

RNAlib-2.6.0b

Generated by Doxygen 1.9.6

1 RNAlib-2.6.0b	1
1.1 A Library for predicting and comparing RNA secondary structures	1
1.2 License	1
1.3 Contributors	2
2 Getting Started	3
2.1 Installation and Configuration	3
2.1.1 Installing the ViennaRNA Package	3
2.1.1.1 Quick-start	3
2.1.1.2 Installation without root privileges	3
2.1.1.3 Notes for MacOS X users	4
2.1.2 Configuring RNAlib features	4
2.1.2.1 Streaming SIMD Extension (SSE) support	5
2.1.2.2 Scripting Interfaces	5
2.1.2.3 Cluster Analysis	5
2.1.2.4 Kinfold	5
2.1.2.5 RNAforester	6
2.1.2.6 Kinwalker	6
2.1.2.7 Link Time Optimization (LTO)	6
2.1.2.8 OpenMP support	6
2.1.2.9 POSIX threads (pthread) support	6
2.1.2.10 SVM Z-score filter in RNALfold	7
2.1.2.11 GNU Scientific Library	7
2.1.2.12 Disable C11/C++11 feature support	7
2.1.2.13 Enable warnings for use of deprecated symbols	7
2.1.2.14 Single precision partition function	7
2.1.2.15 Help	8
2.1.3 Linking against RNAlib	8
2.1.3.1 Compiler and Linker flags	8
2.1.3.2 The pkg-config tool	9
2.2 HelloWorld	10
2.3 HelloWorld (Perl/Python)	12
2.3.1 Perl5	12
2.3.2 Python	13
3 Concepts and Algorithms	15
3.1 RNA Structure	16
3.1.1 RNA Structures	16
3.1.2 Levels of Structure Abstraction	16
3.1.2.1 Primary Structure	16
3.1.2.2 Secondary Structure	16
3.1.2.3 Tertiary Structure	16
3.1.2.4 Quarternary Structure	16

3.1.2.5 Pseudo-Knots	16
3.2 Distance Measures	16
3.2.1 Functions for Tree Edit Distances	17
3.2.2 Functions for String Alignment	18
3.2.3 Functions for Comparison of Base Pair Probabilities	18
3.3 Free Energy of Secondary Structures	18
3.3.1 Secondary Structure Loop Decomposition	19
3.3.1.1 Free Energy Evaluation API	20
3.3.2 Free Energy Parameters	20
3.3.2.1 Free Energy Parameters Modification API	20
3.3.3 Fine-tuning of the Energy Evaluation Model	20
3.4 Secondary Structure Folding Grammar	20
3.4.1 Secondary Structure Folding Recurrences	21
3.4.2 Additional Structural Domains	21
3.4.2.1 Structured Domains	22
3.4.2.2 Unstructured Domains	22
3.4.2.3 Domain Extension API	23
3.4.3 Constraints on the Folding Grammar	23
3.4.3.1 Hard Constraints API	23
3.4.3.2 Soft Constraints API	24
3.5 RNA Secondary Structure Landscapes	24
3.5.1 The Neighborhood of a Secondary Structure	24
3.5.2 The Secondary Structure Landscape API	24
3.6 Minimum Free Energy Algorithm(s)	24
3.6.1 Zuker's Algorithm	24
3.6.2 MFE for circular RNAs	24
3.6.3 MFE Algorithm API	24
3.7 Partition Function and Equilibrium Probability Algorithm(s)	25
3.7.1 Equilibrium Ensemble Statistics	25
3.7.2 Partition Function and Equilibrium Probability API	25
3.8 Suboptimals and (other) Representative Structures	26
3.8.1 Suboptimal Secondary Structures	26
3.8.2 Sampling Secondary Structures from the Ensemble	26
3.8.3 Structure Enumeration and Sampling API	26
3.9 RNA-RNA Interaction	26
3.9.1 	26
3.9.2 Concatenating RNA sequences	26
3.9.3 RNA-RNA interaction as a Stepwise Process	26
3.9.4 RNA-RNA Interaction API	27
3.10 Locally Stable Secondary Structures	27
3.10.1 local_intro	27
3.10.2 local_mfe	27

3.10.3 local_pf	27
3.10.4 Locally Stable Secondary Structure API	27
3.11 Comparative Structure Prediction	27
3.11.1 Incorporate Evolutionary Information	27
3.11.2 Comparative Structure Prediction API	27
3.12 Classified DP variations	27
3.12.1 The Idea of Classified Dynamic Programming	27
3.12.2 Distance Class Partitioning	27
3.12.3 Density of States (DOS)	28
3.12.4 Classified DP API	28
3.13 RNA Sequence Design	28
3.13.1 Generate Sequences that fold into particular Secondary Structures	28
3.13.2 RNA Sequence Design API	28
3.14 Experimental Structure Probing Data	28
3.14.1 Guide the Structure Prediction using Experimental Data	28
3.14.1.1 SHAPE reactivities	28
3.14.2 Structure Probing Data API	28
3.15 Ligand Binding	28
3.15.1 Small Molecules and Proteins that bind to specific RNA Structures	28
3.15.2 ligand_binding_api	28
3.16 (Tertiary) Structure Motifs	29
3.16.1 Incorporating Higher-Order (Tertiary) Structure Motifs	29
3.16.2 RNA G-Quadruplexes	29
3.16.3 (Tertiary) Structure Motif API	29
4 I/O Formats	31
4.1 RNA Structure Notations	31
4.1.1 Representations of Secondary Structures	31
4.1.1.1 Dot-Bracket Notation (a.k.a. Dot-Parenthesis Notation)	31
4.1.1.2 Washington University Secondary Structure (WUSS) notation	32
4.1.1.3 Abstract Shapes	33
4.1.1.4 Tree Representations of Secondary Structures	33
4.1.2 Examples for Structure Parsing and Conversion	34
4.1.3 Structure Parsing and Conversion API	34
4.2 File Formats	35
4.2.1 File formats for Multiple Sequence Alignments (MSA)	35
4.2.1.1 ClustalW format	35
4.2.1.2 Stockholm 1.0 format	36
4.2.1.3 FASTA (Pearson) format	36
4.2.1.4 MAF format	37
4.2.2 File formats to manipulate the RNA folding grammar	38
4.2.2.1 Command Files	38

4.2.3 File Formats for Energy Parameters	40
4.2.3.1 JSON Parameter Files for Modified Bases	40
4.3 Plotting	43
4.3.1 Producing secondary structure graphs	43
4.3.2 Producing (colored) dot plots for base pair probabilities	44
4.3.3 Producing (colored) alignments	44
5 Basic Data Structures	45
5.1 Sequence and Structure Data	45
5.2 The 'Fold Compound'	45
5.3 Model Details	45
6 API Features	47
6.1 RNAlib API v3.0	47
6.1.1 Introduction	47
6.1.2 What are the major changes?	47
6.1.3 How to port your program to the new API	47
6.1.4 Some Examples using RNAlib API v3.0	47
6.2 Callback Functions	48
6.2.1 The purpose of Callback mechanisms	48
6.2.2 List of available Callbacks	48
6.3 Scripting Language interface(s)	49
6.3.1 Introduction	49
6.3.2 Function Renaming	49
6.3.2.1 Global Variables	49
6.3.3 Object oriented Interface for Data Structures	50
6.3.4 Examples	50
6.3.5 SWIG generated Wrapper notes	50
7 Additional Utilities	63
8 Examples	65
8.1 C Examples	65
8.1.1 Hello World Examples	65
8.1.2 First Steps with the Fold Compound	67
8.1.3 Writing Callback Functions	68
8.1.4 Application of Soft Constraints	69
8.1.5 Other Examples	69
8.1.6 Deprecated Examples	70
8.2 Perl5 Examples	71
8.3 Python Examples	72
9 Contributing to the ViennaRNA Package	77

10 Changelog	79
11 Deprecated List	113
12 Bug List	125
13 Module Index	127
13.1 The RNAlib API	127
14 Data Structure Index	131
14.1 Data Structures	131
15 File Index	133
15.1 File List	133
16 Module Documentation	139
16.1 Free Energy Evaluation	139
16.1.1 Detailed Description	139
16.1.2 Function Documentation	142
16.1.2.1 vrna_eval_structure()	142
16.1.2.2 vrna_eval_covar_structure()	142
16.1.2.3 vrna_eval_structure_verbose()	143
16.1.2.4 vrna_eval_structure_v()	143
16.1.2.5 vrna_eval_structure_pt()	144
16.1.2.6 vrna_eval_structure_pt_verbose()	144
16.1.2.7 vrna_eval_structure_pt_v()	145
16.1.2.8 vrna_eval_structure_simple()	146
16.1.2.9 vrna_eval_circ_structure()	146
16.1.2.10 vrna_eval_gquad_structure()	147
16.1.2.11 vrna_eval_circ_gquad_structure()	147
16.1.2.12 vrna_eval_structure_simple_verbose()	148
16.1.2.13 vrna_eval_structure_simple_v()	148
16.1.2.14 vrna_eval_circ_structure_v()	149
16.1.2.15 vrna_eval_gquad_structure_v()	149
16.1.2.16 vrna_eval_circ_gquad_structure_v()	150
16.1.2.17 vrna_eval_consensus_structure_simple()	151
16.1.2.18 vrna_eval_circ_consensus_structure()	151
16.1.2.19 vrna_eval_gquad_consensus_structure()	152
16.1.2.20 vrna_eval_circ_gquad_consensus_structure()	152
16.1.2.21 vrna_eval_consensus_structure_simple_verbose()	153
16.1.2.22 vrna_eval_consensus_structure_simple_v()	154
16.1.2.23 vrna_eval_circ_consensus_structure_v()	154
16.1.2.24 vrna_eval_gquad_consensus_structure_v()	155
16.1.2.25 vrna_eval_circ_gquad_consensus_structure_v()	156

16.1.2.26 vrna_eval_structure_pt_simple()	157
16.1.2.27 vrna_eval_structure_pt_simple_verbose()	157
16.1.2.28 vrna_eval_structure_pt_simple_v()	158
16.1.2.29 vrna_eval_consensus_structure_pt_simple()	158
16.1.2.30 vrna_eval_consensus_structure_pt_simple_verbose()	159
16.1.2.31 vrna_eval_consensus_structure_pt_simple_v()	159
16.2 Energy Evaluation for Individual Loops	159
16.2.1 Detailed Description	159
16.2.2 Function Documentation	160
16.2.2.1 vrna_eval_loop_pt()	160
16.2.2.2 vrna_eval_loop_pt_v()	160
16.3 Energy Evaluation for Atomic Moves	161
16.3.1 Detailed Description	161
16.3.2 Function Documentation	161
16.3.2.1 vrna_eval_move()	161
16.3.2.2 vrna_eval_move_pt()	162
16.4 Deprecated Interface for Free Energy Evaluation	162
16.4.1 Detailed Description	162
16.4.2 Function Documentation	163
16.4.2.1 energy_of_structure()	163
16.4.2.2 energy_of_struct_par()	164
16.4.2.3 energy_of_circ_structure()	164
16.4.2.4 energy_of_circ_struct_par()	165
16.4.2.5 energy_of_structure_pt()	166
16.4.2.6 energy_of_struct_pt_par()	166
16.4.2.7 energy_of_move()	167
16.4.2.8 energy_of_move_pt()	167
16.4.2.9 loop_energy()	168
16.4.2.10 energy_of_struct()	168
16.4.2.11 energy_of_struct_pt()	169
16.4.2.12 energy_of_circ_struct()	170
16.4.2.13 E_Stem()	170
16.4.2.14 exp_E_ExtLoop()	171
16.4.2.15 exp_E_Stem()	172
16.4.2.16 E_IntLoop()	172
16.4.2.17 exp_E_IntLoop()	173
16.5 The RNA Folding Grammar	174
16.5.1 Detailed Description	174
16.5.2 Data Structure Documentation	175
16.5.2.1 struct vrna_gr_aux_s	175
16.5.3 Typedef Documentation	175
16.5.3.1 vrna_grammar_data_free_f	175

16.6 Fine-tuning of the Implemented Models	175
16.6.1 Detailed Description	175
16.6.2 Data Structure Documentation	180
16.6.2.1 struct vrna_md_s	180
16.6.3 Macro Definition Documentation	183
16.6.3.1 VRNA_MODEL_DEFAULT_TEMPERATURE	183
16.6.3.2 VRNA_MODEL_DEFAULT_PF_SCALE	183
16.6.3.3 VRNA_MODEL_DEFAULT_BETA_SCALE	183
16.6.3.4 VRNA_MODEL_DEFAULT_DANGLES	184
16.6.3.5 VRNA_MODEL_DEFAULT_SPECIAL_HP	184
16.6.3.6 VRNA_MODEL_DEFAULT_NO_LP	184
16.6.3.7 VRNA_MODEL_DEFAULT_NO_GU	184
16.6.3.8 VRNA_MODEL_DEFAULT_NO_GU_CLOSURE	184
16.6.3.9 VRNA_MODEL_DEFAULT_CIRC	184
16.6.3.10 VRNA_MODEL_DEFAULT_GQUAD	185
16.6.3.11 VRNA_MODEL_DEFAULT_UNIQ_ML	185
16.6.3.12 VRNA_MODEL_DEFAULT_ENERGY_SET	185
16.6.3.13 VRNA_MODEL_DEFAULT_BACKTRACK	185
16.6.3.14 VRNA_MODEL_DEFAULT_BACKTRACK_TYPE	185
16.6.3.15 VRNA_MODEL_DEFAULT_COMPUTE_BPP	185
16.6.3.16 VRNA_MODEL_DEFAULT_MAX_BP_SPAN	186
16.6.3.17 VRNA_MODEL_DEFAULT_WINDOW_SIZE	186
16.6.3.18 VRNA_MODEL_DEFAULT_LOG_ML	186
16.6.3.19 VRNA_MODEL_DEFAULT_ALI_OLD_EN	186
16.6.3.20 VRNA_MODEL_DEFAULT_ALI_RIBO	186
16.6.3.21 VRNA_MODEL_DEFAULT_ALI_CV_FACT	186
16.6.3.22 VRNA_MODEL_DEFAULT_ALI_NC_FACT	187
16.6.4 Function Documentation	187
16.6.4.1 vrna_md_set_default()	187
16.6.4.2 vrna_md_update()	187
16.6.4.3 vrna_md_copy()	187
16.6.4.4 vrna_md_option_string()	188
16.6.4.5 vrna_md_defaults_reset()	188
16.6.4.6 vrna_md_defaults_temperature()	188
16.6.4.7 vrna_md_defaults_temperature_get()	189
16.6.4.8 vrna_md_defaults_betaScale()	189
16.6.4.9 vrna_md_defaults_betaScale_get()	189
16.6.4.10 vrna_md_defaults_dangles()	190
16.6.4.11 vrna_md_defaults_dangles_get()	190
16.6.4.12 vrna_md_defaults_special_hp()	190
16.6.4.13 vrna_md_defaults_special_hp_get()	190
16.6.4.14 vrna_md_defaults_noLP()	191

16.6.4.15 vrna_md_defaults_noLP_get()	191
16.6.4.16 vrna_md_defaults_noGU()	191
16.6.4.17 vrna_md_defaults_noGU_get()	192
16.6.4.18 vrna_md_defaults_noGUclosure()	192
16.6.4.19 vrna_md_defaults_noGUclosure_get()	192
16.6.4.20 vrna_md_defaults_logML()	192
16.6.4.21 vrna_md_defaults_logML_get()	193
16.6.4.22 vrna_md_defaults_circ()	193
16.6.4.23 vrna_md_defaults_circ_get()	193
16.6.4.24 vrna_md_defaults_gquad()	193
16.6.4.25 vrna_md_defaults_gquad_get()	194
16.6.4.26 vrna_md_defaults_uniq_ML()	194
16.6.4.27 vrna_md_defaults_uniq_ML_get()	194
16.6.4.28 vrna_md_defaults_energy_set()	195
16.6.4.29 vrna_md_defaults_energy_set_get()	195
16.6.4.30 vrna_md_defaults_backtrack()	195
16.6.4.31 vrna_md_defaults_backtrack_get()	196
16.6.4.32 vrna_md_defaults_backtrack_type()	196
16.6.4.33 vrna_md_defaults_backtrack_type_get()	196
16.6.4.34 vrna_md_defaults_compute_bpp()	196
16.6.4.35 vrna_md_defaults_compute_bpp_get()	197
16.6.4.36 vrna_md_defaults_max_bp_span()	197
16.6.4.37 vrna_md_defaults_max_bp_span_get()	197
16.6.4.38 vrna_md_defaults_min_loop_size()	197
16.6.4.39 vrna_md_defaults_min_loop_size_get()	198
16.6.4.40 vrna_md_defaults_window_size()	198
16.6.4.41 vrna_md_defaults_window_size_get()	198
16.6.4.42 vrna_md_defaults_oldAliEn()	199
16.6.4.43 vrna_md_defaults_oldAliEn_get()	199
16.6.4.44 vrna_md_defaults_ribo()	199
16.6.4.45 vrna_md_defaults_ribo_get()	200
16.6.4.46 vrna_md_defaults_cv_fact()	200
16.6.4.47 vrna_md_defaults_cv_fact_get()	200
16.6.4.48 vrna_md_defaults_nc_fact()	200
16.6.4.49 vrna_md_defaults_nc_fact_get()	201
16.6.4.50 vrna_md_defaults_sfact()	201
16.6.4.51 vrna_md_defaults_sfact_get()	201
16.6.4.52 vrna_md_defaults_salt()	201
16.6.4.53 vrna_md_defaults_salt_get()	202
16.6.4.54 vrna_md_defaults_saltMLLower()	202
16.6.4.55 vrna_md_defaults_saltMLLower_get()	202
16.6.4.56 vrna_md_defaults_saltMLUpper()	202

16.6.4.57 vrna_md_defaults_saltMLUpper_get()	203
16.6.4.58 vrna_md_defaults_saltDPXInit()	203
16.6.4.59 vrna_md_defaults_saltDPXInit_get()	203
16.6.4.60 set_model_details()	203
16.6.5 Variable Documentation	203
16.6.5.1 temperature	204
16.6.5.2 pf_scale	204
16.6.5.3 dangles	204
16.6.5.4 tetra_loop	204
16.6.5.5 noLonelyPairs	204
16.6.5.6 energy_set	204
16.6.5.7 do_backtrack	205
16.6.5.8 backtrack_type	205
16.6.5.9 nonstandards	205
16.6.5.10 max_bp_span	205
16.7 Energy Parameters	205
16.7.1 Detailed Description	205
16.7.2 Data Structure Documentation	207
16.7.2.1 struct vrna_param_s	207
16.7.2.2 struct vrna_exp_param_s	207
16.7.3 Typedef Documentation	208
16.7.3.1 paramT	208
16.7.3.2 pf_paramT	208
16.7.4 Function Documentation	208
16.7.4.1 vrna_params()	208
16.7.4.2 vrna_params_copy()	209
16.7.4.3 vrna_exp_params()	209
16.7.4.4 vrna_exp_params_comparative()	210
16.7.4.5 vrna_exp_params_copy()	210
16.7.4.6 vrna_params_subst()	210
16.7.4.7 vrna_exp_params_subst()	211
16.7.4.8 vrna_exp_params_rescale()	211
16.7.4.9 vrna_params_reset()	212
16.7.4.10 vrna_exp_params_reset()	213
16.7.4.11 get_scaled_pf_parameters()	213
16.7.4.12 get_boltzmann_factors()	213
16.7.4.13 get_boltzmann_factor_copy()	214
16.7.4.14 get_scaled_alipf_parameters()	214
16.7.4.15 get_boltzmann_factors_alipf()	215
16.7.4.16 scale_parameters()	215
16.7.4.17 get_scaled_parameters()	215
16.7.4.18 vrna_salt_loop()	216

16.7.4.19 vrna_salt_loop_int()	216
16.7.4.20 vrna_salt_stack()	217
16.8 Extending the Folding Grammar with Additional Domains	217
16.8.1 Detailed Description	217
16.9 Unstructured Domains	217
16.9.1 Detailed Description	217
16.9.2 Data Structure Documentation	219
16.9.2.1 struct vrna_unstructured_domain_s	219
16.9.3 Typedef Documentation	220
16.9.3.1 vrna_ud_f	220
16.9.3.2 vrna_ud_exp_f	220
16.9.3.3 vrna_ud_production_f	221
16.9.3.4 vrna_ud_exp_production_f	221
16.9.3.5 vrna_ud_add_probs_f	221
16.9.3.6 vrna_ud_get_probs_f	221
16.9.4 Function Documentation	221
16.9.4.1 vrna_ud_motifs_centroid()	222
16.9.4.2 vrna_ud_motifs_MEA()	222
16.9.4.3 vrna_ud_motifs_MFE()	223
16.9.4.4 vrna_ud_add_motif()	223
16.9.4.5 vrna_ud_remove()	224
16.9.4.6 vrna_ud_set_data()	224
16.9.4.7 vrna_ud_set_prod_rule_cb()	224
16.9.4.8 vrna_ud_set_exp_prod_rule_cb()	225
16.10 Structured Domains	226
16.10.1 Detailed Description	226
16.11 Constraining the RNA Folding Grammar	226
16.11.1 Detailed Description	226
16.11.2 Macro Definition Documentation	229
16.11.2.1 VRNA_CONSTRAINT_FILE	230
16.11.2.2 VRNA_CONSTRAINT_SOFT_MFE	230
16.11.2.3 VRNA_CONSTRAINT_SOFT_PF	230
16.11.2.4 VRNA_DECOMP_PAIR_HP	230
16.11.2.5 VRNA_DECOMP_PAIR_IL	231
16.11.2.6 VRNA_DECOMP_PAIR_ML	231
16.11.2.7 VRNA_DECOMP_ML_ML_ML	232
16.11.2.8 VRNA_DECOMP_ML_STEM	232
16.11.2.9 VRNA_DECOMP_ML_ML	233
16.11.2.10 VRNA_DECOMP_ML_UP	233
16.11.2.11 VRNA_DECOMP_ML_ML_STEM	234
16.11.2.12 VRNA_DECOMP_ML_COAXIAL	234
16.11.2.13 VRNA_DECOMP_ML_COAXIAL_ENC	234

16.11.2.14 VRNA_DECOMP_EXT_EXT	235
16.11.2.15 VRNA_DECOMP_EXT_UP	235
16.11.2.16 VRNA_DECOMP_EXT_STEM	235
16.11.2.17 VRNA_DECOMP_EXT_EXT_EXT	236
16.11.2.18 VRNA_DECOMP_EXT_STEM_EXT	236
16.11.2.19 VRNA_DECOMP_EXT_STEM_OUTSIDE	236
16.11.2.20 VRNA_DECOMP_EXT_EXT_STEM	237
16.11.2.21 VRNA_DECOMP_EXT_EXT_STEM1	237
16.11.3 Function Documentation	237
16.11.3.1 vrna_constraints_add()	237
16.11.3.2 vrna_message_constraint_options()	238
16.11.3.3 vrna_message_constraint_options_all()	239
16.12 Hard Constraints	239
16.12.1 Detailed Description	239
16.12.2 Data Structure Documentation	241
16.12.2.1 struct vrna_hc_s	241
16.12.2.2 struct vrna_hc_up_s	241
16.12.3 Macro Definition Documentation	242
16.12.3.1 VRNA_CONSTRAINT_DB	242
16.12.3.2 VRNA_CONSTRAINT_DB_ENFORCE_BP	242
16.12.3.3 VRNA_CONSTRAINT_DB_PIPE	242
16.12.3.4 VRNA_CONSTRAINT_DB_DOT	242
16.12.3.5 VRNA_CONSTRAINT_DB_X	243
16.12.3.6 VRNA_CONSTRAINT_DB_RND_BRACK	243
16.12.3.7 VRNA_CONSTRAINT_DB_INTRAMOL	243
16.12.3.8 VRNA_CONSTRAINT_DB_INTERMOL	243
16.12.3.9 VRNA_CONSTRAINT_DB_GQUAD	243
16.12.3.10 VRNA_CONSTRAINT_DB_WUSS	244
16.12.3.11 VRNA_CONSTRAINT_DB_DEFAULT	244
16.12.3.12 VRNA_CONSTRAINT_CONTEXT_EXT_LOOP	244
16.12.3.13 VRNA_CONSTRAINT_CONTEXT_HP_LOOP	244
16.12.3.14 VRNA_CONSTRAINT_CONTEXT_INT_LOOP	244
16.12.3.15 VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC	244
16.12.3.16 VRNA_CONSTRAINT_CONTEXT_MB_LOOP	244
16.12.3.17 VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC	245
16.12.3.18 VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS	245
16.12.4 Typedef Documentation	245
16.12.4.1 vrna_hc_eval_f	245
16.12.5 Function Documentation	246
16.12.5.1 vrna_hc_init()	246
16.12.5.2 vrna_hc_add_up()	246
16.12.5.3 vrna_hc_add_up_batch()	246

16.12.5.4 vrna_hc_add_bp()	247
16.12.5.5 vrna_hc_add_bp_nonspecific()	247
16.12.5.6 vrna_hc_free()	248
16.12.5.7 vrna_hc_add_from_db()	248
16.13 Soft Constraints	248
16.13.1 Detailed Description	248
16.13.2 Data Structure Documentation	249
16.13.2.1 struct vrna_sc_s	249
16.13.3 Typedef Documentation	251
16.13.3.1 vrna_sc_f	251
16.13.3.2 vrna_sc_exp_f	251
16.13.3.3 vrna_sc_bt_f	252
16.13.4 Function Documentation	253
16.13.4.1 vrna_sc_init()	253
16.13.4.2 vrna_sc_set_bp()	253
16.13.4.3 vrna_sc_add_bp()	254
16.13.4.4 vrna_sc_set_up()	254
16.13.4.5 vrna_sc_add_up()	255
16.13.4.6 vrna_sc_remove()	255
16.13.4.7 vrna_sc_free()	256
16.13.4.8 vrna_sc_add_data()	256
16.13.4.9 vrna_sc_add_f()	256
16.13.4.10 vrna_sc_add_bt()	257
16.13.4.11 vrna_sc_add_exp_f()	257
16.14 The RNA Secondary Structure Landscape	258
16.14.1 Detailed Description	258
16.15 Minimum Free Energy (MFE) Algorithms	258
16.15.1 Detailed Description	258
16.16 Partition Function and Equilibrium Properties	259
16.16.1 Detailed Description	259
16.16.2 Function Documentation	259
16.16.2.1 vrna_pf_float_precision()	260
16.17 Global MFE Prediction	260
16.17.1 Detailed Description	260
16.17.2 Function Documentation	261
16.17.2.1 vrna_mfe()	261
16.17.2.2 vrna_mfe_dimer()	261
16.17.2.3 vrna_fold()	262
16.17.2.4 vrna_circfold()	263
16.17.2.5 vrna_alifold()	263
16.17.2.6 vrna_circalifold()	264
16.17.2.7 vrna_cofold()	264

16.18 Local (sliding window) MFE Prediction	265
16.18.1 Detailed Description	265
16.18.2 Typedef Documentation	266
16.18.2.1 vrna_mfe_window_f	266
16.18.3 Function Documentation	267
16.18.3.1 vrna_mfe_window()	267
16.18.3.2 vrna_mfe_window_zscore()	267
16.18.3.3 vrna_Lfold()	268
16.18.3.4 vrna_Lfoldz()	268
16.19 Backtracking MFE structures	269
16.19.1 Detailed Description	269
16.19.2 Function Documentation	270
16.19.2.1 vrna_backtrack5()	270
16.19.2.2 vrna_BT_hp_loop()	270
16.19.2.3 vrna_BT_stack()	271
16.19.2.4 vrna_BT_int_loop()	271
16.19.2.5 vrna_BT_mb_loop()	271
16.20 Global Partition Function and Equilibrium Probabilities	272
16.20.1 Detailed Description	272
16.20.2 Data Structure Documentation	273
16.20.2.1 struct vrna_dimer_pf_s	273
16.20.2.2 struct vrna_multimer_pf_s	273
16.20.3 Function Documentation	273
16.20.3.1 vrna_pf()	274
16.20.3.2 vrna_pf_dimer()	274
16.20.3.3 vrna_pf_fold()	275
16.20.3.4 vrna_pf_circfold()	275
16.20.3.5 vrna_pf_alifold()	276
16.20.3.6 vrna_pf_circalifold()	277
16.20.3.7 vrna_plist_from_probs()	277
16.20.3.8 vrna_pf_co_fold()	278
16.21 Local (sliding window) Partition Function and Equilibrium Probabilities	278
16.21.1 Detailed Description	278
16.21.2 Macro Definition Documentation	279
16.21.2.1 VRNA_PROBS_WINDOW_BPP	280
16.21.2.2 VRNA_PROBS_WINDOW_UP	280
16.21.2.3 VRNA_PROBS_WINDOW_STACKP	280
16.21.2.4 VRNA_PROBS_WINDOW_UP_SPLIT	280
16.21.2.5 VRNA_PROBS_WINDOW_PF	281
16.21.3 Typedef Documentation	281
16.21.3.1 vrna_probs_window_f	281
16.21.4 Function Documentation	282

16.21.4.1 vrna_probs_window()	282
16.21.4.2 vrna_pfl_fold()	283
16.21.4.3 vrna_pfl_fold_cb()	283
16.21.4.4 vrna_pfl_fold_up()	284
16.21.4.5 vrna_pfl_fold_up_cb()	285
16.22 Suboptimals and Representative Structures	285
16.22.1 Detailed Description	285
16.23 Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989	286
16.23.1 Detailed Description	286
16.23.2 Function Documentation	286
16.23.2.1 zuckersubopt()	286
16.23.2.2 zuckersubopt_par()	286
16.23.2.3 vrna_subopt_zuker()	287
16.24 Suboptimal Structures within an Energy Band around the MFE	287
16.24.1 Detailed Description	287
16.24.2 Typedef Documentation	288
16.24.2.1 vrna_subopt_result_f	288
16.24.3 Function Documentation	288
16.24.3.1 vrna_subopt()	288
16.24.3.2 vrna_subopt_cb()	289
16.24.3.3 subopt()	290
16.24.3.4 subopt_par()	290
16.24.3.5 subopt_circ()	290
16.24.4 Variable Documentation	291
16.24.4.1 print_energy	291
16.24.4.2 subopt_sorted	291
16.25 Random Structure Samples from the Ensemble	291
16.25.1 Detailed Description	291
16.25.2 Macro Definition Documentation	293
16.25.2.1 VRNA_PBACKTRACK_DEFAULT	293
16.25.2.2 VRNA_PBACKTRACK_NON_REDUNDANT	293
16.25.3 Typedef Documentation	293
16.25.3.1 vrna_bs_result_f	293
16.25.3.2 vrna_pbacktrack_mem_t	294
16.25.4 Function Documentation	294
16.25.4.1 vrna_pbacktrack5()	294
16.25.4.2 vrna_pbacktrack5_num()	295
16.25.4.3 vrna_pbacktrack5_cb()	296
16.25.4.4 vrna_pbacktrack5_resume()	297
16.25.4.5 vrna_pbacktrack5_resume_cb()	298
16.25.4.6 vrna_pbacktrack()	300
16.25.4.7 vrna_pbacktrack_num()	300

16.25.4.8 vrna_pbacktrack_cb()	301
16.25.4.9 vrna_pbacktrack_resume()	303
16.25.4.10 vrna_pbacktrack_resume_cb()	304
16.25.4.11 vrna_pbacktrack_sub()	305
16.25.4.12 vrna_pbacktrack_sub_num()	306
16.25.4.13 vrna_pbacktrack_sub_cb()	307
16.25.4.14 vrna_pbacktrack_sub_resume()	308
16.25.4.15 vrna_pbacktrack_sub_resume_cb()	310
16.25.4.16 vrna_pbacktrack_mem_free()	311
16.26 Compute the Structure with Maximum Expected Accuracy (MEA)	312
16.26.1 Detailed Description	312
16.26.2 Function Documentation	312
16.26.2.1 vrna_MEA()	312
16.26.2.2 vrna_MEA_from_plist()	313
16.26.2.3 MEA()	313
16.27 Compute the Centroid Structure	314
16.27.1 Detailed Description	314
16.27.2 Function Documentation	314
16.27.2.1 vrna_centroid()	314
16.27.2.2 vrna_centroid_from_plist()	314
16.27.2.3 vrna_centroid_from_probs()	315
16.28 RNA-RNA Interaction	315
16.28.1 Detailed Description	315
16.29 Classified Dynamic Programming Variants	316
16.29.1 Detailed Description	316
16.30 Distance Based Partitioning of the Secondary Structure Space	316
16.30.1 Detailed Description	316
16.31 Computing MFE representatives of a Distance Based Partitioning	316
16.31.1 Detailed Description	317
16.31.2 Data Structure Documentation	317
16.31.2.1 struct vrna_sol_TwoD_t	317
16.31.2.2 struct TwoDfold_vars	318
16.31.3 Typedef Documentation	318
16.31.3.1 vrna_sol_TwoD_t	319
16.31.3.2 TwoDfold_vars	319
16.31.4 Function Documentation	319
16.31.4.1 vrna_mfe_TwoD()	319
16.31.4.2 vrna_backtrack5_TwoD()	320
16.31.4.3 get_TwoDfold_variables()	320
16.31.4.4 destroy_TwoDfold_variables()	321
16.31.4.5 TwoDfoldList()	321
16.31.4.6 TwoDfold_backtrack_f5()	322

16.32 Computing Partition Functions of a Distance Based Partitioning	322
16.32.1 Detailed Description	322
16.32.2 Data Structure Documentation	323
16.32.2.1 struct vrna_sol_TwoD_pf_t	323
16.32.3 Typedef Documentation	323
16.32.3.1 vrna_sol_TwoD_pf_t	323
16.32.4 Function Documentation	323
16.32.4.1 vrna_pf_TwoD()	323
16.33 Stochastic Backtracking of Structures from Distance Based Partitioning	324
16.33.1 Detailed Description	324
16.33.2 Function Documentation	324
16.33.2.1 vrna_pbacktrack_TwoD()	324
16.33.2.2 vrna_pbacktrack5_TwoD()	325
16.34 Predicting various thermodynamic properties	326
16.34.1 Detailed Description	326
16.34.2 Data Structure Documentation	327
16.34.2.1 struct vrna_heat_capacity_s	327
16.34.3 Typedef Documentation	328
16.34.3.1 vrna_heat_capacity_f	328
16.34.3.2 vrna_heat_capacity_t	328
16.34.4 Function Documentation	328
16.34.4.1 vrna_mean_bp_distance_pr()	328
16.34.4.2 vrna_mean_bp_distance()	329
16.34.4.3 vrna_ensemble_defect_pt()	329
16.34.4.4 vrna_ensemble_defect()	330
16.34.4.5 vrna_positional_entropy()	331
16.34.4.6 vrna_stack_prob()	331
16.34.4.7 vrna_pf_dimer_probs()	331
16.34.4.8 vrna_pr_structure()	332
16.34.4.9 vrna_pr_energy()	333
16.34.4.10 vrna_heat_capacity()	333
16.34.4.11 vrna_heat_capacity_cb()	334
16.34.4.12 vrna_heat_capacity_simple()	334
16.35 Compute the Density of States	335
16.35.1 Detailed Description	335
16.35.2 Variable Documentation	335
16.35.2.1 density_of_states	335
16.36 Inverse Folding (Design)	336
16.36.1 Detailed Description	336
16.36.2 Function Documentation	336
16.36.2.1 inverse_fold()	336
16.36.2.2 inverse_pf_fold()	337

16.36.3 Variable Documentation	337
16.36.3.1 final_cost	337
16.36.3.2 give_up	337
16.36.3.3 inv_verbose	337
16.37 Neighborhood Relation and Move Sets for Secondary Structures	337
16.37.1 Detailed Description	337
16.37.2 Data Structure Documentation	340
16.37.2.1 struct vrna_move_s	340
16.37.3 Macro Definition Documentation	341
16.37.3.1 VRNA_MOVESET_INSERTION	341
16.37.3.2 VRNA_MOVESET_DELETION	341
16.37.3.3 VRNA_MOVESET_SHIFT	341
16.37.3.4 VRNA_MOVESET_NO_LP	341
16.37.3.5 VRNA_MOVESET_DEFAULT	342
16.37.3.6 VRNA_NEIGHBOR_CHANGE	342
16.37.3.7 VRNA_NEIGHBOR_INVALID	342
16.37.3.8 VRNA_NEIGHBOR_NEW	342
16.37.4 Typedef Documentation	342
16.37.4.1 vrna_move_update_f	342
16.37.5 Function Documentation	343
16.37.5.1 vrna_move_init()	343
16.37.5.2 vrna_move_list_free()	343
16.37.5.3 vrna_move_apply()	343
16.37.5.4 vrna_move_is_removal()	343
16.37.5.5 vrna_move_is_insertion()	344
16.37.5.6 vrna_move_is_shift()	344
16.37.5.7 vrna_move_compare()	344
16.37.5.8 vrna_loopidx_update()	345
16.37.5.9 vrna_neighbors()	345
16.37.5.10 vrna_neighbors_successive()	346
16.37.5.11 vrna_move_neighbor_diff_cb()	346
16.37.5.12 vrna_move_neighbor_diff()	347
16.38 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima	348
16.38.1 Detailed Description	348
16.38.2 Data Structure Documentation	349
16.38.2.1 struct vrna_path_s	349
16.38.3 Macro Definition Documentation	350
16.38.3.1 VRNA_PATH_TYPE_DOT_BRACKET	350
16.38.3.2 VRNA_PATH_TYPE_MOVES	350
16.38.4 Function Documentation	350
16.38.4.1 vrna_path_free()	350
16.38.4.2 vrna_path_options_free()	351

16.39 Direct Refolding Paths between two Secondary Structures	351
16.39.1 Detailed Description	351
16.39.2 Function Documentation	351
16.39.2.1 vrna_path_findpath_saddle()	351
16.39.2.2 vrna_path_findpath_saddle_ub()	352
16.39.2.3 vrna_path_findpath()	353
16.39.2.4 vrna_path_findpath_ub()	353
16.39.2.5 vrna_path_options_findpath()	354
16.39.2.6 vrna_path_direct()	355
16.39.2.7 vrna_path_direct_ub()	355
16.40 Folding Paths that start at a single Secondary Structure	356
16.40.1 Detailed Description	356
16.40.2 Macro Definition Documentation	357
16.40.2.1 VRNA_PATH_STEEPEST_DESCENT	357
16.40.2.2 VRNA_PATH_RANDOM	357
16.40.2.3 VRNA_PATH_NO_TRANSITION_OUTPUT	357
16.40.2.4 VRNA_PATH_DEFAULT	357
16.40.3 Function Documentation	357
16.40.3.1 vrna_path()	358
16.40.3.2 vrna_path_gradient()	358
16.40.3.3 vrna_path_random()	359
16.41 Experimental Structure Probing Data	360
16.41.1 Detailed Description	360
16.42 SHAPE Reactivity Data	360
16.42.1 Detailed Description	360
16.42.2 Function Documentation	361
16.42.2.1 vrna_sc_add_SHAPE_deigan()	361
16.42.2.2 vrna_sc_add_SHAPE_deigan_ali()	361
16.42.2.3 vrna_sc_add_SHAPE_zarringhalam()	362
16.42.2.4 vrna_sc_SHAPE_to_pr()	363
16.43 Generate Soft Constraints from Data	363
16.43.1 Detailed Description	363
16.43.2 Macro Definition Documentation	364
16.43.2.1 VRNA_OBJECTIVE_FUNCTION_QUADRATIC	364
16.43.2.2 VRNA_OBJECTIVE_FUNCTION_ABSOLUTE	364
16.43.2.3 VRNA_MINIMIZER_DEFAULT	364
16.43.2.4 VRNA_MINIMIZER_CONJUGATE_FR	365
16.43.2.5 VRNA_MINIMIZER_CONJUGATE_PR	365
16.43.2.6 VRNA_MINIMIZER_VECTOR_BFGS	365
16.43.2.7 VRNA_MINIMIZER_VECTOR_BFGS2	365
16.43.2.8 VRNA_MINIMIZER_STEEPEST_DESCENT	365
16.43.3 Typedef Documentation	365

16.43.3.1 progress_callback	365
16.43.4 Function Documentation	365
16.43.4.1 vrna_sc_minimize_pertubation()	366
16.44 Ligands Binding to RNA Structures	367
16.44.1 Detailed Description	367
16.45 Ligands Binding to Unstructured Domains	367
16.46 Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints	367
16.46.1 Detailed Description	367
16.46.2 Data Structure Documentation	368
16.46.2.1 struct vrna_sc_motif_s	368
16.46.3 Function Documentation	368
16.46.3.1 vrna_sc_add_hi_motif()	368
16.47 Structure Modules and Pseudoknots	369
16.47.1 Detailed Description	369
16.48 Pseudoknots	369
16.48.1 Detailed Description	369
16.48.2 Data Structure Documentation	370
16.48.2.1 struct vrna_pk_plex_result_s	370
16.48.3 Typedef Documentation	371
16.48.3.1 vrna_pk_plex_score_f	371
16.48.3.2 vrna_pk_plex_opt_t	371
16.48.3.3 vrna_pk_plex_t	371
16.48.4 Function Documentation	372
16.48.4.1 vrna_pk_plex()	372
16.48.4.2 vrna_pk_plex_accessibility()	372
16.48.4.3 vrna_pk_plex_opt_defaults()	373
16.48.4.4 vrna_pk_plex_opt()	373
16.48.4.5 vrna_pk_plex_opt_fun()	373
16.49 G-Quadruplexes	374
16.49.1 Detailed Description	374
16.49.2 Function Documentation	374
16.49.2.1 get_gquad_matrix()	374
16.49.2.2 parse_gquad()	375
16.49.2.3 backtrack_GQuad_IntLoop()	375
16.49.2.4 backtrack_GQuad_IntLoop_L()	375
16.50 Post-transcriptional Modifications	376
16.50.1 Detailed Description	376
16.50.2 Typedef Documentation	377
16.50.2.1 vrna_sc_mod_param_t	377
16.50.3 Function Documentation	377
16.50.3.1 vrna_sc_mod_read_from_jsonfile()	377
16.50.3.2 vrna_sc_mod_read_from_json()	378

16.50.3.3 vrna_sc_mod_parameters_free()	378
16.50.3.4 vrna_sc_mod_json()	379
16.50.3.5 vrna_sc_mod_jsonfile()	379
16.50.3.6 vrna_sc_mod()	380
16.50.3.7 vrna_sc_mod_m6A()	380
16.50.3.8 vrna_sc_mod_pseudouridine()	381
16.50.3.9 vrna_sc_mod_inosine()	381
16.50.3.10 vrna_sc_mod_7DA()	381
16.50.3.11 vrna_sc_mod_purine()	382
16.50.3.12 vrna_sc_mod_dihydrouridine()	382
16.51 Utilities	383
16.51.1 Detailed Description	383
16.51.2 Macro Definition Documentation	385
16.51.2.1 VRNA_INPUT_FASTA_HEADER	385
16.51.2.2 VRNA_INPUT_CONSTRAINT	385
16.51.3 Function Documentation	385
16.51.3.1 vrna_alloc()	385
16.51.3.2 vrna_realloc()	386
16.51.3.3 vrna_init_rand()	386
16.51.3.4 vrna_init_rand_seed()	386
16.51.3.5 vrna_urn()	387
16.51.3.6 vrna_int_urn()	387
16.51.3.7 vrna_time_stamp()	387
16.51.3.8 get_input_line()	388
16.51.3.9 vrna_idx_row_wise()	388
16.51.3.10 vrna_idx_col_wise()	388
16.51.4 Variable Documentation	389
16.51.4.1 xsubi	389
16.52 Exterior Loops	389
16.52.1 Detailed Description	389
16.52.2 Typedef Documentation	390
16.52.2.1 vrna_mx_pf_aux_el_t	390
16.52.3 Function Documentation	390
16.52.3.1 vrna_E_ext_stem()	390
16.52.3.2 vrna_eval_ext_stem()	391
16.52.3.3 vrna_exp_E_ext_stem()	391
16.53 Hairpin Loops	392
16.53.1 Detailed Description	392
16.53.2 Function Documentation	392
16.53.2.1 vrna_E_hp_loop()	392
16.53.2.2 vrna_E_ext_hp_loop()	393
16.53.2.3 vrna_eval_hp_loop()	393

16.53.2.4 E_Hairpin()	393
16.53.2.5 exp_E_Hairpin()	394
16.53.2.6 vrna_exp_E_hp_loop()	395
16.54 Internal Loops	395
16.54.1 Detailed Description	396
16.54.2 Function Documentation	396
16.54.2.1 vrna_eval_int_loop()	396
16.55 Multibranch Loops	396
16.55.1 Detailed Description	396
16.55.2 Typedef Documentation	397
16.55.2.1 vrna_mx_pf_aux_ml_t	397
16.55.3 Function Documentation	397
16.55.3.1 vrna_E_mb_loop_stack()	397
16.56 Partition Function for Two Hybridized Sequences	397
16.56.1 Detailed Description	398
16.56.2 Function Documentation	398
16.56.2.1 vrna_pf_co_fold()	399
16.56.2.2 vrna_pf_dimer_concentrations()	399
16.57 Partition Function for two Hybridized Sequences as a Stepwise Process	400
16.57.1 Detailed Description	400
16.57.2 Function Documentation	400
16.57.2.1 pf_unstru()	400
16.57.2.2 pf_interact()	401
16.58 Reading/Writing Energy Parameter Sets from/to File	402
16.58.1 Detailed Description	402
16.58.2 Macro Definition Documentation	403
16.58.2.1 VRNA_PARAMETER_FORMAT_DEFAULT	403
16.58.3 Function Documentation	403
16.58.3.1 vrna_params_load()	403
16.58.3.2 vrna_params_save()	403
16.58.3.3 vrna_params_load_from_string()	404
16.58.3.4 vrna_params_load_defaults()	404
16.58.3.5 vrna_params_load_RNA_Turner2004()	405
16.58.3.6 vrna_params_load_RNA_Turner1999()	405
16.58.3.7 vrna_params_load_RNA_Andronescu2007()	405
16.58.3.8 vrna_params_load_RNA_Langdon2018()	406
16.58.3.9 vrna_params_load_RNA_misc_special_hairpins()	406
16.58.3.10 vrna_params_load_DNA_Mathews2004()	406
16.58.3.11 vrna_params_load_DNA_Mathews1999()	407
16.58.3.12 last_parameter_file()	407
16.58.3.13 read_parameter_file()	407
16.58.3.14 write_parameter_file()	407

16.59 Converting Energy Parameter Files	408
16.59.1 Detailed Description	408
16.59.2 Macro Definition Documentation	408
16.59.2.1 VRNA_CONVERT_OUTPUT_ALL	409
16.59.2.2 VRNA_CONVERT_OUTPUT_HP	409
16.59.2.3 VRNA_CONVERT_OUTPUT_STACK	409
16.59.2.4 VRNA_CONVERT_OUTPUT_MM_HP	409
16.59.2.5 VRNA_CONVERT_OUTPUT_MM_INT	409
16.59.2.6 VRNA_CONVERT_OUTPUT_MM_INT_1N	409
16.59.2.7 VRNA_CONVERT_OUTPUT_MM_INT_23	409
16.59.2.8 VRNA_CONVERT_OUTPUT_MM_MULTI	409
16.59.2.9 VRNA_CONVERT_OUTPUT_MM_EXT	409
16.59.2.10 VRNA_CONVERT_OUTPUT_DANGLE5	410
16.59.2.11 VRNA_CONVERT_OUTPUT_DANGLE3	410
16.59.2.12 VRNA_CONVERT_OUTPUT_INT_11	410
16.59.2.13 VRNA_CONVERT_OUTPUT_INT_21	410
16.59.2.14 VRNA_CONVERT_OUTPUT_INT_22	410
16.59.2.15 VRNA_CONVERT_OUTPUT_BULGE	410
16.59.2.16 VRNA_CONVERT_OUTPUT_INT	410
16.59.2.17 VRNA_CONVERT_OUTPUT_ML	410
16.59.2.18 VRNA_CONVERT_OUTPUT_MISC	410
16.59.2.19 VRNA_CONVERT_OUTPUT_SPECIAL_HP	411
16.59.2.20 VRNA_CONVERT_OUTPUT_VANILLA	411
16.59.2.21 VRNA_CONVERT_OUTPUT_NINIO	411
16.59.2.22 VRNA_CONVERT_OUTPUT_DUMP	411
16.59.3 Function Documentation	411
16.59.3.1 convert_parameter_file()	411
16.60 Utilities to deal with Nucleotide Alphabets	412
16.60.1 Detailed Description	412
16.60.2 Data Structure Documentation	413
16.60.2.1 struct vrna_sequence_s	413
16.60.2.2 struct vrna_alignment_s	413
16.60.3 Enumeration Type Documentation	413
16.60.3.1 vrna_seq_type_e	413
16.60.4 Function Documentation	413
16.60.4.1 vrna_ptypes()	413
16.60.4.2 vrna_seq_encode()	414
16.60.4.3 vrna_seq_encode_simple()	414
16.60.4.4 vrna_nucleotide_encode()	414
16.60.4.5 vrna_nucleotide_decode()	415
16.61 (Nucleic Acid Sequence) String Utilitites	415
16.61.1 Detailed Description	415

16.61.2 Macro Definition Documentation	416
16.61.2.1 FILENAME_MAX_LENGTH	417
16.61.2.2 FILENAME_ID_LENGTH	417
16.61.2.3 VRNA_TRIM_LEADING	417
16.61.2.4 VRNA_TRIM_TRAILING	417
16.61.2.5 VRNA_TRIM_IN_BETWEEN	417
16.61.2.6 VRNA_TRIM_SUBST_BY_FIRST	417
16.61.2.7 VRNA_TRIM_DEFAULT	418
16.61.2.8 VRNA_TRIM_ALL	418
16.61.3 Function Documentation	418
16.61.3.1 vrna_strdup_printf()	418
16.61.3.2 vrna_strdup_vprintf()	418
16.61.3.3 vrna_strcat_printf()	419
16.61.3.4 vrna_strcat_vprintf()	419
16.61.3.5 vrna_strtrim()	420
16.61.3.6 vrna_strsplit()	421
16.61.3.7 vrna_random_string()	422
16.61.3.8 vrna_hamming_distance()	422
16.61.3.9 vrna_hamming_distance_bound()	422
16.61.3.10 vrna_seq_toRNA()	424
16.61.3.11 vrna_seq_toupper()	424
16.61.3.12 vrna_seq_reverse()	424
16.61.3.13 vrna_DNA_complement()	425
16.61.3.14 vrna_seq_ungapped()	425
16.61.3.15 vrna_cut_point_insert()	425
16.61.3.16 vrna_cut_point_remove()	426
16.62 Secondary Structure Utilities	426
16.62.1 Detailed Description	426
16.62.2 Function Documentation	427
16.62.2.1 vrna_refBPcnt_matrix()	427
16.62.2.2 vrna_refBPdist_matrix()	427
16.62.2.3 vrna_db_from_probs()	428
16.62.2.4 vrna_db_from_bp_stack()	428
16.63 Dot-Bracket Notation of Secondary Structures	428
16.63.1 Detailed Description	428
16.63.2 Macro Definition Documentation	429
16.63.2.1 VRNA_BRACKETS_ALPHA	429
16.63.2.2 VRNA_BRACKETS_RND	429
16.63.2.3 VRNA_BRACKETS_CLY	430
16.63.2.4 VRNA_BRACKETS_ANG	430
16.63.2.5 VRNA_BRACKETS_SQR	430
16.63.2.6 VRNA_BRACKETS_DEFAULT	430

16.63.2.7 VRNA_BRACKETS_ANY	430
16.63.3 Function Documentation	431
16.63.3.1 vrna_db_pack()	431
16.63.3.2 vrna_db_unpack()	431
16.63.3.3 vrna_db_flatten()	431
16.63.3.4 vrna_db_flatten_to()	432
16.63.3.5 vrna_db_from_ptable()	432
16.63.3.6 vrna_db_from_plist()	433
16.63.3.7 vrna_db_to_element_string()	433
16.63.3.8 vrna_db_pk_remove()	434
16.64 Washington University Secondary Structure (WUSS) notation	434
16.64.1 Detailed Description	434
16.64.2 Function Documentation	435
16.64.2.1 vrna_db_from_WUSS()	435
16.65 Pair Table Representation of Secondary Structures	436
16.65.1 Detailed Description	436
16.65.2 Function Documentation	436
16.65.2.1 vrna_ptable()	436
16.65.2.2 vrna_ptable_from_string()	437
16.65.2.3 vrna_pt_pk_get()	437
16.65.2.4 vrna_ptable_copy()	438
16.65.2.5 vrna_pt_aliases_get()	438
16.65.2.6 vrna_pt_snoop_get()	438
16.65.2.7 vrna_pt_pk_remove()	438
16.66 Pair List Representation of Secondary Structures	439
16.66.1 Detailed Description	439
16.66.2 Data Structure Documentation	440
16.66.2.1 struct vrna_elem_prob_s	440
16.66.3 Function Documentation	440
16.66.3.1 vrna_plist()	440
16.67 Abstract Shapes Representation of Secondary Structures	440
16.67.1 Detailed Description	440
16.67.2 Function Documentation	441
16.67.2.1 vrna_abstract_shapes()	441
16.67.2.2 vrna_abstract_shapes_pt()	442
16.68 Helix List Representation of Secondary Structures	442
16.68.1 Detailed Description	442
16.68.2 Data Structure Documentation	443
16.68.2.1 struct vrna_hx_s	443
16.68.3 Function Documentation	443
16.68.3.1 vrna_hx_from_ptable()	443
16.69 Tree Representation of Secondary Structures	443

16.69.1 Detailed Description	443
16.69.2 Macro Definition Documentation	445
16.69.2.1 VRNA_STRUCTURE_TREE_HIT	445
16.69.2.2 VRNA_STRUCTURE_TREE_SHAPIRO_SHORT	445
16.69.2.3 VRNA_STRUCTURE_TREE_SHAPIRO	445
16.69.2.4 VRNA_STRUCTURE_TREE_SHAPIRO_EXT	445
16.69.2.5 VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT	446
16.69.2.6 VRNA_STRUCTURE_TREE_EXPANDED	446
16.69.3 Function Documentation	446
16.69.3.1 vrna_db_to_tree_string()	446
16.69.3.2 vrna_tree_string_unweight()	447
16.69.3.3 vrna_tree_string_to_db()	447
16.70 Distance measures between Secondary Structures	447
16.70.1 Detailed Description	447
16.70.2 Function Documentation	448
16.70.2.1 vrna_bp_distance_pt()	448
16.70.2.2 vrna_bp_distance()	448
16.71 Multiple Sequence Alignment Utilities	449
16.71.1 Detailed Description	449
16.71.2 Data Structure Documentation	450
16.71.2.1 struct vrna_pinfo_s	450
16.71.3 Macro Definition Documentation	451
16.71.3.1 VRNA_MEASURE_SHANNON_ENTROPY	451
16.71.4 Function Documentation	451
16.71.4.1 vrna_aln_mpi()	451
16.71.4.2 vrna_aln_pinfo()	451
16.71.4.3 vrna_aln_slice()	452
16.71.4.4 vrna_aln_free()	452
16.71.4.5 vrna_aln_uppercase()	452
16.71.4.6 vrna_aln_toRNA()	453
16.71.4.7 vrna_aln_copy()	453
16.71.4.8 vrna_aln_conservation_struct()	453
16.71.4.9 vrna_aln_conservation_col()	454
16.71.4.10 vrna_aln_consensus_sequence()	454
16.71.4.11 vrna_aln_consensus_mis()	455
16.72 Files and I/O	455
16.72.1 Detailed Description	455
16.72.2 Function Documentation	456
16.72.2.1 get_ribosum()	456
16.72.2.2 readribosum()	456
16.72.2.3 vrna_read_line()	457
16.72.2.4 vrna_filename_sanitize()	457

16.72.2.5 vrna_file_exists()	458
16.73 Nucleic Acid Sequences and Structures	458
16.73.1 Detailed Description	458
16.73.2 Macro Definition Documentation	459
16.73.2.1 VRNA_OPTION_MULTILINE	459
16.73.2.2 VRNA_CONSTRAINT_MULTILINE	459
16.73.3 Function Documentation	459
16.73.3.1 vrna_file_helixlist()	459
16.73.3.2 vrna_file_connect()	460
16.73.3.3 vrna_file_bpseq()	460
16.73.3.4 vrna_file_json()	461
16.73.3.5 vrna_file_fasta_read_record()	461
16.73.3.6 vrna_extract_record_rest_structure()	462
16.73.3.7 vrna_file_SHAPE_read()	463
16.73.3.8 vrna_extract_record_rest_constraint()	463
16.73.3.9 read_record()	464
16.74 Multiple Sequence Alignments	464
16.74.1 Detailed Description	464
16.74.2 Macro Definition Documentation	465
16.74.2.1 VRNA_FILE_FORMAT_MSA_CLUSTAL	465
16.74.2.2 VRNA_FILE_FORMAT_MSA_STOCKHOLM	465
16.74.2.3 VRNA_FILE_FORMAT_MSA_FASTA	465
16.74.2.4 VRNA_FILE_FORMAT_MSA_MAF	465
16.74.2.5 VRNA_FILE_FORMAT_MSA_MIS	466
16.74.2.6 VRNA_FILE_FORMAT_MSA_DEFAULT	466
16.74.2.7 VRNA_FILE_FORMAT_MSA_NOCHECK	466
16.74.2.8 VRNA_FILE_FORMAT_MSA_UNKNOWN	466
16.74.2.9 VRNA_FILE_FORMAT_MSA_APPEND	466
16.74.2.10 VRNA_FILE_FORMAT_MSA_QUIET	467
16.74.2.11 VRNA_FILE_FORMAT_MSA_SILENT	467
16.74.3 Function Documentation	467
16.74.3.1 vrna_file_msa_read()	467
16.74.3.2 vrna_file_msa_read_record()	468
16.74.3.3 vrna_file_msa_detect_format()	469
16.74.3.4 vrna_file_msa_write()	470
16.75 Command Files	471
16.75.1 Detailed Description	471
16.75.2 Macro Definition Documentation	471
16.75.2.1 VRNA_CMD_PARSE_HC	472
16.75.2.2 VRNA_CMD_PARSE_SC	472
16.75.2.3 VRNA_CMD_PARSE_UD	472
16.75.2.4 VRNA_CMD_PARSE_SD	472

16.75.2.5 VRNA_CMD_PARSE_DEFAULTS	472
16.75.3 Function Documentation	472
16.75.3.1 vrna_file_commands_read()	473
16.75.3.2 vrna_file_commands_apply()	473
16.75.3.3 vrna_commands_apply()	473
16.75.3.4 vrna_commands_free()	474
16.76 Plotting	474
16.76.1 Detailed Description	474
16.76.2 Data Structure Documentation	475
16.76.2.1 struct vrna_dotplot_auxdata_t	475
16.76.3 Function Documentation	475
16.76.3.1 PS_dot_plot_list()	476
16.76.3.2 PS_dot_plot()	476
16.76.3.3 vrna_file_PS_rnaplot()	476
16.76.3.4 vrna_file_PS_rnaplot_a()	477
16.76.3.5 gmlRNA()	477
16.76.3.6 ssv_rna_plot()	478
16.76.3.7 svg_rna_plot()	478
16.76.3.8 xrna_plot()	479
16.76.3.9 PS_rna_plot()	479
16.76.3.10 PS_rna_plot_a()	479
16.76.3.11 PS_rna_plot_a_gquad()	479
16.77 Layouts and Coordinates	480
16.77.1 Detailed Description	480
16.77.2 Data Structure Documentation	481
16.77.2.1 struct vrna_plot_layout_s	481
16.77.2.2 struct vrna_plot_options_puzzler_t	481
16.77.3 Macro Definition Documentation	481
16.77.3.1 VRNA_PLOT_TYPE_SIMPLE	481
16.77.3.2 VRNA_PLOT_TYPE_NAVIEW	482
16.77.3.3 VRNA_PLOT_TYPE_CIRCULAR	482
16.77.3.4 VRNA_PLOT_TYPE_TURTLE	482
16.77.3.5 VRNA_PLOT_TYPE_PUZZLER	482
16.77.4 Typedef Documentation	482
16.77.4.1 vrna_plot_layout_t	482
16.77.5 Function Documentation	482
16.77.5.1 vrna_plot_layout()	483
16.77.5.2 vrna_plot_layout_simple()	483
16.77.5.3 vrna_plot_layout_circular()	484
16.77.5.4 vrna_plot_layout_turtle()	484
16.77.5.5 vrna_plot_layout_puzzler()	485
16.77.5.6 vrna_plot_layout_free()	485

16.77.5.7 vrna_plot_coords()	485
16.77.5.8 vrna_plot_coords_pt()	486
16.77.5.9 vrna_plot_coords_simple()	487
16.77.5.10 vrna_plot_coords_simple_pt()	488
16.77.5.11 vrna_plot_coords_circular()	488
16.77.5.12 vrna_plot_coords_circular_pt()	489
16.77.5.13 vrna_plot_coords_puzzler()	490
16.77.5.14 vrna_plot_coords_puzzler_pt()	490
16.77.5.15 vrna_plot_options_puzzler()	491
16.77.5.16 vrna_plot_options_puzzler_free()	491
16.77.5.17 vrna_plot_coords_turtle()	492
16.77.5.18 vrna_plot_coords_turtle_pt()	493
16.78 Annotation	493
16.78.1 Detailed Description	493
16.78.2 Function Documentation	493
16.78.2.1 vrna_annotate_covar_db()	494
16.78.2.2 vrna_annotate_covar_pairs()	494
16.79 Alignment Plots	494
16.79.1 Detailed Description	494
16.79.2 Function Documentation	494
16.79.2.1 vrna_file_PS_aln()	494
16.79.2.2 vrna_file_PS_aln_slice()	495
16.80 Search Algorithms	496
16.80.1 Detailed Description	496
16.80.2 Function Documentation	496
16.80.2.1 vrna_search_BMH_num()	496
16.80.2.2 vrna_search_BMH()	497
16.80.2.3 vrna_search_BM_BCT_num()	497
16.80.2.4 vrna_search_BM_BCT()	498
16.81 Combinatorics Algorithms	498
16.81.1 Detailed Description	498
16.81.2 Function Documentation	499
16.81.2.1 vrna_enumerate_necklaces()	499
16.81.2.2 vrna_rotational_symmetry_num()	500
16.81.2.3 vrna_rotational_symmetry_pos_num()	500
16.81.2.4 vrna_rotational_symmetry()	501
16.81.2.5 vrna_rotational_symmetry_pos()	501
16.81.2.6 vrna_rotational_symmetry_db()	502
16.81.2.7 vrna_rotational_symmetry_db_pos()	502
16.81.2.8 vrna_n_multichoose_k()	503
16.81.2.9 vrna_boustrophedon()	504
16.81.2.10 vrna_boustrophedon_pos()	504

16.82 (Abstract) Data Structures	505
16.82.1 Detailed Description	505
16.82.2 Data Structure Documentation	507
16.82.2.1 struct vrna_basepair_s	507
16.82.2.2 struct vrna_cpair_s	507
16.82.2.3 struct vrna_color_s	507
16.82.2.4 struct vrna_data_linear_s	507
16.82.2.5 struct vrna_sect_s	507
16.82.2.6 struct vrna_bp_stack_s	507
16.82.2.7 struct pu_contrib	507
16.82.2.8 struct interact	507
16.82.2.9 struct pu_out	508
16.82.2.10 struct constrain	508
16.82.2.11 struct duplexT	508
16.82.2.12 struct node	508
16.82.2.13 struct snoopT	508
16.82.2.14 struct dupVar	509
16.82.3 Typedef Documentation	509
16.82.3.1 PAIR	509
16.82.3.2 plist	509
16.82.3.3 cpair	509
16.82.3.4 sect	509
16.82.3.5 bondT	509
16.82.4 Function Documentation	509
16.82.4.1 vrna_C11_features()	510
16.83 Messages	510
16.83.1 Detailed Description	510
16.83.2 Function Documentation	511
16.83.2.1 vrna_message_error()	511
16.83.2.2 vrna_message_verror()	511
16.83.2.3 vrna_message_warning()	511
16.83.2.4 vrna_message_vwarning()	512
16.83.2.5 vrna_message_info()	512
16.83.2.6 vrna_message_vinfo()	512
16.83.2.7 vrna_message_input_seq_simple()	513
16.83.2.8 vrna_message_input_seq()	513
16.84 Unit Conversion	513
16.84.1 Detailed Description	513
16.84.2 Enumeration Type Documentation	514
16.84.2.1 vrna_unit_energy_e	514
16.84.2.2 vrna_unit_temperature_e	515
16.84.3 Function Documentation	515

16.84.3.1 vrna_convert_energy()	515
16.84.3.2 vrna_convert_temperature()	515
16.84.3.3 vrna_convert_kcal_to_dcal()	516
16.84.3.4 vrna_convert_dcal_to_kcal()	516
16.85 The Fold Compound	517
16.85.1 Detailed Description	517
16.85.2 Data Structure Documentation	518
16.85.2.1 struct vrna_fc_s	518
16.85.3 Macro Definition Documentation	524
16.85.3.1 VRNA_STATUS_MFE_PRE	524
16.85.3.2 VRNA_STATUS_MFE_POST	525
16.85.3.3 VRNA_STATUS_PF_PRE	525
16.85.3.4 VRNA_STATUS_PF_POST	525
16.85.3.5 VRNA_OPTION_MFE	525
16.85.3.6 VRNA_OPTION_PF	525
16.85.3.7 VRNA_OPTION_EVAL_ONLY	526
16.85.4 Typedef Documentation	526
16.85.4.1 vrna_auxdata_free_f	526
16.85.4.2 vrna_recursion_status_f	526
16.85.5 Enumeration Type Documentation	527
16.85.5.1 vrna_fc_type_e	527
16.85.6 Function Documentation	527
16.85.6.1 vrna_fold_compound()	527
16.85.6.2 vrna_fold_compound_comparative()	528
16.85.6.3 vrna_fold_compound_free()	529
16.85.6.4 vrna_fold_compound_add_auxdata()	529
16.85.6.5 vrna_fold_compound_add_callback()	530
16.86 The Dynamic Programming Matrices	530
16.86.1 Detailed Description	530
16.86.2 Data Structure Documentation	531
16.86.2.1 struct vrna_mx_mfe_s	531
16.86.2.2 struct vrna_mx_pf_s	532
16.86.3 Enumeration Type Documentation	532
16.86.3.1 vrna_mx_type_e	532
16.86.4 Function Documentation	533
16.86.4.1 vrna_mx_add()	533
16.86.4.2 vrna_mx_mfe_free()	534
16.86.4.3 vrna_mx_pf_free()	534
16.87 Hash Tables	534
16.87.1 Detailed Description	534
16.87.2 Data Structure Documentation	535
16.87.2.1 struct vrna_ht_entry_db_t	535

16.87.3 Typedef Documentation	536
16.87.3.1 vrna_hash_table_t	536
16.87.3.2 vrna_ht_cmp_f	536
16.87.3.3 vrna_ht_hashfunc_f	536
16.87.3.4 vrna_ht_free_f	537
16.87.4 Function Documentation	537
16.87.4.1 vrna_ht_init()	537
16.87.4.2 vrna_ht_size()	538
16.87.4.3 vrna_ht_collisions()	538
16.87.4.4 vrna_ht_get()	538
16.87.4.5 vrna_ht_insert()	539
16.87.4.6 vrna_ht_remove()	539
16.87.4.7 vrna_ht_clear()	539
16.87.4.8 vrna_ht_free()	540
16.87.4.9 vrna_ht_db_comp()	540
16.87.4.10 vrna_ht_db_hash_func()	540
16.87.4.11 vrna_ht_db_free_entry()	541
16.88 Heaps	541
16.88.1 Detailed Description	541
16.88.2 Typedef Documentation	542
16.88.2.1 vrna_heap_t	542
16.88.2.2 vrna_heap_cmp_f	542
16.88.2.3 vrna_heap_get_pos_f	544
16.88.2.4 vrna_heap_set_pos_f	544
16.88.3 Function Documentation	544
16.88.3.1 vrna_heap_init()	544
16.88.3.2 vrna_heap_free()	545
16.88.3.3 vrna_heap_size()	545
16.88.3.4 vrna_heap_insert()	546
16.88.3.5 vrna_heap_pop()	546
16.88.3.6 vrna_heap_top()	546
16.88.3.7 vrna_heap_remove()	547
16.88.3.8 vrna_heap_update()	547
16.89 Arrays	548
16.89.1 Detailed Description	548
16.89.2 Data Structure Documentation	549
16.89.2.1 struct vrna_array_header_s	549
16.89.3 Macro Definition Documentation	550
16.89.3.1 vrna_array_init_size	550
16.89.4 Function Documentation	550
16.89.4.1 vrna__array_set_capacity()	550
16.90 Buffers	550

16.90.1 Detailed Description	550
16.90.2 Typedef Documentation	551
16.90.2.1 vrna_stream_output_f	551
16.90.3 Function Documentation	551
16.90.3.1 vrna_cstr()	551
16.90.3.2 vrna_cstr_discard()	552
16.90.3.3 vrna_cstr_free()	552
16.90.3.4 vrna_cstr_close()	552
16.90.3.5 vrna_cstr_fflush()	553
16.90.3.6 vrna_ostream_init()	553
16.90.3.7 vrna_ostream_free()	553
16.90.3.8 vrna_ostream_request()	554
16.90.3.9 vrna_ostream_provide()	554
16.91 Deprecated Interface for Global MFE Prediction	555
16.91.1 Detailed Description	555
16.91.2 Function Documentation	556
16.91.2.1 alifold()	556
16.91.2.2 cofold()	556
16.91.2.3 cofold_par()	557
16.91.2.4 free_co_arrays()	557
16.91.2.5 update_cofold_params()	557
16.91.2.6 update_cofold_params_par()	557
16.91.2.7 export_cofold_arrays_gq()	558
16.91.2.8 export_cofold_arrays()	558
16.91.2.9 initialize_cofold()	559
16.91.2.10 fold_par()	559
16.91.2.11 fold()	560
16.91.2.12 circfold()	560
16.91.2.13 free_arrays()	561
16.91.2.14 update_fold_params()	561
16.91.2.15 update_fold_params_par()	561
16.91.2.16 export_fold_arrays()	562
16.91.2.17 export_fold_arrays_par()	562
16.91.2.18 export_circfold_arrays()	562
16.91.2.19 export_circfold_arrays_par()	562
16.91.2.20 LoopEnergy()	563
16.91.2.21 HairpinE()	563
16.91.2.22 initialize_fold()	563
16.91.2.23 circalifold()	563
16.91.2.24 free_alifold_arrays()	564
16.92 Deprecated Interface for Local (Sliding Window) MFE Prediction	564
16.92.1 Detailed Description	564

16.92.2 Function Documentation	564
16.92.2.1 Lfold()	565
16.92.2.2 Lfoldz()	565
16.93 Deprecated Interface for Global Partition Function Computation	565
16.93.1 Detailed Description	565
16.93.2 Function Documentation	567
16.93.2.1 alipf_fold_par()	567
16.93.2.2 pf_fold_par()	567
16.93.2.3 pf_fold()	568
16.93.2.4 pf_circ_fold()	569
16.93.2.5 free_pf_arrays()	570
16.93.2.6 update_pf_params()	570
16.93.2.7 update_pf_params_par()	570
16.93.2.8 export_bppm()	571
16.93.2.9 get_pf_arrays()	571
16.93.2.10 get_subseq_F()	572
16.93.2.11 mean_bp_distance()	572
16.93.2.12 mean_bp_distance_pr()	572
16.93.2.13 stackProb()	573
16.93.2.14 init_pf_fold()	573
16.93.2.15 co_pf_fold()	573
16.93.2.16 co_pf_fold_par()	573
16.93.2.17 compute_probabilities()	574
16.93.2.18 init_co_pf_fold()	575
16.93.2.19 export_co_bppm()	575
16.93.2.20 free_co_pf_arrays()	575
16.93.2.21 update_co_pf_params()	575
16.93.2.22 update_co_pf_params_par()	576
16.93.2.23 assign_plist_from_db()	576
16.93.2.24 assign_plist_from_pr()	576
16.93.2.25 alipf_fold()	577
16.93.2.26 alipf_circ_fold()	577
16.93.2.27 export_ali_bppm()	578
16.93.2.28 free_alipf_arrays()	578
16.93.2.29 alipbacktrack()	578
16.93.2.30 get_alipf_arrays()	579
16.94 Deprecated Interface for Local (Sliding Window) Partition Function Computation	580
16.94.1 Detailed Description	580
16.94.2 Function Documentation	580
16.94.2.1 update_pf_paramsLP()	580
16.94.2.2 pfl_fold()	580
16.94.2.3 pfl_fold_par()	581

16.94.2.4 putoutpU_prob()	581
16.94.2.5 putoutpU_prob_bin()	583
16.95 Deprecated Interface for Stochastic Backtracking	583
16.95.1 Detailed Description	583
16.95.2 Function Documentation	583
16.95.2.1 pbacktrack()	584
16.95.2.2 pbacktrack5()	584
16.95.2.3 pbacktrack_circ()	584
16.95.3 Variable Documentation	584
16.95.3.1 st_back	585
16.96 Deprecated Interface for Multiple Sequence Alignment Utilities	585
16.96.1 Detailed Description	585
16.96.2 Typedef Documentation	585
16.96.2.1 pair_info	585
16.96.3 Function Documentation	585
16.96.3.1 get_mpi()	586
16.96.3.2 encode_ali_sequence()	586
16.96.3.3 alloc_sequence_arrays()	586
16.96.3.4 free_sequence_arrays()	587
16.97 Deprecated Interface for Secondary Structure Utilities	588
16.97.1 Detailed Description	588
16.97.2 Function Documentation	589
16.97.2.1 b2HIT()	589
16.97.2.2 b2C()	589
16.97.2.3 b2Shapiro()	590
16.97.2.4 add_root()	590
16.97.2.5 expand_Shapiro()	590
16.97.2.6 expand_Full()	591
16.97.2.7 unexpand_Full()	591
16.97.2.8 unweight()	591
16.97.2.9 unexpand_aligned_F()	591
16.97.2.10 parse_structure()	592
16.97.2.11 pack_structure()	592
16.97.2.12 unpack_structure()	592
16.97.2.13 make_pair_table()	593
16.97.2.14 copy_pair_table()	593
16.97.2.15 alimake_pair_table()	593
16.97.2.16 make_pair_table_snoop()	594
16.97.2.17 bp_distance()	594
16.97.2.18 make_referenceBP_array()	594
16.97.2.19 compute_BPdifferences()	594
16.97.2.20 parenthesis_structure()	595

16.97.2.21 parenthesis_zuker()	595
16.97.2.22 bppm_to_structure()	595
16.97.2.23 bppm_symbol()	595
16.98 Deprecated Interface for Plotting Utilities	596
16.98.1 Detailed Description	596
16.98.2 Data Structure Documentation	596
16.98.2.1 struct COORDINATE	596
16.98.3 Function Documentation	596
16.98.3.1 PS_color_aln()	596
16.98.3.2 aliPS_color_aln()	596
16.98.3.3 simple_xy_coordinates()	597
16.98.3.4 simple_circplot_coordinates()	597
16.98.4 Variable Documentation	598
16.98.4.1 rna_plot_type	598
16.99 Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers	598
16.99.1 Detailed Description	598
16.99.2 Typedef Documentation	598
16.99.2.1 path_t	599
16.99.3 Function Documentation	599
16.99.3.1 find_saddle()	599
16.99.3.2 free_path()	599
16.99.3.3 get_path()	599
17 Data Structure Documentation	601
17.1 _struct_en Struct Reference	601
17.1.1 Detailed Description	601
17.2 energy_corrections Struct Reference	601
17.3 LIST Struct Reference	601
17.4 LST_BUCKET Struct Reference	601
17.5 Postorder_list Struct Reference	601
17.5.1 Detailed Description	601
17.6 swString Struct Reference	602
17.6.1 Detailed Description	602
17.7 Tree Struct Reference	602
17.7.1 Detailed Description	602
17.8 TwoDpfold_vars Struct Reference	602
17.8.1 Detailed Description	603
17.9 vrna_dimer_conc_s Struct Reference	603
17.9.1 Detailed Description	603
17.10 vrna_sc_bp_storage_t Struct Reference	603
17.10.1 Detailed Description	603
17.11 vrna_sc_mod_param_s Struct Reference	603

17.12 vrna_string_header_s Struct Reference	603
17.12.1 Detailed Description	604
17.13 vrna_structured_domains_s Struct Reference	604
17.14 vrna_subopt_sol_s Struct Reference	604
17.14.1 Detailed Description	604
17.15 vrna_unstructured_domain_motif_s Struct Reference	604
18 File Documentation	605
18.1 ViennaRNA/2Dfold.h File Reference	605
18.2 2Dfold.h	605
18.3 ViennaRNA/2Dpfold.h File Reference	607
18.3.1 Detailed Description	608
18.3.2 Function Documentation	608
18.3.2.1 get_TwoDpfold_variables()	608
18.3.2.2 destroy_TwoDpfold_variables()	608
18.3.2.3 TwoDpfoldList()	609
18.3.2.4 TwoDpfold_pbacktrack()	609
18.3.2.5 TwoDpfold_pbacktrack5()	610
18.4 2Dpfold.h	611
18.5 ali_plex.h	613
18.6 ViennaRNA/alifold.h File Reference	613
18.6.1 Detailed Description	614
18.6.2 Function Documentation	614
18.6.2.1 energy_of_alistruct()	615
18.6.2.2 update_alifold_params()	615
18.6.3 Variable Documentation	615
18.6.3.1 cv_fact	615
18.6.3.2 nc_fact	615
18.7 alifold.h	616
18.8 ViennaRNA/aln_util.h File Reference	617
18.8.1 Detailed Description	617
18.9 aln_util.h	617
18.10 ViennaRNA/alphabet.h File Reference	617
18.10.1 Detailed Description	618
18.11 alphabet.h	618
18.12 ViennaRNA/boltzmann_sampling.h File Reference	619
18.12.1 Detailed Description	620
18.13 boltzmann_sampling.h	620
18.14 ViennaRNA/centroid.h File Reference	622
18.14.1 Detailed Description	623
18.14.2 Function Documentation	623
18.14.2.1 get_centroid_struct_pl()	623

18.14.2.2 get_centroid_struct_pr()	623
18.15 centroid.h	623
18.16 ViennaRNA/char_stream.h File Reference	624
18.16.1 Detailed Description	624
18.17 char_stream.h	624
18.18 ViennaRNA/datastructures/char_stream.h File Reference	624
18.18.1 Detailed Description	624
18.19 char_stream.h	625
18.20 ViennaRNA/cofold.h File Reference	627
18.20.1 Detailed Description	627
18.21 cofold.h	628
18.22 ViennaRNA/combinatorics.h File Reference	628
18.22.1 Detailed Description	629
18.23 combinatorics.h	629
18.24 ViennaRNA/commands.h File Reference	630
18.24.1 Detailed Description	630
18.25 commands.h	631
18.26 ViennaRNA/concentrations.h File Reference	631
18.26.1 Detailed Description	632
18.26.2 Function Documentation	632
18.26.2.1 get_concentrations()	632
18.27 concentrations.h	632
18.28 ViennaRNA/constraints.h File Reference	633
18.28.1 Detailed Description	633
18.29 constraints.h	633
18.30 ViennaRNA/constraints/hard.h File Reference	634
18.30.1 Detailed Description	636
18.30.2 Macro Definition Documentation	636
18.30.2.1 VRNA_CONSTRAINT_NO_HEADER	636
18.30.2.2 VRNA_CONSTRAINT_DB_ANG_BRACK	636
18.30.3 Enumeration Type Documentation	636
18.30.3.1 vrna_hc_type_e	636
18.30.4 Function Documentation	637
18.30.4.1 vrna_hc_add_data()	637
18.30.4.2 print_tty_constraint()	637
18.30.4.3 print_tty_constraint_full()	637
18.30.4.4 constrain_ptypes()	637
18.31 hard.h	638
18.32 ViennaRNA/constraints/ligand.h File Reference	641
18.32.1 Detailed Description	641
18.33 ligand.h	641
18.34 sc_cb_intern.h	642

18.35 ViennaRNA/constraints/SHAPE.h File Reference	643
18.35.1 Detailed Description	643
18.35.2 Function Documentation	643
18.35.2.1 vrna_sc_SHAPE_parse_method()	643
18.36 SHAPE.h	644
18.37 ViennaRNA/constraints/soft.h File Reference	645
18.37.1 Detailed Description	646
18.37.2 Enumeration Type Documentation	646
18.37.2.1 vrna_sc_type_e	646
18.38 soft.h	646
18.39 ViennaRNA/constraints/soft_special.h File Reference	649
18.39.1 Detailed Description	650
18.40 soft_special.h	650
18.41 ViennaRNA/constraints_hard.h File Reference	651
18.41.1 Detailed Description	651
18.42 constraints_hard.h	651
18.43 ViennaRNA/constraints_ligand.h File Reference	651
18.43.1 Detailed Description	651
18.44 constraints_ligand.h	651
18.45 ViennaRNA/constraints_SHAPE.h File Reference	651
18.45.1 Detailed Description	651
18.46 constraints_SHAPE.h	652
18.47 ViennaRNA/constraints_soft.h File Reference	652
18.47.1 Detailed Description	652
18.48 constraints_soft.h	652
18.49 ViennaRNA/convert_epars.h File Reference	652
18.49.1 Detailed Description	652
18.50 convert_epars.h	652
18.51 ViennaRNA/data_structures.h File Reference	653
18.51.1 Detailed Description	653
18.52 data_structures.h	653
18.53 ViennaRNA/datastructures/array.h File Reference	653
18.53.1 Detailed Description	654
18.54 array.h	654
18.55 ViennaRNA/datastructures/hash_tables.h File Reference	655
18.55.1 Detailed Description	656
18.56 hash_tables.h	656
18.57 ViennaRNA/datastructures/heap.h File Reference	657
18.57.1 Detailed Description	658
18.58 heap.h	658
18.59 lists.h	659
18.60 string.h	660

18.61 ViennaRNA/dist_vars.h File Reference	660
18.61.1 Detailed Description	661
18.61.2 Variable Documentation	661
18.61.2.1 edit_backtrack	661
18.61.2.2 cost_matrix	661
18.62 dist_vars.h	661
18.63 ViennaRNA/dp_matrices.h File Reference	662
18.63.1 Detailed Description	662
18.64 dp_matrices.h	662
18.65 ViennaRNA/duplex.h File Reference	666
18.65.1 Detailed Description	666
18.66 duplex.h	666
18.67 ViennaRNA/edit_cost.h File Reference	666
18.67.1 Detailed Description	666
18.68 edit_cost.h	666
18.69 ViennaRNA/energy_const.h File Reference	667
18.69.1 Detailed Description	667
18.70 energy_const.h	668
18.71 ViennaRNA/energy_par.h File Reference	668
18.71.1 Detailed Description	668
18.72 energy_par.h	668
18.73 ViennaRNA/equilibrium_probs.h File Reference	668
18.73.1 Detailed Description	669
18.74 equilibrium_probs.h	669
18.75 ViennaRNA/eval.h File Reference	670
18.75.1 Detailed Description	673
18.76 eval.h	673
18.77 ViennaRNA/exterior_loops.h File Reference	677
18.77.1 Detailed Description	677
18.78 exterior_loops.h	677
18.79 ViennaRNA/file_formats.h File Reference	677
18.79.1 Detailed Description	677
18.80 file_formats.h	677
18.81 ViennaRNA/io/file_formats.h File Reference	678
18.81.1 Detailed Description	678
18.82 file_formats.h	678
18.83 ViennaRNA/file_formats_msa.h File Reference	680
18.83.1 Detailed Description	680
18.84 file_formats_msa.h	680
18.85 ViennaRNA/io/file_formats_msa.h File Reference	680
18.85.1 Detailed Description	681
18.86 file_formats_msa.h	681

18.87 ViennaRNA/file_utils.h File Reference	682
18.87.1 Detailed Description	682
18.88 file_utils.h	682
18.89 ViennaRNA/findpath.h File Reference	682
18.89.1 Detailed Description	683
18.90 findpath.h	683
18.91 ViennaRNA/landscape/findpath.h File Reference	683
18.91.1 Detailed Description	683
18.92 findpath.h	683
18.93 ViennaRNA/fold.h File Reference	684
18.93.1 Detailed Description	685
18.94 fold.h	685
18.95 ViennaRNA/fold_compound.h File Reference	687
18.95.1 Detailed Description	688
18.96 fold_compound.h	688
18.97 ViennaRNA/fold_vars.h File Reference	690
18.97.1 Detailed Description	691
18.97.2 Variable Documentation	691
18.97.2.1 RibosumFile	691
18.97.2.2 james_rule	691
18.97.2.3 logML	691
18.97.2.4 cut_point	691
18.97.2.5 base_pair	692
18.97.2.6 pr	692
18.97.2.7 iindx	692
18.98 fold_vars.h	692
18.99 ViennaRNA/gquad.h File Reference	692
18.99.1 Detailed Description	693
18.100 gquad.h	693
18.101 ViennaRNA/grammar.h File Reference	712
18.101.1 Detailed Description	712
18.102 grammar.h	712
18.103 ViennaRNA/hairpin_loops.h File Reference	714
18.103.1 Detailed Description	714
18.104 hairpin_loops.h	714
18.105 ViennaRNA/heat_capacity.h File Reference	714
18.105.1 Detailed Description	715
18.106 heat_capacity.h	715
18.107 ViennaRNA/interior_loops.h File Reference	716
18.107.1 Detailed Description	716
18.108 interior_loops.h	716
18.109 ViennaRNA/inverse.h File Reference	716

18.109.1 Detailed Description	717
18.110 inverse.h	717
18.111 ViennaRNA/landscape/move.h File Reference	717
18.111.1 Detailed Description	718
18.112 move.h	718
18.113 ViennaRNA/landscape/paths.h File Reference	719
18.113.1 Detailed Description	719
18.114 paths.h	720
18.115 ViennaRNA/Lfold.h File Reference	720
18.115.1 Detailed Description	721
18.116 Lfold.h	721
18.117 ViennaRNA/loop_energies.h File Reference	721
18.117.1 Detailed Description	721
18.118 loop_energies.h	722
18.119 ViennaRNA/loops/all.h File Reference	722
18.119.1 Detailed Description	722
18.120 all.h	722
18.121 ViennaRNA/loops/external.h File Reference	722
18.121.1 Detailed Description	723
18.122 external.h	723
18.123 ViennaRNA/loops/hairpin.h File Reference	725
18.123.1 Detailed Description	725
18.124 hairpin.h	725
18.125 ViennaRNA/loops/internal.h File Reference	728
18.125.1 Detailed Description	728
18.126 internal.h	729
18.127 ViennaRNA/loops/multibranch.h File Reference	736
18.127.1 Detailed Description	737
18.128 multibranch.h	737
18.129 ViennaRNA/LPfold.h File Reference	739
18.129.1 Detailed Description	739
18.129.2 Function Documentation	740
18.129.2.1 init_pf_foldLP()	740
18.130 LPfold.h	740
18.131 ViennaRNA/MEA.h File Reference	741
18.131.1 Detailed Description	741
18.132 MEA.h	741
18.133 ViennaRNA/mfe.h File Reference	742
18.133.1 Detailed Description	743
18.134 mfe.h	743
18.135 ViennaRNA/mfe_window.h File Reference	744
18.135.1 Detailed Description	744

18.136 mfe_window.h	744
18.137 ViennaRNA/mm.h File Reference	746
18.137.1 Detailed Description	746
18.137.2 Function Documentation	746
18.137.2.1 vrna_maximum_matching()	746
18.137.2.2 vrna_maximum_matching_simple()	746
18.138 mm.h	747
18.139 ViennaRNA/model.h File Reference	747
18.139.1 Detailed Description	752
18.140 model.h	752
18.141 move_set.h	757
18.142 ViennaRNA/multibranch_loops.h File Reference	758
18.142.1 Detailed Description	758
18.143 multibranch_loops.h	758
18.144 ViennaRNA/naview.h File Reference	758
18.144.1 Detailed Description	758
18.145 naview.h	758
18.146 ViennaRNA/landscape/neighbor.h File Reference	759
18.146.1 Detailed Description	759
18.147 neighbor.h	759
18.148 ViennaRNA/neighbor.h File Reference	760
18.148.1 Detailed Description	760
18.149 neighbor.h	761
18.150 pair_mat.h	761
18.151 ViennaRNA/params.h File Reference	763
18.151.1 Detailed Description	763
18.152 params.h	763
18.153 ViennaRNA/params/1.8.4_epars.h File Reference	763
18.153.1 Detailed Description	763
18.154 1.8.4_epars.h	764
18.155 ViennaRNA/params/1.8.4_intloops.h File Reference	768
18.155.1 Detailed Description	768
18.156 1.8.4_intloops.h	768
18.157 ViennaRNA/constraints/basic.h File Reference	915
18.157.1 Detailed Description	916
18.158 basic.h	916
18.159 ViennaRNA/datastructures/basic.h File Reference	917
18.159.1 Detailed Description	918
18.160 basic.h	919
18.161 ViennaRNA/params/basic.h File Reference	921
18.161.1 Detailed Description	922
18.162 basic.h	923

18.163 ViennaRNA/utils/basic.h File Reference	925
18.163.1 Detailed Description	927
18.163.2 Function Documentation	927
18.163.2.1 get_line()	928
18.163.2.2 print_tty_input_seq()	928
18.163.2.3 print_tty_input_seq_str()	928
18.163.2.4 warn_user()	928
18.163.2.5 nrerror()	928
18.163.2.6 space()	929
18.163.2.7 xrealloc()	929
18.163.2.8 init_rand()	929
18.163.2.9 urn()	929
18.163.2.10 int_urn()	929
18.163.2.11 filecopy()	929
18.163.2.12 time_stamp()	930
18.164 basic.h	930
18.165 ViennaRNA/params/constants.h File Reference	932
18.165.1 Detailed Description	932
18.165.2 Macro Definition Documentation	932
18.165.2.1 GASCONST	932
18.165.2.2 K0	933
18.165.2.3 INF	933
18.165.2.4 FORBIDDEN	933
18.165.2.5 BONUS	933
18.165.2.6 NBPAIRS	933
18.165.2.7 TURN	933
18.165.2.8 MAXLOOP	933
18.166 constants.h	933
18.167 ViennaRNA/params/convert.h File Reference	933
18.167.1 Detailed Description	934
18.168 convert.h	934
18.169 default.h	935
18.170 intl11.h	936
18.171 intl11dH.h	940
18.172 intl21.h	945
18.173 intl21dH.h	968
18.174 intl22.h	991
18.175 intl22dH.h	1106
18.176 ViennaRNA/params/io.h File Reference	1221
18.176.1 Detailed Description	1221
18.177 io.h	1222
18.178 ViennaRNA/params/salt.h File Reference	1223

18.178.1 Detailed Description	1223
18.179 salt.h	1223
18.180 ViennaRNA/part_func.h File Reference	1224
18.180.1 Detailed Description	1225
18.180.2 Function Documentation	1225
18.180.2.1 centroid()	1225
18.180.2.2 get_centroid_struct_gquad_pr()	1226
18.180.2.3 mean_bp_dist()	1226
18.180.2.4 expLoopEnergy()	1226
18.180.2.5 expHairpinEnergy()	1226
18.181 part_func.h	1227
18.182 ViennaRNA/part_func_co.h File Reference	1229
18.182.1 Detailed Description	1230
18.182.2 Function Documentation	1230
18.182.2.1 get_plist()	1230
18.183 part_func_co.h	1230
18.184 ViennaRNA/part_func_up.h File Reference	1231
18.184.1 Detailed Description	1232
18.185 part_func_up.h	1232
18.186 ViennaRNA/part_func_window.h File Reference	1232
18.186.1 Detailed Description	1233
18.187 part_func_window.h	1233
18.188 ViennaRNA/perturbation_fold.h File Reference	1235
18.188.1 Detailed Description	1235
18.189 perturbation_fold.h	1236
18.190 pf_multifold.h	1236
18.191 ViennaRNA/pk_plex.h File Reference	1236
18.191.1 Detailed Description	1237
18.192 pk_plex.h	1237
18.193 PKplex.h	1238
18.194 plex.h	1239
18.195 ViennaRNA/plot_aln.h File Reference	1239
18.195.1 Detailed Description	1239
18.196 plot_aln.h	1240
18.197 ViennaRNA/plot_layouts.h File Reference	1240
18.197.1 Detailed Description	1240
18.198 plot_layouts.h	1240
18.199 ViennaRNA/plot_structure.h File Reference	1240
18.199.1 Detailed Description	1240
18.200 plot_structure.h	1240
18.201 ViennaRNA/plot_utils.h File Reference	1241
18.201.1 Detailed Description	1241

18.202 plot_utils.h	1241
18.203 ViennaRNA/plotting/alignments.h File Reference	1241
18.203.1 Detailed Description	1241
18.204 alignments.h	1242
18.205 ViennaRNA/utls/alignments.h File Reference	1242
18.205.1 Detailed Description	1244
18.206 alignments.h	1244
18.207 ViennaRNA/plotting/layouts.h File Reference	1246
18.207.1 Detailed Description	1247
18.208 layouts.h	1247
18.209 ViennaRNA/plotting/probabilities.h File Reference	1249
18.209.1 Detailed Description	1249
18.210 probabilities.h	1249
18.211 ViennaRNA/plotting/RNApuzzler/RNApuzzler.h File Reference	1251
18.211.1 Detailed Description	1251
18.212 RNApuzzler.h	1251
18.213 ViennaRNA/plotting/RNApuzzler/RNAturtle.h File Reference	1252
18.213.1 Detailed Description	1252
18.214 RNAturtle.h	1252
18.215 ViennaRNA/plotting/structures.h File Reference	1253
18.215.1 Detailed Description	1253
18.216 structures.h	1253
18.217 ViennaRNA/utls/structures.h File Reference	1254
18.217.1 Detailed Description	1258
18.218 structures.h	1258
18.219 ProfileAln.h	1262
18.220 ViennaRNA/profiledist.h File Reference	1262
18.220.1 Detailed Description	1262
18.220.2 Function Documentation	1262
18.220.2.1 profile_edit_distance()	1263
18.220.2.2 Make_bp_profile_bppm()	1263
18.220.2.3 free_profile()	1263
18.220.2.4 Make_bp_profile()	1263
18.221 profiledist.h	1263
18.222 ViennaRNA/PS_dot.h File Reference	1264
18.222.1 Detailed Description	1264
18.223 PS_dot.h	1264
18.224 ViennaRNA/read_epars.h File Reference	1264
18.224.1 Detailed Description	1264
18.225 read_epars.h	1265
18.226 ViennaRNA/ribo.h File Reference	1265
18.226.1 Detailed Description	1265

18.227 ribo.h	1265
18.228 ViennaRNA/RNAstruct.h File Reference	1265
18.228.1 Detailed Description	1266
18.229 RNAstruct.h	1266
18.230 ViennaRNA/search/BoyerMoore.h File Reference	1267
18.230.1 Detailed Description	1268
18.231 BoyerMoore.h	1268
18.232 ViennaRNA/sequence.h File Reference	1268
18.232.1 Detailed Description	1269
18.233 sequence.h	1269
18.234 snofold.h	1270
18.235 snoop.h	1271
18.236 special_const.h	1274
18.237 ViennaRNA/datastructures/stream_output.h File Reference	1274
18.237.1 Detailed Description	1274
18.238 stream_output.h	1274
18.239 ViennaRNA/stream_output.h File Reference	1275
18.239.1 Detailed Description	1275
18.240 stream_output.h	1275
18.241 ViennaRNA/string_utils.h File Reference	1276
18.241.1 Detailed Description	1276
18.242 string_utils.h	1276
18.243 ViennaRNA/stringdist.h File Reference	1276
18.243.1 Detailed Description	1276
18.243.2 Function Documentation	1276
18.243.2.1 Make_swString()	1276
18.243.2.2 string_edit_distance()	1277
18.244 stringdist.h	1277
18.245 ViennaRNA/structure_utils.h File Reference	1277
18.245.1 Detailed Description	1277
18.246 structure_utils.h	1277
18.247 ViennaRNA/structured_domains.h File Reference	1278
18.247.1 Detailed Description	1278
18.248 structured_domains.h	1278
18.249 ViennaRNA/subopt.h File Reference	1278
18.249.1 Detailed Description	1279
18.249.2 Typedef Documentation	1279
18.249.2.1 SOLUTION	1279
18.250 subopt.h	1279
18.251 subopt_zuker.h	1280
18.252 ViennaRNA/svm_utils.h File Reference	1281
18.252.1 Detailed Description	1281

18.253 svm_utils.h	1281
18.254 ViennaRNA/treedist.h File Reference	1281
18.254.1 Detailed Description	1281
18.254.2 Function Documentation	1281
18.254.2.1 make_tree()	1281
18.254.2.2 tree_edit_distance()	1282
18.254.2.3 free_tree()	1282
18.255 treedist.h	1282
18.256 ViennaRNA/units.h File Reference	1283
18.256.1 Detailed Description	1283
18.257 units.h	1283
18.258 ViennaRNA/units/units.h File Reference	1283
18.258.1 Detailed Description	1284
18.259 units.h	1284
18.260 ViennaRNA/unstructured_domains.h File Reference	1284
18.260.1 Detailed Description	1286
18.260.2 Function Documentation	1286
18.260.2.1 vrna_ud_get_motif_size_at()	1286
18.260.2.2 vrna_ud_set_prob_cb()	1286
18.261 unstructured_domains.h	1286
18.262 ViennaRNA/io/units.h File Reference	1289
18.262.1 Detailed Description	1289
18.263 utils.h	1289
18.264 ViennaRNA/plotting/utils.h File Reference	1290
18.264.1 Detailed Description	1290
18.265 utils.h	1290
18.266 ViennaRNA/utils.h File Reference	1290
18.266.1 Detailed Description	1290
18.267 utils.h	1291
18.268 cpu.h	1291
18.269 higher_order_functions.h	1291
18.270 ViennaRNA/utils/strings.h File Reference	1291
18.270.1 Detailed Description	1293
18.270.2 Function Documentation	1293
18.270.2.1 str_uppercase()	1293
18.270.2.2 str_DNA2RNA()	1293
18.270.2.3 random_string()	1293
18.270.2.4 hamming()	1294
18.270.2.5 hamming_bound()	1294
18.271 strings.h	1294
18.272 svm.h	1296
18.273 vrna_config.h	1296

18.274 ViennaRNA/landscape/walk.h File Reference	1297
18.274.1 Detailed Description	1298
18.275 walk.h	1298
18.276 ViennaRNA/walk.h File Reference	1299
18.276.1 Detailed Description	1299
18.277 walk.h	1299
18.278 wrap_dlib.h	1299
18.279 zscore.h	1299
Bibliography	1302
Index	1303

Chapter 1

RNAlib-2.6.0b

1.1 A Library for predicting and comparing RNA secondary structures

The core of the ViennaRNA Package ([19], [14]) is formed by a collection of routines for the prediction and comparison of RNA secondary structures. These routines can be accessed through stand-alone programs, such as `RNAfold`, `RNAdistance` etc., which should be sufficient for most users. For those who wish to develop their own programs we provide a library which can be linked to your own code.

This document describes the library and will be primarily useful to programmers. However, it also contains details about the implementation that may be of interest to advanced users. The stand-alone programs are described in separate man pages. The latest version of the package including source code and html versions of the documentation can be found at

<http://www.tbi.univie.ac.at/RNA>

Date

1994-2020

Authors

Ivo Hofacker, Peter Stadler, Ronny Lorenz, and so many more

1.2 License

Disclaimer and Copyright

The programs, library and source code of the Vienna RNA Package are free software. They are distributed in the hope that they will be useful but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Permission is granted for research, educational, and commercial use and modification so long as 1) the package and any derived works are not redistributed for any fee, other than media costs, 2) proper credit is given to the authors and the Institute for Theoretical Chemistry of the University of Vienna.

If you want to include this software in a commercial product, please contact the authors.

1.3 Contributors

Over the past decades since the `ViennaRNA Package` first sprang to life as part of Ivo Hofackers PhD project, several different authors contributed more and more algorithm implementations. In 2008, Ronny Lorenz took over the extensive task to harmonize and simplify the already existing implementations for the sake of easier feature addition. This eventually lead to version 2.0 of the `ViennaRNA Package`. Since then, he (re-)implemented a large portion of the currently existing library features, such as the new, generalized constraints framework, RNA folding grammar domain extensions, and the major part of the scripting language interface. Below is a list of most people who contributed larger parts of the implementations:

- Daniel Wiegreffe (RNAturtle and RNApuzzler secondary structure layouts)
- Andreas Gruber (first approach on RNALfold Z-score filtering)
- Juraj Michalik (non-redundant Boltzmann sampling)
- Gregor Entzian (neighbor, walk)
- Mario Koestl (worked on SWIG interface and related unit testing)
- Dominik Luntzer (perturbation fold)
- Stefan Badelt (cofold evaluation, RNAdesign.pl, cofold findpath extensions)
- Stefan Hammer (parts of SWIG interface and corresponding unit tests)
- Ronny Lorenz (circfold, version 2.0, generic constraints, grammar extensions, and much more)
- Hakim Tafer (RNAplex, RNAsnoop)
- Ulrike Mueckstein (RNAup)
- Stephan Bernhart (RNAcofold, RNApfold, unpaired probabilities, alifold, and so many more)
- Stefan Wuchty (RNAsubopt)
- Ivo Hofacker, Peter Stadler, and Christoph Flamm (almost every implementation up to version 1.8.5)

We also want to thank the following people:

- Sebastian Bonhoeffer's implementation of partition function folding served as a precursor to our `part_func.c`
- Manfred Tacker hacked constrained folding into `fold.c` for the first time
- Martin Fekete made the first attempts at "alignment folding"
- Andrea Tanzer and Martin Raden (Mann) for not stopping to report bugs found through comprehensive usage of our applications and RNAlib
- Thanks also to everyone else who helped testing and finding bugs, especially Christoph Flamm, Martijn Huynen, Baerbel Krakhofer, and many more

If you want to get involved in the development of the `ViennaRNA Package` yourself, please read the [Contributing page](#).

Chapter 2

Getting Started

- [Installation and Configuration](#) describes how to install and configure `RNAlib` for your requirements
- [HelloWorld](#) presents some small example programs to get a first impression on how to use this library
- [HelloWorld \(Perl/Python\)](#) contains small examples that show how to use `RNAlib` even without C/C++ programming skills from within your favorite scripting language

2.1 Installation and Configuration

A documentation on how to configure the different features of `RNAlib`, how to install the ViennaRNA Package, and finally, how to link you own programs against `RNAlib`.

2.1.1 Installing the ViennaRNA Package

For best portability the ViennaRNA package uses the GNU autoconf and automake tools. The instructions below are for installing the ViennaRNA package from source. However, pre-compiled binaries for various Linux distributions, as well as for Windows users are available from Download section of the [main ViennaRNA homepage](#).

2.1.1.1 Quick-start

Usually you'll just unpack, configure and make. To do this type:

```
tar -zxvf ViennaRNA-2.6.0b.tar.gz
cd ViennaRNA-2.6.0b
./configure
make
sudo make install
```

2.1.1.2 Installation without root privileges

If you do not have root privileges on your computer, you might want to install the ViennaRNA Package to a location where you actually have write access to. To do so, you can set the installation prefix of the `./configure` script like so:

```
./configure --prefix=/home/username/ViennaRNA
make install
```

This will install the entire ViennaRNA Package into a new directory `ViennaRNA` directly into the users `username` home directory.

2.1.1.3 Notes for MacOS X users

2.1.1.3.1 Compilation Although users will find `/usr/bin/gcc` and `/usr/bin/g++` executables in their directory tree, these programs are not at all what they pretend to be. Instead of including the GNU programs, Apple decided to install clang/llvm in disguise. Unfortunately, the default version of clang/llvm does not support OpenMP (yet), but only complains at a late stage of the build process when this support is required. Therefore, it seems necessary to deactivate OpenMP support by passing the option `--disable-openmp` to the `./configure` script.

2.1.1.3.2 Missing EXTERN.h include file Furthermore, as far as we are informed, users are discouraged to use the Perl 5 interpreter that is shipped with Mac OS X. Instead, one should install a more recent version from another source, e.g. `homebrew`. If, however, for any reason you do not want to install your own Perl 5 interpreter but use the one from Apple, you need to specify its include path to enable building the ViennaRNA Perl interface. Otherwise, the file `EXTERN.h` will be missing at compile time. To fix this problem, you first need to find out where `EXTERN.h` is located:

```
sudo find /Library -type f -name EXTERN.h
```

Then choose the one that corresponds to your default perl interpreter (find out the version number with `perl -v | grep version`), simply execute the following before running the `./configure` script, e.g.:

```
export CPATH=/Library/Developer/CommandLineTools/SDKs/MacOSX10.15.sdk/System/Library/Perl/5.18/darwin-thread-m
```

if your default perl is v5.18 running on MacOSX10.15. Change the paths according to your current setup. After that, running `./configure` and compilation should run fine.

See also <https://stackoverflow.com/questions/52682304/fatal-error-extern-h-file-not-found>

2.1.1.3.3 Universal binaries Additionally, if you intend to build the ViennaRNA such that it runs on both, x86_64 and the armv8 (such as for the M1 processors in recent MacBooks), architectures, you need to build a so-called universal binary. Note, however, that to accomplish this task, you might need to deactivate any third-party library dependency as in most cases, only one architecture will be available at link time. This includes the Perl 5 and Python interfaces but also MPFR and GSL support, possibly even more. In order to compile and link the programs, library, and scripting language interfaces of the ViennaRNA Package for multiple architectures, we've added a new configure switch that sets up the required changes automatically:

```
./configure --enable-universal-binary
```

Note

Note, that with link time optimization turned on, MacOS X's default compiler (Illum/clang) generates an intermediary binary format that can not easily be combined into a multi-architecture library. Therefore, the `--enable-universal-binary` switch turns off link time optimization!

2.1.2 Configuring RNAlib features

The ViennaRNA Package includes additional executable programs such as RNAforester, Kinfold, and Kinwalker. Furthermore, we include several features in our C-library that may be activated by default, or have to be explicitly turned on at configure-time. Below we list a selection of the available configure options that affect the features included in all executable programs, the RNAlib C-library, and the corresponding scripting language interface(s).

2.1.2.1 Streaming SIMD Extension (SSE) support

Since version 2.3.5 our sources contain code that implements a faster multibranch loop decomposition in global MFE predictions, as used e.g. in RNAfold. This implementation makes use of modern processors capability to execute particular instructions on multiple data simultaneously (SIMD - single instruction multiple data, thanks to W. B. Langdon for providing the modified code). Consequently, the time required to assess the minimum of all multibranch loop decompositions is reduced up to about one half compared to the runtime of the original implementation. This feature is enabled by default since version 2.4.11 and a dispatcher ensures that the correct implementation will be selected at runtime. If for any reason you want to disable this feature at compile-time use the following configure flag:

```
./configure --disable-simd
```

2.1.2.2 Scripting Interfaces

The ViennaRNA Package comes with scripting language interfaces for Perl 5, Python 3.x, and Python 2.x (provided by swig), that allow one to use the implemented algorithms directly without the need of calling an executable program. The interfaces are build by default whenever the autoconf tool-chain detects the required build tools on your system. You may, however, explicitly turn off particular scripting language interface support at configure-time, for instance for Perl 5 and Python 2, before the actual installation.

Example:

```
./configure --without-perl --without-python2
```

Disabling the scripting language support all-together can be accomplished using the following switch:

```
./configure --without-swig
```

2.1.2.3 Cluster Analysis

The programs AnalyseSeqs and AnalyseDists offer some cluster analysis tools (split decomposition, statistical geometry, neighbor joining, Ward's method) for sequences and distance data. To also build these programs add

```
--with-cluster
```

to your configure options.

2.1.2.4 Kinfold

The Kinfold program can be used to simulate the folding dynamics of an RNA molecule, and is compiled by default. Use the

```
--without-kinfold
```

option to skip compilation and installation of Kinfold.

2.1.2.5 RNAforester

The RNAforester program is used for comparing secondary structures using tree alignment. Similar to Kinfold, use the

```
--without-forester
```

option to skip compilation and installation of RNAforester.

2.1.2.6 Kinwalker

The Kinwalker algorithm performs co-transcriptional folding of RNAs, starting at a user specified structure (default↵: open chain) and ending at the minimum free energy structure. Compilation and installation of this program is deactivated by default. Use the

```
--with-kinwalker
```

option to enable building and installation of Kinwalker.

2.1.2.7 Link Time Optimization (LTO)

To increase the performance of our implementations, the ViennaRNA Package tries to make use of the Link Time Optimization (LTO) feature of modern C-compilers. If you are experiencing any troubles at make-time or run-time, or the configure script for some reason detects that your compiler supports this feature although it doesn't, you can deactivate it using the flag

```
./configure --disable-lto
```

Note, that GCC before version 5 is known to produce unreliable LTO code, especially in combination with SIMD (see [Streaming SIMD Extension \(SSE\) support](#)). We therefore recommend using a more recent compiler (GCC 5 or above) or to turn off one of the two features, LTO or SIMD optimized code.

2.1.2.8 OpenMP support

To enable concurrent computation of our implementations and in some cases parallelization of the algorithms we make use of the OpenMP API. This interface is well understood by most modern compilers. However, in some cases it might be necessary to deactivate OpenMP support and therefore transform *RNAlib* into a C-library that is not entirely *thread-safe*. To do so, add the following configure option

```
./configure --disable-openmp
```

2.1.2.9 POSIX threads (pthread) support

To enable concurrent computation of multiple input data in RNAfold, and for our implementation of the concurrent unordered insert, ordered output flush data structure `vrna_ostream_t` we make use of POSIX threads. This should be supported on all modern platforms and usually does not pose any problems. Unfortunately, we use a threadpool implementation that is not compatible with Microsoft Windows yet. Thus, POSIX thread support can not be activated for Windows builds until we have fixed this problem. If you want to compile RNAfold and RNAlib without POSIX threads support for any other reasons, add the following configure option

```
./configure --disable-pthreads
```


2.1.2.10 SVM Z-score filter in RNALfold

By default, RNALfold that comes with the ViennaRNA Package allows for z-score filtering of its predicted results using a support vector machine (SVM). However, the library we use to implement this feature (libsvm) is statically linked to our own RNALib. If this introduces any problems for your own third-party programs that link against RNALib, you can safely switch off the z-scoring implementation using

```
./configure --without-svm
```

2.1.2.11 GNU Scientific Library

The new program RNApvm computes a pseudo-energy perturbation vector that aims to minimize the discrepancy of predicted, and observed pairing probabilities. For that purpose it implements several methods to solve the optimization problem. Many of them are provided by the GNU Scientific Library, which is why the RNApvm program, and the RNALib C-library are required to be linked against libgsl. If this introduces any problems in your own third-party programs that link against RNALib, you can turn off a larger portion of available minimizers in RNApvm and linking against libgsl all-together, using the switch

```
./configure --without-gsl
```

2.1.2.12 Disable C11/C++11 feature support

By default, we use C11/C++11 features in our implementations. This mainly accounts for unnamed unions/structs within *RNALib*. The configure script automatically detects whether or not your compiler understands these features. In case you are using an older compiler, these features will be deactivated by setting a specific pre-processor directive. If for some reason you want to deactivate C11/C++11 features despite the capabilities of your compiler, use the following configure option:

```
./configure --disable-c11
```

2.1.2.13 Enable warnings for use of deprecated symbols

Since version 2.2 we are in the process of transforming the API of our *RNALib*. Hence, several symbols are marked as *deprecated* whenever they have been replaced by the new API. By default, deprecation warnings at compile time are deactivated. If you want to get your terminal spammed by tons of deprecation warnings, enable them using:

```
./configure --enable-warn-deprecated
```

2.1.2.14 Single precision partition function

Calculation of partition functions (via RNAfold -p) uses double precision floats by default, to avoid overflow errors on longer sequences. If your machine has little memory and you don't plan to fold sequences over 1000 bases in length you can compile the package to do the computations in single precision by running

```
./configure --enable-floatpf
```

Note

Using this option is discouraged and not necessary on most modern computers.

2.1.2.15 Help

For a complete list of all `./configure` options and important environment variables, type

```
./configure --help
```

For more general information on the build process see the `INSTALL` file.

2.1.3 Linking against RNAlib

In order to use our implemented algorithms you simply need to link your program to our *RNAlib* C-library that usually comes along with the ViennaRNA Package installation. If you've installed the ViennaRNA Package as a pre-build binary package, you probably need the corresponding development package, e.g. *viennarna-devel*, or *viennarna-dev*. The only thing that is left is to include the ViennaRNA header files into your source code, e.g.:

```
#include <ViennaRNA/mfe.h>
```

and start using our fast and efficient algorithm implementations.

See also

In the [C Examples](#) and [Some Examples using RNAlib API v3.0](#) sections, we list a small set of example code that usually is a good starting point for your application.

2.1.3.1 Compiler and Linker flags

Of course, simply adding the ViennaRNA header files into your source code is usually not enough. You probably need to tell your compiler where to find the header files, and sometimes add additional pre-processor directives. Whenever your installation of *RNAlib* was build with default settings and the header files were installed into their default location, a simple

```
-I/usr/include
```

pre-processor/compile flag should suffice. It can even be omitted in this case, since your compiler should search this directory by default anyway. You only need to change the path from `/usr/include` to the correct location whenever the header files have been installed into a non-standard directory.

On the other hand, if you've compiled *RNAlib* with some non-default settings then you probably need to define some additional pre-processor macros:

- `VRNA_DISABLE_C11_FEATURES` . . . Disable C11/C++11 features.

Warning

Add this directive to your pre-processor/compile flags only if *RNAlib* was build with the `--disable-c11` configure option.

See also

[Disable C11/C++11 feature support](#) and `vrna_C11_features()`

- `VRNA_WARN_DEPRECATED` . . . Enable warnings for using deprecated symbols.

Note

Adding this directive enables compiler warnings whenever you use symbols in *RNAlib* that are marked *deprecated*.

See also

[Enable warnings for use of deprecated symbols](#) and [Deprecated List](#)

- `USE_FLOAT_PF` . . . Use single precision floating point operations instead of double precision in partition function computations.

Warning

Define this macro only if *RNAlib* was build with the `--enable-floatpf` configure option!

See also

[Single precision partition function](#)

Simply add the corresponding definition(s) to your pre-processor/compile flags, for instance:

```
-DVRNA_DISABLE_C11_FEATURES
```

Finally, linking against *RNAlib* is achieved by adding the following linker flag

```
-L/usr/lib -lRNA -fopenmp
```

Again, the path to the library, `/usr/lib`, may be omitted if this path is searched for libraries by default. The second flag tells the linker to include *libRNA.a*, and the remaining two flags activate [Link Time Optimization \(LTO\)](#) and [OpenMP support](#) support, respectively.

Note

Depending on your linker, the last two flags may differ.

Depending on your configure time decisions, you can drop one or both of the last flags.

In case you've compiled *RNAlib* with LTO support (See [Link Time Optimization \(LTO\)](#)) and you are using the same compiler for your third-party project that links against our library, you may add the

```
-flto
```

flag to enable Link Time Optimization.

2.1.3.2 The pkg-config tool

Instead of hard-coding the required compiler and linker flags, you can also let the *pkg-config* tool automatically determine the required flags. This tool is usually packaged for any Linux distribution and should be available for MacOS X and MinGW as well. We ship a file *RNAlib2.pc* which is installed along with the static *libRNA.a* C-library and populated with all required compiler and linker flags that correspond to your configure time decisions.

The compiler flags required for properly building your code that uses *RNAlib* can be easily obtained via

```
pkg-config --cflags RNAlib2
```

You get the corresponding linker flags using

```
pkg-config --libs RNAlib2
```

With this widely accepted standard it is also very easy to integrate *RNAlib* in your *autotools* project, just have a look at the `PKG_CHECK_MODULES` macro.

2.2 HelloWorld

Below, you'll find some more or less simple C programs showing first steps into using *RNAlib*. A complete list of example C programs can be found in the [C Examples](#) section.

Simple MFE prediction for a given sequence

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/fold.h>
#include <ViennaRNA/Utils/basic.h>

int
main()
{
    /* The RNA sequence */
    char *seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGCAGACUAACA";

    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_fold(seq, structure);

    /* print sequence, structure and MFE */
    printf("%s\n%s [ %.2f ]\n", seq, structure, mfe);

    /* cleanup memory */
    free(structure);

    return 0;
}
```

See also

`examples/helloworld_mfe.c` in the source code tarball

Simple MFE prediction for a multiple sequence alignment

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/alifold.h>
#include <ViennaRNA/Utils/basic.h>
#include <ViennaRNA/Utils/alignments.h>

int
main()
{
    /* The RNA sequence alignment */
    const char *sequences[] = {
        "CUGCCUCACAACGUUUGUGCCUCAGUUACCCGUAGAUGUAGUGAGGGU",
        "CUGCCUCACAACAUUUUGUGCCUCAGUUACUACAUAGAUGUAGUGAGGGU",
        "---CUCGACACCACU---GCCUCGGUACCCAUCCGUGCAGUGCGGGU",
        NULL /* indicates end of alignment */
    };

    /* compute the consensus sequence */
    char *cons = consensus(sequences);

    /* allocate memory for MFE consensus structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(sequences[0]) + 1));

    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_alifold(sequences, structure);

    /* print consensus sequence, structure and MFE */
    printf("%s\n%s [ %.2f ]\n", cons, structure, mfe);

    /* cleanup memory */
    free(cons);
    free(structure);

    return 0;
}
```

See also

examples/helloworld_mfe_comparative.c in the source code tarball

Simple Base Pair Probability computation

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/fold.h>
#include <ViennaRNA/part_func.h>
#include <ViennaRNA/utils/basic.h>

int
main()
{
    /* The RNA sequence */
    char      *seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGACAGUAACA";

    /* allocate memory for pairing propensity string (length + 1) */
    char      *propensity = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* pointers for storing and navigating through base pair probabilities */
    vrna_ep_t *ptr, *pair_probabilities = NULL;

    float      en = vrna_pf_fold(seq, propensity, &pair_probabilities);

    /* print sequence, pairing propensity string and ensemble free energy */
    printf("%s\n%s [ %.2f ]\n", seq, propensity, en);

    /* print all base pairs with probability above 50% */
    for (ptr = pair_probabilities; ptr->i != 0; ptr++)
        if (ptr->p > 0.5)
            printf("p(%d, %d) = %g\n", ptr->i, ptr->j, ptr->p);

    /* cleanup memory */
    free(pair_probabilities);
    free(propensity);

    return 0;
}
```

See also

examples/helloworld_probabilities.c in the source code tarball

Deviating from the Default Model

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/model.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/mfe.h>

int
main()
{
    /* initialize random number generator */
    vrna_init_rand();

    /* Generate a random sequence of 50 nucleotides */
    char      *seq = vrna_random_string(50, "ACGU");

    /* allocate memory for MFE structure (length + 1) */
    char      *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* create a new model details structure to store the Model Settings */
    vrna_md_t md;

    /* ALWAYS set default model settings first! */
    vrna_md_set_default(&md);

    /* change temperature and activate G-Quadruplex prediction */
    md.temperature = 25.0; /* 25 Deg Celcius */
}
```

```

md.gquad      = 1;      /* Turn-on G-Quadruples support */

/* create a fold compound */
vrna_fold_compound_t *fc = vrna_fold_compound(seq, &md, VRNA_OPTION_DEFAULT);

/* predict Minmum Free Energy and corresponding secondary structure */
float          mfe = vrna_mfe(fc, structure);

/* print sequence, structure and MFE */
printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);

/* cleanup memory */
free(structure);
vrna_fold_compound_free(fc);

return 0;
}

```

See also

examples/fold_compound_md.c in the source code tarball

2.3 HelloWorld (Perl/Python)

2.3.1 Perl5

Simple MFE prediction for a given sequence

```

use RNA;

# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCCUAUGUUUGUGACUCGCAGACUAACA";

# compute minimum free energy (MFE) and corresponding structure
my ($ss, $mfe) = RNA::fold($seq);

# print output
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;

```

Simple MFE prediction for a multiple sequence alignment

```

use RNA;

# The RNA sequence alignment
my @sequences = (
    "CUGCCUCACAACGUUUUGUGCCUCAGUUACCCGUAGAUGUAGUGAGGGU",
    "CUGCCUCACAACAUAUUUGUGCCUCAGUUACUAAGAUGUAGUGAGGGU",
    "---CUCGACACCACU---GCCUCGGUUAACCAUCGGUGCAGUGCGGGU"
);

# compute the consensus sequence
my $cons = RNA::consensus(\@sequences);

# predict Minmum Free Energy and corresponding secondary structure
my ($ss, $mfe) = RNA::alifold(\@sequences);

# print output
printf "%s\n%s [ %6.2f ]\n", $cons, $ss, $mfe;

```

Deviating from the Default Model

```

use RNA;

# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCCUAUGUUUGUGACUCGCAGACUAACA";

# create a new model details structure
my $md = new RNA::md();

# change temperature and dangle model
$md->{temperature} = 20.0; # 20 Deg Celcius

```

```
$md->{dangles}      = 1;      # Dangle Model 1

# create a fold compound
my $fc = new RNA::fold_compound($seq, $md);

# predict Minnum Free Energy and corresponding secondary structure
my ($ss, $mfe) = $fc->mfe();

# print sequence, structure and MFE
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;
```

2.3.2 Python

Simple MFE prediction for a given sequence

```
import RNA

# The RNA sequence
seq = "GAGUAGUGGAACCAAGGCUAUGUUUGUGACUCGCAGACUAACA"

# compute minimum free energy (MFE) and corresponding structure
(ss, mfe) = RNA.fold(seq)

# print output
print("{}\n{} [ {:.2f} ]".format(seq, ss, mfe))
```

Simple MFE prediction for a multiple sequence alignment

```
import RNA

# The RNA sequence alignment
sequences = [
    "CUGCCUCACAACGUUUGUGCCUCAGUUACCCGUGAGAUGUAGUGAGGGU",
    "CUGCCUCACAACAUAUUUGUGCCUCAGUUACUUAUGAUGUAGUGAGGGU",
    "---CUCGACACCACU---GCCUCGGUUAACCAUCGGUGCAGUGCGGGU"
]

# compute the consensus sequence
cons = RNA.consensus(sequences)

# predict Minnum Free Energy and corresponding secondary structure
(ss, mfe) = RNA.alifold(sequences);

# print output
print("{}\n{} [ {:.2f} ]".format(cons, ss, mfe))
```

Deviating from the Default Model

```
import RNA

# The RNA sequence
seq = "GAGUAGUGGAACCAAGGCUAUGUUUGUGACUCGCAGACUAACA"

# create a new model details structure
md = RNA.md()

# change temperature and dangle model
md.temperature = 20.0 # 20 Deg Celcius
md.dangles     = 1    # Dangle Model 1

# create a fold compound
fc = RNA.fold_compound(seq, md)

# predict Minnum Free Energy and corresponding secondary structure
(ss, mfe) = fc.mfe()

# print sequence, structure and MFE
print("{}\n{} [ {:.2f} ]".format(seq, ss, mfe))
```


Chapter 3

Concepts and Algorithms

This is an overview of the concepts and algorithms for which implementations can be found in this library.

Almost all of them rely on the physics based Nearest Neighbor Model for RNA secondary structure prediction.

- [RNA Structure](#) gives an introduction into the different layers of abstraction for RNA structures
- [Distance Measures](#) introduces different metrics to allow for the comparison of secondary structures
- [Free Energy of Secondary Structures](#) shows how the stability of a secondary structure can be quantified in terms of free energy
- [Secondary Structure Folding Grammar](#) explains the basic recursive decomposition scheme that is applied in secondary structure prediction
- [RNA Secondary Structure Landscapes](#) describes how transition paths between secondary structures span a landscape like graph
- [Minimum Free Energy Algorithm\(s\)](#) compute the most stable conformation in thermodynamic equilibrium
- [Partition Function and Equilibrium Probability Algorithm\(s\)](#) enable one to apply statistical mechanics to derive equilibrium probabilities of structure features
- [Suboptimal and \(other\) Representative Structures](#) allow for alternative description and enumeration of the structure ensemble
- [RNA-RNA Interaction](#) introduces how to model the interaction between RNA molecules
- [Locally Stable Secondary Structures](#) offer insights into structuredness of long sequences and entire genomes
- [Comparative Structure Prediction](#) augment structure prediction with evolutionary conservation of homologous sequences
- [Classified DP variations](#) perform an *a priori* partitioning of the structure ensemble and compute various properties for the resulting classes.
- [RNA Sequence Design](#) constitutes the inverse problem of structure prediction
- [Experimental Structure Probing Data](#) can be used to guide structure prediction, for instance using SHAPE reactivity data
- [Ligand Binding](#) adds more complexity to structure prediction by modelling the interaction between small chemical compounds or proteins and the RNA
- [\(Tertiary\) Structure Motifs](#) extend the abstraction of secondary structure beyond canonical base pair formation

3.1 RNA Structure

3.1.1 RNA Structures

3.1.2 Levels of Structure Abstraction

3.1.2.1 Primary Structure

3.1.2.2 Secondary Structure

3.1.2.3 Tertiary Structure

3.1.2.4 Quarternary Structure

3.1.2.5 Pseudo-Knots

3.2 Distance Measures

A simple measure of dissimilarity between secondary structures of equal length is the base pair distance, given by the number of pairs present in only one of the two structures being compared. I.e. the number of base pairs that have to be opened or closed to transform one structure into the other. It is therefore particularly useful for comparing structures on the same sequence. It is implemented by

```
int bp_distance(const char *str1,
               const char *str2)
```

Compute the "base pair" distance between two secondary structures s1 and s2.

For other cases a distance measure that allows for gaps is preferable. We can define distances between structures as edit distances between trees or their string representations. In the case of string distances this is the same as "sequence alignment". Given a set of edit operations and edit costs, the edit distance is given by the minimum sum of the costs along an edit path converting one object into the other. Edit distances like these always define a metric. The edit operations used by us are insertion, deletion and replacement of nodes. String editing does not pay attention to the matching of brackets, while in tree editing matching brackets represent a single node of the tree. [Tree](#) editing is therefore usually preferable, although somewhat slower. String edit distances are always smaller or equal to tree edit distances.

The different level of detail in the structure representations defined above naturally leads to different measures of distance. For full structures we use a cost of 1 for deletion or insertion of an unpaired base and 2 for a base pair. Replacing an unpaired base for a pair incurs a cost of 1.

Two cost matrices are provided for coarse grained structures:

```

/* Null, H, B, I, M, S, E */
{ 0, 2, 2, 2, 2, 1, 1}, /* Null replaced */
{ 2, 0, 2, 2, 2, INF, INF}, /* H replaced */
{ 2, 2, 0, 1, 2, INF, INF}, /* B replaced */
{ 2, 2, 1, 0, 2, INF, INF}, /* I replaced */
{ 2, 2, 2, 2, 0, INF, INF}, /* M replaced */
{ 1, INF, INF, INF, INF, 0, INF}, /* S replaced */
{ 1, INF, INF, INF, INF, INF, 0}, /* E replaced */

/* Null, H, B, I, M, S, E */
{ 0, 100, 5, 5, 75, 5, 5}, /* Null replaced */
{ 100, 0, 8, 8, 8, INF, INF}, /* H replaced */
{ 5, 8, 0, 3, 8, INF, INF}, /* B replaced */
{ 5, 8, 3, 0, 8, INF, INF}, /* I replaced */
{ 75, 8, 8, 8, 0, INF, INF}, /* M replaced */
{ 5, INF, INF, INF, INF, 0, INF}, /* S replaced */
{ 5, INF, INF, INF, INF, INF, 0}, /* E replaced */

```

The lower matrix uses the costs given in [28]. All distance functions use the following global variables:

```
int cost_matrix;
```

Specify the cost matrix to be used for distance calculations.

```
int edit_backtrack;
```

Produce an alignment of the two structures being compared by tracing the editing path giving the minimum distance.

```
char *aligned_line[4];
```

Contains the two aligned structures after a call to one of the distance functions with [edit_backtrack](#) set to 1.

See also

[utils.h](#), [dist_vars.h](#) and [stringdist.h](#) for more details

3.2.1 Functions for Tree Edit Distances

```
Tree *make_tree (char *struc)
```

Constructs a [Tree](#) (essentially the postorder list) of the structure 'struc', for use in [tree_edit_distance\(\)](#).

```
float tree_edit_distance (Tree *T1,
                        Tree *T2)
```

Calculates the edit distance of the two trees.

```
void free_tree(Tree *t)
```

Free the memory allocated for [Tree](#) t.

See also

[dist_vars.h](#) and [treedist.h](#) for prototypes and more detailed descriptions

3.2.2 Functions for String Alignment

```
swString *Make_swString (char *string)
```

Convert a structure into a format suitable for [string_edit_distance\(\)](#).

```
float      string_edit_distance (swString *T1,  
                                swString *T2)
```

Calculate the string edit distance of T1 and T2.

See also

[dist_vars.h](#) and [stringdist.h](#) for prototypes and more detailed descriptions

3.2.3 Functions for Comparison of Base Pair Probabilities

For comparison of base pair probability matrices, the matrices are first condensed into probability profiles which are then compared by alignment.

```
float *Make_bp_profile_bppm ( double *bppm,  
                             int length)
```

condense pair probability matrix into a vector containing probabilities for unpaired, upstream paired and downstream paired.

```
float profile_edit_distance ( const float *T1,  
                             const float *T2)
```

Align the 2 probability profiles T1, T2

.

See also

[ProfileDist.h](#) for prototypes and more details of the above functions

3.3 Free Energy of Secondary Structures

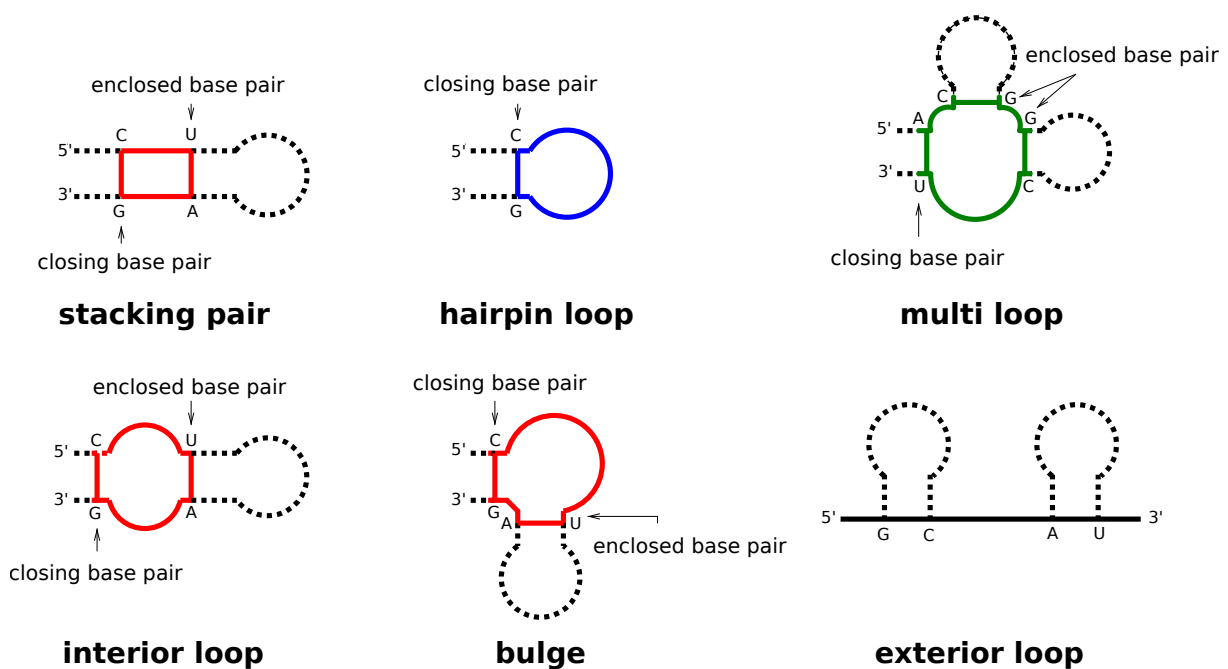
A description on how secondary structures are decomposed into individual loops to eventually evaluate their stability in terms of free energy.

3.3.1 Secondary Structure Loop Decomposition

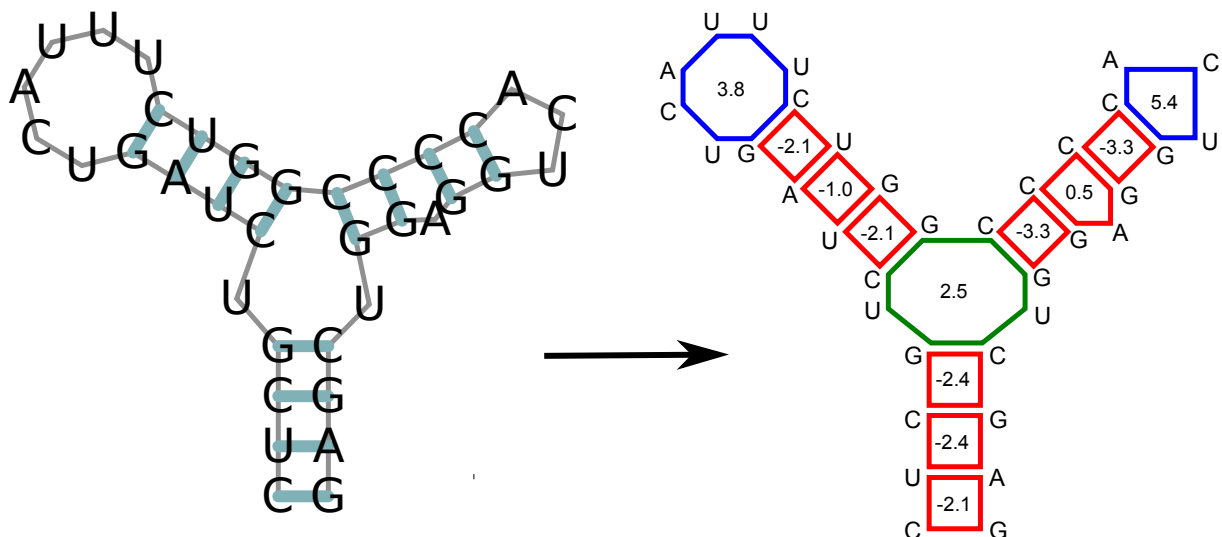
Each base pair in a secondary structure closes a loop, thereby directly enclosing unpaired nucleotides, and/or further base pairs. Our implementation distinguishes four basic types of loops:

- hairpin loops
- interior loops
- multibranch loops
- exterior loop

While the exterior loop is a special case without a closing pair, the other loops are determined by the number of base pairs involved in the loop formation, i.e. hairpin loops are 1-loops, since only a single base pair delimits the loop. interior loops are 2-loops due to their enclosing, and enclosed base pair. All loops where more than two base pairs are involved, are termed multibranch loops.



Any secondary structure can be decomposed into its loops. Each of the loops then can be scored in terms of free energy, and the free energy of an entire secondary structure is simply the sum of free energies of its loops.



3.3.1.1 Free Energy Evaluation API

While we implement some functions that decompose a secondary structure into its individual loops, the majority of methods provided in **RNAlib** are dedicated to free energy evaluation. The corresponding modules are:

See also

[Free Energy Evaluation](#), [Energy Evaluation for Individual Loops](#)

3.3.2 Free Energy Parameters

For secondary structure free energy evaluation we usually utilize the set of Nearest Neighbor Parameters also used in other software, such as *UNAFold* and *RNAstructure*. While the *RNAlib* already contains a compiled-in set of the latest *Turner 2004 Free Energy Parameters*, we defined a file format that allows to change these parameters at runtime. The *ViennaRNA Package* already comes with a set of parameter files containing

- Turner 1999 RNA parameters
- Mathews 1999 DNA parameters
- Andronescu 2007 RNA parameters
- Mathews 2004 DNA parameters

3.3.2.1 Free Energy Parameters Modification API

See also

[Energy Parameters](#), [Reading/Writing Energy Parameter Sets from/to File](#)

3.3.3 Fine-tuning of the Energy Evaluation Model

See also

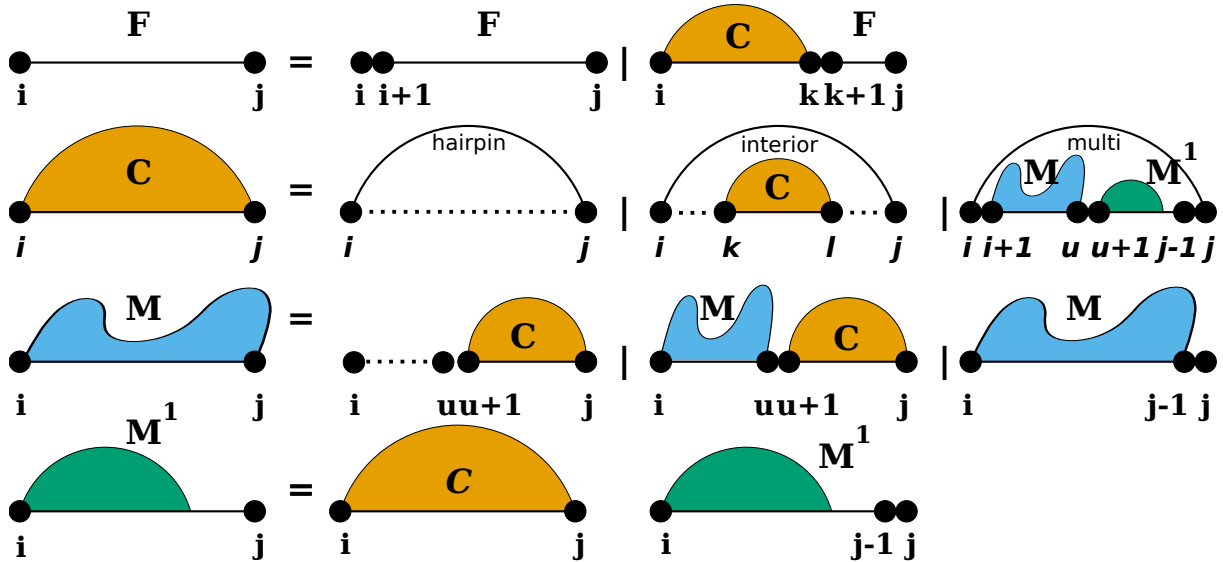
[Fine-tuning of the Implemented Models](#)

3.4 Secondary Structure Folding Grammar

A description of the basic grammar to generate secondary structures, used for almost all prediction algorithms in our library and how to modify it.

3.4.1 Secondary Structure Folding Recurrences

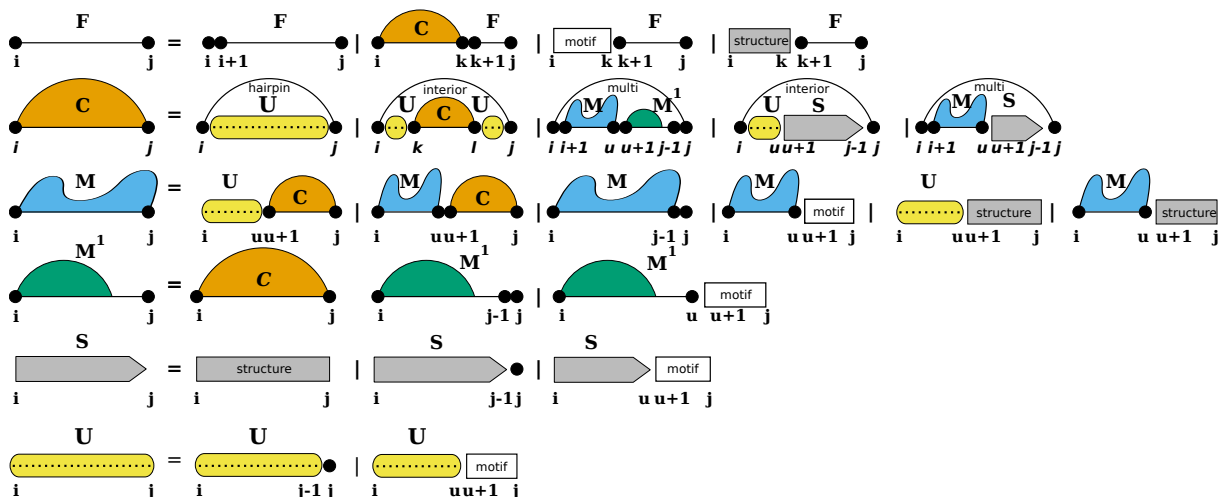
To predict secondary structures composed of the four distinguished loop types introduced before, all algorithms implemented in *RNAlib* follow a specific decomposition scheme, also known as the *RNA folding grammar*, or *Secondary Structure Folding Recurrences*.



However, compared to other RNA secondary structure prediction libraries, our implementation allows for a fine-grained control of the above recursions by constraining both, the individual derivations of the grammar as well as the evaluation of particular loop contributions. Furthermore, we provide a mechanism to extend the above grammar with additional derivation rules, so-called *Domains*.

3.4.2 Additional Structural Domains

Some applications of RNA secondary structure prediction require an extension of the *regular RNA folding grammar*. For instance one would like to include proteins and other ligands binding to unpaired loop regions while competing with conventional base pairing. Another application could be that one may want to include the formation of self-enclosed structural modules, such as *G-quadruplexes*. For such applications, we provide a pair of additional domains that extend the regular RNA folding grammar, [Structured Domains](#) and [Unstructured Domains](#).



While unstructured domains are usually determined by a more or less precise sequence motif, e.g. the binding site for a protein, structured domains are considered self-enclosed modules with a more or less complex pairing pattern. Our extension with these two domains introduces two production rules to fill additional dynamic processing matrices S and U where we store the pre-computed contributions of structured domains (S), and unstructured domains (U).

3.4.2.1 Structured Domains

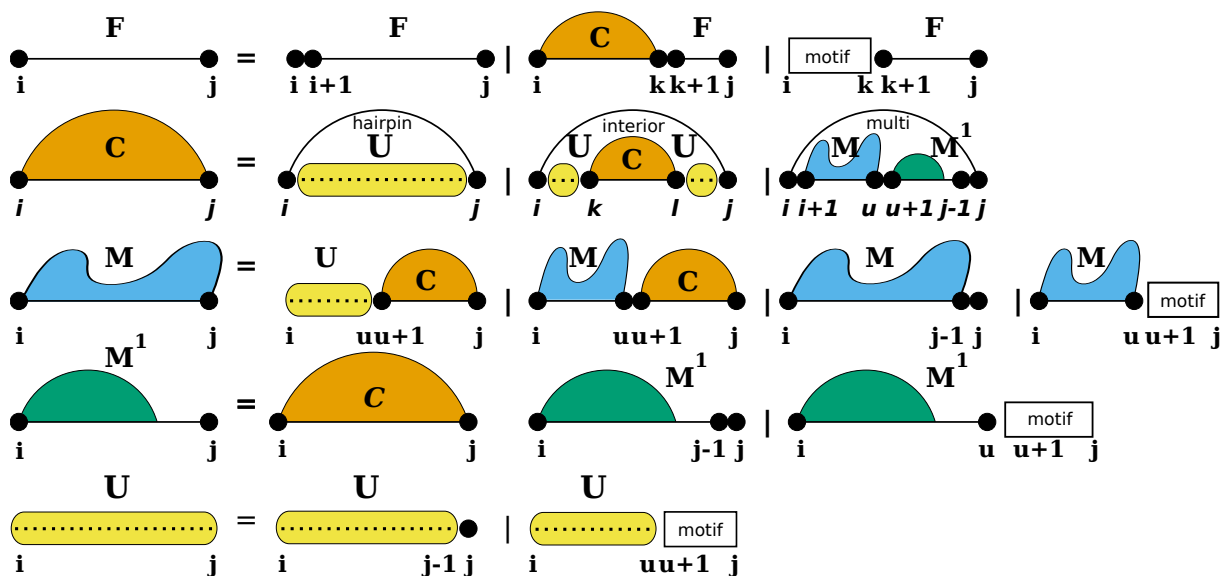
Usually, structured domains represent self-enclosed structural modules that exhibit a more or less complex base pairing pattern. This can be more or less well-defined 3D motifs, such as *G-Quadruplexes*, or loops with additional non-canonical base pair interactions, such as *kink-turns*.

Note

Currently, our implementation only provides the specialized case of *G-Quadruplexes*.

3.4.2.2 Unstructured Domains

Unstructured domains appear in the production rules of the RNA folding grammar wherever new unpaired nucleotides are attached to a growing substructure (see also [21]):



The white boxes represent the stretch of RNA bound to the ligand and represented by a more or less specific sequence motif. The motif itself is considered unable to form base pairs. The additional production rule U is used to precompute the contribution of unpaired stretches possibly bound by one or more ligands. The auxiliary DP matrix for this production rule is filled right before processing the other (regular) production rules of the RNA folding grammar.

3.4.2.3 Domain Extension API

For the sake of flexibility, each of the domains is associated with a specific data structure serving as an abstract interface to the extension. The interface uses callback functions to

- pre-compute arbitrary data, e.g. filling up additional dynamic programming matrices, and
- evaluate the contribution of a paired or unpaired structural feature of the RNA.

Implementations of these callbacks are separate for regular free energy evaluation, e.g. MFE prediction, and partition function applications. A data structure holding arbitrary data required for the callback functions can be associated to the domain as well. While *RNAlib* comes with a default implementation for structured and unstructured domains, the system is entirely user-customizable.

See also

[Unstructured Domains](#), [Structured Domains](#), [G-Quadruplexes](#), [Ligands Binding to Unstructured Domains](#)

3.4.3 Constraints on the Folding Grammar

Secondary Structure constraints can be subdivided into two groups:

- Hard Constraints
- Soft Constraints

While Hard-Constraints directly influence the production rules used in the folding recursions by allowing, disallowing, or enforcing certain decomposition steps, Soft-constraints on the other hand are used to change position specific contributions in the recursions by adding bonuses/penalties in form of pseudo free energies to certain loop configurations.

Note

Secondary structure constraints are always applied at decomposition level, i.e. in each step of the recursive structure decomposition, for instance during MFE prediction.

3.4.3.1 Hard Constraints API

Hard constraints as implemented in our library can be specified for individual loop types, i.e. the atomic derivations of the RNA folding grammar rules. Hence, the pairing behavior of both, single nucleotides and pairs of bases, can be constrained in every loop context separately. Additionally, an abstract implementation using a callback mechanism allows for full control of more complex hard constraints.

See also

[Hard Constraints](#)

3.4.3.2 Soft Constraints API

For the sake of memory efficiency, we do not implement a loop context aware version of soft constraints. The *static* soft constraints as implemented only distinguish unpaired from paired nucleotides. This is usually sufficient for most use-case scenarios. However, similar to hard constraints, an abstract soft constraints implementation using a callback mechanism exists, that allows for any soft constraint that is compatible with the RNA folding grammar. Thus, loop contexts and even individual derivation rules can be addressed separately for maximum flexibility in soft-constraints application.

See also

[Soft Constraints, Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints, SHAPE Reactivity Data](#)

3.5 RNA Secondary Structure Landscapes

A description of the implicit landscape-like network of structures that appears upon modelling the transition of one structure into another.

3.5.1 The Neighborhood of a Secondary Structure

3.5.2 The Secondary Structure Landscape API

3.6 Minimum Free Energy Algorithm(s)

Computing the Minimum Free Energy (MFE), i.e. the most stable conformation in thermodynamic equilibrium.

3.6.1 Zuker's Algorithm

Our library provides fast dynamic programming Minimum Free Energy (MFE) folding algorithms derived from the decomposition scheme as described by "Zuker & Stiegler (1981)" [36].

3.6.2 MFE for circular RNAs

Folding of *circular* RNA sequences is handled as a post-processing step of the forward recursions. See [15] for further details.

3.6.3 MFE Algorithm API

We provide interfaces for the prediction of

- MFE and corresponding secondary structure for single sequences,
- consensus MFE structures of sequence alignments, and
- MFE structure for two hybridized RNA strands

See also

[Minimum Free Energy \(MFE\) Algorithms, RNA-RNA Interaction, Computing MFE representatives of a Distance Based Partition](#)

3.7 Partition Function and Equilibrium Probability Algorithm(s)

3.7.1 Equilibrium Ensemble Statistics

In contrast to methods that compute the property of a single structure in the ensemble, e.g. [Minimum Free Energy Algorithm\(s\)](#), the partition function algorithms always consider the entire equilibrium ensemble. For that purpose, the McCaskill algorithm [23] and its variants can be used to efficiently compute

- the partition function, and from that
- various equilibrium probabilities, for instance base pair probabilities, probabilities of individual structure motifs, and many more.

The principal idea behind this approach is that in equilibrium, statistical mechanics and polymer theory tells us that the frequency or probability $p(s)$ of a particular state s depends on its energy $E(s)$ and follows a Boltzmann distribution, i.e.

$$p(s) \propto e^{-\beta E(s)} \text{ with } \beta = \frac{1}{kT}$$

where $k \approx 1.987 \cdot 10^{-3} \frac{\text{kcal}}{\text{mol K}}$ is the Boltzmann constant, and T the thermodynamic temperature. From that relation, the actual probability of state s can then be obtained using a proper scaling factor, the *canonical partition function*

$$Z = \sum_{s \in \Omega} e^{-\beta E(s)}$$

where Ω is the finite set of all states. Finally, the equilibrium probability of state s can be computed as

$$p(s) = \frac{e^{-\beta E(s)}}{Z}$$

Instead of enumerating all states exhaustively to compute Z one can apply the [Secondary Structure Folding Recurrences](#) again for an efficient computation in cubic time. An *outside* variant of the same recursions is then used to compute probabilities for base pairs, stretches of consecutive unpaired nucleotides, or structural motifs.

See also

Further details of the Partition function and Base Pair Probability algorithm can be obtained from McCaskill 1990 [23]

3.7.2 Partition Function and Equilibrium Probability API

We implement a wide variety of variants of the partition function algorithm according to McCaskill 1990 [23]. See the corresponding submodules for specific implementation details.

See also

[Partition Function and Equilibrium Properties](#), [RNA-RNA Interaction](#), [Partition Function for two Hybridized Sequences as a Step](#), [Computing Partition Functions of a Distance Based Partitioning](#)

3.8 Suboptimal and (other) Representative Structures

3.8.1 Suboptimal Secondary Structures

3.8.2 Sampling Secondary Structures from the Ensemble

3.8.3 Structure Enumeration and Sampling API

See also

[Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989](#), [Suboptimal Structures within an Energy Band around the Minimum Free Energy Structure](#), [Random Structure Samples from the Ensemble](#), [Compute the Structure with Maximum Expected Accuracy \(MEA\)](#), [Compute the Centroid Structure](#)

3.9 RNA-RNA Interaction

3.9.1 `
`

The function of an RNA molecule often depends on its interaction with other RNAs. The following routines therefore allow one to predict structures formed by two RNA molecules upon hybridization.

3.9.2 Concatenating RNA sequences

One approach to co-folding two RNAs consists of concatenating the two sequences and keeping track of the concatenation point in all energy evaluations. Correspondingly, many of the `cofold()` and `co_pf_fold()` routines take one sequence string as argument and use the global variable `cut_point` to mark the concatenation point. Note that while the *RNAcofold* program uses the `'&'` character to mark the chain break in its input, you should not use an `'&'` when using the library routines (set `cut_point` instead).

3.9.3 RNA-RNA interaction as a Stepwise Process

In a second approach to co-folding two RNAs, cofolding is seen as a stepwise process. In the first step the probability of an unpaired region is calculated and in a second step this probability of an unpaired region is multiplied with the probability of an interaction between the two RNAs. This approach is implemented for the interaction between a long target sequence and a short ligand RNA. Function `pf_unstru()` calculates the partition function over all unpaired regions in the input sequence. Function `pf_interact()`, which calculates the partition function over all possible interactions between two sequences, needs both sequence as separate strings as input.

3.9.4 RNA-RNA Interaction API

3.10 Locally Stable Secondary Structures

3.10.1 local_intro

3.10.2 local_mfe

3.10.3 local_pf

3.10.4 Locally Stable Secondary Structure API

3.11 Comparative Structure Prediction

3.11.1 Incorporate Evolutionary Information

Consensus structures can be predicted by a modified version of the [fold\(\)](#) algorithm that takes a set of aligned sequences instead of a single sequence. The energy function consists of the mean energy averaged over the sequences, plus a covariance term that favors pairs with consistent and compensatory mutations and penalizes pairs that cannot be formed by all structures. For details see [\[13\]](#) and [\[1\]](#).

3.11.2 Comparative Structure Prediction API

3.12 Classified DP variations

3.12.1 The Idea of Classified Dynamic Programming

Usually, thermodynamic properties using the basic recursions for [Minimum Free Energy Algorithm\(s\)](#), [Partition Function and Equilibrium](#) and so forth, are computed over the entire structure space. However, sometimes it is desired to partition the structure space *a priori* and compute the above properties for each of the resulting partitions. This approach directly leads to *Classified Dynamic Programming*.

3.12.2 Distance Class Partitioning

The secondary structure space is divided into partitions according to the base pair distance to two given reference structures and all relevant properties are calculated for each of the resulting partitions.

See also

For further details, we refer to Lorenz et al. 2009 [\[20\]](#)

3.12.3 Density of States (DOS)

3.12.4 Classified DP API

3.13 RNA Sequence Design

3.13.1 Generate Sequences that fold into particular Secondary Structures

3.13.2 RNA Sequence Design API

See also

[Inverse Folding \(Design\)](#)

3.14 Experimental Structure Probing Data

3.14.1 Guide the Structure Prediction using Experimental Data

3.14.1.1 SHAPE reactivities

3.14.2 Structure Probing Data API

See also

[Experimental Structure Probing Data](#), [SHAPE Reactivity Data](#), [Generate Soft Constraints from Data](#)

3.15 Ligand Binding

3.15.1 Small Molecules and Proteins that bind to specific RNA Structures

3.15.2 `ligand_binding_api`

In our library, we provide two different ways to incorporate ligand binding to RNA structures:

- [Ligands Binding to Unstructured Domains](#), and
- [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#)

The first approach is implemented as an actual extension of the folding grammar. It adds auxiliary derivation rules for each case when consecutive unpaired nucleotides are evaluated. Therefore, this model is applicable to ligand binding to any loop context.

The second approach, on the other hand, uses the soft-constraints feature to change the energy evaluation of hairpin- or interior-loops. Hence, it can only be applied when a ligand binds to a hairpin-like, or interior-loop like motif.

See also

[Ligands Binding to Unstructured Domains](#), [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#)

3.16 (Tertiary) Structure Motifs

3.16.1 Incorporating Higher-Order (Tertiary) Structure Motifs

3.16.2 RNA G-Quadruplexes

3.16.3 (Tertiary) Structure Motif API

Chapter 4

I/O Formats

Below, you'll find a listing of different sections that introduce the most common notations of sequence and structure data, specifications of bioinformatics sequence and structure file formats, and various output file formats produced by our library.

- [RNA Structure Notations](#) describes the different notations and representations of RNA secondary structures
- [File Formats](#) gives an overview of the file formats compatible with our library
- [Plotting](#) shows the different (PostScript) plotting functions for RNA secondary structures, feature probabilities, and multiple sequence alignments

4.1 RNA Structure Notations

4.1.1 Representations of Secondary Structures

The standard representation of a secondary structure in our library is the [Dot-Bracket Notation](#) (a.k.a. [Dot-Parenthesis Notation](#)), where matching brackets symbolize base pairs and unpaired bases are shown as dots. Based on that notation, more elaborate representations have been developed to include additional information, such as the loop context a nucleotide belongs to and to annotated pseudo-knots.

4.1.1.1 Dot-Bracket Notation (a.k.a. Dot-Parenthesis Notation)

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots . .

As a simple example, consider a helix of size 4 enclosing a hairpin of size 4. In dot-bracket notation, this is annotated as

```
(( (( . . . . ) ) ) )
```

Extended Dot-Bracket Notation

A more generalized version of the original Dot-Bracket notation may use additional pairs of brackets, such as <>, {}, and [], and matching pairs of uppercase/lowercase letters. This allows for annotating pseudo-knots, since different pairs of brackets are not required to be nested.

The following annotations of a simple structure with two crossing helices of size 4 are equivalent:

```
<<<<[ [ [ [ . . . . >>>> ] ] ] ]  
( ( ( ( A A A A . . . . ) ) ) ) a a a a  
A A A A { { { { . . . . a a a a } } } }
```

See also

[vrna_db_pack\(\)](#), [vrna_db_unpack\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#), [vrna_db_from_ptable\(\)](#),
[vrna_db_from_plist\(\)](#), [vrna_db_to_element_string\(\)](#), [vrna_db_pk_remove\(\)](#)

4.1.1.2 Washington University Secondary Structure (WUSS) notation

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

This notation allows for a fine-grained annotation of base pairs and unpaired nucleotides, including pseudo-knots. Below, you'll find a list of secondary structure elements and their corresponding WUSS annotation (See also the infernal user guide at <http://eddylab.org/infernal/Userguide.pdf>)

- **Base pairs**

Nested base pairs are annotated by matching pairs of the symbols `<>`, `()`, `{}`, and `[]`. Each of the matching pairs of parenthesis have their special meaning, however, when used as input in our programs, e.g. structure constraint, these details are usually ignored. Furthermore, base pairs that constitute as pseudo-knot are denoted by letters from the latin alphabet and are, if not denoted otherwise, ignored entirely in our programs.

- **Hairpin loops**

Unpaired nucleotides that constitute the hairpin loop are indicated by underscores, `_`.

Example: `<<<<_____>>>>`

- **Bulges and interior loops**

Residues that constitute a bulge or interior loop are denoted by dashes, `-`.

Example: `(((--<_____>-)))`

- **Multibranch loops**

Unpaired nucleotides in multibranch loops are indicated by commas, `,`.

Example: `(((„<_____>,<_____>))`

- **External residues**

Single stranded nucleotides in the exterior loop, i.e. not enclosed by any other pair are denoted by colons, `:`.

Example: `<<<_____>>>:::`

- **Insertions**

In cases where an alignment represents the consensus with a known structure, insertions relative to the known structure are denoted by periods, `.`. Regions where local structural alignment was invoked, leaving regions of both target and query sequence unaligned, are indicated by tildes, `~`.

Note

These symbols only appear in alignments of a known (query) structure annotation to a target sequence of unknown structure.

- **Pseudo-knots**

The WUSS notation allows for annotation of pseudo-knots using pairs of upper-case/lower-case letters.

Note

Our programs and library functions usually ignore pseudo-knots entirely treating them as unpaired nucleotides, if not stated otherwise.

Example: `<<<_AAA____>>>aaa`

See also

[vrna_db_from_WUSS\(\)](#)

4.1.1.3 Abstract Shapes

Abstract Shapes, introduced by Giegerich et al. in (2004) [12], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

The abstract shapes representation abstracts the structure from individual base pairs and their corresponding location in the sequence, while retaining the inherent nestedness of helices and hairpin loops.

Below is a description of what is included in the abstract shapes abstraction for each respective level together with an example structure:

```
CGUCUUAACUCAUCACCGUGGAGCUGCGACCCUCCUAGAUUCGAAGACGAG
(((((((...(((...))))))...(((...((...))...))))))...))..
```

Shape Level	Description	Result
1	Most accurate - all loops and all unpaired	[_ [_ []] _ [_ []] _ ↔]] _
2	Nesting pattern for all loop types and unpaired regions in external loop and multiloop	[[_ []] [_ []] _]]
3	Nesting pattern for all loop types but no unpaired regions	[[[]] [[]]]
4	Helix nesting pattern in external loop and multiloop	[[[] []]]
5	Most abstract - helix nesting pattern and no unpaired regions	[[[]]]

Note

Our implementations also provide the special Shape Level 0, which does not collapse any structural features but simply convert base pairs and unpaired nucleotides into their corresponding set of symbols for abstract shapes.

See also

[vrna_abstract_shapes\(\)](#), [vrna_abstract_shapes_pt\(\)](#)

4.1.1.4 Tree Representations of Secondary Structures

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

Alternatively, one may find representations with two types of node labels, \mathbb{P} for paired and \mathbb{U} for unpaired; a dot is then replaced by (\mathbb{U}) , and each closed bracket is assigned an additional identifier \mathbb{P} . We call this the expanded notation. In [10] a condensed representation of the secondary structure is proposed, the so-called homeomorphically irreducible tree (HIT) representation. Here a stack is represented as a single pair of matching brackets labeled \mathbb{P} and weighted by the number of base pairs. Correspondingly, a contiguous strain of unpaired bases is shown as one pair of matching brackets labeled \mathbb{U} and weighted by its length. Generally any string consisting of matching brackets and identifiers is equivalent to a plane tree with as many different types of nodes as there are identifiers.

Bruce Shapiro proposed a coarse grained representation [27], which, does not retain the full information of the secondary structure. He represents the different structure elements by single matching brackets and labels them as

- \mathbb{H} (hairpin loop),
- \mathbb{I} (interior loop),
- \mathbb{B} (bulge),
- \mathbb{M} (multi-loop), and
- \mathbb{S} (stack).

We extend his alphabet by an extra letter for external elements **E**. Again these identifiers may be followed by a weight corresponding to the number of unpaired bases or base pairs in the structure element. All tree representations (except for the dot-bracket form) can be encapsulated into a virtual root (labeled **R**).

The following example illustrates the different linear tree representations used by the package:

Consider the secondary structure represented by the dot-bracket string (full tree) `. ((. . ((. . .))) . . ((. .)))) .` which is the most convenient condensed notation used by our programs and library functions.

Then, the following tree representations are equivalent:

- Expanded tree:
`((U)((U)(U)((U)(U)(U)P)P)P)(U)(U)((U)(U)P)P)P)(U)R)`
- HIT representation (Fontana et al. 1993 [10]):
`((U1)((U2)((U3)P3)(U2)((U2)P2)P2)(U1)R)`
- Coarse Grained **Tree** Representation (Shapiro 1988 [27]):
 - Short (with root node R, without stem nodes S):
`((H)((H)M)R)`
 - Full (with root node R):
`((((H)S)((H)S)M)S)R)`
 - Extended (with root node R, with external nodes E):
`(((((H)S)((H)S)M)S)E)R)`
 - Weighted (with root node R, with external nodes E):
`(((((H3)S3)((H2)S2)M4)S2)E2)R)`

The Expanded tree is rather clumsy and mostly included for the sake of completeness. The different versions of Coarse Grained **Tree** Representations are variations of Shapiro's linear tree notation.

For the output of aligned structures from string editing, different representations are needed, where we put the label on both sides. The above examples for tree representations would then look like:

```
* a) (UU) (P (P (P (P (UU) (UU) (P (P (UU) (UU) (UU) P) P) P) (UU) (UU) (P (P (UU) (U. . .
* b) (UU) (P2 (P2 (U2U2) (P2 (U3U3) P3) (U2U2) (P2 (U2U2) P2) P2) (UU) P2) (UU)
* c) (B (M (HH) (HH) M) B)
*   (S (B (S (M (S (HH) S) (S (HH) S) M) S) B) S)
*   (E (S (B (S (M (S (HH) S) (S (HH) S) M) S) B) S) E)
* d) (R (E2 (S2 (B1 (S2 (M4 (S3 (H3) S3) ((H2) S2) M4) S2) B1) S2) E2) R)
*
```

Aligned structures additionally contain the gap character `_`.

See also

[vrna_db_to_tree_string\(\)](#), [vrna_tree_string_unweight\(\)](#), [vrna_tree_string_to_db\(\)](#)

4.1.2 Examples for Structure Parsing and Conversion

4.1.3 Structure Parsing and Conversion API

Several functions are provided for parsing structures and converting to different representations.

```
char *expand_Full(const char *structure)
```

Convert the full structure from bracket notation to the expanded notation including root.

```
char *b2HIT (const char *structure)
```

Converts the full structure from bracket notation to the HIT notation including root.

```
char *b2C (const char *structure)
```

Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.

```
char *b2Shapiro (const char *structure)
```

Converts the full structure from bracket notation to the *weighted* coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.

```
char *expand_Shapiro (const char *coarse);
```

Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from [b2C\(\)](#).

```
char *add_root (const char *structure)
```

Adds a root to an un-rooted tree in any except bracket notation.

```
char *unexpand_Full (const char *ffull)
```

Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.

```
char *unweight (const char *wcoarse)
```

Strip weights from any weighted tree.

```
void unexpand_aligned_F (char *align[2])
```

Converts two aligned structures in expanded notation.

```
void parse_structure (const char *structure)
```

Collects a statistic of structure elements of the full structure in bracket notation.

See also

[RNAstruct.h](#) for prototypes and more detailed description

4.2 File Formats

4.2.1 File formats for Multiple Sequence Alignments (MSA)

4.2.1.1 ClustalW format

The *ClustalW* format is a relatively simple text file containing a single multiple sequence alignment of DNA, RNA, or protein sequences. It was first used as an output format for the *clustalw* programs, but nowadays it may also be generated by various other sequence alignment tools. The specification is straight forward:

- The first line starts with the words

```
CLUSTAL W
```

or

```
CLUSTALW
```

- After the above header there is at least one empty line
- Finally, one or more blocks of sequence data are following, where each block is separated by at least one empty line

Each line in a blocks of sequence data consists of the sequence name followed by the sequence symbols, separated by at least one whitespace character. Usually, the length of a sequence in one block does not exceed 60 symbols. Optionally, an additional whitespace separated cumulative residue count may follow the sequence symbols. Optionally, a block may be followed by a line depicting the degree of conservation of the respective alignment columns.

Note

Sequence names and the sequences must not contain whitespace characters! Allowed gap symbols are the hyphen ("-"), and dot (".").

Note

Sequence names must not contain whitespace characters. Otherwise, the parts after the first whitespace will be dropped. The only allowed gap character is the hyphen ("-").

Here is an example alignment in FASTA format:

```
>AL031296.1/85969-86120
CUGCCUCACAACGUUUGGCCUCAGUUACCCGUAGAUGUAGUGAGGGUAACAAUACUUAC
UCUCGUUGGUGUAAGGAACAGCU
>AANU01225121.1/438-603
CUGCCUCACAACAUUUGGCCUCAGUUACUCAUAGAUGUAGUGAGGGUGACAAUACUUAC
UCUCGUUGGUGUAAGGAACAGCU
>AAWR02037329.1/29294-29150
---CUCGACACCACU---GCCUCGGUUAACCAUCGGUGCAGUGCGGGUAGUAGUACCAAU
GCUAAUAGUUGAGGACCAACU
```

4.2.1.4 MAF format

The multiple alignment format (MAF) is usually used to store multiple alignments on DNA level between entire genomes. It consists of independent blocks of aligned sequences which are annotated by their genomic location. Consequently, an MAF formatted MSA file may contain multiple records. MAF files start with a line

```
##maf
```

which is optionally extended by whitespace delimited key=value pairs. Lines starting with the character("#") are considered comments and usually ignored.

A MAF block starts with character("a") at the beginning of a line, optionally followed by whitespace delimited key=value pairs. The next lines start with character("s") and contain sequence information of the form

```
s src start size strand srcSize sequence
```

where

- *src* is the name of the sequence source
- *start* is the start of the aligned region within the source (0-based)
- *size* is the length of the aligned region without gap characters
- *strand* is either "+" or "-", depicting the location of the aligned region relative to the source
- *srcSize* is the size of the entire sequence source, e.g. the full chromosome
- *sequence* is the aligned sequence including gaps depicted by the hyphen ("-")

Here is an example alignment in MAF format (bluntly taken from the [UCSC Genome browser website](#)):

```
##maf version=1 scoring=tba.v8
# tba.v8 ((human chimp) baboon) (mouse rat))
# multiz.v7
# maf_project.v5 _tba_right.maf3 mouse _tba_C
# single_cov2.v4 single_cov2 /dev/stdin

a score=23262.0
s hg16.chr7 27578828 38 + 158545518 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s panTro1.chr6 28741140 38 + 161576975 AAA-GGGAATGTTAACCAAATGA---ATTGTCTCTTACGGTG
s baboon 116834 38 + 4622798 AAA-GGGAATGTTAACCAAATGA---GTTGTCTCTTATGGTG
s mm4.chr6 53215344 38 + 151104725 -AATGGGAATGTTAAGCAAACGA---ATTGTCTCTCAGTGTG
s rn3.chr4 81344243 40 + 187371129 -AA-GGGGATGCTAAGCCAATGAGTTGTTGTCTCTCAATGTG

a score=5062.0
s hg16.chr7 27699739 6 + 158545518 TAAAGA
s panTro1.chr6 28862317 6 + 161576975 TAAAGA
s baboon 241163 6 + 4622798 TAAAGA
s mm4.chr6 53303881 6 + 151104725 TAAAGA
s rn3.chr4 81444246 6 + 187371129 taagga

a score=6636.0
s hg16.chr7 27707221 13 + 158545518 gcagctgaaaaca
s panTro1.chr6 28869787 13 + 161576975 gcagctgaaaaca
s baboon 249182 13 + 4622798 gcagctgaaaaca
s mm4.chr6 53310102 13 + 151104725 ACAGCTGAAAATA
```

4.2.2 File formats to manipulate the RNA folding grammar

4.2.2.1 Command Files

The RNAlib and many programs of the ViennaRNA Package can parse and apply data from so-called command files. These commands may refer to structure constraints or even extensions of the RNA folding grammar (such as [Unstructured Domains](#)). Commands are given as a line of whitespace delimited data fields. The syntax we use extends the constraint definitions used in the `mfold` / `UNAFold` software, where each line begins with a command character followed by a set of positions.

However, we introduce several new commands, and allow for an optional loop type context specifier in form of a sequence of characters, and an orientation flag that enables one to force a nucleotide to pair upstream, or downstream.

4.2.2.1.1 Constraint commands The following set of commands is recognized:

- `F ...` Force
- `P ...` Prohibit
- `C ...` Conflicts/Context dependency
- `A ...` Allow (for non-canonical pairs)
- `E ...` Soft constraints for unpaired position(s), or base pair(s)

4.2.2.1.2 RNA folding grammar extensions

- `UD ...` Add ligand binding using the [Unstructured Domains](#) feature

4.2.2.1.3 Specification of the loop type context The optional loop type context specifier [LOOP] may be a combination of the following:

- `E ...` Exterior loop
- `H ...` Hairpin loop
- `I ...` Interior loop
- `M ...` Multibranch loop
- `A ...` All loops

For structure constraints, we additionally allow one to address base pairs enclosed by a particular kind of loop, which results in the specifier [WHERE] which consists of [LOOP] plus the following character:

- `i ...` enclosed pair of an Interior loop
- `m ...` enclosed pair of a Multibranch loop

If no [LOOP] or [WHERE] flags are set, all contexts are considered (equivalent to `A`)

4.2.2.1.4 Controlling the orientation of base pairing For particular nucleotides that are forced to pair, the following [ORIENTATION] flags may be used:

- `U ...` Upstream
- `D ...` Downstream

If no [ORIENTATION] flag is set, both directions are considered.

4.2.2.1.5 Sequence coordinates Sequence positions of nucleotides/base pairs are 1— based and consist of three positions i , j , and k . Alternatively, four positions may be provided as a pair of two position ranges $[i : j]$, and $[k : l]$ using the '-' sign as delimiter within each range, i.e. $i - j$, and $k - l$.

4.2.2.1.6 Valid constraint commands Below are resulting general cases that are considered *valid* constraints:

1. **"Forcing a range of nucleotide positions to be paired":**

Syntax:

```
F i 0 k [WHERE] [ORIENTATION]
```

Description:

Enforces the set of k consecutive nucleotides starting at position i to be paired. The optional loop type specifier [WHERE] allows to force them to appear as closing/enclosed pairs of certain types of loops.

2. **"Forcing a set of consecutive base pairs to form":**

Syntax:

```
F i j k [WHERE]
```

Description:

Enforces the base pairs $(i, j), \dots, (i + (k - 1), j - (k - 1))$ to form. The optional loop type specifier [WHERE] allows to specify in which loop context the base pair must appear.

3. **"Prohibiting a range of nucleotide positions to be paired":**

Syntax:

```
P i 0 k [WHERE]
```

Description:

Prohibit a set of k consecutive nucleotides to participate in base pairing, i.e. make these positions unpaired. The optional loop type specifier [WHERE] allows to force the nucleotides to appear within the loop of specific types.

4. **"Prohibiting a set of consecutive base pairs to form":**

Syntax:

```
P i j k [WHERE]
```

Description:

Prohibit the base pairs $(i, j), \dots, (i + (k - 1), j - (k - 1))$ to form. The optional loop type specifier [WHERE] allows to specify the type of loop they are disallowed to be the closing or an enclosed pair of.

5. **"Prohibiting two ranges of nucleotides to pair with each other":**

Syntax:

```
P i-j k-l [WHERE]
```

Description:

Prohibit any nucleotide $p \in [i : j]$ to pair with any other nucleotide $q \in [k : l]$. The optional loop type specifier [WHERE] allows to specify the type of loop they are disallowed to be the closing or an enclosed pair of.

6. **"Enforce a loop context for a range of nucleotide positions":**

Syntax:

```
C i 0 k [WHERE]
```

Description:

This command enforces nucleotides to be unpaired similar to *prohibiting* nucleotides to be paired, as described above. It too marks the corresponding nucleotides to be unpaired, however, the [WHERE] flag can be used to enforce specific loop types the nucleotides must appear in.

7. **"Remove pairs that conflict with a set of consecutive base pairs":**

Syntax:

```
C i j k
```

Description:

Remove all base pairs that conflict with a set of consecutive base pairs $(i, j), \dots, (i + (k - 1), j - (k - 1))$. Two base pairs (i, j) and (p, q) conflict with each other if $i < p < j < q$, or $p < i < q < j$.

8. "Allow a set of consecutive (non-canonical) base pairs to form":

Syntax:

`A i j k [WHERE]`

Description:

This command enables the formation of the consecutive base pairs $(i, j), \dots, (i + (k - 1), j - (k - 1))$, no matter if they are *canonical*, or *non-canonical*. In contrast to the above `F` and `W` commands, which remove conflicting base pairs, the `A` command does not. Therefore, it may be used to allow *non-canonical* base pair interactions. Since the RNAlib does not contain free energy contributions E_{ij} for non-canonical base pairs (i, j) , they are scored as the *maximum* of similar, known contributions. In terms of a *Nussinov* like scoring function the free energy of non-canonical base pairs is therefore estimated as

$$E_{ij} = \min \left[\max_{(i,k) \in \{GC,CG,AU,UA,GU,UG\}} E_{ik}, \max_{(k,j) \in \{GC,CG,AU,UA,GU,UG\}} E_{kj} \right].$$

The optional loop type specifier `[WHERE]` allows to specify in which loop context the base pair may appear.

9. "Apply pseudo free energy to a range of unpaired nucleotide positions":

Syntax:

`E i 0 k e`

Description:

Use this command to apply a pseudo free energy of e to the set of k consecutive nucleotides, starting at position i . The pseudo free energy is applied only if these nucleotides are considered unpaired in the recursions, or evaluations, and is expected to be given in *kcal/mol*.

10. "Apply pseudo free energy to a set of consecutive base pairs":

Syntax

`E i j k e`

Use this command to apply a pseudo free energy of e to the set of base pairs $(i, j), \dots, (i + (k - 1), j - (k - 1))$. Energies are expected to be given in *kcal/mol*.

4.2.2.1.7 Valid domain extensions commands

1. "Add ligand binding to unpaired motif (a.k.a. unstructured domains)":

Syntax:

`UD m e [LOOP]`

Description:

Add ligand binding to unpaired sequence motif m (given in IUPAC format, capital letters) with binding energy e in particular loop type(s).

Example:

`UD AAA -5.0 A`

The above example applies a binding free energy of -5 kcal/mol for a motif AAA that may be present in all loop types.

4.2.3 File Formats for Energy Parameters

4.2.3.1 JSON Parameter Files for Modified Bases

The functions `vrna_sc_mod()`, `vrna_sc_mod_json()` and alike implement an energy correction framework to account for modified bases in the secondary structure predictions. To supply these functions with the energy parameters and general specifications of the base modification, the following JSON data format may be used:

JSON data must consist of a header section **modified_bases**. This header is an object with the mandatory keys:

- **name** specifying a name of the modified base
- **unmodified** that consists of a single upper-case letter of the unmodified version of this base,
- the **one_letter_code** key to specify which letter is used for the modified bases in the subsequent energy parameters, and
- an array of **pairing_partners**.

The latter must be uppercase characters. An optional **sources** key may contain an array of related publications, e.g. those the parameters have been derived from.

Next to the header may follow additional keys to specify the actual energy contributions of the modified base in various loop contexts. All energy contributions must be specified in free energies ΔG in units of $kcal \cdot mol^{-1}$. To allow for rescaling of the free energies at temperatures that differ from the default ($37^{\circ}C$), enthalpy parameters ΔH may be specified as well. Those, however are optional. The keys for free energy (at $37^{\circ}C$) and enthalpy parameters have the suffixes **_energies** and **_enthalpies**, respectively.

The parser and underlying framework currently supports the following loop contexts:

- base pair stacks (via the **stacking** key prefix).
This key must point to an object with one key value pair for each stacking interaction data is provided for. Here, the key consists of four upper-case characters denoting the interacting bases, where the the first two represent one strand in 5' to 3' direction and the last two the opposite strand in 3' to 5' direction. The values are energies in $kcal \cdot mol^{-1}$.
- terminal mismatches (via the **mismatch** key prefix)
This key points to an object with key value pairs for each mismatch energy parameter that is available. Keys are 4 characters long nucleotide one-letter codes as used in base pair stacks above. The second and fourth character denote the two unpaired mismatching bases, while the other two represent the closing base pair.
- dangling ends (via the **dangle5** and **dangle3** key prefixes)
The object behind these keys, again, consists of key value pairs for each dangling end energy parameter. Keys are 3 characters long where the first two represent the two nucleotides that form the base pair, and the third is the unpaired base that either stacks on the 3' or 5' end of the enclosed part of the base pair.
- terminal pairs (via the **terminal** key prefix)
Terminal base pairs, such as AU or GU, sometimes receive an additional energy penalty. The object behind this key may list energy parameters to apply whenever particular base pairs occur at the end of a helix. Each of those parameters is specified as key value pair, where the key consists of two upper-case characters denoting the terminal base pair.

Below is a JSON template specifying most of the possible input parameters. Actual energy parameter files can be found in the source code tarball within the **misc/** subdirectory.

```
{
  "modified_base" : {
    "name" : "My modification (M)",
    "sources" : [
      {
        "authors" : "Author 1, Author 2",
        "title" : "UV-melting of modified oligos",
        "journal" : "Some journal",
        "year" : 2022,
        "doi" : "10.0000/000000"
      }
    ],
    "unmodified" : "G",
    "pairing_partners" : [
      "U", "A"
    ],
    "one_letter_code" : "M"
  },
  "stacking_energies" : {
    "MAUU" : -1.2,
    "AGMC" : -2.73
  },
  "stacking_enthalpies" : {
    "MAUU" : -11.1,
    "AGMC" : -9.73
  },
  "terminal_energies" : {
    "MU" : 0.5,
    "UM" : 0.5
  },
}
```

```

"terminal_enthalpies" : {
  "MU" : 2.0,
  "UM" : 2.0
},
"mismatch_energies" : {
  "CMGM" : -1.11,
  "AGUM" : -0.73
},
"mismatch_enthalpies" : {
  "CMGM" : -11.11,
  "AGUM" : -7.73
},
"dangle5_energies" : {
  "UAM" : -1.01
},
"dangle5_enthalpies" : {
  "UAM" : -6.01
},
"dangle3_energies" : {
  "CGM" : -2.1,
  "GCM" : -1.3
}
}

```

See also

`misc/rna_mod_template_parameters.json` in the source code tarball

An actual example of real-world data may look like

```

{
  "modified_base" : {
    "name" : "Pseudouridine",
    "sources" : [
      {
        "authors": "Graham A. Hudson, Richard J. Bloomingdale, and Brent M. Znosko",
        "title" : "Thermodynamic contribution and nearest-neighbor parameters of pseudouridine-adenosine
        base pairs in oligoribonucleotides",
        "journal" : "RNA 19:1474-1482",
        "year" : 2013,
        "doi" : "10.1261/rna.039610.113"
      }
    ],
    "unmodified" : "U",
    "pairing_partners" : [
      "A"
    ],
    "one_letter_code" : "P"
  },
  "stacking_energies" : {
    "APUA" : -2.8,
    "CPGA" : -2.77,
    "GPCA" : -3.29,
    "UPAA" : -1.62,
    "PAAU" : -2.10,
    "PCAG" : -2.49,
    "PGAC" : -2.2,
    "PUAA" : -2.74
  },
  "stacking_enthalpies" : {
    "APUA" : -22.08,
    "CPGA" : -16.23,
    "GPCA" : -24.07,
    "UPAA" : -20.81,
    "PAAU" : -12.47,
    "PCAG" : -17.29,
    "PGAC" : -11.19,
    "PUAA" : -26.94
  },
  "terminal_energies" : {
    "PA" : 0.31,
    "AP" : 0.31
  },
  "terminal_enthalpies" : {
    "PA" : -2.04,
    "AP" : -2.04
  }
}

```

See also

`misc/rna_mod_pseudouridine_parameters.json` in the source code tarball

4.3 Plotting

Create Plots of Secondary Structures, Feature Motifs, and Sequence Alignments

4.3.1 Producing secondary structure graphs

```
int PS_rna_plot ( char *string,
                  char *structure,
                  char *file)
```

Produce a secondary structure graph in PostScript and write it to 'filename'.

```
int PS_rna_plot_a (
    char *string,
    char *structure,
    char *file,
    char *pre,
    char *post)
```

Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.

```
int gmlRNA (char *string,
            char *structure,
            char *ssfile,
            char option)
```

Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.

```
int ssv_rna_plot (char *string,
                  char *structure,
                  char *ssfile)
```

Produce a secondary structure graph in SStructView format.

```
int svg_rna_plot (char *string,
                  char *structure,
                  char *ssfile)
```

Produce a secondary structure plot in SVG format and write it to a file.

```
int xrna_plot ( char *string,
                 char *structure,
                 char *ssfile)
```

Produce a secondary structure plot for further editing in XRNA.

```
int rna_plot_type
```

Switch for changing the secondary structure layout algorithm.

Two low-level functions provide direct access to the graph layouting algorithms:

```
int simple_xy_coordinates ( short *pair_table,
                            float *X,
                            float *Y)
```

Calculate nucleotide coordinates for secondary structure plot the *Simple way*

```
int naview_xy_coordinates ( short *pair_table,
                            float *X,
                            float *Y)
```

See also

[PS_dot.h](#) and [naview.h](#) for more detailed descriptions.

4.3.2 Producing (colored) dot plots for base pair probabilities

```
int PS_color_dot_plot ( char *string,
                       cpair *pi,
                       char *filename)

int PS_color_dot_plot_turn (char *seq,
                           cpair *pi,
                           char *filename,
                           int winSize)

int PS_dot_plot_list (char *seq,
                     char *filename,
                     plist *pl,
                     plist *mf,
                     char *comment)
```

Produce a postscript dot-plot from two pair lists.

```
int PS_dot_plot_turn (char *seq,
                     struct plist *pl,
                     char *filename,
                     int winSize)
```

See also

[PS_dot.h](#) for more detailed descriptions.

4.3.3 Producing (colored) alignments

```
int PS_color_aln (
    const char *structure,
    const char *filename,
    const char *seqs[],
    const char *names[])
```

Produce PostScript sequence alignment color-annotated by consensus structure.

Chapter 5

Basic Data Structures

- [Sequence and Structure Data](#) shows the most common types for sequence or structure data
- [The 'Fold Compound'](#) is the basic, central container for our implementations of prediction-, evaluation, and other algorithms
- [Model Details](#) provides the means to store the different model parameters

5.1 Sequence and Structure Data

See also

[Secondary Structure Utilities](#)

5.2 The 'Fold Compound'

See also

[The Fold Compound](#)

5.3 Model Details

See also

[Fine-tuning of the Implemented Models](#)

Chapter 6

API Features

- [RNAlib API v3.0](#)
- [Callback Functions](#)
- [Scripting Language interface\(s\)](#)

6.1 RNAlib API v3.0

6.1.1 Introduction

With version 2.2 we introduce the new API that will take over the old one in the future version 3.0. By then, backwards compatibility will be broken, and third party applications using RNAlib need to be ported. This switch of API became necessary, since many new features found their way into the RNAlib where a balance between threadsafety and easy-to-use library functions is hard or even impossible to establish. Furthermore, many old functions of the library are present as slightly modified copies of themselves to provide a crude way to overload functions.

Therefore, we introduce the new v3.0 API very early in our development stage such that developers have enough time to migrate to the new functions and interfaces. We also started to provide encapsulation of the RNAlib functions, data structures, typedefs, and macros by prefixing them with *vrna_* and *VRNA_*, respectively. Header files should also be included using the *ViennaRNA/* namespace, e.g.

```
#include <ViennaRNA/fold.h>
```

instead of just using

```
#include <fold.h>
```

as required for RNAlib 1.x and 2.x.

This eases the work for programmers of third party applications that would otherwise need to put much effort into renaming functions and data types in their own implementations if their names appear in our library. Since we still provide backward compatibility up to the last version of RNAlib 2.x, this advantage may be fully exploited only starting from v3.0 which will be released in the future. However, our plan is to provide the possibility for an early switch-off mechanism of the backward compatibility in one of our next releases of ViennaRNA Package 2.x.

6.1.2 What are the major changes?

...

6.1.3 How to port your program to the new API

...

6.1.4 Some Examples using RNAlib API v3.0

Examples on how to use the new v3.0 API can be found in the [First Steps with the Fold Compound](#) section.

6.2 Callback Functions

With the new [RNAlib API v3.0](#) we introduce so-called callback mechanisms for several functions.

6.2.1 The purpose of Callback mechanisms

Using callback mechanisms, our library enables users not only to retrieve computed data without the need for parsing complicated data structures, but also allows one to tweak our implementation to do additional tasks without the requirement of a re-implementation of basic algorithms.

Our implementation of the callback mechanisms always follows the same scheme: The user:

- defines a function that complies with the interface we've defined, and
- passes a pointer to said function to our implementations

In addition to the specific arguments of our callback interfaces, virtually all callbacks receive an additional *pass-through-pointer* as their last argument. This enables one to:

- encapsulate data, and
- provide thread-safe operations,

since this pointer is simply passed through by our library functions. It may therefore hold the address of an arbitrary, user-defined data structure.

6.2.2 List of available Callbacks

Below, you find an enumeration of the individual callback functions that are available in *RNAlib*.

Global [vrna_auxdata_free_f](#) (void *data)

This callback is supposed to free memory occupied by an auxiliary data structure. It will be called when the [vrna_fold_compound_t](#) is erased from memory through a call to [vrna_fold_compound_free\(\)](#) and will be passed the address of memory previously bound to the [vrna_fold_compound_t](#) via [vrna_fold_compound_add_auxdata\(\)](#).

Global [vrna_bs_result_f](#) (const char *structure, void *data)

This function will be called for each secondary structure that has been successfully backtraced from the partition function DP matrices.

Global [vrna_hc_eval_f](#) (int i, int j, int k, int l, unsigned char d, void *data)

This callback enables one to over-rule default hard constraints in secondary structure decompositions.

Global [vrna_heat_capacity_f](#) (float temp, float heat_capacity, void *data)

This function will be called for each evaluated temperature in the heat capacity prediction.

Global [vrna_mfe_window_f](#) (int start, int end, const char *structure, float en, void *data)

This function will be called for each hit in a sliding window MFE prediction.

Global [vrna_probs_window_f](#) (FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int type, void *data)

This function will be called for each probability data set in the sliding window probability computation implementation of [vrna_probs_window\(\)](#). The argument *type* specifies the type of probability that is passed to this function.

Global [vrna_recursion_status_f](#) (unsigned char status, void *data)

This function will be called to notify a third-party implementation about the status of a currently ongoing recursion. The purpose of this callback mechanism is to provide users with a simple way to ensure pre- and post conditions for auxiliary mechanisms attached to our implementations.

Global [vrna_sc_bt_f](#) (int i, int j, int k, int l, unsigned char d, void *data)

This callback enables one to add auxiliary base pairs in the backtracking steps of hairpin- and interior loops.

Global [vrna_sc_exp_f](#) (int i, int j, int k, int l, unsigned char d, void *data)

This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure (Partition function variant, i.e. contributions must be returned as Boltzmann factors).

Global `vrna_sc_f` `(int i, int j, int k, int l, unsigned char d, void *data)`

This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure.

Global `vrna_subopt_result_f` `(const char *structure, float energy, void *data)`

This function will be called for each suboptimal secondary structure that is successfully backtraced.

Global `vrna_ud_add_probs_f` `(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, FLT_OR_DBL exp_energy, void *data)`

A callback function to store equilibrium probabilities for the unstructured domain feature

Global `vrna_ud_exp_f` `(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`

This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand (Partition function variant, i.e. the Boltzmann factors instead of actual free energies).

Global `vrna_ud_exp_production_f` `(vrna_fold_compound_t *vc, void *data)`

The production rule for the unstructured domain grammar extension (Partition function variant)

Global `vrna_ud_f` `(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`

This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand.

Global `vrna_ud_get_probs_f` `(vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, int motif, void *data)`

A callback function to retrieve equilibrium probabilities for the unstructured domain feature

Global `vrna_ud_production_f` `(vrna_fold_compound_t *vc, void *data)`

The production rule for the unstructured domain grammar extension

6.3 Scripting Language interface(s)

6.3.1 Introduction

For an easy integration into scripting languages, we provide an automatically generated interface to the RNAlib C-library, generated with SWIG.

6.3.2 Function Renaming

To provide a namespace-like separation of function symbols from our C library and third-party code, we use the prefix `vrna_` or `VRNA_` whenever possible. This, however, is not necessary for the scripting language interface, as it uses the separate namespace or package `RNA` anyway. Consequently, symbols that appear to have the `vrna_` or `VRNA_` prefix in the C-library have the corresponding prefix stripped away.

For instance, the C code

```
mfe = vrna_fold(sequence, structure);
```

translates to

```
my ($structure, $mfe) = RNA::fold($sequence)
```

in the Perl 5 interface, and

```
structure, mfe = RNA.fold(sequence)
```

for Python. Note, that in this example we also make use of the possibility to return multiple data at once in the scripting language, while the C library function uses additional parameters to return multiple data.

Functions that are dedicated to work on specific data structures only, e.g. the `vrna_fold_compound_t`, are usually not exported at all. Instead, they are attached as object methods of a corresponding class (see [Object oriented Interface for Data Structures](#) for detailed information).

6.3.2.1 Global Variables

For the Python interface(s) SWIG places global variables of the C-library into an additional namespace `cvar`. For instance, changing the global temperature variable thus becomes

```
RNA.cvar.temperature = 25
```

6.3.3 Object oriented Interface for Data Structures

For data structures, typedefs, and enumerations the `vrna_` prefixes are dropped as well, together with their suffixes `_s`, `_t`, and `_e`, respectively. Furthermore, data structures are usually transformed into classes and relevant functions of the C-library are attached as methods.

6.3.4 Examples

Examples on the basic usage of the scripting language interfaces can be found in the [Perl5 Examples](#) and [Python Examples](#) section.

6.3.5 SWIG generated Wrapper notes

Special notes on how functions, structures, enums, and macro definitions are actually wrapped, can be found below

Global [vrna_abstract_shapes](#) (const char *structure, unsigned int level)

This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

Global [vrna_abstract_shapes_pt](#) (const short *pt, unsigned int level)

This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

Global [vrna_aln_conservation_col](#) (const char **alignment, const vrna_md_t *md_p, unsigned int options)

This function is available in an overloaded form where the last two parameters may be omitted, indicating `md = NULL`, and `options = VRNA_MEASURE_SHANNON_ENTROPY`, respectively.

Global [vrna_aln_conservation_struct](#) (const char **alignment, const char *structure, const vrna_md_t *md)

This function is available in an overloaded form where the last parameter may be omitted, indicating `md = NULL`

Global [vrna_backtrack5](#) (vrna_fold_compound_t *fc, unsigned int length, char *structure)

This function is attached as overloaded method `backtrack()` to objects of type `fold_compound` with default parameter `length` equal to the total length of the RNA.

Global [vrna_boustrophedon](#) (size_t start, size_t end)

This function is available as overloaded global function `boustrophedon()`.

Global [vrna_boustrophedon_pos](#) (size_t start, size_t end, size_t pos)

This function is available as overloaded global function `boustrophedon()`. Omitting the `pos` argument yields the entire sequence from `start` to `end`.

Global [vrna_bp_distance](#) (const char *str1, const char *str2)

This function is available as an overloaded method `bp_distance()`. Note that the SWIG wrapper takes two structure in dot-bracket notation and converts them into pair tables using `vrna_ptable_from_string()`. The resulting pair tables are then internally passed to `vrna_bp_distance_pt()`. To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of `vrna_ptable_from_string()` for available options. (default: `VRNA_BRACKETS_RND`).

Global [vrna_bp_distance_pt](#) (const short *pt1, const short *pt2)

This function is available as an overloaded method `bp_distance()`.

Global [vrna_db_flatten](#) (char *structure, unsigned int options)

This function flattens an input structure string in-place! The second parameter is optional and defaults to `VRNA_BRACKETS_DEFAULT`.

An overloaded version of this function exists, where an additional second parameter can be passed to specify the target brackets, i.e. the type of matching pair characters all brackets will be flattened to. Therefore, in the scripting language interface this function is a replacement for `vrna_db_flatten_to()`.

Global [vrna_db_flatten_to](#) (char *string, const char target[3], unsigned int options)

This function is available as an overloaded version of `vrna_db_flatten()`

Global `vrna_db_from_probs` (const FLT_OR_DBL *pr, unsigned int length)

This function is available as parameter-less method `db_from_probs()` bound to objects of type *fold_compound*. Parameters `pr` and `length` are implicitly taken from the *fold_compound* object the method is bound to. Upon missing base pair probabilities, this method returns an empty string.

Global `vrna_db_pk_remove` (const char *structure, unsigned int options)

This function is available as an overloaded function `db_pk_remove()` where the optional second parameter `options` defaults to `VRNA_BRACKETS_ANY`.

Global `vrna_ensemble_defect` (vrna_fold_compound_t *fc, const char *structure)

This function is attached as method `ensemble_defect()` to objects of type *fold_compound*. Note that the SWIG wrapper takes a structure in dot-bracket notation and converts it into a pair table using `vrna_ptable_from_string()`. The resulting pair table is then internally passed to `vrna_ensemble_defect_pt()`. To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of `vrna_ptable_from_string()` for available options. (default: `VRNA_BRACKETS_RND`).

Global `vrna_ensemble_defect_pt` (vrna_fold_compound_t *fc, const short *pt)

This function is attached as overloaded method `ensemble_defect()` to objects of type *fold_compound*.

Global `vrna_enumerate_necklaces` (const unsigned int *type_counts)

This function is available as global function `enumerate_necklaces()` which accepts lists input, and produces list of lists output.

Global `vrna_eval_circ_consensus_structure` (const char **alignment, const char *structure)

This function is available through an overloaded version of `vrna_eval_circ_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

Global `vrna_eval_circ_consensus_structure_v` (const char **alignment, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of `vrna_eval_circ_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

Global `vrna_eval_circ_gquad_consensus_structure` (const char **alignment, const char *structure)

This function is available through an overloaded version of `vrna_eval_circ_gquad_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

Global `vrna_eval_circ_gquad_consensus_structure_v` (const char **alignment, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of `vrna_eval_circ_gquad_structure()`. Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

Global `vrna_eval_circ_gquad_structure` (const char *string, const char *structure)

In the target scripting language, this function serves as a wrapper for `vrna_eval_circ_gquad_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

Global `vrna_eval_circ_gquad_structure_v` (const char *string, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of `vrna_eval_circ_gquad_structure()`. The last two arguments for this function are optional and default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

Global `vrna_eval_circ_structure` (const char *string, const char *structure)

In the target scripting language, this function serves as a wrapper for `vrna_eval_circ_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

Global `vrna_eval_circ_structure_v` (const char *string, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of `vrna_eval_circ_structure()`. The last two arguments for this function are optional and default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

Global **[vrna_eval_consensus_structure_pt_simple](#)** (const char **alignment, const short *pt)

This function is available through an overloaded version of [vrna_eval_structure_pt_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

Global **[vrna_eval_consensus_structure_pt_simple_v](#)** (const char **alignment, const short *pt, int verbosity_level, FILE *file)

This function is available through an overloaded version of [vrna_eval_structure_pt_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global **[vrna_eval_consensus_structure_pt_simple_verbose](#)** (const char **alignment, const short *pt, FILE *file)

This function is not available. Use [vrna_eval_consensus_structure_pt_v\(\)](#) instead!

Global **[vrna_eval_consensus_structure_simple](#)** (const char **alignment, const char *structure)

This function is available through an overloaded version of [vrna_eval_structure_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

Global **[vrna_eval_consensus_structure_simple_v](#)** (const char **alignment, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of [vrna_eval_structure_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global **[vrna_eval_consensus_structure_simple_verbose](#)** (const char **alignment, const char *structure, FILE *file)

This function is not available. Use [vrna_eval_consensus_structure_simple_v\(\)](#) instead!

Global **[vrna_eval_covar_structure](#)** (vrna_fold_compound_t *fc, const char *structure)

This function is attached as method [eval_covar_structure\(\)](#) to objects of type *fold_compound*

Global **[vrna_eval_gquad_consensus_structure](#)** (const char **alignment, const char *structure)

This function is available through an overloaded version of [vrna_eval_gquad_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

Global **[vrna_eval_gquad_consensus_structure_v](#)** (const char **alignment, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of [vrna_eval_gquad_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global **[vrna_eval_gquad_structure](#)** (const char *string, const char *structure)

In the target scripting language, this function serves as a wrapper for [vrna_eval_gquad_structure_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global **[vrna_eval_gquad_structure_v](#)** (const char *string, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of [vrna_eval_gquad_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global **[vrna_eval_hp_loop](#)** (vrna_fold_compound_t *fc, int i, int j)

This function is attached as method [eval_hp_loop\(\)](#) to objects of type *fold_compound*

Global **[vrna_eval_int_loop](#)** (vrna_fold_compound_t *fc, int i, int j, int k, int l)

This function is attached as method [eval_int_loop\(\)](#) to objects of type *fold_compound*

Global **[vrna_eval_loop_pt](#)** (vrna_fold_compound_t *fc, int i, const short *pt)

This function is attached as method [eval_loop_pt\(\)](#) to objects of type *fold_compound*

Global **[vrna_eval_move](#)** (vrna_fold_compound_t *fc, const char *structure, int m1, int m2)

This function is attached as method [eval_move\(\)](#) to objects of type *fold_compound*

Global [vrna_eval_move_pt](#) ([vrna_fold_compound_t](#) *fc, short *pt, int m1, int m2)

This function is attached as method [eval_move_pt\(\)](#) to objects of type *fold_compound*

Global [vrna_eval_structure](#) ([vrna_fold_compound_t](#) *fc, const char *structure)

This function is attached as method [eval_structure\(\)](#) to objects of type *fold_compound*

Global [vrna_eval_structure_pt](#) ([vrna_fold_compound_t](#) *fc, const short *pt)

This function is attached as method [eval_structure_pt\(\)](#) to objects of type *fold_compound*

Global [vrna_eval_structure_pt_simple](#) (const char *string, const short *pt)

In the target scripting language, this function serves as a wrapper for [vrna_eval_structure_pt_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global [vrna_eval_structure_pt_verbose](#) ([vrna_fold_compound_t](#) *fc, const short *pt, FILE *file)

This function is attached as method [eval_structure_pt_verbose\(\)](#) to objects of type *fold_compound*

Global [vrna_eval_structure_simple](#) (const char *string, const char *structure)

In the target scripting language, this function serves as a wrapper for [vrna_eval_structure_simple_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global [vrna_eval_structure_simple_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)

This function is available through an overloaded version of [vrna_eval_structure_simple\(\)](#). The last two arguments for this function are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

Global [vrna_eval_structure_simple_verbose](#) (const char *string, const char *structure, FILE *file)

This function is not available. Use [vrna_eval_structure_simple_v\(\)](#) instead!

Global [vrna_eval_structure_verbose](#) ([vrna_fold_compound_t](#) *fc, const char *structure, FILE *file)

This function is attached as method [eval_structure_verbose\(\)](#) to objects of type *fold_compound*

Global [vrna_exp_params_rescale](#) ([vrna_fold_compound_t](#) *vc, double *mfe)

This function is attached to [vrna_fc_s](#) objects as overloaded [exp_params_rescale\(\)](#) method.

When no parameter is passed to this method, the resulting action is the same as passing *NULL* as second parameter to [vrna_exp_params_rescale\(\)](#), i.e. default scaling of the partition function. Passing an energy in kcal/mol, e.g. as retrieved by a previous call to the [mfe\(\)](#) method, instructs all subsequent calls to scale the partition function accordingly.

Global [vrna_exp_params_reset](#) ([vrna_fold_compound_t](#) *vc, [vrna_md_t](#) *md_p)

This function is attached to [vrna_fc_s](#) objects as overloaded [exp_params_reset\(\)](#) method.

When no parameter is passed to this method, the resulting action is the same as passing *NULL* as second parameter to [vrna_exp_params_reset\(\)](#), i.e. global default model settings are used. Passing an object of type [vrna_md_s](#) resets the fold compound according to the specifications stored within the [vrna_md_s](#) object.

Global [vrna_exp_params_subst](#) ([vrna_fold_compound_t](#) *vc, [vrna_exp_param_t](#) *params)

This function is attached to [vrna_fc_s](#) objects as overloaded [exp_params_subst\(\)](#) method.

When no parameter is passed, the resulting action is the same as passing *NULL* as second parameter to [vrna_exp_params_subst\(\)](#), i.e. resetting the parameters to the global defaults.

Class [vrna_fc_s](#)

This data structure is wrapped as an object **fold_compound** with several related functions attached as methods.

A new **fold_compound** can be obtained by calling one of its constructors:

- *fold_compound(seq)* – Initialize with a single sequence, or two concatenated sequences separated by an ampersand character '&' (for cofolding)
- *fold_compound(aln)* – Initialize with a sequence alignment *aln* stored as a list of sequences (with gap characters)

The resulting object has a list of attached methods which in most cases directly correspond to functions that mainly operate on the corresponding C data structure:

- `type()` – Get the type of the `fold_compound` (See `vrna_fc_type_e`)
- `length()` – Get the length of the `sequence(s)` or alignment stored within the `fold_compound`

Global `vrna_file_commands_apply` (`vrna_fold_compound_t *vc`, `const char *filename`, `unsigned int options`)

This function is attached as method `file_commands_apply()` to objects of type `fold_compound`

Global `vrna_file_msa_detect_format` (`const char *filename`, `unsigned int options`)

This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to `VRNA_FILE_FORMAT_MSA_DEFAULT`.

Global `vrna_file_msa_read` (`const char *filename`, `char ***names`, `char ***aln`, `char **id`, `char **structure`, `unsigned int options`)

In the target scripting language, only the first and last argument, `filename` and `options`, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

Global `vrna_file_msa_read_record` (`FILE *fp`, `char ***names`, `char ***aln`, `char **id`, `char **structure`, `unsigned int options`)

In the target scripting language, only the first and last argument, `fp` and `options`, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

Global `vrna_file_msa_write` (`const char *filename`, `const char **names`, `const char **aln`, `const char *id`, `const char *structure`, `const char *source`, `unsigned int options`)

In the target scripting language, this function exists as a set of overloaded versions, where the last four parameters may be omitted. If the `options` parameter is missing the options default to (`VRNA_FILE_FORMAT_MSA_STOCKHOLM` | `VRNA_FILE_FORMAT_MSA_APPEND`).

Global `vrna_file_PS_aln` (`const char *filename`, `const char **seqs`, `const char **names`, `const char *structure`, `unsigned int columns`)

This function is available as overloaded function `file_PS_aln()` with three additional parameters `start`, `end`, and `offset` before the `columns` argument. Thus, it resembles the `vrna_file_PS_aln_slice()` function. The last four arguments may be omitted, indicating the default of `start = 0`, `end = 0`, `offset = 0`, and `columns = 60`.

Global `vrna_file_PS_aln_slice` (`const char *filename`, `const char **seqs`, `const char **names`, `const char *structure`, `unsigned int start`, `unsigned int end`, `int offset`, `unsigned int columns`)

This function is available as overloaded function `file_PS_aln()` where the last four parameter may be omitted, indicating `start = 0`, `end = 0`, `offset = 0`, and `columns = 60`.

Global `vrna_hc_add_from_db` (`vrna_fold_compound_t *vc`, `const char *constraint`, `unsigned int options`)

This function is attached as method `hc_add_from_db()` to objects of type `fold_compound`

Global `vrna_hc_init` (`vrna_fold_compound_t *vc`)

This function is attached as method `hc_init()` to objects of type `fold_compound`

Global `vrna_heat_capacity` (`vrna_fold_compound_t *fc`, `float T_min`, `float T_max`, `float T_increment`, `unsigned int mpoints`)

This function is attached as overloaded method `heat_capacity()` to objects of type `fold_compound`. If the optional function arguments `T_min`, `T_max`, `T_increment`, and `mpoints` are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

Global `vrna_heat_capacity_cb` (`vrna_fold_compound_t *fc`, `float T_min`, `float T_max`, `float T_increment`, `unsigned int mpoints`, `vrna_heat_capacity_f cb`, `void *data`)

This function is attached as method `heat_capacity_cb()` to objects of type `fold_compound`

Global **vrna_heat_capacity_simple** (const char *sequence, float T_min, float T_max, float T_increment, unsigned int mpoints)

This function is available as overloaded function **heat_capacity()**. If the optional function arguments T_min, T_max, T_increment, and mpoints are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

Global **vrna_init_rand_seed** (unsigned int seed)

This function is available as an overloaded function **init_rand()** where the argument seed is optional.

Global **vrna_maximum_matching** (vrna_fold_compound_t *fc)

This function is attached as method **maximum_matching()** to objects of type *fold_compound* (i.e. *vrna_fold_compound_t*).

Global **vrna_maximum_matching_simple** (const char *sequence)

This function is available as global function **maximum_matching()**.

Class **vrna_md_s**

This data structure is wrapped as an object **md** with multiple related functions attached as methods.

A new set of default parameters can be obtained by calling the constructor of **md**:

- *md()* – Initialize with default settings

The resulting object has a list of attached methods which directly correspond to functions that mainly operate on the corresponding C data structure:

- *reset()* – *vrna_md_set_default()*
- *set_from_globals()* – *set_model_details()*
- *option_string()* – *vrna_md_option_string()*

Note, that default parameters can be modified by directly setting any of the following global variables. Internally, getting/setting default parameters using their global variable representative translates into calls of the following functions, therefore these wrappers for these functions do not exist in the scripting language interface(s):

Global **vrna_MEA** (vrna_fold_compound_t *fc, double gamma, float *mea)

This function is attached as overloaded method **MEA**(gamma = 1.) to objects of type *fold_compound*. Note, that it returns the MEA structure and MEA value as a tuple (MEA_structure, MEA)

Global **vrna_MEA_from_plist** (vrna_ep_t *plist, const char *sequence, double gamma, vrna_md_t *md, float *mea)

This function is available as overloaded function **MEA_from_plist**(gamma = 1., md = NULL). Note, that it returns the MEA structure and MEA value as a tuple (MEA_structure, MEA)

Global **vrna_mean_bp_distance** (vrna_fold_compound_t *vc)

This function is attached as method **mean_bp_distance()** to objects of type *fold_compound*

Global **vrna_mfe** (vrna_fold_compound_t *vc, char *structure)

This function is attached as method **mfe()** to objects of type *fold_compound*

Global **vrna_mfe_dimer** (vrna_fold_compound_t *vc, char *structure)

This function is attached as method **mfe_dimer()** to objects of type *fold_compound*

Global **vrna_mfe_window** (vrna_fold_compound_t *vc, FILE *file)

This function is attached as method **mfe_window()** to objects of type *fold_compound*

Global **vrna_neighbors** (vrna_fold_compound_t *vc, const short *pt, unsigned int options)

This function is attached as an overloaded method **neighbors()** to objects of type *fold_compound*. The optional parameter options defaults to **VRNA_MOVESET_DEFAULT** if it is omitted.

Global **vrna_params_load** (const char fname[], unsigned int options)

This function is available as overloaded function **params_load**(fname="", options=**VRNA_PARAMETER_FORMAT_DEFAULT**). Here, the empty filename string indicates to load default RNA parameters, i.e. this is equivalent to calling *vrna_params_load_defaults()*.

Global `vrna_params_load_defaults` (void)

This function is available as overloaded function `params_load()`.

Global `vrna_params_load_DNA_Mathews1999` (void)

This function is available as function `params_load_DNA_Mathews1999()`.

Global `vrna_params_load_DNA_Mathews2004` (void)

This function is available as function `params_load_DNA_Mathews2004()`.

Global `vrna_params_load_from_string` (const char *string, const char *name, unsigned int options)

This function is available as overloaded function `params_load_from_string(string, name="", options=VRNA_PARAMETER_FORMAT_DEFAULT)`.

Global `vrna_params_load_RNA_Andronescu2007` (void)

This function is available as function `params_load_RNA_Andronescu2007()`.

Global `vrna_params_load_RNA_Langdon2018` (void)

This function is available as function `params_load_RNA_Langdon2018()`.

Global `vrna_params_load_RNA_misc_special_hairpins` (void)

This function is available as function `params_load_RNA_misc_special_hairpins()`.

Global `vrna_params_load_RNA_Turner1999` (void)

This function is available as function `params_load_RNA_Turner1999()`.

Global `vrna_params_load_RNA_Turner2004` (void)

This function is available as function `params_load_RNA_Turner2004()`.

Global `vrna_params_reset` (vrna_fold_compound_t *vc, vrna_md_t *md_p)

This function is attached to `vrna_fc_s` objects as overloaded `params_reset()` method.

When no parameter is passed to this method, the resulting action is the same as passing `NULL` as second parameter to `vrna_params_reset()`, i.e. global default model settings are used. Passing an object of type `vrna_md_s` resets the fold compound according to the specifications stored within the `vrna_md_s` object.

Global `vrna_params_save` (const char fname[], unsigned int options)

This function is available as overloaded function `params_save(fname, options=VRNA_PARAMETER_FORMAT_DEFAULT)`.

Global `vrna_params_subst` (vrna_fold_compound_t *vc, vrna_param_t *par)

This function is attached to `vrna_fc_s` objects as overloaded `params_subst()` method.

When no parameter is passed, the resulting action is the same as passing `NULL` as second parameter to `vrna_params_subst()`, i.e. resetting the parameters to the global defaults.

Global `vrna_path` (vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)

This function is attached as an overloaded method `path()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

Global `vrna_path_direct` (vrna_fold_compound_t *fc, const char *s1, const char *s2, vrna_path_options_t options)

This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `options` defaults to `NULL` if it is omitted.

Global `vrna_path_direct_ub` (vrna_fold_compound_t *fc, const char *s1, const char *s2, int maxE, vrna_path_options_t options)

This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `maxE` defaults to `#INT_MAX - 1` if it is omitted, while the optional parameter `options` defaults to `NULL`. In case the function did not find a path with $E_{\text{saddle}} < E_{\text{max}}$ it returns an empty list.

Global `vrna_path_findpath` (vrna_fold_compound_t *fc, const char *s1, const char *s2, int width)

This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted.

Global `vrna_path_findpath_saddle` (vrna_fold_compound_t *fc, const char *s1, const char *s2, int width)

This function is attached as an overloaded method `path_findpath_saddle()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted.

Global **`vrna_path_findpath_saddle_ub`** (`vrna_fold_compound_t *fc`, `const char *s1`, `const char *s2`, `int width`, `int maxE`)

This function is attached as an overloaded method `path_findpath_saddle()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to INF. In case the function did not find a path with $E_{saddle} < E_{max}$ the function returns a `NULL` object, i.e. `undef` for Perl and `None` for Python.

Global **`vrna_path_findpath_ub`** (`vrna_fold_compound_t *fc`, `const char *s1`, `const char *s2`, `int width`, `int maxE`)

This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to INF. In case the function did not find a path with $E_{saddle} < E_{max}$ the function returns an empty list.

Global **`vrna_path_gradient`** (`vrna_fold_compound_t *vc`, `short *pt`, `unsigned int options`)

This function is attached as an overloaded method `path_gradient()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

Global **`vrna_path_options_findpath`** (`int width`, `unsigned int type`)

This function is available as overloaded function `path_options_findpath()`. The optional parameter `width` defaults to 10 if omitted, while the optional parameter `type` defaults to `VRNA_PATH_TYPE_DOT_BRACKET`.

Global **`vrna_path_random`** (`vrna_fold_compound_t *vc`, `short *pt`, `unsigned int steps`, `unsigned int options`)

This function is attached as an overloaded method `path_gradient()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_PATH_DEFAULT` if it is omitted.

Global **`vrna_pbacktrack`** (`vrna_fold_compound_t *fc`)

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack5`** (`vrna_fold_compound_t *fc`, `unsigned int length`)

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack5_cb`** (`vrna_fold_compound_t *fc`, `unsigned int num_samples`, `unsigned int length`, `vrna_bs_result_f cb`, `void *data`, `unsigned int options`)

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack5_num`** (`vrna_fold_compound_t *fc`, `unsigned int num_samples`, `unsigned int length`, `unsigned int options`)

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack5_resume`** (`vrna_fold_compound_t *vc`, `unsigned int num_samples`, `unsigned int length`, `vrna_pbacktrack_mem_t *nr_mem`, `unsigned int options`)

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack5_resume_cb`** (`vrna_fold_compound_t *fc`, `unsigned int num_samples`, `unsigned int length`, `vrna_bs_result_f cb`, `void *data`, `vrna_pbacktrack_mem_t *nr_mem`, `unsigned int options`)

This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_cb`** (`vrna_fold_compound_t *fc`, `unsigned int num_samples`, `vrna_bs_result_f cb`, `void *data`, `unsigned int options`)

This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_num`** (`vrna_fold_compound_t *fc`, unsigned int num_samples, unsigned int options)

This function is attached as overloaded method **`pbacktrack()`** to objects of type *fold_compound* where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_resume`** (`vrna_fold_compound_t *fc`, unsigned int num_samples, `vrna_pbacktrack_mem_t *nr_mem`, unsigned int options)

This function is attached as overloaded method **`pbacktrack()`** to objects of type *fold_compound*. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_resume_cb`** (`vrna_fold_compound_t *fc`, unsigned int num_samples, `vrna_bs_result_f cb`, void *data, `vrna_pbacktrack_mem_t *nr_mem`, unsigned int options)

This function is attached as overloaded method **`pbacktrack()`** to objects of type *fold_compound*. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_sub`** (`vrna_fold_compound_t *fc`, unsigned int start, unsigned int end)

This function is attached as overloaded method **`pbacktrack_sub()`** to objects of type *fold_compound*. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_sub_cb`** (`vrna_fold_compound_t *fc`, unsigned int num_samples, unsigned int start, unsigned int end, `vrna_bs_result_f cb`, void *data, unsigned int options)

This function is attached as overloaded method **`pbacktrack()`** to objects of type *fold_compound* where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_sub_num`** (`vrna_fold_compound_t *fc`, unsigned int num_samples, unsigned int start, unsigned int end, unsigned int options)

This function is attached as overloaded method **`pbacktrack_sub()`** to objects of type *fold_compound* where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_sub_resume`** (`vrna_fold_compound_t *vc`, unsigned int num_samples, unsigned int start, unsigned int end, `vrna_pbacktrack_mem_t *nr_mem`, unsigned int options)

This function is attached as overloaded method **`pbacktrack()`** to objects of type *fold_compound*. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pbacktrack_sub_resume_cb`** (`vrna_fold_compound_t *fc`, unsigned int num_samples, unsigned int start, unsigned int end, `vrna_bs_result_f cb`, void *data, `vrna_pbacktrack_mem_t *nr_mem`, unsigned int options)

This function is attached as overloaded method **`pbacktrack()`** to objects of type *fold_compound*. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

Global **`vrna_pf`** (`vrna_fold_compound_t *vc`, char *structure)

This function is attached as method **`pf()`** to objects of type *fold_compound*

Global **`vrna_pf_dimer`** (`vrna_fold_compound_t *vc`, char *structure)

This function is attached as method **`pf_dimer()`** to objects of type *fold_compound*

Global **`vrna_positional_entropy`** (`vrna_fold_compound_t *fc`)

This function is attached as method **`positional_entropy()`** to objects of type *fold_compound*

Global **`vrna_pr_energy`** (`vrna_fold_compound_t *vc`, double e)

This function is attached as method **`pr_energy()`** to objects of type *fold_compound*

Global **`vrna_pr_structure`** (`vrna_fold_compound_t *fc`, const char *structure)

This function is attached as method **`pr_structure()`** to objects of type *fold_compound*

Global `vrna_ptable` (`const char *structure`)

This functions is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

Global `vrna_ptable_from_string` (`const char *structure`, `unsigned int options`)

This functions is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

Global `vrna_rotational_symmetry` (`const char *string`)

This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details.

Global `vrna_rotational_symmetry_db` (`vrna_fold_compound_t *fc`, `const char *structure`)

This function is attached as method `rotational_symmetry_db()` to objects of type `fold_compound` (i.e. `vrna_fold_compound_t`). See `vrna_rotational_symmetry_db_pos()` for details.

Global `vrna_rotational_symmetry_db_pos` (`vrna_fold_compound_t *fc`, `const char *structure`, `unsigned int **positions`)

This function is attached as method `rotational_symmetry_db()` to objects of type `fold_compound` (i.e. `vrna_fold_compound_t`). Thus, the first argument must be omitted. In contrast to our C-implementation, this function doesn't simply return the order of rotational symmetry of the secondary structure, but returns the list `position` of cyclic permutation shifts that result in a rotationally symmetric structure. The length of the list then determines the order of rotational symmetry.

Global `vrna_rotational_symmetry_num` (`const unsigned int *string`, `size_t string_length`)

This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

Global `vrna_rotational_symmetry_pos` (`const char *string`, `unsigned int **positions`)

This function is available as overloaded global function `rotational_symmetry()`. It merges the functionalities of `vrna_rotational_symmetry()`, `vrna_rotational_symmetry_pos()`, `vrna_rotational_symmetry_num()`, and `vrna_rotational_symmetry_pos_num()`. In contrast to our C-implementation, this function doesn't return the order of rotational symmetry as a single value, but returns a list of cyclic permutation shifts that result in a rotationally symmetric string. The length of the list then determines the order of rotational symmetry.

Global `vrna_rotational_symmetry_pos_num` (`const unsigned int *string`, `size_t string_length`, `unsigned int **positions`)

This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

Global `vrna_sc_add_bp` (`vrna_fold_compound_t *vc`, `int i`, `int j`, `FLT_OR_DBL energy`, `unsigned int options`)

This function is attached as an overloaded method `sc_add_bp()` to objects of type `fold_compound`. The method either takes arguments for a single base pair (i,j) with the corresponding energy value:

Global `vrna_sc_add_bt` (`vrna_fold_compound_t *vc`, `vrna_sc_bt_f f`)

This function is attached as method `sc_add_bt()` to objects of type `fold_compound`

Global `vrna_sc_add_data` (`vrna_fold_compound_t *vc`, `void *data`, `vrna_auxdata_free_f free_data`)

This function is attached as method `sc_add_data()` to objects of type `fold_compound`

Global `vrna_sc_add_exp_f` (`vrna_fold_compound_t *vc`, `vrna_sc_exp_f exp_f`)

This function is attached as method `sc_add_exp_f()` to objects of type `fold_compound`

Global `vrna_sc_add_f` (`vrna_fold_compound_t *vc`, `vrna_sc_f f`)

This function is attached as method `sc_add_f()` to objects of type `fold_compound`

Global `vrna_sc_add_hi_motif` (`vrna_fold_compound_t *fc`, `const char *seq`, `const char *structure`, `FLT_OR_DBL energy`, `unsigned int options`)

This function is attached as method `sc_add_hi_motif()` to objects of type `fold_compound`

Global **vrna_sc_add_SHAPE_deigan** (vrna_fold_compound_t *vc, const double *reactivities, double m, double b, unsigned int options)

This function is attached as method **sc_add_SHAPE_deigan()** to objects of type *fold_compound*

Global **vrna_sc_add_SHAPE_deigan_ali** (vrna_fold_compound_t *vc, const char **shape_files, const int *shape_file_association, double m, double b, unsigned int options)

This function is attached as method **sc_add_SHAPE_deigan_ali()** to objects of type *fold_compound*

Global **vrna_sc_add_SHAPE_zarringhalam** (vrna_fold_compound_t *vc, const double *reactivities, double b, double default_value, const char *shape_conversion, unsigned int options)

This function is attached as method **sc_add_SHAPE_zarringhalam()** to objects of type *fold_compound*

Global **vrna_sc_add_up** (vrna_fold_compound_t *vc, int i, FLT_OR_DBL energy, unsigned int options)

This function is attached as an overloaded method **sc_add_up()** to objects of type *fold_compound*. The method either takes arguments for a single nucleotide *i* with the corresponding energy value:

Global **vrna_sc_init** (vrna_fold_compound_t *vc)

This function is attached as method **sc_init()** to objects of type *fold_compound*

Global **vrna_sc_mod** (vrna_fold_compound_t *fc, const vrna_sc_mod_param_t params, const unsigned int *modification_sites)

This function is attached as method **sc_mod()** to objects of type *fold_compound*

Global **vrna_sc_mod_7DA** (vrna_fold_compound_t *fc, const unsigned int *modification_sites)

This function is attached as method **sc_mod_7DA()** to objects of type *fold_compound*

Global **vrna_sc_mod_dihydrouridine** (vrna_fold_compound_t *fc, const unsigned int *modification_sites)

This function is attached as method **sc_mod_dihydrouridine()** to objects of type *fold_compound*

Global **vrna_sc_mod_inosine** (vrna_fold_compound_t *fc, const unsigned int *modification_sites)

This function is attached as method **sc_mod_inosine()** to objects of type *fold_compound*

Global **vrna_sc_mod_json** (vrna_fold_compound_t *fc, const char *json, const unsigned int *modification_sites)

This function is attached as method **sc_mod_json()** to objects of type *fold_compound*

Global **vrna_sc_mod_jsonfile** (vrna_fold_compound_t *fc, const char *json_file, const unsigned int *modification_sites)

This function is attached as method **sc_mod_jsonfile()** to objects of type *fold_compound*

Global **vrna_sc_mod_m6A** (vrna_fold_compound_t *fc, const unsigned int *modification_sites)

This function is attached as method **sc_mod_m6A()** to objects of type *fold_compound*

Global **vrna_sc_mod_parameters_free** (vrna_sc_mod_param_t params)

This function is available as function **sc_mod_parameters_free()**

Global **vrna_sc_mod_pseudouridine** (vrna_fold_compound_t *fc, const unsigned int *modification_sites)

This function is attached as method **sc_mod_pseudouridine()** to objects of type *fold_compound*

Global **vrna_sc_mod_purine** (vrna_fold_compound_t *fc, const unsigned int *modification_sites)

This function is attached as method **sc_mod_purine()** to objects of type *fold_compound*

Global **vrna_sc_mod_read_from_json** (const char *json, vrna_md_t *md)

This function is available as an overloaded function **sc_mod_read_from_json()** where the *md* parameter may be omitted

Global **vrna_sc_mod_read_from_jsonfile** (const char *filename, vrna_md_t *md)

This function is available as an overloaded function **sc_mod_read_from_jsonfile()** where the *md* parameter may be omitted

Global **vrna_sc_remove** (vrna_fold_compound_t *vc)

This function is attached as method **sc_remove()** to objects of type *fold_compound*

Global **vrna_sc_set_bp** (vrna_fold_compound_t *vc, const FLT_OR_DBL **constraints, unsigned int options)

This function is attached as method **sc_set_bp()** to objects of type *fold_compound*

Global **vrna_sc_set_up** (vrna_fold_compound_t *vc, const FLT_OR_DBL *constraints, unsigned int options)

This function is attached as method **sc_set_up()** to objects of type *fold_compound*

Global **vrna_seq_encode** (const char *sequence, vrna_md_t *md)

In the target scripting language, this function is wrapped as overloaded function *seq_encode()* where the last parameter, the *model_details* data structure, is optional. If it is omitted, default model settings are applied, i.e. default nucleotide letter conversion. The wrapped function returns a list/tuple of integer representations of the input sequence.

Global **vrna_strtrim** (char *string, const char *delimiters, unsigned int keep, unsigned int options)

Since many scripting languages treat strings as immutable objects, this function does not modify the input string directly. Instead, it returns the modified string as second return value, together with the number of removed delimiters.

The scripting language interface provides an overloaded version of this function, with default parameters *delimiters=NULL*, *keep=0*, and *options=VRNA_TRIM_DEFAULT*.

Global **vrna_subopt** (vrna_fold_compound_t *fc, int delta, int sorted, FILE *fp)

This function is attached as method **subopt()** to objects of type *fold_compound*

Global **vrna_subopt_cb** (vrna_fold_compound_t *fc, int delta, vrna_subopt_result_f cb, void *data)

This function is attached as method **subopt_cb()** to objects of type *fold_compound*

Global **vrna_subopt_zuker** (vrna_fold_compound_t *fc)

This function is attached as method **subopt_zuker()** to objects of type *fold_compound*

Global **vrna_ud_remove** (vrna_fold_compound_t *vc)

This function is attached as method **ud_remove()** to objects of type *fold_compound*

Global **vrna_ud_set_data** (vrna_fold_compound_t *vc, void *data, vrna_auxdata_free_f free_cb)

This function is attached as method **ud_set_data()** to objects of type *fold_compound*

Global **vrna_ud_set_exp_prod_rule_cb** (vrna_fold_compound_t *vc, vrna_ud_exp_production_f pre_cb, vrna_ud_exp_f exp_e_cb)

This function is attached as method **ud_set_exp_prod_rule_cb()** to objects of type *fold_compound*

Global **vrna_ud_set_prob_cb** (vrna_fold_compound_t *vc, vrna_ud_add_probs_f setter, vrna_ud_get_probs_f getter)

This function is attached as method **ud_set_prob_cb()** to objects of type *fold_compound*

Global **vrna_ud_set_prod_rule_cb** (vrna_fold_compound_t *vc, vrna_ud_production_f pre_cb, vrna_ud_f exp_e_cb)

This function is attached as method **ud_set_prod_rule_cb()** to objects of type *fold_compound*

Chapter 7

Additional Utilities

Chapter 8

Examples

- [C Examples](#)
- [Perl5 Examples](#)
- [Python Examples](#)

8.1 C Examples

8.1.1 Hello World Examples

helloworld_mfe.c

The following is an example showing the minimal requirements to compute the Minimum Free Energy (MFE) and corresponding secondary structure of an RNA sequence

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/fold.h>
#include <ViennaRNA/Utils/basic.h>

int
main()
{
    /* The RNA sequence */
    char *seq = "GAGUAGUGGAACCAGGCUAUGUUUGUGACUCGACAGACUACA";

    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_fold(seq, structure);

    /* print sequence, structure and MFE */
    printf("%s\n%s [ %.2f ]\n", seq, structure, mfe);

    /* cleanup memory */
    free(structure);

    return 0;
}
```

See also

`examples/helloworld_mfe.c` in the source code tarball

helloworld_mfe_comparative.c

Instead of using a single sequence as done above, this example predicts a consensus structure for a multiple sequence alignment

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/alifold.h>
#include <ViennaRNA/Utils/basic.h>
```

```
#include <ViennaRNA/utils/alignments.h>

int
main()
{
    /* The RNA sequence alignment */
    const char *sequences[] = {
        "CUGCCUCACAACGUUUUGGCCUCAGUUACCCGUGAGAUGUAGUGAGGGU",
        "CUGCCUCACAACAUUUUGGCCUCAGUUACUAGUAGUAGUGAGGGU",
        "---CUCGACACCACU---GCCUCGGUUAUCCAUCCGUGCAGUGCGGGU",
        NULL /* indicates end of alignment */
    };

    /* compute the consensus sequence */
    char *cons = consensus(sequences);

    /* allocate memory for MFE consensus structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(sequences[0]) + 1));

    /* predict Minimum Free Energy and corresponding secondary structure */
    float mfe = vrna_alifold(sequences, structure);

    /* print consensus sequence, structure and MFE */
    printf("%s\n%s [ %.2f ]\n", cons, structure, mfe);

    /* cleanup memory */
    free(cons);
    free(structure);

    return 0;
}
```

See also

examples/helloworld_mfe_comparative.c in the source code tarball

helloworld_probabilities.c

This example shows how to compute the partition function and base pair probabilities with minimal implementation effort.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/fold.h>
#include <ViennaRNA/part_func.h>
#include <ViennaRNA/utils/basic.h>

int
main()
{
    /* The RNA sequence */
    char *seq = "GAGUAGUGGAACAGGCUAUGUUUGACUCGCAGACUAACA";

    /* allocate memory for pairing propensity string (length + 1) */
    char *propensity = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* pointers for storing and navigating through base pair probabilities */
    vrna_ep_t *ptr, *pair_probabilities = NULL;

    float en = vrna_pf_fold(seq, propensity, &pair_probabilities);

    /* print sequence, pairing propensity string and ensemble free energy */
    printf("%s\n%s [ %.2f ]\n", seq, propensity, en);

    /* print all base pairs with probability above 50% */
    for (ptr = pair_probabilities; ptr->i != 0; ptr++)
        if (ptr->p > 0.5)
            printf("p(%d, %d) = %g\n", ptr->i, ptr->j, ptr->p);

    /* cleanup memory */
    free(pair_probabilities);
    free(propensity);

    return 0;
}
```

See also

examples/helloworld_probabilities.c in the source code tarball

8.1.2 First Steps with the Fold Compound**fold_compound_mfe.c**

Instead of calling the simple MFE folding interface `vrna_fold()`, this example shows how to first create a `vrna_fold_compound_t` container with the RNA sequence to finally compute the MFE using this container. This is especially useful if non-default model settings are applied or the dynamic programming (DP) matrices of the MFE prediction are required for post-processing operations, or other tasks on the same sequence will be performed.

```
#include <stdlib.h>
#include <stdio.h>

#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/mfe.h>

int
main()
{
    /* initialize random number generator */
    vrna_init_rand();

    /* Generate a random sequence of 50 nucleotides */
    char *seq = vrna_random_string(50, "ACGU");

    /* Create a fold compound for the sequence */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);

    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* predict Minnum Free Energy and corresponding secondary structure */
    float mfe = vrna_mfe(fc, structure);

    /* print sequence, structure and MFE */
    printf("%s\n%s [ %.2f ]\n", seq, structure, mfe);

    /* cleanup memory */
    free(seq);
    free(structure);
    vrna_fold_compound_free(fc);

    return 0;
}
```

See also

examples/fold_compound_mfe.c in the source code tarball

fold_compound_md.c

In the following, we change the model settings (model details) to a temperature of 25 Degree Celcius, and activate G-Quadruplex precision.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <ViennaRNA/model.h>
#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/mfe.h>

int
main()
{
    /* initialize random number generator */
    vrna_init_rand();

    /* Generate a random sequence of 50 nucleotides */
    char *seq = vrna_random_string(50, "ACGU");

    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* create a new model details structure to store the Model Settings */
```

```

vrna_md_t md;

/* ALWAYS set default model settings first! */
vrna_md_set_default(&md);

/* change temperature and activate G-Quadruplex prediction */
md.temperature = 25.0; /* 25 Deg Celcius */
md.gquad       = 1;    /* Turn-on G-Quadruples support */

/* create a fold compound */
vrna_fold_compound_t *fc = vrna_fold_compound(seq, &md, VRNA_OPTION_DEFAULT);

/* predict Minum Free Energy and corresponding secondary structure */
float mfe = vrna_mfe(fc, structure);

/* print sequence, structure and MFE */
printf("%s\n%s [ %6.2f ]\n", seq, structure, mfe);

/* cleanup memory */
free(structure);
vrna_fold_compound_free(fc);

return 0;
}

```

See also

examples/fold_compound_md.c in the source code tarball

8.1.3 Writing Callback Functions

callback_subopt.c

Here is a basic example how to use the callback mechanism in `vrna_subopt_cb()`. It simply defines a callback function (see interface definition for `vrna_subopt_callback`) that prints the result and increases a counter variable.

```

#include <stdlib.h>
#include <stdio.h>

#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/subopt.h>

void
subopt_callback(const char *structure,
               float      energy,
               void       *data)
{
    /* simply print the result and increase the counter variable by 1 */
    if (structure)
        printf("%d.\t%s\t%6.2f\n", ((int *)data)++, structure, energy);
}

int
main()
{
    /* initialize random number generator */
    vrna_init_rand();

    /* Generate a random sequence of 50 nucleotides */
    char *seq = vrna_random_string(50, "ACGU");

    /* Create a fold compound for the sequence */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);

    int counter = 0;

    /*
     * call subopt to enumerate all secondary structures in an energy band of
     * 5 kcal/mol of the MFE and pass it the address of the callback and counter
     * variable
     */
    vrna_subopt_cb(fc, 500, &subopt_callback, (void *)&counter);

    /* cleanup memory */
    free(seq);
    vrna_fold_compound_free(fc);

    return 0;
}

```

See also

examples/callback_subopt.c in the source code tarball

8.1.4 Application of Soft Constraints**soft_constraints_up.c**

In this example, a random RNA sequence is generated to predict its MFE under the constraint that a particular nucleotide receives an additional bonus energy if it remains unpaired.

```
#include <stdlib.h>
#include <stdio.h>

#include <ViennaRNA/fold_compound.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/utils/strings.h>
#include <ViennaRNA/constraints/soft.h>
#include <ViennaRNA/mfe.h>

int
main()
{
    /* initialize random number generator */
    vrna_init_rand();

    /* Generate a random sequence of 50 nucleotides */
    char *seq = vrna_random_string(50, "ACGU");

    /* Create a fold compound for the sequence */
    vrna_fold_compound_t *fc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);

    /* Add soft constraint of -1.7 kcal/mol to nucleotide 5 whenever it appears in an unpaired context */
    vrna_sc_add_up(fc, 5, -1.7, VRNA_OPTION_DEFAULT);

    /* allocate memory for MFE structure (length + 1) */
    char *structure = (char *)vrna_alloc(sizeof(char) * 51);

    /* predict Minum Free Energy and corresponding secondary structure */
    float mfe = vrna_mfe(fc, structure);

    /* print segeunce, structure and MFE */
    printf("%s\n%s [ %.2f ]\n", seq, structure, mfe);

    /* cleanup memory */
    free(seq);
    free(structure);
    vrna_fold_compound_free(fc);

    return 0;
}
```

See also

examples/soft_constraints_up.c in the source code tarball

8.1.5 Other Examples**example1.c**

A more extensive example including MFE, Partition Function, and Centroid structure prediction.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <ViennaRNA/data_structures.h>
#include <ViennaRNA/params/basic.h>
#include <ViennaRNA/utils/basic.h>
#include <ViennaRNA/eval.h>
#include <ViennaRNA/fold.h>
#include <ViennaRNA/part_func.h>

int
main(int argc,
     char *argv[])
{
    char *seq =
        "AGACGACAAGGUUGAAUCGCACCCACAGUCUAUGAGUCGGUGACAACAUAUACGAAAGGCUGUAAAAUCAAUUAUCCACAGGGGGCCCCCGUGUCUAG";
    char *mfe_structure = vrna_alloc(sizeof(char) * (strlen(seq) + 1));
    char *prob_string = vrna_alloc(sizeof(char) * (strlen(seq) + 1));

    /* get a vrna_fold_compound with default settings */
```

```

vrna_fold_compound_t *vc = vrna_fold_compound(seq, NULL, VRNA_OPTION_DEFAULT);

/* call MFE function */
double mfe = (double)vrna_mfe(vc, mfe_structure);

printf("%s\n%s (%6.2f)\n", seq, mfe_structure, mfe);

/* rescale parameters for Boltzmann factors */
vrna_exp_params_rescale(vc, &mfe);

/* call PF function */
FLT_OR_DBL en = vrna_pf(vc, prob_string);

/* print probability string and free energy of ensemble */
printf("%s (%6.2f)\n", prob_string, en);

/* compute centroid structure */
double dist;
char *cent = vrna_centroid(vc, &dist);

/* print centroid structure, its free energy and mean distance to the ensemble */
printf("%s (%6.2f d=%6.2f)\n", cent, vrna_eval_structure(vc, cent), dist);

/* free centroid structure */
free(cent);

/* free pseudo dot-bracket probability string */
free(prob_string);

/* free mfe structure */
free(mfe_structure);

/* free memory occupied by vrna_fold_compound */
vrna_fold_compound_free(vc);

return EXIT_SUCCESS;
}

```

See also

examples/example1.c in the source code tarball

8.1.6 Deprecated Examples

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "utils.h"
#include "fold_vars.h"
#include "fold.h"
#include "part_func.h"
#include "inverse.h"
#include "RNAstruct.h"
#include "treedist.h"
#include "stringdist.h"
#include "profiledist.h"

void
main()
{
    char *seq1 = "CGCAGGGAUACCCGCG", *seq2 = "GCGCCCAUAGGGACGC",
        *struct1, *struct2, *xstruc;
    float e1, e2, tree_dist, string_dist, profile_dist, kT;
    Tree *T1, *T2;
    swString *S1, *S2;
    float *pf1, *pf2;
    FLT_OR_DBL *bppm;

    /* fold at 30C instead of the default 37C */
    temperature = 30.; /* must be set *before* initializing */

    /* allocate memory for structure and fold */
    struct1 = (char *)space(sizeof(char) * (strlen(seq1) + 1));
    e1 = fold(seq1, struct1);

    struct2 = (char *)space(sizeof(char) * (strlen(seq2) + 1));
    e2 = fold(seq2, struct2);

    free_arrays(); /* free arrays used in fold() */

    /* produce tree and string representations for comparison */
    xstruc = expand_Full(struct1);
    T1 = make_tree(xstruc);
    S1 = Make_swString(xstruc);

```



```

free(xstruc);

xstruc = expand_Full(struct2);
T2      = make_tree(xstruc);
S2      = Make_swString(xstruc);
free(xstruc);

/* calculate tree edit distance and aligned structures with gaps */
edit_backtrack = 1;
tree_dist     = tree_edit_distance(T1, T2);
free_tree(T1);
free_tree(T2);
unexpand_aligned_F(aligned_line);
printf("%s\n%s  %3.2f\n", aligned_line[0], aligned_line[1], tree_dist);

/* same thing using string edit (alignment) distance */
string_dist = string_edit_distance(S1, S2);
free(S1);
free(S2);
printf("%s mfe=%5.2f\n%s mfe=%5.2f dist=%3.2f\n",
       aligned_line[0], e1, aligned_line[1], e2, string_dist);

/* for longer sequences one should also set a scaling factor for
 * partition function folding, e.g: */
kT      = (temperature + 273.15) * 1.98717 / 1000.; /* kT in kcal/mol */
pf_scale = exp(-e1 / kT / strlen(seq1));

/* calculate partition function and base pair probabilities */
e1 = pf_fold(seq1, struct1);
/* get the base pair probability matrix for the previous run of pf_fold() */
bppm = export_bppm();
pf1   = Make_bp_profile_bppm(bppm, strlen(seq1));

e2 = pf_fold(seq2, struct2);
/* get the base pair probability matrix for the previous run of pf_fold() */
bppm = export_bppm();
pf2   = Make_bp_profile_bppm(bppm, strlen(seq2));

free_pf_arrays(); /* free space allocated for pf_fold() */

profile_dist = profile_edit_distance(pf1, pf2);
printf("%s free energy=%5.2f\n%s free energy=%5.2f dist=%3.2f\n",
       aligned_line[0], e1, aligned_line[1], e2, profile_dist);

free_profile(pf1);
free_profile(pf2);
}

```

See also

examples/example_old.c in the source code tarball

8.2 Perl5 Examples

Hello World Examples

Using the flat interface

- MFE prediction


```

use RNA;

# The RNA sequence
my $seq = "GAGUAGUGGAACCGGCCUAUGUUUGUGACUCGACAGACUAACA";

# compute minimum free energy (MFE) and corresponding structure
my ($ss, $mfe) = RNA::fold($seq);

# print output
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;

```
- comparative MFE prediction for sequence alignments


```

use RNA;

# The RNA sequence alignment
my @sequences = (
    "CUGCCUCACAAACGUUUGGCCUCAGUUACCCGUAUGUAGUGAGGGU",
    "CUGCCUCACAAAUUUGGCCUCAGUUACUCAUAGAUGUAGUGAGGGU",
    "---CUCGACACCACU---GCCUCGGUACCAUCGGUGCAGUGCGGGU"
);

# compute the consensus sequence

```

```
my $cons = RNA::consensus(\@sequences);

# predict Minum Free Energy and corresponding secondary structure
my ($ss, $mfe) = RNA::alifold(\@sequences);

# print output
printf "%s\n%s [ %6.2f ]\n", $cons, $ss, $mfe;
```

Using the object oriented interface

- MFE prediction

```
#!/usr/bin/perl

use warnings;
use strict;

use RNA;

my $seq1 = "CGCAGGGAUACCCGCG";

# create new fold_compound object
my $fc = new RNA::fold_compound($seq1);

# compute minimum free energy (mfe) and corresponding structure
my ($ss, $mfe) = $fc->mfe();

# print output
printf "%s [ %6.2f ]\n", $ss, $mfe;
```

Changing the Model Settings

Using the object oriented interface

- MFE prediction at different temperature and dangle model

```
use RNA;

# The RNA sequence
my $seq = "GAGUAGUGGAACCAGGCUAUGUUUGACUCGACAGACUAACA";

# create a new model details structure
my $md = new RNA::md();

# change temperature and dangle model
$md->{temperature} = 20.0; # 20 Deg Celcius
$md->{dangles} = 1; # Dangle Model 1

# create a fold compound
my $fc = new RNA::fold_compound($seq, $md);

# predict Minum Free Energy and corresponding secondary structure
my ($ss, $mfe) = $fc->mfe();

# print sequence, structure and MFE
printf "%s\n%s [ %6.2f ]\n", $seq, $ss, $mfe;
```

8.3 Python Examples

MFE Prediction (flat interface)

```
import RNA

# The RNA sequence
seq = "GAGUAGUGGAACCAGGCUAUGUUUGACUCGACAGACUAACA"

# compute minimum free energy (MFE) and corresponding structure
(ss, mfe) = RNA.fold(seq)

# print output
print("{}\n{} [ {:.2f} ]".format(seq, ss, mfe))
```

MFE Prediction (object oriented interface)

```
import RNA;

sequence = "CGCAGGGGAUACCCGCG"

# create new fold_compound object
fc = RNA.fold_compound(sequence)

# compute minimum free energy (mfe) and corresponding structure
(ss, mfe) = fc.mfe()

# print output
print("{} [ {:.2f} ]".format(ss, mfe))
```

Suboptimal Structure Prediction

```
import RNA

sequence = "GGGGAAAACCCC"

# Set global switch for unique ML decomposition
RNA.cvar.uniq_ML = 1

subopt_data = { 'counter' : 1, 'sequence' : sequence }

# Print a subopt result as FASTA record
def print_subopt_result(structure, energy, data):
    if not structure == None:
        print(">subopt {}:d".format(data['counter']))
        print("{}\n{} [ {:.2f} ]".format(data['sequence'], structure, energy))
        # increase structure counter
        data['counter'] = data['counter'] + 1

# Create a 'fold_compound' for our sequence
a = RNA.fold_compound(sequence)

# Enumerate all structures 500 docal/mol = 5 kcal/mol around
# the MFE and print each structure using the function above
a.subopt_cb(500, print_subopt_result, subopt_data);
```

Boltzmann Sampling (a.k.a. Probabilistic Backtracing)

```
import RNA

sequence =
    "UGGGAAUAGUCUCUCCGAGUCUCGCGGGCGACGGGCGAUCUUCGAAAGUGGAAUCCGUACUUAUACCGCCUGUGCGGACUACUAUCCUGACCACAUAGU"

def store_structure(s, data):
    """
    A simple callback function that stores
    a structure sample into a list
    """
    if s:
        data.append(s)

"""
First we prepare a fold_compound object
"""

# create model details
md = RNA.md()

# activate unique multibranch loop decomposition
md.uniq_ML = 1

# create fold compound object
fc = RNA.fold_compound(sequence, md)

# compute MFE
(ss, mfe) = fc.mfe()

# rescale Boltzmann factors according to MFE
fc.exp_params_rescale(mfe)

# compute partition function to fill DP matrices
fc.pf()

"""
Now we are ready to perform Boltzmann sampling
"""

# 1. backtrace a single sub-structure of length 10
```

```

print("{}".format(fc.pbacktrack5(10)))

# 2. backtrace a single sub-structure of length 50
print("{}".format(fc.pbacktrack5(50)))

# 3. backtrace multiple sub-structures of length 10 at once
for s in fc.pbacktrack5(20, 10):
    print("{}".format(s))

# 4. backtrace multiple sub-structures of length 50 at once
for s in fc.pbacktrack5(100, 50):
    print("{}".format(s))

# 5. backtrace a single structure (full length)
print("{}".format(fc.pbacktrack()))

# 6. backtrace multiple structures at once
for s in fc.pbacktrack(100):
    print("{}".format(s))

# 7. backtrace multiple structures non-redundantly
for s in fc.pbacktrack(100, RNA.PBACKTRACK_NON_REDUNDANT):
    print("{}".format(s))

# 8. backtrace multiple structures non-redundantly (with resume option)
num_samples = 500
iterations = 15
d = None # pbacktrack memory object
s_list = []

for i in range(0, iterations):
    d, ss = fc.pbacktrack(num_samples, d, RNA.PBACKTRACK_NON_REDUNDANT)
    s_list = s_list + list(ss)

for s in s_list:
    print("{}".format(s))

# 9. backtrace multiple sub-structures of length 50 in callback mode
ss = []
i = fc.pbacktrack5(100, 50, store_structure, ss)

for s in ss:
    print("{}".format(s))

# 10. backtrace multiple full-length structures in callback mode
ss = list()
i = fc.pbacktrack(100, store_structure, ss)

for s in ss:
    print("{}".format(s))

# 11. non-redundantly backtrace multiple full-length structures in callback mode
ss = list()
i = fc.pbacktrack(100, store_structure, ss, RNA.PBACKTRACK_NON_REDUNDANT)

for s in ss:
    print("{}".format(s))

# 12. non-redundantly backtrace multiple full length structures
# in callback mode with resume option
ss = []
d = None # pbacktrack memory object

for i in range(0, iterations):
    d, i = fc.pbacktrack(num_samples, store_structure, ss, d, RNA.PBACKTRACK_NON_REDUNDANT)

for s in ss:
    print("{}".format(s))

# 13. backtrace a single substructure from the sequence interval [10:50]
print("{}".format(fc.pbacktrack_sub(10, 50)))

# 14. backtrace multiple substructures from the sequence interval [10:50]
for s in fc.pbacktrack_sub(100, 10, 50):
    print("{}".format(s))

# 15. backtrace multiple substructures from the sequence interval [10:50] non-redundantly
for s in fc.pbacktrack_sub(100, 10, 50, RNA.PBACKTRACK_NON_REDUNDANT):
    print("{}".format(s))

```

RNAfold -p -MEA equivalent

```
#!/usr/bin/python
```

```
#
import RNA

seq = "AUUUCCACUAGAGAAGGUCUAGAGUGUUUGUCGUUUGUCAGAGUCCCUAUUCCAGGUACGAACACGGUGGAUAUGUUCGACGACAGGAUCGGCGCACUA"

# create fold_compound data structure (required for all subsequently applied algorithms)
fc = RNA.fold_compound(seq)

# compute MFE and MFE structure
(mfe_struct, mfe) = fc.mfe()

# rescale Boltzmann factors for partition function computation
fc.exp_params_rescale(mfe)

# compute partition function
(pp, pf) = fc.pf()

# compute centroid structure
(centroid_struct, dist) = fc.centroid()

# compute free energy of centroid structure
centroid_en = fc.eval_structure(centroid_struct)

# compute MEA structure
(MEA_struct, MEA) = fc.MEA()

# compute free energy of MEA structure
MEA_en = fc.eval_structure(MEA_struct)

# print everything like RNAfold -p --MEA
print("{}\n{} ( {:.2f} )".format(seq, mfe_struct, mfe))
print("{} ( {:.2f} )".format(pp, pf))
print("{} ( {:.2f} d={:.2f} )".format(centroid_struct, centroid_en, dist))
print("{} ( {:.2f} MEA={:.2f} )".format(MEA_struct, MEA_en, MEA))
print(" frequency of mfe structure in ensemble {:.2f}; ensemble diversity {:.2f} ".format(fc.pr_structure(mfe_struct), fc.mean_bp_distance()))
```

Fun with Soft Constraints

```
import RNA

seq1 = "CUCGUCGCCUUAUCCAGUGCGGCGCUAGACAUCUAGUUAUCGCCGCAA"

# Turn-off dangles globally
RNA.cvar.dangles = 0

# Data structure that will be passed to our MaximumMatching() callback with two components:
# 1. a 'dummy' fold_compound to evaluate loop energies w/o constraints, 2. a fresh set of energy parameters
mm_data = { 'dummy': RNA.fold_compound(seq1), 'params': RNA.param() }

# Nearest Neighbor Parameter reversal functions
revert_NN = {
    RNA.DECOMP_PAIR_HP: lambda i, j, k, l, f, p: - f.eval_hp_loop(i, j) - 100,
    RNA.DECOMP_PAIR_IL: lambda i, j, k, l, f, p: - f.eval_int_loop(i, j, k, l) - 100,
    RNA.DECOMP_PAIR_ML: lambda i, j, k, l, f, p: - p.MLclosing - p.MLintern[0] - (j - i - k + l - 2) * p.MLbase - 100,
    RNA.DECOMP_ML_ML_STEM: lambda i, j, k, l, f, p: - p.MLintern[0] - (l - k - 1) * p.MLbase,
    RNA.DECOMP_ML_STEM: lambda i, j, k, l, f, p: - p.MLintern[0] - (j - i - k + l) * p.MLbase,
    RNA.DECOMP_ML_ML: lambda i, j, k, l, f, p: - (j - i - k + l) * p.MLbase,
    RNA.DECOMP_ML_ML_ML: lambda i, j, k, l, f, p: 0,
    RNA.DECOMP_ML_UP: lambda i, j, k, l, f, p: - (j - i + 1) * p.MLbase,
    RNA.DECOMP_EXT_STEM: lambda i, j, k, l, f, p: - f.eval_ext_stem(k, l),
    RNA.DECOMP_EXT_EXT: lambda i, j, k, l, f, p: 0,
    RNA.DECOMP_EXT_STEM_EXT: lambda i, j, k, l, f, p: - f.eval_ext_stem(i, k),
    RNA.DECOMP_EXT_EXT_STEM: lambda i, j, k, l, f, p: - f.eval_ext_stem(l, j),
}

# Maximum Matching callback function (will be called by RNAlib in each decomposition step)
def MaximumMatching(i, j, k, l, d, data):
    return revert_NN[d](i, j, k, l, data['dummy'], data['params'])

# Create a 'fold_compound' for our sequence
fc = RNA.fold_compound(seq1)

# Add maximum matching soft-constraints
fc.sc_add_f(MaximumMatching)
fc.sc_add_data(mm_data, None)

# Call MFE algorithm
(s, mm) = fc.mfe()

# print result
print("{}\n{} ( MM: {:.2f} )".format(seq1, s, int(-mm)))
```


Chapter 9

Contributing to the ViennaRNA Package

Contents

- General Remarks
- Reporting Bugs
- Pull Request Process
- Contributors License Agreement (CLA)

General Remarks

The ViennaRNA Package is developed by humans and consequently may contain bugs that prevent proper operation of the implemented algorithms. If you think you have found any of those nasty animals, please help us to improve our software by reporting the bug to us.

The ViennaRNA Package also is open-source software, which means that everybody can have a closer look into our implementations to understand and potentially extend its functionality. If you implemented any novel feature into the ViennaRNA Package that might be of interest to a larger community, please don't hesitate to ask for merging of your code into our official source tree. See the Pull Request Process section below to find information on how to do that.

Please note that we have a code of conduct. Please follow it in all your interactions with this project.

If you wish to contribute to this project, please first discuss any proposed changes with the owners and main developers. You may do that either through making an issue at [our official GitHub presence](#), [by email](#), or any other personal communication with the core developer team.

More importantly, if you wish to contribute any files or software, you need to agree to our ViennaRNA Package Contributors License Agreement (CLA)! Otherwise, your contributions can't be merged into our source tree. See below for further information and the full CLA details.

Reporting Bugs

1. Please make an issue at GitHub or notify us by emailing to rna@tbi.univie.ac.at
2. In your report, include as much information as possible, such that we are able to reproduce it. If possible, find a minimal example that triggers the bug.
3. Include the version number for the ViennaRNA Package you experience the bug with.
4. Include at least some minimal information regarding your operating system (Linux, Mac OS X, Windows, etc.)

Pull Request Process

1. Ensure that you have not checked-in any files that are automatically build!

2. When contributing C source code, follow our code formatting guide lines. You may use the tool `uncrustify` together with our config located in `misc/uncrustify.cfg` to accomplish that.
3. Only expose symbols (functions, variables, etc.) to the libraries interface that are absolutely necessary! Hide all other symbols in the corresponding object file(s) by declaring them as `static`.
4. Use the prefixes `vrna_` for any symbol you add to the API of our library! Preprocessor macros in header files require the prefix in capital letters, i.e. `VRNA_`.
5. Use C-style comments at any place necessary to make sure your implementation can still be understood and followed in the future.
6. Add test cases for any new implementation! The test suite is located in the `tests` directory and is split into tests for the C-library, executable programs, and the individual scripting language interfaces.
7. Run `make check` to ensure that all other test suites still run properly with your applied changes!
8. When contributing via GitHub, make a personal fork of our project and create a separate branch for your changes. Then make a pull request to our `user-contrib` branch. Pull requests to the `master` branch will be rejected to keep its history clean.
9. Pull requests that have been successfully merged into the `user-contrib` branch usually find their way into the next release of the ViennaRNA Package. However, please note that the core developers may decide to include your changes in a later version.

Contributors License Agreement

Thank you for your interest in contributing to the ViennaRNA Package ("We" or "Us").

Before contributing, please note that we adopted a standard Contributors License Agreement (CLA) agreement provided by [Project Harmony](#), a community-centered group focused on contributor agreements for free and open source software (FOSS).

This contributor agreement ("Agreement") documents the rights granted by contributors to Us. To make this document effective, please sign it and send it to Us by email to rna@tbi.univie.ac.at.

The respective CLA PDF documents are available in the [doc/CLA directory](#) of the distribution tarball, and online at our [official ViennaRNA Website](#).

Chapter 10

Changelog

Below, you'll find a list of notable changes for each version of the ViennaRNA Package.

Unreleased

Version 2.6.x

Version 2.6.0b

Programs

- Allow for at least as many threads as CPUs are configured if maximum thread number detection fails
- Fix alignment input parsing in `refold.pl`
- Allow for NaCl concentration changes in most executable programs (default 1.021M)
- Add `RNAexplorer` program to the distribution

Library

- Add dynamic array data structure utilities
- Add new soft constraints multi-callback dispatcher
- Add m6A parameters via soft constraints callback mechanism
- Add Pseudouridine-A parameters via soft constraints callback
- Add Dihydrouridine adjustments via soft constraints callback
- Fix potential problems in `free_dp_matrices()` of `LPfold.c`
- Add inosine-U and inosine-C parameters via soft constraints callback
- Add string data structure utilities
- Add arbitrary modified base support (`vrna_sc_mod()`) via soft constraints mechanism and JSON input data
- Add 7DA modification support via soft constraints
- Add Purine (nebularine) modification support
- Refactor function typedefs to make them actual function pointer typedefs
- SWIG: Fix Perl 5 wrapper for `vrna_ud_prob_get()`
- Fix z-score initialization in `vrna_Lfoldz()` and `vrna_mfe_window_zscore_cb()`
- Fix Python 3 wrapper suffix issue
- Fix file close issue in `vrna_file_commands_read()`

Package

- Update dlib to version 19.24
- Adapt Debian dependencies
- Fix compilation issues with RNAforester
- Fix autoconf requirement checks when SVM support is deactivated and swig is missing
- Add `auto` parameters for `-flto` compile/link flags
- Require C++17 due to dependencies to compile `DLIB`

Version 2.5.x

Version 2.5.1 (Release date: 2022-06-02)

Programs

- Refactor `ct2db` program to allow for pseudoknots in output structure

Library

- API: Fix MEA computation for G-quadruplex predictions
- API: Fix memory leak in hard constraints container
- API: Fix RNApuzzler edge-case that resulted in segmentation faults
- API: Fix invalid memory access in `vrna_strjoin()`
- API: Revisit generic soft constraints for sliding-window base pair probability computations
- API: Enable to overwrite automatic unpaired probability determination in MEA computation
- API: Add `#VRNA_PLIST_TYPE_UNPAIRED` and `#VRNA_PLIST_TYPE_TRIPLE` identifiers for `vrna_ep_t`
- API: Add `vrna_init_rand_seed()` to initialize RNG with seed
- API: Add `vrna_zsc_compute_raw()` to obtain mean and sd for Z-score computation
- API: Add `vrna_file_connect_read_record()` function to parse connectivity table (`*.ct`) files
- API: Add `vrna_strtrim()` function
- API: Update sanity checks for input in `vrna_pbacktrack_sub*()`
- API: Allow for pseudo-knots in `vrna_db_from_ptable()`
- API: Do not use `min_loop_size = 0` for multi strand interaction prediction
- API: Remove unnecessary uses of `min_loop_size` at multiple locations
- API: Deprecate cutpoint member of `vrna_fold_compound_t` and prepare for 5'/3' encoding
- API: Refactor sequence addition/preparation for `vrna_fold_compound_t`
- DOC: Update documentation
- SWIG: Add simple dot-plot file wrapper `plot_dp_EPS()`
- SWIG: Add `sequence`, `sequence_encoding` and `sequence_encoding2` attributes to `fold_↔ compound` objects
- SWIG: Fix RNG wrapping and initialize RNG upon module load and update associated functions

- SWIG: Add more access to member variable arrays for various objects used throughout the library
- SWIG: Add memory efficient wrapper for dynamically allocated arrays and matrices
- SWIG: Shadow pair table data structure for efficient interactions between C and target languages
- SWIG: Expose hard constraints members in `fold_compound` objects
- SWIG: Add `exp_E_ext_stem()` method (`vrna_exp_E_ext_stem()`) to `fold_compound` objects
- SWIG: Expose DP matrices within `fold_compound` objects
- SWIG: Fix memory leak in wrapper for `vrna_db_from_ptable()`

Package

- Update dlib to version 19.23
- DOC: Update doxygen.conf for version 1.9.2
- AUTOCONF: Factor-out Naview layout algorithm to allow for deactivating the Naview layout algorithm at configure-time
- AUTOCONF: Make LaTeX checks more portable and update LaTeX package checks
- AUTOCONF: Check whether we can build the swig interface when SVM support is deactivated
- AUTOCONF: Fix condition check for CLA build

Version 2.5.0 (Release date: 2021-11-08)

Programs

- Add `RNAmultifold` program to compute secondary structures for multiple interacting RNAs
- Add multistrand capabilities to `RNAeval`
- Add multistrand capabilities to `RNAsubopt`
- Replace `RNAcofold` with a wrapper to `RNAmultifold`
- Fix computation of BB homodimer base pair probabilities in `RNAcofold`

Library

- API: Fix use of undefined values in deprecated function `PS_dot_plot()`
- API: Fix probability computations for unstructured domains within multibranch loops
- API: Fix index error in ensemble defect computations
- API: Fix hard constraints behavior on non-specific base pairing
- API: Fix segmentation fault for short input sequences in `vrna_hx_from_ptable()`
- API: Fix memory leak in static `rna_layout()` function
- API: Fix corner-case in covariance score computation on sequence alignments that determines which alignment columns may pair and which don't
- API: Add MFE computations for multiple interacting strands
- API: Add partition function computations for multiple interacting strands
- API: Add base pair probability computations for multiple interacting strands
- API: Add suboptimal structure prediction for multiple interacting strands

- API: Add multistrand capabilities to `vrna_eval*()` functions
- API: Add new function `vrna_equilibrium_conc()` for concentration dependency computations of multiple interacting strands with `dlib` backend
- API: Add `vrna_equilibrium_constants()` function to obtain equilibrium constants for different complexes of multiple interacting strands
- API: Add function `vrna_pf_add()` to add ensemble free energies of two ensembles
- API: Add function `vrna_pf_substrands()` to get ensemble free energies for complexes up to a specific number of interacting strands
- API: Add function `vrna_n_multichoose_k()` to obtain a list of k-combinations with repetition
- API: Add `vrna_cstr_discard()` function to allow for discarding char streams prior to flushing
- API: Add `vrna_bp_distance_pt()` function to allow for base pair distance computation with pseudoknots
- API: Add functions `vrna_pbacktrack_sub*()` to allow for stochastic backtracing within arbitrary sequence intervals
- API: Add functions `vrna_boustrophedon()` and `vrna_boustrophedon_pos()` to generate lists of or obtain values from sequences of Boustrophedon distributed integer numbers
- API: Add `vrna_pscore()` and `vrna_pscore_freq()` functions to obtain covariance score for particular alignment columns
- API: Rewrite Zuker suboptimals implementation
- API: Remove old cofold implementations
- API: Make `type` attribute of `vrna_mx_mfe_t` and `vrna_mx_pf_t` a constant
- API: Guard more functions in `utils/structure_utils.c` against `NULL` input
- API: Rename `vrna_E_ext_loop()` to `vrna_eval_ext_stem()`
- API: Use `v3` typedefs in dot-plot function declarations
- SWIG: Fix Python 3 file handle as optional argument in `eval*` functions and methods
- SWIG: Add wrapper for `vrna_pf_add()`
- SWIG: Add wrapper for `vrna_hx_from_ptable()`
- SWIG: Add wrapper for `vrna_db_from_probs()`

Package

- Update `libsvm` to version 3.25
- Make Python 3.x the default Python for the scripting language interfaces
- Add Python3 capability for Mac OS X installer builds
- TESTS: Create TAP driver output for all unit tests (library, executables, SWIG interfaces)
- Remove compile-time switch to deactivate Boustrophedon backtracing scheme (this is the status-quo now)
- Add Contributors License Agreement (CLA) to the Package in `doc/CLA/`

Version 2.4.x

Version 2.4.18 (Release date: 2021-04-22)

Programs

- Fix and refactor `RNApkplex` program
- Fix occasional backtracing errors in `RNALalifold`
- Restrict available dangling end models in `RNALalifold` to 0 and 2
- Prevent segmentation faults upon bogus input data in `RNAfold`, `RNAalifold`, `RNAcofold`, `RNAheat`, and `RNAeval`
- Free MFE DP matrices in `RNAsubopt` Boltzmann sampling when not required anymore

Library

- API: Add `vrna_abstract_shapes()` and `vrna_abstract_shapes_pt()` functions to convert secondary structures into their respective abstract shape notation ala Giegerich et al. 2004
- API: Add functions `vrna_seq_reverse()` and `vrna_DNA_complement()` to create reverse complements of a sequence
- API: Add more soft constraint handling to comparative structure prediction
- API: Add generic soft constraints for sliding window comparative MFE backtracing
- API: Add `vrna_ensemble_defect_pt()` that accepts pair table input instead of dot-bracket string to allow for non-nested reference structures
- API: Add failure/success return values to generic soft constraints application functions
- API: Refactor `RNApkplex` implementation by better using constraints framework and moving out many parts from `RNApkplex.c` into `RNAlib` as separate re-usable functions
- API: Fix energy contributions used in `RNApkplex` implementations
- API: Fix energy evaluation for cofolding with dangle model 1
- API: Fix wrong arithmetic usage for PF variant of combined generic and simple soft constraints applied to external loops
- API: Fix memory size in `vrna_fold_compound_t` initialization
- API: Fix bogus memory access for comparative prediction when preparing hard constraints
- API: Fix wrong index usage in hard constraints for comparative base pair probability computations of internal loops
- API: Fix G-Quadruplex contributions as part of multibranch loops in single sequence base pair probability computations
- API: Fix multibranch loop MFE decomposition step for multiple strand cases
- API: Fix external loop generic hard constraint index updating for partition function computations
- API: Fix memory allocation for auxiliary grammar data structure
- API: Fix incorporation of auxiliary grammar contrib for closing pairs in sliding-window MFE computation
- API: Fix DP matrix initialization in sliding window MFE computations (fixes occasional backtracing issues in comparative sliding-window MFE computations)
- API: Make `vrna_sc_t.type` attribute a constant
- API: Remove upper-triangular hard constraint matrix in favor of full matrix

- API: Always ensure sane base pair span settings after `vrna_fold_compound_prepare()`
- API: Return INF on predictions of `vrna_mfe_dimer()` that fail due to unsatisfiable constraints
- API: Rename internally used hard and soft constraints API symbols
- API: Fix header file inclusions to prevent `#include` cycles
- SWIG: Add wrapper for `vrna_file_fasta_read_record()`
- SWIG: Fix memory leak in wrapper for `vrna_probs_window()`
- SWIG: Refactor and therefore fix soft constraint binding functions for use in comparative structure predictions
- SWIG: Fix typo that prevented properly wrapping `vrna_params_load_RNA_Andronesescu2007()`
- SWIG: Unify wrappers for `vrna_ptable()` and `vrna_ptable_from_string()`

Package

- REFMAN: Refactored structure annotation documentation
- REFMAN: Update Mac OS X install section
- Replace DEF placeholders in energy parameter files with their value of -50
- Update `RNAlocmin` subpackage to properly compile with more stringent C++ compilers
- Update `RNAforester` subpackage to properly compile with more stringent C++ compilers
- Update autotools framework, e.g. checks for pthreads
- Update universal binary build instructions for Mac OS X builds to enable ARM compilation for M1 CPUs

Version 2.4.17 (Release date: 2020-11-25)

Programs

- Fix `RNAup -b` mode with shorter sequence first
- Add `--backtrack-global` option to `RNAfold` (currently only available for `dangles == 2 | 0`)
- Add `--zscore-pre-filter` and `--zscore-report-subsumed` options to `RNAfold`

Library

- API: Fix multiloop backtracing with soft constraints for unpaired positions in `vrna_subopt()` and `vrna_subopt_cb()`
- API: Fix parameter parse in `vrna_params_load_from_string()`
- API: Add `vrna_heat_capacity()` and `vrna_head_capacity_cb()` functions to `RNAlib`
- API: Add backtracing function `vrna_backtrack_window()` for global MFE structure to sliding-window predictions
- API: Add SVG support for `RNApuzzler` structure layouts
- API: Make `vrna_md_t` argument to `vrna_fold_compound()` a constant pointer
- API: Remove missing symbols from header file `ViennaRNA/params/default.h`
- API: Refactor z-score threshold filter handling for sliding-window MFE prediction
- SWIG: Fix typo in interface functions to load DNA parameters
- SWIG: Add python-3.9 autoconf checks

- SWIG: Add `vrna_head_capacity*()` wrappers
- SWIG: Add access to raw energy parameters
- SWIG: Add `alias` and `pair` attribute to objects of type `md`
- SWIG: Add out/varout typemaps for 2-dimensional int-like arrays
- SWIG: Add all data fields to objects of type 'param' and 'exp_param'

Package

- Fix Debian and Windows installer files

Version 2.4.16 (Release date: 2020-10-09)

Programs

- Fix backtracing errors in `RNAalifold` for alignments with more than 32768 columns
- Fix backtracing errors in `RNAalifold` and `RNAalifold` for rare cases when two alignment columns may pair due to covariance score threshold but still yield infinite energies due to energy model
- Refactored manpages/help options for `RNAplfold`, `RNAplot`, `RNApvmim`, `RNAsubopt`, and `RNAup`

Library

- API: Fix undefined behavior due to short int overflows when accessing alignment lengths with alignments larger than 32768 columns. This fixes occasional backtracing errors in `RNAalifold` and `vrna_mfe_window()`
- API: Fix adding pscore to base pairs that yield INF energy in comparative global and local MFE prediction
- API: Add `vrna_convert_kcal_to_dcal()` and vice-versa function for safely converting integer to float representations of energy values
- SWIG: Add a reasonable Python interface for objects of type `vrna_path_t`
- SWIG: Add a wrapper for `vrna_seq_encode()`

Package

- Move `units.h` include file to `ViennaRNA/Utils/units.h`

Version 2.4.15 (Release date: 2020-08-18)

Programs

- Fix compilation of `Kinfold` with GCC 10
- Add `--en-only` flag to `RNAsubopt` to allow for sorting by energy only
- Prevent `RNAcofold` to process input with more than two strands
- Add cutpoint marker to dot-plots created with `RNAcofold -a`
- Update `Kinfold` to version 1.4

Library

- API: Fix removal of strand delimiter in `vrna_plot_dp_PS_list()`
- API: Fix `vrna_enumerate_necklaces()`
- API: Fix bogus backtracing for co-folded structures in `vrna_subopt()` and `vrna_subopt_cb()`
- API: Fix storing co-folded structures for sorted output in `vrna_subopt()`
- API: Fix multibranch loop component hard constraints for multi-strand cases
- API: Prevent adding internal loop energy contributions to enclosed parts with `energy=INF`
- API: Adapt `vrna_db_pack()` / `vrna_db_unpack()` functions to produce comparable strings
- API: Add sorting modes `VRNA_UNSORTED`, `VRNA_SORT_BY_ENERGY_LEXICOGRAPHIC_ASC`, and `VRNA_SORT_BY_ENERGY_ASC` to `vrna_subopt()`
- API: Add `vrna_strjoin()` function
- API: Add missing case to external loop hard constraints
- API: Make hard constraints strand-aware
- SWIG: Fix invalid memory access when using `MEA_from_plist()` in Perl 5 or Python
- SWIG: Enable keyword argument features in Python interface of constructors for `fold_compound`, `md`, `move`, `param`, and `exp_param` objects
- SWIG: Enable autodoc feature for Python interface of constructors for `fold_compound`, `md`, and `move` objects
- SWIG: Enable `toString` conversion for Python interface for objects of type `fold_compound`, `md`, `move`, `params`, `exp_params`, and `subopt_solution`
- SWIG: Add (read-only) attributes `type`, `length`, `strands`, `params`, and `exp_params` to objects of type `fold_compound`
- SWIG: Make attributes of objects of type `param` and `exp_param` read-only
- Add array of strand nicks to EPS dot plot files instead of single cutpoint
- Draw separator line for each strand nick in EPS dot-plots
- Update `libsvm` to version 3.24

Package

- Disable Link-Time-Optimization (LTO) for third-party programs linking against `RNAlib` using `pkg-config`
- TESTS: Fix results dir path for out-of-tree builds
- TESTS: Set default timeout for library tests to 20s

Version 2.4.14 (Release date: 2019-08-13)

Programs

- Fix `RNApvmin` pertubation vector computation
- Add non-redundant sampling option to `RNApvmin`
- Add `RNA DOS` program to compute density of states
- Add `-P DNA` convenience command line parameter to most programs to quickly load DNA parameters without any input file
- MAN: Add example section to man-page of `RNAalifold`

Library

- API: Fix memory leak in `vrna_path_gradient()`
- API: Fix release of memory for `vrna_sequence_remove_all()`
- API: Fix soft-constraints application in `vrna_sc_minimize_perturbation()` that prevented proper computation of the perturbation vector
- API: Add 5' and 3' neighbor nucleotide encoding arrays and name string to `vrna_seq_t`
- API: Add new data structure for multiple sequence alignments
- API: Add `vrna_sequence_order_update()` function
- API: Add non-redundant sampling mode to `vrna_sc_minimize_perturbation()` through passing negative sample-sizes
- API: Add v3.0 API functions for maximum expected accuracy (MEA) computation
- API: Include energy parameter sets into RNAlib and provide functions to load them at runtime
- API: Prepare sequence data in `vrna_fold_compound_t` with `vrna_sequence_add()`
- API: Use `vrna_pbacktrack_num()` instead of `vrna_pbacktrack()` in `vrna_sc_minimize_perturbation()` to speed-up sample generation
- Reduce use of global variable `cut_point` in RNAlib
- SWIG: Use `importlib` in favor of `imp` to determine Python 3 tag extension
- SWIG: Update various wrapper functions
- SWIG: Add wrappers for MEA computation with `vrna_MEA()` and `vrna_MEA_from_plist`
- SWIG: Add wrappers for `vrna_pr_structure()` and `vrna_pr_energy()`

Package

- REFMAN: Fix LaTeX code in `units.h` that prevented proper compilation with `pdflatex`
- Add an R script to create 2D landscape plots from RNA2Dfold output
- Add `gengetopt` to configure-time requirements to build man-pages
- Add new energy parameter file `rna_misc_special_hairpins.par` with additional UV-melting derived parameters for Tri- and Tetra-loops
- Update RNA Tutorial
- Colorize final configure script message
- REFMAN: Always use `pdflatex` to compile reference manual and tutorial
- EXAMPLES: Add Python script that performs computations equivalent to `RNAfold -p --MEA`

Version 2.4.13 (Release date: 2019-05-30)

Programs

- Fix centroid structure prediction for `RNAcofold`
- Fix `--noLP` option for `RNAIfold`

Library

- API: Refactor and fix collision handling in `vrna_hash_table_t`
- API: Fix one access using wrong index for odd dangles in `loops/external.c`
- API: Add two missing `MLbase` contributions for MFE prediction in `loops/multibranch.c`
- API: Refactor multiloop MFE backtracking for odd dangles
- API: Add function `vrna_backtrack5()` to allow for MFE backtracking of sub-sequences starting at the 5'-end
- API: Reduce usage of global macro `TURN` by replacing it with `min_loop_size` field of `vrna_md_t`
- API: Add functions `vrna_path_direct()` and `vrna_path_direct_ub()` that may also return move lists instead of dot-bracket lists
- API: Add functions `vrna_pt_pk_remove()` and `vrna_db_pk_remove()` that remove pseudoknots from an input structure
- API: Fix invalid memory access for lonely pair mode (`--noLP`) in comparative sliding-window MFE prediction
- SWIG: Fix access to global variable `pf_smooth` and `pf_smooth` attribute in `model_details` object
- SWIG: Fix Python reference counting for `Py_None` in `interfaces/findpath.i` wrapper
- SWIG: Refactor reference counting for all Python2 and Python3 wrappers
- REFMAN: Larger updates and restructuring of reference manual

Package

- Install example scripts and source code files, e.g. to `$prefix/share/ViennaRNA/examples`
- Properly pass `GSL`, `PTHREADS`, and `MPFR` flags to sub-projects
- Fix `RNApuzzler` header file installation
- SWIG: Include Python 3.7 and 3.8 in list of autoconf-probed python interpreters
- SWIG: Fix wrapper building for `swig >= 4.0.0`

Version 2.4.12 (Release date: 2019-04-16)

Programs

- Add non-redundant stochastic backtracing option for `RNAalifold`
- Add `--noDP` option to suppress dot-plot output in `RNAfold` and `RNAalifold`
- Add `RNApuzzler` (4) and `RNAturtle` (3) secondary structure layout algorithm options to `RNAfold` and `RNAplot`
- Update help/man page of `RNAfold`
- Allow for multiple input files and parallel input processing in `RNAheat`

Library

- API: Fix declaration of `vrna_move_apply_db()`
- API: Fix `vrna_path()` lexicographical ordering in gradient walks
- API: Enable non-redundant stochastic backtracing for comparative structure prediction
- API: Enable stochastic backtracing for circular comparative structure prediction
- API: Enable stochastic backtracing of subsequences (5' prefixes) for comparative structure prediction
- API: Add `pf_smooth` attribute to `vrna_md_t` data structure to allow for disabling Boltzmann factor energy smoothing
- API: Add functions to allow for resuming non-redundant stochastic backtracing
- API: Add functions to retrieve multiple stochastically backtraced structures (list and callback variants)
- API: Add `vrna_positional_entropy` to compute vector of positional entropies
- API: Add `RNApuzzler` and `RNAturtle` secondary structure layout algorithm (Wiegreffe et al. 2018)
- API: Add v3.0 API for secondary structure layout/coordinate algorithms
- API: Add more helper/utility functions for `vrna_move_t` data structures
- API: Add callback-based neighborhood update function for (subsequent) `vrna_move_t` application
- API: Add abstract heap data structure available as `<ViennaRNA/datastructures/heap.h>`
- API: Refactor and speed-up gradient walk implementation available as `vrna_path_gradient()`
- API: Substitute `vrna_file_PS_aln_sub()` alignment plot function by `vrna_file_PS_aln_slice()` that actually slices out a sub-alignment
- API: Rename `vrna_annotate_covar_struct()` to `vrna_annotate_covar_db()` and add new function `vrna_annotate_covar_db_extended()` to support more bracket types
- API: Calling `vrna_params_reset()` now implies a call to `vrna_exp_params_reset()` as well
- API: Move landscape implementations into separate directory, thus headers should be included as `<ViennaRNA/landscape/move.h>`, `<ViennaRNA/landscape/neighbor.h>`, etc.
- Ensure proper rescaling of energy parameters upon temperature changes
- Refactor soft constraints implementation in stochastic backtracing
- SWIG: Wrap all non-redundant stochastic backtracing functions to scripting language interface(s)
- SWIG: Refactor stochastic backtracing interface(s)
- SWIG: Add proper constructor for objects of type `vrna_ep_t`
- SWIG: Sanitize alignment plot function interface(s)

Package

- Update Ubuntu/Debian and OpenSUSE build instructions
- Reduce intra-package dependency on non-v3.0 API

Version 2.4.11 (Release date: 2018-12-17)

Programs

- Add `--commands` option to `RNAsubopt`
- Add non-redundant Boltzmann sampling mode for `RNAsubopt`

Library

- API: Fix wrong access to base pair soft constraints in equilibrium probability computations
- API: Fix behavior of `vrna_nucleotide_encode()` with lowercase characters in sequence
- API: Fix behavior of `encode_char()` with lowercase characters in sequence
- API: Fix forbidden GU pairs behavior in pscore computation for comparative folding
- API: Fix potential errors due to uninitialized `next` pointers in `vrna_move_t` of `vrna_eval_move_t` ↔ `shift_pt`
- API: Add AVX 512 optimized version of MFE multibranch loop decomposition
- API: Add functions for CPU SIMD feature detection
- API: Add dispatcher to automatically delegate exterior-/multibranch loop MFE decomposition to supported SIMD optimized implementation
- API: Add function `vrna_dist_mountain()` to compute mountain distance between two structures
- API: Add function `vrna_ensemble_defect()` to compute ensemble defect given a target structure
- API: Add non-redundant Boltzmann sampling
- API: Change behavior of `vrna_cstr_free()` and `vrna_cstr_close()` to always flush output before unregistering the stream
- SWIG: Add interface for `vrna_loopidx_from_ptable()`

Package

- Activate compilation for compile-time supported SIMD optimized implementations by default
- Replace `--enable-sse` configure script option with `--disable-simd`

Version 2.4.10 (Release date: 2018-09-26)

Programs

- Fix wrong output filename for binary opening energies in `RNAplfold`
- Enable G-Quadruplex support for partition function computation in `RNAalifold`

Library

- Fix broken SSE4.1 support for multibranch loop MFE computation that resulted in increased run times
- Fix redundant output issue in subopt backtracking with unusually high delta energies ($\geq \text{INF}$)
- Restore default behavior of '|' symbol in dot-bracket hard constraint strings that got lost with version 2.2.0
- Add faster (cache-optimized) version of Nussinov Maximum Matching algorithm
- Change default linker- and loop length computations for G-Quadruplex predictions in comparative prediction modes
- Add hard constraints warning for base pairs that violate the `min_loop_size` of the model
- Update `libsvm` to version 3.23
- API: Add functions to set auxiliary grammar extension rules
- API: Replace upper-triangular hard constraints matrix with full matrix for cache-optimized access
- API: Add G-Quadruplex prediction support for comparative partition function
- API: Remove `VRNA_GQUAD_MISMATCH_PENALTY` and `VRNA_GQUAD_MISMATCH_NUM_ALI` macros
- SWIG: Fix invalid memory access in `subopt()` method of `fold_compound` object when writing to file
- SWIG: Add wrapper for Nussinov Maximum Matching algorithm

Package

- Add `-ftree-vectorize` compile flag by default if supported

Version 2.4.9 (Release date: 2018-07-11)

Programs

- Fix interactive mode behavior for multiple sequence alignment input in `RNAalifold`, `RNAalifold`
- Allow for Stockholm formatted multiple sequence alignment input in `RNAeval` and `RNAplot`
- Allow for multiple input files in `RNAeval` and `RNAplot`
- Allow for parallel processing of input batch jobs in `RNAeval` and `RNAplot`
- Add `-g` option to activate G-Quadruplex support in `RNAheat`
- Warn on unsatisfiable hard constraints from dot-bracket string input in `RNAfold`, `RNAcofold`, and `RNAalifold`

Library

- Fix parameter order bug in `vrna_path_findpath*` functions that resulted in too large search widths
- Fix wrong application of base pair soft constraints in partition function computations
- Fix position ruler string in EPS alignment output files
- Fix MFE backtracking errors that might appear under specific hard constrained base pair patterns
- Refrain from reading anything other than `#=GC SS_cons` to retrieve structures when parsing Stockholm 1.0 format
- Complete soft constraints additions to Boltzmann sampling implementation for single sequences
- Allow for disabling alignment wrapping in `vrna_file_PS_aln*` functions
- Do not remove G-Quadruplex annotation from WUSS formatted structure strings upon calls to `vrna_db_↵
from_WUSS`
- Enable G-Quadruplex related average loop energy correction terms in verbose output of `vrna_eval_*` functions
- Speed-up backward compatibility layer for energy evaluation functions that unnecessarily slowed down third-party tools using the old API
- Allow for passing dot-bracket strings with `"&'strand-end identifier to simplevrna_eval_↵
*functions`
- Remove `implicitexit()` calls from global MFE backtracking implementation.

Version 2.4.8 (Release date: 2018-06-23)

Programs

- Fix compilation of `RNAforester` with C++17 standard
- Fix tty input detection in `RNAcofold`
- Fix bad memory access with `RNAcofold -p`

Library

- API: Fix incorrect unpaired probability computations in [vrna_probs_window\(\)](#)
- API: Fix potential out-of-bounds access situations (for circular RNA folding) in eval.c
- API: Fix comparative exterior internal loop partition function computation for circfold
- SWIG: Fix false-positive use of uninitialized value in Python3/file_py3.i

Package

- TESTS: Add tests for special features in RNAalifold
- TESTS: Add test case for RNACofold -p

Version 2.4.7 (Release date: 2018-06-13)

- Allow for parallel processing across multiple input files in RNAfold
- Allow for arbitrary number of input files in RNAalifold
- Allow for parallel processing of input data in RNAalifold
- Allow for arbitrary number of input files in RNACofold
- Allow for parallel processing of input data in RNACofold
- Enable parallel processing in RNAfold, RNACofold, RNAalifold for MS Windows build
- Add centroid and MEA structure computation to RNACofold
- Add configure time check for LTO capabilities of the linker
- Include ligand binding energies in centroid and MEA structure output of RNAfold
- Refactor ct2db program to process multiple structures from single .ct file
- API: Enable processing of comparative fold_compound with vrna_pr_*() functions
- API: Refactor vrna_ostream_t to enable NULL input in [vrna_ostream_provide\(\)](#)
- API: Major refactoring in loop energy evaluations (MFE and PF)
- API: Make vrna_mx_pf_aux_el_t and vrna_mx_pf_aux_ml_s opaque pointers
- API: Make fold_compound field type a const attribute
- API: Refactor MFE post-processing for circular RNAs
- API: Add motif name/id support for unstructured domains
- API: Remove major part of implicit exit() calls in RNAlib
- API: Add implementations of Boyer-Moore-Horspool search algorithm
- API: Add implementations to determine number of rotational symmetry for strings (of objects)
- API: Make vrna_cmd_t an opaque pointer
- API: Move headers for constraints, datastructures, io, loop energy evaluation, energy parameters, plotting, search, and utilities into separate subdirectories (backward compatibility is maintained)
- API: Add hash table data structure
- API: Fix discrepancy between comparative and single sequence -noLP predictions
- API: Add functions to replace 'old API' interface of [RNAstruct.h](#)

- API: Add functions to replace 'old API' interface of [aln_util.h](#)
- API: Add generic soft constraints support to suboptimal structure prediction sensu Wuchty et al.
- SWIG: Refactor callback execution for Python 2 / 3 interface to reduce overhead
- SWIG: Fix configure-time check for Python 3 interface build
- SWIG: Fix Python 3 IO file stream to C FILE * conversion
- Cosmetic changes in final configure notice
- Major changes in source tree structure of the library
- Add autoconf checks for maintainer tools
- Generate C strings from static PostScript files at configure time (for structure- and dot plots)
- REFMAN: Large updates in API documentation and structure of reference manual

Version 2.4.6 (Release date: 2018-04-19)

- Stabilize rounding of free energy output in RNAalifold
- API: Fix potential rounding errors for comparative free energies in eval.c and mfe.c
- API: Fix regression in exterior loop dangling end contributions for comparative base pair probabilities and Boltzmann sampling (introduced with v2.4.4)
- API: Fix regression with hard constrained base pairs for comparative structure prediction (introduced with v2.4.4)
- TESTS: Add basic tests for RNAalifold executable
- TESTS: Ignore 'frequency of MFE structure' in RNAcifold partition function checks

Version 2.4.5 (Release date: 2018-04-17)

- Allow for arbitrary number of input files in RNAfold
- Allow for parallel processing of input data in RNAfold (UNIX only, no Windows support yet)
- Add SHAPE reactivity support through commandline options for RNApIfold
- Fix unstructured domain motif detection in MFE, centroid, and MEA structures computed by RNAfold
- Limit allowed set of commands in command file for RNAcifold to hard and soft constraints
- API: Add functions to compute equilibrium probability of particular secondary structures
- API: Add dynamic string stream data type and associated functions
- API: Add priority-queue like data structure with unordered fill capability and ordered output callback execution
- API: Add functions to detect unstructured domain motifs in MFE, centroid, and MEA structures
- API: Fix bug in sliding-window partition function computation with SHAPE reactivity and Deigan et al. conversion method
- API: Fix application of '<' and '>' constraint symbols in dot-bracket provided constraints (was broken since v2.4.2)
- API: Fix MEA structure computation in the presence of unstructured domains
- API: Stabilize order of probability entries in EPS dot-plot files
- Fix compiler warnings on wrong type of printf() in naview.c

- Define VRNA_VERSION macro as string literal and add macros for major, minor, and patch numbers
- Stabilize parallel make of Mac OS X installer
- Add energy parameter set from Langdon et al. 2018
- Add autoconf checks for POSIX threads compiler/linker support
- SWIG: Fix 'next' is a perl keyword warnings for Perl5 wrapper
- SWIG: Catch errors and throw exceptions whenever scripting language provided callback functions are not applicable or fail
- SWIG: Add keyword arguments and autodoc feature for Python/Python3 wrappers

Version 2.4.4 (Release date: 2018-03-06)

- Change verbose output for soft-constraints derived ligand binding motifs in RNAfold
- Allow for lowercase letters in ct2db input
- Fix bug in interior-like G-Quadruplex MFE computation for single sequences
- Fix autoconf switch to enable deprecation warnings
- Fix bug in eval_int_loop() that prevented propagation of energy evaluation for loops with nick in strands
- Fix several bugs for SHAPE reactivity related comparative partition function computations
- Fix annotation of PostScript output for soft-constraint derived ligand binding motifs in RNAfold
- Fix constraint indices for multibranch loops in unpaired probability computations of LPfold.c
- Fix dangling end contributions in comparative partition function for exterior loops
- API: Add simplified interface for [vrna_pf_dimer\(\)](#)
- API: Move concentraton dependent implementation for co-folding to separate compile unit
- API: Add new API functions for exterior loop evaluations
- API: Add simplified interfaces for energy evaluation with G-Quadruplexes and circular RNAs
- API: Add findpath functions that allow for specification of an upper bound for the saddle point
- Add configure-time linker check for Python3 interface
- Add automatic CPP suggestions for deprecated function substitutes
- Major restructuring and constraints feature additions in loop type dependent energy evaluation functions
- Major restructuring in MFE implementations
- Major restructuring in PF implementations
- Minor fixes in Boltzmann sampling implementation
- SWIG: Fix wrappers for findpath() implementation
- SWIG: Add tons of energy evaluation wrappers
- SWIG: Fix configure-time check of Perl5 interface build capabilities
- SWIG: Wrap functions from walk.c and neighbor.c
- DOC: Add some missing references to manpages of executable programs
- REFMAN: Heavy re-ordering of the RNAlib reference manual

Version 2.4.3 (Release date: 2017-11-14)

- Fix handling of dangling end contribution at sequence boundaries for sliding window base pair probability computations
- Fix handling of base pair hard constraints in sliding-window implementations
- Fix sliding-window pair probability computations with multibranch-loop unpaired constraints
- Fix sliding-window non-specific base pair hard constraint implementation
- Fix probability computation for stochastic backtracking in RNAsubopt `–stochBT_en` output
- Fix regression in comparative structure prediction for circular RNAs
- Fix LDFLAGS for scripting language interfaces in corresponding Makefiles
- Stabilize partition function scaling by always using `sfact` scaling factor from model details
- Add `–pf_scale` commandline parameter to RNAplfold
- Add constraint framework for single sequence circular RNA structure prediction
- Add RNAfold test suite to check for working implementation of constraints for circular RNAs
- Add a brief contribution guideline CONTRIBUTING.md
- Prevent RNAplfold from creating `inf/-inf` output when solution set is empty with particular hard constraints
- Include RNAforester v2.0.1

Version 2.4.2 (Release date: 2017-10-13)

- Fix G-Quadruplex energy corrections in comparative structure energy evaluations
- Fix discrepancy in comparative exterior loop dangling end contribution of eval vs. MFE predictions
- Fix regression in RNAup unstructuredness and interaction energy computations
- Fix sequence length confusions when FASTA input contains carriage returns
- Fix build problems of RNALocmin with older compilers
- Fix sliding-window hard constraints where single nucleotides are prohibited from pairing
- Fix dot-bracket output string length in sliding-window MFE with G-Quadruplexes
- Fix unpaired probability computations for separate individual loop types in LPfold.c
- Fix bad memory access in RNAsubopt with dot-bracket constraint
- Add full WUSS support for `–SS_cons` constraint option in RNAalifold
- Add commandline option to RNALalifold that enables splitting of energy contributions into separate parts
- Add missing hard constraint cases to sliding-window partition function implementation
- Add CSV output option to RNAcofold
- Use the same model details for SCI computations in RNAalifold
- Abort computations in `vrna_eval_structure_v()` if structure has unexpected length
- Use original MSA in all output generated by RNAalifold and RNALalifold
- API: Add new functions to convert dot-bracket like structure annotations
- API: Add various new utility functions for alignment handling and comparative structure predictions
- API: Add function `vrna_strsplit()` to split string into tokens

- API: Do not convert sequences of input MSA to uppercase letters in [vrna_file_msa_read_record\(\)](#)
- API: Rename `vrna_annotate_bp_covar()` and `vrna_annotate_pr_covar()`
- API: Add new noLP neighbor generation
- SWIG: Add wrapper for functions in `file_utils_msa.h`
- SWIG: Add wrappers for [vrna_pbacktrack\(\)](#) and [vrna_pbacktrack5\(\)](#)
- SWIG: Add [vrna_db_to_element_string\(\)](#) to scripting language interface
- REFMAN: Fix formula to image conversion in HTML output

Version 2.4.1 (Release date: 2017-08-23)

- Fix memory leak in `fold_compound` methods of SWIG interface
- Fix memory leaks in double `**` returning functions of SWIG Perl5 interface
- Fix memory leak in `vrna_ep_t` to-string() function of SWIG interface
- Regression: Fix reverting `pf_scale` to defaults after [vrna_exp_params_rescale\(\)](#)
- Regression: Fix homo-dimer partition function computation in RNAcofold
- Add unit tests for RNAcofold executable
- Add SHAPE reactivity support to RNAcofold
- Add SHAPE reactivity support to RNALaliFold

Version 2.4.0 (Release date: 2017-08-01)

- Bump libsvm to version 3.22
- Print G-Quadruplex corrections in verbose mode of RNAeval
- Change behavior of RNAfold `-outfile` option to something more predictable
- Unify `max_bp_span` usage among sliding window prediction algorithms: RNAplfold, RNALfold, and RNALali-fold now consider any base pair (i,j) with $(j - i + 1) \leq \text{max_bp_span}$
- Add SHAPE reactivity data support to RNALfold
- Add commands-file support for RNALfold, RNAplfold (hard/soft constraints)
- Add RNALocmin - Calculate local minima from structures via gradient walks
- Add RNA Bioinformatics tutorial (PDF version)
- Add hard constraints to sliding-window MFE implementations (RNALfold, RNALaliFold)
- Add hard constraints to sliding-window PF implementations (RNAplfold)
- Add soft constraints to sliding-window MFE implementation for single sequences (RNALfold)
- Add soft constraints to sliding-window PF implementations (RNAplfold)
- Add SWIG interfaces for sliding-window MFE/PF implementations
- Add proper SWIG interface for alignment and structure plotting functions
- Add proper SWIG interface for `duplexfold`, `duplex_subopt`, and its comparative variants
- Add SWIG wrapper for [vrna_exp_params_rescale\(\)](#)
- Add explicit destructor for SWIG generated `vrna_md_t` objects

- Add SWIG perl5 typemap for simple nested STL vectors
- Add dummy field in [vrna_structured_domains_s](#)
- Add note about SSE optimized code in reference manual
- Add SWIG interface for findpath implementation
- Add prepare() functions for ptypes-arrays and `vrna_(exp_)param_t`
- Add warnings for ignored commands in function [vrna_commands_apply\(\)](#)
- Add callback featured functions for sliding window MFE and PF implementations
- Change default behavior of adding soft constraints to a `vrna_fold_compound_t` (store only)
- Several fixes with respect to G-Quadruplex prediction in sliding-window MFE recursions (single sequence and comparative implementation)
- Replace comparative sliding-window MFE recursions (All hits are reported to callback and can be filtered in a post-processing step)
- API: Remove `E_mb_loop_stack()` and introduce new function [vrna_E_mb_loop_stack\(\)](#) as a replacement
- API: change data type of all constraint bit-flags from `char` to `unsigned char`
- API: change data type of `a2s` array in comparative structure prediction from `unsigned short` to `unsigned int`
- API: Change function parameter order in [vrna_probs_window\(\)](#) to follow the style of other callback-aware functions in RNALib
- Move sliding-window MFE implementations to new file `mfe_window.c`
- Fix building PDF Reference manual with non-standard executable paths
- Fix redefinition of macro `ON_SAME_STRAND()` in `subopt.c`
- Fix dangling end issues in sliding-window MFE implementations
- Fix regression for `-canonicalBPonly` switch in `RNAfold/RNAcofold/RNAsubopt`
- Fix building sliding-window MFE implementation without SVM support
- Fix parsing of STOCKHOLM 1.0 MSA files that contain MSA spanning multiple blocks
- Fix Alidot link in RNAalifold manpage
- Fix wrong pre-processor flags when enabling single-precision PF computations
- Fix unit testing perl5 interface by including `builddir/tests` in `PERL5LIB` path
- Fix buffer overflow in hairpin loop sequence motif extraction for circular RNAs
- Fix out-of-bounds memory access in `neighbor.c`
- Restore capability to compile stand-alone findpath utility
- Restore capability to use non-standard alphabets for structure prediction
- Restore old-API random number functions in SWIG interface
- Allow additional control characters in MAF MSA input that do not end a block
- Improve reference manual
- Make functions in [pair_mat.h](#) static inline
- Prevent users from adding out-of-range base pair soft constraints

- Inline print functions in color_output.inc
- Start documenting callback features in reference manual
- Re-write large portions of sliding-window PF implementation
- Introduce soft-constraint state flag
- Clean-up SWIG unit test framework
- Remove obsolete scripts ct2b.pl and colorrna.pl from src/Utils directory
- Remove old RNAfold tutorial

Version 2.3.x

Version 2.3.5 (Release date: 2017-04-14)

- Fix duplication of output filename prefix in RNAfold
- Add V3.0 API for sliding window partition function (a.k.a. RNAPLfold)
- Add G-Quadruplex prediction to RNALalifold
- Add SWIG wrappers for callback-based sliding window comparative MFE prediction
- Add SSE4.1 multiloop decomposition for single sequence MFE prediction
- Enable RNAfold unit tests to run in parallel
- Enable users to turn-off base pair probability computations in RNACofold with -a option
- Split move set in neighbor.c

Version 2.3.4 (Release date: 2017-03-10)

- Fix G-Quadruplex probability computation for single sequences
- Fix double-free when using SHAPE reactivity data in RNAalifold
- Fix out-of-bounds access in strand_number array
- Fix weighting of SHAPE reactivity data in consensus structure prediction when fewer data than sequences are present
- Fix z-score output in RNALfold
- Substitute field name 'A0'/'B0' in data structure `vrna_dimer_conc_s` by 'Ac_start'/'Bc_start' to avoid clashes with termios.h (Mac OSX Python wrapper bug)
- Minimize usage of 'unsafe' `sprintf()` calls
- Enhance auto-id feature in executable programs
- Always sanitize output file names to avoid problems due to strange FASTA headers
- Lift restrictions of FASTA header length in RNAfold, RNACofold, and RNAeval
- Add ViennaRNA/config.h with pre-processor definitions of configure time choices
- Add test-suite for RNAfold
- Add functions to produce colored EPS structure alignments
- Add function to write Stockholm 1.0 formatted alignments
- Add function to sanitize file names

- Add callback based implementation for sliding-window MFE prediction (single sequences, comparative structure prediction)
- Add fast API 3.0 implementations to generate structural neighbors and perform steepest descent / random walks (Thanks to Gregor!)
- Add parameter option to RNALalifold for colored EPS structure alignment and structure plot output
- Add parameter option to RNALalifold to write hits into Stockholm file
- Add parameter option to RNAalifold to write Stockholm 1.0 formatted output
- Add parameter option to RNAalifold to suppress stderr spam
- Add auto-id feature to RNAplot, RNALfold, RNAsubopt, RNApfold, RNAheat
- Add SHAPE reactivity derived pseudo-energies as separate output in RNAalifold
- Add colored output to RNA2Dfold, RNALalifold, RNALfold, RNAduplex, RNAheat, RNAinverse, RNApfold, and RNAsubopt
- Add command line parameters to RNAsubopt to allow for specification of input/output files

Version 2.3.3 (Release date: 2017-01-24)

- Fix multiloop contributions for comparative partition function
- Fix building python2 extension module for OSX

Version 2.3.2 (Release date: 2017-01-18)

- Fix pair probability plist creation with G-Quadruplexes
- Allow for specification of python2/3-config at configure time
- Fix init of vrna_md_t data structure after call to [set_model_details\(\)](#)
- Fix bug in consensus partition function with hard constraints that force nucleotides to be paired
- Fix compilation of functions that use ellipsis/va_list
- Enable generic hard constraints by default
- Fix init of partition function DP matrices for unusually short RNAs
- Fix behavior of RNApfold for unusually short RNAs
- Report SCI of 0 in RNAalifold when sum of single sequence MFEs is 0
- Avoid multiple includes of [pair_mat.h](#)
- Add configure flag to build entirely static executables

Version 2.3.1 (Release date: 2016-11-15)

- Add description for how to use unstructured domains through command files to reference manual and RNAfold manpage
- Fix compilation issue for Windows platforms with MingW
- Add missing newline in non-TTY-color output of [vrna_message_info\(\)](#)
- Fix regression in [vrna_md_update\(\)](#) that resulted in incomplete init of reverse-basepair type array
- Extend coverage of generic hard constraints for partition function computations
- Fix scaling of secondary structure in EPS plot such that it always fits into bounding box
- Several fixes and improvements for SWIG generated scripting language interface(s)

Version 2.3.0 (Release date: 2016-11-01)

- Add grammar extension with structured and unstructured domains
- Add default implementation for unstructured domains to allow for ligand/protein binding to unpaired structure segments (MFE and PF for single sequences)
- Introduced command files that subsume constraint definition files (currently used in RNAfold and RNAcofold)
- Replace explicit calls to `asprintf()` with portable equivalent functions in the library
- Fix configure script to deal with situations where Perl module can't be build
- Fix bug in `doc/Makefile.am` that prevented HTML installation due to long argument list
- Added utility functions that deal with conversion between different units
- Bugfix in SWIG wrapped generic soft constraint feature
- Add `subopt()` and `subopt_zuker()` methods to SWIG wrapped `fold_compound` objects
- Bugfix multiloop decomposition in MFE for circular RNAs
- Add separate function to compute pscore for alignments
- Renamed `VRNA_VC_TYPE_*` macros to `VRNA_FC_TYPE_*`
- Bugfix regression that prevented programs to fail on too long input sequences
- Extend EPS dot-plot in RNAfold to include motif/binding probabilities from unstructured domains
- Add variadic functions for error/warning/info message
- Add ID manipulation feature to RNAeval
- Extend API for soft constraint feature for more fine-grained control
- Add section on SWIG wrapped functions in reference manual
- Fix bug in interior loop computations when hard constraints result in non-canonical base pairs

Version 2.2.x

Version 2.2.10 (Release date: 2016-09-06)

- Do not 'forget' subopt results when output is not written to file handle and sorting is switched off
- Fix bad memory access in `vrna_subopt()` with sorted output
- Add SWIG wrappers for `vrna_subopt_cb()`
- Correctly show if C11 features are activated in configure status
- Fix autoconf checks to allow for cross compilation again

Version 2.2.9 (Release date: 2016-09-01)

- Fix bug in partition function scaling for backward compatibility of `ali_pf_fold()`
- Stabilize v3.0 API when building RNAlib and third party program linking against it with compilers that use different C/C++ standards
- Add details on how to link against RNAlib to the reference manual
- Fix RNAlib2.pc
- Fix bug for temperature setting in RNAplfold

- Use -fflat-lto-objects for static RNAlib library to allow linking without LTO
- Fix interpretation of 'P' hard constraint for single nucleotides in constraint definition files
- Add 'A' command for hard constraints
- Fix several hard constraint corner-cases in MFE and partition function computation when nucleotides must not be unpaired
- Fix order of hard constraints when read from input file
- Allow for non-canonical base pairs in MFE and partition function computations if hard constraints demand it
- Fix behavior of --without-swig configure script option
- Fix bug in hard constraints usage of exterior loop MFE prediction with odd dangles
- Add parsers for Clustal, Stockholm, FASTA, and MAF formatted alignment files
- Enable RNAalifold to use Clustal, Stockholm, FASTA, or MAF alignments as input
- Lift restriction of sequence number in alignments for RNAalifold
- Enable ANSI colors for TTY output in RNAfold, RNAcofold, RNAalifold, RNAsubopt, and warnings/errors issued by RNAlib
- Add various new commandline options to manipulate sequence/alignment IDs in RNAfold, RNAcofold and RNAalifold

Version 2.2.8 (Release date: 2016-08-01)

- Fix bad memory access in RNAalifold
- Fix regression in RNAalifold to restore covariance contribution ratio determination for circular RNA alignments
- Changed output of RNAsubopt in energy-band enumeration mode to print MFE and energy range in kcal/mol instead of 10cal/mol
- Include latest Kinfold sources that make use of v3.0 API, therefore speeding up runtime substantially
- Re-activate warnings in RNAeval when non-canonical base pairs are encountered
- Fix syntactic incompatibilities that potentially prevented compilation with compilers other than gcc
- dd function to compare nucleotides encoded in IUPAC format
- Fix regression in energy evaluation for circular RNA sequences
- Fix regression in suboptimal structure enumeration for circular RNAs
- Allow for P i-j k-l commands in constraint definition files
- Make free energy evaluation functions polymorphic
- Add free energy evaluation functions that allow for specifying verbosity level
- Secure functions in alphabet.c against NULL pointer arguments
- Fix incompatibility with swig >= 3.0.9
- Fix memory leak in swig-generated scripting language interface(s) for user-provided target language soft-constraint callbacks
- Expose additional functions to swig-generated scripting language interface(s)
- Build Python3 interface by default
- Start of more comprehensive scripting language interface documentation

- Fix linking of python2/python3 interfaces when libpython is in non-standard directory
- Restructured viennarna.spec for RPM based distributions
- Several syntactic changes in the implementation to minimize compiler warnings
- Fix `--with-*/--without-*` and `--enable-*/--disable-*` configure script behavior

Version 2.2.7 (Release date: 2016-06-30)

- Fix partition function scaling for long sequences in RNAfold, RNAalifold, and RNAup
- Fix backtracking issue in RNAcofold when `--noLP` option is activated
- Fix hard constraints issue for circular RNAs in generating suboptimal structures
- Rebuild reference manual only when actually required

Version 2.2.6 (Release date: 2016-06-19)

- Plugged memory leak in RNAcofold
- Fixed partition function rescaling bug in RNAup
- Fixed bug in RNALfold with window sizes larger than sequence length
- Re-added SCI parameter for RNAalifold
- Fixed backtracking issue for large G-quadruplexes in RNAalifold
- Fixed missing FASTA id in RNAeval output
- Added option to RNAalifold that allows to specify prefix for output files
- Several fixes and additional functions/methods in scripting language interface(s)
- Added version information for scripting language interface(s)
- Some changes to allow for compilation with newer compilers, such as gcc 6.1

Version 2.2.5 (Release date: 2016-04-09)

- Fixed regression in RNAcofold that prohibited output of concentration computations
- Fixed behavior of RNAfold and RNAcofold when hard constraints create empty solution set (programs now abort with error message)
- Added optional Python 3 interface
- Added RNA::Params Perl 5 sub-package
- Update RNA::Design Perl 5 sub-package
- Simplified usage of v3.0 API with default options
- Wrap more functions of v3.0 API in SWIG generated scripting language interfaces
- Plugged some memory leaks in SWIG generated scripting language interfaces
- Changed parameters of recursion status callback in `vrna_fold_compound_t`
- Enable definition and binding of callback functions from within SWIG target language
- Added optional subpackage Kinwalker
- Added several configure options to ease building and packaging under MacOS X
- Added new utility script RNAdesign.pl

Version 2.2.4 (Release date: 2016-02-19)

- Fixed bug in RNAsubopt that occasionally produced cofolded structures twice
- Removed debugging output in preparations of consensus structure prediction datastructures

Version 2.2.3 (Release date: 2016-02-13)

- Added postscript annotations for found ligand motifs in RNAfold
- Added more documentation for the constraints features in RNAfold and RNAalifold
- Restore backward compatibility of [get_alipf_arrays\(\)](#)

Version 2.2.2 (Release date: 2016-02-08)

- Fix regression bug that occasionally prevented backtracking with RNAcofold --noLP

Version 2.2.1 (Release date: 2016-02-06)

- Fix regression bug that made RNAcofold -a unusable
- Fix regression bug that prohibited RNAfold to compute the MEA structure when G-Quadruplex support was switched on
- Fix bug in Kinfold to enable loading energy parameters from file
- Fix potential use of uninitialized value in RNApdist
- Add manpage for ct2db
- Fix MEA computation when G-Quadruplex support is activated
- Allow for vendor installation of the perl interface using INSTALLDIRS=vendor at configure time
- Install architecture dependent and independent files of the perl and python interface to their correct file system locations

Version 2.2.0 (Release date: 2016-01-25)

- RNAforester is now of version 2.0
- New program RNApvmin to compute pseudo-energy perturbation vector that minimizes discrepancy between observed and predicted pairing probabilities
- SHAPE reactivity support for RNAfold, RNAsubopt, and RNAalifold
- Ligand binding to hairpin- and interior-loop motif support in RNAfold
- New commandline option to limit maximum base pair span for RNAfold, RNAsubopt, RNAcofold, and RNAalifold
- Bugfix in RNAheat to remove numerical instabilities
- Bugfix in RNApdex to allow for computation of interactions without length limitation
- Bugfix in RNAplot for simple layouts and hairpins of size 0
- (generic) hard- and soft-constraints for MFE, partition function, base pair probabilities, stochastic backtracking, and suboptimal secondary structures of single sequences, sequence alignments, and sequence dimers
- libsvm version as required for z-scoring in RNALfold is now 3.20
- Stochastic backtracking for single sequences is faster due to usage of Boustrophedon scheme
- First polymorphic functions [vrna_mfe\(\)](#), [vrna_pf\(\)](#), and [vrna_pbacktrack\(\)](#).

- The FLT_OR_DBL macro is now a typedef
- New functions to convert between different secondary structure representations, such as helix lists, and RNAshapes abstractions
- First object-oriented interface for new API functions in the scripting language interfaces
- new ViennaRNA-perl submodule that augments the Perl interface to RNALib
- Ligand binding to hairpin- and interior-loop motif support in C-library and scripting language interfaces.
- Libraries are generated using libtool
- Linking of libraries and executables defaults to use Link Time Optimization (LTO)
- Large changes in directory structure of the source code files

Version 2.1.x

Version 2.1.9

- Fixed integer underflow bug in RNALfold
- Added Sequence Conservation index (SCI) option to RNAalifold
- Fixed bug in energy evaluation of dangling ends / terminal mismatches of exterior loops and multibranch loops
- Fixed bug in alifold partition function for circular RNAs
- Fixed bug in alifold that scrambled backtracing with activated G-Quadruplex support
- Fixed bug in alifold backtracking for larger G-Quadruplexes

Version 2.1.8

- Repaired incorporation of RNAinverse user provided alphabet
- Fix missing FASTA ID in RNAeval output
- prevent race condition in parallel calls of [Lfold\(\)](#)
- Fixed memory bug in [Lfold\(\)](#) that occurred using long sequences and activated G-Quad support
- Added latest version of switch.pl

Version 2.1.7

- Fixed bug in RNALfold -z
- Python and Perl interface are compiling again under MacOSX
- Fixed handling of C arrays in Python interface
- Added latest version of switch.pl
- Make relplot.pl work with RNACofold output

Version 2.1.6

- New cmdline switches allow for elimination of non-canonical base pairs from constraint structures in RNAfold, RNAalifold and RNAsubopt
- updated moveset functions
- final fix for discrepancy of tri-loop evaluation between partition function and mfe
- pkg-config file now includes the OpenMP linker flag if necessary
- New program ct2db allows for conversion of .ct files into dot-bracket notation (incl. pseudo-knot removal)

Version 2.1.5

- Fix for discrepancy between special hairpin loop evaluation in partition functions and MFE

Version 2.1.4

- Fix of G-quadruplex support in [subopt\(\)](#)
- Fix for discrepancy between special hairpin loop evaluation in partition functions and MFE

Version 2.1.3

- RNAfold: Bugfix for ignoring user specified energy parameter files
- RNACofold: Bugfix for crashing upon constrained folding without specifying a constraint structure
- RNAsubopt: Added G-quadruplex support
- RNAalifold: Added parameter option to specify base pair probability threshold in dotplot
- Fix of several G-quadruplex related bugs
- Added G-quadruplex support in [subopt\(\)](#)

Version 2.1.2

- RNAfold: Bugfix for randomly missing probabilities in dot-plot during batch job execution
- RNAeval: Bugfix for misinterpreted G-quadruplex containing sequences where the quadruplex starts at nucleotide 1
- RNAsubopt: Slight changes to the output of stochastic backtracking and zucker subopt
- Fix of some memory leaks
- Bugfixes in [zuckersubopt\(\)](#), [assign_plist_from_pr\(\)](#)
- New threadsafe variants of putoutpU_prob*() for LPfold()
- Provision of python2 interface support.

Version 2.1.1

- Bugfix to restore backward compatibility with ViennaRNA Package 1.8.x API (this bug also affected proper usage of the perl interface)

Version 2.1.0

- G-Quadruplex support in RNAfold, RNACofold, RNALfold, RNAalifold, RNAeval and RNAplot
- LPfold got a new option to output its computations in split-mode
- several G-Quadruplex related functions were introduced with this release
- several functions for moves in an RNA landscape were introduced
- new function in alipfold.c now enables access to the partition function matrices of [alipf_fold\(\)](#)
- different numeric approach was implement for concentration dependend co-folding to avoid instabilities which occurred under certain circumstances

Version 2.0.x

Version 2.0.7

- Bugfix for RNAplfold where segfault happened upon usage of -O option
- Corrected misbehavior of RNAeval and RNAplot in tty mode

Version 2.0.6

- Bugfix for bad type casting with gcc under MacOSX (resulted in accidental "sequence too long" errors)
- Bugfix for disappearing tri-/hexaloop contributions when read in from certain parameter files
- Bugfix for RNALfold that segfaulted on short strange sequences like AT+ repeats
- Change of RNA2Dfold output format for stochastic backtracking

Version 2.0.5

- Restored z-score computation capabilities in RNALfold

Version 2.0.4

- Bugfix for RNAcofold partition function
- Perl wrapper compatibility to changed RNALib has been restored
- Backward compatibility for partition function calls has been restored

Version 2.0.3

- Bugfix for RNAalifold partition function and base pair probabilities in v2.0.3b
- Added Boltzmann factor scaling in RNAsubopt, RNAalifold, RNAplfold and RNAcofold
- Bugfix for alifold() in v2.0.3b
- Restored threadsafety of folding matrix access in LPfold.c, alifold.c, part_func.c, part_func_co.c and part_func_up.c
- Added several new functions regarding threadsafe function calls in terms of concurrently changing the model details
- Added pkg-config file in the distribution to allow easy checks for certain RNALib2 versions, compiler flags and linker flags.

Version 2.0.2

- added support for Boltzmann factor scaling in RNAfold
- fixed fastaheader to filename bug
- plugged some memory leaks

Version 2.0.1

- First official release of version 2.0
- included latest bugfixes

History

2011-03-10 Ronny Lorenz ronny@tbi.univie.ac.at

- new naming scheme for all shipped energy parameter files
- fixed bugs that appear while compiling with gcc under MacOS X
- fixed bug in RNAup –interaction-first where the longer of the first two sequences was taken as target
- added full FASTA input support to RNAfold, RNAcofold, RNAheat, RNAplfold, RNALfoldz, RNAsubopt and RNALfold

2010-11-24 Ronny Lorenz ronny@tbi.univie.ac.at

- first full pre-release of version 2.0

2009-11-03 Ivo Hofacker ivo@tbi.univie.ac.at

- Fix memory corruption in [PS_color_aln\(\)](#)

2009-09-09 Ivo Hofacker ivo@tbi.univie.ac.at

- Fix bug in RNAplfold when -u and -L parameters are equal
- Fix double call to [free_arrays\(\)](#) in RNAfold.c
- Improve drawing of cofolded structures

2009-05-14 Ivo Hofacker ivo@tbi.univie.ac.at

- Fix occasional segfault in RNAalifold's [print_alifold\(\)](#)

2009-02-24 Ivo Hofacker ivo@tbi.univie.ac.at

- Add -MEA options to RNAfold and RNAalifold
- change [energy_of_alistruct](#) to return float not void

2009-02-24 Ivo Hofacker ivo@tbi.univie.ac.at

- RNAfold will draw structures unless -noPS is used (no more "structure too long" messages)
- Restore the "alifold.out" output from RNAalifold -p
- RNAalifold -circ did not work due to wrong return type
- Accessibility calculation with RNAplfold would give wrong results for $u \leq 30$

2008-12-03 Ivo Hofacker ivo@tbi.univie.ac.at

- Add Zuker style suboptimals to RNAsubopt (-z)
- [get_line\(\)](#) should be much faster when reading huge sequences (e.g. whole chromosomes for RNALfold)

2008-08-12 Ivo Hofacker ivo@tbi.univie.ac.at

- Add Ribosum matrices for covariance scoring in RNAalifold

2008-06-27 Ivo Hofacker ivo@tbi.univie.ac.at

- Change RNAalifold to use Berni's new energy evaluation w/o gaps
- Add stochastic backtracking in RNAalifold

2008-07-04 Ivo Hofacker ivo@tbi.univie.ac.at

- modify output of RNAup (again). Program reading RNAup output will have to be updated!

2008-07-02 Ivo Hofacker ivo@tbi.univie.ac.at

- RNAplfold now computes accessibilities for all regions up to a max length simultaneously. Slightly slower when only 1 value is needed, but much faster if all of them are wanted. This entails a new output format. Programs reading accessibility output from RNAplfold need to be updated!

2008-03-31 Stephan Bernhart berni@tbi.univie.ac.at

- add cofolding to RNAsubopt

2008-01-08 Ivo Hofacker ivo@tbi.univie.ac.at

- ensure circfold works even for open chain

2007-12-13 Ulli Mueckstein ulli@tbi.univie.ac.at

- update RNAup related files RNAup can now include the intramolecular structure of both molecules and handles constraints.

2007-12-05 Ronny Lorenz ronny@tbi.univie.ac.at

- add circfold variants in part_func.c alipfold.c subopt.c

2007-09-19 Ivo Hofacker ivo@tbi.univie.ac.at

- compute the centroid structure of the ensemble in RNAfold -p
- fix a missing factor 2 in [mean_bp_dist\(\)](#). CAUTION ensemble diversities returned by RNAfold -p are now twice as large as in earlier versions.

2007-09-04 Ivo Hofacker ivo@tbi.univie.ac.at

- fix a bug in [Lfold\(\)](#) where base number n-max-4 would never pair

2007-08-26 Ivo Hofacker ivo@tbi.univie.ac.at

- add RNAaliduplex the alignment version of RNAduplex
- introduce a minimal distance between hits produced by duplex_subopt()

2007-07-03 Ivo Hofacker ivo@tbi.univie.ac.at

- add a [loop_energy\(\)](#) function to compute energy of a single loop

2007-06-23 Ivo Hofacker ivo@tbi.univie.ac.at

- add aliLfold() and RNALalifold, alignment variant of [Lfold\(\)](#)

2007-04-30 Ivo Hofacker ivo@tbi.univie.ac.at

- add RNAup to distribution

2007-04-15 Ivo Hofacker ivo@tbi.univie.ac.at

- fix segfault in colorps output (thanks to Andres Varon)

2007-03-03 Ivo Hofacker ivo@tbi.univie.ac.at

- avoid unnormalized doubles in scale[], big speedup for [pf_fold\(\)](#) on very long sequences

2007-02-03 Ivo Hofacker ivo@tbi.univie.ac.at

- RNAalifold can now produce colored structure plots and alignment plots

2007-02-01 Ivo Hofacker ivo@tbi.univie.ac.at

- Fix segfault in RNAplfold because of missing prototype

2006-12-01 Ivo Hofacker ivo@tbi.univie.ac.at

- RNAduplex would segfault when no structure base pairs are possible

2006-08-22 Ivo Hofacker ivo@tbi.univie.ac.at

- add computation stacking probabilities using RNAfold -p2
- add -noPS option for RNAfold to suppress drawing structures

2006-08-09 Stephan Bernhart berni@tbi.univie.ac.at

- RNAplfold can now compute probabilities of unpaired regions (scanning version of RNAup)

2006-06-14 Ivo Hofacker ivo@tbi.univie.ac.at

- compile library with -fpic (if available) for use as shared library in the Perl module.
- fix another bug when calling [Lfold\(\)](#) repeatedly
- fix switch cmdline parsing in RNAalifold (-mis implied -4)
- fix bug in [cofold\(\)](#) with dangles=0

2006-05-08 Ivo Hofacker ivo@tbi.univie.ac.at

- fix segfault in [Lfold\(\)](#) when calling repeatedly
- fix structure parsing in RNAstruct.c (thanks to Michael Pheasant for reporting both bugs)
- add duplexfold() and [alifold\(\)](#) to Perl module
- distinguish window size and max pair span in LPfold

2006-04-05 Ivo Hofacker ivo@tbi.univie.ac.at

- fix performance bug in [co_pf_fold\(\)](#)
- use relative error for termination of Newton iteration

2006-03-02 Ivo Hofacker ivo@tbi.univie.ac.at

- add circular folding in [alifold\(\)](#)

2006-01-18 Ivo Hofacker ivo@tbi.univie.ac.at

- cleanup berni partition cofold code, including several bug fixes

2006-01-16 Ivo Hofacker ivo@tbi.univie.ac.at

- update RNAplfold to working version
- add [PS_dot_plot_turn\(\)](#) in PS_dot.c

2005-11-07 Ivo Hofacker ivo@tbi.univie.ac.at

- add new utilities colorna and coloraln

2005-10-11 Christoph Flamm xtof@tbi.univie.ac.at

- adapt [PS_rna_plot\(\)](#) for drawing co-folded structures

2005-07-24 Ivo Hofacker ivo@tbi.univie.ac.at

- fix a few memory problems in structure comparison routines

2005-04-30 Ivo Hofacker ivo@blini.tbi.univie.ac.at

- add folding of circular RNAs

2005-03-11 Ivo Hofacker ivo@blini.tbi.univie.ac.at

- add -mis option to RNAalifold to give "most informative sequence" as consensus

2005-02-10 Ivo Hofacker ivo@tbi.univie.ac.at

- move [alifold\(\)](#) into the library

2004-12-22 Stephan Bernhart berni@tbi.univie.ac.at

- add partition function version of RNAcofold

2004-12-23 Ivo Hofacker ivo@tbi.univie.ac.at

- add RNAln for fast structural alignments (RNAdist improvement)

2004-08-12 Ivo Hofacker ivo@tbi.univie.ac.at

- fix constrained folding in stochastic backtracking

2004-07-21 Ivo Hofacker ivo@tbi.univie.ac.at

- add RNAduplex, to compute hybrid structures without intra-molecular pairs

2004-02-09 Ivo Hofacker ivo@tbi.univie.ac.at

- fix bug in fold that caused segfaults when using Intel compiler
- add computation of ensemble diversity to RNAfold

2003-09-10 Ivo Hofacker ivo@tbi.univie.ac.at

- add annotation options to RNAplot

2003-08-04 Ivo Hofacker ivo@tbi.univie.ac.at

- stochastic backtracking finally works. Try e.g. RNAsubopt -p 10

2003-07-18 Ivo Hofacker ivo@tbi.univie.ac.at

- add relplot.pl and rotate_ss.pl utilities for reliability annotation and rotation of rna structure plots

2003-01-29 Ivo Hofacker ivo@tbi.univie.ac.at

- add RNALfold program to compute locally optimal structures with maximum pair span.
- add RNAcofold for computing hybrid structure

2002-11-07 Ivo Hofacker ivo@tbi.univie.ac.at

- change [Make_bp_profile\(\)](#) and [profile_edit_distance\(\)](#) to use simple (float *) arrays; makes Perl access much easier. RNAdist -B now works again

2002-10-28 Ivo Hofacker ivo@tbi.univie.ac.at

- Improved Perl module with pod documentation; allow to write things like (\$structure, \$energy) = RNA \leftrightarrow ::fold(\$seq); Compatibility warning: the ptrvalue() and related functions are gone, see the pod documentation for alternatives.

2002-10-29 Ivo Hofacker ivo@tbi.univie.ac.at

- added svg structure plots in PS_dot.c and RNAplot

2002-08-15 Ivo Hofacker ivo@tbi.univie.ac.at

- Improve reading of clustal files (alifold)
- add a sample alifold.cgi script

2001-09-18 Ivo Hofacker ivo@tbi.univie.ac.at

- moved suboptimal folding into the library, thus it's now accessible from the Perl module

2001-08-31 Ivo Hofacker ivo@tbi.univie.ac.at

- added co-folding support in [energy_of_struct\(\)](#), and thus RNAeval

2001-04-30 Ivo Hofacker ivo@tbi.univie.ac.at

- switch from handcrafted makefiles to automake and autoconf

2001-04-05 Ivo Hofacker ivo@tbi.univie.ac.at

- added PS_rna_plot_a to produce structure plots with annotation

2001-03-03 Ivo Hofacker ivo@tbi.univie.ac.at

- add alifold; predict consensus structures from alignment

2000-09-28 Ivo Hofacker ivo@tbi.univie.ac.at

- add -d3 option to RNAfold for co-axial stacking

Chapter 11

Deprecated List

Global **alifold** (const char **strings, char *structure)

Usage of this function is discouraged! Use [vrna_alifold\(\)](#), or [vrna_mfe\(\)](#) instead!

Global **alimake_pair_table** (const char *structure)

Use [vrna_pt_ali_get\(\)](#) instead!

Global **alipbacktrack** (double *prob)

Use [vrna_pbacktrack\(\)](#) instead!

Global **alipf_circ_fold** (const char **sequences, char *structure, vrna_ep_t **pl)

Use [vrna_pf\(\)](#) instead

Global **alipf_fold** (const char **sequences, char *structure, vrna_ep_t **pl)

Use [vrna_pf\(\)](#) instead

Global **alipf_fold_par** (const char **sequences, char *structure, vrna_ep_t **pl, vrna_exp_param_t *parameters, int calculate_bppm, int is_constrained, int is_circular)

Use [vrna_pf\(\)](#) instead

Global **aliPS_color_aln** (const char *structure, const char *filename, const char *seqs[], const char *names[])

Use [vrna_file_PS_aln\(\)](#) instead!

File **aln_util.h**

Use [ViennaRNA/utis/alignments.h](#) instead

Global **assign_plist_from_db** (vrna_ep_t **pl, const char *struc, float pr)

Use [vrna_plist\(\)](#) instead

Global **assign_plist_from_pr** (vrna_ep_t **pl, FLT_OR_DBL *probs, int length, double cutoff)

Use [vrna_plist_from_probs\(\)](#) instead!

Global **b2C** (const char *structure)

See [vrna_db_to_tree_string\(\)](#) and [VRNA_STRUCTURE_TREE_SHAPIRO_SHORT](#) for a replacement

Global **b2HIT** (const char *structure)

See [vrna_db_to_tree_string\(\)](#) and [VRNA_STRUCTURE_TREE_HIT](#) for a replacement

Global **b2Shapiro** (const char *structure)

See [vrna_db_to_tree_string\(\)](#) and [VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT](#) for a replacement

Global **base_pair**

Do not use this variable anymore!

Global **bondT**

Use [vrna_bp_stack_t](#) instead!

Global **bp_distance** (const char *str1, const char *str2)

Use [vrna_bp_distance](#) instead

Global **bppm_symbol** (const float *x)

Use [vrna_bpp_symbol\(\)](#) instead!

Global **bppm_to_structure** (char *structure, FLT_OR_DBL *pr, unsigned int length)

Use [vrna_db_from_probs\(\)](#) instead!

Global **centroid** (int length, double *dist)

This function is deprecated and should not be used anymore as it is not threadsafe!

File **char_stream.h**

Use [ViennaRNA/datastructures/char_stream.h](#) instead

Global **circalifold** (const char **strings, char *structure)

Usage of this function is discouraged! Use [vrna_alicircfold\(\)](#), and [vrna_mfe\(\)](#) instead!

Global **circfold** (const char *sequence, char *structure)

Use [vrna_circfold\(\)](#), or [vrna_mfe\(\)](#) instead!

Global **co_pf_fold** (char *sequence, char *structure)

{Use [vrna_pf_dimer\(\)](#) instead!}

Global **co_pf_fold_par** (char *sequence, char *structure, vrna_exp_param_t *parameters, int calculate_↵
bppm, int is_constrained)

Use [vrna_pf_dimer\(\)](#) instead!

Global **cofold** (const char *sequence, char *structure)

use [vrna_mfe_dimer\(\)](#) instead

Global **cofold_par** (const char *string, char *structure, vrna_param_t *parameters, int is_constrained)

use [vrna_mfe_dimer\(\)](#) instead

Global **compute_BPdifferences** (short *pt1, short *pt2, unsigned int turn)

Use [vrna_refBPdist_matrix\(\)](#) instead

Global **compute_probabilities** (double FAB, double FEA, double FEB, vrna_ep_t *prAB, vrna_ep_t *prA,
vrna_ep_t *prB, int Alength)

{ Use [vrna_pf_dimer_probs\(\)](#) instead!}

Global **constrain_ptypes** (const char *constraint, unsigned int length, char *ptype, int *BP, int min_loop↵
_size, unsigned int idx_type)

Do not use this function anymore! Structure constraints are now handled through [vrna_hc_t](#) and related functions.

File **constraints.h**

Use [ViennaRNA/constraints/basic.h](#) instead

File **constraints_hard.h**

Use [ViennaRNA/constraints/hard.h](#) instead

File **constraints_ligand.h**

Use [ViennaRNA/constraints/ligand.h](#) instead

File **constraints_SHAPE.h**

Use [ViennaRNA/constraints/SHAPE.h](#) instead

File **constraints_soft.h**

Use [ViennaRNA/constraints/soft.h](#) instead

File **convert_epars.h**

Use [ViennaRNA/params/convert.h](#) instead

Global **copy_pair_table** (const short *pt)

Use [vrna_ptable_copy\(\)](#) instead

Global **cpair**

Use [vrna_cpair_t](#) instead!

Global `cv_fact`

See `vrna_md_t.cv_fact`, and `vrna_mfe()` to avoid using global variables

File `data_structures.h`

Use `ViennaRNA/datastructures/basic.h` instead

Global `destroy_TwoDfold_variables` (`TwoDfold_vars` *`our_variables`)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_↵_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `destroy_TwoDpfold_variables` (`TwoDpfold_vars` *`vars`)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_↵_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `E_Stem` (int `type`, int `si1`, int `sj1`, int `extLoop`, `vrna_param_t` *`P`)

Please use one of the functions `vrna_E_ext_stem()` and `E_MLstem()` instead! Use the former for cases where `extLoop` != 0 and the latter otherwise.

File `energy_const.h`

Use `ViennaRNA/params/constants.h` instead

Global `energy_of_alistruct` (const char **`sequences`, const char *`structure`, int `n_seq`, float *`energy`)

Usage of this function is discouraged! Use `vrna_eval_structure()`, and `vrna_eval_covar_structure()` instead!

Global `energy_of_circ_struct` (const char *`string`, const char *`structure`)

This function is deprecated and should not be used in future programs Use `energy_of_circ_structure()` instead!

Global `energy_of_circ_struct_par` (const char *`string`, const char *`structure`, `vrna_param_t` *`parameters`, int `verbosity_level`)

Use `vrna_eval_structure()` or `vrna_eval_structure_verbose()` instead!

Global `energy_of_circ_structure` (const char *`string`, const char *`structure`, int `verbosity_level`)

Use `vrna_eval_structure()` or `vrna_eval_structure_verbose()` instead!

Global `energy_of_move` (const char *`string`, const char *`structure`, int `m1`, int `m2`)

Use `vrna_eval_move()` instead!

Global `energy_of_move_pt` (short *`pt`, short *`s`, short *`s1`, int `m1`, int `m2`)

Use `vrna_eval_move_pt()` instead!

Global `energy_of_struct` (const char *`string`, const char *`structure`)

This function is deprecated and should not be used in future programs! Use `energy_of_structure()` instead!

Global `energy_of_struct_par` (const char *`string`, const char *`structure`, `vrna_param_t` *`parameters`, int `verbosity_level`)

Use `vrna_eval_structure()` or `vrna_eval_structure_verbose()` instead!

Global `energy_of_struct_pt` (const char *`string`, short *`ptable`, short *`s`, short *`s1`)

This function is deprecated and should not be used in future programs! Use `energy_of_structure_pt()` instead!

Global `energy_of_struct_pt_par` (const char *`string`, short *`ptable`, short *`s`, short *`s1`, `vrna_param_t` *`parameters`, int `verbosity_level`)

Use `vrna_eval_structure_pt()` or `vrna_eval_structure_pt_verbose()` instead!

Global `energy_of_structure` (const char *`string`, const char *`structure`, int `verbosity_level`)

Use `vrna_eval_structure()` or `vrna_eval_structure_verbose()` instead!

Global `energy_of_structure_pt` (const char *`string`, short *`ptable`, short *`s`, short *`s1`, int `verbosity_level`)

Use `vrna_eval_structure_pt()` or `vrna_eval_structure_pt_verbose()` instead!

File `energy_par.h`

Use `ViennaRNA/params/default.h` instead

Global `exp_E_ExtLoop` (int `type`, int `si1`, int `sj1`, `vrna_exp_param_t` *`P`)

Use `vrna_exp_E_ext_stem()` instead!

Global **expHairpinEnergy** (int u, int type, short si1, short sj1, const char *string)

Use [exp_E_Hairpin\(\)](#) from [loop_energies.h](#) instead

Global **expLoopEnergy** (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1)

Use [exp_E_IntLoop\(\)](#) from [loop_energies.h](#) instead

Global **export_ali_bppm** (void)

Usage of this function is discouraged! The new [vrna_fold_compound_t](#) allows direct access to the folding matrices, including the pair probabilities! The pair probability array returned here reflects the one of the latest call to [vrna_pf\(\)](#), or any of the old API calls for consensus structure partition function folding.

Global **export_circfold_arrays** (int *Fc_p, int *FcH_p, int *Fcl_p, int *FcM_p, int **fM2_p, int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p)

See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

Global **export_circfold_arrays_par** (int *Fc_p, int *FcH_p, int *Fcl_p, int *FcM_p, int **fM2_p, int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p, vrna_param_t **P_p)

See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

Global **export_co_bppm** (void)

This function is deprecated and will be removed soon! The base pair probability array is available through the [vrna_fold_compound_t](#) data structure, and its associated [vrna_mx_pf_t](#) member.

Global **export_cofold_arrays** (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **fc_p, int **indx_p, char **ptype_p)

folding matrices now reside within the [vrna_fold_compound_t](#). Thus, this function will only work in conjunction with a prior call to the deprecated functions [cofold\(\)](#) or [cofold_par\(\)](#)

Global **export_cofold_arrays_ggq** (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **fc_p, int **ggg_p, int **indx_p, char **ptype_p)

folding matrices now reside within the fold compound. Thus, this function will only work in conjunction with a prior call to [cofold\(\)](#) or [cofold_par\(\)](#)

Global **export_fold_arrays** (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p)

See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

Global **export_fold_arrays_par** (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p, vrna_param_t **P_p)

See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

File **exterior_loops.h**

Use [ViennaRNA/loops/external.h](#) instead

File **file_formats.h**

Use [ViennaRNA/io/file_formats.h](#) instead

File **file_formats_msa.h**

Use [ViennaRNA/io/file_formats_msa.h](#) instead

File **file_utils.h**

Use [ViennaRNA/io/utils.h](#) instead

Global **filecopy** (FILE *from, FILE *to)

Use [vrna_file_copy\(\)](#) instead!

Global **find_saddle** (const char *seq, const char *s1, const char *s2, int width)

Use [vrna_path_findpath_saddle\(\)](#) instead!

File **findpath.h**

Use [ViennaRNA/landscape/findpath.h](#) instead

Global **fold** (const char *sequence, char *structure)

use [vrna_fold\(\)](#), or [vrna_mfe\(\)](#) instead!

Global **fold_par** (const char *sequence, char *structure, vrna_param_t *parameters, int is_constrained, int is_circular)

use [vrna_mfe\(\)](#) instead!

Global **free_alifold_arrays** (void)

Usage of this function is discouraged! It only affects memory being free'd that was allocated by an old API function before. Release of memory occupied by the newly introduced [vrna_fold_compound_t](#) is handled by [vrna_fold_compound_free\(\)](#)

Global **free_alipf_arrays** (void)

Usage of this function is discouraged! This function only free's memory allocated by old API function calls. Memory allocated by any of the new API calls (starting with vrna_) will be not affected!

Global **free_arrays** (void)

See [vrna_fold\(\)](#), [vrna_circfold\(\)](#), or [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

Global **free_co_arrays** (void)

This function will only free memory allocated by a prior call of [cofold\(\)](#) or [cofold_par\(\)](#). See [vrna_mfe_dimer\(\)](#) for how to use the new API

Global **free_co_pf_arrays** (void)

This function will be removed for the new API soon! See [vrna_pf_dimer\(\)](#), [vrna_fold_compound\(\)](#), and [vrna_fold_compound_free\(\)](#) for an alternative

Global **free_path** (vrna_path_t *path)

Use [vrna_path_free\(\)](#) instead!

Global **free_pf_arrays** (void)

See [vrna_fold_compound_t](#) and its related functions for how to free memory occupied by the dynamic programming matrices

Global **get_alipf_arrays** (short ***S_p, short ***S5_p, short ***S3_p, unsigned short ***a2s_p, char ***Ss_p, FLT_OR_DBL **qb_p, FLT_OR_DBL **qm_p, FLT_OR_DBL **q1k_p, FLT_OR_DBL **qln_p, short **pscore)

It is discouraged to use this function! The new [vrna_fold_compound_t](#) allows direct access to all necessary consensus structure prediction related variables!

Global **get_boltzmann_factor_copy** (vrna_exp_param_t *parameters)

Use [vrna_exp_params_copy\(\)](#) instead!

Global **get_boltzmann_factors** (double temperature, double betaScale, vrna_md_t md, double pf_scale)

Use [vrna_exp_params\(\)](#) instead!

Global **get_boltzmann_factors_ali** (unsigned int n_seq, double temperature, double betaScale, vrna_md_t md, double pf_scale)

Use [vrna_exp_params_comparative\(\)](#) instead!

Global **get_centroid_struct_gquad_pr** (int length, double *dist)

This function is deprecated and should not be used anymore as it is not threadsafe!

Global **get_centroid_struct_pl** (int length, double *dist, vrna_ep_t *pl)

This function was renamed to [vrna_centroid_from_plist\(\)](#)

Global **get_centroid_struct_pr** (int length, double *dist, FLT_OR_DBL *pr)

This function was renamed to [vrna_centroid_from_probs\(\)](#)

Global **get_concentrations** (double FEAB, double FEAA, double FEBB, double FEA, double FEB, double *startconc)

{ Use [vrna_pf_dimer_concentrations\(\)](#) instead! }

Global **get_line** (FILE *fp)

Use [vrna_read_line\(\)](#) as a substitute!

Global **get_mpi** (char *Alseq[], int n_seq, int length, int *mini)

Use [vrna_aln_mpi\(\)](#) as a replacement

Global **get_path** (const char *seq, const char *s1, const char *s2, int width)

Use [vrna_path_findpath\(\)](#) instead!

Global **get_plist** (vrna_ep_t *pl, int length, double cut_off)

{ This function is deprecated and will be removed soon!} use [assign_plist_from_pr\(\)](#) instead!

Global **get_scaled_alipf_parameters** (unsigned int n_seq)

Use [vrna_exp_params_comparative\(\)](#) instead!

Global **get_scaled_parameters** (double temperature, vrna_md_t md)

Use [vrna_params\(\)](#) instead!

Global **get_scaled_pf_parameters** (void)

Use [vrna_exp_params\(\)](#) instead!

Global **get_TwoDfold_variables** (const char *seq, const char *structure1, const char *structure2, int circ)

Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound↵_TwoD\(\)](#), [vrna_mfe_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

Global **get_TwoDpfold_variables** (const char *seq, const char *structure1, char *structure2, int circ)

Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound↵_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

File **hairpin_loops.h**

Use [ViennaRNA/loops/hairpin.h](#) instead

Global **HairpinE** (int size, int type, int si1, int sj1, const char *string)

{This function is deprecated and will be removed soon. Use [E_Hairpin\(\)](#) instead!}

Global **hamming** (const char *s1, const char *s2)

Use [vrna_hamming_distance\(\)](#) instead!

Global **hamming_bound** (const char *s1, const char *s2, int n)

Use [vrna_hamming_distance_bound\(\)](#) instead!

Global **iindx**

Do not use this variable anymore!

Global **init_co_pf_fold** (int length)

{ This function is deprecated and will be removed soon!}

Global **init_pf_fold** (int length)

This function is obsolete and will be removed soon!

Global **init_rand** (void)

Use [vrna_init_rand\(\)](#) instead!

Global **initialize_cofold** (int length)

{This function is obsolete and will be removed soon!}

Global **initialize_fold** (int length)

See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

Global **int_urn** (int from, int to)

Use [vrna_int_urn\(\)](#) instead!

File **interior_loops.h**

Use [ViennaRNA/loops/internal.h](#) instead

Global **Lfold** (const char *string, const char *structure, int maxdist)

Use [vrna_mfe_window\(\)](#) instead!

Global **Lfoldz** (const char *string, const char *structure, int maxdist, int zsc, double min_z)

Use [vrna_mfe_window_zscore\(\)](#) instead!

File [loop_energies.h](#)

Use [ViennaRNA/loops/all.h](#) instead

Global [loop_energy](#) (short *ptable, short *s, short *s1, int i)

Use [vrna_eval_loop_pt\(\)](#) instead!

Global [LoopEnergy](#) (int n1, int n2, int type, int type_2, int si1, int sj1, int sp1, int sq1)

{This function is deprecated and will be removed soon. Use [E_IntLoop\(\)](#) instead!}

Global [Make_bp_profile](#) (int length)

This function is deprecated and will be removed soon! See [Make_bp_profile_bppm\(\)](#) for a replacement

Global [make_pair_table](#) (const char *structure)

Use [vrna_ptable\(\)](#) instead

Global [make_pair_table_snoop](#) (const char *structure)

Use [vrna_pt_snoop_get\(\)](#) instead!

Global [make_referenceBP_array](#) (short *reference_pt, unsigned int turn)

Use [vrna_refBPcnt_matrix\(\)](#) instead

Global [MEA](#) (plist *p, char *structure, double gamma)

Use [vrna_MEA\(\)](#) or [vrna_MEA_from_plist\(\)](#) instead!

Global [mean_bp_dist](#) (int length)

This function is not threadsafe and should not be used anymore. Use [mean_bp_distance\(\)](#) instead!

Global [mean_bp_distance](#) (int length)

Use [vrna_mean_bp_distance\(\)](#) or [vrna_mean_bp_distance_pr\(\)](#) instead!

Global [mean_bp_distance_pr](#) (int length, FLT_OR_DBL *pr)

Use [vrna_mean_bp_distance\(\)](#) or [vrna_mean_bp_distance_pr\(\)](#) instead!

File [multibranch_loops.h](#)

Use [ViennaRNA/loops/multibranch.h](#) instead

File [naview.h](#)

Use [ViennaRNA/plotting/naview/naview.h](#) instead

Global [nc_fact](#)

See [vrna_md_t.nc_fact](#), and [vrna_mfe\(\)](#) to avoid using global variables

File [neighbor.h](#)

Use [ViennaRNA/landscape/neighbor.h](#) instead

Global [nerror](#) (const char message[])

Use [vrna_message_error\(\)](#) instead!

Global [pack_structure](#) (const char *struc)

Use [vrna_db_pack\(\)](#) as a replacement

Global [PAIR](#)

Use [vrna_basepair_t](#) instead!

Global [pair_info](#)

Use [vrna_pinfo_t](#) instead!

File [params.h](#)

Use [ViennaRNA/params/basic.h](#) instead

Global [paramT](#)

Use [vrna_param_t](#) instead!

Global [parenthesis_structure](#) (char *structure, vrna_bp_stack_t *bp, int length)

use [vrna_parenthesis_structure\(\)](#) instead

Global **parenthesis_zuker** (char *structure, vrna_bp_stack_t *bp, int length)

use vrna_parenthesis_zuker instead

Global **path_t**

Use vrna_path_t instead!

Global **pbacktrack_circ** (char *sequence)

Use vrna_pbacktrack() instead.

Global **pf_circ_fold** (const char *sequence, char *structure)

Use vrna_pf() instead!

Global **pf_fold_par** (const char *sequence, char *structure, vrna_exp_param_t *parameters, int calculate↵_bppm, int is_constrained, int is_circular)

Use vrna_pf() instead

Global **pf_paramT**

Use vrna_exp_param_t instead!

Global **plist**

Use vrna_ep_t or vrna_elem_prob_s instead!

File **plot_aln.h**

Use ViennaRNA/plotting/alignments.h instead

File **plot_layouts.h**

Use ViennaRNA/plotting/layouts.h instead

File **plot_structure.h**

Use ViennaRNA/plotting/structures.h instead

File **plot_utils.h**

Use ViennaRNA/plotting/utils.h instead

Global **pr**

Do not use this variable anymore!

Global **print_tty_constraint** (unsigned int option)

Use vrna_message_constraints() instead!

Global **print_tty_constraint_full** (void)

Use vrna_message_constraint_options_all() instead!

Global **print_tty_input_seq** (void)

Use vrna_message_input_seq_simple() instead!

Global **print_tty_input_seq_str** (const char *s)

Use vrna_message_input_seq() instead!

Global **PS_color_aln** (const char *structure, const char *filename, const char *seqs[], const char *names[])

Use vrna_file_PS_aln() instead!

File **PS_dot.h**

Use ViennaRNA/plotting/probabilities.h instead

Global **PS_dot_plot** (char *string, char *file)

This function is deprecated and will be removed soon! Use PS_dot_plot_list() instead!

Global **PS_rna_plot** (char *string, char *structure, char *file)

Use vrna_file_PS_rnaplot() instead!

Global **PS_rna_plot_a** (char *string, char *structure, char *file, char *pre, char *post)

Use vrna_file_PS_rnaplot_a() instead!

Global **PS_rna_plot_a_gquad** (char *string, char *structure, char *ssfile, char *pre, char *post)

Use vrna_file_PS_rnaplot_a() instead!

Global **random_string** (int l, const char symbols[])

Use [vrna_random_string\(\)](#) instead!

File **read_epars.h**

Use [ViennaRNA/params/io.h](#) instead

Global **read_parameter_file** (const char fname[])

Use [vrna_params_load\(\)](#) instead!

Global **read_record** (char **header, char **sequence, char ***rest, unsigned int options)

This function is deprecated! Use [vrna_file_fasta_read_record\(\)](#) as a replacment.

Global **scale_parameters** (void)

Use [vrna_params\(\)](#) instead!

Global **sect**

Use [vrna_sect_t](#) instead!

Global **set_model_details** (vrna_md_t *md)

This function will vanish as soon as backward compatibility of RNAlib is dropped (expected in version 3). Use [vrna_md_set_default\(\)](#) instead!

Global **simple_circplot_coordinates** (short *pair_table, float *x, float *y)

Consider switching to [vrna_plot_coords_circular_pt\(\)](#) instead!

Global **simple_xy_coordinates** (short *pair_table, float *X, float *Y)

Consider switching to [vrna_plot_coords_simple_pt\(\)](#) instead!

Global **SOLUTION**

Use [vrna_subopt_solution_t](#) instead!

Global **space** (unsigned size)

Use [vrna_alloc\(\)](#) instead!

Global **st_back**

set the *uniq_ML* flag in [vrna_md_t](#) before passing it to [vrna_fold_compound\(\)](#).

Global **stackProb** (double cutoff)

Use [vrna_stack_prob\(\)](#) instead!

Global **str_DNA2RNA** (char *sequence)

Use [vrna_seq_toRNA\(\)](#) instead!

Global **str_uppercase** (char *sequence)

Use [vrna_seq_toupper\(\)](#) instead!

File **stream_output.h**

Use [ViennaRNA/datastructures/stream_output.h](#) instead

File **string_utils.h**

Use [ViennaRNA/utls/strings.h](#) instead

File **structure_utils.h**

Use [ViennaRNA/utls/structures.h](#) instead

File **svm_utils.h**

Use [ViennaRNA/utls/svm.h](#) instead

Global **temperature**

Use [vrna_md_defaults_temperature\(\)](#), and [vrna_md_defaults_temperature_get\(\)](#) to change, and read the global default temperature settings

Global **time_stamp** (void)

Use [vrna_time_stamp\(\)](#) instead!

Global `TwoDfold_backtrack_f5` (unsigned int j, int k, int l, `TwoDfold_vars` *vars)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound↵_TwoD()`, `vrna_mfe_TwoD()`, `vrna_backtrack5_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `TwoDfold_vars`

This data structure will be removed from the library soon! Use `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `TwoDfoldList` (`TwoDfold_vars` *vars, int distance1, int distance2)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound↵_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `TwoDpfold_pbacktrack` (`TwoDpfold_vars` *vars, int d1, int d2)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound↵_TwoD()`, `vrna_pf_TwoD()`, `vrna_pbacktrack_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `TwoDpfold_pbacktrack5` (`TwoDpfold_vars` *vars, int d1, int d2, unsigned int length)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound↵_TwoD()`, `vrna_pf_TwoD()`, `vrna_pbacktrack5_TwoD()`, and `vrna_fold_compound_free()` instead!

Class `TwoDpfold_vars`

This data structure will be removed from the library soon! Use `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

Global `TwoDpfoldList` (`TwoDpfold_vars` *vars, int maxDistance1, int maxDistance2)

Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound↵_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

File `units.h`

Use `ViennaRNA/utills/units.h` instead

Global `unpack_structure` (const char *packed)

Use `vrna_db_unpack()` as a replacement

Global `update_alifold_params` (void)

Usage of this function is discouraged! The new API uses `vrna_fold_compound_t` to lump all folding related necessities together, including the energy parameters. Use `vrna_update_fold_params()` to update the energy parameters within a `vrna_fold_compound_t`.

Global `update_co_pf_params` (int length)

Use `vrna_exp_params_subst()` instead!

Global `update_co_pf_params_par` (int length, `vrna_exp_param_t` *parameters)

Use `vrna_exp_params_subst()` instead!

Global `update_cofold_params` (void)

See `vrna_params_subst()` for an alternative using the new API

Global `update_cofold_params_par` (`vrna_param_t` *parameters)

See `vrna_params_subst()` for an alternative using the new API

Global `update_fold_params` (void)

For non-default model settings use the new API with `vrna_params_subst()` and `vrna_mfe()` instead!

Global `update_fold_params_par` (`vrna_param_t` *parameters)

For non-default model settings use the new API with `vrna_params_subst()` and `vrna_mfe()` instead!

Global `update_pf_params` (int length)

Use `vrna_exp_params_subst()` instead

Global `update_pf_params_par` (int length, `vrna_exp_param_t` *parameters)

Use `vrna_exp_params_subst()` instead

Global `urn` (void)

Use `vrna_urn()` instead!

File `utils.h`

Use [ViennaRNA/utils/basic.h](#) instead

Use [ViennaRNA/utils/basic.h](#) instead

Global `vrna_cofold` (`const char *sequence, char *structure`)

This function is obsolete since [vrna_mfe\(\)](#)/[vrna_fold\(\)](#) can handle complexes multiple sequences since v2.5.0. Use [vrna_mfe\(\)](#)/[vrna_fold\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

Global `VRNA_CONSTRAINT_FILE`

Use 0 instead!

Global `VRNA_CONSTRAINT_MULTILINE`

see [vrna_extract_record_rest_structure\(\)](#)

Global `VRNA_CONSTRAINT_NO_HEADER`

This mode is not supported anymore!

Global `VRNA_CONSTRAINT_SOFT_MFE`

This flag has no meaning anymore, since constraints are now always stored!

Global `VRNA_CONSTRAINT_SOFT_PF`

Use [VRNA_OPTION_PF](#) instead!

Global `vrna_exp_param_s::id`

This attribute will be removed in version 3

Global `vrna_extract_record_rest_constraint` (`char **cstruc, const char **lines, unsigned int option`)

Use [vrna_extract_record_rest_structure\(\)](#) instead!

Global `vrna_fc_s::pscore_pf_compat`

This attribute will vanish in the future!

Global `vrna_fc_s::ptype_pf_compat`

This attribute will vanish in the future! It's meant for backward compatibility only!

Global `vrna_mfe_dimer` (`vrna_fold_compound_t *vc, char *structure`)

This function is obsolete since [vrna_mfe\(\)](#) can handle complexes multiple sequences since v2.5.0. Use [vrna_mfe\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

File `walk.h`

Use [ViennaRNA/landscape/walk.h](#) instead

Global `warn_user` (`const char message[]`)

Use [vrna_message_warning\(\)](#) instead!

Global `write_parameter_file` (`const char fname[]`)

Use [vrna_params_save\(\)](#) instead!

Global `xrealloc` (`void *p, unsigned size`)

Use [vrna_realloc\(\)](#) instead!

Global `zukersubopt` (`const char *string`)

use [vrna_zukersubopt\(\)](#) instead

Global `zukersubopt_par` (`const char *string, vrna_param_t *parameters`)

use [vrna_zukersubopt\(\)](#) instead

Chapter 12

Bug List

Module [domains_up](#)

Although the additional production rule(s) for unstructured domains as described in [Unstructured Domains](#) are always treated as 'segments possibly bound to one or more ligands', the current implementation requires that at least one ligand is bound. The default implementation already takes care of the required changes, however, upon using callback functions other than the default ones, one has to take care of this fact. Please also note, that this behavior might change in one of the next releases, such that the decomposition schemes as shown above comply with the actual implementation.

Global [VRNA_PROBS_WINDOW_STACKP](#)

Currently, this flag is a placeholder doing nothing as the corresponding implementation for stack probability computation is missing.

Global [vrna_subopt_zuker](#) ([vrna_fold_compound_t](#) *fc)

Due to resizing, any pre-existing constraints will be lost!

Chapter 13

Module Index

13.1 The RNALib API

Our library is grouped into several modules, each addressing different aspects of RNA secondary structure related problems. You can find an overview of the different groups below.

Free Energy Evaluation	139
Energy Evaluation for Individual Loops	159
Exterior Loops	389
Hairpin Loops	392
Internal Loops	395
Multibranch Loops	396
Energy Evaluation for Atomic Moves	161
Deprecated Interface for Free Energy Evaluation	162
The RNA Folding Grammar	174
Fine-tuning of the Implemented Models	175
Energy Parameters	205
Reading/Writing Energy Parameter Sets from/to File	402
Converting Energy Parameter Files	408
Extending the Folding Grammar with Additional Domains	217
Unstructured Domains	217
Structured Domains	226
Constraining the RNA Folding Grammar	226
Hard Constraints	239
Soft Constraints	248
The RNA Secondary Structure Landscape	258
Neighborhood Relation and Move Sets for Secondary Structures	337
(Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima	348
Direct Refolding Paths between two Secondary Structures	351
Folding Paths that start at a single Secondary Structure	356
Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers	598
Minimum Free Energy (MFE) Algorithms	258
Global MFE Prediction	260
Computing MFE representatives of a Distance Based Partitioning	316
Deprecated Interface for Global MFE Prediction	555
Local (sliding window) MFE Prediction	265
Deprecated Interface for Local (Sliding Window) MFE Prediction	564
Backtracking MFE structures	269
Partition Function and Equilibrium Properties	259
Global Partition Function and Equilibrium Probabilities	272
Computing Partition Functions of a Distance Based Partitioning	322
Predicting various thermodynamic properties	326
Deprecated Interface for Global Partition Function Computation	565

Local (sliding window) Partition Function and Equilibrium Probabilities	278
Deprecated Interface for Local (Sliding Window) Partition Function Computation	580
Suboptimals and Representative Structures	285
Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989	286
Suboptimal Structures within an Energy Band around the MFE	287
Random Structure Samples from the Ensemble	291
Stochastic Backtracking of Structures from Distance Based Partitioning	324
Deprecated Interface for Stochastic Backtracking	583
Compute the Structure with Maximum Expected Accuracy (MEA)	312
Compute the Centroid Structure	314
RNA-RNA Interaction	315
Partition Function for Two Hybridized Sequences	397
Partition Function for two Hybridized Sequences as a Stepwise Process	400
Classified Dynamic Programming Variants	316
Distance Based Partitioning of the Secondary Structure Space	316
Computing MFE representatives of a Distance Based Partitioning	316
Computing Partition Functions of a Distance Based Partitioning	322
Stochastic Backtracking of Structures from Distance Based Partitioning	324
Compute the Density of States	335
Inverse Folding (Design)	336
Experimental Structure Probing Data	360
SHAPE Reactivity Data	360
Generate Soft Constraints from Data	363
Ligands Binding to RNA Structures	367
Ligands Binding to Unstructured Domains	367
Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints	367
Structure Modules and Pseudoknots	369
Pseudoknots	369
G-Quadruplexes	374
Post-transcriptional Modifications	376
Utilities	383
Utilities to deal with Nucleotide Alphabets	412
(Nucleic Acid Sequence) String Utilitites	415
Secondary Structure Utilities	426
Dot-Bracket Notation of Secondary Structures	428
Washington University Secondary Structure (WUSS) notation	434
Pair Table Representation of Secondary Structures	436
Pair List Representation of Secondary Structures	439
Abstract Shapes Representation of Secondary Structures	440
Helix List Representation of Secondary Structures	442
Tree Representation of Secondary Structures	443
Distance measures between Secondary Structures	447
Deprecated Interface for Secondary Structure Utilities	588
Multiple Sequence Alignment Utilities	449
Deprecated Interface for Multiple Sequence Alignment Utilities	585
Files and I/O	455
Nucleic Acid Sequences and Structures	458
Multiple Sequence Alignments	464
Command Files	471
Plotting	474
Layouts and Coordinates	480
Annotation	493
Alignment Plots	494
Deprecated Interface for Plotting Utilities	596
Search Algorithms	496

Combinatorics Algorithms	498
(Abstract) Data Structures	505
The Fold Compound	517
The Dynamic Programming Matrices	530
Hash Tables	534
Heaps	541
Arrays	548
Buffers	550
Messages	510
Unit Conversion	513

Chapter 14

Data Structure Index

14.1 Data Structures

Here are the data structures with brief descriptions:

_struct_en	
Data structure for energy_of_move()	601
energy_corrections	601
LIST	601
LST_BUCKET	601
Postorder_list	
Postorder data structure	601
swString	
Some other data structure	602
Tree	
Tree data structure	602
TwoDpfold_vars	
Variables compound for 2Dfold partition function folding	602
vrna_dimer_conc_s	
Data structure for concentration dependency computations	603
vrna_sc_bp_storage_t	
A base pair constraint	603
vrna_sc_mod_param_s	603
vrna_string_header_s	
The header of an array	603
vrna_structured_domains_s	604
vrna_subopt_sol_s	
Solution element from subopt.c	604
vrna_unstructured_domain_motif_s	604

Chapter 15

File Index

15.1 File List

Here is a list of all documented files with brief descriptions:

ViennaRNA/2Dfold.h	
MFE structures for base pair distance classes	605
ViennaRNA/2Dpfold.h	
Partition function implementations for base pair distance classes	607
ViennaRNA/ali_plex.h	613
ViennaRNA/alifold.h	
Functions for comparative structure prediction using RNA sequence alignments	613
ViennaRNA/aln_util.h	
Use ViennaRNA/utis/alignments.h instead	617
ViennaRNA/alphabet.h	
Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets	617
ViennaRNA/boltzmann_sampling.h	
Boltzmann Sampling of secondary structures from the ensemble	619
ViennaRNA/centroid.h	
Centroid structure computation	622
ViennaRNA/char_stream.h	
Use ViennaRNA/datastructures/char_stream.h instead	624
ViennaRNA/cofold.h	
MFE implementations for RNA-RNA interaction	627
ViennaRNA/combinatorics.h	
Various implementations that deal with combinatorial aspects of objects	628
ViennaRNA/commands.h	
Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation	630
ViennaRNA/concentrations.h	
Concentration computations for RNA-RNA interactions	631
ViennaRNA/constraints.h	
Use ViennaRNA/constraints/basic.h instead	633
ViennaRNA/constraints_hard.h	
Use ViennaRNA/constraints/hard.h instead	651
ViennaRNA/constraints_ligand.h	
Use ViennaRNA/constraints/ligand.h instead	651
ViennaRNA/constraints_SHAPE.h	
Use ViennaRNA/constraints/SHAPE.h instead	651
ViennaRNA/constraints_soft.h	
Use ViennaRNA/constraints/soft.h instead	652
ViennaRNA/convert_epars.h	
Use ViennaRNA/params/convert.h instead	652

ViennaRNA/data_structures.h	
Use ViennaRNA/datastructures/basic.h instead	653
ViennaRNA/dist_vars.h	
Global variables for Distance-Package	660
ViennaRNA/dp_matrices.h	
Functions to deal with standard dynamic programming (DP) matrices	662
ViennaRNA/duplex.h	
Functions for simple RNA-RNA duplex interactions	666
ViennaRNA/edit_cost.h	
Global variables for Edit Costs included by treedist.c and stringdist.c	666
ViennaRNA/energy_const.h	
Use ViennaRNA/params/constants.h instead	667
ViennaRNA/energy_par.h	
Use ViennaRNA/params/default.h instead	668
ViennaRNA/equilibrium_probs.h	
Equilibrium Probability implementations	668
ViennaRNA/eval.h	
Functions and variables related to energy evaluation of sequence/structure pairs	670
ViennaRNA/exterior_loops.h	
Use ViennaRNA/loops/external.h instead	677
ViennaRNA/file_formats.h	
Use ViennaRNA/io/file_formats.h instead	677
ViennaRNA/file_formats_msa.h	
Use ViennaRNA/io/file_formats_msa.h instead	680
ViennaRNA/file_utils.h	
Use ViennaRNA/io/utils.h instead	682
ViennaRNA/findpath.h	
Use ViennaRNA/landscape/findpath.h instead	682
ViennaRNA/fold.h	
MFE calculations for single RNA sequences	684
ViennaRNA/fold_compound.h	
The Basic Fold Compound API	687
ViennaRNA/fold_vars.h	
Here all all declarations of the global variables used throughout RNAlib	690
ViennaRNA/gquad.h	
G-quadruplexes	692
ViennaRNA/grammar.h	
Implementations for the RNA folding grammar	712
ViennaRNA/hairpin_loops.h	
Use ViennaRNA/loops/hairpin.h instead	714
ViennaRNA/heat_capacity.h	
Compute heat capacity for an RNA	714
ViennaRNA/interior_loops.h	
Use ViennaRNA/loops/internal.h instead	716
ViennaRNA/inverse.h	
Inverse folding routines	716
ViennaRNA/Lfold.h	
Functions for locally optimal MFE structure prediction	720
ViennaRNA/loop_energies.h	
Use ViennaRNA/loops/all.h instead	721
ViennaRNA/LPfold.h	
Partition function and equilibrium probability implementation for the sliding window algorithm	739
ViennaRNA/MEA.h	
Computes a MEA (maximum expected accuracy) structure	741
ViennaRNA/mfe.h	
Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data	742

ViennaRNA/mfe_window.h	
Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures	744
ViennaRNA/mm.h	
Several Maximum Matching implementations	746
ViennaRNA/model.h	
The model details data structure and its corresponding modifiers	747
ViennaRNA/move_set.h	757
ViennaRNA/multibranch_loops.h	
Use ViennaRNA/loops/multibranch.h instead	758
ViennaRNA/naview.h	
Use ViennaRNA/plotting/naview/naview.h instead	758
ViennaRNA/neighbor.h	
Use ViennaRNA/landscape/neighbor.h instead	760
ViennaRNA/pair_mat.h	761
ViennaRNA/params.h	
Use ViennaRNA/params/basic.h instead	763
ViennaRNA/part_func.h	
Partition function implementations	1224
ViennaRNA/part_func_co.h	
Partition function for two RNA sequences	1229
ViennaRNA/part_func_up.h	
Implementations for accessibility and RNA-RNA interaction as a stepwise process	1231
ViennaRNA/part_func_window.h	
Partition function and equilibrium probability implementation for the sliding window algorithm	1232
ViennaRNA/perturbation_fold.h	
Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments	1235
ViennaRNA/pf_multifold.h	1236
ViennaRNA/pk_plex.h	
Heuristics for two-step pseudoknot forming interaction predictions	1236
ViennaRNA/PKplex.h	1238
ViennaRNA/plex.h	1239
ViennaRNA/plot_aln.h	
Use ViennaRNA/plotting/alignments.h instead	1239
ViennaRNA/plot_layouts.h	
Use ViennaRNA/plotting/layouts.h instead	1240
ViennaRNA/plot_structure.h	
Use ViennaRNA/plotting/structures.h instead	1240
ViennaRNA/plot_utils.h	
Use ViennaRNA/plotting/utils.h instead	1241
ViennaRNA/ProfileAln.h	1262
ViennaRNA/profiledist.h	1262
ViennaRNA/PS_dot.h	
Use ViennaRNA/plotting/probabilities.h instead	1264
ViennaRNA/read_epars.h	
Use ViennaRNA/params/io.h instead	1264
ViennaRNA/ribo.h	
Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments	1265
ViennaRNA/RNAstruct.h	
Parsing and Coarse Graining of Structures	1265
ViennaRNA/sequence.h	
Functions and data structures related to sequence representations ,	1268
ViennaRNA/snofold.h	1270
ViennaRNA/snoop.h	1271
ViennaRNA/special_const.h	1274
ViennaRNA/stream_output.h	
Use ViennaRNA/datastructures/stream_output.h instead	1275

ViennaRNA/string_utils.h	
Use ViennaRNA/utls/strings.h instead	1276
ViennaRNA/stringdist.h	
Functions for String Alignment	1276
ViennaRNA/structure_utils.h	
Use ViennaRNA/utls/structures.h instead	1277
ViennaRNA/structured_domains.h	
This module provides interfaces that deal with additional structured domains in the folding grammar	1278
ViennaRNA/subopt.h	
RNAsubopt and density of states declarations	1278
ViennaRNA/subopt_zuker.h	1280
ViennaRNA/svm_utils.h	
Use ViennaRNA/utls/svm.h instead	1281
ViennaRNA/treedist.h	
Functions for Tree Edit Distances	1281
ViennaRNA/units.h	
Use ViennaRNA/utls/units.h instead	1283
ViennaRNA/unstructured_domains.h	
Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches	1284
ViennaRNA/utls.h	
Use ViennaRNA/utls/basic.h instead	1290
ViennaRNA/vrna_config.h	1296
ViennaRNA/walk.h	
Use ViennaRNA/landscape/walk.h instead	1299
ViennaRNA/wrap_dlib.h	1299
ViennaRNA/zscore.h	1299
ViennaRNA/constraints/basic.h	
Functions and data structures for constraining secondary structure predictions and evaluation	915
ViennaRNA/constraints/hard.h	
Functions and data structures for handling of secondary structure hard constraints	634
ViennaRNA/constraints/ligand.h	
Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework	641
ViennaRNA/constraints/sc_cb_intern.h	642
ViennaRNA/constraints/SHAPE.h	
This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints	643
ViennaRNA/constraints/soft.h	
Functions and data structures for secondary structure soft constraints	645
ViennaRNA/constraints/soft_special.h	
Specialized implementations that utilize the soft constraint callback mechanism	649
ViennaRNA/datastructures/array.h	
A macro-based dynamic array implementation	653
ViennaRNA/datastructures/basic.h	
Various data structures and pre-processor macros	917
ViennaRNA/datastructures/char_stream.h	
Implementation of a dynamic, buffered character stream	624
ViennaRNA/datastructures/hash_tables.h	
Implementations of hash table functions	655
ViennaRNA/datastructures/heap.h	
Implementation of an abstract heap data structure	657
ViennaRNA/datastructures/lists.h	659
ViennaRNA/datastructures/stream_output.h	
An implementation of a buffered, ordered stream output data structure	1274
ViennaRNA/datastructures/string.h	660

ViennaRNA/io/file_formats.h	
Read and write different file formats for RNA sequences, structures	678
ViennaRNA/io/file_formats_msa.h	
Functions dealing with file formats for Multiple Sequence Alignments (MSA)	680
ViennaRNA/io/utils.h	
Several utilities for file handling	1289
ViennaRNA/landscape/findpath.h	
A breadth-first search heuristic for optimal direct folding paths	683
ViennaRNA/landscape/move.h	
Methods to operate with structural neighbors of RNA secondary structures	717
ViennaRNA/landscape/neighbor.h	
Methods to compute the neighbors of an RNA secondary structure	759
ViennaRNA/landscape/paths.h	
API for computing (optimal) (re-)folding paths between secondary structures	719
ViennaRNA/landscape/walk.h	
Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence	1297
ViennaRNA/loops/all.h	
Energy evaluation for MFE and partition function calculations	722
ViennaRNA/loops/external.h	
Energy evaluation of exterior loops for MFE and partition function calculations	722
ViennaRNA/loops/hairpin.h	
Energy evaluation of hairpin loops for MFE and partition function calculations	725
ViennaRNA/loops/internal.h	
Energy evaluation of interior loops for MFE and partition function calculations	728
ViennaRNA/loops/multibranch.h	
Energy evaluation of multibranch loops for MFE and partition function calculations	736
ViennaRNA/params/1.8.4_epars.h	
Free energy parameters for parameter file conversion	763
ViennaRNA/params/1.8.4_intloops.h	
Free energy parameters for interior loop contributions needed by the parameter file conversion functions	768
ViennaRNA/params/basic.h	
Functions to deal with sets of energy parameters	921
ViennaRNA/params/constants.h	
Energy parameter constants	932
ViennaRNA/params/convert.h	
Functions and definitions for energy parameter file format conversion	933
ViennaRNA/params/default.h	
.	935
ViennaRNA/params/intl11.h	
.	936
ViennaRNA/params/intl11dH.h	
.	940
ViennaRNA/params/intl21.h	
.	945
ViennaRNA/params/intl21dH.h	
.	968
ViennaRNA/params/intl22.h	
.	991
ViennaRNA/params/intl22dH.h	
.	1106
ViennaRNA/params/io.h	
Read and write energy parameter files	1221
ViennaRNA/params/salt.h	
Functions to compute salt correction	1223
ViennaRNA/plotting/alignments.h	
Various functions for plotting Sequence / Structure Alignments	1241
ViennaRNA/plotting/layouts.h	
Secondary structure plot layout algorithms	1246
ViennaRNA/plotting/probabilities.h	
Various functions for plotting RNA secondary structures, dot-plots and other visualizations	1249
ViennaRNA/plotting/structures.h	
Various functions for plotting RNA secondary structures	1253

ViennaRNA/plotting/ utils.h	
Various utilities to assist in plotting secondary structures and consensus structures	1290
ViennaRNA/plotting/RNAPuzzler/ RNAPuzzler.h	
Implementation of the RNAPuzzler RNA secondary structure layout algorithm [30]	1251
ViennaRNA/plotting/RNAPuzzler/ RNAturtle.h	
Implementation of the RNAturtle RNA secondary structure layout algorithm [30]	1252
ViennaRNA/search/ BoyerMoore.h	
Variants of the Boyer-Moore string search algorithm	1267
ViennaRNA/utills/ alignments.h	
Various utility- and helper-functions for sequence alignments and comparative structure prediction	1242
ViennaRNA/utills/ basic.h	
General utility- and helper-functions used throughout the <i>ViennaRNA Package</i>	925
ViennaRNA/utills/ cpu.h	1291
ViennaRNA/utills/ higher_order_functions.h	1291
ViennaRNA/utills/ strings.h	
General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package	1291
ViennaRNA/utills/ structures.h	
Various utility- and helper-functions for secondary structure parsing, converting, etc	1254
ViennaRNA/utills/ svm.h	1296
ViennaRNA/utills/ units.h	
Physical Units and Functions to convert them into each other	1283

Chapter 16

Module Documentation

16.1 Free Energy Evaluation

Functions and variables related to free energy evaluation of sequence/structure pairs.

16.1.1 Detailed Description

Functions and variables related to free energy evaluation of sequence/structure pairs.

Several different functions to evaluate the free energy of a particular secondary structure under a particular set of parameters and the Nearest Neighbor Energy model are available. For most of them, two different forms of representations for the secondary structure may be used:

- The Dot-Bracket string
- A pair table representation

Furthermore, the evaluation functions are divided into `basic` and `simplified` variants, where `basic` functions require the use of a `vrna_fold_compound_t` data structure holding the sequence string, and model configuration (settings and parameters). The `simplified` functions, on the other hand, provide often used default model settings that may be called directly with only sequence and structure data.

Finally, `verbose` options exist for some functions that allow one to print the (individual) free energy contributions to some `FILE` stream. Collaboration diagram for Free Energy Evaluation:

Modules

- [Energy Evaluation for Individual Loops](#)
Functions to evaluate the free energy of particular types of loops.
- [Energy Evaluation for Atomic Moves](#)
Functions to evaluate the free energy change of a structure after application of (a set of) atomic moves.
- [Deprecated Interface for Free Energy Evaluation](#)
Deprecated Energy Evaluation functions.

Files

- file [eval.h](#)
Functions and variables related to energy evaluation of sequence/structure pairs.
- file [all.h](#)
Energy evaluation for MFE and partition function calculations.
- file [external.h](#)
Energy evaluation of exterior loops for MFE and partition function calculations.
- file [hairpin.h](#)
Energy evaluation of hairpin loops for MFE and partition function calculations.
- file [internal.h](#)

Energy evaluation of interior loops for MFE and partition function calculations.

- file [multibranch.h](#)

Energy evaluation of multibranch loops for MFE and partition function calculations.

Macros

- `#define VRNA_VERBOSE_QUIET -1`
Quiet level verbosity setting.
- `#define VRNA_VERBOSE_DEFAULT 1`
Default level verbosity setting.

Basic Energy Evaluation Interface with Dot-Bracket Structure String

- float [vrna_eval_structure](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Calculate the free energy of an already folded RNA.
- float [vrna_eval_covar_structure](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Calculate the pseudo energy derived by the covariance scores of a set of aligned sequences.
- float [vrna_eval_structure_verbose](#) ([vrna_fold_compound_t](#) *fc, const char *structure, FILE *file)
Calculate the free energy of an already folded RNA and print contributions on a per-loop base.
- float [vrna_eval_structure_v](#) ([vrna_fold_compound_t](#) *fc, const char *structure, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA and print contributions on a per-loop base.
- float [vrna_eval_structure_cstr](#) ([vrna_fold_compound_t](#) *fc, const char *structure, int verbosity_level, [vrna_cstr_t](#) output_stream)

Basic Energy Evaluation Interface with Structure Pair Table

- int [vrna_eval_structure_pt](#) ([vrna_fold_compound_t](#) *fc, const short *pt)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_verbose](#) ([vrna_fold_compound_t](#) *fc, const short *pt, FILE *file)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_v](#) ([vrna_fold_compound_t](#) *fc, const short *pt, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA.

Simplified Energy Evaluation with Sequence and Dot-Bracket Strings

- float [vrna_eval_structure_simple](#) (const char *string, const char *structure)
Calculate the free energy of an already folded RNA.
- float [vrna_eval_circ_structure](#) (const char *string, const char *structure)
Evaluate the free energy of a sequence/structure pair where the sequence is circular.
- float [vrna_eval_gquad_structure](#) (const char *string, const char *structure)
Evaluate the free energy of a sequence/structure pair where the structure may contain G-Quadruplexes.
- float [vrna_eval_circ_gquad_structure](#) (const char *string, const char *structure)
Evaluate the free energy of a sequence/structure pair where the sequence is circular and the structure may contain G-Quadruplexes.
- float [vrna_eval_structure_simple_verbose](#) (const char *string, const char *structure, FILE *file)
Calculate the free energy of an already folded RNA and print contributions per loop.
- float [vrna_eval_structure_simple_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA and print contributions per loop.
- float [vrna_eval_circ_structure_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Evaluate free energy of a sequence/structure pair, assume sequence to be circular and print contributions per loop.
- float [vrna_eval_gquad_structure_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Evaluate free energy of a sequence/structure pair, allow for G-Quadruplexes in the structure and print contributions per loop.

- float [vrna_eval_circ_gquad_structure_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Evaluate free energy of a sequence/structure pair, assume sequence to be circular, allow for G-Quadruplexes in the structure, and print contributions per loop.

Simplified Energy Evaluation with Sequence Alignments and Consensus Structure Dot-Bracket String

- float [vrna_eval_consensus_structure_simple](#) (const char **alignment, const char *structure)
Calculate the free energy of an already folded RNA sequence alignment.
- float [vrna_eval_circ_consensus_structure](#) (const char **alignment, const char *structure)
Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequences are circular.
- float [vrna_eval_gquad_consensus_structure](#) (const char **alignment, const char *structure)
Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the structure may contain G-Quadruplexes.
- float [vrna_eval_circ_gquad_consensus_structure](#) (const char **alignment, const char *structure)
Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequence is circular and the structure may contain G-Quadruplexes.
- float [vrna_eval_consensus_structure_simple_verbose](#) (const char **alignment, const char *structure, FILE *file)
Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.
- float [vrna_eval_consensus_structure_simple_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.
- float [vrna_eval_circ_consensus_structure_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences and print contributions per loop.
- float [vrna_eval_gquad_consensus_structure_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an RNA sequence alignment, allow for annotated G-Quadruplexes in the structure and print contributions per loop.
- float [vrna_eval_circ_gquad_consensus_structure_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences, allow for annotated G-Quadruplexes in the structure and print contributions per loop.

Simplified Energy Evaluation with Sequence String and Structure Pair Table

- int [vrna_eval_structure_pt_simple](#) (const char *string, const short *pt)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_simple_verbose](#) (const char *string, const short *pt, FILE *file)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_simple_v](#) (const char *string, const short *pt, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA.

Simplified Energy Evaluation with Sequence Alignment and Consensus Structure Pair Table

- int [vrna_eval_consensus_structure_pt_simple](#) (const char **alignment, const short *pt)
Evaluate the Free Energy of a Consensus Secondary Structure given a Sequence Alignment.
- int [vrna_eval_consensus_structure_pt_simple_verbose](#) (const char **alignment, const short *pt, FILE *file)
- int [vrna_eval_consensus_structure_pt_simple_v](#) (const char **alignment, const short *pt, int verbosity_level, FILE *file)

16.1.2 Function Documentation

16.1.2.1 `vrna_eval_structure()`

```
float vrna_eval_structure (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given pair of structure and sequence (alignment). Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not need to contain any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

Note

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE` and `VRNA_FC_TYPE_COMPARATIVE`

See also

[vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_eval_covar_structure\(\)](#)

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>structure</i>	Secondary structure in dot-bracket notation

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is attached as method `eval_structure()` to objects of type `fold_compound`

16.1.2.2 `vrna_eval_covar_structure()`

```
float vrna_eval_covar_structure (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Calculate the pseudo energy derived by the covariance scores of a set of aligned sequences.

Consensus structure prediction is driven by covariance scores of base pairs in rows of the provided alignment. This function allows one to retrieve the total amount of this covariance pseudo energy scores. The `vrna_fold_compound_t` does not need to contain any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound_comparative(alignment, NULL, VRNA_OPTION_EVAL_ONLY);
```

Note

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_COMPARATIVE` only!

See also

[vrna_fold_compound_comparative\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>structure</i>	Secondary (consensus) structure in dot-bracket notation

Returns

The covariance pseudo energy score of the input structure given the input sequence alignment in kcal/mol

SWIG Wrapper Notes This function is attached as method `eval_covar_structure()` to objects of type `fold_compound`

16.1.2.3 `vrna_eval_structure_verbose()`

```
float vrna_eval_structure_verbose (
    vrna_fold_compound_t * fc,
    const char * structure,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions on a per-loop base.

This function is a simplified version of `vrna_eval_structure_v()` that uses the *default* verbosity level.

See also

`vrna_eval_structure_pt()`, `vrna_eval_structure_verbose()`, `vrna_eval_structure_pt_verbose()`,

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>structure</i>	Secondary structure in dot-bracket notation
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is attached as method `eval_structure_verbose()` to objects of type `fold_compound`

16.1.2.4 `vrna_eval_structure_v()`

```
float vrna_eval_structure_v (
    vrna_fold_compound_t * fc,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions on a per-loop base.

This function allows for detailed energy evaluation of a given sequence/structure pair. In contrast to `vrna_eval_structure()` this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive `verbosity_level` activates potential warning message of the energy evaluating functions, while values ≥ 1 allow for detailed control of what data is printed. A negative parameter `verbosity_level` turns off printing all together.

Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The `fold_compound` does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

See also

[vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#),

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>structure</i>	Secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

16.1.2.5 vrna_eval_structure_pt()

```
int vrna_eval_structure_pt (
    vrna_fold_compound_t * fc,
    const short * pt )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair where the structure is provided in `pair_table` format as obtained from [vrna_ptable\(\)](#). Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The `fold_compound` does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

See also

[vrna_ptable\(\)](#), [vrna_eval_structure\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#)

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>pt</i>	Secondary structure as <code>pair_table</code>

Returns

The free energy of the input structure given the input sequence in 10cal/mol

SWIG Wrapper Notes This function is attached as method `eval_structure_pt()` to objects of type `fold_compound`

16.1.2.6 vrna_eval_structure_pt_verbose()

```
int vrna_eval_structure_pt_verbose (
    vrna_fold_compound_t * fc,
    const short * pt,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function is a simplified version of `vrna_eval_structure_simple_v()` that uses the *default* verbosity level.

See also

`vrna_eval_structure_pt_v()`, `vrna_ptable()`, `vrna_eval_structure_pt()`, `vrna_eval_structure_verbose()`

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>pt</i>	Secondary structure as <code>pair_table</code>
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in 10cal/mol

SWIG Wrapper Notes This function is attached as method `eval_structure_pt_verbose()` to objects of type `fold_compound`

16.1.2.7 vrna_eval_structure_pt_v()

```
int vrna_eval_structure_pt_v (
    vrna_fold_compound_t * fc,
    const short * pt,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair where the structure is provided in `pair_table` format as obtained from `vrna_ptable()`. Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The `fold_compound` does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

In contrast to `vrna_eval_structure_pt()` this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive `verbosity_level` activates potential warning message of the energy evaluating functions, while values ≥ 1 allow for detailed control of what data is printed. A negative parameter `verbosity_level` turns off printing all together.

See also

`vrna_ptable()`, `vrna_eval_structure_pt()`, `vrna_eval_structure_verbose()`

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>pt</i>	Secondary structure as <code>pair_table</code>
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in 10cal/mol

16.1.2.8 `vrna_eval_structure_simple()`

```
int vrna_eval_structure_simple (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair. In contrast to `vrna_eval_structure()` this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

See also

[vrna_eval_structure\(\)](#), [vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#),

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes In the target scripting language, this function serves as a wrapper for `vrna_eval_structure_simple_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

16.1.2.9 `vrna_eval_circ_structure()`

```
int vrna_eval_circ_structure (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a sequence/structure pair where the sequence is circular.

See also

[vrna_eval_structure_simple\(\)](#), [vrna_eval_gquad_structure\(\)](#), [vrna_eval_circ_consensus_structure\(\)](#), [vrna_eval_circ_structure_v\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation

Returns

The free energy of the structure given the circular input sequence in kcal/mol

SWIG Wrapper Notes In the target scripting language, this function serves as a wrapper for `vrna_eval_circ_structure_v()` and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to `VRNA_VERBOSITY_QUIET` and `NULL`, respectively.

16.1.2.10 vrna_eval_gquad_structure()

```
int vrna_eval_gquad_structure (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a sequence/structure pair where the structure may contain G-Quadruplexes.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..++...++..++
```

See also

[vrna_eval_structure_simple\(\)](#), [vrna_eval_circ_structure\(\)](#), [vrna_eval_gquad_consensus_structure\(\)](#), [vrna_eval_gquad_structure_v\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation

Returns

The free energy of the structure including contributions of G-quadruplexes in kcal/mol

SWIG Wrapper Notes In the target scripting language, this function serves as a wrapper for [vrna_eval_gquad_structure_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.11 vrna_eval_circ_gquad_structure()

```
int vrna_eval_circ_gquad_structure (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a sequence/structure pair where the sequence is circular and the structure may contain G-Quadruplexes.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots ('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..++...++..++
```

See also

[vrna_eval_structure_simple\(\)](#), [vrna_eval_circ_gquad_consensus_structure\(\)](#), [vrna_eval_circ_gquad_structure_v\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation

Returns

The free energy of the structure including contributions of G-quadruplexes in kcal/mol

SWIG Wrapper Notes In the target scripting language, this function serves as a wrapper for [vrna_eval_circ_gquad_structure_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.12 vrna_eval_structure_simple_verbose()

```
int vrna_eval_structure_simple_verbose (
    const char * string,
    const char * structure,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions per loop.

This function is a simplified version of [vrna_eval_structure_simple_v\(\)](#) that uses the *default* verbosity level.

See also

[vrna_eval_structure_simple_v\(\)](#), [vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_verbose_pt\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is not available. Use [vrna_eval_structure_simple_v\(\)](#) instead!

16.1.2.13 vrna_eval_structure_simple_v()

```
int vrna_eval_structure_simple_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA and print contributions per loop.

This function allows for detailed energy evaluation of a given sequence/structure pair. In contrast to [vrna_eval_structure\(\)](#) this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive `verbosity_level` activates potential warning message of the energy evaluating functions, while values ≥ 1 allow for detailed control of what data is printed. A negative parameter `verbosity_level` turns off printing all together.

In contrast to [vrna_eval_structure_verbose\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

See also

[vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#),

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_structure_simple\(\)](#). The last two arguments for this function are optional and default to [VRNA_VERBOSITY_QUIET](#) and `NULL`, respectively.

16.1.2.14 vrna_eval_circ_structure_v()

```
int vrna_eval_circ_structure_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate free energy of a sequence/structure pair, assume sequence to be circular and print contributions per loop. This function is the same as [vrna_eval_structure_simple_v\(\)](#) but assumes the input sequence to be circularized.

See also

[vrna_eval_structure_simple_v\(\)](#), [vrna_eval_circ_structure\(\)](#), [vrna_eval_structure_verbose\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_circ_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA_VERBOSITY_QUIET](#) and `NULL`, respectively.

16.1.2.15 vrna_eval_gquad_structure_v()

```
int vrna_eval_gquad_structure_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate free energy of a sequence/structure pair, allow for G-Quadruplexes in the structure and print contributions per loop.

This function is the same as [vrna_eval_structure_simple_v\(\)](#) but allows for annotated G-Quadruplexes in the dot-bracket structure input.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..++..++..++
```

See also

[vrna_eval_structure_simple_v\(\)](#), [vrna_eval_gquad_structure\(\)](#), [vrna_eval_structure_verbose\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_gquad_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.16 vrna_eval_circ_gquad_structure_v()

```
int vrna_eval_circ_gquad_structure_v (
    const char * string,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate free energy of a sequence/structure pair, assume sequence to be circular, allow for G-Quadruplexes in the structure, and print contributions per loop.

This function is the same as [vrna_eval_structure_simple_v\(\)](#) but assumes the input sequence to be circular and allows for annotated G-Quadruplexes in the dot-bracket structure input.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..++..++..++
```

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_circ_gquad_structure\(\)](#). The last two arguments for this function are optional and default to [VRNA_VERBOSITY_QUIET](#) and `NULL`, respectively.

16.1.2.17 vrna_eval_consensus_structure_simple()

```
int vrna_eval_consensus_structure_simple (
    const char ** alignment,
    const char * structure )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA sequence alignment.

This function allows for energy evaluation for a given multiple sequence alignment and consensus structure pair. In contrast to [vrna_eval_structure\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_covar_structure\(\)](#), [vrna_eval_structure\(\)](#), [vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters and hyphen ('-') to denote gaps
<i>structure</i>	Consensus Secondary structure in dot-bracket notation

Returns

The free energy of the consensus structure given the input alignment in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_structure_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

16.1.2.18 vrna_eval_circ_consensus_structure()

```
int vrna_eval_circ_consensus_structure (
    const char ** alignment,
    const char * structure )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequences are circular.

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_covar_structure\(\)](#), [vrna_eval_consensus_structure_simple\(\)](#), [vrna_eval_gquad_consensus_structure\(\)](#), [vrna_eval_circ_structure\(\)](#), [vrna_eval_circ_consensus_structure_v\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters
<i>structure</i>	Consensus secondary structure in dot-bracket notation

Returns

The free energy of the consensus structure given the circular input sequence in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_circ_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

16.1.2.19 vrna_eval_gquad_consensus_structure()

```
int vrna_eval_gquad_consensus_structure (
    const char ** alignment,
    const char * structure )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the structure may contain G-Quadruplexes.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..+.....++..++
```

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_covar_structure\(\)](#), [vrna_eval_consensus_structure_simple\(\)](#), [vrna_eval_circ_consensus_structure\(\)](#), [vrna_eval_gquad_structure\(\)](#), [vrna_eval_gquad_consensus_structure_v\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters
<i>structure</i>	Consensus secondary structure in dot-bracket notation

Returns

The free energy of the consensus structure including contributions of G-quadruplexes in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_gquad_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

16.1.2.20 vrna_eval_circ_gquad_consensus_structure()

```
int vrna_eval_circ_gquad_consensus_structure (
```

```

    const char ** alignment,
    const char * structure )
#include <ViennaRNA/eval.h>

```

Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequence is circular and the structure may contain G-Quadruplexes.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```

GGAAGGAAAGGAGG
++..++.....++..++

```

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_covar_structure\(\)](#), [vrna_eval_consensus_structure_simple\(\)](#), [vrna_eval_circ_consensus_structure\(\)](#), [vrna_eval_gquad_structure\(\)](#), [vrna_eval_circ_gquad_consensus_structure_v\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters
<i>structure</i>	Consensus secondary structure in dot-bracket notation

Returns

The free energy of the consensus structure including contributions of G-quadruplexes in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_circ_gquad_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

16.1.2.21 vrna_eval_consensus_structure_simple_verbose()

```

int vrna_eval_consensus_structure_simple_verbose (
    const char ** alignment,
    const char * structure,
    FILE * file )

```

```

#include <ViennaRNA/eval.h>

```

Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop. This function is a simplified version of [vrna_eval_consensus_structure_simple_v\(\)](#) that uses the *default* verbosity level.

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_consensus_structure_simple_v\(\)](#), [vrna_eval_structure_verbose\(\)](#), [vrna_eval_structure_pt\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-')
<i>structure</i>	Consensus secondary structure in dot-bracket notation
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the consensus structure given the aligned input sequences in kcal/mol

SWIG Wrapper Notes This function is not available. Use [vrna_eval_consensus_structure_simple_v\(\)](#) instead!

16.1.2.22 vrna_eval_consensus_structure_simple_v()

```
int vrna_eval_consensus_structure_simple_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop. This function allows for detailed energy evaluation of a given sequence alignment/consensus structure pair. In contrast to [vrna_eval_consensus_structure_simple\(\)](#) this function prints detailed energy contributions based on individual loops to a file handle. If NULL is passed as file handle, this function defaults to print to stdout. Any positive `verbosity_level` activates potential warning message of the energy evaluating functions, while values ≥ 1 allow for detailed control of what data is printed. A negative parameter `verbosity_level` turns off printing all together.

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_consensus_structure\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-')
<i>structure</i>	Consensus secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the consensus structure given the sequence alignment in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_structure_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.23 vrna_eval_circ_consensus_structure_v()

```
int vrna_eval_circ_consensus_structure_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences and print contributions per loop.

This function is identical with [vrna_eval_consensus_structure_simple_v\(\)](#) but assumed the aligned sequences to be circular.

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_consensus_structure_simple_v\(\)](#), [vrna_eval_circ_consensus_structure\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-')
<i>structure</i>	Consensus secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the consensus structure given the sequence alignment in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_circ_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSE_QUIET](#) and NULL, respectively.

16.1.2.24 vrna_eval_gquad_consensus_structure_v()

```
int vrna_eval_gquad_consensus_structure_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an RNA sequence alignment, allow for annotated G- \leftrightarrow Quadruplexes in the structure and print contributions per loop.

This function is identical with [vrna_eval_consensus_structure_simple_v\(\)](#) but allows for annotated G-Quadruplexes in the consensus structure.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..++...++..++
```

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_consensus_structure_simple_v\(\)](#), [vrna_eval_gquad_consensus_structure\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-')
<i>structure</i>	Consensus secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the consensus structure given the sequence alignment in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_gquad_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.25 vrna_eval_circ_gquad_consensus_structure_v()

```
int vrna_eval_circ_gquad_consensus_structure_v (
    const char ** alignment,
    const char * structure,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences, allow for annotated G-Quadruplexes in the structure and print contributions per loop.

This function is identical with [vrna_eval_consensus_structure_simple_v\(\)](#) but assumes the sequences in the alignment to be circular and allows for annotated G-Quadruplexes in the consensus structure.

G-Quadruplexes are annotated as plus signs ('+') for each G involved in the motif. Linker sequences must be denoted by dots('.') as they are considered unpaired. Below is an example of a 2-layer G-quadruplex:

```
GGAAGGAAAGGAGG
++..++..++..++
```

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_consensus_structure_simple_v\(\)](#), [vrna_eval_circ_gquad_consensus_structure\(\)](#), [vrna_eval_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-')
<i>structure</i>	Consensus secondary structure in dot-bracket notation
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the consensus structure given the sequence alignment in kcal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_circ_gquad_structure\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.26 vrna_eval_structure_pt_simple()

```
int vrna_eval_structure_pt_simple (
    const char * string,
    const short * pt )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

In contrast to [vrna_eval_structure_pt\(\)](#) this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

See also

[vrna_ptable\(\)](#), [vrna_eval_structure_simple\(\)](#), [vrna_eval_structure_pt\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>pt</i>	Secondary structure as pair_table

Returns

The free energy of the input structure given the input sequence in 10cal/mol

SWIG Wrapper Notes In the target scripting language, this function serves as a wrapper for [vrna_eval_structure_pt_v\(\)](#) and, thus, allows for two additional, optional arguments, the verbosity level and a file handle which default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.1.2.27 vrna_eval_structure_pt_simple_verbose()

```
int vrna_eval_structure_pt_simple_verbose (
    const char * string,
    const short * pt,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function is a simplified version of [vrna_eval_structure_pt_simple_v\(\)](#) that uses the *default* verbosity level.

See also

[vrna_eval_structure_pt_simple_v\(\)](#), [vrna_ptable\(\)](#), [vrna_eval_structure_pt_verbose\(\)](#), [vrna_eval_structure_simple\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>pt</i>	Secondary structure as pair_table
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in 10cal/mol

16.1.2.28 `vrna_eval_structure_pt_simple_v()`

```
int vrna_eval_structure_pt_simple_v (
    const char * string,
    const short * pt,
    int verbosity_level,
    FILE * file )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

This function allows for energy evaluation of a given sequence/structure pair where the structure is provided in pair_table format as obtained from `vrna_ptable()`. Model details, energy parameters, and possibly soft constraints are used as provided via the parameter 'fc'. The fold_compound does not need to contain any DP matrices, but all the most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_EVAL_ONLY);
```

In contrast to `vrna_eval_structure_pt_verbose()` this function assumes default model details and default energy parameters in order to evaluate the free energy of the secondary structure. Therefore, it serves as a simple interface function for energy evaluation for situations where no changes on the energy model are required.

See also

[vrna_ptable\(\)](#), [vrna_eval_structure_pt_v\(\)](#), [vrna_eval_structure_simple\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>pt</i>	Secondary structure as pair_table
<i>verbosity_level</i>	The level of verbosity of this function
<i>file</i>	A file handle where this function should print to (may be NULL).

Returns

The free energy of the input structure given the input sequence in 10cal/mol

16.1.2.29 `vrna_eval_consensus_structure_pt_simple()`

```
int vrna_eval_consensus_structure_pt_simple (
    const char ** alignment,
    const short * pt )
```

```
#include <ViennaRNA/eval.h>
```

Evaluate the Free Energy of a Consensus Secondary Structure given a Sequence Alignment.

Note

The free energy returned from this function already includes the covariation pseudo energies that is used for comparative structure prediction within this library.

See also

[vrna_eval_consensus_structure_simple\(\)](#), [vrna_eval_structure_pt\(\)](#), [vrna_eval_structure\(\)](#), [vrna_eval_covar_structure\(\)](#)

Parameters

<i>alignment</i>	RNA sequence alignment in uppercase letters. Gaps are denoted by hyphens ('-')
<i>pt</i>	Secondary structure in pair table format

Returns

Free energy of the consensus structure in 10cal/mol

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_structure_pt_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument

16.1.2.30 vrna_eval_consensus_structure_pt_simple_verbose()

```
int vrna_eval_consensus_structure_pt_simple_verbose (
    const char ** alignment,
    const short * pt,
    FILE * file )
#include <ViennaRNA/eval.h>
```

SWIG Wrapper Notes This function is not available. Use [vrna_eval_consensus_structure_pt_v\(\)](#) instead!

16.1.2.31 vrna_eval_consensus_structure_pt_simple_v()

```
int vrna_eval_consensus_structure_pt_simple_v (
    const char ** alignment,
    const short * pt,
    int verbosity_level,
    FILE * file )
#include <ViennaRNA/eval.h>
```

SWIG Wrapper Notes This function is available through an overloaded version of [vrna_eval_structure_pt_simple\(\)](#). Simply pass a sequence alignment as list of strings (including gaps) as first, and the consensus structure as second argument. The last two arguments are optional and default to [VRNA_VERBOSITY_QUIET](#) and NULL, respectively.

16.2 Energy Evaluation for Individual Loops

Functions to evaluate the free energy of particular types of loops.

16.2.1 Detailed Description

Functions to evaluate the free energy of particular types of loops.

To assess the free energy contribution of a particular loop within a secondary structure, two variants are provided:

- The bare free energy E (usually in deka-calories, i.e. multiples of 10cal/mol), and
- The Boltzmann weight $q = \exp(-\beta E)$ of the free energy E (with $\beta = \frac{1}{RT}$, gas constant R and temperature T)

The latter is usually required for partition function computations. Collaboration diagram for Energy Evaluation for Individual Loops:

Modules

- [Exterior Loops](#)
Functions to evaluate the free energy contributions for exterior loops.
- [Hairpin Loops](#)
Functions to evaluate the free energy contributions for hairpin loops.
- [Internal Loops](#)
Functions to evaluate the free energy contributions for internal loops.
- [Multibranch Loops](#)
Functions to evaluate the free energy contributions for multibranch loops.

Files

- file [all.h](#)
Energy evaluation for MFE and partition function calculations.
- file [external.h](#)
Energy evaluation of exterior loops for MFE and partition function calculations.
- file [hairpin.h](#)
Energy evaluation of hairpin loops for MFE and partition function calculations.
- file [internal.h](#)
Energy evaluation of interior loops for MFE and partition function calculations.
- file [multibranch.h](#)
Energy evaluation of multibranch loops for MFE and partition function calculations.

Functions

- int [vrna_eval_loop_pt](#) ([vrna_fold_compound_t](#) *fc, int i, const short *pt)
Calculate energy of a loop.
- int [vrna_eval_loop_pt_v](#) ([vrna_fold_compound_t](#) *fc, int i, const short *pt, int verbosity_level)
Calculate energy of a loop.

16.2.2 Function Documentation

16.2.2.1 vrna_eval_loop_pt()

```
int vrna_eval_loop_pt (
    vrna_fold_compound_t * fc,
    int i,
    const short * pt )
#include <ViennaRNA/eval.h>
Calculate energy of a loop.
```

Parameters

<i>fc</i>	A vrna_fold_compound_t containing the energy parameters and model details
<i>i</i>	position of covering base pair
<i>pt</i>	the pair table of the secondary structure

Returns

free energy of the loop in 10cal/mol

SWIG Wrapper Notes This function is attached as method [eval_loop_pt\(\)](#) to objects of type *fold_compound*

16.2.2.2 vrna_eval_loop_pt_v()

```
int vrna_eval_loop_pt_v (
    vrna_fold_compound_t * fc,
    int i,
    const short * pt,
    int verbosity_level )
#include <ViennaRNA/eval.h>
Calculate energy of a loop.
```

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>i</i>	position of covering base pair
<i>pt</i>	the pair table of the secondary structure
<i>verbosity_level</i>	The level of verbosity of this function

Returns

free energy of the loop in 10cal/mol

16.3 Energy Evaluation for Atomic Moves

Functions to evaluate the free energy change of a structure after application of (a set of) atomic moves.

16.3.1 Detailed Description

Functions to evaluate the free energy change of a structure after application of (a set of) atomic moves.

Here, atomic moves are not to be confused with moves of actual physical atoms. Instead, an atomic move is considered the smallest conformational change a secondary structure can undergo to form another, distinguishable structure. We currently support the following moves

Atomic Moves:

- Opening (dissociation) of a single base pair
- Closing (formation) of a single base pair
- Shifting one pairing partner of an existing pair to a different location

Collaboration diagram for Energy Evaluation for Atomic Moves:

Functions

- float [vrna_eval_move](#) ([vrna_fold_compound_t](#) *fc, const char *structure, int m1, int m2)
Calculate energy of a move (closing or opening of a base pair)
- int [vrna_eval_move_pt](#) ([vrna_fold_compound_t](#) *fc, short *pt, int m1, int m2)
Calculate energy of a move (closing or opening of a base pair)

16.3.2 Function Documentation

16.3.2.1 vrna_eval_move()

```
float vrna_eval_move (
    vrna_fold_compound_t * fc,
    const char * structure,
    int m1,
    int m2 )
```

```
#include <ViennaRNA/eval.h>
```

Calculate energy of a move (closing or opening of a base pair)

If the parameters m1 and m2 are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).

See also

[vrna_eval_move_pt\(\)](#)

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>structure</i>	secondary structure in dot-bracket notation
<i>m1</i>	first coordinate of base pair
<i>m2</i>	second coordinate of base pair

Returns

energy change of the move in kcal/mol (INF / 100. upon any error)

SWIG Wrapper Notes This function is attached as method `eval_move()` to objects of type *fold_compound*

16.3.2.2 vrna_eval_move_pt()

```
int vrna_eval_move_pt (
    vrna_fold_compound_t * fc,
    short * pt,
    int m1,
    int m2 )
```

```
#include <ViennaRNA/eval.h>
```

Calculate energy of a move (closing or opening of a base pair)

If the parameters *m1* and *m2* are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).

See also

[vrna_eval_move\(\)](#)

Parameters

<i>fc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<i>pt</i>	the pair table of the secondary structure
<i>m1</i>	first coordinate of base pair
<i>m2</i>	second coordinate of base pair

Returns

energy change of the move in 10cal/mol

SWIG Wrapper Notes This function is attached as method `eval_move_pt()` to objects of type *fold_compound*

16.4 Deprecated Interface for Free Energy Evaluation

Deprecated Energy Evaluation functions.

16.4.1 Detailed Description

Deprecated Energy Evaluation functions.

Using the functions below is discouraged as they have been marked deprecated and will be removed from the library in the (near) future! Collaboration diagram for Deprecated Interface for Free Energy Evaluation:

Functions

- float [energy_of_structure](#) (const char *string, const char *structure, int verbosity_level)

Calculate the free energy of an already folded RNA using global model detail settings.

- float `energy_of_struct_par` (const char *string, const char *structure, `vrna_param_t` *parameters, int verbosity_level)

Calculate the free energy of an already folded RNA.

- float `energy_of_circ_structure` (const char *string, const char *structure, int verbosity_level)

Calculate the free energy of an already folded circular RNA.

- float `energy_of_circ_struct_par` (const char *string, const char *structure, `vrna_param_t` *parameters, int verbosity_level)

Calculate the free energy of an already folded circular RNA.

- int `energy_of_structure_pt` (const char *string, short *ptable, short *s, short *s1, int verbosity_level)

Calculate the free energy of an already folded RNA.

- int `energy_of_struct_pt_par` (const char *string, short *ptable, short *s, short *s1, `vrna_param_t` *parameters, int verbosity_level)

Calculate the free energy of an already folded RNA.

- float `energy_of_move` (const char *string, const char *structure, int m1, int m2)

Calculate energy of a move (closing or opening of a base pair)

- int `energy_of_move_pt` (short *pt, short *s, short *s1, int m1, int m2)

Calculate energy of a move (closing or opening of a base pair)

- int `loop_energy` (short *ptable, short *s, short *s1, int i)

Calculate energy of a loop.

- float `energy_of_struct` (const char *string, const char *structure)
- int `energy_of_struct_pt` (const char *string, short *ptable, short *s, short *s1)
- float `energy_of_circ_struct` (const char *string, const char *structure)
- int `E_Stem` (int type, int si1, int sj1, int extLoop, `vrna_param_t` *P)

Compute the energy contribution of a stem branching off a loop-region.

- `FLT_OR_DBL exp_E_ExtLoop` (int type, int si1, int sj1, `vrna_exp_param_t` *P)
- `FLT_OR_DBL exp_E_Stem` (int type, int si1, int sj1, int extLoop, `vrna_exp_param_t` *P)
- PRIVATE int `E_IntLoop` (int n1, int n2, int type, int type_2, int si1, int sj1, int sp1, int sq1, `vrna_param_t` *P)
- PRIVATE `FLT_OR_DBL exp_E_IntLoop` (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1, `vrna_exp_param_t` *P)

Variables

- int `cut_point`
first pos of second seq for cofolding
- int `eos_debug`
verbose info from energy_of_struct

16.4.2 Function Documentation

16.4.2.1 energy_of_structure()

```
float energy_of_structure (
    const char * string,
    const char * structure,
    int verbosity_level )
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA using global model detail settings.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

Note

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [energy_of_struct_par\(\)](#) for a completely threadsafe implementation.

Deprecated Use [vrna_eval_structure\(\)](#) or [vrna_eval_structure_verbose\(\)](#) instead!

See also

[vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence
<i>structure</i>	secondary structure in dot-bracket notation
<i>verbosity_level</i>	a flag to turn verbose output on/off

Returns

the free energy of the input structure given the input sequence in kcal/mol

16.4.2.2 energy_of_struct_par()

```
float energy_of_struct_par (
    const char * string,
    const char * structure,
    vrna_param_t * parameters,
    int verbosity_level )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

Deprecated Use [vrna_eval_structure\(\)](#) or [vrna_eval_structure_verbose\(\)](#) instead!

See also

[vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>structure</i>	Secondary structure in dot-bracket notation
<i>parameters</i>	A data structure containing the prescaled energy contributions and the model details.
<i>verbosity_level</i>	A flag to turn verbose output on/off

Returns

The free energy of the input structure given the input sequence in kcal/mol

16.4.2.3 energy_of_circ_structure()

```
float energy_of_circ_structure (
    const char * string,
```

```
const char * structure,
int verbosity_level )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded circular RNA.
```

Note

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [energy_of_circ_struct_par\(\)](#) for a completely threadsafe implementation.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

Deprecated Use [vrna_eval_structure\(\)](#) or [vrna_eval_structure_verbose\(\)](#) instead!

See also

[vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence
<i>structure</i>	Secondary structure in dot-bracket notation
<i>verbosity_level</i>	A flag to turn verbose output on/off

Returns

The free energy of the input structure given the input sequence in kcal/mol

16.4.2.4 energy_of_circ_struct_par()

```
float energy_of_circ_struct_par (
    const char * string,
    const char * structure,
    vrna_param_t * parameters,
    int verbosity_level )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded circular RNA.
If verbosity level is set to a value >0, energies of structure elements are printed to stdout
```

Deprecated Use [vrna_eval_structure\(\)](#) or [vrna_eval_structure_verbose\(\)](#) instead!

See also

[vrna_eval_structure\(\)](#)

Parameters

<i>string</i>	RNA sequence
<i>structure</i>	Secondary structure in dot-bracket notation
<i>parameters</i>	A data structure containing the prescaled energy contributions and the model details.
<i>verbosity_level</i>	A flag to turn verbose output on/off

Returns

The free energy of the input structure given the input sequence in kcal/mol

16.4.2.5 energy_of_structure_pt()

```
int energy_of_structure_pt (
    const char * string,
    short * ptable,
    short * s,
    short * s1,
    int verbosity_level )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

Note

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [energy_of_struct_pt_par\(\)](#) for a completely threadsafe implementation.

Deprecated Use [vrna_eval_structure_pt\(\)](#) or [vrna_eval_structure_pt_verbose\(\)](#) instead!

See also

[vrna_eval_structure_pt\(\)](#)

Parameters

<i>string</i>	RNA sequence
<i>ptable</i>	the pair table of the secondary structure
<i>s</i>	encoded RNA sequence
<i>s1</i>	encoded RNA sequence
<i>verbosity_level</i>	a flag to turn verbose output on/off

Returns

the free energy of the input structure given the input sequence in 10kcal/mol

16.4.2.6 energy_of_struct_pt_par()

```
int energy_of_struct_pt_par (
    const char * string,
    short * ptable,
    short * s,
    short * s1,
    vrna_param_t * parameters,
    int verbosity_level )
```

```
#include <ViennaRNA/eval.h>
```

Calculate the free energy of an already folded RNA.

If verbosity level is set to a value >0, energies of structure elements are printed to stdout

Deprecated Use [vrna_eval_structure_pt\(\)](#) or [vrna_eval_structure_pt_verbose\(\)](#) instead!

See also

[vrna_eval_structure_pt\(\)](#)

Parameters

<i>string</i>	RNA sequence in uppercase letters
<i>ptable</i>	The pair table of the secondary structure
<i>s</i>	Encoded RNA sequence
<i>s1</i>	Encoded RNA sequence
<i>parameters</i>	A data structure containing the prescaled energy contributions and the model details.
<i>verbosity_level</i>	A flag to turn verbose output on/off

Returns

The free energy of the input structure given the input sequence in 10kcal/mol

16.4.2.7 energy_of_move()

```
float energy_of_move (
    const char * string,
    const char * structure,
    int m1,
    int m2 )
```

#include <[ViennaRNA/eval.h](#)>

Calculate energy of a move (closing or opening of a base pair)

If the parameters m1 and m2 are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).

Deprecated Use [vrna_eval_move\(\)](#) instead!

See also

[vrna_eval_move\(\)](#)

Parameters

<i>string</i>	RNA sequence
<i>structure</i>	secondary structure in dot-bracket notation
<i>m1</i>	first coordinate of base pair
<i>m2</i>	second coordinate of base pair

Returns

energy change of the move in kcal/mol

16.4.2.8 energy_of_move_pt()

```
int energy_of_move_pt (
    short * pt,
    short * s,
    short * s1,
    int m1,
    int m2 )
```

```
#include <ViennaRNA/eval.h>
```

Calculate energy of a move (closing or opening of a base pair)

If the parameters *m1* and *m2* are negative, it is deletion (opening) of a base pair, otherwise it is insertion (opening).

Deprecated Use `vrna_eval_move_pt()` instead!

See also

[vrna_eval_move_pt\(\)](#)

Parameters

<i>pt</i>	the pair table of the secondary structure
<i>s</i>	encoded RNA sequence
<i>s1</i>	encoded RNA sequence
<i>m1</i>	first coordinate of base pair
<i>m2</i>	second coordinate of base pair

Returns

energy change of the move in 10cal/mol

16.4.2.9 loop_energy()

```
int loop_energy (
    short * ptable,
    short * s,
    short * s1,
    int i )
```

```
#include <ViennaRNA/eval.h>
```

Calculate energy of a loop.

Deprecated Use `vrna_eval_loop_pt()` instead!

See also

[vrna_eval_loop_pt\(\)](#)

Parameters

<i>ptable</i>	the pair table of the secondary structure
<i>s</i>	encoded RNA sequence
<i>s1</i>	encoded RNA sequence
<i>i</i>	position of covering base pair

Returns

free energy of the loop in 10cal/mol

16.4.2.10 energy_of_struct()

```
float energy_of_struct (
```

```

    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded RNA

```

Note

This function is not entirely threadsafe! Depending on the state of the global variable `eos_debug` it prints energy information to stdout or not...

Deprecated This function is deprecated and should not be used in future programs! Use `energy_of_structure()` instead!

See also

`energy_of_structure`, `energy_of_circ_struct()`, `energy_of_struct_pt()`

Parameters

<i>string</i>	RNA sequence
<i>structure</i>	secondary structure in dot-bracket notation

Returns

the free energy of the input structure given the input sequence in kcal/mol

16.4.2.11 energy_of_struct_pt()

```

int energy_of_struct_pt (
    const char * string,
    short * ptable,
    short * s,
    short * s1 )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded RNA

```

Note

This function is not entirely threadsafe! Depending on the state of the global variable `eos_debug` it prints energy information to stdout or not...

Deprecated This function is deprecated and should not be used in future programs! Use `energy_of_structure_pt()` instead!

See also

`make_pair_table()`, `energy_of_structure()`

Parameters

<i>string</i>	RNA sequence
<i>ptable</i>	the pair table of the secondary structure
<i>s</i>	encoded RNA sequence
<i>s1</i>	encoded RNA sequence

Returns

the free energy of the input structure given the input sequence in 10kcal/mol

16.4.2.12 energy_of_circ_struct()

```
float energy_of_circ_struct (
    const char * string,
    const char * structure )
#include <ViennaRNA/eval.h>
Calculate the free energy of an already folded circular RNA
```

Note

This function is not entirely threadsafe! Depending on the state of the global variable `eos_debug` it prints energy information to stdout or not...

Deprecated This function is deprecated and should not be used in future programs Use `energy_of_circ_structure()` instead!

See also

[energy_of_circ_structure\(\)](#), [energy_of_struct\(\)](#), [energy_of_struct_pt\(\)](#)

Parameters

<i>string</i>	RNA sequence
<i>structure</i>	secondary structure in dot-bracket notation

Returns

the free energy of the input structure given the input sequence in kcal/mol

16.4.2.13 E_Stem()

```
int E_Stem (
    int type,
    int si1,
    int sj1,
    int extLoop,
    vrna_param_t * P )
#include <ViennaRNA/loops/external.h>
Compute the energy contribution of a stem branching off a loop-region.
```

This function computes the energy contribution of a stem that branches off a loop region. This can be the case in multiloops, when a stem branching off increases the degree of the loop but also *immediately interior base pairs* of an exterior loop contribute free energy. To switch the behavior of the function according to the evaluation of a multiloop- or exterior-loop-stem, you pass the flag 'extLoop'. The returned energy contribution consists of a TerminalAU penalty if the pair type is greater than 2, dangling end contributions of mismatching nucleotides adjacent to the stem if only one of the si1, sj1 parameters is greater than 0 and mismatch energies if both mismatching nucleotides are positive values. Thus, to avoid incorporating dangling end or mismatch energies just pass a negative number, e.g. -1 to the mismatch argument.

This is an illustration of how the energy contribution is assembled:

```

      3'   5'
      |   |
      X - Y
5'-si1      sj1-3'
```

Here, (X,Y) is the base pair that closes the stem that branches off a loop region. The nucleotides *si1* and *sj1* are the 5'- and 3'- mismatches, respectively. If the base pair type of (X,Y) is greater than 2 (i.e. an A-U or G-U pair, the TerminalAU penalty will be included in the energy contribution returned. If *si1* and *sj1* are both nonnegative numbers, mismatch energies will also be included. If one of *si1* or *sj1* is a negative value, only 5' or 3' dangling end contributions are taken into account. To prohibit any of these mismatch contributions to be incorporated, just pass a negative number to both, *si1* and *sj1*. In case the argument *extLoop* is 0, the returned energy contribution also includes the *internal-loop-penalty* of a multiloop stem with closing pair type.

See also

`E_MLstem()`
`E_ExtLoop()`

Note

This function is threadsafe

Deprecated Please use one of the functions `vrna_E_ext_stem()` and `E_MLstem()` instead! Use the former for cases where *extLoop* != 0 and the latter otherwise.

Parameters

<i>type</i>	The pair type of the first base pair un the stem
<i>si1</i>	The 5'-mismatching nucleotide
<i>sj1</i>	The 3'-mismatching nucleotide
<i>extLoop</i>	A flag that indicates whether the contribution reflects the one of an exterior loop or not
<i>P</i>	The data structure containing scaled energy parameters

Returns

The Free energy of the branch off the loop in dcal/mol

16.4.2.14 exp_E_ExtLoop()

```
FLT_OR_DBL exp_E_ExtLoop (
    int type,
    int si1,
    int sj1,
    vrna_exp_param_t * P )
```

```
#include <ViennaRNA/loops/external.h>
```

This is the partition function variant of `E_ExtLoop()`

Deprecated Use `vrna_exp_E_ext_stem()` instead!

See also

`E_ExtLoop()`

Returns

The Boltzmann weighted energy contribution of the introduced exterior-loop stem

16.4.2.15 exp_E_Stem()

```
FLT_OR_DBL exp_E_Stem (
    int type,
    int sil,
    int sjl,
    int extLoop,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/external.h>
```

Compute the Boltzmann weighted energy contribution of a stem branching off a loop-region

This is the partition function variant of [E_Stem\(\)](#)

See also

[E_Stem\(\)](#)

Note

This function is threadsafe

Returns

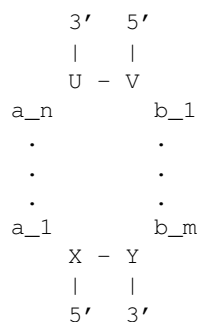
The Boltzmann weighted energy contribution of the branch off the loop

16.4.2.16 E_IntLoop()

```
PRIVATE int E_IntLoop (
    int n1,
    int n2,
    int type,
    int type_2,
    int sil,
    int sjl,
    int spl,
    int sq1,
    vrna_param_t * P )
#include <ViennaRNA/loops/internal.h>
```

Compute the Energy of an interior-loop

This function computes the free energy ΔG of an interior-loop with the following structure:



This general structure depicts an interior-loop that is closed by the base pair (X,Y). The enclosed base pair is (V,U) which leaves the unpaired bases a_1-a_n and b_1-b_n that constitute the loop. In this example, the length of the interior-loop is $(n + m)$ where n or m may be 0 resulting in a bulge-loop or base pair stack. The mismatching

nucleotides for the closing pair (X,Y) are:

5'-mismatch: `a_1`

3'-mismatch: `b_m`

and for the enclosed base pair (V,U):

5'-mismatch: `b_1`

3'-mismatch: `a_n`

Note

Base pairs are always denoted in 5'->3' direction. Thus the enclosed base pair must be 'turned around' when evaluating the free energy of the interior-loop

See also

[scale_parameters\(\)](#)

[vrna_param_t](#)

Note

This function is threadsafe

Parameters

<i>n1</i>	The size of the 'left'-loop (number of unpaired nucleotides)
<i>n2</i>	The size of the 'right'-loop (number of unpaired nucleotides)
<i>type</i>	The pair type of the base pair closing the interior loop
<i>type</i> _↔ <i>_2</i>	The pair type of the enclosed base pair
<i>si1</i>	The 5'-mismatching nucleotide of the closing pair
<i>sj1</i>	The 3'-mismatching nucleotide of the closing pair
<i>sp1</i>	The 3'-mismatching nucleotide of the enclosed pair
<i>sq1</i>	The 5'-mismatching nucleotide of the enclosed pair
<i>P</i>	The datastructure containing scaled energy parameters

Returns

The Free energy of the Interior-loop in dcal/mol

16.4.2.17 exp_E_IntLoop()

```
PRIVATE FLT_OR_DBL exp_E_IntLoop (
    int u1,
    int u2,
    int type,
    int type2,
    short si1,
    short sj1,
    short sp1,
    short sq1,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/internal.h>
```

Compute Boltzmann weight $e^{-\Delta G/kT}$ of interior loop

multiply by `scale[u1+u2+2]` for scaling

See also

[get_scaled_pf_parameters\(\)](#)
[vrna_exp_param_t](#)
[E_IntLoop\(\)](#)

Note

This function is threadsafe

Parameters

<i>u1</i>	The size of the 'left'-loop (number of unpaired nucleotides)
<i>u2</i>	The size of the 'right'-loop (number of unpaired nucleotides)
<i>type</i>	The pair type of the base pair closing the interior loop
<i>type2</i>	The pair type of the enclosed base pair
<i>si1</i>	The 5'-mismatching nucleotide of the closing pair
<i>sj1</i>	The 3'-mismatching nucleotide of the closing pair
<i>sp1</i>	The 3'-mismatching nucleotide of the enclosed pair
<i>sq1</i>	The 5'-mismatching nucleotide of the enclosed pair
<i>P</i>	The datastructure containing scaled Boltzmann weights of the energy parameters

Returns

The Boltzmann weight of the Interior-loop

16.5 The RNA Folding Grammar

The RNA folding grammar as implemented in RNAlib.

16.5.1 Detailed Description

The RNA folding grammar as implemented in RNAlib.

Collaboration diagram for The RNA Folding Grammar:

Modules

- [Fine-tuning of the Implemented Models](#)
Functions and data structures to fine-tune the implemented secondary structure evaluation model.
- [Energy Parameters](#)
All relevant functions to retrieve and copy pre-calculated energy parameter sets as well as reading/writing the energy parameter set from/to file(s).
- [Extending the Folding Grammar with Additional Domains](#)
This module covers simple and straight-forward extensions to the RNA folding grammar.
- [Constraining the RNA Folding Grammar](#)
This module provides general functions that allow for an easy control of constrained secondary structure prediction and evaluation.

Files

- file [grammar.h](#)
Implementations for the RNA folding grammar.

Data Structures

- struct [vrna_gr_aux_s](#)

Typedefs

- typedef void(* [vrna_grammar_data_free_f](#)) (void *data)
Free auxiliary data.

16.5.2 Data Structure Documentation

16.5.2.1 struct vrna_gr_aux_s

Collaboration diagram for vrna_gr_aux_s:

Data Fields

- vrna_grammar_cond_f **cb_proc**
A callback for pre- and post-processing of auxiliary grammar rules.

16.5.3 Typedef Documentation

16.5.3.1 vrna_grammar_data_free_f

```
typedef void(* vrna_grammar_data_free_f) (void *data)
#include <ViennaRNA/grammar.h>
Free auxiliary data.
```

Parameters

<i>data</i>	The auxiliary data to be free'd
-------------	---------------------------------

16.6 Fine-tuning of the Implemented Models

Functions and data structures to fine-tune the implemented secondary structure evaluation model.

16.6.1 Detailed Description

Functions and data structures to fine-tune the implemented secondary structure evaluation model.
Collaboration diagram for Fine-tuning of the Implemented Models:

Files

- file [model.h](#)
The model details data structure and its corresponding modifiers.

Data Structures

- struct [vrna_md_s](#)
The data structure that contains the complete model details used throughout the calculations. [More...](#)

Macros

- #define [VRNA_MODEL_DEFAULT_TEMPERATURE](#) 37.0

- Default temperature for structure prediction and free energy evaluation in °C*

 - #define `VRNA_MODEL_DEFAULT_PF_SCALE` -1
- Default scaling factor for partition function computations.*

 - #define `VRNA_MODEL_DEFAULT_BETA_SCALE` 1.
- Default scaling factor for absolute thermodynamic temperature in Boltzmann factors.*

 - #define `VRNA_MODEL_DEFAULT_DANGLES` 2
- Default dangling end model.*

 - #define `VRNA_MODEL_DEFAULT_SPECIAL_HP` 1
- Default model behavior for lookup of special tri-, tetra-, and hexa-loops.*

 - #define `VRNA_MODEL_DEFAULT_NO_LP` 0
- Default model behavior for so-called 'lonely pairs'.*

 - #define `VRNA_MODEL_DEFAULT_NO_GU` 0
- Default model behavior for G-U base pairs.*

 - #define `VRNA_MODEL_DEFAULT_NO_GU_CLOSURE` 0
- Default model behavior for G-U base pairs closing a loop.*

 - #define `VRNA_MODEL_DEFAULT_CIRC` 0
- Default model behavior to treat a molecule as a circular RNA (DNA)*

 - #define `VRNA_MODEL_DEFAULT_GQUAD` 0
- Default model behavior regarding the treatment of G-Quadruplexes.*

 - #define `VRNA_MODEL_DEFAULT_UNIQ_ML` 0
- Default behavior of the model regarding unique multi-branch loop decomposition.*

 - #define `VRNA_MODEL_DEFAULT_ENERGY_SET` 0
- Default model behavior on which energy set to use.*

 - #define `VRNA_MODEL_DEFAULT_BACKTRACK` 1
- Default model behavior with regards to backtracking of structures.*

 - #define `VRNA_MODEL_DEFAULT_BACKTRACK_TYPE` 'F'
- Default model behavior on what type of backtracking to perform.*

 - #define `VRNA_MODEL_DEFAULT_COMPUTE_BPP` 1
- Default model behavior with regards to computing base pair probabilities.*

 - #define `VRNA_MODEL_DEFAULT_MAX_BP_SPAN` -1
- Default model behavior for the allowed maximum base pair span.*

 - #define `VRNA_MODEL_DEFAULT_WINDOW_SIZE` -1
- Default model behavior for the sliding window approach.*

 - #define `VRNA_MODEL_DEFAULT_LOG_ML` 0
- Default model behavior on how to evaluate the energy contribution of multi-branch loops.*

 - #define `VRNA_MODEL_DEFAULT_ALI_OLD_EN` 0
- Default model behavior for consensus structure energy evaluation.*

 - #define `VRNA_MODEL_DEFAULT_ALI_RIBO` 0
- Default model behavior for consensus structure co-variance contribution assessment.*

 - #define `VRNA_MODEL_DEFAULT_ALI_CV_FACT` 1.
- Default model behavior for weighting the co-variance score in consensus structure prediction.*

 - #define `VRNA_MODEL_DEFAULT_ALI_NC_FACT` 1.
- Default model behavior for weighting the nucleotide conservation? in consensus structure prediction.*

 - #define `VRNA_MODEL_DEFAULT_SALT` 1.021
- Default model salt concentration (M)*

 - #define `VRNA_MODEL_DEFAULT_SALTMLLOWER` 6
- Default model lower bound of multiloop size for salt correction fitting.*

 - #define `VRNA_MODEL_DEFAULT_SALTMLUPPER` 24
- Default model upper bound of multiloop size for salt correction fitting.*

 - #define `VRNA_MODEL_DEFAULT_SALTDPIXINIT` 99999
- Default model value to turn off user-provided salt correction for duplex initialization.*

 - #define `MAXALPHA` 20
- Maximal length of alphabet.*

Typedefs

- typedef struct [vrna_md_s](#) [vrna_md_t](#)
Typename for the model details data structure [vrna_md_s](#).

Functions

- void [vrna_md_set_default](#) ([vrna_md_t](#) *md)
Apply default model details to a provided [vrna_md_t](#) data structure.
- void [vrna_md_update](#) ([vrna_md_t](#) *md)
Update the model details data structure.
- [vrna_md_t](#) * [vrna_md_copy](#) ([vrna_md_t](#) *md_to, const [vrna_md_t](#) *md_from)
Copy/Clone a [vrna_md_t](#) model.
- char * [vrna_md_option_string](#) ([vrna_md_t](#) *md)
Get a corresponding commandline parameter string of the options in a [vrna_md_t](#).
- void [vrna_md_defaults_reset](#) ([vrna_md_t](#) *md_p)
Reset the global default model details to a specific set of parameters, or their initial values.
- void [vrna_md_defaults_temperature](#) (double T)
Set default temperature for energy evaluation of loops.
- double [vrna_md_defaults_temperature_get](#) (void)
Get default temperature for energy evaluation of loops.
- void [vrna_md_defaults_betaScale](#) (double b)
Set default scaling factor of thermodynamic temperature in Boltzmann factors.
- double [vrna_md_defaults_betaScale_get](#) (void)
Get default scaling factor of thermodynamic temperature in Boltzmann factors.
- void [vrna_md_defaults_dangles](#) (int d)
Set default dangle model for structure prediction.
- int [vrna_md_defaults_dangles_get](#) (void)
Get default dangle model for structure prediction.
- void [vrna_md_defaults_special_hp](#) (int flag)
Set default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
- int [vrna_md_defaults_special_hp_get](#) (void)
Get default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
- void [vrna_md_defaults_noLP](#) (int flag)
Set default behavior for prediction of canonical secondary structures.
- int [vrna_md_defaults_noLP_get](#) (void)
Get default behavior for prediction of canonical secondary structures.
- void [vrna_md_defaults_noGU](#) (int flag)
Set default behavior for treatment of G-U wobble pairs.
- int [vrna_md_defaults_noGU_get](#) (void)
Get default behavior for treatment of G-U wobble pairs.
- void [vrna_md_defaults_noGUclosure](#) (int flag)
Set default behavior for G-U pairs as closing pair for loops.
- int [vrna_md_defaults_noGUclosure_get](#) (void)
Get default behavior for G-U pairs as closing pair for loops.
- void [vrna_md_defaults_logML](#) (int flag)
Set default behavior recomputing free energies of multi-branch loops using a logarithmic model.
- int [vrna_md_defaults_logML_get](#) (void)
Get default behavior recomputing free energies of multi-branch loops using a logarithmic model.
- void [vrna_md_defaults_circ](#) (int flag)
Set default behavior whether input sequences are circularized.
- int [vrna_md_defaults_circ_get](#) (void)

- Get default behavior whether input sequences are circularized.*

 - void [vrna_md_defaults_gquad](#) (int flag)
- Set default behavior for treatment of G-Quadruplexes.*

 - int [vrna_md_defaults_gquad_get](#) (void)
- Get default behavior for treatment of G-Quadruplexes.*

 - void [vrna_md_defaults_uniq_ML](#) (int flag)
- Set default behavior for creating additional matrix for unique multi-branch loop prediction.*

 - int [vrna_md_defaults_uniq_ML_get](#) (void)
- Get default behavior for creating additional matrix for unique multi-branch loop prediction.*

 - void [vrna_md_defaults_energy_set](#) (int e)
- Set default energy set.*

 - int [vrna_md_defaults_energy_set_get](#) (void)
- Get default energy set.*

 - void [vrna_md_defaults_backtrack](#) (int flag)
- Set default behavior for whether to backtrack secondary structures.*

 - int [vrna_md_defaults_backtrack_get](#) (void)
- Get default behavior for whether to backtrack secondary structures.*

 - void [vrna_md_defaults_backtrack_type](#) (char t)
- Set default backtrack type, i.e. which DP matrix is used.*

 - char [vrna_md_defaults_backtrack_type_get](#) (void)
- Get default backtrack type, i.e. which DP matrix is used.*

 - void [vrna_md_defaults_compute_bpp](#) (int flag)
- Set the default behavior for whether to compute base pair probabilities after partition function computation.*

 - int [vrna_md_defaults_compute_bpp_get](#) (void)
- Get the default behavior for whether to compute base pair probabilities after partition function computation.*

 - void [vrna_md_defaults_max_bp_span](#) (int span)
- Set default maximal base pair span.*

 - int [vrna_md_defaults_max_bp_span_get](#) (void)
- Get default maximal base pair span.*

 - void [vrna_md_defaults_min_loop_size](#) (int size)
- Set default minimal loop size.*

 - int [vrna_md_defaults_min_loop_size_get](#) (void)
- Get default minimal loop size.*

 - void [vrna_md_defaults_window_size](#) (int size)
- Set default window size for sliding window structure prediction approaches.*

 - int [vrna_md_defaults_window_size_get](#) (void)
- Get default window size for sliding window structure prediction approaches.*

 - void [vrna_md_defaults_oldAliEn](#) (int flag)
- Set default behavior for whether to use old energy model for comparative structure prediction.*

 - int [vrna_md_defaults_oldAliEn_get](#) (void)
- Get default behavior for whether to use old energy model for comparative structure prediction.*

 - void [vrna_md_defaults_ribo](#) (int flag)
- Set default behavior for whether to use Ribosum Scoring in comparative structure prediction.*

 - int [vrna_md_defaults_ribo_get](#) (void)
- Get default behavior for whether to use Ribosum Scoring in comparative structure prediction.*

 - void [vrna_md_defaults_cv_fact](#) (double factor)
- Set the default co-variance scaling factor used in comparative structure prediction.*

 - double [vrna_md_defaults_cv_fact_get](#) (void)
- Get the default co-variance scaling factor used in comparative structure prediction.*

 - void [vrna_md_defaults_nc_fact](#) (double factor)
- double [vrna_md_defaults_nc_fact_get](#) (void)

- void `vrna_md_defaults_sfact` (double factor)
Set the default scaling factor used to avoid under-/overflows in partition function computation.
- double `vrna_md_defaults_sfact_get` (void)
Get the default scaling factor used to avoid under-/overflows in partition function computation.
- void `vrna_md_defaults_salt` (double salt)
Set the default salt concentration.
- double `vrna_md_defaults_salt_get` (void)
Get the default salt concentration.
- void `vrna_md_defaults_saltMLLower` (int lower)
Set the default multiloop size lower bound for loop salt correction linear fitting.
- int `vrna_md_defaults_saltMLLower_get` (void)
Get the default multiloop size lower bound for loop salt correction linear fitting.
- void `vrna_md_defaults_saltMLUpper` (int upper)
Set the default multiloop size upper bound for loop salt correction linear fitting.
- int `vrna_md_defaults_saltMLUpper_get` (void)
Get the default multiloop size upper bound for loop salt correction linear fitting.
- void `vrna_md_defaults_saltDPXInit` (int value)
Set user-provided salt correction for duplex initialization. If value is 99999 the default value from fitting is used.
- int `vrna_md_defaults_saltDPXInit_get` (void)
Get user-provided salt correction for duplex initialization. If value is 99999 the default value from fitting is used.
- void `set_model_details` (`vrna_md_t *md`)
Set default model details.

Variables

- double `temperature`
Rescale energy parameters to a temperature in degC.
- double `pf_scale`
A scaling factor used by `pf_fold()` to avoid overflows.
- int `dangles`
Switch the energy model for dangling end contributions (0, 1, 2, 3)
- int `tetra_loop`
Include special stabilizing energies for some tri-, tetra- and hexa-loops;
- int `noLonelyPairs`
Global switch to avoid/allow helices of length 1.
- int `noGU`
Global switch to forbid/allow GU base pairs at all.
- int `no_closingGU`
GU allowed only inside stacks if set to 1.
- int `circ`
backward compatibility variable.. this does not effect anything
- int `gquad`
Allow G-quadruplex formation.
- int `uniq_ML`
do ML decomposition uniquely (for subopt)
- int `energy_set`
0 = BP; 1=any with GC; 2=any with AU-parameter
- int `do_backtrack`
do backtracking, i.e. compute secondary structures or base pair probabilities
- char `backtrack_type`
A backtrack array marker for `inverse_fold()`

- char * [nonstandards](#)
contains allowed non standard base pairs
- int [max_bp_span](#)
Maximum allowed base pair span.
- int **oldAliEn**
use old alifold energies (with gaps)
- int **ribo**
use ribosum matrices
- int **logML**
if nonzero use logarithmic ML energy in energy_of_struct
- double **salt**
salt concentration
- int **saltDPXInit**
Salt correction for duplex initialization.

16.6.2 Data Structure Documentation

16.6.2.1 struct vrna_md_s

The data structure that contains the complete model details used throughout the calculations.

For convenience reasons, we provide the type name [vrna_md_t](#) to address this data structure without the use of the struct keyword

See also

[vrna_md_set_default\(\)](#), [set_model_details\(\)](#), [vrna_md_update\(\)](#), [vrna_md_t](#)

SWIG Wrapper Notes This data structure is wrapped as an object **md** with multiple related functions attached as methods.

A new set of default parameters can be obtained by calling the constructor of **md**:

- `md()` – Initialize with default settings

The resulting object has a list of attached methods which directly correspond to functions that mainly operate on the corresponding C data structure:

- `reset()` – [vrna_md_set_default\(\)](#)
- `set_from_globals()` – [set_model_details\(\)](#)
- `option_string()` – [vrna_md_option_string\(\)](#)

Note, that default parameters can be modified by directly setting any of the following global variables. Internally, getting/setting default parameters using their global variable representative translates into calls of the following functions, therefore these wrappers for these functions do not exist in the scripting language interface(s):

global variable	C getter	C setter
temperature	vrna_md_defaults_temperature_get()	vrna_md_defaults_temperature()
dangles	vrna_md_defaults_dangles_get()	vrna_md_defaults_dangles()
betaScale	vrna_md_defaults_betaScale_get()	vrna_md_defaults_betaScale()
tetra_loop	this is an alias of <i>special_hp</i>	
special_hp	vrna_md_defaults_special_hp_get()	vrna_md_defaults_special_hp()
noLonelyPairs	this is an alias of <i>noLP</i>	
noLP	vrna_md_defaults_noLP_get()	vrna_md_defaults_noLP()

global variable	C getter	C setter
noGU	vrna_md_defaults_noGU_get()	vrna_md_defaults_noGU()
no_closingGU	this is an alias of <i>noGUclosure</i>	
noGUclosure	vrna_md_defaults_noGUclosure_get()	vrna_md_defaults_noGUclosure()
logML	vrna_md_defaults_logML_get()	vrna_md_defaults_logML()
circ	vrna_md_defaults_circ_get()	vrna_md_defaults_circ()
gquad	vrna_md_defaults_gquad_get()	vrna_md_defaults_gquad()
uniq_ML	vrna_md_defaults_uniq_ML_get()	vrna_md_defaults_uniq_ML()
energy_set	vrna_md_defaults_energy_set_get()	vrna_md_defaults_energy_set()
backtrack	vrna_md_defaults_backtrack_get()	vrna_md_defaults_backtrack()
backtrack_type	vrna_md_defaults_backtrack_type_get()	vrna_md_defaults_backtrack_type()
do_backtrack	this is an alias of <i>compute_bpp</i>	
compute_bpp	vrna_md_defaults_compute_bpp_get()	vrna_md_defaults_compute_bpp()
max_bp_span	vrna_md_defaults_max_bp_span_get()	vrna_md_defaults_max_bp_span()
min_loop_size	vrna_md_defaults_min_loop_size_get()	vrna_md_defaults_min_loop_size()
window_size	vrna_md_defaults_window_size_get()	vrna_md_defaults_window_size()
oldAliEn	vrna_md_defaults_oldAliEn_get()	vrna_md_defaults_oldAliEn()
ribo	vrna_md_defaults_ribo_get()	vrna_md_defaults_ribo()
cv_fact	vrna_md_defaults_cv_fact_get()	vrna_md_defaults_cv_fact()
nc_fact	vrna_md_defaults_nc_fact_get()	vrna_md_defaults_nc_fact()
sfact	vrna_md_defaults_sfact_get()	vrna_md_defaults_sfact()

Data Fields

- double **temperature**
The temperature used to scale the thermodynamic parameters.
- double **betaScale**
A scaling factor for the thermodynamic temperature of the Boltzmann factors.
- int **pf_smooth**
A flag specifying whether energies in Boltzmann factors need to be smoothed.
- int **dangles**
Specifies the dangle model used in any energy evaluation (0,1,2 or 3)
- int **special_hp**
Include special hairpin contributions for tri, tetra and hexaloops.
- int **noLP**
Only consider canonical structures, i.e. no 'lonely' base pairs.
- int **noGU**
Do not allow GU pairs.
- int **noGUclosure**
Do not allow loops to be closed by GU pair.
- int **logML**
Use logarithmic scaling for multiloops.
- int **circ**
Assume RNA to be circular instead of linear.
- int **gquad**
Include G-quadruplexes in structure prediction.
- int **uniq_ML**
Flag to ensure unique multi-branch loop decomposition during folding.
- int **energy_set**
Specifies the energy set that defines set of compatible base pairs.

- int **backtrack**
Specifies whether or not secondary structures should be backtraced.
- char **backtrack_type**
Specifies in which matrix to backtrack.
- int **compute_bpp**
Specifies whether or not backward recursions for base pair probability (bpp) computation will be performed.
- char **nonstandards** [64]
contains allowed non standard bases
- int **max_bp_span**
maximum allowed base pair span
- int **min_loop_size**
Minimum size of hairpin loops.
- int **window_size**
Size of the sliding window for locally optimal structure prediction.
- int **oldAliEn**
Use old alifold energy model.
- int **ribo**
Use ribosum scoring table in alifold energy model.
- double **cv_fact**
Co-variance scaling factor for consensus structure prediction.
- double **nc_fact**
Scaling factor to weight co-variance contributions of non-canonical pairs.
- double **sfact**
Scaling factor for partition function scaling.
- int **rtype** [8]
Reverse base pair type array.
- short **alias** [MAXALPHA+1]
alias of an integer nucleotide representation
- int **pair** [MAXALPHA+1][MAXALPHA+1]
Integer representation of a base pair.
- float **pair_dist** [7][7]
Base pair dissimilarity, a.k.a. distance matrix.
- double **salt**
Salt (monovalent) concentration (M) in buffer.
- int **saltMLLower**
Lower bound of multiloop size to use in loop salt correction linear fitting.
- int **saltMLUpper**
Upper bound of multiloop size to use in loop salt correction linear fitting.
- int **saltDPXInit**
User-provided salt correction for duplex initialization (in dcal/mol). If set to 99999 the default salt correction is used. If set to 0 there is no salt correction for duplex initialization.

16.6.2.1.1 Field Documentation

16.6.2.1.1.1 dangles `int vrna_md_s::dangles`

Specifies the dangle model used in any energy evaluation (0,1,2 or 3)

If set to 0 no stabilizing energies are assigned to bases adjacent to helices in free ends and multiloops (so called dangling ends). Normally (dangles = 1) dangling end energies are assigned only to unpaired bases and a base cannot participate simultaneously in two dangling ends. In the partition function algorithm `vrna_pf()` these checks are neglected. To provide comparability between free energy minimization and partition function algorithms, the default setting is 2. This treatment of dangling ends gives more favorable energies to helices directly adjacent to one another, which can be beneficial since such helices often do engage in stabilizing interactions through co-axial stacking.

If set to 3 co-axial stacking is explicitly included for adjacent helices in multiloops. The option affects only mfe folding and energy evaluation (`vrna_mfe()` and `vrna_eval_structure()`), as well as suboptimal folding (`vrna_subopt()`) via re-evaluation of energies. Co-axial stacking with one intervening mismatch is not considered so far.

Note

Some function do not implement all dangle model but only a subset of (0,1,2,3). In particular, partition function algorithms can only handle 0 and 2. Read the documentation of the particular recurrences or energy evaluation function for information about the provided dangle model.

16.6.2.1.1.2 min_loop_size `int vrna_md_s::min_loop_size`

Minimum size of hairpin loops.

Note

The default value for this field is [TURN](#), however, it may be 0 in cofolding context.

16.6.3 Macro Definition Documentation**16.6.3.1 VRNA_MODEL_DEFAULT_TEMPERATURE**

```
#define VRNA_MODEL_DEFAULT_TEMPERATURE 37.0
#include <ViennaRNA/model.h>
```

Default temperature for structure prediction and free energy evaluation in °C

See also

[vrna_md_t.temperature](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.2 VRNA_MODEL_DEFAULT_PF_SCALE

```
#define VRNA_MODEL_DEFAULT_PF_SCALE -1
#include <ViennaRNA/model.h>
```

Default scaling factor for partition function computations.

See also

[vrna_exp_param_t.pf_scale](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.3 VRNA_MODEL_DEFAULT_BETA_SCALE

```
#define VRNA_MODEL_DEFAULT_BETA_SCALE 1.
#include <ViennaRNA/model.h>
```

Default scaling factor for absolute thermodynamic temperature in Boltzmann factors.

See also

[vrna_exp_param_t.alpha](#), [vrna_md_t.betaScale](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.4 VRNA_MODEL_DEFAULT_DANGLES

```
#define VRNA_MODEL_DEFAULT_DANGLES 2
#include <ViennaRNA/model.h>
```

Default dangling end model.

See also

[vrna_md_t.dangles](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.5 VRNA_MODEL_DEFAULT_SPECIAL_HP

```
#define VRNA_MODEL_DEFAULT_SPECIAL_HP 1
#include <ViennaRNA/model.h>
```

Default model behavior for lookup of special tri-, tetra-, and hexa-loops.

See also

[vrna_md_t.special_hp](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.6 VRNA_MODEL_DEFAULT_NO_LP

```
#define VRNA_MODEL_DEFAULT_NO_LP 0
#include <ViennaRNA/model.h>
```

Default model behavior for so-called 'lonely pairs'.

See also

[vrna_md_t.noLP](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.7 VRNA_MODEL_DEFAULT_NO_GU

```
#define VRNA_MODEL_DEFAULT_NO_GU 0
#include <ViennaRNA/model.h>
```

Default model behavior for G-U base pairs.

See also

[vrna_md_t.noGU](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.8 VRNA_MODEL_DEFAULT_NO_GU_CLOSURE

```
#define VRNA_MODEL_DEFAULT_NO_GU_CLOSURE 0
#include <ViennaRNA/model.h>
```

Default model behavior for G-U base pairs closing a loop.

See also

[vrna_md_t.noGUclosure](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.9 VRNA_MODEL_DEFAULT_CIRC

```
#define VRNA_MODEL_DEFAULT_CIRC 0
#include <ViennaRNA/model.h>
```

Default model behavior to treat a molecule as a circular RNA (DNA)

See also

[vrna_md_t.circ](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.10 VRNA_MODEL_DEFAULT_GQUAD

```
#define VRNA_MODEL_DEFAULT_GQUAD 0
#include <ViennaRNA/model.h>
```

Default model behavior regarding the treatment of G-Quadruplexes.

See also

[vrna_md_t.gquad](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.11 VRNA_MODEL_DEFAULT_UNIQ_ML

```
#define VRNA_MODEL_DEFAULT_UNIQ_ML 0
#include <ViennaRNA/model.h>
```

Default behavior of the model regarding unique multi-branch loop decomposition.

See also

[vrna_md_t.uniq_ML](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.12 VRNA_MODEL_DEFAULT_ENERGY_SET

```
#define VRNA_MODEL_DEFAULT_ENERGY_SET 0
#include <ViennaRNA/model.h>
```

Default model behavior on which energy set to use.

See also

[vrna_md_t.energy_set](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.13 VRNA_MODEL_DEFAULT_BACKTRACK

```
#define VRNA_MODEL_DEFAULT_BACKTRACK 1
#include <ViennaRNA/model.h>
```

Default model behavior with regards to backtracking of structures.

See also

[vrna_md_t.backtrack](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.14 VRNA_MODEL_DEFAULT_BACKTRACK_TYPE

```
#define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'
#include <ViennaRNA/model.h>
```

Default model behavior on what type of backtracking to perform.

See also

[vrna_md_t.backtrack_type](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.15 VRNA_MODEL_DEFAULT_COMPUTE_BPP

```
#define VRNA_MODEL_DEFAULT_COMPUTE_BPP 1
#include <ViennaRNA/model.h>
```

Default model behavior with regards to computing base pair probabilities.

See also

[vrna_md_t.compute_bpp](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.16 VRNA_MODEL_DEFAULT_MAX_BP_SPAN

```
#define VRNA_MODEL_DEFAULT_MAX_BP_SPAN -1
#include <ViennaRNA/model.h>
```

Default model behavior for the allowed maximum base pair span.

See also

[vrna_md_t.max_bp_span](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.17 VRNA_MODEL_DEFAULT_WINDOW_SIZE

```
#define VRNA_MODEL_DEFAULT_WINDOW_SIZE -1
#include <ViennaRNA/model.h>
```

Default model behavior for the sliding window approach.

See also

[vrna_md_t.window_size](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.18 VRNA_MODEL_DEFAULT_LOG_ML

```
#define VRNA_MODEL_DEFAULT_LOG_ML 0
#include <ViennaRNA/model.h>
```

Default model behavior on how to evaluate the energy contribution of multi-branch loops.

See also

[vrna_md_t.logML](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.19 VRNA_MODEL_DEFAULT_ALI_OLD_EN

```
#define VRNA_MODEL_DEFAULT_ALI_OLD_EN 0
#include <ViennaRNA/model.h>
```

Default model behavior for consensus structure energy evaluation.

See also

[vrna_md_t.oldAliEn](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.20 VRNA_MODEL_DEFAULT_ALI_RIBO

```
#define VRNA_MODEL_DEFAULT_ALI_RIBO 0
#include <ViennaRNA/model.h>
```

Default model behavior for consensus structure co-variance contribution assessment.

See also

[vrna_md_t.ribo](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.21 VRNA_MODEL_DEFAULT_ALI_CV_FACT

```
#define VRNA_MODEL_DEFAULT_ALI_CV_FACT 1.
#include <ViennaRNA/model.h>
```

Default model behavior for weighting the co-variance score in consensus structure prediction.

See also

[vrna_md_t.cv_fact](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.3.22 VRNA_MODEL_DEFAULT_ALI_NC_FACT

```
#define VRNA_MODEL_DEFAULT_ALI_NC_FACT 1.
#include <ViennaRNA/model.h>
```

Default model behavior for weighting the nucleotide conservation? in consensus structure prediction.

See also

[vrna_md_t.nc_fact](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#)

16.6.4 Function Documentation

16.6.4.1 vrna_md_set_default()

```
void vrna_md_set_default (
    vrna_md_t * md )
#include <ViennaRNA/model.h>
```

Apply default model details to a provided [vrna_md_t](#) data structure.

Use this function to initialize a [vrna_md_t](#) data structure with its default values

Parameters

<i>md</i>	A pointer to the data structure that is about to be initialized
-----------	---

16.6.4.2 vrna_md_update()

```
void vrna_md_update (
    vrna_md_t * md )
#include <ViennaRNA/model.h>
```

Update the model details data structure.

This function should be called after changing the [vrna_md_t.energy_set](#) attribute since it re-initializes base pairing related arrays within the [vrna_md_t](#) data structure. In particular, [vrna_md_t.pair](#), [vrna_md_t.alias](#), and [vrna_md_t.rtype](#) are set to the values that correspond to the specified [vrna_md_t.energy_set](#) option

See also

[vrna_md_t](#), [vrna_md_t.energy_set](#), [vrna_md_t.pair](#), [vrna_md_t.rtype](#), [vrna_md_t.alias](#), [vrna_md_set_default\(\)](#)

16.6.4.3 vrna_md_copy()

```
vrna_md_t * vrna_md_copy (
    vrna_md_t * md_to,
    const vrna_md_t * md_from )
#include <ViennaRNA/model.h>
```

Copy/Clone a [vrna_md_t](#) model.

Use this function to clone a given model either inplace (target container [md_to](#) given) or create a copy by cloning the source model and returning it ([md_to](#) == NULL).

Parameters

<i>md_to</i>	The model to be overwritten (if non-NULL and md_to != md_from)
<i>md_from</i>	The model to copy (if non-NULL)

Returns

A pointer to the copy model (or NULL if `md_from == NULL`)

16.6.4.4 vrna_md_option_string()

```
char * vrna_md_option_string (
    vrna_md_t * md )
```

```
#include <ViennaRNA/model.h>
```

Get a corresponding commandline parameter string of the options in a [vrna_md_t](#).

Note

This function is not threadsafe!

16.6.4.5 vrna_md_defaults_reset()

```
void vrna_md_defaults_reset (
    vrna_md_t * md_p )
```

```
#include <ViennaRNA/model.h>
```

Reset the global default model details to a specific set of parameters, or their initial values.

This function resets the global default model details to their initial values, i.e. as specified by the ViennaRNA Package release, upon passing NULL as argument. Alternatively it resets them according to a set of provided parameters.

Note

The global default parameters affect all function calls of RNAlib where model details are not explicitly provided. Hence, any change of them is not considered threadsafe

Warning

This function first resets the global default settings to factory defaults, and only then applies user provided settings (if any). User settings that do not meet specifications are skipped.

See also

[vrna_md_set_default\(\)](#), [vrna_md_t](#)

Parameters

<code>md_p</code>	A set of model details to use as global default (if NULL is passed, factory defaults are restored)
-------------------	--

16.6.4.6 vrna_md_defaults_temperature()

```
void vrna_md_defaults_temperature (
    double T )
```

```
#include <ViennaRNA/model.h>
```

Set default temperature for energy evaluation of loops.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_TEMPERATURE](#)

Parameters

<i>T</i>	Temperature in centigrade
----------	---------------------------

16.6.4.7 vrna_md_defaults_temperature_get()

```
double vrna_md_defaults_temperature_get (
    void )
#include <ViennaRNA/model.h>
Get default temperature for energy evaluation of loops.
```

See also

[vrna_md_defaults_temperature\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_TEMPERATURE](#)

Returns

The global default settings for temperature in centigrade

16.6.4.8 vrna_md_defaults_betaScale()

```
void vrna_md_defaults_betaScale (
    double b )
#include <ViennaRNA/model.h>
Set default scaling factor of thermodynamic temperature in Boltzmann factors.
Boltzmann factors are then computed as  $\exp(-E/(b \cdot kT))$ .
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_BETA_SCALE](#)

Parameters

<i>b</i>	The scaling factor, default is 1.0
----------	------------------------------------

16.6.4.9 vrna_md_defaults_betaScale_get()

```
double vrna_md_defaults_betaScale_get (
    void )
#include <ViennaRNA/model.h>
Get default scaling factor of thermodynamic temperature in Boltzmann factors.
```

See also

[vrna_md_defaults_betaScale\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_BETA_SCALE](#)

Returns

The global default thermodynamic temperature scaling factor

16.6.4.10 vrna_md_defaults_dangles()

```
void vrna_md_defaults_dangles (
    int d )
#include <ViennaRNA/model.h>
Set default dangle model for structure prediction.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_DANGLES](#)

Parameters

<i>d</i>	The dangle model
----------	------------------

16.6.4.11 vrna_md_defaults_dangles_get()

```
int vrna_md_defaults_dangles_get (
    void )
#include <ViennaRNA/model.h>
Get default dangle model for structure prediction.
```

See also

[vrna_md_defaults_dangles\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_DANG](#)

Returns

The global default settings for the dangle model

16.6.4.12 vrna_md_defaults_special_hp()

```
void vrna_md_defaults_special_hp (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_SPECIAL_HP](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.13 vrna_md_defaults_special_hp_get()

```
int vrna_md_defaults_special_hp_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
```


See also

[vrna_md_defaults_special_hp\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_SPECIAL_HP](#)

Returns

The global default settings for the treatment of special hairpin loops

16.6.4.14 `vrna_md_defaults_noLP()`

```
void vrna_md_defaults_noLP (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior for prediction of canonical secondary structures.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_NO_LP](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.15 `vrna_md_defaults_noLP_get()`

```
int vrna_md_defaults_noLP_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default behavior for prediction of canonical secondary structures.

See also

[vrna_md_defaults_noLP\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_NO_LP](#)

Returns

The global default settings for predicting canonical secondary structures

16.6.4.16 `vrna_md_defaults_noGU()`

```
void vrna_md_defaults_noGU (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior for treatment of G-U wobble pairs.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_NO_GU](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.17 vrna_md_defaults_noGU_get()

```
int vrna_md_defaults_noGU_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior for treatment of G-U wobble pairs.

See also

[vrna_md_defaults_noGU\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_NO_GU](#)

Returns

The global default settings for treatment of G-U wobble pairs

16.6.4.18 vrna_md_defaults_noGUclosure()

```
void vrna_md_defaults_noGUclosure (
    int flag )
#include <ViennaRNA/model.h>
```

Set default behavior for G-U pairs as closing pair for loops.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_NO_GU_CLOSURE](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.19 vrna_md_defaults_noGUclosure_get()

```
int vrna_md_defaults_noGUclosure_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior for G-U pairs as closing pair for loops.

See also

[vrna_md_defaults_noGUclosure\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_NO_GU_CLOSURE](#)

Returns

The global default settings for treatment of G-U pairs closing a loop

16.6.4.20 vrna_md_defaults_logML()

```
void vrna_md_defaults_logML (
    int flag )
#include <ViennaRNA/model.h>
```

Set default behavior recomputing free energies of multi-branch loops using a logarithmic model.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_LOG_ML](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.21 vrna_md_defaults_logML_get()

```
int vrna_md_defaults_logML_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default behavior recomputing free energies of multi-branch loops using a logarithmic model.

See also

[vrna_md_defaults_logML\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_LOG_M](#)

Returns

The global default settings for logarithmic model in multi-branch loop free energy evaluation

16.6.4.22 vrna_md_defaults_circ()

```
void vrna_md_defaults_circ (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior whether input sequences are circularized.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_CIRC](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.23 vrna_md_defaults_circ_get()

```
int vrna_md_defaults_circ_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default behavior whether input sequences are circularized.

See also

[vrna_md_defaults_circ\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_CIRC](#)

Returns

The global default settings for treating input sequences as circular

16.6.4.24 vrna_md_defaults_gquad()

```
void vrna_md_defaults_gquad (
    int flag )
```

```
#include <ViennaRNA/model.h>
Set default behavior for treatment of G-Quadruplexes.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_GQUAD](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.25 vrna_md_defaults_gquad_get()

```
int vrna_md_defaults_gquad_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for treatment of G-Quadruplexes.
```

See also

[vrna_md_defaults_gquad\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_GQUAD](#)

Returns

The global default settings for treatment of G-Quadruplexes

16.6.4.26 vrna_md_defaults_uniq_ML()

```
void vrna_md_defaults_uniq_ML (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for creating additional matrix for unique multi-branch loop prediction.
```

Note

Activating this option usually results in higher memory consumption!

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_UNIQ_ML](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.27 vrna_md_defaults_uniq_ML_get()

```
int vrna_md_defaults_uniq_ML_get (
    void )
#include <ViennaRNA/model.h>
Get default behavior for creating additional matrix for unique multi-branch loop prediction.
```

See also

[vrna_md_defaults_uniq_ML\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_UNIQ](#)

Returns

The global default settings for creating additional matrices for unique multi-branch loop prediction

16.6.4.28 `vrna_md_defaults_energy_set()`

```
void vrna_md_defaults_energy_set (
    int e )
#include <ViennaRNA/model.h>
Set default energy set.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ENERGY_SET](#)

Parameters

<i>e</i>	Energy set (0, 1, 2, 3)
----------	-------------------------

16.6.4.29 `vrna_md_defaults_energy_set_get()`

```
int vrna_md_defaults_energy_set_get (
    void )
#include <ViennaRNA/model.h>
Get default energy set.
```

See also

[vrna_md_defaults_energy_set\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_EN](#)

Returns

The global default settings for the energy set

16.6.4.30 `vrna_md_defaults_backtrack()`

```
void vrna_md_defaults_backtrack (
    int flag )
#include <ViennaRNA/model.h>
Set default behavior for whether to backtrack secondary structures.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_BACKTRACK](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.31 vrna_md_defaults_backtrack_get()

```
int vrna_md_defaults_backtrack_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default behavior for whether to backtrack secondary structures.

See also

[vrna_md_defaults_backtrack\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_BAC](#)

Returns

The global default settings for backtracking structures

16.6.4.32 vrna_md_defaults_backtrack_type()

```
void vrna_md_defaults_backtrack_type (
    char t )
```

```
#include <ViennaRNA/model.h>
```

Set default backtrack type, i.e. which DP matrix is used.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_BACKTRACK_TYPE](#)

Parameters

<i>t</i>	The type ('F', 'C', or 'M')
----------	-----------------------------

16.6.4.33 vrna_md_defaults_backtrack_type_get()

```
char vrna_md_defaults_backtrack_type_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default backtrack type, i.e. which DP matrix is used.

See also

[vrna_md_defaults_backtrack_type\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_BACKTRACK_TYPE](#)

Returns

The global default settings that specify which DP matrix is used for backtracking

16.6.4.34 vrna_md_defaults_compute_bpp()

```
void vrna_md_defaults_compute_bpp (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set the default behavior for whether to compute base pair probabilities after partition function computation.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_COMPUTE_BPP](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.35 vrna_md_defaults_compute_bpp_get()

```
int vrna_md_defaults_compute_bpp_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default behavior for whether to compute base pair probabilities after partition function computation.

See also

[vrna_md_defaults_compute_bpp\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_C](#)

Returns

The global default settings that specify whether base pair probabilities are computed together with partition function

16.6.4.36 vrna_md_defaults_max_bp_span()

```
void vrna_md_defaults_max_bp_span (
    int span )
#include <ViennaRNA/model.h>
```

Set default maximal base pair span.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_MAX_BP_SPAN](#)

Parameters

<i>span</i>	Maximal base pair span
-------------	------------------------

16.6.4.37 vrna_md_defaults_max_bp_span_get()

```
int vrna_md_defaults_max_bp_span_get (
    void )
#include <ViennaRNA/model.h>
```

Get default maximal base pair span.

See also

[vrna_md_defaults_max_bp_span\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_C](#)

Returns

The global default settings for maximum base pair span

16.6.4.38 vrna_md_defaults_min_loop_size()

```
void vrna_md_defaults_min_loop_size (
    int size )
```

```
#include <ViennaRNA/model.h>
Set default minimal loop size.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [TURN](#)

Parameters

<i>size</i>	Minimal size, i.e. number of unpaired nucleotides for a hairpin loop
-------------	--

16.6.4.39 vrna_md_defaults_min_loop_size_get()

```
int vrna_md_defaults_min_loop_size_get (
    void )
#include <ViennaRNA/model.h>
Get default minimal loop size.
```

See also

[vrna_md_defaults_min_loop_size\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [TURN](#)

Returns

The global default settings for minimal size of hairpin loops

16.6.4.40 vrna_md_defaults_window_size()

```
void vrna_md_defaults_window_size (
    int size )
#include <ViennaRNA/model.h>
Set default window size for sliding window structure prediction approaches.
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_WINDOW_SIZE](#)

Parameters

<i>size</i>	The size of the sliding window
-------------	--------------------------------

16.6.4.41 vrna_md_defaults_window_size_get()

```
int vrna_md_defaults_window_size_get (
    void )
#include <ViennaRNA/model.h>
Get default window size for sliding window structure prediction approaches.
```

See also

[vrna_md_defaults_window_size\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_W](#)

Returns

The global default settings for the size of the sliding window

16.6.4.42 vrna_md_defaults_oldAliEn()

```
void vrna_md_defaults_oldAliEn (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior for whether to use old energy model for comparative structure prediction.

Note

This option is outdated. Activating the old energy model usually results in worse consensus structure predictions.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_OLD_EN](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.43 vrna_md_defaults_oldAliEn_get()

```
int vrna_md_defaults_oldAliEn_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get default behavior for whether to use old energy model for comparative structure prediction.

See also

[vrna_md_defaults_oldAliEn\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_O](#)

Returns

The global default settings for using old energy model for comparative structure prediction

16.6.4.44 vrna_md_defaults_ribo()

```
void vrna_md_defaults_ribo (
    int flag )
```

```
#include <ViennaRNA/model.h>
```

Set default behavior for whether to use Ribosum Scoring in comparative structure prediction.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_RIBO](#)

Parameters

<i>flag</i>	On/Off switch (0 = OFF, else = ON)
-------------	------------------------------------

16.6.4.45 vrna_md_defaults_ribo_get()

```
int vrna_md_defaults_ribo_get (
    void )
#include <ViennaRNA/model.h>
```

Get default behavior for whether to use Ribosum Scoring in comparative structure prediction.

See also

[vrna_md_defaults_ribo\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_RIBO](#)

Returns

The global default settings for using Ribosum scoring in comparative structure prediction

16.6.4.46 vrna_md_defaults_cv_fact()

```
void vrna_md_defaults_cv_fact (
    double factor )
#include <ViennaRNA/model.h>
```

Set the default co-variance scaling factor used in comparative structure prediction.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_CV_FACT](#)

Parameters

<i>factor</i>	The co-variance factor
---------------	------------------------

16.6.4.47 vrna_md_defaults_cv_fact_get()

```
double vrna_md_defaults_cv_fact_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default co-variance scaling factor used in comparative structure prediction.

See also

[vrna_md_defaults_cv_fact\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_CV_FACT](#)

Returns

The global default settings for the co-variance factor

16.6.4.48 vrna_md_defaults_nc_fact()

```
void vrna_md_defaults_nc_fact (
    double factor )
#include <ViennaRNA/model.h>
```

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_NC_FACT](#)

Parameters

<i>factor</i>	
---------------	--

16.6.4.49 vrna_md_defaults_nc_fact_get()

```
double vrna_md_defaults_nc_fact_get (
    void )
#include <ViennaRNA/model.h>
```

See also

[vrna_md_defaults_nc_fact\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#), [VRNA_MODEL_DEFAULT_ALI_NC](#)

Returns

16.6.4.50 vrna_md_defaults_sfact()

```
void vrna_md_defaults_sfact (
    double factor )
#include <ViennaRNA/model.h>
```

Set the default scaling factor used to avoid under-/overflows in partition function computation.

See also

[vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#)

Parameters

<i>factor</i>	The scaling factor (default: 1.07)
---------------	------------------------------------

16.6.4.51 vrna_md_defaults_sfact_get()

```
double vrna_md_defaults_sfact_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default scaling factor used to avoid under-/overflows in partition function computation.

See also

[vrna_md_defaults_sfact\(\)](#), [vrna_md_defaults_reset\(\)](#), [vrna_md_set_default\(\)](#), [vrna_md_t](#)

Returns

The global default settings of the scaling factor

16.6.4.52 vrna_md_defaults_salt()

```
void vrna_md_defaults_salt (
    double salt )
#include <ViennaRNA/model.h>
```

Set the default salt concentration.

Parameters

<i>salt</i>	The sodium concentration in M (default: 1.021)
-------------	--

16.6.4.53 vrna_md_defaults_salt_get()

```
double vrna_md_defaults_salt_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default salt concentration.

Returns

The default salt concentration

16.6.4.54 vrna_md_defaults_saltMLLower()

```
void vrna_md_defaults_saltMLLower (
    int lower )
#include <ViennaRNA/model.h>
```

Set the default multiloop size lower bound for loop salt correction linear fitting.

Parameters

<i>lower</i>	Size lower bound (number of backbone in loop)
--------------	---

16.6.4.55 vrna_md_defaults_saltMLLower_get()

```
int vrna_md_defaults_saltMLLower_get (
    void )
#include <ViennaRNA/model.h>
```

Get the default multiloop size lower bound for loop salt correction linear fitting.

Returns

The default lower bound

16.6.4.56 vrna_md_defaults_saltMLUpper()

```
void vrna_md_defaults_saltMLUpper (
    int upper )
#include <ViennaRNA/model.h>
```

Set the default multiloop size upper bound for loop salt correction linear fitting.

Parameters

<i>upper</i>	Size Upper bound (number of backbone in loop)
--------------	---

16.6.4.57 vrna_md_defaults_saltMLUpper_get()

```
int vrna_md_defaults_saltMLUpper_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get the default multiloop size upper bound for loop salt correction linear fitting.

Returns

The default upper bound

16.6.4.58 vrna_md_defaults_saltDPXInit()

```
void vrna_md_defaults_saltDPXInit (
    int value )
```

```
#include <ViennaRNA/model.h>
```

Set user-provided salt correction for duplex initialization. If value is 99999 the default value from fitting is used.

Parameters

<i>value</i>	The value of salt correction for duplex initialization (in dcal/mol)
--------------	--

16.6.4.59 vrna_md_defaults_saltDPXInit_get()

```
int vrna_md_defaults_saltDPXInit_get (
    void )
```

```
#include <ViennaRNA/model.h>
```

Get user-provided salt correction for duplex initialization. If value is 99999 the default value from fitting is used.

Returns

The user-provided salt correction for duplex initialization

16.6.4.60 set_model_details()

```
void set_model_details (
    vrna_md_t * md )
```

```
#include <ViennaRNA/model.h>
```

Set default model details.

Use this function if you wish to initialize a `vrna_md_t` data structure with its default values, i.e. the global model settings as provided by the deprecated global variables.

Deprecated This function will vanish as soon as backward compatibility of RNAlib is dropped (expected in version 3). Use `vrna_md_set_default()` instead!

Parameters

<i>md</i>	A pointer to the data structure that is about to be initialized
-----------	---

16.6.5 Variable Documentation

16.6.5.1 temperature

```
double temperature [extern]
```

```
#include <ViennaRNA/model.h>
```

Rescale energy parameters to a temperature in degC.

Default is 37C. You have to call the `update_..._params()` functions after changing this parameter.

Deprecated Use `vrna_md_defaults_temperature()`, and `vrna_md_defaults_temperature_get()` to change, and read the global default temperature settings

See also

[vrna_md_defaults_temperature\(\)](#), [vrna_md_defaults_temperature_get\(\)](#), [vrna_md_defaults_reset\(\)](#)

16.6.5.2 pf_scale

```
double pf_scale [extern]
```

```
#include <ViennaRNA/model.h>
```

A scaling factor used by `pf_fold()` to avoid overflows.

Should be set to approximately $\exp((-F/kT)/length)$, where F is an estimate for the ensemble free energy, for example the minimum free energy. You must call `update_pf_params()` after changing this parameter.

If `pf_scale` is -1 (the default), an estimate will be provided automatically when computing partition functions, e.g. `pf_fold()`. The automatic estimate is usually insufficient for sequences more than a few hundred bases long.

16.6.5.3 dangles

```
int dangles [extern]
```

```
#include <ViennaRNA/model.h>
```

Switch the energy model for dangling end contributions (0, 1, 2, 3)

If set to 0 no stabilizing energies are assigned to bases adjacent to helices in free ends and multiloops (so called dangling ends). Normally (`dangles` = 1) dangling end energies are assigned only to unpaired bases and a base cannot participate simultaneously in two dangling ends. In the partition function algorithm `pf_fold()` these checks are neglected. If `dangles` is set to 2, all folding routines will follow this convention. This treatment of dangling ends gives more favorable energies to helices directly adjacent to one another, which can be beneficial since such helices often do engage in stabilizing interactions through co-axial stacking.

If `dangles` = 3 co-axial stacking is explicitly included for adjacent helices in multiloops. The option affects only mfe folding and energy evaluation (`fold()` and `energy_of_structure()`), as well as suboptimal folding (`subopt()`) via re-evaluation of energies. Co-axial stacking with one intervening mismatch is not considered so far.

Default is 2 in most algorithms, partition function algorithms can only handle 0 and 2

16.6.5.4 tetra_loop

```
int tetra_loop [extern]
```

```
#include <ViennaRNA/model.h>
```

Include special stabilizing energies for some tri-, tetra- and hexa-loops;. default is 1.

16.6.5.5 noLonelyPairs

```
int noLonelyPairs [extern]
```

```
#include <ViennaRNA/model.h>
```

Global switch to avoid/allow helices of length 1.

Disallow all pairs which can only occur as lonely pairs (i.e. as helix of length 1). This avoids lonely base pairs in the predicted structures in most cases.

16.6.5.6 energy_set

```
int energy_set [extern]
```

```
#include <ViennaRNA/model.h>
```

0 = BP; 1=any with GC; 2=any with AU-parameter

If set to 1 or 2: fold sequences from an artificial alphabet ABCD..., where A pairs B, C pairs D, etc. using either GC (1) or AU parameters (2); default is 0, you probably don't want to change it.

16.6.5.7 do_backtrack

```
int do_backtrack [extern]
```

```
#include <ViennaRNA/model.h>
```

do backtracking, i.e. compute secondary structures or base pair probabilities

If 0, do not calculate pair probabilities in [pf_fold\(\)](#); this is about twice as fast. Default is 1.

16.6.5.8 backtrack_type

```
char backtrack_type [extern]
```

```
#include <ViennaRNA/model.h>
```

A backtrack array marker for [inverse_fold\(\)](#)

If set to 'C': force (1,N) to be paired, 'M' fold as if the sequence were inside a multiloop. Otherwise ('F') the usual mfe structure is computed.

16.6.5.9 nonstandards

```
char* nonstandards [extern]
```

```
#include <ViennaRNA/model.h>
```

contains allowed non standard base pairs

Lists additional base pairs that will be allowed to form in addition to GC, CG, AU, UA, GU and UG. Nonstandard base pairs are given a stacking energy of 0.

16.6.5.10 max_bp_span

```
int max_bp_span [extern]
```

```
#include <ViennaRNA/model.h>
```

Maximum allowed base pair span.

A value of -1 indicates no restriction for distant base pairs.

16.7 Energy Parameters

All relevant functions to retrieve and copy pre-calculated energy parameter sets as well as reading/writing the energy parameter set from/to file(s).

16.7.1 Detailed Description

All relevant functions to retrieve and copy pre-calculated energy parameter sets as well as reading/writing the energy parameter set from/to file(s).

All relevant functions to compute salt correction at a given salt concentration and temperature.

This module covers all relevant functions for pre-calculation of the energy parameters necessary for the folding routines provided by RNAlib. Furthermore, the energy parameter set in the RNAlib can be easily exchanged by a user-defined one. It is also possible to write the current energy parameter set into a text file.

The corrections for loop and stack are taken from Einert and Netz, 2011 All corrections returned are in dcal/mol
Collaboration diagram for Energy Parameters:

Modules

- [Reading/Writing Energy Parameter Sets from/to File](#)

Read and Write energy parameter sets from and to files or strings.

Files

- file [basic.h](#)
Functions to deal with sets of energy parameters.
- file [constants.h](#)
Energy parameter constants.
- file [convert.h](#)
Functions and definitions for energy parameter file format conversion.
- file [io.h](#)
Read and write energy parameter files.
- file [salt.h](#)
Functions to compute salt correction.

Data Structures

- struct [vrna_param_s](#)
The datastructure that contains temperature scaled energy parameters. [More...](#)
- struct [vrna_exp_param_s](#)
The data structure that contains temperature scaled Boltzmann weights of the energy parameters. [More...](#)

Typedefs

- typedef struct [vrna_param_s](#) **vrna_param_t**
Typename for the free energy parameter data structure [vrna_params](#).
- typedef struct [vrna_exp_param_s](#) **vrna_exp_param_t**
Typename for the Boltzmann factor data structure [vrna_exp_params](#).
- typedef struct [vrna_param_s](#) paramT
Old typename of [vrna_param_s](#).
- typedef struct [vrna_exp_param_s](#) pf_paramT
Old typename of [vrna_exp_param_s](#).

Functions

- [vrna_param_t](#) * [vrna_params](#) ([vrna_md_t](#) *md)
Get a data structure containing prescaled free energy parameters.
- [vrna_param_t](#) * [vrna_params_copy](#) ([vrna_param_t](#) *par)
Get a copy of the provided free energy parameters.
- [vrna_exp_param_t](#) * [vrna_exp_params](#) ([vrna_md_t](#) *md)
Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors.
- [vrna_exp_param_t](#) * [vrna_exp_params_comparative](#) (unsigned int n_seq, [vrna_md_t](#) *md)
Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors (alifold version)
- [vrna_exp_param_t](#) * [vrna_exp_params_copy](#) ([vrna_exp_param_t](#) *par)
Get a copy of the provided free energy parameters (provided as Boltzmann factors)
- void [vrna_params_subst](#) ([vrna_fold_compound_t](#) *vc, [vrna_param_t](#) *par)
Update/Reset energy parameters data structure within a [vrna_fold_compound_t](#).
- void [vrna_exp_params_subst](#) ([vrna_fold_compound_t](#) *vc, [vrna_exp_param_t](#) *params)
Update the energy parameters for subsequent partition function computations.
- void [vrna_exp_params_rescale](#) ([vrna_fold_compound_t](#) *vc, double *mfe)
Rescale Boltzmann factors for partition function computations.
- void [vrna_params_reset](#) ([vrna_fold_compound_t](#) *vc, [vrna_md_t](#) *md_p)
Reset free energy parameters within a [vrna_fold_compound_t](#) according to provided, or default model details.
- void [vrna_exp_params_reset](#) ([vrna_fold_compound_t](#) *vc, [vrna_md_t](#) *md_p)

Reset Boltzmann factors for partition function computations within a `vrna_fold_compound_t` according to provided, or default model details.

- `vrna_exp_param_t * get_scaled_pf_parameters` (void)
- `vrna_exp_param_t * get_boltzmann_factors` (double `temperature`, double `betaScale`, `vrna_md_t` `md`, double `pf_scale`)

Get precomputed Boltzmann factors of the loop type dependent energy contributions with independent thermodynamic temperature.

- `vrna_exp_param_t * get_boltzmann_factor_copy` (`vrna_exp_param_t *`parameters)

Get a copy of already precomputed Boltzmann factors.

- `vrna_exp_param_t * get_scaled_alipf_parameters` (unsigned int `n_seq`)

Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant)

- `vrna_exp_param_t * get_boltzmann_factors_ali` (unsigned int `n_seq`, double `temperature`, double `betaScale`, `vrna_md_t` `md`, double `pf_scale`)

Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant) with independent thermodynamic temperature.

- `vrna_param_t * scale_parameters` (void)

Get precomputed energy contributions for all the known loop types.

- `vrna_param_t * get_scaled_parameters` (double `temperature`, `vrna_md_t` `md`)

Get precomputed energy contributions for all the known loop types.

- double `vrna_salt_loop` (int `L`, double `salt`, double `T`)

Get salt correction for a loop at a given salt concentration and temperature.

- int `vrna_salt_loop_int` (int `L`, double `salt`, double `T`)

Get salt correction for a loop at a given salt concentration and temperature.

- int `vrna_salt_stack` (double `salt`, double `T`)

Get salt correction for a stack at a given salt concentration and temperature.

16.7.2 Data Structure Documentation

16.7.2.1 struct vrna_param_s

The datastructure that contains temperature scaled energy parameters.

Collaboration diagram for `vrna_param_s`:

Data Fields

- double **temperature**
Temperature used for loop contribution scaling.
- `vrna_md_t` **model_details**
Model details to be used in the recursions.
- char **param_file** [256]
The filename the parameters were derived from, or empty string if they represent the default.

16.7.2.2 struct vrna_exp_param_s

The data structure that contains temperature scaled Boltzmann weights of the energy parameters.

Collaboration diagram for `vrna_exp_param_s`:

Data Fields

- int `id`
An identifier for the data structure.
- double **pf_scale**
Scaling factor to avoid over-/underflows.
- double **temperature**

- double `alpha`
Temperature used for loop contribution scaling.
- `vrna_md_t model_details`
Scaling factor for the thermodynamic temperature.
Model details to be used in the recursions.
- char `param_file` [256]
The filename the parameters were derived from, or empty string if they represent the default.

16.7.2.2.1 Field Documentation

16.7.2.2.1.1 `id` `int vrna_exp_param_s::id`

An identifier for the data structure.

Deprecated This attribute will be removed in version 3

16.7.2.2.1.2 `alpha` `double vrna_exp_param_s::alpha`

Scaling factor for the thermodynamic temperature.

This allows for temperature scaling in Boltzmann factors independently from the energy contributions. The resulting Boltzmann factors are then computed by $e^{-E/(\alpha \cdot K \cdot T)}$

16.7.3 Typedef Documentation

16.7.3.1 `paramT`

```
typedef struct vrna_param_s paramT
#include <ViennaRNA/params/basic.h>
Old typename of vrna_param_s.
```

Deprecated Use `vrna_param_t` instead!

16.7.3.2 `pf_paramT`

```
typedef struct vrna_exp_param_s pf_paramT
#include <ViennaRNA/params/basic.h>
Old typename of vrna_exp_param_s.
```

Deprecated Use `vrna_exp_param_t` instead!

16.7.4 Function Documentation

16.7.4.1 `vrna_params()`

```
vrna_param_t * vrna_params (
    vrna_md_t * md )
#include <ViennaRNA/params/basic.h>
```

Get a data structure containing prescaled free energy parameters.

If a NULL pointer is passed for the model details parameter, the default model parameters are stored within the requested `vrna_param_t` structure.

See also

`vrna_md_t`, `vrna_md_set_default()`, `vrna_exp_params()`

Parameters

<i>md</i>	A pointer to the model details to store inside the structure (Maybe NULL)
-----------	---

Returns

A pointer to the memory location where the requested parameters are stored

16.7.4.2 `vrna_params_copy()`

```
vrna_param_t * vrna_params_copy (
    vrna_param_t * par )
```

```
#include <ViennaRNA/params/basic.h>
```

Get a copy of the provided free energy parameters.

If NULL is passed as parameter, a default set of energy parameters is created and returned.

See also

[vrna_params\(\)](#), [vrna_param_t](#)

Parameters

<i>par</i>	The free energy parameters that are to be copied (Maybe NULL)
------------	---

Returns

A copy or a default set of the (provided) parameters

16.7.4.3 `vrna_exp_params()`

```
vrna_exp_param_t * vrna_exp_params (
    vrna_md_t * md )
```

```
#include <ViennaRNA/params/basic.h>
```

Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors.

This function returns a data structure that contains all necessary precomputed energy contributions for each type of loop.

In contrast to [vrna_params\(\)](#), the free energies within this data structure are stored as their Boltzmann factors, i.e.

$\exp(-E/kT)$

where E is the free energy.

If a NULL pointer is passed for the model details parameter, the default model parameters are stored within the requested [vrna_exp_param_t](#) structure.

See also

[vrna_md_t](#), [vrna_md_set_default\(\)](#), [vrna_params\(\)](#), [vrna_rescale_pf_params\(\)](#)

Parameters

<i>md</i>	A pointer to the model details to store inside the structure (Maybe NULL)
-----------	---

Returns

A pointer to the memory location where the requested parameters are stored

16.7.4.4 `vrna_exp_params_comparative()`

```
vrna_exp_param_t * vrna_exp_params_comparative (
    unsigned int n_seq,
    vrna_md_t * md )
```

```
#include <ViennaRNA/params/basic.h>
```

Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors (alifold version)

If a NULL pointer is passed for the model details parameter, the default model parameters are stored within the requested `vrna_exp_param_t` structure.

See also

[vrna_md_t](#), [vrna_md_set_default\(\)](#), [vrna_exp_params\(\)](#), [vrna_params\(\)](#)

Parameters

<code>n_seq</code>	The number of sequences in the alignment
<code>md</code>	A pointer to the model details to store inside the structure (Maybe NULL)

Returns

A pointer to the memory location where the requested parameters are stored

16.7.4.5 `vrna_exp_params_copy()`

```
vrna_exp_param_t * vrna_exp_params_copy (
    vrna_exp_param_t * par )
```

```
#include <ViennaRNA/params/basic.h>
```

Get a copy of the provided free energy parameters (provided as Boltzmann factors)

If NULL is passed as parameter, a default set of energy parameters is created and returned.

See also

[vrna_exp_params\(\)](#), [vrna_exp_param_t](#)

Parameters

<code>par</code>	The free energy parameters that are to be copied (Maybe NULL)
------------------	---

Returns

A copy or a default set of the (provided) parameters

16.7.4.6 `vrna_params_subst()`

```
void vrna_params_subst (
    vrna_fold_compound_t * vc,
    vrna_param_t * par )
```

```
#include <ViennaRNA/params/basic.h>
```

Update/Reset energy parameters data structure within a [vrna_fold_compound_t](#).

Passing NULL as second argument leads to a reset of the energy parameters within vc to their default values.

Otherwise, the energy parameters provided will be copied over into vc.

See also

[vrna_params_reset\(\)](#), [vrna_param_t](#), [vrna_md_t](#), [vrna_params\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t that is about to receive updated energy parameters
<i>par</i>	The energy parameters used to substitute those within <i>vc</i> (Maybe NULL)

SWIG Wrapper Notes This function is attached to [vrna_fc_s](#) objects as overloaded **params_subst()** method.

When no parameter is passed, the resulting action is the same as passing *NULL* as second parameter to [vrna_params_subst\(\)](#), i.e. resetting the parameters to the global defaults.

16.7.4.7 vrna_exp_params_subst()

```
void vrna_exp_params_subst (
    vrna_fold_compound_t * vc,
    vrna_exp_param_t * params )
#include <ViennaRNA/params/basic.h>
```

Update the energy parameters for subsequent partition function computations.

This function can be used to properly assign new energy parameters for partition function computations to a [vrna_fold_compound_t](#). For this purpose, the data of the provided pointer *params* will be copied into *vc* and a recomputation of the partition function scaling factor is issued, if the *pf_scale* attribute of *params* is less than 1.0.

Passing NULL as second argument leads to a reset of the energy parameters within *vc* to their default values

See also

[vrna_exp_params_reset\(\)](#), [vrna_exp_params_rescale\(\)](#), [vrna_exp_param_t](#), [vrna_md_t](#), [vrna_exp_params\(\)](#)

Parameters

<i>vc</i>	The fold compound data structure
<i>params</i>	A pointer to the new energy parameters

SWIG Wrapper Notes This function is attached to [vrna_fc_s](#) objects as overloaded **exp_params_subst()** method.

When no parameter is passed, the resulting action is the same as passing *NULL* as second parameter to [vrna_exp_params_subst\(\)](#), i.e. resetting the parameters to the global defaults.

16.7.4.8 vrna_exp_params_rescale()

```
void vrna_exp_params_rescale (
    vrna_fold_compound_t * vc,
    double * mfe )
#include <ViennaRNA/params/basic.h>
```

Rescale Boltzmann factors for partition function computations.

This function may be used to (automatically) rescale the Boltzmann factors used in partition function computations. Since partition functions over subsequences can easily become extremely large, the RNALib internally rescales them to avoid numerical over- and/or underflow. Therefore, a proper scaling factor *s* needs to be chosen that in turn is then used to normalize the corresponding partition functions $\hat{q}[i, j] = q[i, j]/s^{(j-i+1)}$.

This function provides two ways to automatically adjust the scaling factor.

1. Automatic guess

2. Automatic adjustment according to MFE

Passing `NULL` as second parameter activates the *automatic guess mode*. Here, the scaling factor is recomputed according to a mean free energy of `184.3*length` cal for random sequences.

Note

This recomputation only takes place if the `pf_scale` attribute of the `exp_params` data structure contained in `vc` has a value below `1.0`.

On the other hand, if the MFE for a sequence is known, it can be used to recompute a more robust scaling factor, since it represents the lowest free energy of the entire ensemble of structures, i.e. the highest Boltzmann factor. To activate this second mode of *automatic adjustment according to MFE*, a pointer to the MFE value needs to be passed as second argument. This value is then taken to compute the scaling factor as $s = \exp((s_{fact} * MFE)/kT/length)$, where `sfact` is an additional scaling weight located in the `vrna_md_t` data structure of `exp_params` in `vc`.

The computed scaling factor s will be stored as `pf_scale` attribute of the `exp_params` data structure in `vc`.

See also

[vrna_exp_params_subst\(\)](#), [vrna_md_t](#), [vrna_exp_param_t](#), [vrna_fold_compound_t](#)

Parameters

<code>vc</code>	The fold compound data structure
<code>mfe</code>	A pointer to the MFE (in kcal/mol) or <code>NULL</code>

SWIG Wrapper Notes This function is attached to `vrna_fc_s` objects as overloaded `exp_params_rescale()` method.

When no parameter is passed to this method, the resulting action is the same as passing `NULL` as second parameter to `vrna_exp_params_rescale()`, i.e. default scaling of the partition function. Passing an energy in kcal/mol, e.g. as retrieved by a previous call to the `mfe()` method, instructs all subsequent calls to scale the partition function accordingly.

16.7.4.9 vrna_params_reset()

```
void vrna_params_reset (
    vrna_fold_compound_t * vc,
    vrna_md_t * md_p )
#include <ViennaRNA/params/basic.h>
```

Reset free energy parameters within a `vrna_fold_compound_t` according to provided, or default model details.

This function allows one to rescale free energy parameters for subsequent structure prediction or evaluation according to a set of model details, e.g. temperature values. To do so, the caller provides either a pointer to a set of model details to be used for rescaling, or `NULL` if global default setting should be used.

See also

[vrna_exp_params_reset\(\)](#), [vrna_params_subs\(\)](#)

Parameters

<code>vc</code>	The fold compound data structure
<code>md_p</code>	A pointer to the new model details (or <code>NULL</code> for reset to defaults)

SWIG Wrapper Notes This function is attached to `vrna_fc_s` objects as overloaded `params_reset()` method.

When no parameter is passed to this method, the resulting action is the same as passing *NULL* as second parameter to [vrna_params_reset\(\)](#), i.e. global default model settings are used. Passing an object of type [vrna_md_s](#) resets the fold compound according to the specifications stored within the [vrna_md_s](#) object.

16.7.4.10 vrna_exp_params_reset()

```
vrna_exp_params_reset (
    vrna_fold_compound_t * vc,
    vrna_md_t * md_p )
#include <ViennaRNA/params/basic.h>
```

Reset Boltzmann factors for partition function computations within a [vrna_fold_compound_t](#) according to provided, or default model details.

This function allows one to rescale Boltzmann factors for subsequent partition function computations according to a set of model details, e.g. temperature values. To do so, the caller provides either a pointer to a set of model details to be used for rescaling, or *NULL* if global default setting should be used.

See also

[vrna_params_reset\(\)](#), [vrna_exp_params_subst\(\)](#), [vrna_exp_params_rescale\(\)](#)

Parameters

<i>vc</i>	The fold compound data structure
<i>md_p</i>	A pointer to the new model details (or <i>NULL</i> for reset to defaults)

SWIG Wrapper Notes This function is attached to [vrna_fc_s](#) objects as overloaded **exp_params_reset()** method.

When no parameter is passed to this method, the resulting action is the same as passing *NULL* as second parameter to [vrna_exp_params_reset\(\)](#), i.e. global default model settings are used. Passing an object of type [vrna_md_s](#) resets the fold compound according to the specifications stored within the [vrna_md_s](#) object.

16.7.4.11 get_scaled_pf_parameters()

```
vrna_exp_param_t * get_scaled_pf_parameters (
    void )
#include <ViennaRNA/params/basic.h>
```

get a data structure of type [vrna_exp_param_t](#) which contains the Boltzmann weights of several energy parameters scaled according to the current temperature

Deprecated Use [vrna_exp_params\(\)](#) instead!

Returns

The data structure containing Boltzmann weights for use in partition function calculations

16.7.4.12 get_boltzmann_factors()

```
vrna_exp_param_t * get_boltzmann_factors (
    double temperature,
    double betaScale,
```

```

    vrna_md_t md,
    double pf_scale )
#include <ViennaRNA/params/basic.h>

```

Get precomputed Boltzmann factors of the loop type dependent energy contributions with independent thermodynamic temperature.

This function returns a data structure that contains all necessary precalculated Boltzmann factors for each loop type contribution.

In contrast to [get_scaled_pf_parameters\(\)](#), this function enables setting of independent temperatures for both, the individual energy contributions as well as the thermodynamic temperature used in $\exp(-\Delta G/kT)$

Deprecated Use [vrna_exp_params\(\)](#) instead!

See also

[get_scaled_pf_parameters\(\)](#), [get_boltzmann_factor_copy\(\)](#)

Parameters

<i>temperature</i>	The temperature in degrees Celcius used for (re-)scaling the energy contributions
<i>betaScale</i>	A scaling value that is used as a multiplication factor for the absolute temperature of the system
<i>md</i>	The model details to be used
<i>pf_scale</i>	The scaling factor for the Boltzmann factors

Returns

A set of precomputed Boltzmann factors

16.7.4.13 [get_boltzmann_factor_copy\(\)](#)

```

vrna_exp_param_t * get_boltzmann_factor_copy (
    vrna_exp_param_t * parameters )
#include <ViennaRNA/params/basic.h>

```

Get a copy of already precomputed Boltzmann factors.

Deprecated Use [vrna_exp_params_copy\(\)](#) instead!

See also

[get_boltzmann_factors\(\)](#), [get_scaled_pf_parameters\(\)](#)

Parameters

<i>parameters</i>	The input data structure that shall be copied
-------------------	---

Returns

A copy of the provided Boltzmann factor data set

16.7.4.14 [get_scaled_alipf_parameters\(\)](#)

```

vrna_exp_param_t * get_scaled_alipf_parameters (
    unsigned int n_seq )
#include <ViennaRNA/params/basic.h>

```


Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant)

Deprecated Use [vrna_exp_params_comparative\(\)](#) instead!

16.7.4.15 `get_boltzmann_factors_ali()`

```
vrna_exp_param_t * get_boltzmann_factors_ali (
    unsigned int n_seq,
    double temperature,
    double betaScale,
    vrna_md_t md,
    double pf_scale )
```

```
#include <ViennaRNA/params/basic.h>
```

Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant) with independent thermodynamic temperature.

Deprecated Use [vrna_exp_params_comparative\(\)](#) instead!

16.7.4.16 `scale_parameters()`

```
vrna_param_t * scale_parameters (
    void )
```

```
#include <ViennaRNA/params/basic.h>
```

Get precomputed energy contributions for all the known loop types.

Note

OpenMP: This function relies on several global model settings variables and thus is not to be considered threadsafe. See [get_scaled_parameters\(\)](#) for a completely threadsafe implementation.

Deprecated Use [vrna_params\(\)](#) instead!

Returns

A set of precomputed energy contributions

16.7.4.17 `get_scaled_parameters()`

```
vrna_param_t * get_scaled_parameters (
    double temperature,
    vrna_md_t md )
```

```
#include <ViennaRNA/params/basic.h>
```

Get precomputed energy contributions for all the known loop types.

Call this function to retrieve precomputed energy contributions, i.e. scaled according to the temperature passed. Furthermore, this function assumes a data structure that contains the model details as well, such that subsequent folding recursions are able to retrieve the correct model settings

Deprecated Use [vrna_params\(\)](#) instead!

See also

[vrna_md_t, set_model_details\(\)](#)

Parameters

<i>temperature</i>	The temperature in degrees Celcius
<i>md</i>	The model details

Returns

precomputed energy contributions and model settings

16.7.4.18 vrna_salt_loop()

```
double vrna_salt_loop (
    int L,
    double salt,
    double T )
#include <ViennaRNA/params/salt.h>
```

Get salt correction for a loop at a given salt concentration and temperature.

Parameters

<i>L</i>	backbone number in loop
<i>salt</i>	salt concentration (M)
<i>T</i>	absolute temperature (K)

Returns

Salt correction for loop in dcal/mol

16.7.4.19 vrna_salt_loop_int()

```
int vrna_salt_loop_int (
    int L,
    double salt,
    double T )
#include <ViennaRNA/params/salt.h>
```

Get salt correction for a loop at a given salt concentration and temperature.

This functions is same as vrna_salt_loop but returns rounded salt correction in integer

See also

[vrna_salt_loop](#)

Parameters

<i>L</i>	backbone number in loop
<i>salt</i>	salt concentration (M)
<i>T</i>	absolute temperature (K)

Returns

Rounded salt correction for loop in dcal/mol

16.7.4.20 vrna_salt_stack()

```
int vrna_salt_stack (
    double salt,
    double T )
#include <ViennaRNA/params/salt.h>
```

Get salt correction for a stack at a given salt concentration and temperature.

Parameters

<i>salt</i>	salt concentration (M)
<i>T</i>	absolute temperature (K)

Returns

Rounded salt correction for stack in dcal/mol

16.8 Extending the Folding Grammar with Additional Domains

This module covers simple and straight-forward extensions to the RNA folding grammar.

16.8.1 Detailed Description

This module covers simple and straight-forward extensions to the RNA folding grammar.
 Collaboration diagram for Extending the Folding Grammar with Additional Domains:

Modules

- [Unstructured Domains](#)
Add and modify unstructured domains to the RNA folding grammar.
- [Structured Domains](#)
Add and modify structured domains to the RNA folding grammar.

16.9 Unstructured Domains

Add and modify unstructured domains to the RNA folding grammar.

16.9.1 Detailed Description

Add and modify unstructured domains to the RNA folding grammar.

This module provides the tools to add and modify unstructured domains to the production rules of the RNA folding grammar. Usually this functionality is utilized for incorporating ligand binding to unpaired stretches of an RNA.

Bug Although the additional production rule(s) for unstructured domains as described in [Unstructured Domains](#) are always treated as 'segments possibly bound to one or more ligands', the current implementation requires that at least one ligand is bound. The default implementation already takes care of the required changes, however, upon using callback functions other than the default ones, one has to take care of this fact. Please also note, that this behavior might change in one of the next releases, such that the decomposition schemes as shown above comply with the actual implementation.

A default implementation allows one to readily use this feature by simply adding sequence motifs and corresponding binding free energies with the function [vrna_ud_add_motif\(\)](#) (see also [Ligands Binding to Unstructured Domains](#)). The grammar extension is realized using a callback function that

- evaluates the binding free energy of a ligand to its target sequence segment (white boxes in the figures above), or
- returns the free energy of an unpaired stretch possibly bound by a ligand, stored in the additional *UDP* matrix.

The callback is passed the segment positions, the loop context, and which of the two above mentioned evaluations are required. A second callback implements the pre-processing step that prepares the *UDP* matrix by evaluating all possible cases of the additional production rule. Both callbacks have a default implementation in *RNAlib*, but may be over-written by a user-implementation, making it fully user-customizable.

For equilibrium probability computations, two additional callbacks exist. One to store/add and one to retrieve the probability of unstructured domains at particular positions. Our implementation already takes care of computing the probabilities, but users of the unstructured domain feature are required to provide a mechanism to efficiently store/add the corresponding values into some external data structure. Collaboration diagram for Unstructured Domains:

Files

- file [unstructured_domains.h](#)

Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches.

Data Structures

- struct [vrna_unstructured_domain_s](#)

Data structure to store all functionality for ligand binding. [More...](#)

Macros

- `#define VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP 1U`
Flag to indicate ligand bound to unpaired stretch in the exterior loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP 2U`
Flag to indicate ligand bound to unpaired stretch in a hairpin loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP 4U`
Flag to indicate ligand bound to unpaired stretch in an interior loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP 8U`
Flag to indicate ligand bound to unpaired stretch in a multibranch loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_MOTIF 16U`
Flag to indicate ligand binding without additional unbound nucleotides (motif-only)
- `#define VRNA_UNSTRUCTURED_DOMAIN_ALL_LOOPS`
Flag to indicate ligand bound to unpaired stretch in any loop (convenience macro)

Typedefs

- `typedef struct vrna_unstructured_domain_s vrna_ud_t`
Typename for the ligand binding extension data structure [vrna_unstructured_domain_s](#).
- `typedef int(* vrna_ud_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`
Callback to retrieve binding free energy of a ligand bound to an unpaired sequence segment.
- `typedef FLT_OR_DBL(* vrna_ud_exp_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`
Callback to retrieve Boltzmann factor of the binding free energy of a ligand bound to an unpaired sequence segment.
- `typedef void(* vrna_ud_production_f) (vrna_fold_compound_t *vc, void *data)`
Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature.
- `typedef void(* vrna_ud_exp_production_f) (vrna_fold_compound_t *vc, void *data)`
Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature (partition function variant)
- `typedef void(* vrna_ud_add_probs_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, FLT_OR_DBL exp_energy, void *data)`
Callback to store/add equilibrium probability for a ligand bound to an unpaired sequence segment.
- `typedef FLT_OR_DBL(* vrna_ud_get_probs_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, int motif, void *data)`
Callback to retrieve equilibrium probability for a ligand bound to an unpaired sequence segment.

Functions

- `vrna_ud_motif_t * vrna_ud_motifs_centroid (vrna_fold_compound_t *fc, const char *structure)`
Detect unstructured domains in centroid structure.
- `vrna_ud_motif_t * vrna_ud_motifs_MEA (vrna_fold_compound_t *fc, const char *structure, vrna_ep_t *probability_list)`
Detect unstructured domains in MEA structure.
- `vrna_ud_motif_t * vrna_ud_motifs_MFE (vrna_fold_compound_t *fc, const char *structure)`
Detect unstructured domains in MFE structure.
- `void vrna_ud_add_motif (vrna_fold_compound_t *vc, const char *motif, double motif_en, const char *motif_name, unsigned int loop_type)`
Add an unstructured domain motif, e.g. for ligand binding.
- `void vrna_ud_remove (vrna_fold_compound_t *vc)`
Remove ligand binding to unpaired stretches.
- `void vrna_ud_set_data (vrna_fold_compound_t *vc, void *data, vrna_auxdata_free_f free_cb)`
Attach an auxiliary data structure.
- `void vrna_ud_set_prod_rule_cb (vrna_fold_compound_t *vc, vrna_ud_production_f pre_cb, vrna_ud_f e_cb)`
Attach production rule callbacks for free energies computations.
- `void vrna_ud_set_exp_prod_rule_cb (vrna_fold_compound_t *vc, vrna_ud_exp_production_f pre_cb, vrna_ud_exp_f exp_e_cb)`
Attach production rule for partition function.

16.9.2 Data Structure Documentation

16.9.2.1 struct vrna_unstructured_domain_s

Data structure to store all functionality for ligand binding.

Collaboration diagram for `vrna_unstructured_domain_s`:

Data Fields

- `int uniq_motif_count`
The unique number of motifs of different lengths.
- `unsigned int * uniq_motif_size`
An array storing a unique list of motif lengths.
- `int motif_count`
Total number of distinguished motifs.
- `char ** motif`
Motif sequences.
- `char ** motif_name`
Motif identifier/name.
- `unsigned int * motif_size`
Motif lengths.
- `double * motif_en`
Ligand binding free energy contribution.
- `unsigned int * motif_type`
Type of motif, i.e. loop type the ligand binds to.
- `vrna_ud_production_f prod_cb`
Callback to ligand binding production rule, i.e. create/fill DP free energy matrices.
- `vrna_ud_exp_production_f exp_prod_cb`
Callback to ligand binding production rule, i.e. create/fill DP partition function matrices.
- `vrna_ud_f energy_cb`
Callback to evaluate free energy of ligand binding to a particular unpaired stretch.

- [vrna_ud_exp_f](#) **exp_energy_cb**
Callback to evaluate Boltzmann factor of ligand binding to a particular unpaired stretch.
- void * **data**
Auxiliary data structure passed to energy evaluation callbacks.
- [vrna_auxdata_free_f](#) **free_data**
Callback to free auxiliary data structure.
- [vrna_ud_add_probs_f](#) **probs_add**
Callback to store/add outside partition function.
- [vrna_ud_get_probs_f](#) **probs_get**
Callback to retrieve outside partition function.

16.9.2.1.1 Field Documentation

16.9.2.1.1.1 prod_cb [vrna_ud_production_f](#) `vrna_unstructured_domain_s::prod_cb`

Callback to ligand binding production rule, i.e. create/fill DP free energy matrices.

This callback will be executed right before the actual secondary structure decompositions, and, therefore, any implementation must not interleave with the regular DP matrices.

16.9.3 Typedef Documentation

16.9.3.1 vrna_ud_f

```
typedef int(* vrna_ud_f) (vrna\_fold\_compound\_t *vc, int i, int j, unsigned int loop_type, void *data)
```

```
#include <ViennaRNA/unstructured\_domains.h>
```

Callback to retrieve binding free energy of a ligand bound to an unpaired sequence segment.

Notes on Callback Functions This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand.

Parameters

<i>vc</i>	The current vrna_fold_compound_t
<i>i</i>	The start of the unstructured domain (5' end)
<i>j</i>	The end of the unstructured domain (3' end)
<i>loop_type</i>	The loop context of the unstructured domain
<i>data</i>	Auxiliary data

Returns

The auxiliary energy contribution in deka-cal/mol

16.9.3.2 vrna_ud_exp_f

```
typedef FLT\_OR\_DBL(* vrna_ud_exp_f) (vrna\_fold\_compound\_t *vc, int i, int j, unsigned int loop_type, void *data)
```

```
#include <ViennaRNA/unstructured\_domains.h>
```

Callback to retrieve Boltzmann factor of the binding free energy of a ligand bound to an unpaired sequence segment.

Notes on Callback Functions This function will be called to determine the additional energy contribution of a specific unstructured domain, e.g. the binding free energy of some ligand (Partition function variant, i.e. the Boltzmann factors instead of actual free energies).

Parameters

<i>vc</i>	The current vrna_fold_compound_t
<i>i</i>	The start of the unstructured domain (5' end)
<i>j</i>	The end of the unstructured domain (3' end)
<i>loop_type</i>	The loop context of the unstructured domain
<i>data</i>	Auxiliary data

Returns

The auxiliary energy contribution as Boltzmann factor

16.9.3.3 vrna_ud_production_f

```
typedef void(* vrna_ud_production_f) (vrna_fold_compound_t *vc, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature.

Notes on Callback Functions The production rule for the unstructured domain grammar extension

16.9.3.4 vrna_ud_exp_production_f

```
typedef void(* vrna_ud_exp_production_f) (vrna_fold_compound_t *vc, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature (partition function variant)

Notes on Callback Functions The production rule for the unstructured domain grammar extension (Partition function variant)

16.9.3.5 vrna_ud_add_probs_f

```
typedef void(* vrna_ud_add_probs_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int
```

```
loop_type, FLT_OR_DBL exp_energy, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback to store/add equilibrium probability for a ligand bound to an unpaired sequence segment.

Notes on Callback Functions A callback function to store equilibrium probabilities for the unstructured domain feature

16.9.3.6 vrna_ud_get_probs_f

```
typedef FLT_OR_DBL(* vrna_ud_get_probs_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int
```

```
loop_type, int motif, void *data)
```

```
#include <ViennaRNA/unstructured_domains.h>
```

Callback to retrieve equilibrium probability for a ligand bound to an unpaired sequence segment.

Notes on Callback Functions A callback function to retrieve equilibrium probabilities for the unstructured domain feature

16.9.4 Function Documentation

16.9.4.1 vrna_ud_motifs_centroid()

```
vrna_ud_motif_t * vrna_ud_motifs_centroid (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/unstructured_domains.h>
```

Detect unstructured domains in centroid structure.

Given a centroid structure and a set of unstructured domains compute the list of unstructured domain motifs present in the centroid. Since we do not explicitly annotate unstructured domain motifs in dot-bracket strings, this function can be used to check for the presence and location of unstructured domain motifs under the assumption that the dot-bracket string is the centroid structure of the equilibrium ensemble.

See also

[vrna_centroid\(\)](#)

Parameters

<i>fc</i>	The fold_compound data structure with pre-computed equilibrium probabilities and model settings
<i>structure</i>	The centroid structure in dot-bracket notation

Returns

A list of unstructured domain motifs (possibly NULL). The last element terminates the list with `start=0`, `number=-1`

16.9.4.2 vrna_ud_motifs_MEA()

```
vrna_ud_motif_t * vrna_ud_motifs_MEA (
    vrna_fold_compound_t * fc,
    const char * structure,
    vrna_ep_t * probability_list )
#include <ViennaRNA/unstructured_domains.h>
```

Detect unstructured domains in MEA structure.

Given an MEA structure and a set of unstructured domains compute the list of unstructured domain motifs present in the MEA structure. Since we do not explicitly annotate unstructured domain motifs in dot-bracket strings, this function can be used to check for the presence and location of unstructured domain motifs under the assumption that the dot-bracket string is the MEA structure of the equilibrium ensemble.

See also

[MEA\(\)](#)

Parameters

<i>fc</i>	The fold_compound data structure with pre-computed equilibrium probabilities and model settings
<i>structure</i>	The MEA structure in dot-bracket notation
<i>probability_list</i>	The list of probabilities to extract the MEA structure from

Returns

A list of unstructured domain motifs (possibly NULL). The last element terminates the list with `start=0`, `number=-1`

16.9.4.3 `vrna_ud_motifs_MFE()`

```
vrna_ud_motif_t * vrna_ud_motifs_MFE (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/unstructured_domains.h>
```

Detect unstructured domains in MFE structure.

Given an MFE structure and a set of unstructured domains compute the list of unstructured domain motifs present in the MFE structure. Since we do not explicitly annotate unstructured domain motifs in dot-bracket strings, this function can be used to check for the presence and location of unstructured domain motifs under the assumption that the dot-bracket string is the MFE structure of the equilibrium ensemble.

See also

[vrna_mfe\(\)](#)

Parameters

<i>fc</i>	The fold_compound data structure with model settings
<i>structure</i>	The MFE structure in dot-bracket notation

Returns

A list of unstructured domain motifs (possibly NULL). The last element terminates the list with `start=0`, `number=-1`

16.9.4.4 `vrna_ud_add_motif()`

```
void vrna_ud_add_motif (
    vrna_fold_compound_t * vc,
    const char * motif,
    double motif_en,
    const char * motif_name,
    unsigned int loop_type )
#include <ViennaRNA/unstructured_domains.h>
```

Add an unstructured domain motif, e.g. for ligand binding.

This function adds a ligand binding motif and the associated binding free energy to the `vrna_ud_t` attribute of a `vrna_fold_compound_t`. The motif data will then be used in subsequent secondary structure predictions. Multiple calls to this function with different motifs append all additional data to a list of ligands, which all will be evaluated. Ligand motif data can be removed from the `vrna_fold_compound_t` again using the `vrna_ud_remove()` function. The loop type parameter allows one to limit the ligand binding to particular loop type, such as the exterior loop, hairpin loops, interior loops, or multibranch loops.

See also

[VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP](#), [VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP](#), [VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP](#), [VRNA_UNSTRUCTURED_DOMAIN_ALL_LOOPS](#), [vrna_ud_remove\(\)](#)

Parameters

<i>vc</i>	The <code>vrna_fold_compound_t</code> data structure the ligand motif should be bound to
<i>motif</i>	The sequence motif the ligand binds to
<i>motif_en</i>	The binding free energy of the ligand in kcal/mol
<i>motif_name</i>	The name/id of the motif (may be NULL)
<i>loop_type</i>	The loop type the ligand binds to

16.9.4.5 vrna_ud_remove()

```
void vrna_ud_remove (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/unstructured_domains.h>
```

Remove ligand binding to unpaired stretches.

This function removes all ligand motifs that were bound to a `vrna_fold_compound_t` using the `vrna_ud_add_motif()` function.

Parameters

vc	The <code>vrna_fold_compound_t</code> data structure the ligand motif data should be removed from
----	---

SWIG Wrapper Notes This function is attached as method `ud_remove()` to objects of type `fold_compound`

16.9.4.6 vrna_ud_set_data()

```
void vrna_ud_set_data (
    vrna_fold_compound_t * vc,
    void * data,
    vrna_auxdata_free_f free_cb )
#include <ViennaRNA/unstructured_domains.h>
```

Attach an auxiliary data structure.

This function binds an arbitrary, auxiliary data structure for user-implemented ligand binding. The optional callback `free_cb` will be passed the bound data structure whenever the `vrna_fold_compound_t` is removed from memory to avoid memory leaks.

See also

`vrna_ud_set_prod_rule_cb()`, `vrna_ud_set_exp_prod_rule_cb()`, `vrna_ud_remove()`

Parameters

vc	The <code>vrna_fold_compound_t</code> data structure the auxiliary data structure should be bound to
data	A pointer to the auxiliary data structure
free_cb	A pointer to a callback function that free's memory occupied by <code>data</code>

SWIG Wrapper Notes This function is attached as method `ud_set_data()` to objects of type `fold_compound`

16.9.4.7 vrna_ud_set_prod_rule_cb()

```
void vrna_ud_set_prod_rule_cb (
    vrna_fold_compound_t * vc,
    vrna_ud_production_f pre_cb,
    vrna_ud_f e_cb )
#include <ViennaRNA/unstructured_domains.h>
```

Attach production rule callbacks for free energies computations.

Use this function to bind a user-implemented grammar extension for unstructured domains.

The callback `e_cb` needs to evaluate the free energy contribution $f(i, j)$ of the unpaired segment $[i, j]$. It will be executed in each of the regular secondary structure production rules. Whenever the callback is passed the `VRNA_UNSTRUCTURED_DOMAIN_MOTIF` flag via its `loop_type` parameter the contribution of

$$f(i,j) = \boxed{} \mid \overbrace{}^B$$

i *j* *i* *j*

See also

[vrna_ud_set_prod_rule_cb\(\)](#)

Parameters

<code>vc</code>	The vrna_fold_compound_t data structure the callback will be bound to
<code>pre_cb</code>	A pointer to a callback function for the B production rule
<code>exp_e_cb</code>	A pointer to a callback function that retrieves the partition function for a segment $[i, j]$ that may be bound by one or more ligands.

SWIG Wrapper Notes This function is attached as method `ud_set_exp_prod_rule_cb()` to objects of type `fold↔_compound`

16.10 Structured Domains

Add and modify structured domains to the RNA folding grammar.

16.10.1 Detailed Description

Add and modify structured domains to the RNA folding grammar.

This module provides the tools to add and modify structured domains to the production rules of the RNA folding grammar. Usually this functionality is utilized for incorporating self-enclosed structural modules that exhibit a more or less complex base pairing pattern. Collaboration diagram for Structured Domains:

Files

- file [structured_domains.h](#)

This module provides interfaces that deal with additional structured domains in the folding grammar.

16.11 Constraining the RNA Folding Grammar

This module provides general functions that allow for an easy control of constrained secondary structure prediction and evaluation.

16.11.1 Detailed Description

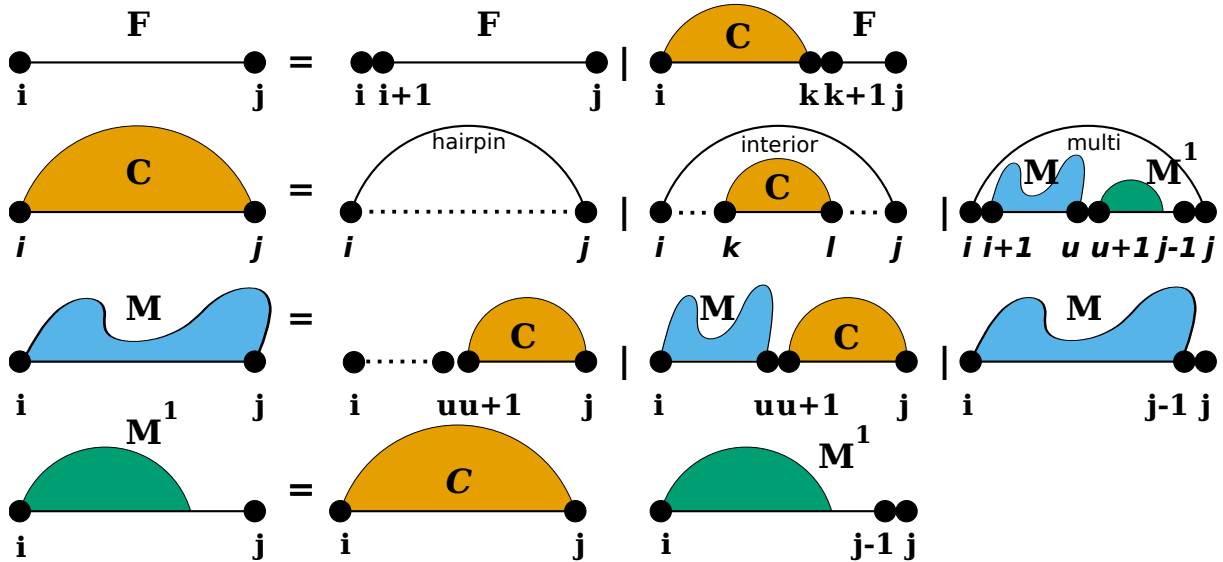
This module provides general functions that allow for an easy control of constrained secondary structure prediction and evaluation.

Secondary Structure constraints can be subdivided into two groups:

- [Hard Constraints](#), and
- [Soft Constraints](#).

While Hard-Constraints directly influence the production rules used in the folding recursions by allowing, disallowing, or enforcing certain decomposition steps, Soft-constraints on the other hand are used to change position specific contributions in the recursions by adding bonuses/penalties in form of pseudo free energies to certain loop configurations.

Secondary structure constraints are always applied at decomposition level, i.e. in each step of the recursive structure decomposition, for instance during MFE prediction. Below is a visualization of the decomposition scheme



For [Hard Constraints](#) the following option flags may be used to constrain the pairing behavior of single, or pairs of nucleotides:

- [VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#) - Hard constraints flag, base pair in the exterior loop.
- [VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#) - Hard constraints flag, base pair encloses hairpin loop.
- [VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#) - Hard constraints flag, base pair encloses an interior loop.
- [VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC](#) - Hard constraints flag, base pair encloses a multi branch loop.
- [VRNA_CONSTRAINT_CONTEXT_MB_LOOP](#) - Hard constraints flag, base pair is enclosed in an interior loop.
- [VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC](#) - Hard constraints flag, base pair is enclosed in a multi branch loop.
- [VRNA_CONSTRAINT_CONTEXT_ENFORCE](#) - Hard constraint flag to indicate enforcement of constraints.
- [VRNA_CONSTRAINT_CONTEXT_NO_REMOVE](#) - Hard constraint flag to indicate not to remove base pairs that conflict with a given constraint.
- [VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS](#) - Constraint context flag indicating any loop context.

However, for [Soft Constraints](#) we do not allow for simple loop type dependent constraining. But soft constraints are equipped with generic constraint support. This enables the user to pass arbitrary callback functions that return auxiliary energy contributions for evaluation the evaluation of any decomposition.

The callback will then always be notified about the type of decomposition that is happening, and the corresponding delimiting sequence positions. The following decomposition steps are distinguished, and should be captured by the user's implementation of the callback:

- [VRNA_DECOMP_PAIR_HP](#) - Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.
- [VRNA_DECOMP_PAIR_IL](#) - Indicator for interior loop decomposition step.

- [VRNA_DECOMP_PAIR_ML](#) - Indicator for multibranch loop decomposition step.
- [VRNA_DECOMP_ML_ML_ML](#) - Indicator for decomposition of multibranch loop part.
- [VRNA_DECOMP_ML_STEM](#) - Indicator for decomposition of multibranch loop part.
- [VRNA_DECOMP_ML_ML](#) - Indicator for decomposition of multibranch loop part.
- [VRNA_DECOMP_ML_UP](#) - Indicator for decomposition of multibranch loop part.
- [VRNA_DECOMP_ML_ML_STEM](#) - Indicator for decomposition of multibranch loop part.
- [VRNA_DECOMP_ML_COAXIAL](#) - Indicator for decomposition of multibranch loop part.
- [VRNA_DECOMP_EXT_EXT](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_UP](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_STEM](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_EXT_EXT](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_STEM_EXT](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_STEM_OUTSIDE](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_EXT_STEM](#) - Indicator for decomposition of exterior loop part.
- [VRNA_DECOMP_EXT_EXT_STEM1](#) - Indicator for decomposition of exterior loop part.

Simplified interfaces to the soft constraints framework can be obtained by the implementations in the submodules

- [SHAPE Reactivity Data](#) and
- [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#).

An implementation that generates soft constraints for unpaired nucleotides by minimizing the discrepancy between their predicted and expected pairing probability is available in submodule [Generate Soft Constraints from Data](#). Collaboration diagram for Constraining the RNA Folding Grammar:

Modules

- [Hard Constraints](#)
This module covers all functionality for hard constraints in secondary structure prediction.
- [Soft Constraints](#)
Functions and data structures for secondary structure soft constraints.

Files

- file [basic.h](#)
Functions and data structures for constraining secondary structure predictions and evaluation.

Macros

- `#define VRNA_CONSTRAINT_FILE 0`
Flag for [vrna_constraints_add\(\)](#) to indicate that constraints are present in a text file.
- `#define VRNA_CONSTRAINT_SOFT_MFE 0`
Indicate generation of constraints for MFE folding.
- `#define VRNA_CONSTRAINT_SOFT_PF VRNA_OPTION_PF`
Indicate generation of constraints for partition function computation.
- `#define VRNA_DECOMP_PAIR_HP (unsigned char)1`
Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.

- #define `VRNA_DECOMP_PAIR_IL` (unsigned char)2
Indicator for interior loop decomposition step.
- #define `VRNA_DECOMP_PAIR_ML` (unsigned char)3
Indicator for multibranch loop decomposition step.
- #define `VRNA_DECOMP_ML_ML_ML` (unsigned char)5
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_ML_STEM` (unsigned char)6
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_ML_ML` (unsigned char)7
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_ML_UP` (unsigned char)8
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_ML_ML_STEM` (unsigned char)9
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_ML_COAXIAL` (unsigned char)10
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_ML_COAXIAL_ENC` (unsigned char)11
Indicator for decomposition of multibranch loop part.
- #define `VRNA_DECOMP_EXT_EXT` (unsigned char)12
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_UP` (unsigned char)13
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_STEM` (unsigned char)14
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_EXT_EXT` (unsigned char)15
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_STEM_EXT` (unsigned char)16
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_STEM_OUTSIDE` (unsigned char)17
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_EXT_STEM` (unsigned char)18
Indicator for decomposition of exterior loop part.
- #define `VRNA_DECOMP_EXT_EXT_STEM1` (unsigned char)19
Indicator for decomposition of exterior loop part.

Functions

- void `vrna_constraints_add` (`vrna_fold_compound_t` *vc, const char *constraint, unsigned int options)
Add constraints to a `vrna_fold_compound_t` data structure.
- void `vrna_message_constraint_options` (unsigned int option)
Print a help message for pseudo dot-bracket structure constraint characters to stdout. (constraint support is specified by option parameter)
- void `vrna_message_constraint_options_all` (void)
Print structure constraint characters to stdout (full constraint support)

16.11.2 Macro Definition Documentation

16.11.2.1 VRNA_CONSTRAINT_FILE

```
#define VRNA_CONSTRAINT_FILE 0
```

```
#include <ViennaRNA/constraints/basic.h>
```

Flag for `vrna_constraints_add()` to indicate that constraints are present in a text file.

See also

`vrna_constraints_add()`

Deprecated Use 0 instead!

16.11.2.2 VRNA_CONSTRAINT_SOFT_MFE

```
#define VRNA_CONSTRAINT_SOFT_MFE 0
```

```
#include <ViennaRNA/constraints/basic.h>
```

Indicate generation of constraints for MFE folding.

Deprecated This flag has no meaning anymore, since constraints are now always stored!

16.11.2.3 VRNA_CONSTRAINT_SOFT_PF

```
#define VRNA_CONSTRAINT_SOFT_PF VRNA_OPTION_PF
```

```
#include <ViennaRNA/constraints/basic.h>
```

Indicate generation of constraints for partition function computation.

Deprecated Use `VRNA_OPTION_PF` instead!

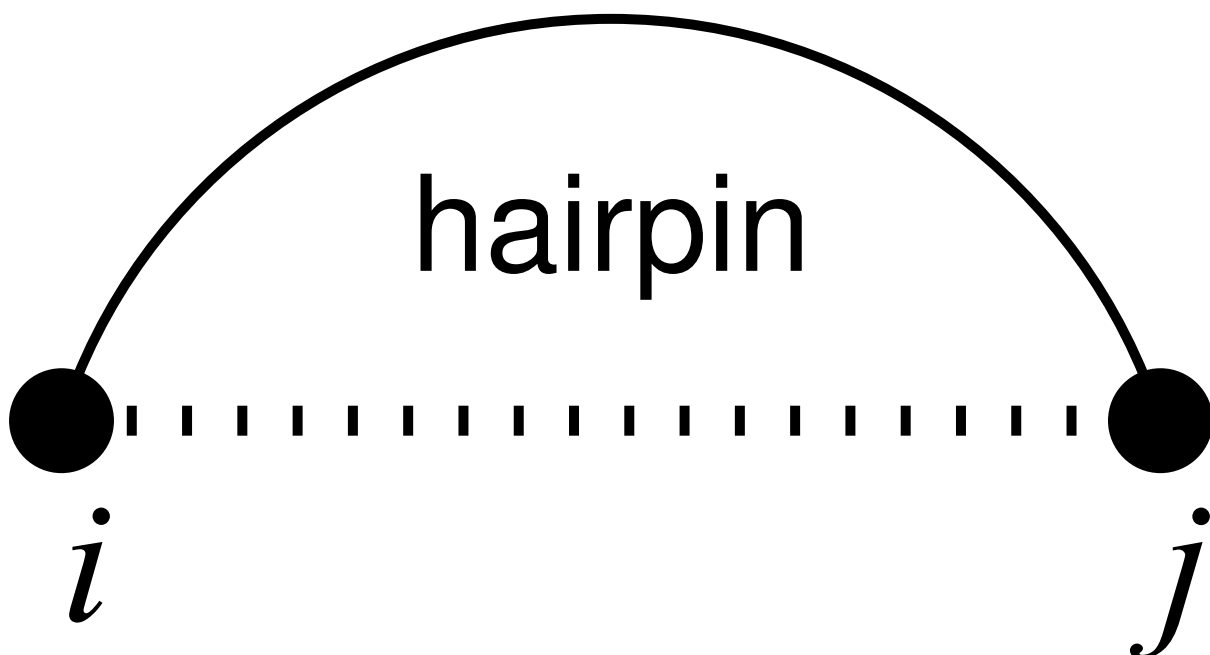
16.11.2.4 VRNA_DECOMP_PAIR_HP

```
#define VRNA_DECOMP_PAIR_HP (unsigned char)1
```

```
#include <ViennaRNA/constraints/basic.h>
```

Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a hairpin loop enclosed by the base pair (i, j) .

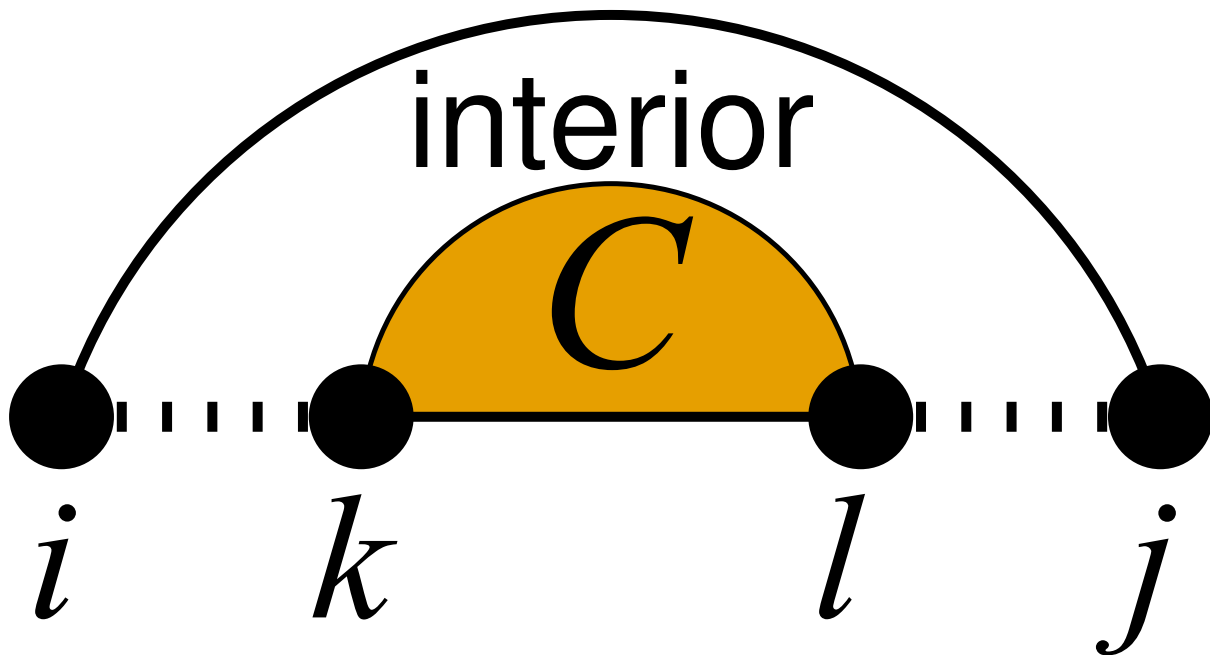


16.11.2.5 VRNA_DECOMP_PAIR_IL

```
#define VRNA_DECOMP_PAIR_IL (unsigned char)2
#include <ViennaRNA/constraints/basic.h>
```

Indicator for interior loop decomposition step.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an interior loop enclosed by the base pair (i, j) , and enclosing the base pair (k, l) .

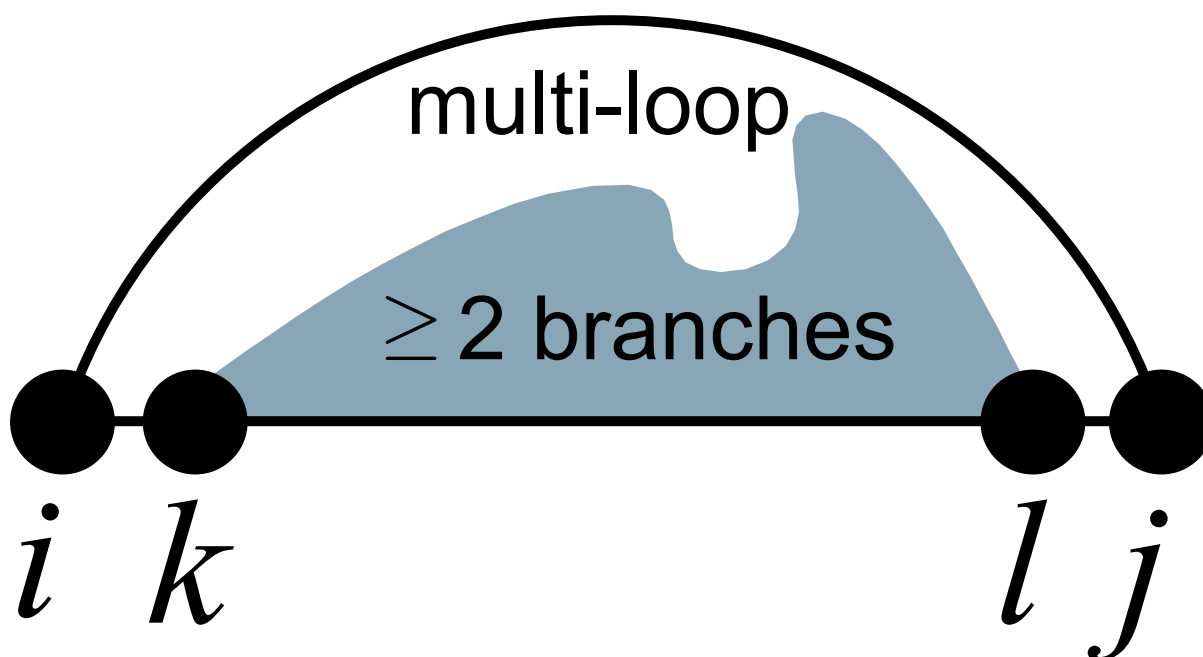


16.11.2.6 VRNA_DECOMP_PAIR_ML

```
#define VRNA_DECOMP_PAIR_ML (unsigned char)3
#include <ViennaRNA/constraints/basic.h>
```

Indicator for multibranch loop decomposition step.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop enclosed by the base pair (i, j) , and consisting of some enclosed multi loop content from k to l .

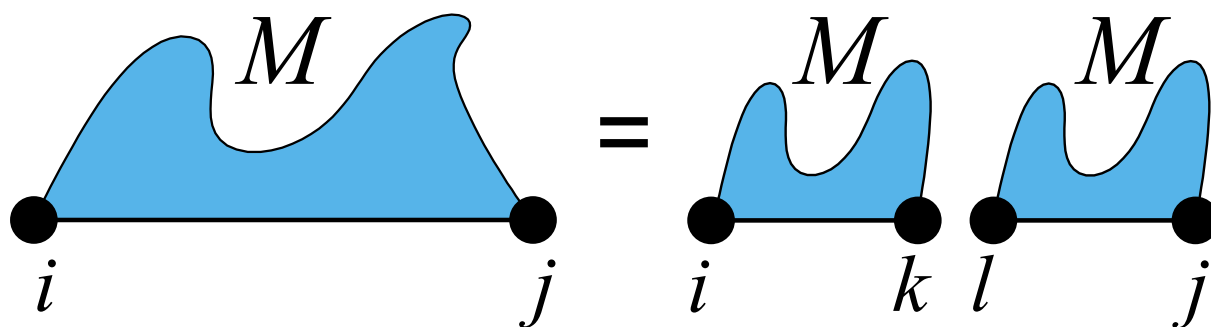


16.11.2.7 VRNA_DECOMP_ML_ML_ML

```
#define VRNA_DECOMP_ML_ML_ML (unsigned char)5
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop part in the interval $[i : j]$, which will be decomposed into two multibranch loop parts $[i : k]$, and $[l : j]$.

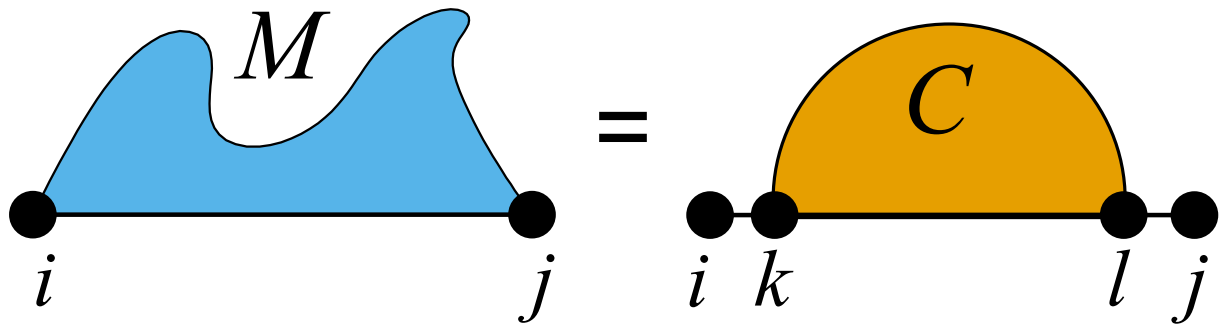


16.11.2.8 VRNA_DECOMP_ML_STEM

```
#define VRNA_DECOMP_ML_STEM (unsigned char)6
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multi-branch loop part in the interval $[i : j]$, which will be considered a single stem branching off with base pair (k, l) .



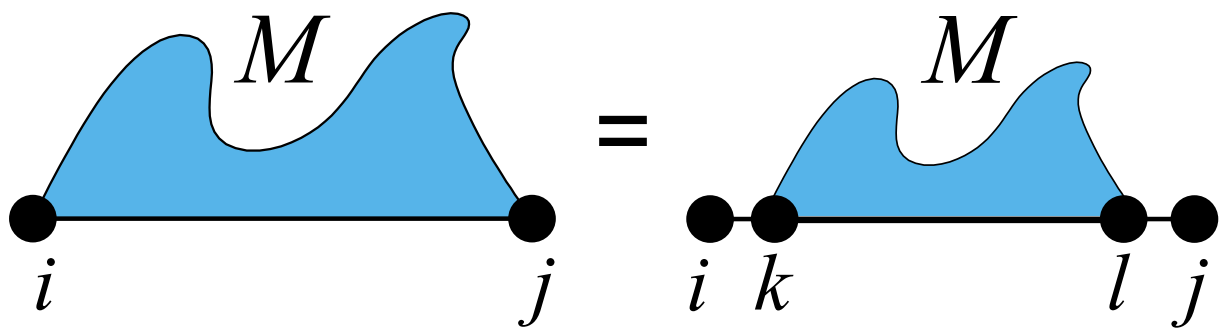
16.11.2.9 VRNA_DECOMP_ML_ML

```
#define VRNA_DECOMP_ML_ML (unsigned char)7
```

```
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval $[i : j]$, which will be decomposed into a (usually) smaller multibranch loop part $[k : l]$.



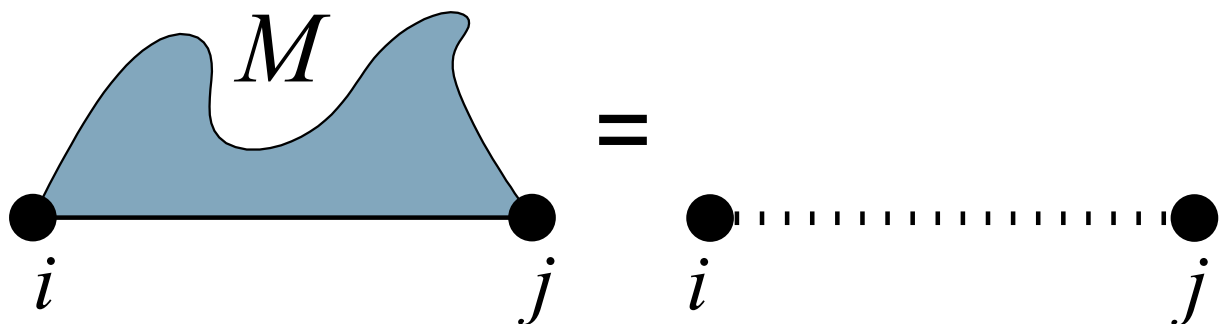
16.11.2.10 VRNA_DECOMP_ML_UP

```
#define VRNA_DECOMP_ML_UP (unsigned char)8
```

```
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval $[i : j]$, which will be considered a multibranch loop part that only consists of unpaired nucleotides.

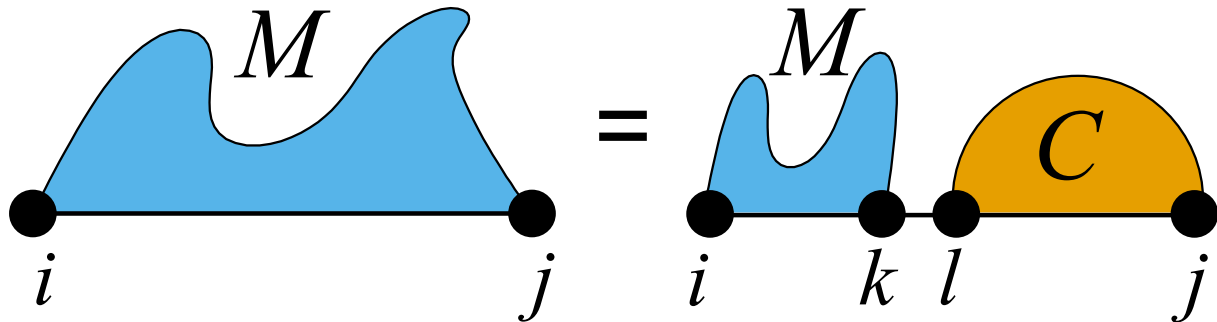


16.11.2.11 VRNA_DECOMP_ML_ML_STEM

```
#define VRNA_DECOMP_ML_ML_STEM (unsigned char)9
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval $[i : j]$, which will be decomposed into a multibranch loop part $[i : k]$, and a stem with enclosing base pair (l, j) .

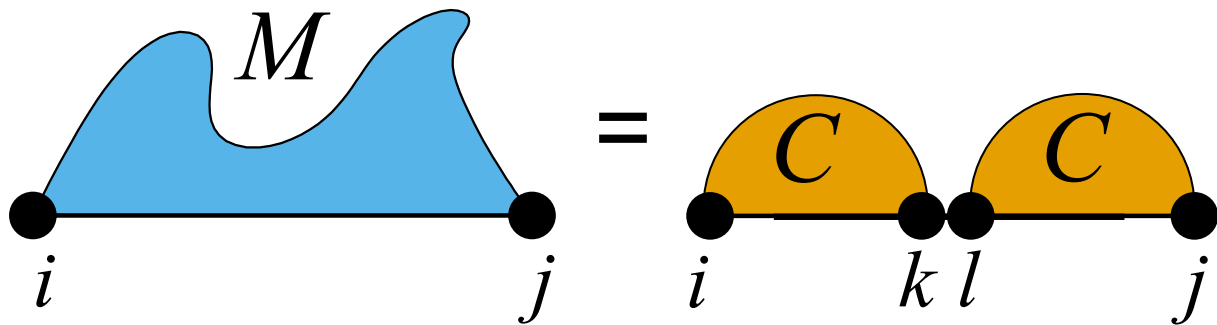


16.11.2.12 VRNA_DECOMP_ML_COAXIAL

```
#define VRNA_DECOMP_ML_COAXIAL (unsigned char)10
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval $[i : j]$, where two stems with enclosing pairs (i, k) and (l, j) are coaxially stacking onto each other.

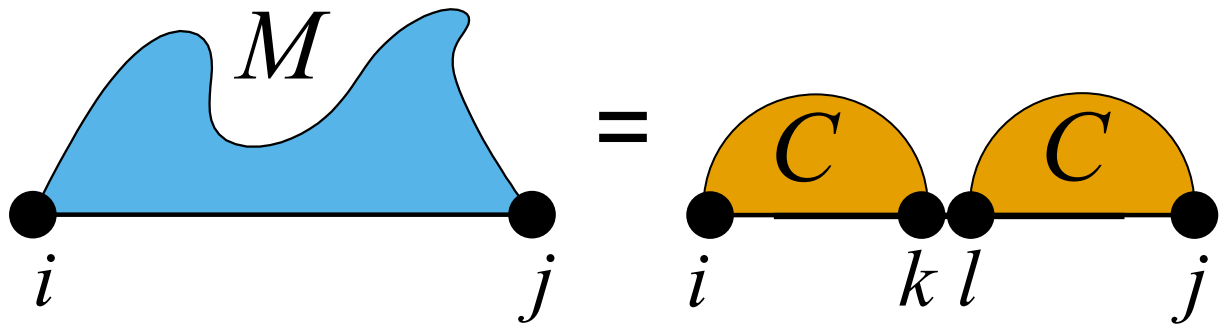


16.11.2.13 VRNA_DECOMP_ML_COAXIAL_ENC

```
#define VRNA_DECOMP_ML_COAXIAL_ENC (unsigned char)11
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of multibranch loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates a multibranch loop part in the interval $[i : j]$, where two stems with enclosing pairs (i, k) and (l, j) are coaxially stacking onto each other.

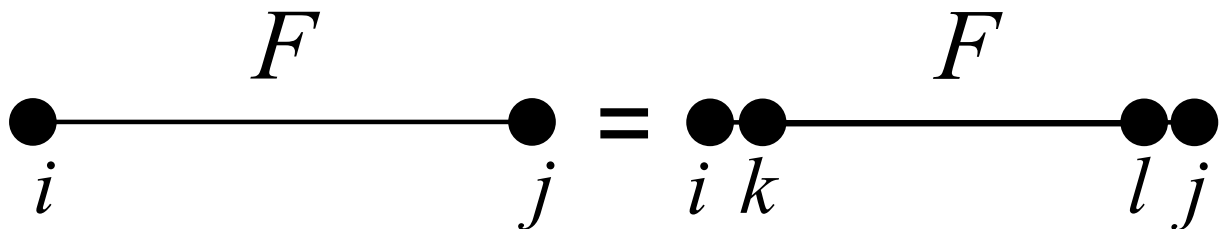


16.11.2.14 VRNA_DECOMP_EXT_EXT

```
#define VRNA_DECOMP_EXT_EXT (unsigned char)12
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be decomposed into a (usually) smaller exterior loop part $[k : l]$.

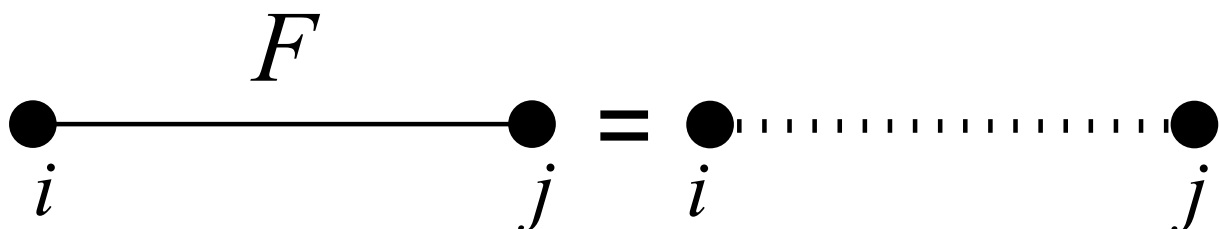


16.11.2.15 VRNA_DECOMP_EXT_UP

```
#define VRNA_DECOMP_EXT_UP (unsigned char)13
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be considered as an exterior loop component consisting of only unpaired nucleotides.

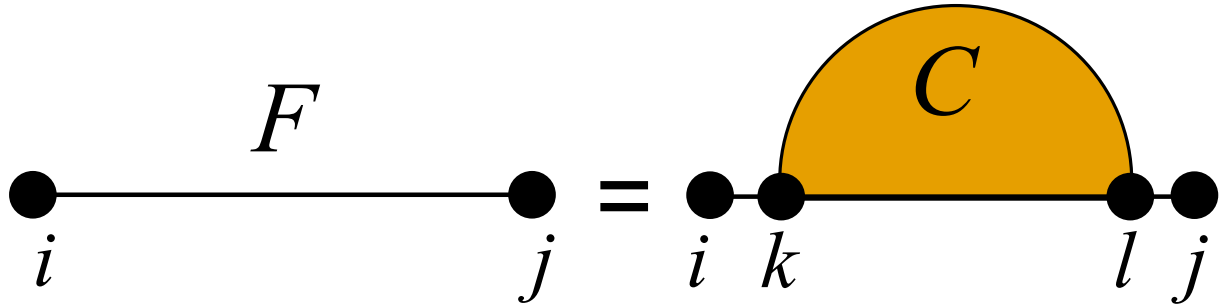


16.11.2.16 VRNA_DECOMP_EXT_STEM

```
#define VRNA_DECOMP_EXT_STEM (unsigned char)14
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be considered a stem with enclosing pair (k, l) .

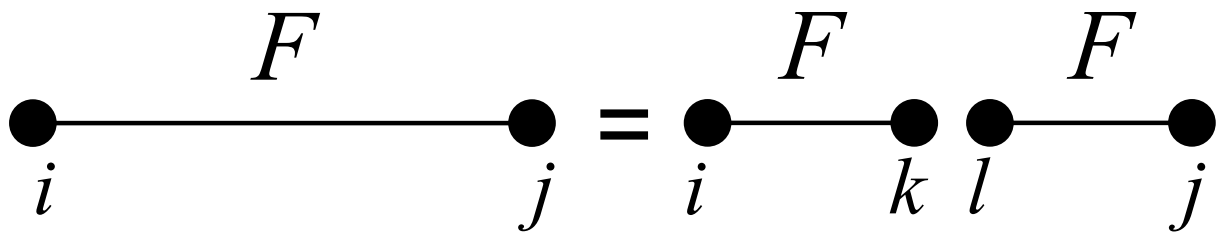


16.11.2.17 VRNA_DECOMP_EXT_EXT_EXT

```
#define VRNA_DECOMP_EXT_EXT_EXT (unsigned char)15
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be decomposed into two exterior loop parts $[i : k]$ and $[l : j]$.

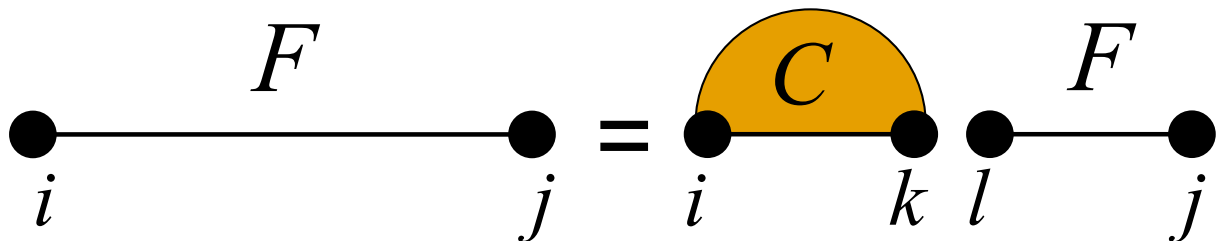


16.11.2.18 VRNA_DECOMP_EXT_STEM_EXT

```
#define VRNA_DECOMP_EXT_STEM_EXT (unsigned char)16
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be decomposed into a stem branching off with base pair (i, k) , and an exterior loop part $[l : j]$.



16.11.2.19 VRNA_DECOMP_EXT_STEM_OUTSIDE

```
#define VRNA_DECOMP_EXT_STEM_OUTSIDE (unsigned char)17
#include <ViennaRNA/constraints/basic.h>
```

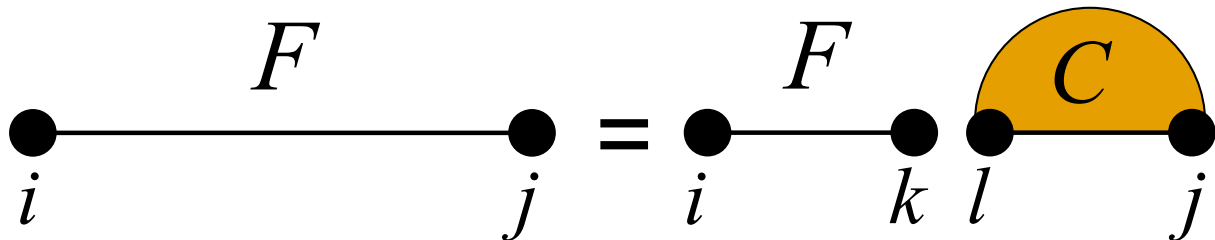
Indicator for decomposition of exterior loop part.

16.11.2.20 VRNA_DECOMP_EXT_EXT_STEM

```
#define VRNA_DECOMP_EXT_EXT_STEM (unsigned char)18
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be decomposed into an exterior loop part $[i : k]$, and a stem branching off with base pair (l, j) .

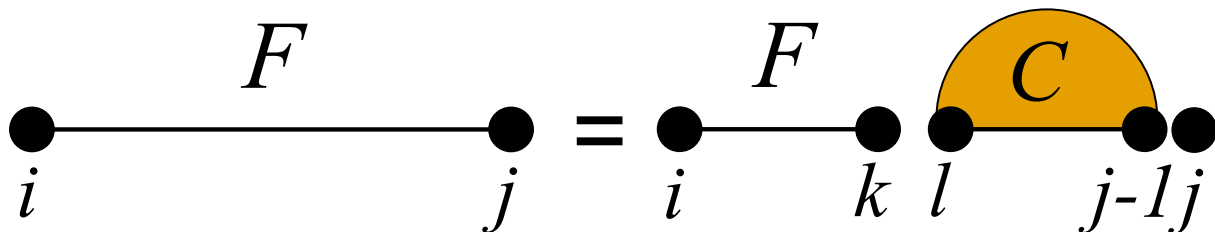


16.11.2.21 VRNA_DECOMP_EXT_EXT_STEM1

```
#define VRNA_DECOMP_EXT_EXT_STEM1 (unsigned char)19
#include <ViennaRNA/constraints/basic.h>
```

Indicator for decomposition of exterior loop part.

This flag notifies the soft or hard constraint callback function that the current decomposition step evaluates an exterior loop part in the interval $[i : j]$, which will be decomposed into an exterior loop part $[i : k]$, and a stem branching off with base pair $(l, j - 1)$.



16.11.3 Function Documentation

16.11.3.1 vrna_constraints_add()

```
void vrna_constraints_add (
    vrna_fold_compound_t * vc,
    const char * constraint,
    unsigned int options )
#include <ViennaRNA/constraints/basic.h>
```

Add constraints to a `vrna_fold_compound_t` data structure.

Use this function to add/update the hard/soft constraints. The function allows for passing a string 'constraint' that can either be a filename that points to a constraints definition file or it may be a pseudo dot-bracket notation indicating hard constraints. For the latter, the user has to pass the `VRNA_CONSTRAINT_DB` option. Also, the user has to specify, which characters are allowed to be interpreted as constraints by passing the corresponding options via the third parameter.

See also

[vrna_hc_init\(\)](#), [vrna_hc_add_up\(\)](#), [vrna_hc_add_up_batch\(\)](#), [vrna_hc_add_bp\(\)](#), [vrna_sc_init\(\)](#), [vrna_sc_set_up\(\)](#), [vrna_sc_set_bp\(\)](#), [vrna_sc_add_SHAPE_deigan\(\)](#), [vrna_sc_add_SHAPE_zarringhalam\(\)](#), [vrna_hc_free\(\)](#), [vrna_sc_free\(\)](#), [VRNA_CONSTRAINT_DB](#), [VRNA_CONSTRAINT_DB_DEFAULT](#), [VRNA_CONSTRAINT_DB_PIPE](#), [VRNA_CONSTRAINT_DB_DOT](#), [VRNA_CONSTRAINT_DB_X](#), [VRNA_CONSTRAINT_DB_ANG_BRACK](#), [VRNA_CONSTRAINT_DB_RND_BRACK](#), [VRNA_CONSTRAINT_DB_INTRAMOL](#), [VRNA_CONSTRAINT_DB_INTERMOL](#), [VRNA_CONSTRAINT_DB_GQUAD](#)

The following is an example for adding hard constraints given in pseudo dot-bracket notation. Here, `vc` is the [vrna_fold_compound_t](#) object, `structure` is a char array with the hard constraint in dot-bracket notation, and `enforceConstraints` is a flag indicating whether or not constraints for base pairs should be enforced instead of just doing a removal of base pair that conflict with the constraint.

```
unsigned int constraint_options = VRNA_CONSTRAINT_DB_DEFAULT;

if (enforceConstraints)
    constraint_options |= VRNA_CONSTRAINT_DB_ENFORCE_BP;

if (canonicalBPonly)
    constraint_options |= VRNA_CONSTRAINT_DB_CANONICAL_BP;

vrna_constraints_add(fc, (const char *)cstruc, constraint_options);
```

In constrast to the above, constraints may also be read from file:

```
vrna_constraints_add(fc, constraints_file, VRNA_OPTION_DEFAULT);
```

See also

[vrna_hc_add_from_db\(\)](#), [vrna_hc_add_up\(\)](#), [vrna_hc_add_up_batch\(\)](#) [vrna_hc_add_bp_unspecific\(\)](#), [vrna_hc_add_bp\(\)](#)

Parameters

<i>vc</i>	The fold compound
<i>constraint</i>	A string with either the filename of the constraint definitions or a pseudo dot-bracket notation of the hard constraint. May be NULL.
<i>options</i>	The option flags

16.11.3.2 [vrna_message_constraint_options\(\)](#)

```
void vrna_message_constraint_options (
    unsigned int option )
#include <ViennaRNA/constraints/hard.h>
```

Print a help message for pseudo dot-bracket structure constraint characters to stdout. (constraint support is specified by option parameter)

Currently available options are:

[VRNA_CONSTRAINT_DB_PIPE](#) (paired with another base)
[VRNA_CONSTRAINT_DB_DOT](#) (no constraint at all)
[VRNA_CONSTRAINT_DB_X](#) (base must not pair)
[VRNA_CONSTRAINT_DB_ANG_BRACK](#) (paired downstream/upstream)
[VRNA_CONSTRAINT_DB_RND_BRACK](#) (base i pairs base j)

pass a collection of options as one value like this:

```
vrna_message_constraints(option_1 | option_2 | option_n)
```

See also

[vrna_message_constraint_options_all\(\)](#), [vrna_constraints_add\(\)](#), [VRNA_CONSTRAINT_DB](#), [VRNA_CONSTRAINT_DB_PIPE](#), [VRNA_CONSTRAINT_DB_DOT](#), [VRNA_CONSTRAINT_DB_X](#), [VRNA_CONSTRAINT_DB_ANG_BRACK](#), [VRNA_CONSTRAINT_DB_RND_BRACK](#), [VRNA_CONSTRAINT_DB_INTERMOL](#), [VRNA_CONSTRAINT_DB_INTRAMOL](#)

Parameters

<i>option</i>	Option switch that tells which constraint help will be printed
---------------	--

16.11.3.3 vrna_message_constraint_options_all()

```
void vrna_message_constraint_options_all (
    void )
#include <ViennaRNA/constraints/hard.h>
Print structure constraint characters to stdout (full constraint support)
```

See also

[vrna_message_constraint_options\(\)](#), [vrna_constraints_add\(\)](#), [VRNA_CONSTRAINT_DB](#), [VRNA_CONSTRAINT_DB_PIPE](#), [VRNA_CONSTRAINT_DB_DOT](#), [VRNA_CONSTRAINT_DB_X](#), [VRNA_CONSTRAINT_DB_ANG_BRACK](#), [VRNA_CONSTRAINT_DB_RND_BRACK](#), [VRNA_CONSTRAINT_DB_INTERMOL](#), [VRNA_CONSTRAINT_DB_INTRAMOL](#)

16.12 Hard Constraints

This module covers all functionality for hard constraints in secondary structure prediction.

16.12.1 Detailed Description

This module covers all functionality for hard constraints in secondary structure prediction.
Collaboration diagram for Hard Constraints:

Files

- file [hard.h](#)

Functions and data structures for handling of secondary structure hard constraints.

Data Structures

- struct [vrna_hc_s](#)
The hard constraints data structure. [More...](#)
- struct [vrna_hc_up_s](#)
A single hard constraint for a single nucleotide. [More...](#)

Macros

- #define [VRNA_CONSTRAINT_DB](#) 16384U
Flag for [vrna_constraints_add\(\)](#) to indicate that constraint is passed in pseudo dot-bracket notation.
- #define [VRNA_CONSTRAINT_DB_ENFORCE_BP](#) 32768U
Switch for dot-bracket structure constraint to enforce base pairs.
- #define [VRNA_CONSTRAINT_DB_PIPE](#) 65536U
Flag that is used to indicate the pipe '|' sign in pseudo dot-bracket notation of hard constraints.
- #define [VRNA_CONSTRAINT_DB_DOT](#) 131072U
dot '.' switch for structure constraints (no constraint at all)
- #define [VRNA_CONSTRAINT_DB_X](#) 262144U
'x' switch for structure constraint (base must not pair)
- #define [VRNA_CONSTRAINT_DB_RND_BRACK](#) 1048576U
round brackets '(',')' switch for structure constraint (base i pairs base j)
- #define [VRNA_CONSTRAINT_DB_INTRAMOL](#) 2097152U

- Flag that is used to indicate the character 'l' in pseudo dot-bracket notation of hard constraints.
- #define `VRNA_CONSTRAINT_DB_INTERMOL` 4194304U
Flag that is used to indicate the character 'e' in pseudo dot-bracket notation of hard constraints.
- #define `VRNA_CONSTRAINT_DB_GQUAD` 8388608U
'+' switch for structure constraint (base is involved in a gquad)
- #define `VRNA_CONSTRAINT_DB_WUSS` 33554432U
Flag to indicate Washington University Secondary Structure (WUSS) notation of the hard constraint string.
- #define `VRNA_CONSTRAINT_DB_DEFAULT`
Switch for dot-bracket structure constraint with default symbols.
- #define `VRNA_CONSTRAINT_CONTEXT_EXT_LOOP` (unsigned char)0x01
Hard constraints flag, base pair in the exterior loop.
- #define `VRNA_CONSTRAINT_CONTEXT_HP_LOOP` (unsigned char)0x02
Hard constraints flag, base pair encloses hairpin loop.
- #define `VRNA_CONSTRAINT_CONTEXT_INT_LOOP` (unsigned char)0x04
Hard constraints flag, base pair encloses an interior loop.
- #define `VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC` (unsigned char)0x08
Hard constraints flag, base pair encloses a multi branch loop.
- #define `VRNA_CONSTRAINT_CONTEXT_MB_LOOP` (unsigned char)0x10
Hard constraints flag, base pair is enclosed in an interior loop.
- #define `VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC` (unsigned char)0x20
Hard constraints flag, base pair is enclosed in a multi branch loop.
- #define `VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS`
Constraint context flag indicating any loop context.

Typedefs

- typedef struct `vrna_hc_s` `vrna_hc_t`
Typename for the hard constraints data structure `vrna_hc_s`.
- typedef struct `vrna_hc_up_s` `vrna_hc_up_t`
Typename for the single nucleotide hard constraint data structure `vrna_hc_up_s`.
- typedef unsigned char(* `vrna_hc_eval_f`) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to evaluate whether or not a particular decomposition step is contributing to the solution space.

Functions

- void `vrna_hc_init` (`vrna_fold_compound_t` *vc)
Initialize/Reset hard constraints to default values.
- void `vrna_hc_add_up` (`vrna_fold_compound_t` *vc, int i, unsigned char option)
Make a certain nucleotide unpaired.
- int `vrna_hc_add_up_batch` (`vrna_fold_compound_t` *vc, `vrna_hc_up_t` *constraints)
Apply a list of hard constraints for single nucleotides.
- int `vrna_hc_add_bp` (`vrna_fold_compound_t` *vc, int i, int j, unsigned char option)
Favorize/Enforce a certain base pair (i,j)
- void `vrna_hc_add_bp_nonspecific` (`vrna_fold_compound_t` *vc, int i, int d, unsigned char option)
Enforce a nucleotide to be paired (upstream/downstream)
- void `vrna_hc_free` (`vrna_hc_t` *hc)
Free the memory allocated by a `vrna_hc_t` data structure.
- int `vrna_hc_add_from_db` (`vrna_fold_compound_t` *vc, const char *constraint, unsigned int options)
Add hard constraints from pseudo dot-bracket notation.

16.12.2 Data Structure Documentation

16.12.2.1 struct vrna_hc_s

The hard constraints data structure.

The content of this data structure determines the decomposition pattern used in the folding recursions. Attribute 'matrix' is used as source for the branching pattern of the decompositions during all folding recursions. Any entry in matrix[i,j] consists of the 6 LSB that allows one to distinguish the following types of base pairs:

- in the exterior loop ([VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#))
- enclosing a hairpin ([VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#))
- enclosing an interior loop ([VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#))
- enclosed by an exterior loop ([VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC](#))
- enclosing a multi branch loop ([VRNA_CONSTRAINT_CONTEXT_MB_LOOP](#))
- enclosed by a multi branch loop ([VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC](#))

The four linear arrays 'up_xxx' provide the number of available unpaired nucleotides (including position i) 3' of each position in the sequence.

See also

[vrna_hc_init\(\)](#), [vrna_hc_free\(\)](#), [VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_MB_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC](#)

Data Fields

- int * **up_ext**
A linear array that holds the number of allowed unpaired nucleotides in an exterior loop.
- int * **up_hp**
A linear array that holds the number of allowed unpaired nucleotides in a hairpin loop.
- int * **up_int**
A linear array that holds the number of allowed unpaired nucleotides in an interior loop.
- int * **up_ml**
A linear array that holds the number of allowed unpaired nucleotides in a multi branched loop.
- [vrna_hc_eval_f](#) f
A function pointer that returns whether or not a certain decomposition may be evaluated.
- void * **data**
A pointer to some structure where the user may store necessary data to evaluate its generic hard constraint function.
- [vrna_auxdata_free_f](#) free_data
A pointer to a function to free memory occupied by auxiliary data.

16.12.2.1.1 Field Documentation

16.12.2.1.1.1 free_data [vrna_auxdata_free_f](#) vrna_hc_s::free_data

A pointer to a function to free memory occupied by auxiliary data.

The function this pointer is pointing to will be called upon destruction of the [vrna_hc_s](#), and provided with the [vrna_hc_s.data](#) pointer that may hold auxiliary data. Hence, to avoid leaking memory, the user may use this pointer to free memory occupied by auxiliary data.

16.12.2.2 struct vrna_hc_up_s

A single hard constraint for a single nucleotide.

Data Fields

- int **position**
The sequence position (1-based)
- unsigned char **options**
The hard constraint option

16.12.3 Macro Definition Documentation

16.12.3.1 VRNA_CONSTRAINT_DB

```
#define VRNA_CONSTRAINT_DB 16384U
```

```
#include <ViennaRNA/constraints/hard.h>
```

Flag for `vrna_constraints_add()` to indicate that constraint is passed in pseudo dot-bracket notation.

See also

[vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.2 VRNA_CONSTRAINT_DB_ENFORCE_BP

```
#define VRNA_CONSTRAINT_DB_ENFORCE_BP 32768U
```

```
#include <ViennaRNA/constraints/hard.h>
```

Switch for dot-bracket structure constraint to enforce base pairs.

This flag should be used to really enforce base pairs given in dot-bracket constraint rather than just weakly-enforcing them.

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.3 VRNA_CONSTRAINT_DB_PIPE

```
#define VRNA_CONSTRAINT_DB_PIPE 65536U
```

```
#include <ViennaRNA/constraints/hard.h>
```

Flag that is used to indicate the pipe '|' sign in pseudo dot-bracket notation of hard constraints.

Use this definition to indicate the pipe sign '|' (paired with another base)

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.4 VRNA_CONSTRAINT_DB_DOT

```
#define VRNA_CONSTRAINT_DB_DOT 131072U
```

```
#include <ViennaRNA/constraints/hard.h>
```

dot '.' switch for structure constraints (no constraint at all)

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.5 VRNA_CONSTRAINT_DB_X

```
#define VRNA_CONSTRAINT_DB_X 262144U
#include <ViennaRNA/constraints/hard.h>
'x' switch for structure constraint (base must not pair)
```

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.6 VRNA_CONSTRAINT_DB_RND_BRACK

```
#define VRNA_CONSTRAINT_DB_RND_BRACK 1048576U
#include <ViennaRNA/constraints/hard.h>
round brackets '(',')' switch for structure constraint (base i pairs base j)
```

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.7 VRNA_CONSTRAINT_DB_INTRAMOL

```
#define VRNA_CONSTRAINT_DB_INTRAMOL 2097152U
#include <ViennaRNA/constraints/hard.h>
Flag that is used to indicate the character 'I' in pseudo dot-bracket notation of hard constraints.
Use this definition to indicate the usage of 'I' character (intramolecular pairs only)
```

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.8 VRNA_CONSTRAINT_DB_INTERMOL

```
#define VRNA_CONSTRAINT_DB_INTERMOL 4194304U
#include <ViennaRNA/constraints/hard.h>
Flag that is used to indicate the character 'e' in pseudo dot-bracket notation of hard constraints.
Use this definition to indicate the usage of 'e' character (intermolecular pairs only)
```

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.9 VRNA_CONSTRAINT_DB_GQUAD

```
#define VRNA_CONSTRAINT_DB_GQUAD 8388608U
#include <ViennaRNA/constraints/hard.h>
'+' switch for structure constraint (base is involved in a quad)
```

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

Warning

This flag is for future purposes only! No implementation recognizes it yet.

16.12.3.10 VRNA_CONSTRAINT_DB_WUSS

```
#define VRNA_CONSTRAINT_DB_WUSS 33554432U
```

```
#include <ViennaRNA/constraints/hard.h>
```

Flag to indicate Washington University Secondary Structure (WUSS) notation of the hard constraint string.

This secondary structure notation for RNAs is usually used as consensus secondary structure (SS_cons) entry in Stockholm formatted files

16.12.3.11 VRNA_CONSTRAINT_DB_DEFAULT

```
#define VRNA_CONSTRAINT_DB_DEFAULT
```

```
#include <ViennaRNA/constraints/hard.h>
```

Value:

```
(VRNA_CONSTRAINT_DB \
| VRNA_CONSTRAINT_DB_PIPE \
| VRNA_CONSTRAINT_DB_DOT \
| VRNA_CONSTRAINT_DB_X \
| VRNA_CONSTRAINT_DB_ANG_BRACK \
| VRNA_CONSTRAINT_DB_RND_BRACK \
| VRNA_CONSTRAINT_DB_INTRAMOL \
| VRNA_CONSTRAINT_DB_INTERMOL \
| VRNA_CONSTRAINT_DB_GQUAD \
)
```

Switch for dot-bracket structure constraint with default symbols.

This flag conveniently combines all possible symbols in dot-bracket notation for hard constraints and

[VRNA_CONSTRAINT_DB](#)

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

16.12.3.12 VRNA_CONSTRAINT_CONTEXT_EXT_LOOP

```
#define VRNA_CONSTRAINT_CONTEXT_EXT_LOOP (unsigned char)0x01
```

```
#include <ViennaRNA/constraints/hard.h>
```

Hard constraints flag, base pair in the exterior loop.

16.12.3.13 VRNA_CONSTRAINT_CONTEXT_HP_LOOP

```
#define VRNA_CONSTRAINT_CONTEXT_HP_LOOP (unsigned char)0x02
```

```
#include <ViennaRNA/constraints/hard.h>
```

Hard constraints flag, base pair encloses hairpin loop.

16.12.3.14 VRNA_CONSTRAINT_CONTEXT_INT_LOOP

```
#define VRNA_CONSTRAINT_CONTEXT_INT_LOOP (unsigned char)0x04
```

```
#include <ViennaRNA/constraints/hard.h>
```

Hard constraints flag, base pair encloses an interior loop.

16.12.3.15 VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC

```
#define VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC (unsigned char)0x08
```

```
#include <ViennaRNA/constraints/hard.h>
```

Hard constraints flag, base pair encloses a multi branch loop.

16.12.3.16 VRNA_CONSTRAINT_CONTEXT_MB_LOOP

```
#define VRNA_CONSTRAINT_CONTEXT_MB_LOOP (unsigned char)0x10
```

```
#include <ViennaRNA/constraints/hard.h>
```

Hard constraints flag, base pair is enclosed in an interior loop.

16.12.3.17 VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC

```
#define VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC (unsigned char)0x20
```

```
#include <ViennaRNA/constraints/hard.h>
```

Hard constraints flag, base pair is enclosed in a multi branch loop.

16.12.3.18 VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS

```
#define VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS
```

```
#include <ViennaRNA/constraints/hard.h>
```

Value:

```
char) (VRNA_CONSTRAINT_CONTEXT_CLOSING_LOOPS | \
                                             (unsigned
                                             VRNA_CONSTRAINT_CONTEXT_ENCLOSED_LOOPS)
```

Constraint context flag indicating any loop context.

16.12.4 Typedef Documentation

16.12.4.1 vrna_hc_eval_f

```
typedef unsigned char(* vrna_hc_eval_f) (int i, int j, int k, int l, unsigned char d, void
*data)
```

```
#include <ViennaRNA/constraints/hard.h>
```

Callback to evaluate whether or not a particular decomposition step is contributing to the solution space.

This is the prototype for callback functions used by the folding recursions to evaluate generic hard constraints. The first four parameters passed indicate the delimiting nucleotide positions of the decomposition, and the parameter denotes the decomposition step. The last parameter *data* is the auxiliary data structure associated to the hard constraints via [vrna_hc_add_data\(\)](#), or NULL if no auxiliary data was added.

Notes on Callback Functions This callback enables one to over-rule default hard constraints in secondary structure decompositions.

See also

[VRNA_DECOMP_PAIR_HP](#), [VRNA_DECOMP_PAIR_IL](#), [VRNA_DECOMP_PAIR_ML](#), [VRNA_DECOMP_ML_ML_ML](#), [VRNA_DECOMP_ML_STEM](#), [VRNA_DECOMP_ML_ML](#), [VRNA_DECOMP_ML_UP](#), [VRNA_DECOMP_ML_ML_STEM](#), [VRNA_DECOMP_ML_COAXIAL](#), [VRNA_DECOMP_EXT_EXT](#), [VRNA_DECOMP_EXT_UP](#), [VRNA_DECOMP_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_EXT](#), [VRNA_DECOMP_EXT_STEM_EXT](#), [VRNA_DECOMP_EXT_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_STEM1](#), [vrna_hc_add_f\(\)](#), [vrna_hc_add_data\(\)](#)

Parameters

<i>i</i>	Left (5') delimiter position of substructure
<i>j</i>	Right (3') delimiter position of substructure
<i>k</i>	Left delimiter of decomposition
<i>l</i>	Right delimiter of decomposition
<i>d</i>	Decomposition step indicator
<i>data</i>	Auxiliary data

Returns

A non-zero value if the decomposition is valid, 0 otherwise

16.12.5 Function Documentation**16.12.5.1 vrna_hc_init()**

```
void vrna_hc_init (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/constraints/hard.h>
```

Initialize/Reset hard constraints to default values.

This function resets the hard constraints to their default values, i.e. all positions may be unpaired in all contexts, and base pairs are allowed in all contexts, if they resemble canonical pairs. Previously set hard constraints will be removed before initialization.

See also

[vrna_hc_add_bp\(\)](#), [vrna_hc_add_bp_nonspecific\(\)](#), [vrna_hc_add_up\(\)](#)

Parameters

<i>vc</i>	The fold compound
-----------	-------------------

SWIG Wrapper Notes This function is attached as method **hc_init()** to objects of type *fold_compound*

16.12.5.2 vrna_hc_add_up()

```
void vrna_hc_add_up (
    vrna_fold_compound_t * vc,
    int i,
    unsigned char option )
#include <ViennaRNA/constraints/hard.h>
```

Make a certain nucleotide unpaired.

See also

[vrna_hc_add_bp\(\)](#), [vrna_hc_add_bp_nonspecific\(\)](#), [vrna_hc_init\(\)](#), [VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the hard constraints are associated with
<i>i</i>	The position that needs to stay unpaired (1-based)
<i>option</i>	The options flag indicating how/where to store the hard constraints

16.12.5.3 vrna_hc_add_up_batch()

```
int vrna_hc_add_up_batch (
    vrna_fold_compound_t * vc,
    vrna_hc_up_t * constraints )
```



```
#include <ViennaRNA/constraints/hard.h>
```

Apply a list of hard constraints for single nucleotides.

Parameters

<i>vc</i>	The vrna_fold_compound_t the hard constraints are associated with
<i>constraints</i>	The list off constraints to apply, last entry must have position attribute set to 0

16.12.5.4 vrna_hc_add_bp()

```
int vrna_hc_add_bp (
    vrna_fold_compound_t * vc,
    int i,
    int j,
    unsigned char option )
#include <ViennaRNA/constraints/hard.h>
```

Favorize/Enforce a certain base pair (i,j)

See also

[vrna_hc_add_bp_nonspecific\(\)](#), [vrna_hc_add_up\(\)](#), [vrna_hc_init\(\)](#), [VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_MB_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC](#), [VRNA_CONSTRAINT_CONTEXT_ENFORCE](#), [VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the hard constraints are associated with
<i>i</i>	The 5' located nucleotide position of the base pair (1-based)
<i>j</i>	The 3' located nucleotide position of the base pair (1-based)
<i>option</i>	The options flag indicating how/where to store the hard constraints

16.12.5.5 vrna_hc_add_bp_nonspecific()

```
void vrna_hc_add_bp_nonspecific (
    vrna_fold_compound_t * vc,
    int i,
    int d,
    unsigned char option )
#include <ViennaRNA/constraints/hard.h>
```

Enforce a nucleotide to be paired (upstream/downstream)

See also

[vrna_hc_add_bp\(\)](#), [vrna_hc_add_up\(\)](#), [vrna_hc_init\(\)](#), [VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_MB_LOOP](#), [VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC](#), [VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the hard constraints are associated with
<i>i</i>	The position that needs to stay unpaired (1-based)

Parameters

<i>d</i>	The direction of base pairing ($d < 0$: pairs upstream, $d > 0$: pairs downstream, $d == 0$: no direction)
<i>option</i>	The options flag indicating in which loop type context the pairs may appear

16.12.5.6 vrna_hc_free()

```
void vrna_hc_free (
    vrna_hc_t * hc )
#include <ViennaRNA/constraints/hard.h>
Free the memory allocated by a vrna\_hc\_t data structure.
Use this function to free all memory that was allocated for a data structure of type vrna\_hc\_t.
```

See also

[get_hard_constraints\(\)](#), [vrna_hc_t](#)

16.12.5.7 vrna_hc_add_from_db()

```
int vrna_hc_add_from_db (
    vrna_fold_compound_t * vc,
    const char * constraint,
    unsigned int options )
#include <ViennaRNA/constraints/hard.h>
Add hard constraints from pseudo dot-bracket notation.
This function allows one to apply hard constraints from a pseudo dot-bracket notation. The options parameter controls, which characters are recognized by the parser. Use the VRNA\_CONSTRAINT\_DB\_DEFAULT convenience macro, if you want to allow all known characters
```

See also

[VRNA_CONSTRAINT_DB_PIPE](#), [VRNA_CONSTRAINT_DB_DOT](#), [VRNA_CONSTRAINT_DB_X](#), [VRNA_CONSTRAINT_DB_](#)
[VRNA_CONSTRAINT_DB_RND_BRACK](#), [VRNA_CONSTRAINT_DB_INTRAMOL](#), [VRNA_CONSTRAINT_DB_INTERMOL](#),
[VRNA_CONSTRAINT_DB_GQUAD](#)

Parameters

<i>vc</i>	The fold compound
<i>constraint</i>	A pseudo dot-bracket notation of the hard constraint.
<i>options</i>	The option flags

SWIG Wrapper Notes This function is attached as method [hc_add_from_db\(\)](#) to objects of type *fold_compound*

16.13 Soft Constraints

Functions and data structures for secondary structure soft constraints.

16.13.1 Detailed Description

Functions and data structures for secondary structure soft constraints.
Soft-constraints are used to change position specific contributions in the recursions by adding bonuses/penalties in form of pseudo free energies to certain loop configurations. Collaboration diagram for Soft Constraints:

Files

- file [soft.h](#)
Functions and data structures for secondary structure soft constraints.
- file [soft_special.h](#)
Specialized implementations that utilize the soft constraint callback mechanism.

Data Structures

- struct [vrna_sc_s](#)
The soft constraints data structure. [More...](#)

Typedefs

- typedef struct [vrna_sc_s](#) [vrna_sc_t](#)
Typename for the soft constraints data structure [vrna_sc_s](#).
- typedef int(* [vrna_sc_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to retrieve pseudo energy contribution for soft constraint feature.
- typedef [FLT_OR_DBL](#)(* [vrna_sc_exp_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to retrieve pseudo energy contribution as Boltzmann Factors for soft constraint feature.
- typedef [vrna_basepair_t](#) *(* [vrna_sc_bt_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to retrieve auxiliary base pairs for soft constraint feature.

Functions

- void [vrna_sc_init](#) ([vrna_fold_compound_t](#) *vc)
Initialize an empty soft constraints data structure within a [vrna_fold_compound_t](#).
- int [vrna_sc_set_bp](#) ([vrna_fold_compound_t](#) *vc, const [FLT_OR_DBL](#) **constraints, unsigned int options)
Set soft constraints for paired nucleotides.
- int [vrna_sc_add_bp](#) ([vrna_fold_compound_t](#) *vc, int i, int j, [FLT_OR_DBL](#) energy, unsigned int options)
Add soft constraints for paired nucleotides.
- int [vrna_sc_set_up](#) ([vrna_fold_compound_t](#) *vc, const [FLT_OR_DBL](#) *constraints, unsigned int options)
Set soft constraints for unpaired nucleotides.
- int [vrna_sc_add_up](#) ([vrna_fold_compound_t](#) *vc, int i, [FLT_OR_DBL](#) energy, unsigned int options)
Add soft constraints for unpaired nucleotides.
- void [vrna_sc_remove](#) ([vrna_fold_compound_t](#) *vc)
Remove soft constraints from [vrna_fold_compound_t](#).
- void [vrna_sc_free](#) ([vrna_sc_t](#) *sc)
Free memory occupied by a [vrna_sc_t](#) data structure.
- int [vrna_sc_add_data](#) ([vrna_fold_compound_t](#) *vc, void *data, [vrna_auxdata_free_f](#) free_data)
Add an auxiliary data structure for the generic soft constraints callback function.
- int [vrna_sc_add_f](#) ([vrna_fold_compound_t](#) *vc, [vrna_sc_f](#) f)
Bind a function pointer for generic soft constraint feature (MFE version)
- int [vrna_sc_add_bt](#) ([vrna_fold_compound_t](#) *vc, [vrna_sc_bt_f](#) f)
Bind a backtracking function pointer for generic soft constraint feature.
- int [vrna_sc_add_exp_f](#) ([vrna_fold_compound_t](#) *vc, [vrna_sc_exp_f](#) exp_f)
Bind a function pointer for generic soft constraint feature (PF version)

16.13.2 Data Structure Documentation

16.13.2.1 struct [vrna_sc_s](#)

The soft constraints data structure.
Collaboration diagram for [vrna_sc_s](#):

Data Fields

- `int ** energy_up`
Energy contribution for stretches of unpaired nucleotides.
- `FLT_OR_DBL ** exp_energy_up`
Boltzmann Factors of the energy contributions for unpaired sequence stretches.
- `int * up_storage`
Storage container for energy contributions per unpaired nucleotide.
- `vrna_sc_bp_storage_t ** bp_storage`
Storage container for energy contributions per base pair.
- `int * energy_stack`
Pseudo Energy contribution per base pair involved in a stack.
- `FLT_OR_DBL * exp_energy_stack`
Boltzmann weighted pseudo energy contribution per nucleotide involved in a stack.
- `vrna_sc_f f`
A function pointer used for pseudo energy contribution in MFE calculations.
- `vrna_sc_bt_f bt`
A function pointer used to obtain backtraced base pairs in loop regions that were altered by soft constrained pseudo energy contributions.
- `vrna_sc_exp_f exp_f`
A function pointer used for pseudo energy contribution boltzmann factors in PF calculations.
- `void * data`
A pointer to the data object provided for for pseudo energy contribution functions of the generic soft constraints feature.
- `int * energy_bp`
Energy contribution for base pairs.
- `FLT_OR_DBL * exp_energy_bp`
Boltzmann Factors of the energy contribution for base pairs.
- `int ** energy_bp_local`
Energy contribution for base pairs (sliding window approach)
- `FLT_OR_DBL ** exp_energy_bp_local`
Boltzmann Factors of the energy contribution for base pairs (sliding window approach)

16.13.2.1.1 Field Documentation

16.13.2.1.1.1 `f vrna_sc_f vrna_sc_s::f`

A function pointer used for pseudo energy contribution in MFE calculations.

See also

[vrna_sc_add_f\(\)](#)

16.13.2.1.1.2 `bt vrna_sc_bt_f vrna_sc_s::bt`

A function pointer used to obtain backtraced base pairs in loop regions that were altered by soft constrained pseudo energy contributions.

See also

[vrna_sc_add_bt\(\)](#)

16.13.2.1.1.3 exp_f `vrna_sc_exp_f vrna_sc_s::exp_f`

A function pointer used for pseudo energy contribution boltzmann factors in PF calculations.

See also

[vrna_sc_add_exp_f\(\)](#)

16.13.3 Typedef Documentation**16.13.3.1 vrna_sc_f**

```
typedef int(* vrna_sc_f) (int i, int j, int k, int l, unsigned char d, void *data)
#include <ViennaRNA/constraints/soft.h>
```

Callback to retrieve pseudo energy contribution for soft constraint feature.

This is the prototype for callback functions used by the folding recursions to evaluate generic soft constraints. The first four parameters passed indicate the delimiting nucleotide positions of the decomposition, and the parameter `denotes` the decomposition step. The last parameter `data` is the auxiliary data structure associated to the hard constraints via [vrna_sc_add_data\(\)](#), or NULL if no auxiliary data was added.

Notes on Callback Functions This callback enables one to add (pseudo-)energy contributions to individual decompositions of the secondary structure.

See also

[VRNA_DECOMP_PAIR_HP](#), [VRNA_DECOMP_PAIR_IL](#), [VRNA_DECOMP_PAIR_ML](#), [VRNA_DECOMP_ML_ML_ML](#), [VRNA_DECOMP_ML_STEM](#), [VRNA_DECOMP_ML_ML](#), [VRNA_DECOMP_ML_UP](#), [VRNA_DECOMP_ML_ML_STEM](#), [VRNA_DECOMP_ML_COAXIAL](#), [VRNA_DECOMP_EXT_EXT](#), [VRNA_DECOMP_EXT_UP](#), [VRNA_DECOMP_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_EXT](#), [VRNA_DECOMP_EXT_STEM_EXT](#), [VRNA_DECOMP_EXT_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_STEM1](#), [vrna_sc_add_f\(\)](#), [vrna_sc_add_exp_f\(\)](#), [vrna_sc_add_bt\(\)](#), [vrna_sc_add_data\(\)](#)

Parameters

<i>i</i>	Left (5') delimiter position of substructure
<i>j</i>	Right (3') delimiter position of substructure
<i>k</i>	Left delimiter of decomposition
<i>l</i>	Right delimiter of decomposition
<i>d</i>	Decomposition step indicator
<i>data</i>	Auxiliary data

Returns

Pseudo energy contribution in deka-kalories per mol

16.13.3.2 vrna_sc_exp_f

```
typedef FLT_OR_DBL(* vrna_sc_exp_f) (int i, int j, int k, int l, unsigned char d, void *data)
#include <ViennaRNA/constraints/soft.h>
```

Callback to retrieve pseudo energy contribution as Boltzmann Factors for soft constraint feature.

This is the prototype for callback functions used by the partition function recursions to evaluate generic soft constraints. The first four parameters passed indicate the delimiting nucleotide positions of the decomposition, and the parameter `denotes` the decomposition step. The last parameter `data` is the auxiliary data structure associated to the hard constraints via [vrna_sc_add_data\(\)](#), or NULL if no auxiliary data was added.

Notes on Callback Functions This callback enables one to add (pseudo-)energy contributions to individual de-

compositions of the secondary structure (Partition function variant, i.e. contributions must be returned as Boltzmann factors).

See also

[VRNA_DECOMP_PAIR_HP](#), [VRNA_DECOMP_PAIR_IL](#), [VRNA_DECOMP_PAIR_ML](#), [VRNA_DECOMP_ML_ML_ML](#), [VRNA_DECOMP_ML_STEM](#), [VRNA_DECOMP_ML_ML](#), [VRNA_DECOMP_ML_UP](#), [VRNA_DECOMP_ML_ML_STEM](#), [VRNA_DECOMP_ML_COAXIAL](#), [VRNA_DECOMP_EXT_EXT](#), [VRNA_DECOMP_EXT_UP](#), [VRNA_DECOMP_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_EXT](#), [VRNA_DECOMP_EXT_STEM_EXT](#), [VRNA_DECOMP_EXT_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_STEM1](#), [vrna_sc_add_exp_f\(\)](#), [vrna_sc_add_f\(\)](#), [vrna_sc_add_bt\(\)](#), [vrna_sc_add_data\(\)](#)

Parameters

<i>i</i>	Left (5') delimiter position of substructure
<i>j</i>	Right (3') delimiter position of substructure
<i>k</i>	Left delimiter of decomposition
<i>l</i>	Right delimiter of decomposition
<i>d</i>	Decomposition step indicator
<i>data</i>	Auxiliary data

Returns

Pseudo energy contribution in deka-kalories per mol

16.13.3.3 vrna_sc_bt_f

```
typedef vrna\_basepair\_t *(* vrna_sc_bt_f) (int i, int j, int k, int l, unsigned char d, void *data)
```

```
#include <ViennaRNA/constraints/soft.h>
```

Callback to retrieve auxiliary base pairs for soft constraint feature.

Notes on Callback Functions This callback enables one to add auxiliary base pairs in the backtracking steps of hairpin- and interior loops.

See also

[VRNA_DECOMP_PAIR_HP](#), [VRNA_DECOMP_PAIR_IL](#), [VRNA_DECOMP_PAIR_ML](#), [VRNA_DECOMP_ML_ML_ML](#), [VRNA_DECOMP_ML_STEM](#), [VRNA_DECOMP_ML_ML](#), [VRNA_DECOMP_ML_UP](#), [VRNA_DECOMP_ML_ML_STEM](#), [VRNA_DECOMP_ML_COAXIAL](#), [VRNA_DECOMP_EXT_EXT](#), [VRNA_DECOMP_EXT_UP](#), [VRNA_DECOMP_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_EXT](#), [VRNA_DECOMP_EXT_STEM_EXT](#), [VRNA_DECOMP_EXT_EXT_STEM](#), [VRNA_DECOMP_EXT_EXT_STEM1](#), [vrna_sc_add_bt\(\)](#), [vrna_sc_add_f\(\)](#), [vrna_sc_add_exp_f\(\)](#), [vrna_sc_add_data\(\)](#)

Parameters

<i>i</i>	Left (5') delimiter position of substructure
<i>j</i>	Right (3') delimiter position of substructure
<i>k</i>	Left delimiter of decomposition
<i>l</i>	Right delimiter of decomposition
<i>d</i>	Decomposition step indicator
<i>data</i>	Auxiliary data

Returns

List of additional base pairs

16.13.4 Function Documentation

16.13.4.1 `vrna_sc_init()`

```
void vrna_sc_init (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/constraints/soft.h>
```

Initialize an empty soft constraints data structure within a `vrna_fold_compound_t`.

This function adds a proper soft constraints data structure to the `vrna_fold_compound_t` data structure. If soft constraints already exist within the fold compound, they are removed.

Note

Accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE` and `VRNA_FC_TYPE_COMPARATIVE`

See also

`vrna_sc_set_bp()`, `vrna_sc_set_up()`, `vrna_sc_add_SHAPE_deigan()`, `vrna_sc_add_SHAPE_zarringhalam()`, `vrna_sc_remove()`, `vrna_sc_add_f()`, `vrna_sc_add_exp_f()`, `vrna_sc_add_pre()`, `vrna_sc_add_post()`

Parameters

<code>vc</code>	The <code>vrna_fold_compound_t</code> where an empty soft constraint feature is to be added to
-----------------	--

SWIG Wrapper Notes This function is attached as method `sc_init()` to objects of type `fold_compound`

16.13.4.2 `vrna_sc_set_bp()`

```
int vrna_sc_set_bp (
    vrna_fold_compound_t * vc,
    const FLT_OR_DBL ** constraints,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
```

Set soft constraints for paired nucleotides.

Note

This function replaces any pre-existing soft constraints with the ones supplied in `constraints`.

See also

`vrna_sc_add_bp()`, `vrna_sc_set_up()`, `vrna_sc_add_up()`

Parameters

<code>vc</code>	The <code>vrna_fold_compound_t</code> the soft constraints are associated with
<code>constraints</code>	A two-dimensional array of pseudo free energies in <i>kcal/mol</i>
<code>options</code>	The options flag indicating how/where to store the soft constraints

Returns

Non-zero on successful application of the constraint, 0 otherwise.

SWIG Wrapper Notes This function is attached as method **sc_set_bp()** to objects of type *fold_compound*

16.13.4.3 vrna_sc_add_bp()

```
int vrna_sc_add_bp (
    vrna_fold_compound_t * vc,
    int i,
    int j,
    FLT_OR_DBL energy,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
Add soft constraints for paired nucleotides.
```

See also

[vrna_sc_set_bp\(\)](#), [vrna_sc_set_up\(\)](#), [vrna_sc_add_up\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the soft constraints are associated with
<i>i</i>	The 5' position of the base pair the soft constraint is added for
<i>j</i>	The 3' position of the base pair the soft constraint is added for
<i>energy</i>	The free energy (soft-constraint) in <i>kcal/mol</i>
<i>options</i>	The options flag indicating how/where to store the soft constraints

Returns

Non-zero on successful application of the constraint, 0 otherwise.

SWIG Wrapper Notes This function is attached as an overloaded method **sc_add_bp()** to objects of type *fold_compound*. The method either takes arguments for a single base pair (i,j) with the corresponding energy value:

```
fold_compound.sc_add_bp(i, j, energy, options)
```

or an entire 2-dimensional matrix with dimensions n x n that stores free energy contributions for any base pair (i,j) with $1 \leq i < j \leq n$:

```
fold_compound.sc_add_bp(matrix, options)
```

In both variants, the *options* argument is optional can may be omitted.

16.13.4.4 vrna_sc_set_up()

```
int vrna_sc_set_up (
    vrna_fold_compound_t * vc,
    const FLT_OR_DBL * constraints,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
Set soft constraints for unpaired nucleotides.
```

Note

This function replaces any pre-existing soft constraints with the ones supplied in *constraints*.

See also

[vrna_sc_add_up\(\)](#), [vrna_sc_set_bp\(\)](#), [vrna_sc_add_bp\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the soft constraints are associated with
<i>constraints</i>	A vector of pseudo free energies in <i>kcal/mol</i>
<i>options</i>	The options flag indicating how/where to store the soft constraints

Returns

Non-zero on successful application of the constraint, 0 otherwise.

SWIG Wrapper Notes This function is attached as method `sc_set_up()` to objects of type *fold_compound*

16.13.4.5 vrna_sc_add_up()

```
int vrna_sc_add_up (
    vrna_fold_compound_t * vc,
    int i,
    FLT_OR_DBL energy,
    unsigned int options )
#include <ViennaRNA/constraints/soft.h>
Add soft constraints for unpaired nucleotides.
```

See also

[vrna_sc_set_up\(\)](#), [vrna_sc_add_bp\(\)](#), [vrna_sc_set_bp\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the soft constraints are associated with
<i>i</i>	The nucleotide position the soft constraint is added for
<i>energy</i>	The free energy (soft-constraint) in <i>kcal/mol</i>
<i>options</i>	The options flag indicating how/where to store the soft constraints

Returns

Non-zero on successful application of the constraint, 0 otherwise.

SWIG Wrapper Notes This function is attached as an overloaded method `sc_add_up()` to objects of type *fold_compound*. The method either takes arguments for a single nucleotide *i* with the corresponding energy value:

```
fold_compound.sc_add_up(i, energy, options)
or an entire vector that stores free energy contributions for each nucleotide i with 1 ≤ i ≤ n:
fold_compound.sc_add_bp(vector, options)
In both variants, the options argument is optional can may be omitted.
```

16.13.4.6 vrna_sc_remove()

```
void vrna_sc_remove (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/constraints/soft.h>
Remove soft constraints from vrna\_fold\_compound\_t.
```

Note

Accepts [vrna_fold_compound_t](#) of type [VRNA_FC_TYPE_SINGLE](#) and [VRNA_FC_TYPE_COMPARATIVE](#)

Parameters

vc	The vrna_fold_compound_t possibly containing soft constraints
----	---

SWIG Wrapper Notes This function is attached as method **sc_remove()** to objects of type *fold_compound*

16.13.4.7 vrna_sc_free()

```
void vrna_sc_free (
    vrna_sc_t * sc )
#include <ViennaRNA/constraints/soft.h>
Free memory occupied by a vrna\_sc\_t data structure.
```

Parameters

sc	The data structure to free from memory
----	--

16.13.4.8 vrna_sc_add_data()

```
int vrna_sc_add_data (
    vrna_fold_compound_t * vc,
    void * data,
    vrna_auxdata_free_f free_data )
#include <ViennaRNA/constraints/soft.h>
Add an auxiliary data structure for the generic soft constraints callback function.
```

See also

[vrna_sc_add_f\(\)](#), [vrna_sc_add_exp_f\(\)](#), [vrna_sc_add_bt\(\)](#)

Parameters

vc	The fold compound the generic soft constraint function should be bound to
data	A pointer to the data structure that holds required data for function 'f'
free_data	A pointer to a function that free's the memory occupied by <i>data</i> (Maybe NULL)

Returns

Non-zero on successful binding the data (and free-function), 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_add_data()** to objects of type *fold_compound*

16.13.4.9 vrna_sc_add_f()

```
int vrna_sc_add_f (
    vrna_fold_compound_t * vc,
    vrna_sc_f f )
#include <ViennaRNA/constraints/soft.h>
```

Bind a function pointer for generic soft constraint feature (MFE version)

This function allows one to easily bind a function pointer and corresponding data structure to the soft constraint part [vrna_sc_t](#) of the [vrna_fold_compound_t](#). The function for evaluating the generic soft constraint feature has to return a pseudo free energy \hat{E} in *dacal/mol*, where $1\text{dacal/mol} = 10\text{cal/mol}$.

See also

[vrna_sc_add_data\(\)](#), [vrna_sc_add_bt\(\)](#), [vrna_sc_add_exp_f\(\)](#)

Parameters

<i>vc</i>	The fold compound the generic soft constraint function should be bound to
<i>f</i>	A pointer to the function that evaluates the generic soft constraint feature

Returns

Non-zero on successful binding the callback function, 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_add_f()** to objects of type *fold_compound*

16.13.4.10 vrna_sc_add_bt()

```
int vrna_sc_add_bt (
    vrna_fold_compound_t * vc,
    vrna_sc_bt_f f )
#include <ViennaRNA/constraints/soft.h>
```

Bind a backtracking function pointer for generic soft constraint feature.

This function allows one to easily bind a function pointer to the soft constraint part [vrna_sc_t](#) of the [vrna_fold_compound_t](#). The provided function should be used for backtracking purposes in loop regions that were altered via the generic soft constraint feature. It has to return an array of [vrna_basepair_t](#) data structures, were the last element in the list is indicated by a value of -1 in it's i position.

See also

[vrna_sc_add_data\(\)](#), [vrna_sc_add_f\(\)](#), [vrna_sc_add_exp_f\(\)](#)

Parameters

<i>vc</i>	The fold compound the generic soft constraint function should be bound to
<i>f</i>	A pointer to the function that returns additional base pairs

Returns

Non-zero on successful binding the callback function, 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_add_bt()** to objects of type *fold_compound*

16.13.4.11 vrna_sc_add_exp_f()

```
int vrna_sc_add_exp_f (
    vrna_fold_compound_t * vc,
    vrna_sc_exp_f exp_f )
#include <ViennaRNA/constraints/soft.h>
```

Bind a function pointer for generic soft constraint feature (PF version)

This function allows one to easily bind a function pointer and corresponding data structure to the soft constraint part [vrna_sc_t](#) of the [vrna_fold_compound_t](#). The function for evaluating the generic soft constraint feature has to return a pseudo free energy \hat{E} as Boltzmann factor, i.e. $\exp(-\hat{E}/kT)$. The required unit for E is *cal/mol*.

See also

[vrna_sc_add_bt\(\)](#), [vrna_sc_add_f\(\)](#), [vrna_sc_add_data\(\)](#)

Parameters

<code>vc</code>	The fold compound the generic soft constraint function should be bound to
<code>exp_f</code>	A pointer to the function that evaluates the generic soft constraint feature

Returns

Non-zero on successful binding the callback function, 0 otherwise

SWIG Wrapper Notes This function is attached as method `sc_add_exp_f()` to objects of type *fold_compound*

16.14 The RNA Secondary Structure Landscape

16.14.1 Detailed Description

Collaboration diagram for The RNA Secondary Structure Landscape:

Modules

- [Neighborhood Relation and Move Sets for Secondary Structures](#)
Different functions to generate structural neighbors of a secondary structure according to a particular Move Set.
- [\(Re-\)folding Paths, Saddle Points, Energy Barriers, and Local Minima](#)
API for various RNA folding path algorithms.

16.15 Minimum Free Energy (MFE) Algorithms

Predicting the Minimum Free Energy (MFE) and a corresponding (consensus) secondary structure.

16.15.1 Detailed Description

Predicting the Minimum Free Energy (MFE) and a corresponding (consensus) secondary structure.

In a nutshell we provide two different flavors for MFE prediction:

- [Global MFE Prediction](#) - to compute the MFE for the entire sequence
- [Local \(sliding window\) MFE Prediction](#) - to compute MFEs for each window using a sliding window approach

Each of these flavors, again, provides two implementations to either compute the MFE based on

- single RNA (DNA) sequence(s), or
- a comparative approach using multiple sequence alignments (MSA).

For the latter, a consensus secondary structure is predicted and our implementations compute an average of free energies for each sequence in the MSA plus an additional covariance pseudo-energy term.

The implementations for [Backtracking MFE structures](#) are generally agnostic with respect to whether local or global structure prediction is in place. Collaboration diagram for Minimum Free Energy (MFE) Algorithms:

Modules

- [Global MFE Prediction](#)
Variations of the global Minimum Free Energy (MFE) prediction algorithm.
- [Local \(sliding window\) MFE Prediction](#)
Variations of the local (sliding window) Minimum Free Energy (MFE) prediction algorithm.
- [Backtracking MFE structures](#)
Backtracking related interfaces.

Files

- file [mfe.h](#)
Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.
- file [mfe_window.h](#)
Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.

16.16 Partition Function and Equilibrium Properties

Compute the partition function to assess various equilibrium properties.

16.16.1 Detailed Description

Compute the partition function to assess various equilibrium properties.

Similar to our [Minimum Free Energy \(MFE\) Algorithms](#), we provide two different flavors for partition function computations:

- [Global Partition Function and Equilibrium Probabilities](#) - to compute the partition function for a full length sequence
- [Local \(sliding window\) Partition Function and Equilibrium Probabilities](#) - to compute the partition function of each window using a sliding window approach

While the global partition function approach supports predictions using single sequences as well as consensus partition functions for multiple sequence alignments (MSA), we currently do not support MSA input for the local variant.

Comparative prediction computes an average of the free energy contributions plus an additional covariance pseudo-energy term, exactly as we do for the [Minimum Free Energy \(MFE\) Algorithms](#) implementation.

Boltzmann weights for the free energy contributions of individual loops can be found in [Energy Evaluation for Individual Loops](#). Our implementations also provide a stochastic backtracking procedure to draw [Random Structure Samples from the Ensemble](#) according to their equilibrium probability. Collaboration diagram for Partition Function and Equilibrium Properties:

Modules

- [Global Partition Function and Equilibrium Probabilities](#)
Variations of the global partition function algorithm.
- [Local \(sliding window\) Partition Function and Equilibrium Probabilities](#)
Scanning version using a sliding window approach to compute equilibrium probabilities.

Files

- file [concentrations.h](#)
Concentration computations for RNA-RNA interactions.
- file [part_func.h](#)
Partition function implementations.
- file [part_func_window.h](#)
Partition function and equilibrium probability implementation for the sliding window algorithm.

Functions

- int [vrna_pf_float_precision](#) (void)
Find out whether partition function computations are using single precision floating points.

16.16.2 Function Documentation

16.16.2.1 vrna_pf_float_precision()

```
int vrna_pf_float_precision (
    void )
#include <ViennaRNA/part_func.h>
```

Find out whether partition function computations are using single precision floating points.

See also

[FLT_OR_DBL](#)

Returns

1 if single precision is used, 0 otherwise

16.17 Global MFE Prediction

Variations of the global Minimum Free Energy (MFE) prediction algorithm.

16.17.1 Detailed Description

Variations of the global Minimum Free Energy (MFE) prediction algorithm.

We provide implementations of the global MFE prediction algorithm for

- Single sequences,
- Multiple sequence alignments (MSA), and
- RNA-RNA hybrids

Collaboration diagram for Global MFE Prediction:

Modules

- [Computing MFE representatives of a Distance Based Partitioning](#)
Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.
- [Deprecated Interface for Global MFE Prediction](#)

Files

- file [mfe.h](#)
Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.

Basic global MFE prediction interface

- float [vrna_mfe](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Compute minimum free energy and an appropriate secondary structure of an RNA sequence, or RNA sequence alignment.
- float [vrna_mfe_dimer](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Compute the minimum free energy of two interacting RNA molecules.

Simplified global MFE prediction using sequence(s) or multiple sequence alignment(s)

- float [vrna_fold](#) (const char *sequence, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding secondary structure for an RNA sequence.
- float [vrna_circfold](#) (const char *sequence, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding secondary structure for a circular RNA sequence.

- float [vrna_alifold](#) (const char **sequences, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for an RNA sequence alignment using a comparative method.
- float [vrna_circalifold](#) (const char **sequences, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for a sequence alignment of circular RNAs using a comparative method.
- float [vrna_cofold](#) (const char *sequence, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.

16.17.2 Function Documentation

16.17.2.1 vrna_mfe()

```
float vrna_mfe (
    vrna_fold_compound_t * vc,
    char * structure )
```

```
#include <ViennaRNA/mfe.h>
```

Compute minimum free energy and an appropriate secondary structure of an RNA sequence, or RNA sequence alignment.

Depending on the type of the provided [vrna_fold_compound_t](#), this function predicts the MFE for a single sequence (or connected component of multiple sequences), or an averaged MFE for a sequence alignment. If backtracking is activated, it also constructs the corresponding secondary structure, or consensus structure. Therefore, the second parameter, *structure*, has to point to an allocated block of memory with a size of at least `strlen(sequence) + 1` to store the backtracked MFE structure. (For consensus structures, this is the length of the alignment + 1. If `NULL` is passed, no backtracking will be performed.

Note

This function is polymorphic. It accepts [vrna_fold_compound_t](#) of type [VRNA_FC_TYPE_SINGLE](#), and [VRNA_FC_TYPE_COMPARATIVE](#).

See also

[vrna_fold_compound_t](#), [vrna_fold_compound\(\)](#), [vrna_fold\(\)](#), [vrna_circfold\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_alifold\(\)](#), [vrna_circalifold\(\)](#)

Parameters

<i>vc</i>	fold compound
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to (Maybe <code>NULL</code>)

Returns

the minimum free energy (MFE) in kcal/mol

SWIG Wrapper Notes This function is attached as method **mfe()** to objects of type *fold_compound*

16.17.2.2 vrna_mfe_dimer()

```
float vrna_mfe_dimer (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute the minimum free energy of two interacting RNA molecules.
The code is analog to the [vrna_mfe\(\)](#) function.

Deprecated This function is obsolete since [vrna_mfe\(\)](#) can handle complexes multiple sequences since v2.5.0. Use [vrna_mfe\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

See also

[vrna_mfe\(\)](#)

Parameters

<i>vc</i>	fold compound
<i>structure</i>	Will hold the barcket dot structure of the dimer molecule

Returns

minimum free energy of the structure

SWIG Wrapper Notes This function is attached as method **mfe_dimer()** to objects of type *fold_compound*

16.17.2.3 vrna_fold()

```
float vrna_fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding secondary structure for an RNA sequence. This simplified interface to [vrna_mfe\(\)](#) computes the MFE and, if required, a secondary structure for an RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_circfold\(\)](#), [vrna_mfe\(\)](#)

Parameters

<i>sequence</i>	RNA sequence
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.17.2.4 vrna_circfold()

```
float vrna_circfold (
    const char * sequence,
    char * structure )
```

```
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding secondary structure for a circular RNA sequence.

This simplified interface to [vrna_mfe\(\)](#) computes the MFE and, if required, a secondary structure for a circular RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [15] for further details.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_fold\(\)](#), [vrna_mfe\(\)](#)

Parameters

<i>sequence</i>	RNA sequence
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.17.2.5 vrna_alifold()

```
float vrna_alifold (
    const char ** sequences,
    char * structure )
```

```
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for an RNA sequence alignment using a comparative method.

This simplified interface to [vrna_mfe\(\)](#) computes the MFE and, if required, a consensus secondary structure for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_circalifold\(\)](#), [vrna_mfe\(\)](#)

Parameters

<i>sequences</i>	RNA sequence alignment
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.17.2.6 vrna_circalifold()

```
float vrna_circalifold (
    const char ** sequences,
    char * structure )
```

```
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for a sequence alignment of circular RNAs using a comparative method.

This simplified interface to [vrna_mfe\(\)](#) computes the MFE and, if required, a consensus secondary structure for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [15] for further details.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_alifold\(\)](#), [vrna_mfe\(\)](#)

Parameters

<i>sequences</i>	Sequence alignment of circular RNAs
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.17.2.7 vrna_cofold()

```
float vrna_cofold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.

This simplified interface to [vrna_mfe\(\)](#) computes the MFE and, if required, a secondary structure for two RNA sequences upon dimerization using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing, e.g. suboptimal backtracking, etc.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

Deprecated This function is obsolete since [vrna_mfe\(\)](#)/[vrna_fold\(\)](#) can handle complexes multiple sequences since v2.5.0. Use [vrna_mfe\(\)](#)/[vrna_fold\(\)](#) for connected component MFE instead and compute MFEs of unconnected states separately.

See also

[vrna_fold\(\)](#), [vrna_mfe\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#), [vrna_cut_point_insert\(\)](#)

Parameters

<i>sequence</i>	two RNA sequences separated by the '&' character
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.18 Local (sliding window) MFE Prediction

Variations of the local (sliding window) Minimum Free Energy (MFE) prediction algorithm.

16.18.1 Detailed Description

Variations of the local (sliding window) Minimum Free Energy (MFE) prediction algorithm.

We provide implementations for the local (sliding window) MFE prediction algorithm for

- Single sequences,
- Multiple sequence alignments (MSA), and

Note, that our implementation scans an RNA sequence (or MSA) from the 3' to the 5' end, and reports back locally optimal (consensus) structures, the corresponding free energy, and the position of the sliding window in global coordinates.

For any particular RNA sequence (or MSA) multiple locally optimal (consensus) secondary structures may be predicted. Thus, we tried to implement an interface that allows for an effortless conversion of the corresponding hits into any target data structure. As a consequence, we provide two distinct ways to retrieve the corresponding predictions, either

- through directly writing to an open `FILE` stream on-the-fly, or
- through a callback function mechanism.

The latter allows one to store the results in any possible target data structure. Our implementations then pass the results through the user-implemented callback as soon as the prediction for a particular window is finished. Collaboration diagram for Local (sliding window) MFE Prediction:

Modules

- [Deprecated Interface for Local \(Sliding Window\) MFE Prediction](#)

Files

- file [mfe_window.h](#)

Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.

Typedefs

- typedef void(* [vrna_mfe_window_f](#)) (int start, int end, const char *structure, float en, void *data)

The default callback for sliding window MFE structure predictions.

Basic local (sliding window) MFE prediction interface

- float [vrna_mfe_window](#) ([vrna_fold_compound_t](#) *vc, FILE *file)

Local MFE prediction using a sliding window approach.

- float [vrna_mfe_window_cb](#) ([vrna_fold_compound_t](#) *vc, [vrna_mfe_window_f](#) cb, void *data)

- float [vrna_mfe_window_zscore](#) ([vrna_fold_compound_t](#) *vc, double min_z, FILE *file)

Local MFE prediction using a sliding window approach (with z-score cut-off)

- float [vrna_mfe_window_zscore_cb](#) ([vrna_fold_compound_t](#) *vc, double min_z, [vrna_mfe_window_zscore_f](#) cb, void *data)

Simplified local MFE prediction using sequence(s) or multiple sequence alignment(s)

- float [vrna_Lfold](#) (const char *string, int window_size, FILE *file)

Local MFE prediction using a sliding window approach (simplified interface)

- float [vrna_Lfold_cb](#) (const char *string, int window_size, [vrna_mfe_window_f](#) cb, void *data)

- float [vrna_Lfoldz](#) (const char *string, int window_size, double min_z, FILE *file)

Local MFE prediction using a sliding window approach with z-score cut-off (simplified interface)

- float [vrna_Lfoldz_cb](#) (const char *string, int window_size, double min_z, [vrna_mfe_window_zscore_f](#) cb, void *data)

- float [vrna_alifold](#) (const char **alignment, int maxdist, FILE *fp)

- float [vrna_alifold_cb](#) (const char **alignment, int maxdist, [vrna_mfe_window_f](#) cb, void *data)

16.18.2 Typedef Documentation

16.18.2.1 [vrna_mfe_window_f](#)

```
typedef void(* vrna_mfe_window_f) (int start, int end, const char *structure, float en, void *data)
```

```
#include <ViennaRNA/mfe_window.h>
```

The default callback for sliding window MFE structure predictions.

Notes on Callback Functions This function will be called for each hit in a sliding window MFE prediction.

Parameters

See also

[vrna_mfe_window\(\)](#)

Parameters

<i>start</i>	provides the first position of the hit (1-based, relative to entire sequence/alignment)
<i>end</i>	provides the last position of the hit (1-based, relative to the entire sequence/alignment)
<i>structure</i>	provides the (sub)structure in dot-bracket notation
<i>en</i>	is the free energy of the structure hit in kcal/mol
<i>data</i>	is some arbitrary data pointer passed through by the function executing the callback

16.18.3 Function Documentation

16.18.3.1 vrna_mfe_window()

```
float vrna_mfe_window (
    vrna_fold_compound_t * vc,
    FILE * file )
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach.

Computes minimum free energy structures using a sliding window approach, where base pairs may not span outside the window. In contrast to [vrna_mfe\(\)](#), where a maximum base pair span may be set using the [vrna_md_t.max_bp_span](#) attribute and one globally optimal structure is predicted, this function uses a sliding window to retrieve all locally optimal structures within each window. The size of the sliding window is set in the [vrna_md_t.window_size](#) attribute, prior to the retrieval of the [vrna_fold_compound_t](#) using [vrna_fold_compound\(\)](#) with option [VRNA_OPTION_WINDOW](#)

The predicted structures are written on-the-fly, either to stdout, if a NULL pointer is passed as file parameter, or to the corresponding filehandle.

See also

[vrna_fold_compound\(\)](#), [vrna_mfe_window_zscore\(\)](#), [vrna_mfe\(\)](#), [vrna_Lfold\(\)](#), [vrna_Lfoldz\(\)](#), [VRNA_OPTION_WINDOW](#), [vrna_md_t.max_bp_span](#), [vrna_md_t.window_size](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t with preallocated memory for the DP matrices
<i>file</i>	The output file handle where predictions are written to (maybe NULL)

SWIG Wrapper Notes This function is attached as method **mfe_window()** to objects of type *fold_compound*

16.18.3.2 vrna_mfe_window_zscore()

```
float vrna_mfe_window_zscore (
    vrna_fold_compound_t * vc,
    double min_z,
    FILE * file )
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach (with z-score cut-off)

Computes minimum free energy structures using a sliding window approach, where base pairs may not span outside the window. This function is the z-score version of [vrna_mfe_window\(\)](#), i.e. only predictions above a certain z-score cut-off value are printed. As for [vrna_mfe_window\(\)](#), the size of the sliding window is set in the [vrna_md_t.window_size](#) attribute, prior to the retrieval of the [vrna_fold_compound_t](#) using [vrna_fold_compound\(\)](#) with option [VRNA_OPTION_WINDOW](#).

The predicted structures are written on-the-fly, either to stdout, if a NULL pointer is passed as file parameter, or to the corresponding filehandle.

See also

[vrna_fold_compound\(\)](#), [vrna_mfe_window_zscore\(\)](#), [vrna_mfe\(\)](#), [vrna_Lfold\(\)](#), [vrna_Lfoldz\(\)](#), [VRNA_OPTION_WINDOW](#), [vrna_md_t.max_bp_span](#), [vrna_md_t.window_size](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t with preallocated memory for the DP matrices
<i>min_z</i>	The minimal z-score for a predicted structure to appear in the output
<i>file</i>	The output file handle where predictions are written to (maybe NULL)

16.18.3.3 vrna_Lfold()

```
float vrna_Lfold (
    const char * string,
    int window_size,
    FILE * file )
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach (simplified interface)

This simplified interface to [vrna_mfe_window\(\)](#) computes the MFE and locally optimal secondary structure using default options. Structures are predicted using a sliding window approach, where base pairs may not span outside the window. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe_window\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_mfe_window\(\)](#), [vrna_Lfoldz\(\)](#), [vrna_mfe_window_zscore\(\)](#)

Parameters

<i>string</i>	The nucleic acid sequence
<i>window_size</i>	The window size for locally optimal structures
<i>file</i>	The output file handle where predictions are written to (if NULL, output is written to stdout)

16.18.3.4 vrna_Lfoldz()

```
float vrna_Lfoldz (
```

```
const char * string,
int window_size,
double min_z,
FILE * file )
```

```
#include <ViennaRNA/mfe_window.h>
```

Local MFE prediction using a sliding window approach with z-score cut-off (simplified interface)

This simplified interface to [vrna_mfe_window_zscore\(\)](#) computes the MFE and locally optimal secondary structure using default options. Structures are predicted using a sliding window approach, where base pairs may not span outside the window. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing. This function is the z-score version of [vrna_Lfold\(\)](#), i.e. only predictions above a certain z-score cut-off value are printed.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_mfe_window\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_mfe_window_zscore\(\)](#), [vrna_Lfold\(\)](#), [vrna_mfe_window\(\)](#)

Parameters

<i>string</i>	The nucleic acid sequence
<i>window_size</i>	The window size for locally optimal structures
<i>min_z</i>	The minimal z-score for a predicted structure to appear in the output
<i>file</i>	The output file handle where predictions are written to (if NULL, output is written to stdout)

16.19 Backtracking MFE structures

Backtracking related interfaces.

16.19.1 Detailed Description

Backtracking related interfaces.

Collaboration diagram for Backtracking MFE structures:

Functions

- float [vrna_backtrack5](#) ([vrna_fold_compound_t](#) *fc, unsigned int length, char *structure)
Backtrack an MFE (sub)structure.
- int [vrna_BT_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int en, [vrna_bp_stack_t](#) *bp_stack, int *stack_count)
Backtrack a hairpin loop closed by (i, j).
- int [vrna_BT_stack](#) ([vrna_fold_compound_t](#) *fc, int *i, int *j, int *en, [vrna_bp_stack_t](#) *bp_stack, int *stack_count)
Backtrack a stacked pair closed by (i, j).
- int [vrna_BT_int_loop](#) ([vrna_fold_compound_t](#) *fc, int *i, int *j, int en, [vrna_bp_stack_t](#) *bp_stack, int *stack_count)
Backtrack an interior loop closed by (i, j).
- int [vrna_BT_mb_loop](#) ([vrna_fold_compound_t](#) *fc, int *i, int *j, int *k, int en, int *component1, int *component2)
Backtrack the decomposition of a multi branch loop closed by (i, j).

16.19.2 Function Documentation

16.19.2.1 `vrna_backtrack5()`

```
float vrna_backtrack5 (
    vrna_fold_compound_t * fc,
    unsigned int length,
    char * structure )
#include <ViennaRNA/mfe.h>
```

Backtrack an MFE (sub)structure.

This function allows one to backtrack the MFE structure for a (sub)sequence

Note

On error, the function returns INF / 100. and stores the empty string in `structure`.

Precondition

Requires pre-filled MFE dynamic programming matrices, i.e. one has to call `vrna_mfe()` prior to calling this function

See also

`vrna_mfe()`, `vrna_pbacktrack5()`

Parameters

<i>fc</i>	fold compound
<i>length</i>	The length of the subsequence, starting from the 5' end
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to. (Must have size of at least $\$p \text{ length} + 1$)

Returns

The minimum free energy (MFE) for the specified `length` in kcal/mol and a corresponding secondary structure in dot-bracket notation (stored in `structure`)

SWIG Wrapper Notes This function is attached as overloaded method `backtrack()` to objects of type `fold_compound` with default parameter `length` equal to the total length of the RNA.

16.19.2.2 `vrna_BT_hp_loop()`

```
int vrna_BT_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j,
    int en,
    vrna_bp_stack_t * bp_stack,
    int * stack_count )
#include <ViennaRNA/loops/hairpin.h>
```

Backtrack a hairpin loop closed by (i, j) .

Note

This function is polymorphic! The provided `vrna_fold_compound_t` may be of type `VRNA_FC_TYPE_SINGLE` or `VRNA_FC_TYPE_COMPARATIVE`

16.19.2.3 vrna_BT_stack()

```
int vrna_BT_stack (
    vrna_fold_compound_t * fc,
    int * i,
    int * j,
    int * en,
    vrna_bp_stack_t * bp_stack,
    int * stack_count )
#include <ViennaRNA/loops/internal.h>
Backtrack a stacked pair closed by  $(i, j)$ .
```

16.19.2.4 vrna_BT_int_loop()

```
int vrna_BT_int_loop (
    vrna_fold_compound_t * fc,
    int * i,
    int * j,
    int en,
    vrna_bp_stack_t * bp_stack,
    int * stack_count )
#include <ViennaRNA/loops/internal.h>
Backtrack an interior loop closed by  $(i, j)$ .
```

16.19.2.5 vrna_BT_mb_loop()

```
int vrna_BT_mb_loop (
    vrna_fold_compound_t * fc,
    int * i,
    int * j,
    int * k,
    int en,
    int * component1,
    int * component2 )
#include <ViennaRNA/loops/multibranch.h>
Backtrack the decomposition of a multi branch loop closed by  $(i, j)$ .
```

Parameters

<i>fc</i>	The <code>vrna_fold_compound_t</code> filled with all relevant data for backtracking
<i>i</i>	5' position of base pair closing the loop (will be set to 5' position of leftmost decomposed block upon successful backtracking)
<i>j</i>	3' position of base pair closing the loop (will be set to 3' position of rightmost decomposed block upon successful backtracking)
<i>k</i>	Split position that delimits leftmost from rightmost block, $[i, k]$ and $[k+1, j]$, respectively. (Will be set upon successful backtracking)
<i>en</i>	The energy contribution of the substructure enclosed by (i, j)
<i>component1</i>	Type of leftmost block (1 = ML, 2 = C)
<i>component2</i>	Type of rightmost block (1 = ML, 2 = C)

Returns

1, if backtracking succeeded, 0 otherwise.

16.20 Global Partition Function and Equilibrium Probabilities

Variations of the global partition function algorithm.

16.20.1 Detailed Description

Variations of the global partition function algorithm.

We provide implementations of the global partition function algorithm for

- Single sequences,
- Multiple sequence alignments (MSA), and
- RNA-RNA hybrids

Collaboration diagram for Global Partition Function and Equilibrium Probabilities:

Modules

- [Computing Partition Functions of a Distance Based Partitioning](#)
Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.
- [Predicting various thermodynamic properties](#)
Compute various thermodynamic properties using the partition function.
- [Deprecated Interface for Global Partition Function Computation](#)

Files

- file [part_func.h](#)
Partition function implementations.

Data Structures

- struct [vrna_dimer_pf_s](#)
Data structure returned by [vrna_pf_dimer\(\)](#) [More...](#)
- struct [vrna_multimer_pf_s](#)

Functions

- [vrna_ep_t](#) * [vrna_plist_from_probs](#) ([vrna_fold_compound_t](#) *vc, double cut_off)
Create a [vrna_ep_t](#) from base pair probability matrix.

Basic global partition function interface

- [FLT_OR_DBL](#) [vrna_pf](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Compute the partition function Q for a given RNA sequence, or sequence alignment.
- [vrna_dimer_pf_t](#) [vrna_pf_dimer](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.
- [FLT_OR_DBL](#) * [vrna_pf_substrands](#) ([vrna_fold_compound_t](#) *fc, [size_t](#) complex_size)
- [FLT_OR_DBL](#) [vrna_pf_add](#) ([FLT_OR_DBL](#) dG1, [FLT_OR_DBL](#) dG2, double kT)

Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- float [vrna_pf_fold](#) (const char *sequence, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for an RNA sequence using a comparative method.
- float [vrna_pf_circfold](#) (const char *sequence, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for a circular RNA sequences using a comparative method.
- float [vrna_pf_alifold](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for an RNA sequence alignment using a comparative method.
- float [vrna_pf_circalifold](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for an alignment of circular RNA sequences using a comparative method.
- [vrna_dimer_pf_t vrna_pf_co_fold](#) (const char *seq, char *structure, [vrna_ep_t](#) **pl)
Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

16.20.2 Data Structure Documentation

16.20.2.1 struct vrna_dimer_pf_s

Data structure returned by [vrna_pf_dimer\(\)](#)

Data Fields

- double **F0AB**
Null model without DuplexInit.
- double **FAB**
all states with DuplexInit correction
- double **FcAB**
true hybrid states only
- double **FA**
monomer A
- double **FB**
monomer B

16.20.2.2 struct vrna_multimer_pf_s

Data Fields

- double **F_connected**
Fully connected ensemble (incl. DuplexInitiation and rotational symmetry correction.
- double * **F_monomers**
monomers
- size_t **num_monomers**
Number of monomers.

16.20.3 Function Documentation

16.20.3.1 vrna_pf()

```
float vrna_pf (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Compute the partition function Q for a given RNA sequence, or sequence alignment.

If *structure* is not a NULL pointer on input, it contains on return a string consisting of the letters ". , | { } () " denoting bases that are essentially unpaired, weakly paired, strongly paired without preference, weakly upstream (downstream) paired, or strongly up- (down-)stream paired bases, respectively. If the model's compute_bpp is set to 0 base pairing probabilities will not be computed (saving CPU time), otherwise after calculations took place *pr* will contain the probability that bases *i* and *j* pair.

Note

This function is polymorphic. It accepts *vrna_fold_compound_t* of type *VRNA_FC_TYPE_SINGLE*, and *VRNA_FC_TYPE_COMPARATIVE*.

This function may return INF / 100. in case of contradicting constraints or numerical over-/underflow. In the latter case, a corresponding warning will be issued to *stdout*.

See also

[vrna_fold_compound_t](#), [vrna_fold_compound\(\)](#), [vrna_pf_fold\(\)](#), [vrna_pf_circfold\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_pf_alifold\(\)](#), [vrna_pf_circalifold\(\)](#), [vrna_db_from_probs\(\)](#), [vrna_exp_params\(\)](#), [vrna_aln_pinfo\(\)](#)

Parameters

in, out	<i>vc</i>	The fold compound data structure
in, out	<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

SWIG Wrapper Notes This function is attached as method **pf()** to objects of type *fold_compound*

16.20.3.2 vrna_pf_dimer()

```
vrna_dimer_pf_t vrna_pf_dimer (
    vrna_fold_compound_t * vc,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

This is the cofold partition function folding.

Note

This function may return INF / 100. for the FA, FB, FAB, F0AB members of the output data structure in case of contradicting constraints or numerical over-/underflow. In the latter case, a corresponding warning will be issued to *stdout*.

See also

[vrna_fold_compound\(\)](#) for how to retrieve the necessary data structure

Parameters

<i>vc</i>	the fold compound data structure
<i>structure</i>	Will hold the structure or constraints

Returns

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

SWIG Wrapper Notes This function is attached as method **pf_dimer()** to objects of type *fold_compound*

16.20.3.3 `vrna_pf_fold()`

```
float vrna_pf_fold (
    const char * sequence,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function Q (and base pair probabilities) for an RNA sequence using a comparative method. This simplified interface to `vrna_pf()` computes the partition function and, if required, base pair probabilities for an RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use `vrna_pf()`, and the data structure `vrna_fold_compound_t` instead.

See also

[vrna_pf_circfold\(\)](#), [vrna_pf\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#)

Parameters

<i>sequence</i>	RNA sequence
<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)
<i>pl</i>	A pointer to a list of vrna_ep_t to store pairing probabilities (Maybe NULL)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.20.3.4 `vrna_pf_circfold()`

```
float vrna_pf_circfold (
    const char * sequence,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function Q (and base pair probabilities) for a circular RNA sequences using a comparative method.

This simplified interface to `vrna_pf()` computes the partition function and, if required, base pair probabilities for a circular RNA sequence using default options. Memory required for dynamic programming (DP) matrices will be

allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_pf\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [15] for further details.

See also

[vrna_pf_fold\(\)](#), [vrna_pf\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#)

Parameters

<i>sequence</i>	A circular RNA sequence
<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)
<i>pl</i>	A pointer to a list of vrna_ep_t to store pairing probabilities (Maybe NULL)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.20.3.5 vrna_pf_alifold()

```
float vrna_pf_alifold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function Q (and base pair probabilities) for an RNA sequence alignment using a comparative method.

This simplified interface to [vrna_pf\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_pf\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_pf_circalifold\(\)](#), [vrna_pf\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_fold_compound_t](#)

Parameters

<i>sequences</i>	RNA sequence alignment
<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)
<i>pl</i>	A pointer to a list of vrna_ep_t to store pairing probabilities (Maybe NULL)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.20.3.6 `vrna_pf_circalifold()`

```
float vrna_pf_circalifold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Compute Partition function Q (and base pair probabilities) for an alignment of circular RNA sequences using a comparative method.

This simplified interface to `vrna_pf()` computes the partition function and, if required, base pair probabilities for an RNA sequence alignment using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use `vrna_pf()`, and the data structure `vrna_fold_compound_t` instead.

Folding of circular RNA sequences is handled as a post-processing step of the forward recursions. See [15] for further details.

See also

[vrna_pf_alifold\(\)](#), [vrna_pf\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_fold_compound_t](#)

Parameters

<i>sequences</i>	Sequence alignment of circular RNAs
<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)
<i>pl</i>	A pointer to a list of <code>vrna_ep_t</code> to store pairing probabilities (Maybe NULL)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.20.3.7 `vrna_plist_from_probs()`

```
vrna_ep_t * vrna_plist_from_probs (
    vrna_fold_compound_t * vc,
    double cut_off )
#include <ViennaRNA/utils/structures.h>
```

Create a `vrna_ep_t` from base pair probability matrix.

The probability matrix provided via the `vrna_fold_compound_t` is parsed and all pair probabilities above the given threshold are used to create an entry in the plist

The end of the plist is marked by sequence positions i as well as j equal to 0. This condition should be used to stop looping over its entries

Parameters

in	<i>vc</i>	The fold compound
in	<i>cut_off</i>	The cutoff value

Returns

A pointer to the plist that is to be created

16.20.3.8 vrna_pf_co_fold()

```
vrna_dimer_pf_t vrna_pf_co_fold (
    const char * seq,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

This simplified interface to [vrna_pf_dimer\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA-RNA interaction using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_pf_dimer\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_pf_dimer\(\)](#)

Parameters

<i>seq</i>	Two concatenated RNA sequences with a delimiting '&' in between
<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)
<i>pl</i>	A pointer to a list of vrna_ep_t to store pairing probabilities (Maybe NULL)

Returns

vrna_dimer_pf_t structure containing a set of energies needed for concentration computations.

16.21 Local (sliding window) Partition Function and Equilibrium Probabilities

Scanning version using a sliding window approach to compute equilibrium probabilities.

16.21.1 Detailed Description

Scanning version using a sliding window approach to compute equilibrium probabilities.

Collaboration diagram for Local (sliding window) Partition Function and Equilibrium Probabilities:

Modules

- [Deprecated Interface for Local \(Sliding Window\) Partition Function Computation](#)

Files

- file [part_func_window.h](#)
Partition function and equilibrium probability implementation for the sliding window algorithm.

Macros

- `#define VRNA_EXT_LOOP 1U`
Exterior loop.
- `#define VRNA_HP_LOOP 2U`
Hairpin loop.
- `#define VRNA_INT_LOOP 4U`
Internal loop.
- `#define VRNA_MB_LOOP 8U`
Multibranch loop.
- `#define VRNA_ANY_LOOP (VRNA_EXT_LOOP | VRNA_HP_LOOP | VRNA_INT_LOOP | VRNA_MB_LOOP)`
Any loop.
- `#define VRNA_PROBS_WINDOW_BPP 4096U`
Trigger base pairing probabilities.
- `#define VRNA_PROBS_WINDOW_UP 8192U`
Trigger unpaired probabilities.
- `#define VRNA_PROBS_WINDOW_STACKP 16384U`
Trigger base pair stack probabilities.
- `#define VRNA_PROBS_WINDOW_UP_SPLIT 32768U`
Trigger detailed unpaired probabilities split up into different loop type contexts.
- `#define VRNA_PROBS_WINDOW_PF 65536U`
Trigger partition function.

Typedefs

- `typedef void(* vrna_probs_window_f) (FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int type, void *data)`
Sliding window probability computation callback.

Basic local partition function interface

- `int vrna_probs_window (vrna_fold_compound_t *fc, int ulength, unsigned int options, vrna_probs_window_f cb, void *data)`
Compute various equilibrium probabilities under a sliding window approach.

Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- `vrna_ep_t * vrna_pfl_fold (const char *sequence, int window_size, int max_bp_span, float cutoff)`
Compute base pair probabilities using a sliding-window approach.
- `int vrna_pfl_fold_cb (const char *sequence, int window_size, int max_bp_span, vrna_probs_window_f cb, void *data)`
Compute base pair probabilities using a sliding-window approach (callback version)
- `double ** vrna_pfl_fold_up (const char *sequence, int ulength, int window_size, int max_bp_span)`
Compute probability of contiguous unpaired segments.
- `int vrna_pfl_fold_up_cb (const char *sequence, int ulength, int window_size, int max_bp_span, vrna_probs_window_f cb, void *data)`
Compute probability of contiguous unpaired segments.

16.21.2 Macro Definition Documentation

16.21.2.1 VRNA_PROBS_WINDOW_BPP

```
#define VRNA_PROBS_WINDOW_BPP 4096U
#include <ViennaRNA/part_func_window.h>
```

Trigger base pairing probabilities.

Passing this flag to [vrna_probs_window\(\)](#) activates callback execution for base pairing probabilities. In turn, the corresponding callback receives this flag through the `type` argument whenever base pairing probabilities are provided.

Detailed information for the algorithm to compute unpaired probabilities can be taken from [3].

See also

[vrna_probs_window\(\)](#)

16.21.2.2 VRNA_PROBS_WINDOW_UP

```
#define VRNA_PROBS_WINDOW_UP 8192U
#include <ViennaRNA/part_func_window.h>
```

Trigger unpaired probabilities.

Passing this flag to [vrna_probs_window\(\)](#) activates callback execution for unpaired probabilities. In turn, the corresponding callback receives this flag through the `type` argument whenever unpaired probabilities are provided.

Detailed information for the algorithm to compute unpaired probabilities can be taken from [4].

See also

[vrna_probs_window\(\)](#)

16.21.2.3 VRNA_PROBS_WINDOW_STACKP

```
#define VRNA_PROBS_WINDOW_STACKP 16384U
#include <ViennaRNA/part_func_window.h>
```

Trigger base pair stack probabilities.

Passing this flag to [vrna_probs_window\(\)](#) activates callback execution for stacking probabilities. In turn, the corresponding callback receives this flag through the `type` argument whenever stack probabilities are provided.

Bug Currently, this flag is a placeholder doing nothing as the corresponding implementation for stack probability computation is missing.

See also

[vrna_probs_window\(\)](#)

16.21.2.4 VRNA_PROBS_WINDOW_UP_SPLIT

```
#define VRNA_PROBS_WINDOW_UP_SPLIT 32768U
#include <ViennaRNA/part_func_window.h>
```

Trigger detailed unpaired probabilities split up into different loop type contexts.

Passing this flag to [vrna_probs_window\(\)](#) activates callback execution for unpaired probabilities. In contrast to [VRNA_PROBS_WINDOW_UP](#) this flag requests unpaired probabilities to be split up into different loop type contexts. In turn, the corresponding callback receives the [VRNA_PROBS_WINDOW_UP](#) flag OR-ed together with the corresponding loop type, i.e.:

- [VRNA_EXT_LOOP](#) - Exterior loop.
- [VRNA_HP_LOOP](#) - Hairpin loop.
- [VRNA_INT_LOOP](#) - Internal loop.

- [VRNA_MB_LOOP](#) - Multibranch loop.
- [VRNA_ANY_LOOP](#) - Any loop.

See also

[vrna_probs_window\(\)](#), [VRNA_PROBS_WINDOW_UP](#)

16.21.2.5 VRNA_PROBS_WINDOW_PF

```
#define VRNA_PROBS_WINDOW_PF 65536U
#include <ViennaRNA/part_func_window.h>
```

Trigger partition function.

Passing this flag to [vrna_probs_window\(\)](#) activates callback execution for partition function. In turn, the corresponding callback receives this flag through its `type` argument whenever partition function data is provided.

Note

Instead of actually providing the partition function Z , the callback is always provided with the corresponding ensemble free energy $\Delta G = -RT \ln Z$.

See also

[vrna_probs_window\(\)](#)

16.21.3 Typedef Documentation

16.21.3.1 vrna_probs_window_f

```
typedef void(* vrna_probs_window_f) (FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int
type, void *data)
#include <ViennaRNA/part_func_window.h>
```

Sliding window probability computation callback.

Notes on Callback Functions This function will be called for each probability data set in the sliding window probability computation implementation of [vrna_probs_window\(\)](#). The argument *type* specifies the type of probability that is passed to this function.

Types:

- [VRNA_PROBS_WINDOW_BPP](#) - Trigger base pairing probabilities.
- [VRNA_PROBS_WINDOW_UP](#) - Trigger unpaired probabilities.
- [VRNA_PROBS_WINDOW_PF](#) - Trigger partition function.

The above types usually come exclusively. However, for unpaired probabilities, the [VRNA_PROBS_WINDOW_UP](#) flag is OR-ed together with one of the loop type contexts

- [VRNA_EXT_LOOP](#) - Exterior loop.
- [VRNA_HP_LOOP](#) - Hairpin loop.
- [VRNA_INT_LOOP](#) - Internal loop.
- [VRNA_MB_LOOP](#) - Multibranch loop.
- [VRNA_ANY_LOOP](#) - Any loop.

to indicate the particular type of data available through the `pr` pointer.

See also

[vrna_probs_window\(\)](#), [vrna_pfl_fold_up_cb\(\)](#)

Parameters

<i>pr</i>	An array of probabilities
<i>pr_size</i>	The length of the probability array
<i>i</i>	The i-position (5') of the probabilities
<i>max</i>	The (theoretical) maximum length of the probability array
<i>type</i>	The type of data that is provided
<i>data</i>	Auxiliary data

16.21.4 Function Documentation

16.21.4.1 `vrna_probs_window()`

```
int vrna_probs_window (
    vrna_fold_compound_t * fc,
    int ulength,
    unsigned int options,
    vrna_probs_window_f cb,
    void * data )
```

```
#include <ViennaRNA/part_func_window.h>
```

Compute various equilibrium probabilities under a sliding window approach.

This function applies a sliding window scan for the sequence provided with the argument `fc` and reports back equilibrium probabilities through the callback function `cb`. The data reported to the callback depends on the `options` flag.

Note

The parameter `ulength` only affects computation and resulting data if unpaired probability computations are requested through the `options` flag.

Options:

- `VRNA_PROBS_WINDOW_BPP` - Trigger base pairing probabilities.
- `VRNA_PROBS_WINDOW_UP` - Trigger unpaired probabilities.
- `VRNA_PROBS_WINDOW_UP_SPLIT` - Trigger detailed unpaired probabilities split up into different loop type contexts.

Options may be OR-ed together

See also

`vrna_pfl_fold_cb()`, `vrna_pfl_fold_up_cb()`

Parameters

<i>fc</i>	The fold compound with sequence data, model settings and precomputed energy parameters
<i>ulength</i>	The maximal length of an unpaired segment (only for unpaired probability computations)
<i>cb</i>	The callback function which collects the pair probability data for further processing
<i>data</i>	Some arbitrary data structure that is passed to the callback <code>cb</code>
<i>options</i>	Option flags to control the behavior of this function

Returns

0 on failure, non-zero on success

16.21.4.2 vrna_pfl_fold()

```
vrna_ep_t * vrna_pfl_fold (
    const char * sequence,
    int window_size,
    int max_bp_span,
    float cutoff )
#include <ViennaRNA/part_func_window.h>
```

Compute base pair probabilities using a sliding-window approach.

This is a simplified wrapper to [vrna_probs_window\(\)](#) that given a nucleic acid sequence, a window size, a maximum base pair span, and a cutoff value computes the pair probabilities for any base pair in any window. The pair probabilities are returned as a list and the user has to take care to free() the memory occupied by the list.

Note

This function uses default model settings! For custom model settings, we refer to the function [vrna_probs_window\(\)](#).

In case of any computation errors, this function returns NULL

See also

[vrna_probs_window\(\)](#), [vrna_pfl_fold_cb\(\)](#), [vrna_pfl_fold_up\(\)](#)

Parameters

<i>sequence</i>	The nucleic acid input sequence
<i>window_size</i>	The size of the sliding window
<i>max_bp_span</i>	The maximum distance along the backbone between two nucleotides that form a base pairs
<i>cutoff</i>	A cutoff value that omits all pairs with lower probability

Returns

A list of base pair probabilities, terminated by an entry with [vrna_ep_t.i](#) and [vrna_ep_t.j](#) set to 0

16.21.4.3 vrna_pfl_fold_cb()

```
int vrna_pfl_fold_cb (
    const char * sequence,
    int window_size,
    int max_bp_span,
    vrna_probs_window_f cb,
    void * data )
#include <ViennaRNA/part_func_window.h>
```

Compute base pair probabilities using a sliding-window approach (callback version)

This is a simplified wrapper to [vrna_probs_window\(\)](#) that given a nucleic acid sequence, a window size, a maximum base pair span, and a cutoff value computes the pair probabilities for any base pair in any window. It is similar to [vrna_pfl_fold\(\)](#) but uses a callback mechanism to return the pair probabilities.

Read the details for [vrna_probs_window\(\)](#) for details on the callback implementation!

Note

This function uses default model settings! For custom model settings, we refer to the function [vrna_probs_window\(\)](#).

See also

[vrna_probs_window\(\)](#), [vrna_pfl_fold\(\)](#), [vrna_pfl_fold_up_cb\(\)](#)

Parameters

<i>sequence</i>	The nucleic acid input sequence
<i>window_size</i>	The size of the sliding window
<i>max_bp_span</i>	The maximum distance along the backbone between two nucleotides that form a base pairs
<i>cb</i>	The callback function which collects the pair probability data for further processing
<i>data</i>	Some arbitrary data structure that is passed to the callback <i>cb</i>

Returns

0 on failure, non-zero on success

16.21.4.4 vrna_pfl_fold_up()

```
double ** vrna_pfl_fold_up (
    const char * sequence,
    int ulength,
    int window_size,
    int max_bp_span )
#include <ViennaRNA/part_func_window.h>
```

Compute probability of contiguous unpaired segments.

This is a simplified wrapper to [vrna_probs_window\(\)](#) that given a nucleic acid sequence, a maximum length of unpaired segments (*ulength*), a window size, and a maximum base pair span computes the equilibrium probability of any segment not exceeding *ulength*. The probabilities to be unpaired are returned as a 1-based, 2-dimensional matrix with dimensions $N \times M$, where N is the length of the sequence and M is the maximum segment length. As an example, the probability of a segment of size 5 starting at position 100 is stored in the matrix entry $X[100][5]$. It is the users responsibility to free the memory occupied by this matrix.

Note

This function uses default model settings! For custom model settings, we refer to the function [vrna_probs_window\(\)](#).

Parameters

<i>sequence</i>	The nucleic acid input sequence
<i>ulength</i>	The maximal length of an unpaired segment
<i>window_size</i>	The size of the sliding window
<i>max_bp_span</i>	The maximum distance along the backbone between two nucleotides that form a base pairs

Returns

The probabilities to be unpaired for any segment not exceeding *ulength*

16.21.4.5 vrna_pfl_fold_up_cb()

```
int vrna_pfl_fold_up_cb (
    const char * sequence,
    int ulength,
    int window_size,
    int max_bp_span,
    vrna_probs_window_f cb,
    void * data )
#include <ViennaRNA/part_func_window.h>
```

Compute probability of contiguous unpaired segments.

This is a simplified wrapper to [vrna_probs_window\(\)](#) that given a nucleic acid sequence, a maximum length of unpaired segments (`ulength`), a window size, and a maximum base pair span computes the equilibrium probability of any segment not exceeding `ulength`. It is similar to [vrna_pfl_fold_up\(\)](#) but uses a callback mechanism to return the unpaired probabilities.

Read the details for [vrna_probs_window\(\)](#) for details on the callback implementation!

Note

This function uses default model settings! For custom model settings, we refer to the function [vrna_probs_window\(\)](#).

Parameters

<i>sequence</i>	The nucleic acid input sequence
<i>ulength</i>	The maximal length of an unpaired segment
<i>window_size</i>	The size of the sliding window
<i>max_bp_span</i>	The maximum distance along the backbone between two nucleotides that form a base pairs
<i>cb</i>	The callback function which collects the pair probability data for further processing
<i>data</i>	Some arbitrary data structure that is passed to the callback <code>cb</code>

Returns

0 on failure, non-zero on success

16.22 Suboptimals and Representative Structures

Sample and enumerate suboptimal secondary structures from RNA sequence data.

16.22.1 Detailed Description

Sample and enumerate suboptimal secondary structures from RNA sequence data.

Collaboration diagram for Suboptimals and Representative Structures:

Modules

- [Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989](#)
- [Suboptimal Structures within an Energy Band around the MFE](#)
- [Random Structure Samples from the Ensemble](#)

Functions to draw random structure samples from the ensemble according to their equilibrium probability.

- [Compute the Structure with Maximum Expected Accuracy \(MEA\)](#)
- [Compute the Centroid Structure](#)

Files

- file [boltzmann_sampling.h](#)

- Boltzmann Sampling of secondary structures from the ensemble.
- file [centroid.h](#)
Centroid structure computation.
- file [MEA.h](#)
Computes a MEA (maximum expected accuracy) structure.
- file [mm.h](#)
Several Maximum Matching implementations.
- file [subopt.h](#)
RNAsubopt and density of states declarations.

16.23 Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989

16.23.1 Detailed Description

Collaboration diagram for Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989:

Functions

- [SOLUTION](#) * [zukersubopt](#) (const char *string)
Compute Zuker type suboptimal structures.
- [SOLUTION](#) * [zukersubopt_par](#) (const char *string, [vrna_param_t](#) *parameters)
Compute Zuker type suboptimal structures.
- [vrna_subopt_solution_t](#) * [vrna_subopt_zuker](#) ([vrna_fold_compound_t](#) *fc)
Compute Zuker type suboptimal structures.

16.23.2 Function Documentation

16.23.2.1 [zukersubopt\(\)](#)

```
SOLUTION * zukersubopt (
    const char * string )
#include <ViennaRNA/subopt.h>
```

Compute Zuker type suboptimal structures.

Compute Suboptimal structures according to M. Zuker, i.e. for every possible base pair the minimum energy structure containing the resp. base pair. Returns a list of these structures and their energies.

Deprecated use [vrna_zukersubopt\(\)](#) instead

Parameters

<i>string</i>	RNA sequence
---------------	--------------

Returns

List of zuker suboptimal structures

16.23.2.2 [zukersubopt_par\(\)](#)

```
SOLUTION * zukersubopt\_par (
    const char * string,
    vrna\_param\_t * parameters )
```



```
#include <ViennaRNA/subopt.h>
Compute Zuker type suboptimal structures.
```

Deprecated use `vrna_zukersubopt()` instead

16.23.2.3 vrna_subopt_zuker()

```
vrna_subopt_solution_t * vrna_subopt_zuker (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/subopt_zuker.h>
```

Compute Zuker type suboptimal structures.

Compute Suboptimal structures according to M. Zuker [35], i.e. for every possible base pair the minimum energy structure containing the resp. base pair. Returns a list of these structures and their energies.

Note

This function internally uses the cofold implementation to compute the suboptimal structures. For that purpose, the function doubles the sequence and enlarges the DP matrices, which in fact will grow by a factor of 4 during the computation! At the end of the structure prediction, everything will be re-set to its original requirements, i.e. normal sequence, normal (empty) DP matrices.

Bug Due to resizing, any pre-existing constraints will be lost!

See also

[vrna_subopt\(\)](#), [zukersubopt\(\)](#), [zukersubopt_par\(\)](#)

Parameters

vc	fold compound
----	---------------

Returns

List of zuker suboptimal structures

SWIG Wrapper Notes This function is attached as method **subopt_zuker()** to objects of type *fold_compound*

16.24 Suboptimal Structures within an Energy Band around the MFE

16.24.1 Detailed Description

Collaboration diagram for Suboptimal Structures within an Energy Band around the MFE:

Typedefs

- typedef void(* [vrna_subopt_result_f](#)) (const char *structure, float energy, void *data)
Callback for [vrna_subopt_cb\(\)](#)

Functions

- [vrna_subopt_solution_t](#) * [vrna_subopt](#) ([vrna_fold_compound_t](#) *fc, int delta, int sorted, FILE *fp)
Returns list of subopt structures or writes to fp.
- void [vrna_subopt_cb](#) ([vrna_fold_compound_t](#) *fc, int delta, [vrna_subopt_result_f](#) cb, void *data)
Generate suboptimal structures within an energy band around the MFE.
- [SOLUTION](#) * [subopt](#) (char *seq, char *structure, int delta, FILE *fp)

Returns list of subopt structures or writes to fp.

- **SOLUTION** * [subopt_par](#) (char *seq, char *structure, [vrna_param_t](#) *parameters, int delta, int is_↔ constrained, int is_circular, FILE *fp)

Returns list of subopt structures or writes to fp.

- **SOLUTION** * [subopt_circ](#) (char *seq, char *sequence, int delta, FILE *fp)

Returns list of circular subopt structures or writes to fp.

Variables

- double [print_energy](#)
printing threshold for use with logML
- int [subopt_sorted](#)
Sort output by energy.

16.24.2 Typedef Documentation

16.24.2.1 vrna_subopt_result_f

```
typedef void(* vrna_subopt_result_f) (const char *structure, float energy, void *data)
#include <ViennaRNA/subopt.h>
Callback for vrna\_subopt\_cb\(\)
```

Notes on Callback Functions This function will be called for each suboptimal secondary structure that is successfully backtraced.

See also

[vrna_subopt_cb\(\)](#)

Parameters

<i>structure</i>	The suboptimal secondary structure in dot-bracket notation
<i>energy</i>	The free energy of the secondary structure in kcal/mol
<i>data</i>	Some arbitrary, auxiliary data address as passed to vrna_subopt_cb()

16.24.3 Function Documentation

16.24.3.1 vrna_subopt()

```
vrna_subopt_solution_t * vrna_subopt (
    vrna_fold_compound_t * vc,
    int delta,
    int sorted,
    FILE * fp )
```

```
#include <ViennaRNA/subopt.h>
```

Returns list of subopt structures or writes to fp.

This function produces **all** suboptimal secondary structures within 'delta' * 0.01 kcal/mol of the optimum, see [33]. The results are either directly written to a 'fp' (if 'fp' is not NULL), or (fp==NULL) returned in a [vrna_subopt_solution_t](#) * list terminated by an entry where the 'structure' member is NULL.

Note

This function requires all multibranch loop DP matrices for unique multibranch loop backtracing. Therefore, the supplied `vrna_fold_compound_t` `vc` (argument 1) must be initialized with `vrna_md_t.uniq_ML = 1`, for instance like this:

```
vrna_md_t md;
vrna_md_set_default(&md);
md.uniq_ML = 1;

vrna_fold_compound_t *vc=vrna_fold_compound("GGGGGGAAAAACCCCC", &md, VRNA_OPTION_DEFAULT);
```

See also

`vrna_subopt_cb()`, `vrna_subopt_zuker()`

Parameters

<i>fc</i>	
<i>delta</i>	
<i>sorted</i>	Sort results by energy in ascending order
<i>fp</i>	

Returns

SWIG Wrapper Notes This function is attached as method `subopt()` to objects of type `fold_compound`

16.24.3.2 vrna_subopt_cb()

```
void vrna_subopt_cb (
    vrna_fold_compound_t * vc,
    int delta,
    vrna_subopt_result_f cb,
    void * data )
```

```
#include <ViennaRNA/subopt.h>
```

Generate suboptimal structures within an energy band around the MFE.

This is the most generic implementation of the suboptimal structure generator according to Wuchty et al. 1999 [33]. Identical to `vrna_subopt()`, it computes all secondary structures within an energy band `delta` around the MFE. However, this function does not print the resulting structures and their corresponding free energies to a file pointer, or returns them as a list. Instead, it calls a user-provided callback function which it passes the structure in dot-bracket format, the corresponding free energy in kcal/mol, and a user-provided data structure each time a structure was backtracked successfully. This function indicates the final output, i.e. the end of the backtracking procedure by passing NULL instead of an actual dot-bracket string to the callback.

Note

This function requires all multibranch loop DP matrices for unique multibranch loop backtracing. Therefore, the supplied `vrna_fold_compound_t` `vc` (argument 1) must be initialized with `vrna_md_t.uniq_ML = 1`, for instance like this:

```
vrna_md_t md;
vrna_md_set_default(&md);
md.uniq_ML = 1;

vrna_fold_compound_t *vc=vrna_fold_compound("GGGGGGAAAAACCCCC", &md, VRNA_OPTION_DEFAULT);
```

See also

`vrna_subopt_result_f`, `vrna_subopt()`, `vrna_subopt_zuker()`

Parameters

<i>fc</i>	fold compound with the sequence data
<i>delta</i>	Energy band around the MFE in 10cal/mol, i.e. deka-calories
<i>cb</i>	Pointer to a callback function that handles the backtracked structure and its free energy in kcal/mol
<i>data</i>	Pointer to some data structure that is passed along to the callback

SWIG Wrapper Notes This function is attached as method **subopt_cb()** to objects of type *fold_compound*

16.24.3.3 subopt()

```
SOLUTION * subopt (
    char * seq,
    char * structure,
    int delta,
    FILE * fp )
```

```
#include <ViennaRNA/subopt.h>
```

Returns list of subopt structures or writes to fp.

This function produces **all** suboptimal secondary structures within 'delta' * 0.01 kcal/mol of the optimum. The results are either directly written to a 'fp' (if 'fp' is not NULL), or (fp==NULL) returned in a **SOLUTION** * list terminated by an entry where the 'structure' pointer is NULL.

Parameters

<i>seq</i>	
<i>structure</i>	
<i>delta</i>	
<i>fp</i>	

Returns

16.24.3.4 subopt_par()

```
SOLUTION * subopt_par (
    char * seq,
    char * structure,
    vrna_param_t * parameters,
    int delta,
    int is_constrained,
    int is_circular,
    FILE * fp )
```

```
#include <ViennaRNA/subopt.h>
```

Returns list of subopt structures or writes to fp.

16.24.3.5 subopt_circ()

```
SOLUTION * subopt_circ (
    char * seq,
    char * sequence,
```

```

        int delta,
        FILE * fp )
#include <ViennaRNA/subopt.h>

```

Returns list of circular subopt structures or writes to fp.

This function is similar to [subopt\(\)](#) but calculates secondary structures assuming the RNA sequence to be circular instead of linear

Parameters

<i>seq</i>	
<i>sequence</i>	
<i>delta</i>	
<i>fp</i>	

Returns

16.24.4 Variable Documentation

16.24.4.1 `print_energy`

```

double print_energy [extern]
#include <ViennaRNA/subopt.h>
printing threshold for use with logML

```

16.24.4.2 `subopt_sorted`

```

int subopt_sorted [extern]
#include <ViennaRNA/subopt.h>
Sort output by energy.

```

16.25 Random Structure Samples from the Ensemble

Functions to draw random structure samples from the ensemble according to their equilibrium probability.

16.25.1 Detailed Description

Functions to draw random structure samples from the ensemble according to their equilibrium probability.
Collaboration diagram for Random Structure Samples from the Ensemble:

Modules

- [Stochastic Backtracking of Structures from Distance Based Partitioning](#)
Contains functions related to stochastic backtracking from a specified distance class.
- [Deprecated Interface for Stochastic Backtracking](#)

Macros

- `#define VRNA_PBACKTRACK_DEFAULT 0`
Boltzmann sampling flag indicating default backtracing mode.
- `#define VRNA_PBACKTRACK_NON_REDUNDANT 1`
Boltzmann sampling flag indicating non-redundant backtracing mode.

Typedefs

- typedef void(* [vrna_bs_result_f](#)) (const char *structure, void *data)
Callback for Boltzmann sampling.
- typedef struct vrna_pbacktrack_memory_s * [vrna_pbacktrack_mem_t](#)
Boltzmann sampling memory data structure.

Functions

- char * [vrna_pbacktrack5](#) ([vrna_fold_compound_t](#) *fc, unsigned int length)
Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.
- char ** [vrna_pbacktrack5_num](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int length, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- unsigned int [vrna_pbacktrack5_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int length, [vrna_bs_result_f](#) cb, void *data, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- char ** [vrna_pbacktrack5_resume](#) ([vrna_fold_compound_t](#) *vc, unsigned int num_samples, unsigned int length, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- unsigned int [vrna_pbacktrack5_resume_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int length, [vrna_bs_result_f](#) cb, void *data, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- char * [vrna_pbacktrack](#) ([vrna_fold_compound_t](#) *fc)
Sample a secondary structure from the Boltzmann ensemble according its probability.
- char ** [vrna_pbacktrack_num](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int options)
Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.
- unsigned int [vrna_pbacktrack_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, [vrna_bs_result_f](#) cb, void *data, unsigned int options)
Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.
- char ** [vrna_pbacktrack_resume](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)
Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.
- unsigned int [vrna_pbacktrack_resume_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, [vrna_bs_result_f](#) cb, void *data, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)
Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.
- char * [vrna_pbacktrack_sub](#) ([vrna_fold_compound_t](#) *fc, unsigned int start, unsigned int end)
Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.
- char ** [vrna_pbacktrack_sub_num](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int start, unsigned int end, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- unsigned int [vrna_pbacktrack_sub_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int start, unsigned int end, [vrna_bs_result_f](#) cb, void *data, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- char ** [vrna_pbacktrack_sub_resume](#) ([vrna_fold_compound_t](#) *vc, unsigned int num_samples, unsigned int start, unsigned int end, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

- unsigned int [vrna_pbacktrack_sub_resume_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int start, unsigned int end, [vrna_bs_result_f](#) cb, void *data, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

- void [vrna_pbacktrack_mem_free](#) ([vrna_pbacktrack_mem_t](#) s)

Release memory occupied by a Boltzmann sampling memory data structure.

16.25.2 Macro Definition Documentation

16.25.2.1 VRNA_PBACKTRACK_DEFAULT

```
#define VRNA_PBACKTRACK_DEFAULT 0
#include <ViennaRNA/boltzmann_sampling.h>
Boltzmann sampling flag indicating default backtracing mode.
```

See also

[vrna_pbacktrack5_num\(\)](#), [vrna_pbacktrack5_cb\(\)](#), [vrna_pbacktrack5_resume\(\)](#), [vrna_pbacktrack5_resume_cb\(\)](#), [vrna_pbacktrack_num\(\)](#), [vrna_pbacktrack_cb\(\)](#), [vrna_pbacktrack_resume\(\)](#), [vrna_pbacktrack_resume_cb\(\)](#)

16.25.2.2 VRNA_PBACKTRACK_NON_REDUNDANT

```
#define VRNA_PBACKTRACK_NON_REDUNDANT 1
#include <ViennaRNA/boltzmann_sampling.h>
Boltzmann sampling flag indicating non-redundant backtracing mode.
This flag will turn the Boltzmann sampling into non-redundant backtracing mode along the lines of Michalik et al. 2017 [24]
```

See also

[vrna_pbacktrack5_num\(\)](#), [vrna_pbacktrack5_cb\(\)](#), [vrna_pbacktrack5_resume\(\)](#), [vrna_pbacktrack5_resume_cb\(\)](#), [vrna_pbacktrack_num\(\)](#), [vrna_pbacktrack_cb\(\)](#), [vrna_pbacktrack_resume\(\)](#), [vrna_pbacktrack_resume_cb\(\)](#)

16.25.3 Typedef Documentation

16.25.3.1 vrna_bs_result_f

```
typedef void(* vrna_bs_result_f) (const char *structure, void *data)
#include <ViennaRNA/boltzmann_sampling.h>
Callback for Boltzmann sampling.
```

Notes on Callback Functions This function will be called for each secondary structure that has been successfully backtraced from the partition function DP matrices.

See also

[vrna_pbacktrack5_cb\(\)](#), [vrna_pbacktrack_cb\(\)](#), [vrna_pbacktrack5_resume_cb\(\)](#), [vrna_pbacktrack_resume_cb\(\)](#)

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
<i>data</i>	Some arbitrary, auxiliary data address as provided to the calling function

16.25.3.2 vrna_pbacktrack_mem_t

```
typedef struct vrna_pbacktrack_memory_s* vrna_pbacktrack_mem_t
#include <ViennaRNA/boltzmann_sampling.h>
```

Boltzmann sampling memory data structure.

This structure is required for properly resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking.

Initialize with `NULL` and pass its address to the corresponding functions `vrna_pbacktrack5_resume()`, etc.

Note

Do not forget to release memory occupied by this data structure before losing its context! Use `vrna_pbacktrack_mem_free()`.

See also

`vrna_pbacktrack5_resume()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_mem_free()`

16.25.4 Function Documentation

16.25.4.1 vrna_pbacktrack5()

```
char * vrna_pbacktrack5 (
    vrna_fold_compound_t * fc,
    unsigned int length )
#include <ViennaRNA/boltzmann_sampling.h>
```

Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a secondary structure.

The parameter `length` specifies the length of the substructure starting from the 5' end.

The structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

See also

`vrna_pbacktrack5_num()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack()`

Parameters

<code>fc</code>	The fold compound data structure
<code>length</code>	The length of the subsequence to consider (starting with 5' end)

Returns

A sampled secondary structure in dot-bracket notation (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.2 vrna_pbacktrack5_num()

```
char ** vrna_pbacktrack5_num (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according to their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_num()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>length</code>	The length of the subsequence to consider (starting with 5' end)
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.3 vrna_pbacktrack5_cb()

```
unsigned int vrna_pbacktrack5_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    vrna_bs_result_f cb,
    void * data,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack5()` and `vrna_pbacktrack5_num()` this function yields the structure samples through a callback mechanism.

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5()`, `vrna_pbacktrack5_num()`, `vrna_pbacktrack_cb()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

Parameters

<i>fc</i>	The fold compound data structure
<i>num_samples</i>	The size of the sample set, i.e. number of structures
<i>length</i>	The length of the subsequence to consider (starting with 5' end)
<i>cb</i>	The callback that receives the sampled structure
<i>data</i>	A data structure passed through to the callback <i>cb</i>
<i>options</i>	A bitwise OR-flag indicating the backtracing mode.

Returns

The number of structures actually backtraced

SWIG Wrapper Notes This function is attached as overloaded method [pbacktrack5\(\)](#) to objects of type *fold_compound* where the last argument *options* is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.4 *vrna_pbacktrack5_resume()*

```
char ** vrna_pbacktrack5_resume (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according to their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of *num_samples* secondary structures. The parameter *length* specifies the length of the substructure starting from the 5' end.

Any structure *s* with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant *k* and thermodynamic temperature *T*.

Using the *options* flag one can switch between regular ([VRNA_PBACKTRACK_DEFAULT](#)) backtracing mode, and non-redundant sampling ([VRNA_PBACKTRACK_NON_REDUNDANT](#)) along the lines of Michalik et al. 2017 [24].

In contrast to [vrna_pbacktrack5_cb\(\)](#) this function allows for resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking. For that purpose, the user passes the address of a Boltzmann sampling data structure ([vrna_pbacktrack_mem_t](#)) which will be re-used in each round of sampling, i.e. each successive call to [vrna_pbacktrack5_resume_cb\(\)](#) or [vrna_pbacktrack5_resume\(\)](#).

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;

// sample the first 100 structures
vrna_pbacktrack5_resume(fc,
    100,
    fc->length,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack5_resume(fc,
    500,
    fc->length,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>length</code>	The length of the subsequence to consider (starting with 5' end)
<code>nr_mem</code>	The address of the Boltzmann sampling memory data structure
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack5()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.5 vrna_pbacktrack5_resume_cb()

```
unsigned int vrna_pbacktrack5_resume_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int length,
    vrna_bs_result_f cb,
    void * data,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
```

```
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T . Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracking mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack5_resume()` this function yields the structure samples through a callback mechanism. A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;

// sample the first 100 structures
vrna_pbacktrack5_resume_cb(fc,
    100,
    fc->length,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack5_resume_cb(fc,
    500,
    fc->length,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5_resume()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>length</code>	The length of the subsequence to consider (starting with 5' end)
<code>cb</code>	The callback that receives the sampled structure
<code>data</code>	A data structure passed through to the callback <code>cb</code>
<code>nr_mem</code>	The address of the Boltzmann sampling memory data structure
<code>options</code>	A bitwise OR-flag indicating the backtracking mode.

Returns

The number of structures actually backtraced

SWIG Wrapper Notes This function is attached as overloaded method [pbacktrack5\(\)](#) to objects of type *fold_compound*. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.6 vrna_pbacktrack()

```
char * vrna_pbacktrack (
    vrna_fold_compound_t * fc )
#include <ViennaRNA/boltzmann_sampling.h>
```

Sample a secondary structure from the Boltzmann ensemble according its probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a secondary structure.

The structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with [vrna_fold_compound\(\)](#) or similar. This can be done easily by passing [vrna_fold_compound\(\)](#) a model details parameter with `vrna_md_t.uniq_ML = 1`.

[vrna_pf\(\)](#) has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

See also

[vrna_pbacktrack5\(\)](#), [vrna_pbacktrack_num](#), [vrna_pbacktrack_cb\(\)](#)

Parameters

<code>fc</code>	The fold compound data structure
-----------------	----------------------------------

Returns

A sampled secondary structure in dot-bracket notation (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method [pbacktrack\(\)](#) to objects of type *fold_compound*. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.7 vrna_pbacktrack_num()

```
char ** vrna_pbacktrack_num (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack()`, `vrna_pbacktrack_cb()`, `vrna_pbacktrack5_num()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.8 vrna_pbacktrack_cb()

```
unsigned int vrna_pbacktrack_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    vrna_bs_result_f cb,
```

```

    void * data,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>

```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack()` and `vrna_pbacktrack_num()` this function yields the structure samples through a callback mechanism.

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack()`, `vrna_pbacktrack_num()`, `vrna_pbacktrack5_cb()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>cb</code>	The callback that receives the sampled structure
<code>data</code>	A data structure passed through to the callback <code>cb</code>
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

The number of structures actually backtraced

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.9 `vrna_pbacktrack_resume()`

```
char ** vrna_pbacktrack_resume (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples from the Boltzmann ensemble according to their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack_cb()` this function allows for resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking. For that purpose, the user passes the address of a Boltzmann sampling data structure (`vrna_pbacktrack_mem_t`) which will be re-used in each round of sampling, i.e. each successive call to `vrna_pbacktrack_resume_cb()` or `vrna_pbacktrack_resume()`.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;

// sample the first 100 structures
vrna_pbacktrack_resume(fc,
    100,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack_resume(fc,
    500,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_cb()`, `vrna_pbacktrack5_resume()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

Parameters

<i>fc</i>	The fold compound data structure
<i>num_samples</i>	The size of the sample set, i.e. number of structures
<i>nr_mem</i>	The address of the Boltzmann sampling memory data structure
<i>options</i>	A bitwise OR-flag indicating the backtracing mode.

Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the list of structures, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.10 `vrna_pbacktrack_resume_cb()`

```
unsigned int vrna_pbacktrack_resume_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    vrna_bs_result_f cb,
    void * data,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
```

```
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack5_resume()` this function yields the structure samples through a callback mechanism.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;

// sample the first 100 structures
vrna_pbacktrack5_resume_cb(fc,
    100,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack5_resume_cb(fc,
    500,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack_resume()`, `vrna_pbacktrack_cb()`, `vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>cb</code>	The callback that receives the sampled structure
<code>data</code>	A data structure passed through to the callback <code>cb</code>
<code>nr_mem</code>	The address of the Boltzmann sampling memory data structure
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

The number of structures actually backtraced

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.11 vrna_pbacktrack_sub()

```
char * vrna_pbacktrack_sub (
    vrna_fold_compound_t * fc,
    unsigned int start,
    unsigned int end )
```

```
#include <ViennaRNA/boltzmann_sampling.h>
```

Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a secondary structure. The parameters `start` and `end` specify the interval $[start : end]$ of the subsequence with $1 \leq start < end \leq n$ for sequence length n , the structure $s_{start,end}$ should be drawn from.

The resulting substructure $s_{start,end}$ with free energy $E(s_{start,end})$ is picked from the Boltzmann distributed sub ensemble of all structures within the interval $[start : end]$ according to its probability

$$p(s_{start,end}) = \frac{\exp(-E(s_{start,end})/kT)}{Z_{start,end}}$$

with partition function $Z_{start,end} = \sum_{s_{start,end}} \exp(-E(s_{start,end})/kT)$, Boltzmann constant k and thermodynamic temperature T .

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

See also

`vrna_pbacktrack_sub_num()`, `vrna_pbacktrack_sub_cb()`, `vrna_pbacktrack()`

Parameters

<code>fc</code>	The fold compound data structure
<code>start</code>	The start of the subsequence to consider, i.e. 5'-end position(1-based)
<code>end</code>	The end of the subsequence to consider, i.e. 3'-end position (1-based)

Returns

A sampled secondary structure in dot-bracket notation (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack_sub()` to objects of type `fold_compound`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.12 vrna_pbacktrack_sub_num()

```
char ** vrna_pbacktrack_sub_num (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end. Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack_sub()`, `vrna_pbacktrack_sub_cb()`, `vrna_pbacktrack_num()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>start</code>	The start of the subsequence to consider, i.e. 5'-end position (1-based)
<code>end</code>	The end of the subsequence to consider, i.e. 3'-end position (1-based)
<code>options</code>	A bitwise OR-flag indicating the backtracking mode.

Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack_sub()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.13 vrna_pbacktrack_sub_cb()

```
unsigned int vrna_pbacktrack_sub_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    vrna_bs_result_f cb,
    void * data,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according to their probability.

Perform a probabilistic (stochastic) backtracking in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end.

Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T . Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack5()` and `vrna_pbacktrack5_num()` this function yields the structure samples through a callback mechanism.

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5()`, `vrna_pbacktrack5_num()`, `vrna_pbacktrack_cb()`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>start</code>	The start of the subsequence to consider, i.e. 5'-end position (1-based)
<code>end</code>	The end of the subsequence to consider, i.e. 3'-end position (1-based)
<code>cb</code>	The callback that receives the sampled structure
<code>data</code>	A data structure passed through to the callback <code>cb</code>
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

The number of structures actually backtraced

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound` where the last argument `options` is optional with default value `options = VRNA_PBACKTRACK_DEFAULT`. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.14 `vrna_pbacktrack_sub_resume()`

```
char ** vrna_pbacktrack_sub_resume (
```

```
vrna_fold_compound_t * fc,
unsigned int num_samples,
unsigned int start,
unsigned int end,
vrna_pbacktrack_mem_t * nr_mem,
unsigned int options )
```

```
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of `num_samples` secondary structures. The parameter `length` specifies the length of the substructure starting from the 5' end. Any structure s with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant k and thermodynamic temperature T .

Using the `options` flag one can switch between regular (`VRNA_PBACKTRACK_DEFAULT`) backtracing mode, and non-redundant sampling (`VRNA_PBACKTRACK_NON_REDUNDANT`) along the lines of Michalik et al. 2017 [24].

In contrast to `vrna_pbacktrack5_cb()` this function allows for resuming a previous sampling round in specialized Boltzmann sampling, such as non-redundant backtracking. For that purpose, the user passes the address of a Boltzmann sampling data structure (`vrna_pbacktrack_mem_t`) which will be re-used in each round of sampling, i.e. each successive call to `vrna_pbacktrack5_resume_cb()` or `vrna_pbacktrack5_resume()`.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;

// sample the first 100 structures
vrna_pbacktrack5_resume(fc,
    100,
    fc->length,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack5_resume(fc,
    500,
    fc->length,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```

Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

Parameters

<i>fc</i>	The fold compound data structure
<i>num_samples</i>	The size of the sample set, i.e. number of structures
<i>start</i>	The start of the subsequence to consider, i.e. 5'-end position(1-based)
<i>end</i>	The end of the subsequence to consider, i.e. 3'-end position (1-based)
<i>nr_mem</i>	The address of the Boltzmann sampling memory data structure
<i>options</i>	A bitwise OR-flag indicating the backtracing mode.

Returns

A set of secondary structure samples in dot-bracket notation terminated by NULL (or NULL on error)

SWIG Wrapper Notes This function is attached as overloaded method **pbacktrack()** to objects of type *fold_compound*. In addition to the list of structures, this function also returns the *nr_mem* data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.15 vrna_pbacktrack_sub_resume_cb()

```
unsigned int vrna_pbacktrack_sub_resume_cb (
    vrna_fold_compound_t * fc,
    unsigned int num_samples,
    unsigned int start,
    unsigned int end,
    vrna_bs_result_f cb,
    void * data,
    vrna_pbacktrack_mem_t * nr_mem,
    unsigned int options )
#include <ViennaRNA/boltzmann_sampling.h>
```

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

Perform a probabilistic (stochastic) backtracing in the partition function DP arrays to obtain a set of *num_samples* secondary structures. The parameter *length* specifies the length of the substructure starting from the 5' end. Any structure *s* with free energy $E(s)$ is picked from the Boltzmann distributed ensemble according to its probability

$$p(s) = \frac{\exp(-E(s)/kT)}{Z}$$

with partition function $Z = \sum_s \exp(-E(s)/kT)$, Boltzmann constant *k* and thermodynamic temperature *T*.

Using the *options* flag one can switch between regular (**VRNA_PBACKTRACK_DEFAULT**) backtracing mode, and non-redundant sampling (**VRNA_PBACKTRACK_NON_REDUNDANT**) along the lines of Michalik et al. 2017 [24].

In contrast to **vrna_pbacktrack5_resume()** this function yields the structure samples through a callback mechanism.

A successive sample call to this function may look like:

```
vrna_pbacktrack_mem_t nonredundant_memory = NULL;

// sample the first 100 structures
vrna_pbacktrack5_resume_cb(fc,
    100,
    fc->length,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// sample another 500 structures
vrna_pbacktrack5_resume_cb(fc,
    500,
    fc->length,
    &callback_function,
    (void *)&callback_data,
    &nonredundant_memory,
    options);

// release memory occupied by the non-redundant memory data structure
vrna_pbacktrack_mem_free(nonredundant_memory);
```


Precondition

Unique multiloop decomposition has to be active upon creation of `fc` with `vrna_fold_compound()` or similar. This can be done easily by passing `vrna_fold_compound()` a model details parameter with `vrna_md_t.uniq_ML = 1`.

`vrna_pf()` has to be called first to fill the partition function matrices

Note

This function is polymorphic. It accepts `vrna_fold_compound_t` of type `VRNA_FC_TYPE_SINGLE`, and `VRNA_FC_TYPE_COMPARATIVE`.

Warning

In non-redundant sampling mode (`VRNA_PBACKTRACK_NON_REDUNDANT`), this function may not yield the full number of requested samples. This may happen if a) the number of requested structures is larger than the total number of structures in the ensemble, b) numeric instabilities prevent the backtracking function to enumerate structures with high free energies, or c) any other error occurs.

See also

`vrna_pbacktrack5_resume()`, `vrna_pbacktrack5_cb()`, `vrna_pbacktrack_resume_cb()`, `vrna_pbacktrack_mem_t`, `VRNA_PBACKTRACK_DEFAULT`, `VRNA_PBACKTRACK_NON_REDUNDANT`, `vrna_pbacktrack_mem_free`

Parameters

<code>fc</code>	The fold compound data structure
<code>num_samples</code>	The size of the sample set, i.e. number of structures
<code>start</code>	The start of the subsequence to consider, i.e. 5'-end position (1-based)
<code>end</code>	The end of the subsequence to consider, i.e. 3'-end position (1-based)
<code>cb</code>	The callback that receives the sampled structure
<code>data</code>	A data structure passed through to the callback <code>cb</code>
<code>nr_mem</code>	The address of the Boltzmann sampling memory data structure
<code>options</code>	A bitwise OR-flag indicating the backtracing mode.

Returns

The number of structures actually backtraced

SWIG Wrapper Notes This function is attached as overloaded method `pbacktrack()` to objects of type `fold_compound`. In addition to the number of structures backtraced, this function also returns the `nr_mem` data structure as first element. See also [Python Examples - Boltzmann Sampling](#)

16.25.4.16 vrna_pbacktrack_mem_free()

```
void vrna_pbacktrack_mem_free (
    vrna_pbacktrack_mem_t s )
#include <ViennaRNA/boltzmann_sampling.h>
```

Release memory occupied by a Boltzmann sampling memory data structure.

See also

`vrna_pbacktrack_mem_t`, `vrna_pbacktrack5_resume()`, `vrna_pbacktrack5_resume_cb()`, `vrna_pbacktrack_resume()`, `vrna_pbacktrack_resume_cb()`

Parameters

s	The non-redundancy memory data structure
---	--

16.26 Compute the Structure with Maximum Expected Accuracy (MEA)

16.26.1 Detailed Description

Collaboration diagram for Compute the Structure with Maximum Expected Accuracy (MEA):

Functions

- char * [vrna_MEA](#) ([vrna_fold_compound_t](#) *fc, double gamma, float *mea)
Compute a MEA (maximum expected accuracy) structure.
- char * [vrna_MEA_from_plist](#) ([vrna_ep_t](#) *plist, const char *sequence, double gamma, [vrna_md_t](#) *md, float *mea)
Compute a MEA (maximum expected accuracy) structure from a list of probabilities.
- float [MEA](#) ([plist](#) *p, char *structure, double gamma)
Computes a MEA (maximum expected accuracy) structure.

16.26.2 Function Documentation

16.26.2.1 vrna_MEA()

```
char * vrna_MEA (
    vrna\_fold\_compound\_t * fc,
    double gamma,
    float * mea )
#include <ViennaRNA/MEA.h>
Compute a MEA (maximum expected accuracy) structure.
The algorithm maximizes the expected accuracy
```

$$A(S) = \sum_{(i,j) \in S} 2\gamma p_{ij} + \sum_{i \notin S} p_i^u$$

Higher values of γ result in more base pairs of lower probability and thus higher sensitivity. Low values of γ result in structures containing only highly likely pairs (high specificity). The code of the MEA function also demonstrates the use of sparse dynamic programming scheme to reduce the time and memory complexity of folding.

Precondition

[vrna_pf\(\)](#) must be executed on input parameter `fc`

Parameters

<i>fc</i>	The fold compound data structure with pre-filled base pair probability matrix
<i>gamma</i>	The weighting factor for base pairs vs. unpaired nucleotides
<i>mea</i>	A pointer to a variable where the MEA value will be written to

Returns

An MEA structure (or NULL on any error)

SWIG Wrapper Notes This function is attached as overloaded method **MEA**(gamma = 1.) to objects of type *fold_compound*. Note, that it returns the MEA structure and MEA value as a tuple (MEA_structure, MEA)

16.26.2.2 vrna_MEA_from_plist()

```
char * vrna_MEA_from_plist (
    vrna_ep_t * plist,
    const char * sequence,
    double gamma,
    vrna_md_t * md,
    float * mea )
#include <ViennaRNA/MEA.h>
```

Compute a MEA (maximum expected accuracy) structure from a list of probabilities. The algorithm maximizes the expected accuracy

$$A(S) = \sum_{(i,j) \in S} 2\gamma p_{ij} + \sum_{i \notin S} p_i^u$$

Higher values of γ result in more base pairs of lower probability and thus higher sensitivity. Low values of γ result in structures containing only highly likely pairs (high specificity). The code of the MEA function also demonstrates the use of sparse dynamic programming scheme to reduce the time and memory complexity of folding.

Note

The unpaired probabilities $p_i^u = 1 - \sum_{j \neq i} p_{ij}$ are usually computed from the supplied pairing probabilities p_{ij} as stored in *plist* entries of type **VRNA_PLIST_TYPE_BASEPAIR**. To overwrite individual p_o^u values simply add entries with type **VRNA_PLIST_TYPE_UNPAIRED**

To include G-Quadruplex support, the corresponding field in *md* must be set.

Parameters

<i>plist</i>	A list of base pair probabilities the MEA structure is computed from
<i>sequence</i>	The RNA sequence that corresponds to the list of probability values
<i>gamma</i>	The weighting factor for base pairs vs. unpaired nucleotides
<i>md</i>	A model details data structure (maybe NULL)
<i>mea</i>	A pointer to a variable where the MEA value will be written to

Returns

An MEA structure (or NULL on any error)

SWIG Wrapper Notes This function is available as overloaded function **MEA_from_plist**(gamma = 1., md = NULL). Note, that it returns the MEA structure and MEA value as a tuple (MEA_structure, MEA)

16.26.2.3 MEA()

```
float MEA (
    plist * p,
    char * structure,
    double gamma )
```

```
#include <ViennaRNA/MEA.h>
```

Computes a MEA (maximum expected accuracy) structure.

The algorithm maximizes the expected accuracy

$$A(S) = \sum_{(i,j) \in S} 2\gamma p_{ij} + \sum_{i \notin S} p_i^u$$

Higher values of γ result in more base pairs of lower probability and thus higher sensitivity. Low values of γ result in structures containing only highly likely pairs (high specificity). The code of the MEA function also demonstrates the use of sparse dynamic programming scheme to reduce the time and memory complexity of folding.

Deprecated Use `vrna_MEA()` or `vrna_MEA_from_plist()` instead!

16.27 Compute the Centroid Structure

16.27.1 Detailed Description

Collaboration diagram for Compute the Centroid Structure:

Functions

- char * `vrna_centroid` (`vrna_fold_compound_t` *vc, double *dist)
Get the centroid structure of the ensemble.
- char * `vrna_centroid_from_plist` (int length, double *dist, `vrna_ep_t` *pl)
Get the centroid structure of the ensemble.
- char * `vrna_centroid_from_probs` (int length, double *dist, `FLT_OR_DBL` *probs)
Get the centroid structure of the ensemble.

16.27.2 Function Documentation

16.27.2.1 `vrna_centroid()`

```
char * vrna_centroid (
    vrna_fold_compound_t * vc,
    double * dist )
```

```
#include <ViennaRNA/centroid.h>
```

Get the centroid structure of the ensemble.

The centroid is the structure with the minimal average distance to all other structures

$$< d(S) > = \sum_{(i,j) \in S} (1 - p_{ij}) + \sum_{(i,j) \notin S} p_{ij}$$

Thus, the centroid is simply the structure containing all pairs with $p_{ij} > 0.5$. The distance of the centroid to the ensemble is written to the memory addressed by *dist*.

Parameters

in	vc	The fold compound data structure
out	dist	A pointer to the distance variable where the centroid distance will be written to

Returns

The centroid structure of the ensemble in dot-bracket notation (NULL on error)

16.27.2.2 `vrna_centroid_from_plist()`

```
char * vrna_centroid_from_plist (
```

```

    int length,
    double * dist,
    vrna_ep_t * pl )
#include <ViennaRNA/centroid.h>

```

Get the centroid structure of the ensemble.

This function is a threadsafe replacement for `centroid()` with a `vrna_ep_t` input

The centroid is the structure with the minimal average distance to all other structures

$$\langle d(S) \rangle = \sum_{(i,j) \in S} (1 - p_{ij}) + \sum_{(i,j) \notin S} p_{ij}$$

Thus, the centroid is simply the structure containing all pairs with $p_{ij} > 0.5$. The distance of the centroid to the ensemble is written to the memory addressed by `dist`.

Parameters

in	<i>length</i>	The length of the sequence
out	<i>dist</i>	A pointer to the distance variable where the centroid distance will be written to
in	<i>pl</i>	A pair list containing base pair probability information about the ensemble

Returns

The centroid structure of the ensemble in dot-bracket notation (NULL on error)

16.27.2.3 vrna_centroid_from_probs()

```

char * vrna_centroid_from_probs (
    int length,
    double * dist,
    FLT_OR_DBL * probs )
#include <ViennaRNA/centroid.h>

```

Get the centroid structure of the ensemble.

This function is a threadsafe replacement for `centroid()` with a probability array input

The centroid is the structure with the minimal average distance to all other structures

$$\langle d(S) \rangle = \sum_{(i,j) \in S} (1 - p_{ij}) + \sum_{(i,j) \notin S} p_{ij}$$

Thus, the centroid is simply the structure containing all pairs with $p_{ij} > 0.5$. The distance of the centroid to the ensemble is written to the memory addressed by `dist`.

Parameters

in	<i>length</i>	The length of the sequence
out	<i>dist</i>	A pointer to the distance variable where the centroid distance will be written to
in	<i>probs</i>	An upper triangular matrix containing base pair probabilities (access via <code>iindx vrna_idx_row_wise()</code>)

Returns

The centroid structure of the ensemble in dot-bracket notation (NULL on error)

16.28 RNA-RNA Interaction

16.28.1 Detailed Description

Collaboration diagram for RNA-RNA Interaction:

Modules

- [Partition Function for Two Hybridized Sequences](#)

Partition Function Cofolding.

- [Partition Function for two Hybridized Sequences as a Stepwise Process](#)

RNA-RNA interaction as a stepwise process.

Files

- file [concentrations.h](#)
Concentration computations for RNA-RNA interactions.
- file [duplex.h](#)
Functions for simple RNA-RNA duplex interactions.
- file [part_func_up.h](#)
Implementations for accessibility and RNA-RNA interaction as a stepwise process.

16.29 Classified Dynamic Programming Variants

16.29.1 Detailed Description

Collaboration diagram for Classified Dynamic Programming Variants:

Modules

- [Distance Based Partitioning of the Secondary Structure Space](#)
- [Compute the Density of States](#)

16.30 Distance Based Partitioning of the Secondary Structure Space

16.30.1 Detailed Description

Collaboration diagram for Distance Based Partitioning of the Secondary Structure Space:

Modules

- [Computing MFE representatives of a Distance Based Partitioning](#)
Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.
- [Computing Partition Functions of a Distance Based Partitioning](#)
Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.
- [Stochastic Backtracking of Structures from Distance Based Partitioning](#)
Contains functions related to stochastic backtracking from a specified distance class.

Files

- file [2Dfold.h](#)
MFE structures for base pair distance classes.
- file [2Dpfold.h](#)
Partition function implementations for base pair distance classes.

16.31 Computing MFE representatives of a Distance Based Partitioning

Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.

16.31.1 Detailed Description

Compute the minimum free energy (MFE) and secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures basepair distance to two fixed reference structures.

See also

For further details, we refer to Lorenz et al. 2009 [20]

Collaboration diagram for Computing MFE representatives of a Distance Based Partitioning:

Data Structures

- struct [vrna_sol_TwoD_t](#)
Solution element returned from [vrna_mfe_TwoD\(\)](#) [More...](#)
- struct [TwoDfold_vars](#)
Variables compound for 2Dfold MFE folding. [More...](#)

Typedefs

- typedef struct [vrna_sol_TwoD_t](#) [vrna_sol_TwoD_t](#)
Solution element returned from [vrna_mfe_TwoD\(\)](#)
- typedef struct [TwoDfold_vars](#) [TwoDfold_vars](#)
Variables compound for 2Dfold MFE folding.

Functions

- [vrna_sol_TwoD_t](#) * [vrna_mfe_TwoD](#) ([vrna_fold_compound_t](#) *vc, int distance1, int distance2)
Compute MFE's and representative for distance partitioning.
- char * [vrna_backtrack5_TwoD](#) ([vrna_fold_compound_t](#) *vc, int k, int l, unsigned int j)
Backtrack a minimum free energy structure from a 5' section of specified length.
- [TwoDfold_vars](#) * [get_TwoDfold_variables](#) (const char *seq, const char *structure1, const char *structure2, int circ)
Get a structure of type [TwoDfold_vars](#) prefilled with current global settings.
- void [destroy_TwoDfold_variables](#) ([TwoDfold_vars](#) *our_variables)
Destroy a [TwoDfold_vars](#) datastructure without memory loss.
- [TwoDfold_solution](#) * [TwoDfoldList](#) ([TwoDfold_vars](#) *vars, int distance1, int distance2)
Compute MFE's and representative for distance partitioning.
- char * [TwoDfold_backtrack_f5](#) (unsigned int j, int k, int l, [TwoDfold_vars](#) *vars)
Backtrack a minimum free energy structure from a 5' section of specified length.

16.31.2 Data Structure Documentation

16.31.2.1 struct vrna_sol_TwoD_t

Solution element returned from [vrna_mfe_TwoD\(\)](#)

This element contains free energy and structure for the appropriate kappa (k), lambda (l) neighborhood The datastructure contains two integer attributes 'k' and 'l' as well as an attribute 'en' of type float representing the free energy in kcal/mol and an attribute 's' of type char* containing the secondary structure representative, A value of INF in k denotes the end of a list

See also

[vrna_mfe_TwoD\(\)](#)

Data Fields

- int **k**
Distance to first reference.
- int **l**
Distance to second reference.
- float **en**
Free energy in kcal/mol.
- char * **s**
MFE representative structure in dot-bracket notation.

16.31.2.2 struct TwoDfold_vars

Variables compound for 2Dfold MFE folding.

Deprecated This data structure will be removed from the library soon! Use [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_mfe_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

Collaboration diagram for TwoDfold_vars:

Data Fields

- [vrna_param_t](#) * **P**
Precomputed energy parameters and model details.
- int **do_backtrack**
Flag whether to do backtracing of the structure(s) or not.
- char * **ptype**
Precomputed array of pair types.
- char * **sequence**
The input sequence
- short * **S1**
The input sequences in numeric form.
- unsigned int **maxD1**
Maximum allowed base pair distance to first reference.
- unsigned int **maxD2**
Maximum allowed base pair distance to second reference.
- unsigned int * **mm1**
Maximum matching matrix, reference struct 1 disallowed.
- unsigned int * **mm2**
Maximum matching matrix, reference struct 2 disallowed.
- int * **my_iindx**
Index for moving in quadratic distance dimensions.
- unsigned int * **referenceBPs1**
Matrix containing number of basepairs of reference structure1 in interval [i,j].
- unsigned int * **referenceBPs2**
Matrix containing number of basepairs of reference structure2 in interval [i,j].
- unsigned int * **bpdist**
Matrix containing base pair distance of reference structure 1 and 2 on interval [i,j].

16.31.3 Typedef Documentation

16.31.3.1 vrna_sol_TwoD_t

```
typedef struct vrna_sol_TwoD_t vrna_sol_TwoD_t
#include <ViennaRNA/2Dfold.h>
```

Solution element returned from [vrna_mfe_TwoD\(\)](#)

This element contains free energy and structure for the appropriate kappa (k), lambda (l) neighborhood. The datastructure contains two integer attributes 'k' and 'l' as well as an attribute 'en' of type float representing the free energy in kcal/mol and an attribute 's' of type char* containing the secondary structure representative.

A value of INF in k denotes the end of a list.

See also

[vrna_mfe_TwoD\(\)](#)

16.31.3.2 TwoDfold_vars

```
typedef struct TwoDfold_vars TwoDfold_vars
#include <ViennaRNA/2Dfold.h>
```

Variables compound for 2Dfold MFE folding.

Deprecated This data structure will be removed from the library soon! Use [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_mfe_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

16.31.4 Function Documentation

16.31.4.1 vrna_mfe_TwoD()

```
vrna_sol_TwoD_t * vrna_mfe_TwoD (
    vrna_fold_compound_t * vc,
    int distance1,
    int distance2 )
```

```
#include <ViennaRNA/2Dfold.h>
```

Compute MFE's and representative for distance partitioning.

This function computes the minimum free energies and a representative secondary structure for each distance class according to the two references specified in the datastructure 'vars'. The maximum basepair distance to each of both references may be set by the arguments 'distance1' and 'distance2', respectively. If both distance arguments are set to '-1', no restriction is assumed and the calculation is performed for each distance class possible.

The returned list contains an entry for each distance class. If a maximum basepair distance to either of the references was passed, an entry with k=l=-1 will be appended in the list, denoting the class where all structures exceeding the maximum will be thrown into. The end of the list is denoted by an attribute value of INF in the k-attribute of the list entry.

See also

[vrna_fold_compound_TwoD\(\)](#), [vrna_fold_compound_free\(\)](#), [vrna_pf_TwoD\(\)](#), [vrna_backtrack5_TwoD\(\)](#), [vrna_sol_TwoD_t](#), [vrna_fold_compound_t](#)

Parameters

<i>vc</i>	The datastructure containing all precomputed folding attributes
<i>distance1</i>	maximum distance to reference1 (-1 means no restriction)
<i>distance2</i>	maximum distance to reference2 (-1 means no restriction)

Returns

A list of minimum free energies (and corresponding structures) for each distance class

16.31.4.2 vrna_backtrack5_TwoD()

```
char * vrna_backtrack5_TwoD (
    vrna_fold_compound_t * vc,
    int k,
    int l,
    unsigned int j )
```

```
#include <ViennaRNA/2Dfold.h>
```

Backtrack a minimum free energy structure from a 5' section of specified length.

This function allows one to backtrack a secondary structure beginning at the 5' end, a specified length and residing in a specific distance class. If the argument 'k' gets a value of -1, the structure that is backtracked is assumed to reside in the distance class where all structures exceeding the maximum basepair distance specified in [vrna_mfe_TwoD\(\)](#) belong to.

Note

The argument 'vars' must contain precalculated energy values in the energy matrices, i.e. a call to [vrna_mfe_TwoD\(\)](#) preceding this function is mandatory!

See also

[vrna_mfe_TwoD\(\)](#)

Parameters

<i>vc</i>	The datastructure containing all precomputed folding attributes
<i>j</i>	The length in nucleotides beginning from the 5' end
<i>k</i>	distance to reference1 (may be -1)
<i>l</i>	distance to reference2

16.31.4.3 get_TwoDfold_variables()

```
TwoDfold_vars * get_TwoDfold_variables (
    const char * seq,
    const char * structure1,
    const char * structure2,
    int circ )
```

```
#include <ViennaRNA/2Dfold.h>
```

Get a structure of type [TwoDfold_vars](#) prefilled with current global settings.

This function returns a datastructure of type [TwoDfold_vars](#). The data fields inside the [TwoDfold_vars](#) are prefilled by global settings and all memory allocations necessary to start a computation are already done for the convenience of the user

Note

Make sure that the reference structures are compatible with the sequence according to Watson-Crick- and Wobble-base pairing

Deprecated Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_mfe_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

Parameters

<i>seq</i>	The RNA sequence
<i>structure1</i>	The first reference structure in dot-bracket notation
<i>structure2</i>	The second reference structure in dot-bracket notation
<i>circ</i>	A switch to indicate the assumption to fold a circular instead of linear RNA (0=OFF, 1=ON)

Returns

A datastructure prefilled with folding options and allocated memory

16.31.4.4 `destroy_TwoDfold_variables()`

```
void destroy_TwoDfold_variables (
    TwoDfold_vars * our_variables )
#include <ViennaRNA/2Dfold.h>
```

Destroy a `TwoDfold_vars` datastructure without memory loss.

This function free's all allocated memory that depends on the datastructure given.

Deprecated Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

Parameters

<i>our_variables</i>	A pointer to the datastructure to be destroyed
----------------------	--

16.31.4.5 `TwoDfoldList()`

```
TwoDfold_solution * TwoDfoldList (
    TwoDfold_vars * vars,
    int distance1,
    int distance2 )
#include <ViennaRNA/2Dfold.h>
```

Compute MFE's and representative for distance partitioning.

This function computes the minimum free energies and a representative secondary structure for each distance class according to the two references specified in the datastructure 'vars'. The maximum basepair distance to each of both references may be set by the arguments 'distance1' and 'distance2', respectively. If both distance arguments are set to '-1', no restriction is assumed and the calculation is performed for each distance class possible.

The returned list contains an entry for each distance class. If a maximum basepair distance to either of the references was passed, an entry with `k=-1` will be appended in the list, denoting the class where all structures exceeding the maximum will be thrown into. The end of the list is denoted by an attribute value of `INF` in the `k`-attribute of the list entry.

Deprecated Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_mfe_TwoD()`, and `vrna_fold_compound_free()` instead!

Parameters

<i>vars</i>	the datastructure containing all predefined folding attributes
<i>distance1</i>	maximum distance to reference1 (-1 means no restriction)
<i>distance2</i>	maximum distance to reference2 (-1 means no restriction)

16.31.4.6 TwoDfold_backtrack_f5()

```
char * TwoDfold_backtrack_f5 (
    unsigned int j,
    int k,
    int l,
    TwoDfold_vars * vars )
#include <ViennaRNA/2Dfold.h>
```

Backtrack a minimum free energy structure from a 5' section of specified length.

This function allows one to backtrack a secondary structure beginning at the 5' end, a specified length and residing in a specific distance class. If the argument 'k' gets a value of -1, the structure that is backtracked is assumed to reside in the distance class where all structures exceeding the maximum basepair distance specified in TwoDfold() belong to.

Note

The argument 'vars' must contain precalculated energy values in the energy matrices, i.e. a call to TwoDfold() preceding this function is mandatory!

Deprecated Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_mfe_TwoD\(\)](#), [vrna_backtrack5_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

Parameters

<i>j</i>	The length in nucleotides beginning from the 5' end
<i>k</i>	distance to reference1 (may be -1)
<i>l</i>	distance to reference2
<i>vars</i>	the datastructure containing all predefined folding attributes

16.32 Computing Partition Functions of a Distance Based Partitioning

Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.

16.32.1 Detailed Description

Compute the partition function and stochastically sample secondary structures for a partitioning of the secondary structure space according to the base pair distance to two fixed reference structures.

Collaboration diagram for Computing Partition Functions of a Distance Based Partitioning:

Data Structures

- struct [vrna_sol_TwoD_pf_t](#)
Solution element returned from [vrna_pf_TwoD\(\)](#) [More...](#)

Typedefs

- typedef struct [vrna_sol_TwoD_pf_t](#) [vrna_sol_TwoD_pf_t](#)
Solution element returned from [vrna_pf_TwoD\(\)](#)

Functions

- [vrna_sol_TwoD_pf_t](#) * [vrna_pf_TwoD](#) ([vrna_fold_compound_t](#) *vc, int maxDistance1, int maxDistance2)

Compute the partition function for all distance classes.

16.32.2 Data Structure Documentation

16.32.2.1 struct vrna_sol_TwoD_pf_t

Solution element returned from [vrna_pf_TwoD\(\)](#)

This element contains the partition function for the appropriate kappa (k), lambda (l) neighborhood. The datastructure contains two integer attributes 'k' and 'l' as well as an attribute 'q' of type [FLT_OR_DBL](#).

A value of INF in k denotes the end of a list.

See also

[vrna_pf_TwoD\(\)](#)

Data Fields

- int k
Distance to first reference.
- int l
Distance to second reference.
- [FLT_OR_DBL](#) q
partition function

16.32.3 Typedef Documentation

16.32.3.1 vrna_sol_TwoD_pf_t

```
typedef struct vrna_sol_TwoD_pf_t vrna_sol_TwoD_pf_t
```

```
#include <ViennaRNA/2Dpfold.h>
```

Solution element returned from [vrna_pf_TwoD\(\)](#)

This element contains the partition function for the appropriate kappa (k), lambda (l) neighborhood. The datastructure contains two integer attributes 'k' and 'l' as well as an attribute 'q' of type [FLT_OR_DBL](#).

A value of INF in k denotes the end of a list.

See also

[vrna_pf_TwoD\(\)](#)

16.32.4 Function Documentation

16.32.4.1 vrna_pf_TwoD()

```
vrna_sol_TwoD_pf_t * vrna_pf_TwoD (
    vrna_fold_compound_t * vc,
    int maxDistance1,
    int maxDistance2 )
```

```
#include <ViennaRNA/2Dpfold.h>
```

Compute the partition function for all distance classes.

This function computes the partition functions for all distance classes according to the two reference structures specified in the datastructure 'vars'. Similar to [vrna_mfe_TwoD\(\)](#) the arguments maxDistance1 and maxDistance2 specify the maximum distance to both reference structures. A value of '-1' in either of them makes the appropriate distance restrictionless, i.e. all basepair distances to the reference are taken into account during computation. In case there is a restriction, the returned solution contains an entry where the attribute k=l=-1 contains the partition function for all structures exceeding the restriction. A value of INF in the attribute 'k' of the returned list denotes the end of the list.

See also

[vrna_fold_compound_TwoD\(\)](#), [vrna_fold_compound_free\(\)](#), [vrna_fold_compound_vrna_sol_TwoD_pf_t](#)

Parameters

<code>vc</code>	The datastructure containing all necessary folding attributes and matrices
<code>maxDistance1</code>	The maximum basepair distance to reference1 (may be -1)
<code>maxDistance2</code>	The maximum basepair distance to reference2 (may be -1)

Returns

A list of partition funtions for the corresponding distance classes

16.33 Stochastic Backtracking of Structures from Distance Based Partitioning

Contains functions related to stochastic backtracking from a specified distance class.

16.33.1 Detailed Description

Contains functions related to stochastic backtracking from a specified distance class.

Collaboration diagram for Stochastic Backtracking of Structures from Distance Based Partitioning:

Functions

- `char * vrna_pbacktrack_TwoD (vrna_fold_compound_t *vc, int d1, int d2)`
Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability.
- `char * vrna_pbacktrack5_TwoD (vrna_fold_compound_t *vc, int d1, int d2, unsigned int length)`
Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

16.33.2 Function Documentation

16.33.2.1 vrna_pbacktrack_TwoD()

```
char * vrna_pbacktrack_TwoD (
    vrna_fold_compound_t * vc,
    int d1,
    int d2 )
```

```
#include <ViennaRNA/2Dpfold.h>
```

Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability. If the argument 'd1' is set to '-1', the structure will be backtracked in the distance class where all structures exceeding the maximum basepair distance to either of the references reside.

Precondition

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to [vrna_pf_TwoD\(\)](#) preceding this function is mandatory!

See also

[vrna_pf_TwoD\(\)](#)

Parameters

in, out	vc	The vrna_fold_compound_t datastructure containing all necessary folding attributes and matrices
in	d1	The distance to reference1 (may be -1)
in	d2	The distance to reference2

Returns

A sampled secondary structure in dot-bracket notation

16.33.2.2 vrna_pbacktrack5_TwoD()

```
char * vrna_pbacktrack5_TwoD (
    vrna_fold_compound_t * vc,
    int d1,
    int d2,
    unsigned int length )
```

```
#include <ViennaRNA/2Dpfold.h>
```

Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

This function does essentially the same as [vrna_pbacktrack_TwoD\(\)](#) with the only difference that partial structures, i.e. structures beginning from the 5' end with a specified length of the sequence, are backtracked

Note

This function does not work (since it makes no sense) for circular RNA sequences!

Precondition

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to [vrna_pf_TwoD\(\)](#) preceding this function is mandatory!

See also

[vrna_pbacktrack_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#)

Parameters

in, out	vc	The vrna_fold_compound_t datastructure containing all necessary folding attributes and matrices
in	d1	The distance to reference1 (may be -1)
in	d2	The distance to reference2
in	length	The length of the structure beginning from the 5' end

Returns

A sampled secondary structure in dot-bracket notation

16.34 Predicting various thermodynamic properties

Compute various thermodynamic properties using the partition function.

16.34.1 Detailed Description

Compute various thermodynamic properties using the partition function.

Many thermodynamic properties can be derived from the partition function

$$Q = \sum_{s \in \omega} e^{\frac{-E(s)}{kT}}.$$

In particular, for nucleic acids in equilibrium the probability $p(F)$ of a particular structural feature F follows Boltzmann's law, i.e.

$$p(F) \propto \sum_{s|F \in s} e^{\frac{-E(s)}{kT}}.$$

The actual probabilities can then be obtained from the ratio of those structures containing F and *all* structures, i.e.

$$p(F) = \frac{1}{Q} \sum_{s|F \in s} e^{\frac{-E(s)}{kT}}.$$

Consequently, a particular secondary structure s has equilibrium probability

$$p(s) = \frac{1}{Q} e^{\frac{-E(s)}{kT}}$$

which can be easily computed once Q and $E(s)$ are known.

On the other hand, efficient dynamic programming algorithms exist to compute the equilibrium probabilities

$$p_{ij} = \frac{1}{Q} \sum_{s|(i,j) \in s} e^{\frac{-E(s)}{kT}}$$

of base pairs (i, j) without the need for exhaustive enumeration of s .

This interface provides the functions for all thermodynamic property computations implemented in *RNAlib*. Collaboration diagram for Predicting various thermodynamic properties:

Files

- file [equilibrium_probs.h](#)
Equilibrium Probability implementations.
- file [heat_capacity.h](#)
Compute heat capacity for an RNA.

Data Structures

- struct [vrna_heat_capacity_s](#)
A single result from heat capacity computations. [More...](#)

Typedefs

- typedef void(* [vrna_heat_capacity_f](#)) (float temp, float heat_capacity, void *data)
The callback for heat capacity predictions.
- typedef struct [vrna_heat_capacity_s](#) [vrna_heat_capacity_t](#)
A single result from heat capacity computations.

Base pair probabilities and derived computations

- int [vrna_pairing_probs](#) ([vrna_fold_compound_t](#) *vc, char *structure)
- double [vrna_mean_bp_distance_pr](#) (int length, [FLT_OR_DBL](#) *pr)
Get the mean base pair distance in the thermodynamic ensemble from a probability matrix.
- double [vrna_mean_bp_distance](#) ([vrna_fold_compound_t](#) *vc)
Get the mean base pair distance in the thermodynamic ensemble.
- double [vrna_ensemble_defect_pt](#) ([vrna_fold_compound_t](#) *fc, const short *pt)
Compute the Ensemble Defect for a given target structure provided as a [vrna_ptable](#).
- double [vrna_ensemble_defect](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Compute the Ensemble Defect for a given target structure.
- double * [vrna_positional_entropy](#) ([vrna_fold_compound_t](#) *fc)
Compute a vector of positional entropies.
- [vrna_ep_t](#) * [vrna_stack_prob](#) ([vrna_fold_compound_t](#) *vc, double cutoff)
Compute stacking probabilities.

Multimer probabilities computations

- void [vrna_pf_dimer_probs](#) (double FAB, double FA, double FB, [vrna_ep_t](#) *prAB, const [vrna_ep_t](#) *prA, const [vrna_ep_t](#) *prB, int Alength, const [vrna_exp_param_t](#) *exp_params)
Compute Boltzmann probabilities of dimerization without homodimers.

Structure probability computations

- double [vrna_pr_structure](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Compute the equilibrium probability of a particular secondary structure.
- double [vrna_pr_energy](#) ([vrna_fold_compound_t](#) *vc, double e)

Basic heat capacity function interface

- [vrna_heat_capacity_t](#) * [vrna_heat_capacity](#) ([vrna_fold_compound_t](#) *fc, float T_min, float T_max, float T_increment, unsigned int mpoints)
Compute the specific heat for an RNA.
- int [vrna_heat_capacity_cb](#) ([vrna_fold_compound_t](#) *fc, float T_min, float T_max, float T_increment, unsigned int mpoints, [vrna_heat_capacity_f](#) cb, void *data)
Compute the specific heat for an RNA (callback variant)

Simplified heat capacity computation

- [vrna_heat_capacity_t](#) * [vrna_heat_capacity_simple](#) (const char *sequence, float T_min, float T_max, float T_increment, unsigned int mpoints)
Compute the specific heat for an RNA (simplified variant)

16.34.2 Data Structure Documentation

16.34.2.1 struct [vrna_heat_capacity_s](#)

A single result from heat capacity computations.

See also

[vrna_heat_capacity\(\)](#)

Data Fields

- float **temperature**
The temperature in °C.
- float **heat_capacity**
*The specific heat at this temperature in Kcal/(Mol * K)*

16.34.3 Typedef Documentation

16.34.3.1 vrna_heat_capacity_f

```
typedef void(* vrna_heat_capacity_f) (float temp, float heat_capacity, void *data)
#include <ViennaRNA/heat_capacity.h>
```

The callback for heat capacity predictions.

Notes on Callback Functions This function will be called for each evaluated temperature in the heat capacity prediction.

See also

[vrna_heat_capacity_cb\(\)](#)

Parameters

<i>temp</i>	The current temperature this results corresponds to in °C
<i>heat_capacity</i>	The heat capacity in Kcal/(Mol * K)
<i>data</i>	Some arbitrary data pointer passed through by the function executing the callback

16.34.3.2 vrna_heat_capacity_t

```
typedef struct vrna_heat_capacity_s vrna_heat_capacity_t
#include <ViennaRNA/heat_capacity.h>
```

A single result from heat capacity computations.
This is a convenience typedef for [vrna_heat_capacity_s](#), i.e. results as obtained from [vrna_heat_capacity\(\)](#)

16.34.4 Function Documentation

16.34.4.1 vrna_mean_bp_distance_pr()

```
double vrna_mean_bp_distance_pr (
    int length,
    FLT_OR_DBL * pr )
#include <ViennaRNA/equilibrium_probs.h>
```

Get the mean base pair distance in the thermodynamic ensemble from a probability matrix.

$$\langle d \rangle = \sum_{a,b} p_a p_b d(S_a, S_b)$$

this can be computed from the pair probs p_{ij} as

$$\langle d \rangle = \sum_{ij} p_{ij} (1 - p_{ij})$$

Parameters

<i>length</i>	The length of the sequence
<i>pr</i>	The matrix containing the base pair probabilities

Returns

The mean pair distance of the structure ensemble

16.34.4.2 `vrna_mean_bp_distance()`

```
double vrna_mean_bp_distance (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/equilibrium_probs.h>
Get the mean base pair distance in the thermodynamic ensemble.
```

$$\langle d \rangle = \sum_{a,b} p_a p_b d(S_a, S_b)$$

this can be computed from the pair probs p_{ij} as

$$\langle d \rangle = \sum_{ij} p_{ij} (1 - p_{ij})$$

Parameters

<i>vc</i>	The fold compound data structure
-----------	----------------------------------

Returns

The mean pair distance of the structure ensemble

SWIG Wrapper Notes This function is attached as method `mean_bp_distance()` to objects of type `fold_compound`

16.34.4.3 `vrna_ensemble_defect_pt()`

```
double vrna_ensemble_defect_pt (
    vrna_fold_compound_t * fc,
    const short * pt )
#include <ViennaRNA/equilibrium_probs.h>
Compute the Ensemble Defect for a given target structure provided as a vrna_ptable.
Given a target structure  $s$ , compute the average dissimilarity of a randomly drawn structure from the ensemble, i.e.:
```

$$ED(s) = 1 - \frac{1}{n} \sum_{ij, (i,j) \in s} p_{ij} - \frac{1}{n} \sum_i (1 - s_i) q_i$$

with sequence length n , the probability p_{ij} of a base pair (i, j) , the probability $q_i = 1 - \sum_j p_{ij}$ of nucleotide i being unpaired, and the indicator variable $s_i = 1$ if $\exists (i, j) \in s$, and $s_i = 0$ otherwise.

Precondition

The `vrna_fold_compound_t` input parameter `fc` must contain a valid base pair probability matrix. This means that partition function and base pair probabilities must have been computed using `fc` before execution of this function!

See also

[vrna_pf\(\)](#), [vrna_pairing_probs\(\)](#), [vrna_ensemble_defect\(\)](#)

Parameters

<i>fc</i>	A fold_compound with pre-computed base pair probabilities
<i>pt</i>	A pair table representing a target structure

Returns

The ensemble defect with respect to the target structure, or -1. upon failure, e.g. pre-conditions are not met

SWIG Wrapper Notes This function is attached as overloaded method **ensemble_defect()** to objects of type *fold_compound*.

16.34.4.4 vrna_ensemble_defect()

```
double vrna_ensemble_defect (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute the Ensemble Defect for a given target structure.

This is a wrapper around [vrna_ensemble_defect_pt\(\)](#). Given a target structure s , compute the average dissimilarity of a randomly drawn structure from the ensemble, i.e.:

$$ED(s) = 1 - \frac{1}{n} \sum_{ij, (i,j) \in s} p_{ij} - \frac{1}{n} \sum_i (1 - s_i) q_i$$

with sequence length n , the probability p_{ij} of a base pair (i, j) , the probability $q_i = 1 - \sum_j p_{ij}$ of nucleotide i being unpaired, and the indicator variable $s_i = 1$ if $\exists (i, j) \in s$, and $s_i = 0$ otherwise.

Precondition

The [vrna_fold_compound_t](#) input parameter fc must contain a valid base pair probability matrix. This means that partition function and base pair probabilities must have been computed using fc before execution of this function!

See also

[vrna_pf\(\)](#), [vrna_pairing_probs\(\)](#), [vrna_ensemble_defect_pt\(\)](#)

Parameters

<i>fc</i>	A fold_compound with pre-computed base pair probabilities
<i>structure</i>	A target structure in dot-bracket notation

Returns

The ensemble defect with respect to the target structure, or -1. upon failure, e.g. pre-conditions are not met

SWIG Wrapper Notes This function is attached as method **ensemble_defect()** to objects of type *fold_compound*. Note that the SWIG wrapper takes a structure in dot-bracket notation and converts it into a pair table using [vrna_ptable_from_string\(\)](#). The resulting pair table is then internally passed to [vrna_ensemble_defect_pt\(\)](#). To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of [vrna_ptable_from_string\(\)](#) for available options. (default: **VRNA_BRACKETS_RND**).

16.34.4.5 vrna_positional_entropy()

```
double * vrna_positional_entropy (
    vrna_fold_compound_t * fc )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute a vector of positional entropies.

This function computes the positional entropies from base pair probabilities as

$$S(i) = - \sum_j p_{ij} \log(p_{ij}) - q_i \log(q_i)$$

with unpaired probabilities $q_i = 1 - \sum_j p_{ij}$.

Low entropy regions have little structural flexibility and the reliability of the predicted structure is high. High entropy implies many structural alternatives. While these alternatives may be functionally important, they make structure prediction more difficult and thus less reliable.

Precondition

This function requires pre-computed base pair probabilities! Thus, [vrna_pf\(\)](#) must be called beforehand.

Parameters

<i>fc</i>	A fold_compound with pre-computed base pair probabilities
-----------	---

Returns

A 1-based vector of positional entropies $S(i)$. (position 0 contains the sequence length)

SWIG Wrapper Notes This function is attached as method **positional_entropy()** to objects of type *fold_compound*

16.34.4.6 vrna_stack_prob()

```
vrna_ep_t * vrna_stack_prob (
    vrna_fold_compound_t * vc,
    double cutoff )
#include <ViennaRNA/equilibrium_probs.h>
```

Compute stacking probabilities.

For each possible base pair (i, j) , compute the probability of a stack (i, j) , $(i + 1, j - 1)$.

Parameters

<i>vc</i>	The fold compound data structure with precomputed base pair probabilities
<i>cutoff</i>	A cutoff value that limits the output to stacks with $p > \text{cutoff}$.

Returns

A list of stacks with enclosing base pair (i, j) and probability p

16.34.4.7 vrna_pf_dimer_probs()

```
void vrna_pf_dimer_probs (
    double FAB,
    double FA,
```

```

double FB,
vrna_ep_t * prAB,
const vrna_ep_t * prA,
const vrna_ep_t * prB,
int Alength,
const vrna_exp_param_t * exp_params )
#include <ViennaRNA/equilibrium_probs.h>

```

Compute Boltzmann probabilities of dimerization without homodimers.

Given the pair probabilities and free energies (in the null model) for a dimer AB and the two constituent monomers A and B, compute the conditional pair probabilities given that a dimer AB actually forms. Null model pair probabilities are given as a list as produced by [vrna_plist_from_probs\(\)](#), the dimer probabilities 'prAB' are modified in place.

Parameters

<i>FAB</i>	free energy of dimer AB
<i>FA</i>	free energy of monomer A
<i>FB</i>	free energy of monomer B
<i>prAB</i>	pair probabilities for dimer
<i>prA</i>	pair probabilities monomer
<i>prB</i>	pair probabilities monomer
<i>Alength</i>	Length of molecule A
<i>exp_params</i>	The precomputed Boltzmann factors

16.34.4.8 vrna_pr_structure()

```

double vrna_pr_structure (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/equilibrium_probs.h>

```

Compute the equilibrium probability of a particular secondary structure.

The probability $p(s)$ of a particular secondary structure s can be computed as

$$p(s) = \frac{\exp(-\beta E(s))}{Z}$$

from the structures free energy $E(s)$ and the partition function

$$Z = \sum_s \exp(-\beta E(s)), \quad \text{with} \quad \beta = \frac{1}{RT}$$

where R is the gas constant and T the thermodynamic temperature.

Precondition

The fold compound fc must have went through a call to [vrna_pf\(\)](#) to fill the dynamic programming matrices with the corresponding partition function.

Parameters

<i>fc</i>	The fold compound data structure with precomputed partition function
<i>structure</i>	The secondary structure to compute the probability for in dot-bracket notation

Returns

The probability of the input structure (range $[0 : 1]$)

SWIG Wrapper Notes This function is attached as method **pr_structure()** to objects of type *fold_compound*

16.34.4.9 vrna_pr_energy()

```
double vrna_pr_energy (
    vrna_fold_compound_t * fc,
    double e )
#include <ViennaRNA/equilibrium_probs.h>
```

SWIG Wrapper Notes This function is attached as method **pr_energy()** to objects of type *fold_compound*

16.34.4.10 vrna_heat_capacity()

```
vrna_heat_capacity_t * vrna_heat_capacity (
    vrna_fold_compound_t * fc,
    float T_min,
    float T_max,
    float T_increment,
    unsigned int mpoints )
#include <ViennaRNA/heat_capacity.h>
```

Compute the specific heat for an RNA.

This function computes an RNAs specific heat in a given temperature range from the partition function by numeric differentiation. The result is returned as a list of pairs of temperature in °C and specific heat in Kcal/(Mol*K).

Users can specify the temperature range for the computation from *T_min* to *T_max*, as well as the increment step size *T_increment*. The latter also determines how many times the partition function is computed. Finally, the parameter *mpoints* determines how smooth the curve should be. The algorithm itself fits a parabola to $2 \cdot mpoints + 1$ data points to calculate 2nd derivatives. Increasing this parameter produces a smoother curve.

See also

[vrna_heat_capacity_cb\(\)](#), [vrna_heat_capacity_t](#), [vrna_heat_capacity_s](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t with the RNA sequence to analyze
<i>T_min</i>	Lowest temperature in °C
<i>T_max</i>	Highest temperature in °C
<i>T_increment</i>	Stepsize for temperature incrementation in °C (a reasonable choice might be 1 °C)
<i>mpoints</i>	The number of interpolation points to calculate 2nd derivative (a reasonable choice might be 2, min: 1, max: 100)

Returns

A list of pairs of temperatures and corresponding heat capacity or *NULL* upon any failure. The last entry of the list is indicated by a **temperature** field set to a value smaller than *T_min*

SWIG Wrapper Notes This function is attached as overloaded method **heat_capacity()** to objects of type *fold_compound*. If the optional function arguments *T_min*, *T_max*, *T_increment*, and *mpoints* are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

16.34.4.11 vrna_heat_capacity_cb()

```
int vrna_heat_capacity_cb (
    vrna_fold_compound_t * fc,
    float T_min,
    float T_max,
    float T_increment,
    unsigned int mpoints,
    vrna_heat_capacity_f cb,
    void * data )
#include <ViennaRNA/heat_capacity.h>
```

Compute the specific heat for an RNA (callback variant)

Similar to [vrna_heat_capacity\(\)](#), this function computes an RNAs specific heat in a given temperature range from the partition function by numeric differentiation. Instead of returning a list of temperature/specific heat pairs, however, this function returns the individual results through a callback mechanism. The provided function will be called for each result and passed the corresponding temperature and specific heat values along with the arbitrary data as provided through the `data` pointer argument.

Users can specify the temperature range for the computation from `T_min` to `T_max`, as well as the increment step size `T_increment`. The latter also determines how many times the partition function is computed. Finally, the parameter `mpoints` determines how smooth the curve should be. The algorithm itself fits a parabola to $2 \cdot mpoints + 1$ data points to calculate 2nd derivatives. Increasing this parameter produces a smoother curve.

See also

[vrna_heat_capacity\(\)](#), [vrna_heat_capacity_f](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t with the RNA sequence to analyze
<i>T_min</i>	Lowest temperature in °C
<i>T_max</i>	Highest temperature in °C
<i>T_increment</i>	Stepsize for temperature incrementation in °C (a reasonable choice might be 1 °C)
<i>mpoints</i>	The number of interpolation points to calculate 2nd derivative (a reasonable choice might be 2, min: 1, max: 100)
<i>cb</i>	The user-defined callback function that receives the individual results
<i>data</i>	An arbitrary data structure that will be passed to the callback in conjunction with the results

Returns

Returns 0 upon failure, and non-zero otherwise

SWIG Wrapper Notes This function is attached as method **heat_capacity_cb()** to objects of type *fold_compound*

16.34.4.12 vrna_heat_capacity_simple()

```
vrna_heat_capacity_t * vrna_heat_capacity_simple (
    const char * sequence,
    float T_min,
    float T_max,
    float T_increment,
    unsigned int mpoints )
#include <ViennaRNA/heat_capacity.h>
```

Compute the specific heat for an RNA (simplified variant)

Similar to [vrna_heat_capacity\(\)](#), this function computes an RNAs specific heat in a given temperature range from the partition function by numeric differentiation. This simplified version, however, only requires the RNA sequence as input instead of a `vrna_fold_compound_t` data structure. The result is returned as a list of pairs of temperature in °C and specific heat in Kcal/(Mol*K).

Users can specify the temperature range for the computation from `T_min` to `T_max`, as well as the increment step size `T_increment`. The latter also determines how many times the partition function is computed. Finally, the parameter `mpoints` determines how smooth the curve should be. The algorithm itself fits a parabola to $2 \cdot \text{mpoints} + 1$ data points to calculate 2nd derivatives. Increasing this parameter produces a smoother curve.

See also

[vrna_heat_capacity\(\)](#), [vrna_heat_capacity_cb\(\)](#), [vrna_heat_capacity_t](#), [vrna_heat_capacity_s](#)

Parameters

<i>sequence</i>	The RNA sequence input (must be uppercase)
<i>T_min</i>	Lowest temperature in °C
<i>T_max</i>	Highest temperature in °C
<i>T_increment</i>	Stepsize for temperature incrementation in °C (a reasonable choice might be 1 °C)
<i>mpoints</i>	The number of interpolation points to calculate 2nd derivative (a reasonable choice might be 2, min: 1, max: 100)

Returns

A list of pairs of temperatures and corresponding heat capacity or *NULL* upon any failure. The last entry of the list is indicated by a **temperature** field set to a value smaller than `T_min`

SWIG Wrapper Notes This function is available as overloaded function **heat_capacity()**. If the optional function arguments `T_min`, `T_max`, `T_increment`, and `mpoints` are omitted, they default to 0.0, 100.0, 1.0 and 2, respectively.

16.35 Compute the Density of States

16.35.1 Detailed Description

Collaboration diagram for Compute the Density of States:

Variables

- int [density_of_states](#) [`MAXDOS`+1]
The Density of States.

16.35.2 Variable Documentation

16.35.2.1 density_of_states

```
int density_of_states[MAXDOS+1] [extern]
#include <ViennaRNA/subopt.h>
```

The Density of States.

This array contains the density of states for an RNA sequences after a call to [subopt_par\(\)](#), [subopt\(\)](#) or [subopt_circ\(\)](#).

Precondition

Call one of the functions [subopt_par\(\)](#), [subopt\(\)](#) or [subopt_circ\(\)](#) prior accessing the contents of this array

See also

[subopt_par\(\)](#), [subopt\(\)](#), [subopt_circ\(\)](#)

16.36 Inverse Folding (Design)

RNA sequence design.

16.36.1 Detailed Description

RNA sequence design.

Files

- file [inverse.h](#)
Inverse folding routines.

Functions

- float [inverse_fold](#) (char *start, const char *target)
Find sequences with predefined structure.
- float [inverse_pf_fold](#) (char *start, const char *target)
Find sequence that maximizes probability of a predefined structure.

Variables

- char * [symbolset](#)
This global variable points to the allowed bases, initially "AUGC". It can be used to design sequences from reduced alphabets.
- float [final_cost](#)
- int [give_up](#)
- int [inv_verbose](#)

16.36.2 Function Documentation

16.36.2.1 [inverse_fold\(\)](#)

```
float inverse_fold (  
    char * start,  
    const char * target )  
#include <ViennaRNA/inverse.h>
```

Find sequences with predefined structure.

This function searches for a sequence with minimum free energy structure provided in the parameter 'target', starting with sequence 'start'. It returns 0 if the search was successful, otherwise a structure distance in terms of the energy difference between the search result and the actual target 'target' is returned. The found sequence is returned in 'start'. If [give_up](#) is set to 1, the function will return as soon as it is clear that the search will be unsuccessful, this speeds up the algorithm if you are only interested in exact solutions.

Parameters

<i>start</i>	The start sequence
<i>target</i>	The target secondary structure in dot-bracket notation

Returns

The distance to the target in case a search was unsuccessful, 0 otherwise

16.36.2.2 inverse_pf_fold()

```
float inverse_pf_fold (
    char * start,
    const char * target )
#include <ViennaRNA/inverse.h>
```

Find sequence that maximizes probability of a predefined structure.

This function searches for a sequence with maximum probability to fold into the provided structure 'target' using the partition function algorithm. It returns $-kT \cdot \log(p)$ where p is the frequency of 'target' in the ensemble of possible structures. This is usually much slower than [inverse_fold\(\)](#).

Parameters

<i>start</i>	The start sequence
<i>target</i>	The target secondary structure in dot-bracket notation

Returns

The distance to the target in case a search was unsuccessful, 0 otherwise

16.36.3 Variable Documentation**16.36.3.1 final_cost**

```
float final_cost [extern]
#include <ViennaRNA/inverse.h>
when to stop inverse\_pf\_fold\(\)
```

16.36.3.2 give_up

```
int give_up [extern]
#include <ViennaRNA/inverse.h>
default 0: try to minimize structure distance even if no exact solution can be found
```

16.36.3.3 inv_verbose

```
int inv_verbose [extern]
#include <ViennaRNA/inverse.h>
print out substructure on which inverse\_fold\(\) fails
```

16.37 Neighborhood Relation and Move Sets for Secondary Structures

Different functions to generate structural neighbors of a secondary structure according to a particular Move Set.

16.37.1 Detailed Description

Different functions to generate structural neighbors of a secondary structure according to a particular Move Set.

This module contains methods to compute the neighbors of an RNA secondary structure. Neighbors of a given structure are all structures that differ in exactly one base pair. That means one can insert an delete base pairs in the given structure. These insertions and deletions of base pairs are usually called moves. A third move which is considered in these methods is a shift move. A shifted base pair has one stable position and one position that changes. These moves are encoded as follows:

- insertion: (i, j) where $i, j > 0$

- deletion: (i, j) where $i, j < 0$
 shift: (i, j) where either $i > 0, j < 0$ or $i < 0, j > 0$
 The negative position of a shift indicates the position that has changed.

Example:

```
We have given a sequence and a structure.
Sequence  AAGGAAACC
Structure ..(.....)
Indices   123456789
```

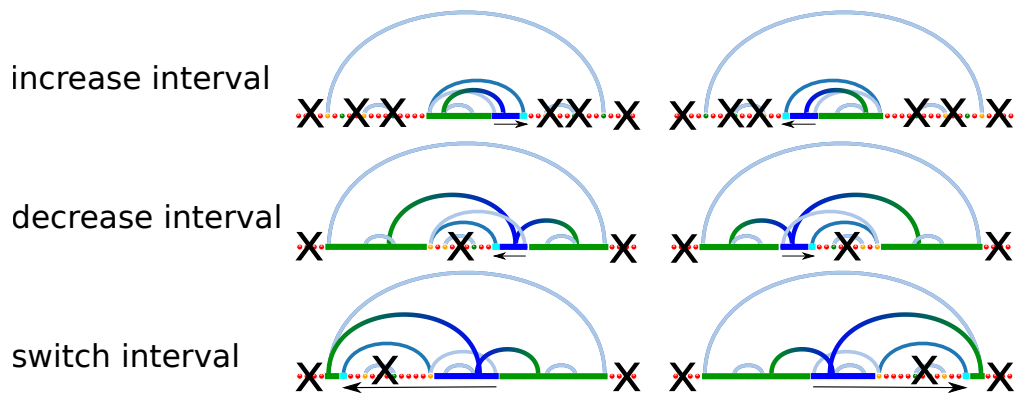
```
The given base pair is (3,9) and the neighbors are the insertion (4, 8), the deletion (-3,-9), the
shift (3,-8)
and the shift (-4, 9).
This leads to the neighbored structures:
... (....)
.....
... (....)
.... (....)
```

A simple method to construct all insertions is to iterate over the positions of a sequence twice. The first iteration has the index i in $[1, \text{sequence length}]$, the second iteration has the index j in $[i+1, \text{sequence length}]$. All pairs (i, j) with compatible letters and which are non-crossing with present base pairs are valid neighbored insertion moves. Valid deletion moves are all present base pairs with negative sign. Valid shift moves are constructed by taking all paired positions as fix position of a shift move and iterating over all positions of the sequence. If the letters of a position are compatible and if it the move is non-crossing with existing base pairs, we have a valid shift move. The method of generating shift moves can be accelerated by skipping neighbored base pairs.

If we need to construct all neighbors several times for subsequent moves, we can speed up the task by using the move set of the previous structure. The previous move set has to be filtered, such that all moves that would cross the next selected move are non-crossing. Next, the selected move has to be removed. Then one has to only to generate all moves that were not possible before. One move is the inverted selected move (if it was an insertion, simply make the indices negative). The generation of all other new moves is different and depends on the selected move. It is easy for an insertion move, because we have only to include all non-crossing shift moves, that are possible with the new base pair. For that we can either iterate over the sequence or we can select all crossing shift moves in the filter procedure and convert them into shifts.

The generation of new moves given a deletion is a little bit more complex, because we can create more moves. At first we can insert the deleted pair as insertion move. Then we generate all insertions that would have crossed the deleted base pair. Finally we construct all crossing shift moves.

If the given move is a shift, we can save much time by specifying the intervals for the generation of new moves. The interval which was enclosed by the positive position of the shift move and the previous paired position is the freed interval after applying the move. This freed interval includes all positions and base pairs that we need to construct new insertions and shifts. All these new moves have one position in the freed interval and the other position in the environment of the freed interval. The environment are all position which are outside the freed interval, but within the same enclosing loop of the shift move. The environment for valid base pairs can be divided into one or more intervals, depending on the shift move. The following examples describe a few scenarios to specify the intervals of the environment.



- freed interval
- environment for new non-crossing moves
- X intervals that would produce crossing pairs
- ↪ new shift moves from pairs in one interval to positions in the other interval
- points to the new position of the shift move

Given the intervals of the environment and the freed interval, the new shift moves can be constructed quickly. One has to take all positions of pairs from the environment in order to create valid pairs with positions in the freed interval. The same procedure can be applied for the other direction. This is taking all paired positions within the freed interval in order to look for pairs with valid positions in the intervals of the environment. Collaboration diagram for Neighborhood Relation and Move Sets for Secondary Structures:

Files

- file [move.h](#)
Methods to operate with structural neighbors of RNA secondary structures.
- file [neighbor.h](#)
Methods to compute the neighbors of an RNA secondary structure.

Data Structures

- struct [vrna_move_s](#)
An atomic representation of the transition / move from one structure to its neighbor. [More...](#)

Macros

- `#define VRNA_MOVESET_INSERTION 4`
Option flag indicating insertion move.
- `#define VRNA_MOVESET_DELETION 8`
Option flag indicating deletion move.
- `#define VRNA_MOVESET_SHIFT 16`
Option flag indicating shift move.
- `#define VRNA_MOVESET_NO_LP 32`
Option flag indicating moves without lonely base pairs.
- `#define VRNA_MOVESET_DEFAULT (VRNA_MOVESET_INSERTION | VRNA_MOVESET_DELETION)`
Option flag indicating default move set, i.e. insertions/deletion of a base pair.
- `#define VRNA_NEIGHBOR_CHANGE 1`
State indicator for a neighbor that has been changed.
- `#define VRNA_NEIGHBOR_INVALID 2`
State indicator for a neighbor that has been invalidated.
- `#define VRNA_NEIGHBOR_NEW 3`
State indicator for a neighbor that has become newly available.

Typedefs

- typedef struct [vrna_move_s](#) [vrna_move_t](#)
A single move that transforms a secondary structure into one of its neighbors.
- typedef void(* [vrna_move_update_f](#)) ([vrna_fold_compound_t](#) *fc, [vrna_move_t](#) neighbor, unsigned int state, void *data)
Prototype of the neighborhood update callback.

Functions

- [vrna_move_t](#) [vrna_move_init](#) (int pos_5, int pos_3)
Create an atomic move.
- void [vrna_move_list_free](#) ([vrna_move_t](#) *moves)
- void [vrna_move_apply](#) (short *pt, const [vrna_move_t](#) *m)
Apply a particular move / transition to a secondary structure, i.e. transform a structure.
- int [vrna_move_is_removal](#) (const [vrna_move_t](#) *m)
Test whether a move is a base pair removal.
- int [vrna_move_is_insertion](#) (const [vrna_move_t](#) *m)
Test whether a move is a base pair insertion.
- int [vrna_move_is_shift](#) (const [vrna_move_t](#) *m)
Test whether a move is a base pair shift.
- int [vrna_move_compare](#) (const [vrna_move_t](#) *a, const [vrna_move_t](#) *b, const short *pt)
Compare two moves.
- void [vrna_loopidx_update](#) (int *loopidx, const short *pt, int length, const [vrna_move_t](#) *m)
Alters the loopIndices array that was constructed with [vrna_loopidx_from_ptable\(\)](#).
- [vrna_move_t](#) * [vrna_neighbors](#) ([vrna_fold_compound_t](#) *vc, const short *pt, unsigned int options)
Generate neighbors of a secondary structure.
- [vrna_move_t](#) * [vrna_neighbors_successive](#) (const [vrna_fold_compound_t](#) *vc, const [vrna_move_t](#) *curr↔_move, const short *prev_pt, const [vrna_move_t](#) *prev_neighbors, int size_prev_neighbors, int *size_↔neighbors, unsigned int options)
Generate neighbors of a secondary structure (the fast way)
- int [vrna_move_neighbor_diff_cb](#) ([vrna_fold_compound_t](#) *fc, short *ptable, [vrna_move_t](#) move, [vrna_move_update_f](#) cb, void *data, unsigned int options)
Apply a move to a secondary structure and indicate which neighbors have changed consequentially.
- [vrna_move_t](#) * [vrna_move_neighbor_diff](#) ([vrna_fold_compound_t](#) *fc, short *ptable, [vrna_move_t](#) move, [vrna_move_t](#) **invalid_moves, unsigned int options)
Apply a move to a secondary structure and indicate which neighbors have changed consequentially.

16.37.2 Data Structure Documentation

16.37.2.1 struct [vrna_move_s](#)

An atomic representation of the transition / move from one structure to its neighbor.

An atomic transition / move may be one of the following:

- a **base pair insertion**,
- a **base pair removal**, or
- a **base pair shift** where an existing base pair changes one of its pairing partner.

These moves are encoded by two integer values that represent the affected 5' and 3' nucleotide positions. Furthermore, we use the following convention on the signedness of these encodings:

- both values are positive for *insertion moves*
- both values are negative for *base pair removals*
- both values have different signedness for *shift moves*, where the positive value indicates the nucleotide that stays constant, and the others absolute value is the new pairing partner

Note

A value of 0 in either field is used as list-end indicator and doesn't represent any valid move.

Collaboration diagram for `vrna_move_s`:

Data Fields

- `int pos_5`
The (absolute value of the) 5' position of a base pair, or any position of a shifted pair.
- `int pos_3`
The (absolute value of the) 3' position of a base pair, or any position of a shifted pair.
- `vrna_move_t * next`
The next base pair (if an elementary move changes more than one base pair), or `NULL` Has to be terminated with move 0,0.

16.37.3 Macro Definition Documentation**16.37.3.1 VRNA_MOVESET_INSERTION**

```
#define VRNA_MOVESET_INSERTION 4
#include <ViennaRNA/landscape/move.h>
```

Option flag indicating insertion move.

See also

[vrna_neighbors\(\)](#), [vrna_neighbors_successive](#), [vrna_path\(\)](#)

16.37.3.2 VRNA_MOVESET_DELETION

```
#define VRNA_MOVESET_DELETION 8
#include <ViennaRNA/landscape/move.h>
```

Option flag indicating deletion move.

See also

[vrna_neighbors\(\)](#), [vrna_neighbors_successive](#), [vrna_path\(\)](#)

16.37.3.3 VRNA_MOVESET_SHIFT

```
#define VRNA_MOVESET_SHIFT 16
#include <ViennaRNA/landscape/move.h>
```

Option flag indicating shift move.

See also

[vrna_neighbors\(\)](#), [vrna_neighbors_successive](#), [vrna_path\(\)](#)

16.37.3.4 VRNA_MOVESET_NO_LP

```
#define VRNA_MOVESET_NO_LP 32
#include <ViennaRNA/landscape/move.h>
```

Option flag indicating moves without lonely base pairs.

See also

[vrna_neighbors\(\)](#), [vrna_neighbors_successive](#), [vrna_path\(\)](#)

16.37.3.5 VRNA_MOVESET_DEFAULT

```
#define VRNA_MOVESET_DEFAULT (VRNA_MOVESET_INSERTION | VRNA_MOVESET_DELETION)
#include <ViennaRNA/landscape/move.h>
```

Option flag indicating default move set, i.e. insertions/deletion of a base pair.

See also

[vrna_neighbors\(\)](#), [vrna_neighbors_successive](#), [vrna_path\(\)](#)

16.37.3.6 VRNA_NEIGHBOR_CHANGE

```
#define VRNA_NEIGHBOR_CHANGE 1
#include <ViennaRNA/landscape/neighbor.h>
```

State indicator for a neighbor that has been changed.

See also

[vrna_move_neighbor_diff_cb\(\)](#)

16.37.3.7 VRNA_NEIGHBOR_INVALID

```
#define VRNA_NEIGHBOR_INVALID 2
#include <ViennaRNA/landscape/neighbor.h>
```

State indicator for a neighbor that has been invalidated.

See also

[vrna_move_neighbor_diff_cb\(\)](#)

16.37.3.8 VRNA_NEIGHBOR_NEW

```
#define VRNA_NEIGHBOR_NEW 3
#include <ViennaRNA/landscape/neighbor.h>
```

State indicator for a neighbor that has become newly available.

See also

[vrna_move_neighbor_diff_cb\(\)](#)

16.37.4 Typedef Documentation

16.37.4.1 vrna_move_update_f

```
typedef void(* vrna_move_update_f) (vrna_fold_compound_t *fc, vrna_move_t neighbor, unsigned
int state, void *data)
#include <ViennaRNA/landscape/neighbor.h>
```

Prototype of the neighborhood update callback.

See also

[vrna_move_neighbor_diff_cb\(\)](#), [VRNA_NEIGHBOR_CHANGE](#), [VRNA_NEIGHBOR_INVALID](#), [VRNA_NEIGHBOR_NEW](#)

Parameters

<i>fc</i>	The fold compound the calling function is working on
<i>neighbor</i>	The move that generates the (changed or new) neighbor
<i>state</i>	The state of the neighbor (move) as supplied by argument <i>neighbor</i>
<i>data</i>	Some arbitrary data pointer as passed to vrna_move_neighbor_diff_cb()

16.37.5 Function Documentation

16.37.5.1 `vrna_move_init()`

```
vrna_move_t vrna_move_init (
    int pos_5,
    int pos_3 )
#include <ViennaRNA/landscape/move.h>
Create an atomic move.
```

See also

[vrna_move_s](#)

Parameters

<code>pos_5</code>	The 5' position of the move (positive for insertions, negative for removal, any value for shift moves)
<code>pos_3</code>	The 3' position of the move (positive for insertions, negative for removal, any value for shift moves)

Returns

An atomic move as specified by `pos_5` and `pos_3`

16.37.5.2 `vrna_move_list_free()`

```
void vrna_move_list_free (
    vrna_move_t * moves )
#include <ViennaRNA/landscape/move.h>
delete all moves in a zero terminated list.
```

16.37.5.3 `vrna_move_apply()`

```
void vrna_move_apply (
    short * pt,
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Apply a particular move / transition to a secondary structure, i.e. transform a structure.
```

Parameters

<code>in, out</code>	<code>pt</code>	The pair table representation of the secondary structure
<code>in</code>	<code>m</code>	The move to apply

16.37.5.4 `vrna_move_is_removal()`

```
int vrna_move_is_removal (
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Test whether a move is a base pair removal.
```

Parameters

<i>m</i>	The move to test against
----------	--------------------------

Returns

Non-zero if the move is a base pair removal, 0 otherwise

16.37.5.5 vrna_move_is_insertion()

```
int vrna_move_is_insertion (
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Test whether a move is a base pair insertion.
```

Parameters

<i>m</i>	The move to test against
----------	--------------------------

Returns

Non-zero if the move is a base pair insertion, 0 otherwise

16.37.5.6 vrna_move_is_shift()

```
int vrna_move_is_shift (
    const vrna_move_t * m )
#include <ViennaRNA/landscape/move.h>
Test whether a move is a base pair shift.
```

Parameters

<i>m</i>	The move to test against
----------	--------------------------

Returns

Non-zero if the move is a base pair shift, 0 otherwise

16.37.5.7 vrna_move_compare()

```
int vrna_move_compare (
    const vrna_move_t * a,
    const vrna_move_t * b,
    const short * pt )
#include <ViennaRNA/landscape/move.h>
```

Compare two moves.

The function compares two moves *a* and *b* and returns whether move *a* is lexicographically smaller (-1), larger (1) or equal to move *b*.

If any of the moves *a* or *b* is a shift move, this comparison only makes sense in a structure context. Thus, the third argument with the current structure must be provided.

Note

This function returns 0 (equality) upon any error, e.g. missing input

Warning

Currently, shift moves are not supported!

Parameters

<i>a</i>	The first move of the comparison
<i>b</i>	The second move of the comparison
<i>pt</i>	The pair table of the current structure that is compatible with both moves (maybe NULL if moves are guaranteed to be no shifts)

Returns

-1 if $a < b$, 1 if $a > b$, 0 otherwise

16.37.5.8 vrna_loopidx_update()

```
void vrna_loopidx_update (
    int * loopidx,
    const short * pt,
    int length,
    const vrna_move_t * m )
```

#include <ViennaRNA/landscape/neighbor.h>

Alters the loopIndices array that was constructed with [vrna_loopidx_from_ptable\(\)](#).

The loopIndex of the current move will be inserted. The correctness of the input will not be checked because the speed should be optimized.

Parameters

in, out	<i>loopidx</i>	The loop index data structure that needs an update
in	<i>pt</i>	A pair table on which the move will be executed
	<i>length</i>	The length of the structure
in	<i>m</i>	The move that is applied to the current structure

16.37.5.9 vrna_neighbors()

```
vrna_move_t * vrna_neighbors (
    vrna_fold_compound_t * vc,
    const short * pt,
    unsigned int options )
```

#include <ViennaRNA/landscape/neighbor.h>

Generate neighbors of a secondary structure.

This function allows one to generate all structural neighbors (according to a particular move set) of an RNA secondary structure. The neighborhood is then returned as a list of transitions / moves required to transform the current structure into the actual neighbor.

See also

[vrna_neighbors_successive\(\)](#), [vrna_move_apply\(\)](#), [VRNA_MOVESET_INSERTION](#), [VRNA_MOVESET_DELETION](#), [VRNA_MOVESET_SHIFT](#), [VRNA_MOVESET_DEFAULT](#)

Parameters

in	<i>vc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
in	<i>pt</i>	The pair table representation of the structure
	<i>options</i>	Options to modify the behavior of this function, e.g. available move set

Returns

Neighbors as a list of moves / transitions (the last element in the list has both of its fields set to 0)

SWIG Wrapper Notes This function is attached as an overloaded method `neighbors()` to objects of type `fold_compound`. The optional parameter `options` defaults to `VRNA_MOVESET_DEFAULT` if it is omitted.

16.37.5.10 vrna_neighbors_successive()

```
vrna_move_t * vrna_neighbors_successive (
    const vrna_fold_compound_t * vc,
    const vrna_move_t * curr_move,
    const short * prev_pt,
    const vrna_move_t * prev_neighbors,
    int size_prev_neighbors,
    int * size_neighbors,
    unsigned int options )
#include <ViennaRNA/landscape/neighbor.h>
```

Generate neighbors of a secondary structure (the fast way)

This function implements a fast way to generate all neighbors of a secondary structure that results from successive applications of individual moves. The speed-up results from updating an already known list of valid neighbors before the individual move towards the current structure took place. In essence, this function removes neighbors that are not accessible anymore and inserts neighbors emerging after a move took place.

See also

[vrna_neighbors\(\)](#), [vrna_move_apply\(\)](#), [VRNA_MOVESET_INSERTION](#), [VRNA_MOVESET_DELETION](#), [VRNA_MOVESET_SHIFT](#), [VRNA_MOVESET_DEFAULT](#)

Parameters

in	<i>vc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
in	<i>curr_move</i>	The move that was/will be applied to <code>prev_pt</code>
in	<i>prev_pt</i>	A pair table representation of the structure before <code>curr_move</code> is/was applied
in	<i>prev_neighbors</i>	The list of neighbors of <code>prev_pt</code>
	<i>size_prev_neighbors</i>	The size of <code>prev_neighbors</code> , i.e. the lists length
out	<i>size_neighbors</i>	A pointer to store the size / length of the new neighbor list
	<i>options</i>	Options to modify the behavior of this function, e.g. available move set

Returns

Neighbors as a list of moves / transitions (the last element in the list has both of its fields set to 0)

16.37.5.11 vrna_move_neighbor_diff_cb()

```
int vrna_move_neighbor_diff_cb (
    vrna_fold_compound_t * fc,
```

```

    short * ptable,
    vrna_move_t move,
    vrna_move_update_f cb,
    void * data,
    unsigned int options )
#include <ViennaRNA/landscape/neighbor.h>

```

Apply a move to a secondary structure and indicate which neighbors have changed consequentially.

This function applies a move to a secondary structure and explores the local neighborhood of the affected loop. Any changes to previously compatible neighbors that have been affected by this loop will be reported through a callback function. In particular, any of the three cases might appear:

- A previously available neighbor move has changed, usually the free energy change of the move ([VRNA_NEIGHBOR_CHANGE](#))
- A previously available neighbor move became invalid ([VRNA_NEIGHBOR_INVALID](#))
- A new neighbor move becomes available ([VRNA_NEIGHBOR_NEW](#))

See also

[vrna_move_neighbor_diff\(\)](#), [VRNA_NEIGHBOR_CHANGE](#), [VRNA_NEIGHBOR_INVALID](#), [VRNA_NEIGHBOR_NEW](#), [vrna_move_update_f](#)

Parameters

<i>fc</i>	A fold compound for the RNA sequence(s) that this function operates on
<i>ptable</i>	The current structure as pair table
<i>move</i>	The move to apply
<i>cb</i>	The address of the callback function that is passed the neighborhood changes
<i>data</i>	An arbitrary data pointer that will be passed through to the callback function <i>cb</i>
<i>options</i>	Options to modify the behavior of this function, .e.g available move set

Returns

Non-zero on success, 0 otherwise

16.37.5.12 vrna_move_neighbor_diff()

```

vrna_move_t * vrna_move_neighbor_diff (
    vrna_fold_compound_t * fc,
    short * ptable,
    vrna_move_t move,
    vrna_move_t ** invalid_moves,
    unsigned int options )
#include <ViennaRNA/landscape/neighbor.h>

```

Apply a move to a secondary structure and indicate which neighbors have changed consequentially.

Similar to [vrna_move_neighbor_diff_cb\(\)](#), this function applies a move to a secondary structure and reports back the neighbors of the current structure become affected by this move. Instead of executing a callback for each of the affected neighbors, this function compiles two lists of neighbor moves, one that is returned and consists of all moves that are novel or may have changed in energy, and a second, *invalid_moves*, that consists of all the neighbor moves that become invalid, respectively.

Parameters

<i>fc</i>	A fold compound for the RNA sequence(s) that this function operates on
<i>ptable</i>	The current structure as pair table

Parameters

<i>move</i>	The move to apply
<i>invalid_moves</i>	The address of a move list where the function stores those moves that become invalid
<i>options</i>	Options to modify the behavior of this function, .e.g available move set

Returns

A list of moves that might have changed in energy or are novel compared to the structure before application of the move

16.38 (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima

API for various RNA folding path algorithms.

16.38.1 Detailed Description

API for various RNA folding path algorithms.

This part of our API allows for generating RNA secondary structure (re-)folding paths between two secondary structures or simply starting from a single structure. This is most important if an estimate of the refolding energy barrier between two structures is required, or a structure's corresponding local minimum needs to be determined, e.g. through a gradient-descent walk.

This part of the interface is further split into the following sections:

- [Direct Refolding Paths between two Secondary Structures](#), and
- [Folding Paths that start at a single Secondary Structure](#)

Collaboration diagram for (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima:

Modules

- [Direct Refolding Paths between two Secondary Structures](#)
Heuristics to explore direct, optimal (re-)folding paths between two secondary structures.
- [Folding Paths that start at a single Secondary Structure](#)
Implementation of gradient- and random walks starting from a single secondary structure.
- [Deprecated Interface for \(Re-\)folding Paths, Saddle Points, and Energy Barriers](#)

Files

- file [findpath.h](#)
A breadth-first search heuristic for optimal direct folding paths.
- file [paths.h](#)
API for computing (optimal) (re-)folding paths between secondary structures.
- file [walk.h](#)
Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence.

Data Structures

- struct [vrna_path_s](#)
An element of a refolding path list. [More...](#)

Macros

- `#define VRNA_PATH_TYPE_DOT_BRACKET 1U`
Flag to indicate producing a (re-)folding path as list of dot-bracket structures.
- `#define VRNA_PATH_TYPE_MOVES 2U`
Flag to indicate producing a (re-)folding path as list of transition moves.

Typedefs

- `typedef struct vrna_path_s vrna_path_t`
Typename for the refolding path data structure `vrna_path_s`.
- `typedef struct vrna_path_options_s * vrna_path_options_t`
Options data structure for (re-)folding path implementations.

Functions

- `void vrna_path_free (vrna_path_t *path)`
Release (free) memory occupied by a (re-)folding path.
- `void vrna_path_options_free (vrna_path_options_t options)`
Release (free) memory occupied by an options data structure for (re-)folding path implementations.

16.38.2 Data Structure Documentation

16.38.2.1 struct vrna_path_s

An element of a refolding path list.

Usually, one has to deal with an array of `vrna_path_s`, e.g. returned from one of the refolding-path algorithms. Since in most cases the length of the list is not known in advance, such lists have an *end-of-list* marker, which is either:

- a value of `NULL` for `vrna_path_s::s` if `vrna_path_s::type = VRNA_PATH_TYPE_DOT_BRACKET`, or
- a `vrna_path_s::move` with zero in both fields `vrna_move_t::pos_5` and `vrna_move_t::pos_3` if `vrna_path_s::type = VRNA_PATH_TYPE_MOVES`.

In the following we show an example for how to cover both cases of iteration:

```
vrna_path_t *ptr = path; // path was returned from one of the refolding path functions, e.g.
                        vrna_path_direct()

if (ptr) {
    if (ptr->type == VRNA_PATH_TYPE_DOT_BRACKET) {
        for (; ptr->s; ptr++)
            printf("%s [%6.2f]\n", ptr->s, ptr->en);
    } else if (ptr->type == VRNA_PATH_TYPE_MOVES) {
        for (; ptr->move.pos_5 != 0; ptr++)
            printf("move %d:%d, dG = %6.2f\n", ptr->move.pos_5, ptr->move.pos_3, ptr->en);
    }
}
```

See also

[vrna_path_free\(\)](#)

Collaboration diagram for `vrna_path_s`:

Data Fields

- unsigned int `type`
The type of the path element.
- double `en`
Free energy of current structure.
- char * `s`
Secondary structure in dot-bracket notation.
- `vrna_move_t` `move`
Move that transforms the previous structure into it's next neighbor along the path.

16.38.2.1.1 Field Documentation

16.38.2.1.1.1 `type` `unsigned int vrna_path_s::type`

The type of the path element.

A value of `VRNA_PATH_TYPE_DOT_BRACKET` indicates that `vrna_path_s::s` consists of the secondary structure in dot-bracket notation, and `vrna_path_s::en` the corresponding free energy.

On the other hand, if the value is `VRNA_PATH_TYPE_MOVES`, `vrna_path_s::s` is `NULL` and `vrna_path_s::move` is set to the transition move that transforms a previous structure into it's neighbor along the path. In this case, the attribute `vrna_path_s::en` states the change in free energy with respect to the structure before application of `vrna_path_s::move`.

16.38.3 Macro Definition Documentation

16.38.3.1 `VRNA_PATH_TYPE_DOT_BRACKET`

```
#define VRNA_PATH_TYPE_DOT_BRACKET 1U
#include <ViennaRNA/landscape/paths.h>
```

Flag to indicate producing a (re-)folding path as list of dot-bracket structures.

See also

[vrna_path_t](#), [vrna_path_options_findpath\(\)](#), [vrna_path_direct\(\)](#), [vrna_path_direct_ub\(\)](#)

16.38.3.2 `VRNA_PATH_TYPE_MOVES`

```
#define VRNA_PATH_TYPE_MOVES 2U
#include <ViennaRNA/landscape/paths.h>
```

Flag to indicate producing a (re-)folding path as list of transition moves.

See also

[vrna_path_t](#), [vrna_path_options_findpath\(\)](#), [vrna_path_direct\(\)](#), [vrna_path_direct_ub\(\)](#)

16.38.4 Function Documentation

16.38.4.1 `vrna_path_free()`

```
void vrna_path_free (
    vrna_path_t * path )
#include <ViennaRNA/landscape/paths.h>
```

Release (free) memory occupied by a (re-)folding path.

See also

[vrna_path_direct\(\)](#), [vrna_path_direct_ub\(\)](#), [vrna_path_findpath\(\)](#), [vrna_path_findpath_ub\(\)](#)

Parameters

<i>path</i>	The refolding path to be free'd
-------------	---------------------------------

16.38.4.2 vrna_path_options_free()

```
void vrna_path_options_free (
    vrna_path_options_t options )
#include <ViennaRNA/landscape/paths.h>
```

Release (free) memory occupied by an options data structure for (re-)folding path implementations.

See also

[vrna_path_options_findpath\(\)](#), [vrna_path_direct\(\)](#), [vrna_path_direct_ub\(\)](#)

Parameters

<i>options</i>	The options data structure to be free'd
----------------	---

16.39 Direct Refolding Paths between two Secondary Structures

Heuristics to explore direct, optimal (re-)folding paths between two secondary structures.

16.39.1 Detailed Description

Heuristics to explore direct, optimal (re-)folding paths between two secondary structures.

Collaboration diagram for Direct Refolding Paths between two Secondary Structures:

Functions

- [int vrna_path_findpath_saddle](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width)
Find energy of a saddle point between 2 structures (search only direct path)
- [int vrna_path_findpath_saddle_ub](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width, int maxE)
Find energy of a saddle point between 2 structures (search only direct path)
- [vrna_path_t](#) * [vrna_path_findpath](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width)
Find refolding path between 2 structures (search only direct path)
- [vrna_path_t](#) * [vrna_path_findpath_ub](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width, int maxE)
Find refolding path between 2 structures (search only direct path)
- [vrna_path_options_t](#) [vrna_path_options_findpath](#) (int width, unsigned int type)
Create options data structure for findpath direct (re-)folding path heuristic.
- [vrna_path_t](#) * [vrna_path_direct](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, [vrna_path_options_t](#) options)
Determine an optimal direct (re-)folding path between two secondary structures.
- [vrna_path_t](#) * [vrna_path_direct_ub](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int maxE, [vrna_path_options_t](#) options)
Determine an optimal direct (re-)folding path between two secondary structures.

16.39.2 Function Documentation

16.39.2.1 vrna_path_findpath_saddle()

```
int vrna_path_findpath_saddle (
    vrna_fold_compound_t * fc,
    const char * s1,
```

```

        const char * s2,
        int width )
#include <ViennaRNA/landscape/findpath.h>

```

Find energy of a saddle point between 2 structures (search only direct path)

This function uses an implementation of the *findpath* algorithm [9] for near-optimal direct refolding path prediction. Model details, and energy parameters are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);
```

See also

[vrna_path_findpath_saddle_ub\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#), [vrna_path_findpath\(\)](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t with precomputed sequence encoding and model details
<i>s1</i>	The start structure in dot-bracket notation
<i>s2</i>	The target structure in dot-bracket notation
<i>width</i>	A number specifying how many strutures are being kept at each step during the search

Returns

The saddle energy in 10cal/mol

SWIG Wrapper Notes This function is attached as an overloaded method *path_findpath_saddle()* to objects of type *fold_compound*. The optional parameter *width* defaults to 1 if it is omitted.

16.39.2.2 vrna_path_findpath_saddle_ub()

```

int vrna_path_findpath_saddle_ub (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int width,
    int maxE )
#include <ViennaRNA/landscape/findpath.h>

```

Find energy of a saddle point between 2 structures (search only direct path)

This function uses an implementation of the *findpath* algorithm [9] for near-optimal direct refolding path prediction. Model details, and energy parameters are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);
```

Warning

The argument `maxE` (E_{max}) enables one to specify an upper bound, or maximum free energy for the saddle point between the two input structures. If no path with $E_{saddle} < E_{max}$ is found, the function simply returns `maxE`

See also

[vrna_path_findpath_saddle\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#), [vrna_path_findpath\(\)](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t with precomputed sequence encoding and model details
<i>s1</i>	The start structure in dot-bracket notation
<i>s2</i>	The target structure in dot-bracket notation
<i>width</i>	A number specifying how many strutures are being kept at each step during the search
<i>maxE</i>	An upper bound for the saddle point energy in 10cal/mol

Returns

The saddle energy in 10cal/mol

SWIG Wrapper Notes This function is attached as an overloaded method `path_findpath_saddle()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to INF. In case the function did not find a path with $E_{\text{saddle}} < E_{\text{max}}$ the function returns a `NULL` object, i.e. `undef` for Perl and `None` for Python.

16.39.2.3 `vrna_path_findpath()`

```
vrna_path_t * vrna_path_findpath (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int width )
```

```
#include <ViennaRNA/landscape/findpath.h>
```

Find refolding path between 2 structures (search only direct path)

This function uses an implementation of the *findpath* algorithm [9] for near-optimal direct refolding path prediction.

Model details, and energy parameters are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);
```

See also

[vrna_path_findpath_ub\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#), [vrna_path_findpath_saddle\(\)](#)

Parameters

<i>fc</i>	The <code>vrna_fold_compound_t</code> with precomputed sequence encoding and model details
<i>s1</i>	The start structure in dot-bracket notation
<i>s2</i>	The target structure in dot-bracket notation
<i>width</i>	A number specifying how many structures are being kept at each step during the search

Returns

The saddle energy in 10cal/mol

SWIG Wrapper Notes This function is attached as an overloaded method `path_findpath()` to objects of type `fold_compound`. The optional parameter `width` defaults to 1 if it is omitted.

16.39.2.4 `vrna_path_findpath_ub()`

```
vrna_path_t * vrna_path_findpath_ub (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int width,
    int maxE )
```

```
#include <ViennaRNA/landscape/findpath.h>
```

Find refolding path between 2 structures (search only direct path)

This function uses an implementation of the *findpath* algorithm [9] for near-optimal direct refolding path prediction.

Model details, and energy parameters are used as provided via the parameter 'fc'. The `vrna_fold_compound_t` does not require memory for any DP matrices, but requires all most basic init values as one would get from a call like this:

```
fc = vrna_fold_compound(sequence, NULL, VRNA_OPTION_DEFAULT);
```

Warning

The argument `maxE` enables one to specify an upper bound, or maximum free energy for the saddle point between the two input structures. If no path with $E_{saddle} < E_{max}$ is found, the function simply returns `NULL`

See also

[vrna_path_findpath\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_t](#), [vrna_path_findpath_saddle\(\)](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t with precomputed sequence encoding and model details
<i>s1</i>	The start structure in dot-bracket notation
<i>s2</i>	The target structure in dot-bracket notation
<i>width</i>	A number specifying how many strutures are being kept at each step during the search
<i>maxE</i>	An upper bound for the saddle point energy in 10cal/mol

Returns

The saddle energy in 10cal/mol

SWIG Wrapper Notes This function is attached as an overloaded method `path_findpath()` to objects of type `fold↔_compound`. The optional parameter `width` defaults to 1 if it is omitted, while the optional parameter `maxE` defaults to INF. In case the function did not find a path with $E_{saddle} < E_{max}$ the function returns an empty list.

16.39.2.5 vrna_path_options_findpath()

```
vrna_path_options_t vrna_path_options_findpath (
    int width,
    unsigned int type )
```

```
#include <ViennaRNA/landscape/paths.h>
```

Create options data structure for findpath direct (re-)folding path heuristic.

This function returns an options data structure that switches the [vrna_path_direct\(\)](#) and [vrna_path_direct_ub\(\)](#) API functions to use the *findpath* [9] heuristic. The parameter `width` specifies the width of the breadth-first search while the second parameter `type` allows one to set the type of the returned (re-)folding path.

Currently, the following return types are available:

- A list of dot-bracket structures and corresponding free energy (flag: [VRNA_PATH_TYPE_DOT_BRACKET](#))
- A list of transition moves and corresponding free energy changes (flag: [VRNA_PATH_TYPE_MOVES](#))

See also

[VRNA_PATH_TYPE_DOT_BRACKET](#), [VRNA_PATH_TYPE_MOVES](#), [vrna_path_options_free\(\)](#), [vrna_path_direct\(\)](#), [vrna_path_direct_ub\(\)](#)

Parameters

<i>width</i>	Width of the breath-first search strategy
<i>type</i>	Setting that specifies how the return (re-)folding path should be encoded

Returns

An options data structure with settings for the findpath direct path heuristic

SWIG Wrapper Notes This function is available as overloaded function `path_options_findpath()`. The optional parameter `width` defaults to 10 if omitted, while the optional parameter `type` defaults to `VRNA_PATH_TYPE_DOT_BRACKET`.

16.39.2.6 `vrna_path_direct()`

```
vrna_path_t * vrna_path_direct (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    vrna_path_options_t options )
#include <ViennaRNA/landscape/paths.h>
```

Determine an optimal direct (re-)folding path between two secondary structures.

This is the generic wrapper function to retrieve (an optimal) (re-)folding path between two secondary structures `s1` and `s2`. The actual algorithm that is used to generate the (re-)folding path is determined by the settings specified in the `options` data structure. This data structure also determines the return type, which might be either:

- a list of dot-bracket structures with corresponding free energy, or
- a list of transition moves with corresponding free energy change

If the `options` parameter is passed a `NULL` pointer, this function defaults to the *findpath heuristic* [9] with a breadth-first search width of 10, and the returned path consists of dot-bracket structures with corresponding free energies.

See also

[vrna_path_direct_ub\(\)](#), [vrna_path_options_findpath\(\)](#), [vrna_path_options_free\(\)](#), [vrna_path_free\(\)](#)

Parameters

<code>fc</code>	The <code>vrna_fold_compound_t</code> with precomputed sequence encoding and model details
<code>s1</code>	The start structure in dot-bracket notation
<code>s2</code>	The target structure in dot-bracket notation
<code>options</code>	An options data structure that specifies the path heuristic and corresponding settings (maybe <code>NULL</code>)

Returns

An optimal (re-)folding path between the two input structures

SWIG Wrapper Notes This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `options` defaults to `NULL` if it is omitted.

16.39.2.7 `vrna_path_direct_ub()`

```
vrna_path_t * vrna_path_direct_ub (
    vrna_fold_compound_t * fc,
    const char * s1,
    const char * s2,
    int maxE,
    vrna_path_options_t options )
#include <ViennaRNA/landscape/paths.h>
```

Determine an optimal direct (re-)folding path between two secondary structures.

This function is similar to `vrna_path_direct()`, but allows to specify an *upper-bound* for the saddle point energy. The underlying algorithms will stop determining an (optimal) (re-)folding path, if none can be found that has a saddle point below the specified upper-bound threshold `maxE`.

Warning

The argument `maxE` enables one to specify an upper bound, or maximum free energy for the saddle point between the two input structures. If no path with $E_{saddle} < E_{max}$ is found, the function simply returns `NULL`.

See also

`vrna_path_direct_ub()`, `vrna_path_options_findpath()`, `vrna_path_options_free()`, `vrna_path_free()`

Parameters

<code>fc</code>	The <code>vrna_fold_compound_t</code> with precomputed sequence encoding and model details
<code>s1</code>	The start structure in dot-bracket notation
<code>s2</code>	The target structure in dot-bracket notation
<code>maxE</code>	Upper bound for the saddle point along the (re-)folding path
<code>options</code>	An options data structure that specifies the path heuristic and corresponding settings (maybe <code>NULL</code>)

Returns

An optimal (re-)folding path between the two input structures

SWIG Wrapper Notes This function is attached as an overloaded method `path_direct()` to objects of type `fold_compound`. The optional parameter `maxE` defaults to `#INT_MAX - 1` if it is omitted, while the optional parameter `options` defaults to `NULL`. In case the function did not find a path with $E_{saddle} < E_{max}$ it returns an empty list.

16.40 Folding Paths that start at a single Secondary Structure

Implementation of gradient- and random walks starting from a single secondary structure.

16.40.1 Detailed Description

Implementation of gradient- and random walks starting from a single secondary structure.
Collaboration diagram for Folding Paths that start at a single Secondary Structure:

Macros

- `#define VRNA_PATH_STEEPEST_DESCENT 128`
Option flag to request a steepest descent / gradient path.
- `#define VRNA_PATH_RANDOM 256`
Option flag to request a random walk path.
- `#define VRNA_PATH_NO_TRANSITION_OUTPUT 512`
Option flag to omit returning the transition path.
- `#define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)`
Option flag to request defaults (steepest descent / default move set)

Functions

- `vrna_move_t * vrna_path (vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)`
Compute a path, store the final structure, and return a list of transition moves from the start to the final structure.
- `vrna_move_t * vrna_path_gradient (vrna_fold_compound_t *vc, short *pt, unsigned int options)`

Compute a steepest descent / gradient path, store the final structure, and return a list of transition moves from the start to the final structure.

- [vrna_move_t](#) * [vrna_path_random](#) ([vrna_fold_compound_t](#) *vc, short *pt, unsigned int steps, unsigned int options)

Generate a random walk / path of a given length, store the final structure, and return a list of transition moves from the start to the final structure.

16.40.2 Macro Definition Documentation

16.40.2.1 VRNA_PATH_STEEPEST_DESCENT

```
#define VRNA_PATH_STEEPEST_DESCENT 128
#include <ViennaRNA/landscape/walk.h>
```

Option flag to request a steepest descent / gradient path.

See also

[vrna_path\(\)](#)

16.40.2.2 VRNA_PATH_RANDOM

```
#define VRNA_PATH_RANDOM 256
#include <ViennaRNA/landscape/walk.h>
```

Option flag to request a random walk path.

See also

[vrna_path\(\)](#)

16.40.2.3 VRNA_PATH_NO_TRANSITION_OUTPUT

```
#define VRNA_PATH_NO_TRANSITION_OUTPUT 512
#include <ViennaRNA/landscape/walk.h>
```

Option flag to omit returning the transition path.

See also

[vrna_path\(\)](#), [vrna_path_gradient\(\)](#), [vrna_path_random\(\)](#)

16.40.2.4 VRNA_PATH_DEFAULT

```
#define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)
#include <ViennaRNA/landscape/walk.h>
```

Option flag to request defaults (steepest descent / default move set)

See also

[vrna_path\(\)](#), [VRNA_PATH_STEEPEST_DESCENT](#), [VRNA_MOVESET_DEFAULT](#)

16.40.3 Function Documentation

16.40.3.1 `vrna_path()`

```
vrna_move_t * vrna_path (
    vrna_fold_compound_t * vc,
    short * pt,
    unsigned int steps,
    unsigned int options )
#include <ViennaRNA/landscape/walk.h>
```

Compute a path, store the final structure, and return a list of transition moves from the start to the final structure. This function computes, given a start structure in pair table format, a transition path, updates the pair table to the final structure of the path. Finally, if not requested otherwise by using the [VRNA_PATH_NO_TRANSITION_OUTPUT](#) flag in the `options` field, this function returns a list of individual transitions that lead from the start to the final structure if requested.

The currently available transition paths are

- Steepest Descent / Gradient walk (flag: [VRNA_PATH_STEEPEST_DESCENT](#))
- Random walk (flag: [VRNA_PATH_RANDOM](#))

The type of transitions must be set through the `options` parameter

Note

Since the result is written to the input structure you may want to use [vrna_ptable_copy\(\)](#) before calling this function to keep the initial structure

See also

[vrna_path_gradient\(\)](#), [vrna_path_random\(\)](#), [vrna_ptable\(\)](#), [vrna_ptable_copy\(\)](#), [vrna_fold_compound\(\)](#), [VRNA_PATH_STEEPEST_DESCENT](#), [VRNA_PATH_RANDOM](#), [VRNA_MOVESET_DEFAULT](#), [VRNA_MOVESET_SHIFT](#), [VRNA_PATH_NO_TRANSITION_OUTPUT](#)

Parameters

in	<i>vc</i>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
in, out	<i>pt</i>	The pair table containing the start structure. Used to update to the final structure after execution of this function
in	<i>options</i>	Options to modify the behavior of this function

Returns

A list of transition moves (default), or NULL (if options & [VRNA_PATH_NO_TRANSITION_OUTPUT](#))

SWIG Wrapper Notes This function is attached as an overloaded method `path()` to objects of type `fold_compound`. The optional parameter `options` defaults to [VRNA_PATH_DEFAULT](#) if it is omitted.

16.40.3.2 `vrna_path_gradient()`

```
vrna_move_t * vrna_path_gradient (
    vrna_fold_compound_t * vc,
    short * pt,
    unsigned int options )
#include <ViennaRNA/landscape/walk.h>
```

Compute a steepest descent / gradient path, store the final structure, and return a list of transition moves from the start to the final structure.

This function computes, given a start structure in pair table format, a steepest descent path, updates the pair table to the final structure of the path. Finally, if not requested otherwise by using the [VRNA_PATH_NO_TRANSITION_OUTPUT](#) flag in the `options` field, this function returns a list of individual transitions that lead from the start to the final structure if requested.

Note

Since the result is written to the input structure you may want to use [vrna_ptable_copy\(\)](#) before calling this function to keep the initial structure

See also

[vrna_path_random\(\)](#), [vrna_path\(\)](#), [vrna_ptable\(\)](#), [vrna_ptable_copy\(\)](#), [vrna_fold_compound\(\)](#) [VRNA_MOVESET_DEFAULT](#), [VRNA_MOVESET_SHIFT](#), [VRNA_PATH_NO_TRANSITION_OUTPUT](#)

Parameters

<code>in</code>	<code>vc</code>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<code>in, out</code>	<code>pt</code>	The pair table containing the start structure. Used to update to the final structure after execution of this function
<code>in</code>	<code>options</code>	Options to modify the behavior of this function

Returns

A list of transition moves (default), or NULL (if options & [VRNA_PATH_NO_TRANSITION_OUTPUT](#))

SWIG Wrapper Notes This function is attached as an overloaded method `path_gradient()` to objects of type `fold_compound`. The optional parameter `options` defaults to [VRNA_PATH_DEFAULT](#) if it is omitted.

16.40.3.3 vrna_path_random()

```
vrna_move_t * vrna_path_random (
    vrna_fold_compound_t * vc,
    short * pt,
    unsigned int steps,
    unsigned int options )
#include <ViennaRNA/landscape/walk.h>
```

Generate a random walk / path of a given length, store the final structure, and return a list of transition moves from the start to the final structure.

This function generates, given a start structure in pair table format, a random walk / path, updates the pair table to the final structure of the path. Finally, if not requested otherwise by using the [VRNA_PATH_NO_TRANSITION_OUTPUT](#) flag in the `options` field, this function returns a list of individual transitions that lead from the start to the final structure if requested.

Note

Since the result is written to the input structure you may want to use [vrna_ptable_copy\(\)](#) before calling this function to keep the initial structure

See also

[vrna_path_gradient\(\)](#), [vrna_path\(\)](#), [vrna_ptable\(\)](#), [vrna_ptable_copy\(\)](#), [vrna_fold_compound\(\)](#) [VRNA_MOVESET_DEFAULT](#), [VRNA_MOVESET_SHIFT](#), [VRNA_PATH_NO_TRANSITION_OUTPUT](#)

Parameters

<code>in</code>	<code>vc</code>	A <code>vrna_fold_compound_t</code> containing the energy parameters and model details
<code>in, out</code>	<code>pt</code>	The pair table containing the start structure. Used to update to the final structure after execution of this function
<code>in</code>	<code>steps</code>	The length of the path, i.e. the total number of transitions / moves
<code>in</code>	<code>options</code>	Options to modify the behavior of this function

Returns

A list of transition moves (default), or NULL (if options & [VRNA_PATH_NO_TRANSITION_OUTPUT](#))

SWIG Wrapper Notes This function is attached as an overloaded method *path_gradient()* to objects of type *fold_compound*. The optional parameter *options* defaults to [VRNA_PATH_DEFAULT](#) if it is omitted.

16.41 Experimental Structure Probing Data

Include Experimental Structure Probing Data to Guide Structure Predictions.

16.41.1 Detailed Description

Include Experimental Structure Probing Data to Guide Structure Predictions.

Collaboration diagram for Experimental Structure Probing Data:

Modules

- [SHAPE Reactivity Data](#)

Incorporate SHAPE reactivity structure probing data into the folding recursions by means of soft constraints.

- [Generate Soft Constraints from Data](#)

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

16.42 SHAPE Reactivity Data

Incorporate SHAPE reactivity structure probing data into the folding recursions by means of soft constraints.

16.42.1 Detailed Description

Incorporate SHAPE reactivity structure probing data into the folding recursions by means of soft constraints.

Details for our implementation to incorporate SHAPE reactivity data to guide secondary structure prediction can be found in [22] Collaboration diagram for SHAPE Reactivity Data:

Files

- file [SHAPE.h](#)

This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints.

Functions

- int [vrna_sc_add_SHAPE_deigan](#) ([vrna_fold_compound_t](#) *vc, const double *reactivities, double m, double b, unsigned int options)

Add SHAPE reactivity data as soft constraints (Deigan et al. method)

- int [vrna_sc_add_SHAPE_deigan_al](#)i ([vrna_fold_compound_t](#) *vc, const char **shape_files, const int *shape_file_association, double m, double b, unsigned int options)

Add SHAPE reactivity data from files as soft constraints for consensus structure prediction (Deigan et al. method)

- int [vrna_sc_add_SHAPE_zarringhalam](#) ([vrna_fold_compound_t](#) *vc, const double *reactivities, double b, double default_value, const char *shape_conversion, unsigned int options)

Add SHAPE reactivity data as soft constraints (Zarringhalam et al. method)

- int [vrna_sc_SHAPE_to_pr](#) (const char *shape_conversion, double *values, int length, double default_value)

Convert SHAPE reactivity values to probabilities for being unpaired.

16.42.2 Function Documentation

16.42.2.1 `vrna_sc_add_SHAPE_deigan()`

```
int vrna_sc_add_SHAPE_deigan (
    vrna_fold_compound_t * vc,
    const double * reactivities,
    double m,
    double b,
    unsigned int options )
#include <ViennaRNA/constraints/SHAPE.h>
Add SHAPE reactivity data as soft constraints (Deigan et al. method)
This approach of SHAPE directed RNA folding uses the simple linear ansatz
```

$$\Delta G_{\text{SHAPE}}(i) = m \ln(\text{SHAPE reactivity}(i) + 1) + b$$

to convert SHAPE reactivity values to pseudo energies whenever a nucleotide i contributes to a stacked pair. A positive slope m penalizes high reactivities in paired regions, while a negative intercept b results in a confirmatory “bonus” free energy for correctly predicted base pairs. Since the energy evaluation of a base pair stack involves two pairs, the pseudo energies are added for all four contributing nucleotides. Consequently, the energy term is applied twice for pairs inside a helix and only once for pairs adjacent to other structures. For all other loop types the energy model remains unchanged even when the experimental data highly disagrees with a certain motif.

See also

For further details, we refer to [8].

[vrna_sc_remove\(\)](#), [vrna_sc_add_SHAPE_zarringhalam\(\)](#), [vrna_sc_minimize_pertubation\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the soft constraints are associated with
<i>reactivities</i>	A vector of normalized SHAPE reactivities
<i>m</i>	The slope of the conversion function
<i>b</i>	The intercept of the conversion function
<i>options</i>	The options flag indicating how/where to store the soft constraints

Returns

1 on successful extraction of the method, 0 on errors

SWIG Wrapper Notes This function is attached as method `sc_add_SHAPE_deigan()` to objects of type `fold_compound`

16.42.2.2 `vrna_sc_add_SHAPE_deigan_al()`

```
int vrna_sc_add_SHAPE_deigan_al (
    vrna_fold_compound_t * vc,
    const char ** shape_files,
    const int * shape_file_association,
    double m,
    double b,
    unsigned int options )
#include <ViennaRNA/constraints/SHAPE.h>
Add SHAPE reactivity data from files as soft constraints for consensus structure prediction (Deigan et al. method)
```

Parameters

<i>vc</i>	The vrna_fold_compound_t the soft constraints are associated with
<i>shape_files</i>	A set of filenames that contain normalized SHAPE reactivity data
<i>shape_file_association</i>	An array of integers that associate the files with sequences in the alignment
<i>m</i>	The slope of the conversion function
<i>b</i>	The intercept of the conversion function
<i>options</i>	The options flag indicating how/where to store the soft constraints

Returns

1 on successful extraction of the method, 0 on errors

SWIG Wrapper Notes This function is attached as method `sc_add_SHAPE_deigan_ali()` to objects of type `fold↔_compound`

16.42.2.3 `vrna_sc_add_SHAPE_zarringhalam()`

```
int vrna_sc_add_SHAPE_zarringhalam (
    vrna_fold_compound_t * vc,
    const double * reactivities,
    double b,
    double default_value,
    const char * shape_conversion,
    unsigned int options )
```

#include <[ViennaRNA/constraints/SHAPE.h](#)>

Add SHAPE reactivity data as soft constraints (Zarringhalam et al. method)

This method first converts the observed SHAPE reactivity of nucleotide i into a probability q_i that position i is unpaired by means of a non-linear map. Then pseudo-energies of the form

$$\Delta G_{\text{SHAPE}}(x, i) = \beta |x_i - q_i|$$

are computed, where $x_i = 0$ if position i is unpaired and $x_i = 1$ if i is paired in a given secondary structure. The parameter β serves as scaling factor. The magnitude of discrepancy between prediction and experimental observation is represented by $|x_i - q_i|$.

See also

For further details, we refer to [34]

[vrna_sc_remove\(\)](#), [vrna_sc_add_SHAPE_deigan\(\)](#), [vrna_sc_minimize_pertubation\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t the soft constraints are associated with
<i>reactivities</i>	A vector of normalized SHAPE reactivities
<i>b</i>	The scaling factor β of the conversion function
<i>default_value</i>	The default value for a nucleotide where reactivity data is missing for
<i>shape_conversion</i>	A flag that specifies how to convert reactivities to probabilities
<i>options</i>	The options flag indicating how/where to store the soft constraints

Returns

1 on successful extraction of the method, 0 on errors

SWIG Wrapper Notes This function is attached as method `sc_add_SHAPE_zarringhalam()` to objects of type `fold_compound`

16.42.2.4 `vrna_sc_SHAPE_to_pr()`

```
int vrna_sc_SHAPE_to_pr (
    const char * shape_conversion,
    double * values,
    int length,
    double default_value )
#include <ViennaRNA/constraints/SHAPE.h>
```

Convert SHAPE reactivity values to probabilities for being unpaired.

This function parses the informations from a given file and stores the result in the preallocated string sequence and the `FLT_OR_DBL` array values.

See also

[vrna_file_SHAPE_read\(\)](#)

Parameters

<i>shape_conversion</i>	String defining the method used for the conversion process
<i>values</i>	Pointer to an array of SHAPE reactivities
<i>length</i>	Length of the array of SHAPE reactivities
<i>default_value</i>	Result used for position with invalid/missing reactivity values

16.43 Generate Soft Constraints from Data

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

16.43.1 Detailed Description

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

Collaboration diagram for Generate Soft Constraints from Data:

Files

- file [perturbation_fold.h](#)

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

Macros

- `#define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0`
Use the sum of squared aberrations as objective function.
- `#define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1`
Use the sum of absolute aberrations as objective function.
- `#define VRNA_MINIMIZER_DEFAULT 0`

Use a custom implementation of the gradient descent algorithm to minimize the objective function.

- #define [VRNA_MINIMIZER_CONJUGATE_FR](#) 1

Use the GNU Scientific Library implementation of the Fletcher-Reeves conjugate gradient algorithm to minimize the objective function.

- #define [VRNA_MINIMIZER_CONJUGATE_PR](#) 2

Use the GNU Scientific Library implementation of the Polak-Ribiere conjugate gradient algorithm to minimize the objective function.

- #define [VRNA_MINIMIZER_VECTOR_BFGS](#) 3

Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.

- #define [VRNA_MINIMIZER_VECTOR_BFGS2](#) 4

Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.

- #define [VRNA_MINIMIZER_STEEPEST_DESCENT](#) 5

Use the GNU Scientific Library implementation of the steepest descent algorithm to minimize the objective function.

Typedefs

- typedef void(* [progress_callback](#)) (int iteration, double score, double *epsilon)

Callback for following the progress of the minimization process.

Functions

- void [vrna_sc_minimize_pertubation](#) ([vrna_fold_compound_t](#) *vc, const double *q_prob_unpaired, int objective_function, double sigma_squared, double tau_squared, int algorithm, int sample_size, double *epsilon, double initialStepSize, double minStepSize, double minImprovement, double minimizerTolerance, [progress_callback](#) callback)

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

16.43.2 Macro Definition Documentation

16.43.2.1 VRNA_OBJECTIVE_FUNCTION_QUADRATIC

```
#define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0
#include <ViennaRNA/perturbation_fold.h>
```

Use the sum of squared aberrations as objective function.

$$F(\vec{\epsilon}) = \sum_{i=1}^n \frac{\epsilon_i^2}{\tau^2} + \sum_{i=1}^n \frac{(p_i(\vec{\epsilon}) - q_i)^2}{\sigma^2} \rightarrow \min$$

16.43.2.2 VRNA_OBJECTIVE_FUNCTION_ABSOLUTE

```
#define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1
#include <ViennaRNA/perturbation_fold.h>
```

Use the sum of absolute aberrations as objective function.

$$F(\vec{\epsilon}) = \sum_{i=1}^n \frac{|\epsilon_i|}{\tau^2} + \sum_{i=1}^n \frac{|p_i(\vec{\epsilon}) - q_i|}{\sigma^2} \rightarrow \min$$

16.43.2.3 VRNA_MINIMIZER_DEFAULT

```
#define VRNA_MINIMIZER_DEFAULT 0
#include <ViennaRNA/perturbation_fold.h>
```

Use a custom implementation of the gradient descent algorithm to minimize the objective function.

16.43.2.4 VRNA_MINIMIZER_CONJUGATE_FR

```
#define VRNA_MINIMIZER_CONJUGATE_FR 1
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the Fletcher-Reeves conjugate gradient algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

16.43.2.5 VRNA_MINIMIZER_CONJUGATE_PR

```
#define VRNA_MINIMIZER_CONJUGATE_PR 2
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the Polak-Ribiere conjugate gradient algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

16.43.2.6 VRNA_MINIMIZER_VECTOR_BFGS

```
#define VRNA_MINIMIZER_VECTOR_BFGS 3
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

16.43.2.7 VRNA_MINIMIZER_VECTOR_BFGS2

```
#define VRNA_MINIMIZER_VECTOR_BFGS2 4
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

16.43.2.8 VRNA_MINIMIZER_STEEPEST_DESCENT

```
#define VRNA_MINIMIZER_STEEPEST_DESCENT 5
#include <ViennaRNA/perturbation_fold.h>
```

Use the GNU Scientific Library implementation of the steepest descent algorithm to minimize the objective function.

Please note that this algorithm can only be used when the GNU Scientific Library is available on your system

16.43.3 Typedef Documentation**16.43.3.1 progress_callback**

```
typedef void(* progress_callback) (int iteration, double score, double *epsilon)
#include <ViennaRNA/perturbation_fold.h>
```

Callback for following the progress of the minimization process.

Parameters

<i>iteration</i>	The number of the current iteration
<i>score</i>	The score of the objective function
<i>epsilon</i>	The perturbation vector yielding the reported score

16.43.4 Function Documentation

16.43.4.1 vrna_sc_minimize_pertubation()

```
void vrna_sc_minimize_pertubation (
    vrna_fold_compound_t * vc,
    const double * q_prob_unpaired,
    int objective_function,
    double sigma_squared,
    double tau_squared,
    int algorithm,
    int sample_size,
    double * epsilon,
    double initialStepSize,
    double minStepSize,
    double minImprovement,
    double minimizerTolerance,
    progress_callback callback )
```

```
#include <ViennaRNA/perturbation_fold.h>
```

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

Use an iterative minimization algorithm to find a vector of perturbation energies whose incorporation as soft constraints shifts the predicted pairing probabilities closer to the experimentally observed probabilities. The algorithm aims to minimize an objective function that penalizes discrepancies between predicted and observed pairing probabilities and energy model adjustments, i.e. an appropriate vector of perturbation energies satisfies

$$F(\vec{\epsilon}) = \sum_{\mu} \frac{\epsilon_{\mu}^2}{\tau^2} + \sum_{i=1}^n \frac{(p_i(\vec{\epsilon}) - q_i)^2}{\sigma^2} \rightarrow \min.$$

An initialized fold compound and an array containing the observed probability for each nucleotide to be unbound are required as input data. The parameters `objective_function`, `sigma_squared` and `tau_squared` are responsible for adjusting the aim of the objective function. Dependend on which type of objective function is selected, either squared or absolute aberrations are contributing to the objective function. The ratio of the parameters `sigma_squared` and `tau_squared` can be used to adjust the algorithm to find a solution either close to the thermodynamic prediction (`sigma_squared >> tau_squared`) or close to the experimental data (`tau_squared >> sigma_squared`). The minimization can be performed by making use of a custom gradient descent implementation or using one of the minimizing algorithms provided by the GNU Scientific Library. All algorithms require the evaluation of the gradient of the objective function, which includes the evaluation of conditional pairing probabilities. Since an exact evaluation is expensive, the probabilities can also be estimated from sampling by setting an appropriate sample size. The found vector of perturbation energies will be stored in the array `epsilon`. The progress of the minimization process can be tracked by implementing and passing a callback function.

See also

For further details we refer to [29].

Parameters

<code>vc</code>	Pointer to a fold compound
<code>q_prob_unpaired</code>	Pointer to an array containing the probability to be unpaired for each nucleotide
<code>objective_function</code>	The type of objective function to be used (VRNA_OBJECTIVE_FUNCTION_QUADRATIC / VRNA_OBJECTIVE_FUNCTION_LINEAR)
<code>sigma_squared</code>	A factor used for weighting the objective function. More weight on this factor will lead to a solution close to the null vector.
<code>tau_squared</code>	A factor used for weighting the objective function. More weight on this factor will lead to a solution close to the data provided in <code>q_prob_unpaired</code> .
<code>algorithm</code>	The minimization algorithm (VRNA_MINIMIZER_*)
<code>sample_size</code>	The number of sampled sequences used for estimating the pairing probabilities. A value ≤ 0 will lead to an exact evaluation.
<code>epsilon</code>	A pointer to an array used for storing the calculated vector of perturbation energies
<code>callback</code>	A pointer to a callback function used for reporting the current minimization progress

16.44 Ligands Binding to RNA Structures

Simple Extensions to Model Ligand Binding to RNA Structures.

16.44.1 Detailed Description

Simple Extensions to Model Ligand Binding to RNA Structures.

Collaboration diagram for Ligands Binding to RNA Structures:

Modules

- [Ligands Binding to Unstructured Domains](#)
Add ligand binding to loop regions using the [Unstructured Domains](#) feature.
- [Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints](#)
Ligand binding to specific hairpin/interior loop like motifs using the [Soft Constraints](#) feature.

Files

- file [ligand.h](#)
Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework.

16.45 Ligands Binding to Unstructured Domains

Add ligand binding to loop regions using the [Unstructured Domains](#) feature.

Add ligand binding to loop regions using the [Unstructured Domains](#) feature.

Sometime, certain ligands, like single strand binding (SSB) proteins, compete with intramolecular base pairing of the RNA. In situations, where the dissociation constant of the ligand is known and the ligand binds to a consecutive stretch of single-stranded nucleotides we can use the [Unstructured Domains](#) functionality to extend the RNA folding grammar. This module provides a convenience default implementation that covers most of the application scenarios. The function `vrna_ud_add_motif()` attaches a ligands sequence motif and corresponding binding free energy to the list of known ligand motifs within a `vrna_fold_compound_t.domains_up` attribute. The first call to this function initializes the [Unstructured Domains](#) feature with our default implementation. Subsequent calls of secondary structure prediction algorithms with the modified `vrna_fold_compound_t` then directly include the competition of the ligand with regulas base pairing. Since we utilize the unstructured domain extension, The ligand binding model can be removed again using the `vrna_ud_remove()` function. Collaboration diagram for Ligands Binding to Unstructured Domains:

16.46 Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints

Ligand binding to specific hairpin/interior loop like motifs using the [Soft Constraints](#) feature.

16.46.1 Detailed Description

Ligand binding to specific hairpin/interior loop like motifs using the [Soft Constraints](#) feature.

Collaboration diagram for Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints:

Data Structures

- struct [vrna_sc_motif_s](#)

Typedefs

- typedef struct [vrna_sc_motif_s](#) [vrna_sc_motif_t](#)
Type definition for soft constraint motif.

Functions

- int `vrna_sc_add_hi_motif` (`vrna_fold_compound_t` *fc, const char *seq, const char *structure, FLT_OR_DBL energy, unsigned int options)

Add soft constraints for hairpin or interior loop binding motif.

16.46.2 Data Structure Documentation

16.46.2.1 struct `vrna_sc_motif_s`

16.46.3 Function Documentation

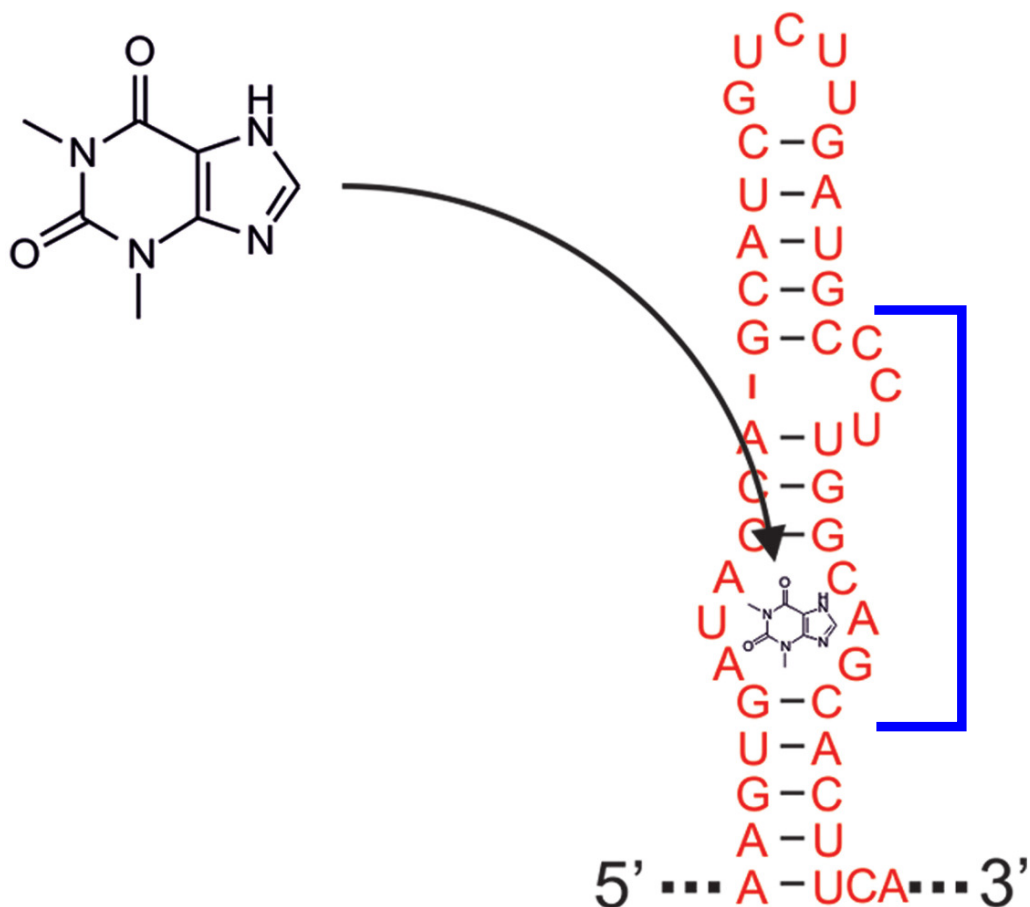
16.46.3.1 `vrna_sc_add_hi_motif()`

```
int vrna_sc_add_hi_motif (
    vrna_fold_compound_t * fc,
    const char * seq,
    const char * structure,
    FLT_OR_DBL energy,
    unsigned int options )
```

#include <ViennaRNA/constraints/ligand.h>

Add soft constraints for hairpin or interior loop binding motif.

Here is an example that adds a theophylline binding motif. Free energy contribution is derived from $k_d = 0.1\mu M$, taken from Jenison et al. 1994. At 1M concentration the corresponding binding free energy amounts to -9.93 kcal/mol .



```
vrna_sc_add_hi_motif(fc,
    "GAUACCAG&CCCUUGGCAGC",
```

```
" (... ((( (&) ... ))) ... )",
-9.93, VRNA_OPTION_DEFAULT);
```

Parameters

<i>fc</i>	The vrna_fold_compound_t the motif is applied to
<i>seq</i>	The sequence motif (may be interspaced by '&' character)
<i>structure</i>	The structure motif (may be interspaced by '&' character)
<i>energy</i>	The free energy of the motif (e.g. binding free energy)
<i>options</i>	Options

Returns

non-zero value if application of the motif using soft constraints was successful

SWIG Wrapper Notes This function is attached as method `sc_add_hi_motif()` to objects of type *fold_compound*

16.47 Structure Modules and Pseudoknots

16.47.1 Detailed Description

Collaboration diagram for Structure Modules and Pseudoknots:

Modules

- [Pseudoknots](#)
Implementations to predict pseudoknotted structures.
- [G-Quadruplexes](#)
Various functions related to G-quadruplex computations.

Files

- file [gquad.h](#)
G-quadruplexes.

16.48 Pseudoknots

Implementations to predict pseudoknotted structures.

16.48.1 Detailed Description

Implementations to predict pseudoknotted structures.

Collaboration diagram for Pseudoknots:

Files

- file [pk_plex.h](#)
Heuristics for two-step pseudoknot forming interaction predictions.

Data Structures

- struct [vrna_pk_plex_result_s](#)
A result of the RNA PKplex interaction prediction. [More...](#)

Typedefs

- typedef int(* [vrna_pk_plex_score_f](#)) (const short *pt, int start_5, int end_5, int start_3, int end_3, void *data)
Pseudoknot loop scoring function prototype.
- typedef struct vrna_pk_plex_option_s * [vrna_pk_plex_opt_t](#)
RNA PKplex options object.
- typedef struct [vrna_pk_plex_result_s](#) [vrna_pk_plex_t](#)
Convenience typedef for results of the RNA PKplex prediction.

Functions

- [vrna_pk_plex_t](#) * [vrna_pk_plex](#) ([vrna_fold_compound_t](#) *fc, const int **accessibility, [vrna_pk_plex_opt_t](#) options)
Predict Pseudoknot interactions in terms of a two-step folding process.
- int ** [vrna_pk_plex_accessibility](#) (const char *sequence, unsigned int [unpaired](#), double cutoff)
Obtain a list of opening energies suitable for PKplex computations.
- [vrna_pk_plex_opt_t](#) [vrna_pk_plex_opt_defaults](#) (void)
Default options for PKplex algorithm.
- [vrna_pk_plex_opt_t](#) [vrna_pk_plex_opt](#) (unsigned int delta, unsigned int max_interaction_length, int pk_↔ penalty)
Simple options for PKplex algorithm.
- [vrna_pk_plex_opt_t](#) [vrna_pk_plex_opt_fun](#) (unsigned int delta, unsigned int max_interaction_length, [vrna_pk_plex_score_f](#) scoring_function, void *scoring_data)
Simple options for PKplex algorithm.

16.48.2 Data Structure Documentation

16.48.2.1 struct vrna_pk_plex_result_s

A result of the RNA PKplex interaction prediction.

See also

[vrna_pk_plex_t](#)

Data Fields

- char * **structure**
Secondary Structure in dot-bracket notation.
- double **energy**
Net free energy in kcal/mol.
- double **dGpk**
Free energy of PK loop in kcal/mol.
- double **dGint**
Free energy of PK forming duplex interaction.
- double **dG1**
Opening energy for the 5' interaction site used in the heuristic.
- double **dG2**
Opening energy for the 3' interaction site used in the heuristic.
- unsigned int **start_5**
Start coordinate of the 5' interaction site.
- unsigned int **end_5**
End coordinate of the 5' interaction site.
- unsigned int **start_3**
Start coordinate of the 3' interaction site.
- unsigned int **end_3**
End coordinate of the 3' interaction site.

16.48.3 Typedef Documentation

16.48.3.1 `vrna_pk_plex_score_f`

```
typedef int(* vrna_pk_plex_score_f) (const short *pt, int start_5, int end_5, int start_3, int end_3, void *data)
```

```
#include <ViennaRNA/pk_plex.h>
```

Pseudoknot loop scoring function prototype.

This function is used to evaluate a formed pseudoknot (PK) interaction in `vrna_pk_plex()`. It is supposed to take a PK-free secondary structure as input and coordinates of an additional interaction site. From this data, the energy (penalty) to score the PK loop is derived and returned in decakal/mol. Upon passing zero in any of the interaction site coordinates (`start_5`, `end_5`, `start_3`, `end_3`) or a `NULL` pointer in `pt`, the function must return a PK loop score. This minimum PK loop score is used in the first phase of the heuristic implemented in `vrna_pk_plex()` to assess whether a particular interaction is further taken into account in a later, more thorough evaluation step.

The simplest scoring function would simply return a constant score for any PK loop, no matter what type of loop is formed and how large the loop is. This is the default if `vrna_pk_plex_opt_defaults()` or `vrna_pk_plex_opt()` is used to generate options for `vrna_pk_plex()`.

See also

[vrna_pk_plex_opt_fun\(\)](#), [vrna_pk_plex\(\)](#)

Parameters

<code>pt</code>	The secondary structure (without pseudoknot) in pair table notation
<code>start_5</code>	The start coordinate of the 5' site of the pseudoknot interaction
<code>end_5</code>	The end coordinate of the 5' site of the pseudoknot interaction
<code>start_3</code>	The start coordinate of the 3' site of the pseudoknot interaction
<code>end_3</code>	The end coordinate of the 3' site of the pseudoknot interaction
<code>data</code>	An arbitrary data structure passed from the calling function

Returns

The energy (penalty) of the resulting pseudoknot

16.48.3.2 `vrna_pk_plex_opt_t`

```
typedef struct vrna_pk_plex_option_s* vrna_pk_plex_opt_t
```

```
#include <ViennaRNA/pk_plex.h>
```

RNA PKplex options object.

See also

[vrna_pk_plex_opt_defaults\(\)](#), [vrna_pk_plex_opt\(\)](#), [vrna_pk_plex_opt_fun\(\)](#), [vrna_pk_plex\(\)](#), [vrna_pk_plex_score_f](#)

16.48.3.3 `vrna_pk_plex_t`

```
typedef struct vrna_pk_plex_result_s vrna_pk_plex_t
```

```
#include <ViennaRNA/pk_plex.h>
```

Convenience typedef for results of the RNA PKplex prediction.

See also

[#vrna_pk_plex_results_s](#), [vrna_pk_plex\(\)](#)

16.48.4 Function Documentation

16.48.4.1 vrna_pk_plex()

```
vrna_pk_plex_t * vrna_pk_plex (
    vrna_fold_compound_t * fc,
    const int ** accessibility,
    vrna_pk_plex_opt_t options )
#include <ViennaRNA/pk_plex.h>
```

Predict Pseudoknot interactions in terms of a two-step folding process.

Computes simple pseudoknot interactions according to the PKplex algorithm. This simple heuristic first compiles a list of potential interaction sites that may form a pseudoknot. The resulting candidate interactions are then fixed and an PK-free MFE structure for the remainder of the sequence is computed.

The `accessibility` argument is a list of opening energies for potential interaction sites. It is used in the first step of the algorithm to identify potential interactions. Upon passing `NULL`, the opening energies are determined automatically based on the current model settings.

Depending on the `options`, the function can return the MFE (incl. PK loops) or suboptimal structures within an energy band around the MFE. The PK loop is internally scored by a scoring function that in the simplest cases assigns a constant value for each PK loop. More complicated scoring functions can be passed as well, see [vrna_pk_plex_score_f](#) and [vrna_pk_plex_opt_fun\(\)](#).

The function returns `NULL` on any error. Otherwise, a list of structures and interaction coordinates with corresponding energy contributions is returned. If no PK-interaction that satisfies the options is found, the list only consists of the PK-free MFE structure.

Parameters

<code>fc</code>	fold compound with the input sequence and model settings
<code>accessibility</code>	An array of opening energies for the implemented heuristic (maybe <code>NULL</code>)
<code>options</code>	An vrna_pk_plex_opt_t options data structure that determines the algorithm parameters

Returns

A list of potentially pseudoknotted structures (Last element in the list indicated by `NULL` value in [vrna_pk_plex_result_s.structure](#))

16.48.4.2 vrna_pk_plex_accessibility()

```
int ** vrna_pk_plex_accessibility (
    const char * sequence,
    unsigned int unpaired,
    double cutoff )
#include <ViennaRNA/pk_plex.h>
```

Obtain a list of opening energies suitable for PKplex computations.

See also

[vrna_pk_plex\(\)](#)

Parameters

<code>sequence</code>	The RNA sequence
<code>unpaired</code>	The maximum number of unpaired nucleotides, i.e. length of interaction
<code>cutoff</code>	A cutoff value for unpaired probabilities

Returns

Opening energies as required for [vrna_pk_plex\(\)](#)

16.48.4.3 [vrna_pk_plex_opt_defaults\(\)](#)

```
vrna_pk_plex_opt_t vrna_pk_plex_opt_defaults (
    void )
#include <ViennaRNA/pk_plex.h>
Default options for PKplex algorithm.
```

See also

[vrna_pk_plex\(\)](#), [vrna_pk_plex_opt\(\)](#), [vrna_pk_plex_opt_fun\(\)](#)

Returns

An options data structure suitable for PKplex computations

16.48.4.4 [vrna_pk_plex_opt\(\)](#)

```
vrna_pk_plex_opt_t vrna_pk_plex_opt (
    unsigned int delta,
    unsigned int max_interaction_length,
    int pk_penalty )
#include <ViennaRNA/pk_plex.h>
Simple options for PKplex algorithm.
```

See also

[vrna_pk_plex\(\)](#), [vrna_pk_plex_opt_defaults\(\)](#), [vrna_pk_plex_opt_fun\(\)](#)

Parameters

<i>delta</i>	Size of energy band around MFE for suboptimal results in dekal/mol
<i>max_interaction_length</i>	Maximum length of interaction
<i>pk_penalty</i>	Energy constant to score the PK forming loop

Returns

An options data structure suitable for PKplex computations

16.48.4.5 [vrna_pk_plex_opt_fun\(\)](#)

```
vrna_pk_plex_opt_t vrna_pk_plex_opt_fun (
    unsigned int delta,
    unsigned int max_interaction_length,
    vrna_pk_plex_score_f scoring_function,
    void * scoring_data )
#include <ViennaRNA/pk_plex.h>
Simple options for PKplex algorithm.
```

See also

[vrna_pk_plex\(\)](#), [vrna_pk_plex_opt_defaults\(\)](#), [vrna_pk_plex_opt\(\)](#), [vrna_pk_plex_score_f](#)

Parameters

<i>delta</i>	Size of energy band around MFE for suboptimal results in dekalcal/mol
<i>max_interaction_length</i>	Maximum length of interaction
<i>scoring_function</i>	Energy evaluating function to score the PK forming loop
<i>scoring_data</i>	An arbitrary data structure passed to the scoring function (maybe <i>NUL</i>)

Returns

An options data structure suitable for PKplex computations

16.49 G-Quadruplexes

Various functions related to G-quadruplex computations.

16.49.1 Detailed Description

Various functions related to G-quadruplex computations.

Collaboration diagram for G-Quadruplexes:

Functions

- `int * get_gquad_matrix` (short *S, `vrna_param_t` *P)
Get a triangular matrix prefilled with minimum free energy contributions of G-quadruplexes.
- `int parse_gquad` (const char *struc, int *L, int l[3])
- PRIVATE `int backtrack_GQuad_IntLoop` (int c, int i, int j, int type, short *S, int *ggg, int *index, int *p, int *q, `vrna_param_t` *P)
- PRIVATE `int backtrack_GQuad_IntLoop_L` (int c, int i, int j, int type, short *S, int **ggg, int maxdist, int *p, int *q, `vrna_param_t` *P)

16.49.2 Function Documentation

16.49.2.1 get_gquad_matrix()

```
int * get_gquad_matrix (
    short * S,
    vrna_param_t * P )
#include <ViennaRNA/gquad.h>
```

Get a triangular matrix prefilled with minimum free energy contributions of G-quadruplexes.

At each position ij in the matrix, the minimum free energy of any G-quadruplex delimited by i and j is stored. If no G-quadruplex formation is possible, the matrix element is set to INF. Access the elements in the matrix via `matrix[indx[j]+i]`. To get the integer array `indx` see `get_jindx()`.

See also

`get_jindx()`, `encode_sequence()`

Parameters

<i>S</i>	The encoded sequence
<i>P</i>	A pointer to the data structure containing the precomputed energy contributions

Returns

A pointer to the G-quadruplex contribution matrix

16.49.2.2 parse_gquad()

```
int parse_gquad (
    const char * struc,
    int * L,
    int l[3] )
#include <ViennaRNA/gquad.h>
given a dot-bracket structure (possibly) containing gquads encoded by '+' signs, find first gquad, return end position
or 0 if none found Upon return L and l[] contain the number of stacked layers, as well as the lengths of the linker
regions. To parse a string with many gquads, call parse_gquad repeatedly e.g. end1 = parse_gquad(struc, &L, l); ...
; end2 = parse_gquad(struc+end1, &L, l); end2+=end1; ... ; end3 = parse_gquad(struc+end2, &L, l); end3+=end2;
... ;
```

16.49.2.3 backtrack_GQuad_IntLoop()

```
PRIVATE int backtrack_GQuad_IntLoop (
    int c,
    int i,
    int j,
    int type,
    short * S,
    int * ggg,
    int * index,
    int * p,
    int * q,
    vrna_param_t * P )
#include <ViennaRNA/gquad.h>
backtrack an interior loop like enclosed g-quadruplex with closing pair (i,j)
```

Parameters

<i>c</i>	The total contribution the loop should resemble
<i>i</i>	position i of enclosing pair
<i>j</i>	position j of enclosing pair
<i>type</i>	base pair type of enclosing pair (must be reverse type)
<i>S</i>	integer encoded sequence
<i>ggg</i>	triangular matrix containing g-quadruplex contributions
<i>index</i>	the index for accessing the triangular matrix
<i>p</i>	here the 5' position of the gquad is stored
<i>q</i>	here the 3' position of the gquad is stored
<i>P</i>	the datastructure containing the precalculated contributions

Returns

1 on success, 0 if no gquad found

16.49.2.4 backtrack_GQuad_IntLoop_L()

```
PRIVATE int backtrack_GQuad_IntLoop_L (
    int c,
```

```

    int i,
    int j,
    int type,
    short * S,
    int ** ggg,
    int maxdist,
    int * p,
    int * q,
    vrna_param_t * P )
#include <ViennaRNA/gquad.h>
backtrack an interior loop like enclosed g-quadruplex with closing pair (i,j) with underlying Lfold matrix

```

Parameters

<i>c</i>	The total contribution the loop should resemble
<i>i</i>	position i of enclosing pair
<i>j</i>	position j of enclosing pair
<i>type</i>	base pair type of enclosing pair (must be reverse type)
<i>S</i>	integer encoded sequence
<i>ggg</i>	triangular matrix containing g-quadruplex contributions
<i>p</i>	here the 5' position of the gquad is stored
<i>q</i>	here the 3' position of the gquad is stored
<i>P</i>	the datastructure containing the precalculated contibutions

Returns

1 on success, 0 if no gquad found

16.50 Post-transcriptional Modifications

Support of modified bases in secondary structure prediction.

16.50.1 Detailed Description

Support of modified bases in secondary structure prediction.

Energy parameter corrections for modified bases.

Many RNAs are known to be (heavily) modified post-trasncptionaly. The best known examples are tRNAs and rRNAs. To-date, more than 150 different modifications are listed in the MODOMICS database (<http://genesilico.pl/modomics/>) [5].

Many of the modified bases change the pairing behavior compared to their unmodified version, affecting not only the pairing partner preference, but also the resulting stability of the loops the base pairs may form.

Here, we provide a simple soft constraints callback implementation to correct for some well known modified bases where energy parameters are available for. This mechanism also supports arbitrary new modified base energy parameters supplied in JSON format (see [JSON Parameter Files for Modified Bases](#) for details). Collaboration diagram for Post-transcriptional Modifications:

Files

- file [soft_special.h](#)

Specialized implementations that utilize the soft constraint callback mechanism.

Typedefs

- typedef struct [vrna_sc_mod_param_s](#) * [vrna_sc_mod_param_t](#)

Modified base parameter data structure.

Functions

- [vrna_sc_mod_param_t vrna_sc_mod_read_from_jsonfile](#) (const char *filename, [vrna_md_t](#) *md)
Parse and extract energy parameters for a modified base from a JSON file.
- [vrna_sc_mod_param_t vrna_sc_mod_read_from_json](#) (const char *json, [vrna_md_t](#) *md)
Parse and extract energy parameters for a modified base from a JSON string.
- void [vrna_sc_mod_parameters_free](#) ([vrna_sc_mod_param_t](#) params)
Release memory occupied by a modified base parameter data structure.
- int [vrna_sc_mod_json](#) ([vrna_fold_compound_t](#) *fc, const char *json, const unsigned int *modification_sites)
Prepare soft constraint callbacks for modified base as specified in JSON string.
- int [vrna_sc_mod_jsonfile](#) ([vrna_fold_compound_t](#) *fc, const char *json_file, const unsigned int *modification_sites)
Prepare soft constraint callbacks for modified base as specified in JSON string.
- int [vrna_sc_mod](#) ([vrna_fold_compound_t](#) *fc, const [vrna_sc_mod_param_t](#) params, const unsigned int *modification_sites)
Prepare soft constraint callbacks for modified base as specified in JSON string.
- int [vrna_sc_mod_m6A](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for N6-methyl-adenosine (m6A)
- int [vrna_sc_mod_pseudouridine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for Pseudouridine.
- int [vrna_sc_mod_inosine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for Inosine.
- int [vrna_sc_mod_7DA](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for 7-deaza-adenosine (7DA)
- int [vrna_sc_mod_purine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for Purine (a.k.a. nebularine)
- int [vrna_sc_mod_dihydrouridine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for dihydrouridine.

16.50.2 Typedef Documentation

16.50.2.1 [vrna_sc_mod_param_t](#)

```
typedef struct vrna\_sc\_mod\_param\_s* vrna\_sc\_mod\_param\_t
#include <ViennaRNA/constraints/soft\_special.h>
```

Modified base parameter data structure.

See also

[vrna_sc_mod_read_from_jsonfile\(\)](#), [vrna_sc_mod_read_from_json\(\)](#), [vrna_sc_mod\(\)](#)

16.50.3 Function Documentation

16.50.3.1 [vrna_sc_mod_read_from_jsonfile\(\)](#)

```
vrna\_sc\_mod\_param\_t vrna\_sc\_mod\_read\_from\_jsonfile (
    const char * filename,
    vrna\_md\_t * md )
```

```
#include <ViennaRNA/constraints/soft\_special.h>
```

Parse and extract energy parameters for a modified base from a JSON file.

See also

[vrna_sc_mod_read_from_json\(\)](#), [vrna_sc_mod_parameters_free\(\)](#), [vrna_sc_mod\(\)](#), [JSON Parameter Files for Modified Bases](#)

Parameters

<i>filename</i>	The JSON file containing the specifications of the modified base
<i>md</i>	A model-details data structure (for look-up of canonical base pairs)

Returns

Parameters of the modified base

SWIG Wrapper Notes This function is available as an overloaded function `sc_mod_read_from_jsonfile()` where the `md` parameter may be omitted

16.50.3.2 vrna_sc_mod_read_from_json()

```
vrna_sc_mod_param_t vrna_sc_mod_read_from_json (
    const char * json,
    vrna_md_t * md )
#include <ViennaRNA/constraints/soft_special.h>
Parse and extract energy parameters for a modified base from a JSON string.
```

See also

[vrna_sc_mod_read_from_jsonfile\(\)](#), [vrna_sc_mod_parameters_free\(\)](#), [vrna_sc_mod\(\)](#), [JSON Parameter Files for Modified Base](#)

Parameters

<i>filename</i>	The JSON file containing the specifications of the modified base
<i>md</i>	A model-details data structure (for look-up of canonical base pairs)

Returns

Parameters of the modified base

SWIG Wrapper Notes This function is available as an overloaded function `sc_mod_read_from_json()` where the `md` parameter may be omitted

16.50.3.3 vrna_sc_mod_parameters_free()

```
void vrna_sc_mod_parameters_free (
    vrna_sc_mod_param_t params )
#include <ViennaRNA/constraints/soft_special.h>
Release memory occupied by a modified base parameter data structure.
Properly free a vrna\_sc\_mod\_param\_t data structure
```

Parameters

<i>params</i>	The data structure to free
---------------	----------------------------

SWIG Wrapper Notes This function is available as function `sc_mod_parameters_free()`

16.50.3.4 `vrna_sc_mod_json()`

```
int vrna_sc_mod_json (
    vrna_fold_compound_t * fc,
    const char * json,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Prepare soft constraint callbacks for modified base as specified in JSON string.

This function prepares all requirements to acknowledge modified bases as specified in the provided `json` string.

All subsequent predictions will treat each modification site special and adjust energy contributions if necessary.

See also

[vrna_sc_mod_jsonfile\(\)](#), [vrna_sc_mod\(\)](#), [vrna_sc_mod_m6A\(\)](#), [vrna_sc_mod_pseudouridine\(\)](#), [vrna_sc_mod_inosine\(\)](#), [vrna_sc_mod_7DA\(\)](#), [vrna_sc_mod_purine\(\)](#), [vrna_sc_mod_dihydrouridine\(\)](#), [JSON Parameter Files for Modified Bases](#)

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>json</i>	The JSON formatted string with the modified base parameters
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

SWIG Wrapper Notes This function is attached as method `sc_mod_json()` to objects of type *fold_compound*

16.50.3.5 `vrna_sc_mod_jsonfile()`

```
int vrna_sc_mod_jsonfile (
    vrna_fold_compound_t * fc,
    const char * json_file,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Prepare soft constraint callbacks for modified base as specified in JSON string.

Similar to [vrna_sc_mod_json\(\)](#), this function prepares all requirements to acknowledge modified bases as specified in the provided `json` file. All subsequent predictions will treat each modification site special and adjust energy contributions if necessary.

See also

[vrna_sc_mod_json\(\)](#), [vrna_sc_mod\(\)](#), [vrna_sc_mod_m6A\(\)](#), [vrna_sc_mod_pseudouridine\(\)](#), [vrna_sc_mod_inosine\(\)](#), [vrna_sc_mod_7DA\(\)](#), [vrna_sc_mod_purine\(\)](#), [vrna_sc_mod_dihydrouridine\(\)](#), [JSON Parameter Files for Modified Bases](#)

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>json</i>	The JSON formatted string with the modified base parameters
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

SWIG Wrapper Notes This function is attached as method `sc_mod_jsonfile()` to objects of type *fold_compound*

16.50.3.6 vrna_sc_mod()

```
int vrna_sc_mod (
    vrna_fold_compound_t * fc,
    const vrna_sc_mod_param_t params,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Prepare soft constraint callbacks for modified base as specified in JSON string.

This function takes a `vrna_sc_mod_param_t` data structure as obtained from `vrna_sc_mod_read_from_json()` or `vrna_sc_mod_read_from_jsonfile()` and prepares all requirements to acknowledge modified bases as specified in the provided `params` data structure. All subsequent predictions will treat each modification site special and adjust energy contributions if necessary.

See also

`vrna_sc_mod_read_from_json()`, `vrna_sc_mod_read_from_jsonfile()`, `vrna_sc_mod_json()`, `vrna_sc_mod_jsonfile()`, `vrna_sc_mod_m6A()`, `vrna_sc_mod_pseudouridine()`, `vrna_sc_mod_inosine()`, `vrna_sc_mod_7DA()`, `vrna_sc_mod_purine()`, `vrna_sc_mod_dihydrouridine()`

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>json</i>	The JSON formatted string with the modified base parameters
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method `sc_mod()` to objects of type `fold_compound`

16.50.3.7 vrna_sc_mod_m6A()

```
int vrna_sc_mod_m6A (
    vrna_fold_compound_t * fc,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Add soft constraint callbacks for N6-methyl-adenosine (m6A)

This is a convenience wrapper to add support for m6A using the soft constraint callback mechanism. Modification sites are provided as a list of sequence positions (1-based). Energy parameter corrections are derived from [18].

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method `sc_mod_m6A()` to objects of type `fold_compound`

16.50.3.8 vrna_sc_mod_pseudouridine()

```
int vrna_sc_mod_pseudouridine (
    vrna_fold_compound_t * fc,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Add soft constraint callbacks for Pseudouridine.

This is a convenience wrapper to add support for pseudouridine using the soft constraint callback mechanism. Modification sites are provided as a list of sequence positions (1-based). Energy parameter corrections are derived from [16].

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_mod_pseudouridine()** to objects of type *fold_compound*

16.50.3.9 vrna_sc_mod_inosine()

```
int vrna_sc_mod_inosine (
    vrna_fold_compound_t * fc,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Add soft constraint callbacks for Inosine.

This is a convenience wrapper to add support for inosine using the soft constraint callback mechanism. Modification sites are provided as a list of sequence positions (1-based). Energy parameter corrections are derived from [32] and [31].

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_mod_inosine()** to objects of type *fold_compound*

16.50.3.10 vrna_sc_mod_7DA()

```
int vrna_sc_mod_7DA (
    vrna_fold_compound_t * fc,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Add soft constraint callbacks for 7-deaza-adenosine (7DA)

This is a convenience wrapper to add support for 7-deaza-adenosine using the soft constraint callback mechanism. Modification sites are provided as a list of sequence positions (1-based). Energy parameter corrections are derived from [25].

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_mod_7DA()** to objects of type *fold_compound*

16.50.3.11 vrna_sc_mod_purine()

```
int vrna_sc_mod_purine (
    vrna_fold_compound_t * fc,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Add soft constraint callbacks for Purine (a.k.a. nebularine)

This is a convenience wrapper to add support for Purine using the soft constraint callback mechanism. Modification sites are provided as a list of sequence positions (1-based). Energy parameter corrections are derived from [17].

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method **sc_mod_purine()** to objects of type *fold_compound*

16.50.3.12 vrna_sc_mod_dihydrouridine()

```
int vrna_sc_mod_dihydrouridine (
    vrna_fold_compound_t * fc,
    const unsigned int * modification_sites )
#include <ViennaRNA/constraints/soft_special.h>
```

Add soft constraint callbacks for dihydrouridine.

This is a convenience wrapper to add support for dihydrouridine using the soft constraint callback mechanism. Modification sites are provided as a list of sequence positions (1-based). This implementation simply assumes that dihydrouridines favor destacking and destabilize base pair stacks by at least 1.5kcal/mol, as suggested in [7].

Parameters

<i>fc</i>	The fold_compound the corrections should be bound to
<i>modification_sites</i>	A list of modification site, i.e. positions that contain the modified base (1-based, last element in the list indicated by 0)

Returns

Non-zero if corrections have been added to the fold_compound, 0 otherwise

SWIG Wrapper Notes This function is attached as method `sc_mod_dihydrouridine()` to objects of type `fold_↔
compound`

16.51 Utilities

16.51.1 Detailed Description

Collaboration diagram for Utilities:

Modules

- [Utilities to deal with Nucleotide Alphabets](#)
Functions to cope with various aspects related to the nucleotide sequence alphabet.
- [\(Nucleic Acid Sequence\) String Utilitites](#)
Functions to parse, convert, manipulate, create, and compare (nucleic acid sequence) strings.
- [Secondary Structure Utilities](#)
Functions to create, parse, convert, manipulate, and compare secondary structure representations.
- [Multiple Sequence Alignment Utilities](#)
Functions to extract features from and to manipulate multiple sequence alignments.
- [Files and I/O](#)
Functions to parse, write, and convert various file formats and to deal with file system related issues.
- [Plotting](#)
Functions for Creating Secondary Structure Plots, Dot-Plots, and More.
- [Search Algorithms](#)
Implementations of various search algorithms to detect strings of objects within other strings of objects.
- [Combinatorics Algorithms](#)
Implementations to solve various combinatorial aspects for strings of objects.
- [\(Abstract\) Data Structures](#)
All datastructures and typedefs shared among the ViennaRNA Package can be found here.
- [Messages](#)
Functions to print various kind of messages.
- [Unit Conversion](#)
Functions to convert between various physical units.

Files

- file [alphabet.h](#)
Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.
- file [combinatorics.h](#)
Various implementations that deal with combinatorial aspects of objects.
- file [commands.h](#)
Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.
- file [sequence.h](#)
Functions and data structures related to sequence representations ,.
- file [file_formats_msa.h](#)
Functions dealing with file formats for Multiple Sequence Alignments (MSA)
- file [utils.h](#)
Several utilities for file handling.
- file [utils.h](#)

- *Various utilities to assist in plotting secondary structures and consensus structures.*
- file [alignments.h](#)
Various utility- and helper-functions for sequence alignments and comparative structure prediction.
- file [basic.h](#)
General utility- and helper-functions used throughout the ViennaRNA Package.
- file [strings.h](#)
General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.
- file [units.h](#)
Physical Units and Functions to convert them into each other.
- file [BoyerMoore.h](#)
Variants of the Boyer-Moore string search algorithm.
- file [char_stream.h](#)
Implementation of a dynamic, buffered character stream.
- file [stream_output.h](#)
An implementation of a buffered, ordered stream output data structure.

Macros

- **#define VRNA_INPUT_ERROR 1U**
Output flag of [get_input_line\(\)](#): "An ERROR has occurred, maybe EOF".
- **#define VRNA_INPUT_QUIT 2U**
Output flag of [get_input_line\(\)](#): "the user requested quitting the program".
- **#define VRNA_INPUT_MISC 4U**
Output flag of [get_input_line\(\)](#): "something was read".
- **#define VRNA_INPUT_FASTA_HEADER 8U**
*Input/Output flag of [get_input_line\(\)](#):
if used as input option this tells [get_input_line\(\)](#) that the data to be read should comply with the FASTA format.*
- **#define VRNA_INPUT_CONSTRAINT 32U**
*Input flag for [get_input_line\(\)](#):
Tell [get_input_line\(\)](#) that we assume to read a structure constraint.*
- **#define VRNA_INPUT_NO_TRUNCATION 256U**
Input switch for [get_input_line\(\)](#): "do not truncate the line by eliminating white spaces at end of line".
- **#define VRNA_INPUT_NO_REST 512U**
Input switch for [vrna_file_fasta_read_record\(\)](#): "do fill rest array".
- **#define VRNA_INPUT_NO_SPAN 1024U**
Input switch for [vrna_file_fasta_read_record\(\)](#): "never allow data to span more than one line".
- **#define VRNA_INPUT_NOSKIP_BLANK_LINES 2048U**
Input switch for [vrna_file_fasta_read_record\(\)](#): "do not skip empty lines".
- **#define VRNA_INPUT_BLANK_LINE 4096U**
Output flag for [vrna_file_fasta_read_record\(\)](#): "read an empty line".
- **#define VRNA_INPUT_NOSKIP_COMMENTS 128U**
Input switch for [get_input_line\(\)](#): "do not skip comment lines".
- **#define VRNA_INPUT_COMMENT 8192U**
Output flag for [vrna_file_fasta_read_record\(\)](#): "read a comment".
- **#define MIN2(A, B) ((A) < (B) ? (A) : (B))**
Get the minimum of two comparable values.
- **#define MAX2(A, B) ((A) > (B) ? (A) : (B))**
Get the maximum of two comparable values.
- **#define MIN3(A, B, C) (MIN2((MIN2((A), (B))), (C)))**
Get the minimum of three comparable values.
- **#define MAX3(A, B, C) (MAX2((MAX2((A), (B))), (C)))**
Get the maximum of three comparable values.

Functions

- void * [vrna_alloc](#) (unsigned size)
Allocate space safely.
- void * [vrna_realloc](#) (void *p, unsigned size)
Reallocate space safely.
- void [vrna_init_rand](#) (void)
Initialize seed for random number generator.
- void [vrna_init_rand_seed](#) (unsigned int seed)
Initialize the random number generator with a pre-defined seed.
- double [vrna_urn](#) (void)
get a random number from [0..1]
- int [vrna_int_urn](#) (int from, int to)
Generates a pseudo random integer in a specified range.
- char * [vrna_time_stamp](#) (void)
Get a timestamp.
- unsigned int [get_input_line](#) (char **string, unsigned int options)
- int * [vrna_idx_row_wise](#) (unsigned int length)
Get an index mapper array (iidx) for accessing the energy matrices, e.g. in partition function related functions.
- int * [vrna_idx_col_wise](#) (unsigned int length)
Get an index mapper array (idx) for accessing the energy matrices, e.g. in MFE related functions.

Variables

- unsigned short [xsubi](#) [3]
Current 48 bit random number.

16.51.2 Macro Definition Documentation

16.51.2.1 VRNA_INPUT_FASTA_HEADER

```
#define VRNA_INPUT_FASTA_HEADER 8U
```

#include <[ViennaRNA/utils/basic.h](#)>

Input/Output flag of [get_input_line\(\)](#):
if used as input option this tells [get_input_line\(\)](#) that the data to be read should comply with the FASTA format.
the function will return this flag if a fasta header was read

16.51.2.2 VRNA_INPUT_CONSTRAINT

```
#define VRNA_INPUT_CONSTRAINT 32U
```

#include <[ViennaRNA/utils/basic.h](#)>

Input flag for [get_input_line\(\)](#):
Tell [get_input_line\(\)](#) that we assume to read a structure constraint.

16.51.3 Function Documentation

16.51.3.1 vrna_alloc()

```
void * vrna_alloc (
    unsigned size )
```

#include <[ViennaRNA/utils/basic.h](#)>

Allocate space safely.

Parameters

<i>size</i>	The size of the memory to be allocated in bytes
-------------	---

Returns

A pointer to the allocated memory

16.51.3.2 vrna_realloc()

```
void * vrna_realloc (
    void * p,
    unsigned size )
#include <ViennaRNA/utils/basic.h>
Reallocate space safely.
```

Parameters

<i>p</i>	A pointer to the memory region to be reallocated
<i>size</i>	The size of the memory to be allocated in bytes

Returns

A pointer to the newly allocated memory

16.51.3.3 vrna_init_rand()

```
void vrna_init_rand (
    void )
#include <ViennaRNA/utils/basic.h>
Initialize seed for random number generator.
```

See also

[vrna_init_rand_seed\(\)](#), [vrna_urn\(\)](#)

16.51.3.4 vrna_init_rand_seed()

```
void vrna_init_rand_seed (
    unsigned int seed )
#include <ViennaRNA/utils/basic.h>
Initialize the random number generator with a pre-defined seed.
```

See also

[vrna_init_rand\(\)](#), [vrna_urn\(\)](#)

Parameters

<i>seed</i>	The seed for the random number generator
-------------	--

SWIG Wrapper Notes This function is available as an overloaded function [init_rand\(\)](#) where the argument *seed* is optional.

16.51.3.5 `vrna_urn()`

```
double vrna_urn (
    void )
#include <ViennaRNA/utils/basic.h>
get a random number from [0..1]
```

See also

[vrna_int_urn\(\)](#), [vrna_init_rand\(\)](#), [vrna_init_rand_seed\(\)](#)

Note

Usually implemented by calling *erand48()*.

Returns

A random number in range [0..1]

16.51.3.6 `vrna_int_urn()`

```
int vrna_int_urn (
    int from,
    int to )
#include <ViennaRNA/utils/basic.h>
Generates a pseudo random integer in a specified range.
```

See also

[vrna_urn\(\)](#), [vrna_init_rand\(\)](#)

Parameters

<i>from</i>	The first number in range
<i>to</i>	The last number in range

Returns

A pseudo random number in range [from, to]

16.51.3.7 `vrna_time_stamp()`

```
char * vrna_time_stamp (
    void )
#include <ViennaRNA/utils/basic.h>
Get a timestamp.
Returns a string containing the current date in the format
```

```
Fri Mar 19 21:10:57 1993
```

Returns

A string containing the timestamp

16.51.3.8 `get_input_line()`

```
unsigned int get_input_line (
    char ** string,
    unsigned int options )
#include <ViennaRNA/utils/basic.h>
```

Retrieve a line from 'stdin' safely while skipping comment characters and other features This function returns the type of input it has read if recognized. An option argument allows one to switch between different reading modes.

Currently available options are:

[VRNA_INPUT_COMMENT](#), [VRNA_INPUT_NOSKIP_COMMENTS](#), [VRNA_INPUT_NO_TRUNCATION](#)

pass a collection of options as one value like this:

```
get_input_line(string, option_1 | option_2 | option_n)
```

If the function recognizes the type of input, it will report it in the return value. It also reports if a user defined 'quit' command (-sign on 'stdin') was given. Possible return values are:

[VRNA_INPUT_FASTA_HEADER](#), [VRNA_INPUT_ERROR](#), [VRNA_INPUT_MISC](#), [VRNA_INPUT_QUIT](#)

Parameters

<i>string</i>	A pointer to the character array that contains the line read
<i>options</i>	A collection of options for switching the functions behavior

Returns

A flag with information about what has been read

16.51.3.9 `vrna_idx_row_wise()`

```
int * vrna_idx_row_wise (
    unsigned int length )
#include <ViennaRNA/utils/basic.h>
```

Get an index mapper array (iindx) for accessing the energy matrices, e.g. in partition function related functions.

Access of a position "(i,j)" is then accomplished by using

```
(i,j) ~ iindx[i]-j
```

This function is necessary as most of the two-dimensional energy matrices are actually one-dimensional arrays throughout the ViennaRNA Package

Consult the implemented code to find out about the mapping formula ;)

See also

[vrna_idx_col_wise\(\)](#)

Parameters

<i>length</i>	The length of the RNA sequence
---------------	--------------------------------

Returns

The mapper array

16.51.3.10 `vrna_idx_col_wise()`

```
int * vrna_idx_col_wise (
    unsigned int length )
```

```
#include <ViennaRNA/utils/basic.h>
```

Get an index mapper array (indx) for accessing the energy matrices, e.g. in MFE related functions.
Access of a position "(i,j)" is then accomplished by using

```
(i,j) ~ indx[j]+i
```

This function is necessary as most of the two-dimensional energy matrices are actually one-dimensional arrays throughout the ViennaRNAPackage

Consult the implemented code to find out about the mapping formula ;)

See also

[vrna_idx_row_wise\(\)](#)

Parameters

<i>length</i>	The length of the RNA sequence
---------------	--------------------------------

Returns

The mapper array

16.51.4 Variable Documentation

16.51.4.1 xsubi

```
unsigned short xsubi[3] [extern]
```

```
#include <ViennaRNA/utils/basic.h>
```

Current 48 bit random number.

This variable is used by [vrna_urn\(\)](#). These should be set to some random number seeds before the first call to [vrna_urn\(\)](#).

See also

[vrna_urn\(\)](#)

16.52 Exterior Loops

Functions to evaluate the free energy contributions for exterior loops.

16.52.1 Detailed Description

Functions to evaluate the free energy contributions for exterior loops.

Collaboration diagram for Exterior Loops:

Files

- file [external.h](#)

Energy evaluation of exterior loops for MFE and partition function calculations.

Boltzmann weight (partition function) interface

- typedef struct vrna_mx_pf_aux_el_s * [vrna_mx_pf_aux_el_t](#)

Auxiliary helper arrays for fast exterior loop computations.

- [FLT_OR_DBL vrna_exp_E_ext_stem](#) (unsigned int type, int n5d, int n3d, [vrna_exp_param_t](#) *p)

Evaluate a stem branching off the exterior loop (Boltzmann factor version)

- `vrna_mx_pf_aux_el_t vrna_exp_E_ext_fast_init (vrna_fold_compound_t *fc)`
- `void vrna_exp_E_ext_fast_rotate (vrna_mx_pf_aux_el_t aux_mx)`
- `void vrna_exp_E_ext_fast_free (vrna_mx_pf_aux_el_t aux_mx)`
- `FLT_OR_DBL vrna_exp_E_ext_fast (vrna_fold_compound_t *fc, int i, int j, vrna_mx_pf_aux_el_t aux_mx)`
- `void vrna_exp_E_ext_fast_update (vrna_fold_compound_t *fc, int j, vrna_mx_pf_aux_el_t aux_mx)`

Basic free energy interface

- `int vrna_E_ext_stem` (unsigned int type, int n5d, int n3d, `vrna_param_t` *p)
Evaluate a stem branching off the exterior loop.
- `int vrna_eval_ext_stem` (`vrna_fold_compound_t` *fc, int i, int j)
Evaluate the free energy of a base pair in the exterior loop.
- `int vrna_E_ext_loop_5` (`vrna_fold_compound_t` *fc)
- `int vrna_E_ext_loop_3` (`vrna_fold_compound_t` *fc, int i)

16.52.2 Typedef Documentation

16.52.2.1 vrna_mx_pf_aux_el_t

```
typedef struct vrna_mx_pf_aux_el_s* vrna_mx_pf_aux_el_t
#include <ViennaRNA/loops/external.h>
```

Auxiliary helper arrays for fast exterior loop computations.

See also

`vrna_exp_E_ext_fast_init()`, `vrna_exp_E_ext_fast_rotate()`, `vrna_exp_E_ext_fast_free()`, `vrna_exp_E_ext_fast()`

16.52.3 Function Documentation

16.52.3.1 vrna_E_ext_stem()

```
int vrna_E_ext_stem (
    unsigned int type,
    int n5d,
    int n3d,
    vrna_param_t * p )
#include <ViennaRNA/loops/external.h>
```

Evaluate a stem branching off the exterior loop.

Given a base pair (i, j) encoded by *type*, compute the energy contribution including dangling-end/terminal-mismatch contributions. Instead of returning the energy contribution per-se, this function returns the corresponding Boltzmann factor. If either of the adjacent nucleotides $(i - 1)$ and $(j + 1)$ must not contribute stacking energy, the corresponding encoding must be -1 .

See also

`vrna_E_exp_stem()`

Parameters

<i>type</i>	The base pair encoding
<i>n5d</i>	The encoded nucleotide directly adjacent at the 5' side of the base pair (may be -1)
<i>n3d</i>	The encoded nucleotide directly adjacent at the 3' side of the base pair (may be -1)
<i>p</i>	The pre-computed energy parameters

Returns

The energy contribution of the introduced exterior-loop stem

16.52.3.2 `vrna_eval_ext_stem()`

```
int vrna_eval_ext_stem (
    vrna_fold_compound_t * fc,
    int i,
    int j )
```

```
#include <ViennaRNA/loops/external.h>
```

Evaluate the free energy of a base pair in the exterior loop.

Evaluate the free energy of a base pair connecting two nucleotides in the exterior loop and take hard constraints into account.

Typically, this is simply dangling end contributions of the adjacent nucleotides, potentially a terminal A-U mismatch penalty, and maybe some generic soft constraint contribution for that decomposition.

Note

For dangles == 1 || 3 this function also evaluates the three additional pairs $(i + 1, j)$, $(i, j - 1)$, and $(i + 1, j - 1)$ and returns the minimum for all four possibilities in total.

Parameters

<i>fc</i>	Fold compound to work on (defines the model and parameters)
<i>i</i>	5' position of the base pair
<i>j</i>	3' position of the base pair

Returns

Free energy contribution that arises when this pair is formed in the exterior loop

16.52.3.3 `vrna_exp_E_ext_stem()`

```
FLT_OR_DBL vrna_exp_E_ext_stem (
    unsigned int type,
    int n5d,
    int n3d,
    vrna_exp_param_t * p )
```

```
#include <ViennaRNA/loops/external.h>
```

Evaluate a stem branching off the exterior loop (Boltzmann factor version)

Given a base pair (i, j) encoded by *type*, compute the energy contribution including dangling-end/terminal-mismatch contributions. Instead of returning the energy contribution per-se, this function returns the corresponding Boltzmann factor. If either of the adjacent nucleotides $(i - 1)$ and $(j + 1)$ must not contribute stacking energy, the corresponding encoding must be -1 .

See also

[vrna_E_ext_stem\(\)](#)

Parameters

<i>type</i>	The base pair encoding
<i>n5d</i>	The encoded nucleotide directly adjacent at the 5' side of the base pair (may be -1)
<i>n3d</i>	The encoded nucleotide directly adjacent at the 3' side of the base pair (may be -1)
<i>p</i>	The pre-computed energy parameters (Boltzmann factor version)

Returns

The Boltzmann weighted energy contribution of the introduced exterior-loop stem

16.53 Hairpin Loops

Functions to evaluate the free energy contributions for hairpin loops.

16.53.1 Detailed Description

Functions to evaluate the free energy contributions for hairpin loops.

Collaboration diagram for Hairpin Loops:

Files

- file [hairpin.h](#)
Energy evaluation of hairpin loops for MFE and partition function calculations.

Basic free energy interface

- int [vrna_E_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate the free energy of a hairpin loop and consider hard constraints if they apply.
- int [vrna_E_ext_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate the free energy of an exterior hairpin loop and consider possible hard constraints.
- int [vrna_eval_ext_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate free energy of an exterior hairpin loop.
- int [vrna_eval_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate free energy of a hairpin loop.
- PRIVATE int [E_Hairpin](#) (int size, int type, int si1, int sj1, const char *string, [vrna_param_t](#) *P)
Compute the Energy of a hairpin-loop.

Boltzmann weight (partition function) interface

- PRIVATE [FLT_OR_DBL exp_E_Hairpin](#) (int u, int type, short si1, short sj1, const char *string, [vrna_exp_param_t](#) *P)
Compute Boltzmann weight $e^{-\Delta G/kT}$ of a hairpin loop.
- [FLT_OR_DBL vrna_exp_E_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
High-Level function for hairpin loop energy evaluation (partition function variant)

16.53.2 Function Documentation

16.53.2.1 vrna_E_hp_loop()

```
int vrna_E_hp_loop (
    vrna\_fold\_compound\_t * fc,
    int i,
    int j )
```

```
#include <ViennaRNA/loops/hairpin.h>
```

Evaluate the free energy of a hairpin loop and consider hard constraints if they apply.

This function evaluates the free energy of a hairpin loop

In case the base pair is not allowed due to a constraint conflict, this function returns INF.

Note

This function is polymorphic! The provided [vrna_fold_compound_t](#) may be of type [VRNA_FC_TYPE_SINGLE](#) or [VRNA_FC_TYPE_COMPARATIVE](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t that stores all relevant model settings
<i>i</i>	The 5' nucleotide of the base pair (3' to evaluate the pair as exterior hairpin loop)
<i>j</i>	The 3' nucleotide of the base pair (5' to evaluate the pair as exterior hairpin loop)

Returns

The free energy of the hairpin loop in 10cal/mol

16.53.2.2 `vrna_E_ext_hp_loop()`

```
int vrna_E_ext_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
```

Evaluate the free energy of an exterior hairpin loop and consider possible hard constraints.

Note

This function is polymorphic! The provided [vrna_fold_compound_t](#) may be of type [VRNA_FC_TYPE_SINGLE](#) or [VRNA_FC_TYPE_COMPARATIVE](#)

16.53.2.3 `vrna_eval_hp_loop()`

```
int vrna_eval_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
```

Evaluate free energy of a hairpin loop.

Note

This function is polymorphic! The provided [vrna_fold_compound_t](#) may be of type [VRNA_FC_TYPE_SINGLE](#) or [VRNA_FC_TYPE_COMPARATIVE](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t for the particular energy evaluation
<i>i</i>	5'-position of the base pair
<i>j</i>	3'-position of the base pair

Returns

Free energy of the hairpin loop closed by (*i*, *j*) in deka-kal/mol

SWIG Wrapper Notes This function is attached as method `eval_hp_loop()` to objects of type `fold_compound`

16.53.2.4 `E_Hairpin()`

```
PRIVATE int E_Hairpin (
    int size,
```

```

        int type,
        int si1,
        int sj1,
        const char * string,
        vrna_param_t * P )
#include <ViennaRNA/loops/hairpin.h>

```

Compute the Energy of a hairpin-loop.

To evaluate the free energy of a hairpin-loop, several parameters have to be known. A general hairpin-loop has this structure:

```

      a3 a4
a2      a5
a1      a6
  X - Y
  |   |
 5'   3'

```

where X-Y marks the closing pair [e.g. a (**G,C**) pair]. The length of this loop is 6 as there are six unpaired nucleotides (a1-a6) enclosed by (X,Y). The 5' mismatching nucleotide is a1 while the 3' mismatch is a6. The nucleotide sequence of this loop is "a1.a2.a3.a4.a5.a6"

Note

The parameter sequence should contain the sequence of the loop in capital letters of the nucleic acid alphabet if the loop size is below 7. This is useful for unusually stable tri-, tetra- and hexa-loops which are treated differently (based on experimental data) if they are tabulated.

See also

[scale_parameters\(\)](#)
[vrna_param_t](#)

Warning

Not (really) thread safe! A threadsafe implementation will replace this function in a future release!
 Energy evaluation may change due to updates in global variable "tetra_loop"

Parameters

<i>size</i>	The size of the loop (number of unpaired nucleotides)
<i>type</i>	The pair type of the base pair closing the hairpin
<i>si1</i>	The 5'-mismatching nucleotide
<i>sj1</i>	The 3'-mismatching nucleotide
<i>string</i>	The sequence of the loop (May be NULL, otherwise mst be at least <i>size</i> + 2 long)
<i>P</i>	The datastructure containing scaled energy parameters

Returns

The Free energy of the Hairpin-loop in dcal/mol

16.53.2.5 exp_E_Hairpin()

```

PRIVATE FLT_OR_DBL exp_E_Hairpin (
    int u,

```

```

    int type,
    short si1,
    short sj1,
    const char * string,
    vrna_exp_param_t * P )
#include <ViennaRNA/loops/hairpin.h>
Compute Boltzmann weight  $e^{-\Delta G/kT}$  of a hairpin loop.
multiply by scale[u+2]

```

See also

[get_scaled_pf_parameters\(\)](#)
[vrna_exp_param_t](#)
[E_Hairpin\(\)](#)

Warning

Not (really) thread safe! A threadsafe implementation will replace this function in a future release!
 Energy evaluation may change due to updates in global variable "tetra_loop"

Parameters

<i>u</i>	The size of the loop (number of unpaired nucleotides)
<i>type</i>	The pair type of the base pair closing the hairpin
<i>si1</i>	The 5'-mismatching nucleotide
<i>sj1</i>	The 3'-mismatching nucleotide
<i>string</i>	The sequence of the loop (May be NULL, otherwise mst be at least <i>size</i> + 2 long)
<i>P</i>	The datastructure containing scaled Boltzmann weights of the energy parameters

Returns

The Boltzmann weight of the Hairpin-loop

16.53.2.6 vrna_exp_E_hp_loop()

```

FLT_OR_DBL vrna_exp_E_hp_loop (
    vrna_fold_compound_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/hairpin.h>
High-Level function for hairpin loop energy evaluation (partition function variant)

```

See also

[vrna_E_hp_loop\(\)](#) for it's free energy counterpart

Note

This function is polymorphic! The provided [vrna_fold_compound_t](#) may be of type [VRNA_FC_TYPE_SINGLE](#) or [VRNA_FC_TYPE_COMPARATIVE](#)

16.54 Internal Loops

Functions to evaluate the free energy contributions for internal loops.

16.54.1 Detailed Description

Functions to evaluate the free energy contributions for internal loops.
Collaboration diagram for Internal Loops:

Files

- file [internal.h](#)
Energy evaluation of interior loops for MFE and partition function calculations.

Basic free energy interface

- int [vrna_E_int_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
- int [vrna_eval_int_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int k, int l)
Evaluate the free energy contribution of an interior loop with delimiting base pairs (i, j) and (k, l) .
- int [vrna_E_ext_int_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int *ip, int *iq)
- int [vrna_E_stack](#) ([vrna_fold_compound_t](#) *fc, int i, int j)

Boltzmann weight (partition function) interface

- [FLT_OR_DBL vrna_exp_E_int_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
- [FLT_OR_DBL vrna_exp_E_interior_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int k, int l)

16.54.2 Function Documentation

16.54.2.1 vrna_eval_int_loop()

```
int vrna_eval_int_loop (
    vrna\_fold\_compound\_t * fc,
    int i,
    int j,
    int k,
    int l )
#include <ViennaRNA/loops/internal.h>
```

Evaluate the free energy contribution of an interior loop with delimiting base pairs (i, j) and (k, l) .

Note

This function is polymorphic, i.e. it accepts [vrna_fold_compound_t](#) of type [VRNA_FC_TYPE_SINGLE](#) as well as [VRNA_FC_TYPE_COMPARATIVE](#)

SWIG Wrapper Notes This function is attached as method [eval_int_loop\(\)](#) to objects of type *fold_compound*

16.55 Multibranch Loops

Functions to evaluate the free energy contributions for multibranch loops.

16.55.1 Detailed Description

Functions to evaluate the free energy contributions for multibranch loops.
Collaboration diagram for Multibranch Loops:

Files

- file [multibranch.h](#)
Energy evaluation of multibranch loops for MFE and partition function calculations.

Boltzmann weight (partition function) interface

- typedef struct vrna_mx_pf_aux_ml_s * [vrna_mx_pf_aux_ml_t](#)
Auxiliary helper arrays for fast exterior loop computations.
- [FLT_OR_DBL vrna_exp_E_mb_loop_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, [vrna_mx_pf_aux_ml_t](#) aux_mx)
- [vrna_mx_pf_aux_ml_t vrna_exp_E_ml_fast_init](#) ([vrna_fold_compound_t](#) *fc)
- void [vrna_exp_E_ml_fast_rotate](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- void [vrna_exp_E_ml_fast_free](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- const [FLT_OR_DBL](#) * [vrna_exp_E_ml_fast_qqm](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- const [FLT_OR_DBL](#) * [vrna_exp_E_ml_fast_qqm1](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- [FLT_OR_DBL vrna_exp_E_ml_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, [vrna_mx_pf_aux_ml_t](#) aux_mx)

Basic free energy interface

- int [vrna_E_mb_loop_stack](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate energy of a multi branch helices stacking onto closing pair (i,j)
- int [vrna_E_mb_loop_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int *dmli1, int *dmli2)
- int [E_ml_rightmost_stem](#) (int i, int j, [vrna_fold_compound_t](#) *fc)
- int [vrna_E_ml_stems_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int *fmi, int *dmli)

16.55.2 Typedef Documentation

16.55.2.1 vrna_mx_pf_aux_ml_t

```
typedef struct vrna_mx_pf_aux_ml_s* vrna\_mx\_pf\_aux\_ml\_t
#include <ViennaRNA/loops/multibranch.h>
```

Auxiliary helper arrays for fast exterior loop computations.

See also

[vrna_exp_E_ml_fast_init\(\)](#), [vrna_exp_E_ml_fast_rotate\(\)](#), [vrna_exp_E_ml_fast_free\(\)](#), [vrna_exp_E_ml_fast\(\)](#)

16.55.3 Function Documentation

16.55.3.1 vrna_E_mb_loop_stack()

```
int vrna\_E\_mb\_loop\_stack (
    vrna\_fold\_compound\_t * fc,
    int i,
    int j )
#include <ViennaRNA/loops/multibranch.h>
```

Evaluate energy of a multi branch helices stacking onto closing pair (i,j)
Computes total free energy for coaxial stacking of (i,j) with (i+1.k) or (k+1.j-1)

16.56 Partition Function for Two Hybridized Sequences

Partition Function Cofolding.

16.56.1 Detailed Description

Partition Function Cofolding.

To simplify the implementation the partition function computation is done internally in a null model that does not include the duplex initiation energy, i.e. the entropic penalty for producing a dimer from two monomers). The resulting free energies and pair probabilities are initially relative to that null model. In a second step the free energies can be corrected to include the dimerization penalty, and the pair probabilities can be divided into the conditional pair probabilities given that a re dimer is formed or not formed. See [2] for further details.

As for folding one RNA molecule, this computes the partition function of all possible structures and the base pair probabilities. Uses the same global [pf_scale](#) variable to avoid overflows.

After computing the partition functions of all possible dimers one can compute the probabilities of base pairs, the concentrations out of start concentrations and sofar and soaway.

Dimer formation is inherently concentration dependent. Given the free energies of the monomers A and B and dimers AB, AA, and BB one can compute the equilibrium concentrations, given input concentrations of A and B, see e.g. Dimitrov & Zuker (2004) Collaboration diagram for Partition Function for Two Hybridized Sequences:

Files

- file [concentrations.h](#)
Concentration computations for RNA-RNA interactions.
- file [part_func_up.h](#)
Implementations for accessibility and RNA-RNA interaction as a stepwise process.

Typedefs

- typedef struct [vrna_dimer_pf_s](#) [vrna_dimer_pf_t](#)
Typename for the data structure that stores the dimer partition functions, [vrna_dimer_pf_s](#), as returned by [vrna_pf_dimer\(\)](#)
- typedef struct [vrna_dimer_pf_s](#) [cofoldF](#)
Backward compatibility typedef for [vrna_dimer_pf_s](#).

Variables

- int **mirnatog**
Toggles no intrabp in 2nd mol.
- double **F_monomer** [2]
Free energies of the two monomers.
- typedef struct [vrna_dimer_conc_s](#) [vrna_dimer_conc_t](#)
Typename for the data structure that stores the dimer concentrations, [vrna_dimer_conc_s](#), as required by [vrna_pf_dimer_concentration\(\)](#)
- typedef struct [vrna_dimer_conc_s](#) **ConcEnt**
Backward compatibility typedef for [vrna_dimer_conc_s](#).
- [vrna_dimer_conc_t](#) * [vrna_pf_dimer_concentrations](#) (double FcAB, double FcAA, double FcBB, double FEA, double FEB, const double *startconc, const [vrna_exp_param_t](#) *exp_params)
Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.

Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- [vrna_dimer_pf_t](#) [vrna_pf_co_fold](#) (const char *seq, char *structure, [vrna_ep_t](#) **pl)
Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

16.56.2 Function Documentation

16.56.2.1 vrna_pf_co_fold()

```
vrna_dimer_pf_t vrna_pf_co_fold (
    const char * seq,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/part_func.h>
```

Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

This simplified interface to [vrna_pf_dimer\(\)](#) computes the partition function and, if required, base pair probabilities for an RNA-RNA interaction using default options. Memory required for dynamic programming (DP) matrices will be allocated and free'd on-the-fly. Hence, after return of this function, the recursively filled matrices are not available any more for any post-processing.

Note

In case you want to use the filled DP matrices for any subsequent post-processing step, or you require other conditions than specified by the default model details, use [vrna_pf_dimer\(\)](#), and the data structure [vrna_fold_compound_t](#) instead.

See also

[vrna_pf_dimer\(\)](#)

Parameters

<i>seq</i>	Two concatenated RNA sequences with a delimiting '&' in between
<i>structure</i>	A pointer to the character array where position-wise pairing propensity will be stored. (Maybe NULL)
<i>pl</i>	A pointer to a list of vrna_ep_t to store pairing probabilities (Maybe NULL)

Returns

[vrna_dimer_pf_t](#) structure containing a set of energies needed for concentration computations.

16.56.2.2 vrna_pf_dimer_concentrations()

```
vrna_dimer_conc_t * vrna_pf_dimer_concentrations (
    double FcAB,
    double FcAA,
    double FcBB,
    double FEA,
    double FEB,
    const double * startconc,
    const vrna_exp_param_t * exp_params )
#include <ViennaRNA/concentrations.h>
```

Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.

This function takes an array 'startconc' of input concentrations with alternating entries for the initial concentrations of molecules A and B (terminated by two zeroes), then computes the resulting equilibrium concentrations from the free energies for the dimers. Dimer free energies should be the dimer-only free energies, i.e. the FcAB entries from the [vrna_dimer_pf_t](#) struct.

Parameters

<i>FcAB</i>	Free energy of AB dimer (FcAB entry)
<i>FcAA</i>	Free energy of AA dimer (FcAB entry)
<i>FcBB</i>	Free energy of BB dimer (FcAB entry)

Parameters

<i>FEA</i>	Free energy of monomer A
<i>FEB</i>	Free energy of monomer B
<i>startconc</i>	List of start concentrations [a0],[b0],[a1],[b1],...,[an],[bn],[0],[0]
<i>exp_params</i>	The precomputed Boltzmann factors

Returns

vrna_dimer_conc_t array containing the equilibrium energies and start concentrations

16.57 Partition Function for two Hybridized Sequences as a Stepwise Process

RNA-RNA interaction as a stepwise process.

16.57.1 Detailed Description

RNA-RNA interaction as a stepwise process.

In this approach to cofolding the interaction between two RNA molecules is seen as a stepwise process. In a first step, the target molecule has to adopt a structure in which a binding site is accessible. In a second step, the ligand molecule will hybridize with a region accessible to an interaction. Consequently the algorithm is designed as a two step process: The first step is the calculation of the probability that a region within the target is unpaired, or equivalently, the calculation of the free energy needed to expose a region. In the second step we compute the free energy of an interaction for every possible binding site. Collaboration diagram for Partition Function for two Hybridized Sequences as a Stepwise Process:

Functions

- `pu_contrib * pf_unstru` (char *sequence, int max_w)
Calculate the partition function over all unpaired regions of a maximal length.
- `interact * pf_interact` (const char *s1, const char *s2, `pu_contrib *p_c`, `pu_contrib *p_c2`, int max_w, char *cstruc, int incr3, int incr5)
Calculates the probability of a local interaction between two sequences.
- void `free_interact` (`interact *pin`)
Frees the output of function `pf_interact()`.
- void `free_pu_contrib_struct` (`pu_contrib *pu`)
Frees the output of function `pf_unstru()`.

16.57.2 Function Documentation

16.57.2.1 pf_unstru()

```
pu_contrib * pf_unstru (
    char * sequence,
    int max_w )
#include <ViennaRNA/part_func_up.h>
```

Calculate the partition function over all unpaired regions of a maximal length.

You have to call function `pf_fold()` providing the same sequence before calling `pf_unstru()`. If you want to calculate unpaired regions for a constrained structure, set variable 'structure' in function '`pf_fold()`' to the constrain string. It returns a `pu_contrib` struct containing four arrays of dimension [i = 1 to length(sequence)][j = 0 to u-1] containing all possible contributions to the probabilities of unpaired regions of maximum length u. Each array in `pu_contrib` contains one of the contributions to the total probability of being unpaired: The probability of being unpaired within an

exterior loop is in array `pu_contrib->E`, the probability of being unpaired within a hairpin loop is in array `pu_contrib->H`, the probability of being unpaired within an interior loop is in array `pu_contrib->I` and probability of being unpaired within a multi-loop is in array `pu_contrib->M`. The total probability of being unpaired is the sum of the four arrays of `pu_contrib`.

This function frees everything allocated automatically. To free the output structure call `free_pu_contrib()`.

Parameters

<i>sequence</i>	
<i>max_w</i>	

Returns

16.57.2.2 pf_interact()

```
interact * pf_interact (
    const char * s1,
    const char * s2,
    pu_contrib * p_c,
    pu_contrib * p_c2,
    int max_w,
    char * cstruc,
    int incr3,
    int incr5 )
```

```
#include <ViennaRNA/part_func_up.h>
```

Calculates the probability of a local interaction between two sequences.

The function considers the probability that the region of interaction is unpaired within 's1' and 's2'. The longer sequence has to be given as 's1'. The shorter sequence has to be given as 's2'. Function `pf_unstru()` has to be called for 's1' and 's2', where the probabilities of being unpaired have to be given in 'p_c' and 'p_c2', respectively. If you do not want to include the probabilities of being unpaired for 's2' set 'p_c2' to NULL. If variable 'cstruc' is not NULL, constrained folding is done: The available constraints for intermolecular interaction are: '.' (no constrain), 'x' (the base has no intermolecular interaction) and '|' (the corresponding base has to be paired intermolecularly).

The parameter 'w' determines the maximal length of the interaction. The parameters 'incr5' and 'incr3' allows inclusion of unpaired residues left ('incr5') and right ('incr3') of the region of interaction in 's1'. If the 'incr' options are used, function `pf_unstru()` has to be called with $w = w + \text{incr5} + \text{incr3}$ for the longer sequence 's1'.

It returns a structure of type `interact` which contains the probability of the best local interaction including residue i in P_i and the minimum free energy in G_i , where i is the position in sequence 's1'. The member `Gikjl` of structure `interact` is the best interaction between region $[k,i]$ $k < i$ in longer sequence 's1' and region $[j,l]$ $j < l$ in 's2'. `Gikjl_wo` is `Gikjl` without the probability of being unpaired.

Use `free_interact()` to free the returned structure, all other stuff is freed inside `pf_interact()`.

Parameters

<i>s1</i>	
<i>s2</i>	
<i>p_c</i>	
<i>p_c2</i>	
<i>max_w</i>	
<i>cstruc</i>	
<i>incr3</i>	
<i>incr5</i>	

Returns

16.58 Reading/Writing Energy Parameter Sets from/to File

Read and Write energy parameter sets from and to files or strings.

16.58.1 Detailed Description

Read and Write energy parameter sets from and to files or strings.

Collaboration diagram for Reading/Writing Energy Parameter Sets from/to File:

Modules

- [Converting Energy Parameter Files](#)
Convert energy parameter files into the latest format.

Macros

- `#define VRNA_PARAMETER_FORMAT_DEFAULT 0`
Default Energy Parameter File format.

Functions

- `int vrna_params_load` (const char fname[], unsigned int options)
Load energy parameters from a file.
- `int vrna_params_save` (const char fname[], unsigned int options)
Save energy parameters to a file.
- `int vrna_params_load_from_string` (const char *string, const char *name, unsigned int options)
Load energy paramters from string.
- `int vrna_params_load_defaults` (void)
Load default RNA energy parameter set.
- `int vrna_params_load_RNA_Turner2004` (void)
Load Turner 2004 RNA energy parameter set.
- `int vrna_params_load_RNA_Turner1999` (void)
Load Turner 1999 RNA energy parameter set.
- `int vrna_params_load_RNA_Andronescu2007` (void)
Load Andronsecu 2007 RNA energy parameter set.
- `int vrna_params_load_RNA_Langdon2018` (void)
Load Langdon 2018 RNA energy parameter set.
- `int vrna_params_load_RNA_misc_special_hairpins` (void)
Load Misc Special Hairpin RNA energy parameter set.
- `int vrna_params_load_DNA_Mathews2004` (void)
Load Mathews 2004 DNA energy parameter set.
- `int vrna_params_load_DNA_Mathews1999` (void)
Load Mathews 1999 DNA energy parameter set.
- `const char * last_parameter_file` (void)
Get the file name of the parameter file that was most recently loaded.
- `void read_parameter_file` (const char fname[])
Read energy parameters from a file.
- `void write_parameter_file` (const char fname[])
Write energy parameters to a file.

16.58.2 Macro Definition Documentation

16.58.2.1 VRNA_PARAMETER_FORMAT_DEFAULT

```
#define VRNA_PARAMETER_FORMAT_DEFAULT 0
#include <ViennaRNA/params/io.h>
Default Energy Parameter File format.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#)

16.58.3 Function Documentation

16.58.3.1 vrna_params_load()

```
int vrna_params_load (
    const char fname[],
    unsigned int options )
#include <ViennaRNA/params/io.h>
Load energy parameters from a file.
```

See also

[vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_defaults\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#), [vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Parameters

<i>fname</i>	The path to the file containing the energy parameters
<i>options</i>	File format bit-mask (usually VRNA_PARAMETER_FORMAT_DEFAULT)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as overloaded function **params_load**(fname="", options=[VRNA_PARAMETER_FORMAT_DEFAULT](#)). Here, the empty filename string indicates to load default RNA parameters, i.e. this is equivalent to calling [vrna_params_load_defaults\(\)](#).

16.58.3.2 vrna_params_save()

```
int vrna_params_save (
    const char fname[],
    unsigned int options )
#include <ViennaRNA/params/io.h>
Save energy parameters to a file.
```

See also

[vrna_params_load\(\)](#)

Parameters

<i>fname</i>	A filename (path) for the file where the current energy parameters will be written to
<i>options</i>	File format bit-mask (usually VRNA_PARAMETER_FORMAT_DEFAULT)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as overloaded function **params_save**(fname, options=[VRNA_PARAMETER_FORMAT_DEFAULT](#))

16.58.3.3 **vrna_params_load_from_string()**

```
int vrna_params_load_from_string (
    const char * string,
    const char * name,
    unsigned int options )
#include <ViennaRNA/params/io.h>
```

Load energy paramters from string.

The string must follow the default energy parameter file convention! The optional `name` argument allows one to specify a name for the parameter set which is stored internally.

See also

[vrna_params_load\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_defaults\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#), [vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Parameters

<i>string</i>	A 0-terminated string containing energy parameters
<i>name</i>	A name for the parameter set in <code>string</code> (Maybe NULL)
<i>options</i>	File format bit-mask (usually VRNA_PARAMETER_FORMAT_DEFAULT)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as overloaded function **params_load_from_string**(string, name="", options=[VRNA_PARAMETER_FORMAT_DEFAULT](#)).

16.58.3.4 **vrna_params_load_defaults()**

```
int vrna_params_load_defaults (
    void )
#include <ViennaRNA/params/io.h>
```

Load default RNA energy parameter set.

This is a convenience function to load the Turner 2004 RNA free energy parameters. It's the same as calling [vrna_params_load_RNA_Turner2004\(\)](#)

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#), [vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as overloaded function `params_load()`.

16.58.3.5 vrna_params_load_RNA_Turner2004()

```
int vrna_params_load_RNA_Turner2004 (
    void )
#include <ViennaRNA/params/io.h>
Load Turner 2004 RNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_defaults\(\)](#),
[vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#),
[vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_RNA_Turner2004()`.

16.58.3.6 vrna_params_load_RNA_Turner1999()

```
int vrna_params_load_RNA_Turner1999 (
    void )
#include <ViennaRNA/params/io.h>
Load Turner 1999 RNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#),
[vrna_params_load_defaults\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#),
[vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_RNA_Turner1999()`.

16.58.3.7 vrna_params_load_RNA_Andronescu2007()

```
int vrna_params_load_RNA_Andronescu2007 (
    void )
#include <ViennaRNA/params/io.h>
Load Andronescu 2007 RNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#),
[vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_defaults\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#),
[vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_RNA_Andronescu2007()`.

16.58.3.8 vrna_params_load_RNA_Langdon2018()

```
int vrna_params_load_RNA_Langdon2018 (
    void )
#include <ViennaRNA/params/io.h>
Load Langdon 2018 RNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_defaults\(\)](#), [vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathev](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_RNA_Langdon2018()`.

16.58.3.9 vrna_params_load_RNA_misc_special_hairpins()

```
int vrna_params_load_RNA_misc_special_hairpins (
    void )
#include <ViennaRNA/params/io.h>
Load Misc Special Hairpin RNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#), [vrna_params_load_defaults\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_DNA_Mathews1999\(\)](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_RNA_misc_special_hairpins()`.

16.58.3.10 vrna_params_load_DNA_Mathews2004()

```
int vrna_params_load_DNA_Mathews2004 (
    void )
#include <ViennaRNA/params/io.h>
Load Mathews 2004 DNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#), [vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_defaults\(\)](#), [vrna_params_load_DNA_Mathews1999\(\)](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_DNA_Mathews2004()`.

16.58.3.11 vrna_params_load_DNA_Mathews1999()

```
int vrna_params_load_DNA_Mathews1999 (
    void )
#include <ViennaRNA/params/io.h>
Load Mathews 1999 DNA energy parameter set.
```

See also

[vrna_params_load\(\)](#), [vrna_params_load_from_string\(\)](#), [vrna_params_save\(\)](#), [vrna_params_load_RNA_Turner2004\(\)](#), [vrna_params_load_RNA_Turner1999\(\)](#), [vrna_params_load_RNA_Andronescu2007\(\)](#), [vrna_params_load_RNA_Langdon2018\(\)](#), [vrna_params_load_RNA_misc_special_hairpins\(\)](#), [vrna_params_load_DNA_Mathews2004\(\)](#), [vrna_params_load_defaults\(\)](#)

Returns

Non-zero on success, 0 on failure

SWIG Wrapper Notes This function is available as function `params_load_DNA_Mathews1999()`.

16.58.3.12 last_parameter_file()

```
const char * last_parameter_file (
    void )
#include <ViennaRNA/params/io.h>
Get the file name of the parameter file that was most recently loaded.
```

Returns

The file name of the last parameter file, or NULL if parameters are still at defaults

16.58.3.13 read_parameter_file()

```
void read_parameter_file (
    const char fname[] )
#include <ViennaRNA/params/io.h>
Read energy parameters from a file.
```

Deprecated Use [vrna_params_load\(\)](#) instead!

Parameters

<i>fname</i>	The path to the file containing the energy parameters
--------------	---

16.58.3.14 write_parameter_file()

```
void write_parameter_file (
    const char fname[] )
#include <ViennaRNA/params/io.h>
Write energy parameters to a file.
```

Deprecated Use [vrna_params_save\(\)](#) instead!

Parameters

<i>fname</i>	A filename (path) for the file where the current energy parameters will be written to
--------------	---

16.59 Converting Energy Parameter Files

Convert energy parameter files into the latest format.

16.59.1 Detailed Description

Convert energy parameter files into the latest format.

To preserve some backward compatibility the RNAlib also provides functions to convert energy parameter files from the format used in version 1.4-1.8 into the new format used since version 2.0 Collaboration diagram for Converting Energy Parameter Files:

Files

- file [1.8.4_epars.h](#)
Free energy parameters for parameter file conversion.
- file [1.8.4_intloops.h](#)
Free energy parameters for interior loop contributions needed by the parameter file conversion functions.

Macros

- `#define VRNA_CONVERT_OUTPUT_ALL 1U`
- `#define VRNA_CONVERT_OUTPUT_HP 2U`
- `#define VRNA_CONVERT_OUTPUT_STACK 4U`
- `#define VRNA_CONVERT_OUTPUT_MM_HP 8U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT 16U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_1N 32U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_23 64U`
- `#define VRNA_CONVERT_OUTPUT_MM_MULTI 128U`
- `#define VRNA_CONVERT_OUTPUT_MM_EXT 256U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE5 512U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE3 1024U`
- `#define VRNA_CONVERT_OUTPUT_INT_11 2048U`
- `#define VRNA_CONVERT_OUTPUT_INT_21 4096U`
- `#define VRNA_CONVERT_OUTPUT_INT_22 8192U`
- `#define VRNA_CONVERT_OUTPUT_BULGE 16384U`
- `#define VRNA_CONVERT_OUTPUT_INT 32768U`
- `#define VRNA_CONVERT_OUTPUT_ML 65536U`
- `#define VRNA_CONVERT_OUTPUT_MISC 131072U`
- `#define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U`
- `#define VRNA_CONVERT_OUTPUT_VANILLA 524288U`
- `#define VRNA_CONVERT_OUTPUT_NINIO 1048576U`
- `#define VRNA_CONVERT_OUTPUT_DUMP 2097152U`

Functions

- void [convert_parameter_file](#) (const char *iname, const char *oname, unsigned int options)

16.59.2 Macro Definition Documentation

16.59.2.1 VRNA_CONVERT_OUTPUT_ALL

```
#define VRNA_CONVERT_OUTPUT_ALL 1U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of a complete parameter set

16.59.2.2 VRNA_CONVERT_OUTPUT_HP

```
#define VRNA_CONVERT_OUTPUT_HP 2U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of hairpin contributions

16.59.2.3 VRNA_CONVERT_OUTPUT_STACK

```
#define VRNA_CONVERT_OUTPUT_STACK 4U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of base pair stack contributions

16.59.2.4 VRNA_CONVERT_OUTPUT_MM_HP

```
#define VRNA_CONVERT_OUTPUT_MM_HP 8U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of hairpin mismatch contribution

16.59.2.5 VRNA_CONVERT_OUTPUT_MM_INT

```
#define VRNA_CONVERT_OUTPUT_MM_INT 16U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of interior loop mismatch contribution

16.59.2.6 VRNA_CONVERT_OUTPUT_MM_INT_1N

```
#define VRNA_CONVERT_OUTPUT_MM_INT_1N 32U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of 1:n interior loop mismatch contribution

16.59.2.7 VRNA_CONVERT_OUTPUT_MM_INT_23

```
#define VRNA_CONVERT_OUTPUT_MM_INT_23 64U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of 2:3 interior loop mismatch contribution

16.59.2.8 VRNA_CONVERT_OUTPUT_MM_MULTI

```
#define VRNA_CONVERT_OUTPUT_MM_MULTI 128U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of multi loop mismatch contribution

16.59.2.9 VRNA_CONVERT_OUTPUT_MM_EXT

```
#define VRNA_CONVERT_OUTPUT_MM_EXT 256U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of exterior loop mismatch contribution

16.59.2.10 VRNA_CONVERT_OUTPUT_DANGLE5

```
#define VRNA_CONVERT_OUTPUT_DANGLE5 512U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 5' dangle contribution
```

16.59.2.11 VRNA_CONVERT_OUTPUT_DANGLE3

```
#define VRNA_CONVERT_OUTPUT_DANGLE3 1024U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 3' dangle contribution
```

16.59.2.12 VRNA_CONVERT_OUTPUT_INT_11

```
#define VRNA_CONVERT_OUTPUT_INT_11 2048U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 1:1 interior loop contribution
```

16.59.2.13 VRNA_CONVERT_OUTPUT_INT_21

```
#define VRNA_CONVERT_OUTPUT_INT_21 4096U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 2:1 interior loop contribution
```

16.59.2.14 VRNA_CONVERT_OUTPUT_INT_22

```
#define VRNA_CONVERT_OUTPUT_INT_22 8192U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of 2:2 interior loop contribution
```

16.59.2.15 VRNA_CONVERT_OUTPUT_BULGE

```
#define VRNA_CONVERT_OUTPUT_BULGE 16384U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of bulge loop contribution
```

16.59.2.16 VRNA_CONVERT_OUTPUT_INT

```
#define VRNA_CONVERT_OUTPUT_INT 32768U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of interior loop contribution
```

16.59.2.17 VRNA_CONVERT_OUTPUT_ML

```
#define VRNA_CONVERT_OUTPUT_ML 65536U
#include <ViennaRNA/params/convert.h>
Flag to indicate printing of multi loop contribution
```

16.59.2.18 VRNA_CONVERT_OUTPUT_MISC

```
#define VRNA_CONVERT_OUTPUT_MISC 131072U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of misc contributions (such as terminalAU)

16.59.2.19 VRNA_CONVERT_OUTPUT_SPECIAL_HP

```
#define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of special hairpin contributions (tri-, tetra-, hexa-loops)

16.59.2.20 VRNA_CONVERT_OUTPUT_VANILLA

```
#define VRNA_CONVERT_OUTPUT_VANILLA 524288U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of given parameters only

Note

This option overrides all other output options, except [VRNA_CONVERT_OUTPUT_DUMP](#) !

16.59.2.21 VRNA_CONVERT_OUTPUT_NINIO

```
#define VRNA_CONVERT_OUTPUT_NINIO 1048576U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate printing of interior loop asymmetry contribution

16.59.2.22 VRNA_CONVERT_OUTPUT_DUMP

```
#define VRNA_CONVERT_OUTPUT_DUMP 2097152U
#include <ViennaRNA/params/convert.h>
```

Flag to indicate dumping the energy contributions from the library instead of an input file

16.59.3 Function Documentation

16.59.3.1 convert_parameter_file()

```
void convert_parameter_file (
    const char * iname,
    const char * oname,
    unsigned int options )
#include <ViennaRNA/params/convert.h>
```

Convert/dump a Vienna 1.8.4 formatted energy parameter file

The options argument allows one to control the different output modes.

Currently available options are:

[VRNA_CONVERT_OUTPUT_ALL](#), [VRNA_CONVERT_OUTPUT_HP](#), [VRNA_CONVERT_OUTPUT_STACK](#)
[VRNA_CONVERT_OUTPUT_MM_HP](#), [VRNA_CONVERT_OUTPUT_MM_INT](#), [VRNA_CONVERT_OUTPUT_MM_INT_1N](#)
[VRNA_CONVERT_OUTPUT_MM_INT_23](#), [VRNA_CONVERT_OUTPUT_MM_MULTI](#), [VRNA_CONVERT_OUTPUT_MM_EXT](#)
[VRNA_CONVERT_OUTPUT_DANGLE5](#), [VRNA_CONVERT_OUTPUT_DANGLE3](#), [VRNA_CONVERT_OUTPUT_INT_11](#)
[VRNA_CONVERT_OUTPUT_INT_21](#), [VRNA_CONVERT_OUTPUT_INT_22](#), [VRNA_CONVERT_OUTPUT_BULGE](#)
[VRNA_CONVERT_OUTPUT_INT](#), [VRNA_CONVERT_OUTPUT_ML](#), [VRNA_CONVERT_OUTPUT_MISC](#)
[VRNA_CONVERT_OUTPUT_SPECIAL_HP](#), [VRNA_CONVERT_OUTPUT_VANILLA](#), [VRNA_CONVERT_OUTPUT_NINIO](#)
[VRNA_CONVERT_OUTPUT_DUMP](#)

The defined options are fine for bitwise compare- and assignment-operations, e. g.: pass a collection of options as a single value like this:

```
convert_parameter_file(ifile, ofile, option_1 | option_2 | option_n)
```

Parameters

<i>iname</i>	The input file name (If NULL input is read from stdin)
<i>oname</i>	The output file name (If NULL output is written to stdout)
<i>options</i>	The options (as described above)

16.60 Utilities to deal with Nucleotide Alphabets

Functions to cope with various aspects related to the nucleotide sequence alphabet.

16.60.1 Detailed Description

Functions to cope with various aspects related to the nucleotide sequence alphabet.

Collaboration diagram for Utilities to deal with Nucleotide Alphabets:

Files

- file [alphabet.h](#)
Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.
- file [sequence.h](#)
Functions and data structures related to sequence representations ,.

Data Structures

- struct [vrna_sequence_s](#)
Data structure representing a nucleotide sequence. [More...](#)
- struct [vrna_alignment_s](#)

Typedefs

- typedef struct [vrna_sequence_s](#) [vrna_seq_t](#)
Typename for nucleotide sequence representation data structure [vrna_sequence_s](#).

Enumerations

- enum [vrna_seq_type_e](#) { [VRNA_SEQ_UNKNOWN](#) , [VRNA_SEQ_RNA](#) , [VRNA_SEQ_DNA](#) }
A enumerator used in [vrna_sequence_s](#) to distinguish different nucleotide sequences.

Functions

- char * [vrna_ptypes](#) (const short *S, [vrna_md_t](#) *md)
Get an array of the numerical encoding for each possible base pair (i,j)
- short * [vrna_seq_encode](#) (const char *sequence, [vrna_md_t](#) *md)
Get a numerical representation of the nucleotide sequence.
- short * [vrna_seq_encode_simple](#) (const char *sequence, [vrna_md_t](#) *md)
Get a numerical representation of the nucleotide sequence (simple version)
- int [vrna_nucleotide_encode](#) (char c, [vrna_md_t](#) *md)
Encode a nucleotide character to numerical value.
- char [vrna_nucleotide_decode](#) (int enc, [vrna_md_t](#) *md)
Decode a numerical representation of a nucleotide back into nucleotide alphabet.

16.60.2 Data Structure Documentation

16.60.2.1 struct vrna_sequence_s

Data structure representing a nucleotide sequence.

Data Fields

- [vrna_seq_type_e](#) type
The type of sequence.
- char * **string**
The string representation of the sequence.
- short * **encoding**
The integer representation of the sequence.
- unsigned int **length**
The length of the sequence.

16.60.2.2 struct vrna_alignment_s

Collaboration diagram for vrna_alignment_s:

16.60.3 Enumeration Type Documentation

16.60.3.1 vrna_seq_type_e

```
enum vrna\_seq\_type\_e
```

```
#include <ViennaRNA/sequence.h>
```

A enumerator used in [vrna_sequence_s](#) to distinguish different nucleotide sequences.

Enumerator

VRNA_SEQ_UNKNOWN	Nucleotide sequence represents an Unkown type.
VRNA_SEQ_RNA	Nucleotide sequence represents an RNA type.
VRNA_SEQ_DNA	Nucleotide sequence represents a DNA type.

16.60.4 Function Documentation

16.60.4.1 vrna_ptypes()

```
char * vrna_ptypes (
    const short * S,
    vrna\_md\_t * md )
#include <ViennaRNA/alphabet.h>
```

Get an array of the numerical encoding for each possible base pair (i,j)

Note

This array is always indexed in column-wise order, in contrast to previously different indexing between mfe and pf variants!

See also

[vrna_idx_col_wise\(\)](#), [vrna_fold_compound_t](#)

16.60.4.2 `vrna_seq_encode()`

```
short * vrna_seq_encode (
    const char * sequence,
    vrna_md_t * md )
#include <ViennaRNA/alphabet.h>
```

Get a numerical representation of the nucleotide sequence.

Parameters

<i>sequence</i>	The input sequence in upper-case letters
<i>md</i>	A pointer to a <code>vrna_md_t</code> data structure that specifies the conversion type

Returns

A list of integer encodings for each sequence letter (1-based). Position 0 denotes the length of the list

SWIG Wrapper Notes In the target scripting language, this function is wrapped as overloaded function `seq_encode()` where the last parameter, the `model_details` data structure, is optional. If it is omitted, default model settings are applied, i.e. default nucleotide letter conversion. The wrapped function returns a list/tuple of integer representations of the input sequence.

16.60.4.3 `vrna_seq_encode_simple()`

```
short * vrna_seq_encode_simple (
    const char * sequence,
    vrna_md_t * md )
#include <ViennaRNA/alphabet.h>
```

Get a numerical representation of the nucleotide sequence (simple version)

16.60.4.4 `vrna_nucleotide_encode()`

```
int vrna_nucleotide_encode (
    char c,
    vrna_md_t * md )
#include <ViennaRNA/alphabet.h>
```

Encode a nucleotide character to numerical value.

This function encodes a nucleotide character to its numerical representation as required by many functions in RNAlib.

See also

[vrna_nucleotide_decode\(\)](#), [vrna_seq_encode\(\)](#)

Parameters

<i>c</i>	The nucleotide character to encode
<i>md</i>	The model details that determine the kind of encoding

Returns

The encoded nucleotide

16.60.4.5 vrna_nucleotide_decode()

```
char vrna_nucleotide_decode (
    int enc,
    vrna_md_t * md )
```

```
#include <ViennaRNA/alphabet.h>
```

Decode a numerical representation of a nucleotide back into nucleotide alphabet.

This function decodes a numerical representation of a nucleotide character back into nucleotide alphabet

See also

[vrna_nucleotide_encode\(\)](#), [vrna_seq_encode\(\)](#)

Parameters

<i>enc</i>	The encoded nucleotide
<i>md</i>	The model details that determine the kind of decoding

Returns

The decoded nucleotide character

16.61 (Nucleic Acid Sequence) String Utilities

Functions to parse, convert, manipulate, create, and compare (nucleic acid sequence) strings.

16.61.1 Detailed Description

Functions to parse, convert, manipulate, create, and compare (nucleic acid sequence) strings.

Collaboration diagram for (Nucleic Acid Sequence) String Utilities:

Files

- file [strings.h](#)

General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.

Macros

- `#define XSTR(s) STR(s)`
Stringify a macro after expansion.
- `#define STR(s) #s`
Stringify a macro argument.
- `#define FILENAME_MAX_LENGTH 80`
Maximum length of filenames that are generated by our programs.
- `#define FILENAME_ID_LENGTH 42`
Maximum length of id taken from fasta header for filename generation.
- `#define VRNA_TRIM_LEADING 1U`
Trim only characters leading the string.
- `#define VRNA_TRIM_TRAILING 2U`

- Trim only characters trailing the string.*
- #define `VRNA_TRIM_IN_BETWEEN` 4U
- Trim only characters within the string.*
- #define `VRNA_TRIM_SUBST_BY_FIRST` 8U
- Replace remaining characters after trimming with the first delimiter in list.*
- #define `VRNA_TRIM_DEFAULT` (`VRNA_TRIM_LEADING` | `VRNA_TRIM_TRAILING`)
- Default settings for trimming, i.e. trim leading and trailing.*
- #define `VRNA_TRIM_ALL` (`VRNA_TRIM_DEFAULT` | `VRNA_TRIM_IN_BETWEEN`)
- Trim characters anywhere in the string.*

Functions

- char * `vrna_strdup_printf` (const char *format,...)
Safely create a formatted string.
- char * `vrna_strdup_vprintf` (const char *format, va_list argp)
Safely create a formatted string.
- int `vrna_strcat_printf` (char **dest, const char *format,...)
Safely append a formatted string to another string.
- int `vrna_strcat_vprintf` (char **dest, const char *format, va_list args)
Safely append a formatted string to another string.
- unsigned int `vrna_strtrim` (char *string, const char *delimiters, unsigned int keep, unsigned int options)
Trim a string by removing (multiple) occurrences of a particular character.
- char ** `vrna_strsplit` (const char *string, const char *delimiter)
Split a string into tokens using a delimiting character.
- char * `vrna_random_string` (int l, const char symbols[])
Create a random string using characters from a specified symbol set.
- int `vrna_hamming_distance` (const char *s1, const char *s2)
Calculate hamming distance between two sequences.
- int `vrna_hamming_distance_bound` (const char *s1, const char *s2, int n)
Calculate hamming distance between two sequences up to a specified length.
- void `vrna_seq_toRNA` (char *sequence)
Convert an input sequence (possibly containing DNA alphabet characters) to RNA alphabet.
- void `vrna_seq_toupper` (char *sequence)
Convert an input sequence to uppercase.
- void `vrna_seq_reverse` (char *sequence)
Reverse a string in-place.
- char * `vrna_DNA_complement` (const char *sequence)
Retrieve a DNA sequence which resembles the complement of the input sequence.
- char * `vrna_seq_ungapped` (const char *sequence)
Remove gap characters from a nucleotide sequence.
- char * `vrna_cut_point_insert` (const char *string, int cp)
Add a separating '&' character into a string according to cut-point position.
- char * `vrna_cut_point_remove` (const char *string, int *cp)
Remove a separating '&' character from a string.

16.61.2 Macro Definition Documentation

16.61.2.1 FILENAME_MAX_LENGTH

```
#define FILENAME_MAX_LENGTH 80
```

```
#include <ViennaRNA/utils/strings.h>
```

Maximum length of filenames that are generated by our programs.

This definition should be used throughout the complete ViennaRNA package wherever a static array holding filenames of output files is declared.

16.61.2.2 FILENAME_ID_LENGTH

```
#define FILENAME_ID_LENGTH 42
```

```
#include <ViennaRNA/utils/strings.h>
```

Maximum length of id taken from fasta header for filename generation.

this has to be smaller than FILENAME_MAX_LENGTH since in most cases, some suffix will be appended to the ID

16.61.2.3 VRNA_TRIM_LEADING

```
#define VRNA_TRIM_LEADING 1U
```

```
#include <ViennaRNA/utils/strings.h>
```

Trim only characters leading the string.

See also

[vrna_strtrim\(\)](#)

16.61.2.4 VRNA_TRIM_TRAILING

```
#define VRNA_TRIM_TRAILING 2U
```

```
#include <ViennaRNA/utils/strings.h>
```

Trim only characters trailing the string.

See also

[vrna_strtrim\(\)](#)

16.61.2.5 VRNA_TRIM_IN_BETWEEN

```
#define VRNA_TRIM_IN_BETWEEN 4U
```

```
#include <ViennaRNA/utils/strings.h>
```

Trim only characters within the string.

See also

[vrna_strtrim\(\)](#)

16.61.2.6 VRNA_TRIM_SUBST_BY_FIRST

```
#define VRNA_TRIM_SUBST_BY_FIRST 8U
```

```
#include <ViennaRNA/utils/strings.h>
```

Replace remaining characters after trimming with the first delimiter in list.

See also

[vrna_strtrim\(\)](#)

16.61.2.7 VRNA_TRIM_DEFAULT

```
#define VRNA_TRIM_DEFAULT ( VRNA_TRIM_LEADING | VRNA_TRIM_TRAILING )
#include <ViennaRNA/utils/strings.h>
Default settings for trimming, i.e. trim leading and trailing.
```

See also

[vrna_strtrim\(\)](#)

16.61.2.8 VRNA_TRIM_ALL

```
#define VRNA_TRIM_ALL ( VRNA_TRIM_DEFAULT | VRNA_TRIM_IN_BETWEEN )
#include <ViennaRNA/utils/strings.h>
Trim characters anywhere in the string.
```

See also

[vrna_strtrim\(\)](#)

16.61.3 Function Documentation

16.61.3.1 vrna_strdup_printf()

```
char * vrna_strdup_printf (
    const char * format,
    ... )
#include <ViennaRNA/utils/strings.h>
```

Safely create a formatted string.

This function is a safe implementation for creating a formatted character array, similar to *sprintf*. Internally, it uses the *asprintf* function if available to dynamically allocate a large enough character array to store the supplied content. If *asprintf* is not available, mimic it's behavior using *vsprintf*.

Note

The returned pointer of this function should always be passed to *free()* to release the allocated memory

See also

[vrna_strdup_vprintf\(\)](#), [vrna_strcat_printf\(\)](#)

Parameters

<i>format</i>	The format string (See also <i>asprintf</i>)
...	The list of variables used to fill the format string

Returns

The formatted, null-terminated string, or NULL if something has gone wrong

16.61.3.2 vrna_strdup_vprintf()

```
char * vrna_strdup_vprintf (
    const char * format,
    va_list argp )
```

```
#include <ViennaRNA/utils/strings.h>
```

Safely create a formatted string.
This function is the *va_list* version of [vrna_strdup_printf\(\)](#)

Note

The returned pointer of this function should always be passed to *free()* to release the allocated memory

See also

[vrna_strdup_printf\(\)](#), [vrna_strcat_printf\(\)](#), [vrna_strcat_vprintf\(\)](#)

Parameters

<i>format</i>	The format string (See also asprintf())
<i>argp</i>	The list of arguments to fill the format string

Returns

The formatted, null-terminated string, or NULL if something has gone wrong

16.61.3.3 vrna_strcat_printf()

```
int vrna_strcat_printf (
    char ** dest,
    const char * format,
    ... )
```

```
#include <ViennaRNA/utils/strings.h>
```

Safely append a formatted string to another string.

This function is a safe implementation for appending a formatted character array, similar to a combination of *strcat* and *sprintf*. The function automatically allocates enough memory to store both, the previous content stored at *dest* and the appended format string. If the *dest* pointer is NULL, the function allocate memory only for the format string. The function returns the number of characters in the resulting string or -1 in case of an error.

See also

[vrna_strcat_vprintf\(\)](#), [vrna_strdup_printf\(\)](#), [vrna_strdup_vprintf\(\)](#)

Parameters

<i>dest</i>	The address of a char *pointer where the formatted string is to be appended
<i>format</i>	The format string (See also sprintf())
...	The list of variables used to fill the format string

Returns

The number of characters in the final string, or -1 on error

16.61.3.4 vrna_strcat_vprintf()

```
int vrna_strcat_vprintf (
    char ** dest,
    const char * format,
    va_list args )
```

```
#include <ViennaRNA/utils/strings.h>
```

Safely append a formatted string to another string.
This function is the *va_list* version of [vrna_strcat_printf\(\)](#)

See also

[vrna_strcat_printf\(\)](#), [vrna_strdup_printf\(\)](#), [vrna_strdup_vprintf\(\)](#)

Parameters

<i>dest</i>	The address of a char *pointer where the formatted string is to be appended
<i>format</i>	The format string (See also sprintf)
<i>args</i>	The list of argument to fill the format string

Returns

The number of characters in the final string, or -1 on error

16.61.3.5 vrna_strtrim()

```
unsigned int vrna_strtrim (
    char * string,
    const char * delimiters,
    unsigned int keep,
    unsigned int options )
```

```
#include <ViennaRNA/utils/strings.h>
```

Trim a string by removing (multiple) occurrences of a particular character.

This function removes (multiple) consecutive occurrences of a set of characters (*delimiters*) within an input string. It may be used to remove leading and/or trailing whitespaces or to restrict the maximum number of consecutive occurrences of the delimiting characters *delimiters*. Setting *keep*=0 removes all occurrences, while other values reduce multiple consecutive occurrences to at most *keep* delimiters. This might be useful if one would like to reduce multiple whitespaces to a single one, or to remove empty fields within a comma-separated value string.

The parameter *delimiters* may be a pointer to a 0-terminated char string containing a set of any ASCII character. If *NULL* is passed as delimiter set or an empty char string, all whitespace characters are trimmed. The *options* parameter is a bit vector that specifies which part of the string should undergo trimming. The implementation distinguishes the leading ([VRNA_TRIM_LEADING](#)), trailing ([VRNA_TRIM_TRAILING](#)), and in-between ([VRNA_TRIM_IN_BETWEEN](#)) part with respect to the delimiter set. Combinations of these parts can be specified by using logical-or operator.

The following example code removes all leading and trailing whitespace characters from the input string:

```
char    string[20] = " \t blablabla ";
unsigned int r      = vrna_strtrim(&(string[0]),
                                NULL,
                                0,
                                VRNA_TRIM_DEFAULT);
```

Note

The delimiter always consists of a single character from the set of characters provided. In case of alternative delimiters and non-null *keep* parameter, the first *keep* delimiters are preserved within the string. Use [VRNA_TRIM_SUBST_BY_FIRST](#) to substitute all remaining delimiting characters with the first from the *delimiters* list.

See also

[VRNA_TRIM_LEADING](#), [VRNA_TRIM_TRAILING](#), [VRNA_TRIM_IN_BETWEEN](#), [VRNA_TRIM_SUBST_BY_FIRST](#), [VRNA_TRIM_DEFAULT](#), [VRNA_TRIM_ALL](#)

Parameters

<i>string</i>	The '\0'-terminated input string to trim
<i>delimiters</i>	The delimiter characters as 0-terminated char array (or <i>NULL</i>)
<i>keep</i>	The maximum number of consecutive occurrences of the delimiter in the output string
<i>options</i>	The option bit vector specifying the mode of operation

Returns

The number of delimiters removed from the string

SWIG Wrapper Notes Since many scripting languages treat strings as immutable objects, this function does not modify the input string directly. Instead, it returns the modified string as second return value, together with the number of removed delimiters.

The scripting language interface provides an overloaded version of this function, with default parameters `delimiters=NULL`, `keep=0`, and `options=VRNA_TRIM_DEFAULT`.

16.61.3.6 `vrna_strsplit()`

```
char ** vrna_strsplit (
    const char * string,
    const char * delimiter )
#include <ViennaRNA/utils/strings.h>
```

Split a string into tokens using a delimiting character.

This function splits a string into an array of strings using a single character that delimits the elements within the string. The default delimiter is the ampersand ' & ' and will be used when `NULL` is passed as a second argument. The returned list is `NULL` terminated, i.e. the last element is `NULL`. If the delimiter is not found, the returned list contains exactly one element: the input string.

For instance, the following code:

```
char **tok = vrna_strsplit("GGGG&CCCC&AAAAA", NULL);

for (char **ptr = tok; *ptr; ptr++) {
    printf("%s\n", *ptr);
    free(*ptr);
}
free(tok);
```

produces this output:

```
* GGGG
* CCCC
* AAAAA
*
```

and properly free's the memory occupied by the returned element array.

Note

This function internally uses `strtok_r()` and is therefore considered to be thread-safe. Also note, that it is the users responsibility to free the memory of the array and that of the individual element strings!

In case the input string consists of consecutive delimiters, starts or ends with one or multiple delimiters, empty strings are produced in the output list, indicating the empty fields of data resulting from the split. Use [vrna_strtrim\(\)](#) prior to a call to this function to remove any leading, trailing, or in-between empty fields.

See also

[vrna_strtrim\(\)](#)

Parameters

<i>string</i>	The input string that should be split into elements
<i>delimiter</i>	The delimiting character. If <code>NULL</code> , the delimiter is "&"

Returns

A `NULL` terminated list of the elements in the string

16.61.3.7 `vrna_random_string()`

```
char * vrna_random_string (
    int l,
    const char symbols[] )
#include <ViennaRNA/utils/strings.h>
```

Create a random string using characters from a specified symbol set.

Parameters

<i>l</i>	The length of the sequence
<i>symbols</i>	The symbol set

Returns

A random string of length 'l' containing characters from the symbolset

16.61.3.8 `vrna_hamming_distance()`

```
int vrna_hamming_distance (
    const char * s1,
    const char * s2 )
#include <ViennaRNA/utils/strings.h>
```

Calculate hamming distance between two sequences.

Parameters

<i>s1</i>	The first sequence
<i>s2</i>	The second sequence

Returns

The hamming distance between s1 and s2

16.61.3.9 `vrna_hamming_distance_bound()`

```
int vrna_hamming_distance_bound (
    const char * s1,
    const char * s2,
    int n )
#include <ViennaRNA/utils/strings.h>
```

Calculate hamming distance between two sequences up to a specified length.

This function is similar to [vrna_hamming_distance\(\)](#) but instead of comparing both sequences up to their actual length only the first 'n' characters are taken into account

Parameters

<i>s1</i>	The first sequence
<i>s2</i>	The second sequence
<i>n</i>	The length of the subsequences to consider (starting from the 5' end)

Returns

The hamming distance between *s1* and *s2*

16.61.3.10 vrna_seq_toRNA()

```
void vrna_seq_toRNA (
    char * sequence )
#include <ViennaRNA/utils/strings.h>
```

Convert an input sequence (possibly containing DNA alphabet characters) to RNA alphabet.
This function substitutes *T* and *t* with *U* and *u*, respectively

Parameters

<i>sequence</i>	The sequence to be converted
-----------------	------------------------------

16.61.3.11 vrna_seq_toupper()

```
void vrna_seq_toupper (
    char * sequence )
#include <ViennaRNA/utils/strings.h>
```

Convert an input sequence to uppercase.

Parameters

<i>sequence</i>	The sequence to be converted
-----------------	------------------------------

16.61.3.12 vrna_seq_reverse()

```
void vrna_seq_reverse (
    char * sequence )
#include <ViennaRNA/utils/strings.h>
```

Reverse a string in-place.
This function reverses a character string in the form of an array of characters in-place, i.e. it changes the input parameter.

Postcondition

After execution, the input *sequence* consists of the reverse string prior to the execution.

See also

[vrna_DNA_complement\(\)](#)

Parameters

<i>sequence</i>	The string to reverse
-----------------	-----------------------

16.61.3.13 vrna_DNA_complement()

```
char * vrna_DNA_complement (
    const char * sequence )
#include <ViennaRNA/utils/strings.h>
```

Retrieve a DNA sequence which resembles the complement of the input sequence.

This function returns a new DNA string which is the complement of the input, i.e. the nucleotide letters A,C,G, and T are substituted by their complements T,G,C, and A, respectively.

Any characters not belonging to the alphabet of the 4 canonical bases of DNA are not altered.

Note

This function also handles lower-case input sequences and treats U of the RNA alphabet equally to T

See also

[vrna_seq_reverse\(\)](#)

Parameters

<i>sequence</i>	the input DNA sequence
-----------------	------------------------

Returns

The complement of the input DNA sequence

16.61.3.14 vrna_seq_ungapped()

```
char * vrna_seq_ungapped (
    const char * sequence )
#include <ViennaRNA/utils/strings.h>
```

Remove gap characters from a nucleotide sequence.

Parameters

<i>sequence</i>	The original, null-terminated nucleotide sequence
-----------------	---

Returns

A copy of the input sequence with all gap characters removed

16.61.3.15 vrna_cut_point_insert()

```
char * vrna_cut_point_insert (
    const char * string,
    int cp )
#include <ViennaRNA/utils/strings.h>
```

Add a separating '&' character into a string according to cut-point position.

If the cut-point position is less or equal to zero, this function just returns a copy of the provided string. Otherwise, the cut-point character is set at the corresponding position

Parameters

<i>string</i>	The original string
<i>cp</i>	The cut-point position

Returns

A copy of the provided string including the cut-point character

16.61.3.16 vrna_cut_point_remove()

```
char * vrna_cut_point_remove (
    const char * string,
    int * cp )
#include <ViennaRNA/utils/strings.h>
```

Remove a separating '&' character from a string.

This function removes the cut-point indicating '&' character from a string and memorizes its position in a provided integer variable. If not '&' is found in the input, the integer variable is set to -1. The function returns a copy of the input string with the '&' being sliced out.

Parameters

<i>string</i>	The original string
<i>cp</i>	The cut-point position

Returns

A copy of the input string with the '&' being sliced out

16.62 Secondary Structure Utilities

Functions to create, parse, convert, manipulate, and compare secondary structure representations.

16.62.1 Detailed Description

Functions to create, parse, convert, manipulate, and compare secondary structure representations.

Collaboration diagram for Secondary Structure Utilities:

Modules

- [Dot-Bracket Notation of Secondary Structures](#)

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots . .

- [Washington University Secondary Structure \(WUSS\) notation](#)

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

- [Pair Table Representation of Secondary Structures](#)
- [Pair List Representation of Secondary Structures](#)
- [Abstract Shapes Representation of Secondary Structures](#)

Abstract Shapes, introduced by Giegerich et al. in (2004) [12], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

- [Helix List Representation of Secondary Structures](#)

- [Tree Representation of Secondary Structures](#)

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

- [Distance measures between Secondary Structures](#)
- [Deprecated Interface for Secondary Structure Utilities](#)

Files

- file [structures.h](#)

Various utility- and helper-functions for secondary structure parsing, converting, etc.

Functions

- `int * vrna_loopidx_from_ptable` (const short *pt)
Get a loop index representation of a structure.
- `unsigned int * vrna_refBPcnt_matrix` (const short *reference_pt, unsigned int turn)
Make a reference base pair count matrix.
- `unsigned int * vrna_refBPdist_matrix` (const short *pt1, const short *pt2, unsigned int turn)
Make a reference base pair distance matrix.
- `char * vrna_db_from_probs` (const FLT_OR_DBL *pr, unsigned int length)
Create a dot-bracket like structure string from base pair probability matrix.
- `char vrna_bpp_symbol` (const float *x)
Get a pseudo dot bracket notation for a given probability information.
- `char * vrna_db_from_bp_stack` (vrna_bp_stack_t *bp, unsigned int length)
Create a dot-bracket/parenthesis structure from backtracking stack.

16.62.2 Function Documentation

16.62.2.1 vrna_refBPcnt_matrix()

```
unsigned int * vrna_refBPcnt_matrix (
    const short * reference_pt,
    unsigned int turn )
#include <ViennaRNA/utils/structures.h>
```

Make a reference base pair count matrix.

Get an upper triangular matrix containing the number of basepairs of a reference structure for each interval [i,j] with i<j. Access it via iindx!!!

16.62.2.2 vrna_refBPdist_matrix()

```
unsigned int * vrna_refBPdist_matrix (
    const short * pt1,
    const short * pt2,
    unsigned int turn )
#include <ViennaRNA/utils/structures.h>
```

Make a reference base pair distance matrix.

Get an upper triangular matrix containing the base pair distance of two reference structures for each interval [i,j] with i<j. Access it via iindx!!!

16.62.2.3 vrna_db_from_probs()

```
char * vrna_db_from_probs (
    const FLT_OR_DBL * pr,
    unsigned int length )
#include <ViennaRNA/utils/structures.h>
```

Create a dot-bracket like structure string from base pair probability matrix.

SWIG Wrapper Notes This function is available as parameter-less method **db_from_probs()** bound to objects of type *fold_compound*. Parameters *pr* and *length* are implicitly taken from the *fold_compound* object the method is bound to. Upon missing base pair probabilities, this method returns an empty string.

16.62.2.4 vrna_db_from_bp_stack()

```
char * vrna_db_from_bp_stack (
    vrna_bp_stack_t * bp,
    unsigned int length )
#include <ViennaRNA/utils/structures.h>
```

Create a dot-bracket/parenthesis structure from backtracking stack.

This function is capable to create dot-bracket structures from suboptimal structure prediction sensu M. Zuker

Parameters

<i>bp</i>	Base pair stack containing the traced base pairs
<i>length</i>	The length of the structure

Returns

The secondary structure in dot-bracket notation as provided in the input

16.63 Dot-Bracket Notation of Secondary Structures

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots ..

16.63.1 Detailed Description

The Dot-Bracket notation as introduced already in the early times of the ViennaRNA Package denotes base pairs by matching pairs of parenthesis () and unpaired nucleotides by dots ..

As a simple example, consider a helix of size 4 enclosing a hairpin of size 4. In dot-bracket notation, this is annotated as

```
(((((....))))
```

Extended Dot-Bracket Notation

A more generalized version of the original Dot-Bracket notation may use additional pairs of brackets, such as <>, {}, and [], and matching pairs of uppercase/lowercase letters. This allows for anoting pseudo-knots, since different pairs of brackets are not required to be nested.

The follwing annotations of a simple structure with two crossing helices of size 4 are equivalent:

```
<<<<[[[ [....>>>>]]]]
```

```
((((AAAA....)))) aaaa
```

AAAA{{{ { [.... aaaa] }}} Collaboration diagram for Dot-Bracket Notation of Secondary Structures:

Macros

- #define **VRNA_BRACKETS_ALPHA** 4U

Bitflag to indicate secondary structure notations using uppercase/lowercase letters from the latin alphabet.

- `#define VRNA_BRACKETS_RND 8U`
Bitflag to indicate secondary structure notations using round brackets (parenthesis), ()
- `#define VRNA_BRACKETS_CLY 16U`
Bitflag to indicate secondary structure notations using curly brackets, { }
- `#define VRNA_BRACKETS_ANG 32U`
Bitflag to indicate secondary structure notations using angular brackets, < >
- `#define VRNA_BRACKETS_SQR 64U`
Bitflag to indicate secondary structure notations using square brackets, []
- `#define VRNA_BRACKETS_DEFAULT`
Default bitmask to indicate secondary structure notation using any pair of brackets.
- `#define VRNA_BRACKETS_ANY`
Bitmask to indicate secondary structure notation using any pair of brackets or uppercase/lowercase alphabet letters.

Functions

- `char * vrna_db_pack (const char *struc)`
Pack secondary structure, 5:1 compression using base 3 encoding.
- `char * vrna_db_unpack (const char *packed)`
Unpack secondary structure previously packed with `vrna_db_pack()`
- `void vrna_db_flatten (char *structure, unsigned int options)`
Substitute pairs of brackets in a string with parenthesis.
- `void vrna_db_flatten_to (char *string, const char target[3], unsigned int options)`
Substitute pairs of brackets in a string with another type of pair characters.
- `char * vrna_db_from_ptable (const short *pt)`
Convert a pair table into dot-parenthesis notation.
- `char * vrna_db_from_plist (vrna_ep_t *pairs, unsigned int n)`
Convert a list of base pairs into dot-bracket notation.
- `char * vrna_db_to_element_string (const char *structure)`
Convert a secondary structure in dot-bracket notation to a nucleotide annotation of loop contexts.
- `char * vrna_db_pk_remove (const char *structure, unsigned int options)`
Remove pseudo-knots from an input structure.

16.63.2 Macro Definition Documentation

16.63.2.1 VRNA_BRACKETS_ALPHA

```
#define VRNA_BRACKETS_ALPHA 4U
```

```
#include <ViennaRNA/utils/structures.h>
```

Bitflag to indicate secondary structure notations using uppercase/lowercase letters from the latin alphabet.

See also

[vrna_ptable_from_string\(\)](#)

16.63.2.2 VRNA_BRACKETS_RND

```
#define VRNA_BRACKETS_RND 8U
```

```
#include <ViennaRNA/utils/structures.h>
```

Bitflag to indicate secondary structure notations using round brackets (parenthesis), ()

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#)

16.63.2.3 VRNA_BRACKETS_CLY

```
#define VRNA_BRACKETS_CLY 16U
#include <ViennaRNA/utils/structures.h>
Bitflag to indicate secondary structure notations using curly brackets, { }
```

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#)

16.63.2.4 VRNA_BRACKETS_ANG

```
#define VRNA_BRACKETS_ANG 32U
#include <ViennaRNA/utils/structures.h>
Bitflag to indicate secondary structure notations using angular brackets, <>
```

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#)

16.63.2.5 VRNA_BRACKETS_SQR

```
#define VRNA_BRACKETS_SQR 64U
#include <ViennaRNA/utils/structures.h>
Bitflag to indicate secondary structure notations using square brackets, [ ]
```

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#)

16.63.2.6 VRNA_BRACKETS_DEFAULT

```
#define VRNA_BRACKETS_DEFAULT
#include <ViennaRNA/utils/structures.h>
```

Value:

```
(VRNA_BRACKETS_RND | \
 VRNA_BRACKETS_CLY | \
 VRNA_BRACKETS_ANG | \
 VRNA_BRACKETS_SQR)
```

Default bitmask to indicate secondary structure notation using any pair of brackets.

This set of matching brackets/parenthesis is always nested, i.e. pseudo-knot free, in WUSS format. However, in general different kinds of brackets are mostly used for annotating pseudo-knots. Thus special care has to be taken to remove pseudo-knots if this bitmask is used in functions that return secondary structures without pseudo-knots!

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#), [vrna_db_pk_remove\(\)](#) [vrna_pt_pk_remove\(\)](#)

16.63.2.7 VRNA_BRACKETS_ANY

```
#define VRNA_BRACKETS_ANY
#include <ViennaRNA/utils/structures.h>
```

Value:

```
(VRNA_BRACKETS_RND | \
 VRNA_BRACKETS_CLY | \
 VRNA_BRACKETS_ANG | \
 VRNA_BRACKETS_SQR | \
 VRNA_BRACKETS_ALPHA)
```

Bitmask to indicate secondary structure notation using any pair of brackets or uppercase/lowercase alphabet letters.

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_pk_remove\(\)](#), [vrna_db_flatten\(\)](#), [vrna_db_flatten_to\(\)](#)

16.63.3 Function Documentation

16.63.3.1 `vrna_db_pack()`

```
char * vrna_db_pack (
    const char * struc )
#include <ViennaRNA/utils/structures.h>
```

Pack secondary secondary structure, 5:1 compression using base 3 encoding.

Returns a binary string encoding of the secondary structure using a 5:1 compression scheme. The string is NULL terminated and can therefore be used with standard string functions such as `strcmp()`. Useful for programs that need to keep many structures in memory.

See also

[vrna_db_unpack\(\)](#)

Parameters

<i>struc</i>	The secondary structure in dot-bracket notation
--------------	---

Returns

The binary encoded structure

16.63.3.2 `vrna_db_unpack()`

```
char * vrna_db_unpack (
    const char * packed )
#include <ViennaRNA/utils/structures.h>
```

Unpack secondary structure previously packed with [vrna_db_pack\(\)](#)

Translate a compressed binary string produced by [vrna_db_pack\(\)](#) back into the familiar dot-bracket notation.

See also

[vrna_db_pack\(\)](#)

Parameters

<i>packed</i>	The binary encoded packed secondary structure
---------------	---

Returns

The unpacked secondary structure in dot-bracket notation

16.63.3.3 `vrna_db_flatten()`

```
void vrna_db_flatten (
    char * structure,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Substitute pairs of brackets in a string with parenthesis.

This function can be used to replace brackets of unusual types, such as angular brackets <> , to dot-bracket format. The `options` parameter is used to specify which types of brackets will be replaced by round parenthesis () .

See also

[vrna_db_flatten_to\(\)](#), [VRNA_BRACKETS_RND](#), [VRNA_BRACKETS_ANG](#), [VRNA_BRACKETS_CLY](#), [VRNA_BRACKETS_SQR](#), [VRNA_BRACKETS_DEFAULT](#)

Parameters

<i>structure</i>	The structure string where brackets are flattened in-place
<i>options</i>	A bitmask to specify which types of brackets should be flattened out

SWIG Wrapper Notes This function flattens an input structure string in-place! The second parameter is optional and defaults to [VRNA_BRACKETS_DEFAULT](#).

An overloaded version of this function exists, where an additional second parameter can be passed to specify the target brackets, i.e. the type of matching pair characters all brackets will be flattened to. Therefore, in the scripting language interface this function is a replacement for [vrna_db_flatten_to\(\)](#).

16.63.3.4 vrna_db_flatten_to()

```
void vrna_db_flatten_to (
    char * string,
    const char target[3],
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Substitute pairs of brackets in a string with another type of pair characters.

This function can be used to replace brackets in a structure annotation string, such as square brackets [], to another type of pair characters, e.g. angular brackets <> .

The `target` array must contain a character for the 'pair open' annotation at position 0, and one for 'pair close' at position 1. The `options` parameter is used to specify which types of brackets will be replaced by the new pairs.

See also

[vrna_db_flatten\(\)](#), [VRNA_BRACKETS_RND](#), [VRNA_BRACKETS_ANG](#), [VRNA_BRACKETS_CLY](#), [VRNA_BRACKETS_SQR](#), [VRNA_BRACKETS_DEFAULT](#)

Parameters

<i>string</i>	The structure string where brackets are flattened in-place
<i>target</i>	The new pair characters the string will be flattened to
<i>options</i>	A bitmask to specify which types of brackets should be flattened out

SWIG Wrapper Notes This function is available as an overloaded version of [vrna_db_flatten\(\)](#)

16.63.3.5 vrna_db_from_ptable()

```
char * vrna_db_from_ptable (
    const short * pt )
#include <ViennaRNA/utils/structures.h>
```

Convert a pair table into dot-parenthesis notation.

This function also converts pair table formatted structures that contain pseudoknots. Non-nested base pairs result in additional pairs of parenthesis and brackets within the resulting dot-parenthesis string. The following pairs are available: (), [], {}, <>, as well as pairs of matching upper-/lower-case characters from the alphabet A-Z.

Note

In cases where the level of non-nested base pairs exceeds the maximum number of 30 different base pair indicators (4 parenthesis/brackets, 26 matching characters), a warning is printed and the remaining base pairs are left out from the conversion.

Parameters

<i>pt</i>	The pair table to be copied
-----------	-----------------------------

Returns

A char pointer to the dot-bracket string

16.63.3.6 `vrna_db_from_plist()`

```
char * vrna_db_from_plist (
    vrna_ep_t * pairs,
    unsigned int n )
#include <ViennaRNA/utils/structures.h>
```

Convert a list of base pairs into dot-bracket notation.

See also

[vrna_plist\(\)](#)

Parameters

<i>pairs</i>	A vrna_ep_t containing the pairs to be included in the dot-bracket string
<i>n</i>	The length of the structure (number of nucleotides)

Returns

The dot-bracket string containing the provided base pairs

16.63.3.7 `vrna_db_to_element_string()`

```
char * vrna_db_to_element_string (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Convert a secondary structure in dot-bracket notation to a nucleotide annotation of loop contexts.

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
------------------	---

Returns

A string annotating each nucleotide according to it's structural context

16.63.3.8 vrna_db_pk_remove()

```
char * vrna_db_pk_remove (
    const char * structure,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Remove pseudo-knots from an input structure.

This function removes pseudo-knots from an input structure by determining the minimum number of base pairs that need to be removed to make the structure pseudo-knot free.

To accomplish that, we use a dynamic programming algorithm similar to the Nussinov maximum matching approach.

The input structure must be in a dot-bracket string like form where crossing base pairs are denoted by the use of additional types of matching brackets, e.g. <>, {}, [], {}. Furthermore, crossing pairs may be annotated by matching uppercase/lowercase letters from the alphabet A-Z. For the latter, the uppercase letter must be the 5' and the lowercase letter the 3' nucleotide of the base pair. The actual type of brackets to be recognized by this function must be specified through the `options` parameter.

Note

Brackets in the input structure string that are not covered by the `options` bitmask will be silently ignored!

See also

[vrna_pt_pk_remove\(\)](#), [vrna_db_flatten\(\)](#), [VRNA_BRACKETS_RND](#), [VRNA_BRACKETS_ANG](#), [VRNA_BRACKETS_CLY](#), [VRNA_BRACKETS_SQR](#), [VRNA_BRACKETS_ALPHA](#), [VRNA_BRACKETS_DEFAULT](#), [VRNA_BRACKETS_ANY](#)

Parameters

<i>structure</i>	Input structure in dot-bracket format that may include pseudo-knots
<i>options</i>	A bitmask to specify which types of brackets should be processed

Returns

The input structure devoid of pseudo-knots in dot-bracket notation

SWIG Wrapper Notes This function is available as an overloaded function `db_pk_remove()` where the optional second parameter `options` defaults to [VRNA_BRACKETS_ANY](#).

16.64 Washington University Secondary Structure (WUSS) notation

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

16.64.1 Detailed Description

The WUSS notation, as frequently used for consensus secondary structures in [Stockholm 1.0 format](#).

This notation allows for a fine-grained annotation of base pairs and unpaired nucleotides, including pseudo-knots. Below, you'll find a list of secondary structure elements and their corresponding WUSS annotation (See also the infernal user guide at <http://eddylab.org/infernal/Userguide.pdf>)

- **Base pairs**

Nested base pairs are annotated by matching pairs of the symbols <>, (), {}, and []. Each of the matching pairs of parenthesis have their special meaning, however, when used as input in our programs, e.g.

structure constraint, these details are usually ignored. Furthermore, base pairs that constitute as pseudo-knot are denoted by letters from the latin alphabet and are, if not denoted otherwise, ignored entirely in our programs.

- **Hairpin loops**

Unpaired nucleotides that constitute the hairpin loop are indicated by underscores, `_`.

Example: `<<<<<_____>>>>>`

- **Bulges and interior loops**

Residues that constitute a bulge or interior loop are denoted by dashes, `-`.

Example: `(((--<_____>>-)))`

- **Multibranch loops**

Unpaired nucleotides in multibranch loops are indicated by commas, `,`.

Example: `(((„<_____>>, <_____>>)))`

- **External residues**

Single stranded nucleotides in the exterior loop, i.e. not enclosed by any other pair are denoted by colons, `:`.

Example: `<<<_____>>>:::`

- **Insertions**

In cases where an alignment represents the consensus with a known structure, insertions relative to the known structure are denoted by periods, `.`. Regions where local structural alignment was invoked, leaving regions of both target and query sequence unaligned, are indicated by tildes, `~`.

Note

These symbols only appear in alignments of a known (query) structure annotation to a target sequence of unknown structure.

- **Pseudo-knots**

The WUSS notation allows for annotation of pseudo-knots using pairs of upper-case/lower-case letters.

Note

Our programs and library functions usually ignore pseudo-knots entirely treating them as unpaired nucleotides, if not stated otherwise.

Example: `<<<_AAA_____>>>aaa`

Collaboration diagram for Washington University Secondary Structure (WUSS) notation:

Functions

- `char * vrna_db_from_WUSS (const char *wuss)`
Convert a WUSS annotation string to dot-bracket format.

16.64.2 Function Documentation

16.64.2.1 vrna_db_from_WUSS()

```
char * vrna_db_from_WUSS (
    const char * wuss )
#include <ViennaRNA/utils/structures.h>
Convert a WUSS annotation string to dot-bracket format.
```

Note

This function flattens all brackets, and treats pseudo-knots annotated by matching pairs of upper/lowercase letters as unpaired nucleotides

Parameters

<code>wuss</code>	The input string in WUSS notation
-------------------	-----------------------------------

Returns

A dot-bracket notation of the input secondary structure

16.65 Pair Table Representation of Secondary Structures

16.65.1 Detailed Description

Collaboration diagram for Pair Table Representation of Secondary Structures:

Functions

- short * [vrna_ptable](#) (const char *structure)
Create a pair table from a dot-bracket notation of a secondary structure.
- short * [vrna_ptable_from_string](#) (const char *structure, unsigned int options)
Create a pair table for a secondary structure string.
- short * [vrna_pt_pk_get](#) (const char *structure)
Create a pair table of a secondary structure (pseudo-knot version)
- short * [vrna_ptable_copy](#) (const short *pt)
Get an exact copy of a pair table.
- short * [vrna_pt_ali_get](#) (const char *structure)
Create a pair table of a secondary structure (snoop align version)
- short * [vrna_pt_snoop_get](#) (const char *structure)
Create a pair table of a secondary structure (snoop version)
- short * [vrna_pt_pk_remove](#) (const short *ptable, unsigned int options)
Remove pseudo-knots from a pair table.

16.65.2 Function Documentation

16.65.2.1 [vrna_ptable\(\)](#)

```
short * vrna_ptable (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table from a dot-bracket notation of a secondary structure.

Returns a newly allocated table, such that table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure.

See also

[vrna_ptable_from_string\(\)](#), [vrna_db_from_ptable\(\)](#)

Parameters

<code>structure</code>	The secondary structure in dot-bracket notation
------------------------	---

Returns

A pointer to the created `pair_table`

SWIG Wrapper Notes This functions is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

16.65.2.2 vrna_ptable_from_string()

```
short * vrna_ptable_from_string (
    const char * structure,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table for a secondary structure string.

This function takes an input string of a secondary structure annotation in [Dot-Bracket Notation](#) (a.k.a. [Dot-Parenthesis Notation](#)) or dot-bracket-ext-notation, and converts it into a pair table representation.

Note

This function also extracts crossing base pairs, i.e. pseudo-knots if more than a single matching bracket type is allowed through the bitmask `options`.

See also

[vrna_ptable\(\)](#), [vrna_db_from_ptable\(\)](#), [vrna_db_flatten_to\(\)](#), [vrna_pt_pk_remove\(\)](#) `VRNA_BRACKETS_RND`, `VRNA_BRACKETS_ANG`, `VRNA_BRACKETS_CLY`, `VRNA_BRACKETS_SQR`, `VRNA_BRACKETS_ALPHA`, `VRNA_BRACKETS_DEFAULT`, `VRNA_BRACKETS_ANY`

Parameters

<i>structure</i>	Secondary structure in dot-bracket-ext-notation
<i>options</i>	A bitmask to specify which brackets are recognized during conversion to pair table

Returns

A pointer to a new pair table of the provided secondary structure

SWIG Wrapper Notes This functions is wrapped as overloaded function `ptable()` that takes an optional argument `options` to specify which type of matching brackets should be considered during conversion. The default set is round brackets, i.e. `VRNA_BRACKETS_RND`.

16.65.2.3 vrna_pt_pk_get()

```
short * vrna_pt_pk_get (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table of a secondary structure (pseudo-knot version)

Returns a newly allocated table, such that `table[i]=j` if (i,j) pair or 0 if i is unpaired, `table[0]` contains the length of the structure.

In contrast to [vrna_ptable\(\)](#) this function also recognizes the base pairs denoted by '[' and ']' brackets. Thus, this function behaves like

```
vrna_ptable_from_string(structure, #VRNA_BRACKETS_RND | VRNA_BRACKETS_SQR)
```

See also

[vrna_ptable_from_string\(\)](#)

Parameters

<i>structure</i>	The secondary structure in (extended) dot-bracket notation
------------------	--

Returns

A pointer to the created pair_table

16.65.2.4 vrna_ptable_copy()

```
short * vrna_ptable_copy (
    const short * pt )
#include <ViennaRNA/utils/structures.h>
```

Get an exact copy of a pair table.

Parameters

<i>pt</i>	The pair table to be copied
-----------	-----------------------------

Returns

A pointer to the copy of 'pt'

16.65.2.5 vrna_pt_ali_get()

```
short * vrna_pt_ali_get (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table of a secondary structure (snoop align version)

16.65.2.6 vrna_pt_snoop_get()

```
short * vrna_pt_snoop_get (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table of a secondary structure (snoop version)
returns a newly allocated table, such that: table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure. The special pseudoknotted H/ACA-mRNA structure is taken into account.

16.65.2.7 vrna_pt_pk_remove()

```
short * vrna_pt_pk_remove (
    const short * ptable,
    unsigned int options )
#include <ViennaRNA/utils/structures.h>
```

Remove pseudo-knots from a pair table.

This function removes pseudo-knots from an input structure by determining the minimum number of base pairs that need to be removed to make the structure pseudo-knot free.

To accomplish that, we use a dynamic programming algorithm similar to the Nussinov maximum matching approach.

See also

[vrna_db_pk_remove\(\)](#)

Parameters

<i>ptable</i>	Input structure that may include pseudo-knots
<i>options</i>	

Returns

The input structure devoid of pseudo-knots

16.66 Pair List Representation of Secondary Structures

16.66.1 Detailed Description

Collaboration diagram for Pair List Representation of Secondary Structures:

Data Structures

- struct [vrna_elem_prob_s](#)
Data structure representing a single entry of an element probability list (e.g. list of pair probabilities) [More...](#)

Macros

- `#define VRNA_PLIST_TYPE_BASEPAIR 0`
A Base Pair element.
- `#define VRNA_PLIST_TYPE_GQUAD 1`
A G-Quadruplex element.
- `#define VRNA_PLIST_TYPE_H_MOTIF 2`
A Hairpin loop motif element.
- `#define VRNA_PLIST_TYPE_I_MOTIF 3`
An Internal loop motif element.
- `#define VRNA_PLIST_TYPE_UD_MOTIF 4`
An Unstructured Domain motif element.
- `#define VRNA_PLIST_TYPE_STACK 5`
A Base Pair stack element.
- `#define VRNA_PLIST_TYPE_UNPAIRED 6`
An unpaired base.
- `#define VRNA_PLIST_TYPE_TRIPLE 7`
One pair of a base triplet.

Typedefs

- typedef struct [vrna_elem_prob_s](#) [vrna_ep_t](#)
Convenience typedef for data structure [vrna_elem_prob_s](#).

Functions

- [vrna_ep_t](#) * [vrna_plist](#) (const char *struc, float pr)
Create a [vrna_ep_t](#) from a dot-bracket string.

16.66.2 Data Structure Documentation

16.66.2.1 struct vrna_elem_prob_s

Data structure representing a single entry of an element probability list (e.g. list of pair probabilities)

See also

[vrna_plist\(\)](#), [vrna_plist_from_probs\(\)](#), [vrna_db_from_plist\(\)](#), [VRNA_PLIST_TYPE_BASEPAIR](#), [VRNA_PLIST_TYPE_GQUAD](#), [VRNA_PLIST_TYPE_H_MOTIF](#), [VRNA_PLIST_TYPE_I_MOTIF](#), [VRNA_PLIST_TYPE_UD_MOTIF](#), [VRNA_PLIST_TYPE_STACK](#)

Data Fields

- int **i**
Start position (usually 5' nucleotide that starts the element, e.g. base pair)
- int **j**
End position (usually 3' nucleotide that ends the element, e.g. base pair)
- float **p**
Probability of the element.
- int **type**
Type of the element.

16.66.3 Function Documentation

16.66.3.1 vrna_plist()

```
vrna_ep_t * vrna_plist (
    const char * struc,
    float pr )
#include <ViennaRNA/utils/structures.h>
```

Create a [vrna_ep_t](#) from a dot-bracket string.

The dot-bracket string is parsed and for each base pair an entry in the plist is created. The probability of each pair in the list is set by a function parameter.

The end of the plist is marked by sequence positions i as well as j equal to 0. This condition should be used to stop looping over its entries

Parameters

<i>struc</i>	The secondary structure in dot-bracket notation
<i>pr</i>	The probability for each base pair used in the plist

Returns

The plist array

16.67 Abstract Shapes Representation of Secondary Structures

Abstract Shapes, introduced by Giegerich et al. in (2004) [12], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

16.67.1 Detailed Description

Abstract Shapes, introduced by Giegerich et al. in (2004) [12], collapse the secondary structure while retaining the nestedness of helices and hairpin loops.

The abstract shapes representation abstracts the structure from individual base pairs and their corresponding location in the sequence, while retaining the inherent nestedness of helices and hairpin loops.

Below is a description of what is included in the abstract shapes abstraction for each respective level together with an example structure:

```
CGUCUUAACUCAUCACCGUGUGGAGCUGCGACCCUCCUAGAUUCGAAGACGAG
(((((((...(((...))))))...(((...))...))))))..
```

Shape Level	Description	Result
1	Most accurate - all loops and all unpaired	[[_[]]_[]_↔]]] _
2	Nesting pattern for all loop types and unpaired regions in external loop and multiloop	[[_[]] [_[]_]]
3	Nesting pattern for all loop types but no unpaired regions	[[[]] [[]]]
4	Helix nesting pattern in external loop and multiloop	[[] [[]]]
5	Most abstract - helix nesting pattern and no unpaired regions	[[] [[]]]

Note

Our implementations also provide the special Shape Level 0, which does not collapse any structural features but simply convert base pairs and unpaired nucleotides into their corresponding set of symbols for abstract shapes.

Collaboration diagram for Abstract Shapes Representation of Secondary Structures:

Functions

- char * [vrna_abstract_shapes](#) (const char *structure, unsigned int level)
Convert a secondary structure in dot-bracket notation to its abstract shapes representation.
- char * [vrna_abstract_shapes_pt](#) (const short *pt, unsigned int level)
Convert a secondary structure to its abstract shapes representation.

16.67.2 Function Documentation

16.67.2.1 vrna_abstract_shapes()

```
char * vrna_abstract_shapes (
    const char * structure,
    unsigned int level )
#include <ViennaRNA/utils/structures.h>
```

Convert a secondary structure in dot-bracket notation to its abstract shapes representation.

This function converts a secondary structure into its abstract shapes representation as presented by Giegerich et al. 2004 [12].

See also

[vrna_abstract_shapes_pt\(\)](#)

Parameters

<i>structure</i>	A secondary structure in dot-bracket notation
<i>level</i>	The abstraction level (integer in the range of 0 to 5)

Returns

The secondary structure in abstract shapes notation

SWIG Wrapper Notes This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

16.67.2.2 vrna_abstract_shapes_pt()

```
char * vrna_abstract_shapes_pt (
    const short * pt,
    unsigned int level )
#include <ViennaRNA/utils/structures.h>
```

Convert a secondary structure to its abstract shapes representation.

This function converts a secondary structure into its abstract shapes representation as presented by Giegerich et al. 2004 [12]. This function is equivalent to `vrna_db_to_shapes()`, but requires a pair table input instead of a dot-bracket structure.

Note

The length of the structure must be present at `pt[0]`!

See also

[vrna_abstract_shapes\(\)](#)

Parameters

<i>pt</i>	A secondary structure in pair table format
<i>level</i>	The abstraction level (integer in the range of 0 to 5)

Returns

The secondary structure in abstract shapes notation

SWIG Wrapper Notes This function is available as an overloaded function `abstract_shapes()` where the optional second parameter `level` defaults to 5.

16.68 Helix List Representation of Secondary Structures**16.68.1 Detailed Description**

Collaboration diagram for Helix List Representation of Secondary Structures:

Data Structures

- struct [vrna_hx_s](#)
Data structure representing an entry of a helix list. [More...](#)

Typedefs

- typedef struct [vrna_hx_s](#) [vrna_hx_t](#)
Convenience typedef for data structure [vrna_hx_s](#).

Functions

- [vrna_hx_t * vrna_hx_from_ptable](#) (short *pt)

Convert a pair table representation of a secondary structure into a helix list.

- `vrna_hx_t * vrna_hx_merge` (const `vrna_hx_t *list`, int `maxdist`)

Create a merged helix list from another helix list.

16.68.2 Data Structure Documentation

16.68.2.1 struct vrna_hx_s

Data structure representing an entry of a helix list.

16.68.3 Function Documentation

16.68.3.1 vrna_hx_from_ptable()

```
vrna_hx_t * vrna_hx_from_ptable (
    short * pt )
#include <ViennaRNA/utils/structures.h>
Convert a pair table representation of a secondary structure into a helix list.
```

Parameters

<code>pt</code>	The secondary structure in pair table representation
-----------------	--

Returns

The secondary structure represented as a helix list

16.69 Tree Representation of Secondary Structures

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

16.69.1 Detailed Description

Secondary structures can be readily represented as trees, where internal nodes represent base pairs, and leaves represent unpaired nucleotides. The dot-bracket structure string already is a tree represented by a string of parenthesis (base pairs) and dots for the leaf nodes (unpaired nucleotides).

Alternatively, one may find representations with two types of node labels, P for paired and U for unpaired; a dot is then replaced by (U) , and each closed bracket is assigned an additional identifier P . We call this the expanded notation. In [10] a condensed representation of the secondary structure is proposed, the so-called homeomorphically irreducible tree (HIT) representation. Here a stack is represented as a single pair of matching brackets labeled P and weighted by the number of base pairs. Correspondingly, a contiguous strain of unpaired bases is shown as one pair of matching brackets labeled U and weighted by its length. Generally any string consisting of matching brackets and identifiers is equivalent to a plane tree with as many different types of nodes as there are identifiers.

Bruce Shapiro proposed a coarse grained representation [27], which, does not retain the full information of the secondary structure. He represents the different structure elements by single matching brackets and labels them as

- H (hairpin loop),
- I (interior loop),
- B (bulge),
- M (multi-loop), and
- S (stack).

We extend his alphabet by an extra letter for external elements **E**. Again these identifiers may be followed by a weight corresponding to the number of unpaired bases or base pairs in the structure element. All tree representations (except for the dot-bracket form) can be encapsulated into a virtual root (labeled **R**).

The following example illustrates the different linear tree representations used by the package:

Consider the secondary structure represented by the dot-bracket string (full tree) `. ((. . (((. . .))) . . ((. .)))) .` which is the most convenient condensed notation used by our programs and library functions.

Then, the following tree representations are equivalent:

- Expanded tree:
`((U) (((U) (U) (((U) (U) (U) P) P) P) (U) (U) (((U) (U) P) P) P) (U) R)`
- HIT representation (Fontana et al. 1993 [10]):
`((U1) ((U2) ((U3) P3) (U2) ((U2) P2) P2) (U1) R)`
- Coarse Grained [Tree](#) Representation (Shapiro 1988 [27]):
 - Short (with root node **R**, without stem nodes **S**):
`((H) ((H) M) R)`
 - Full (with root node **R**):
`((((H) S) ((H) S) M) S) R)`
 - Extended (with root node **R**, with external nodes **E**):
`(((((H) S) ((H) S) M) S) E) R)`
 - Weighted (with root node **R**, with external nodes **E**):
`(((((H3) S3) ((H2) S2) M4) S2) E2) R)`

The Expanded tree is rather clumsy and mostly included for the sake of completeness. The different versions of Coarse Grained [Tree](#) Representations are variations of Shapiro's linear tree notation.

For the output of aligned structures from string editing, different representations are needed, where we put the label on both sides. The above examples for tree representations would then look like:

```
* a) (UU) (P (P (P (P (UU) (UU) (P (P (P (UU) (UU) (UU) P) P) P) (UU) (UU) (P (P (UU) (U...
* b) (UU) (P2 (P2 (U2U2) (P2 (U3U3) P3) (U2U2) (P2 (U2U2) P2) P2) (UU) P2) (UU)
* c) (B (M (HH) (HH) M) B)
*   (S (B (S (M (S (HH) S) (S (HH) S) M) S) B) S)
*   (E (S (B (S (M (S (HH) S) (S (HH) S) M) S) B) S) E)
* d) (R (E2 (S2 (B1 (S2 (M4 (S3 (H3) S3) ( (H2) S2) M4) S2) B1) S2) E2) R)
*
```

Aligned structures additionally contain the gap character `_`. Collaboration diagram for Tree Representation of Secondary Structures:

Macros

- `#define VRNA_STRUCTURE_TREE_HIT 1U`
Homeomorphically Irreducible [Tree](#) (HIT) representation of a secondary structure.
- `#define VRNA_STRUCTURE_TREE_SHAPIRO_SHORT 2U`
(short) Coarse Grained representation of a secondary structure
- `#define VRNA_STRUCTURE_TREE_SHAPIRO 3U`
(full) Coarse Grained representation of a secondary structure
- `#define VRNA_STRUCTURE_TREE_SHAPIRO_EXT 4U`
(extended) Coarse Grained representation of a secondary structure
- `#define VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT 5U`
(weighted) Coarse Grained representation of a secondary structure
- `#define VRNA_STRUCTURE_TREE_EXPANDED 6U`
Expanded [Tree](#) representation of a secondary structure.

Functions

- char * [vrna_db_to_tree_string](#) (const char *structure, unsigned int type)
Convert a Dot-Bracket structure string into tree string representation.
- char * [vrna_tree_string_unweight](#) (const char *structure)
Remove weights from a linear string tree representation of a secondary structure.
- char * [vrna_tree_string_to_db](#) (const char *tree)
Convert a linear tree string representation of a secondary structure back to Dot-Bracket notation.

16.69.2 Macro Definition Documentation

16.69.2.1 VRNA_STRUCTURE_TREE_HIT

```
#define VRNA_STRUCTURE_TREE_HIT 1U
#include <ViennaRNA/utils/structures.h>
```

Homeomorphically Irreducible [Tree](#) (HIT) representation of a secondary structure.

See also

[vrna_db_to_tree_string\(\)](#)

16.69.2.2 VRNA_STRUCTURE_TREE_SHAPIRO_SHORT

```
#define VRNA_STRUCTURE_TREE_SHAPIRO_SHORT 2U
#include <ViennaRNA/utils/structures.h>
```

(short) Coarse Grained representation of a secondary structure

See also

[vrna_db_to_tree_string\(\)](#)

16.69.2.3 VRNA_STRUCTURE_TREE_SHAPIRO

```
#define VRNA_STRUCTURE_TREE_SHAPIRO 3U
#include <ViennaRNA/utils/structures.h>
```

(full) Coarse Grained representation of a secondary structure

See also

[vrna_db_to_tree_string\(\)](#)

16.69.2.4 VRNA_STRUCTURE_TREE_SHAPIRO_EXT

```
#define VRNA_STRUCTURE_TREE_SHAPIRO_EXT 4U
#include <ViennaRNA/utils/structures.h>
```

(extended) Coarse Grained representation of a secondary structure

See also

[vrna_db_to_tree_string\(\)](#)

16.69.2.5 VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT

```
#define VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT 5U
#include <ViennaRNA/utils/structures.h>
(weighted) Coarse Grained representation of a secondary structure
```

See also

[vrna_db_to_tree_string\(\)](#)

16.69.2.6 VRNA_STRUCTURE_TREE_EXPANDED

```
#define VRNA_STRUCTURE_TREE_EXPANDED 6U
#include <ViennaRNA/utils/structures.h>
Expanded Tree representation of a secondary structure.
```

See also

[vrna_db_to_tree_string\(\)](#)

16.69.3 Function Documentation

16.69.3.1 vrna_db_to_tree_string()

```
char * vrna_db_to_tree_string (
    const char * structure,
    unsigned int type )
#include <ViennaRNA/utils/structures.h>
```

Convert a Dot-Bracket structure string into tree string representation.

This function allows one to convert a secondary structure in dot-bracket notation into one of the various tree representations for secondary structures. The resulting tree is then represented as a string of parenthesis and node symbols, similar to to the Newick format.

Currently we support conversion into the following formats, denoted by the value of parameter `type`:

- [VRNA_STRUCTURE_TREE_HIT](#) - Homeomorphically Irreducible [Tree](#) (HIT) representation of a secondary structure. (See also Fontana et al. 1993 [10])
- [VRNA_STRUCTURE_TREE_SHAPIRO_SHORT](#) - (short) Coarse Grained representation of a secondary structure (same as Shapiro 1988 [27], but with root node `R` and without `S` nodes for the stems)
- [VRNA_STRUCTURE_TREE_SHAPIRO](#) - (full) Coarse Grained representation of a secondary structure (See also Shapiro 1988 [27])
- [VRNA_STRUCTURE_TREE_SHAPIRO_EXT](#) - (extended) Coarse Grained representation of a secondary structure (same as Shapiro 1988 [27], but external nodes denoted as `E`)
- [VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT](#) - (weighted) Coarse Grained representation of a secondary structure (same as [VRNA_STRUCTURE_TREE_SHAPIRO_EXT](#) but with additional weights for number of unpaired nucleotides in loop, and number of pairs in stems)
- [VRNA_STRUCTURE_TREE_EXPANDED](#) - Expanded [Tree](#) representation of a secondary structure.

See also

[Tree Representations of Secondary Structures](#)

Parameters

<i>structure</i>	The null-terminated dot-bracket structure string
<i>type</i>	A switch to determine the type of tree string representation

Returns

A tree representation of the input `structure`

16.69.3.2 `vrna_tree_string_unweight()`

```
char * vrna_tree_string_unweight (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Remove weights from a linear string tree representation of a secondary structure.

This function strips the weights of a linear string tree representation such as `HIT`, or Coarse Grained [Tree](#) sensu Shapiro [27]

See also

[vrna_db_to_tree_string\(\)](#)

Parameters

<i>structure</i>	A linear string tree representation of a secondary structure with weights
------------------	---

Returns

A linear string tree representation of a secondary structure without weights

16.69.3.3 `vrna_tree_string_to_db()`

```
char * vrna_tree_string_to_db (
    const char * tree )
#include <ViennaRNA/utils/structures.h>
```

Convert a linear tree string representation of a secondary structure back to Dot-Bracket notation.

Warning

This function only accepts *Expanded* and *HIT* tree representations!

See also

[vrna_db_to_tree_string\(\)](#), [VRNA_STRUCTURE_TREE_EXPANDED](#), [VRNA_STRUCTURE_TREE_HIT](#), [Tree Representations of Secondary Structures](#)

Parameters

<i>tree</i>	A linear tree string representation of a secondary structure
-------------	--

Returns

A dot-bracket notation of the secondary structure provided in `tree`

16.70 Distance measures between Secondary Structures**16.70.1 Detailed Description**

Collaboration diagram for Distance measures between Secondary Structures:

Functions

- int [vrna_bp_distance_pt](#) (const short *pt1, const short *pt2)
Compute the "base pair" distance between two pair tables pt1 and pt2 of secondary structures.
- int [vrna_bp_distance](#) (const char *str1, const char *str2)
Compute the "base pair" distance between two secondary structures s1 and s2.

16.70.2 Function Documentation

16.70.2.1 vrna_bp_distance_pt()

```
int vrna_bp_distance_pt (
    const short * pt1,
    const short * pt2 )
```

```
#include <ViennaRNA/utils/structures.h>
```

Compute the "base pair" distance between two pair tables pt1 and pt2 of secondary structures.

The pair tables should have the same length. dist = number of base pairs in one structure but not in the other same as edit distance with open-pair close-pair as move-set

See also

[vrna_bp_distance\(\)](#)

Parameters

<i>pt1</i>	First structure in dot-bracket notation
<i>pt2</i>	Second structure in dot-bracket notation

Returns

The base pair distance between pt1 and pt2

SWIG Wrapper Notes This function is available as an overloaded method [bp_distance\(\)](#).

16.70.2.2 vrna_bp_distance()

```
int vrna_bp_distance (
    const char * str1,
    const char * str2 )
```

```
#include <ViennaRNA/utils/structures.h>
```

Compute the "base pair" distance between two secondary structures s1 and s2.

This is a wrapper around [vrna_bp_distance_pt\(\)](#). The sequences should have the same length. dist = number of base pairs in one structure but not in the other same as edit distance with open-pair close-pair as move-set

See also

[vrna_bp_distance_pt\(\)](#)

Parameters

<i>str1</i>	First structure in dot-bracket notation
<i>str2</i>	Second structure in dot-bracket notation

Returns

The base pair distance between `str1` and `str2`

SWIG Wrapper Notes This function is available as an overloaded method `bp_distance()`. Note that the SWIG wrapper takes two structure in dot-bracket notation and converts them into pair tables using `vrna_ptable_from_string()`. The resulting pair tables are then internally passed to `vrna_bp_distance_pt()`. To control which kind of matching brackets will be used during conversion, the optional argument `options` can be used. See also the description of `vrna_ptable_from_string()` for available options. (default: `VRNA_BRACKETS_RND`).

16.71 Multiple Sequence Alignment Utilities

Functions to extract features from and to manipulate multiple sequence alignments.

16.71.1 Detailed Description

Functions to extract features from and to manipulate multiple sequence alignments.
Collaboration diagram for Multiple Sequence Alignment Utilities:

Modules

- [Deprecated Interface for Multiple Sequence Alignment Utilities](#)

Files

- file [alignments.h](#)
Various utility- and helper-functions for sequence alignments and comparative structure prediction.

Data Structures

- struct [vrna_pinfo_s](#)
A base pair info structure. [More...](#)

Macros

- `#define VRNA_ALN_DEFAULT 0U`
Use default alignment settings.
- `#define VRNA_ALN_RNA 1U`
Convert to RNA alphabet.
- `#define VRNA_ALN_DNA 2U`
Convert to DNA alphabet.
- `#define VRNA_ALN_UPPERCASE 4U`
Convert to uppercase nucleotide letters.
- `#define VRNA_ALN_LOWERCASE 8U`
Convert to lowercase nucleotide letters.
- `#define VRNA_MEASURE_SHANNON_ENTROPY 1U`
Flag indicating Shannon Entropy measure.

Typedefs

- typedef struct [vrna_pinfo_s](#) [vrna_pinfo_t](#)
Typename for the base pair info representing data structure [vrna_pinfo_s](#).

Functions

- int `vrna_aln_mpi` (const char **alignment)
Get the mean pairwise identity in steps from ?to?(ident)
- `vrna_pinfo_t * vrna_aln_pinfo` (`vrna_fold_compound_t *vc`, const char *structure, double threshold)
Retrieve an array of `vrna_pinfo_t` structures from precomputed pair probabilities.
- char ** `vrna_aln_slice` (const char **alignment, unsigned int i, unsigned int j)
Slice out a subalignment from a larger alignment.
- void `vrna_aln_free` (char **alignment)
Free memory occupied by a set of aligned sequences.
- char ** `vrna_aln_uppercase` (const char **alignment)
Create a copy of an alignment with only uppercase letters in the sequences.
- char ** `vrna_aln_toRNA` (const char **alignment)
Create a copy of an alignment where DNA alphabet is replaced by RNA alphabet.
- char ** `vrna_aln_copy` (const char **alignment, unsigned int options)
Make a copy of a multiple sequence alignment.
- float * `vrna_aln_conservation_struct` (const char **alignment, const char *structure, const `vrna_md_t *md`)
Compute base pair conservation of a consensus structure.
- float * `vrna_aln_conservation_col` (const char **alignment, const `vrna_md_t *md_p`, unsigned int options)
Compute nucleotide conservation in an alignment.
- char * `vrna_aln_consensus_sequence` (const char **alignment, const `vrna_md_t *md_p`)
Compute the consensus sequence for a given multiple sequence alignment.
- char * `vrna_aln_consensus_mis` (const char **alignment, const `vrna_md_t *md_p`)
Compute the Most Informative Sequence (MIS) for a given multiple sequence alignment.

16.71.2 Data Structure Documentation

16.71.2.1 struct `vrna_pinfo_s`

A base pair info structure.

For each base pair (i,j) with i,j in [0, n-1] the structure lists:

- its probability 'p'
- an entropy-like measure for its well-definedness 'ent'
- the frequency of each type of pair in 'bp[]'
 - 'bp[0]' contains the number of non-compatible sequences
 - 'bp[1]' the number of CG pairs, etc.

Data Fields

- unsigned **i**
nucleotide position i
- unsigned **j**
nucleotide position j
- float **p**
Probability.
- float **ent**
*Pseudo entropy for $p(i, j) = S_i + S_j - p_{ij} * \ln(p_{ij})$.*
- short **bp** [8]
Frequencies of pair_types.
- char **comp**
1 iff pair is in mfe structure

16.71.3 Macro Definition Documentation

16.71.3.1 VRNA_MEASURE_SHANNON_ENTROPY

```
#define VRNA_MEASURE_SHANNON_ENTROPY 1U
#include <ViennaRNA/utils/alignments.h>
```

Flag indicating Shannon Entropy measure.
Shannon Entropy is defined as $H = -\sum_c p_c \cdot \log_2 p_c$

16.71.4 Function Documentation

16.71.4.1 vrna_aln_mpi()

```
int vrna_aln_mpi (
    const char ** alignment )
#include <ViennaRNA/utils/alignments.h>
```

Get the mean pairwise identity in steps from ?to?(ident)

Parameters

<i>alignment</i>	Aligned sequences
------------------	-------------------

Returns

The mean pairwise identity

16.71.4.2 vrna_aln_pinfo()

```
vrna_pinfo_t * vrna_aln_pinfo (
    vrna_fold_compound_t * vc,
    const char * structure,
    double threshold )
#include <ViennaRNA/utils/alignments.h>
```

Retrieve an array of [vrna_pinfo_t](#) structures from precomputed pair probabilities.

This array of structures contains information about positionwise pair probabilities, base pair entropy and more

See also

[vrna_pinfo_t](#), and [vrna_pf\(\)](#)

Parameters

<i>vc</i>	The vrna_fold_compound_t of type VRNA_FC_TYPE_COMPARATIVE with precomputed partition function matrices
<i>structure</i>	An optional structure in dot-bracket notation (Maybe NULL)
<i>threshold</i>	Do not include results with pair probabilities below threshold

Returns

The [vrna_pinfo_t](#) array

16.71.4.3 vrna_aln_slice()

```
char ** vrna_aln_slice (
    const char ** alignment,
    unsigned int i,
    unsigned int j )
#include <ViennaRNA/utils/alignments.h>
```

Slice out a subalignment from a larger alignment.

Note

The user is responsible to free the memory occupied by the returned subalignment

See also

[vrna_aln_free\(\)](#)

Parameters

<i>alignment</i>	The input alignment
<i>i</i>	The first column of the subalignment (1-based)
<i>j</i>	The last column of the subalignment (1-based)

Returns

The subalignment between column *i* and *j*

16.71.4.4 vrna_aln_free()

```
void vrna_aln_free (
    char ** alignment )
#include <ViennaRNA/utils/alignments.h>
```

Free memory occupied by a set of aligned sequences.

Parameters

<i>alignment</i>	The input alignment
------------------	---------------------

16.71.4.5 vrna_aln_uppercase()

```
char ** vrna_aln_uppercase (
    const char ** alignment )
#include <ViennaRNA/utils/alignments.h>
```

Create a copy of an alignment with only uppercase letters in the sequences.

See also

[vrna_aln_copy](#)

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
------------------	--

Returns

A copy of the input alignment where lowercase sequence letters are replaced by uppercase letters

16.71.4.6 vrna_aln_toRNA()

```
char ** vrna_aln_toRNA (
    const char ** alignment )
#include <ViennaRNA/utils/alignments.h>
```

Create a copy of an alignment where DNA alphabet is replaced by RNA alphabet.

See also

[vrna_aln_copy](#)

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
------------------	--

Returns

A copy of the input alignment where DNA alphabet is replaced by RNA alphabet (T -> U)

16.71.4.7 vrna_aln_copy()

```
char ** vrna_aln_copy (
    const char ** alignment,
    unsigned int options )
#include <ViennaRNA/utils/alignments.h>
```

Make a copy of a multiple sequence alignment.

This function allows one to create a copy of a multiple sequence alignment. The *options* parameter additionally allows for sequence manipulation, such as converting DNA to RNA alphabet, and conversion to uppercase letters.

See also

[vrna_aln_copy\(\)](#), [VRNA_ALN_RNA](#), [VRNA_ALN_UPPERCASE](#), [VRNA_ALN_DEFAULT](#)

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
<i>options</i>	Option flags indicating whether the aligned sequences should be converted

Returns

A (manipulated) copy of the input alignment

16.71.4.8 vrna_aln_conservation_struct()

```
float * vrna_aln_conservation_struct (
    const char ** alignment,
    const char * structure,
    const vrna_md_t * md )
#include <ViennaRNA/utils/alignments.h>
```

Compute base pair conservation of a consensus structure.

This function computes the base pair conservation (fraction of canonical base pairs) of a consensus structure given a multiple sequence alignment. The base pair types that are considered canonical may be specified using the [vrna_md_t.pair](#) array. Passing *NULL* as parameter *md* results in default pairing rules, i.e. canonical Watson-Crick and GU Wobble pairs.

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
<i>structure</i>	The consensus structure in dot-bracket notation
<i>md</i>	Model details that specify compatible base pairs (Maybe <i>NULL</i>)

Returns

A 1-based vector of base pair conservations

SWIG Wrapper Notes This function is available in an overloaded form where the last parameter may be omitted, indicating *md* = *NULL*

16.71.4.9 vrna_aln_conservation_col()

```
float * vrna_aln_conservation_col (
    const char ** alignment,
    const vrna_md_t * md,
    unsigned int options )
#include <ViennaRNA/utils/alignments.h>
```

Compute nucleotide conservation in an alignment.

This function computes the conservation of nucleotides in alignment columns. The simplest measure is Shannon Entropy and can be selected by passing the [VRNA_MEASURE_SHANNON_ENTROPY](#) flag in the *options* parameter.

Note

Currently, only [VRNA_MEASURE_SHANNON_ENTROPY](#) is supported as conservation measure.

See also

[VRNA_MEASURE_SHANNON_ENTROPY](#)

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
<i>md</i>	Model details that specify known nucleotides (Maybe <i>NULL</i>)
<i>options</i>	A flag indicating which measure of conservation should be applied

Returns

A 1-based vector of column conservations

SWIG Wrapper Notes This function is available in an overloaded form where the last two parameters may be omitted, indicating *md* = *NULL*, and *options* = [VRNA_MEASURE_SHANNON_ENTROPY](#), respectively.

16.71.4.10 vrna_aln_consensus_sequence()

```
char * vrna_aln_consensus_sequence (
```



```

    const char ** alignment,
    const vrna_md_t * md_p )
#include <ViennaRNA/utils/alignments.h>
Compute the consensus sequence for a given multiple sequence alignment.

```

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
<i>md_p</i>	Model details that specify known nucleotides (Maybe <i>NULL</i>)

Returns

The consensus sequence of the alignment, i.e. the most frequent nucleotide for each alignment column

16.71.4.11 vrna_aln_consensus_mis()

```

char * vrna_aln_consensus_mis (
    const char ** alignment,
    const vrna_md_t * md_p )
#include <ViennaRNA/utils/alignments.h>
Compute the Most Informative Sequence (MIS) for a given multiple sequence alignment.
The most informative sequence (MIS) [11] displays for each alignment column the nucleotides with frequency
greater than the background frequency, projected into IUPAC notation. Columns where gaps are over-represented
are in lower case.

```

Parameters

<i>alignment</i>	The input sequence alignment (last entry must be <i>NULL</i> terminated)
<i>md_p</i>	Model details that specify known nucleotides (Maybe <i>NULL</i>)

Returns

The most informative sequence for the alignment

16.72 Files and I/O

Functions to parse, write, and convert various file formats and to deal with file system related issues.

16.72.1 Detailed Description

Functions to parse, write, and convert various file formats and to deal with file system related issues.
 Collaboration diagram for Files and I/O:

Modules

- [Nucleic Acid Sequences and Structures](#)
Functions to read/write different file formats for nucleic acid sequences and secondary structures.
- [Multiple Sequence Alignments](#)
Functions to read/write multiple sequence alignments (MSA) in various file formats.
- [Command Files](#)
Functions to parse and interpret the content of [Command Files](#).

Files

- file [commands.h](#)
Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.
- file [ribo.h](#)
Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments.
- file [file_formats.h](#)
Read and write different file formats for RNA sequences, structures.
- file [file_formats_msa.h](#)
Functions dealing with file formats for Multiple Sequence Alignments (MSA)
- file [utils.h](#)
Several utilities for file handling.

Functions

- float ** [get_ribosum](#) (const char **Alseq, int n_seq, int length)
Retrieve a RiboSum Scoring Matrix for a given Alignment.
- float ** [readribosum](#) (char *name)
Read a RiboSum or other user-defined Scoring Matrix and Store into global Memory.
- void [vrna_file_copy](#) (FILE *from, FILE *to)
Inefficient 'cp'.
- char * [vrna_read_line](#) (FILE *fp)
Read a line of arbitrary length from a stream.
- int [vrna_mkdir_p](#) (const char *path)
Recursively create a directory tree.
- char * [vrna_basename](#) (const char *path)
Extract the filename from a file path.
- char * [vrna_dirname](#) (const char *path)
Extract the directory part of a file path.
- char * [vrna_filename_sanitize](#) (const char *name, const char *replacement)
Sanitize a file name.
- int [vrna_file_exists](#) (const char *filename)
Check if a file already exists in the file system.

16.72.2 Function Documentation

16.72.2.1 [get_ribosum\(\)](#)

```
float ** get_ribosum (
    const char ** Alseq,
    int n_seq,
    int length )
#include <ViennaRNA/ribo.h>
Retrieve a RiboSum Scoring Matrix for a given Alignment.
```

16.72.2.2 [readribosum\(\)](#)

```
float ** readribosum (
    char * name )
#include <ViennaRNA/ribo.h>
Read a RiboSum or other user-defined Scoring Matrix and Store into global Memory.
```

16.72.2.3 vrna_read_line()

```
char * vrna_read_line (
    FILE * fp )
#include <ViennaRNA/io/utils.h>
```

Read a line of arbitrary length from a stream.

Returns a pointer to the resulting string. The necessary memory is allocated and should be released using *free()* when the string is no longer needed.

Parameters

<i>fp</i>	A file pointer to the stream where the function should read from
-----------	--

Returns

A pointer to the resulting string

16.72.2.4 vrna_filename_sanitize()

```
char * vrna_filename_sanitize (
    const char * name,
    const char * replacement )
#include <ViennaRNA/io/utils.h>
```

Sanitize a file name.

Returns a new file name where all invalid characters are substituted by a replacement character. If no replacement character is supplied, invalid characters are simply removed from the filename. File names may also never exceed a length of 255 characters. Longer file names will undergo a 'smart' truncation process, where the filenames' suffix, i.e. everything after the last dot '.', is attempted to be kept intact. Hence, only the filename part before the suffix is reduced in such a way that the total filename complies to the length restriction of 255 characters. If no suffix is present or the suffix itself already exceeds the maximum length, the filename is simply truncated from the back of the string.

For now we consider the following characters invalid:

- backslash '\'
- slash '/'
- question mark '?'
- percent sign '%'
- asterisk '*'
- colon ':'
- pipe symbol '|'
- double quote '"'
- triangular brackets '<' and '>'

Furthermore, the (resulting) file name must not be a reserved file name, such as:

- '.'
- '..'

Note

This function allocates a new block of memory for the sanitized string. It also may return (a) NULL if the input is pointing to NULL, or (b) an empty string if the input only consists of invalid characters which are simply removed!

Parameters

<i>name</i>	The input file name
<i>replacement</i>	The replacement character, or NULL

Returns

The sanitized file name, or NULL

16.72.2.5 vrna_file_exists()

```
int vrna_file_exists (
    const char * filename )
#include <ViennaRNA/io/utils.h>
Check if a file already exists in the file system.
```

Parameters

<i>filename</i>	The name of (path to) the file to check for existence
-----------------	---

Returns

0 if it doesn't exist, 1 otherwise

16.73 Nucleic Acid Sequences and Structures

Functions to read/write different file formats for nucleic acid sequences and secondary structures.

16.73.1 Detailed Description

Functions to read/write different file formats for nucleic acid sequences and secondary structures.
Collaboration diagram for Nucleic Acid Sequences and Structures:

Files

- file [file_formats.h](#)
Read and write different file formats for RNA sequences, structures.

Macros

- #define [VRNA_OPTION_MULTILINE](#) 32U
Tell a function that an input is assumed to span several lines.
- #define [VRNA_CONSTRAINT_MULTILINE](#) 32U
parse multiline constraint

Functions

- void [vrna_file_helixlist](#) (const char *seq, const char *db, float energy, FILE *file)
Print a secondary structure as helix list.
- void [vrna_file_connect](#) (const char *seq, const char *db, float energy, const char *identifier, FILE *file)
Print a secondary structure as connect table.
- void [vrna_file_bpseq](#) (const char *seq, const char *db, FILE *file)
Print a secondary structure in bpseq format.

- void [vrna_file_json](#) (const char *seq, const char *db, double energy, const char *identifier, FILE *file)
Print a secondary structure in jsonformat.
- unsigned int [vrna_file_fasta_read_record](#) (char **header, char **sequence, char ***rest, FILE *file, unsigned int options)
- char * [vrna_extract_record_rest_structure](#) (const char **lines, unsigned int length, unsigned int option)
Extract a dot-bracket structure string from (multiline)character array.
- int [vrna_file_SHAPE_read](#) (const char *file_name, int length, double default_value, char *sequence, double *values)
Read data from a given SHAPE reactivity input file.
- void [vrna_extract_record_rest_constraint](#) (char **cstruc, const char **lines, unsigned int option)
Extract a hard constraint encoded as pseudo dot-bracket string.
- unsigned int [read_record](#) (char **header, char **sequence, char ***rest, unsigned int options)
Get a data record from stdin.

16.73.2 Macro Definition Documentation

16.73.2.1 VRNA_OPTION_MULTILINE

```
#define VRNA_OPTION_MULTILINE 32U
```

```
#include <ViennaRNA/io/file_formats.h>
```

Tell a function that an input is assumed to span several lines.

If used as input-option a function might also be returning this state telling that it has read data from multiple lines.

See also

[vrna_extract_record_rest_structure\(\)](#), [vrna_file_fasta_read_record\(\)](#)

16.73.2.2 VRNA_CONSTRAINT_MULTILINE

```
#define VRNA_CONSTRAINT_MULTILINE 32U
```

```
#include <ViennaRNA/io/file_formats.h>
```

parse multiline constraint

Deprecated see [vrna_extract_record_rest_structure\(\)](#)

16.73.3 Function Documentation

16.73.3.1 vrna_file_helixlist()

```
void vrna_file_helixlist (
    const char * seq,
    const char * db,
    float energy,
    FILE * file )
```

```
#include <ViennaRNA/io/file_formats.h>
```

Print a secondary structure as helix list.

Parameters

<i>seq</i>	The RNA sequence
<i>db</i>	The structure in dot-bracket format
<i>energy</i>	Free energy of the structure in kcal/mol
<i>file</i>	The file handle used to print to (print defaults to 'stdout' if(file == NULL))

16.73.3.2 vrna_file_connect()

```
void vrna_file_connect (
    const char * seq,
    const char * db,
    float energy,
    const char * identifier,
    FILE * file )
```

#include <ViennaRNA/io/file_formats.h>

Print a secondary structure as connect table.

Connect table file format looks like this:

```
* 300 ENERGY = 7.0 example
* 1 G      0   2   22   1
* 2 G      1   3   21   2
*
```

where the headerline is followed by 6 columns with:

1. Base number: index n
2. Base (A, C, G, T, U, X)
3. Index n-1 (0 if first nucleotide)
4. Index n+1 (0 if last nucleotide)
5. Number of the base to which n is paired. No pairing is indicated by 0 (zero).
6. Natural numbering.

Parameters

<i>seq</i>	The RNA sequence
<i>db</i>	The structure in dot-bracket format
<i>energy</i>	The free energy of the structure
<i>identifier</i>	An optional identifier for the sequence
<i>file</i>	The file handle used to print to (print defaults to 'stdout' if(file == NULL))

16.73.3.3 vrna_file_bpseq()

```
void vrna_file_bpseq (
    const char * seq,
    const char * db,
    FILE * file )
```

#include <ViennaRNA/io/file_formats.h>

Print a secondary structure in bpseq format.

Parameters

<i>seq</i>	The RNA sequence
<i>db</i>	The structure in dot-bracket format
<i>file</i>	The file handle used to print to (print defaults to 'stdout' if(file == NULL))

16.73.3.4 vrna_file_json()

```
void vrna_file_json (
    const char * seq,
    const char * db,
    double energy,
    const char * identifier,
    FILE * file )
#include <ViennaRNA/io/file_formats.h>
Print a secondary structure in jsonformat.
```

Parameters

<i>seq</i>	The RNA sequence
<i>db</i>	The structure in dot-bracket format
<i>energy</i>	The free energy
<i>identifier</i>	An identifier for the sequence
<i>file</i>	The file handle used to print to (print defaults to 'stdout' if(file == NULL))

16.73.3.5 vrna_file_fasta_read_record()

```
unsigned int vrna_file_fasta_read_record (
    char ** header,
    char ** sequence,
    char *** rest,
    FILE * file,
    unsigned int options )
#include <ViennaRNA/io/file_formats.h>
@brief Get a (fasta) data set from a file or stdin
```

This function may be used to obtain complete datasets from a filehandle or stdin. A dataset is always defined to contain at least a sequence. If data starts with a fasta header, i.e. a line like

```
@verbatim >some header info @endverbatim
```

then `vrna_file_fasta_read_record()` will assume that the sequence that follows the header may span over several lines. To disable this behavior and to assign a single line to the argument 'sequence' one can pass `#VRNA_INPUT_NO_SPAN` in the 'options' argument. If no fasta header is read in the beginning of a data block, a sequence must not span over multiple lines!\n

Unless the options `#VRNA_INPUT_NOSKIP_COMMENTS` or `#VRNA_INPUT_NOSKIP_BLANK_LINES` are passed, a sequence may be interrupted by lines starting with a comment character or empty lines.\n

A sequence is regarded as completely read if it was either assumed to not span over multiple lines, a secondary structure or structure constraint follows the sequence on the next line, or a new header marks the beginning of a new sequence...\n

All lines following the sequence (this includes comments) that do not initiate a new dataset according to the above definition are available through the line-array 'rest'. Here one can usually find the structure constraint or other information belonging to the current dataset. Filling of 'rest' may be prevented by passing `#VRNA_INPUT_NO_REST` to the options argument.\n

@note This function will exit any program with an error message if no sequence could be read!
 @note This function is NOT threadsafe! It uses a global variable to store information about the next data block.

The main purpose of this function is to be able to easily parse blocks of data in the header of a loop where all calculations for the appropriate data is done inside the loop. The loop may be then left on certain return values, e.g.:

```
@code
```

```
char *id, *seq, **rest; int i; id = seq = NULL; rest = NULL; while(!(vrna_file_fasta_read_record(&id, &seq, &rest,
NULL, 0) & (VRNA_INPUT_ERROR | VRNA_INPUT_QUIT))){ if(id) printf("%s\n", id); printf("%s\n", seq); if(rest)
for(i=0;rest[i];i++){ printf("%s\n", rest[i]); free(rest[i]); } free(rest); free(seq); free(id); } In the example above, the
```

while loop will be terminated when [vrna_file_fasta_read_record\(\)](#) returns either an error, EOF, or a user initiated quit request.

As long as data is read from stdin (we are passing NULL as the file pointer), the id is printed if it is available for the current block of data. The sequence will be printed in any case and if some more lines belong to the current block of data each line will be printed as well.

Note

Do not forget to free the memory occupied by header, sequence and rest!

Parameters

<i>header</i>	A pointer which will be set such that it points to the header of the record
<i>sequence</i>	A pointer which will be set such that it points to the sequence of the record
<i>rest</i>	A pointer which will be set such that it points to an array of lines which also belong to the record
<i>file</i>	A file handle to read from (if NULL, this function reads from stdin)
<i>options</i>	Some options which may be passed to alter the behavior of the function, use 0 for no options

Returns

A flag with information about what the function actually did read

16.73.3.6 vrna_extract_record_rest_structure()

```
char * vrna_extract_record_rest_structure (
    const char ** lines,
    unsigned int length,
    unsigned int option )
#include <ViennaRNA/io/file_formats.h>
```

Extract a dot-bracket structure string from (multiline)character array.

This function extracts a dot-bracket structure string from the 'rest' array as returned by [vrna_file_fasta_read_record\(\)](#) and returns it. All occurrences of comments within the 'lines' array will be skipped as long as they do not break the structure string. If no structure could be read, this function returns NULL.

Precondition

The argument 'lines' has to be a 2-dimensional character array as obtained by [vrna_file_fasta_read_record\(\)](#)

See also

[vrna_file_fasta_read_record\(\)](#)

Parameters

<i>lines</i>	The (multiline) character array to be parsed
<i>length</i>	The assumed length of the dot-bracket string (passing a value < 1 results in no length limit)
<i>option</i>	Some options which may be passed to alter the behavior of the function, use 0 for no options

Returns

The dot-bracket string read from lines or NULL

16.73.3.7 vrna_file_SHAPE_read()

```
int vrna_file_SHAPE_read (
    const char * file_name,
    int length,
    double default_value,
    char * sequence,
    double * values )
#include <ViennaRNA/io/file_formats.h>
```

Read data from a given SHAPE reactivity input file.

This function parses the informations from a given file and stores the result in the preallocated string sequence and the double array values.

Parameters

<i>file_name</i>	Path to the constraints file
<i>length</i>	Length of the sequence (file entries exceeding this limit will cause an error)
<i>default_value</i>	Value for missing indices
<i>sequence</i>	Pointer to an array used for storing the sequence obtained from the SHAPE reactivity file
<i>values</i>	Pointer to an array used for storing the values obtained from the SHAPE reactivity file

16.73.3.8 vrna_extract_record_rest_constraint()

```
void vrna_extract_record_rest_constraint (
    char ** cstruc,
    const char ** lines,
    unsigned int option )
#include <ViennaRNA/io/file_formats.h>
```

Extract a hard constraint encoded as pseudo dot-bracket string.

Deprecated Use [vrna_extract_record_rest_structure\(\)](#) instead!

Precondition

The argument 'lines' has to be a 2-dimensional character array as obtained by [vrna_file_fasta_read_record\(\)](#)

See also

[vrna_file_fasta_read_record\(\)](#), [VRNA_CONSTRAINT_DB_PIPE](#), [VRNA_CONSTRAINT_DB_DOT](#), [VRNA_CONSTRAINT_DB_VRNA_CONSTRAINT_DB_ANG_BRACK](#), [VRNA_CONSTRAINT_DB_RND_BRACK](#)

Parameters

<i>cstruc</i>	A pointer to a character array that is used as pseudo dot-bracket output
<i>lines</i>	A 2-dimensional character array with the extension lines from the FASTA input
<i>option</i>	The option flags that define the behavior and recognition pattern of this function

16.73.3.9 read_record()

```
unsigned int read_record (
    char ** header,
    char ** sequence,
    char *** rest,
    unsigned int options )
#include <ViennaRNA/io/file_formats.h>
Get a data record from stdin.
```

Deprecated This function is deprecated! Use `vrna_file_fasta_read_record()` as a replacment.

16.74 Multiple Sequence Alignments

Functions to read/write multiple sequence alignments (MSA) in various file formats.

16.74.1 Detailed Description

Functions to read/write multiple sequence alignments (MSA) in various file formats.
Collaboration diagram for Multiple Sequence Alignments:

Files

- file [file_formats_msa.h](#)
Functions dealing with file formats for Multiple Sequence Alignments (MSA)

Macros

- #define [VRNA_FILE_FORMAT_MSA_CLUSTAL](#) 1U
Option flag indicating ClustalW formatted files.
- #define [VRNA_FILE_FORMAT_MSA_STOCKHOLM](#) 2U
Option flag indicating Stockholm 1.0 formatted files.
- #define [VRNA_FILE_FORMAT_MSA_FASTA](#) 4U
Option flag indicating FASTA (Pearson) formatted files.
- #define [VRNA_FILE_FORMAT_MSA_MAF](#) 8U
Option flag indicating MAF formatted files.
- #define [VRNA_FILE_FORMAT_MSA_MIS](#) 16U
Option flag indicating most informative sequence (MIS) output.
- #define [VRNA_FILE_FORMAT_MSA_DEFAULT](#)
Option flag indicating the set of default file formats.
- #define [VRNA_FILE_FORMAT_MSA_NOCHECK](#) 4096U
Option flag to disable validation of the alignment.
- #define [VRNA_FILE_FORMAT_MSA_UNKNOWN](#) 8192U
Return flag of [vrna_file_msa_detect_format\(\)](#) to indicate unknown or malformed alignment.
- #define [VRNA_FILE_FORMAT_MSA_APPEND](#) 16384U
Option flag indicating to append data to a multiple sequence alignment file rather than overwriting it.
- #define [VRNA_FILE_FORMAT_MSA_QUIET](#) 32768U
Option flag to suppress unnecessary spam messages on `stderr`
- #define [VRNA_FILE_FORMAT_MSA_SILENT](#) 65536U
Option flag to completely silence any warnings on `stderr`

Functions

- int [vrna_file_msa_read](#) (const char *filename, char ***names, char ***aln, char **id, char **structure, unsigned int options)
Read a multiple sequence alignment from file.
- int [vrna_file_msa_read_record](#) (FILE *fp, char ***names, char ***aln, char **id, char **structure, unsigned int options)
Read a multiple sequence alignment from file handle.
- unsigned int [vrna_file_msa_detect_format](#) (const char *filename, unsigned int options)
Detect the format of a multiple sequence alignment file.
- int [vrna_file_msa_write](#) (const char *filename, const char **names, const char **aln, const char *id, const char *structure, const char *source, unsigned int options)
Write multiple sequence alignment file.

16.74.2 Macro Definition Documentation

16.74.2.1 VRNA_FILE_FORMAT_MSA_CLUSTAL

```
#define VRNA_FILE_FORMAT_MSA_CLUSTAL 1U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag indicating ClustalW formatted files.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#), [vrna_file_msa_detect_format\(\)](#)

16.74.2.2 VRNA_FILE_FORMAT_MSA_STOCKHOLM

```
#define VRNA_FILE_FORMAT_MSA_STOCKHOLM 2U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag indicating Stockholm 1.0 formatted files.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#), [vrna_file_msa_detect_format\(\)](#)

16.74.2.3 VRNA_FILE_FORMAT_MSA_FASTA

```
#define VRNA_FILE_FORMAT_MSA_FASTA 4U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag indicating FASTA (Pearson) formatted files.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#), [vrna_file_msa_detect_format\(\)](#)

16.74.2.4 VRNA_FILE_FORMAT_MSA_MAF

```
#define VRNA_FILE_FORMAT_MSA_MAF 8U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag indicating MAF formatted files.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#), [vrna_file_msa_detect_format\(\)](#)

16.74.2.5 VRNA_FILE_FORMAT_MSA_MIS

```
#define VRNA_FILE_FORMAT_MSA_MIS 16U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag indicating most informative sequence (MIS) output.

The default reference sequence output for an alignment is simply a consensus sequence. This flag allows to write the most informative sequence (MIS) instead.

See also

[vrna_file_msa_write\(\)](#)

16.74.2.6 VRNA_FILE_FORMAT_MSA_DEFAULT

```
#define VRNA_FILE_FORMAT_MSA_DEFAULT
#include <ViennaRNA/io/file_formats_msa.h>
```

Value:

```
( \
  VRNA_FILE_FORMAT_MSA_CLUSTAL \
| VRNA_FILE_FORMAT_MSA_STOCKHOLM \
| VRNA_FILE_FORMAT_MSA_FASTA \
| VRNA_FILE_FORMAT_MSA_MAF \
)
```

Option flag indicating the set of default file formats.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#), [vrna_file_msa_detect_format\(\)](#)

16.74.2.7 VRNA_FILE_FORMAT_MSA_NOCHECK

```
#define VRNA_FILE_FORMAT_MSA_NOCHECK 4096U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag to disable validation of the alignment.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#)

16.74.2.8 VRNA_FILE_FORMAT_MSA_UNKNOWN

```
#define VRNA_FILE_FORMAT_MSA_UNKNOWN 8192U
#include <ViennaRNA/io/file_formats_msa.h>
```

Return flag of [vrna_file_msa_detect_format\(\)](#) to indicate unknown or malformed alignment.

See also

[vrna_file_msa_detect_format\(\)](#)

16.74.2.9 VRNA_FILE_FORMAT_MSA_APPEND

```
#define VRNA_FILE_FORMAT_MSA_APPEND 16384U
#include <ViennaRNA/io/file_formats_msa.h>
```

Option flag indicating to append data to a multiple sequence alignment file rather than overwriting it.

See also

[vrna_file_msa_write\(\)](#)

16.74.2.10 VRNA_FILE_FORMAT_MSA_QUIET

```
#define VRNA_FILE_FORMAT_MSA_QUIET 32768U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag to suppress unnecessary spam messages on stderr
```

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#)

16.74.2.11 VRNA_FILE_FORMAT_MSA_SILENT

```
#define VRNA_FILE_FORMAT_MSA_SILENT 65536U
#include <ViennaRNA/io/file_formats_msa.h>
Option flag to completely silence any warnings on stderr
```

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_read_record\(\)](#)

16.74.3 Function Documentation

16.74.3.1 vrna_file_msa_read()

```
int vrna_file_msa_read (
    const char * filename,
    char *** names,
    char *** aln,
    char ** id,
    char ** structure,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
```

Read a multiple sequence alignment from file.

This function reads the (first) multiple sequence alignment from an input file. The read alignment is split into the sequence id/name part and the actual sequence information and stored in memory as arrays of ids/names and sequences. If the alignment file format allows for additional information, such as an ID of the entire alignment or consensus structure information, this data is retrieved as well and made available. The `options` parameter allows to specify the set of alignment file formats that should be used to retrieve the data. If 0 is passed as option, the list of alignment file formats defaults to [VRNA_FILE_FORMAT_MSA_DEFAULT](#).

Currently, the list of parsable multiple sequence alignment file formats consists of:

- [ClustalW format](#)
- [Stockholm 1.0 format](#)
- [FASTA \(Pearson\) format](#)
- [MAF format](#)

Note

After successfully reading an alignment, this function performs a validation of the data that includes uniqueness of the sequence identifiers, and equal sequence lengths. This check can be deactivated by passing [VRNA_FILE_FORMAT_MSA_NOCHECK](#) in the `options` parameter.

It is the users responsibility to free any memory occupied by the output arguments `names`, `aln`, `id`, and `structure` after calling this function. The function automatically sets the latter two arguments to `NULL` in case no corresponding data could be retrieved from the input alignment.

See also

[vrna_file_msa_read_record\(\)](#), [VRNA_FILE_FORMAT_MSA_CLUSTAL](#), [VRNA_FILE_FORMAT_MSA_STOCKHOLM](#), [VRNA_FILE_FORMAT_MSA_FASTA](#), [VRNA_FILE_FORMAT_MSA_MAF](#), [VRNA_FILE_FORMAT_MSA_DEFAULT](#), [VRNA_FILE_FORMAT_MSA_NOCHECK](#)

Parameters

<i>filename</i>	The name of input file that contains the alignment
<i>names</i>	An address to the pointer where sequence identifiers should be written to
<i>aln</i>	An address to the pointer where aligned sequences should be written to
<i>id</i>	An address to the pointer where the alignment ID should be written to (Maybe NULL)
<i>structure</i>	An address to the pointer where consensus structure information should be written to (Maybe NULL)
<i>options</i>	Options to manipulate the behavior of this function

Returns

The number of sequences in the alignment, or -1 if no alignment record could be found

SWIG Wrapper Notes In the target scripting language, only the first and last argument, `filename` and `options`, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

```
num_seq, names, aln, id, structure = RNA.file_msa_read("msa.stk", RNA.FILE_FORMAT_MSA_STOCKHOLM)
```

After successfully reading the first record, the variable `num_seq` contains the number of sequences in the alignment (the actual return value of the C-function), while the variables `names`, `aln`, `id`, and `structure` are lists of the sequence names and aligned sequences, as well as strings holding the alignment ID and the structure as stated in the `SS_cons` line, respectively. Note, the last two return values may be empty strings in case the alignment does not provide the required data.

This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to [VRNA_FILE_FORMAT_MSA_STOCKHOLM](#).

16.74.3.2 vrna_file_msa_read_record()

```
int vrna_file_msa_read_record (
    FILE * fp,
    char *** names,
    char *** aln,
    char ** id,
    char ** structure,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
```

Read a multiple sequence alignment from file handle.

Similar to [vrna_file_msa_read\(\)](#), this function reads a multiple sequence alignment from an input file handle. Since using a file handle, this function is not limited to the first alignment record, but allows for looping over all alignments within the input.

The read alignment is split into the sequence id/name part and the actual sequence information and stored in memory as arrays of ids/names and sequences. If the alignment file format allows for additional information, such as an ID of the entire alignment or consensus structure information, this data is retrieved as well and made available. The `options` parameter allows to specify the alignment file format used to retrieve the data. A single format must be specified here, see [vrna_file_msa_detect_format\(\)](#) for helping to determine the correct MSA file format.

Currently, the list of parsable multiple sequence alignment file formats consists of:

- [ClustalW format](#)
- [Stockholm 1.0 format](#)
- [FASTA \(Pearson\) format](#)
- [MAF format](#)

Note

After successfully reading an alignment, this function performs a validation of the data that includes uniqueness of the sequence identifiers, and equal sequence lengths. This check can be deactivated by passing [VRNA_FILE_FORMAT_MSA_NOCHECK](#) in the `options` parameter.

It is the users responsibility to free any memory occupied by the output arguments `names`, `aln`, `id`, and `structure` after calling this function. The function automatically sets the latter two arguments to `NULL` in case no corresponding data could be retrieved from the input alignment.

See also

[vrna_file_msa_read\(\)](#), [vrna_file_msa_detect_format\(\)](#), [VRNA_FILE_FORMAT_MSA_CLUSTAL](#), [VRNA_FILE_FORMAT_MSA_STOCKHOLM](#), [VRNA_FILE_FORMAT_MSA_FASTA](#), [VRNA_FILE_FORMAT_MSA_MAF](#), [VRNA_FILE_FORMAT_MSA_DEFAULT](#), [VRNA_FILE_FORMAT_MSA_NOCHECK](#)

Parameters

<i>fp</i>	The file pointer the data will be retrieved from
<i>names</i>	An address to the pointer where sequence identifiers should be written to
<i>aln</i>	An address to the pointer where aligned sequences should be written to
<i>id</i>	An address to the pointer where the alignment ID should be written to (Maybe NULL)
<i>structure</i>	An address to the pointer where consensus structure information should be written to (Maybe NULL)
<i>options</i>	Options to manipulate the behavior of this function

Returns

The number of sequences in the alignment, or -1 if no alignment record could be found

SWIG Wrapper Notes In the target scripting language, only the first and last argument, `fp` and `options`, are passed to the corresponding function. The other arguments, which serve as output in the C-library, are available as additional return values. Hence, a function call in python may look like this:

```
f = open('msa.stk', 'r')
num_seq, names, aln, id, structure = RNA.file_msa_read_record(f, RNA.FILE_FORMAT_MSA_STOCKHOLM)
f.close()
```

After successfully reading the first record, the variable `num_seq` contains the number of sequences in the alignment (the actual return value of the C-function), while the variables `names`, `aln`, `id`, and `structure` are lists of the sequence names and aligned sequences, as well as strings holding the alignment ID and the structure as stated in the `SS_cons` line, respectively. Note, the last two return values may be empty strings in case the alignment does not provide the required data.

This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to [VRNA_FILE_FORMAT_MSA_STOCKHOLM](#).

16.74.3.3 vrna_file_msa_detect_format()

```
unsigned int vrna_file_msa_detect_format (
    const char * filename,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
```

Detect the format of a multiple sequence alignment file.

This function attempts to determine the format of a file that supposedly contains a multiple sequence alignment (MSA). This is useful in cases where a MSA file contains more than a single record and therefore [vrna_file_msa_read\(\)](#) can not be applied, since it only retrieves the first. Here, one can try to guess the correct file format using this function and then loop over the file, record by record using one of the low-level record retrieval functions for the corresponding MSA file format.

Note

This function parses the entire first record within the specified file. As a result, it returns [VRNA_FILE_FORMAT_MSA_UNKNOWN](#) not only if it can't detect the file's format, but also in cases where the file doesn't contain sequences!

See also

[vrna_file_msa_read\(\)](#), [vrna_file_stockholm_read_record\(\)](#), [vrna_file_clustal_read_record\(\)](#), [vrna_file_fasta_read_record\(\)](#)

Parameters

<i>filename</i>	The name of input file that contains the alignment
<i>options</i>	Options to manipulate the behavior of this function

Returns

The MSA file format, or [VRNA_FILE_FORMAT_MSA_UNKNOWN](#)

SWIG Wrapper Notes This function exists as an overloaded version where the `options` parameter may be omitted! In that case, the `options` parameter defaults to [VRNA_FILE_FORMAT_MSA_DEFAULT](#).

16.74.3.4 vrna_file_msa_write()

```
int vrna_file_msa_write (
    const char * filename,
    const char ** names,
    const char ** aln,
    const char * id,
    const char * structure,
    const char * source,
    unsigned int options )
#include <ViennaRNA/io/file_formats_msa.h>
Write multiple sequence alignment file.
```

Note

Currently, we only support [Stockholm 1.0 format](#) output

See also

[VRNA_FILE_FORMAT_MSA_STOCKHOLM](#), [VRNA_FILE_FORMAT_MSA_APPEND](#), [VRNA_FILE_FORMAT_MSA_MIS](#)

Parameters

<i>filename</i>	The output filename
<i>names</i>	The array of sequence names / identifies
<i>aln</i>	The array of aligned sequences
<i>id</i>	An optional ID for the alignment
<i>structure</i>	An optional consensus structure
<i>source</i>	A string describing the source of the alignment
<i>options</i>	Options to manipulate the behavior of this function

Returns

Non-null upon successfully writing the alignment to file

SWIG Wrapper Notes In the target scripting language, this function exists as a set of overloaded versions, where the last four parameters may be omitted. If the `options` parameter is missing the options default to (`VRNA_FILE_FORMAT_MSA_STOCKHOLM` | `VRNA_FILE_FORMAT_MSA_APPEND`).

16.75 Command Files

Functions to parse and interpret the content of [Command Files](#).

16.75.1 Detailed Description

Functions to parse and interpret the content of [Command Files](#).

Collaboration diagram for Command Files:

Files

- file [commands.h](#)
Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.

Macros

- `#define VRNA_CMD_PARSE_HC 1U`
Command parse/apply flag indicating hard constraints.
- `#define VRNA_CMD_PARSE_SC 2U`
Command parse/apply flag indicating soft constraints.
- `#define VRNA_CMD_PARSE_UD 4U`
Command parse/apply flag indicating unstructured domains.
- `#define VRNA_CMD_PARSE_SD 8U`
Command parse/apply flag indicating structured domains.
- `#define VRNA_CMD_PARSE_DEFAULTS`
Command parse/apply flag indicating default set of commands.

Typedefs

- `typedef struct vrna_command_s * vrna_cmd_t`
A data structure that contains commands.

Functions

- `vrna_cmd_t vrna_file_commands_read` (const char *filename, unsigned int options)
Extract a list of commands from a command file.
- `int vrna_file_commands_apply` (`vrna_fold_compound_t` *vc, const char *filename, unsigned int options)
Apply a list of commands from a command file.
- `int vrna_commands_apply` (`vrna_fold_compound_t` *vc, `vrna_cmd_t` commands, unsigned int options)
Apply a list of commands to a [vrna_fold_compound_t](#).
- `void vrna_commands_free` (`vrna_cmd_t` commands)
Free memory occupied by a list of commands.

16.75.2 Macro Definition Documentation

16.75.2.1 VRNA_CMD_PARSE_HC

```
#define VRNA_CMD_PARSE_HC 1U
#include <ViennaRNA/commands.h>
```

Command parse/apply flag indicating hard constraints.

See also

[vrna_cmd_t](#), [vrna_file_commands_read\(\)](#), [vrna_file_commands_apply\(\)](#), [vrna_commands_apply\(\)](#)

16.75.2.2 VRNA_CMD_PARSE_SC

```
#define VRNA_CMD_PARSE_SC 2U
#include <ViennaRNA/commands.h>
```

Command parse/apply flag indicating soft constraints.

See also

[vrna_cmd_t](#), [vrna_file_commands_read\(\)](#), [vrna_file_commands_apply\(\)](#), [vrna_commands_apply\(\)](#)

16.75.2.3 VRNA_CMD_PARSE_UD

```
#define VRNA_CMD_PARSE_UD 4U
#include <ViennaRNA/commands.h>
```

Command parse/apply flag indicating unstructured domains.

See also

[vrna_cmd_t](#), [vrna_file_commands_read\(\)](#), [vrna_file_commands_apply\(\)](#), [vrna_commands_apply\(\)](#)

16.75.2.4 VRNA_CMD_PARSE_SD

```
#define VRNA_CMD_PARSE_SD 8U
#include <ViennaRNA/commands.h>
```

Command parse/apply flag indicating structured domains.

See also

[vrna_cmd_t](#), [vrna_file_commands_read\(\)](#), [vrna_file_commands_apply\(\)](#), [vrna_commands_apply\(\)](#)

16.75.2.5 VRNA_CMD_PARSE_DEFAULTS

```
#define VRNA_CMD_PARSE_DEFAULTS
#include <ViennaRNA/commands.h>
```

Value:

```
(VRNA_CMD_PARSE_HC \
| VRNA_CMD_PARSE_SC \
| VRNA_CMD_PARSE_UD \
| VRNA_CMD_PARSE_SD \
)
```

Command parse/apply flag indicating default set of commands.

See also

[vrna_cmd_t](#), [vrna_file_commands_read\(\)](#), [vrna_file_commands_apply\(\)](#), [vrna_commands_apply\(\)](#)

16.75.3 Function Documentation

16.75.3.1 vrna_file_commands_read()

```
vrna_cmd_t vrna_file_commands_read (
    const char * filename,
    unsigned int options )
#include <ViennaRNA/commands.h>
```

Extract a list of commands from a command file.

Read a list of commands specified in the input file and return them as list of abstract commands

See also

[vrna_commands_apply\(\)](#), [vrna_file_commands_apply\(\)](#), [vrna_commands_free\(\)](#)

Parameters

<i>filename</i>	The filename
<i>options</i>	Options to limit the type of commands read from the file

Returns

A list of abstract commands

16.75.3.2 vrna_file_commands_apply()

```
int vrna_file_commands_apply (
    vrna_fold_compound_t * vc,
    const char * filename,
    unsigned int options )
#include <ViennaRNA/commands.h>
```

Apply a list of commands from a command file.

This function is a shortcut to directly parse a commands file and apply all successfully parsed commands to a [vrna_fold_compound_t](#) data structure. It is the same as:

```
int r;
struct vrna_command_s *cmds;

cmds = vrna_file_commands_read(filename, options);
r = vrna_commands_apply(vc, cmds, options);

vrna_commands_free(cmds);

return r;
```

Parameters

<i>vc</i>	The vrna_fold_compound_t the command list will be applied to
<i>filename</i>	The filename
<i>options</i>	Options to limit the type of commands read from the file

Returns

The number of commands successfully applied

SWIG Wrapper Notes This function is attached as method **file_commands_apply()** to objects of type *fold_compound*

16.75.3.3 vrna_commands_apply()

```
int vrna_commands_apply (
    vrna_fold_compound_t * vc,
```

```

        vrna_cmd_t commands,
        unsigned int options )
#include <ViennaRNA/commands.h>
Apply a list of commands to a vrna\_fold\_compound\_t.

```

Parameters

<i>vc</i>	The vrna_fold_compound_t the command list will be applied to
<i>commands</i>	The commands to apply
<i>options</i>	Options to limit the type of commands read from the file

Returns

The number of commands successfully applied

16.75.3.4 [vrna_commands_free\(\)](#)

```

void vrna_commands_free (
        vrna_cmd_t commands )
#include <ViennaRNA/commands.h>
Free memory occupied by a list of commands.
Release memory occupied by a list of commands

```

Parameters

<i>commands</i>	A pointer to a list of commands
-----------------	---------------------------------

16.76 Plotting

Functions for Creating Secondary Structure Plots, Dot-Plots, and More.

16.76.1 Detailed Description

Functions for Creating Secondary Structure Plots, Dot-Plots, and More.
Collaboration diagram for Plotting:

Modules

- [Layouts and Coordinates](#)
Functions to compute coordinate layouts for secondary structure plots.
- [Annotation](#)
Functions to generate annotations for Secondary Structure Plots, Dot-Plots, and Others.
- [Alignment Plots](#)
Functions to generate Alignment plots with annotated consensus structure.
- [Deprecated Interface for Plotting Utilities](#)

Files

- file [alignments.h](#)
Various functions for plotting Sequence / Structure Alignments.
- file [layouts.h](#)
Secondary structure plot layout algorithms.

- file [probabilities.h](#)
Various functions for plotting RNA secondary structures, dot-plots and other visualizations.
- file [structures.h](#)
Various functions for plotting RNA secondary structures.
- file [utils.h](#)
Various utilities to assist in plotting secondary structures and consensus structures.
- file [RNApuzzler.h](#)
Implementation of the RNApuzzler RNA secondary structure layout algorithm [30].
- file [RNAturtle.h](#)
Implementation of the RNAturtle RNA secondary structure layout algorithm [30].

Data Structures

- struct [vrna_dotplot_auxdata_t](#)

Functions

- int [PS_dot_plot_list](#) (char *seq, char *filename, [vrna_ep_t](#) *pl, [vrna_ep_t](#) *mf, char *comment)
Produce a postscript dot-plot from two pair lists.
- int [PS_dot_plot](#) (char *string, char *file)
Produce postscript dot-plot.
- int [vrna_file_PS_rnaplot](#) (const char *seq, const char *structure, const char *file, [vrna_md_t](#) *md_p)
Produce a secondary structure graph in PostScript and write it to 'filename'.
- int [vrna_file_PS_rnaplot_a](#) (const char *seq, const char *structure, const char *file, const char *pre, const char *post, [vrna_md_t](#) *md_p)
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.
- int [gmlRNA](#) (char *string, char *structure, char *ssfile, char option)
Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.
- int [ssv_rna_plot](#) (char *string, char *structure, char *ssfile)
Produce a secondary structure graph in SStructView format.
- int [svg_rna_plot](#) (char *string, char *structure, char *ssfile)
Produce a secondary structure plot in SVG format and write it to a file.
- int [xrna_plot](#) (char *string, char *structure, char *ssfile)
Produce a secondary structure plot for further editing in XRNA.
- int [PS_rna_plot](#) (char *string, char *structure, char *file)
Produce a secondary structure graph in PostScript and write it to 'filename'.
- int [PS_rna_plot_a](#) (char *string, char *structure, char *file, char *pre, char *post)
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.
- int [PS_rna_plot_a_gquad](#) (char *string, char *structure, char *ssfile, char *pre, char *post)
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename' (detect and draw g-quadruplexes)

16.76.2 Data Structure Documentation

16.76.2.1 struct [vrna_dotplot_auxdata_t](#)

Collaboration diagram for [vrna_dotplot_auxdata_t](#):

16.76.3 Function Documentation

16.76.3.1 PS_dot_plot_list()

```
int PS_dot_plot_list (
    char * seq,
    char * filename,
    vrna_ep_t * pl,
    vrna_ep_t * mf,
    char * comment )
#include <ViennaRNA/plotting/probabilities.h>
```

Produce a postscript dot-plot from two pair lists.

This function reads two plist structures (e.g. base pair probabilities and a secondary structure) as produced by [assign_plist_from_pr\(\)](#) and [assign_plist_from_db\(\)](#) and produces a postscript "dot plot" that is written to 'filename'. Using base pair probabilities in the first and mfe structure in the second plist, the resulting "dot plot" represents each base pairing probability by a square of corresponding area in a upper triangle matrix. The lower part of the matrix contains the minimum free energy structure.

See also

[assign_plist_from_pr\(\)](#), [assign_plist_from_db\(\)](#)

Parameters

<i>seq</i>	The RNA sequence
<i>filename</i>	A filename for the postscript output
<i>pl</i>	The base pair probability pairlist
<i>mf</i>	The mfe secondary structure pairlist
<i>comment</i>	A comment

Returns

1 if postscript was successfully written, 0 otherwise

16.76.3.2 PS_dot_plot()

```
int PS_dot_plot (
    char * string,
    char * file )
#include <ViennaRNA/plotting/probabilities.h>
```

Produce postscript dot-plot.

Wrapper to [PS_dot_plot_list](#)

Reads base pair probabilities produced by [pf_fold\(\)](#) from the global array [pr](#) and the pair list [base_pair](#) produced by [fold\(\)](#) and produces a postscript "dot plot" that is written to 'filename'. The "dot plot" represents each base pairing probability by a square of corresponding area in a upper triangle matrix. The lower part of the matrix contains the minimum free energy

Note

DO NOT USE THIS FUNCTION ANYMORE SINCE IT IS NOT THREADSAFE

Deprecated This function is deprecated and will be removed soon! Use [PS_dot_plot_list\(\)](#) instead!

16.76.3.3 vrna_file_PS_rnaplot()

```
int vrna_file_PS_rnaplot (
    const char * seq,
    const char * structure,
```

```

    const char * file,
    vrna_md_t * md_p )
#include <ViennaRNA/plotting/structures.h>

```

Produce a secondary structure graph in PostScript and write it to 'filename'.

Note that this function has changed from previous versions and now expects the structure to be plotted in dot-bracket notation as an argument. It does not make use of the global [base_pair](#) array anymore.

Parameters

<i>seq</i>	The RNA sequence
<i>structure</i>	The secondary structure in dot-bracket notation
<i>file</i>	The filename of the postscript output
<i>md_p</i>	Model parameters used to generate a commandline option string in the output (Maybe NULL)

Returns

1 on success, 0 otherwise

16.76.3.4 vrna_file_PS_rnaplot_a()

```

int vrna_file_PS_rnaplot_a (
    const char * seq,
    const char * structure,
    const char * file,
    const char * pre,
    const char * post,
    vrna_md_t * md_p )
#include <ViennaRNA/plotting/structures.h>

```

Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'. Same as [vrna_file_PS_rnaplot\(\)](#) but adds extra PostScript macros for various annotations (see generated PS code). The 'pre' and 'post' variables contain PostScript code that is verbatim copied in the resulting PS file just before and after the structure plot. If both arguments ('pre' and 'post') are NULL, no additional macros will be printed into the PostScript.

Parameters

<i>seq</i>	The RNA sequence
<i>structure</i>	The secondary structure in dot-bracket notation
<i>file</i>	The filename of the postscript output
<i>pre</i>	PostScript code to appear before the secondary structure plot
<i>post</i>	PostScript code to appear after the secondary structure plot
<i>md_p</i>	Model parameters used to generate a commandline option string in the output (Maybe NULL)

Returns

1 on success, 0 otherwise

16.76.3.5 gmlRNA()

```

int gmlRNA (
    char * string,
    char * structure,

```

```

        char * ssfile,
        char option )
#include <ViennaRNA/plotting/structures.h>

```

Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.

If 'option' is an uppercase letter the RNA sequence is used to label nodes, if 'option' equals 'X' or 'x' the resulting file will contain coordinates for an initial layout of the graph.

Parameters

<i>string</i>	The RNA sequence
<i>structure</i>	The secondary structure in dot-bracket notation
<i>ssfile</i>	The filename of the gml output
<i>option</i>	The option flag

Returns

1 on success, 0 otherwise

16.76.3.6 ssv_rna_plot()

```

int ssv_rna_plot (
    char * string,
    char * structure,
    char * ssfile )
#include <ViennaRNA/plotting/structures.h>

```

Produce a secondary structure graph in SStructView format.

Write coord file for SStructView

Parameters

<i>string</i>	The RNA sequence
<i>structure</i>	The secondary structure in dot-bracket notation
<i>ssfile</i>	The filename of the ssv output

Returns

1 on success, 0 otherwise

16.76.3.7 svg_rna_plot()

```

int svg_rna_plot (
    char * string,
    char * structure,
    char * ssfile )
#include <ViennaRNA/plotting/structures.h>

```

Produce a secondary structure plot in SVG format and write it to a file.

Parameters

<i>string</i>	The RNA sequence
<i>structure</i>	The secondary structure in dot-bracket notation
<i>ssfile</i>	The filename of the svg output

Returns

1 on success, 0 otherwise

16.76.3.8 xrna_plot()

```
int xrna_plot (
    char * string,
    char * structure,
    char * ssfile )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure plot for further editing in XRNA.
```

Parameters

<i>string</i>	The RNA sequence
<i>structure</i>	The secondary structure in dot-bracket notation
<i>ssfile</i>	The filename of the xrna output

Returns

1 on success, 0 otherwise

16.76.3.9 PS_rna_plot()

```
int PS_rna_plot (
    char * string,
    char * structure,
    char * file )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure graph in PostScript and write it to 'filename'.
```

Deprecated Use `vrna_file_PS_rnaplot()` instead!

16.76.3.10 PS_rna_plot_a()

```
int PS_rna_plot_a (
    char * string,
    char * structure,
    char * file,
    char * pre,
    char * post )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.
```

Deprecated Use `vrna_file_PS_rnaplot_a()` instead!

16.76.3.11 PS_rna_plot_a_gquad()

```
int PS_rna_plot_a_gquad (
    char * string,
    char * structure,
```

```

        char * ssfile,
        char * pre,
        char * post )
#include <ViennaRNA/plotting/structures.h>
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'
(detect and draw g-quadruplexes)

```

Deprecated Use `vrna_file_PS_rnaplot_a()` instead!

16.77 Layouts and Coordinates

Functions to compute coordinate layouts for secondary structure plots.

16.77.1 Detailed Description

Functions to compute coordinate layouts for secondary structure plots.
Collaboration diagram for Layouts and Coordinates:

Data Structures

- struct [vrna_plot_layout_s](#)
- struct [vrna_plot_options_puzzler_t](#)
Options data structure for RNApuzzler algorithm implementation. [More...](#)

Macros

- #define [VRNA_PLOT_TYPE_SIMPLE](#) 0
Definition of Plot type simple
- #define [VRNA_PLOT_TYPE_NAVIEW](#) 1
Definition of Plot type Naview
- #define [VRNA_PLOT_TYPE_CIRCULAR](#) 2
Definition of Plot type Circular
- #define [VRNA_PLOT_TYPE_TURTLE](#) 3
Definition of Plot type Turtle [30].
- #define [VRNA_PLOT_TYPE_PUZZLER](#) 4
Definition of Plot type RNApuzzler [30].

Typedefs

- typedef struct [vrna_plot_layout_s](#) [vrna_plot_layout_t](#)
RNA secondary structure figure layout.

Functions

- [vrna_plot_layout_t](#) * [vrna_plot_layout](#) (const char *structure, unsigned int plot_type)
Create a layout (coordinates, etc.) for a secondary structure plot.
- [vrna_plot_layout_t](#) * [vrna_plot_layout_simple](#) (const char *structure)
Create a layout (coordinates, etc.) for a simple secondary structure plot.
- [vrna_plot_layout_t](#) * [vrna_plot_layout_circular](#) (const char *structure)
Create a layout (coordinates, etc.) for a circular secondary structure plot.
- [vrna_plot_layout_t](#) * [vrna_plot_layout_turtle](#) (const char *structure)
Create a layout (coordinates, etc.) for a secondary structure plot using the Turtle Algorithm [30].
- [vrna_plot_layout_t](#) * [vrna_plot_layout_puzzler](#) (const char *structure, [vrna_plot_options_puzzler_t](#) *options)
Create a layout (coordinates, etc.) for a secondary structure plot using the RNApuzzler Algorithm [30].

- void [vrna_plot_layout_free](#) ([vrna_plot_layout_t](#) *layout)
Free memory occupied by a figure layout data structure.
- int [vrna_plot_coords](#) (const char *structure, float **x, float **y, int plot_type)
Compute nucleotide coordinates for secondary structure plot.
- int [vrna_plot_coords_pt](#) (const short *pt, float **x, float **y, int plot_type)
Compute nucleotide coordinates for secondary structure plot.
- int [vrna_plot_coords_simple](#) (const char *structure, float **x, float **y)
Compute nucleotide coordinates for secondary structure plot the Simple way
- int [vrna_plot_coords_simple_pt](#) (const short *pt, float **x, float **y)
Compute nucleotide coordinates for secondary structure plot the Simple way
- int [vrna_plot_coords_circular](#) (const char *structure, float **x, float **y)
Compute coordinates of nucleotides mapped in equal distances onto a unit circle.
- int [vrna_plot_coords_circular_pt](#) (const short *pt, float **x, float **y)
Compute nucleotide coordinates for a Circular Plot
- int [vrna_plot_coords_puzzler](#) (const char *structure, float **x, float **y, double **arc_coords, [vrna_plot_options_puzzler_t](#) *options)
Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [30].
- int [vrna_plot_coords_puzzler_pt](#) (short const *const pair_table, float **x, float **y, double **arc_coords, [vrna_plot_options_puzzler_t](#) *puzzler)
Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [30].
- [vrna_plot_options_puzzler_t](#) * [vrna_plot_options_puzzler](#) (void)
Create an RNApuzzler options data structure.
- void [vrna_plot_options_puzzler_free](#) ([vrna_plot_options_puzzler_t](#) *options)
Free memory occupied by an RNApuzzler options data structure.
- int [vrna_plot_coords_turtle](#) (const char *structure, float **x, float **y, double **arc_coords)
Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [30].
- int [vrna_plot_coords_turtle_pt](#) (short const *const pair_table, float **x, float **y, double **arc_coords)
Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [30].

16.77.2 Data Structure Documentation

16.77.2.1 struct vrna_plot_layout_s

16.77.2.2 struct vrna_plot_options_puzzler_t

Options data structure for RNApuzzler algorithm implementation.

16.77.3 Macro Definition Documentation

16.77.3.1 VRNA_PLOT_TYPE_SIMPLE

```
#define VRNA_PLOT_TYPE_SIMPLE 0
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *simple*

This is the plot type definition for several RNA structure plotting functions telling them to use **Simple** plotting algorithm

See also

[rna_plot_type](#), [vrna_file_PS_rnaplot_a\(\)](#), [vrna_file_PS_rnaplot\(\)](#), [svg_rna_plot\(\)](#), [gmlRNA\(\)](#), [ssv_rna_plot\(\)](#), [xrna_plot\(\)](#)

16.77.3.2 VRNA_PLOT_TYPE_NAVIEW

```
#define VRNA_PLOT_TYPE_NAVIEW 1
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *Naview*

This is the plot type definition for several RNA structure plotting functions telling them to use **Naview** plotting algorithm [6].

See also

[rna_plot_type](#), [vrna_file_PS_rnaplot_a\(\)](#), [vrna_file_PS_rnaplot\(\)](#), [svg_rna_plot\(\)](#), [gmlRNA\(\)](#), [ssv_rna_plot\(\)](#), [xrna_plot\(\)](#)

16.77.3.3 VRNA_PLOT_TYPE_CIRCULAR

```
#define VRNA_PLOT_TYPE_CIRCULAR 2
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *Circular*

This is the plot type definition for several RNA structure plotting functions telling them to produce a **Circular plot**

See also

[rna_plot_type](#), [vrna_file_PS_rnaplot_a\(\)](#), [vrna_file_PS_rnaplot\(\)](#), [svg_rna_plot\(\)](#), [gmlRNA\(\)](#), [ssv_rna_plot\(\)](#), [xrna_plot\(\)](#)

16.77.3.4 VRNA_PLOT_TYPE_TURTLE

```
#define VRNA_PLOT_TYPE_TURTLE 3
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *Turtle* [30].

16.77.3.5 VRNA_PLOT_TYPE_PUZZLER

```
#define VRNA_PLOT_TYPE_PUZZLER 4
#include <ViennaRNA/plotting/layouts.h>
```

Definition of Plot type *RNApuzzler* [30].

16.77.4 Typedef Documentation

16.77.4.1 vrna_plot_layout_t

```
typedef struct vrna_plot_layout_s vrna_plot_layout_t
#include <ViennaRNA/plotting/layouts.h>
```

RNA secondary structure figure layout.

See also

[vrna_plot_layout\(\)](#), [vrna_plot_layout_free\(\)](#), [vrna_plot_layout_simple\(\)](#), [vrna_plot_layout_circular\(\)](#), [vrna_plot_layout_naview\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_layout_puzzler\(\)](#)

16.77.5 Function Documentation

16.77.5.1 vrna_plot_layout()

```
vrna_plot_layout_t * vrna_plot_layout (
    const char * structure,
    unsigned int plot_type )
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a secondary structure plot.

This function can be used to create a secondary structure nucleotide layout that is then further processed by an actual plotting function. The layout algorithm can be specified using the `plot_type` parameter, and the following algorithms are currently supported:

- [VRNA_PLOT_TYPE_SIMPLE](#)
- [VRNA_PLOT_TYPE_NAVIEW](#)
- [VRNA_PLOT_TYPE_CIRCULAR](#)
- [VRNA_PLOT_TYPE_TURTLE](#)
- [VRNA_PLOT_TYPE_PUZZLER](#)

Passing an unsupported selection leads to the default algorithm [VRNA_PLOT_TYPE_NAVIEW](#)

Note

If only X-Y coordinates of the corresponding structure layout are required, consider using [vrna_plot_coords\(\)](#) instead!

See also

[vrna_plot_layout_free\(\)](#), [vrna_plot_layout_simple\(\)](#), [vrna_plot_layout_naview\(\)](#), [vrna_plot_layout_circular\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_layout_puzzler\(\)](#), [vrna_plot_coords\(\)](#), [vrna_file_PS_rnaplot_layout\(\)](#)

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
<i>plot_type</i>	The layout algorithm to be used

Returns

The layout data structure for the provided secondary structure

16.77.5.2 vrna_plot_layout_simple()

```
vrna_plot_layout_t * vrna_plot_layout_simple (
    const char * structure )
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a *simple* secondary structure plot.

This function basically is a wrapper to [vrna_plot_layout\(\)](#) that passes the `plot_type` [VRNA_PLOT_TYPE_SIMPLE](#).

Note

If only X-Y coordinates of the corresponding structure layout are required, consider using [vrna_plot_coords_simple\(\)](#) instead!

See also

[vrna_plot_layout_free\(\)](#), [vrna_plot_layout\(\)](#), [vrna_plot_layout_naview\(\)](#), [vrna_plot_layout_circular\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_layout_puzzler\(\)](#), [vrna_plot_coords_simple\(\)](#), [vrna_file_PS_rnaplot_layout\(\)](#)

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
------------------	---

Returns

The layout data structure for the provided secondary structure

16.77.5.3 vrna_plot_layout_circular()

```
vrna_plot_layout_t * vrna_plot_layout_circular (
    const char * structure )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a *circular* secondary structure plot.

This function basically is a wrapper to [vrna_plot_layout\(\)](#) that passes the `plot_type` `VRNA_PLOT_TYPE_CIRCULAR`.

Note

If only X-Y coordinates of the corresponding structure layout are required, consider using [vrna_plot_coords_circular\(\)](#) instead!

See also

[vrna_plot_layout_free\(\)](#), [vrna_plot_layout\(\)](#), [vrna_plot_layout_navigate\(\)](#), [vrna_plot_layout_simple\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_layout_puzzler\(\)](#), [vrna_plot_coords_circular\(\)](#), [vrna_file_PS_rnaplot_layout\(\)](#)

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
------------------	---

Returns

The layout data structure for the provided secondary structure

16.77.5.4 vrna_plot_layout_turtle()

```
vrna_plot_layout_t * vrna_plot_layout_turtle (
    const char * structure )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a secondary structure plot using the *Turtle Algorithm* [30].

This function basically is a wrapper to [vrna_plot_layout\(\)](#) that passes the `plot_type` `VRNA_PLOT_TYPE_TURTLE`.

Note

If only X-Y coordinates of the corresponding structure layout are required, consider using [vrna_plot_coords_turtle\(\)](#) instead!

See also

[vrna_plot_layout_free\(\)](#), [vrna_plot_layout\(\)](#), [vrna_plot_layout_simple\(\)](#), [vrna_plot_layout_circular\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_layout_navigate\(\)](#), [vrna_plot_layout_puzzler\(\)](#), [vrna_plot_coords_turtle\(\)](#), [vrna_file_PS_rnaplot_layout\(\)](#)

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
------------------	---

Returns

The layout data structure for the provided secondary structure

16.77.5.5 vrna_plot_layout_puzzler()

```
vrna_plot_layout_t * vrna_plot_layout_puzzler (
    const char * structure,
    vrna_plot_options_puzzler_t * options )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Create a layout (coordinates, etc.) for a secondary structure plot using the *RNApuzzler Algorithm* [30].

This function basically is a wrapper to [vrna_plot_layout\(\)](#) that passes the `plot_type` `VRNA_PLOT_TYPE_PUZZLER`.

Note

If only X-Y coordinates of the corresponding structure layout are required, consider using [vrna_plot_coords_puzzler\(\)](#) instead!

See also

[vrna_plot_layout_free\(\)](#), [vrna_plot_layout\(\)](#), [vrna_plot_layout_simple\(\)](#), [vrna_plot_layout_circular\(\)](#), [vrna_plot_layout_navigate\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_coords_puzzler\(\)](#), [vrna_file_PS_rnaplot_layout\(\)](#)

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
------------------	---

Returns

The layout data structure for the provided secondary structure

16.77.5.6 vrna_plot_layout_free()

```
void vrna_plot_layout_free (
    vrna_plot_layout_t * layout )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Free memory occupied by a figure layout data structure.

See also

[vrna_plot_layout_t](#), [vrna_plot_layout\(\)](#), [vrna_plot_layout_simple\(\)](#), [vrna_plot_layout_circular\(\)](#), [vrna_plot_layout_navigate\(\)](#), [vrna_plot_layout_turtle\(\)](#), [vrna_plot_layout_puzzler\(\)](#), [vrna_file_PS_rnaplot_layout\(\)](#)

Parameters

<i>layout</i>	The layout data structure to free
---------------	-----------------------------------

16.77.5.7 vrna_plot_coords()

```
int vrna_plot_coords (
    const char * structure,
    float ** x,
    float ** y,
    int plot_type )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Compute nucleotide coordinates for secondary structure plot.

This function takes a secondary structure and computes X-Y coordinates for each nucleotide that then can be used to create a structure plot. The parameter `plot_type` is used to select the underlying layout algorithm. Currently, the following selections are provided:

- [VRNA_PLOT_TYPE_SIMPLE](#)
- [VRNA_PLOT_TYPE_NAVIEW](#)
- [VRNA_PLOT_TYPE_CIRCULAR](#)
- [VRNA_PLOT_TYPE_TURTLE](#)
- [VRNA_PLOT_TYPE_PUZZLER](#)

Passing an unsupported selection leads to the default algorithm [VRNA_PLOT_TYPE_NAVIEW](#)

Here is a simple example how to use this function, assuming variable `structure` contains a valid dot-bracket string:

```
float *x, *y;

if (vrna_plot_coords(structure, &x, &y)) {
    printf("all fine");
} else {
    printf("some failure occurred!");
}

free(x);
free(y);
```

Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addresses `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords_pt\(\)](#), [vrna_plot_coords_simple\(\)](#), [vrna_plot_coords_naview\(\)](#) [vrna_plot_coords_circular\(\)](#), [vrna_plot_coords_turtle\(\)](#), [vrna_plot_coords_puzzler\(\)](#)

Parameters

	<i>structure</i>	The secondary structure in dot-bracket notation
in, out	<i>x</i>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
in, out	<i>y</i>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)
	<i>plot_type</i>	The layout algorithm to be used

Returns

The length of the structure on success, 0 otherwise

16.77.5.8 vrna_plot_coords_pt()

```
int vrna_plot_coords_pt (
    const short * pt,
    float ** x,
    float ** y,
    int plot_type )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Compute nucleotide coordinates for secondary structure plot.

Same as [vrna_plot_coords\(\)](#) but takes a pair table with the structure information as input.

Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addressess `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords\(\)](#), [vrna_plot_coords_simple_pt\(\)](#), [vrna_plot_coords_navview_pt\(\)](#), [vrna_plot_coords_circular_pt\(\)](#), [vrna_plot_coords_turtle_pt\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#)

Parameters

	<i>pt</i>	The pair table that holds the secondary structure
<i>in, out</i>	<i>x</i>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<i>in, out</i>	<i>y</i>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)
	<i>plot_type</i>	The layout algorithm to be used

Returns

The length of the structure on success, 0 otherwise

16.77.5.9 vrna_plot_coords_simple()

```
int vrna_plot_coords_simple (
    const char * structure,
    float ** x,
    float ** y )
```

#include <[ViennaRNA/plotting/layouts.h](#)>

Compute nucleotide coordinates for secondary structure plot the *Simple way*

This function basically is a wrapper to [vrna_plot_coords\(\)](#) that passes the `plot_type` [VRNA_PLOT_TYPE_SIMPLE](#).

Here is a simple example how to use this function, assuming variable `structure` contains a valid dot-bracket string:

```
float *x, *y;

if (vrna_plot_coords_simple(structure, &x, &y)) {
    printf("all fine");
} else {
    printf("some failure occured!");
}

free(x);
free(y);
```

Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addressess `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords\(\)](#), [vrna_plot_coords_simple_pt\(\)](#), [vrna_plot_coords_circular\(\)](#), [vrna_plot_coords_navview\(\)](#), [vrna_plot_coords_turtle\(\)](#), [vrna_plot_coords_puzzler\(\)](#)

Parameters

	<i>structure</i>	The secondary structure in dot-bracket notation
--	------------------	---

Parameters

<code>in, out</code>	<code>x</code>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<code>in, out</code>	<code>y</code>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)

Returns

The length of the structure on success, 0 otherwise

16.77.5.10 vrna_plot_coords_simple_pt()

```
int vrna_plot_coords_simple_pt (
    const short * pt,
    float ** x,
    float ** y )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Compute nucleotide coordinates for secondary structure plot the *Simple way*

Same as [vrna_plot_coords_simple\(\)](#) but takes a pair table with the structure information as input.

Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addressess `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords_pt\(\)](#), [vrna_plot_coords_simple\(\)](#), [vrna_plot_coords_circular_pt\(\)](#), [vrna_plot_coords_↔naview_pt\(\)](#), [vrna_plot_coords_turtle_pt\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#)

Parameters

	<code>pt</code>	The pair table that holds the secondary structure
<code>in, out</code>	<code>x</code>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<code>in, out</code>	<code>y</code>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)

Returns

The length of the structure on success, 0 otherwise

16.77.5.11 vrna_plot_coords_circular()

```
int vrna_plot_coords_circular (
    const char * structure,
    float ** x,
    float ** y )
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Compute coordinates of nucleotides mapped in equal distancies onto a unit circle.

This function basically is a wrapper to [vrna_plot_coords\(\)](#) that passes the `plot_type` [VRNA_PLOT_TYPE_CIRCULAR](#).

In order to draw nice arcs using quadratic bezier curves that connect base pairs one may calculate a second tangential point P^t in addition to the actual R^2 coordinates. the simplest way to do so may be to compute a radius scaling factor rs in the interval $[0, 1]$ that weights the proportion of base pair span to the actual length of the

sequence. This scaling factor can then be used to calculate the coordinates for P^t , i.e.

$$P_x^t[i] = X[i] * rs$$

and

$$P_y^t[i] = Y[i] * rs$$

Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addressess `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords\(\)](#), [vrna_plot_coords_circular_pt\(\)](#), [vrna_plot_coords_simple\(\)](#), [vrna_plot_coords_naview\(\)](#), [vrna_plot_coords_turtle\(\)](#), [vrna_plot_coords_puzzler\(\)](#)

Parameters

	<i>structure</i>	The secondary structure in dot-bracket notation
<code>in, out</code>	<code>x</code>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<code>in, out</code>	<code>y</code>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)

Returns

The length of the structure on success, 0 otherwise

16.77.5.12 vrna_plot_coords_circular_pt()

```
int vrna_plot_coords_circular_pt (
    const short * pt,
    float ** x,
    float ** y )
```

#include <ViennaRNA/plotting/layouts.h>

Compute nucleotide coordinates for a *Circular Plot*

Same as [vrna_plot_coords_circular\(\)](#) but takes a pair table with the structure information as input.

Note

On success, this function allocates memory for X and Y coordinates and assigns the pointers at addressess `x` and `y` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords_pt\(\)](#), [vrna_plot_coords_circular\(\)](#), [vrna_plot_coords_simple_pt\(\)](#), [vrna_plot_coords_naview_pt\(\)](#), [vrna_plot_coords_turtle_pt\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#)

Parameters

	<i>pt</i>	The pair table that holds the secondary structure
<code>in, out</code>	<code>x</code>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<code>in, out</code>	<code>y</code>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)

Returns

The length of the structure on success, 0 otherwise

16.77.5.13 vrna_plot_coords_puzzler()

```
int vrna_plot_coords_puzzler (
    const char * structure,
    float ** x,
    float ** y,
    double ** arc_coords,
    vrna_plot_options_puzzler_t * options )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [30].
This function basically is a wrapper to vrna\_plot\_coords\(\) that passes the plot_type VRNA\_PLOT\_TYPE\_PUZZLER.
Here is a simple example how to use this function, assuming variable structure contains a valid dot-bracket
string and using the default options (options = NULL):
float *x, *y;
double *arcs;

if (vrna_plot_coords_puzzler(structure, &x, &y, &arcs, NULL)) {
    printf("all fine");
} else {
    printf("some failure occured!");
}

free(x);
free(y);
free(arcs);
```

Note

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#), [vrna_plot_coords_circular\(\)](#), [vrna_plot_coords_simple\(\)](#), [vrna_plot_coords_turtle\(\)](#), [vrna_plot_coords_naview\(\)](#), [vrna_plot_options_puzzler\(\)](#)

Parameters

	<i>structure</i>	The secondary structure in dot-bracket notation
in, out	<i>x</i>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
in, out	<i>y</i>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)
in, out	<i>arc_coords</i>	The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure)
	<i>options</i>	The options for the RNApuzzler algorithm (or NULL)

Returns

The length of the structure on success, 0 otherwise

16.77.5.14 vrna_plot_coords_puzzler_pt()

```
int vrna_plot_coords_puzzler_pt (
    short const *const pair_table,
```

```

float ** x,
float ** y,
double ** arc_coords,
vrna_plot_options_puzzler_t * puzzler )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Compute nucleotide coordinates for secondary structure plot using the RNApuzzler algorithm [30].
Same as vrna\_plot\_coords\_puzzler\(\) but takes a pair table with the structure information as input.

```

Note

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords_pt\(\)](#), [vrna_plot_coords_puzzler\(\)](#), [vrna_plot_coords_circular_pt\(\)](#), [vrna_plot_coords_simple_pt\(\)](#), [vrna_plot_coords_turtle_pt\(\)](#), [vrna_plot_coords_navview_pt\(\)](#)

Parameters

	<i>pt</i>	The pair table that holds the secondary structure
<code>in, out</code>	<code>x</code>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<code>in, out</code>	<code>y</code>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)
<code>in, out</code>	<code>arc_coords</code>	The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure)
	<code>options</code>	The options for the RNApuzzler algorithm (or NULL)

Returns

The length of the structure on success, 0 otherwise

16.77.5.15 vrna_plot_options_puzzler()

```

vrna_plot_options_puzzler_t * vrna_plot_options_puzzler (
    void )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Create an RNApuzzler options data structure.

```

See also

[vrna_plot_options_puzzler_free\(\)](#), [vrna_plot_coords_puzzler\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#), [vrna_plot_layout_puzzler\(\)](#)

Returns

An RNApuzzler options data structure with default settings

16.77.5.16 vrna_plot_options_puzzler_free()

```

void vrna_plot_options_puzzler_free (
    vrna_plot_options_puzzler_t * options )
#include <ViennaRNA/plotting/RNApuzzler/RNApuzzler.h>
Free memory occupied by an RNApuzzler options data structure.

```

See also

[vrna_plot_options_puzzler\(\)](#), [vrna_plot_coords_puzzler\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#), [vrna_plot_layout_puzzler\(\)](#)

Parameters

<i>options</i>	A pointer to the options data structure to free
----------------	---

16.77.5.17 vrna_plot_coords_turtle()

```
int vrna_plot_coords_turtle (
    const char * structure,
    float ** x,
    float ** y,
    double ** arc_coords )
#include <ViennaRNA/plotting/RNApuzzler/RNAturtle.h>
```

Compute nucleotide coordinates for secondary structure plot using the *RNAturtle* algorithm [30].

This function basically is a wrapper to [vrna_plot_coords\(\)](#) that passes the `plot_type` [VRNA_PLOT_TYPE_TURTLE](#).

Here is a simple example how to use this function, assuming variable `structure` contains a valid dot-bracket string:

```
float *x, *y;
double *arcs;

if (vrna_plot_coords_turtle(structure, &x, &y, &arcs)) {
    printf("all fine");
} else {
    printf("some failure occurred!");
}

free(x);
free(y);
free(arcs);
```

Note

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords\(\)](#), [vrna_plot_coords_turtle_pt\(\)](#), [vrna_plot_coords_circular\(\)](#), [vrna_plot_coords_simple\(\)](#), [vrna_plot_coords_navigate\(\)](#), [vrna_plot_coords_puzzler\(\)](#)

Parameters

	<i>structure</i>	The secondary structure in dot-bracket notation
<i>in, out</i>	<i>x</i>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
<i>in, out</i>	<i>y</i>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)
<i>in, out</i>	<i>arc_coords</i>	The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure)

Returns

The length of the structure on success, 0 otherwise

16.77.5.18 vrna_plot_coords_turtle_pt()

```
int vrna_plot_coords_turtle_pt (
    short const *const pair_table,
    float ** x,
    float ** y,
    double ** arc_coords )
#include <ViennaRNA/plotting/RNApuzzler/RNAturtle.h>
```

Compute nucleotide coordinates for secondary structure plot using the *RNAturtle* algorithm [30]. Same as [vrna_plot_coords_turtle\(\)](#) but takes a pair table with the structure information as input.

Note

On success, this function allocates memory for X, Y and arc coordinates and assigns the pointers at addresses `x`, `y` and `arc_coords` to the corresponding memory locations. It's the users responsibility to cleanup this memory after usage!

See also

[vrna_plot_coords_pt\(\)](#), [vrna_plot_coords_turtle\(\)](#), [vrna_plot_coords_circular_pt\(\)](#), [vrna_plot_coords_simple_pt\(\)](#), [vrna_plot_coords_puzzler_pt\(\)](#), [vrna_plot_coords_navigate_pt\(\)](#)

Parameters

	<i>pt</i>	The pair table that holds the secondary structure
in, out	<i>x</i>	The address of a pointer of X coordinates (pointer will point to memory, or NULL on failure)
in, out	<i>y</i>	The address of a pointer of Y coordinates (pointer will point to memory, or NULL on failure)
in, out	<i>arc_coords</i>	The address of a pointer that will hold arc coordinates (pointer will point to memory, or NULL on failure)

Returns

The length of the structure on success, 0 otherwise

16.78 Annotation

Functions to generate annotations for Secondary Structure Plots, Dot-Plots, and Others.

16.78.1 Detailed Description

Functions to generate annotations for Secondary Structure Plots, Dot-Plots, and Others.

Collaboration diagram for Annotation:

Functions

- `char ** vrna_annotate_covar_db` (const char **alignment, const char *structure, [vrna_md_t](#) *md_p)
Produce covariance annotation for an alignment given a secondary structure.
- `vrna_cpair_t * vrna_annotate_covar_pairs` (const char **alignment, [vrna_ep_t](#) *pl, [vrna_ep_t](#) *mfel, double threshold, [vrna_md_t](#) *md)
Produce covariance annotation for an alignment given a set of base pairs.

16.78.2 Function Documentation

16.78.2.1 vrna_annotate_covar_db()

```
char ** vrna_annotate_covar_db (
    const char ** alignment,
    const char * structure,
    vrna_md_t * md_p )
#include <ViennaRNA/plotting/utils.h>
```

Produce covariance annotation for an alignment given a secondary structure.

16.78.2.2 vrna_annotate_covar_pairs()

```
vrna_cpair_t * vrna_annotate_covar_pairs (
    const char ** alignment,
    vrna_ep_t * pl,
    vrna_ep_t * mfe1,
    double threshold,
    vrna_md_t * md )
#include <ViennaRNA/plotting/utils.h>
```

Produce covariance annotation for an alignment given a set of base pairs.

16.79 Alignment Plots

Functions to generate Alignment plots with annotated consensus structure.

16.79.1 Detailed Description

Functions to generate Alignment plots with annotated consensus structure.

Collaboration diagram for Alignment Plots:

Functions

- int [vrna_file_PS_aln](#) (const char *filename, const char **seqs, const char **names, const char *structure, unsigned int columns)
Create an annotated PostScript alignment plot.
- int [vrna_file_PS_aln_slice](#) (const char *filename, const char **seqs, const char **names, const char *structure, unsigned int start, unsigned int end, int offset, unsigned int columns)
Create an annotated PostScript alignment plot.

16.79.2 Function Documentation

16.79.2.1 vrna_file_PS_aln()

```
int vrna_file_PS_aln (
    const char * filename,
    const char ** seqs,
    const char ** names,
    const char * structure,
    unsigned int columns )
#include <ViennaRNA/plotting/alignments.h>
```

Create an annotated PostScript alignment plot.

See also

[vrna_file_PS_aln_slice\(\)](#)

Parameters

<i>filename</i>	The output file name
<i>seqs</i>	The aligned sequences
<i>names</i>	The names of the sequences
<i>structure</i>	The consensus structure in dot-bracket notation
<i>columns</i>	The number of columns before the alignment is wrapped as a new block (a value of 0 indicates no wrapping)

SWIG Wrapper Notes This function is available as overloaded function `file_PS_aln()` with three additional parameters `start`, `end`, and `offset` before the `columns` argument. Thus, it resembles the `vrna_file_PS_aln_slice()` function. The last four arguments may be omitted, indicating the default of `start = 0`, `end = 0`, `offset = 0`, and `columns = 60`.

16.79.2.2 `vrna_file_PS_aln_slice()`

```
int vrna_file_PS_aln_slice (
    const char * filename,
    const char ** seqs,
    const char ** names,
    const char * structure,
    unsigned int start,
    unsigned int end,
    int offset,
    unsigned int columns )
#include <ViennaRNA/plotting/alignments.h>
```

Create an annotated PostScript alignment plot.

Similar to `vrna_file_PS_aln()` but allows the user to print a particular slice of the alignment by specifying a `start` and `end` position. The additional `offset` parameter allows for adjusting the alignment position ruler value.

See also

[vrna_file_PS_aln_slice\(\)](#)

Parameters

<i>filename</i>	The output file name
<i>seqs</i>	The aligned sequences
<i>names</i>	The names of the sequences
<i>structure</i>	The consensus structure in dot-bracket notation
<i>start</i>	The start of the alignment slice (a value of 0 indicates the first position of the alignment, i.e. no slicing at 5' side)
<i>end</i>	The end of the alignment slice (a value of 0 indicates the last position of the alignment, i.e. no slicing at 3' side)
<i>offset</i>	The alignment coordinate offset for the position ruler.
<i>columns</i>	The number of columns before the alignment is wrapped as a new block (a value of 0 indicates no wrapping)

SWIG Wrapper Notes This function is available as overloaded function `file_PS_aln()` where the last four parameter may be omitted, indicating `start = 0`, `end = 0`, `offset = 0`, and `columns = 60`.

16.80 Search Algorithms

Implementations of various search algorithms to detect strings of objects within other strings of objects.

16.80.1 Detailed Description

Implementations of various search algorithms to detect strings of objects within other strings of objects.

Collaboration diagram for Search Algorithms:

Files

- file [BoyerMoore.h](#)

Variants of the Boyer-Moore string search algorithm.

Functions

- const unsigned int * [vrna_search_BMH_num](#) (const unsigned int *needle, size_t needle_size, const unsigned int *haystack, size_t haystack_size, size_t start, size_t *badchars, unsigned char cyclic)

Search for a string of elements in a larger string of elements using the Boyer-Moore-Horspool algorithm.

- const char * [vrna_search_BMH](#) (const char *needle, size_t needle_size, const char *haystack, size_t haystack_size, size_t start, size_t *badchars, unsigned char cyclic)

Search for an ASCII pattern within a larger ASCII string using the Boyer-Moore-Horspool algorithm.

- size_t * [vrna_search_BM_BCT_num](#) (const unsigned int *pattern, size_t pattern_size, unsigned int num_max)

Retrieve a Boyer-Moore Bad Character Table for a pattern of elements represented by natural numbers.

- size_t * [vrna_search_BM_BCT](#) (const char *pattern)

Retrieve a Boyer-Moore Bad Character Table for a NULL-terminated pattern of ASCII characters.

16.80.2 Function Documentation

16.80.2.1 vrna_search_BMH_num()

```
const unsigned int * vrna_search_BMH_num (
    const unsigned int * needle,
    size_t needle_size,
    const unsigned int * haystack,
    size_t haystack_size,
    size_t start,
    size_t * badchars,
    unsigned char cyclic )
#include <ViennaRNA/search/BoyerMoore.h>
```

Search for a string of elements in a larger string of elements using the Boyer-Moore-Horspool algorithm.

To speed-up subsequent searches with this function, the Bad Character Table should be precomputed and passed as argument `badchars`.

See also

[vrna_search_BM_BCT_num\(\)](#), [vrna_search_BMH\(\)](#)

Parameters

<i>needle</i>	The pattern of object representations to search for
<i>needle_size</i>	The size (length) of the pattern provided in <code>needle</code>
<i>haystack</i>	The string of objects the search will be performed on
<i>haystack_size</i>	The size (length) of the <code>haystack</code> string

Parameters

<i>start</i>	The position within <code>haystack</code> where to start the search
<i>badchars</i>	A pre-computed Bad Character Table obtained from vrna_search_BM_BCT_num() (If NULL, a Bad Character Table will be generated automatically)
<i>cyclic</i>	Allow for cyclic matches if non-zero, stop search at end of haystack otherwise

Returns

A pointer to the first occurrence of `needle` within `haystack` after position `start`

16.80.2.2 `vrna_search_BMH()`

```
const char * vrna_search_BMH (
    const char * needle,
    size_t needle_size,
    const char * haystack,
    size_t haystack_size,
    size_t start,
    size_t * badchars,
    unsigned char cyclic )
#include <ViennaRNA/search/BoyerMoore.h>
```

Search for an ASCII pattern within a larger ASCII string using the Boyer-Moore-Horspool algorithm.

To speed-up subsequent searches with this function, the Bad Character Table should be precomputed and passed as argument `badchars`. Furthermore, both, the lengths of `needle` and the length of `haystack` should be pre-computed and must be passed along with each call.

See also

[vrna_search_BM_BCT\(\)](#), [vrna_search_BMH_num\(\)](#)

Parameters

<i>needle</i>	The NULL-terminated ASCII pattern to search for
<i>needle_size</i>	The size (length) of the pattern provided in <code>needle</code>
<i>haystack</i>	The NULL-terminated ASCII string of the search will be performed on
<i>haystack_size</i>	The size (length) of the <code>haystack</code> string
<i>start</i>	The position within <code>haystack</code> where to start the search
<i>badchars</i>	A pre-computed Bad Character Table obtained from vrna_search_BM_BCT() (If NULL, a Bad Character Table will be generated automatically)
<i>cyclic</i>	Allow for cyclic matches if non-zero, stop search at end of haystack otherwise

Returns

A pointer to the first occurrence of `needle` within `haystack` after position `start`

16.80.2.3 `vrna_search_BM_BCT_num()`

```
size_t * vrna_search_BM_BCT_num (
    const unsigned int * pattern,
    size_t pattern_size,
    unsigned int num_max )
#include <ViennaRNA/search/BoyerMoore.h>
```

Retrieve a Boyer-Moore Bad Character Table for a pattern of elements represented by natural numbers.

Note

We store the maximum number representation of an element `num_max` at position 0. So the actual bad character table `T` starts at `T[1]` for an element represented by number 0.

See also

[vrna_search_BMH_num\(\)](#), [vrna_search_BM_BCT\(\)](#)

Parameters

<i>pattern</i>	The pattern of element representations used in the subsequent search
<i>pattern_size</i>	The size (length) of the pattern provided in <code>pattern</code>
<i>num_max</i>	The maximum number representation of an element, i.e. the size of the alphabet

Returns

A Bad Character Table for use in our Boyer-Moore search algorithm implementation(s)

16.80.2.4 vrna_search_BM_BCT()

```
size_t * vrna_search_BM_BCT (
    const char * pattern )
#include <ViennaRNA/search/BoyerMoore.h>
```

Retrieve a Boyer-Moore Bad Character Table for a NULL-terminated pattern of ASCII characters.

Note

We store the maximum number representation of an element, i.e. 127 at position 0. So the actual bad character table `T` starts at `T[1]` for an element represented by ASCII code 0.

See also

[vrna_search_BMH\(\)](#), [vrna_search_BM_BCT_num\(\)](#)

Parameters

<i>pattern</i>	The NULL-terminated pattern of ASCII characters used in the subsequent search
----------------	---

Returns

A Bad Character Table for use in our Boyer-Moore search algorithm implementation(s)

16.81 Combinatorics Algorithms

Implementations to solve various combinatorial aspects for strings of objects.

16.81.1 Detailed Description

Implementations to solve various combinatorial aspects for strings of objects.

Collaboration diagram for Combinatorics Algorithms:

Files

- file [combinatorics.h](#)

Various implementations that deal with combinatorial aspects of objects.

Functions

- unsigned int ** [vrna_enumerate_necklaces](#) (const unsigned int *type_counts)
Enumerate all necklaces with fixed content.
- unsigned int [vrna_rotational_symmetry_num](#) (const unsigned int *string, size_t string_length)
Determine the order of rotational symmetry for a string of objects represented by natural numbers.
- unsigned int [vrna_rotational_symmetry_pos_num](#) (const unsigned int *string, size_t string_length, unsigned int **positions)
Determine the order of rotational symmetry for a string of objects represented by natural numbers.
- unsigned int [vrna_rotational_symmetry](#) (const char *string)
Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.
- unsigned int [vrna_rotational_symmetry_pos](#) (const char *string, unsigned int **positions)
Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.
- unsigned int [vrna_rotational_symmetry_db](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Determine the order of rotational symmetry for a dot-bracket structure.
- unsigned int [vrna_rotational_symmetry_db_pos](#) ([vrna_fold_compound_t](#) *fc, const char *structure, unsigned int **positions)
Determine the order of rotational symmetry for a dot-bracket structure.
- unsigned int ** [vrna_n_multichoose_k](#) (size_t n, size_t k)
Obtain a list of k-combinations with repetition (n multichoose k)
- unsigned int * [vrna_boustrophedon](#) (size_t start, size_t end)
Generate a sequence of Boustrophedon distributed numbers.
- unsigned int [vrna_boustrophedon_pos](#) (size_t start, size_t end, size_t pos)
Obtain the i-th element in a Boustrophedon distributed interval of natural numbers.

16.81.2 Function Documentation

16.81.2.1 [vrna_enumerate_necklaces\(\)](#)

```
unsigned int ** vrna_enumerate_necklaces (
    const unsigned int * type_counts )
#include <ViennaRNA/combinatorics.h>
```

Enumerate all necklaces with fixed content.

This function implements *A fast algorithm to generate necklaces with fixed content* as published by Joe Sawada in 2003 [26].

The function receives a list of counts (the elements on the necklace) for each type of object within a necklace. The list starts at index 0 and ends with an entry that has a count of 0. The algorithm then enumerates all non-cyclic permutations of the content, returned as a list of necklaces. This list, again, is zero-terminated, i.e. the last entry of the list is a NULL pointer.

Parameters

<code>type_counts</code>	A 0-terminated list of entity counts
--------------------------	--------------------------------------

Returns

A list of all non-cyclic permutations of the entities

SWIG Wrapper Notes This function is available as global function `enumerate_necklaces()` which accepts lists input, and produces list of lists output.

16.81.2.2 `vrna_rotational_symmetry_num()`

```
unsigned int vrna_rotational_symmetry_num (
    const unsigned int * string,
    size_t string_length )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a string of objects represented by natural numbers.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with its start. For example, a string of the form 011011 has rotational symmetry of order 2

This is a simplified version of [vrna_rotational_symmetry_pos_num\(\)](#) that may be useful if one is only interested in the degree of rotational symmetry but not the actual set of rotational symmetric strings.

See also

[vrna_rotational_symmetry_pos_num\(\)](#), [vrna_rotational_symmetry\(\)](#)

Parameters

<i>string</i>	The string of elements encoded as natural numbers
<i>string_length</i>	The length of the string

Returns

The order of rotational symmetry

SWIG Wrapper Notes This function is available as global function `rotational_symmetry()`. See [vrna_rotational_symmetry_pos\(\)](#) for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

16.81.2.3 `vrna_rotational_symmetry_pos_num()`

```
unsigned int vrna_rotational_symmetry_pos_num (
    const unsigned int * string,
    size_t string_length,
    unsigned int ** positions )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a string of objects represented by natural numbers.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with its start. For example, a string of the form 011011 has rotational symmetry of order 2

If the argument `positions` is not NULL, the function stores an array of string start positions for rotational shifts that map the string back onto itself. This array has length of order of rotational symmetry, i.e. the number returned by this function. The first element `positions[0]` always contains a shift value of 0 representing the trivial rotation.

Note

Do not forget to release the memory occupied by `positions` after a successful execution of this function.

See also

[vrna_rotational_symmetry_num\(\)](#), [vrna_rotational_symmetry\(\)](#), [vrna_rotational_symmetry_pos\(\)](#)

Parameters

<i>string</i>	The string of elements encoded as natural numbers
<i>string_length</i>	The length of the string
<i>positions</i>	A pointer to an (undefined) list of alternative string start positions that lead to an identity mapping (may be NULL)

Returns

The order of rotational symmetry

SWIG Wrapper Notes This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details. Note, that in the target language the length of the list `string` is always known a-priori, so the parameter `string_length` must be omitted.

16.81.2.4 `vrna_rotational_symmetry()`

```
unsigned int vrna_rotational_symmetry (
    const char * string )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with it's start. For example, a string of the form `AABAAB` has rotational symmetry of order 2

This is a simplified version of `vrna_rotational_symmetry_pos()` that may be useful if one is only interested in the degree of rotational symmetry but not the actual set of rotational symmetric strings.

See also

[vrna_rotational_symmetry_pos\(\)](#), [vrna_rotational_symmetry_num\(\)](#)

Parameters

<i>string</i>	A NULL-terminated string of characters
---------------	--

Returns

The order of rotational symmetry

SWIG Wrapper Notes This function is available as global function `rotational_symmetry()`. See `vrna_rotational_symmetry_pos()` for details.

16.81.2.5 `vrna_rotational_symmetry_pos()`

```
unsigned int vrna_rotational_symmetry_pos (
    const char * string,
    unsigned int ** positions )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.

The algorithm applies a fast search of the provided string within itself, assuming the end of the string wraps around to connect with it's start. For example, a string of the form `AABAAB` has rotational symmetry of order 2

If the argument `positions` is not NULL, the function stores an array of string start positions for rotational shifts that map the string back onto itself. This array has length of order of rotational symmetry, i.e. the number returned by this function. The first element `positions[0]` always contains a shift value of 0 representing the trivial rotation.

Note

Do not forget to release the memory occupied by `positions` after a successful execution of this function.

See also

[vrna_rotational_symmetry\(\)](#), [vrna_rotational_symmetry_num\(\)](#), [vrna_rotational_symmetry_num_pos\(\)](#)

Parameters

<i>string</i>	A NULL-terminated string of characters
<i>positions</i>	A pointer to an (undefined) list of alternative string start positions that lead to an identity mapping (may be NULL)

Returns

The order of rotational symmetry

SWIG Wrapper Notes This function is available as overloaded global function **rotational_symmetry()**. It merges the functionalities of [vrna_rotational_symmetry\(\)](#), [vrna_rotational_symmetry_pos\(\)](#), [vrna_rotational_symmetry_num\(\)](#), and [vrna_rotational_symmetry_pos_num\(\)](#). In contrast to our C-implementation, this function doesn't return the order of rotational symmetry as a single value, but returns a list of cyclic permutation shifts that result in a rotationally symmetric string. The length of the list then determines the order of rotational symmetry.

16.81.2.6 vrna_rotational_symmetry_db()

```
unsigned int vrna_rotational_symmetry_db (
    vrna_fold_compound_t * fc,
    const char * structure )
#include <ViennaRNA/combinatorics.h>
```

Determine the order of rotational symmetry for a dot-bracket structure.

Given a (permutation of multiple) RNA strand(s) and a particular secondary structure in dot-bracket notation, compute the degree of rotational symmetry. In case there is only a single linear RNA strand, the structure always has degree 1, as there are no rotational symmetries due to the direction of the nucleic acid sequence and the fixed positions of 5' and 3' ends. However, for circular RNAs, rotational symmetries might arise if the sequence consists of a concatenation of k identical subsequences.

This is a simplified version of [vrna_rotational_symmetry_db_pos\(\)](#) that may be useful if one is only interested in the degree of rotational symmetry but not the actual set of rotational symmetric strings.

See also

[vrna_rotational_symmetry_db_pos\(\)](#), [vrna_rotational_symmetry\(\)](#), [vrna_rotational_symmetry_num\(\)](#)

Parameters

<i>fc</i>	A fold_compound data structure containing the nucleic acid sequence(s), their order, and model settings
<i>structure</i>	The dot-bracket structure the degree of rotational symmetry is checked for

Returns

The degree of rotational symmetry of the `structure` (0 in case of any errors)

SWIG Wrapper Notes This function is attached as method **rotational_symmetry_db()** to objects of type `fold_compound` (i.e. [vrna_fold_compound_t](#)). See [vrna_rotational_symmetry_db_pos\(\)](#) for details.

16.81.2.7 vrna_rotational_symmetry_db_pos()

```
unsigned int vrna_rotational_symmetry_db_pos (
    vrna_fold_compound_t * fc,
```



```

    const char * structure,
    unsigned int ** positions )
#include <ViennaRNA/combinatorics.h>

```

Determine the order of rotational symmetry for a dot-bracket structure.

Given a (permutation of multiple) RNA strand(s) and a particular secondary structure in dot-bracket notation, compute the degree of rotational symmetry. In case there is only a single linear RNA strand, the structure always has degree 1, as there are no rotational symmetries due to the direction of the nucleic acid sequence and the fixed positions of 5' and 3' ends. However, for circular RNAs, rotational symmetries might arise if the sequence consists of a concatenation of k identical subsequences.

If the argument `positions` is not NULL, the function stores an array of string start positions for rotational shifts that map the string back onto itself. This array has length of order of rotational symmetry, i.e. the number returned by this function. The first element `positions[0]` always contains a shift value of 0 representing the trivial rotation.

Note

Do not forget to release the memory occupied by `positions` after a successful execution of this function.

See also

[vrna_rotational_symmetry_db\(\)](#), [vrna_rotational_symmetry_pos\(\)](#), [vrna_rotational_symmetry_pos_num\(\)](#)

Parameters

<i>fc</i>	A fold_compound data structure containing the nucleic acid sequence(s), their order, and model settings
<i>structure</i>	The dot-bracket structure the degree of rotational symmetry is checked for
<i>positions</i>	A pointer to an (undefined) list of alternative string start positions that lead to an identity mapping (may be NULL)

Returns

The degree of rotational symmetry of the `structure` (0 in case of any errors)

SWIG Wrapper Notes This function is attached as method `rotational_symmetry_db()` to objects of type `fold_compound` (i.e. `vrna_fold_compound_t`). Thus, the first argument must be omitted. In contrast to our C-implementation, this function doesn't simply return the order of rotational symmetry of the secondary structure, but returns the list `position` of cyclic permutation shifts that result in a rotationally symmetric structure. The length of the list then determines the order of rotational symmetry.

16.81.2.8 vrna_n_multichoose_k()

```

unsigned int ** vrna_n_multichoose_k (
    size_t n,
    size_t k )
#include <ViennaRNA/combinatorics.h>

```

Obtain a list of k -combinations with repetition (n multichoose k)

This function compiles a list of k -combinations, or k -multicombination, i.e. a list of multisubsets of size k from a set of integer values from 0 to $n - 1$. For that purpose, we enumerate $n + k - 1$ choose k and decrease each index position i by i to obtain n multichoose k .

Parameters

<i>n</i>	Maximum number to choose from (interval of integers from 0 to $n - 1$)
<i>k</i>	Number of elements to choose, i.e. size of each multisubset

Returns

A list of lists of elements of combinations (last entry is terminated by **NULL**)

16.81.2.9 vrna_boustrophedon()

```
unsigned int vrna_boustrophedon (
    size_t start,
    size_t end )
#include <ViennaRNA/combinatorics.h>
```

Generate a sequence of Boustrophedon distributed numbers.

This function generates a sequence of positive natural numbers within the interval $[start, end]$ in a Boustrophedon fashion. That is, the numbers $start, \dots, end$ in the resulting list are alternating between left and right ends of the interval while progressing to the inside, i.e. the list consists of a sequence of natural numbers of the form:

$$start, end, start + 1, end - 1, start + 2, end - 2, \dots$$

The resulting list is 1-based and contains the length of the sequence of numbers at it's 0-th position.

Upon failure, the function returns **NULL**

See also

[vrna_boustrophedon_pos\(\)](#)

Parameters

<i>start</i>	The first number of the list (left side of the interval)
<i>end</i>	The last number of the list (right side of the interval)

Returns

A list of alternating numbers from the interval $[start, end]$ (or **NULL** on error)

SWIG Wrapper Notes This function is available as overloaded global function **boustrophedon()**.

16.81.2.10 vrna_boustrophedon_pos()

```
unsigned int vrna_boustrophedon_pos (
    size_t start,
    size_t end,
    size_t pos )
#include <ViennaRNA/combinatorics.h>
```

Obtain the i-th element in a Boustrophedon distributed interval of natural numbers.

See also

[vrna_boustrophedon\(\)](#)

Parameters

<i>start</i>	The first number of the list (left side of the interval)
<i>end</i>	The last number of the list (right side of the interval)
<i>pos</i>	The index of the number within the Boustrophedon distributed sequence (1-based)

Returns

The `pos-th` element in the Boustrophedon distributed sequence of natural numbers of the interval

SWIG Wrapper Notes This function is available as overloaded global function **boustrophedon()**. Omitting the `pos` argument yields the entire sequence from `start` to `end`.

16.82 (Abstract) Data Structures

All datastructures and typedefs shared among the ViennaRNA Package can be found here.

16.82.1 Detailed Description

All datastructures and typedefs shared among the ViennaRNA Package can be found here.

Collaboration diagram for (Abstract) Data Structures:

Modules

- [The Fold Compound](#)
This module provides interfaces that deal with the most basic data structure used in structure predicting and energy evaluating function of the RNAlib.
- [The Dynamic Programming Matrices](#)
This module provides interfaces that deal with creation and destruction of dynamic programming matrices used within the RNAlib.
- [Hash Tables](#)
Various implementations of hash table functions.
- [Heaps](#)
Interface for an abstract implementation of a heap data structure.
- [Arrays](#)
Interface for an abstract implementation of an array data structure.
- [Buffers](#)
Functions that provide dynamically buffered stream-like data structures.

Files

- file [dp_matrices.h](#)
Functions to deal with standard dynamic programming (DP) matrices.
- file [array.h](#)
A macro-based dynamic array implementation.
- file [basic.h](#)
Various data structures and pre-processor macros.

Data Structures

- struct [vrna_basepair_s](#)
Base pair data structure used in subopt.c. [More...](#)
- struct [vrna_cpair_s](#)
this datastructure is used as input parameter in functions of PS_dot.c [More...](#)
- struct [vrna_color_s](#)
- struct [vrna_data_linear_s](#)
- struct [vrna_sect_s](#)
Stack of partial structures for backtracking. [More...](#)
- struct [vrna_bp_stack_s](#)
Base pair stack element. [More...](#)
- struct [pu_contrib](#)

- contributions to p_u [More...](#)*
- struct [interact](#)
 - interaction data structure for RNAup [More...](#)*
- struct [pu_out](#)
 - Collection of all free_energy of beeing unpaired values for output. [More...](#)*
- struct [constrain](#)
 - constraints for cofolding [More...](#)*
- struct [duplexT](#)
 - Data structure for RNAduplex. [More...](#)*
- struct [node](#)
 - Data structure for RNAsnoop (fold energy list) [More...](#)*
- struct [snoopT](#)
 - Data structure for RNAsnoop. [More...](#)*
- struct [dupVar](#)
 - Data structure used in RNAppkplex. [More...](#)*

Typedefs

- typedef struct [vrna_basepair_s](#) [vrna_basepair_t](#)
 - Typename for the base pair representing data structure [vrna_basepair_s](#).*
- typedef struct [vrna_elem_prob_s](#) [vrna_plist_t](#)
 - Typename for the base pair list representing data structure [vrna_elem_prob_s](#).*
- typedef struct [vrna_bp_stack_s](#) [vrna_bp_stack_t](#)
 - Typename for the base pair stack representing data structure [vrna_bp_stack_s](#).*
- typedef struct [vrna_cpair_s](#) [vrna_cpair_t](#)
 - Typename for data structure [vrna_cpair_s](#).*
- typedef struct [vrna_sect_s](#) [vrna_sect_t](#)
 - Typename for stack of partial structures [vrna_sect_s](#).*
- typedef double [FLT_OR_DBL](#)
 - Typename for floating point number in partition function computations.*
- typedef struct [vrna_basepair_s](#) [PAIR](#)
 - Old typename of [vrna_basepair_s](#).*
- typedef struct [vrna_elem_prob_s](#) [plist](#)
 - Old typename of [vrna_elem_prob_s](#).*
- typedef struct [vrna_cpair_s](#) [cpair](#)
 - Old typename of [vrna_cpair_s](#).*
- typedef struct [vrna_sect_s](#) [sect](#)
 - Old typename of [vrna_sect_s](#).*
- typedef struct [vrna_bp_stack_s](#) [bondT](#)
 - Old typename of [vrna_bp_stack_s](#).*
- typedef struct [pu_contrib](#) [pu_contrib](#)
 - contributions to p_u*
- typedef struct [interact](#) [interact](#)
 - interaction data structure for RNAup*
- typedef struct [pu_out](#) [pu_out](#)
 - Collection of all free_energy of beeing unpaired values for output.*
- typedef struct [constrain](#) [constrain](#)
 - constraints for cofolding*
- typedef struct [node](#) [folden](#)
 - Data structure for RNAsnoop (fold energy list)*
- typedef struct [dupVar](#) [dupVar](#)
 - Data structure used in RNAppkplex.*

Functions

- void `vrna_C11_features` (void)

Dummy symbol to check whether the library was build using C11/C++11 features.

16.82.2 Data Structure Documentation

16.82.2.1 struct `vrna_basepair_s`

Base pair data structure used in subopt.c.

16.82.2.2 struct `vrna_cpair_s`

this datastructure is used as input parameter in functions of PS_dot.c

16.82.2.3 struct `vrna_color_s`

16.82.2.4 struct `vrna_data_linear_s`

Collaboration diagram for `vrna_data_linear_s`:

16.82.2.5 struct `vrna_sect_s`

Stack of partial structures for backtracking.

16.82.2.6 struct `vrna_bp_stack_s`

Base pair stack element.

16.82.2.7 struct `pu_contrib`

contributions to `p_u`

Data Fields

- double ** **H**
hairpin loops
- double ** **I**
interior loops
- double ** **M**
multi loops
- double ** **E**
exterior loop
- int **length**
length of the input sequence
- int **w**
longest unpaired region

16.82.2.8 struct `interact`

interaction data structure for RNAup

Data Fields

- double * **Pi**
probabilities of interaction
- double * **Gi**
free energies of interaction
- double **Gikjl**
full free energy for interaction between [k,i] $k < i$ in longer seq and [j,l] $j < l$ in shorter seq
- double **Gikjl_wo**
Gikjl without contributions for prob_unpaired.
- int **i**
 $k < i$ in longer seq
- int **k**
 $k < i$ in longer seq
- int **j**
 $j < l$ in shorter seq
- int **l**
 $j < l$ in shorter seq
- int **length**
length of longer sequence

16.82.2.9 struct pu_out

Collection of all free_energy of beeing unpaired values for output.

Data Fields

- int **len**
sequence length
- int **u_vals**
number of different -u values
- int **contribs**
[-c "SHIME"]
- char ** **header**
header line
- double ** **u_values**
*(the -u values * [-c "SHIME"]) * seq len*

16.82.2.10 struct constrain

constraints for cofolding

16.82.2.11 struct duplexT

Data structure for RNAduplex.

16.82.2.12 struct node

Data structure for RNAsnoop (fold energy list)
Collaboration diagram for node:

16.82.2.13 struct snoopT

Data structure for RNAsnoop.

16.82.2.14 struct dupVar

Data structure used in RNAppkplex.

16.82.3 Typedef Documentation

16.82.3.1 PAIR

```
typedef struct vrna_basepair_s PAIR
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_basepair_s.
```

Deprecated Use `vrna_basepair_t` instead!

16.82.3.2 plist

```
typedef struct vrna_elem_prob_s plist
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_elem_prob_s.
```

Deprecated Use `vrna_ep_t` or `vrna_elem_prob_s` instead!

16.82.3.3 cpair

```
typedef struct vrna_cpair_s cpair
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_cpair_s.
```

Deprecated Use `vrna_cpair_t` instead!

16.82.3.4 sect

```
typedef struct vrna_sect_s sect
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_sect_s.
```

Deprecated Use `vrna_sect_t` instead!

16.82.3.5 bondT

```
typedef struct vrna_bp_stack_s bondT
#include <ViennaRNA/datastructures/basic.h>
Old typename of vrna_bp_stack_s.
```

Deprecated Use `vrna_bp_stack_t` instead!

16.82.4 Function Documentation

16.82.4.1 vrna_C11_features()

```
void vrna_C11_features (
    void )
#include <ViennaRNA/datastructures/basic.h>
```

Dummy symbol to check whether the library was build using C11/C++11 features.

By default, several data structures of our new v3.0 API use C11/C++11 features, such as unnamed unions, unnamed structs. However, these features can be deactivated at compile time to allow building the library and executables with compilers that do not support these features.

Now, the problem arises that once our static library is compiled and a third-party application is supposed to link against it, it needs to know, at compile time, how to correctly address particular data structures. This is usually implicitly taken care of through the API exposed in our header files. Unfortunately, we had some preprocessor directives in our header files that changed the API depending on the capabilities of the compiler the third-party application is build with. This in turn prohibited the use of an RNAlib compiled without C11/C++11 support in a program that compiles/links with enabled C11/C++11 support and vice-versa.

Therefore, we introduce this dummy symbol which can be used to check, whether the static library was build with C11/C++11 features.

Note

If the symbol is present, the library was build with enabled C11/C++11 features support and no action is required. However, if the symbol is missing in RNAlib $\geq 2.2.9$, programs that link to RNAlib must define a pre-processor identifier `VRNA_DISABLE_C11_FEATURES` before including any ViennaRNA Package header file, for instance by adding a `CPPFLAG`

```
CPPFLAGS+=-DVRNA_DISABLE_C11_FEATURES
```

Since

v2.2.9

16.83 Messages

Functions to print various kind of messages.

16.83.1 Detailed Description

Functions to print various kind of messages.

Collaboration diagram for Messages:

Functions

- void [vrna_message_error](#) (const char *format,...)
Print an error message and die.
- void [vrna_message_verror](#) (const char *format, va_list args)
Print an error message and die.
- void [vrna_message_warning](#) (const char *format,...)
Print a warning message.
- void [vrna_message_vwarning](#) (const char *format, va_list args)
Print a warning message.
- void [vrna_message_info](#) (FILE *fp, const char *format,...)
Print an info message.
- void [vrna_message_vinfo](#) (FILE *fp, const char *format, va_list args)
Print an info message.
- void [vrna_message_input_seq_simple](#) (void)
Print a line to stdout that asks for an input sequence.
- void [vrna_message_input_seq](#) (const char *s)
Print a line with a user defined string and a ruler to stdout.

16.83.2 Function Documentation

16.83.2.1 `vrna_message_error()`

```
void vrna_message_error (
    const char * format,
    ... )
#include <ViennaRNA/utils/basic.h>
```

Print an error message and die.

This function is a wrapper to `fprintf(stderr, ...)` that puts a capital **ERROR:** in front of the message and then exits the calling program.

See also

[vrna_message_verror\(\)](#), [vrna_message_warning\(\)](#), [vrna_message_info\(\)](#)

Parameters

<i>format</i>	The error message to be printed
...	Optional arguments for the formatted message string

16.83.2.2 `vrna_message_verror()`

```
void vrna_message_verror (
    const char * format,
    va_list args )
#include <ViennaRNA/utils/basic.h>
```

Print an error message and die.

This function is a wrapper to `vfprintf(stderr, ...)` that puts a capital **ERROR:** in front of the message and then exits the calling program.

See also

[vrna_message_error\(\)](#), [vrna_message_warning\(\)](#), [vrna_message_info\(\)](#)

Parameters

<i>format</i>	The error message to be printed
<i>args</i>	The argument list for the formatted message string

16.83.2.3 `vrna_message_warning()`

```
void vrna_message_warning (
    const char * format,
    ... )
#include <ViennaRNA/utils/basic.h>
```

Print a warning message.

This function is a wrapper to `fprintf(stderr, ...)` that puts a capital **WARNING:** in front of the message.

See also

[vrna_message_vwarning\(\)](#), [vrna_message_error\(\)](#), [vrna_message_info\(\)](#)

Parameters

<i>format</i>	The warning message to be printed
...	Optional arguments for the formatted message string

16.83.2.4 vrna_message_vwarning()

```
void vrna_message_vwarning (
    const char * format,
    va_list args )
#include <ViennaRNA/utils/basic.h>
```

Print a warning message.

This function is a wrapper to *fprintf(stderr, ...)* that puts a capital **WARNING:** in front of the message.

See also

[vrna_message_vwarning\(\)](#), [vrna_message_error\(\)](#), [vrna_message_info\(\)](#)

Parameters

<i>format</i>	The warning message to be printed
<i>args</i>	The argument list for the formatted message string

16.83.2.5 vrna_message_info()

```
void vrna_message_info (
    FILE * fp,
    const char * format,
    ... )
#include <ViennaRNA/utils/basic.h>
```

Print an info message.

This function is a wrapper to *fprintf(...)*.

See also

[vrna_message_vinfo\(\)](#), [vrna_message_error\(\)](#), [vrna_message_warning\(\)](#)

Parameters

<i>fp</i>	The file pointer where the message is printed to
<i>format</i>	The warning message to be printed
...	Optional arguments for the formatted message string

16.83.2.6 vrna_message_vinfo()

```
void vrna_message_vinfo (
    FILE * fp,
    const char * format,
    va_list args )
#include <ViennaRNA/utils/basic.h>
```

Print an info message.

This function is a wrapper to `fprintf(...)`.

See also

[vrna_message_vinfo\(\)](#), [vrna_message_error\(\)](#), [vrna_message_warning\(\)](#)

Parameters

<i>fp</i>	The file pointer where the message is printed to
<i>format</i>	The info message to be printed
<i>args</i>	The argument list for the formatted message string

16.83.2.7 vrna_message_input_seq_simple()

```
void vrna_message_input_seq_simple (
    void )
#include <ViennaRNA/utils/basic.h>
Print a line to stdout that asks for an input sequence.
There will also be a ruler (scale line) printed that helps orientation of the sequence positions
```

16.83.2.8 vrna_message_input_seq()

```
void vrna_message_input_seq (
    const char * s )
#include <ViennaRNA/utils/basic.h>
Print a line with a user defined string and a ruler to stdout.
(usually this is used to ask for user input) There will also be a ruler (scale line) printed that helps orientation of the
sequence positions
```

Parameters

<i>s</i>	A user defined string that will be printed to stdout
----------	--

16.84 Unit Conversion

Functions to convert between various physical units.

16.84.1 Detailed Description

Functions to convert between various physical units.

Collaboration diagram for Unit Conversion:

Files

- file [units.h](#)
Physical Units and Functions to convert them into each other.

Enumerations

- enum [vrna_unit_energy_e](#) {
[VRNA_UNIT_J](#), [VRNA_UNIT_KJ](#), [VRNA_UNIT_CAL_IT](#), [VRNA_UNIT_DACAL_IT](#),
[VRNA_UNIT_KCAL_IT](#), [VRNA_UNIT_CAL](#), [VRNA_UNIT_DACAL](#), [VRNA_UNIT_KCAL](#),
[VRNA_UNIT_G_TNT](#), [VRNA_UNIT_KG_TNT](#), [VRNA_UNIT_T_TNT](#), [VRNA_UNIT_EV](#),
[VRNA_UNIT_WH](#), [VRNA_UNIT_KWH](#) }

Energy / Work Units.

- enum `vrna_unit_temperature_e` {
`VRNA_UNIT_K`, `VRNA_UNIT_DEG_C`, `VRNA_UNIT_DEG_F`, `VRNA_UNIT_DEG_R`,
`VRNA_UNIT_DEG_N`, `VRNA_UNIT_DEG_DE`, `VRNA_UNIT_DEG_RE`, `VRNA_UNIT_DEG_RO` }

Temperature Units.

Functions

- double `vrna_convert_energy` (double energy, `vrna_unit_energy_e` from, `vrna_unit_energy_e` to)
Convert between energy / work units.
- double `vrna_convert_temperature` (double temp, `vrna_unit_temperature_e` from, `vrna_unit_temperature_e` to)
Convert between temperature units.
- int `vrna_convert_kcal_to_dcal` (double energy)
Convert floating point energy value into integer representation.
- double `vrna_convert_dcal_to_kcal` (int energy)
Convert an integer representation of free energy in deka-cal/mol to kcal/mol.

16.84.2 Enumeration Type Documentation

16.84.2.1 `vrna_unit_energy_e`

```
enum vrna_unit_energy_e
#include <ViennaRNA/utils/units.h>
Energy / Work Units.
```

See also

[`vrna_convert_energy\(\)`](#)

Enumerator

<code>VRNA_UNIT_J</code>	Joule ($1\text{ J} = 1\text{ kg} \cdot \text{m}^2\text{s}^{-2}$)
<code>VRNA_UNIT_KJ</code>	Kilojoule ($1\text{ kJ} = 1,000\text{ J}$)
<code>VRNA_UNIT_CAL_IT</code>	Calorie (International (Steam) Table, $1\text{ cal}_{IT} = 4.1868\text{ J}$)
<code>VRNA_UNIT_DACAL_IT</code>	Decacalorie (International (Steam) Table, $1\text{ dcal}_{IT} = 10\text{ cal}_{IT} = 41.868\text{ J}$)
<code>VRNA_UNIT_KCAL_IT</code>	Kilocalorie (International (Steam) Table, $1\text{ kcal}_{IT} = 4.1868\text{ kJ}$)
<code>VRNA_UNIT_CAL</code>	Calorie (Thermochemical, $1\text{ cal}_{th} = 4.184\text{ J}$)
<code>VRNA_UNIT_DACAL</code>	Decacalorie (Thermochemical, $1\text{ dcal}_{th} = 10\text{ cal}_{th} = 41.84\text{ J}$)
<code>VRNA_UNIT_KCAL</code>	Kilocalorie (Thermochemical, $1\text{ kcal}_{th} = 4.184\text{ kJ}$)
<code>VRNA_UNIT_G_TNT</code>	g TNT ($1\text{ g TNT} = 1,000\text{ cal}_{th} = 4,184\text{ J}$)
<code>VRNA_UNIT_KG_TNT</code>	kg TNT ($1\text{ kg TNT} = 1,000\text{ kcal}_{th} = 4,184\text{ kJ}$)
<code>VRNA_UNIT_T_TNT</code>	ton TNT ($1\text{ t TNT} = 1,000,000\text{ kcal}_{th} = 4,184\text{ MJ}$)
<code>VRNA_UNIT_EV</code>	Electronvolt ($1\text{ eV} = 1.602176565 \times 10^{-19}\text{ J}$)
<code>VRNA_UNIT_WH</code>	Watt hour ($1\text{ W} \cdot \text{h} = 1\text{ W} \cdot 3,600\text{ s} = 3,600\text{ J} = 3.6\text{ kJ}$)
<code>VRNA_UNIT_KWH</code>	Kilowatt hour ($1\text{ kW} \cdot \text{h} = 1\text{ kW} \cdot 3,600\text{ s} = 3,600\text{ kJ} = 3.6\text{ MJ}$)

16.84.2.2 vrna_unit_temperature_e

```
enum vrna_unit_temperature_e
#include <ViennaRNA/utils/units.h>
Temperature Units.
```

See also

[vrna_convert_temperature\(\)](#)

Enumerator

VRNA_UNIT_K	Kelvin (K)
VRNA_UNIT_DEG_C	Degree Celcius ($^{\circ}\text{C}$) ($[^{\circ}\text{C}] = [K] - 273.15$)
VRNA_UNIT_DEG_F	Degree Fahrenheit ($^{\circ}\text{F}$) ($[^{\circ}\text{F}] = [K] \times \frac{9}{5} - 459.67$)
VRNA_UNIT_DEG_R	Degree Rankine ($^{\circ}\text{R}$) ($[^{\circ}\text{R}] = [K] \times \frac{9}{5}$)
VRNA_UNIT_DEG_N	Degree Newton ($^{\circ}\text{N}$) ($[^{\circ}\text{N}] = ([K] - 273.15) \times \frac{33}{100}$)
VRNA_UNIT_DEG_DE	Degree Delisle ($^{\circ}\text{De}$) ($[^{\circ}\text{De}] = (373.15 - [K]) \times \frac{3}{2}$)
VRNA_UNIT_DEG_RE	Degree Réaumur ($^{\circ}\text{Ré}$) ($[^{\circ}\text{Ré}] = ([K] - 273.15) \times \frac{4}{5}$)
VRNA_UNIT_DEG_RO	Degree Rømer ($^{\circ}\text{Rø}$) ($[^{\circ}\text{Rø}] = ([K] - 273.15) \times \frac{21}{40} + 7.5$)

16.84.3 Function Documentation**16.84.3.1 vrna_convert_energy()**

```
double vrna_convert_energy (
    double energy,
    vrna_unit_energy_e from,
    vrna_unit_energy_e to )
#include <ViennaRNA/utils/units.h>
Convert between energy / work units.
```

See also

[vrna_unit_energy_e](#)

Parameters

<i>energy</i>	Input energy value
<i>from</i>	Input unit
<i>to</i>	Output unit

Returns

Energy value in Output unit

16.84.3.2 vrna_convert_temperature()

```
double vrna_convert_temperature (
    double temp,
```

```
    vrna_unit_temperature_e from,  
    vrna_unit_temperature_e to )  
#include <ViennaRNA/utils/units.h>  
Convert between temperature units.
```

See also

[vrna_unit_temperature_e](#)

Parameters

<i>temp</i>	Input temperature value
<i>from</i>	Input unit
<i>to</i>	Output unit

Returns

Temperature value in Output unit

16.84.3.3 vrna_convert_kcal_to_dcal()

```
int vrna_convert_kcal_to_dcal (  
    double energy )  
#include <ViennaRNA/utils/units.h>  
Convert floating point energy value into integer representation.  
This function converts a floating point value in kcal/mol into its corresponding deka-cal/mol integer representation  
as used throughout RNAlib.
```

See also

[vrna_convert_dcal_to_kcal\(\)](#)

Parameters

<i>energy</i>	The energy value in kcal/mol
---------------	------------------------------

Returns

The energy value in deka-cal/mol

16.84.3.4 vrna_convert_dcal_to_kcal()

```
double vrna_convert_dcal_to_kcal (  
    int energy )  
#include <ViennaRNA/utils/units.h>  
Convert an integer representation of free energy in deka-cal/mol to kcal/mol.  
This function converts a free energy value given as integer in deka-cal/mol into the corresponding floating point  
number in kcal/mol
```

See also

[vrna_convert_kcal_to_dcal\(\)](#)

Parameters

<i>energy</i>	The energy in deka-cal/mol
---------------	----------------------------

Returns

The energy in kcal/mol

16.85 The Fold Compound

This module provides interfaces that deal with the most basic data structure used in structure predicting and energy evaluating function of the RNALib.

16.85.1 Detailed Description

This module provides interfaces that deal with the most basic data structure used in structure predicting and energy evaluating function of the RNALib.

Throughout the entire RNALib, the [vrna_fold_compound_t](#), is used to group information and data that is required for structure prediction and energy evaluation. Here, you'll find interface functions to create, modify, and delete [vrna_fold_compound_t](#) data structures. Collaboration diagram for The Fold Compound:

Files

- file [fold_compound.h](#)
The Basic Fold Compound API.

Data Structures

- struct [vrna_fc_s](#)
The most basic data structure required by many functions throughout the RNALib. [More...](#)

Macros

- #define [VRNA_STATUS_MFE_PRE](#) (unsigned char)1
Status message indicating that MFE computations are about to begin.
- #define [VRNA_STATUS_MFE_POST](#) (unsigned char)2
Status message indicating that MFE computations are finished.
- #define [VRNA_STATUS_PF_PRE](#) (unsigned char)3
Status message indicating that Partition function computations are about to begin.
- #define [VRNA_STATUS_PF_POST](#) (unsigned char)4
Status message indicating that Partition function computations are finished.
- #define [VRNA_OPTION_DEFAULT](#) 0U
Option flag to specify default settings/requirements.
- #define [VRNA_OPTION_MFE](#) 1U
Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
- #define [VRNA_OPTION_PF](#) 2U
Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
- #define [VRNA_OPTION_HYBRID](#) 4U
Option flag to specify requirement of dimer DP matrices.
- #define [VRNA_OPTION_EVAL_ONLY](#) 8U
Option flag to specify that neither MFE, nor PF DP matrices are required.
- #define [VRNA_OPTION_WINDOW](#) 16U
Option flag to specify requirement of DP matrices for local folding approaches.

Typedefs

- typedef struct [vrna_fc_s](#) **vrna_fold_compound_t**
Typename for the fold_compound data structure [vrna_fc_s](#).
- typedef void(* [vrna_auxdata_free_f](#)) (void *data)
Callback to free memory allocated for auxiliary user-provided data.
- typedef void(* [vrna_recursion_status_f](#)) (unsigned char status, void *data)
Callback to perform specific user-defined actions before, or after recursive computations.

Enumerations

- enum [vrna_fc_type_e](#) { [VRNA_FC_TYPE_SINGLE](#) , [VRNA_FC_TYPE_COMPARATIVE](#) }
An enumerator that is used to specify the type of a [vrna_fold_compound_t](#).

Functions

- [vrna_fold_compound_t](#) * [vrna_fold_compound](#) (const char *sequence, const [vrna_md_t](#) *md_p, unsigned int options)
Retrieve a [vrna_fold_compound_t](#) data structure for single sequences and hybridizing sequences.
- [vrna_fold_compound_t](#) * [vrna_fold_compound_comparative](#) (const char **sequences, [vrna_md_t](#) *md_p, unsigned int options)
Retrieve a [vrna_fold_compound_t](#) data structure for sequence alignments.
- void [vrna_fold_compound_free](#) ([vrna_fold_compound_t](#) *fc)
Free memory occupied by a [vrna_fold_compound_t](#).
- void [vrna_fold_compound_add_auxdata](#) ([vrna_fold_compound_t](#) *fc, void *data, [vrna_auxdata_free_f](#) f)
Add auxiliary data to the [vrna_fold_compound_t](#).
- void [vrna_fold_compound_add_callback](#) ([vrna_fold_compound_t](#) *fc, [vrna_recursion_status_f](#) f)
Add a recursion status callback to the [vrna_fold_compound_t](#).

16.85.2 Data Structure Documentation

16.85.2.1 struct vrna_fc_s

The most basic data structure required by many functions throughout the RNAlib.

Note

Please read the documentation of this data structure carefully! Some attributes are only available for specific types this data structure can adopt.

Warning

Reading/Writing from/to attributes that are not within the scope of the current type usually result in undefined behavior!

See also

[vrna_fold_compound_t.type](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_fold_compound_free\(\)](#), [VRNA_FC_TYPE_SINGLE](#), [VRNA_FC_TYPE_COMPARATIVE](#)

SWIG Wrapper Notes This data structure is wrapped as an object **fold_compound** with several related functions attached as methods.

A new **fold_compound** can be obtained by calling one of its constructors:

- *fold_compound(seq)* – Initialize with a single sequence, or two concatenated sequences separated by an ampersand character '&' (for cofolding)

- `fold_compound(aln)` – Initialize with a sequence alignment `aln` stored as a list of sequences (with gap characters)

The resulting object has a list of attached methods which in most cases directly correspond to functions that mainly operate on the corresponding `C` data structure:

- `type()` – Get the type of the `fold_compound` (See [vrna_fc_type_e](#))
- `length()` – Get the length of the [sequence\(s\)](#) or alignment stored within the `fold_compound`

Collaboration diagram for `vrna_fc_s`:

Data Fields

Common data fields

- const [vrna_fc_type_e](#) `type`
The type of the [vrna_fold_compound_t](#).
- unsigned int `length`
The length of the sequence (or sequence alignment)
- int `cutpoint`
The position of the (cofold) cutpoint within the provided sequence. If there is no cutpoint, this field will be set to -1.
- unsigned int * `strand_number`
The strand number a particular nucleotide is associated with.
- unsigned int * `strand_order`
The strand order, i.e. permutation of current concatenated sequence.
- unsigned int * `strand_order_uniq`
The strand order array where identical sequences have the same ID.
- unsigned int * `strand_start`
The start position of a particular strand within the current concatenated sequence.
- unsigned int * `strand_end`
The end (last) position of a particular strand within the current concatenated sequence.
- unsigned int `strands`
Number of interacting strands.
- [vrna_seq_t](#) * `nucleotides`
Set of nucleotide sequences.
- [vrna_msa_t](#) * `alignment`
Set of alignments.
- [vrna_hc_t](#) * `hc`
The hard constraints data structure used for structure prediction.
- [vrna_mx_mfe_t](#) * `matrices`
The MFE DP matrices.
- [vrna_mx_pf_t](#) * `exp_matrices`
The PF DP matrices
- [vrna_param_t](#) * `params`
The precomputed free energy contributions for each type of loop.
- [vrna_exp_param_t](#) * `exp_params`
The precomputed free energy contributions as Boltzmann factors
- int * `iindx`
DP matrix accessor
- int * `jindx`
DP matrix accessor

User-defined data fields

- `vrna_recursion_status_f stat_cb`
Recursion status callback (usually called just before, and after recursive computations in the library).
- `void * auxdata`
A pointer to auxiliary, user-defined data.
- `vrna_auxdata_free_f free_auxdata`
A callback to free auxiliary user data whenever the fold_compound itself is free'd.

Secondary Structure Decomposition (grammar) related data fields

- `vrna_sd_t * domains_struct`
Additional structured domains.
- `vrna_ud_t * domains_up`
Additional unstructured domains.
- `vrna_gr_aux_t * aux_grammar`
Additional decomposition grammar rules.

Data fields available for single/hybrid structure prediction

Data fields for consensus structure prediction

Additional data fields for Distance Class Partitioning

These data fields are typically populated with meaningful data only if used in the context of Distance Class Partitioning

- unsigned int **maxD1**
Maximum allowed base pair distance to first reference.
- unsigned int **maxD2**
Maximum allowed base pair distance to second reference.
- short * **reference_pt1**
A pairtable of the first reference structure.
- short * **reference_pt2**
A pairtable of the second reference structure.
- unsigned int * **referenceBPs1**
Matrix containing number of basepairs of reference structure1 in interval [i,j].
- unsigned int * **referenceBPs2**
Matrix containing number of basepairs of reference structure2 in interval [i,j].
- unsigned int * **bpdist**
Matrix containing base pair distance of reference structure 1 and 2 on interval [i,j].
- unsigned int * **mm1**
Maximum matching matrix, reference struct 1 disallowed.
- unsigned int * **mm2**
Maximum matching matrix, reference struct 2 disallowed.

Additional data fields for local folding

These data fields are typically populated with meaningful data only if used in the context of local folding

- int **window_size**
window size for local folding sliding window approach
- char ** **ptype_local**
Pair type array (for local folding)
- `vrna_zsc_dat_t zscore_data`
Data structure with settings for z-score computations.

16.85.2.1.1 Field Documentation

16.85.2.1.1.1 type `const vrna_fc_type_e vrna_fc_s::type`

The type of the [vrna_fold_compound_t](#).

Currently possible values are [VRNA_FC_TYPE_SINGLE](#), and [VRNA_FC_TYPE_COMPARATIVE](#)

Warning

Do not edit this attribute, it will be automagically set by the corresponding `get()` methods for the [vrna_fold_compound_t](#). The value specified in this attribute dictates the set of other attributes to use within this data structure.

16.85.2.1.1.2 stat_cb `vrna_recursion_status_f vrna_fc_s::stat_cb`

Recursion status callback (usually called just before, and after recursive computations in the library).

See also

[vrna_recursion_status_f\(\)](#), [vrna_fold_compound_add_callback\(\)](#)

16.85.2.1.1.3 auxdata `void* vrna_fc_s::auxdata`

A pointer to auxiliary, user-defined data.

See also

[vrna_fold_compound_add_auxdata\(\)](#), [vrna_fold_compound_t.free_auxdata](#)

16.85.2.1.1.4 free_auxdata `vrna_auxdata_free_f vrna_fc_s::free_auxdata`

A callback to free auxiliary user data whenever the fold_compound itself is free'd.

See also

[vrna_fold_compound_t.auxdata](#), [vrna_auxdata_free_f\(\)](#)

16.85.2.1.1.5 sequence `char* vrna_fc_s::sequence`

The input sequence string.

Warning

Only available if

`type==VRNA_FC_TYPE_SINGLE`

16.85.2.1.1.6 sequence_encoding `short* vrna_fc_s::sequence_encoding`

Numerical encoding of the input sequence.

See also

[vrna_sequence_encode\(\)](#)

Warning

Only available if

`type==VRNA_FC_TYPE_SINGLE`

16.85.2.1.1.7 ptype `char* vrna_fc_s::ptype`

Pair type array.

Contains the numerical encoding of the pair type for each pair (i,j) used in MFE, Partition function and Evaluation computations.

Note

This array is always indexed via jindx, in contrast to previously different indexing between mfe and pf variants!

Warning

Only available if

```
type==VRNA_FC_TYPE_SINGLE
```

See also

[vrna_idx_col_wise\(\)](#), [vrna_ptypes\(\)](#)

16.85.2.1.1.8 ptype_pf_compat `char* vrna_fc_s::ptype_pf_compat`

ptype array indexed via iindx

Deprecated This attribute will vanish in the future! It's meant for backward compatibility only!

Warning

Only available if

```
type==VRNA_FC_TYPE_SINGLE
```

16.85.2.1.1.9 sc `vrna_sc_t* vrna_fc_s::sc`

The soft constraints for usage in structure prediction and evaluation.

Warning

Only available if

```
type==VRNA_FC_TYPE_SINGLE
```

16.85.2.1.1.10 sequences `char** vrna_fc_s::sequences`

The aligned sequences.

Note

The end of the alignment is indicated by a NULL pointer in the second dimension

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.11 n_seq unsigned int vrna_fc_s::n_seq
The number of sequences in the alignment.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.12 cons_seq char* vrna_fc_s::cons_seq
The consensus sequence of the aligned sequences.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.13 S_cons short* vrna_fc_s::S_cons
Numerical encoding of the consensus sequence.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.14 S short** vrna_fc_s::S
Numerical encoding of the sequences in the alignment.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.15 S5 short** vrna_fc_s::S5
S5[s][i] holds next base 5' of i in sequence s.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.16 S3 short** vrna_fc_s::S3
S3[s][i] holds next base 3' of i in sequence s.

Warning

Only available if

```
type==VRNA_FC_TYPE_COMPARATIVE
```

16.85.2.1.1.17 pscore `int* vrna_fc_s::pscore`
 Precomputed array of pair types expressed as pairing scores.

Warning

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

16.85.2.1.1.18 pscore_local `int** vrna_fc_s::pscore_local`
 Precomputed array of pair types expressed as pairing scores.

Warning

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

16.85.2.1.1.19 pscore_pf_compat `short* vrna_fc_s::pscore_pf_compat`
 Precomputed array of pair types expressed as pairing scores indexed via iindx.

Deprecated This attribute will vanish in the future!

Warning

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

16.85.2.1.1.20 scs `vrna_sc_t** vrna_fc_s::scs`
 A set of soft constraints (for each sequence in the alignment)

Warning

Only available if

`type==VRNA_FC_TYPE_COMPARATIVE`

16.85.3 Macro Definition Documentation

16.85.3.1 VRNA_STATUS_MFE_PRE

```
#define VRNA_STATUS_MFE_PRE (unsigned char)1
#include <ViennaRNA/fold_compound.h>
```

Status message indicating that MFE computations are about to begin.

See also

[vrna_fold_compound_t.stat_cb](#), [vrna_recursion_status_f\(\)](#), [vrna_mfe\(\)](#), [vrna_fold\(\)](#), [vrna_circfold\(\)](#),
[vrna_alifold\(\)](#), [vrna_circalifold\(\)](#), [vrna_cofold\(\)](#)

16.85.3.2 VRNA_STATUS_MFE_POST

```
#define VRNA_STATUS_MFE_POST (unsigned char)2
#include <ViennaRNA/fold_compound.h>
```

Status message indicating that MFE computations are finished.

See also

[vrna_fold_compound_t.stat_cb](#), [vrna_recursion_status_f\(\)](#), [vrna_mfe\(\)](#), [vrna_fold\(\)](#), [vrna_circfold\(\)](#), [vrna_alifold\(\)](#), [vrna_circalifold\(\)](#), [vrna_cofold\(\)](#)

16.85.3.3 VRNA_STATUS_PF_PRE

```
#define VRNA_STATUS_PF_PRE (unsigned char)3
#include <ViennaRNA/fold_compound.h>
```

Status message indicating that Partition function computations are about to begin.

See also

[vrna_fold_compound_t.stat_cb](#), [vrna_recursion_status_f\(\)](#), [vrna_pf\(\)](#)

16.85.3.4 VRNA_STATUS_PF_POST

```
#define VRNA_STATUS_PF_POST (unsigned char)4
#include <ViennaRNA/fold_compound.h>
```

Status message indicating that Partition function computations are finished.

See also

[vrna_fold_compound_t.stat_cb](#), [vrna_recursion_status_f\(\)](#), [vrna_pf\(\)](#)

16.85.3.5 VRNA_OPTION_MFE

```
#define VRNA_OPTION_MFE 1U
#include <ViennaRNA/fold_compound.h>
```

Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.

See also

[vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [VRNA_OPTION_EVAL_ONLY](#)

16.85.3.6 VRNA_OPTION_PF

```
#define VRNA_OPTION_PF 2U
#include <ViennaRNA/fold_compound.h>
```

Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.

See also

[vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [VRNA_OPTION_EVAL_ONLY](#)

16.85.3.7 VRNA_OPTION_EVAL_ONLY

```
#define VRNA_OPTION_EVAL_ONLY 8U
```

```
#include <ViennaRNA/fold_compound.h>
```

Option flag to specify that neither MFE, nor PF DP matrices are required.

Use this flag in conjunction with [VRNA_OPTION_MFE](#), and [VRNA_OPTION_PF](#) to save memory for a [vrna_fold_compound_t](#) obtained from [vrna_fold_compound\(\)](#), or [vrna_fold_compound_comparative\(\)](#) in cases where only energy evaluation but no structure prediction is required.

See also

[vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_eval_structure\(\)](#)

16.85.4 Typedef Documentation

16.85.4.1 vrna_auxdata_free_f

```
typedef void(* vrna_auxdata_free_f) (void *data)
```

```
#include <ViennaRNA/fold_compound.h>
```

Callback to free memory allocated for auxiliary user-provided data.

This type of user-implemented function usually deletes auxiliary data structures. The user must take care to free all the memory occupied by the data structure passed.

Notes on Callback Functions This callback is supposed to free memory occupied by an auxiliary data structure. It will be called when the [vrna_fold_compound_t](#) is erased from memory through a call to [vrna_fold_compound_free\(\)](#) and will be passed the address of memory previously bound to the [vrna_fold_compound_t](#) via [vrna_fold_compound_add_auxdata\(\)](#).

See also

[vrna_fold_compound_add_auxdata\(\)](#), [vrna_fold_compound_free\(\)](#), [vrna_fold_compound_add_callback\(\)](#)

Parameters

<i>data</i>	The data that needs to be free'd
-------------	----------------------------------

16.85.4.2 vrna_recursion_status_f

```
typedef void(* vrna_recursion_status_f) (unsigned char status, void *data)
```

```
#include <ViennaRNA/fold_compound.h>
```

Callback to perform specific user-defined actions before, or after recursive computations.

Notes on Callback Functions This function will be called to notify a third-party implementation about the status of a currently ongoing recursion. The purpose of this callback mechanism is to provide users with a simple way to ensure pre- and post conditions for auxiliary mechanisms attached to our implementations.

See also

[vrna_fold_compound_add_auxdata\(\)](#), [vrna_fold_compound_add_callback\(\)](#), [vrna_mfe\(\)](#), [vrna_pf\(\)](#), [VRNA_STATUS_MFE_PRE](#), [VRNA_STATUS_MFE_POST](#), [VRNA_STATUS_PF_PRE](#), [VRNA_STATUS_PF_POST](#)

Parameters

<i>status</i>	The status indicator
<i>data</i>	The data structure that was assigned with vrna_fold_compound_add_auxdata()

16.85.5 Enumeration Type Documentation

16.85.5.1 vrna_fc_type_e

```
enum vrna_fc_type_e
```

```
#include <ViennaRNA/fold_compound.h>
```

An enumerator that is used to specify the type of a [vrna_fold_compound_t](#).

Enumerator

VRNA_FC_TYPE_SINGLE	Type is suitable for single, and hybridizing sequences
VRNA_FC_TYPE_COMPARATIVE	Type is suitable for sequence alignments (consensus structure prediction)

16.85.6 Function Documentation

16.85.6.1 vrna_fold_compound()

```
vrna_fold_compound_t * vrna_fold_compound (
```

```
    const char * sequence,
```

```
    const vrna_md_t * md_p,
```

```
    unsigned int options )
```

```
#include <ViennaRNA/fold_compound.h>
```

Retrieve a [vrna_fold_compound_t](#) data structure for single sequences and hybridizing sequences.

This function provides an easy interface to obtain a prefilled [vrna_fold_compound_t](#) by passing a single sequence, or two concatenated sequences as input. For the latter, sequences need to be separated by an '&' character like this:

```
char *sequence = "GGGG&CCCC";
```

The optional parameter `md_p` can be used to specify the model details for successive computations based on the content of the generated [vrna_fold_compound_t](#). Passing NULL will instruct the function to use default model details. The third parameter `options` may be used to specify dynamic programming (DP) matrix requirements.

Options

- [VRNA_OPTION_DEFAULT](#) - Option flag to specify default settings/requirements.
- [VRNA_OPTION_MFE](#) - Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
- [VRNA_OPTION_PF](#) - Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
- [VRNA_OPTION_WINDOW](#) - Option flag to specify requirement of DP matrices for local folding approaches.

The above options may be OR-ed together.

If you just need the folding compound serving as a container for your data, you can simply pass [VRNA_OPTION_DEFAULT](#) to the `option` parameter. This creates a [vrna_fold_compound_t](#) without DP matrices, thus saving memory. Subsequent calls of any structure prediction function will then take care of allocating

the memory required for the DP matrices. If you only intend to evaluate structures instead of actually predicting them, you may use the `VRNA_OPTION_EVAL_ONLY` macro. This will seriously speedup the creation of the `vrna_fold_compound_t`.

Note

The sequence string must be uppercase, and should contain only RNA (resp. DNA) alphabet depending on what energy parameter set is used

See also

`vrna_fold_compound_free()`, `vrna_fold_compound_comparative()`, `vrna_md_t`

Parameters

<i>sequence</i>	A single sequence, or two concatenated sequences separated by an '&' character
<i>md_p</i>	An optional set of model details
<i>options</i>	The options for DP matrices memory allocation

Returns

A prefilled `vrna_fold_compound_t` ready to be used for computations (may be `NULL` on error)

16.85.6.2 `vrna_fold_compound_comparative()`

```
vrna_fold_compound_t * vrna_fold_compound_comparative (
    const char ** sequences,
    vrna_md_t * md_p,
    unsigned int options )
```

```
#include <ViennaRNA/fold_compound.h>
```

Retrieve a `vrna_fold_compound_t` data structure for sequence alignments.

This function provides an easy interface to obtain a prefilled `vrna_fold_compound_t` by passing an alignment of sequences.

The optional parameter `md_p` can be used to specify the model details for successive computations based on the content of the generated `vrna_fold_compound_t`. Passing `NULL` will instruct the function to use default model details. The third parameter `options` may be used to specify dynamic programming (DP) matrix requirements.

Options

- `VRNA_OPTION_DEFAULT` - Option flag to specify default settings/requirements.
- `VRNA_OPTION_MFE` - Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
- `VRNA_OPTION_PF` - Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
- `VRNA_OPTION_WINDOW` - Option flag to specify requirement of DP matrices for local folding approaches.

The above options may be OR-ed together.

If you just need the folding compound serving as a container for your data, you can simply pass `VRNA_OPTION_DEFAULT` to the `option` parameter. This creates a `vrna_fold_compound_t` without DP matrices, thus saving memory. Subsequent calls of any structure prediction function will then take care of allocating the memory required for the DP matrices. If you only intend to evaluate structures instead of actually predicting them, you may use the `VRNA_OPTION_EVAL_ONLY` macro. This will seriously speedup the creation of the `vrna_fold_compound_t`.

Note

The sequence strings must be uppercase, and should contain only RNA (resp. DNA) alphabet including gap characters depending on what energy parameter set is used.

See also

[vrna_fold_compound_free\(\)](#), [vrna_fold_compound\(\)](#), [vrna_md_t](#), [VRNA_OPTION_MFE](#), [VRNA_OPTION_PF](#), [VRNA_OPTION_EVAL_ONLY](#), [read_clustal\(\)](#)

Parameters

<i>sequences</i>	A sequence alignment including 'gap' characters
<i>md_p</i>	An optional set of model details
<i>options</i>	The options for DP matrices memory allocation

Returns

A prefilled `vrna_fold_compound_t` ready to be used for computations (may be `NULL` on error)

16.85.6.3 vrna_fold_compound_free()

```
void vrna_fold_compound_free (
    vrna_fold_compound_t * fc )
#include <ViennaRNA/fold_compound.h>
Free memory occupied by a vrna\_fold\_compound\_t.
```

See also

[vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_mx_mfe_free\(\)](#), [vrna_mx_pf_free\(\)](#)

Parameters

<i>fc</i>	The vrna_fold_compound_t that is to be erased from memory
-----------	---

16.85.6.4 vrna_fold_compound_add_auxdata()

```
void vrna_fold_compound_add_auxdata (
    vrna_fold_compound_t * fc,
    void * data,
    vrna_auxdata_free_f f )
#include <ViennaRNA/fold_compound.h>
Add auxiliary data to the vrna\_fold\_compound\_t.
```

This function allows one to bind arbitrary data to a [vrna_fold_compound_t](#) which may later on be used by one of the callback functions, e.g. [vrna_recursion_status_f\(\)](#). To allow for proper cleanup of the memory occupied by this auxiliary data, the user may also provide a pointer to a cleanup function that free's the corresponding memory. This function will be called automatically when the [vrna_fold_compound_t](#) is free'd with [vrna_fold_compound_free\(\)](#).

Note

Before attaching the arbitrary data pointer, this function will call the [vrna_auxdata_free_f\(\)](#) on any pre-existing data that is already attached.

See also

[vrna_auxdata_free_f\(\)](#)

Parameters

<i>fc</i>	The fold_compound the arbitrary data pointer should be associated with
<i>data</i>	A pointer to an arbitrary data structure
<i>f</i>	A pointer to function that free's memory occupied by the arbitrary data (May be NULL)

16.85.6.5 vrna_fold_compound_add_callback()

```
void vrna_fold_compound_add_callback (
    vrna_fold_compound_t * fc,
    vrna_recursion_status_f f )
#include <ViennaRNA/fold_compound.h>
```

Add a recursion status callback to the [vrna_fold_compound_t](#).

Binding a recursion status callback function to a [vrna_fold_compound_t](#) allows one to perform arbitrary operations just before, or after an actual recursive computations, e.g. MFE prediction, is performed by the RNAlib. The callback function will be provided with a pointer to its [vrna_fold_compound_t](#), and a status message. Hence, it has complete access to all variables that influence the recursive computations.

See also

[vrna_recursion_status_f\(\)](#), [vrna_fold_compound_t](#), [VRNA_STATUS_MFE_PRE](#), [VRNA_STATUS_MFE_POST](#), [VRNA_STATUS_PF_PRE](#), [VRNA_STATUS_PF_POST](#)

Parameters

<i>fc</i>	The fold_compound the callback function should be attached to
<i>f</i>	The pointer to the recursion status callback function

16.86 The Dynamic Programming Matrices

This module provides interfaces that deal with creation and destruction of dynamic programming matrices used within the RNAlib.

16.86.1 Detailed Description

This module provides interfaces that deal with creation and destruction of dynamic programming matrices used within the RNAlib.

Collaboration diagram for The Dynamic Programming Matrices:

Data Structures

- struct [vrna_mx_mfe_s](#)
Minimum Free Energy (MFE) Dynamic Programming (DP) matrices data structure required within the [vrna_fold_compound_t](#). More...
- struct [vrna_mx_pf_s](#)
Partition function (PF) Dynamic Programming (DP) matrices data structure required within the [vrna_fold_compound_t](#). More...

Typedefs

- typedef struct [vrna_mx_mfe_s](#) [vrna_mx_mfe_t](#)
Typename for the Minimum Free Energy (MFE) DP matrices data structure [vrna_mx_mfe_s](#).
- typedef struct [vrna_mx_pf_s](#) [vrna_mx_pf_t](#)
Typename for the Partition Function (PF) DP matrices data structure [vrna_mx_pf_s](#).

Enumerations

- enum [vrna_mx_type_e](#) { [VRNA_MX_DEFAULT](#) , [VRNA_MX_WINDOW](#) , [VRNA_MX_2DFOLD](#) }
An enumerator that is used to specify the type of a polymorphic Dynamic Programming (DP) matrix data structure.

Functions

- int [vrna_mx_add](#) ([vrna_fold_compound_t](#) *vc, [vrna_mx_type_e](#) type, unsigned int options)
Add Dynamic Programming (DP) matrices (allocate memory)
- void [vrna_mx_mfe_free](#) ([vrna_fold_compound_t](#) *vc)
Free memory occupied by the Minimum Free Energy (MFE) Dynamic Programming (DP) matrices.
- void [vrna_mx_pf_free](#) ([vrna_fold_compound_t](#) *vc)
Free memory occupied by the Partition Function (PF) Dynamic Programming (DP) matrices.

16.86.2 Data Structure Documentation

16.86.2.1 struct [vrna_mx_mfe_s](#)

Minimum Free Energy (MFE) Dynamic Programming (DP) matrices data structure required within the [vrna_fold_compound_t](#).

Data Fields

Common fields for MFE matrices

- const [vrna_mx_type_e](#) type
- unsigned int **length**
Length of the sequence, therefore an indicator of the size of the DP matrices.
- unsigned int [strands](#)

Default DP matrices

Note

These data fields are available if
`vrna_mx_mfe_t.type == VRNA_MX_DEFAULT`

Local Folding DP matrices using window approach

Note

These data fields are available if
`vrna_mx_mfe_t.type == VRNA_MX_WINDOW`

Distance Class DP matrices

Note

These data fields are available if
`vrna_mx_mfe_t.type == VRNA_MX_2DFOLD`

16.86.2.1.1 Field Documentation

16.86.2.1.1.1 type const [vrna_mx_type_e](#) [vrna_mx_mfe_s::type](#)
Type of the DP matrices

16.86.2.1.1.2 strands unsigned int `vrna_mx_mfe_s::strands`
 Number of strands

16.86.2.2 struct `vrna_mx_pf_s`

Partition function (PF) Dynamic Programming (DP) matrices data structure required within the `vrna_fold_compound_t`.

Data Fields

Common fields for DP matrices

- const `vrna_mx_type_e` type
- unsigned int `length`
- `FLT_OR_DBL` * `scale`
- `FLT_OR_DBL` * `expMLbase`

Default PF matrices

Note

These data fields are available if
`vrna_mx_pf_t.type == VRNA_MX_DEFAULT`

Local Folding DP matrices using window approach

Note

These data fields are available if
`vrna_mx_mfe_t.type == VRNA_MX_WINDOW`

Distance Class DP matrices

Note

These data fields are available if
`vrna_mx_pf_t.type == VRNA_MX_2DFOLD`

16.86.2.2.1 Field Documentation

16.86.2.2.1.1 type const `vrna_mx_type_e` `vrna_mx_pf_s::type`
 Type of the DP matrices

16.86.2.2.1.2 length unsigned int `vrna_mx_pf_s::length`
 Size of the DP matrices (i.e. sequence length)

16.86.2.2.1.3 scale `FLT_OR_DBL`* `vrna_mx_pf_s::scale`
 Boltzmann factor scaling

16.86.2.2.1.4 expMLbase `FLT_OR_DBL`* `vrna_mx_pf_s::expMLbase`
 Boltzmann factors for unpaired bases in multibranch loop

16.86.3 Enumeration Type Documentation

16.86.3.1 `vrna_mx_type_e`

```
enum vrna_mx_type_e
#include <ViennaRNA/dp_matrices.h>
```

An enumerator that is used to specify the type of a polymorphic Dynamic Programming (DP) matrix data structure.

See also

`vrna_mx_mfe_t`, `vrna_mx_pf_t`

Enumerator

VRNA_MX_DEFAULT	Default DP matrices.
VRNA_MX_WINDOW	DP matrices suitable for local structure prediction using window approach. See also vrna_mfe_window() , vrna_mfe_window_zscore() , pfl_fold()
VRNA_MX_2DFOLD	DP matrices suitable for distance class partitioned structure prediction. See also vrna_mfe_TwoD() , vrna_pf_TwoD()

16.86.4 Function Documentation

16.86.4.1 `vrna_mx_add()`

```
int vrna_mx_add (
    vrna_fold_compound_t * vc,
    vrna_mx_type_e type,
    unsigned int options )
#include <ViennaRNA/dp_matrices.h>
```

Add Dynamic Programming (DP) matrices (allocate memory)

This function adds DP matrices of a specific type to the provided `vrna_fold_compound_t`, such that successive DP recursion can be applied. The function caller has to specify which type of DP matrix is requested, see `vrna_mx_type_e`, and what kind of recursive algorithm will be applied later on, using the parameters `type`, and `options`, respectively. For the latter, Minimum free energy (MFE), and Partition function (PF) computations are distinguished. A third option that may be passed is `VRNA_OPTION_HYBRID`, indicating that auxiliary DP arrays are required for RNA-RNA interaction prediction.

Note

Usually, there is no need to call this function, since the constructors of `vrna_fold_compound_t` are handling all the DP matrix memory allocation.

See also

[vrna_mx_mfe_add\(\)](#), [vrna_mx_pf_add\(\)](#), [vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_fold_compound_free\(\)](#), [vrna_mx_pf_free\(\)](#), [vrna_mx_mfe_free\(\)](#), [vrna_mx_type_e](#), `VRNA_OPTION_MFE`, `VRNA_OPTION_PF`, `VRNA_OPTION_HYBRID`, `VRNA_OPTION_EVAL_ONLY`

Parameters

<i>vc</i>	The <code>vrna_fold_compound_t</code> that holds pointers to the DP matrices
<i>type</i>	The type of DP matrices requested
<i>options</i>	Option flags that specify the kind of DP matrices, such as MFE or PF arrays, and auxiliary requirements

Returns

1 if DP matrices were properly allocated and attached, 0 otherwise

16.86.4.2 vrna_mx_mfe_free()

```
void vrna_mx_mfe_free (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/dp_matrices.h>
```

Free memory occupied by the Minimum Free Energy (MFE) Dynamic Programming (DP) matrices.

See also

[vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_fold_compound_free\(\)](#), [vrna_mx_pf_free\(\)](#)

Parameters

vc	The vrna_fold_compound_t storing the MFE DP matrices that are to be erased from memory
----	--

16.86.4.3 vrna_mx_pf_free()

```
void vrna_mx_pf_free (
    vrna_fold_compound_t * vc )
#include <ViennaRNA/dp_matrices.h>
```

Free memory occupied by the Partition Function (PF) Dynamic Programming (DP) matrices.

See also

[vrna_fold_compound\(\)](#), [vrna_fold_compound_comparative\(\)](#), [vrna_fold_compound_free\(\)](#), [vrna_mx_mfe_free\(\)](#)

Parameters

vc	The vrna_fold_compound_t storing the PF DP matrices that are to be erased from memory
----	---

16.87 Hash Tables

Various implementations of hash table functions.

16.87.1 Detailed Description

Various implementations of hash table functions.

Hash tables are common data structures that allow for fast random access to the data that is stored within.

Here, we provide an abstract implementation of a hash table interface and a concrete implementation for pairs of secondary structure and corresponding free energy value. Collaboration diagram for Hash Tables:

Files

- file [hash_tables.h](#)

Implementations of hash table functions.

Data Structures

- struct [vrna_ht_entry_db_t](#)

Default hash table entry. [More...](#)

Abstract interface

- typedef struct vrna_hash_table_s * [vrna_hash_table_t](#)
A hash table object.
- typedef int(* [vrna_ht_cmp_f](#)) (void *x, void *y)
Callback function to compare two hash table entries.
- typedef int() [vrna_callback_ht_compare_entries](#)(void *x, void *y)
- typedef unsigned int(* [vrna_ht_hashfunc_f](#)) (void *x, unsigned long hashtable_size)
Callback function to generate a hash key, i.e. hash function.
- typedef unsigned int() [vrna_callback_ht_hash_function](#)(void *x, unsigned long hashtable_size)
- typedef int(* [vrna_ht_free_f](#)) (void *x)
Callback function to free a hash table entry.
- typedef int() [vrna_callback_ht_free_entry](#)(void *x)
- [vrna_hash_table_t](#) [vrna_ht_init](#) (unsigned int b, [vrna_ht_cmp_f](#) compare_function, [vrna_ht_hashfunc_f](#) hash_function, [vrna_ht_free_f](#) free_hash_entry)
Get an initialized hash table.
- unsigned long [vrna_ht_size](#) ([vrna_hash_table_t](#) ht)
Get the size of the hash table.
- unsigned long [vrna_ht_collisions](#) (struct vrna_hash_table_s *ht)
Get the number of collisions in the hash table.
- void * [vrna_ht_get](#) ([vrna_hash_table_t](#) ht, void *x)
Get an element from the hash table.
- int [vrna_ht_insert](#) ([vrna_hash_table_t](#) ht, void *x)
Insert an object into a hash table.
- void [vrna_ht_remove](#) ([vrna_hash_table_t](#) ht, void *x)
Remove an object from the hash table.
- void [vrna_ht_clear](#) ([vrna_hash_table_t](#) ht)
Clear the hash table.
- void [vrna_ht_free](#) ([vrna_hash_table_t](#) ht)
Free all memory occupied by the hash table.

Dot-Bracket / Free Energy entries

- int [vrna_ht_db_comp](#) (void *x, void *y)
Default hash table entry comparison.
- unsigned int [vrna_ht_db_hash_func](#) (void *x, unsigned long hashtable_size)
Default hash function.
- int [vrna_ht_db_free_entry](#) (void *hash_entry)
Default function to free memory occupied by a hash entry.

16.87.2 Data Structure Documentation

16.87.2.1 struct vrna_ht_entry_db_t

Default hash table entry.

See also

[vrna_ht_init\(\)](#), [vrna_ht_db_comp\(\)](#), [vrna_ht_db_hash_func\(\)](#), [vrna_ht_db_free_entry\(\)](#)

Data Fields

- char * [structure](#)
- float [energy](#)

16.87.2.1.1 Field Documentation

16.87.2.1.1.1 **structure** `char* vrna_ht_entry_db_t::structure`

A secondary structure in dot-bracket notation

16.87.2.1.1.2 **energy** `float vrna_ht_entry_db_t::energy`

The free energy of `structure`

16.87.3 Typedef Documentation

16.87.3.1 `vrna_hash_table_t`

```
typedef struct vrna_hash_table_s* vrna_hash_table_t
#include <ViennaRNA/datastructures/hash_tables.h>
```

A hash table object.

See also

[vrna_ht_init\(\)](#), [vrna_ht_free\(\)](#)

16.87.3.2 `vrna_ht_cmp_f`

```
typedef int(* vrna_ht_cmp_f) (void *x, void *y)
#include <ViennaRNA/datastructures/hash_tables.h>
```

Callback function to compare two hash table entries.

See also

[vrna_ht_init\(\)](#), [vrna_ht_db_comp\(\)](#)

Parameters

<i>x</i>	A hash table entry
<i>y</i>	A hash table entry

Returns

-1 if *x* is smaller, +1 if *x* is larger than *y*. 0 if *x* == *y*

16.87.3.3 `vrna_ht_hashfunc_f`

```
typedef unsigned int(* vrna_ht_hashfunc_f) (void *x, unsigned long hashtable_size)
#include <ViennaRNA/datastructures/hash_tables.h>
```

Callback function to generate a hash key, i.e. hash function.

See also

[vrna_ht_init\(\)](#), [vrna_ht_db_hash_func\(\)](#)

Parameters

<i>x</i>	A hash table entry
<i>hashtable_size</i>	The size of the hash table

Returns

The hash table key for entry x

16.87.3.4 `vrna_ht_free_f`

```
typedef int(* vrna_ht_free_f) (void *x)
#include <ViennaRNA/datastructures/hash_tables.h>
```

Callback function to free a hash table entry.

See also

[vrna_ht_init\(\)](#), [vrna_ht_db_free_entry\(\)](#)

Parameters

x	A hash table entry
-----	--------------------

Returns

0 on success

16.87.4 Function Documentation

16.87.4.1 `vrna_ht_init()`

```
vrna_hash_table_t vrna_ht_init (
    unsigned int b,
    vrna_ht_cmp_f compare_function,
    vrna_ht_hashfunc_f hash_function,
    vrna_ht_free_f free_hash_entry )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Get an initialized hash table.

This function returns a ready-to-use hash table with pre-allocated memory for a particular number of entries.

Note

If all function pointers are `NULL`, this function initializes the hash table with *default functions*, i.e.

- [vrna_ht_db_comp\(\)](#) for the `compare_function`,
- [vrna_ht_db_hash_func\(\)](#) for the `hash_function`, and
- [vrna_ht_db_free_entry\(\)](#) for the `free_hash_entry`

arguments.

Warning

If `hash_bits` is larger than 27 you have to compile it with the flag `gcc -mmodel=large`.

Parameters

b	Number of bits for the hash table. This determines the size ($2^b - 1$).
<code>compare_function</code>	A function pointer to compare any two entries in the hash table (may be <code>NULL</code>)
<code>hash_function</code>	A function pointer to retrieve the hash value of any entry (may be <code>NULL</code>)
<code>free_hash_entry</code>	A function pointer to free the memory occupied by any entry (may be <code>NULL</code>)

Returns

An initialized, empty hash table, or `NULL` on any error

16.87.4.2 vrna_ht_size()

```
unsigned long vrna_ht_size (
    vrna_hash_table_t ht )
#include <ViennaRNA/datastructures/hash_tables.h>
Get the size of the hash table.
```

Parameters

<i>ht</i>	The hash table
-----------	----------------

Returns

The size of the hash table, i.e. the maximum number of entries

16.87.4.3 vrna_ht_collisions()

```
unsigned long vrna_ht_collisions (
    struct vrna_hash_table_s * ht )
#include <ViennaRNA/datastructures/hash_tables.h>
Get the number of collisions in the hash table.
```

Parameters

<i>ht</i>	The hash table
-----------	----------------

Returns

The number of collisions in the hash table

16.87.4.4 vrna_ht_get()

```
void * vrna_ht_get (
    vrna_hash_table_t ht,
    void * x )
#include <ViennaRNA/datastructures/hash_tables.h>
Get an element from the hash table.
```

This function takes an object `x` and performs a look-up whether the object is stored within the hash table `ht`. If the object is already stored in `ht`, the function simply returns the entry, otherwise it returns `NULL`.

See also

[vrna_ht_insert\(\)](#), [vrna_hash_delete\(\)](#), [vrna_ht_init\(\)](#)

Parameters

<i>ht</i>	The hash table
<i>x</i>	The hash entry to look-up

Returns

The entry `x` if it is stored in `ht`, `NULL` otherwise

16.87.4.5 vrna_ht_insert()

```
int vrna_ht_insert (
    vrna_hash_table_t ht,
    void * x )
#include <ViennaRNA/datastructures/hash_tables.h>
Insert an object into a hash table.
Writes the pointer to your hash entry into the table.
```

Warning

In case of collisions, this function simply increments the hash key until a free entry in the hash table is found.

See also

[vrna_ht_init\(\)](#), [vrna_hash_delete\(\)](#), [vrna_ht_clear\(\)](#)

Parameters

<i>ht</i>	The hash table
<i>x</i>	The hash entry

Returns

0 on success, 1 if the value is already in the hash table, -1 on error.

16.87.4.6 vrna_ht_remove()

```
void vrna_ht_remove (
    vrna_hash_table_t ht,
    void * x )
#include <ViennaRNA/datastructures/hash_tables.h>
Remove an object from the hash table.
Deletes the pointer to your hash entry from the table.
```

Note

This function doesn't free any memory occupied by the hash entry.

Parameters

<i>ht</i>	The hash table
<i>x</i>	The hash entry

16.87.4.7 vrna_ht_clear()

```
void vrna_ht_clear (
    vrna_hash_table_t ht )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Clear the hash table.

This function removes all entries from the hash table and automatically free's the memory occupied by each entry using the bound [vrna_ht_free_f\(\)](#) function.

See also

[vrna_ht_free\(\)](#), [vrna_ht_init\(\)](#)

Parameters

<i>ht</i>	The hash table
-----------	----------------

16.87.4.8 vrna_ht_free()

```
void vrna_ht_free (
    vrna_hash_table_t ht )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Free all memory occupied by the hash table.

This function removes all entries from the hash table by calling the [vrna_ht_free_f\(\)](#) function for each entry. Finally, the memory occupied by the hash table itself is free'd as well.

Parameters

<i>ht</i>	The hash table
-----------	----------------

16.87.4.9 vrna_ht_db_comp()

```
int vrna_ht_db_comp (
    void * x,
    void * y )
#include <ViennaRNA/datastructures/hash_tables.h>
```

Default hash table entry comparison.

This is the default comparison function for hash table entries. It assumes the both entries *x* and *y* are of type [vrna_ht_entry_db_t](#) and compares the `structure` attribute of both entries

See also

[vrna_ht_entry_db_t](#), [vrna_ht_init\(\)](#), [vrna_ht_db_hash_func\(\)](#), [vrna_ht_db_free_entry\(\)](#)

Parameters

<i>x</i>	A hash table entry of type vrna_ht_entry_db_t
<i>y</i>	A hash table entry of type vrna_ht_entry_db_t

Returns

-1 if *x* is smaller, +1 if *x* is larger than *y*. 0 if both are equal.

16.87.4.10 vrna_ht_db_hash_func()

```
unsigned int vrna_ht_db_hash_func (
    void * x,
    unsigned long hashtable_size )
```

```
#include <ViennaRNA/datastructures/hash_tables.h>
```

Default hash function.

This is the default hash function for hash table insertion/lookup. It assumes that entries are of type [vrna_ht_entry_db_t](#) and uses the Bob Jenkins 1996 mix function to create a hash key from the `structure` attribute of the hash entry.

See also

[vrna_ht_entry_db_t](#), [vrna_ht_init\(\)](#), [vrna_ht_db_comp\(\)](#), [vrna_ht_db_free_entry\(\)](#)

Parameters

<code>x</code>	A hash table entry to compute the key for
<code>hashtable_size</code>	The size of the hash table

Returns

The hash key for entry `x`

16.87.4.11 vrna_ht_db_free_entry()

```
int vrna_ht_db_free_entry (
    void * hash_entry )
```

```
#include <ViennaRNA/datastructures/hash_tables.h>
```

Default function to free memory occupied by a hash entry.

This function assumes that hash entries are of type [vrna_ht_entry_db_t](#) and free's the memory occupied by that entry.

See also

[vrna_ht_entry_db_t](#), [vrna_ht_init\(\)](#), [vrna_ht_db_comp\(\)](#), [vrna_ht_db_hash_func\(\)](#)

Parameters

<code>hash_entry</code>	The hash entry to remove from memory
-------------------------	--------------------------------------

Returns

0 on success

16.88 Heaps

Interface for an abstract implementation of a heap data structure.

16.88.1 Detailed Description

Interface for an abstract implementation of a heap data structure.

Collaboration diagram for Heaps:

Files

- file [heap.h](#)

Implementation of an abstract heap data structure.

Typedefs

- typedef struct vrna_heap_s * [vrna_heap_t](#)
An abstract heap data structure.
- typedef int(* [vrna_heap_cmp_f](#)) (const void *a, const void *b, void *data)
Heap compare function prototype.
- typedef size_t(* [vrna_heap_get_pos_f](#)) (const void *a, void *data)
Retrieve the position of a particular heap entry within the heap.
- typedef void(* [vrna_heap_set_pos_f](#)) (const void *a, size_t pos, void *data)
Store the position of a particular heap entry within the heap.

Functions

- [vrna_heap_t](#) [vrna_heap_init](#) (size_t n, [vrna_heap_cmp_f](#) cmp, [vrna_heap_get_pos_f](#) get_entry_pos, [vrna_heap_set_pos_f](#) set_entry_pos, void *data)
Initialize a heap data structure.
- void [vrna_heap_free](#) ([vrna_heap_t](#) h)
Free memory occupied by a heap data structure.
- size_t [vrna_heap_size](#) (struct vrna_heap_s *h)
Get the size of a heap data structure, i.e. the number of stored elements.
- void [vrna_heap_insert](#) ([vrna_heap_t](#) h, void *v)
Insert an element into the heap.
- void * [vrna_heap_pop](#) ([vrna_heap_t](#) h)
Pop (remove and return) the object at the root of the heap.
- const void * [vrna_heap_top](#) ([vrna_heap_t](#) h)
Get the object at the root of the heap.
- void * [vrna_heap_remove](#) ([vrna_heap_t](#) h, const void *v)
Remove an arbitrary element within the heap.
- void * [vrna_heap_update](#) ([vrna_heap_t](#) h, void *v)
Update an arbitrary element within the heap.

16.88.2 Typedef Documentation

16.88.2.1 vrna_heap_t

```
typedef struct vrna_heap_s* vrna\_heap\_t
#include <ViennaRNA/datastructures/heap.h>
```

An abstract heap data structure.

See also

[vrna_heap_init\(\)](#), [vrna_heap_free\(\)](#), [vrna_heap_insert\(\)](#), [vrna_heap_pop\(\)](#), [vrna_heap_top\(\)](#), [vrna_heap_remove\(\)](#), [vrna_heap_update\(\)](#)

16.88.2.2 vrna_heap_cmp_f

```
typedef int(* vrna\_heap\_cmp\_f) (const void *a, const void *b, void *data)
#include <ViennaRNA/datastructures/heap.h>
```

Heap compare function prototype.

Use this prototype to design the compare function for the heap implementation. The arbitrary data pointer `data` may be used to get access to further information required to actually compare the two values `a` and `b`.

Note

The heap implementation acts as a *min-heap*, therefore, the minimum element will be present at the heap's root. In case a *max-heap* is required, simply reverse the logic of this compare function.

Parameters

<i>a</i>	The first object to compare
<i>b</i>	The second object to compare
<i>data</i>	An arbitrary data pointer passed through from the heap implementation

Returns

A value less than zero if $a < b$, a value greater than zero if $a > b$, and 0 otherwise

16.88.2.3 vrna_heap_get_pos_f

```
typedef size_t(* vrna_heap_get_pos_f) (const void *a, void *data)
#include <ViennaRNA/datastructures/heap.h>
```

Retrieve the position of a particular heap entry within the heap.

Parameters

<i>a</i>	The object to look-up within the heap
<i>data</i>	An arbitrary data pointer passed through from the heap implementation

Returns

The position of the element *a* within the heap, or 0 if it is not in the heap

16.88.2.4 vrna_heap_set_pos_f

```
typedef void(* vrna_heap_set_pos_f) (const void *a, size_t pos, void *data)
#include <ViennaRNA/datastructures/heap.h>
```

Store the position of a particular heap entry within the heap.

Parameters

<i>a</i>	The object whose position shall be stored
<i>pos</i>	The current position of <i>a</i> within the heap, or 0 if <i>a</i> was deleted
<i>data</i>	An arbitrary data pointer passed through from the heap implementation

16.88.3 Function Documentation**16.88.3.1 vrna_heap_init()**

```
vrna_heap_t vrna_heap_init (
    size_t n,
    vrna_heap_cmp_f cmp,
    vrna_heap_get_pos_f get_entry_pos,
    vrna_heap_set_pos_f set_entry_pos,
    void * data )
#include <ViennaRNA/datastructures/heap.h>
```

Initialize a heap data structure.

This function initializes a heap data structure. The implementation is based on a *min-heap*, i.e. the minimal element is located at the root of the heap. However, by reversing the logic of the compare function, one can easily transform this into a *max-heap* implementation.

Beside the regular operations on a heap data structure, we implement removal and update of arbitrary elements within the heap. For that purpose, however, one requires a reverse-index lookup system that, (i) for a given element stores the current position in the heap, and (ii) allows for fast lookup of an elements current position within the heap. The corresponding getter- and setter- functions may be provided through the arguments `get_entry_pos` and `set_entry_pos`, respectively.

Sometimes, it is difficult to simply compare two data structures without any context. Therefore, the compare function is provided with a user-defined data pointer that can hold any context required.

Warning

If any of the arguments `get_entry_pos` or `set_entry_pos` is NULL, the operations `vrna_heap_update()` and `vrna_heap_remove()` won't work.

See also

[vrna_heap_free\(\)](#), [vrna_heap_insert\(\)](#), [vrna_heap_pop\(\)](#), [vrna_heap_top\(\)](#), [vrna_heap_remove\(\)](#), [vrna_heap_update\(\)](#), [vrna_heap_t](#), [vrna_heap_cmp_f](#), [vrna_heap_get_pos_f](#), [vrna_heap_set_pos_f](#)

Parameters

<i>n</i>	The initial size of the heap, i.e. the number of elements to store
<i>cmp</i>	The address of a compare function that will be used to fulfill the partial order requirement
<i>get_entry_pos</i>	The address of a function that retrieves the position of an element within the heap (or NULL)
<i>set_entry_pos</i>	The address of a function that stores the position of an element within the heap (or NULL)
<i>data</i>	An arbitrary data pointer passed through to the compare function <code>cmp</code> , and the set/get functions <code>get_entry_pos</code> / <code>set_entry_pos</code>

Returns

An initialized heap data structure, or NULL on error

16.88.3.2 vrna_heap_free()

```
void vrna_heap_free (
    vrna_heap_t h )
#include <ViennaRNA/datastructures/heap.h>
Free memory occupied by a heap data structure.
```

See also

[vrna_heap_init\(\)](#)

Parameters

<i>h</i>	The heap that should be free'd
----------	--------------------------------

16.88.3.3 vrna_heap_size()

```
size_t vrna_heap_size (
    struct vrna_heap_s * h )
```

```
#include <ViennaRNA/datastructures/heap.h>
```

Get the size of a heap data structure, i.e. the number of stored elements.

Parameters

<i>h</i>	The heap data structure
----------	-------------------------

Returns

The number of elements currently stored in the heap, or 0 upon any error

16.88.3.4 vrna_heap_insert()

```
void vrna_heap_insert (
    vrna_heap_t h,
    void * v )
```

```
#include <ViennaRNA/datastructures/heap.h>
```

Insert an element into the heap.

See also

[vrna_heap_init\(\)](#), [vrna_heap_pop\(\)](#), [vrna_heap_top\(\)](#), [vrna_heap_free\(\)](#), [vrna_heap_remove\(\)](#), [vrna_heap_update\(\)](#)

Parameters

<i>h</i>	The heap data structure
<i>v</i>	A pointer to the object that is about to be inserted into the heap

16.88.3.5 vrna_heap_pop()

```
void * vrna_heap_pop (
    vrna_heap_t h )
```

```
#include <ViennaRNA/datastructures/heap.h>
```

Pop (remove and return) the object at the root of the heap.

This function removes the root from the heap and returns it to the caller.

See also

[vrna_heap_init\(\)](#), [vrna_heap_top\(\)](#), [vrna_heap_insert\(\)](#), [vrna_heap_free\(\)](#), [vrna_heap_remove\(\)](#), [vrna_heap_update\(\)](#)

Parameters

<i>h</i>	The heap data structure
----------	-------------------------

Returns

The object at the root of the heap, i.e. the minimal element (or NULL if (a) the heap is empty or (b) any error occurred)

16.88.3.6 vrna_heap_top()

```
const void * vrna_heap_top (
    vrna_heap_t h )
```

```
#include <ViennaRNA/datastructures/heap.h>
```

Get the object at the root of the heap.

See also

[vrna_heap_init\(\)](#), [vrna_heap_pop\(\)](#), [vrna_heap_insert\(\)](#), [vrna_heap_free\(\)](#), [vrna_heap_remove\(\)](#), [vrna_heap_update\(\)](#)

Parameters

<i>h</i>	The heap data structure
----------	-------------------------

Returns

The object at the root of the heap, i.e. the minimal element (or NULL if (a) the heap is empty or (b) any error occurred)

16.88.3.7 vrna_heap_remove()

```
void * vrna_heap_remove (
    vrna_heap_t h,
    const void * v )
```

```
#include <ViennaRNA/datastructures/heap.h>
```

Remove an arbitrary element within the heap.

See also

[vrna_heap_init\(\)](#), [vrna_heap_get_pos_f](#), [vrna_heap_set_pos_f](#), [vrna_heap_pop\(\)](#), [vrna_heap_free\(\)](#)

Warning

This function won't work if the heap was not properly initialized with callback functions for fast reverse-index mapping!

Parameters

<i>h</i>	The heap data structure
<i>v</i>	The object to remove from the heap

Returns

The object that was removed from the heap (or NULL if (a) it wasn't found or (b) any error occurred)

16.88.3.8 vrna_heap_update()

```
void * vrna_heap_update (
    vrna_heap_t h,
    void * v )
```

```
#include <ViennaRNA/datastructures/heap.h>
```

Update an arbitrary element within the heap.

Note

If the object that is to be updated is not currently stored in the heap, it will be inserted. In this case, the function returns NULL.

Warning

This function won't work if the heap was not properly initialized with callback functions for fast reverse-index mapping!

See also

[vrna_heap_init\(\)](#), [vrna_heap_get_pos_f](#), [vrna_heap_set_pos_f](#), [vrna_heap_pop\(\)](#), [vrna_heap_remove\(\)](#), [vrna_heap_free\(\)](#)

Parameters

<i>h</i>	The heap data structure
<i>v</i>	The object to update

Returns

The 'previous' object within the heap that now got replaced by *v* (or NULL if (a) it wasn't found or (b) any error occurred)

16.89 Arrays

Interface for an abstract implementation of an array data structure.

16.89.1 Detailed Description

Interface for an abstract implementation of an array data structure.

Arrays of a particular `Type` are defined and initialized using the following code:

```
vrna_array(Type) my_array;
vrna_array_init(my_array);
```

or equivalently:

```
vrna_array_make(Type, my_array);
```

Dynamic arrays can be used like regular pointers, i.e. elements are simply addressed using the `[]` operator, e.g.:

```
my_array[1] = 42;
```

Using the [vrna_array_append\(\)](#) macro, items can be safely appended and the array will grow accordingly if required:

```
vrna_array_append(my_array, item);
```

Finally, memory occupied by an array must be released using the [vrna_array_free\(\)](#) macro:

```
vrna_array_free(my_array);
```

Use the [vrna_array_size\(\)](#) macro to get the number of items stored in an array, e.g. for looping over its elements:

```
// define and initialize
vrna_array_make(int, my_array);

// append some items
vrna_array_append(my_array, 42);
vrna_array_append(my_array, 23);
vrna_array_append(my_array, 5);

// loop over items and print
for (size_t i = 0; i < vrna_array_size(my_array); i++)
    printf("%d\n", my_array[i]);

// release memory of the array
vrna_array_free(my_array);
```

Under the hood, arrays are preceded by a header that actually stores the number of items they contain and the capacity of elements they are able to store. The general ideas for this implementation are taken from Ginger Bill's C Helper Library (public domain). Collaboration diagram for Arrays:

Files

- file [array.h](#)

A macro-based dynamic array implementation.

Data Structures

- struct [vrna_array_header_s](#)
The header of an array. [More...](#)

Macros

- `#define vrna_array(Type) Type *`
Define an array.
- `#define vrna_array_make(Type, Name) Type * Name; vrna_array_init(Name)`
Make an array *Name* of type *Type*.
- `#define VRNA_ARRAY_GROW_FORMULA(n) (1.4 * (n) + 8)`
The default growth formula for array.
- `#define VRNA_ARRAY_HEADER(input) ((vrna_array_header_t *) (input) - 1)`
Retrieve a pointer to the header of an array *input*.
- `#define vrna_array_size(input) (VRNA_ARRAY_HEADER(input)->num)`
Get the number of elements of an array *input*.
- `#define vrna_array_capacity(input) (VRNA_ARRAY_HEADER(input)->size)`
Get the size of an array *input*, i.e. its actual capacity.
- `#define vrna_array_set_capacity(a, capacity)`
Explicitly set the capacity of an array *a*.
- `#define vrna_array_init_size(a, init_size)`
Initialize an array *a* with a particular pre-allocated size *init_size*.
- `#define vrna_array_init(a) vrna_array_init_size(a, VRNA_ARRAY_GROW_FORMULA(0));`
Initialize an array *a*.
- `#define vrna_array_free(a)`
Release memory of an array *a*.
- `#define vrna_array_append(a, item)`
Safely append an item to an array *a*.
- `#define vrna_array_grow(a, min_capacity)`
Grow an array *a* to provide a minimum capacity *min_capacity*.

Typedefs

- typedef struct [vrna_array_header_s](#) [vrna_array_header_t](#)
The header of an array.

Functions

- `VRNA_NO_INLINE void * vrna__array_set_capacity (void *array, size_t capacity, size_t element_size)`
Explicitly set the capacity of an array.

16.89.2 Data Structure Documentation

16.89.2.1 struct vrna_array_header_s

The header of an array.

Data Fields

- `size_t num`
The number of elements in an array.
- `size_t size`
The actual capacity of an array.

16.89.3 Macro Definition Documentation

16.89.3.1 vrna_array_init_size

```
#define vrna_array_init_size(
    a,
    init_size )
#include <ViennaRNA/datastructures/array.h>
```

Value:

```
do { \
    void **a_ptr = (void **)&(a); \
    size_t size = sizeof(*(a)) * (init_size) + sizeof(vrna_array_header_t); \
    vrna_array_header_t *h = (void *)vrna_alloc(size); \
    h->num = 0; \
    h->size = init_size; \
    *a_ptr = (void *) (h + 1); \
} while (0)
```

Initialize an array a with a particular pre-allocated size init_size.

16.89.4 Function Documentation

16.89.4.1 vrna__array_set_capacity()

```
VRNA_NO_INLINE void * vrna__array_set_capacity (
    void * array,
    size_t capacity,
    size_t element_size )
#include <ViennaRNA/datastructures/array.h>
```

Explicitely set the capacity of an array.

Note

Do not use this function. Rather resort to the [vrna_array_set_capacity](#) macro

16.90 Buffers

Functions that provide dynamically buffered stream-like data structures.

16.90.1 Detailed Description

Functions that provide dynamically buffered stream-like data structures.

Collaboration diagram for Buffers:

Files

- file [char_stream.h](#)
Implementation of a dynamic, buffered character stream.
- file [stream_output.h](#)
An implementation of a buffered, ordered stream output data structure.

Typedefs

- typedef struct vrna_ordered_stream_s * **vrna_ostream_t**
An ordered output stream structure with unordered insert capabilities.
- typedef void(* [vrna_stream_output_f](#)) (void *auxdata, unsigned int i, void *data)
Ordered stream processing callback.

Functions

- `vrna_cstr_t vrna_cstr` (`size_t size`, `FILE *output`)
*Create a dynamic char * stream data structure.*
- `void vrna_cstr_discard` (`struct vrna_cstr_s *buf`)
*Discard the current content of the dynamic char * stream data structure.*
- `void vrna_cstr_free` (`vrna_cstr_t buf`)
*Free the memory occupied by a dynamic char * stream data structure.*
- `void vrna_cstr_close` (`vrna_cstr_t buf`)
*Free the memory occupied by a dynamic char * stream and close the output stream.*
- `void vrna_cstr_fflush` (`struct vrna_cstr_s *buf`)
*Flush the dynamic char * output stream.*
- `vrna_ostream_t vrna_ostream_init` (`vrna_stream_output_f output`, `void *auxdata`)
Get an initialized ordered output stream.
- `void vrna_ostream_free` (`vrna_ostream_t dat`)
Free an initialized ordered output stream.
- `void vrna_ostream_request` (`vrna_ostream_t dat`, `unsigned int num`)
Request index in ordered output stream.
- `void vrna_ostream_provide` (`vrna_ostream_t dat`, `unsigned int i`, `void *data`)
Provide output stream data for a particular index.

16.90.2 Typedef Documentation

16.90.2.1 vrna_stream_output_f

```
typedef void(* vrna_stream_output_f) (void *auxdata, unsigned int i, void *data)
```

```
#include <ViennaRNA/datastructures/stream_output.h>
```

Ordered stream processing callback.

This callback will be processed in sequential order as soon as sequential data in the output stream becomes available.

Note

The callback must also release the memory occupied by the data passed since the stream will lose any reference to it after the callback has been executed.

Parameters

<i>auxdata</i>	A shared pointer for all calls, as provided by the second argument to <code>vrna_ostream_init()</code>
<i>i</i>	The index number of the data passed to <code>data</code>
<i>data</i>	A block of data ready for processing

16.90.3 Function Documentation

16.90.3.1 vrna_cstr()

```
vrna_cstr_t vrna_cstr (
    size_t size,
    FILE * output )
```

```
#include <ViennaRNA/datastructures/char_stream.h>
```

Create a dynamic char * stream data structure.

See also

[vrna_cstr_free\(\)](#), [vrna_cstr_close\(\)](#), [vrna_cstr_fflush\(\)](#), [vrna_cstr_discard\(\)](#), [vrna_cstr_printf\(\)](#)

Parameters

<i>size</i>	The initial size of the buffer in characters
<i>output</i>	An optional output file stream handle that is used to write the collected data to (defaults to <i>stdout</i> if <i>NULL</i>)

16.90.3.2 vrna_cstr_discard()

```
void vrna_cstr_discard (
    struct vrna_cstr_s * buf )
#include <ViennaRNA/datastructures/char_stream.h>
Discard the current content of the dynamic char * stream data structure.
```

See also

[vrna_cstr_free\(\)](#), [vrna_cstr_close\(\)](#), [vrna_cstr_fflush\(\)](#), [vrna_cstr_printf\(\)](#)

Parameters

<i>buf</i>	The dynamic char * stream data structure to free
------------	--

16.90.3.3 vrna_cstr_free()

```
void vrna_cstr_free (
    vrna_cstr_t buf )
#include <ViennaRNA/datastructures/char_stream.h>
Free the memory occupied by a dynamic char * stream data structure.
This function first flushes any remaining character data within the stream and then free's the memory occupied by the data structure.
```

See also

[vrna_cstr_close\(\)](#), [vrna_cstr_fflush\(\)](#), [vrna_cstr\(\)](#)

Parameters

<i>buf</i>	The dynamic char * stream data structure to free
------------	--

16.90.3.4 vrna_cstr_close()

```
void vrna_cstr_close (
    vrna_cstr_t buf )
#include <ViennaRNA/datastructures/char_stream.h>
Free the memory occupied by a dynamic char * stream and close the output stream.
This function first flushes any remaining character data within the stream then closes the attached output file stream (if any), and finally free's the memory occupied by the data structure.
```

See also

[vrna_cstr_free\(\)](#), [vrna_cstr_fflush\(\)](#), [vrna_cstr\(\)](#)

Parameters

<i>buf</i>	The dynamic char * stream data structure to free
------------	--

16.90.3.5 vrna_cstr_fflush()

```
void vrna_cstr_fflush (
    struct vrna_cstr_s * buf )
#include <ViennaRNA/datastructures/char_stream.h>
```

Flush the dynamic char * output stream.

This function flushes the collected char * stream, either by writing to the attached file handle, or simply by writing to *stdout* if no file handle has been attached upon construction using [vrna_cstr\(\)](#).

Postcondition

The stream buffer is empty after execution of this function

See also

[vrna_cstr\(\)](#), [vrna_cstr_close\(\)](#), [vrna_cstr_free\(\)](#)

Parameters

<i>buf</i>	The dynamic char * stream data structure to flush
------------	---

16.90.3.6 vrna_ostream_init()

```
vrna_ostream_t vrna_ostream_init (
    vrna_stream_output_f output,
    void * auxdata )
#include <ViennaRNA/datastructures/stream_output.h>
```

Get an initialized ordered output stream.

See also

[vrna_ostream_free\(\)](#), [vrna_ostream_request\(\)](#), [vrna_ostream_provide\(\)](#)

Parameters

<i>output</i>	A callback function that processes and releases data in the stream
<i>auxdata</i>	A pointer to auxiliary data passed as first argument to the <i>output</i> callback

Returns

An initialized ordered output stream

16.90.3.7 vrna_ostream_free()

```
void vrna_ostream_free (
```

```

        vrna_ostream_t dat )
#include <ViennaRNA/datastructures/stream_output.h>
Free an initialized ordered output stream.

```

See also

[vrna_ostream_init\(\)](#)

Parameters

<i>dat</i>	The output stream for which occupied memory should be free'd
------------	--

16.90.3.8 vrna_ostream_request()

```

void vrna_ostream_request (
        vrna_ostream_t dat,
        unsigned int num )
#include <ViennaRNA/datastructures/stream_output.h>

```

Request index in ordered output stream.

This function must be called prior to [vrna_ostream_provide\(\)](#) to indicate that data associated with a certain index number is expected to be inserted into the stream in the future.

See also

[vrna_ostream_init\(\)](#), [vrna_ostream_provide\(\)](#), [vrna_ostream_free\(\)](#)

Parameters

<i>dat</i>	The output stream for which the index is requested
<i>num</i>	The index to request data for

16.90.3.9 vrna_ostream_provide()

```

void vrna_ostream_provide (
        vrna_ostream_t dat,
        unsigned int i,
        void * data )
#include <ViennaRNA/datastructures/stream_output.h>

```

Provide output stream data for a particular index.

Precondition

The index data is provided for must have been requested using [vrna_ostream_request\(\)](#) beforehand.

See also

[vrna_ostream_request\(\)](#)

Parameters

<i>dat</i>	The output stream for which data is provided
<i>i</i>	The index of the provided data
<i>data</i>	The data provided

16.91 Deprecated Interface for Global MFE Prediction

16.91.1 Detailed Description

Collaboration diagram for Deprecated Interface for Global MFE Prediction:

Files

- file [alifold.h](#)
Functions for comparative structure prediction using RNA sequence alignments.
- file [cofold.h](#)
MFE implementations for RNA-RNA interaction.
- file [fold.h](#)
MFE calculations for single RNA sequences.

Functions

- float [cofold](#) (const char *sequence, char *structure)
Compute the minimum free energy of two interacting RNA molecules.
- float [cofold_par](#) (const char *string, char *structure, [vrna_param_t](#) *parameters, int is_constrained)
Compute the minimum free energy of two interacting RNA molecules.
- void [free_co_arrays](#) (void)
Free memory occupied by [cofold\(\)](#)
- void [update_cofold_params](#) (void)
Recalculate parameters.
- void [update_cofold_params_par](#) ([vrna_param_t](#) *parameters)
Recalculate parameters.
- void [export_cofold_arrays_gq](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **fc_p, int **ggg_p, int **indx_p, char **ptype_p)
Export the arrays of partition function cofold (with gquadruplex support)
- void [export_cofold_arrays](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **fc_p, int **indx_p, char **ptype_p)
Export the arrays of partition function cofold.
- void [initialize_cofold](#) (int length)
- float [fold_par](#) (const char *sequence, char *structure, [vrna_param_t](#) *parameters, int is_constrained, int is_circular)
Compute minimum free energy and an appropriate secondary structure of an RNA sequence.
- float [fold](#) (const char *sequence, char *structure)
Compute minimum free energy and an appropriate secondary structure of an RNA sequence.
- float [circfold](#) (const char *sequence, char *structure)
Compute minimum free energy and an appropriate secondary structure of a circular RNA sequence.
- void [free_arrays](#) (void)
Free arrays for mfe folding.
- void [update_fold_params](#) (void)
Recalculate energy parameters.
- void [update_fold_params_par](#) ([vrna_param_t](#) *parameters)
Recalculate energy parameters.
- void [export_fold_arrays](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p)
- void [export_fold_arrays_par](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p, [vrna_param_t](#) **P_p)
- void [export_circfold_arrays](#) (int *Fc_p, int *FcH_p, int *FcI_p, int *FcM_p, int **fM2_p, int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p)
- void [export_circfold_arrays_par](#) (int *Fc_p, int *FcH_p, int *FcI_p, int *FcM_p, int **fM2_p, int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p, [vrna_param_t](#) **P_p)

- int [LoopEnergy](#) (int n1, int n2, int type, int type_2, int si1, int sj1, int sp1, int sq1)
- int [HairpinE](#) (int size, int type, int si1, int sj1, const char *string)
- void [initialize_fold](#) (int length)
- float [alifold](#) (const char **strings, char *structure)
Compute MFE and according consensus structure of an alignment of sequences.
- float [circularfold](#) (const char **strings, char *structure)
Compute MFE and according structure of an alignment of sequences assuming the sequences are circular instead of linear.
- void [free_alifold_arrays](#) (void)
Free the memory occupied by MFE alifold functions.

16.91.2 Function Documentation

16.91.2.1 alifold()

```
float alifold (
    const char ** strings,
    char * structure )
#include <ViennaRNA/alifold.h>
```

Compute MFE and according consensus structure of an alignment of sequences.

This function predicts the consensus structure for the aligned 'sequences' and returns the minimum free energy; the mfe structure in bracket notation is returned in 'structure'.

Sufficient space must be allocated for 'structure' before calling [alifold\(\)](#).

Deprecated Usage of this function is discouraged! Use [vrna_alifold\(\)](#), or [vrna_mfe\(\)](#) instead!

See also

[vrna_alifold\(\)](#), [vrna_mfe\(\)](#)

Parameters

<i>strings</i>	A pointer to a NULL terminated array of character arrays
<i>structure</i>	A pointer to a character array that may contain a constraining consensus structure (will be overwritten by a consensus structure that exhibits the MFE)

Returns

The free energy score in kcal/mol

16.91.2.2 cofold()

```
float cofold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/cofold.h>
```

Compute the minimum free energy of two interacting RNA molecules.

The code is analog to the [fold\(\)](#) function. If [cut_point](#) == -1 results should be the same as with [fold\(\)](#).

Deprecated use [vrna_mfe_dimer\(\)](#) instead

Parameters

<i>sequence</i>	The two sequences concatenated
<i>structure</i>	Will hold the barcket dot structure of the dimer molecule

Returns

minimum free energy of the structure

16.91.2.3 cofold_par()

```
float cofold_par (
    const char * string,
    char * structure,
    vrna_param_t * parameters,
    int is_constrained )
```

```
#include <ViennaRNA/cofold.h>
```

Compute the minimum free energy of two interacting RNA molecules.

Deprecated use [vrna_mfe_dimer\(\)](#) instead

16.91.2.4 free_co_arrays()

```
void free_co_arrays (
    void )
```

```
#include <ViennaRNA/cofold.h>
```

Free memory occupied by [cofold\(\)](#)

Deprecated This function will only free memory allocated by a prior call of [cofold\(\)](#) or [cofold_par\(\)](#). See [vrna_mfe_dimer\(\)](#) for how to use the new API

Note

folding matrices now reside in the fold compound, and should be free'd there

See also

[vrna_fc_destroy\(\)](#), [vrna_mfe_dimer\(\)](#)

16.91.2.5 update_cofold_params()

```
void update_cofold_params (
    void )
```

```
#include <ViennaRNA/cofold.h>
```

Recalculate parameters.

Deprecated See [vrna_params_subst\(\)](#) for an alternative using the new API

16.91.2.6 update_cofold_params_par()

```
void update_cofold_params_par (
    vrna_param_t * parameters )
```

```
#include <ViennaRNA/cofold.h>
```

Recalculate parameters.

Deprecated See [vrna_params_subst\(\)](#) for an alternative using the new API

16.91.2.7 export_cofold_arrays_gq()

```
void export_cofold_arrays_gq (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** fc_p,
    int ** ggg_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/cofold.h>
```

Export the arrays of partition function cofold (with gquadruplex support)

Export the cofold arrays for use e.g. in the concentration Computations or suboptimal secondary structure backtracking

Deprecated folding matrices now reside within the fold compound. Thus, this function will only work in conjunction with a prior call to [cofold\(\)](#) or [cofold_par\(\)](#)

See also

[vrna_mfe_dimer\(\)](#) for the new API

Parameters

<i>f5_p</i>	A pointer to the 'f5' array, i.e. array containing best free energy in interval [1..j]
<i>c_p</i>	A pointer to the 'c' array, i.e. array containing best free energy in interval [i..j] given that i pairs with j
<i>fML_p</i>	A pointer to the 'M' array, i.e. array containing best free energy in interval [i..j] for any multiloop segment with at least one stem
<i>fM1_p</i>	A pointer to the 'M1' array, i.e. array containing best free energy in interval [i..j] for multiloop segment with exactly one stem
<i>fc_p</i>	A pointer to the 'fc' array, i.e. array ...
<i>ggg_p</i>	A pointer to the 'ggg' array, i.e. array containing best free energy of a gquadruplex delimited by [i..j]
<i>indx_p</i>	A pointer to the indexing array used for accessing the energy matrices
<i>ptype↔_p</i>	A pointer to the ptype array containing the base pair types for each possibility (i,j)

16.91.2.8 export_cofold_arrays()

```
void export_cofold_arrays (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** fc_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/cofold.h>
```

Export the arrays of partition function cofold.

Export the cofold arrays for use e.g. in the concentration Computations or suboptimal secondary structure backtracking

Deprecated folding matrices now reside within the [vrna_fold_compound_t](#). Thus, this function will only work in conjunction with a prior call to the deprecated functions [cofold\(\)](#) or [cofold_par\(\)](#)

See also

[vrna_mfe_dimer\(\)](#) for the new API

Parameters

<i>f5_p</i>	A pointer to the 'f5' array, i.e. array containing best free energy in interval [1..j]
<i>c_p</i>	A pointer to the 'c' array, i.e. array containing best free energy in interval [i..j] given that i pairs with j
<i>fML_p</i>	A pointer to the 'M' array, i.e. array containing best free energy in interval [i..j] for any multiloop segment with at least one stem
<i>fM1_p</i>	A pointer to the 'M1' array, i.e. array containing best free energy in interval [i..j] for multiloop segment with exactly one stem
<i>fc_p</i>	A pointer to the 'fc' array, i.e. array ...
<i>indx_p</i>	A pointer to the indexing array used for accessing the energy matrices
<i>ptype←_p</i>	A pointer to the ptype array containing the base pair types for each possibility (i,j)

16.91.2.9 initialize_cofold()

```
void initialize_cofold (
    int length )
#include <ViennaRNA/cofold.h>
allocate arrays for folding
```

Deprecated {This function is obsolete and will be removed soon!}

16.91.2.10 fold_par()

```
float fold_par (
    const char * sequence,
    char * structure,
    vrna_param_t * parameters,
    int is_constrained,
    int is_circular )
#include <ViennaRNA/fold.h>
```

Compute minimum free energy and an appropriate secondary structure of an RNA sequence.

The first parameter given, the RNA sequence, must be *uppercase* and should only contain an alphabet Σ that is understood by the RNAlib

(e.g. $\Sigma = \{A, U, C, G\}$)

The second parameter, *structure*, must always point to an allocated block of memory with a size of at least $\text{strlen}(\text{sequence}) + 1$

If the third parameter is NULL, global model detail settings are assumed for the folding recursions. Otherwise, the provided parameters are used.

The fourth parameter indicates whether a secondary structure constraint in enhanced dot-bracket notation is passed through the structure parameter or not. If so, the characters " $| x < >$ " are recognized to mark bases that are paired, unpaired, paired upstream, or downstream, respectively. Matching brackets " $()$ " denote base pairs, dots "." are used for unconstrained bases.

To indicate that the RNA sequence is circular and thus has to be post-processed, set the last parameter to non-zero. After a successful call of [fold_par\(\)](#), a backtracked secondary structure (in dot-bracket notation) that exhibits the minimum of free energy will be written to the memory *structure* is pointing to. The function returns the minimum of free energy for any fold of the sequence given.

Note

OpenMP: Passing NULL to the 'parameters' argument involves access to several global model detail variables and thus is not to be considered threadsafe

Deprecated use [vrna_mfe\(\)](#) instead!

See also

[vrna_mfe\(\)](#), [fold\(\)](#), [circfold\(\)](#), [vrna_md_t](#), [set_energy_model\(\)](#), [get_scaled_parameters\(\)](#)

Parameters

<i>sequence</i>	RNA sequence
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to
<i>parameters</i>	A data structure containing the pre-scaled energy contributions and the model details. (NULL may be passed, see OpenMP notes above)
<i>is_constrained</i>	Switch to indicate that a structure constraint is passed via the structure argument (0==off)
<i>is_circular</i>	Switch to (de-)activate post-processing steps in case RNA sequence is circular (0==off)

Returns

the minimum free energy (MFE) in kcal/mol

16.91.2.11 fold()

```
float fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/fold.h>
```

Compute minimum free energy and an appropriate secondary structure of an RNA sequence.

This function essentially does the same thing as [fold_par\(\)](#). However, it takes its model details, i.e. [temperature](#), [dangles](#), [tetra_loop](#), [noGU](#), [no_closingGU](#), [fold_constrained](#), [noLonelyPairs](#) from the current global settings within the library

Deprecated use [vrna_fold\(\)](#), or [vrna_mfe\(\)](#) instead!

See also

[fold_par\(\)](#), [circfold\(\)](#)

Parameters

<i>sequence</i>	RNA sequence
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.91.2.12 circfold()

```
float circfold (
```

```

    const char * sequence,
    char * structure )
#include <ViennaRNA/fold.h>

```

Compute minimum free energy and an appropriate secondary structure of a circular RNA sequence.

This function essentially does the same thing as [fold_par\(\)](#). However, it takes its model details, i.e. [temperature](#), [dangles](#), [tetra_loop](#), [noGU](#), [no_closingGU](#), [fold_constrained](#), [noLonelyPairs](#) from the current global settings within the library

Deprecated Use [vrna_circfold\(\)](#), or [vrna_mfe\(\)](#) instead!

See also

[fold_par\(\)](#), [circfold\(\)](#)

Parameters

<i>sequence</i>	RNA sequence
<i>structure</i>	A pointer to the character array where the secondary structure in dot-bracket notation will be written to

Returns

the minimum free energy (MFE) in kcal/mol

16.91.2.13 free_arrays()

```

void free_arrays (
    void )
#include <ViennaRNA/fold.h>

```

Free arrays for mfe folding.

Deprecated See [vrna_fold\(\)](#), [vrna_circfold\(\)](#), or [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

16.91.2.14 update_fold_params()

```

void update_fold_params (
    void )
#include <ViennaRNA/fold.h>

```

Recalculate energy parameters.

Deprecated For non-default model settings use the new API with [vrna_params_subst\(\)](#) and [vrna_mfe\(\)](#) instead!

16.91.2.15 update_fold_params_par()

```

void update_fold_params_par (
    vrna_param_t * parameters )
#include <ViennaRNA/fold.h>

```

Recalculate energy parameters.

Deprecated For non-default model settings use the new API with [vrna_params_subst\(\)](#) and [vrna_mfe\(\)](#) instead!

16.91.2.16 export_fold_arrays()

```
void export_fold_arrays (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/fold.h>
```

Deprecated See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

16.91.2.17 export_fold_arrays_par()

```
void export_fold_arrays_par (
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** indx_p,
    char ** ptype_p,
    vrna_param_t ** P_p )
#include <ViennaRNA/fold.h>
```

Deprecated See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

16.91.2.18 export_circfold_arrays()

```
void export_circfold_arrays (
    int * Fc_p,
    int * FcH_p,
    int * FcI_p,
    int * FcM_p,
    int ** fM2_p,
    int ** f5_p,
    int ** c_p,
    int ** fML_p,
    int ** fM1_p,
    int ** indx_p,
    char ** ptype_p )
#include <ViennaRNA/fold.h>
```

Deprecated See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

16.91.2.19 export_circfold_arrays_par()

```
void export_circfold_arrays_par (
    int * Fc_p,
    int * FcH_p,
    int * FcI_p,
    int * FcM_p,
    int ** fM2_p,
    int ** f5_p,
    int ** c_p,
```

```

    int ** fML_p,
    int ** fMl_p,
    int ** indx_p,
    char ** ptype_p,
    vrna_param_t ** P_p )
#include <ViennaRNA/fold.h>

```

Deprecated See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

16.91.2.20 LoopEnergy()

```

int LoopEnergy (
    int n1,
    int n2,
    int type,
    int type_2,
    int sil,
    int sj1,
    int sp1,
    int sq1 )
#include <ViennaRNA/fold.h>

```

Deprecated {This function is deprecated and will be removed soon. Use [E_IntLoop\(\)](#) instead!}

16.91.2.21 HairpinE()

```

int HairpinE (
    int size,
    int type,
    int sil,
    int sj1,
    const char * string )
#include <ViennaRNA/fold.h>

```

Deprecated {This function is deprecated and will be removed soon. Use [E_Hairpin\(\)](#) instead!}

16.91.2.22 initialize_fold()

```

void initialize_fold (
    int length )
#include <ViennaRNA/fold.h>
Allocate arrays for folding

```

Deprecated See [vrna_mfe\(\)](#) and [vrna_fold_compound_t](#) for the usage of the new API!

16.91.2.23 circalifold()

```

float circalifold (
    const char ** strings,
    char * structure )
#include <ViennaRNA/alifold.h>

```

Compute MFE and according structure of an alignment of sequences assuming the sequences are circular instead of linear.

Deprecated Usage of this function is discouraged! Use `vrna_alicircfold()`, and `vrna_mfe()` instead!

See also

`vrna_alicircfold()`, `vrna_alifold()`, `vrna_mfe()`

Parameters

<i>strings</i>	A pointer to a NULL terminated array of character arrays
<i>structure</i>	A pointer to a character array that may contain a constraining consensus structure (will be overwritten by a consensus structure that exhibits the MFE)

Returns

The free energy score in kcal/mol

16.91.2.24 free_alifold_arrays()

```
void free_alifold_arrays (
    void )
#include <ViennaRNA/alifold.h>
Free the memory occupied by MFE alifold functions.
```

Deprecated Usage of this function is discouraged! It only affects memory being free'd that was allocated by an old API function before. Release of memory occupied by the newly introduced `vrna_fold_compound_t` is handled by `vrna_fold_compound_free()`

See also

`vrna_fold_compound_free()`

16.92 Deprecated Interface for Local (Sliding Window) MFE Prediction

16.92.1 Detailed Description

Collaboration diagram for Deprecated Interface for Local (Sliding Window) MFE Prediction:

Files

- file `Lfold.h`
Functions for locally optimal MFE structure prediction.

Functions

- float `Lfold` (const char *string, const char *structure, int maxdist)
The local analog to `fold()`.
- float `Lfoldz` (const char *string, const char *structure, int maxdist, int zsc, double min_z)

16.92.2 Function Documentation

16.92.2.1 Lfold()

```
float Lfold (
    const char * string,
    const char * structure,
    int maxdist )
#include <ViennaRNA/Lfold.h>
```

The local analog to `fold()`.

Computes the minimum free energy structure including only base pairs with a span smaller than 'maxdist'

Deprecated Use `vrna_mfe_window()` instead!

16.92.2.2 Lfoldz()

```
float Lfoldz (
    const char * string,
    const char * structure,
    int maxdist,
    int zsc,
    double min_z )
#include <ViennaRNA/Lfold.h>
```

Deprecated Use `vrna_mfe_window_zscore()` instead!

16.93 Deprecated Interface for Global Partition Function Computation

16.93.1 Detailed Description

Collaboration diagram for Deprecated Interface for Global Partition Function Computation:

Files

- file `part_func_co.h`
Partition function for two RNA sequences.

Functions

- float `pf_fold_par` (const char *sequence, char *structure, `vrna_exp_param_t` *parameters, int calculate_↔
bppm, int is_constrained, int is_circular)
Compute the partition function Q for a given RNA sequence.
- float `pf_fold` (const char *sequence, char *structure)
Compute the partition function Q of an RNA sequence.
- float `pf_circ_fold` (const char *sequence, char *structure)
Compute the partition function of a circular RNA sequence.
- void `free_pf_arrays` (void)
Free arrays for the partition function recursions.
- void `update_pf_params` (int length)
Recalculate energy parameters.
- void `update_pf_params_par` (int length, `vrna_exp_param_t` *parameters)
Recalculate energy parameters.
- `FLT_OR_DBL` * `export_bppm` (void)
Get a pointer to the base pair probability array.
- int `get_pf_arrays` (short **S_p, short **S1_p, char **ptype_p, `FLT_OR_DBL` **qb_p, `FLT_OR_DBL` **qm↔
_p, `FLT_OR_DBL` **q1k_p, `FLT_OR_DBL` **qln_p)

- Get the pointers to (almost) all relevant computation arrays used in partition function computation.*

 - double `get_subseq_F` (int i, int j)
- Get the free energy of a subsequence from the q[] array.*

 - double `mean_bp_distance` (int length)
- Get the mean base pair distance of the last partition function computation.*

 - double `mean_bp_distance_pr` (int length, FLT_OR_DBL *pr)
- Get the mean base pair distance in the thermodynamic ensemble.*

 - `vrna_ep_t` * `stackProb` (double cutoff)
- Get the probability of stacks.*

 - void `init_pf_fold` (int length)
- Allocate space for `pf_fold()`*

 - `vrna_dimer_pf_t` `co_pf_fold` (char *sequence, char *structure)
- Calculate partition function and base pair probabilities.*

 - `vrna_dimer_pf_t` `co_pf_fold_par` (char *sequence, char *structure, `vrna_exp_param_t` *parameters, int calculate_bppm, int is_constrained)
- Calculate partition function and base pair probabilities.*

 - void `compute_probabilities` (double FAB, double FEA, double FEB, `vrna_ep_t` *prAB, `vrna_ep_t` *prA, `vrna_ep_t` *prB, int Alength)
- Compute Boltzmann probabilities of dimerization without homodimers.*

 - void `init_co_pf_fold` (int length)
- FLT_OR_DBL * `export_co_bppm` (void)
- Get a pointer to the base pair probability array.*

 - void `free_co_pf_arrays` (void)
- Free the memory occupied by `co_pf_fold()`*

 - void `update_co_pf_params` (int length)
- Recalculate energy parameters.*

 - void `update_co_pf_params_par` (int length, `vrna_exp_param_t` *parameters)
- Recalculate energy parameters.*

 - void `assign_plist_from_db` (`vrna_ep_t` **pl, const char *struc, float pr)
- Create a `vrna_ep_t` from a dot-bracket string.*

 - void `assign_plist_from_pr` (`vrna_ep_t` **pl, FLT_OR_DBL *probs, int length, double cutoff)
- Create a `vrna_ep_t` from a probability matrix.*
- float `alipf_fold_par` (const char **sequences, char *structure, `vrna_ep_t` **pl, `vrna_exp_param_t` *parameters, int calculate_bppm, int is_constrained, int is_circular)
- float `alipf_fold` (const char **sequences, char *structure, `vrna_ep_t` **pl)
- The partition function version of `alifold()` works in analogy to `pf_fold()`. Pair probabilities and information about sequence covariations are returned via the 'pi' variable as a list of `vrna_pinfo_t` structs. The list is terminated by the first entry with `pi.i = 0`.*

 - float `alipf_circ_fold` (const char **sequences, char *structure, `vrna_ep_t` **pl)
- FLT_OR_DBL * `export_ali_bppm` (void)
- Get a pointer to the base pair probability array.*

 - void `free_alipf_arrays` (void)
- Free the memory occupied by folding matrices allocated by `alipf_fold`, `alipf_circ_fold`, etc.*
- char * `alipbacktrack` (double *prob)
- Sample a consensus secondary structure from the Boltzmann ensemble according its probability.*

 - int `get_alipf_arrays` (short ***S_p, short ***S5_p, short ***S3_p, unsigned short ***a2s_p, char ***Ss←_p, FLT_OR_DBL **qb_p, FLT_OR_DBL **qm_p, FLT_OR_DBL **q1k_p, FLT_OR_DBL **qln_p, short **pscore)
- Get pointers to (almost) all relevant arrays used in `alifold`'s partition function computation.*

16.93.2 Function Documentation

16.93.2.1 alipf_fold_par()

```
float alipf_fold_par (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl,
    vrna_exp_param_t * parameters,
    int calculate_bppm,
    int is_constrained,
    int is_circular )
#include <ViennaRNA/alifold.h>
```

Deprecated Use `vrna_pf()` instead

Parameters

<i>sequences</i>	
<i>structure</i>	
<i>pl</i>	
<i>parameters</i>	
<i>calculate_bppm</i>	
<i>is_constrained</i>	
<i>is_circular</i>	

Returns

16.93.2.2 pf_fold_par()

```
float pf_fold_par (
    const char * sequence,
    char * structure,
    vrna_exp_param_t * parameters,
    int calculate_bppm,
    int is_constrained,
    int is_circular )
#include <ViennaRNA/part_func.h>
```

Compute the partition function Q for a given RNA sequence.

If *structure* is not a NULL pointer on input, it contains on return a string consisting of the letters " . , | { } () " denoting bases that are essentially unpaired, weakly paired, strongly paired without preference, weakly upstream (downstream) paired, or strongly up- (down-)stream paired bases, respectively. If `fold_constrained` is not 0, the *structure* string is interpreted on input as a list of constraints for the folding. The character "x" marks bases that must be unpaired, matching brackets " () " denote base pairs, all other characters are ignored. Any pairs conflicting with the constraint will be forbidden. This is usually sufficient to ensure the constraints are honored. If the parameter `calculate_bppm` is set to 0 base pairing probabilities will not be computed (saving CPU time), otherwise after calculations took place `pr` will contain the probability that bases *i* and *j* pair.

Deprecated Use `vrna_pf()` instead

Note

The global array `pr` is deprecated and the user who wants the calculated base pair probabilities for further computations is advised to use the function `export_bppm()`

Postcondition

After successful run the hidden folding matrices are filled with the appropriate Boltzmann factors. Depending on whether the global variable `do_backtrack` was set the base pair probabilities are already computed and may be accessed for further usage via the `export_bppm()` function. A call of `free_pf_arrays()` will free all memory allocated by this function. Successive calls will first free previously allocated memory before starting the computation.

See also

`vrna_pf()`, `bppm_to_structure()`, `export_bppm()`, `vrna_exp_params()`, `free_pf_arrays()`

Parameters

in	<i>sequence</i>	The RNA sequence input
in, out	<i>structure</i>	A pointer to a char array where a base pair probability information can be stored in a pseudo-dot-bracket notation (may be NULL, too)
in	<i>parameters</i>	Data structure containing the precalculated Boltzmann factors
in	<i>calculate_bppm</i>	Switch to Base pair probability calculations on/off (0==off)
in	<i>is_constrained</i>	Switch to indicate that a structure constraint is passed via the structure argument (0==off)
in	<i>is_circular</i>	Switch to (de-)activate postprocessing steps in case RNA sequence is circular (0==off)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.93.2.3 pf_fold()

```
float pf_fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Compute the partition function Q of an RNA sequence.

If *structure* is not a NULL pointer on input, it contains on return a string consisting of the letters ". , | { } () " denoting bases that are essentially unpaired, weakly paired, strongly paired without preference, weakly upstream (downstream) paired, or strongly up- (down-)stream paired bases, respectively. If `fold_constrained` is not 0, the *structure* string is interpreted on input as a list of constraints for the folding. The character "x" marks bases that must be unpaired, matching brackets " () " denote base pairs, all other characters are ignored. Any pairs conflicting with the constraint will be forbidden. This is usually sufficient to ensure the constraints are honored. If `do_backtrack` has been set to 0 base pairing probabilities will not be computed (saving CPU time), otherwise `pr` will contain the probability that bases i and j pair.

Note

The global array `pr` is deprecated and the user who wants the calculated base pair probabilities for further computations is advised to use the function `export_bppm()`.

OpenMP: This function is not entirely threadsafe. While the recursions are working on their own copies of data the model details for the recursions are determined from the global settings just before entering the recursions. Consider using `pf_fold_par()` for a really threadsafe implementation.

Precondition

This function takes its model details from the global variables provided in *RNAlib*

Postcondition

After successful run the hidden folding matrices are filled with the appropriate Boltzmann factors. Depending on whether the global variable `do_backtrack` was set the base pair probabilities are already computed and may be accessed for further usage via the `export_bppm()` function. A call of `free_pf_arrays()` will free all memory allocated by this function. Successive calls will first free previously allocated memory before starting the computation.

See also

`pf_fold_par()`, `pf_circ_fold()`, `bppm_to_structure()`, `export_bppm()`

Parameters

<i>sequence</i>	The RNA sequence input
<i>structure</i>	A pointer to a char array where a base pair probability information can be stored in a pseudo-dot-bracket notation (may be NULL, too)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.93.2.4 pf_circ_fold()

```
float pf_circ_fold (
    const char * sequence,
    char * structure )
#include <ViennaRNA/part_func.h>
```

Compute the partition function of a circular RNA sequence.

Note

The global array `pr` is deprecated and the user who wants the calculated base pair probabilities for further computations is advised to use the function `export_bppm()`.

OpenMP: This function is not entirely threadsafe. While the recursions are working on their own copies of data the model details for the recursions are determined from the global settings just before entering the recursions. Consider using `pf_fold_par()` for a really threadsafe implementation.

Precondition

This function takes its model details from the global variables provided in *RNAlib*

Postcondition

After successful run the hidden folding matrices are filled with the appropriate Boltzmann factors. Depending on whether the global variable `do_backtrack` was set the base pair probabilities are already computed and may be accessed for further usage via the `export_bppm()` function. A call of `free_pf_arrays()` will free all memory allocated by this function. Successive calls will first free previously allocated memory before starting the computation.

See also

`vrna_pf()`

Deprecated Use `vrna_pf()` instead!

Parameters

<code>in</code>	<i>sequence</i>	The RNA sequence input
<code>in, out</code>	<i>structure</i>	A pointer to a char array where a base pair probability information can be stored in a pseudo-dot-bracket notation (may be NULL, too)

Returns

The ensemble free energy $G = -RT \cdot \log(Q)$ in kcal/mol

16.93.2.5 free_pf_arrays()

```
void free_pf_arrays (
    void )
#include <ViennaRNA/part_func.h>
```

Free arrays for the partition function recursions.

Call this function if you want to free all allocated memory associated with the partition function forward recursion.

Note

Successive calls of [pf_fold\(\)](#), [pf_circ_fold\(\)](#) already check if they should free any memory from a previous run.

OpenMP notice:

This function should be called before leaving a thread in order to avoid leaking memory

Deprecated See [vrna_fold_compound_t](#) and its related functions for how to free memory occupied by the dynamic programming matrices

Postcondition

All memory allocated by [pf_fold_par\(\)](#), [pf_fold\(\)](#) or [pf_circ_fold\(\)](#) will be free'd

See also

[pf_fold_par\(\)](#), [pf_fold\(\)](#), [pf_circ_fold\(\)](#)

16.93.2.6 update_pf_params()

```
void update_pf_params (
    int length )
#include <ViennaRNA/part_func.h>
```

Recalculate energy parameters.

Call this function to recalculate the pair matrix and energy parameters after a change in folding parameters like [temperature](#)

Deprecated Use [vrna_exp_params_subst\(\)](#) instead

16.93.2.7 update_pf_params_par()

```
void update_pf_params_par (
    int length,
    vrna_exp_param_t * parameters )
#include <ViennaRNA/part_func.h>
```

Recalculate energy parameters.

Deprecated Use [vrna_exp_params_subst\(\)](#) instead

16.93.2.8 export_bppm()

```
FLT_OR_DBL * export_bppm (
    void )
#include <ViennaRNA/part_func.h>
Get a pointer to the base pair probability array.
Accessing the base pair probabilities for a pair (i,j) is achieved by
FLT_OR_DBL *pr = export_bppm();
pr_ij = pr[iindx[i]-j];
```

Precondition

Call [pf_fold_par\(\)](#), [pf_fold\(\)](#) or [pf_circ_fold\(\)](#) first to fill the base pair probability array

See also

[pf_fold\(\)](#), [pf_circ_fold\(\)](#), [vrna_idx_row_wise\(\)](#)

Returns

A pointer to the base pair probability array

16.93.2.9 get_pf_arrays()

```
int get_pf_arrays (
    short ** S_p,
    short ** S1_p,
    char ** ptype_p,
    FLT_OR_DBL ** qb_p,
    FLT_OR_DBL ** qm_p,
    FLT_OR_DBL ** q1k_p,
    FLT_OR_DBL ** qln_p )
#include <ViennaRNA/part_func.h>
```

Get the pointers to (almost) all relevant computation arrays used in partition function computation.

Precondition

In order to assign meaningful pointers, you have to call [pf_fold_par\(\)](#) or [pf_fold\(\)](#) first!

See also

[pf_fold_par\(\)](#), [pf_fold\(\)](#), [pf_circ_fold\(\)](#)

Parameters

out	<i>S_p</i>	A pointer to the 'S' array (integer representation of nucleotides)
out	<i>S1_p</i>	A pointer to the 'S1' array (2nd integer representation of nucleotides)
out	<i>ptype</i> _{← _p}	A pointer to the pair type matrix
out	<i>qb_p</i>	A pointer to the Q^B matrix
out	<i>qm_p</i>	A pointer to the Q^M matrix
out	<i>q1k_p</i>	A pointer to the 5' slice of the Q matrix ($q1k(k) = Q(1, k)$)
out	<i>qln_p</i>	A pointer to the 3' slice of the Q matrix ($qln(l) = Q(l, n)$)

Returns

Non Zero if everything went fine, 0 otherwise

16.93.2.10 get_subseq_F()

```
double get_subseq_F (
    int i,
    int j )
#include <ViennaRNA/part_func.h>
Get the free energy of a subsequence from the q[] array.
```

16.93.2.11 mean_bp_distance()

```
double mean_bp_distance (
    int length )
#include <ViennaRNA/part_func.h>
Get the mean base pair distance of the last partition function computation.
```

Deprecated Use [vrna_mean_bp_distance\(\)](#) or [vrna_mean_bp_distance_pr\(\)](#) instead!

See also

[vrna_mean_bp_distance\(\)](#), [vrna_mean_bp_distance_pr\(\)](#)

Parameters

<i>length</i>	
---------------	--

Returns

mean base pair distance in thermodynamic ensemble

16.93.2.12 mean_bp_distance_pr()

```
double mean_bp_distance_pr (
    int length,
    FLT_OR_DBL * pr )
#include <ViennaRNA/part_func.h>
Get the mean base pair distance in the thermodynamic ensemble.
This is a threadsafe implementation of mean\_bp\_dist\(\) !
 $\langle d \rangle = \sum_{a,b} p_a p_b d(S_a, S_b)$ 
this can be computed from the pair probs  $p_{ij}$  as
 $\langle d \rangle = \sum_{ij} p_{ij} (1 - p_{ij})$ 
```

Deprecated Use [vrna_mean_bp_distance\(\)](#) or [vrna_mean_bp_distance_pr\(\)](#) instead!

Parameters

<i>length</i>	The length of the sequence
<i>pr</i>	The matrix containing the base pair probabilities

Returns

The mean pair distance of the structure ensemble

16.93.2.13 stackProb()

```
vrna_ep_t * stackProb (
    double cutoff )
#include <ViennaRNA/part_func.h>
Get the probability of stacks.
```

Deprecated Use `vrna_stack_prob()` instead!

16.93.2.14 init_pf_fold()

```
void init_pf_fold (
    int length )
#include <ViennaRNA/part_func.h>
Allocate space for pf_fold()
```

Deprecated This function is obsolete and will be removed soon!

16.93.2.15 co_pf_fold()

```
vrna_dimer_pf_t co_pf_fold (
    char * sequence,
    char * structure )
#include <ViennaRNA/part_func_co.h>
Calculate partition function and base pair probabilities.
This is the cofold partition function folding. The second molecule starts at the cut_point nucleotide.
```

Note

OpenMP: Since this function relies on the global parameters `do_backtrack`, `dangles`, `temperature` and `pf_scale` it is not threadsafe according to concurrent changes in these variables! Use `co_pf_fold_par()` instead to circumvent this issue.

Deprecated {Use `vrna_pf_dimer()` instead!}

Parameters

<i>sequence</i>	Concatenated RNA sequences
<i>structure</i>	Will hold the structure or constraints

Returns

`vrna_dimer_pf_t` structure containing a set of energies needed for concentration computations.

16.93.2.16 co_pf_fold_par()

```
vrna_dimer_pf_t co_pf_fold_par (
    char * sequence,
    char * structure,
```

```

    vrna_exp_param_t * parameters,
    int calculate_bppm,
    int is_constrained )
#include <ViennaRNA/part_func_co.h>
Calculate partition function and base pair probabilities.
This is the cofold partition function folding. The second molecule starts at the cut\_point nucleotide.

```

Deprecated Use [vrna_pf_dimer\(\)](#) instead!

See also

[get_boltzmann_factors\(\)](#), [co_pf_fold\(\)](#)

Parameters

<i>sequence</i>	Concatenated RNA sequences
<i>structure</i>	Pointer to the structure constraint
<i>parameters</i>	Data structure containing the precalculated Boltzmann factors
<i>calculate_bppm</i>	Switch to turn Base pair probability calculations on/off (0==off)
<i>is_constrained</i>	Switch to indicate that a structure constraint is passed via the structure argument (0==off)

Returns

[vrna_dimer_pf_t](#) structure containing a set of energies needed for concentration computations.

16.93.2.17 compute_probabilities()

```

void compute_probabilities (
    double FAB,
    double FEA,
    double FEB,
    vrna_ep_t * prAB,
    vrna_ep_t * prA,
    vrna_ep_t * prB,
    int Alength )
#include <ViennaRNA/part_func_co.h>

```

Compute Boltzmann probabilities of dimerization without homodimers.

Given the pair probabilities and free energies (in the null model) for a dimer AB and the two constituent monomers A and B, compute the conditional pair probabilities given that a dimer AB actually forms. Null model pair probabilities are given as a list as produced by [assign_plist_from_pr\(\)](#), the dimer probabilities 'prAB' are modified in place.

Deprecated { Use [vrna_pf_dimer_probs\(\)](#) instead! }

Parameters

<i>FAB</i>	free energy of dimer AB
<i>FEA</i>	free energy of monomer A
<i>FEB</i>	free energy of monomer B
<i>prAB</i>	pair probabilities for dimer
<i>prA</i>	pair probabilities monomer
<i>prB</i>	pair probabilities monomer
<i>Alength</i>	Length of molecule A

16.93.2.18 init_co_pf_fold()

```
void init_co_pf_fold (
    int length )
#include <ViennaRNA/part_func_co.h>
DO NOT USE THIS FUNCTION ANYMORE
```

Deprecated { This function is deprecated and will be removed soon!}

16.93.2.19 export_co_bppm()

```
FLT_OR_DBL * export_co_bppm (
    void )
#include <ViennaRNA/part_func_co.h>
Get a pointer to the base pair probability array.
Accessing the base pair probabilities for a pair (i,j) is achieved by

FLT_OR_DBL *pr = export_bppm(); pr_ij = pr[iindx[i]-j];
```

Deprecated This function is deprecated and will be removed soon! The base pair probability array is available through the `vrna_fold_compound_t` data structure, and its associated `vrna_mx_pf_t` member.

See also

[vrna_idx_row_wise\(\)](#)

Returns

A pointer to the base pair probability array

16.93.2.20 free_co_pf_arrays()

```
void free_co_pf_arrays (
    void )
#include <ViennaRNA/part_func_co.h>
Free the memory occupied by co\_pf\_fold\(\)
```

Deprecated This function will be removed for the new API soon! See [vrna_pf_dimer\(\)](#), [vrna_fold_compound\(\)](#), and [vrna_fold_compound_free\(\)](#) for an alternative

16.93.2.21 update_co_pf_params()

```
void update_co_pf_params (
    int length )
#include <ViennaRNA/part_func_co.h>
Recalculate energy parameters.
This function recalculates all energy parameters given the current model settings.
```

Deprecated Use [vrna_exp_params_subst\(\)](#) instead!

Parameters

<i>length</i>	Length of the current RNA sequence
---------------	------------------------------------

16.93.2.22 `update_co_pf_params_par()`

```
void update_co_pf_params_par (
    int length,
    vrna_exp_param_t * parameters )
#include <ViennaRNA/part_func_co.h>
```

Recalculate energy parameters.

This function recalculates all energy parameters given the current model settings. It's second argument can either be NULL or a data structure containing the precomputed Boltzmann factors. In the first scenario, the necessary data structure will be created automatically according to the current global model settings, i.e. this mode might not be threadsafe. However, if the provided data structure is not NULL, threadsafety for the model parameters [dangles](#), [pf_scale](#) and [temperature](#) is regained, since their values are taken from this data structure during subsequent calculations.

Deprecated Use `vrna_exp_params_subst()` instead!

Parameters

<i>length</i>	Length of the current RNA sequence
<i>parameters</i>	data structure containing the precomputed Boltzmann factors

16.93.2.23 `assign_plist_from_db()`

```
void assign_plist_from_db (
    vrna_ep_t ** pl,
    const char * struc,
    float pr )
#include <ViennaRNA/utils/structures.h>
```

Create a `vrna_ep_t` from a dot-bracket string.

The dot-bracket string is parsed and for each base pair an entry in the plist is created. The probability of each pair in the list is set by a function parameter.

The end of the plist is marked by sequence positions *i* as well as *j* equal to 0. This condition should be used to stop looping over its entries

Deprecated Use `vrna_plist()` instead

Parameters

<i>pl</i>	A pointer to the <code>vrna_ep_t</code> that is to be created
<i>struc</i>	The secondary structure in dot-bracket notation
<i>pr</i>	The probability for each base pair

16.93.2.24 `assign_plist_from_pr()`

```
void assign_plist_from_pr (
    vrna_ep_t ** pl,
    FLT_OR_DBL * probs,
    int length,
    double cutoff )
#include <ViennaRNA/utils/structures.h>
```

Create a `vrna_ep_t` from a probability matrix.

The probability matrix given is parsed and all pair probabilities above the given threshold are used to create an entry in the plist

The end of the plist is marked by sequence positions `i` as well as `j` equal to 0. This condition should be used to stop looping over its entries

Note

This function is threadsafe

Deprecated Use `vrna_plist_from_probs()` instead!

Parameters

out	<i>pl</i>	A pointer to the <code>vrna_ep_t</code> that is to be created
in	<i>probs</i>	The probability matrix used for creating the plist
in	<i>length</i>	The length of the RNA sequence
in	<i>cutoff</i>	The cutoff value

16.93.2.25 alipf_fold()

```
float alipf_fold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/alifold.h>
```

The partition function version of `alifold()` works in analogy to `pf_fold()`. Pair probabilities and information about sequence covariations are returned via the 'pi' variable as a list of `vrna_pinfo_t` structs. The list is terminated by the first entry with `pi.i = 0`.

Deprecated Use `vrna_pf()` instead

Parameters

<i>sequences</i>	
<i>structure</i>	
<i>pl</i>	

Returns

16.93.2.26 alipf_circ_fold()

```
float alipf_circ_fold (
    const char ** sequences,
    char * structure,
    vrna_ep_t ** pl )
#include <ViennaRNA/alifold.h>
```

Deprecated Use `vrna_pf()` instead

Parameters

<i>sequences</i>	
<i>structure</i>	
<i>pl</i>	

Returns

16.93.2.27 export_ali_bppm()

```
FLT_OR_DBL * export_ali_bppm (
    void )
#include <ViennaRNA/alifold.h>
Get a pointer to the base pair probability array.
Accessing the base pair probabilities for a pair (i,j) is achieved by
FLT_OR_DBL *pr = export_bppm(); pr_ij = pr[iindx[i]-j];
```

Deprecated Usage of this function is discouraged! The new [vrna_fold_compound_t](#) allows direct access to the folding matrices, including the pair probabilities! The pair probability array returned here reflects the one of the latest call to [vrna_pf\(\)](#), or any of the old API calls for consensus structure partition function folding.

See also

[vrna_fold_compound_t](#), [vrna_fold_compound_comparative\(\)](#), and [vrna_pf\(\)](#)

Returns

A pointer to the base pair probability array

16.93.2.28 free_alipf_arrays()

```
void free_alipf_arrays (
    void )
#include <ViennaRNA/alifold.h>
Free the memory occupied by folding matrices allocated by alipf_fold, alipf_circ_fold, etc.
```

Deprecated Usage of this function is discouraged! This function only free's memory allocated by old API function calls. Memory allocated by any of the new API calls (starting with [vrna_](#)) will be not affected!

See also

[vrna_fold_compound_t](#), [vrna_vrna_fold_compound_free\(\)](#)

16.93.2.29 alipbacktrack()

```
char * alipbacktrack (
    double * prob )
#include <ViennaRNA/alifold.h>
Sample a consensus secondary structure from the Boltzmann ensemble according its probability.
```

Deprecated Use [vrna_pbacktrack\(\)](#) instead!

Parameters

<i>prob</i>	to be described (berni)
-------------	-------------------------

Returns

A sampled consensus secondary structure in dot-bracket notation

16.93.2.30 get_alipf_arrays()

```
int get_alipf_arrays (
    short *** S_p,
    short *** S5_p,
    short *** S3_p,
    unsigned short *** a2s_p,
    char *** Ss_p,
    FLT_OR_DBL ** qb_p,
    FLT_OR_DBL ** qm_p,
    FLT_OR_DBL ** q1k_p,
    FLT_OR_DBL ** qln_p,
    short ** pscore )
```

```
#include <ViennaRNA/alifold.h>
```

Get pointers to (almost) all relevant arrays used in alifold's partition function computation.

Note

To obtain meaningful pointers, call `alipf_fold` first!

See also

`pf_alifold()`, [alipf_circ_fold\(\)](#)

Deprecated It is discouraged to use this function! The new [vrna_fold_compound_t](#) allows direct access to all necessary consensus structure prediction related variables!

See also

[vrna_fold_compound_t](#), [vrna_fold_compound_comparative\(\)](#), [vrna_pf\(\)](#)

Parameters

<i>S_p</i>	A pointer to the 'S' array (integer representation of nucleotides)
<i>S5_p</i>	A pointer to the 'S5' array
<i>S3_p</i>	A pointer to the 'S3' array
<i>a2s↔ _p</i>	A pointer to the alignment-column to sequence position mapping array
<i>Ss_p</i>	A pointer to the 'Ss' array
<i>qb_p</i>	A pointer to the Q^B matrix
<i>qm_p</i>	A pointer to the Q^M matrix
<i>q1k↔ _p</i>	A pointer to the 5' slice of the Q matrix ($q1k(k) = Q(1, k)$)
<i>qln_p</i>	A pointer to the 3' slice of the Q matrix ($qln(l) = Q(l, n)$)
<i>pscore</i>	A pointer to the start of a pscore list

Returns

Non Zero if everything went fine, 0 otherwise

16.94 Deprecated Interface for Local (Sliding Window) Partition Function Computation

16.94.1 Detailed Description

Collaboration diagram for Deprecated Interface for Local (Sliding Window) Partition Function Computation:

Files

- file [LPfold.h](#)

Partition function and equilibrium probability implementation for the sliding window algorithm.

Functions

- void [update_pf_paramsLP](#) (int length)
- [vrna_ep_t](#) * [pfl_fold](#) (char *sequence, int winSize, int pairSize, float cutoffb, double **pU, [vrna_ep_t](#) **dpp2, FILE *pUfp, FILE *spup)
Compute partition functions for locally stable secondary structures.
- [vrna_ep_t](#) * [pfl_fold_par](#) (char *sequence, int winSize, int pairSize, float cutoffb, double **pU, [vrna_ep_t](#) **dpp2, FILE *pUfp, FILE *spup, [vrna_exp_param_t](#) *parameters)
Compute partition functions for locally stable secondary structures.
- void [putoutpU_prob](#) (double **pU, int length, int ulength, FILE *fp, int energies)
Writes the unpaired probabilities (pU) or opening energies into a file.
- void [putoutpU_prob_bin](#) (double **pU, int length, int ulength, FILE *fp, int energies)
Writes the unpaired probabilities (pU) or opening energies into a binary file.

16.94.2 Function Documentation

16.94.2.1 update_pf_paramsLP()

```
void update_pf_paramsLP (
    int length )
#include <ViennaRNA/LPfold.h>
```

Parameters

<i>length</i>	
---------------	--

16.94.2.2 pfl_fold()

```
vrna\_ep\_t * pfl_fold (
    char * sequence,
    int winSize,
    int pairSize,
    float cutoffb,
    double ** pU,
    vrna\_ep\_t ** dpp2,
    FILE * pUfp,
    FILE * spup )
```

```
#include <ViennaRNA/LPfold.h>
```

Compute partition functions for locally stable secondary structures.

`pfl_fold` computes partition functions for every window of size '`winSize`' possible in a RNA molecule, allowing only pairs with a span smaller than '`pairSize`'. It returns the mean pair probabilities averaged over all windows containing the pair in '`pl`'. '`winSize`' should always be \geq '`pairSize`'. Note that in contrast to `Lfold()`, bases outside of the window do not influence the structure at all. Only probabilities higher than '`cutoffb`' are kept.

If '`pU`' is supplied (i.e. is not the NULL pointer), `pfl_fold()` will also compute the mean probability that regions of length '`u`' and smaller are unpaired. The parameter '`u`' is supplied in '`pup[0][0]`'. On return the '`pup`' array will contain these probabilities, with the entry on '`pup[x][y]`' containing the mean probability that `x` and the `y-1` preceding bases are unpaired. The '`pU`' array needs to be large enough to hold `n+1 float*` entries, where `n` is the sequence length.

If an array `dpp2` is supplied, the probability of base pair (`i,j`) given that there already exists a base pair (`i+1,j-1`) is also computed and saved in this array. If `pUfp` is given (i.e. not NULL), `pU` is not saved but put out immediately. If `spup` is given (i.e. is not NULL), the pair probabilities in `pl` are not saved but put out immediately.

Parameters

<i>sequence</i>	RNA sequence
<i>winSize</i>	size of the window
<i>pairSize</i>	maximum size of base pair
<i>cutoffb</i>	cutoffb for base pairs
<i>pU</i>	array holding all unpaired probabilities
<i>dpp2</i>	array of dependent pair probabilities
<i>pUfp</i>	file pointer for pU
<i>spup</i>	file pointer for pair probabilities

Returns

list of pair probabilities

16.94.2.3 pfl_fold_par()

```
vrna_ep_t * pfl_fold_par (
    char * sequence,
    int winSize,
    int pairSize,
    float cutoffb,
    double ** pU,
    vrna_ep_t ** dpp2,
    FILE * pUfp,
    FILE * spup,
    vrna_exp_param_t * parameters )
```

```
#include <ViennaRNA/LPfold.h>
```

Compute partition functions for locally stable secondary structures.

16.94.2.4 putoutpU_prob()

```
void putoutpU_prob (
    double ** pU,
    int length,
    int ulength,
    FILE * fp,
    int energies )
```

```
#include <ViennaRNA/LPfold.h>
```

Writes the unpaired probabilities (`pU`) or opening energies into a file.

Can write either the unpaired probabilities (accessibilities) p_U or the opening energies $-\log(p_U)kT$ into a file

Parameters

<i>pU</i>	pair probabilities
<i>length</i>	length of RNA sequence
<i>ulength</i>	maximum length of unpaired stretch
<i>fp</i>	file pointer of destination file
<i>energies</i>	switch to put out as opening energies

16.94.2.5 putoutpU_prob_bin()

```
void putoutpU_prob_bin (
    double ** pU,
    int length,
    int ulength,
    FILE * fp,
    int energies )
#include <ViennaRNA/LPfold.h>
```

Writes the unpaired probabilities (pU) or opening energies into a binary file.

Can write either the unpaired probabilities (accessibilities) pU or the opening energies $-\log(pU)kT$ into a file

Parameters

<i>pU</i>	pair probabilities
<i>length</i>	length of RNA sequence
<i>ulength</i>	maximum length of unpaired stretch
<i>fp</i>	file pointer of destination file
<i>energies</i>	switch to put out as opening energies

16.95 Deprecated Interface for Stochastic Backtracking

16.95.1 Detailed Description

Collaboration diagram for Deprecated Interface for Stochastic Backtracking:

Functions

- char * [pbacktrack](#) (char *sequence)
Sample a secondary structure from the Boltzmann ensemble according its probability.
- char * [pbacktrack5](#) (char *sequence, int length)
Sample a sub-structure from the Boltzmann ensemble according its probability.
- char * [pbacktrack_circ](#) (char *sequence)
Sample a secondary structure of a circular RNA from the Boltzmann ensemble according its probability.

Variables

- int [st_back](#)
Flag indicating that auxiliary arrays are needed throughout the computations. This is essential for stochastic backtracking.

16.95.2 Function Documentation

16.95.2.1 pbacktrack()

```
char * pbacktrack (
    char * sequence )
#include <ViennaRNA/part_func.h>
```

Sample a secondary structure from the Boltzmann ensemble according its probability.

Precondition

[st_back](#) has to be set to 1 before calling [pf_fold\(\)](#) or [pf_fold_par\(\)](#)

[pf_fold_par\(\)](#) or [pf_fold\(\)](#) have to be called first to fill the partition function matrices

Parameters

<i>sequence</i>	The RNA sequence
-----------------	------------------

Returns

A sampled secondary structure in dot-bracket notation

16.95.2.2 pbacktrack5()

```
char * pbacktrack5 (
    char * sequence,
    int length )
#include <ViennaRNA/part_func.h>
```

Sample a sub-structure from the Boltzmann ensemble according its probability.

16.95.2.3 pbacktrack_circ()

```
char * pbacktrack_circ (
    char * sequence )
#include <ViennaRNA/part_func.h>
```

Sample a secondary structure of a circular RNA from the Boltzmann ensemble according its probability.

This function does the same as [pbacktrack\(\)](#) but assumes the RNA molecule to be circular

Precondition

[st_back](#) has to be set to 1 before calling [pf_fold\(\)](#) or [pf_fold_par\(\)](#)

[pf_fold_par\(\)](#) or [pf_circ_fold\(\)](#) have to be called first to fill the partition function matrices

Deprecated Use [vrna_pbacktrack\(\)](#) instead.

Parameters

<i>sequence</i>	The RNA sequence
-----------------	------------------

Returns

A sampled secondary structure in dot-bracket notation

16.95.3 Variable Documentation

16.95.3.1 st_back

```
int st_back [extern]
#include <ViennaRNA/part_func.h>
```

Flag indicating that auxiliary arrays are needed throughout the computations. This is essential for stochastic backtracking.

Set this variable to 1 prior to a call of `pf_fold()` to ensure that all matrices needed for stochastic backtracking are filled in the forward recursions

Deprecated set the `uniq_ML` flag in `vrna_md_t` before passing it to `vrna_fold_compound()`.

See also

`pbacktrack()`, `pbacktrack_circ`

16.96 Deprecated Interface for Multiple Sequence Alignment Utilities

16.96.1 Detailed Description

Collaboration diagram for Deprecated Interface for Multiple Sequence Alignment Utilities:

Typedefs

- typedef struct `vrna_pinfo_s` `pair_info`
Old typename of `vrna_pinfo_s`.

Functions

- int `get_mpi` (char *Aseq[], int n_seq, int length, int *mini)
Get the mean pairwise identity in steps from ?to?(ident)
- void `encode_al_i_sequence` (const char *sequence, short *S, short *s5, short *s3, char *ss, unsigned short *as, int `circ`)
Get arrays with encoded sequence of the alignment.
- void `alloc_sequence_arrays` (const char **sequences, short ***S, short ***S5, short ***S3, unsigned short ***a2s, char ***Ss, int `circ`)
Allocate memory for sequence array used to deal with aligned sequences.
- void `free_sequence_arrays` (unsigned int n_seq, short ***S, short ***S5, short ***S3, unsigned short ***a2s, char ***Ss)
Free the memory of the sequence arrays used to deal with aligned sequences.

16.96.2 Typedef Documentation

16.96.2.1 pair_info

```
typedef struct vrna_pinfo_s pair_info
#include <ViennaRNA/utils/alignments.h>
Old typename of vrna_pinfo_s.
```

Deprecated Use `vrna_pinfo_t` instead!

16.96.3 Function Documentation

16.96.3.1 get_mpi()

```
int get_mpi (
    char * Alseq[],
    int n_seq,
    int length,
    int * mini )
```

#include <ViennaRNA/utils/alignments.h>

Get the mean pairwise identity in steps from ?to?(ident)

Deprecated Use `vrna_aln_mpi()` as a replacement

Parameters

<i>Alseq</i>	
<i>n_seq</i>	The number of sequences in the alignment
<i>length</i>	The length of the alignment
<i>mini</i>	

Returns

The mean pairwise identity

16.96.3.2 encode_al_sequence()

```
void encode_al_sequence (
    const char * sequence,
    short * S,
    short * s5,
    short * s3,
    char * ss,
    unsigned short * as,
    int circ )
```

#include <ViennaRNA/utils/alignments.h>

Get arrays with encoded sequence of the alignment.

this function assumes that in S, S5, s3, ss and as enough space is already allocated (size must be at least sequence length+2)

Parameters

<i>sequence</i>	The gapped sequence from the alignment
<i>S</i>	pointer to an array that holds encoded sequence
<i>s5</i>	pointer to an array that holds the next base 5' of alignment position i
<i>s3</i>	pointer to an array that holds the next base 3' of alignment position i
<i>ss</i>	
<i>as</i>	
<i>circ</i>	assume the molecules to be circular instead of linear (circ=0)

16.96.3.3 alloc_sequence_arrays()

```
void alloc_sequence_arrays (
    const char ** sequences,
    short *** S,
```

```

short *** S5,
short *** S3,
unsigned short *** a2s,
char *** Ss,
int circ )

```

#include <[ViennaRNA/utils/alignments.h](#)>

Allocate memory for sequence array used to deal with aligned sequences.

Note that these arrays will also be initialized according to the sequence alignment given

See also

[free_sequence_arrays\(\)](#)

Parameters

<i>sequences</i>	The aligned sequences
<i>S</i>	A pointer to the array of encoded sequences
<i>S5</i>	A pointer to the array that contains the next 5' nucleotide of a sequence position
<i>S3</i>	A pointer to the array that contains the next 3' nucleotide of a sequence position
<i>a2s</i>	A pointer to the array that contains the alignment to sequence position mapping
<i>Ss</i>	A pointer to the array that contains the ungapped sequence
<i>circ</i>	assume the molecules to be circular instead of linear (circ=0)

16.96.3.4 free_sequence_arrays()

```

void free_sequence_arrays (
    unsigned int n_seq,
    short *** S,
    short *** S5,
    short *** S3,
    unsigned short *** a2s,
    char *** Ss )

```

#include <[ViennaRNA/utils/alignments.h](#)>

Free the memory of the sequence arrays used to deal with aligned sequences.

This function frees the memory previously allocated with [alloc_sequence_arrays\(\)](#)

See also

[alloc_sequence_arrays\(\)](#)

Parameters

<i>n_seq</i>	The number of aligned sequences
<i>S</i>	A pointer to the array of encoded sequences
<i>S5</i>	A pointer to the array that contains the next 5' nucleotide of a sequence position
<i>S3</i>	A pointer to the array that contains the next 3' nucleotide of a sequence position
<i>a2s</i>	A pointer to the array that contains the alignment to sequence position mapping
<i>Ss</i>	A pointer to the array that contains the ungapped sequence

16.97 Deprecated Interface for Secondary Structure Utilities

16.97.1 Detailed Description

Collaboration diagram for Deprecated Interface for Secondary Structure Utilities:

Files

- file [RNAstruct.h](#)
Parsing and Coarse Graining of Structures.

Functions

- char * [b2HIT](#) (const char *structure)
Converts the full structure from bracket notation to the HIT notation including root.
- char * [b2C](#) (const char *structure)
Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.
- char * [b2Shapiro](#) (const char *structure)
Converts the full structure from bracket notation to the weighted coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.
- char * [add_root](#) (const char *structure)
Adds a root to an un-rooted tree in any except bracket notation.
- char * [expand_Shapiro](#) (const char *coarse)
Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from [b2C\(\)](#).
- char * [expand_Full](#) (const char *structure)
Convert the full structure from bracket notation to the expanded notation including root.
- char * [unexpand_Full](#) (const char *ffull)
Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.
- char * [unweight](#) (const char *wcoarse)
Strip weights from any weighted tree.
- void [unexpand_aligned_F](#) (char *align[2])
Converts two aligned structures in expanded notation.
- void [parse_structure](#) (const char *structure)
Collects a statistic of structure elements of the full structure in bracket notation.
- char * [pack_structure](#) (const char *struc)
Pack secondary secondary structure, 5:1 compression using base 3 encoding.
- char * [unpack_structure](#) (const char *packed)
Unpack secondary structure previously packed with [pack_structure\(\)](#)
- short * [make_pair_table](#) (const char *structure)
Create a pair table of a secondary structure.
- short * [copy_pair_table](#) (const short *pt)
Get an exact copy of a pair table.
- short * [alimake_pair_table](#) (const char *structure)
- short * [make_pair_table_snoop](#) (const char *structure)
- int [bp_distance](#) (const char *str1, const char *str2)
Compute the "base pair" distance between two secondary structures s1 and s2.
- unsigned int * [make_referenceBP_array](#) (short *reference_pt, unsigned int turn)
Make a reference base pair count matrix.
- unsigned int * [compute_BPdifferences](#) (short *pt1, short *pt2, unsigned int turn)
Make a reference base pair distance matrix.
- void [parenthesis_structure](#) (char *structure, [vrna_bp_stack_t](#) *bp, int length)
Create a dot-bracket/parenthesis structure from backtracking stack.

- void [parenthesis_zuker](#) (char *structure, [vrna_bp_stack_t](#) *bp, int length)
Create a dot-bracket/parenthesis structure from backtracking stack obtained by zuker suboptimal calculation in cofold.c.
- void [bppm_to_structure](#) (char *structure, [FLT_OR_DBL](#) *pr, unsigned int length)
Create a dot-bracket like structure string from base pair probability matrix.
- char [bppm_symbol](#) (const float *x)
Get a pseudo dot bracket notation for a given probability information.

Variables

- int **loop_size** [STRUC]
contains a list of all loop sizes. loop_size[0] contains the number of external bases.
- int **helix_size** [STRUC]
contains a list of all stack sizes.
- int **loop_degree** [STRUC]
contains the corresponding list of loop degrees.
- int **loops**
contains the number of loops (and therefore of stacks).
- int **unpaired**
contains the number of unpaired bases.
- int **pairs**
contains the number of base pairs in the last parsed structure.

16.97.2 Function Documentation

16.97.2.1 b2HIT()

```
char * b2HIT (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
Converts the full structure from bracket notation to the HIT notation including root.
```

Deprecated See [vrna_db_to_tree_string\(\)](#) and [VRNA_STRUCTURE_TREE_HIT](#) for a replacement

Parameters

<i>structure</i>	
------------------	--

Returns

16.97.2.2 b2C()

```
char * b2C (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.
```

Deprecated See [vrna_db_to_tree_string\(\)](#) and [VRNA_STRUCTURE_TREE_SHAPIRO_SHORT](#) for a replacement

Parameters

<i>structure</i>	
------------------	--

Returns

16.97.2.3 b2Shapiro()

```
char * b2Shapiro (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
```

Converts the full structure from bracket notation to the *weighted* coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.

Deprecated See [vrna_db_to_tree_string\(\)](#) and [VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT](#) for a replacement

Parameters

<i>structure</i>	
------------------	--

Returns

16.97.2.4 add_root()

```
char * add_root (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
```

Adds a root to an un-rooted tree in any except bracket notation.

Parameters

<i>structure</i>	
------------------	--

Returns

16.97.2.5 expand_Shapiro()

```
char * expand_Shapiro (
    const char * coarse )
#include <ViennaRNA/RNAstruct.h>
```

Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from [b2C\(\)](#).

Parameters

<i>coarse</i>	
---------------	--

Returns**16.97.2.6 expand_Full()**

```
char * expand_Full (
    const char * structure )
#include <ViennaRNA/RNAstruct.h>
Convert the full structure from bracket notation to the expanded notation including root.
```

Parameters

<i>structure</i>	
------------------	--

Returns**16.97.2.7 unexpand_Full()**

```
char * unexpand_Full (
    const char * ffull )
#include <ViennaRNA/RNAstruct.h>
Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.
```

Parameters

<i>ffull</i>	
--------------	--

Returns**16.97.2.8 unweight()**

```
char * unweight (
    const char * wcoarse )
#include <ViennaRNA/RNAstruct.h>
Strip weights from any weighted tree.
```

Parameters

<i>wcoarse</i>	
----------------	--

Returns**16.97.2.9 unexpand_aligned_F()**

```
void unexpand_aligned_F (
    char * align[2] )
```

```
#include <ViennaRNA/RNAstruct.h>
```

Converts two aligned structures in expanded notation.

Takes two aligned structures as produced by [tree_edit_distance\(\)](#) function back to bracket notation with '_' as the gap character. The result overwrites the input.

Parameters

<i>align</i>	
--------------	--

16.97.2.10 parse_structure()

```
void parse_structure (
    const char * structure )
```

```
#include <ViennaRNA/RNAstruct.h>
```

Collects a statistic of structure elements of the full structure in bracket notation.

The function writes to the following global variables: [loop_size](#), [loop_degree](#), [helix_size](#), [loops](#), [pairs](#), [unpaired](#)

Parameters

<i>structure</i>	
------------------	--

16.97.2.11 pack_structure()

```
char * pack_structure (
    const char * struc )
```

```
#include <ViennaRNA/utils/structures.h>
```

Pack secondary secondary structure, 5:1 compression using base 3 encoding.

Returns a binary string encoding of the secondary structure using a 5:1 compression scheme. The string is NULL terminated and can therefore be used with standard string functions such as `strcmp()`. Useful for programs that need to keep many structures in memory.

Deprecated Use [vrna_db_pack\(\)](#) as a replacement

Parameters

<i>struc</i>	The secondary structure in dot-bracket notation
--------------	---

Returns

The binary encoded structure

16.97.2.12 unpack_structure()

```
char * unpack_structure (
    const char * packed )
```

```
#include <ViennaRNA/utils/structures.h>
```

Unpack secondary structure previously packed with [pack_structure\(\)](#)

Translate a compressed binary string produced by [pack_structure\(\)](#) back into the familiar dot-bracket notation.

Deprecated Use [vrna_db_unpack\(\)](#) as a replacement

Parameters

<i>packed</i>	The binary encoded packed secondary structure
---------------	---

Returns

The unpacked secondary structure in dot-bracket notation

16.97.2.13 make_pair_table()

```
short * make_pair_table (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Create a pair table of a secondary structure.

Returns a newly allocated table, such that table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure.

Deprecated Use [vrna_ptable\(\)](#) instead

Parameters

<i>structure</i>	The secondary structure in dot-bracket notation
------------------	---

Returns

A pointer to the created pair_table

16.97.2.14 copy_pair_table()

```
short * copy_pair_table (
    const short * pt )
#include <ViennaRNA/utils/structures.h>
```

Get an exact copy of a pair table.

Deprecated Use [vrna_ptable_copy\(\)](#) instead

Parameters

<i>pt</i>	The pair table to be copied
-----------	-----------------------------

Returns

A pointer to the copy of 'pt'

16.97.2.15 alimake_pair_table()

```
short * alimake_pair_table (
    const char * structure )
#include <ViennaRNA/utils/structures.h>
```

Pair table for snoop align

Deprecated Use [vrna_pt_ali_get\(\)](#) instead!

16.97.2.16 make_pair_table_snoop()

```
short * make_pair_table_snoop (
    const char * structure )
```

```
#include <ViennaRNA/utils/structures.h>
```

returns a newly allocated table, such that: table[i]=j if (i,j) pair or 0 if i is unpaired, table[0] contains the length of the structure. The special pseudoknotted H/ACA-mRNA structure is taken into account.

Deprecated Use [vrna_pt_snoop_get\(\)](#) instead!

16.97.2.17 bp_distance()

```
int bp_distance (
    const char * str1,
    const char * str2 )
```

```
#include <ViennaRNA/utils/structures.h>
```

Compute the "base pair" distance between two secondary structures s1 and s2.

The sequences should have the same length. dist = number of base pairs in one structure but not in the other same as edit distance with open-pair close-pair as move-set

Deprecated Use [vrna_bp_distance](#) instead

Parameters

<i>str1</i>	First structure in dot-bracket notation
<i>str2</i>	Second structure in dot-bracket notation

Returns

The base pair distance between str1 and str2

16.97.2.18 make_referenceBP_array()

```
unsigned int * make_referenceBP_array (
    short * reference_pt,
    unsigned int turn )
```

```
#include <ViennaRNA/utils/structures.h>
```

Make a reference base pair count matrix.

Get an upper triangular matrix containing the number of basepairs of a reference structure for each interval [i,j] with i<j. Access it via iindx!!!

Deprecated Use [vrna_refBPcnt_matrix\(\)](#) instead

16.97.2.19 compute_BPdifferences()

```
unsigned int * compute_BPdifferences (
    short * pt1,
    short * pt2,
    unsigned int turn )
```

```
#include <ViennaRNA/utils/structures.h>
```

Make a reference base pair distance matrix.

Get an upper triangular matrix containing the base pair distance of two reference structures for each interval [i,j] with i<j. Access it via iindx!!!

Deprecated Use `vrna_refBPdist_matrix()` instead

16.97.2.20 `parenthesis_structure()`

```
void parenthesis_structure (
    char * structure,
    vrna_bp_stack_t * bp,
    int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket/parenthesis structure from backtracking stack.
```

Deprecated use `vrna_parenthesis_structure()` instead

Note

This function is threadsafe

16.97.2.21 `parenthesis_zuker()`

```
void parenthesis_zuker (
    char * structure,
    vrna_bp_stack_t * bp,
    int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket/parenthesis structure from backtracking stack obtained by zuker suboptimal calculation in cofold.c.
```

Deprecated use `vrna_parenthesis_zuker` instead

Note

This function is threadsafe

16.97.2.22 `bppm_to_structure()`

```
void bppm_to_structure (
    char * structure,
    FLT_OR_DBL * pr,
    unsigned int length )
#include <ViennaRNA/utils/structures.h>
Create a dot-bracket like structure string from base pair probability matrix.
```

Deprecated Use `vrna_db_from_probs()` instead!

16.97.2.23 `bppm_symbol()`

```
char bppm_symbol (
    const float * x )
#include <ViennaRNA/utils/structures.h>
Get a pseudo dot bracket notation for a given probability information.
```

Deprecated Use `vrna_bpp_symbol()` instead!

16.98 Deprecated Interface for Plotting Utilities

16.98.1 Detailed Description

Collaboration diagram for Deprecated Interface for Plotting Utilities:

Data Structures

- struct [COORDINATE](#)
this is a workaround for the SWIG Perl Wrapper RNA plot function that returns an array of type [COORDINATE](#) [More...](#)

Functions

- int [PS_color_aln](#) (const char *structure, const char *filename, const char *seqs[], const char *names[])
Produce PostScript sequence alignment color-annotated by consensus structure.
- int [aliPS_color_aln](#) (const char *structure, const char *filename, const char *seqs[], const char *names[])
PS_color_aln for duplexes.
- int [simple_xy_coordinates](#) (short *pair_table, float *X, float *Y)
Calculate nucleotide coordinates for secondary structure plot the Simple way
- int [simple_circplot_coordinates](#) (short *pair_table, float *x, float *y)
Calculate nucleotide coordinates for Circular Plot

Variables

- int [rna_plot_type](#)
Switch for changing the secondary structure layout algorithm.

16.98.2 Data Structure Documentation

16.98.2.1 struct COORDINATE

this is a workaround for the SWIG Perl Wrapper RNA plot function that returns an array of type [COORDINATE](#)

16.98.3 Function Documentation

16.98.3.1 PS_color_aln()

```
int PS_color_aln (
    const char * structure,
    const char * filename,
    const char * seqs[],
    const char * names[] )
#include <ViennaRNA/plotting/alignments.h>
Produce PostScript sequence alignment color-annotated by consensus structure.
```

Deprecated Use [vrna_file_PS_aln\(\)](#) instead!

16.98.3.2 aliPS_color_aln()

```
int aliPS_color_aln (
    const char * structure,
    const char * filename,
    const char * seqs[],
    const char * names[] )
```

```
#include <ViennaRNA/plotting/alignments.h>
PS_color_aln for duplexes.
```

Deprecated Use `vrna_file_PS_aln()` instead!

16.98.3.3 simple_xy_coordinates()

```
int simple_xy_coordinates (
    short * pair_table,
    float * X,
    float * Y )
#include <ViennaRNA/plotting/layouts.h>
Calculate nucleotide coordinates for secondary structure plot the Simple way
```

See also

`make_pair_table()`, `rna_plot_type`, `simple_circplot_coordinates()`, `naview_xy_coordinates()`, `vrna_file_PS_rnaplot_a()`, `vrna_file_PS_rnaplot`, `svg_rna_plot()`

Deprecated Consider switching to `vrna_plot_coords_simple_pt()` instead!

Parameters

<i>pair_table</i>	The pair table of the secondary structure
<i>X</i>	a pointer to an array with enough allocated space to hold the x coordinates
<i>Y</i>	a pointer to an array with enough allocated space to hold the y coordinates

Returns

length of sequence on success, 0 otherwise

16.98.3.4 simple_circplot_coordinates()

```
int simple_circplot_coordinates (
    short * pair_table,
    float * x,
    float * y )
#include <ViennaRNA/plotting/layouts.h>
Calculate nucleotide coordinates for Circular Plot
This function calculates the coordinates of nucleotides mapped in equal distances onto a unit circle.
```

Note

In order to draw nice arcs using quadratic bezier curves that connect base pairs one may calculate a second tangential point P^t in addition to the actual R^2 coordinates. the simplest way to do so may be to compute a radius scaling factor rs in the interval $[0, 1]$ that weights the proportion of base pair span to the actual length of the sequence. This scaling factor can then be used to calculate the coordinates for P^t , i.e. $P_x^t[i] = X[i] * rs$ and $P_y^t[i] = Y[i] * rs$.

See also

`make_pair_table()`, `rna_plot_type`, `simple_xy_coordinates()`, `naview_xy_coordinates()`, `vrna_file_PS_rnaplot_a()`, `vrna_file_PS_rnaplot`, `svg_rna_plot()`

Deprecated Consider switching to `vrna_plot_coords_circular_pt()` instead!

Parameters

<i>pair_table</i>	The pair table of the secondary structure
<i>x</i>	a pointer to an array with enough allocated space to hold the x coordinates
<i>y</i>	a pointer to an array with enough allocated space to hold the y coordinates

Returns

length of sequence on success, 0 otherwise

16.98.4 Variable Documentation

16.98.4.1 rna_plot_type

```
int rna_plot_type [extern]
```

```
#include <ViennaRNA/plotting/layouts.h>
```

Switch for changing the secondary structure layout algorithm.

Current possibilities are 0 for a simple radial drawing or 1 for the modified radial drawing taken from the *naview* program of [6].

Note

To provide thread safety please do not rely on this global variable in future implementations but pass a plot type flag directly to the function that decides which layout algorithm it may use!

See also

[VRNA_PLOT_TYPE_SIMPLE](#), [VRNA_PLOT_TYPE_NAVIEW](#), [VRNA_PLOT_TYPE_CIRCULAR](#)

16.99 Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers

16.99.1 Detailed Description

Collaboration diagram for Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers:

Typedefs

- typedef struct [vrna_path_s](#) [path_t](#)
Old typename of [vrna_path_s](#).

Functions

- int [find_saddle](#) (const char *seq, const char *s1, const char *s2, int width)
Find energy of a saddle point between 2 structures (search only direct path)
- void [free_path](#) ([vrna_path_t](#) *path)
Free memory allocated by [get_path\(\)](#) function.
- [vrna_path_t](#) * [get_path](#) (const char *seq, const char *s1, const char *s2, int width)
Find refolding path between 2 structures (search only direct path)

16.99.2 Typedef Documentation

16.99.2.1 path_t

```
typedef struct vrna_path_s path_t
#include <ViennaRNA/landscape/paths.h>
Old typename of vrna_path_s.
```

Deprecated Use `vrna_path_t` instead!

16.99.3 Function Documentation

16.99.3.1 find_saddle()

```
int find_saddle (
    const char * seq,
    const char * s1,
    const char * s2,
    int width )
#include <ViennaRNA/landscape/findpath.h>
Find energy of a saddle point between 2 structures (search only direct path)
```

Deprecated Use `vrna_path_findpath_saddle()` instead!

Parameters

<i>seq</i>	RNA sequence
<i>s1</i>	A pointer to the character array where the first secondary structure in dot-bracket notation will be written to
<i>s2</i>	A pointer to the character array where the second secondary structure in dot-bracket notation will be written to
<i>width</i>	integer how many strutures are being kept during the search

Returns

the saddle energy in 10cal/mol

16.99.3.2 free_path()

```
void free_path (
    vrna_path_t * path )
#include <ViennaRNA/landscape/findpath.h>
Free memory allocated by get_path() function.
```

Deprecated Use `vrna_path_free()` instead!

Parameters

<i>path</i>	pointer to memory to be freed
-------------	-------------------------------

16.99.3.3 get_path()

```
vrna_path_t * get_path (
```

```
    const char * seq,  
    const char * s1,  
    const char * s2,  
    int width )  
#include <ViennaRNA/landscape/findpath.h>  
Find refolding path between 2 structures (search only direct path)
```

Deprecated Use `vrna_path_findpath()` instead!

Parameters

<i>seq</i>	RNA sequence
<i>s1</i>	A pointer to the character array where the first secondary structure in dot-bracket notation will be written to
<i>s2</i>	A pointer to the character array where the second secondary structure in dot-bracket notation will be written to
<i>width</i>	integer how many structures are being kept during the search

Returns

direct refolding path between two structures

Chapter 17

Data Structure Documentation

17.1 `_struct_en` Struct Reference

Data structure for [energy_of_move\(\)](#)

17.1.1 Detailed Description

Data structure for [energy_of_move\(\)](#)

The documentation for this struct was generated from the following file:

- ViennaRNA/move_set.h

17.2 `energy_corrections` Struct Reference

The documentation for this struct was generated from the following file:

- ViennaRNA/constraints/sc_cb_intern.h

17.3 `LIST` Struct Reference

Collaboration diagram for LIST:

The documentation for this struct was generated from the following file:

- ViennaRNA/datastructures/lists.h

17.4 `LST_BUCKET` Struct Reference

Collaboration diagram for LST_BUCKET:

The documentation for this struct was generated from the following file:

- ViennaRNA/datastructures/lists.h

17.5 `Postorder_list` Struct Reference

Postorder data structure.

17.5.1 Detailed Description

Postorder data structure.

The documentation for this struct was generated from the following file:

- ViennaRNA/[dist_vars.h](#)

17.6 swString Struct Reference

Some other data structure.

17.6.1 Detailed Description

Some other data structure.

The documentation for this struct was generated from the following file:

- ViennaRNA/[dist_vars.h](#)

17.7 Tree Struct Reference

[Tree](#) data structure.

Collaboration diagram for Tree:

17.7.1 Detailed Description

[Tree](#) data structure.

The documentation for this struct was generated from the following file:

- ViennaRNA/[dist_vars.h](#)

17.8 TwoDpfold_vars Struct Reference

Variables compound for 2Dfold partition function folding.

Collaboration diagram for TwoDpfold_vars:

Data Fields

- char * **ptype**
Precomputed array of pair types.
- char * **sequence**
The input sequence
- short * **S1**
The input sequences in numeric form.
- unsigned int **maxD1**
Maximum allowed base pair distance to first reference.
- unsigned int **maxD2**
Maximum allowed base pair distance to second reference.
- int * **my_iindx**
Index for moving in quadratic distance dimensions.
- int * **jindx**
Index for moving in the triangular matrix qm1.
- unsigned int * **referenceBPs1**
Matrix containing number of basepairs of reference structure1 in interval [i,j].
- unsigned int * **referenceBPs2**
Matrix containing number of basepairs of reference structure2 in interval [i,j].
- unsigned int * **bpdist**
Matrix containing base pair distance of reference structure 1 and 2 on interval [i,j].
- unsigned int * **mm1**
Maximum matching matrix, reference struct 1 disallowed.
- unsigned int * **mm2**
Maximum matching matrix, reference struct 2 disallowed.

17.8.1 Detailed Description

Variables compound for 2Dfold partition function folding.

Deprecated This data structure will be removed from the library soon! Use [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

The documentation for this struct was generated from the following file:

- [ViennaRNA/2Dpfold.h](#)

17.9 vrna_dimer_conc_s Struct Reference

Data structure for concentration dependency computations.

Data Fields

- double **Ac_start**
start concentration A
- double **Bc_start**
start concentration B
- double **ABc**
End concentration AB.

17.9.1 Detailed Description

Data structure for concentration dependency computations.

The documentation for this struct was generated from the following file:

- [ViennaRNA/concentrations.h](#)

17.10 vrna_sc_bp_storage_t Struct Reference

A base pair constraint.

17.10.1 Detailed Description

A base pair constraint.

The documentation for this struct was generated from the following file:

- [ViennaRNA/constraints/soft.h](#)

17.11 vrna_sc_mod_param_s Struct Reference

The documentation for this struct was generated from the following file:

- [ViennaRNA/constraints/sc_cb_intern.h](#)

17.12 vrna_string_header_s Struct Reference

The header of an array.

Data Fields

- size_t **len**
The length of the string.
- size_t **size**
The actual capacity of an array.

17.12.1 Detailed Description

The header of an array.

The documentation for this struct was generated from the following file:

- [ViennaRNA/datastructures/string.h](#)

17.13 `vrna_structured_domains_s` Struct Reference

The documentation for this struct was generated from the following file:

- [ViennaRNA/structured_domains.h](#)

17.14 `vrna_subopt_sol_s` Struct Reference

Solution element from subopt.c.

Data Fields

- float **energy**
Free Energy of structure in kcal/mol.
- char * **structure**
Structure in dot-bracket notation.

17.14.1 Detailed Description

Solution element from subopt.c.

The documentation for this struct was generated from the following file:

- [ViennaRNA/subopt.h](#)

17.15 `vrna_unstructured_domain_motif_s` Struct Reference

The documentation for this struct was generated from the following file:

- [ViennaRNA/unstructured_domains.h](#)

Chapter 18

File Documentation

18.1 ViennaRNA/2Dfold.h File Reference

MFE structures for base pair distance classes.

Include dependency graph for 2Dfold.h:

18.2 2Dfold.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_TWO_D_FOLD_H
00002 #define VIENNA_RNA_PACKAGE_TWO_D_FOLD_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00035 #include <ViennaRNA/fold_compound.h>
00036 #include <ViennaRNA/datastructures/basic.h>
00037 #include <ViennaRNA/params/basic.h>
00038
00053 typedef struct vrna_sol_TwoD_t {
00054     int k;
00055     int l;
00056     float en;
00057     char *s;
00058 } vrna_sol_TwoD_t;
00059
00060
00088 vrna_sol_TwoD_t *
00089 vrna_mfe_TwoD(vrna_fold_compound_t *vc,
00090              int distance1,
00091              int distance2);
00092
00093
00112 char *
00113 vrna_backtrack5_TwoD(vrna_fold_compound_t *vc,
00114                    int k,
00115                    int l,
00116                    unsigned int j);
00117
00118
00119 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00120
00121 #define TwoDfold_solution vrna_sol_TwoD_t /* restore compatibility of struct rename */
00122
00130 typedef struct TwoDfold_vars {
00131     vrna_param_t *P;
00132     int do_backtrack;
00133     char *ptype;
00134     char *sequence;
00135     short *S, *S1;
00136     unsigned int maxD1;
00137     unsigned int maxD2;
```

```

00140 unsigned int      *mm1;
00141 unsigned int      *mm2;
00143 int               *my_iindx;
00145 double           temperature;
00146
00147 unsigned int      *referenceBPs1;
00148 unsigned int      *referenceBPs2;
00149 unsigned int      *bpdist;
00151 short            *reference_pt1;
00152 short            *reference_pt2;
00153 int              circ;
00154 int              dangles;
00155 unsigned int      seq_length;
00156
00157 int              ***E_F5;
00158 int              ***E_F3;
00159 int              ***E_C;
00160 int              ***E_M;
00161 int              ***E_M1;
00162 int              ***E_M2;
00163
00164 int              **E_Fc;
00165 int              **E_FcH;
00166 int              **E_FcI;
00167 int              **E_FcM;
00168
00169 int              **l_min_values;
00170 int              **l_max_values;
00171 int              *k_min_values;
00172 int              *k_max_values;
00173
00174 int              **l_min_values_m;
00175 int              **l_max_values_m;
00176 int              *k_min_values_m;
00177 int              *k_max_values_m;
00178
00179 int              **l_min_values_m1;
00180 int              **l_max_values_m1;
00181 int              *k_min_values_m1;
00182 int              *k_max_values_m1;
00183
00184 int              **l_min_values_f;
00185 int              **l_max_values_f;
00186 int              *k_min_values_f;
00187 int              *k_max_values_f;
00188
00189 int              **l_min_values_f3;
00190 int              **l_max_values_f3;
00191 int              *k_min_values_f3;
00192 int              *k_max_values_f3;
00193
00194 int              **l_min_values_m2;
00195 int              **l_max_values_m2;
00196 int              *k_min_values_m2;
00197 int              *k_max_values_m2;
00198
00199 int              *l_min_values_fc;
00200 int              *l_max_values_fc;
00201 int              k_min_values_fc;
00202 int              k_max_values_fc;
00203
00204 int              *l_min_values_fcH;
00205 int              *l_max_values_fcH;
00206 int              k_min_values_fcH;
00207 int              k_max_values_fcH;
00208
00209 int              *l_min_values_fcI;
00210 int              *l_max_values_fcI;
00211 int              k_min_values_fcI;
00212 int              k_max_values_fcI;
00213
00214 int              *l_min_values_fcM;
00215 int              *l_max_values_fcM;
00216 int              k_min_values_fcM;
00217 int              k_max_values_fcM;
00218
00219 /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
00220 int              *E_F5_rem;
00221 int              *E_F3_rem;
00222 int              *E_C_rem;
00223 int              *E_M_rem;
00224 int              *E_M1_rem;
00225 int              *E_M2_rem;
00226
00227 int              E_Fc_rem;
00228 int              E_FcH_rem;
00229 int              E_FcI_rem;

```



```

00230     int                E_FcM_rem;
00231
00232 #ifdef COUNT_STATES
00233     unsigned long      ***N_F5;
00234     unsigned long      ***N_C;
00235     unsigned long      ***N_M;
00236     unsigned long      ***N_M1;
00237 #endif
00238
00239     vrna_fold_compound_t *compatibility;
00240 } TwoDfold_vars;
00241
00260 DEPRECATED(TwoDfold_vars *
00261     get_TwoDfold_variables(const char *seq,
00262                           const char *structure1,
00263                           const char *structure2,
00264                           int circ),
00265     "Use the new API and corresponding functions vrna_fold_compound_TwoD(), etc. instead");
00266
00277 DEPRECATED(void
00278     destroy_TwoDfold_variables(TwoDfold_vars *our_variables),
00279     "Use the new API and vrna_fold_compound_free() instead");
00280
00306 DEPRECATED(TwoDfold_solution *
00307     TwoDfoldList(TwoDfold_vars *vars,
00308                 int distance1,
00309                 int distance2),
00310     "Use the new API and vrna_mfe_TwoD() instead");
00311
00332 DEPRECATED(char *TwoDfold_backtrack_f5(unsigned int j,
00333                                         int k,
00334                                         int l,
00335                                         TwoDfold_vars *vars),
00336     "Use the new API and vrna_backtrack5_TwoD() instead");
00337
00341 DEPRECATED(TwoDfold_solution **TwoDfold(TwoDfold_vars *our_variables,
00342                                         int distance1,
00343                                         int distance2),
00344     "Use the new API and vrna_mfe_TwoD() instead");
00345
00346
00347 #endif
00348
00353 #endif

```

18.3 ViennaRNA/2Dpfold.h File Reference

Partition function implementations for base pair distance classes.

Include dependency graph for 2Dpfold.h:

Data Structures

- struct [vrna_sol_TwoD_pf_t](#)
Solution element returned from [vrna_pf_TwoD\(\)](#) [More...](#)
- struct [TwoDpfold_vars](#)
Variables compound for 2Dfold partition function folding.

Typedefs

- typedef struct [vrna_sol_TwoD_pf_t](#) [vrna_sol_TwoD_pf_t](#)
Solution element returned from [vrna_pf_TwoD\(\)](#)

Functions

- [vrna_sol_TwoD_pf_t](#) * [vrna_pf_TwoD](#) ([vrna_fold_compound_t](#) *vc, int maxDistance1, int maxDistance2)
Compute the partition function for all distance classes.
- char * [vrna_pbacktrack_TwoD](#) ([vrna_fold_compound_t](#) *vc, int d1, int d2)
Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability.
- char * [vrna_pbacktrack5_TwoD](#) ([vrna_fold_compound_t](#) *vc, int d1, int d2, unsigned int length)
Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

- `TwoDpfold_vars * get_TwoDpfold_variables` (const char *seq, const char *structure1, char *structure2, int circ)
Get a datastructure containing all necessary attributes and global folding switches.
- void `destroy_TwoDpfold_variables` (TwoDpfold_vars *vars)
Free all memory occupied by a TwoDpfold_vars datastructure.
- TwoDpfold_solution * `TwoDpfoldList` (TwoDpfold_vars *vars, int maxDistance1, int maxDistance2)
Compute the partition function for all distance classes.
- char * `TwoDpfold_pbacktrack` (TwoDpfold_vars *vars, int d1, int d2)
Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability.
- char * `TwoDpfold_pbacktrack5` (TwoDpfold_vars *vars, int d1, int d2, unsigned int length)
Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

18.3.1 Detailed Description

Partition function implementations for base pair distance classes.

18.3.2 Function Documentation

18.3.2.1 get_TwoDpfold_variables()

```
TwoDpfold_vars * get_TwoDpfold_variables (
    const char * seq,
    const char * structure1,
    char * structure2,
    int circ )
```

Get a datastructure containing all necessary attributes and global folding switches.

This function prepares all necessary attributes and matrices etc which are needed for a call of `TwoDpfold()`. A snapshot of all current global model switches (dangles, temperature and so on) is done and stored in the returned datastructure. Additionally, all matrices that will hold the partition function values are prepared.

Deprecated Use the new API that relies on `vrna_fold_compound_t` and the corresponding functions `vrna_fold_compound_TwoD()`, `vrna_pf_TwoD()`, and `vrna_fold_compound_free()` instead!

Parameters

<i>seq</i>	the RNA sequence in uppercase format with letters from the alphabet {AUCG}
<i>structure1</i>	the first reference structure in dot-bracket notation
<i>structure2</i>	the second reference structure in dot-bracket notation
<i>circ</i>	a switch indicating if the sequence is linear (0) or circular (1)

Returns

the datastructure containing all necessary partition function attributes

18.3.2.2 destroy_TwoDpfold_variables()

```
void destroy_TwoDpfold_variables (
    TwoDpfold_vars * vars )
```

Free all memory occupied by a `TwoDpfold_vars` datastructure.

This function free's all memory occupied by a datastructure obtained from `get_TwoDpfold_variables()` or `get_TwoDpfold_variables_from_MFE()`

Deprecated Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

See also

[get_TwoDpfold_variables\(\)](#), [get_TwoDpfold_variables_from_MFE\(\)](#)

Parameters

<i>vars</i>	the datastructure to be free'd
-------------	--------------------------------

18.3.2.3 TwoDpfoldList()

```
TwoDpfold_solution * TwoDpfoldList (
    TwoDpfold_vars * vars,
    int maxDistance1,
    int maxDistance2 )
```

Compute the partition function for all distance classes.

This function computes the partition functions for all distance classes according the two reference structures specified in the datastructure 'vars'. Similar to [TwoDfold\(\)](#) the arguments `maxDistance1` and `maxDistance2` specify the maximum distance to both reference structures. A value of '-1' in either of them makes the appropriate distance restrictionless, i.e. all basepair distances to the reference are taken into account during computation. In case there is a restriction, the returned solution contains an entry where the attribute `k` is -1 contains the partition function for all structures exceeding the restriction. A values of INF in the attribute 'k' of the returned list denotes the end of the list

Deprecated Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

See also

[get_TwoDpfold_variables\(\)](#), [destroy_TwoDpfold_variables\(\)](#), [vrna_sol_TwoD_pf_t](#)

Parameters

<i>vars</i>	the datastructure containing all necessary folding attributes and matrices
<i>maxDistance1</i>	the maximum basepair distance to reference1 (may be -1)
<i>maxDistance2</i>	the maximum basepair distance to reference2 (may be -1)

Returns

a list of partition funtions for the appropriate distance classes

18.3.2.4 TwoDpfold_pbacktrack()

```
char * TwoDpfold_pbacktrack (
    TwoDpfold_vars * vars,
    int d1,
    int d2 )
```

Sample secondary structure representatives from a set of distance classes according to their Boltzmann probability. If the argument 'd1' is set to '-1', the structure will be backtracked in the distance class where all structures exceeding the maximum basepair distance to either of the references reside.

Precondition

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to `TwoDpfold()` preceding this function is mandatory!

Deprecated Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#), [vrna_pbacktrack_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

See also

`TwoDpfold()`

Parameters

in	<i>vars</i>	the datastructure containing all necessary folding attributes and matrices
in	<i>d1</i>	the distance to reference1 (may be -1)
in	<i>d2</i>	the distance to reference2

Returns

A sampled secondary structure in dot-bracket notation

18.3.2.5 TwoDpfold_pbacktrack5()

```
char * TwoDpfold_pbacktrack5 (
    TwoDpfold_vars * vars,
    int d1,
    int d2,
    unsigned int length )
```

Sample secondary structure representatives with a specified length from a set of distance classes according to their Boltzmann probability.

This function does essentially the same as [TwoDpfold_pbacktrack\(\)](#) with the only difference that partial structures, i.e. structures beginning from the 5' end with a specified length of the sequence, are backtracked

Note

This function does not work (since it makes no sense) for circular RNA sequences!

Precondition

The argument 'vars' must contain precalculated partition function matrices, i.e. a call to `TwoDpfold()` preceding this function is mandatory!

Deprecated Use the new API that relies on [vrna_fold_compound_t](#) and the corresponding functions [vrna_fold_compound_TwoD\(\)](#), [vrna_pf_TwoD\(\)](#), [vrna_pbacktrack5_TwoD\(\)](#), and [vrna_fold_compound_free\(\)](#) instead!

See also

[TwoDpfold_pbacktrack\(\)](#), `TwoDpfold()`

Parameters

in	<i>vars</i>	the datastructure containing all necessary folding attributes and matrices
in	<i>d1</i>	the distance to reference1 (may be -1)
in	<i>d2</i>	the distance to reference2
in	<i>length</i>	the length of the structure beginning from the 5' end

Returns

A sampled secondary structure in dot-bracket notation

18.4 2Dpfold.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_TWO_D_PF_FOLD_H
00002 #define VIENNA_RNA_PACKAGE_TWO_D_PF_FOLD_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00032 #include <ViennaRNA/fold_compound.h>
00033 #include <ViennaRNA/datastructures/basic.h>
00034 #include <ViennaRNA/fold_compound.h>
00035 #include <ViennaRNA/params/basic.h>
00036
00049 typedef struct vrna_sol_TwoD_pf_t {
00050     int k;
00051     int l;
00052     FLT_OR_DBL q;
00053 } vrna_sol_TwoD_pf_t;
00054
00077 vrna_sol_TwoD_pf_t *
00078 vrna_pf_TwoD(vrna_fold_compound_t *vc,
00079             int maxDistance1,
00080             int maxDistance2);
00081
00082 /* End of group kl_neighborhood_pf */
00084
00108 char *
00109 vrna_pbacktrack_TwoD(vrna_fold_compound_t *vc,
00110                     int d1,
00111                     int d2);
00112
00113
00133 char *
00134 vrna_pbacktrack5_TwoD(vrna_fold_compound_t *vc,
00135                      int d1,
00136                      int d2,
00137                      unsigned int length);
00138
00139 /* End of group kl_neighborhood_stochbt */
00143
00144 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00145
00146 #define TwoDpfold_solution vrna_sol_TwoD_pf_t /* restore compatibility of struct rename */
00147
00155 typedef struct {
00156     unsigned int alloc;
00157     char *ptype;
00158     char *sequence;
00159     short *S, *S1;
00160     unsigned int maxD1;
00161     unsigned int maxD2;
00163     double temperature; /* temperature in last call to scale_pf_params */
00164     double init_temp; /* temperature in last call to scale_pf_params */
00165     FLT_OR_DBL scale;
00166     FLT_OR_DBL pf_scale;
00167     vrna_exp_param_t *pf_params; /* holds all [unscaled] pf parameters */
00168
00169     int *my_iindx;
00170     int *jindx;
00172     short *reference_pt1;
00173     short *reference_pt2;
00174
00175     unsigned int *referenceBPs1;
00176     unsigned int *referenceBPs2;
00177     unsigned int *bpdist;
00179     unsigned int *mm1;
00180     unsigned int *mm2;
00182     int circ;
00183     int dangles;

```

```

00184 unsigned int      seq_length;
00185
00186 FLT_OR_DBL         ***Q;
00187 FLT_OR_DBL         ***Q_B;
00188 FLT_OR_DBL         ***Q_M;
00189 FLT_OR_DBL         ***Q_M1;
00190 FLT_OR_DBL         ***Q_M2;
00191
00192 FLT_OR_DBL         **Q_C;
00193 FLT_OR_DBL         **Q_cH;
00194 FLT_OR_DBL         **Q_cI;
00195 FLT_OR_DBL         **Q_cM;
00196
00197 int                *l_min_values;
00198 int                *l_max_values;
00199 int                *k_min_values;
00200 int                *k_max_values;
00201
00202 int                *l_min_values_b;
00203 int                *l_max_values_b;
00204 int                *k_min_values_b;
00205 int                *k_max_values_b;
00206
00207 int                *l_min_values_m;
00208 int                *l_max_values_m;
00209 int                *k_min_values_m;
00210 int                *k_max_values_m;
00211
00212 int                *l_min_values_m1;
00213 int                *l_max_values_m1;
00214 int                *k_min_values_m1;
00215 int                *k_max_values_m1;
00216
00217 int                *l_min_values_m2;
00218 int                *l_max_values_m2;
00219 int                *k_min_values_m2;
00220 int                *k_max_values_m2;
00221
00222 int                *l_min_values_qc;
00223 int                *l_max_values_qc;
00224 int                *k_min_values_qc;
00225 int                *k_max_values_qc;
00226
00227 int                *l_min_values_qcH;
00228 int                *l_max_values_qcH;
00229 int                *k_min_values_qcH;
00230 int                *k_max_values_qcH;
00231
00232 int                *l_min_values_qcI;
00233 int                *l_max_values_qcI;
00234 int                *k_min_values_qcI;
00235 int                *k_max_values_qcI;
00236
00237 int                *l_min_values_qcM;
00238 int                *l_max_values_qcM;
00239 int                *k_min_values_qcM;
00240 int                *k_max_values_qcM;
00241
00242 /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
00243 FLT_OR_DBL         *Q_rem;
00244 FLT_OR_DBL         *Q_B_rem;
00245 FLT_OR_DBL         *Q_M_rem;
00246 FLT_OR_DBL         *Q_M1_rem;
00247 FLT_OR_DBL         *Q_M2_rem;
00248
00249 FLT_OR_DBL         Q_c_rem;
00250 FLT_OR_DBL         Q_cH_rem;
00251 FLT_OR_DBL         Q_cI_rem;
00252 FLT_OR_DBL         Q_cM_rem;
00253
00254 vrna_fold_compound_t *compatibility;
00255 } TwoDpfold_vars;
00256
00275 DEPRECATED(TwoDpfold_vars *
00276             get_TwoDpfold_variables(const char *seq,
00277                                     const char *structure1,
00278                                     char *structure2,
00279                                     int circ),
00280             "Use the new API and vrna_fold_compound_TwoD() instead");
00281
00295 DEPRECATED(void
00296             destroy_TwoDpfold_variables(TwoDpfold_vars *vars),
00297             "Use the new API and vrna_fold_compound_free() instead");
00298
00323 DEPRECATED(TwoDpfold_solution *
00324             TwoDpfoldList(TwoDpfold_vars *vars,
00325                           int maxDistance1,

```

```

00326             int                maxDistance2),
00327     "Use the new API and vrna_pf_TwoD() instead");
00328
00350 DEPRECATED(char *
00351     TwoDpfold_pbacktrack(TwoDpfold_vars *vars,
00352             int                d1,
00353             int                d2),
00354     "Use the new API and vrna_pbacktrack_TwoD() instead");
00355
00379 DEPRECATED(char *
00380     TwoDpfold_pbacktrack5(TwoDpfold_vars *vars,
00381             int                d1,
00382             int                d2,
00383             unsigned int      length),
00384     "Use the new API and vrna_pbacktrack5_TwoD() instead");
00385
00391 DEPRECATED(FLT_OR_DBL **TwoDpfold(TwoDpfold_vars *our_variables,
00392             int                maxDistance1,
00393             int                maxDistance2),
00394     "Use the new API and vrna_pf_TwoD() instead");
00395
00401 DEPRECATED(FLT_OR_DBL **TwoDpfold_circ(TwoDpfold_vars *our_variables,
00402             int                maxDistance1,
00403             int                maxDistance2),
00404     "Use the new API and vrna_pf_TwoD() instead");
00405
00406 #endif
00407
00408 #endif

```

18.5 ali_plex.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_ALI_PLEX_H
00002 #define VIENNA_RNA_PACKAGE_ALI_PLEX_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005
00006 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00007
00011 duplexT **aliLduplexfold(const char *s1[],
00012     const char *s2[],
00013     const int threshold,
00014     const int extension_cost,
00015     const int alignment_length,
00016     const int delta,
00017     const int fast,
00018     const int il_a,
00019     const int il_b,
00020     const int b_a,
00021     const int b_b);
00022
00023
00027 duplexT **aliLduplexfold_XS(const char *s1[],
00028     const char *s2[],
00029     const int **access_s1,
00030     const int **access_s2,
00031     const int threshold,
00032     const int alignment_length,
00033     const int delta,
00034     const int fast,
00035     const int il_a,
00036     const int il_b,
00037     const int b_a,
00038     const int b_b);
00039
00040
00041 /*
00042  * extern duplexT aliduplexfold(const char *s1[], const char *s2[], const int extension_cost);
00043  * extern duplexT aliduplexfold_XS(const char *s1[], const char *s2[], const int **access_s1,
00044  * const int **access_s2, const int i_pos, const int j_pos, const int threshold);
00045  */
00046 #endif
00047
00048 #endif

```

18.6 ViennaRNA/alifold.h File Reference

Functions for comparative structure prediction using RNA sequence alignments.
 Include dependency graph for alifold.h:

Functions

- float [energy_of_alistruct](#) (const char **sequences, const char *structure, int n_seq, float *energy)
Calculate the free energy of a consensus structure given a set of aligned sequences.
- void [update_alifold_params](#) (void)
Update the energy parameters for alifold function.

- float [alifold](#) (const char **strings, char *structure)
Compute MFE and according consensus structure of an alignment of sequences.
- float [circularifold](#) (const char **strings, char *structure)
Compute MFE and according structure of an alignment of sequences assuming the sequences are circular instead of linear.
- void [free_alifold_arrays](#) (void)
Free the memory occupied by MFE alifold functions.

- float [alipf_fold_par](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl, [vrna_exp_param_t](#) *parameters, int calculate_bppm, int is_constrained, int is_circular)
- float [alipf_fold](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl)
The partition function version of [alifold\(\)](#) works in analogy to [pf_fold\(\)](#). Pair probabilities and information about sequence covariations are returned via the 'pi' variable as a list of [vrna_pinfo_t](#) structs. The list is terminated by the first entry with pi.i = 0.
- float [alipf_circ_fold](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl)
- [FLT_OR_DBL](#) * [export_ali_bppm](#) (void)
Get a pointer to the base pair probability array.
- void [free_alipf_arrays](#) (void)
Free the memory occupied by folding matrices allocated by [alipf_fold](#), [alipf_circ_fold](#), etc.
- char * [alipbacktrack](#) (double *prob)
Sample a consensus secondary structure from the Boltzmann ensemble according its probability.
- int [get_alipf_arrays](#) (short ***S_p, short ***S5_p, short ***S3_p, unsigned short ***a2s_p, char ***Ss↔_p, [FLT_OR_DBL](#) **qb_p, [FLT_OR_DBL](#) **qm_p, [FLT_OR_DBL](#) **q1k_p, [FLT_OR_DBL](#) **qln_p, short **pscore)
Get pointers to (almost) all relevant arrays used in alifold's partition function computation.

Variables

- double [cv_fact](#)
This variable controls the weight of the covariance term in the energy function of alignment folding algorithms.
- double [nc_fact](#)
This variable controls the magnitude of the penalty for non-compatible sequences in the covariance term of alignment folding algorithms.

18.6.1 Detailed Description

Functions for comparative structure prediction using RNA sequence alignments.

18.6.2 Function Documentation

18.6.2.1 energy_of_alistruct()

```
float energy_of_alistruct (
    const char ** sequences,
    const char * structure,
    int n_seq,
    float * energy )
```

Calculate the free energy of a consensus structure given a set of aligned sequences.

Deprecated Usage of this function is discouraged! Use [vrna_eval_structure\(\)](#), and [vrna_eval_covar_structure\(\)](#) instead!

Parameters

<i>sequences</i>	The NULL terminated array of sequences
<i>structure</i>	The consensus structure
<i>n_seq</i>	The number of sequences in the alignment
<i>energy</i>	A pointer to an array of at least two floats that will hold the free energies (energy[0] will contain the free energy, energy[1] will be filled with the covariance energy term)

Returns

free energy in kcal/mol

18.6.2.2 update_alifold_params()

```
void update_alifold_params (
    void )
```

Update the energy parameters for alifold function.

Call this to recalculate the pair matrix and energy parameters after a change in folding parameters like [temperature](#)

Deprecated Usage of this function is discouraged! The new API uses [vrna_fold_compound_t](#) to lump all folding related necessities together, including the energy parameters. Use [vrna_update_fold_params\(\)](#) to update the energy parameters within a [vrna_fold_compound_t](#).

18.6.3 Variable Documentation

18.6.3.1 cv_fact

```
double cv_fact [extern]
```

This variable controls the weight of the covariance term in the energy function of alignment folding algorithms.

Deprecated See [vrna_md_t.cv_fact](#), and [vrna_mfe\(\)](#) to avoid using global variables

Default is 1.

18.6.3.2 nc_fact

```
double nc_fact [extern]
```

This variable controls the magnitude of the penalty for non-compatible sequences in the covariance term of alignment folding algorithms.

Deprecated See [vrna_md_t.nc_fact](#), and [vrna_mfe\(\)](#) to avoid using global variables

Default is 1.

18.7 alifold.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_ALIFOLD_H
00002 #define VIENNA_RNA_PACKAGE_ALIFOLD_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/params/basic.h>
00006 #include <ViennaRNA/ribo.h>
00007 #include <ViennaRNA/mfe.h>
00008 #include <ViennaRNA/part_func.h>
00009 #include <ViennaRNA/utils/alignments.h>
00010 #include <ViennaRNA/utils/structures.h>
00011 #include <ViennaRNA/boltzmann_sampling.h>
00012
00013 #ifdef VRNA_WARN_DEPRECATED
00014 # if defined(__clang__)
00015 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00016 # elif defined(__GNUC__)
00017 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00018 # else
00019 #  define DEPRECATED(func, msg) func
00020 # endif
00021 #else
00022 # define DEPRECATED(func, msg) func
00023 #endif
00024
00032 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00033
00034 /*
00035 #####
00036 # DEPRECATED FUNCTIONS #
00037 #####
00038 */
00039
00063 DEPRECATED(float alifold( const char **strings, char *structure),
00064             "Use vrna_alifold() or vrna_mfe() instead");
00065
00077 DEPRECATED(float circalifold( const char **strings, char *structure),
00078             "Use vrna_alicircfold() or vrna_mfe() instead");
00079
00091 DEPRECATED(void free_alifold_arrays(void),
00092             "This function is obsolete");
00093
00094 /* End group mfe_global_deprecated */
00112 DEPRECATED(float energy_of_alistruct(const char **sequences, const char *structure, int n_seq, float
*energy),
00113             "Use vrna_eval_structure() and vrna_eval_covar_structure() instead");
00114
00115 DEPRECATED(float energy_of_ali_gquad_structure(const char **sequences, const char *structure, int
n_seq, float *energy),
00116             "Use vrna_eval_structure() and vrna_eval_covar_structure() instead");
00117
00128 DEPRECATED(extern double cv_fact,
00129             "Use the cv_fact attribute of the vrna_md_t datastructure instead");
00140 DEPRECATED(extern double nc_fact,
00141             "Use the nc_fact attribute of the vrna_md_t datastructure instead");
00142
00162 DEPRECATED(float alipf_fold_par( const char **sequences,
00163                                   char *structure,
00164                                   vrna_ep_t **pl,
00165                                   vrna_exp_param_t *parameters,
00166                                   int calculate_bppm,
00167                                   int is_constrained,
00168                                   int is_circular),
00169             "Use vrna_pf_alifold() or vrna_pf() instead");
00170
00187 DEPRECATED(float alipf_fold( const char **sequences, char *structure, vrna_ep_t **pl),
00188             "Use vrna_pf_alifold() or vrna_pf() instead");
00189
00200 DEPRECATED(float alipf_circ_fold(const char **sequences, char *structure, vrna_ep_t **pl),
00201             "Use vrna_pf_circalifold() or vrna_pf() instead");
00202
00203
00220 DEPRECATED(FLT_OR_DBL *export_ali_bppm(void),
00221             "Use the new API with vrna_fold_compound_t datastructure instead");
00222
00233 DEPRECATED(void free_alipf_arrays(void),
00234             "This function is obsolete");
00235
00244 DEPRECATED(char *alipbacktrack(double *prob),
00245             "Use the new API and vrna_pbacktrack() instead");
00246
00271 DEPRECATED(int get_alipf_arrays(short ***S_p,
00272                                   short ***S5_p,
00273                                   short ***S3_p,
```

```

00274         unsigned short ***a2s_p,
00275         char ***Ss_p,
00276         FLT_OR_DBL **qb_p,
00277         FLT_OR_DBL **qm_p,
00278         FLT_OR_DBL **qlk_p,
00279         FLT_OR_DBL **qln_p,
00280         short **pscore),
00281     "Use the new API with vrna_fold_compound_t datastructure instead");
00282
00283
00284 /* End group part_func_global_deprecated */
00298 DEPRECATED(void update_alifold_params(void),
00299     "Use the new API with vrna_fold_compound_t datastructure instead");
00300
00301 #endif
00302
00303
00304 #endif

```

18.8 ViennaRNA/aln_util.h File Reference

Use [ViennaRNA/utis/alignments.h](#) instead.

Include dependency graph for aln_util.h:

18.8.1 Detailed Description

Use [ViennaRNA/utis/alignments.h](#) instead.

Deprecated Use [ViennaRNA/utis/alignments.h](#) instead

18.9 aln_util.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_ALN_UTILS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_ALN_UTILS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/aln_util.h>! Use <ViennaRNA/utis/alignments.h>
00013         instead!"
00013 # endif
00014 #include <ViennaRNA/utis/alignments.h>
00015 #endif
00016
00017 #endif

```

18.10 ViennaRNA/alphabet.h File Reference

Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.

Include dependency graph for alphabet.h: This graph shows which files directly or indirectly include this file:

Functions

- char * [vrna_ptypes](#) (const short *S, [vrna_md_t](#) *md)

Get an array of the numerical encoding for each possible base pair (i,j)
- short * [vrna_seq_encode](#) (const char *sequence, [vrna_md_t](#) *md)

Get a numerical representation of the nucleotide sequence.
- short * [vrna_seq_encode_simple](#) (const char *sequence, [vrna_md_t](#) *md)

Get a numerical representation of the nucleotide sequence (simple version)
- int [vrna_nucleotide_encode](#) (char c, [vrna_md_t](#) *md)

Encode a nucleotide character to numerical value.
- char [vrna_nucleotide_decode](#) (int enc, [vrna_md_t](#) *md)

Decode a numerical representation of a nucleotide back into nucleotide alphabet.

18.10.1 Detailed Description

Functions to process, convert, and generally handle different nucleotide and/or base pair alphabets.

,

18.11 alphabet.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_ALPHABET_H
00002 #define VIENNA_RNA_PACKAGE_ALPHABET_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00029 #include <ViennaRNA/fold_compound.h>
00030 #include <ViennaRNA/model.h>
00031
00032 unsigned int
00033 vrna_sequence_length_max(unsigned int options);
00034
00035
00036 int
00037 vrna_nucleotide_IUPAC_identity(char a,
00038                               char b);
00039
00040
00041 void
00042 vrna_ptypes_prepare(vrna_fold_compound_t *fc,
00043                   unsigned int options);
00044
00045
00055 char *
00056 vrna_ptypes(const short *S,
00057            vrna_md_t *md);
00058
00059
00067 short *
00068 vrna_seq_encode(const char *sequence,
00069               vrna_md_t *md);
00070
00071
00076 short *
00077 vrna_seq_encode_simple(const char *sequence,
00078                    vrna_md_t *md);
00079
00080
00092 int
00093 vrna_nucleotide_encode(char c,
00094                      vrna_md_t *md);
00095
00096
00108 char
00109 vrna_nucleotide_decode(int enc,
00110                      vrna_md_t *md);
00111
00112
00113 void
00114 vrna_aln_encode(const char *sequence,
00115               short **S_p,
00116               short **S5_p,
00117               short **S3_p,
00118               char **SS_p,
00119               unsigned int **as_p,
00120               vrna_md_t *md);
00121
00122
00123 unsigned int
00124 vrna_get_ptype_md(int i,
00125                 int j,
00126                 vrna_md_t *md);
00127
00128
00129 unsigned int
00130 vrna_get_ptype(int ij,
```

```

00131         char *ptype);
00132
00133
00134 unsigned int
00135 vrna_get_ptype_window(int i,
00136                      int j,
00137                      char **ptype);
00138
00139
00140 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00141
00142 DEPRECATED(char *get_ptypes(const short *S,
00143                          vrna_md_t *md,
00144                          unsigned int idx_type),
00145           "Use vrna_pytypes() instead");
00146
00147 #endif
00148
00149 #endif
00150
00151 #endif
00152
00153 #endif

```

18.12 ViennaRNA/boltzmann_sampling.h File Reference

Boltzmann Sampling of secondary structures from the ensemble.

Include dependency graph for boltzmann_sampling.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_PBACKTRACK_DEFAULT 0`
Boltzmann sampling flag indicating default backtracing mode.
- `#define VRNA_PBACKTRACK_NON_REDUNDANT 1`
Boltzmann sampling flag indicating non-redundant backtracing mode.

Typedefs

- `typedef void(* vrna_bs_result_f) (const char *structure, void *data)`
Callback for Boltzmann sampling.
- `typedef struct vrna_pbacktrack_memory_s * vrna_pbacktrack_mem_t`
Boltzmann sampling memory data structure.

Functions

- `char * vrna_pbacktrack5 (vrna_fold_compound_t *fc, unsigned int length)`
Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.
- `char ** vrna_pbacktrack5_num (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, unsigned int options)`
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- `unsigned int vrna_pbacktrack5_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_bs_result_f cb, void *data, unsigned int options)`
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- `char ** vrna_pbacktrack5_resume (vrna_fold_compound_t *vc, unsigned int num_samples, unsigned int length, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- `unsigned int vrna_pbacktrack5_resume_cb (vrna_fold_compound_t *fc, unsigned int num_samples, unsigned int length, vrna_bs_result_f cb, void *data, vrna_pbacktrack_mem_t *nr_mem, unsigned int options)`
Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.
- `char * vrna_pbacktrack (vrna_fold_compound_t *fc)`

- Sample a secondary structure from the Boltzmann ensemble according its probability.*

 - char ** [vrna_pbacktrack_num](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int options)

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

 - unsigned int [vrna_pbacktrack_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, [vrna_bs_result_f](#) cb, void *data, unsigned int options)

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

 - char ** [vrna_pbacktrack_resume](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

 - unsigned int [vrna_pbacktrack_resume_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, [vrna_bs_result_f](#) cb, void *data, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)

Obtain a set of secondary structure samples from the Boltzmann ensemble according their probability.

 - char * [vrna_pbacktrack_sub](#) ([vrna_fold_compound_t](#) *fc, unsigned int start, unsigned int end)

Sample a secondary structure of a subsequence from the Boltzmann ensemble according its probability.

 - char ** [vrna_pbacktrack_sub_num](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int start, unsigned int end, unsigned int options)

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

 - unsigned int [vrna_pbacktrack_sub_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int start, unsigned int end, [vrna_bs_result_f](#) cb, void *data, unsigned int options)

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

 - char ** [vrna_pbacktrack_sub_resume](#) ([vrna_fold_compound_t](#) *vc, unsigned int num_samples, unsigned int start, unsigned int end, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

 - unsigned int [vrna_pbacktrack_sub_resume_cb](#) ([vrna_fold_compound_t](#) *fc, unsigned int num_samples, unsigned int start, unsigned int end, [vrna_bs_result_f](#) cb, void *data, [vrna_pbacktrack_mem_t](#) *nr_mem, unsigned int options)

Obtain a set of secondary structure samples for a subsequence from the Boltzmann ensemble according their probability.

 - void [vrna_pbacktrack_mem_free](#) ([vrna_pbacktrack_mem_t](#) s)

Release memory occupied by a Boltzmann sampling memory data structure.

18.12.1 Detailed Description

Boltzmann Sampling of secondary structures from the ensemble.

A.k.a. Stochastic backtracking

18.13 `boltzmann_sampling.h`

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_BOLTZMANN_SAMPLING_H
00002 #define VIENNA_RNA_PACKAGE_BOLTZMANN_SAMPLING_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)
00009 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00010 # elif defined(__GNUC__)
00011 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00012 # else
00013 #   define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
00018
00043 #define VRNA_PBACKTRACK_DEFAULT 0
```

```

00044
00055 #define VRNA_PBACKTRACK_NON_REDUNDANT 1
00056
00072 typedef void (*vrna_bs_result_f)(const char *structure,
00073                                 void *data);
00074
00075 DEPRECATED(typedef void (vrna_boltzmann_sampling_callback)(const char *structure,
00076                                                           void *data),
00077            "Use vrna_bs_result_f instead!");
00078
00079
00095 typedef struct vrna_pbacktrack_memory_s *vrna_pbacktrack_mem_t;
00096
00097 #include <ViennaRNA/fold_compound.h>
00098
00128 char *
00129 vrna_pbacktrack5(vrna_fold_compound_t *fc,
00130                 unsigned int length);
00131
00132
00177 char **
00178 vrna_pbacktrack5_num(vrna_fold_compound_t *fc,
00179                     unsigned int num_samples,
00180                     unsigned int length,
00181                     unsigned int options);
00182
00183
00233 unsigned int
00234 vrna_pbacktrack5_cb(vrna_fold_compound_t *fc,
00235                    unsigned int num_samples,
00236                    unsigned int length,
00237                    vrna_bs_result_f cb,
00238                    void *data,
00239                    unsigned int options);
00240
00241
00316 char **
00317 vrna_pbacktrack5_resume(vrna_fold_compound_t *vc,
00318                        unsigned int num_samples,
00319                        unsigned int length,
00320                        vrna_pbacktrack_mem_t *nr_mem,
00321                        unsigned int options);
00322
00323
00401 unsigned int
00402 vrna_pbacktrack5_resume_cb(vrna_fold_compound_t *fc,
00403                           unsigned int num_samples,
00404                           unsigned int length,
00405                           vrna_bs_result_f cb,
00406                           void *data,
00407                           vrna_pbacktrack_mem_t *nr_mem,
00408                           unsigned int options);
00409
00410
00438 char *
00439 vrna_pbacktrack(vrna_fold_compound_t *fc);
00440
00441
00484 char **
00485 vrna_pbacktrack_num(vrna_fold_compound_t *fc,
00486                    unsigned int num_samples,
00487                    unsigned int options);
00488
00489
00537 unsigned int
00538 vrna_pbacktrack_cb(vrna_fold_compound_t *fc,
00539                   unsigned int num_samples,
00540                   vrna_bs_result_f cb,
00541                   void *data,
00542                   unsigned int options);
00543
00544
00615 char **
00616 vrna_pbacktrack_resume(vrna_fold_compound_t *fc,
00617                       unsigned int num_samples,
00618                       vrna_pbacktrack_mem_t *nr_mem,
00619                       unsigned int options);
00620
00621
00695 unsigned int
00696 vrna_pbacktrack_resume_cb(vrna_fold_compound_t *fc,
00697                          unsigned int num_samples,
00698                          vrna_bs_result_f cb,
00699                          void *data,
00700                          vrna_pbacktrack_mem_t *nr_mem,
00701                          unsigned int options);
00702

```

```

00703
00704
00705
00706
00707
00708
00709
00742 char *
00743 vrna_pbacktrack_sub(vrna_fold_compound_t *fc,
00744                     unsigned int start,
00745                     unsigned int end);
00746
00747
00793 char **
00794 vrna_pbacktrack_sub_num(vrna_fold_compound_t *fc,
00795                         unsigned int num_samples,
00796                         unsigned int start,
00797                         unsigned int end,
00798                         unsigned int options);
00799
00800
00851 unsigned int
00852 vrna_pbacktrack_sub_cb(vrna_fold_compound_t *fc,
00853                       unsigned int num_samples,
00854                       unsigned int start,
00855                       unsigned int end,
00856                       vrna_bs_result_f cb,
00857                       void *data,
00858                       unsigned int options);
00859
00860
00936 char **
00937 vrna_pbacktrack_sub_resume(vrna_fold_compound_t *vc,
00938                           unsigned int num_samples,
00939                           unsigned int start,
00940                           unsigned int end,
00941                           vrna_pbacktrack_mem_t *nr_mem,
00942                           unsigned int options);
00943
00944
01023 unsigned int
01024 vrna_pbacktrack_sub_resume_cb(vrna_fold_compound_t *fc,
01025                              unsigned int num_samples,
01026                              unsigned int start,
01027                              unsigned int end,
01028                              vrna_bs_result_f cb,
01029                              void *data,
01030                              vrna_pbacktrack_mem_t *nr_mem,
01031                              unsigned int options);
01032
01033
01042 void
01043 vrna_pbacktrack_mem_free(vrna_pbacktrack_mem_t s);
01044
01045
01049 #endif

```

18.14 ViennaRNA/centroid.h File Reference

Centroid structure computation.

Include dependency graph for centroid.h: This graph shows which files directly or indirectly include this file:

Functions

- char * [vrna_centroid](#) (vrna_fold_compound_t *vc, double *dist)
Get the centroid structure of the ensemble.
- char * [vrna_centroid_from_plist](#) (int length, double *dist, vrna_ep_t *pl)
Get the centroid structure of the ensemble.
- char * [vrna_centroid_from_probs](#) (int length, double *dist, FLT_OR_DBL *probs)
Get the centroid structure of the ensemble.
- char * [get_centroid_struct_pl](#) (int length, double *dist, vrna_ep_t *pl)
Get the centroid structure of the ensemble.
- char * [get_centroid_struct_pr](#) (int length, double *dist, FLT_OR_DBL *pr)
Get the centroid structure of the ensemble.

18.14.1 Detailed Description

Centroid structure computation.

18.14.2 Function Documentation

18.14.2.1 `get_centroid_struct_pl()`

```
char * get_centroid_struct_pl (
    int length,
    double * dist,
    vrna_ep_t * pl )
```

Get the centroid structure of the ensemble.

Deprecated This function was renamed to `vrna_centroid_from_plist()`

18.14.2.2 `get_centroid_struct_pr()`

```
char * get_centroid_struct_pr (
    int length,
    double * dist,
    FLT_OR_DBL * pr )
```

Get the centroid structure of the ensemble.

Deprecated This function was renamed to `vrna_centroid_from_probs()`

18.15 centroid.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CENTROID_H
00002 #define VIENNA_RNA_PACKAGE_CENTROID_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/fold_compound.h>
00006 #include <ViennaRNA/utils/structures.h>
00007
00008 #ifdef VRNA_WARN_DEPRECATED
00009 # if defined(__clang__)
00010 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00011 # elif defined(__GNUC__)
00012 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00013 # else
00014 #  define DEPRECATED(func, msg) func
00015 # endif
00016 #else
00017 # define DEPRECATED(func, msg) func
00018 #endif
00019
00039 char *
00040 vrna_centroid(vrna_fold_compound_t *vc,
00041              double *dist);
00042
00043
00060 char *
00061 vrna_centroid_from_plist(int length,
00062                          double *dist,
00063                          vrna_ep_t *pl);
00064
00065
00082 char *
00083 vrna_centroid_from_probs(int length,
00084                          double *dist,
00085                          FLT_OR_DBL *probs);
00086
00087
00088 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00089
00095 DEPRECATED(char *get_centroid_struct_pl(int length,
```

```

00096             double      *dist,
00097             vrna_ep_t *pl),
00098     "Use vrna_centroid_from_plist() instead");
00099
00105 DEPRECATED(char *get_centroid_struct_pr(int      length,
00106             double      *dist,
00107             FLT_OR_DBL *pr),
00108     "Use vrna_centroid_from_probs() instead");
00109
00110 #endif
00111
00112 #endif

```

18.16 ViennaRNA/char_stream.h File Reference

Use [ViennaRNA/datastructures/char_stream.h](#) instead.

Include dependency graph for char_stream.h:

18.16.1 Detailed Description

Use [ViennaRNA/datastructures/char_stream.h](#) instead.

Deprecated Use [ViennaRNA/datastructures/char_stream.h](#) instead

18.17 char_stream.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_CHAR_STREAM_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_CHAR_STREAM_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/char_stream.h>! Use
00013         <ViennaRNA/datastructures/char_stream.h> instead!"
00013 # endif
00014 #include <ViennaRNA/datastructures/char_stream.h>
00015 #endif
00016
00017 #endif

```

18.18 ViennaRNA/datastructures/char_stream.h File Reference

Implementation of a dynamic, buffered character stream.

Include dependency graph for char_stream.h: This graph shows which files directly or indirectly include this file:

Functions

- `vrna_cstr_t vrna_cstr` (size_t size, FILE *output)
*Create a dynamic char * stream data structure.*
- `void vrna_cstr_discard` (struct vrna_cstr_s *buf)
*Discard the current content of the dynamic char * stream data structure.*
- `void vrna_cstr_free` (vrna_cstr_t buf)
*Free the memory occupied by a dynamic char * stream data structure.*
- `void vrna_cstr_close` (vrna_cstr_t buf)
*Free the memory occupied by a dynamic char * stream and close the output stream.*
- `void vrna_cstr_fflush` (struct vrna_cstr_s *buf)
*Flush the dynamic char * output stream.*

18.18.1 Detailed Description

Implementation of a dynamic, buffered character stream.

,

18.19 char_stream.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_CHAR_STREAM_H
00002 #define VIENNA_RNA_PACKAGE_CHAR_STREAM_H
00003
00016 #include <stdarg.h>
00017 #include <stdio.h>
00018
00019 /* below is our own implementation of a dynamic char * stream */
00020 typedef struct vrna_cstr_s *vrna_cstr_t;
00021
00030 vrna_cstr_t
00031 vrna_cstr(size_t size,
00032          FILE *output);
00033
00034
00042 void
00043 vrna_cstr_discard(struct vrna_cstr_s *buf);
00044
00045
00056 void
00057 vrna_cstr_free(vrna_cstr_t buf);
00058
00059
00071 void
00072 vrna_cstr_close(vrna_cstr_t buf);
00073
00074
00088 void
00089 vrna_cstr_fflush(struct vrna_cstr_s *buf);
00090
00091
00092 const char *
00093 vrna_cstr_string(vrna_cstr_t buf);
00094
00095
00096 int
00097 vrna_cstr_vprintf(vrna_cstr_t buf,
00098                  const char *format,
00099                  va_list args);
00100
00101
00102 int
00103 vrna_cstr_printf(vrna_cstr_t buf,
00104                  const char *format,
00105                  ...);
00106
00107
00108 void
00109 vrna_cstr_message_info(vrna_cstr_t buf,
00110                       const char *format,
00111                       ...);
00112
00113
00114 void
00115 vrna_cstr_message_vinfo(vrna_cstr_t buf,
00116                        const char *format,
00117                        va_list args);
00118
00119
00120 void
00121 vrna_cstr_message_warning(struct vrna_cstr_s *buf,
00122                          const char *format,
00123                          ...);
00124
00125
00126 void
00127 vrna_cstr_message_vwarning(struct vrna_cstr_s *buf,
00128                           const char *format,
00129                           va_list args);
00130
00131
00132 void
00133 vrna_cstr_print_fasta_header(vrna_cstr_t buf,
00134                             const char *head);
00135
00136
00137 void
00138 vrna_cstr_printf_structure(struct vrna_cstr_s *buf,
00139                           const char *structure,
00140                           const char *format,
00141                           ...);
00142
00143
00144 void

```

```
00145 vrna_cstr_vprintf_structure(struct vrna_cstr_s *buf,
00146                             const char *structure,
00147                             const char *format,
00148                             va_list args);
00149
00150
00151 void
00152 vrna_cstr_printf_comment(struct vrna_cstr_s *buf,
00153                         const char *format,
00154                         ...);
00155
00156
00157 void
00158 vrna_cstr_vprintf_comment(struct vrna_cstr_s *buf,
00159                          const char *format,
00160                          va_list args);
00161
00162
00163 void
00164 vrna_cstr_printf_thead(struct vrna_cstr_s *buf,
00165                       const char *format,
00166                       ...);
00167
00168
00169 void
00170 vrna_cstr_vprintf_thead(struct vrna_cstr_s *buf,
00171                        const char *format,
00172                        va_list args);
00173
00174
00175 void
00176 vrna_cstr_printf_tbody(struct vrna_cstr_s *buf,
00177                       const char *format,
00178                       ...);
00179
00180
00181 void
00182 vrna_cstr_vprintf_tbody(struct vrna_cstr_s *buf,
00183                        const char *format,
00184                        va_list args);
00185
00186
00187 void
00188 vrna_cstr_print_eval_sd_corr(struct vrna_cstr_s *buf);
00189
00190
00191 void
00192 vrna_cstr_print_eval_ext_loop(struct vrna_cstr_s *buf,
00193                              int energy);
00194
00195
00196 void
00197 vrna_cstr_print_eval_hp_loop(struct vrna_cstr_s *buf,
00198                             int i,
00199                             int j,
00200                             char si,
00201                             char sj,
00202                             int energy);
00203
00204
00205 void
00206 vrna_cstr_print_eval_hp_loop_revert(struct vrna_cstr_s *buf,
00207                                    int i,
00208                                    int j,
00209                                    char si,
00210                                    char sj,
00211                                    int energy);
00212
00213
00214 void
00215 vrna_cstr_print_eval_int_loop(struct vrna_cstr_s *buf,
00216                              int i,
00217                              int j,
00218                              char si,
00219                              char sj,
00220                              int k,
00221                              int l,
00222                              char sk,
00223                              char sl,
00224                              int energy);
00225
00226
00227 void
00228 vrna_cstr_print_eval_int_loop_revert(struct vrna_cstr_s *buf,
00229                                     int i,
00230                                     int j,
00231                                     char si,
```

```

00232             char          sj,
00233             int           k,
00234             int           l,
00235             char          sk,
00236             char          sl,
00237             int           energy);
00238
00239 void
00240 vrna_cstr_print_eval_mb_loop(struct vrna_cstr_s *buf,
00241                             int           i,
00242                             int           j,
00243                             char          si,
00244                             char          sj,
00245                             int           energy);
00246
00247 void
00248 vrna_cstr_print_eval_mb_loop_revert(struct vrna_cstr_s *buf,
00249                                    int           i,
00250                                    int           j,
00251                                    char          si,
00252                                    char          sj,
00253                                    int           energy);
00254
00255 void
00256 vrna_cstr_print_eval_gquad(struct vrna_cstr_s *buf,
00257                           int           i,
00258                           int           L,
00259                           int           l[3],
00260                           int           energy);
00261
00262 #endif

```

18.20 ViennaRNA/cofold.h File Reference

MFE implementations for RNA-RNA interaction.

Include dependency graph for cofold.h:

Functions

- float [cofold](#) (const char *sequence, char *structure)
Compute the minimum free energy of two interacting RNA molecules.
- float [cofold_par](#) (const char *string, char *structure, [vrna_param_t](#) *parameters, int is_constrained)
Compute the minimum free energy of two interacting RNA molecules.
- void [free_co_arrays](#) (void)
Free memory occupied by [cofold\(\)](#)
- void [update_cofold_params](#) (void)
Recalculate parameters.
- void [update_cofold_params_par](#) ([vrna_param_t](#) *parameters)
Recalculate parameters.
- void [export_cofold_arrays_gq](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **fc_p, int **ggg_p, int **indx_p, char **ptype_p)
Export the arrays of partition function cofold (with gquadruplex support)
- void [export_cofold_arrays](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **fc_p, int **indx_p, char **ptype_p)
Export the arrays of partition function cofold.
- void [initialize_cofold](#) (int length)

18.20.1 Detailed Description

MFE implementations for RNA-RNA interaction.

18.21 cofold.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_COFOLD_H
00002 #define VIENNA_RNA_PACKAGE_COFOLD_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/params/basic.h>
00006 #include <ViennaRNA/mfe.h>
00007
00008 #ifdef VRNA_WARN_DEPRECATED
00009 # if defined(__clang__)
00010 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00011 # elif defined(__GNUC__)
00012 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00013 # else
00014 #  define DEPRECATED(func, msg) func
00015 # endif
00016 #else
00017 # define DEPRECATED(func, msg) func
00018 #endif
00019
00026 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00027
00042 DEPRECATED(float
00043             cofold(const char *sequence,
00044                   char *structure),
00045             "Use vrna_cofold() instead");
00046
00054 DEPRECATED(float
00055             cofold_par(const char *string,
00056                       char *structure,
00057                       vrna_param_t *parameters,
00058                       int is_constrained),
00059             "Use the new API and vrna_mfe_dimer() instead");
00060
00072 DEPRECATED(void
00073             free_co_arrays(void),
00074             "This function is obsolete");
00075
00082 DEPRECATED(void
00083             update_cofold_params(void),
00084             "This function is obsolete");
00085
00092 DEPRECATED(void
00093             update_cofold_params_par(vrna_param_t *parameters),
00094             "Use the new API with vrna_fold_compound_t instead");
00095
00096
00118 DEPRECATED(void
00119             export_cofold_arrays_gg(int **f5_p,
00120                                     int **c_p,
00121                                     int **fML_p,
00122                                     int **fML1_p,
00123                                     int **fc_p,
00124                                     int **ggg_p,
00125                                     int **indx_p,
00126                                     char **ptype_p),
00127             "Use the new API with vrna_fold_compound_t instead");
00128
00149 DEPRECATED(void
00150             export_cofold_arrays(int **f5_p,
00151                                 int **c_p,
00152                                 int **fML_p,
00153                                 int **fML1_p,
00154                                 int **fc_p,
00155                                 int **indx_p,
00156                                 char **ptype_p),
00157             "Use the new API with vrna_fold_compound_t instead");
00158
00159
00166 DEPRECATED(void
00167             initialize_cofold(int length),
00168             "This function is obsolete");
00169
00170 #endif
00171
00172 #endif

```

18.22 ViennaRNA/combinatorics.h File Reference

Various implementations that deal with combinatorial aspects of objects.
 Include dependency graph for combinatorics.h:

Functions

- unsigned int **[vrna_enumerate_necklaces](#)** (const unsigned int *type_counts)
Enumerate all necklaces with fixed content.
- unsigned int **[vrna_rotational_symmetry_num](#)** (const unsigned int *string, size_t string_length)
Determine the order of rotational symmetry for a string of objects represented by natural numbers.
- unsigned int **[vrna_rotational_symmetry_pos_num](#)** (const unsigned int *string, size_t string_length, unsigned int **positions)
Determine the order of rotational symmetry for a string of objects represented by natural numbers.
- unsigned int **[vrna_rotational_symmetry](#)** (const char *string)
Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.
- unsigned int **[vrna_rotational_symmetry_pos](#)** (const char *string, unsigned int **positions)
Determine the order of rotational symmetry for a NULL-terminated string of ASCII characters.
- unsigned int **[vrna_rotational_symmetry_db](#)** (**[vrna_fold_compound_t](#)** *fc, const char *structure)
Determine the order of rotational symmetry for a dot-bracket structure.
- unsigned int **[vrna_rotational_symmetry_db_pos](#)** (**[vrna_fold_compound_t](#)** *fc, const char *structure, unsigned int **positions)
Determine the order of rotational symmetry for a dot-bracket structure.
- unsigned int **[vrna_n_multichoose_k](#)** (size_t n, size_t k)
Obtain a list of k-combinations with repetition (n multichoose k)
- unsigned int * **[vrna_boustrophedon](#)** (size_t start, size_t end)
Generate a sequence of Boustrophedon distributed numbers.
- unsigned int **[vrna_boustrophedon_pos](#)** (size_t start, size_t end, size_t pos)
Obtain the i-th element in a Boustrophedon distributed interval of natural numbers.

18.22.1 Detailed Description

Various implementations that deal with combinatorial aspects of objects.

18.23 combinatorics.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_COMBINATORICS_H
00002 #define VIENNA_RNA_PACKAGE_COMBINATORICS_H
00003
00010 #include <ViennaRNA/fold_compound.h>
00011
00033 unsigned int **
00034 vrna_enumerate_necklaces(const unsigned int *type_counts);
00035
00036
00056 unsigned int
00057 vrna_rotational_symmetry_num(const unsigned int *string,
00058                             size_t string_length);
00059
00060
00087 unsigned int
00088 vrna_rotational_symmetry_pos_num(const unsigned int *string,
00089                                 size_t string_length,
00090                                 unsigned int **positions);
00091
00092
00110 unsigned int
00111 vrna_rotational_symmetry(const char *string);
00112
00113
00138 unsigned int
00139 vrna_rotational_symmetry_pos(const char *string,
00140                             unsigned int **positions);
00141
00142
00165 unsigned int
00166 vrna_rotational_symmetry_db(vrna_fold_compound_t *fc,
00167                             const char *structure);
00168
```

```

00169
00200 unsigned int
00201 vrna_rotational_symmetry_db_pos(vrna_fold_compound_t *fc,
00202                                 const char *structure,
00203                                 unsigned int **positions);
00204
00205
00218 unsigned int **
00219 vrna_n_multichoose_k(size_t n,
00220                     size_t k);
00221
00222
00245 unsigned int *
00246 vrna_boustrophedon(size_t start,
00247                   size_t end);
00248
00249
00261 unsigned int
00262 vrna_boustrophedon_pos(size_t start,
00263                       size_t end,
00264                       size_t pos);
00265
00266
00270 #endif

```

18.24 ViennaRNA/commands.h File Reference

Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation. Include dependency graph for commands.h:

Macros

- `#define VRNA_CMD_PARSE_HC 1U`
Command parse/apply flag indicating hard constraints.
- `#define VRNA_CMD_PARSE_SC 2U`
Command parse/apply flag indicating soft constraints.
- `#define VRNA_CMD_PARSE_UD 4U`
Command parse/apply flag indicating unstructured domains.
- `#define VRNA_CMD_PARSE_SD 8U`
Command parse/apply flag indicating structured domains.
- `#define VRNA_CMD_PARSE_DEFAULTS`
Command parse/apply flag indicating default set of commands.

Typedefs

- `typedef struct vrna_command_s * vrna_cmd_t`
A data structure that contains commands.

Functions

- `vrna_cmd_t vrna_file_commands_read` (const char *filename, unsigned int options)
Extract a list of commands from a command file.
- `int vrna_file_commands_apply` (vrna_fold_compound_t *vc, const char *filename, unsigned int options)
Apply a list of commands from a command file.
- `int vrna_commands_apply` (vrna_fold_compound_t *vc, vrna_cmd_t commands, unsigned int options)
Apply a list of commands to a [vrna_fold_compound_t](#).
- `void vrna_commands_free` (vrna_cmd_t commands)
Free memory occupied by a list of commands.

18.24.1 Detailed Description

Parse and apply different commands that alter the behavior of secondary structure prediction and evaluation.

, ,

18.25 commands.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_COMMANDS_H
00002 #define VIENNA_RNA_PACKAGE_COMMANDS_H
00003
00018 typedef struct vrna_command_s *vrna_cmd_t;
00019
00020
00021 #include <ViennaRNA/fold_compound.h>
00022
00027 #define VRNA_CMD_PARSE_HC      1U
00032 #define VRNA_CMD_PARSE_SC      2U
00037 #define VRNA_CMD_PARSE_UD      4U
00042 #define VRNA_CMD_PARSE_SD      8U
00047 #define VRNA_CMD_PARSE_DEFAULTS (VRNA_CMD_PARSE_HC \
00048                                   | VRNA_CMD_PARSE_SC \
00049                                   | VRNA_CMD_PARSE_UD \
00050                                   | VRNA_CMD_PARSE_SD \
00051                                   )
00052
00053 #define VRNA_CMD_PARSE_SILENT   16U
00054
00068 vrna_cmd_t
00069 vrna_file_commands_read(const char *filename,
00070                        unsigned int options);
00071
00072
00086 int
00087 vrna_file_commands_apply(vrna_fold_compound_t *vc,
00088                        const char *filename,
00089                        unsigned int options);
00090
00091
00100 int
00101 vrna_commands_apply(vrna_fold_compound_t *vc,
00102                   vrna_cmd_t commands,
00103                   unsigned int options);
00104
00105
00112 void
00113 vrna_commands_free(vrna_cmd_t commands);
00114
00115
00120 #endif

```

18.26 ViennaRNA/concentrations.h File Reference

Concentration computations for RNA-RNA interactions.

Include dependency graph for concentrations.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_dimer_conc_s](#)
Data structure for concentration dependency computations.

Functions

- [vrna_dimer_conc_t * get_concentrations](#) (double FEAB, double FEAA, double FEBB, double FEA, double FEB, double *startconc)
Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.
- typedef struct [vrna_dimer_conc_s](#) [vrna_dimer_conc_t](#)
Typename for the data structure that stores the dimer concentrations, [vrna_dimer_conc_s](#), as required by [vrna_pf←_dimer_concentration\(\)](#)
- typedef struct [vrna_dimer_conc_s](#) [ConcEnt](#)
Backward compatibility typedef for [vrna_dimer_conc_s](#).
- [vrna_dimer_conc_t * vrna_pf_dimer_concentrations](#) (double FcAB, double FcAA, double FcBB, double FEA, double FEB, const double *startconc, const [vrna_exp_param_t](#) *exp_params)
Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.

18.26.1 Detailed Description

Concentration computations for RNA-RNA interactions.

18.26.2 Function Documentation

18.26.2.1 `get_concentrations()`

```
vrna_dimer_conc_t * get_concentrations (
    double FEAB,
    double FEAA,
    double FEBB,
    double FEA,
    double FEB,
    double * startconc )
```

Given two start monomer concentrations a and b, compute the concentrations in thermodynamic equilibrium of all dimers and the monomers.

This function takes an array 'startconc' of input concentrations with alternating entries for the initial concentrations of molecules A and B (terminated by two zeroes), then computes the resulting equilibrium concentrations from the free energies for the dimers. Dimer free energies should be the dimer-only free energies, i.e. the FcAB entries from the `vrna_dimer_pf_t` struct.

Deprecated { Use `vrna_pf_dimer_concentrations()` instead!}

Parameters

<i>FEAB</i>	Free energy of AB dimer (FcAB entry)
<i>FEAA</i>	Free energy of AA dimer (FcAB entry)
<i>FEBB</i>	Free energy of BB dimer (FcAB entry)
<i>FEA</i>	Free energy of monomer A
<i>FEB</i>	Free energy of monomer B
<i>startconc</i>	List of start concentrations [a0],[b0],[a1],[b1],...,[an],[bn],[0],[0]

Returns

`vrna_dimer_conc_t` array containing the equilibrium energies and start concentrations

18.27 concentrations.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONCENTRATIONS_H
00002 #define VIENNA_RNA_PACKAGE_CONCENTRATIONS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #   define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00028 typedef struct vrna_dimer_conc_s vrna_dimer_conc_t;
00029
00030
00031 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00032
00036 typedef struct vrna_dimer_conc_s ConcEnt;
```

```

00037
00038 #endif
00039
00040 #include <ViennaRNA/params/basic.h>
00041
00042 struct vrna_dimer_conc_s {
00043     double   Ac_start;
00044     double   Bc_start;
00045     double   ABc;
00046     double   AAc;
00047     double   BBc;
00048     double   Ac;
00049     double   Bc;
00050 };
00051
00052 vrna_dimer_conc_t *vrna_pf_dimer_concentrations(double FcAB,
00053                                                  double FcAA,
00054                                                  double FcBB,
00055                                                  double FEA,
00056                                                  double FEB,
00057                                                  const double *startconc,
00058                                                  const vrna_exp_param_t *exp_params);
00059
00060 double *
00061 vrna_equilibrium_constants(const double *dG_complexes,
00062                           const double *dG_strands,
00063                           const unsigned int **A,
00064                           double kT,
00065                           size_t strands,
00066                           size_t complexes);
00067
00068 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00069 /*
00070  * DEPRECATED FUNCTIONS
00071  */
00072 DEPRECATED(vrna_dimer_conc_t *get_concentrations(double FEAB,
00073                                                  double FEAA,
00074                                                  double FEBB,
00075                                                  double FEA,
00076                                                  double FEB,
00077                                                  double *startconc),
00078           "Use vrna_pf_dimer_concentrations() instead");
00079 #endif
00080 #endif

```

18.28 ViennaRNA/constraints.h File Reference

Use [ViennaRNA/constraints/basic.h](#) instead.

Include dependency graph for constraints.h:

18.28.1 Detailed Description

Use [ViennaRNA/constraints/basic.h](#) instead.

Deprecated Use [ViennaRNA/constraints/basic.h](#) instead

18.29 constraints.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_DEPRECATED_H
00003
00004 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00005 # ifdef VRNA_WARN_DEPRECATED
00006 #warning "Including deprecated header file <ViennaRNA/constraints.h>! Use
00007 <ViennaRNA/constraints/basic.h> instead!"
00008 # endif
00009 #include <ViennaRNA/constraints/basic.h>
00010 #include <ViennaRNA/constraints/hard.h>
00011 #include <ViennaRNA/constraints/soft.h>
00012 #include <ViennaRNA/constraints/SHAPE.h>

```

```

00018 #include <ViennaRNA/constraints/ligand.h>
00019 #endif
00020
00021 #endif

```

18.30 ViennaRNA/constraints/hard.h File Reference

Functions and data structures for handling of secondary structure hard constraints.

Include dependency graph for hard.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_hc_s](#)
The hard constraints data structure. [More...](#)
- struct [vrna_hc_up_s](#)
A single hard constraint for a single nucleotide. [More...](#)

Macros

- #define [VRNA_CONSTRAINT_NO_HEADER](#) 0
do not print the header information line
- #define [VRNA_CONSTRAINT_DB](#) 16384U
Flag for [vrna_constraints_add\(\)](#) to indicate that constraint is passed in pseudo dot-bracket notation.
- #define [VRNA_CONSTRAINT_DB_ENFORCE_BP](#) 32768U
Switch for dot-bracket structure constraint to enforce base pairs.
- #define [VRNA_CONSTRAINT_DB_PIPE](#) 65536U
Flag that is used to indicate the pipe '|' sign in pseudo dot-bracket notation of hard constraints.
- #define [VRNA_CONSTRAINT_DB_DOT](#) 131072U
dot '.' switch for structure constraints (no constraint at all)
- #define [VRNA_CONSTRAINT_DB_X](#) 262144U
'x' switch for structure constraint (base must not pair)
- #define [VRNA_CONSTRAINT_DB_ANG_BRACK](#) 524288U
angle brackets '<', '>' switch for structure constraint (paired downstream/upstream)
- #define [VRNA_CONSTRAINT_DB_RND_BRACK](#) 1048576U
round brackets '(', ')' switch for structure constraint (base i pairs base j)
- #define [VRNA_CONSTRAINT_DB_INTRAMOL](#) 2097152U
Flag that is used to indicate the character 'I' in pseudo dot-bracket notation of hard constraints.
- #define [VRNA_CONSTRAINT_DB_INTERMOL](#) 4194304U
Flag that is used to indicate the character 'e' in pseudo dot-bracket notation of hard constraints.
- #define [VRNA_CONSTRAINT_DB_GQUAD](#) 8388608U
'+' switch for structure constraint (base is involved in a quad)
- #define [VRNA_CONSTRAINT_DB_WUSS](#) 33554432U
Flag to indicate Washington University Secondary Structure (WUSS) notation of the hard constraint string.
- #define [VRNA_CONSTRAINT_DB_DEFAULT](#)
Switch for dot-bracket structure constraint with default symbols.
- #define [VRNA_CONSTRAINT_CONTEXT_EXT_LOOP](#) (unsigned char)0x01
Hard constraints flag, base pair in the exterior loop.
- #define [VRNA_CONSTRAINT_CONTEXT_HP_LOOP](#) (unsigned char)0x02
Hard constraints flag, base pair encloses hairpin loop.
- #define [VRNA_CONSTRAINT_CONTEXT_INT_LOOP](#) (unsigned char)0x04
Hard constraints flag, base pair encloses an interior loop.
- #define [VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC](#) (unsigned char)0x08
Hard constraints flag, base pair encloses a multi branch loop.

- #define [VRNA_CONSTRAINT_CONTEXT_MB_LOOP](#) (unsigned char)0x10
Hard constraints flag, base pair is enclosed in an interior loop.
- #define [VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC](#) (unsigned char)0x20
Hard constraints flag, base pair is enclosed in a multi branch loop.
- #define [VRNA_CONSTRAINT_CONTEXT_ENFORCE](#) (unsigned char)0x40
Hard constraint flag to indicate enforcement of constraints.
- #define [VRNA_CONSTRAINT_CONTEXT_NO_REMOVE](#) (unsigned char)0x80
Hard constraint flag to indicate not to remove base pairs that conflict with a given constraint.
- #define [VRNA_CONSTRAINT_CONTEXT_NONE](#) (unsigned char)0
Constraint context flag that forbids any loop.
- #define [VRNA_CONSTRAINT_CONTEXT_CLOSING_LOOPS](#)
Constraint context flag indicating base pairs that close any loop.
- #define [VRNA_CONSTRAINT_CONTEXT_ENCLOSED_LOOPS](#)
Constraint context flag indicating base pairs enclosed by any loop.
- #define [VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS](#)
Constraint context flag indicating any loop context.

Typedefs

- typedef struct [vrna_hc_s](#) [vrna_hc_t](#)
Typename for the hard constraints data structure [vrna_hc_s](#).
- typedef struct [vrna_hc_up_s](#) [vrna_hc_up_t](#)
Typename for the single nucleotide hard constraint data structure [vrna_hc_up_s](#).
- typedef unsigned char(* [vrna_hc_eval_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to evaluate whether or not a particular decomposition step is contributing to the solution space.

Enumerations

- enum [vrna_hc_type_e](#) { [VRNA_HC_DEFAULT](#) , [VRNA_HC_WINDOW](#) }
The hard constraints type.

Functions

- void [vrna_message_constraint_options](#) (unsigned int option)
Print a help message for pseudo dot-bracket structure constraint characters to stdout. (constraint support is specified by option parameter)
- void [vrna_message_constraint_options_all](#) (void)
Print structure constraint characters to stdout (full constraint support)
- void [vrna_hc_init](#) ([vrna_fold_compound_t](#) *vc)
Initialize/Reset hard constraints to default values.
- void [vrna_hc_add_up](#) ([vrna_fold_compound_t](#) *vc, int i, unsigned char option)
Make a certain nucleotide unpaired.
- int [vrna_hc_add_up_batch](#) ([vrna_fold_compound_t](#) *vc, [vrna_hc_up_t](#) *constraints)
Apply a list of hard constraints for single nucleotides.
- int [vrna_hc_add_bp](#) ([vrna_fold_compound_t](#) *vc, int i, int j, unsigned char option)
Favorize/Enforce a certain base pair (i,j)
- void [vrna_hc_add_bp_nonspecific](#) ([vrna_fold_compound_t](#) *vc, int i, int d, unsigned char option)
Enforce a nucleotide to be paired (upstream/downstream)
- void [vrna_hc_free](#) ([vrna_hc_t](#) *hc)
Free the memory allocated by a [vrna_hc_t](#) data structure.
- void [vrna_hc_add_f](#) ([vrna_fold_compound_t](#) *vc, [vrna_hc_eval_f](#) f)
Add a function pointer pointer for the generic hard constraint feature.

- void [vrna_hc_add_data](#) ([vrna_fold_compound_t](#) *vc, void *data, [vrna_auxdata_free_f](#) f)
Add an auxiliary data structure for the generic hard constraints callback function.
- int [vrna_hc_add_from_db](#) ([vrna_fold_compound_t](#) *vc, const char *constraint, unsigned int options)
Add hard constraints from pseudo dot-bracket notation.
- void [print_tty_constraint](#) (unsigned int option)
Print structure constraint characters to stdout. (constraint support is specified by option parameter)
- void [print_tty_constraint_full](#) (void)
Print structure constraint characters to stdout (full constraint support)
- void [constrain_ptypes](#) (const char *constraint, unsigned int length, char *ptype, int *BP, int min_loop_size, unsigned int idx_type)
Insert constraining pair types according to constraint structure string.

18.30.1 Detailed Description

Functions and data structures for handling of secondary structure hard constraints.

18.30.2 Macro Definition Documentation

18.30.2.1 VRNA_CONSTRAINT_NO_HEADER

```
#define VRNA_CONSTRAINT_NO_HEADER 0
```

do not print the header information line

Deprecated This mode is not supported anymore!

18.30.2.2 VRNA_CONSTRAINT_DB_ANG_BRACK

```
#define VRNA_CONSTRAINT_DB_ANG_BRACK 524288U
```

angle brackets '<', '>' switch for structure constraint (paired downstream/upstream)

See also

[vrna_hc_add_from_db\(\)](#), [vrna_constraints_add\(\)](#), [vrna_message_constraint_options\(\)](#), [vrna_message_constraint_options_all\(\)](#)

18.30.3 Enumeration Type Documentation

18.30.3.1 vrna_hc_type_e

```
enum vrna_hc_type_e
```

The hard constraints type.

Global and local structure prediction methods use a slightly different way to handle hard constraints internally. This enum is used to distinguish both types.

Enumerator

VRNA_HC_DEFAULT	Default Hard Constraints.
VRNA_HC_WINDOW	Hard Constraints suitable for local structure prediction using window approach. See also vrna_mfe_window() , vrna_mfe_window_zscore() , pfl_fold()

18.30.4 Function Documentation

18.30.4.1 `vrna_hc_add_data()`

```
void vrna_hc_add_data (
    vrna_fold_compound_t * vc,
    void * data,
    vrna_auxdata_free_f f )
```

Add an auxiliary data structure for the generic hard constraints callback function.

See also

[vrna_hc_add_f\(\)](#)

Parameters

<i>vc</i>	The fold compound the generic hard constraint function should be bound to
<i>data</i>	A pointer to the data structure that holds required data for function 'f'
<i>f</i>	A pointer to a function that free's the memory occupied by <i>data</i> (Maybe <code>NULL</code>)

18.30.4.2 `print_tty_constraint()`

```
void print_tty_constraint (
    unsigned int option )
```

Print structure constraint characters to stdout. (constraint support is specified by option parameter)

Deprecated Use [vrna_message_constraints\(\)](#) instead!

Parameters

<i>option</i>	Option switch that tells which constraint help will be printed
---------------	--

18.30.4.3 `print_tty_constraint_full()`

```
void print_tty_constraint_full (
    void )
```

Print structure constraint characters to stdout (full constraint support)

Deprecated Use [vrna_message_constraint_options_all\(\)](#) instead!

18.30.4.4 `constrain_ptypes()`

```
void constrain_ptypes (
    const char * constraint,
    unsigned int length,
    char * ptype,
    int * BP,
    int min_loop_size,
    unsigned int idx_type )
```

Insert constraining pair types according to constraint structure string.

Deprecated Do not use this function anymore! Structure constraints are now handled through [vrna_hc_t](#) and related functions.

Parameters

<i>constraint</i>	The structure constraint string
<i>length</i>	The actual length of the sequence (constraint may be shorter)
<i>ptype</i>	A pointer to the basepair type array
<i>BP</i>	(not used anymore)
<i>min_loop_size</i>	The minimal loop size (usually TURN)
<i>idx_type</i>	Define the access type for base pair type array (0 = indx, 1 = iindx)

18.31 hard.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(" ", msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00016
00034 typedef struct vrna_hc_s vrna_hc_t;
00035
00040 typedef struct vrna_hc_up_s vrna_hc_up_t;
00041
00042 typedef struct vrna_hc_depot_s vrna_hc_depot_t;
00043
00044 #include <ViennaRNA/fold_compound.h>
00045 #include <ViennaRNA/constraints/basic.h>
00046
00078 typedef unsigned char (*vrna_hc_eval_f)(int i,
00079 int j,
00080 int k,
00081 int l,
00082 unsigned char d,
00083 void *data);
00084
00085 DEPRECATED(typedef unsigned char (vrna_callback_hc_evaluate)(int i,
00086 int j,
00087 int k,
00088 int l,
00089 unsigned char d,
00090 void *data),
00091 "Use vrna_hc_eval_f instead!");
00092
00093
00099 #define VRNA_CONSTRAINT_NO_HEADER 0
00100
00109 #define VRNA_CONSTRAINT_DB 16384U
00110
00122 #define VRNA_CONSTRAINT_DB_ENFORCE_BP 32768U
00123
00135 #define VRNA_CONSTRAINT_DB_PIPE 65536U
00136
00145 #define VRNA_CONSTRAINT_DB_DOT 131072U
00154 #define VRNA_CONSTRAINT_DB_X 262144U
00161 #define VRNA_CONSTRAINT_DB_ANG_BRACK 524288U
00170 #define VRNA_CONSTRAINT_DB_RND_BRACK 1048576U
00171
00183 #define VRNA_CONSTRAINT_DB_INTRAMOL 2097152U
00184
00196 #define VRNA_CONSTRAINT_DB_INTERMOL 4194304U
00197
00208 #define VRNA_CONSTRAINT_DB_GQUAD 8388608U
00209

```



```
00210 #define VRNA_CONSTRAINT_DB_CANONICAL_BP      16777216U
00211
00220 #define VRNA_CONSTRAINT_DB_WUSS              33554432U
00221
00222
00234 #define VRNA_CONSTRAINT_DB_DEFAULT \
00235     (VRNA_CONSTRAINT_DB \
00236      | VRNA_CONSTRAINT_DB_PIPE \
00237      | VRNA_CONSTRAINT_DB_DOT \
00238      | VRNA_CONSTRAINT_DB_X \
00239      | VRNA_CONSTRAINT_DB_ANG_BRACK \
00240      | VRNA_CONSTRAINT_DB_RND_BRACK \
00241      | VRNA_CONSTRAINT_DB_INTRAMOL \
00242      | VRNA_CONSTRAINT_DB_INTERMOL \
00243      | VRNA_CONSTRAINT_DB_GQUAD \
00244     )
00245
00252 #define VRNA_CONSTRAINT_CONTEXT_EXT_LOOP      (unsigned char)0x01
00253
00260 #define VRNA_CONSTRAINT_CONTEXT_HP_LOOP      (unsigned char)0x02
00261
00268 #define VRNA_CONSTRAINT_CONTEXT_INT_LOOP      (unsigned char)0x04
00269
00276 #define VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC (unsigned char)0x08
00277
00284 #define VRNA_CONSTRAINT_CONTEXT_MB_LOOP      (unsigned char)0x10
00285
00292 #define VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC (unsigned char)0x20
00293
00297 #define VRNA_CONSTRAINT_CONTEXT_ENFORCE      (unsigned char)0x40
00298
00302 #define VRNA_CONSTRAINT_CONTEXT_NO_REMOVE    (unsigned char)0x80
00303
00304
00308 #define VRNA_CONSTRAINT_CONTEXT_NONE         (unsigned char)0
00309
00313 #define VRNA_CONSTRAINT_CONTEXT_CLOSING_LOOPS (unsigned char)(VRNA_CONSTRAINT_CONTEXT_EXT_LOOP | \
00314                                                                VRNA_CONSTRAINT_CONTEXT_HP_LOOP | \
00315                                                                VRNA_CONSTRAINT_CONTEXT_INT_LOOP | \
00316                                                                VRNA_CONSTRAINT_CONTEXT_MB_LOOP)
00317
00321 #define VRNA_CONSTRAINT_CONTEXT_ENCLOSED_LOOPS (unsigned char)(VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC |
00322                                                                VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC)
00323
00330 #define VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS     (unsigned char)(VRNA_CONSTRAINT_CONTEXT_CLOSING_LOOPS |
00331                                                                VRNA_CONSTRAINT_CONTEXT_ENCLOSED_LOOPS)
00332
00333
00334 #define VRNA_CONSTRAINT_WINDOW_UPDATE_5      1U
00335
00336 #define VRNA_CONSTRAINT_WINDOW_UPDATE_3      2U
00337
00344 typedef enum {
00345     VRNA_HC_DEFAULT,
00346     VRNA_HC_WINDOW
00350 } vrna_hc_type_e;
00351
00352
00377 struct vrna_hc_s {
00378     vrna_hc_type_e type;
00379     unsigned int    n;
00380
00381     unsigned char   state;
00382
00383 #ifndef VRNA_DISABLE_C11_FEATURES
00384     /* C11 support for unnamed unions/structs */
00385     union {
00386         struct {
00387 #endif
00388         unsigned char *mx;
00389 #ifndef VRNA_DISABLE_C11_FEATURES
00390         };
00391     struct {
00392 #endif
00393         unsigned char **matrix_local;
00394 #ifndef VRNA_DISABLE_C11_FEATURES
00395         };
00396     };
00397 #endif
00398
00399     int          *up_ext;
00402     int          *up_hp;
00405     int          *up_int;
00408     int          *up_ml;
00412     vrna_hc_eval_f f;
```

```
00416 void *data;
00421 vrna_auxdata_free_f free_data;
00432 vrna_hc_depot_t *depot;
00433 };
00434
00440 struct vrna_hc_up_s {
00441     int position;
00442     int strand;
00443     unsigned char options;
00444 };
00445
00468 void
00469 vrna_message_constraint_options(unsigned int option);
00470
00471
00482 void
00483 vrna_message_constraint_options_all(void);
00484
00485
00500 void
00501 vrna_hc_init(vrna_fold_compound_t *vc);
00502
00503
00504 void
00505 vrna_hc_init_window(vrna_fold_compound_t *vc);
00506
00507
00508 int
00509 vrna_hc_prepare(vrna_fold_compound_t *fc,
00510                unsigned int options);
00511
00512
00513 void
00514 vrna_hc_update(vrna_fold_compound_t *fc,
00515               unsigned int i,
00516               unsigned int options);
00517
00518
00533 void
00534 vrna_hc_add_up(vrna_fold_compound_t *vc,
00535               int i,
00536               unsigned char option);
00537
00538
00539 int
00540 vrna_hc_add_up_strand(vrna_fold_compound_t *fc,
00541                      unsigned int i,
00542                      unsigned int strand,
00543                      unsigned char option);
00544
00545
00555 int
00556 vrna_hc_add_up_batch(vrna_fold_compound_t *vc,
00557                     vrna_hc_up_t *constraints);
00558
00559
00560 int
00561 vrna_hc_add_up_strand_batch(vrna_fold_compound_t *fc,
00562                            vrna_hc_up_t *constraints);
00563
00564
00581 int
00582 vrna_hc_add_bp(vrna_fold_compound_t *vc,
00583               int i,
00584               int j,
00585               unsigned char option);
00586
00587
00588 int
00589 vrna_hc_add_bp_strand(vrna_fold_compound_t *fc,
00590                      unsigned int i,
00591                      unsigned int strand_i,
00592                      unsigned int j,
00593                      unsigned int strand_j,
00594                      unsigned char option);
00595
00596
00614 void
00615 vrna_hc_add_bp_nonspecific(vrna_fold_compound_t *vc,
00616                           int i,
00617                           int d,
00618                           unsigned char option);
00619
00620
00632 void
00633 vrna_hc_free(vrna_hc_t *hc);
00634
```

```

00635
00640 void
00641 vrna_hc_add_f(vrna_fold_compound_t *vc,
00642              vrna_hc_eval_f f);
00643
00644
00656 void
00657 vrna_hc_add_data(vrna_fold_compound_t *vc,
00658                 void *data,
00659                 vrna_auxdata_free_f f);
00660
00661
00680 int
00681 vrna_hc_add_from_db(vrna_fold_compound_t *vc,
00682                   const char *constraint,
00683                   unsigned int options);
00684
00685
00686 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00687
00695 DEPRECATED(void
00696             print_tty_constraint(unsigned int option),
00697             "Use vrna_message_constraint_options() instead");
00698
00705 DEPRECATED(void
00706             print_tty_constraint_full(void),
00707             "Use vrna_message_constraint_options_all() instead");
00708
00721 DEPRECATED(void
00722             constrain_ptypes(const char *constraint,
00723                             unsigned int length,
00724                             char *ptype,
00725                             int *BP,
00726                             int min_loop_size,
00727                             unsigned int idx_type),
00728             "Use the new API and the hard constraint framework instead");
00729
00730 #endif
00731
00732 #endif

```

18.32 ViennaRNA/constraints/ligand.h File Reference

Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework. Include dependency graph for ligand.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_sc_motif_s](#)

Typedefs

- typedef struct [vrna_sc_motif_s](#) [vrna_sc_motif_t](#)
Type definition for soft constraint motif.

Functions

- int [vrna_sc_add_hi_motif](#) ([vrna_fold_compound_t](#) *fc, const char *seq, const char *structure, [FLT_OR_DBL](#) energy, unsigned int options)
Add soft constraints for hairpin or interior loop binding motif.

18.32.1 Detailed Description

Functions for incorporation of ligands binding to hairpin and interior loop motifs using the soft constraints framework.

18.33 ligand.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LIGAND_H
00002 #define VIENNA_RNA_PACKAGE_LIGAND_H

```

```

00003
00019 typedef struct vrna_sc_motif_s vrna_sc_motif_t;
00020
00021 #include <ViennaRNA/datastructures/basic.h>
00022 #include <ViennaRNA/fold_compound.h>
00023
00024 struct vrna_sc_motif_s {
00025     int i;
00026     int j;
00027     int k;
00028     int l;
00029     int number;
00030 };
00031
00032
00059 int
00060 vrna_sc_add_hi_motif(vrna_fold_compound_t *fc,
00061                     const char *seq,
00062                     const char *structure,
00063                     FLT_OR_DBL energy,
00064                     unsigned int options);
00065
00066
00067 vrna_sc_motif_t *
00068 vrna_sc_ligand_detect_motifs(vrna_fold_compound_t *fc,
00069                             const char *structure);
00070
00071
00072 vrna_sc_motif_t *
00073 vrna_sc_ligand_get_all_motifs(vrna_fold_compound_t *fc);
00074
00075
00080 #endif

```

18.34 sc_cb_intern.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_INTERN_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_INTERN_H
00003
00004 #define MOD_PARAMS_STACK_dG (1 < 0)
00005 #define MOD_PARAMS_STACK_dH (1 < 1)
00006 #define MOD_PARAMS_MISMATCH_dG (1 < 2)
00007 #define MOD_PARAMS_MISMATCH_dH (1 < 3)
00008 #define MOD_PARAMS_TERMINAL_dG (1 < 4)
00009 #define MOD_PARAMS_TERMINAL_dH (1 < 5)
00010 #define MOD_PARAMS_DANGLES_dG (1 < 6)
00011 #define MOD_PARAMS_DANGLES_dH (1 < 7)
00012
00013 /*
00014 #define DEBUG
00015 */
00016 #define MAX_ALPHABET (6)
00017 #define MAX_PAIRS (NBPAIRS + 1 + 25)
00018
00019
00020 /* a container to store the data read from a json parameter file */
00021 struct vrna_sc_mod_param_s {
00022     unsigned int available;
00023
00024     char *name;
00025     char one_letter_code;
00026     char unmodified;
00027     char pairing_partners[7];
00028     unsigned int pairing_partners_encoding[7];
00029     unsigned int unmodified_encoding;
00030
00031     size_t num_ptypes;
00032     size_t ptypes[MAX_ALPHABET][MAX_ALPHABET];
00033
00034     int stack_dG[MAX_PAIRS][MAX_ALPHABET][MAX_ALPHABET];
00035     int stack_dH[MAX_PAIRS][MAX_ALPHABET][MAX_ALPHABET];
00036
00037     int dangle5_dG[MAX_PAIRS][MAX_ALPHABET];
00038     int dangle5_dH[MAX_PAIRS][MAX_ALPHABET];
00039     int dangle3_dG[MAX_PAIRS][MAX_ALPHABET];
00040     int dangle3_dH[MAX_PAIRS][MAX_ALPHABET];
00041
00042     int mismatch_dG[MAX_PAIRS][MAX_ALPHABET][MAX_ALPHABET];
00043     int mismatch_dH[MAX_PAIRS][MAX_ALPHABET][MAX_ALPHABET];
00044
00045     int terminal_dG[MAX_PAIRS];
00046     int terminal_dH[MAX_PAIRS];
00047 };
00048
00049 /* the actual data structure passed around while evaluating */

```

```

00050 typedef struct {
00051     short    *enc;
00052     size_t   ptypes[MAX_ALPHABET][MAX_ALPHABET];
00053
00054     int       stack_diff[MAX_PAIRS][MAX_ALPHABET][MAX_ALPHABET];
00055
00056     int       dangle5_diff[MAX_PAIRS][MAX_ALPHABET];
00057     int       dangle3_diff[MAX_PAIRS][MAX_ALPHABET];
00058
00059     int       mismatch_diff[MAX_PAIRS][MAX_ALPHABET][MAX_ALPHABET];
00060
00061     int       terminal_diff[MAX_PAIRS];
00062 } energy_corrections;
00063
00064
00065 #endif

```

18.35 ViennaRNA/constraints/SHAPE.h File Reference

This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints.

Include dependency graph for SHAPE.h: This graph shows which files directly or indirectly include this file:

Functions

- int [vrna_sc_add_SHAPE_deigan](#) ([vrna_fold_compound_t](#) *vc, const double *reactivities, double m, double b, unsigned int options)
Add SHAPE reactivity data as soft constraints (Deigan et al. method)
- int [vrna_sc_add_SHAPE_deigan_aln](#) ([vrna_fold_compound_t](#) *vc, const char **shape_files, const int *shape_file_association, double m, double b, unsigned int options)
Add SHAPE reactivity data from files as soft constraints for consensus structure prediction (Deigan et al. method)
- int [vrna_sc_add_SHAPE_zarrinhalam](#) ([vrna_fold_compound_t](#) *vc, const double *reactivities, double b, double default_value, const char *shape_conversion, unsigned int options)
Add SHAPE reactivity data as soft constraints (Zarrinhalam et al. method)
- int [vrna_sc_SHAPE_parse_method](#) (const char *method_string, char *method, float *param_1, float *param_2)
Parse a character string and extract the encoded SHAPE reactivity conversion method and possibly the parameters for conversion into pseudo free energies.
- int [vrna_sc_SHAPE_to_pr](#) (const char *shape_conversion, double *values, int length, double default_value)
Convert SHAPE reactivity values to probabilities for being unpaired.

18.35.1 Detailed Description

This module provides function to incorporate SHAPE reactivity data into the folding recursions by means of soft constraints.

18.35.2 Function Documentation

18.35.2.1 [vrna_sc_SHAPE_parse_method\(\)](#)

```

int vrna_sc_SHAPE_parse_method (
    const char * method_string,
    char * method,
    float * param_1,
    float * param_2 )

```

Parse a character string and extract the encoded SHAPE reactivity conversion method and possibly the parameters for conversion into pseudo free energies.

Parameters

<i>method_string</i>	The string that contains the encoded SHAPE reactivity conversion method
<i>method</i>	A pointer to the memory location where the method character will be stored
<i>param_1</i>	A pointer to the memory location where the first parameter of the corresponding method will be stored
<i>param_2</i>	A pointer to the memory location where the second parameter of the corresponding method will be stored

Returns

1 on successful extraction of the method, 0 on errors

18.36 SHAPE.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_H
00003
00004 #include <ViennaRNA/fold_compound.h>
00005
00023 void
00024 vrna_constraints_add_SHAPE(vrna_fold_compound_t *vc,
00025                          const char *shape_file,
00026                          const char *shape_method,
00027                          const char *shape_conversion,
00028                          int verbose,
00029                          unsigned int constraint_type);
00030
00031
00032 void
00033 vrna_constraints_add_SHAPE_aliases(vrna_fold_compound_t *vc,
00034                                   const char *shape_method,
00035                                   const char **shape_files,
00036                                   const int *shape_file_association,
00037                                   int verbose,
00038                                   unsigned int constraint_type);
00039
00040
00066 int
00067 vrna_sc_add_SHAPE_deigan(vrna_fold_compound_t *vc,
00068                         const double *reactivities,
00069                         double m,
00070                         double b,
00071                         unsigned int options);
00072
00073
00086 int
00087 vrna_sc_add_SHAPE_deigan_aliases(vrna_fold_compound_t *vc,
00088                                  const char **shape_files,
00089                                  const int *shape_file_association,
00090                                  double m,
00091                                  double b,
00092                                  unsigned int options);
00093
00094
00117 int
00118 vrna_sc_add_SHAPE_zarrinhalam(vrna_fold_compound_t *vc,
00119                               const double *reactivities,
00120                               double b,
00121                               double default_value,
00122                               const char *shape_conversion,
00123                               unsigned int options);
00124
00125
00138 int
00139 vrna_sc_SHAPE_parse_method(const char *method_string,
00140                           char *method,
00141                           float *param_1,
00142                           float *param_2);
00143
00144
00159 int
00160 vrna_sc_SHAPE_to_pr(const char *shape_conversion,
00161                   double *values,
00162                   int length,
00163                   double default_value);
00164

```

```
00165
00166 #endif
```

18.37 ViennaRNA/constraints/soft.h File Reference

Functions and data structures for secondary structure soft constraints.

Include dependency graph for soft.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_sc_bp_storage_t](#)
A base pair constraint.
- struct [vrna_sc_s](#)
The soft constraints data structure. [More...](#)

Typedefs

- typedef struct [vrna_sc_s](#) [vrna_sc_t](#)
Typename for the soft constraints data structure [vrna_sc_s](#).
- typedef int(* [vrna_sc_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to retrieve pseudo energy contribution for soft constraint feature.
- typedef [FLT_OR_DBL](#)(* [vrna_sc_exp_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to retrieve pseudo energy contribution as Boltzmann Factors for soft constraint feature.
- typedef [vrna_basepair_t](#) *(* [vrna_sc_bt_f](#)) (int i, int j, int k, int l, unsigned char d, void *data)
Callback to retrieve auxiliary base pairs for soft constraint feature.

Enumerations

- enum [vrna_sc_type_e](#) { [VRNA_SC_DEFAULT](#) , [VRNA_SC_WINDOW](#) }
The type of a soft constraint.

Functions

- void [vrna_sc_init](#) ([vrna_fold_compound_t](#) *vc)
Initialize an empty soft constraints data structure within a [vrna_fold_compound_t](#).
- int [vrna_sc_set_bp](#) ([vrna_fold_compound_t](#) *vc, const [FLT_OR_DBL](#) **constraints, unsigned int options)
Set soft constraints for paired nucleotides.
- int [vrna_sc_add_bp](#) ([vrna_fold_compound_t](#) *vc, int i, int j, [FLT_OR_DBL](#) energy, unsigned int options)
Add soft constraints for paired nucleotides.
- int [vrna_sc_set_up](#) ([vrna_fold_compound_t](#) *vc, const [FLT_OR_DBL](#) *constraints, unsigned int options)
Set soft constraints for unpaired nucleotides.
- int [vrna_sc_add_up](#) ([vrna_fold_compound_t](#) *vc, int i, [FLT_OR_DBL](#) energy, unsigned int options)
Add soft constraints for unpaired nucleotides.
- void [vrna_sc_remove](#) ([vrna_fold_compound_t](#) *vc)
Remove soft constraints from [vrna_fold_compound_t](#).
- void [vrna_sc_free](#) ([vrna_sc_t](#) *sc)
Free memory occupied by a [vrna_sc_t](#) data structure.
- int [vrna_sc_add_data](#) ([vrna_fold_compound_t](#) *vc, void *data, [vrna_auxdata_free_f](#) free_data)
Add an auxiliary data structure for the generic soft constraints callback function.
- int [vrna_sc_add_f](#) ([vrna_fold_compound_t](#) *vc, [vrna_sc_f](#) f)
Bind a function pointer for generic soft constraint feature (MFE version)
- int [vrna_sc_add_bt](#) ([vrna_fold_compound_t](#) *vc, [vrna_sc_bt_f](#) f)
Bind a backtracking function pointer for generic soft constraint feature.
- int [vrna_sc_add_exp_f](#) ([vrna_fold_compound_t](#) *vc, [vrna_sc_exp_f](#) exp_f)
Bind a function pointer for generic soft constraint feature (PF version)

18.37.1 Detailed Description

Functions and data structures for secondary structure soft constraints.

18.37.2 Enumeration Type Documentation

18.37.2.1 vrna_sc_type_e

enum [vrna_sc_type_e](#)

The type of a soft constraint.

Enumerator

VRNA_SC_DEFAULT	Default Soft Constraints.
VRNA_SC_WINDOW	Soft Constraints suitable for local structure prediction using window approach. See also vrna_mfe_window() , vrna_mfe_window_zscore() , pfl_fold()

18.38 soft.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_H
00003
00026 typedef struct vrna_sc_s vrna_sc_t;
00027
00028 #include <ViennaRNA/datastructures/basic.h>
00029 #include <ViennaRNA/fold_compound.h>
00030 #include <ViennaRNA/constraints/basic.h>
00031
00064 typedef int (*vrna_sc_f) (int          i,
00065                          int          j,
00066                          int          k,
00067                          int          l,
00068                          unsigned char d,
00069                          void          *data);
00070
00071 DEPRECATED typedef int (vrna_callback_sc_energy) (int i,
00072                                                  int j,
00073                                                  int k,
00074                                                  int l,
00075                                                  unsigned char d,
00076                                                  void *data),
00077          "Use vrna_sc_f instead!");
00078
00079
00080 typedef int (*vrna_sc_direct_f) (vrna_fold_compound_t *fc,
00081                                int          i,
00082                                int          j,
00083                                int          k,
00084                                int          l,
00085                                void          *data);
00086
00119 typedef FLT_OR_DBL (*vrna_sc_exp_f) (int          i,
00120                                     int          j,
00121                                     int          k,
00122                                     int          l,
00123                                     unsigned char d,
00124                                     void          *data);
00125
00126 DEPRECATED typedef FLT_OR_DBL (vrna_callback_sc_exp_energy) (int i,
00127                                                             int j,
00128                                                             int k,
00129                                                             int l,
00130                                                             unsigned char d,
00131                                                             void *data),
00132          "Use vrna_sc_exp_f instead!");
00133
00134

```



```

00135 typedef FLT_OR_DBL (*vrna_sc_exp_direct_f) (vrna_fold_compound_t *fc,
00136                                             int i,
00137                                             int j,
00138                                             int k,
00139                                             int l,
00140                                             void *data);
00141
00168 typedef vrna_basepair_t *(*vrna_sc_bt_f) (int i,
00169                                           int j,
00170                                           int k,
00171                                           int l,
00172                                           unsigned char d,
00173                                           void *data);
00174
00175 DEPRECATED (typedef vrna_basepair_t *(*vrna_callback_sc_backtrack) (int i,
00176                                                                    int j,
00177                                                                    int k,
00178                                                                    int l,
00179                                                                    unsigned char d,
00180                                                                    void *data),
00181            "Use vrna_sc_bt_f instead");
00182
00183
00187 typedef enum {
00188     VRNA_SC_DEFAULT,
00189     VRNA_SC_WINDOW
00190 } vrna_sc_type_e;
00191
00192
00199 typedef struct {
00200     unsigned int interval_start;
00201     unsigned int interval_end;
00202     int e;
00203 } vrna_sc_bp_storage_t;
00204
00205
00211 struct vrna_sc_s {
00212     const vrna_sc_type_e type;
00213     unsigned int n;
00214
00215     unsigned char state;
00216
00217     int **energy_up;
00218     FLT_OR_DBL **exp_energy_up;
00219     int *up_storage;
00220     vrna_sc_bp_storage_t **bp_storage;
00221
00222 #ifndef VRNA_DISABLE_C11_FEATURES
00223     /* C11 support for unnamed unions/structs */
00224     union {
00225         struct {
00226             int *energy_bp;
00227             FLT_OR_DBL *exp_energy_bp;
00228         }
00229 #ifndef VRNA_DISABLE_C11_FEATURES
00230         /* C11 support for unnamed unions/structs */
00231         struct {
00232             int **energy_bp_local;
00233             FLT_OR_DBL **exp_energy_bp_local;
00234         }
00235 #ifndef VRNA_DISABLE_C11_FEATURES
00236         /* C11 support for unnamed unions/structs */
00237         struct {
00238             int *energy_stack;
00239             FLT_OR_DBL *exp_energy_stack;
00240             /* generic soft constraints below */
00241             vrna_sc_f f;
00242             vrna_sc_bt_f bt;
00243             vrna_sc_exp_f exp_f;
00244             void *data;
00245             vrna_auxdata_free_f free_data;
00246         };
00247     };
00248 void
00249 vrna_sc_init(vrna_fold_compound_t *vc);
00250
00251 void
00252 vrna_sc_prepare(vrna_fold_compound_t *vc,
00253                unsigned int options);
00254
00255 int
00256 vrna_sc_update(vrna_fold_compound_t *vc,

```

```

00298         unsigned int      i,
00299         unsigned int      options);
00300
00301
00317 int
00318 vrna_sc_set_bp(vrna_fold_compound_t *vc,
00319               const FLT_OR_DBL      **constraints,
00320               unsigned int          options);
00321
00322
00337 int
00338 vrna_sc_add_bp(vrna_fold_compound_t *vc,
00339               int                    i,
00340               int                    j,
00341               FLT_OR_DBL            energy,
00342               unsigned int          options);
00343
00344
00360 int
00361 vrna_sc_set_up(vrna_fold_compound_t *vc,
00362               const FLT_OR_DBL      **constraints,
00363               unsigned int          options);
00364
00365
00379 int
00380 vrna_sc_add_up(vrna_fold_compound_t *vc,
00381               int                    i,
00382               FLT_OR_DBL            energy,
00383               unsigned int          options);
00384
00385
00386 int
00387 vrna_sc_set_stack(vrna_fold_compound_t *vc,
00388               const FLT_OR_DBL      **constraints,
00389               unsigned int          options);
00390
00391
00392 int
00393 vrna_sc_set_stack_comparative(vrna_fold_compound_t *fc,
00394                               const FLT_OR_DBL      **constraints,
00395                               unsigned int          options);
00396
00397
00398 int
00399 vrna_sc_add_stack(vrna_fold_compound_t *vc,
00400               int                    i,
00401               FLT_OR_DBL            energy,
00402               unsigned int          options);
00403
00404
00405 int
00406 vrna_sc_add_stack_comparative(vrna_fold_compound_t *fc,
00407                               int                    i,
00408                               const FLT_OR_DBL      **energies,
00409                               unsigned int          options);
00410
00411
00421 void
00422 vrna_sc_remove(vrna_fold_compound_t *vc);
00423
00424
00432 void
00433 vrna_sc_free(vrna_sc_t *sc);
00434
00435
00448 int
00449 vrna_sc_add_data(vrna_fold_compound_t *vc,
00450               void                    *data,
00451               vrna_auxdata_free_f    free_data);
00452
00453
00454 int
00455 vrna_sc_add_data_comparative(vrna_fold_compound_t *vc,
00456                               void                    **data,
00457                               vrna_auxdata_free_f    *free_data);
00458
00459
00476 int
00477 vrna_sc_add_f(vrna_fold_compound_t *vc,
00478               vrna_sc_f              f);
00479
00480
00481 size_t
00482 vrna_sc_multi_cb_add(vrna_fold_compound_t *fc,
00483                     vrna_sc_direct_f    cb,
00484                     vrna_sc_exp_direct_f cb_exp,
00485                     void                    *data,

```

```

00486             vrna_auxdata_free_f free_data,
00487             unsigned int         decomp_type);
00488
00489
00490 int
00491 vrna_sc_add_f_comparative(vrna_fold_compound_t *vc,
00492                          vrna_sc_f            *f);
00493
00494
00513 int
00514 vrna_sc_add_bt(vrna_fold_compound_t *vc,
00515               vrna_sc_bt_f          f);
00516
00517
00535 int
00536 vrna_sc_add_exp_f(vrna_fold_compound_t *vc,
00537                  vrna_sc_exp_f         exp_f);
00538
00539
00540 int
00541 vrna_sc_add_exp_f_comparative(vrna_fold_compound_t *vc,
00542                               vrna_sc_exp_f         *exp_f);
00543
00544
00545 #endif

```

18.39 ViennaRNA/constraints/soft_special.h File Reference

Specialized implementations that utilize the soft constraint callback mechanism.

Typedefs

- typedef struct [vrna_sc_mod_param_s](#) * [vrna_sc_mod_param_t](#)
Modified base parameter data structure.

Functions

- [vrna_sc_mod_param_t vrna_sc_mod_read_from_jsonfile](#) (const char *filename, [vrna_md_t](#) *md)
Parse and extract energy parameters for a modified base from a JSON file.
- [vrna_sc_mod_param_t vrna_sc_mod_read_from_json](#) (const char *json, [vrna_md_t](#) *md)
Parse and extract energy parameters for a modified base from a JSON string.
- void [vrna_sc_mod_parameters_free](#) ([vrna_sc_mod_param_t](#) params)
Release memory occupied by a modified base parameter data structure.
- int [vrna_sc_mod_json](#) ([vrna_fold_compound_t](#) *fc, const char *json, const unsigned int *modification_sites)
Prepare soft constraint callbacks for modified base as specified in JSON string.
- int [vrna_sc_mod_jsonfile](#) ([vrna_fold_compound_t](#) *fc, const char *json_file, const unsigned int *modification_sites)
Prepare soft constraint callbacks for modified base as specified in JSON string.
- int [vrna_sc_mod](#) ([vrna_fold_compound_t](#) *fc, const [vrna_sc_mod_param_t](#) params, const unsigned int *modification_sites)
Prepare soft constraint callbacks for modified base as specified in JSON string.
- int [vrna_sc_mod_m6A](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for N6-methyl-adenosine (m6A)
- int [vrna_sc_mod_pseudouridine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for Pseudouridine.
- int [vrna_sc_mod_inosine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for Inosine.
- int [vrna_sc_mod_7DA](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for 7-deaza-adenosine (7DA)
- int [vrna_sc_mod_purine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for Purine (a.k.a. nebularine)
- int [vrna_sc_mod_dihydrouridine](#) ([vrna_fold_compound_t](#) *fc, const unsigned int *modification_sites)
Add soft constraint callbacks for dihydrouridine.

18.39.1 Detailed Description

Specialized implementations that utilize the soft constraint callback mechanism.

,

18.40 soft_special.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_SPECIAL_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_SPECIAL_H
00003
00037 typedef struct vrna_sc_mod_param_s *vrna_sc_mod_param_t;
00038
00039
00050 vrna_sc_mod_param_t
00051 vrna_sc_mod_read_from_jsonfile(const char *filename,
00052                               vrna_md_t *md);
00053
00054
00065 vrna_sc_mod_param_t
00066 vrna_sc_mod_read_from_json(const char *json,
00067                            vrna_md_t *md);
00068
00069
00077 void
00078 vrna_sc_mod_parameters_free(vrna_sc_mod_param_t params);
00079
00080
00097 int
00098 vrna_sc_mod_json(vrna_fold_compound_t *fc,
00099                 const char *json,
00100                 const unsigned int *modification_sites);
00101
00102
00120 int
00121 vrna_sc_mod_jsonfile(vrna_fold_compound_t *fc,
00122                     const char *json_file,
00123                     const unsigned int *modification_sites);
00124
00125
00146 int
00147 vrna_sc_mod(vrna_fold_compound_t *fc,
00148            const vrna_sc_mod_param_t params,
00149            const unsigned int *modification_sites);
00150
00151
00164 int
00165 vrna_sc_mod_m6A(vrna_fold_compound_t *fc,
00166                const unsigned int *modification_sites);
00167
00168
00181 int
00182 vrna_sc_mod_pseudouridine(vrna_fold_compound_t *fc,
00183                          const unsigned int *modification_sites);
00184
00185
00198 int
00199 vrna_sc_mod_inosine(vrna_fold_compound_t *fc,
00200                   const unsigned int *modification_sites);
00201
00202
00215 int
00216 vrna_sc_mod_7DA(vrna_fold_compound_t *fc,
00217                const unsigned int *modification_sites);
00218
00219
00232 int
00233 vrna_sc_mod_purine(vrna_fold_compound_t *fc,
00234                  const unsigned int *modification_sites);
00235
00236
00250 int
00251 vrna_sc_mod_dihydrouridine(vrna_fold_compound_t *fc,
00252                          const unsigned int *modification_sites);
00253
00254
00258 #endif
```

18.41 ViennaRNA/constraints_hard.h File Reference

Use [ViennaRNA/constraints/hard.h](#) instead.

Include dependency graph for constraints_hard.h:

18.41.1 Detailed Description

Use [ViennaRNA/constraints/hard.h](#) instead.

Deprecated Use [ViennaRNA/constraints/hard.h](#) instead

18.42 constraints_hard.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_HARD_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #  ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/constraints_hard.h>! Use
<ViennaRNA/constraints/hard.h> instead!"
00013 #  endif
00014 #include <ViennaRNA/constraints/hard.h>
00015 #endif
00016
00017 #endif
```

18.43 ViennaRNA/constraints_ligand.h File Reference

Use [ViennaRNA/constraints/ligand.h](#) instead.

Include dependency graph for constraints_ligand.h:

18.43.1 Detailed Description

Use [ViennaRNA/constraints/ligand.h](#) instead.

Deprecated Use [ViennaRNA/constraints/ligand.h](#) instead

18.44 constraints_ligand.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_LIGAND_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_LIGAND_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #  ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/constraints_ligand.h>! Use
<ViennaRNA/constraints/ligand.h> instead!"
00013 #  endif
00014 #include <ViennaRNA/constraints/ligand.h>
00015 #endif
00016
00017 #endif
```

18.45 ViennaRNA/constraints_SHAPE.h File Reference

Use [ViennaRNA/constraints/SHAPE.h](#) instead.

Include dependency graph for constraints_SHAPE.h:

18.45.1 Detailed Description

Use [ViennaRNA/constraints/SHAPE.h](#) instead.

Deprecated Use [ViennaRNA/constraints/SHAPE.h](#) instead

18.46 constraints_SHAPE.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SHAPE_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/constraints_SHAPE.h>! Use
    <ViennaRNA/constraints/SHAPE.h> instead!"
00013 # endif
00014 #include <ViennaRNA/constraints/SHAPE.h>
00015 #endif
00016
00017 #endif
```

18.47 ViennaRNA/constraints_soft.h File Reference

Use [ViennaRNA/constraints/soft.h](#) instead.

Include dependency graph for constraints_soft.h:

18.47.1 Detailed Description

Use [ViennaRNA/constraints/soft.h](#) instead.

Deprecated Use [ViennaRNA/constraints/soft.h](#) instead

18.48 constraints_soft.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_SOFT_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/constraints_soft.h>! Use
    <ViennaRNA/constraints/soft.h> instead!"
00013 # endif
00014 #include <ViennaRNA/constraints/soft.h>
00015 #endif
00016
00017 #endif
```

18.49 ViennaRNA/convert_epars.h File Reference

Use [ViennaRNA/params/convert.h](#) instead.

Include dependency graph for convert_epars.h:

18.49.1 Detailed Description

Use [ViennaRNA/params/convert.h](#) instead.

Deprecated Use [ViennaRNA/params/convert.h](#) instead

18.50 convert_epars.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONVERT_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_CONVERT_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/convert_epars.h>! Use
    <ViennaRNA/params/convert.h> instead!"
00013 # endif
00014 #include <ViennaRNA/params/convert.h>
00015 #endif
00016
00017 #endif
```

18.51 ViennaRNA/data_structures.h File Reference

Use [ViennaRNA/datastructures/basic.h](#) instead.

Include dependency graph for data_structures.h:

18.51.1 Detailed Description

Use [ViennaRNA/datastructures/basic.h](#) instead.

Deprecated Use [ViennaRNA/datastructures/basic.h](#) instead

18.52 data_structures.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/data_structures.h>! Use
    <ViennaRNA/datastructures/basic.h> instead!"
00013 # endif
00014 #include <ViennaRNA/datastructures/basic.h>
00015 #endif
00016
00017 #endif
```

18.53 ViennaRNA/datastructures/array.h File Reference

A macro-based dynamic array implementation.

Include dependency graph for array.h:

Data Structures

- struct [vrna_array_header_s](#)
The header of an array. [More...](#)

Macros

- #define [vrna_array](#)(Type) Type *
Define an array.
- #define [vrna_array_make](#)(Type, Name) Type * Name; [vrna_array_init](#)(Name)
Make an array *Name* of type *Type*.
- #define [VRNA_ARRAY_GROW_FORMULA](#)(n) (1.4 * (n) + 8)
The default growth formula for array.
- #define [VRNA_ARRAY_HEADER](#)(input) (([vrna_array_header_t](#) *) (input) - 1)
Retrieve a pointer to the header of an array *input*.
- #define [vrna_array_size](#)(input) ([VRNA_ARRAY_HEADER](#)(input)->num)
Get the number of elements of an array *input*.
- #define [vrna_array_capacity](#)(input) ([VRNA_ARRAY_HEADER](#)(input)->size)
Get the size of an array *input*, i.e. its actual capacity.
- #define [vrna_array_set_capacity](#)(a, capacity)
Explicitly set the capacity of an array *a*.
- #define [vrna_array_init_size](#)(a, init_size)
Initialize an array *a* with a particular pre-allocated size *init_size*.
- #define [vrna_array_init](#)(a) [vrna_array_init_size](#)(a, [VRNA_ARRAY_GROW_FORMULA](#)(0));
Initialize an array *a*.
- #define [vrna_array_free](#)(a)

*Release memory of an array *a*.*

- **#define `vrna_array_append`**(*a*, *item*)

*Safely append an item to an array *a*.*

- **#define `vrna_array_grow`**(*a*, *min_capacity*)

*Grow an array *a* to provide a minimum capacity *min_capacity*.*

Typedefs

- typedef struct `vrna_array_header_s` `vrna_array_header_t`

The header of an array.

Functions

- `VRNA_NO_INLINE` void * `vrna__array_set_capacity` (void *array, size_t capacity, size_t element_size)

Explicitly set the capacity of an array.

18.53.1 Detailed Description

A macro-based dynamic array implementation.

18.54 `array.h`

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_ARRAY_H
00002 #define VIENNA_RNA_PACKAGE_ARRAY_H
00003
00004 #include <stddef.h>
00005
00006
00007 #if !defined(VRNA_NO_INLINE)
00008     #if defined(_MSC_VER)
00009         #define VRNA_NO_INLINE __declspec(noinline)
00010     #else
00011         #define VRNA_NO_INLINE __attribute__((noinline))
00012     #endif
00013 #endif
00014
00091 typedef struct vrna_array_header_s {
00092     size_t  num;
00093     size_t  size;
00094 } vrna_array_header_t;
00095
00099 #define vrna_array(Type) Type *
00100
00104 #define vrna_array_make(Type, Name) Type * Name; vrna_array_init(Name)
00105
00106
00107 #ifndef VRNA_ARRAY_GROW_FORMULA
00111 #define VRNA_ARRAY_GROW_FORMULA(n) (1.4 * (n) + 8)
00112 #endif
00113
00117 #define VRNA_ARRAY_HEADER(input) ((vrna_array_header_t *) (input) - 1)
00121 #define vrna_array_size(input) (VRNA_ARRAY_HEADER(input)->num)
00125 #define vrna_array_capacity(input) (VRNA_ARRAY_HEADER(input)->size)
00126
00130 #define vrna_array_set_capacity(a, capacity) do { \
00131     if (a) { \
00132         void **a_ptr = (void **)&(a); \
00133         *a_ptr = vrna__array_set_capacity((a), (capacity), sizeof(*(a))); \
00134     } \
00135 } while (0)
00136
00137
00143 VRNA_NO_INLINE void *
00144 vrna__array_set_capacity(void *array,
00145                          size_t capacity,
00146                          size_t element_size);
00147
00152 #define vrna_array_init_size(a, init_size) do { \
00153     void **a_ptr = (void **)&(a); \
00154     size_t size = sizeof(*(a)) * (init_size) + sizeof(vrna_array_header_t); \
00155     vrna_array_header_t *h = (void *)vrna_alloc(size); \
```



```

00156     h->num           = 0; \
00157     h->size          = init_size; \
00158     *a_ptr           = (void *) (h + 1); \
00159 } while (0)
00160
00164 #define vrna_array_init(a)  vrna_array_init_size(a, VRNA_ARRAY_GROW_FORMULA(0));
00165
00166
00170 #define vrna_array_free(a) do { \
00171     vrna_array_header_t *h = VRNA_ARRAY_HEADER(a); \
00172     free(h); \
00173 } while (0)
00174
00175
00179 #define vrna_array_append(a, item) do { \
00180     if (vrna_array_capacity(a) < vrna_array_size(a) + 1) \
00181         vrna_array_grow(a, 0); \
00182     (a)[vrna_array_size(a)++] = (item); \
00183 } while (0)
00184
00185
00189 #define vrna_array_grow(a, min_capacity) do { \
00190     size_t new_capacity = VRNA_ARRAY_GROW_FORMULA(vrna_array_capacity(a)); \
00191     if (new_capacity < (min_capacity)) \
00192         new_capacity = (min_capacity); \
00193     vrna_array_set_capacity(a, new_capacity); \
00194 } while (0)
00195
00201 #endif

```

18.55 ViennaRNA/datastructures/hash_tables.h File Reference

Implementations of hash table functions.

Data Structures

- struct [vrna_ht_entry_db_t](#)
Default hash table entry. [More...](#)

Functions

Dot-Bracket / Free Energy entries

- int [vrna_ht_db_comp](#) (void *x, void *y)
Default hash table entry comparison.
- unsigned int [vrna_ht_db_hash_func](#) (void *x, unsigned long hashtable_size)
Default hash function.
- int [vrna_ht_db_free_entry](#) (void *hash_entry)
Default function to free memory occupied by a hash entry.

Abstract interface

- typedef struct vrna_hash_table_s * [vrna_hash_table_t](#)
A hash table object.
- typedef int(* [vrna_ht_cmp_f](#)) (void *x, void *y)
Callback function to compare two hash table entries.
- typedef int() [vrna_callback_ht_compare_entries](#)(void *x, void *y)
- typedef unsigned int(* [vrna_ht_hashfunc_f](#)) (void *x, unsigned long hashtable_size)
Callback function to generate a hash key, i.e. hash function.
- typedef unsigned int() [vrna_callback_ht_hash_function](#)(void *x, unsigned long hashtable_size)
- typedef int(* [vrna_ht_free_f](#)) (void *x)
Callback function to free a hash table entry.
- typedef int() [vrna_callback_ht_free_entry](#)(void *x)
- [vrna_hash_table_t](#) [vrna_ht_init](#) (unsigned int b, [vrna_ht_cmp_f](#) compare_function, [vrna_ht_hashfunc_f](#) hash_function, [vrna_ht_free_f](#) free_hash_entry)

Get an initialized hash table.

- unsigned long [vrna_ht_size](#) ([vrna_hash_table_t](#) ht)

Get the size of the hash table.

- unsigned long [vrna_ht_collisions](#) (struct [vrna_hash_table_s](#) *ht)

Get the number of collisions in the hash table.

- void * [vrna_ht_get](#) ([vrna_hash_table_t](#) ht, void *x)

Get an element from the hash table.

- int [vrna_ht_insert](#) ([vrna_hash_table_t](#) ht, void *x)

Insert an object into a hash table.

- void [vrna_ht_remove](#) ([vrna_hash_table_t](#) ht, void *x)

Remove an object from the hash table.

- void [vrna_ht_clear](#) ([vrna_hash_table_t](#) ht)

Clear the hash table.

- void [vrna_ht_free](#) ([vrna_hash_table_t](#) ht)

Free all memory occupied by the hash table.

18.55.1 Detailed Description

Implementations of hash table functions.

18.56 hash_tables.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_HASH_UTIL_H
00002 #define VIENNA_RNA_PACKAGE_HASH_UTIL_H
00003
00004 /* Taken from the barriers tool and modified by GE. */
00005
00006 #ifdef VRNA_WARN_DEPRECATED
00007 # if defined(DEPRECATED)
00008 #  undef DEPRECATED
00009 # endif
00010 # if defined(__clang__)
00011 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00012 # elif defined(__GNUC__)
00013 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00014 # else
00015 #  define DEPRECATED(func, msg) func
00016 # endif
00017 #else
00018 # define DEPRECATED(func, msg) func
00019 #endif
00020
00050 typedef struct vrna_hash_table_s *vrna_hash_table_t;
00051
00052
00060 typedef int (*vrna_ht_cmp_f)(void *x,
00061                               void *y);
00062
00063 DEPRECATED(typedef int (vrna_callback_ht_compare_entries)(void *x,
00064                                                           void *y),
00065            "Use vrna_ht_cmp_f instead!");
00066
00067
00068
00076 typedef unsigned int (*vrna_ht_hashfunc_f)(void *x,
00077                                             unsigned long hashtable_size);
00078
00079 DEPRECATED(typedef unsigned int (vrna_callback_ht_hash_function)(void *x,
00080                                                                 unsigned long hashtable_size),
00081            "Use vrna_ht_hashfunc_f instead!");
00082
00083
00090 typedef int (*vrna_ht_free_f)(void *x);
00091
00092 DEPRECATED(typedef int (vrna_callback_ht_free_entry)(void *x),
00093            "Use vrna_ht_free_f instead!");
00094
00095
00121 vrna_hash_table_t
00122 vrna_ht_init(unsigned int b,
00123              vrna_ht_cmp_f compare_function,
```

```

00124         vrna_ht_hashfunc_f    hash_function,
00125         vrna_ht_free_f        free_hash_entry);
00126
00127
00134 unsigned long
00135 vrna_ht_size(vrna_hash_table_t ht);
00136
00137
00144 unsigned long
00145 vrna_ht_collisions(struct vrna_hash_table_s *ht);
00146
00147
00162 void *
00163 vrna_ht_get(vrna_hash_table_t ht,
00164            void *x);
00165
00166
00182 int
00183 vrna_ht_insert(vrna_hash_table_t ht,
00184              void *x);
00185
00186
00198 void
00199 vrna_ht_remove(vrna_hash_table_t ht,
00200              void *x);
00201
00202
00214 void
00215 vrna_ht_clear(vrna_hash_table_t ht);
00216
00217
00228 void
00229 vrna_ht_free(vrna_hash_table_t ht);
00230
00231
00232 /* End of abstract interface */
00244 typedef struct {
00245     char *structure;
00246     float energy;
00247 } vrna_ht_entry_db_t;
00248
00249
00263 int
00264 vrna_ht_db_comp(void *x,
00265               void *y);
00266
00267
00282 unsigned int
00283 vrna_ht_db_hash_func(void *x,
00284                   unsigned long hashtable_size);
00285
00286
00298 int vrna_ht_db_free_entry(void *hash_entry);
00299
00300
00301 /* End of dot-bracket interface */
00308 #endif

```

18.57 ViennaRNA/datastructures/heap.h File Reference

Implementation of an abstract heap data structure.

Typedefs

- typedef struct vrna_heap_s * [vrna_heap_t](#)
An abstract heap data structure.
- typedef int(* [vrna_heap_cmp_f](#)) (const void *a, const void *b, void *data)
Heap compare function prototype.
- typedef size_t(* [vrna_heap_get_pos_f](#)) (const void *a, void *data)
Retrieve the position of a particular heap entry within the heap.
- typedef void(* [vrna_heap_set_pos_f](#)) (const void *a, size_t pos, void *data)
Store the position of a particular heap entry within the heap.

Functions

- `vrna_heap_t vrna_heap_init` (`size_t n`, `vrna_heap_cmp_f cmp`, `vrna_heap_get_pos_f get_entry_pos`, `vrna_heap_set_pos_f set_entry_pos`, `void *data`)
Initialize a heap data structure.
- `void vrna_heap_free` (`vrna_heap_t h`)
Free memory occupied by a heap data structure.
- `size_t vrna_heap_size` (`struct vrna_heap_s *h`)
Get the size of a heap data structure, i.e. the number of stored elements.
- `void vrna_heap_insert` (`vrna_heap_t h`, `void *v`)
Insert an element into the heap.
- `void * vrna_heap_pop` (`vrna_heap_t h`)
Pop (remove and return) the object at the root of the heap.
- `const void * vrna_heap_top` (`vrna_heap_t h`)
Get the object at the root of the heap.
- `void * vrna_heap_remove` (`vrna_heap_t h`, `const void *v`)
Remove an arbitrary element within the heap.
- `void * vrna_heap_update` (`vrna_heap_t h`, `void *v`)
Update an arbitrary element within the heap.

18.57.1 Detailed Description

Implementation of an abstract heap data structure.

18.58 heap.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_HEAP_H
00002 #define VIENNA_RNA_PACKAGE_HEAP_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)
00009 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00010 # elif defined(__GNUC__)
00011 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00012 # else
00013 #   define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
00018
00040 typedef struct vrna_heap_s *vrna_heap_t;
00041
00042
00059 typedef int (*vrna_heap_cmp_f)(const void *a,
00060                               const void *b,
00061                               void *data);
00062
00063 DEPRECATED(typedef int (vrna_callback_heap_cmp)(const void *a,
00064                                                const void *b,
00065                                                void *data),
00066           "Use vrna_heap_cmp_f instead!");
00067
00068
00076 typedef size_t (*vrna_heap_get_pos_f)(const void *a,
00077                                       void *data);
00078
00079 DEPRECATED(typedef size_t (vrna_callback_heap_get_pos)(const void *a,
00080                                                       void *data),
00081           "Use vrna_heap_get_pos_f instead!");
00082
00083
00091 typedef void (*vrna_heap_set_pos_f)(const void *a,
00092                                     size_t pos,
00093                                     void *data);
00094
```

```

00095 DEPRECATED(typedef void (vrna_callback_heap_set_pos)(const void *a,
00096                                                         size_t    pos,
00097                                                         void      *data),
00098               "USe vrna_heap_set_pos_f instead!");
00099
00100
00134 vrna_heap_t
00135 vrna_heap_init(size_t          n,
00136               vrna_heap_cmp_f  cmp,
00137               vrna_heap_get_pos_f get_entry_pos,
00138               vrna_heap_set_pos_f set_entry_pos,
00139               void              *data);
00140
00141
00149 void
00150 vrna_heap_free(vrna_heap_t h);
00151
00152
00159 size_t
00160 vrna_heap_size(struct vrna_heap_s *h);
00161
00162
00172 void
00173 vrna_heap_insert(vrna_heap_t h,
00174                 void         *v);
00175
00176
00188 void *
00189 vrna_heap_pop(vrna_heap_t h);
00190
00191
00201 const void *
00202 vrna_heap_top(vrna_heap_t h);
00203
00204
00218 void *
00219 vrna_heap_remove(vrna_heap_t h,
00220                  const void  *v);
00221
00222
00239 void *
00240 vrna_heap_update(vrna_heap_t h,
00241                 void         *v);
00242
00243
00248 #endif

```

18.59 lists.h

```

00001 /*
00002  $Log: lists.h,v $
00003  Revision 1.2  2000/10/10 08:50:01  ivo
00004  some annotation for lclint
00005
00006  Revision 1.1  1997/08/04 21:05:32  walter
00007  Initial revision
00008
00009 */
00010
00011 #ifndef __LIST_H
00012 #define __LIST_H
00013
00014 /*----- Macros and type definitions -----*/
00015
00016 typedef struct LST_BUCKET {
00017     struct LST_BUCKET *next;
00018 }
00019 LST_BUCKET;
00020
00021 typedef struct {
00022     int count; /* Number of elements currently in list */
00023     LST_BUCKET *head; /* Pointer to head element of list */
00024     LST_BUCKET *z; /* Pointer to last node of list */
00025     LST_BUCKET hz[2]; /* Space for head and z nodes */
00026 }
00027 LIST;
00028
00029 /* Return a pointer to the user space given the address of the header of
00030  * a node.
00031  */
00032
00033 #define LST_USERSPACE(h) ((void*)((LST_BUCKET*)(h) + 1))
00034
00035 /* Return a pointer to the header of a node, given the address of the
00036  * user space.

```

```

00037  */
00038
00039 #define LST_HEADER(n)    ((LST_BUCKET*)(n) - 1)
00040
00041 /* Return a pointer to the user space of the list's head node. This user
00042  * space does not actually exist, but it is useful to be able to address
00043  * it to enable insertion at the start of the list.
00044  */
00045
00046 #define LST_HEAD(l)     LST_USERSPACE((l)->head)
00047
00048 /* Determine if a list is empty
00049  */
00050
00051 #define LST_EMPTY(l)     ((l)->count == 0)
00052
00053 /*----- Function Prototypes -----*/
00054
00055 /*@only@*//*@out@*/ void *lst_newnode (int size);
00056 void lst_freemode (/*@only@*/ void *node);
00057 /*@only@*//*@out@*/ LIST *lst_init (void);
00058 void lst_kill (LIST * l, void (*freeNode) ());
00059 void lst_insertafter (LIST * l, /*@keep@*/ void *node, void *after);
00060 void *lst_deletenext (/*@only@*/ LIST * l, void *node);
00061 /*@dependent@*/ void *lst_first (LIST * l);
00062 /*@dependent@*/ void *lst_next (void *prev);
00063 void lst_mergesort (LIST * l, int (*cmp_func) ());
00064
00065 #endif

```

18.60 string.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_STRING_H
00002 #define VIENNA_RNA_PACKAGE_STRING_H
00003
00004 #include <stddef.h>
00005 #include <string.h>
00006
00007 typedef char *vrna_string_t;
00008
00012 typedef struct vrna_string_header_s {
00013     size_t  len;
00014     size_t  size;
00015     size_t  shift_post;
00016     char    backup;
00017 } vrna_string_header_t;
00018
00019
00020 #define VRNA_STRING_HEADER(s) ((vrna_string_header_t *)s - 1)
00021
00022 vrna_string_t
00023 vrna_string_make(char const *str);
00024
00025 void
00026 vrna_string_free(vrna_string_t str);
00027
00028 vrna_string_t
00029 vrna_string_append(vrna_string_t str,
00030                  vrna_string_t const other);
00031
00032 vrna_string_t
00033 vrna_string_append_cstring(vrna_string_t str,
00034                          char const *other);
00035
00036
00037 #endif

```

18.61 ViennaRNA/dist_vars.h File Reference

Global variables for Distance-Package.

This graph shows which files directly or indirectly include this file:

Data Structures

- struct [Postorder_list](#)
Postorder data structure.
- struct [Tree](#)

Tree data structure.

- struct `swString`

Some other data structure.

Variables

- int `edit_backtrack`

Produce an alignment of the two structures being compared by tracing the editing path giving the minimum distance.

- char * `aligned_line` [4]

Contains the two aligned structures after a call to one of the distance functions with `edit_backtrack` set to 1.

- int `cost_matrix`

Specify the cost matrix to be used for distance calculations.

18.61.1 Detailed Description

Global variables for Distance-Package.

18.61.2 Variable Documentation

18.61.2.1 edit_backtrack

```
int edit_backtrack [extern]
```

Produce an alignment of the two structures being compared by tracing the editing path giving the minimum distance. set to 1 if you want backtracking

18.61.2.2 cost_matrix

```
int cost_matrix [extern]
```

Specify the cost matrix to be used for distance calculations.

if 0, use the default cost matrix (upper matrix in example), otherwise use Shapiro's costs (lower matrix).

18.62 dist_vars.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_DIST_VARS_H
00002 #define VIENNA_RNA_PACKAGE_DIST_VARS_H
00003
00009 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00010
00017 extern int    edit_backtrack;
00018
00023 extern char  *aligned_line[4];
00024
00031 extern int    cost_matrix;
00032
00033 /* Global type defs for Distance-Package */
00034
00038 typedef struct {
00039     int    type;
00040     int    weight;
00041     int    father;
00042     int    sons;
00043     int    leftmostleaf;
00044 } Postorder_list;
00045
00049 typedef struct {
00050     Postorder_list *postorder_list;
00051     int             *keyroots;
00052 } Tree;
00053
00057 typedef struct {
00058     int    type;
00059     int    sign;
00060     float  weight;
00061 } swString;
```

```
00062 #endif
00063
00064 #endif
```

18.63 ViennaRNA/dp_matrices.h File Reference

Functions to deal with standard dynamic programming (DP) matrices.

Include dependency graph for dp_matrices.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_mx_mfe_s](#)
Minimum Free Energy (MFE) Dynamic Programming (DP) matrices data structure required within the [vrna_fold_compound_t](#). [More...](#)
- struct [vrna_mx_pf_s](#)
Partition function (PF) Dynamic Programming (DP) matrices data structure required within the [vrna_fold_compound_t](#). [More...](#)

Typedefs

- typedef struct [vrna_mx_mfe_s](#) [vrna_mx_mfe_t](#)
Typename for the Minimum Free Energy (MFE) DP matrices data structure [vrna_mx_mfe_s](#).
- typedef struct [vrna_mx_pf_s](#) [vrna_mx_pf_t](#)
Typename for the Partition Function (PF) DP matrices data structure [vrna_mx_pf_s](#).

Enumerations

- enum [vrna_mx_type_e](#) { [VRNA_MX_DEFAULT](#) , [VRNA_MX_WINDOW](#) , [VRNA_MX_2DFOLD](#) }
An enumerator that is used to specify the type of a polymorphic Dynamic Programming (DP) matrix data structure.

Functions

- int [vrna_mx_add](#) ([vrna_fold_compound_t](#) *vc, [vrna_mx_type_e](#) type, unsigned int options)
Add Dynamic Programming (DP) matrices (allocate memory)
- void [vrna_mx_mfe_free](#) ([vrna_fold_compound_t](#) *vc)
Free memory occupied by the Minimum Free Energy (MFE) Dynamic Programming (DP) matrices.
- void [vrna_mx_pf_free](#) ([vrna_fold_compound_t](#) *vc)
Free memory occupied by the Partition Function (PF) Dynamic Programming (DP) matrices.

18.63.1 Detailed Description

Functions to deal with standard dynamic programming (DP) matrices.

18.64 dp_matrices.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_DP_MATRICES_H
00002 #define VIENNA_RNA_PACKAGE_DP_MATRICES_H
00003
00020 typedef struct  vrna_mx_mfe_s vrna_mx_mfe_t;
00022 typedef struct  vrna_mx_pf_s  vrna_mx_pf_t;
00023
00024 #include <ViennaRNA/datastructures/basic.h>
00025 #include <ViennaRNA/fold_compound.h>
00026
00032 typedef enum {
00033     VRNA_MX_DEFAULT,
00034     VRNA_MX_WINDOW,
00038     VRNA_MX_2DFOLD
00041 } vrna_mx_type_e;
```



```
00042
00046 struct vrna_mx_mfe_s {
00050     const vrna_mx_type_e type;
00051     unsigned int length;
00052     unsigned int strands;
00057 #ifndef VRNA_DISABLE_C11_FEATURES
00058     /* C11 support for unnamed unions/structs */
00059     union {
00060         struct {
00061 #endif
00067         int *c;
00068         int *f5;
00069         int *f3;
00070         int **fms5;
00071         int **fms3;
00072         int *fML;
00073         int *fM1;
00074         int *fM2;
00075         int *ggg;
00076         int Fc;
00077         int FcH;
00078         int FcI;
00079         int FcM;
00084 #ifndef VRNA_DISABLE_C11_FEATURES
00085     /* C11 support for unnamed unions/structs */
00086 };
00087 struct {
00088 #endif
00094     int **c_local;
00095     int *f3_local;
00096     int **fML_local;
00097     int **ggg_local;
00101 #ifndef VRNA_DISABLE_C11_FEATURES
00102     /* C11 support for unnamed unions/structs */
00103 };
00104 struct {
00105 #endif
00106
00112     int ***E_F5;
00113     int **l_min_F5;
00114     int **l_max_F5;
00115     int *k_min_F5;
00116     int *k_max_F5;
00117
00118     int ***E_F3;
00119     int **l_min_F3;
00120     int **l_max_F3;
00121     int *k_min_F3;
00122     int *k_max_F3;
00123
00124     int ***E_C;
00125     int **l_min_C;
00126     int **l_max_C;
00127     int *k_min_C;
00128     int *k_max_C;
00129
00130     int ***E_M;
00131     int **l_min_M;
00132     int **l_max_M;
00133     int *k_min_M;
00134     int *k_max_M;
00135
00136     int ***E_M1;
00137     int **l_min_M1;
00138     int **l_max_M1;
00139     int *k_min_M1;
00140     int *k_max_M1;
00141
00142     int ***E_M2;
00143     int **l_min_M2;
00144     int **l_max_M2;
00145     int *k_min_M2;
00146     int *k_max_M2;
00147
00148     int **E_Fc;
00149     int *l_min_Fc;
00150     int *l_max_Fc;
00151     int k_min_Fc;
00152     int k_max_Fc;
00153
00154     int **E_FcH;
00155     int *l_min_FcH;
00156     int *l_max_FcH;
00157     int k_min_FcH;
00158     int k_max_FcH;
00159
00160     int **E_FcI;
```

```

00161     int             *l_min_FcI;
00162     int             *l_max_FcI;
00163     int             k_min_FcI;
00164     int             k_max_FcI;
00165
00166     int             **E_FcM;
00167     int             *l_min_FcM;
00168     int             *l_max_FcM;
00169     int             k_min_FcM;
00170     int             k_max_FcM;
00171
00172     /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
00173     int             *E_F5_rem;
00174     int             *E_F3_rem;
00175     int             *E_C_rem;
00176     int             *E_M_rem;
00177     int             *E_M1_rem;
00178     int             *E_M2_rem;
00179
00180     int             E_Fc_rem;
00181     int             E_FcH_rem;
00182     int             E_FcI_rem;
00183     int             E_FcM_rem;
00184
00185     #ifdef COUNT_STATES
00186     unsigned long   ***N_F5;
00187     unsigned long   ***N_C;
00188     unsigned long   ***N_M;
00189     unsigned long   ***N_M1;
00190     #endif
00191
00196     #ifndef VRNA_DISABLE_C11_FEATURES
00197     /* C11 support for unnamed unions/structs */
00198     };
00199     };
00200     #endif
00201     };
00202
00206     struct vrna_mx_pf_s {
00210     const vrna_mx_type_e type;
00211     unsigned int         length;
00212     FLT_OR_DBL           *scale;
00213     FLT_OR_DBL           *expMLbase;
00219     #ifndef VRNA_DISABLE_C11_FEATURES
00220     /* C11 support for unnamed unions/structs */
00221     union {
00222         struct {
00223     #endif
00224
00230     FLT_OR_DBL *q;
00231     FLT_OR_DBL *qb;
00232     FLT_OR_DBL *qm;
00233     FLT_OR_DBL *qm1;
00234     FLT_OR_DBL *probs;
00235     FLT_OR_DBL *qlk;
00236     FLT_OR_DBL *qln;
00237     FLT_OR_DBL *G;
00238
00239     FLT_OR_DBL qo;
00240     FLT_OR_DBL *qm2;
00241     FLT_OR_DBL qho;
00242     FLT_OR_DBL qio;
00243     FLT_OR_DBL qmo;
00244
00249     #ifndef VRNA_DISABLE_C11_FEATURES
00250     /* C11 support for unnamed unions/structs */
00251     };
00252     struct {
00253     #endif
00254
00260     FLT_OR_DBL **q_local;
00261     FLT_OR_DBL **qb_local;
00262     FLT_OR_DBL **qm_local;
00263     FLT_OR_DBL **pR;
00264     FLT_OR_DBL **qm2_local;
00265     FLT_OR_DBL **QI5;
00266     FLT_OR_DBL **q2l;
00267     FLT_OR_DBL **qmb;
00268     FLT_OR_DBL **G_local;
00273     #ifndef VRNA_DISABLE_C11_FEATURES
00274     /* C11 support for unnamed unions/structs */
00275     };
00276     struct {
00277     #endif
00278
00284     FLT_OR_DBL ***Q;
00285     int **l_min_Q;

```

```
00286     int **l_max_Q;
00287     int *k_min_Q;
00288     int *k_max_Q;
00289
00290
00291     FLT_OR_DBL ***Q_B;
00292     int **l_min_Q_B;
00293     int **l_max_Q_B;
00294     int *k_min_Q_B;
00295     int *k_max_Q_B;
00296
00297     FLT_OR_DBL ***Q_M;
00298     int **l_min_Q_M;
00299     int **l_max_Q_M;
00300     int *k_min_Q_M;
00301     int *k_max_Q_M;
00302
00303     FLT_OR_DBL ***Q_M1;
00304     int **l_min_Q_M1;
00305     int **l_max_Q_M1;
00306     int *k_min_Q_M1;
00307     int *k_max_Q_M1;
00308
00309     FLT_OR_DBL ***Q_M2;
00310     int **l_min_Q_M2;
00311     int **l_max_Q_M2;
00312     int *k_min_Q_M2;
00313     int *k_max_Q_M2;
00314
00315     FLT_OR_DBL **Q_c;
00316     int *l_min_Q_c;
00317     int *l_max_Q_c;
00318     int *k_min_Q_c;
00319     int *k_max_Q_c;
00320
00321     FLT_OR_DBL **Q_cH;
00322     int *l_min_Q_cH;
00323     int *l_max_Q_cH;
00324     int *k_min_Q_cH;
00325     int *k_max_Q_cH;
00326
00327     FLT_OR_DBL **Q_cI;
00328     int *l_min_Q_cI;
00329     int *l_max_Q_cI;
00330     int *k_min_Q_cI;
00331     int *k_max_Q_cI;
00332
00333     FLT_OR_DBL **Q_cM;
00334     int *l_min_Q_cM;
00335     int *l_max_Q_cM;
00336     int *k_min_Q_cM;
00337     int *k_max_Q_cM;
00338
00339     /* auxiliary arrays for remaining set of coarse graining (k,l) > (k_max, l_max) */
00340     FLT_OR_DBL *Q_rem;
00341     FLT_OR_DBL *Q_B_rem;
00342     FLT_OR_DBL *Q_M_rem;
00343     FLT_OR_DBL *Q_M1_rem;
00344     FLT_OR_DBL *Q_M2_rem;
00345
00346     FLT_OR_DBL Q_c_rem;
00347     FLT_OR_DBL Q_cH_rem;
00348     FLT_OR_DBL Q_cI_rem;
00349     FLT_OR_DBL Q_cM_rem;
00350 #ifndef VRNA_DISABLE_C11_FEATURES
00351     /* C11 support for unnamed unions/structs */
00352 };
00353 };
00354 #endif
00355 };
00356
00357 int
00358 vrna_mx_add(vrna_fold_compound_t *vc,
00359             vrna_mx_type_e type,
00360             unsigned int options);
00361
00362 int
00363 vrna_mx_mfe_add(vrna_fold_compound_t *vc,
00364                vrna_mx_type_e mx_type,
00365                unsigned int options);
00366
00367 int
00368 vrna_mx_pf_add(vrna_fold_compound_t *vc,
00369               vrna_mx_type_e mx_type,
00370               unsigned int options);
```

```

00406
00407
00408 int
00409 vrna_mx_prepare(vrna_fold_compound_t *vc,
00410                unsigned int options);
00411
00412
00420 void
00421 vrna_mx_mfe_free(vrna_fold_compound_t *vc);
00422
00423
00431 void
00432 vrna_mx_pf_free(vrna_fold_compound_t *vc);
00433
00434
00439 #endif

```

18.65 ViennaRNA/duplex.h File Reference

Functions for simple RNA-RNA duplex interactions.
 Include dependency graph for duplex.h:

18.65.1 Detailed Description

Functions for simple RNA-RNA duplex interactions.

18.66 duplex.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_DUPLICATION_H
00002 #define VIENNA_RNA_PACKAGE_DUPLICATION_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005
00006 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00007
00015 duplexT duplexfold(const char *s1,
00016                   const char *s2);
00017
00018
00019 duplexT *duplex_subopt(const char *s1,
00020                      const char *s2,
00021                      int delta,
00022                      int w);
00023
00024
00025 duplexT aliduplexfold(const char *s1[],
00026                     const char *s2[]);
00027
00028
00029 duplexT *aliduplex_subopt(const char *s1[],
00030                          const char *s2[],
00031                          int delta,
00032                          int w);
00033
00034
00035 #endif
00036
00037 #endif

```

18.67 ViennaRNA/edit_cost.h File Reference

global variables for Edit Costs included by treedist.c and stringdist.c

18.67.1 Detailed Description

global variables for Edit Costs included by treedist.c and stringdist.c

18.68 edit_cost.h

[Go to the documentation of this file.](#)

```

00001
00006 #define PRIVATE static
00007
00008 PRIVATE char sep = ':';
00009 PRIVATE char *coding = "Null:U:P:H:B:I:M:S:E:R";
00010
00011 #define DIST_INF 10000 /* infinity */
00012
00013 typedef int CostMatrix[10][10];
00014
00015 PRIVATE CostMatrix *EditCost; /* will point to UsualCost or ShapiroCost */
00016
00017 PRIVATE CostMatrix UsualCost =
00018 {
00019
00020 /*      Null,      U,      P,      H,      B,      I,      M,      S,      E,      R
00021 */
00022 { 0,      1,      2,      2,      2,      2,      2,      1,      1,
DIST_INF}, /* Null replaced */
00023 { 1,      0,      1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,
DIST_INF}, /* U replaced */
00024 { 2,      1,      0, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,
DIST_INF}, /* P replaced */
00025 { 2, DIST_INF, DIST_INF, 0,      2,      2,      2, DIST_INF, DIST_INF,
DIST_INF}, /* H replaced */
00026 { 2, DIST_INF, DIST_INF, 2,      0,      1,      2, DIST_INF, DIST_INF,
DIST_INF}, /* B replaced */
00027 { 2, DIST_INF, DIST_INF, 2,      1,      0,      2, DIST_INF, DIST_INF,
DIST_INF}, /* I replaced */
00028 { 2, DIST_INF, DIST_INF, 2,      2,      2,      0, DIST_INF, DIST_INF,
DIST_INF}, /* M replaced */
00029 { 1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0, DIST_INF,
DIST_INF}, /* S replaced */
00030 { 1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0,
DIST_INF}, /* E replaced */
00031 { DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,
0}, /* R replaced */
00032
00033 };
00034
00035
00036 PRIVATE CostMatrix ShapiroCost =
00037 {
00038
00039 /*      Null,      U,      P,      H,      B,      I,      M,      S,      E,      R
00040 */
00041 { 0,      1,      2,      100,      5,      5,      75,      5,      5,
DIST_INF}, /* Null replaced */
00042 { 1,      0,      1, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,
DIST_INF}, /* U replaced */
00043 { 2,      1,      0, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,
DIST_INF}, /* P replaced */
00044 { 100, DIST_INF, DIST_INF, 0,      8,      8,      8, DIST_INF, DIST_INF,
DIST_INF}, /* H replaced */
00045 { 5, DIST_INF, DIST_INF, 8,      0,      3,      8, DIST_INF, DIST_INF,
DIST_INF}, /* B replaced */
00046 { 5, DIST_INF, DIST_INF, 8,      3,      0,      8, DIST_INF, DIST_INF,
DIST_INF}, /* I replaced */
00047 { 75, DIST_INF, DIST_INF, 8,      8,      8,      0, DIST_INF, DIST_INF,
DIST_INF}, /* M replaced */
00048 { 5, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0, DIST_INF,
DIST_INF}, /* S replaced */
00049 { 5, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, 0,
DIST_INF}, /* E replaced */
00050 { DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF, DIST_INF,
0}, /* R replaced */
00051
00052 };
00053

```

18.69 ViennaRNA/energy_const.h File Reference

Use [ViennaRNA/params/constants.h](#) instead.

Include dependency graph for energy_const.h:

18.69.1 Detailed Description

Use [ViennaRNA/params/constants.h](#) instead.

Deprecated Use [ViennaRNA/params/constants.h](#) instead

18.70 energy_const.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/energy_const.h>! Use
    <ViennaRNA/params/constants.h> instead!"
00013 # endif
00014 #include <ViennaRNA/params/constants.h>
00015 #endif
00016
00017 #endif
```

18.71 ViennaRNA/energy_par.h File Reference

Use [ViennaRNA/params/default.h](#) instead.

Include dependency graph for energy_par.h:

18.71.1 Detailed Description

Use [ViennaRNA/params/default.h](#) instead.

Deprecated Use [ViennaRNA/params/default.h](#) instead

18.72 energy_par.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/energy_par.h>! Use <ViennaRNA/params/default.h>
    instead!"
00013 # endif
00014 #include <ViennaRNA/params/default.h>
00015 #endif
00016
00017 #endif
```

18.73 ViennaRNA/equilibrium_probs.h File Reference

Equilibrium Probability implementations.

Include dependency graph for equilibrium_probs.h: This graph shows which files directly or indirectly include this file:

Functions

Base pair probabilities and derived computations

- `int vrna_pairing_probs (vrna_fold_compound_t *vc, char *structure)`
- `double vrna_mean_bp_distance_pr (int length, FLT_OR_DBL *pr)`
Get the mean base pair distance in the thermodynamic ensemble from a probability matrix.
- `double vrna_mean_bp_distance (vrna_fold_compound_t *vc)`
Get the mean base pair distance in the thermodynamic ensemble.
- `double vrna_ensemble_defect_pt (vrna_fold_compound_t *fc, const short *pt)`
*Compute the Ensemble Defect for a given target structure provided as a **vrna_ptable**.*
- `double vrna_ensemble_defect (vrna_fold_compound_t *fc, const char *structure)`
Compute the Ensemble Defect for a given target structure.
- `double * vrna_positional_entropy (vrna_fold_compound_t *fc)`
Compute a vector of positional entropies.
- `vrna_ep_t * vrna_stack_prob (vrna_fold_compound_t *vc, double cutoff)`

Compute stacking probabilities.

Multimer probabilities computations

- void `vrna_pf_dimer_probs` (double FAB, double FA, double FB, `vrna_ep_t` *prAB, const `vrna_ep_t` *prA, const `vrna_ep_t` *prB, int Alength, const `vrna_exp_param_t` *exp_params)

Compute Boltzmann probabilities of dimerization without homodimers.

Structure probability computations

- double `vrna_pr_structure` (`vrna_fold_compound_t` *fc, const char *structure)

Compute the equilibrium probability of a particular secondary structure.

- double `vrna_pr_energy` (`vrna_fold_compound_t` *vc, double e)

18.73.1 Detailed Description

Equilibrium Probability implementations.

This file includes various implementations for equilibrium probability computations based on the partition function of an RNA sequence, two concatenated sequences, or a sequence alignment.

18.74 equilibrium_probs.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_EQUILIBRIUM_PROBS_H
00002 #define VIENNA_RNA_PACKAGE_EQUILIBRIUM_PROBS_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/fold_compound.h>
00006 #include <ViennaRNA/Utils/structures.h>
00007 #include <ViennaRNA/params/basic.h>
00008
00020 /*
00021 #####
00022 # BASE PAIR PROBABILITY RELATED FUNCTIONS #
00023 #####
00024 */
00025
00026
00062 int
00063 vrna_pairing_probs(vrna_fold_compound_t *vc,
00064                   char *structure);
00065
00066
00082 double
00083 vrna_mean_bp_distance_pr(int length,
00084                          FLT_OR_DBL *pr);
00085
00086
00101 double
00102 vrna_mean_bp_distance(vrna_fold_compound_t *vc);
00103
00104
00127 double
00128 vrna_ensemble_defect_pt(vrna_fold_compound_t *fc,
00129                        const short *pt);
00130
00131
00155 double
00156 vrna_ensemble_defect(vrna_fold_compound_t *fc,
00157                     const char *structure);
00158
00159
00182 double *
00183 vrna_positional_entropy(vrna_fold_compound_t *fc);
00184
00185
00196 vrna_ep_t *
00197 vrna_stack_prob(vrna_fold_compound_t *vc,
00198                double cutoff);
00199
00200
00201 /* End base pair related functions */
00227 void
00228 vrna_pf_dimer_probs(double FAB,
00229                    double FA,
00230                    double FB,
```

```

00231         vrna_ep_t           *prAB,
00232         const vrna_ep_t     *prA,
00233         const vrna_ep_t     *prB,
00234         int                 Alength,
00235         const vrna_exp_param_t *exp_params);
00236
00237
00238 /* End multimer probability related functions */
00268 double
00269 vrna_pr_structure(vrna_fold_compound_t *fc,
00270                  const char *structure);
00271
00272
00273 double
00274 vrna_pr_energy(vrna_fold_compound_t *vc,
00275                double e);
00276
00277
00278 /* End structure probability related functions */
00281 /* End thermodynamics */
00284 #endif

```

18.75 ViennaRNA/eval.h File Reference

Functions and variables related to energy evaluation of sequence/structure pairs.

Include dependency graph for eval.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_VERBOSITY_QUIET -1`
Quiet level verbosity setting.
- `#define VRNA_VERBOSITY_DEFAULT 1`
Default level verbosity setting.

Functions

- `int vrna_eval_loop_pt(vrna_fold_compound_t *fc, int i, const short *pt)`
Calculate energy of a loop.
- `int vrna_eval_loop_pt_v(vrna_fold_compound_t *fc, int i, const short *pt, int verbosity_level)`
Calculate energy of a loop.
- `float vrna_eval_move(vrna_fold_compound_t *fc, const char *structure, int m1, int m2)`
Calculate energy of a move (closing or opening of a base pair)
- `int vrna_eval_move_pt(vrna_fold_compound_t *fc, short *pt, int m1, int m2)`
Calculate energy of a move (closing or opening of a base pair)
- `float energy_of_structure(const char *string, const char *structure, int verbosity_level)`
Calculate the free energy of an already folded RNA using global model detail settings.
- `float energy_of_struct_par(const char *string, const char *structure, vrna_param_t *parameters, int verbosity_level)`
Calculate the free energy of an already folded RNA.
- `float energy_of_circ_structure(const char *string, const char *structure, int verbosity_level)`
Calculate the free energy of an already folded circular RNA.
- `float energy_of_circ_struct_par(const char *string, const char *structure, vrna_param_t *parameters, int verbosity_level)`
Calculate the free energy of an already folded circular RNA.
- `int energy_of_structure_pt(const char *string, short *ptable, short *s, short *s1, int verbosity_level)`
Calculate the free energy of an already folded RNA.
- `int energy_of_struct_pt_par(const char *string, short *ptable, short *s, short *s1, vrna_param_t *parameters, int verbosity_level)`
Calculate the free energy of an already folded RNA.
- `float energy_of_move(const char *string, const char *structure, int m1, int m2)`

- *Calculate energy of a move (closing or opening of a base pair)*
• int [energy_of_move_pt](#) (short *pt, short *s, short *s1, int m1, int m2)
- *Calculate energy of a move (closing or opening of a base pair)*
• int [loop_energy](#) (short *ptable, short *s, short *s1, int i)
- *Calculate energy of a loop.*
- float [energy_of_struct](#) (const char *string, const char *structure)
- int [energy_of_struct_pt](#) (const char *string, short *ptable, short *s, short *s1)
- float [energy_of_circ_struct](#) (const char *string, const char *structure)

Basic Energy Evaluation Interface with Dot-Bracket Structure String

- float [vrna_eval_structure](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Calculate the free energy of an already folded RNA.
- float [vrna_eval_covar_structure](#) ([vrna_fold_compound_t](#) *fc, const char *structure)
Calculate the pseudo energy derived by the covariance scores of a set of aligned sequences.
- float [vrna_eval_structure_verbose](#) ([vrna_fold_compound_t](#) *fc, const char *structure, FILE *file)
Calculate the free energy of an already folded RNA and print contributions on a per-loop base.
- float [vrna_eval_structure_v](#) ([vrna_fold_compound_t](#) *fc, const char *structure, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA and print contributions on a per-loop base.
- float [vrna_eval_structure_cstr](#) ([vrna_fold_compound_t](#) *fc, const char *structure, int verbosity_level, [vrna_cstr_t](#) output_stream)

Basic Energy Evaluation Interface with Structure Pair Table

- int [vrna_eval_structure_pt](#) ([vrna_fold_compound_t](#) *fc, const short *pt)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_verbose](#) ([vrna_fold_compound_t](#) *fc, const short *pt, FILE *file)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_v](#) ([vrna_fold_compound_t](#) *fc, const short *pt, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA.

Simplified Energy Evaluation with Sequence and Dot-Bracket Strings

- float [vrna_eval_structure_simple](#) (const char *string, const char *structure)
Calculate the free energy of an already folded RNA.
- float [vrna_eval_circ_structure](#) (const char *string, const char *structure)
Evaluate the free energy of a sequence/structure pair where the sequence is circular.
- float [vrna_eval_gquad_structure](#) (const char *string, const char *structure)
Evaluate the free energy of a sequence/structure pair where the structure may contain G-Quadruplexes.
- float [vrna_eval_circ_gquad_structure](#) (const char *string, const char *structure)
Evaluate the free energy of a sequence/structure pair where the sequence is circular and the structure may contain G-Quadruplexes.
- float [vrna_eval_structure_simple_verbose](#) (const char *string, const char *structure, FILE *file)
Calculate the free energy of an already folded RNA and print contributions per loop.
- float [vrna_eval_structure_simple_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA and print contributions per loop.
- float [vrna_eval_circ_structure_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Evaluate free energy of a sequence/structure pair, assume sequence to be circular and print contributions per loop.
- float [vrna_eval_gquad_structure_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Evaluate free energy of a sequence/structure pair, allow for G-Quadruplexes in the structure and print contributions per loop.
- float [vrna_eval_circ_gquad_structure_v](#) (const char *string, const char *structure, int verbosity_level, FILE *file)
Evaluate free energy of a sequence/structure pair, assume sequence to be circular, allow for G-Quadruplexes in the structure, and print contributions per loop.

Simplified Energy Evaluation with Sequence Alignments and Consensus Structure Dot-Bracket String

- float [vrna_eval_consensus_structure_simple](#) (const char **alignment, const char *structure)
Calculate the free energy of an already folded RNA sequence alignment.
- float [vrna_eval_circ_consensus_structure](#) (const char **alignment, const char *structure)
Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequences are circular.
- float [vrna_eval_gquad_consensus_structure](#) (const char **alignment, const char *structure)
Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the structure may contain G-Quadruplexes.
- float [vrna_eval_circ_gquad_consensus_structure](#) (const char **alignment, const char *structure)
Evaluate the free energy of a multiple sequence alignment/consensus structure pair where the sequence is circular and the structure may contain G-Quadruplexes.
- float [vrna_eval_consensus_structure_simple_verbose](#) (const char **alignment, const char *structure, FILE *file)
Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.
- float [vrna_eval_consensus_structure_simple_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an RNA sequence alignment and print contributions per loop.
- float [vrna_eval_circ_consensus_structure_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences and print contributions per loop.
- float [vrna_eval_gquad_consensus_structure_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an RNA sequence alignment, allow for annotated G- \leftrightarrow Quadruplexes in the structure and print contributions per loop.
- float [vrna_eval_circ_gquad_consensus_structure_v](#) (const char **alignment, const char *structure, int verbosity_level, FILE *file)
Evaluate the free energy of a consensus structure for an alignment of circular RNA sequences, allow for annotated G-Quadruplexes in the structure and print contributions per loop.

Simplified Energy Evaluation with Sequence String and Structure Pair Table

- int [vrna_eval_structure_pt_simple](#) (const char *string, const short *pt)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_simple_verbose](#) (const char *string, const short *pt, FILE *file)
Calculate the free energy of an already folded RNA.
- int [vrna_eval_structure_pt_simple_v](#) (const char *string, const short *pt, int verbosity_level, FILE *file)
Calculate the free energy of an already folded RNA.

Simplified Energy Evaluation with Sequence Alignment and Consensus Structure Pair Table

- int [vrna_eval_consensus_structure_pt_simple](#) (const char **alignment, const short *pt)
Evaluate the Free Energy of a Consensus Secondary Structure given a Sequence Alignment.
- int [vrna_eval_consensus_structure_pt_simple_verbose](#) (const char **alignment, const short *pt, FILE *file)
- int [vrna_eval_consensus_structure_pt_simple_v](#) (const char **alignment, const short *pt, int verbosity_level, FILE *file)

Variables

- int [cut_point](#)
first pos of second seq for cofolding
- int [eos_debug](#)
verbose info from energy_of_struct

18.75.1 Detailed Description

Functions and variables related to energy evaluation of sequence/structure pairs.

18.76 eval.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_EVAL_H
00002 #define VIENNA_RNA_PACKAGE_EVAL_H
00003
00004 #include <stdio.h>
00005 #include <ViennaRNA/datastructures/basic.h>
00006 #include <ViennaRNA/fold_compound.h>
00007 #include <ViennaRNA/datastructures/char_stream.h>
00008 #include <ViennaRNA/landscape/move.h>
00009 #include <ViennaRNA/params/basic.h> /* for deprecated functions */
00010
00011 #ifdef VRNA_WARN_DEPRECATED
00012 # if defined(__clang__)
00013 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00014 # elif defined(__GNUC__)
00015 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00016 # else
00017 #  define DEPRECATED(func, msg) func
00018 # endif
00019 #else
00020 # define DEPRECATED(func, msg) func
00021 #endif
00022
00023
00058 #define VRNA_VERBOSITY_QUIET -1
00059
00060
00064 #define VRNA_VERBOSITY_DEFAULT 1
00065
00066
00093 float
00094 vrna_eval_structure(vrna_fold_compound_t *fc,
00095                    const char *structure);
00096
00097
00118 float
00119 vrna_eval_covar_structure(vrna_fold_compound_t *fc,
00120                          const char *structure);
00121
00122
00136 float
00137 vrna_eval_structure_verbose(vrna_fold_compound_t *fc,
00138                            const char *structure,
00139                            FILE *file);
00140
00141
00167 float
00168 vrna_eval_structure_v(vrna_fold_compound_t *fc,
00169                      const char *structure,
00170                      int verbosity_level,
00171                      FILE *file);
00172
00173
00174 float
00175 vrna_eval_structure_cstr(vrna_fold_compound_t *fc,
00176                         const char *structure,
00177                         int verbosity_level,
00178                         vrna_cstr_t output_stream);
00179
00180
00181 /* End basic eval interface */
00208 int
00209 vrna_eval_structure_pt(vrna_fold_compound_t *fc,
00210                      const short *pt);
00211
00212
00226 int
00227 vrna_eval_structure_pt_verbose(vrna_fold_compound_t *fc,
00228                               const short *pt,
00229                               FILE *file);
00230
00231
00257 int
00258 vrna_eval_structure_pt_v(vrna_fold_compound_t *fc,
00259                         const short *pt,
00260                         int verbosity_level,
00261                         FILE *file);
00262

```

```
00263
00264 /* End basic eval interface with pair table */
00288 float
00289 vrna_eval_structure_simple(const char *string,
00290                           const char *structure);
00291
00292
00303 float
00304 vrna_eval_circ_structure(const char *string,
00305                          const char *structure);
00306
00307
00325 float
00326 vrna_eval_gquad_structure(const char *string,
00327                           const char *structure);
00328
00329
00348 float
00349 vrna_eval_circ_gquad_structure(const char *string,
00350                                const char *structure);
00351
00352
00367 float
00368 vrna_eval_structure_simple_verbose(const char *string,
00369                                    const char *structure,
00370                                    FILE *file);
00371
00372
00396 float
00397 vrna_eval_structure_simple_v(const char *string,
00398                              const char *structure,
00399                              int verbosity_level,
00400                              FILE *file);
00401
00402
00418 float
00419 vrna_eval_circ_structure_v(const char *string,
00420                            const char *structure,
00421                            int verbosity_level,
00422                            FILE *file);
00423
00424
00447 float
00448 vrna_eval_gquad_structure_v(const char *string,
00449                             const char *structure,
00450                             int verbosity_level,
00451                             FILE *file);
00452
00453
00474 float
00475 vrna_eval_circ_gquad_structure_v(const char *string,
00476                                  const char *structure,
00477                                  int verbosity_level,
00478                                  FILE *file);
00479
00480
00481 /* End simplified eval interface */
00511 float
00512 vrna_eval_consensus_structure_simple(const char **alignment,
00513                                     const char *structure);
00514
00515
00531 float
00532 vrna_eval_circ_consensus_structure(const char **alignment,
00533                                    const char *structure);
00534
00535
00558 float
00559 vrna_eval_gquad_consensus_structure(const char **alignment,
00560                                     const char *structure);
00561
00562
00585 float
00586 vrna_eval_circ_gquad_consensus_structure(const char **alignment,
00587                                           const char *structure);
00588
00589
00609 float
00610 vrna_eval_consensus_structure_simple_verbose(const char **alignment,
00611                                               const char *structure,
00612                                               FILE *file);
00613
00614
00639 float
00640 vrna_eval_consensus_structure_simple_v(const char **alignment,
00641                                         const char *structure,
00642                                         int verbosity_level,
```

```

00643             FILE      *file);
00644
00645
00665 float
00666 vrna_eval_circ_consensus_structure_v(const char **alignment,
00667                                     const char *structure,
00668                                     int      verbosity_level,
00669                                     FILE      *file);
00670
00671
00698 float
00699 vrna_eval_gquad_consensus_structure_v(const char **alignment,
00700                                     const char *structure,
00701                                     int      verbosity_level,
00702                                     FILE      *file);
00703
00704
00731 float
00732 vrna_eval_circ_gquad_consensus_structure_v(const char **alignment,
00733                                     const char *structure,
00734                                     int      verbosity_level,
00735                                     FILE      *file);
00736
00737
00738 /* End simplified comparative eval interface */
00761 int
00762 vrna_eval_structure_pt_simple(const char *string,
00763                             const short *pt);
00764
00765
00779 int
00780 vrna_eval_structure_pt_simple_verbose(const char *string,
00781                                     const short *pt,
00782                                     FILE      *file);
00783
00784
00809 int
00810 vrna_eval_structure_pt_simple_v(const char *string,
00811                                const short *pt,
00812                                int      verbosity_level,
00813                                FILE      *file);
00814
00815
00816 /* End simplified eval interface with pair table */
00838 int
00839 vrna_eval_consensus_structure_pt_simple(const char **alignment,
00840                                     const short *pt);
00841
00842
00843 int
00844 vrna_eval_consensus_structure_pt_simple_verbose(const char **alignment,
00845                                     const short *pt,
00846                                     FILE      *file);
00847
00848
00849 int
00850 vrna_eval_consensus_structure_pt_simple_v(const char **alignment,
00851                                     const short *pt,
00852                                     int      verbosity_level,
00853                                     FILE      *file);
00854
00855
00856 /* End simplified eval interface with pair table */
00887 int
00888 vrna_eval_loop_pt(vrna_fold_compound_t *fc,
00889                  int      i,
00890                  const short *pt);
00891
00892
00902 int
00903 vrna_eval_loop_pt_v(vrna_fold_compound_t *fc,
00904                    int      i,
00905                    const short *pt,
00906                    int      verbosity_level);
00907
00908
00943 float
00944 vrna_eval_move(vrna_fold_compound_t *fc,
00945               const char *structure,
00946               int      m1,
00947               int      m2);
00948
00949
00964 int
00965 vrna_eval_move_pt(vrna_fold_compound_t *fc,
00966                  short *pt,
00967                  int      m1,

```

```
00968             int                m2);
00969
00970
00971 int
00972 vrna_eval_move_pt_simple(const char *string,
00973                         short      *pt,
00974                         int        m1,
00975                         int        m2);
00976
00977
00978 int
00979 vrna_eval_move_shift_pt(vrna_fold_compound_t *fc,
00980                       vrna_move_t          *m,
00981                       short                 *structure);
00982
00983
00984 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00985
01003 extern int cut_point;
01004
01008 extern int eos_debug;
01009
01028 DEPRECATED(float energy_of_structure(const char *string,
01029                                     const char *structure,
01030                                     int        verbosity_level),
01031            "Use vrna_eval_structure_simple() and vrna_eval_structure() instead");
01032
01048 DEPRECATED(float energy_of_struct_par(const char *string,
01049                                       const char *structure,
01050                                       vrna_param_t *parameters,
01051                                       int        verbosity_level),
01052            "Use vrna_eval_structure() instead");
01053
01072 DEPRECATED(float energy_of_circ_structure(const char *string,
01073                                           const char *structure,
01074                                           int        verbosity_level),
01075            "Use vrna_eval_circ_structure_simple() and vrna_eval_structure() instead");
01076
01092 DEPRECATED(float energy_of_circ_struct_par(const char *string,
01093                                             const char *structure,
01094                                             vrna_param_t *parameters,
01095                                             int        verbosity_level),
01096            "Use vrna_eval_structure() instead");
01097
01098
01099 DEPRECATED(float energy_of_gquad_structure(const char *string,
01100                                           const char *structure,
01101                                           int        verbosity_level),
01102            "Use vrna_eval_structure_simple() instead");
01103
01104 DEPRECATED(float energy_of_gquad_struct_par(const char *string,
01105                                             const char *structure,
01106                                             vrna_param_t *parameters,
01107                                             int        verbosity_level),
01108            "Use vrna_eval_structure() instead");
01109
01110
01131 DEPRECATED(int energy_of_structure_pt(const char *string,
01132                                       short      *ptable,
01133                                       short      *s,
01134                                       short      *s1,
01135                                       int        verbosity_level),
01136            "Use vrna_eval_structure_pt_simple() and vrna_eval_structure_pt() instead");
01137
01155 DEPRECATED(int energy_of_struct_pt_par(const char *string,
01156                                         short      *ptable,
01157                                         short      *s,
01158                                         short      *s1,
01159                                         vrna_param_t *parameters,
01160                                         int        verbosity_level),
01161            "Use vrna_eval_structure_pt() instead");
01162
01163
01180 DEPRECATED(float energy_of_move(const char *string,
01181                                  const char *structure,
01182                                  int        m1,
01183                                  int        m2),
01184            "Use vrna_eval_move() instead");
01185
01186
01205 DEPRECATED(int energy_of_move_pt(short *pt,
01206                                   short *s,
01207                                   short *s1,
01208                                   int    m1,
01209                                   int    m2),
01210            "Use vrna_eval_move_pt_simple() and vrna_eval_move_pt() instead");
01211
```

```

01225 DEPRECATED(int    loop_energy(short  *ptable,
01226                                short  *s,
01227                                short  *sl,
01228                                int    i),
01229                "Use vrna_eval_loop_pt() instead");
01230
01245 DEPRECATED(float energy_of_struct(const char *string,
01246                                const char *structure),
01247                "Use vrna_eval_structure_simple() instead");
01248
01265 DEPRECATED(int energy_of_struct_pt(const char *string,
01266                                short  *ptable,
01267                                short  *s,
01268                                short  *sl),
01269                "Use vrna_eval_structure_pt_simple() instead");
01270
01285 DEPRECATED(float energy_of_circ_struct(const char *string,
01286                                const char *structure),
01287                "Use vrna_eval_circ_structure_simple() and vrna_eval_structure() instead");
01288
01289 #endif
01290
01295 #endif

```

18.77 ViennaRNA/exterior_loops.h File Reference

Use [ViennaRNA/loops/external.h](#) instead.

Include dependency graph for exterior_loops.h:

18.77.1 Detailed Description

Use [ViennaRNA/loops/external.h](#) instead.

Deprecated Use [ViennaRNA/loops/external.h](#) instead

18.78 exterior_loops.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #  ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/exterior_loops.h>! Use
00013 #  <ViennaRNA/loops/external.h> instead!"
00013 #  endif
00014 #include <ViennaRNA/loops/external.h>
00015 #endif
00016
00017 #endif

```

18.79 ViennaRNA/file_formats.h File Reference

Use [ViennaRNA/io/file_formats.h](#) instead.

Include dependency graph for file_formats.h:

18.79.1 Detailed Description

Use [ViennaRNA/io/file_formats.h](#) instead.

Deprecated Use [ViennaRNA/io/file_formats.h](#) instead

18.80 file_formats.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_DEPRECATED_H
00003

```

```

00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #  ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/file_formats.h>! Use
    <ViennaRNA/io/file_formats.h> instead!"
00013 #  endif
00014 #include <ViennaRNA/io/file_formats.h>
00015 #include <ViennaRNA/io/file_formats_msa.h>
00016 #endif
00017
00018 #endif

```

18.81 ViennaRNA/io/file_formats.h File Reference

Read and write different file formats for RNA sequences, structures.

Include dependency graph for file_formats.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_OPTION_MULTILINE 32U`
Tell a function that an input is assumed to span several lines.
- `#define VRNA_CONSTRAINT_MULTILINE 32U`
parse multiline constraint

Functions

- void `vrna_file_helixlist` (const char *seq, const char *db, float energy, FILE *file)
Print a secondary structure as helix list.
- void `vrna_file_connect` (const char *seq, const char *db, float energy, const char *identifier, FILE *file)
Print a secondary structure as connect table.
- void `vrna_file_bpseq` (const char *seq, const char *db, FILE *file)
Print a secondary structure in bpseq format.
- void `vrna_file_json` (const char *seq, const char *db, double energy, const char *identifier, FILE *file)
Print a secondary structure in jsonformat.
- unsigned int `vrna_file_fasta_read_record` (char **header, char **sequence, char ***rest, FILE *file, unsigned int options)
- char * `vrna_extract_record_rest_structure` (const char **lines, unsigned int length, unsigned int option)
Extract a dot-bracket structure string from (multiline)character array.
- int `vrna_file_SHAPE_read` (const char *file_name, int length, double default_value, char *sequence, double *values)
Read data from a given SHAPE reactivity input file.
- void `vrna_extract_record_rest_constraint` (char **cstruc, const char **lines, unsigned int option)
Extract a hard constraint encoded as pseudo dot-bracket string.
- unsigned int `read_record` (char **header, char **sequence, char ***rest, unsigned int options)
Get a data record from stdin.

18.81.1 Detailed Description

Read and write different file formats for RNA sequences, structures.

18.82 file_formats.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_H
00002 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 #  if defined(__clang__)
00006 #    define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))

```



```

00007 # elif defined(__GNUC__)
00008 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00009 # else
00010 #   define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00028 #include <stdio.h>
00029
00030 #include <ViennaRNA/datastructures/basic.h>
00031
00040 void
00041 vrna_file_helixlist(const char *seq,
00042                    const char *db,
00043                    float energy,
00044                    FILE *file);
00045
00046
00070 void
00071 vrna_file_connect(const char *seq,
00072                  const char *db,
00073                  float energy,
00074                  const char *identifier,
00075                  FILE *file);
00076
00077
00085 void
00086 vrna_file_bpseq(const char *seq,
00087                 const char *db,
00088                 FILE *file);
00089
00090
00091 #if VRNA_WITH_JSON_SUPPORT
00092
00102 void
00103 vrna_file_json(const char *seq,
00104                const char *db,
00105                double energy,
00106                const char *identifier,
00107                FILE *file);
00108
00109
00110 #endif
00111
00121 #define VRNA_OPTION_MULTILINE 32U
00126 #define VRNA_CONSTRAINT_MULTILINE 32U
00127
00193 unsigned int
00194 vrna_file_fasta_read_record(char **header,
00195                             char **sequence,
00196                             char **rest,
00197                             FILE *file,
00198                             unsigned int options);
00199
00200
00217 char *
00218 vrna_extract_record_rest_structure(const char **lines,
00219                                   unsigned int length,
00220                                   unsigned int option);
00221
00222
00235 int
00236 vrna_file_SHAPE_read(const char *file_name,
00237                      int length,
00238                      double default_value,
00239                      char *sequence,
00240                      double *values);
00241
00242 #define VRNA_INPUT_VERBOSE 16384U
00243
00244
00245 int
00246 vrna_file_connect_read_record(FILE *fp,
00247                               char **id,
00248                               char **sequence,
00249                               char **structure,
00250                               char **remainder,
00251                               unsigned int options);
00252
00253 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00254
00271 DEPRECATED(void vrna_extract_record_rest_constraint(char **cstruc,
00272                                                     const char **lines,
00273                                                     unsigned int option),
00274            "This function is obsolete");

```

```

00275
00280 DEPRECATED(char *extract_record_rest_structure(const char **lines,
00281                                                unsigned int length,
00282                                                unsigned int option),
00283             "Use vrna_extract_record_rest_structure() instead");
00284
00291 DEPRECATED(unsigned int read_record(char **header,
00292                                     char **sequence,
00293                                     char ***rest,
00294                                     unsigned int options),
00295             "Use vrna_file_fasta_read_record() instead");
00296
00297
00298 DEPRECATED(unsigned int get_multi_input_line(char **string,
00299                                               unsigned int options),
00300             "This function is obsolete");
00301
00302 #endif
00303
00308 #endif

```

18.83 ViennaRNA/file_formats_msa.h File Reference

Use [ViennaRNA/io/file_formats_msa.h](#) instead.

Include dependency graph for file_formats_msa.h:

18.83.1 Detailed Description

Use [ViennaRNA/io/file_formats_msa.h](#) instead.

Deprecated Use [ViennaRNA/io/file_formats_msa.h](#) instead

18.84 file_formats_msa.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/file_formats_msa.h>! Use
00013 <ViennaRNA/io/file_formats_msa.h> instead!"
00013 # endif
00014 #include <ViennaRNA/io/file_formats_msa.h>
00015 #endif
00016
00017 #endif

```

18.85 ViennaRNA/io/file_formats_msa.h File Reference

Functions dealing with file formats for Multiple Sequence Alignments (MSA)

Include dependency graph for file_formats_msa.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_FILE_FORMAT_MSA_CLUSTAL 1U`
Option flag indicating ClustalW formatted files.
- `#define VRNA_FILE_FORMAT_MSA_STOCKHOLM 2U`
Option flag indicating Stockholm 1.0 formatted files.
- `#define VRNA_FILE_FORMAT_MSA_FASTA 4U`
Option flag indicating FASTA (Pearson) formatted files.
- `#define VRNA_FILE_FORMAT_MSA_MAF 8U`
Option flag indicating MAF formatted files.
- `#define VRNA_FILE_FORMAT_MSA_MIS 16U`

- *Option flag indicating most informative sequence (MIS) output.*
- `#define VRNA_FILE_FORMAT_MSA_DEFAULT`
Option flag indicating the set of default file formats.
- `#define VRNA_FILE_FORMAT_MSA_NOCHECK 4096U`
Option flag to disable validation of the alignment.
- `#define VRNA_FILE_FORMAT_MSA_UNKNOWN 8192U`
Return flag of `vrna_file_msa_detect_format()` to indicate unknown or malformed alignment.
- `#define VRNA_FILE_FORMAT_MSA_APPEND 16384U`
Option flag indicating to append data to a multiple sequence alignment file rather than overwriting it.
- `#define VRNA_FILE_FORMAT_MSA_QUIET 32768U`
Option flag to suppress unnecessary spam messages on `stderr`
- `#define VRNA_FILE_FORMAT_MSA_SILENT 65536U`
Option flag to completely silence any warnings on `stderr`

Functions

- `int vrna_file_msa_read` (const char *filename, char ***names, char ***aln, char **id, char **structure, unsigned int options)
Read a multiple sequence alignment from file.
- `int vrna_file_msa_read_record` (FILE *fp, char ***names, char ***aln, char **id, char **structure, unsigned int options)
Read a multiple sequence alignment from file handle.
- `unsigned int vrna_file_msa_detect_format` (const char *filename, unsigned int options)
Detect the format of a multiple sequence alignment file.
- `int vrna_file_msa_write` (const char *filename, const char **names, const char **aln, const char *id, const char *structure, const char *source, unsigned int options)
Write multiple sequence alignment file.

18.85.1 Detailed Description

Functions dealing with file formats for Multiple Sequence Alignments (MSA)

, ,

18.86 file_formats_msa.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_H
00002 #define VIENNA_RNA_PACKAGE_FILE_FORMATS_MSA_H
00003
00010 #include <stdio.h>
00011
00022 #define VRNA_FILE_FORMAT_MSA_CLUSTAL      1U
00023
00028 #define VRNA_FILE_FORMAT_MSA_STOCKHOLM   2U
00029
00034 #define VRNA_FILE_FORMAT_MSA_FASTA       4U
00035
00040 #define VRNA_FILE_FORMAT_MSA_MAF         8U
00041
00050 #define VRNA_FILE_FORMAT_MSA_MIS         16U
00051
00056 #define VRNA_FILE_FORMAT_MSA_DEFAULT     ( \
00057     VRNA_FILE_FORMAT_MSA_CLUSTAL \
00058     | VRNA_FILE_FORMAT_MSA_STOCKHOLM \
00059     | VRNA_FILE_FORMAT_MSA_FASTA \
00060     | VRNA_FILE_FORMAT_MSA_MAF \
00061 )
00062
00067 #define VRNA_FILE_FORMAT_MSA_NOCHECK     4096U
00068
00073 #define VRNA_FILE_FORMAT_MSA_UNKNOWN     8192U
00074
00079 #define VRNA_FILE_FORMAT_MSA_APPEND     16384U
00080
```

```

00085 #define VRNA_FILE_FORMAT_MSA_QUIET          32768U
00086
00091 #define VRNA_FILE_FORMAT_MSA_SILENT          65536U
00092
00145 int
00146 vrna_file_msa_read(const char    *filename,
00147                   char          ***names,
00148                   char          ***aln,
00149                   char          **id,
00150                   char          **structure,
00151                   unsigned int  options);
00152
00153
00210 int
00211 vrna_file_msa_read_record(FILE      *fp,
00212                          char      ***names,
00213                          char      ***aln,
00214                          char      **id,
00215                          char      **structure,
00216                          unsigned int options);
00217
00218
00244 unsigned int
00245 vrna_file_msa_detect_format(const char *filename,
00246                          unsigned int options);
00247
00248
00266 int
00267 vrna_file_msa_write(const char    *filename,
00268                   const char    **names,
00269                   const char    **aln,
00270                   const char    *id,
00271                   const char    *structure,
00272                   const char    *source,
00273                   unsigned int  options);
00274
00275
00280 #endif

```

18.87 ViennaRNA/file_utils.h File Reference

Use [ViennaRNA/io/utils.h](#) instead.

Include dependency graph for file_utils.h:

18.87.1 Detailed Description

Use [ViennaRNA/io/utils.h](#) instead.

Deprecated Use [ViennaRNA/io/utils.h](#) instead

18.88 file_utils.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_FILE_UTILS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_FILE_UTILS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/file_utils.h>! Use <ViennaRNA/io/utils.h>
00013 #   endif
00014 #include <ViennaRNA/io/utils.h>
00015 #endif
00016
00017 #endif

```

18.89 ViennaRNA/findpath.h File Reference

Use [ViennaRNA/landscape/findpath.h](#) instead.

Include dependency graph for findpath.h:

18.89.1 Detailed Description

Use [ViennaRNA/landscape/findpath.h](#) instead.

Deprecated Use [ViennaRNA/landscape/findpath.h](#) instead

18.90 findpath.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_FINDPATH_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_FINDPATH_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/findpath.h>! Use
    <ViennaRNA/landscape/findpath.h> instead!"
00013 # endif
00014 #include <ViennaRNA/landscape/findpath.h>
00015 #endif
00016
00017 #endif
```

18.91 ViennaRNA/landscape/findpath.h File Reference

A breadth-first search heuristic for optimal direct folding paths.

Include dependency graph for findpath.h: This graph shows which files directly or indirectly include this file:

Functions

- [int vrna_path_findpath_saddle](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width)
Find energy of a saddle point between 2 structures (search only direct path)
- [int vrna_path_findpath_saddle_ub](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width, int maxE)
Find energy of a saddle point between 2 structures (search only direct path)
- [vrna_path_t](#) * [vrna_path_findpath](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width)
Find refolding path between 2 structures (search only direct path)
- [vrna_path_t](#) * [vrna_path_findpath_ub](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int width, int maxE)
Find refolding path between 2 structures (search only direct path)
- [int find_saddle](#) (const char *seq, const char *s1, const char *s2, int width)
Find energy of a saddle point between 2 structures (search only direct path)
- [void free_path](#) ([vrna_path_t](#) *path)
Free memory allocated by [get_path\(\)](#) function.
- [vrna_path_t](#) * [get_path](#) (const char *seq, const char *s1, const char *s2, int width)
Find refolding path between 2 structures (search only direct path)

18.91.1 Detailed Description

A breadth-first search heuristic for optimal direct folding paths.

18.92 findpath.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_FIND_PATH_H
00002 #define VIENNA_RNA_PACKAGE_FIND_PATH_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
```

```

00010 # define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00030 #include <ViennaRNA/fold_compound.h>
00031 #include <ViennaRNA/landscape/paths.h>
00032
00054 int
00055 vrna_path_findpath_saddle(vrna_fold_compound_t *fc,
00056                          const char *s1,
00057                          const char *s2,
00058                          int width);
00059
00060
00087 int
00088 vrna_path_findpath_saddle_ub(vrna_fold_compound_t *fc,
00089                             const char *s1,
00090                             const char *s2,
00091                             int width,
00092                             int maxE);
00093
00094
00116 vrna_path_t *
00117 vrna_path_findpath(vrna_fold_compound_t *fc,
00118                  const char *s1,
00119                  const char *s2,
00120                  int width);
00121
00122
00150 vrna_path_t *
00151 vrna_path_findpath_ub(vrna_fold_compound_t *fc,
00152                     const char *s1,
00153                     const char *s2,
00154                     int width,
00155                     int maxE);
00156
00157
00158 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00159
00176 DEPRECATED(int
00177             find_saddle(const char *seq,
00178                        const char *s1,
00179                        const char *s2,
00180                        int width),
00181             "Use vrna_path_findpath_saddle() instead!");
00182
00183
00193 DEPRECATED(void
00194             free_path(vrna_path_t *path),
00195             "Use vrna_path_free() instead!");
00196
00197
00214 DEPRECATED(vrna_path_t *
00215             get_path(const char *seq,
00216                    const char *s1,
00217                    const char *s2,
00218                    int width),
00219             "Use vrna_path_findpath() instead!");
00220
00221
00222 #endif
00223
00228 #endif

```

18.93 ViennaRNA/fold.h File Reference

MFE calculations for single RNA sequences.

Include dependency graph for fold.h:

Functions

- float **fold_par** (const char *sequence, char *structure, [vrna_param_t](#) *parameters, int is_constrained, int is_circular)
Compute minimum free energy and an appropriate secondary structure of an RNA sequence.
- float **fold** (const char *sequence, char *structure)
Compute minimum free energy and an appropriate secondary structure of an RNA sequence.

- float [circfold](#) (const char *sequence, char *structure)
Compute minimum free energy and an appropriate secondary structure of a circular RNA sequence.
- void [free_arrays](#) (void)
Free arrays for mfe folding.
- void [update_fold_params](#) (void)
Recalculate energy parameters.
- void [update_fold_params_par](#) (vrna_param_t *parameters)
Recalculate energy parameters.
- void [export_fold_arrays](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p)
- void [export_fold_arrays_par](#) (int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p, vrna_param_t **P_p)
- void [export_circfold_arrays](#) (int *Fc_p, int *FcH_p, int *Fcl_p, int *FcM_p, int **fM2_p, int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p)
- void [export_circfold_arrays_par](#) (int *Fc_p, int *FcH_p, int *Fcl_p, int *FcM_p, int **fM2_p, int **f5_p, int **c_p, int **fML_p, int **fM1_p, int **indx_p, char **ptype_p, vrna_param_t **P_p)
- int [LoopEnergy](#) (int n1, int n2, int type, int type_2, int si1, int sj1, int sp1, int sq1)
- int [HairpinE](#) (int size, int type, int si1, int sj1, const char *string)
- void [initialize_fold](#) (int length)

18.93.1 Detailed Description

MFE calculations for single RNA sequences.

18.94 fold.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_FOLD_H
00002 #define VIENNA_RNA_PACKAGE_FOLD_H
00003
00004 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00005
00006 #include <ViennaRNA/datastructures/basic.h>
00007 #include <ViennaRNA/params/basic.h>
00008 #include <ViennaRNA/mfe.h>
00009 #include <ViennaRNA/eval.h>
00010
00011 #ifdef VRNA_WARN_DEPRECATED
00012 # if defined(__clang__)
00013 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00014 # elif defined(__GNUC__)
00015 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00016 # else
00017 #  define DEPRECATED(func, msg) func
00018 # endif
00019 #else
00020 # define DEPRECATED(func, msg) func
00021 #endif
00022
00077 DEPRECATED(float
00078 fold_par( const char *sequence,
00079          char *structure,
00080          vrna_param_t *parameters,
00081          int is_constrained,
00082          int is_circular),
00083 "Use the new API and vrna_mfe() instead");
00084
00101 DEPRECATED(float fold( const char *sequence, char *structure),
00102 "Use vrna_fold() or vrna_mfe() instead");
00103
00120 DEPRECATED(float circfold( const char *sequence, char *structure),
00121 "Use vrna_circfold() or vrna_mfe() instead");
00122
00123
00130 DEPRECATED(void free_arrays(void),
00131 "This function is obsolete");
00132
00133
00134
00141 DEPRECATED(void update_fold_params(void),
00142 "This function is obsolete");
00143
```

```

00150 DEPRECATED(void update_fold_params_par(vrna_param_t *parameters),
00151 "Use the new API with vrna_fold_compound_t datastructure instead");
00152
00158 DEPRECATED(void
00159 export_fold_arrays( int **f5_p,
00160                     int **c_p,
00161                     int **fML_p,
00162                     int **fM1_p,
00163                     int **indx_p,
00164                     char **ptype_p),
00165 "Use the new API with vrna_fold_compound_t datastructure instead");
00166
00172 DEPRECATED(void
00173 export_fold_arrays_par( int **f5_p,
00174                        int **c_p,
00175                        int **fML_p,
00176                        int **fM1_p,
00177                        int **indx_p,
00178                        char **ptype_p,
00179                        vrna_param_t **P_p),
00180 "Use the new API with vrna_fold_compound_t datastructure instead");
00181
00187 DEPRECATED(void
00188 export_circfold_arrays( int *Fc_p,
00189                        int *FcH_p,
00190                        int *FcI_p,
00191                        int *FcM_p,
00192                        int **fM2_p,
00193                        int **f5_p,
00194                        int **c_p,
00195                        int **fML_p,
00196                        int **fM1_p,
00197                        int **indx_p,
00198                        char **ptype_p),
00199 "Use the new API with vrna_fold_compound_t datastructure instead");
00200
00206 DEPRECATED(void
00207 export_circfold_arrays_par( int *Fc_p,
00208                            int *FcH_p,
00209                            int *FcI_p,
00210                            int *FcM_p,
00211                            int **fM2_p,
00212                            int **f5_p,
00213                            int **c_p,
00214                            int **fML_p,
00215                            int **fM1_p,
00216                            int **indx_p,
00217                            char **ptype_p,
00218                            vrna_param_t **P_p),
00219 "Use the new API with vrna_fold_compound_t datastructure instead");
00220
00221
00222
00223 /* finally moved the loop energy function declarations to this header... */
00224 /* BUT: The functions only exist for backward compatibility reasons! */
00225 /* You better include "loop_energies.h" and call the functions: */
00226 /* E_Hairpin() and E_IntLoop() which are (almost) threadsafe as they get */
00227 /* a pointer to the energy parameter data structure as additional argument */
00228
00233 DEPRECATED(int LoopEnergy(int n1,
00234                           int n2,
00235                           int type,
00236                           int type_2,
00237                           int sil,
00238                           int sj1,
00239                           int spl,
00240                           int sql),
00241 "This function is obsolete");
00242
00247 DEPRECATED(int HairpinE(int size,
00248                          int type,
00249                          int sil,
00250                          int sj1,
00251                          const char *string),
00252 "Use E_Hairpin() instead");
00253
00259 DEPRECATED(void initialize_fold(int length),
00260 "This function is obsolete");
00261
00265 DEPRECATED(char *backtrack_fold_from_pair(char *sequence,
00266                                             int i,
00267                                             int j),
00268 "This function is obsolete. Consider using vrna_backtrack_from_intervals() instead");
00269
00270
00271 #endif
00272

```



```
00277 #endif
```

18.95 ViennaRNA/fold_compound.h File Reference

The Basic Fold Compound API.

Include dependency graph for fold_compound.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_fc_s](#)

The most basic data structure required by many functions throughout the RNAlib. [More...](#)

Macros

- #define [VRNA_STATUS_MFE_PRE](#) (unsigned char)1
Status message indicating that MFE computations are about to begin.
- #define [VRNA_STATUS_MFE_POST](#) (unsigned char)2
Status message indicating that MFE computations are finished.
- #define [VRNA_STATUS_PF_PRE](#) (unsigned char)3
Status message indicating that Partition function computations are about to begin.
- #define [VRNA_STATUS_PF_POST](#) (unsigned char)4
Status message indicating that Partition function computations are finished.
- #define [VRNA_OPTION_DEFAULT](#) 0U
Option flag to specify default settings/requirements.
- #define [VRNA_OPTION_MFE](#) 1U
Option flag to specify requirement of Minimum Free Energy (MFE) DP matrices and corresponding set of energy parameters.
- #define [VRNA_OPTION_PF](#) 2U
Option flag to specify requirement of Partition Function (PF) DP matrices and corresponding set of Boltzmann factors.
- #define [VRNA_OPTION_HYBRID](#) 4U
Option flag to specify requirement of dimer DP matrices.
- #define [VRNA_OPTION_EVAL_ONLY](#) 8U
Option flag to specify that neither MFE, nor PF DP matrices are required.
- #define [VRNA_OPTION_WINDOW](#) 16U
Option flag to specify requirement of DP matrices for local folding approaches.

Typedefs

- typedef struct [vrna_fc_s](#) [vrna_fold_compound_t](#)
Typename for the fold_compound data structure [vrna_fc_s](#).
- typedef void(* [vrna_auxdata_free_f](#)) (void *data)
Callback to free memory allocated for auxiliary user-provided data.
- typedef void(* [vrna_recursion_status_f](#)) (unsigned char status, void *data)
Callback to perform specific user-defined actions before, or after recursive computations.

Enumerations

- enum [vrna_fc_type_e](#) { [VRNA_FC_TYPE_SINGLE](#) , [VRNA_FC_TYPE_COMPARATIVE](#) }
An enumerator that is used to specify the type of a [vrna_fold_compound_t](#).

Functions

- `vrna_fold_compound_t * vrna_fold_compound` (const char *sequence, const `vrna_md_t` *md_p, unsigned int options)

Retrieve a `vrna_fold_compound_t` data structure for single sequences and hybridizing sequences.

- `vrna_fold_compound_t * vrna_fold_compound_comparative` (const char **sequences, `vrna_md_t` *md_p, unsigned int options)

Retrieve a `vrna_fold_compound_t` data structure for sequence alignments.

- void `vrna_fold_compound_free` (`vrna_fold_compound_t` *fc)

Free memory occupied by a `vrna_fold_compound_t`.

- void `vrna_fold_compound_add_auxdata` (`vrna_fold_compound_t` *fc, void *data, `vrna_auxdata_free_f` f)

Add auxiliary data to the `vrna_fold_compound_t`.

- void `vrna_fold_compound_add_callback` (`vrna_fold_compound_t` *fc, `vrna_recursion_status_f` f)

Add a recursion status callback to the `vrna_fold_compound_t`.

18.95.1 Detailed Description

The Basic Fold Compound API.

18.96 fold_compound.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_FOLD_COMPOUND_H
00002 #define VIENNA_RNA_PACKAGE_FOLD_COMPOUND_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00038 typedef struct vrna_fc_s vrna_fold_compound_t;
00039
00058 typedef void (*vrna_auxdata_free_f)(void *data);
00059
00060 DEPRECATED(typedef void (vrna_callback_free_auxdata)(void *data),
00061           "Use vrna_auxdata_free_f instead!");
00062
00081 typedef void (*vrna_recursion_status_f)(unsigned char status,
00082                                         void *data);
00083
00084 DEPRECATED(typedef void (vrna_callback_recursion_status)(unsigned char status,
00085                                                         void *data),
00086           "Use vrna_recursion_status_f instead!");
00087
00088
00095 #define VRNA_STATUS_MFE_PRE      (unsigned char)1
00096
00103 #define VRNA_STATUS_MFE_POST    (unsigned char)2
00104
00110 #define VRNA_STATUS_PF_PRE      (unsigned char)3
00111
00117 #define VRNA_STATUS_PF_POST     (unsigned char)4
00118
00119
00120 #include <ViennaRNA/model.h>
00121 #include <ViennaRNA/params/basic.h>
00122 #include <ViennaRNA/sequence.h>
00123 #include <ViennaRNA/dp_matrices.h>
00124 #include <ViennaRNA/constraints/hard.h>
00125 #include <ViennaRNA/constraints/soft.h>
00126 #include <ViennaRNA/grammar.h>
00127 #include <ViennaRNA/structured_domains.h>
00128 #include <ViennaRNA/unstructured_domains.h>
00129
00130 #ifdef VRNA_WITH_SVM
00131 #include <ViennaRNA/zscore.h>
00132 #endif
```

```
00133
00134
00138 typedef enum {
00139     VRNA_FC_TYPE_SINGLE,
00140     VRNA_FC_TYPE_COMPARATIVE
00141 } vrna_fc_type_e;
00142
00143
00156 struct vrna_fc_s {
00161     const vrna_fc_type_e type;
00168     unsigned int length;
00169 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00170     DEPRECATED(int cutpoint,
00171                "Use strand_* members instead");
00175 #endif
00176     unsigned int *strand_number;
00177     unsigned int *strand_order;
00178     unsigned int *strand_order_uniq;
00179     unsigned int *strand_start;
00180     unsigned int *strand_end;
00182     unsigned int strands;
00183     vrna_seq_t *nucleotides;
00184     vrna_msa_t *alignment;
00186     vrna_hc_t *hc;
00188     vrna_mx_mfe_t *matrices;
00189     vrna_mx_pf_t *exp_matrices;
00191     vrna_param_t *params;
00192     vrna_exp_param_t *exp_params;
00194     int *iindx;
00195     int *jindx;
00203     vrna_recursion_status_f stat_cb;
00208     void *auxdata;
00212     vrna_auxdata_free_f free_auxdata;
00223     /* data structure to adjust additional structural domains, such as G-quadruplexes */
00224     vrna_sd_t *domains_struct;
00226     /* data structure to adjust additional contributions to unpaired stretches, e.g. due to protein
binding */
00227     vrna_ud_t *domains_up;
00229     /* auxiliary (user-defined) extension to the folding grammar */
00230     vrna_gr_aux_t *aux_grammar;
00236 #ifndef VRNA_DISABLE_C11_FEATURES
00237     /* C11 support for unnamed unions/structs */
00238     union {
00239         struct {
00240 #endif
00241
00246         char *sequence;
00249         short *sequence_encoding;
00253         short *encoding5;
00254         short *encoding3;
00255
00256         short *sequence_encoding2;
00257
00258
00259         char *ptype;
00268         char *ptype_pf_compat;
00273         vrna_sc_t *sc;
00281 #ifndef VRNA_DISABLE_C11_FEATURES
00282     /* C11 support for unnamed unions/structs */
00283     };
00284     struct {
00285 #endif
00286
00291         char **sequences;
00295         unsigned int n_seq;
00298         char *cons_seq;
00301         short *S_cons;
00304         short **S;
00307         short **S5;
00310         short **S3;
00313         char **Ss;
00314         unsigned int **a2s;
00315         int *pscore;
00318         int *pscore_local;
00321         short *pscore_pf_compat;
00325         vrna_sc_t **scs;
00328         int oldAliEn;
00329
00333 #ifndef VRNA_DISABLE_C11_FEATURES
00334     };
00335     };
00336 #endif
00337
00344     unsigned int maxD1;
00345     unsigned int maxD2;
00346     short *reference_pt1;
00347     short *reference_pt2;
```

```

00349 unsigned int *referenceBPs1;
00350 unsigned int *referenceBPs2;
00351 unsigned int *bpdist;
00353 unsigned int *mm1;
00354 unsigned int *mm2;
00366 int window_size;
00367 char **ptype_local;
00368 #ifdef VRNA_WITH_SVM
00369 vrna_zsc_dat_t zscore_data;
00370 #endif
00371
00375 };
00376
00377
00378 /* the definitions below should be used for functions that return/receive/destroy fold compound data
    structures */
00379
00383 #define VRNA_OPTION_DEFAULT 0U
00384
00391 #define VRNA_OPTION_MFE 1U
00392
00399 #define VRNA_OPTION_PF 2U
00400
00404 #define VRNA_OPTION_HYBRID 4U
00405
00415 #define VRNA_OPTION_EVAL_ONLY 8U
00416
00420 #define VRNA_OPTION_WINDOW 16U
00421
00459 vrna_fold_compound_t *
00460 vrna_fold_compound(const char *sequence,
00461                    const vrna_md_t *md_p,
00462                    unsigned int options);
00463
00464
00502 vrna_fold_compound_t *
00503 vrna_fold_compound_comparative(const char **sequences,
00504                                vrna_md_t *md_p,
00505                                unsigned int options);
00506
00507
00508 vrna_fold_compound_t *
00509 vrna_fold_compound_comparative2(const char **sequences,
00510                                 const char **names,
00511                                 const unsigned char *orientation,
00512                                 const unsigned long long *start,
00513                                 const unsigned long long *genome_size,
00514                                 vrna_md_t *md_p,
00515                                 unsigned int options);
00516
00517
00518 vrna_fold_compound_t *
00519 vrna_fold_compound_TwoD(const char *sequence,
00520                        const char *s1,
00521                        const char *s2,
00522                        vrna_md_t *md_p,
00523                        unsigned int options);
00524
00525
00526 int
00527 vrna_fold_compound_prepare(vrna_fold_compound_t *fc,
00528                           unsigned int options);
00529
00530
00538 void
00539 vrna_fold_compound_free(vrna_fold_compound_t *fc);
00540
00541
00559 void
00560 vrna_fold_compound_add_auxdata(vrna_fold_compound_t *fc,
00561                               void *data,
00562                               vrna_auxdata_free_f f);
00563
00564
00580 void
00581 vrna_fold_compound_add_callback(vrna_fold_compound_t *fc,
00582                                vrna_recursion_status_f f);
00583
00584
00589 #endif

```

18.97 ViennaRNA/fold_vars.h File Reference

Here all all declarations of the global variables used throughout RNAlib.

Include dependency graph for fold_vars.h: This graph shows which files directly or indirectly include this file:

Variables

- int **fold_constrained**
Global switch to activate/deactivate folding with structure constraints.
- int **csv**
generate comma seperated output
- char * [RibosumFile](#)
- int [james_rule](#)
- int [logML](#)
- int [cut_point](#)
Marks the position (starting from 1) of the first nucleotide of the second molecule within the concatenated sequence.
- [bondT](#) * [base_pair](#)
Contains a list of base pairs after a call to [fold\(\)](#).
- [FLT_OR_DBL](#) * [pr](#)
A pointer to the base pair probability matrix.
- int * [iindx](#)
index array to move through pr.

18.97.1 Detailed Description

Here all all declarations of the global variables used throughout RNAlib.

18.97.2 Variable Documentation

18.97.2.1 RibosumFile

```
char* RibosumFile [extern]
```

warning this variable will vanish in the future ribosums will be compiled in instead

18.97.2.2 james_rule

```
int james_rule [extern]
```

interior loops of size 2 get energy 0.8Kcal and no mismatches, default 1

18.97.2.3 logML

```
int logML [extern]
```

use logarithmic multiloop energy function

18.97.2.4 cut_point

```
int cut_point [extern]
```

Marks the position (starting from 1) of the first nucleotide of the second molecule within the concatenated sequence. To evaluate the energy of a duplex structure (a structure formed by two strands), concatenate the to sequences and set it to the first base of the second strand in the concatenated sequence. The default value of -1 stands for single molecule folding. The cut_point variable is also used by [vrna_file_PS_rnaplot\(\)](#) and [PS_dot_plot\(\)](#) to mark the chain break in postscript plots.

18.97.2.5 base_pair

`bondT* base_pair [extern]`

Contains a list of base pairs after a call to `fold()`.

`base_pair[0].i` contains the total number of pairs.

Deprecated Do not use this variable anymore!

18.97.2.6 pr

`FLT_OR_DBL* pr [extern]`

A pointer to the base pair probability matrix.

Deprecated Do not use this variable anymore!

18.97.2.7 iindx

`int* iindx [extern]`

index array to move through `pr`.

The probability for base `i` and `j` to form a pair is in `pr[iindx[i]-j]`.

Deprecated Do not use this variable anymore!

18.98 fold_vars.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_FOLD_VARS_H
00002 #define VIENNA_RNA_PACKAGE_FOLD_VARS_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 /* For now, we include model.h by default to provide backwards compatibility
00006    However, this will most likely change, since fold_vars.h is scheduled to
00007    vanish from the sources at latest in ViennaRNA Package v3
00008 */
00009 #include <ViennaRNA/model.h>
00010
00011
00012 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00013
00022 extern int    fold_constrained;
00023
00027 extern int    csv;
00028
00033 extern char  *RibosumFile;
00034
00039 extern int    james_rule;
00040
00044 extern int    logML;
00045
00057 extern int    cut_point;
00058
00065 extern bondT *base_pair;
00066
00072 extern FLT_OR_DBL *pr;
00073
00080 extern int    *iindx;
00081
00082
00083 #endif
00084
00085
00086 #endif
```

18.99 ViennaRNA/gquad.h File Reference

G-quadruplexes.

Include dependency graph for `gquad.h`:

Functions

- int `get_gquad_matrix` (short *S, vrna_param_t *P)
Get a triangular matrix prefilled with minimum free energy contributions of G-quadruplexes.
- int `parse_gquad` (const char *struc, int *L, int l[3])
- PRIVATE int `backtrack_GQuad_IntLoop` (int c, int i, int j, int type, short *S, int *ggg, int *index, int *p, int *q, vrna_param_t *P)
- PRIVATE int `backtrack_GQuad_IntLoop_L` (int c, int i, int j, int type, short *S, int **ggg, int maxdist, int *p, int *q, vrna_param_t *P)

18.99.1 Detailed Description

G-quadruplexes.

18.100 gquad.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_GQUAD_H
00002 #define VIENNA_RNA_PACKAGE_GQUAD_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/fold_compound.h>
00006 #include <ViennaRNA/params/basic.h>
00007
00008 #ifndef INLINE
00009 #ifdef __GNUC__
00010 # define INLINE inline
00011 #else
00012 # define INLINE
00013 #endif
00014 #endif
00015
00030 int E_gquad(int L,
00031             int l[3],
00032             vrna_param_t *P);
00033
00034
00035 FLT_OR_DBL exp_E_gquad(int L,
00036                        int l[3],
00037                        vrna_exp_param_t *pf);
00038
00039
00040 void E_gquad_ali_en(int i,
00041                    int L,
00042                    int l[3],
00043                    const short **S,
00044                    unsigned int **a2s,
00045                    unsigned int n_seq,
00046                    vrna_param_t *P,
00047                    int en[2]);
00048
00049
00066 int *get_gquad_matrix(short *S,
00067                      vrna_param_t *P);
00068
00069
00070 int *get_gquad_ali_matrix(unsigned int n,
00071                          short *S_cons,
00072                          short **S,
00073                          unsigned int **a2s,
00074                          int n_seq,
00075                          vrna_param_t *P);
00076
00077
00078 FLT_OR_DBL *get_gquad_pf_matrix(short *S,
00079                                FLT_OR_DBL *scale,
00080                                vrna_exp_param_t *pf);
00081
00082
00083 FLT_OR_DBL *get_gquad_pf_matrix_comparative(unsigned int n,
00084                                             short *S_cons,
00085                                             short **S,
00086                                             unsigned int **a2s,
00087                                             FLT_OR_DBL *scale,
00088                                             unsigned int n_seq,
00089                                             vrna_exp_param_t *pf);
00090
00091
```

```

00092 int **get_gquad_L_matrix(short      *S,
00093                          int        start,
00094                          int        maxdist,
00095                          int        n,
00096                          int        **g,
00097                          vrna_param_t *P);
00098
00099
00100 void      vrna_gquad_mx_local_update(vrna_fold_compound_t *vc,
00101                                     int                  start);
00102
00103
00104 void get_gquad_pattern_mfe(short      *S,
00105                          int        i,
00106                          int        j,
00107                          vrna_param_t *P,
00108                          int        *L,
00109                          int        l[3]);
00110
00111
00112 void
00113 get_gquad_pattern_exhaustive(short      *S,
00114                             int        i,
00115                             int        j,
00116                             vrna_param_t *P,
00117                             int        *L,
00118                             int        *l,
00119                             int        threshold);
00120
00121
00122 void get_gquad_pattern_pf(short      *S,
00123                          int        i,
00124                          int        j,
00125                          vrna_exp_param_t *pf,
00126                          int        *L,
00127                          int        l[3]);
00128
00129
00130 plist *get_plist_gquad_from_pr(short      *S,
00131                               int        gi,
00132                               int        gj,
00133                               FLT_OR_DBL *G,
00134                               FLT_OR_DBL *probs,
00135                               FLT_OR_DBL *scale,
00136                               vrna_exp_param_t *pf);
00137
00138
00139 plist *get_plist_gquad_from_pr_max(short      *S,
00140                                   int        gi,
00141                                   int        gj,
00142                                   FLT_OR_DBL *G,
00143                                   FLT_OR_DBL *probs,
00144                                   FLT_OR_DBL *scale,
00145                                   int        *L,
00146                                   int        l[3],
00147                                   vrna_exp_param_t *pf);
00148
00149
00150 plist *get_plist_gquad_from_db(const char *structure,
00151                               float      pr);
00152
00153
00154 plist *
00155 vrna_get_plist_gquad_from_pr(vrna_fold_compound_t *fc,
00156                             int        gi,
00157                             int        gj);
00158
00159
00160 plist *
00161 vrna_get_plist_gquad_from_pr_max(vrna_fold_compound_t *fc,
00162                                  int        gi,
00163                                  int        gj,
00164                                  int        *lmax,
00165                                  int        lmax[3]);
00166
00167
00168 int      get_gquad_count(short *S,
00169                        int    i,
00170                        int    j);
00171
00172
00173 int      get_gquad_layer_count(short *S,
00174                               int    i,
00175                               int    j);
00176
00177
00178 void get_gquad_pattern_mfe_aln(short **S,

```



```

00179             unsigned int **a2s,
00180             short      *S_cons,
00181             int         n_seq,
00182             int         i,
00183             int         j,
00184             vrna_param_t *P,
00185             int         *L,
00186             int         l[3]);
00187
00188
00189 int parse_gquad(const char *struc,
00200                int         *L,
00201                int         l[3]);
00202
00203
00204 INLINE PRIVATE int backtrack_GQuad_IntLoop(int      c,
00205                                           int      i,
00206                                           int      j,
00207                                           int      type,
00208                                           short    *S,
00209                                           int      *ggg,
00210                                           int      *index,
00211                                           int      *p,
00212                                           int      *q,
00213                                           vrna_param_t *P);
00214
00215
00216 INLINE PRIVATE int backtrack_GQuad_IntLoop_comparative(int      c,
00217                                                         int      i,
00218                                                         int      j,
00219                                                         unsigned int *type,
00220                                                         short    *S_cons,
00221                                                         short    **S5,
00222                                                         short    **S3,
00223                                                         unsigned int **a2s,
00224                                                         int      *ggg,
00225                                                         int      *index,
00226                                                         int      *p,
00227                                                         int      *q,
00228                                                         int      n_seq,
00229                                                         vrna_param_t *P);
00230
00231
00232 INLINE PRIVATE int backtrack_GQuad_IntLoop_L(int      c,
00233                                               int      i,
00234                                               int      j,
00235                                               int      type,
00236                                               short    *S,
00237                                               int      *ggg,
00238                                               int      maxdist,
00239                                               int      *p,
00240                                               int      *q,
00241                                               vrna_param_t *P);
00242
00243
00244 PRIVATE INLINE int
00245 vrna_BT_gquad_int(vrna_fold_compound_t *vc,
00246                  int      i,
00247                  int      j,
00248                  int      en,
00249                  vrna_bp_stack_t *bp_stack,
00250                  int      *stack_count);
00251
00252
00253 PRIVATE INLINE int
00254 vrna_BT_gquad_mfe(vrna_fold_compound_t *vc,
00255                  int      i,
00256                  int      j,
00257                  vrna_bp_stack_t *bp_stack,
00258                  int      *stack_count)
00259 {
00260     /*
00261      * here we do some fancy stuff to backtrace the stacksize and linker lengths
00262      * of the g-quadruplex that should reside within position i,j
00263      */
00264     short    *S;
00265     int      l[3], L, a, n_seq;
00266     vrna_param_t *P;
00267
00268     if (vc) {
00269         P = vc->params;
00270         switch (vc->type) {
00271             case VRNA_FC_TYPE_SINGLE:
00272                 S = vc->sequence_encoding2;
00273                 L = -1;
00274
00275                 get_gquad_pattern_mfe(S, i, j, P, &L, l);

```

```

00276         break;
00277
00278     case VRNA_FC_TYPE_COMPARATIVE:
00279         n_seq = vc->n_seq;
00280         L = -1;
00281         get_gquad_pattern_mfe_ali(vc->S, vc->a2s, vc->S_cons, n_seq, i, j, P, &L, l);
00282         break;
00283     }
00284
00285     if (L != -1) {
00286         /* fill the G's of the quadruplex into base_pair2 */
00287         for (a = 0; a < L; a++) {
00288             bp_stack[++(*stack_count)].i = i + a;
00289             bp_stack[(*stack_count)].j = i + a;
00290             bp_stack[++(*stack_count)].i = i + L + l[0] + a;
00291             bp_stack[(*stack_count)].j = i + L + l[0] + a;
00292             bp_stack[++(*stack_count)].i = i + L + l[0] + L + l[1] + a;
00293             bp_stack[(*stack_count)].j = i + L + l[0] + L + l[1] + a;
00294             bp_stack[++(*stack_count)].i = i + L + l[0] + L + l[1] + L + l[2] + a;
00295             bp_stack[(*stack_count)].j = i + L + l[0] + L + l[1] + L + l[2] + a;
00296         }
00297         return 1;
00298     } else {
00299         return 0;
00300     }
00301 }
00302
00303 return 0;
00304 }
00305
00306
00307 PRIVATE INLINE int
00308 vrna_BT_gquad_int(vrna_fold_compound_t *vc,
00309                  int i,
00310                  int j,
00311                  int en,
00312                  vrna_bp_stack_t *bp_stack,
00313                  int *stack_count)
00314 {
00315     int energy, dangles, *idx, ij, p, q, maxl, minl, c0, l1, *ggg;
00316     unsigned char type;
00317     char *ptype;
00318     short si, sj, *S, *S1;
00319
00320     vrna_param_t *P;
00321     vrna_md_t *md;
00322
00323     idx = vc->jindx;
00324     ij = idx[j] + i;
00325     P = vc->params;
00326     md = &(P->model_details);
00327     ptype = vc->ptype;
00328     type = (unsigned char)ptype[ij];
00329     S1 = vc->sequence_encoding;
00330     S = vc->sequence_encoding2;
00331     dangles = md->dangles;
00332     si = S1[i + 1];
00333     sj = S1[j - 1];
00334     ggg = vc->matrices->ggg;
00335     energy = 0;
00336
00337     if (dangles == 2)
00338         energy += P->mismatchI[type][si][sj];
00339
00340     if (type > 2)
00341         energy += P->TerminalAU;
00342
00343     p = i + 1;
00344     if (S1[p] == 3) {
00345         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00346             minl = j - i + p - MAXLOOP - 2;
00347             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00348             minl = MAX2(c0, minl);
00349             c0 = j - 3;
00350             maxl = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00351             maxl = MIN2(c0, maxl);
00352             for (q = minl; q < maxl; q++) {
00353                 if (S[q] != 3)
00354                     continue;
00355
00356                 if (en == energy + ggg[idx[q] + p] + P->internal_loop[j - q - 1])
00357                     return vrna_BT_gquad_mfe(vc, p, q, bp_stack, stack_count);
00358             }
00359         }
00360     }
00361
00362     for (p = i + 2;

```

```

00363     p < j - VRNA_GQUAD_MIN_BOX_SIZE;
00364     p++) {
00365     l1 = p - i - 1;
00366     if (l1 > MAXLOOP)
00367         break;
00368
00369     if (S1[p] != 3)
00370         continue;
00371
00372     minl = j - i + p - MAXLOOP - 2;
00373     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00374     minl = MAX2(c0, minl);
00375     c0 = j - 1;
00376     maxl = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00377     maxl = MIN2(c0, maxl);
00378     for (q = minl; q < maxl; q++) {
00379         if (S1[q] != 3)
00380             continue;
00381
00382         if (en == energy + ggg[idx[q] + p] + P->internal_loop[l1 + j - q - 1])
00383             return vrna_BT_gquad_mfe(vc, p, q, bp_stack, stack_count);
00384     }
00385 }
00386
00387 q = j - 1;
00388 if (S1[q] == 3)
00389     for (p = i + 4;
00390          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
00391          p++) {
00392         l1 = p - i - 1;
00393         if (l1 > MAXLOOP)
00394             break;
00395
00396         if (S1[p] != 3)
00397             continue;
00398
00399         if (en == energy + ggg[idx[q] + p] + P->internal_loop[l1])
00400             return vrna_BT_gquad_mfe(vc, p, q, bp_stack, stack_count);
00401     }
00402
00403 return 0;
00404 }
00405
00406
00424 INLINE PRIVATE int
00425 backtrack_GQuad_IntLoop(int          c,
00426                          int          i,
00427                          int          j,
00428                          int          type,
00429                          short        *S,
00430                          int          *ggg,
00431                          int          *index,
00432                          int          *p,
00433                          int          *q,
00434                          vrna_param_t *P)
00435 {
00436     int energy, dangles, k, l, maxl, minl, c0, l1;
00437     short si, sj;
00438
00439     dangles = P->model_details.dangles;
00440     si = S[i + 1];
00441     sj = S[j - 1];
00442     energy = 0;
00443
00444     if (dangles == 2)
00445         energy += P->mismatchI[type][si][sj];
00446
00447     if (type > 2)
00448         energy += P->TerminalAU;
00449
00450     k = i + 1;
00451     if (S[k] == 3) {
00452         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00453             minl = j - i + k - MAXLOOP - 2;
00454             c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00455             minl = MAX2(c0, minl);
00456             c0 = j - 3;
00457             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00458             maxl = MIN2(c0, maxl);
00459             for (l = minl; l < maxl; l++) {
00460                 if (S[l] != 3)
00461                     continue;
00462
00463                 if (c == energy + ggg[index[l] + k] + P->internal_loop[j - l - 1]) {
00464                     *p = k;
00465                     *q = l;
00466                     return 1;

```

```

00467     }
00468     }
00469     }
00470 }
00471
00472 for (k = i + 2;
00473      k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00474      k++) {
00475     l1 = k - i - 1;
00476     if (l1 > MAXLOOP)
00477         break;
00478
00479     if (S[k] != 3)
00480         continue;
00481
00482     minl = j - i + k - MAXLOOP - 2;
00483     c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00484     minl = MAX2(c0, minl);
00485     c0 = j - 1;
00486     maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00487     maxl = MIN2(c0, maxl);
00488     for (l = minl; l < maxl; l++) {
00489         if (S[l] != 3)
00490             continue;
00491
00492         if (c == energy + ggg[index[l] + k] + P->internal_loop[l1 + j - 1 - 1]) {
00493             *p = k;
00494             *q = l;
00495             return 1;
00496         }
00497     }
00498 }
00499
00500 l = j - 1;
00501 if (S[l] == 3)
00502     for (k = i + 4;
00503          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00504          k++) {
00505         l1 = k - i - 1;
00506         if (l1 > MAXLOOP)
00507             break;
00508
00509         if (S[k] != 3)
00510             continue;
00511
00512         if (c == energy + ggg[index[l] + k] + P->internal_loop[l1]) {
00513             *p = k;
00514             *q = l;
00515             return 1;
00516         }
00517     }
00518
00519 return 0;
00520 }
00521
00522
00523 INLINE PRIVATE int
00524 backtrack_GQuad_IntLoop_comparative(int c,
00525                                     int i,
00526                                     int j,
00527                                     unsigned int *type,
00528                                     short *S_cons,
00529                                     short **S5,
00530                                     short **S3,
00531                                     unsigned int **a2s,
00532                                     int *ggg,
00533                                     int *index,
00534                                     int *p,
00535                                     int *q,
00536                                     int n_seq,
00537                                     vrna_param_t *P)
00538 {
00539     int energy, dangles, k, l, maxl, minl, c0, l1, ss, tt, u1, u2, eee;
00540
00541     dangles = P->model_details.dangles;
00542     energy = 0;
00543
00544     for (ss = 0; ss < n_seq; ss++) {
00545         tt = type[ss];
00546         if (tt == 0)
00547             tt = 7;
00548
00549         if (dangles == 2)
00550             energy += P->mismatchI[tt][S3[ss][i]][S5[ss][j]];
00551
00552         if (tt > 2)
00553             energy += P->TerminalAU;
00554     }

```

```

00554     }
00555
00556     k = i + 1;
00557     if (S_cons[k] == 3) {
00558         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00559             minl = j - i + k - MAXLOOP - 2;
00560             c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00561             minl = MAX2(c0, minl);
00562             c0 = j - 3;
00563             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00564             maxl = MIN2(c0, maxl);
00565             for (l = minl; l < maxl; l++) {
00566                 if (S_cons[l] != 3)
00567                     continue;
00568
00569                 eee = 0;
00570
00571                 for (ss = 0; ss < n_seq; ss++) {
00572                     u1 = a2s[ss][j - 1] - a2s[ss][l];
00573                     eee += P->internal_loop[u1];
00574                 }
00575
00576                 if (c == energy + ggg[index[l] + k] + eee) {
00577                     *p = k;
00578                     *q = l;
00579                     return 1;
00580                 }
00581             }
00582         }
00583     }
00584
00585     for (k = i + 2;
00586          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00587          k++) {
00588         l1 = k - i - 1;
00589         if (l1 > MAXLOOP)
00590             break;
00591
00592         if (S_cons[k] != 3)
00593             continue;
00594
00595         minl = j - i + k - MAXLOOP - 2;
00596         c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00597         minl = MAX2(c0, minl);
00598         c0 = j - 1;
00599         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00600         maxl = MIN2(c0, maxl);
00601         for (l = minl; l < maxl; l++) {
00602             if (S_cons[l] != 3)
00603                 continue;
00604
00605             eee = 0;
00606
00607             for (ss = 0; ss < n_seq; ss++) {
00608                 u1 = a2s[ss][k - 1] - a2s[ss][i];
00609                 u2 = a2s[ss][j - 1] - a2s[ss][l];
00610                 eee += P->internal_loop[u1 + u2];
00611             }
00612
00613             if (c == energy + ggg[index[l] + k] + eee) {
00614                 *p = k;
00615                 *q = l;
00616                 return 1;
00617             }
00618         }
00619     }
00620
00621     l = j - 1;
00622     if (S_cons[l] == 3)
00623         for (k = i + 4;
00624              k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00625              k++) {
00626             l1 = k - i - 1;
00627             if (l1 > MAXLOOP)
00628                 break;
00629
00630             if (S_cons[k] != 3)
00631                 continue;
00632
00633             eee = 0;
00634
00635             for (ss = 0; ss < n_seq; ss++) {
00636                 u1 = a2s[ss][k - 1] - a2s[ss][i];
00637                 eee += P->internal_loop[u1];
00638             }
00639
00640             if (c == energy + ggg[index[l] + k] + eee) {

```

```

00641         *p = k;
00642         *q = l;
00643         return 1;
00644     }
00645 }
00646
00647 return 0;
00648 }
00649
00650
00651 INLINE PRIVATE int
00652 backtrack_GQuad_IntLoop_L(int          c,
00653                           int          i,
00654                           int          j,
00655                           int          type,
00656                           short        *S,
00657                           int          **ggg,
00658                           int          maxdist,
00659                           int          *p,
00660                           int          *q,
00661                           vrna_param_t *P)
00662 {
00663     int energy, dangles, k, l, maxl, minl, c0, ll;
00664     short si, sj;
00665
00666     dangles = P->model_details.dangles;
00667     si = S[i + 1];
00668     sj = S[j - 1];
00669     energy = 0;
00670
00671     if (dangles == 2)
00672         energy += P->mismatchI[type][si][sj];
00673
00674     if (type > 2)
00675         energy += P->TerminalAU;
00676
00677     k = i + 1;
00678     if (S[k] == 3) {
00679         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00680             minl = j - i + k - MAXLOOP - 2;
00681             c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00682             minl = MAX2(c0, minl);
00683             c0 = j - 3;
00684             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00685             maxl = MIN2(c0, maxl);
00686             for (l = minl; l < maxl; l++) {
00687                 if (S[l] != 3)
00688                     continue;
00689
00690                 if (c == energy + ggg[k][l - k] + P->internal_loop[j - l - 1]) {
00691                     *p = k;
00692                     *q = l;
00693                     return 1;
00694                 }
00695             }
00696         }
00697     }
00698
00699     for (k = i + 2;
00700          k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00701          k++) {
00702         ll = k - i - 1;
00703         if (ll > MAXLOOP)
00704             break;
00705
00706         if (S[k] != 3)
00707             continue;
00708
00709         minl = j - i + k - MAXLOOP - 2;
00710         c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00711         minl = MAX2(c0, minl);
00712         c0 = j - 1;
00713         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00714         maxl = MIN2(c0, maxl);
00715         for (l = minl; l < maxl; l++) {
00716             if (S[l] != 3)
00717                 continue;
00718
00719             if (c == energy + ggg[k][l - k] + P->internal_loop[ll + j - l - 1]) {
00720                 *p = k;
00721                 *q = l;
00722                 return 1;
00723             }
00724         }
00725     }
00726
00727     l = j - 1;

```

```

00744     if (S[l] == 3)
00745         for (k = i + 4;
00746             k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00747             k++) {
00748             ll = k - i - 1;
00749             if (ll > MAXLOOP)
00750                 break;
00751
00752             if (S[k] != 3)
00753                 continue;
00754
00755             if (c == energy + ggg[k][l - k] + P->internal_loop[ll]) {
00756                 *p = k;
00757                 *q = l;
00758                 return 1;
00759             }
00760         }
00761
00762     return 0;
00763 }
00764
00765
00766 INLINE PRIVATE int
00767 backtrack_GQuad_IntLoop_L_comparative(int c,
00768                                       int i,
00769                                       int j,
00770                                       unsigned int *type,
00771                                       short *S_cons,
00772                                       short **S5,
00773                                       short **S3,
00774                                       unsigned int **a2s,
00775                                       int **ggg,
00776                                       int *p,
00777                                       int *q,
00778                                       int n_seq,
00779                                       vrna_param_t *P)
00780 {
00781     /*
00782     * The case that is handled here actually resembles something like
00783     * an interior loop where the enclosing base pair is of regular
00784     * kind and the enclosed pair is not a canonical one but a g-quadruplex
00785     * that should then be decomposed further...
00786     */
00787     int mm, dangle_model, k, l, maxl, minl, c0, ll, ss, tt, eee, u1, u2;
00788
00789     dangle_model = P->model_details.dangles;
00790
00791     mm = 0;
00792     for (ss = 0; ss < n_seq; ss++) {
00793         tt = type[ss];
00794
00795         if (dangle_model == 2)
00796             mm += P->mismatchI[tt][S3[ss][i]][S5[ss][j]];
00797
00798         if (tt > 2)
00799             mm += P->TerminalAU;
00800     }
00801
00802     for (k = i + 2;
00803         k < j - VRNA_GQUAD_MIN_BOX_SIZE;
00804         k++) {
00805         if (S_cons[k] != 3)
00806             continue;
00807
00808         ll = k - i - 1;
00809         if (ll > MAXLOOP)
00810             break;
00811
00812         minl = j - i + k - MAXLOOP - 2;
00813         c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00814         minl = MAX2(c0, minl);
00815         c0 = j - 1;
00816         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00817         maxl = MIN2(c0, maxl);
00818         for (l = minl; l < maxl; l++) {
00819             if (S_cons[l] != 3)
00820                 continue;
00821
00822             eee = 0;
00823
00824             for (ss = 0; ss < n_seq; ss++) {
00825                 u1 = a2s[ss][k - 1] - a2s[ss][i];
00826                 u2 = a2s[ss][j - 1] - a2s[ss][l];
00827                 eee += P->internal_loop[u1 + u2];
00828             }
00829
00830             c0 = mm +

```

```

00831         ggg[k][l - k] +
00832         eee;
00833
00834     if (c == c0) {
00835         *p = k;
00836         *q = l;
00837         return 1;
00838     }
00839 }
00840 }
00841 k = i + 1;
00842 if (S_cons[k] == 3) {
00843     if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00844         minl = j - i + k - MAXLOOP - 2;
00845         c0 = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00846         minl = MAX2(c0, minl);
00847         c0 = j - 3;
00848         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00849         maxl = MIN2(c0, maxl);
00850         for (l = minl; l < maxl; l++) {
00851             if (S_cons[l] != 3)
00852                 continue;
00853
00854             eee = 0;
00855
00856             for (ss = 0; ss < n_seq; ss++) {
00857                 ul = a2s[ss][j - 1] - a2s[ss][l];
00858                 eee += P->internal_loop[ul];
00859             }
00860
00861             if (c == mm + ggg[k][l - k] + eee) {
00862                 *p = k;
00863                 *q = l;
00864                 return 1;
00865             }
00866         }
00867     }
00868 }
00869
00870 l = j - 1;
00871 if (S_cons[l] == 3) {
00872     for (k = i + 4; k < j - VRNA_GQUAD_MIN_BOX_SIZE; k++) {
00873         ll = k - i - 1;
00874         if (ll > MAXLOOP)
00875             break;
00876
00877         if (S_cons[k] != 3)
00878             continue;
00879
00880         eee = 0;
00881
00882         for (ss = 0; ss < n_seq; ss++) {
00883             ul = a2s[ss][k - 1] - a2s[ss][i];
00884             eee += P->internal_loop[ul];
00885         }
00886
00887         if (c == mm + ggg[k][l - k] + eee) {
00888             *p = k;
00889             *q = l;
00890             return 1;
00891         }
00892     }
00893 }
00894
00895 return 0;
00896 }
00897
00898
00899 PRIVATE INLINE
00900 int
00901 E_GQuad_IntLoop(int i,
00902                 int j,
00903                 int type,
00904                 short *S,
00905                 int *ggg,
00906                 int *index,
00907                 vrna_param_t *P)
00908 {
00909     int energy, ge, dangles, p, q, ll, minq, maxq, c0;
00910     short si, sj;
00911
00912     dangles = P->model_details.dangles;
00913     si = S[i + 1];
00914     sj = S[j - 1];
00915     energy = 0;
00916
00917     if (dangles == 2)

```



```

00918     energy += P->mismatchI[type][si][sj];
00919
00920     if (type > 2)
00921         energy += P->TerminalAU;
00922
00923     ge = INF;
00924
00925     p = i + 1;
00926     if (S[p] == 3) {
00927         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00928             minq = j - i + p - MAXLOOP - 2;
00929             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00930             minq = MAX2(c0, minq);
00931             c0 = j - 3;
00932             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00933             maxq = MIN2(c0, maxq);
00934             for (q = minq; q < maxq; q++) {
00935                 if (S[q] != 3)
00936                     continue;
00937
00938                 c0 = energy + ggg[index[q] + p] + P->internal_loop[j - q - 1];
00939                 ge = MIN2(ge, c0);
00940             }
00941         }
00942     }
00943
00944     for (p = i + 2;
00945          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
00946          p++) {
00947         l1 = p - i - 1;
00948         if (l1 > MAXLOOP)
00949             break;
00950
00951         if (S[p] != 3)
00952             continue;
00953
00954         minq = j - i + p - MAXLOOP - 2;
00955         c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00956         minq = MAX2(c0, minq);
00957         c0 = j - 1;
00958         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
00959         maxq = MIN2(c0, maxq);
00960         for (q = minq; q < maxq; q++) {
00961             if (S[q] != 3)
00962                 continue;
00963
00964             c0 = energy + ggg[index[q] + p] + P->internal_loop[l1 + j - q - 1];
00965             ge = MIN2(ge, c0);
00966         }
00967     }
00968
00969     q = j - 1;
00970     if (S[q] == 3)
00971         for (p = i + 4;
00972              p < j - VRNA_GQUAD_MIN_BOX_SIZE;
00973              p++) {
00974             l1 = p - i - 1;
00975             if (l1 > MAXLOOP)
00976                 break;
00977
00978             if (S[p] != 3)
00979                 continue;
00980
00981             c0 = energy + ggg[index[q] + p] + P->internal_loop[l1];
00982             ge = MIN2(ge, c0);
00983         }
00984
00985 #if 0
00986     /* here comes the additional stuff for the odd dangle models */
00987     if (dangles % 1) {
00988         en1 = energy + P->dangle5[type][si];
00989         en2 = energy + P->dangle5[type][sj];
00990         en3 = energy + P->mismatchI[type][si][sj];
00991
00992         /* first case with 5' dangle (i.e. j-1) onto enclosing pair */
00993         p = i + 1;
00994         if (S[p] == 3) {
00995             if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
00996                 minq = j - i + p - MAXLOOP - 2;
00997                 c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
00998                 minq = MAX2(c0, minq);
00999                 c0 = j - 4;
01000                 maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01001                 maxq = MIN2(c0, maxq);
01002                 for (q = minq; q < maxq; q++) {
01003                     if (S[q] != 3)
01004                         continue;

```

```

01005
01006         c0 = en1 + ggg[index[q] + p] + P->internal_loop[j - q - 1];
01007         ge = MIN2(ge, c0);
01008     }
01009 }
01010 }
01011
01012 for (p = i + 2; p < j - VRNA_GQUAD_MIN_BOX_SIZE; p++) {
01013     l1 = p - i - 1;
01014     if (l1 > MAXLOOP)
01015         break;
01016
01017     if (S[p] != 3)
01018         continue;
01019
01020     minq = j - i + p - MAXLOOP - 2;
01021     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01022     minq = MAX2(c0, minq);
01023     c0 = j - 2;
01024     maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01025     maxq = MIN2(c0, maxq);
01026     for (q = minq; q < maxq; q++) {
01027         if (S[q] != 3)
01028             continue;
01029
01030         c0 = en1 + ggg[index[q] + p] + P->internal_loop[l1 + j - q - 1];
01031         ge = MIN2(ge, c0);
01032     }
01033 }
01034
01035 q = j - 2;
01036 if (S[q] == 3)
01037     for (p = i + 4; p < j - VRNA_GQUAD_MIN_BOX_SIZE; p++) {
01038         l1 = p - i - 1;
01039         if (l1 > MAXLOOP)
01040             break;
01041
01042         if (S[p] != 3)
01043             continue;
01044
01045         c0 = en1 + ggg[index[q] + p] + P->internal_loop[l1 + 1];
01046         ge = MIN2(ge, c0);
01047     }
01048
01049     /* second case with 3' dangle (i.e. i+1) onto enclosing pair */
01050 }
01051
01052 #endif
01053 return ge;
01054 }
01055
01056
01057 PRIVATE INLINE
01058 int
01059 E_GQuad_IntLoop_comparative(int i,
01060                             int j,
01061                             unsigned int *tt,
01062                             short *S_cons,
01063                             short **S5,
01064                             short **S3,
01065                             unsigned int **a2s,
01066                             int *ggg,
01067                             int *index,
01068                             int n_seq,
01069                             vrna_param_t *P)
01070 {
01071     unsigned int type;
01072     int eee, energy, ge, p, q, l1, u1, u2, minq, maxq, c0, s;
01073     vrna_md_t *md;
01074
01075     md = &(P->model_details);
01076     energy = 0;
01077
01078     for (s = 0; s < n_seq; s++) {
01079         type = tt[s];
01080         if (md->dangles == 2)
01081             energy += P->mismatchI[type][S3[s][i]][S5[s][j]];
01082
01083         if (type > 2)
01084             energy += P->TerminalAU;
01085     }
01086
01087     ge = INF;
01088
01089     p = i + 1;
01090     if (S_cons[p] == 3) {
01091         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {

```

```

01092     minq = j - i + p - MAXLOOP - 2;
01093     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01094     minq = MAX2(c0, minq);
01095     c0 = j - 3;
01096     maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01097     maxq = MIN2(c0, maxq);
01098     for (q = minq; q < maxq; q++) {
01099         if (S_cons[q] != 3)
01100             continue;
01101
01102         eee = 0;
01103
01104         for (s = 0; s < n_seq; s++) {
01105             u1 = a2s[s][j - 1] - a2s[s][q];
01106             eee += P->internal_loop[u1];
01107         }
01108
01109         c0 = energy +
01110             ggg[index[q] + p] +
01111             eee;
01112         ge = MIN2(ge, c0);
01113     }
01114 }
01115 }
01116
01117 for (p = i + 2;
01118     p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01119     p++) {
01120     l1 = p - i - 1;
01121     if (l1 > MAXLOOP)
01122         break;
01123
01124     if (S_cons[p] != 3)
01125         continue;
01126
01127     minq = j - i + p - MAXLOOP - 2;
01128     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01129     minq = MAX2(c0, minq);
01130     c0 = j - 1;
01131     maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01132     maxq = MIN2(c0, maxq);
01133     for (q = minq; q < maxq; q++) {
01134         if (S_cons[q] != 3)
01135             continue;
01136
01137         eee = 0;
01138
01139         for (s = 0; s < n_seq; s++) {
01140             u1 = a2s[s][p - 1] - a2s[s][i];
01141             u2 = a2s[s][j - 1] - a2s[s][q];
01142             eee += P->internal_loop[u1 + u2];
01143         }
01144
01145         c0 = energy +
01146             ggg[index[q] + p] +
01147             eee;
01148         ge = MIN2(ge, c0);
01149     }
01150 }
01151
01152 q = j - 1;
01153 if (S_cons[q] == 3)
01154     for (p = i + 4;
01155         p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01156         p++) {
01157         l1 = p - i - 1;
01158         if (l1 > MAXLOOP)
01159             break;
01160
01161         if (S_cons[p] != 3)
01162             continue;
01163
01164         eee = 0;
01165
01166         for (s = 0; s < n_seq; s++) {
01167             u1 = a2s[s][p - 1] - a2s[s][i];
01168             eee += P->internal_loop[u1];
01169         }
01170
01171         c0 = energy +
01172             ggg[index[q] + p] +
01173             eee;
01174         ge = MIN2(ge, c0);
01175     }
01176
01177 return ge;
01178 }

```

```

01179
01180
01181 PRIVATE INLINE
01182 int
01183 E_GQuad_IntLoop_L_comparative(int i,
01184                                int j,
01185                                unsigned int *tt,
01186                                short *S_cons,
01187                                short **S5,
01188                                short **S3,
01189                                unsigned int **a2s,
01190                                int **ggg,
01191                                int n_seq,
01192                                vrna_param_t *P)
01193 {
01194     unsigned int type;
01195     int eee, energy, ge, p, q, ll, u1, u2, minq, maxq, c0, s;
01196     vrna_md_t *md;
01197
01198     md = &(P->model_details);
01199     energy = 0;
01200
01201     for (s = 0; s < n_seq; s++) {
01202         type = tt[s];
01203         if (md->dangles == 2)
01204             energy += P->mismatchI[type][S3[s][i]][S5[s][j]];
01205
01206         if (type > 2)
01207             energy += P->TerminalAU;
01208     }
01209
01210     ge = INF;
01211
01212     p = i + 1;
01213     if (S_cons[p] == 3) {
01214         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
01215             minq = j - i + p - MAXLOOP - 2;
01216             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01217             minq = MAX2(c0, minq);
01218             c0 = j - 3;
01219             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01220             maxq = MIN2(c0, maxq);
01221             for (q = minq; q < maxq; q++) {
01222                 if (S_cons[q] != 3)
01223                     continue;
01224
01225                 eee = 0;
01226
01227                 for (s = 0; s < n_seq; s++) {
01228                     u1 = a2s[s][j - 1] - a2s[s][q];
01229                     eee += P->internal_loop[u1];
01230                 }
01231
01232                 c0 = energy +
01233                     ggg[p][q - p] +
01234                     eee;
01235                 ge = MIN2(ge, c0);
01236             }
01237         }
01238     }
01239
01240     for (p = i + 2;
01241          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01242          p++) {
01243         ll = p - i - 1;
01244         if (ll > MAXLOOP)
01245             break;
01246
01247         if (S_cons[p] != 3)
01248             continue;
01249
01250         minq = j - i + p - MAXLOOP - 2;
01251         c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01252         minq = MAX2(c0, minq);
01253         c0 = j - 1;
01254         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01255         maxq = MIN2(c0, maxq);
01256         for (q = minq; q < maxq; q++) {
01257             if (S_cons[q] != 3)
01258                 continue;
01259
01260             eee = 0;
01261
01262             for (s = 0; s < n_seq; s++) {
01263                 u1 = a2s[s][p - 1] - a2s[s][i];
01264                 u2 = a2s[s][j - 1] - a2s[s][q];
01265                 eee += P->internal_loop[u1 + u2];

```

```

01266     }
01267
01268     c0 = energy +
01269         ggg[p][q - p] +
01270         eee;
01271     ge = MIN2(ge, c0);
01272 }
01273 }
01274
01275 q = j - 1;
01276 if (S_cons[q] == 3)
01277     for (p = i + 4;
01278         p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01279         p++) {
01280         l1 = p - i - 1;
01281         if (l1 > MAXLOOP)
01282             break;
01283
01284         if (S_cons[p] != 3)
01285             continue;
01286
01287         eee = 0;
01288
01289         for (s = 0; s < n_seq; s++) {
01290             u1 = a2s[s][p - 1] - a2s[s][i];
01291             eee += P->internal_loop[u1];
01292         }
01293
01294         c0 = energy +
01295             ggg[p][q - p] +
01296             eee;
01297         ge = MIN2(ge, c0);
01298     }
01299
01300     return ge;
01301 }
01302
01303
01304 PRIVATE INLINE
01305 int *
01306 E_GQuad_IntLoop_exhaustive(int      i,
01307                             int      j,
01308                             int      **p_p,
01309                             int      **q_p,
01310                             int      type,
01311                             short    *S,
01312                             int      *ggg,
01313                             int      threshold,
01314                             int      *index,
01315                             vrna_param_t *P)
01316 {
01317     int  energy, *ge, dangles, p, q, l1, minq, maxq, c0;
01318     short si, sj;
01319     int  cnt = 0;
01320
01321     dangles = P->model_details.dangles;
01322     si = S[i + 1];
01323     sj = S[j - 1];
01324     energy = 0;
01325
01326     if (dangles == 2)
01327         energy += P->mismatchI[type][si][sj];
01328
01329     if (type > 2)
01330         energy += P->TerminalAU;
01331
01332     /* guess how many gquads are possible in interval [i+1,j-1] */
01333     *p_p = (int *)vrna_alloc(sizeof(int) * 256);
01334     *q_p = (int *)vrna_alloc(sizeof(int) * 256);
01335     ge = (int *)vrna_alloc(sizeof(int) * 256);
01336
01337     p = i + 1;
01338     if (S[p] == 3) {
01339         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
01340             minq = j - i + p - MAXLOOP - 2;
01341             c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01342             minq = MAX2(c0, minq);
01343             c0 = j - 3;
01344             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01345             maxq = MIN2(c0, maxq);
01346             for (q = minq; q < maxq; q++) {
01347                 if (S[q] != 3)
01348                     continue;
01349
01350                 c0 = energy + ggg[index[q] + p] + P->internal_loop[j - q - 1];
01351                 if (c0 <= threshold) {
01352                     ge[cnt] = energy + P->internal_loop[j - q - 1];

```

```

01353         (*p_p)[cnt] = p;
01354         (*q_p)[cnt++] = q;
01355     }
01356 }
01357 }
01358 }
01359
01360 for (p = i + 2;
01361      p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01362      p++) {
01363     ll = p - i - 1;
01364     if (ll > MAXLOOP)
01365         break;
01366
01367     if (S[p] != 3)
01368         continue;
01369
01370     minq = j - i + p - MAXLOOP - 2;
01371     c0 = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01372     minq = MAX2(c0, minq);
01373     c0 = j - 1;
01374     maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01375     maxq = MIN2(c0, maxq);
01376     for (q = minq; q < maxq; q++) {
01377         if (S[q] != 3)
01378             continue;
01379
01380         c0 = energy + ggg[index[q] + p] + P->internal_loop[ll + j - q - 1];
01381         if (c0 <= threshold) {
01382             ge[cnt] = energy + P->internal_loop[ll + j - q - 1];
01383             (*p_p)[cnt] = p;
01384             (*q_p)[cnt++] = q;
01385         }
01386     }
01387 }
01388
01389 q = j - 1;
01390 if (S[q] == 3)
01391     for (p = i + 4;
01392          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01393          p++) {
01394         ll = p - i - 1;
01395         if (ll > MAXLOOP)
01396             break;
01397
01398         if (S[p] != 3)
01399             continue;
01400
01401         c0 = energy + ggg[index[q] + p] + P->internal_loop[ll];
01402         if (c0 <= threshold) {
01403             ge[cnt] = energy + P->internal_loop[ll];
01404             (*p_p)[cnt] = p;
01405             (*q_p)[cnt++] = q;
01406         }
01407     }
01408
01409 (*p_p)[cnt] = -1;
01410
01411 return ge;
01412 }
01413
01414 PRIVATE INLINE
01415 int
01416 E_GQuad_IntLoop_L(int i,
01417                   int j,
01418                   int type,
01419                   short *S,
01420                   int **ggg,
01421                   int maxdist,
01422                   vrna_param_t *P)
01423 {
01424     int energy, ge, dangles, p, q, ll, minq, maxq, c0;
01425     short si, sj;
01426
01427     dangles = P->model_details.dangles;
01428     si = S[i + 1];
01429     sj = S[j - 1];
01430     energy = 0;
01431
01432     if (dangles == 2)
01433         energy += P->mismatchI[type][si][sj];
01434
01435     if (type > 2)
01436         energy += P->TerminalAU;
01437
01438     ge = INF;

```

```

01440
01441     p = i + 1;
01442     if (S[p] == 3) {
01443         if (p < j - VRNA_GQUAD_MIN_BOX_SIZE) {
01444             minq = j - i + p - MAXLOOP - 2;
01445             c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01446             minq = MAX2(c0, minq);
01447             c0   = j - 3;
01448             maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01449             maxq = MIN2(c0, maxq);
01450             for (q = minq; q < maxq; q++) {
01451                 if (S[q] != 3)
01452                     continue;
01453
01454                 c0 = energy + ggg[p][q - p] + P->internal_loop[j - q - 1];
01455                 ge = MIN2(ge, c0);
01456             }
01457         }
01458     }
01459
01460     for (p = i + 2;
01461          p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01462          p++) {
01463         l1 = p - i - 1;
01464         if (l1 > MAXLOOP)
01465             break;
01466
01467         if (S[p] != 3)
01468             continue;
01469
01470         minq = j - i + p - MAXLOOP - 2;
01471         c0   = p + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01472         minq = MAX2(c0, minq);
01473         c0   = j - 1;
01474         maxq = p + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01475         maxq = MIN2(c0, maxq);
01476         for (q = minq; q < maxq; q++) {
01477             if (S[q] != 3)
01478                 continue;
01479
01480             c0 = energy + ggg[p][q - p] + P->internal_loop[l1 + j - q - 1];
01481             ge = MIN2(ge, c0);
01482         }
01483     }
01484
01485     q = j - 1;
01486     if (S[q] == 3)
01487         for (p = i + 4;
01488              p < j - VRNA_GQUAD_MIN_BOX_SIZE;
01489              p++) {
01490             l1 = p - i - 1;
01491             if (l1 > MAXLOOP)
01492                 break;
01493
01494             if (S[p] != 3)
01495                 continue;
01496
01497             c0 = energy + ggg[p][q - p] + P->internal_loop[l1];
01498             ge = MIN2(ge, c0);
01499         }
01500
01501     return ge;
01502 }
01503
01504
01505 PRIVATE INLINE
01506 FLT_OR_DBL
01507 exp_E_GQuad_IntLoop(int          i,
01508                     int          j,
01509                     int          type,
01510                     short        *S,
01511                     FLT_OR_DBL  *G,
01512                     FLT_OR_DBL  *scale,
01513                     int          *index,
01514                     vrna_exp_param_t *pf)
01515 {
01516     int      k, l, minl, maxl, u, r;
01517     FLT_OR_DBL q, qe;
01518     double     *expintern;
01519     short      si, sj;
01520
01521     q      = 0;
01522     si     = S[i + 1];
01523     sj     = S[j - 1];
01524     qe     = (FLT_OR_DBL)pf->expmismatchI[type][si][sj];
01525     expintern = &(pf->expinternal[0]);
01526

```

```

01527     if (type > 2)
01528         qe *= (FLT_OR_DBL)pf->expTermAU;
01529
01530     k = i + 1;
01531     if (S[k] == 3) {
01532         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
01533             minl = j - MAXLOOP - 1;
01534             u = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01535             minl = MAX2(u, minl);
01536             u = j - 3;
01537             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01538             maxl = MIN2(u, maxl);
01539             for (l = minl; l < maxl; l++) {
01540                 if (S[l] != 3)
01541                     continue;
01542
01543                 if (G[index[k] - 1] == 0.)
01544                     continue;
01545
01546                 q += qe
01547                     * G[index[k] - 1]
01548                     * (FLT_OR_DBL)expintern[j - 1 - 1]
01549                     * scale[j - 1 + 1];
01550             }
01551         }
01552     }
01553
01554     for (k = i + 2;
01555          k <= j - VRNA_GQUAD_MIN_BOX_SIZE;
01556          k++) {
01557         u = k - i - 1;
01558         if (u > MAXLOOP)
01559             break;
01560
01561         if (S[k] != 3)
01562             continue;
01563
01564         minl = j - i + k - MAXLOOP - 2;
01565         r = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01566         minl = MAX2(r, minl);
01567         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01568         r = j - 1;
01569         maxl = MIN2(r, maxl);
01570         for (l = minl; l < maxl; l++) {
01571             if (S[l] != 3)
01572                 continue;
01573
01574             if (G[index[k] - 1] == 0.)
01575                 continue;
01576
01577             q += qe
01578                 * G[index[k] - 1]
01579                 * (FLT_OR_DBL)expintern[u + j - 1 - 1]
01580                 * scale[u + j - 1 + 1];
01581         }
01582     }
01583
01584     l = j - 1;
01585     if (S[l] == 3)
01586         for (k = i + 4; k <= j - VRNA_GQUAD_MIN_BOX_SIZE; k++) {
01587             u = k - i - 1;
01588             if (u > MAXLOOP)
01589                 break;
01590
01591             if (S[k] != 3)
01592                 continue;
01593
01594             if (G[index[k] - 1] == 0.)
01595                 continue;
01596
01597             q += qe
01598                 * G[index[k] - 1]
01599                 * (FLT_OR_DBL)expintern[u]
01600                 * scale[u + 2];
01601         }
01602
01603     return q;
01604 }
01605
01606
01607 PRIVATE INLINE
01608 FLT_OR_DBL
01609 exp_E_GQuad_IntLoop_comparative(int i,
01610                                  int j,
01611                                  unsigned int *tt,
01612                                  short *S_cons,
01613                                  short **S5,

```



```

01614             short          **S3,
01615             unsigned int    **a2s,
01616             FLT_OR_DBL      *G,
01617             FLT_OR_DBL      *scale,
01618             int              *index,
01619             int              n_seq,
01620             vrna_exp_param_t *pf)
01621 {
01622     unsigned int type;
01623     int k, l, minl, maxl, u, ul, u2, r, s;
01624     FLT_OR_DBL q, qe, qqq;
01625     double *expintern;
01626     vrna_md_t *md;
01627
01628     q = 0;
01629     qe = 1.;
01630     md = &(pf->model_details);
01631     expintern = &(pf->expinternal[0]);
01632
01633     for (s = 0; s < n_seq; s++) {
01634         type = tt[s];
01635         if (md->dangles == 2)
01636             qe *= (FLT_OR_DBL)pf->expmismatchI[type][S3[s][i]][S5[s][j]];
01637
01638         if (type > 2)
01639             qe *= (FLT_OR_DBL)pf->expTermAU;
01640     }
01641
01642     k = i + 1;
01643     if (S_cons[k] == 3) {
01644         if (k < j - VRNA_GQUAD_MIN_BOX_SIZE) {
01645             minl = j - MAXLOOP - 1;
01646             u = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01647             minl = MAX2(u, minl);
01648             u = j - 3;
01649             maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01650             maxl = MIN2(u, maxl);
01651             for (l = minl; l < maxl; l++) {
01652                 if (S_cons[l] != 3)
01653                     continue;
01654
01655                 if (G[index[k] - 1] == 0.)
01656                     continue;
01657
01658                 qqq = 1.;
01659
01660                 for (s = 0; s < n_seq; s++) {
01661                     ul = a2s[s][j - 1] - a2s[s][l];
01662                     qqq *= expintern[ul];
01663                 }
01664
01665                 q += qe *
01666                     G[index[k] - 1] *
01667                     qqq *
01668                     scale[j - 1 + 1];
01669             }
01670         }
01671     }
01672
01673     for (k = i + 2;
01674          k <= j - VRNA_GQUAD_MIN_BOX_SIZE;
01675          k++) {
01676         u = k - i - 1;
01677         if (u > MAXLOOP)
01678             break;
01679
01680         if (S_cons[k] != 3)
01681             continue;
01682
01683         minl = j - i + k - MAXLOOP - 2;
01684         r = k + VRNA_GQUAD_MIN_BOX_SIZE - 1;
01685         minl = MAX2(r, minl);
01686         maxl = k + VRNA_GQUAD_MAX_BOX_SIZE + 1;
01687         r = j - 1;
01688         maxl = MIN2(r, maxl);
01689         for (l = minl; l < maxl; l++) {
01690             if (S_cons[l] != 3)
01691                 continue;
01692
01693             if (G[index[k] - 1] == 0.)
01694                 continue;
01695
01696             qqq = 1.;
01697
01698             for (s = 0; s < n_seq; s++) {
01699                 ul = a2s[s][k - 1] - a2s[s][i];
01700                 u2 = a2s[s][j - 1] - a2s[s][l];

```

```

01701         qqg *= expintern[u1 + u2];
01702     }
01703
01704     q += qe *
01705         G[index[k] - 1] *
01706         qqg *
01707         scale[u + j - 1 + 1];
01708 }
01709 }
01710
01711 l = j - 1;
01712 if (S_cons[l] == 3)
01713     for (k = i + 4; k <= j - VRNA_GQUAD_MIN_BOX_SIZE; k++) {
01714         u = k - i - 1;
01715         if (u > MAXLOOP)
01716             break;
01717
01718         if (S_cons[k] != 3)
01719             continue;
01720
01721         if (G[index[k] - 1] == 0.)
01722             continue;
01723
01724         qqg = 1.;
01725
01726         for (s = 0; s < n_seq; s++) {
01727             u1 = a2s[s][k - 1] - a2s[s][i];
01728             qqg *= expintern[u1];
01729         }
01730
01731         q += qe *
01732             G[index[k] - 1] *
01733             qqg *
01734             scale[u + 2];
01735     }
01736
01737     return q;
01738 }
01739
01740
01746 #endif

```

18.101 ViennaRNA/grammar.h File Reference

Implementations for the RNA folding grammar.

Include dependency graph for grammar.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_gr_aux_s](#)

Typedefs

- typedef void(* [vrna_grammar_data_free_f](#)) (void *data)

Free auxiliary data.

18.101.1 Detailed Description

Implementations for the RNA folding grammar.

18.102 grammar.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_GRAMMAR_H
00002 #define VIENNA_RNA_PACKAGE_GRAMMAR_H
00003
00016 #include <ViennaRNA/fold_compound.h>
00017
00018 typedef int (*vrna_grammar_rule_f)(vrna_fold_compound_t *vc,
00019                                     int i,
00020                                     int j,
00021                                     void *data);
00022

```

```
00023
00024 typedef void (*vrna_grammar_rule_f_aux) (vrna_fold_compound_t *vc,
00025                                           int i,
00026                                           int j,
00027                                           void *data);
00028
00029
00030 typedef FLT_OR_DBL (*vrna_grammar_rule_f_exp) (vrna_fold_compound_t *vc,
00031                                                int i,
00032                                                int j,
00033                                                void *data);
00034
00035
00036 typedef void (*vrna_grammar_rule_f_aux_exp) (vrna_fold_compound_t *vc,
00037                                              int i,
00038                                              int j,
00039                                              void *data);
00040
00041
00042 typedef void (*vrna_grammar_cond_f) (vrna_fold_compound_t *fc,
00043                                     unsigned char stage,
00044                                     void *data);
00045
00046
00051 typedef void (*vrna_grammar_data_free_f) (void *data);
00052
00053
00054 typedef struct vrna_gr_aux_s vrna_gr_aux_t;
00055
00056
00057 struct vrna_gr_aux_s {
00058     vrna_grammar_cond_f      cb_proc;
00059     vrna_grammar_rule_f      cb_aux_f;
00060     vrna_grammar_rule_f      cb_aux_c;
00061     vrna_grammar_rule_f      cb_aux_m;
00062     vrna_grammar_rule_f      cb_aux_ml;
00063     vrna_grammar_rule_f_aux   cb_aux;
00064
00065     vrna_grammar_rule_f_exp   cb_aux_exp_f;
00066     vrna_grammar_rule_f_exp   cb_aux_exp_c;
00067     vrna_grammar_rule_f_exp   cb_aux_exp_m;
00068     vrna_grammar_rule_f_exp   cb_aux_exp_ml;
00069     vrna_grammar_rule_f_aux_exp cb_aux_exp;
00070
00071     void *data;
00072     vrna_grammar_data_free_f free_data;
00073 };
00074
00075
00076
00077 int
00078 vrna_gr_set_aux_f(vrna_fold_compound_t *fc,
00079                  vrna_grammar_rule_f cb);
00080
00081
00082 int
00083 vrna_gr_set_aux_exp_f(vrna_fold_compound_t *fc,
00084                      vrna_grammar_rule_f_exp cb);
00085
00086
00087 int
00088 vrna_gr_set_aux_c(vrna_fold_compound_t *fc,
00089                  vrna_grammar_rule_f cb);
00090
00091
00092 int
00093 vrna_gr_set_aux_exp_c(vrna_fold_compound_t *fc,
00094                     vrna_grammar_rule_f_exp cb);
00095
00096
00097 int
00098 vrna_gr_set_aux_m(vrna_fold_compound_t *fc,
00099                  vrna_grammar_rule_f cb);
00100
00101
00102 int
00103 vrna_gr_set_aux_exp_m(vrna_fold_compound_t *fc,
00104                     vrna_grammar_rule_f_exp cb);
00105
00106
00107 int
00108 vrna_gr_set_aux_ml(vrna_fold_compound_t *fc,
00109                   vrna_grammar_rule_f cb);
00110
00111
00112 int
00113 vrna_gr_set_aux_exp_ml(vrna_fold_compound_t *fc,
00114                      vrna_grammar_rule_f_exp cb);
```

```

00115
00116
00117 int
00118 vrna_gr_set_aux(vrna_fold_compound_t *fc,
00119                vrna_grammar_rule_f_aux cb);
00120
00121
00122 int
00123 vrna_gr_set_aux_exp(vrna_fold_compound_t *fc,
00124                    vrna_grammar_rule_f_aux_exp cb);
00125
00126
00127 int
00128 vrna_gr_set_data(vrna_fold_compound_t *fc,
00129                 void *data,
00130                 vrna_grammar_data_free_f free_data);
00131
00132
00133 int
00134 vrna_gr_set_cond(vrna_fold_compound_t *fc,
00135                 vrna_grammar_cond_f cb);
00136
00137
00138 int
00139 vrna_gr_reset(vrna_fold_compound_t *fc);
00140
00141
00151 #endif

```

18.103 ViennaRNA/hairpin_loops.h File Reference

Use [ViennaRNA/loops/hairpin.h](#) instead.

Include dependency graph for hairpin_loops.h:

18.103.1 Detailed Description

Use [ViennaRNA/loops/hairpin.h](#) instead.

Deprecated Use [ViennaRNA/loops/hairpin.h](#) instead

18.104 hairpin_loops.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/hairpin_loops.h>! Use
00013         <ViennaRNA/loops/hairpin.h> instead!"
00013 # endif
00014 #include <ViennaRNA/loops/hairpin.h>
00015 #endif
00016
00017 #endif

```

18.105 ViennaRNA/heat_capacity.h File Reference

Compute heat capacity for an RNA.

Include dependency graph for heat_capacity.h:

Data Structures

- struct [vrna_heat_capacity_s](#)

A single result from heat capacity computations. [More...](#)

Typedefs

- typedef void(* [vrna_heat_capacity_f](#)) (float temp, float heat_capacity, void *data)

The callback for heat capacity predictions.

- typedef struct `vrna_heat_capacity_s` `vrna_heat_capacity_t`

A single result from heat capacity computations.

Functions

Basic heat capacity function interface

- `vrna_heat_capacity_t * vrna_heat_capacity` (`vrna_fold_compound_t *fc`, float `T_min`, float `T_max`, float `T_increment`, unsigned int `mpoints`)

Compute the specific heat for an RNA.

- int `vrna_heat_capacity_cb` (`vrna_fold_compound_t *fc`, float `T_min`, float `T_max`, float `T_increment`, unsigned int `mpoints`, `vrna_heat_capacity_f` `cb`, void `*data`)

Compute the specific heat for an RNA (callback variant)

Simplified heat capacity computation

- `vrna_heat_capacity_t * vrna_heat_capacity_simple` (const char `*sequence`, float `T_min`, float `T_max`, float `T_increment`, unsigned int `mpoints`)

Compute the specific heat for an RNA (simplified variant)

18.105.1 Detailed Description

Compute heat capacity for an RNA.

This file includes the interface to all functions related to predicting the heat capacity for an RNA.

18.106 heat_capacity.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_MELTING_H
00002 #define VIENNA_RNA_PACKAGE_MELTING_H
00003
00004 #include <stdio.h>
00005
00006 #include <ViennaRNA/datastructures/basic.h>
00007
00008 #ifdef VRNA_WARN_DEPRECATED
00009 # if defined(DEPRECATED)
00010 #  undef DEPRECATED
00011 # endif
00012 # if defined(__clang__)
00013 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00014 # elif defined(__GNUC__)
00015 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00016 # else
00017 #  define DEPRECATED(func, msg) func
00018 # endif
00019 #else
00020 # define DEPRECATED(func, msg) func
00021 #endif
00022
00023 typedef void (*vrna_heat_capacity_f)(float temp,
00024                                     float heat_capacity,
00025                                     void *data);
00026
00027 DEPRECATED(typedef void (vrna_heat_capacity_callback)(float temp,
00028                                                       float heat_capacity,
00029                                                       void *data),
00030            "Use vrna_heat_capacity_f instead!");
00031
00032 typedef struct vrna_heat_capacity_s vrna_heat_capacity_t;
00033
00034 struct vrna_heat_capacity_s {
00035     float temperature;
00036     float heat_capacity;
00037 };
00038
00039 vrna_heat_capacity_t *
00040 vrna_heat_capacity(vrna_fold_compound_t *fc,
```

```

00113             float           T_min,
00114             float           T_max,
00115             float           T_increment,
00116             unsigned int    mpoints);
00117
00118
00148 int
00149 vrna_heat_capacity_cb(vrna_fold_compound_t *fc,
00150                     float           T_min,
00151                     float           T_max,
00152                     float           T_increment,
00153                     unsigned int    mpoints,
00154                     vrna_heat_capacity_f cb,
00155                     void           *data);
00156
00157
00158 /* End basic interface */
00192 vrna_heat_capacity_t *
00193 vrna_heat_capacity_simple(const char *sequence,
00194                         float       T_min,
00195                         float       T_max,
00196                         float       T_increment,
00197                         unsigned int mpoints);
00198
00199 /* End basic interface */
00202 /* End thermodynamics */
00205 #endif

```

18.107 ViennaRNA/interior_loops.h File Reference

Use [ViennaRNA/loops/internal.h](#) instead.

Include dependency graph for interior_loops.h:

18.107.1 Detailed Description

Use [ViennaRNA/loops/internal.h](#) instead.

Deprecated Use [ViennaRNA/loops/internal.h](#) instead

18.108 interior_loops.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/interior_loops.h>! Use
00013         <ViennaRNA/loops/internal.h> instead!"
00013 # endif
00014 #include <ViennaRNA/loops/internal.h>
00015 #endif
00016
00017 #endif

```

18.109 ViennaRNA/inverse.h File Reference

Inverse folding routines.

Functions

- float [inverse_fold](#) (char *start, const char *target)
Find sequences with predefined structure.
- float [inverse_pf_fold](#) (char *start, const char *target)
Find sequence that maximizes probability of a predefined structure.

Variables

- char * **symbolset**
This global variable points to the allowed bases, initially "AUGC". It can be used to design sequences from reduced alphabets.
- float **final_cost**
- int **give_up**
- int **inv_verbose**

18.109.1 Detailed Description

Inverse folding routines.

18.110 inverse.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_INVERSE_H
00002 #define VIENNA_RNA_PACKAGE_INVERSE_H
00003
00021 extern char *symbolset;
00023 extern float final_cost;
00025 extern int give_up;
00027 extern int inv_verbose;
00028
00045 float inverse_fold( char *start,
00046                    const char *target);
00047
00061 float inverse_pf_fold(char *start,
00062                      const char *target);
00063
00067 #endif
```

18.111 ViennaRNA/landscape/move.h File Reference

Methods to operate with structural neighbors of RNA secondary structures.

This graph shows which files directly or indirectly include this file:

Data Structures

- struct **vrna_move_s**
An atomic representation of the transition / move from one structure to its neighbor. [More...](#)

Macros

- #define **VRNA_MOVESET_INSERTION** 4
Option flag indicating insertion move.
- #define **VRNA_MOVESET_DELETION** 8
Option flag indicating deletion move.
- #define **VRNA_MOVESET_SHIFT** 16
Option flag indicating shift move.
- #define **VRNA_MOVESET_NO_LP** 32
Option flag indicating moves without lonely base pairs.
- #define **VRNA_MOVESET_DEFAULT** (VRNA_MOVESET_INSERTION | VRNA_MOVESET_DELETION)
Option flag indicating default move set, i.e. insertions/deletion of a base pair.

Typedefs

- typedef struct **vrna_move_s** **vrna_move_t**
A single move that transforms a secondary structure into one of its neighbors.

Functions

- `vrna_move_t vrna_move_init` (int pos_5, int pos_3)
Create an atomic move.
- void `vrna_move_list_free` (vrna_move_t *moves)
- void `vrna_move_apply` (short *pt, const vrna_move_t *m)
Apply a particular move / transition to a secondary structure, i.e. transform a structure.
- int `vrna_move_is_removal` (const vrna_move_t *m)
Test whether a move is a base pair removal.
- int `vrna_move_is_insertion` (const vrna_move_t *m)
Test whether a move is a base pair insertion.
- int `vrna_move_is_shift` (const vrna_move_t *m)
Test whether a move is a base pair shift.
- int `vrna_move_compare` (const vrna_move_t *a, const vrna_move_t *b, const short *pt)
Compare two moves.

18.111.1 Detailed Description

Methods to operate with structural neighbors of RNA secondary structures.

18.112 move.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_MOVE_H
00002 #define VIENNA_RNA_PACKAGE_MOVE_H
00003
00004
00020 typedef struct vrna_move_s vrna_move_t;
00021
00026 #define VRNA_MOVESET_INSERTION 4
00027
00032 #define VRNA_MOVESET_DELETION 8
00033
00038 #define VRNA_MOVESET_SHIFT 16
00043 #define VRNA_MOVESET_NO_LP 32
00044
00049 #define VRNA_MOVESET_DEFAULT (VRNA_MOVESET_INSERTION | VRNA_MOVESET_DELETION)
00050
00051
00073 struct vrna_move_s {
00074     int pos_5;
00075     int pos_3;
00076     vrna_move_t *next;
00077 };
00080
00081
00091 vrna_move_t
00092 vrna_move_init(int pos_5,
00093               int pos_3);
00094
00095
00099 void
00100 vrna_move_list_free(vrna_move_t *moves);
00101
00102
00109 void
00110 vrna_move_apply(short *pt,
00111                 const vrna_move_t *m);
00112
00113
00114 void
00115 vrna_move_apply_db(char *structure,
00116                   const short *pt,
00117                   const vrna_move_t *m);
00118
00119
00126 int
00127 vrna_move_is_removal(const vrna_move_t *m);
00128
00129
00136 int
00137 vrna_move_is_insertion(const vrna_move_t *m);
00138

```



```

00139
00146 int
00147 vrna_move_is_shift(const vrna_move_t *m);
00148
00149
00170 int
00171 vrna_move_compare(const vrna_move_t *a,
00172                  const vrna_move_t *b,
00173                  const short      *pt);
00174
00175
00180 #endif

```

18.113 ViennaRNA/landscape/paths.h File Reference

API for computing (optimal) (re-)folding paths between secondary structures.

Include dependency graph for paths.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_path_s](#)
An element of a refolding path list. [More...](#)

Macros

- #define [VRNA_PATH_TYPE_DOT_BRACKET](#) 1U
Flag to indicate producing a (re-)folding path as list of dot-bracket structures.
- #define [VRNA_PATH_TYPE_MOVES](#) 2U
Flag to indicate producing a (re-)folding path as list of transition moves.

Typedefs

- typedef struct [vrna_path_s](#) [vrna_path_t](#)
Typename for the refolding path data structure [vrna_path_s](#).
- typedef struct [vrna_path_options_s](#) * [vrna_path_options_t](#)
Options data structure for (re-)folding path implementations.
- typedef struct [vrna_path_s](#) [path_t](#)
Old typename of [vrna_path_s](#).

Functions

- void [vrna_path_free](#) ([vrna_path_t](#) *path)
Release (free) memory occupied by a (re-)folding path.
- void [vrna_path_options_free](#) ([vrna_path_options_t](#) options)
Release (free) memory occupied by an options data structure for (re-)folding path implementations.
- [vrna_path_options_t](#) [vrna_path_options_findpath](#) (int width, unsigned int type)
Create options data structure for findpath direct (re-)folding path heuristic.
- [vrna_path_t](#) * [vrna_path_direct](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, [vrna_path_options_t](#) options)
Determine an optimal direct (re-)folding path between two secondary structures.
- [vrna_path_t](#) * [vrna_path_direct_ub](#) ([vrna_fold_compound_t](#) *fc, const char *s1, const char *s2, int maxE, [vrna_path_options_t](#) options)
Determine an optimal direct (re-)folding path between two secondary structures.

18.113.1 Detailed Description

API for computing (optimal) (re-)folding paths between secondary structures.

18.114 paths.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PATHS_H
00002 #define VIENNA_RNA_PACKAGE_PATHS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00042 typedef struct vrna_path_s vrna_path_t;
00043
00044
00049 typedef struct vrna_path_options_s *vrna_path_options_t;
00050
00051
00052 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00053
00059 DEPRECATED typedef struct vrna_path_s path_t,
00060               "Use vrna_path_t instead!";
00061
00062 #endif
00063
00064 #include <ViennaRNA/fold_compound.h>
00065 #include <ViennaRNA/landscape/move.h>
00066
00071 #define VRNA_PATH_TYPE_DOT_BRACKET 1U
00072
00077 #define VRNA_PATH_TYPE_MOVES 2U
00078
00108 struct vrna_path_s {
00109     unsigned int type;
00119     double en;
00120     char *s;
00121     vrna_move_t move;
00122 };
00123
00124
00131 void
00132 vrna_path_free(vrna_path_t *path);
00133
00134
00141 void
00142 vrna_path_options_free(vrna_path_options_t options);
00143
00144
00173 vrna_path_options_t
00174 vrna_path_options_findpath(int width,
00175                             unsigned int type);
00176
00177
00203 vrna_path_t *
00204 vrna_path_direct(vrna_fold_compound_t *fc,
00205                  const char *s1,
00206                  const char *s2,
00207                  vrna_path_options_t options);
00208
00209
00232 vrna_path_t *
00233 vrna_path_direct_ub(vrna_fold_compound_t *fc,
00234                     const char *s1,
00235                     const char *s2,
00236                     int maxE,
00237                     vrna_path_options_t options);
00238
00239
00242 #endif
```

18.115 ViennaRNA/Lfold.h File Reference

Functions for locally optimal MFE structure prediction.

Include dependency graph for Lfold.h:

Functions

- float [Lfold](#) (const char *string, const char *structure, int maxdist)
The local analog to [fold\(\)](#).
- float [Lfoldz](#) (const char *string, const char *structure, int maxdist, int zsc, double min_z)

18.115.1 Detailed Description

Functions for locally optimal MFE structure prediction.

18.116 Lfold.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_LFOLD_H
00002 #define VIENNA_RNA_PACKAGE_LFOLD_H
00003
00004 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00005
00012 #ifdef VRNA_WARN_DEPRECATED
00013 # if defined(__clang__)
00014 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00015 # elif defined(__GNUC__)
00016 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00017 # else
00018 #  define DEPRECATED(func, msg) func
00019 # endif
00020 #else
00021 # define DEPRECATED(func, msg) func
00022 #endif
00023
00024 #include <ViennaRNA/mfe_window.h>
00025
00036 DEPRECATED(float Lfold(const char *string,
00037                        const char *structure,
00038                        int maxdist),
00039            "Use vrna_Lfold() or vrna_Lfold_cb() instead");
00040
00041 #ifdef VRNA_WITH_SVM
00049 DEPRECATED(float Lfoldz(const char *string,
00050                        const char *structure,
00051                        int maxdist,
00052                        int zsc,
00053                        double min_z),
00054            "Use vrna_Lfoldz() or vrna_Lfoldz_cb() instead");
00055 #endif
00056
00062 DEPRECATED(float aliLfold(const char **AS,
00063                        const char *structure,
00064                        int maxdist),
00065            "Use vrna_aliLfold() or vrna_aliLfold_cb() instead");
00066
00067
00074 DEPRECATED(float aliLfold_cb(const char **AS,
00075                        int maxdist,
00076                        vrna_mfe_window_f cb,
00077                        void *data),
00078            "Use vrna_aliLfold() or vrna_aliLfold_cb() instead");
00079
00080
00081 #endif
00082
00083 #endif
```

18.117 ViennaRNA/loop_energies.h File Reference

Use [ViennaRNA/loops/all.h](#) instead.

Include dependency graph for loop_energies.h:

18.117.1 Detailed Description

Use [ViennaRNA/loops/all.h](#) instead.

Deprecated Use [ViennaRNA/loops/all.h](#) instead

18.118 loop_energies.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_ALL_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_ALL_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/loop_energies.h>! Use <ViennaRNA/loops/all.h>
         instead!"
00013 # endif
00014 #include <ViennaRNA/loops/all.h>
00015 #endif
00016
00017 #endif
```

18.119 ViennaRNA/loops/all.h File Reference

Energy evaluation for MFE and partition function calculations.

Include dependency graph for all.h: This graph shows which files directly or indirectly include this file:

18.119.1 Detailed Description

Energy evaluation for MFE and partition function calculations.

This file contains functions for the calculation of the free energy ΔG of a hairpin- [[E_Hairpin\(\)](#)] or interior-loop [[E_IntLoop\(\)](#)].

The unit of the free energy returned is $10^{-2} * \text{kcal/mol}$

In case of computing the partition function, this file also supplies functions which return the Boltzmann weights $e^{-\Delta G/kT}$ for a hairpin- [[exp_E_Hairpin\(\)](#)] or interior-loop [[exp_E_IntLoop\(\)](#)].

18.120 all.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_ALL_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_ALL_H
00003
00027 /* below we include the loop type specific energy evaluation functions */
00028
00029 #include <ViennaRNA/loops/external.h>
00030
00031 #include <ViennaRNA/loops/hairpin.h>
00032
00033 #include <ViennaRNA/loops/internal.h>
00034
00035 #include <ViennaRNA/loops/multibranch.h>
00036
00041 #endif
```

18.121 ViennaRNA/loops/external.h File Reference

Energy evaluation of exterior loops for MFE and partition function calculations.

Include dependency graph for external.h: This graph shows which files directly or indirectly include this file:

Functions

- int [E_Stem](#) (int type, int si1, int sj1, int extLoop, [vrna_param_t](#) *P)
Compute the energy contribution of a stem branching off a loop-region.
- [FLT_OR_DBL exp_E_ExtLoop](#) (int type, int si1, int sj1, [vrna_exp_param_t](#) *P)
- [FLT_OR_DBL exp_E_Stem](#) (int type, int si1, int sj1, int extLoop, [vrna_exp_param_t](#) *P)

Basic free energy interface

- int [vrna_E_ext_stem](#) (unsigned int type, int n5d, int n3d, [vrna_param_t](#) *p)
Evaluate a stem branching off the exterior loop.

- int `vrna_eval_ext_stem` (`vrna_fold_compound_t` *fc, int i, int j)
Evaluate the free energy of a base pair in the exterior loop.
- int `vrna_E_ext_loop_5` (`vrna_fold_compound_t` *fc)
- int `vrna_E_ext_loop_3` (`vrna_fold_compound_t` *fc, int i)

Boltzmann weight (partition function) interface

- typedef struct `vrna_mx_pf_aux_el_s` * `vrna_mx_pf_aux_el_t`
Auxiliary helper arrays for fast exterior loop computations.
- `FLT_OR_DBL vrna_exp_E_ext_stem` (unsigned int type, int n5d, int n3d, `vrna_exp_param_t` *p)
Evaluate a stem branching off the exterior loop (Boltzmann factor version)
- `vrna_mx_pf_aux_el_t vrna_exp_E_ext_fast_init` (`vrna_fold_compound_t` *fc)
- void `vrna_exp_E_ext_fast_rotate` (`vrna_mx_pf_aux_el_t` aux_mx)
- void `vrna_exp_E_ext_fast_free` (`vrna_mx_pf_aux_el_t` aux_mx)
- `FLT_OR_DBL vrna_exp_E_ext_fast` (`vrna_fold_compound_t` *fc, int i, int j, `vrna_mx_pf_aux_el_t` aux_mx)
- void `vrna_exp_E_ext_fast_update` (`vrna_fold_compound_t` *fc, int j, `vrna_mx_pf_aux_el_t` aux_mx)

18.121.1 Detailed Description

Energy evaluation of exterior loops for MFE and partition function calculations.

, ,

18.122 external.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_EXTERNAL_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/fold_compound.h>
00006 #include <ViennaRNA/params/basic.h>
00007
00008 #ifdef VRNA_WARN_DEPRECATED
00009 # if defined(DEPRECATED)
00010 #  undef DEPRECATED
00011 # endif
00012 # if defined(__clang__)
00013 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00014 # elif defined(__GNUC__)
00015 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00016 # else
00017 #  define DEPRECATED(func, msg) func
00018 # endif
00019 #else
00020 # define DEPRECATED(func, msg) func
00021 #endif
00022
00059 int
00060 vrna_E_ext_stem(unsigned int type,
00061                int n5d,
00062                int n3d,
00063                vrna_param_t *p);
00064
00065
00085 int
00086 vrna_eval_ext_stem(vrna_fold_compound_t *fc,
00087                   int i,
00088                   int j);
00089
00090
00091 int
00092 vrna_E_ext_loop_5(vrna_fold_compound_t *fc);
00093
00094
00095 int
00096 vrna_E_ext_loop_3(vrna_fold_compound_t *fc,
00097                  int i);
00098
00099
00100 /* End basic interface */
00116 typedef struct vrna_mx_pf_aux_el_s *vrna_mx_pf_aux_el_t;
00117
00118
```

```

00136 FLT_OR_DBL
00137 vrna_exp_E_ext_stem(unsigned int      type,
00138                   int                n5d,
00139                   int                n3d,
00140                   vrna_exp_param_t  *p);
00141
00142
00143 vrna_mx_pf_aux_el_t
00144 vrna_exp_E_ext_fast_init(vrna_fold_compound_t *fc);
00145
00146
00147 void
00148 vrna_exp_E_ext_fast_rotate(vrna_mx_pf_aux_el_t aux_mx);
00149
00150
00151 void
00152 vrna_exp_E_ext_fast_free(vrna_mx_pf_aux_el_t aux_mx);
00153
00154
00155 FLT_OR_DBL
00156 vrna_exp_E_ext_fast(vrna_fold_compound_t *fc,
00157                   int                i,
00158                   int                j,
00159                   vrna_mx_pf_aux_el_t aux_mx);
00160
00161
00162 void
00163 vrna_exp_E_ext_fast_update(vrna_fold_compound_t *fc,
00164                          int                j,
00165                          vrna_mx_pf_aux_el_t aux_mx);
00166
00167
00168 /* End partition function interface */
00182 int
00183 vrna_BT_ext_loop_f5(vrna_fold_compound_t *fc,
00184                   int                *k,
00185                   int                *i,
00186                   int                *j,
00187                   vrna_bp_stack_t    *bp_stack,
00188                   int                *stack_count);
00189
00190
00191 int
00192 vrna_BT_ext_loop_f3(vrna_fold_compound_t *fc,
00193                   int                *k,
00194                   int                maxdist,
00195                   int                *i,
00196                   int                *j,
00197                   vrna_bp_stack_t    *bp_stack,
00198                   int                *stack_count);
00199
00200
00201 int
00202 vrna_BT_ext_loop_f3_pp(vrna_fold_compound_t *fc,
00203                      int                *i,
00204                      int                maxdist);
00205
00206
00211 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00212
00268 DEPRECATED(int E_Stem(int      type,
00269                      int      sil,
00270                      int      sjl,
00271                      int      extLoop,
00272                      vrna_param_t *P),
00273           "This function is obsolete. Use vrna_E_ext_stem() or E_MLstem() instead");
00274
00275
00276 DEPRECATED(int E_ExtLoop(int      type,
00277                      int      sil,
00278                      int      sjl,
00279                      vrna_param_t *P),
00280           "Use vrna_E_ext_stem() instead");
00281
00282
00290 DEPRECATED(FLT_OR_DBL exp_E_ExtLoop(int      type,
00291                      int      sil,
00292                      int      sjl,
00293                      vrna_exp_param_t *P),
00294           "Use vrna_exp_E_ext_stem() instead");
00295
00296
00305 DEPRECATED(FLT_OR_DBL exp_E_Stem(int      type,
00306                      int      sil,
00307                      int      sjl,
00308                      int      extLoop,
00309                      vrna_exp_param_t *P),

```

```

00310         "This function is obsolete");
00311
00312
00313 #endif
00314
00320 #endif

```

18.123 ViennaRNA/loops/hairpin.h File Reference

Energy evaluation of hairpin loops for MFE and partition function calculations.

Include dependency graph for hairpin.h: This graph shows which files directly or indirectly include this file:

Functions

- int [vrna_BT_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int en, [vrna_bp_stack_t](#) *bp_stack, int *stack_↔count)
Backtrack a hairpin loop closed by (i, j).

Basic free energy interface

- int [vrna_E_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate the free energy of a hairpin loop and consider hard constraints if they apply.
- int [vrna_E_ext_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate the free energy of an exterior hairpin loop and consider possible hard constraints.
- int [vrna_eval_ext_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate free energy of an exterior hairpin loop.
- int [vrna_eval_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate free energy of a hairpin loop.
- PRIVATE int [E_Hairpin](#) (int size, int type, int si1, int sj1, const char *string, [vrna_param_t](#) *P)
Compute the Energy of a hairpin-loop.

Boltzmann weight (partition function) interface

- PRIVATE [FLT_OR_DBL exp_E_Hairpin](#) (int u, int type, short si1, short sj1, const char *string, [vrna_exp_param_t](#) *P)
Compute Boltzmann weight $e^{-\Delta G/kT}$ of a hairpin loop.
- [FLT_OR_DBL vrna_exp_E_hp_loop](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
High-Level function for hairpin loop energy evaluation (partition function variant)

18.123.1 Detailed Description

Energy evaluation of hairpin loops for MFE and partition function calculations.

```

, ,

```

18.124 hairpin.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_HAIRPIN_H
00003
00004 #include <math.h>
00005 #include <string.h>
00006 #include <ViennaRNA/utils/basic.h>
00007 #include <ViennaRNA/datastructures/basic.h>
00008 #include <ViennaRNA/fold_compound.h>
00009 #include <ViennaRNA/params/basic.h>
00010 #include <ViennaRNA/params/salt.h>
00011
00012 #ifdef VRNA_WARN_DEPRECATED
00013 # if defined(DEPRECATED)
00014 #   undef DEPRECATED
00015 # endif
00016 # if defined(__clang__)
00017 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00018 # elif defined(__GNUC__)

```

```

00019 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00020 # else
00021 # define DEPRECATED(func, msg) func
00022 # endif
00023 #else
00024 # define DEPRECATED(func, msg) func
00025 #endif
00026
00027 #ifdef __GNUC__
00028 # define INLINE inline
00029 #else
00030 # define INLINE
00031 #endif
00032
00071 int
00072 vrna_E_hp_loop(vrna_fold_compound_t *fc,
00073               int i,
00074               int j);
00075
00076
00085 int
00086 vrna_E_ext_hp_loop(vrna_fold_compound_t *fc,
00087                   int i,
00088                   int j);
00089
00090
00094 int
00095 vrna_eval_ext_hp_loop(vrna_fold_compound_t *fc,
00096                      int i,
00097                      int j);
00098
00099
00111 int
00112 vrna_eval_hp_loop(vrna_fold_compound_t *fc,
00113                  int i,
00114                  int j);
00115
00116
00149 PRIVATE INLINE int
00150 E_Hairpin(int size,
00151           int type,
00152           int sil,
00153           int sjl,
00154           const char *string,
00155           vrna_param_t *P)
00156 {
00157     int energy, salt_correction;
00158
00159     salt_correction = 0;
00160
00161     if (P->model_details.salt != VRNA_MODEL_DEFAULT_SALT) {
00162         if (size <= MAXLOOP)
00163             salt_correction = P->SaltLoop[size+1];
00164         else
00165             salt_correction = vrna_salt_loop_int(size+1, P->model_details.salt, P->temperature+K0);
00166     }
00167
00168     if (size <= 30)
00169         energy = P->hairpin[size];
00170     else
00171         energy = P->hairpin[30] + (int) (P->lxc * log((size) / 30.));
00172
00173     energy += salt_correction;
00174
00175     if (size < 3)
00176         return energy; /* should only be the case when folding alignments */
00177
00178     if ((string) && (P->model_details.special_hp)) {
00179         if (size == 4) {
00180             /* check for tetraloop bonus */
00181             char tl[7] = {
00182                 0
00183             }, *ts;
00184             memcpy(tl, string, sizeof(char) * 6);
00185             tl[6] = '\0';
00186             if ((ts = strstr(P->Tetraloops, tl)))
00187                 return P->Tetraloop_E[(ts - P->Tetraloops) / 7] + salt_correction;
00188         } else if (size == 6) {
00189             char tl[9] = {
00190                 0
00191             }, *ts;
00192             memcpy(tl, string, sizeof(char) * 8);
00193             tl[8] = '\0';
00194             if ((ts = strstr(P->Hexaloops, tl)))
00195                 return P->Hexaloop_E[(ts - P->Hexaloops) / 9] + salt_correction;
00196         } else if (size == 3) {
00197             char tl[6] = {

```



```

00198         0
00199     }, *ts;
00200     memcpy(tl, string, sizeof(char) * 5);
00201     tl[5] = '\0';
00202     if ((ts = strstr(P->Triloops, tl)))
00203         return P->Triloop_E[(ts - P->Triloops) / 6] + salt_correction;
00204
00205     return energy + (type > 2 ? P->TerminalAU : 0);
00206 }
00207 }
00208
00209 energy += P->mismatchH[type][sil][sjl];
00210
00211 return energy;
00212 }
00213
00214
00215 /* End basic interface */
00243 PRIVATE INLINE FLT_OR_DBL
00244 exp_E_Hairpin(int          u,
00245               int          type,
00246               short        sil,
00247               short        sjl,
00248               const char    *string,
00249               vrna_exp_param_t *P)
00250 {
00251     double q, kT, salt_correction;
00252
00253     kT = P->kT; /* kT in cal/mol */
00254     salt_correction = 1.;
00255
00256     if (P->model_details.salt != VRNA_MODEL_DEFAULT_SALT) {
00257         if (u <= MAXLOOP)
00258             salt_correction = P->expSaltLoop[u+1];
00259         else
00260             salt_correction = exp(-vrna_salt_loop_int(u+1, P->model_details.salt, P->temperature+K0) * 10. /
00261 kT);
00262     }
00263
00264     if (u <= 30)
00265         q = P->exphairpin[u];
00266     else
00267         q = P->exphairpin[30] * exp(-(P->lxc * log(u / 30.)) * 10. / kT);
00268
00269     q *= salt_correction;
00270
00271     if (u < 3)
00272         return (FLT_OR_DBL)q; /* should only be the case when folding alignments */
00273
00274     if ((string) && (P->model_details.special_hp)) {
00275         if (u == 4) {
00276             char tl[7] = {
00277                 0
00278             }, *ts;
00279             memcpy(tl, string, sizeof(char) * 6);
00280             tl[6] = '\0';
00281             if ((ts = strstr(P->Tetraloops, tl))) {
00282                 if (type != 7)
00283                     return (FLT_OR_DBL) (P->exptetra[(ts - P->Tetraloops) / 7] * salt_correction);
00284                 else
00285                     q *= P->exptetra[(ts - P->Tetraloops) / 7];
00286             }
00287         } else if (u == 6) {
00288             char tl[9] = {
00289                 0
00290             }, *ts;
00291             memcpy(tl, string, sizeof(char) * 8);
00292             tl[8] = '\0';
00293             if ((ts = strstr(P->Hexaloops, tl)))
00294                 return (FLT_OR_DBL) (P->exphex[(ts - P->Hexaloops) / 9] * salt_correction);
00295         } else if (u == 3) {
00296             char tl[6] = {
00297                 0
00298             }, *ts;
00299             memcpy(tl, string, sizeof(char) * 5);
00300             tl[5] = '\0';
00301             if ((ts = strstr(P->Triloops, tl)))
00302                 return (FLT_OR_DBL) (P->exptri[(ts - P->Triloops) / 6] * salt_correction);
00303         }
00304         if (type > 2)
00305             return (FLT_OR_DBL) (q * P->expTermAU);
00306         else
00307             return (FLT_OR_DBL) q;
00308     }
00309
00310     q *= P->expmismatchH[type][sil][sjl];

```

```

00311
00312     return (FLT_OR_DBL)q;
00313 }
00314
00315 FLT_OR_DBL
00326 vrna_exp_E_hp_loop(vrna_fold_compound_t *fc,
00327                   int i,
00328                   int j);
00329
00330
00331 /* End partition function interface */
00332 int
00333 vrna_BT_hp_loop(vrna_fold_compound_t *fc,
00334                int i,
00335                int j,
00336                int en,
00337                vrna_bp_stack_t *bp_stack,
00338                int *stack_count);
00339
00340
00341 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00342
00343 #endif
00344 #endif
00345 #endif

```

18.125 ViennaRNA/loops/internal.h File Reference

Energy evaluation of interior loops for MFE and partition function calculations.

Include dependency graph for internal.h: This graph shows which files directly or indirectly include this file:

Functions

- int [vrna_BT_stack](#) (vrna_fold_compound_t *fc, int *i, int *j, int *en, vrna_bp_stack_t *bp_stack, int *stack_count)
Backtrack a stacked pair closed by (i, j) .
- int [vrna_BT_int_loop](#) (vrna_fold_compound_t *fc, int *i, int *j, int en, vrna_bp_stack_t *bp_stack, int *stack_count)
Backtrack an interior loop closed by (i, j) .
- PRIVATE int [E_IntLoop](#) (int n1, int n2, int type, int type_2, int si1, int sj1, int sp1, int sq1, vrna_param_t *P)
- PRIVATE FLT_OR_DBL [exp_E_IntLoop](#) (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1, vrna_exp_param_t *P)

Basic free energy interface

- int [vrna_E_int_loop](#) (vrna_fold_compound_t *fc, int i, int j)
- int [vrna_eval_int_loop](#) (vrna_fold_compound_t *fc, int i, int j, int k, int l)
Evaluate the free energy contribution of an interior loop with delimiting base pairs (i, j) and (k, l) .
- int [vrna_E_ext_int_loop](#) (vrna_fold_compound_t *fc, int i, int j, int *ip, int *iq)
- int [vrna_E_stack](#) (vrna_fold_compound_t *fc, int i, int j)

Boltzmann weight (partition function) interface

- FLT_OR_DBL [vrna_exp_E_int_loop](#) (vrna_fold_compound_t *fc, int i, int j)
- FLT_OR_DBL [vrna_exp_E_interior_loop](#) (vrna_fold_compound_t *fc, int i, int j, int k, int l)

18.125.1 Detailed Description

Energy evaluation of interior loops for MFE and partition function calculations.

, ,

18.126 internal.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_INTERNAL_H
00003
00004 #include <math.h>
00005
00006 #include <ViennaRNA/utils/basic.h>
00007 #include <ViennaRNA/params/default.h>
00008 #include <ViennaRNA/datastructures/basic.h>
00009 #include <ViennaRNA/fold_compound.h>
00010 #include <ViennaRNA/params/basic.h>
00011 #include <ViennaRNA/constraints/hard.h>
00012 #include <ViennaRNA/constraints/soft.h>
00013 #include <ViennaRNA/params/salt.h>
00014
00015 #ifdef VRNA_WARN_DEPRECATED
00016 # if defined(DEPRECATED)
00017 #   undef DEPRECATED
00018 # endif
00019 # if defined(__clang__)
00020 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00021 # elif defined(__GNUC__)
00022 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00023 # else
00024 #   define DEPRECATED(func, msg) func
00025 # endif
00026 #else
00027 # define DEPRECATED(func, msg) func
00028 #endif
00029
00030 #ifdef __GNUC__
00031 # define INLINE inline
00032 #else
00033 # define INLINE
00034 #endif
00035
00053 int
00054 vrna_E_int_loop(vrna_fold_compound_t *fc,
00055                int i,
00056                int j);
00057
00058
00066 int
00067 vrna_eval_int_loop(vrna_fold_compound_t *fc,
00068                  int i,
00069                  int j,
00070                  int k,
00071                  int l);
00072
00073
00074 int
00075 vrna_E_ext_int_loop(vrna_fold_compound_t *fc,
00076                   int i,
00077                   int j,
00078                   int *ip,
00079                   int *iq);
00080
00081
00082 int
00083 vrna_E_stack(vrna_fold_compound_t *fc,
00084             int i,
00085             int j);
00086
00087
00088 /* End basic interface */
00089 /* j < i indicates circular folding, i.e. collect contributions for exterior int loops */
00090 FLT_OR_DBL
00100 vrna_exp_E_int_loop(vrna_fold_compound_t *fc,
00101                   int i,
00102                   int j);
00103
00104
00105 FLT_OR_DBL
00106 vrna_exp_E_interior_loop(vrna_fold_compound_t *fc,
00107                       int i,
00108                       int j,
00109                       int k,
00110                       int l);
00111
00112
00113 /* End partition function interface */
00131 int
00132 vrna_BT_stack(vrna_fold_compound_t *fc,
00133              int *i,

```

```

00134         int                *j,
00135         int                *en,
00136         vrna_bp_stack_t    *bp_stack,
00137         int                *stack_count);
00138
00139
00144 int
00145 vrna_BT_int_loop(vrna_fold_compound_t *fc,
00146                 int                *i,
00147                 int                *j,
00148                 int                en,
00149                 vrna_bp_stack_t    *bp_stack,
00150                 int                *stack_count);
00151
00152
00158 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00165 #ifdef ON_SAME_STRAND
00166 #undef ON_SAME_STRAND
00167 #endif
00168
00169 #define ON_SAME_STRAND(I, J, C)  (((I) >= (C)) || ((J) < (C)))
00170
00215 PRIVATE INLINE int E_IntLoop(int          n1,
00216                             int          n2,
00217                             int          type,
00218                             int          type_2,
00219                             int          sil,
00220                             int          sj1,
00221                             int          spl,
00222                             int          sql,
00223                             vrna_param_t *P);
00224
00225
00245 PRIVATE INLINE FLT_OR_DBL exp_E_IntLoop(int          u1,
00246                                         int          u2,
00247                                         int          type,
00248                                         int          type2,
00249                                         short        sil,
00250                                         short        sj1,
00251                                         short        spl,
00252                                         short        sql,
00253                                         vrna_exp_param_t *P);
00254
00255
00256 PRIVATE INLINE int E_IntLoop_Co(int          type,
00257                                int          type_2,
00258                                int          i,
00259                                int          j,
00260                                int          p,
00261                                int          q,
00262                                int          cutpoint,
00263                                short        sil,
00264                                short        sj1,
00265                                short        spl,
00266                                short        sql,
00267                                int          dangles,
00268                                vrna_param_t *P);
00269
00270
00271 /*
00272  * ugly but fast interior loop evaluation
00273  *
00274  * Avoid including this function in your own code. It only serves
00275  * as a fast inline block internally re-used throughout the RNAlib. It
00276  * evaluates the free energy of interior loops in single sequences or sequence
00277  * hybrids. Soft constraints are also applied if available.
00278  *
00279  * NOTE: do not include into doxygen reference manual!
00280  */
00281 PRIVATE INLINE int
00282 ubf_eval_int_loop(int          i,
00283                  int          j,
00284                  int          p,
00285                  int          q,
00286                  int          il,
00287                  int          jl,
00288                  int          pl,
00289                  int          ql,
00290                  short        si,
00291                  short        sj,
00292                  short        sp,
00293                  short        sq,
00294                  unsigned char type,
00295                  unsigned char type_2,
00296                  int          *rtype,
00297                  int          ij,
00298                  int          cp,

```

```

00299         vrna_param_t *P,
00300         vrna_sc_t *sc)
00301 {
00302     int energy, u1, u2;
00303
00304     u1 = p1 - i;
00305     u2 = j1 - q;
00306
00307     if ((cp < 0) || (ON_SAME_STRAND(i, p, cp) && ON_SAME_STRAND(q, j, cp))) {
00308         /* regular interior loop */
00309         energy = E_IntLoop(u1, u2, type, type_2, si, sj, sp, sq, P);
00310     } else {
00311         /* interior loop like cofold structure */
00312         short Si, Sj;
00313         Si = ON_SAME_STRAND(i, i1, cp) ? si : -1;
00314         Sj = ON_SAME_STRAND(j1, j, cp) ? sj : -1;
00315         energy = E_IntLoop_Co(rtype[type], rtype[type_2],
00316                               i, j, p, q,
00317                               cp,
00318                               Si, Sj,
00319                               sp, sq,
00320                               P->model_details.dangles,
00321                               P);
00322     }
00323
00324     /* add soft constraints */
00325     if (sc) {
00326         if (sc->energy_up)
00327             energy += sc->energy_up[i1][u1]
00328                     + sc->energy_up[q1][u2];
00329
00330         if (sc->energy_bp)
00331             energy += sc->energy_bp[ij];
00332
00333         if (sc->energy_stack)
00334             if (u1 + u2 == 0) {
00335                 int a = sc->energy_stack[i]
00336                     + sc->energy_stack[p]
00337                     + sc->energy_stack[q]
00338                     + sc->energy_stack[j];
00339                 energy += a;
00340             }
00341
00342         if (sc->f)
00343             energy += sc->f(i, j, p, q, VRNA_DECOMP_PAIR_IL, sc->data);
00344     }
00345
00346     return energy;
00347 }
00348
00349
00350 PRIVATE INLINE int
00351 ubf_eval_int_loop2(int i,
00352                    int j,
00353                    int p,
00354                    int q,
00355                    int i1,
00356                    int j1,
00357                    int p1,
00358                    int q1,
00359                    short si,
00360                    short sj,
00361                    short sp,
00362                    short sq,
00363                    unsigned char type,
00364                    unsigned char type_2,
00365                    int *rtype,
00366                    int ij,
00367                    unsigned int *sn,
00368                    unsigned int *ss,
00369                    vrna_param_t *P,
00370                    vrna_sc_t *sc)
00371 {
00372     int energy, u1, u2;
00373
00374     u1 = p1 - i;
00375     u2 = j1 - q;
00376
00377     if ((sn[i] == sn[p]) && (sn[q] == sn[j])) {
00378         /* regular interior loop */
00379         energy = E_IntLoop(u1, u2, type, type_2, si, sj, sp, sq, P);
00380     } else {
00381         /* interior loop like cofold structure */
00382         short Si, Sj;
00383         Si = (sn[i1] == sn[i]) ? si : -1;
00384         Sj = (sn[j] == sn[j1]) ? sj : -1;
00385         energy = E_IntLoop_Co(rtype[type], rtype[type_2],

```

```

00386             i, j, p, q,
00387             ss[l],
00388             Si, Sj,
00389             sp, sq,
00390             P->model_details.dangles,
00391             P);
00392     }
00393
00394     /* add soft constraints */
00395     if (sc) {
00396         if (sc->energy_up)
00397             energy += sc->energy_up[i1][u1]
00398                 + sc->energy_up[q1][u2];
00399
00400         if (sc->energy_bp)
00401             energy += sc->energy_bp[ij];
00402
00403         if (sc->energy_stack)
00404             if (u1 + u2 == 0) {
00405                 int a = sc->energy_stack[i]
00406                     + sc->energy_stack[p]
00407                     + sc->energy_stack[q]
00408                     + sc->energy_stack[j];
00409                 energy += a;
00410             }
00411
00412         if (sc->f)
00413             energy += sc->f(i, j, p, q, VRNA_DECOMP_PAIR_IL, sc->data);
00414     }
00415
00416     return energy;
00417 }
00418
00419
00420 /*
00421  * ugly but fast exterior interior loop evaluation
00422  *
00423  * Avoid including this function in your own code. It only serves
00424  * as a fast inline block internally re-used throughout the RNALib. It
00425  * evaluates the free energy of interior loops in single sequences or sequence
00426  * hybrids. Soft constraints are also applied if available.
00427  *
00428  * NOTE: do not include into doxygen reference manual!
00429  */
00430 PRIVATE INLINE int
00431 ubf_eval_ext_int_loop(int i,
00432                       int j,
00433                       int p,
00434                       int q,
00435                       int i1,
00436                       int j1,
00437                       int p1,
00438                       int q1,
00439                       short si,
00440                       short sj,
00441                       short sp,
00442                       short sq,
00443                       unsigned char type,
00444                       unsigned char type_2,
00445                       int length,
00446                       vrna_param_t *P,
00447                       vrna_sc_t *sc)
00448 {
00449     int energy, u1, u2, u3;
00450
00451     u1 = i1;
00452     u2 = p1 - j;
00453     u3 = length - q;
00454
00455     energy = E_IntLoop(u2, u1 + u3, type, type_2, si, sj, sp, sq, P);
00456
00457     /* add soft constraints */
00458     if (sc) {
00459         if (sc->energy_up) {
00460             energy += sc->energy_up[j1][u2]
00461                 + ((u3 > 0) ? sc->energy_up[q1][u3] : 0)
00462                 + ((u1 > 0) ? sc->energy_up[i1][u1] : 0);
00463         }
00464
00465         if (sc->energy_stack)
00466             if (u1 + u2 + u3 == 0)
00467                 energy += sc->energy_stack[i]
00468                     + sc->energy_stack[p]
00469                     + sc->energy_stack[q]
00470                     + sc->energy_stack[j];
00471
00472         if (sc->f)

```

```

00473     energy += sc->f(i, j, p, q, VRNA_DECOMP_PAIR_IL, sc->data);
00474 }
00475
00476 return energy;
00477 }
00478
00479
00480 PRIVATE INLINE int
00481 E_IntLoop(int      n1,
00482           int      n2,
00483           int      type,
00484           int      type_2,
00485           int      si1,
00486           int      sj1,
00487           int      sp1,
00488           int      sq1,
00489           vrna_param_t *P)
00490 {
00491     /* compute energy of degree 2 loop (stack bulge or interior) */
00492     int n1, ns, u, energy, salt_stack_correction, salt_loop_correction, backbones;
00493
00494     salt_stack_correction = P->SaltStack;
00495     salt_loop_correction = 0;
00496
00497     if (n1 > n2) {
00498         n1 = n1;
00499         ns = n2;
00500     } else {
00501         n1 = n2;
00502         ns = n1;
00503     }
00504
00505     if (n1 == 0) {
00506         return P->stack[type][type_2] + salt_stack_correction; /* stack */
00507     }
00508
00509     backbones = n1+ns+2;
00510
00511     if (P->model_details.salt != VRNA_MODEL_DEFAULT_SALT) {
00512         /* salt correction for loop */
00513         if (backbones <= MAXLOOP+1)
00514             salt_loop_correction = P->SaltLoop[backbones];
00515         else
00516             salt_loop_correction = vrna_salt_loop_int(backbones, P->model_details.salt, P->temperature+K0);
00517     }
00518
00519     if (ns == 0) {
00520         /* bulge */
00521         energy = (n1 <= MAXLOOP) ? P->bulge[n1] :
00522             (P->bulge[30] + (int)(P->lxc * log(n1 / 30.)));
00523         if (n1 == 1) {
00524             energy += P->stack[type][type_2];
00525         } else {
00526             if (type > 2)
00527                 energy += P->TerminalAU;
00528
00529             if (type_2 > 2)
00530                 energy += P->TerminalAU;
00531         }
00532
00533         return energy + salt_loop_correction;
00534     } else {
00535         /* interior loop */
00536         if (ns == 1) {
00537             if (n1 == 1) /* 1x1 loop */
00538                 return P->int11[type][type_2][si1][sj1] + salt_loop_correction;
00539
00540             if (n1 == 2) {
00541                 /* 2x1 loop */
00542                 if (n1 == 1)
00543                     energy = P->int21[type][type_2][si1][sq1][sj1];
00544                 else
00545                     energy = P->int21[type_2][type][sq1][si1][sp1];
00546
00547                 return energy + salt_loop_correction;
00548             } else {
00549                 /* 1xn loop */
00550                 energy =
00551                     (n1 + 1 <=
00552                     MAXLOOP) ? (P->internal_loop[n1 + 1]) : (P->internal_loop[30] +
00553                     (int)(P->lxc * log((n1 + 1) / 30.)));
00554                 energy += MIN2(MAX_NINIO, (n1 - ns) * P->ninio[2]);
00555                 energy += P->mismatchlnI[type][si1][sj1] + P->mismatchlnI[type_2][sq1][sp1];
00556                 return energy + salt_loop_correction;
00557             }
00558         } else if (ns == 2) {
00559             if (n1 == 2) {

```

```

00560         /* 2x2 loop */
00561         return P->int22[type][type_2][sil][spl][sql][sjl] + salt_loop_correction;
00562     } else if (nl == 3) {
00563         /* 2x3 loop */
00564         energy = P->internal_loop[5] + P->ninio[2];
00565         energy += P->mismatch23I[type][sil][sjl] + P->mismatch23I[type_2][sql][spl];
00566         return energy + salt_loop_correction;
00567     }
00568 }
00569
00570 {
00571     /* generic interior loop (no else here!)*
00572     u = nl + ns;
00573     energy =
00574         (u <=
00575          MAXLOOP) ? (P->internal_loop[u]) : (P->internal_loop[30] + (int)(P->lxc * log((u) / 30.)));
00576
00577     energy += MIN2(MAX_NINIO, (nl - ns) * P->ninio[2]);
00578
00579     energy += P->mismatchI[type][sil][sjl] + P->mismatchI[type_2][sql][spl];
00580 }
00581 }
00582
00583 return energy + salt_loop_correction;
00584 }
00585
00586
00587 PRIVATE INLINE FLT_OR_DBL
00588 exp_E_IntLoop(int u1,
00589               int u2,
00590               int type,
00591               int type2,
00592               short sil,
00593               short sjl,
00594               short spl,
00595               short sql,
00596               vrna_exp_param_t *P)
00597 {
00598     int ul, us, no_close = 0;
00599     double z = 0.;
00600     int noGUclosure = P->model_details.noGUclosure;
00601     int backbones;
00602     double salt_stack_correction = P->expSaltStack;
00603     double salt_loop_correction = 1.;
00604
00605     if ((noGUclosure) && ((type2 == 3) || (type2 == 4) || (type == 3) || (type == 4)))
00606         no_close = 1;
00607
00608     if (u1 > u2) {
00609         ul = u1;
00610         us = u2;
00611     } else {
00612         ul = u2;
00613         us = u1;
00614     }
00615
00616     /* salt correction for loop */
00617     backbones = ul+us+2;
00618
00619     if (P->model_details.salt != VRNA_MODEL_DEFAULT_SALT) {
00620         if (backbones <= MAXLOOP+1)
00621             salt_loop_correction = P->expSaltLoop[backbones];
00622         else
00623             salt_loop_correction = exp(-vrna_salt_loop_int(backbones, P->model_details.salt,
00624 P->temperature+K0) * 10. / P->kT);
00625     }
00626
00627     if (ul == 0) {
00628         /* stack */
00629         z = P->expstack[type][type2] * salt_stack_correction;
00630     } else if (!no_close) {
00631         if (us == 0) {
00632             /* bulge */
00633             z = P->expbulge[ul];
00634             if (ul == 1) {
00635                 z *= P->expstack[type][type2];
00636             } else {
00637                 if (type > 2)
00638                     z *= P->expTermAU;
00639
00640                 if (type2 > 2)
00641                     z *= P->expTermAU;
00642             }
00643         }
00644         return (FLT_OR_DBL)(z * salt_loop_correction);
00645     } else if (us == 1) {
00646         if (ul == 1)
00647             /* 1x1 loop */

```



```

00646         return (FLT_OR_DBL) (P->expint11[type][type2][sil][sj1] * salt_loop_correction);
00647
00648     if (ul == 2) {
00649         /* 2x1 loop */
00650         if (ul == 1)
00651             return (FLT_OR_DBL) (P->expint21[type][type2][sil][sq1][sj1] * salt_loop_correction);
00652         else
00653             return (FLT_OR_DBL) (P->expint21[type2][type][sq1][sil][sp1] * salt_loop_correction);
00654     } else {
00655         /* 1xn loop */
00656         z = P->expinternal[ul + us] * P->expmismatch1nI[type][sil][sj1] *
00657             P->expmismatch1nI[type2][sq1][sp1];
00658         return (FLT_OR_DBL) (z * P->expninio[2][ul - us] * salt_loop_correction);
00659     }
00660 } else if (us == 2) {
00661     if (ul == 2) {
00662         /* 2x2 loop */
00663         return (FLT_OR_DBL) (P->expint22[type][type2][sil][sp1][sq1][sj1] * salt_loop_correction);
00664     } else if (ul == 3) {
00665         /* 2x3 loop */
00666         z = P->expinternal[5] * P->expmismatch23I[type][sil][sj1] *
00667             P->expmismatch23I[type2][sq1][sp1];
00668         return (FLT_OR_DBL) (z * P->expninio[2][1] * salt_loop_correction);
00669     }
00670 }
00671
00672 /* generic interior loop (no else here!) */
00673 z = P->expinternal[ul + us] * P->expmismatchI[type][sil][sj1] *
00674     P->expmismatchI[type2][sq1][sp1];
00675 return (FLT_OR_DBL) (z * P->expninio[2][ul - us] * salt_loop_correction);
00676 }
00677
00678 return (FLT_OR_DBL) z;
00679 }
00680
00681
00682 PRIVATE INLINE int
00683 E_IntLoop_Co(int          type,
00684              int          type_2,
00685              int          i,
00686              int          j,
00687              int          p,
00688              int          q,
00689              int          cutpoint,
00690              short        sil,
00691              short        sj1,
00692              short        spl,
00693              short        sq1,
00694              int          dangles,
00695              vrna_param_t *P)
00696 {
00697     int e, energy, ci, cj, cp, cq, d3, d5, d5_2, d3_2, tmm, tmm_2;
00698     int salt_loop_correction, backbones;
00699
00700     salt_loop_correction = 0;
00701
00702     backbones = p - i + j - q;
00703     /* salt correction for loop */
00704     if (P->model_details.salt != VRNA_MODEL_DEFAULT_SALT) {
00705         if (backbones <= MAXLOOP+1)
00706             salt_loop_correction = P->SaltLoop[backbones];
00707         else
00708             salt_loop_correction = vrna_salt_loop_int(backbones, P->model_details.salt, P->temperature+K0);
00709     }
00710
00711     energy = 0;
00712     if (type > 2)
00713         energy += P->TerminalAU;
00714
00715     if (type_2 > 2)
00716         energy += P->TerminalAU;
00717
00718     if (!dangles)
00719         return energy + salt_loop_correction;
00720
00721     ci = ON_SAME_STRAND(i, i + 1, cutpoint);
00722     cj = ON_SAME_STRAND(j - 1, j, cutpoint);
00723     cp = ON_SAME_STRAND(p - 1, p, cutpoint);
00724     cq = ON_SAME_STRAND(q, q + 1, cutpoint);
00725
00726     d3 = ci ? P->dangle3[type][sil] : 0;
00727     d5 = cj ? P->dangle5[type][sj1] : 0;
00728     d5_2 = cp ? P->dangle5[type_2][spl] : 0;
00729     d3_2 = cq ? P->dangle3[type_2][sq1] : 0;
00730
00731     tmm = (cj && ci) ? P->mismatchExt[type][sj1][sil] : d5 + d3;
00732     tmm_2 = (cp && cq) ? P->mismatchExt[type_2][spl][sq1] : d5_2 + d3_2;

```

```

00733
00734     if (dangles == 2)
00735         return energy + tmm + tmm_2 + salt_loop_correction;
00736
00737     /* now we may have non-double dangles only */
00738     if (p - i > 2) {
00739         if (j - q > 2) {
00740             /* all degrees of freedom */
00741             e = MIN2(tmm, d5);
00742             e = MIN2(e, d3);
00743             energy += e;
00744             e = MIN2(tmm_2, d5_2);
00745             e = MIN2(e, d3_2);
00746             energy += e;
00747         } else if (j - q == 2) {
00748             /* all degrees of freedom in 5' part between i and p */
00749             e = MIN2(tmm + d5_2, d3 + d5_2);
00750             e = MIN2(e, d5 + d5_2);
00751             e = MIN2(e, d3 + tmm_2);
00752             e = MIN2(e, d3 + d3_2);
00753             e = MIN2(e, tmm_2); /* no dangles on enclosing pair */
00754             e = MIN2(e, d5_2); /* no dangles on enclosing pair */
00755             e = MIN2(e, d3_2); /* no dangles on enclosing pair */
00756             energy += e;
00757         } else {
00758             /* no unpaired base between q and j */
00759             energy += d3 + d5_2;
00760         }
00761     } else if (p - i == 2) {
00762         if (j - q > 2) {
00763             /* all degrees of freedom in 3' part between q and j */
00764             e = MIN2(tmm + d3_2, d5 + d3_2);
00765             e = MIN2(e, d5 + d3_2);
00766             e = MIN2(e, d3 + d3_2);
00767             e = MIN2(e, d5 + tmm_2);
00768             e = MIN2(e, tmm_2);
00769             e = MIN2(e, d5_2);
00770             e = MIN2(e, d3_2);
00771             energy += e;
00772         } else if (j - q == 2) {
00773             /* one possible dangling base between either side */
00774             e = MIN2(tmm, tmm_2);
00775             e = MIN2(e, d3);
00776             e = MIN2(e, d5);
00777             e = MIN2(e, d5_2);
00778             e = MIN2(e, d3_2);
00779             e = MIN2(e, d3 + d3_2);
00780             e = MIN2(e, d5 + d5_2);
00781             energy += e;
00782         } else {
00783             /* one unpaired base between i and p */
00784             energy += MIN2(d3, d5_2);
00785         }
00786     } else {
00787         /* no unpaired base between i and p */
00788         if (j - q > 2) {
00789             /* all degrees of freedom in 3' part between q and j */
00790             energy += d5 + d3_2;
00791         } else if (j - q == 2) {
00792             /* one unpaired base between q and j */
00793             energy += MIN2(d5, d3_2);
00794         }
00795     }
00796
00797     return energy + salt_loop_correction;
00798 }
00799
00800
00805 #endif
00806
00807 #endif

```

18.127 ViennaRNA/loops/multibranch.h File Reference

Energy evaluation of multibranch loops for MFE and partition function calculations.

Include dependency graph for multibranch.h: This graph shows which files directly or indirectly include this file:

Functions

- int [vrna_BT_mb_loop](#) ([vrna_fold_compound_t](#) *fc, int *i, int *j, int *k, int en, int *component1, int *component2)

Backtrack the decomposition of a multi branch loop closed by (i, j) .

Basic free energy interface

- int [vrna_E_mb_loop_stack](#) ([vrna_fold_compound_t](#) *fc, int i, int j)
Evaluate energy of a multi branch helices stacking onto closing pair (i, j)
- int [vrna_E_mb_loop_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int *dmli1, int *dmli2)
- int [E_ml_rightmost_stem](#) (int i, int j, [vrna_fold_compound_t](#) *fc)
- int [vrna_E_ml_stems_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, int *fmi, int *dmli)

Boltzmann weight (partition function) interface

- typedef struct [vrna_mx_pf_aux_ml_s](#) * [vrna_mx_pf_aux_ml_t](#)
Auxiliary helper arrays for fast exterior loop computations.
- [FLT_OR_DBL vrna_exp_E_mb_loop_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, [vrna_mx_pf_aux_ml_t](#) aux_mx)
- [vrna_mx_pf_aux_ml_t vrna_exp_E_ml_fast_init](#) ([vrna_fold_compound_t](#) *fc)
- void [vrna_exp_E_ml_fast_rotate](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- void [vrna_exp_E_ml_fast_free](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- const [FLT_OR_DBL](#) * [vrna_exp_E_ml_fast_qqm](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- const [FLT_OR_DBL](#) * [vrna_exp_E_ml_fast_qqm1](#) ([vrna_mx_pf_aux_ml_t](#) aux_mx)
- [FLT_OR_DBL vrna_exp_E_ml_fast](#) ([vrna_fold_compound_t](#) *fc, int i, int j, [vrna_mx_pf_aux_ml_t](#) aux_mx)

18.127.1 Detailed Description

Energy evaluation of multibranch loops for MFE and partition function calculations.

, ,

18.128 multibranch.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_H
00003
00004 #include <ViennaRNA/utils/basic.h>
00005 #include <ViennaRNA/datastructures/basic.h>
00006 #include <ViennaRNA/fold_compound.h>
00007 #include <ViennaRNA/params/basic.h>
00008
00009 #ifdef VRNA_WARN_DEPRECATED
00010 # if defined(DEPRECATED)
00011 #  undef DEPRECATED
00012 # endif
00013 # if defined(__clang__)
00014 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00015 # elif defined(__GNUC__)
00016 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00017 # else
00018 #  define DEPRECATED(func, msg) func
00019 # endif
00020 #else
00021 # define DEPRECATED(func, msg) func
00022 #endif
00023
00024 #ifdef __GNUC__
00025 # define INLINE inline
00026 #else
00027 # define INLINE
00028 #endif
00029
00054 int
00055 vrna\_E\_mb\_loop\_stack(vrna\_fold\_compound\_t *fc,
00056                     int
00057                     int
00058                     j);
00059
00060 int
00061 vrna\_E\_mb\_loop\_fast(vrna\_fold\_compound\_t *fc,
00062                     int
00063                     int
00064                     int
00065                     *dmli1,
```

```

00065             int                *dmli2);
00066
00067
00068 int
00069 E_ml_rightmost_stem(int                i,
00070                     int                j,
00071                     vrna_fold_compound_t *fc);
00072
00073
00074 int
00075 vrna_E_ml_stems_fast(vrna_fold_compound_t *fc,
00076                     int                i,
00077                     int                j,
00078                     int                *fmi,
00079                     int                *dmli);
00080
00081
00082 /* End basic interface */
00083 typedef struct vrna_mx_pf_aux_ml_s *vrna_mx_pf_aux_ml_t;
00084
00085
00086 FLT_OR_DBL
00087 vrna_exp_E_ml_loop_fast(vrna_fold_compound_t *fc,
00088                         int                i,
00089                         int                j,
00090                         vrna_mx_pf_aux_ml_t aux_mx);
00091
00092
00093 vrna_mx_pf_aux_ml_t
00094 vrna_exp_E_ml_fast_init(vrna_fold_compound_t *fc);
00095
00096
00097 void
00098 vrna_exp_E_ml_fast_rotate(vrna_mx_pf_aux_ml_t aux_mx);
00099
00100
00101 void
00102 vrna_exp_E_ml_fast_free(vrna_mx_pf_aux_ml_t aux_mx);
00103
00104
00105 const FLT_OR_DBL *
00106 vrna_exp_E_ml_fast_qqm(vrna_mx_pf_aux_ml_t aux_mx);
00107
00108
00109 const FLT_OR_DBL *
00110 vrna_exp_E_ml_fast_qqm1(vrna_mx_pf_aux_ml_t aux_mx);
00111
00112
00113 FLT_OR_DBL
00114 vrna_exp_E_ml_fast(vrna_fold_compound_t *fc,
00115                   int                i,
00116                   int                j,
00117                   vrna_mx_pf_aux_ml_t aux_mx);
00118
00119
00120 /* End partition function interface */
00121 int
00122 vrna_BT_mb_loop(vrna_fold_compound_t *fc,
00123                int                *i,
00124                int                *j,
00125                int                *k,
00126                int                en,
00127                int                *component1,
00128                int                *component2);
00129
00130
00131 int
00132 vrna_BT_mb_loop_split(vrna_fold_compound_t *fc,
00133                      int                *i,
00134                      int                *j,
00135                      int                *k,
00136                      int                *l,
00137                      int                *component1,
00138                      int                *component2,
00139                      vrna_bp_stack_t *bp_stack,
00140                      int                *stack_count);
00141
00142
00143 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00144
00145 PRIVATE INLINE int
00146 E_MLstem(int                type,
00147          int                sil,
00148          int                sj1,
00149          vrna_param_t *P)
00150 {
00151     int energy = 0;

```

```

00223
00224     if (sil >= 0 && sj1 >= 0)
00225         energy += P->mismatchM[type][sil][sj1];
00226     else if (sil >= 0)
00227         energy += P->dangle5[type][sil];
00228     else if (sj1 >= 0)
00229         energy += P->dangle3[type][sj1];
00230
00231     if (type > 2)
00232         energy += P->TerminalAU;
00233
00234     energy += P->MLintern[type];
00235
00236     return energy;
00237 }
00238
00239
00246 PRIVATE INLINE FLT_OR_DBL
00247 exp_E_MLstem(int          type,
00248              int          sil,
00249              int          sj1,
00250              vrna_exp_param_t *P)
00251 {
00252     double energy = 1.0;
00253
00254     if (sil >= 0 && sj1 >= 0)
00255         energy = P->expmismatchM[type][sil][sj1];
00256     else if (sil >= 0)
00257         energy = P->expdangle5[type][sil];
00258     else if (sj1 >= 0)
00259         energy = P->expdangle3[type][sj1];
00260
00261     if (type > 2)
00262         energy *= P->expTermAU;
00263
00264     energy *= P->expMLintern[type];
00265     return (FLT_OR_DBL)energy;
00266 }
00267
00268
00269 #endif
00270
00275 #endif

```

18.129 ViennaRNA/LPfold.h File Reference

Partition function and equilibrium probability implementation for the sliding window algorithm.
Include dependency graph for LPfold.h:

Functions

- void [update_pf_paramsLP](#) (int length)
- [vrna_ep_t * pfl_fold](#) (char *sequence, int winSize, int pairSize, float cutoffb, double **pU, [vrna_ep_t](#) **dpp2, FILE *pUfp, FILE *spup)
Compute partition functions for locally stable secondary structures.
- [vrna_ep_t * pfl_fold_par](#) (char *sequence, int winSize, int pairSize, float cutoffb, double **pU, [vrna_ep_t](#) **dpp2, FILE *pUfp, FILE *spup, [vrna_exp_param_t](#) *parameters)
Compute partition functions for locally stable secondary structures.
- void [putoutpU_prob](#) (double **pU, int length, int ulength, FILE *fp, int energies)
Writes the unpaired probabilities (pU) or opening energies into a file.
- void [putoutpU_prob_bin](#) (double **pU, int length, int ulength, FILE *fp, int energies)
Writes the unpaired probabilities (pU) or opening energies into a binary file.
- void [init_pf_foldLP](#) (int length)

18.129.1 Detailed Description

Partition function and equilibrium probability implementation for the sliding window algorithm.

This file contains the implementation for sliding window partition function and equilibrium probabilities. It also provides the unpaired probability implementation from Bernhart et al. 2011 [4]

18.129.2 Function Documentation

18.129.2.1 init_pf_foldLP()

```
void init_pf_foldLP (
    int length )
```

Dunno if this function was ever used by external programs linking to RNALib, but it was declared PUBLIC before. Anyway, never use this function as it will be removed soon and does nothing at all

18.130 LPfold.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_LPFOLD_H
00002 #define VIENNA_RNA_PACKAGE_LPFOLD_H
00003
00004 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00005
00016 #include <stdio.h>
00017
00018 #include <ViennaRNA/datastructures/basic.h>
00019 #include <ViennaRNA/params/basic.h>
00020 #include <ViennaRNA/part_func_window.h>
00021
00022 #ifdef VRNA_WARN_DEPRECATED
00023 # if defined(__clang__)
00024 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00025 # elif defined(__GNUC__)
00026 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00027 # else
00028 #  define DEPRECATED(func, msg) func
00029 # endif
00030 #else
00031 # define DEPRECATED(func, msg) func
00032 #endif
00033
00041 DEPRECATED(void update_pf_paramsLP(int length),
00042 "This function is obsolete");
00043
00044
00051 DEPRECATED(void update_pf_paramsLP_par(int length,
00052                                         vrna_exp_param_t *parameters),
00053 "Use the new API with vrna_folc_compound_t instead");
00054
00055
00093 DEPRECATED(vrna_ep_t *pfl_fold(char *sequence,
00094                                  int winSize,
00095                                  int pairSize,
00096                                  float cutofffb,
00097                                  double **pU,
00098                                  vrna_ep_t **dpp2,
00099                                  FILE *pUfp,
00100                                  FILE *spup),
00101 "Use vrna_pfl_fold(), vrna_pfl_fold_cb(), vrna_pfl_fold_up(), or vrna_pfl_fold_up_cb() instead");
00102
00103
00110 DEPRECATED(vrna_ep_t *pfl_fold_par(char *sequence,
00111                                     int winSize,
00112                                     int pairSize,
00113                                     float cutofffb,
00114                                     double **pU,
00115                                     vrna_ep_t **dpp2,
00116                                     FILE *pUfp,
00117                                     FILE *spup,
00118                                     vrna_exp_param_t *parameters),
00119 "Use the new API and vrna_probs_window() instead");
00120
00121
00122 DEPRECATED(void putoutpU_prob_par(double **pU,
00123                                   int length,
00124                                   int ulength,
00125                                   FILE *fp,
00126                                   int energies,
00127                                   vrna_exp_param_t *parameters),
00128 "");
00129
00130
00145 DEPRECATED(void putoutpU_prob(double **pU,
00146                                int length,
```

```

00147             int    ulength,
00148             FILE    *fp,
00149             int      energies),
00150 "");
00151
00152
00153 DEPRECATED(void putoutpU_prob_bin_par(double      **pU,
00154             int      length,
00155             int      ulength,
00156             FILE      *fp,
00157             int      energies,
00158             vrna_exp_param_t *parameters),
00159 "");
00160
00161
00162 DEPRECATED(void putoutpU_prob_bin(double **pU,
00163             int      length,
00164             int      ulength,
00165             FILE      *fp,
00166             int      energies),
00167 "");
00168
00169
00170 DEPRECATED(void init_pf_foldLP(int length),
00171 "This function is obsolete");
00172
00173 #endif
00174
00175 #endif

```

18.131 ViennaRNA/MEA.h File Reference

Computes a MEA (maximum expected accuracy) structure.

Include dependency graph for MEA.h:

Functions

- char * [vrna_MEA](#) ([vrna_fold_compound_t](#) *fc, double gamma, float *mea)
Compute a MEA (maximum expected accuracy) structure.
- char * [vrna_MEA_from_plist](#) ([vrna_ep_t](#) *plist, const char *sequence, double gamma, [vrna_md_t](#) *md, float *mea)
Compute a MEA (maximum expected accuracy) structure from a list of probabilities.
- float [MEA](#) ([plist](#) *p, char *structure, double gamma)
Computes a MEA (maximum expected accuracy) structure.

18.131.1 Detailed Description

Computes a MEA (maximum expected accuracy) structure.

18.132 MEA.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_MEA_H
00002 #define VIENNA_RNA_PACKAGE_MEA_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005 #include <ViennaRNA/params/basic.h>
00006
00007 char *
00008 vrna_MEA(vrna_fold_compound_t *fc,
00009          double gamma,
00010          float *mea);
00011
00012 char *
00013 vrna_MEA_from_plist(vrna_ep_t *plist,
00014                     const char *sequence,
00015                     double gamma,
00016                     vrna_md_t *md,
00017                     float *mea);
00018
00019 #endif

```

```

00078 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00079
00080 #ifdef VRNA_WARN_DEPRECATED
00081 # if defined(__clang__)
00082 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00083 # elif defined(__GNUC__)
00084 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00085 # else
00086 #  define DEPRECATED(func, msg) func
00087 # endif
00088 #else
00089 # define DEPRECATED(func, msg) func
00090 #endif
00091
00092
00108 DEPRECATED(float
00109     MEA(plist *p,
00110         char *structure,
00111         double gamma),
00112     "Use vrna_MEA() or vrna_MEA_from_plist() instead!");
00113
00114
00115 DEPRECATED(float
00116     MEA_seq(plist *p,
00117             const char *sequence,
00118             char *structure,
00119             double gamma,
00120             vrna_exp_param_t *pf),
00121     "Use vrna_MEA() or vrna_MEA_from_plist() instead!");
00122
00123
00124 #endif
00125
00126 #endif

```

18.133 ViennaRNA/mfe.h File Reference

Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.

Include dependency graph for mfe.h: This graph shows which files directly or indirectly include this file:

Functions

- float [vrna_backtrack5](#) ([vrna_fold_compound_t](#) *fc, unsigned int length, char *structure)
Backtrace an MFE (sub)structure.

Basic global MFE prediction interface

- float [vrna_mfe](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Compute minimum free energy and an appropriate secondary structure of an RNA sequence, or RNA sequence alignment.
- float [vrna_mfe_dimer](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Compute the minimum free energy of two interacting RNA molecules.

Simplified global MFE prediction using sequence(s) or multiple sequence alignment(s)

- float [vrna_fold](#) (const char *sequence, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding secondary structure for an RNA sequence.
- float [vrna_circfold](#) (const char *sequence, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding secondary structure for a circular RNA sequence.
- float [vrna_alifold](#) (const char **sequences, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for an RNA sequence alignment using a comparative method.
- float [vrna_circalifold](#) (const char **sequences, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding consensus secondary structure for a sequence alignment of circular RNAs using a comparative method.
- float [vrna_cofold](#) (const char *sequence, char *structure)
Compute Minimum Free Energy (MFE), and a corresponding secondary structure for two dimerized RNA sequences.

18.133.1 Detailed Description

Compute Minimum Free energy (MFE) and backtrace corresponding secondary structures from RNA sequence data.

This file includes (almost) all function declarations within the RNAlib that are related to MFE folding...

18.134 mfe.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_MFE_H
00002 #define VIENNA_RNA_PACKAGE_MFE_H
00003
00004 #include <stdio.h>
00005 #include <ViennaRNA/datastructures/basic.h>
00006 #include <ViennaRNA/fold_compound.h>
00007
00008 #ifdef VRNA_WARN_DEPRECATED
00009 # if defined(__clang__)
00010 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00011 # elif defined(__GNUC__)
00012 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00013 # else
00014 #  define DEPRECATED(func, msg) func
00015 # endif
00016 #else
00017 # define DEPRECATED(func, msg) func
00018 #endif
00019
00020
00097 float
00098 vrna_mfe(vrna_fold_compound_t *vc,
00099         char *structure);
00100
00101
00117 DEPRECATED(float
00118             vrna_mfe_dimer(vrna_fold_compound_t *vc,
00119                           char *structure),
00120             "Use vrna_mfe() instead");
00121
00122
00153 float
00154 vrna_fold(const char *sequence,
00155           char *structure);
00156
00157
00180 float
00181 vrna_circfold(const char *sequence,
00182              char *structure);
00183
00184
00206 float
00207 vrna_alifold(const char **sequences,
00208             char *structure);
00209
00210
00235 float
00236 vrna_circalifold(const char **sequences,
00237                 char *structure);
00238
00239
00264 DEPRECATED(float
00265             vrna_cofold(const char *sequence,
00266                       char *structure),
00267             "Use vrna_fold() instead");
00268
00269
00289 int
00290 vrna_backtrack_from_intervals(vrna_fold_compound_t *vc,
00291                             vrna_bp_stack_t *bp_stack,
00292                             sect bt_stack[],
00293                             int s);
00294
00295
00317 float
00318 vrna_backtrack5(vrna_fold_compound_t *fc,
00319                unsigned int length,
00320                char *structure);
00321
00322
00323 int
00324 vrna_backtrack_window(vrna_fold_compound_t *fc,

```

```

00325         const char      *Lfold_filename,
00326         long             file_pos,
00327         char             **structure,
00328         double            mfe);
00329
00330
00337 #endif

```

18.135 ViennaRNA/mfe_window.h File Reference

Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.

Include dependency graph for mfe_window.h: This graph shows which files directly or indirectly include this file:

Typedefs

- typedef void(* [vrna_mfe_window_f](#)) (int start, int end, const char *structure, float en, void *data)
The default callback for sliding window MFE structure predictions.

Functions

Basic local (sliding window) MFE prediction interface

- float [vrna_mfe_window](#) ([vrna_fold_compound_t](#) *vc, FILE *file)
Local MFE prediction using a sliding window approach.
- float [vrna_mfe_window_cb](#) ([vrna_fold_compound_t](#) *vc, [vrna_mfe_window_f](#) cb, void *data)
- float [vrna_mfe_window_zscore](#) ([vrna_fold_compound_t](#) *vc, double min_z, FILE *file)
Local MFE prediction using a sliding window approach (with z-score cut-off)
- float [vrna_mfe_window_zscore_cb](#) ([vrna_fold_compound_t](#) *vc, double min_z, [vrna_mfe_window_f](#) cb, void *data)

Simplified local MFE prediction using sequence(s) or multiple sequence alignment(s)

- float [vrna_Lfold](#) (const char *string, int window_size, FILE *file)
Local MFE prediction using a sliding window approach (simplified interface)
- float [vrna_Lfold_cb](#) (const char *string, int window_size, [vrna_mfe_window_f](#) cb, void *data)
- float [vrna_Lfoldz](#) (const char *string, int window_size, double min_z, FILE *file)
Local MFE prediction using a sliding window approach with z-score cut-off (simplified interface)
- float [vrna_Lfoldz_cb](#) (const char *string, int window_size, double min_z, [vrna_mfe_window_zscore_f](#) cb, void *data)
- float [vrna_alifold](#) (const char **alignment, int maxdist, FILE *fp)
- float [vrna_alifold_cb](#) (const char **alignment, int maxdist, [vrna_mfe_window_f](#) cb, void *data)

18.135.1 Detailed Description

Compute local Minimum Free Energy (MFE) using a sliding window approach and backtrace corresponding secondary structures.

This file includes the interface to all functions related to predicting locally stable secondary structures.

18.136 mfe_window.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_MFE_WINDOW_H
00002 #define VIENNA_RNA_PACKAGE_MFE_WINDOW_H
00003
00004 #include <stdio.h>
00005 #include <ViennaRNA/fold_compound.h>
00006
00007 #ifdef VRNA_WITH_SVM
00008 #include <ViennaRNA/zscore.h>
00009 #endif
00010

```

```

00011 #ifdef VRNA_WARN_DEPRECATED
00012 # if defined(DEPRECATED)
00013 #   undef DEPRECATED
00014 # endif
00015 # if defined(__clang__)
00016 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00017 # elif defined(__GNUC__)
00018 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00019 # else
00020 #   define DEPRECATED(func, msg) func
00021 # endif
00022 #else
00023 # define DEPRECATED(func, msg) func
00024 #endif
00025
00079 typedef void (*vrna_mfe_window_f)(int start,
00080                                   int end,
00081                                   const char *structure,
00082                                   float en,
00083                                   void *data);
00084
00085 DEPRECATED(typedef void (vrna_mfe_window_callback)(int start,
00086                                                    int end,
00087                                                    const char *structure,
00088                                                    float en,
00089                                                    void *data),
00090            "Use vrna_mfe_window_f instead!");
00091
00092
00093
00094 #ifdef VRNA_WITH_SVM
00095 typedef void (*vrna_mfe_window_zscore_f)(int start,
00096                                           int end,
00097                                           const char *structure,
00098                                           float en,
00099                                           float zscore,
00100                                           void *data);
00101
00102 DEPRECATED(typedef void (vrna_mfe_window_zscore_callback)(int start,
00103                                                           int end,
00104                                                           const char *structure,
00105                                                           float en,
00106                                                           float zscore,
00107                                                           void *data),
00108            "Use vrna_mfe_window_zscore_f instead!");
00109 #endif
00110
00141 float
00142 vrna_mfe_window(vrna_fold_compound_t *vc,
00143                FILE *file);
00144
00145
00146 float
00147 vrna_mfe_window_cb(vrna_fold_compound_t *vc,
00148                   vrna_mfe_window_f cb,
00149                   void *data);
00150
00151
00152 #ifdef VRNA_WITH_SVM
00178 float
00179 vrna_mfe_window_zscore(vrna_fold_compound_t *vc,
00180                       double min_z,
00181                       FILE *file);
00182
00183
00184 float
00185 vrna_mfe_window_zscore_cb(vrna_fold_compound_t *vc,
00186                          double min_z,
00187                          vrna_mfe_window_zscore_f cb,
00188                          void *data);
00189
00190
00191 #endif
00192
00193 /* End basic local MFE interface */
00221 float
00222 vrna_Lfold(const char *string,
00223           int window_size,
00224           FILE *file);
00225
00226
00227 float
00228 vrna_Lfold_cb(const char *string,
00229              int window_size,
00230              vrna_mfe_window_f cb,
00231              void *data);
00232

```

```

00233
00234 #ifdef VRNA_WITH_SVM
00259 float
00260 vrna_Lfoldz(const char *string,
00261             int window_size,
00262             double min_z,
00263             FILE *file);
00264
00265
00266 float
00267 vrna_Lfoldz_cb(const char *string,
00268                int window_size,
00269                double min_z,
00270                vrna_mfe_window_zscore_f cb,
00271                void *data);
00272
00273
00274 #endif
00275
00276 float vrna_aliiLfold(const char **alignment,
00277                     int maxdist,
00278                     FILE *fp);
00279
00280
00281 float vrna_aliiLfold_cb(const char **alignment,
00282                        int maxdist,
00283                        vrna_mfe_window_f cb,
00284                        void *data);
00285
00286
00287 /* End simplified local MFE interface */
00290 /* End group mfe_fold_window */
00294 #endif

```

18.137 ViennaRNA/mm.h File Reference

Several Maximum Matching implementations.
 Include dependency graph for mm.h:

Functions

- int [vrna_maximum_matching](#) ([vrna_fold_compound_t](#) *fc)
- int [vrna_maximum_matching_simple](#) (const char *sequence)

18.137.1 Detailed Description

Several Maximum Matching implementations.
 This file contains the declarations for several maximum matching implementations

18.137.2 Function Documentation

18.137.2.1 vrna_maximum_matching()

```

int vrna_maximum_matching (
    vrna_fold_compound_t * fc )

```

SWIG Wrapper Notes This function is attached as method **maximum_matching()** to objects of type `fold_compound` (i.e. [vrna_fold_compound_t](#)).

18.137.2.2 vrna_maximum_matching_simple()

```

int vrna_maximum_matching_simple (
    const char * sequence )

```

SWIG Wrapper Notes This function is available as global function **maximum_matching()**.

18.138 mm.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_MM_H
00002 #define VIENNA_RNA_PACKAGE_MM_H
00003
00013 #include <ViennaRNA/fold_compound.h>
00014
00015 int
00016 vrna_maximum_matching(vrna_fold_compound_t *fc);
00017
00018 int
00019 vrna_maximum_matching_simple(const char *sequence);
00020
00021
00022 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00023
00024 unsigned int
00025 maximumMatching(const char *string);
00026
00027 unsigned int *
00028 maximumMatchingConstraint(const char *string,
00029                          short *ptable);
00030
00031 unsigned int *
00032 maximumMatching2Constraint(const char *string,
00033                           short *ptable,
00034                           short *ptable2);
00035
00036 #endif
00037
00038 #endif
```

18.139 ViennaRNA/model.h File Reference

The model details data structure and its corresponding modifiers.

This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_md_s](#)

The data structure that contains the complete model details used throughout the calculations. [More...](#)

Macros

- #define [VRNA_MODEL_DEFAULT_TEMPERATURE](#) 37.0
Default temperature for structure prediction and free energy evaluation in °C
- #define [VRNA_MODEL_DEFAULT_PF_SCALE](#) -1
Default scaling factor for partition function computations.
- #define [VRNA_MODEL_DEFAULT_BETA_SCALE](#) 1.
Default scaling factor for absolute thermodynamic temperature in Boltzmann factors.
- #define [VRNA_MODEL_DEFAULT_DANGLES](#) 2
Default dangling end model.
- #define [VRNA_MODEL_DEFAULT_SPECIAL_HP](#) 1
Default model behavior for lookup of special tri-, tetra-, and hexa-loops.
- #define [VRNA_MODEL_DEFAULT_NO_LP](#) 0
Default model behavior for so-called 'lonely pairs'.
- #define [VRNA_MODEL_DEFAULT_NO_GU](#) 0
Default model behavior for G-U base pairs.
- #define [VRNA_MODEL_DEFAULT_NO_GU_CLOSURE](#) 0
Default model behavior for G-U base pairs closing a loop.
- #define [VRNA_MODEL_DEFAULT_CIRC](#) 0

- *Default model behavior to treat a molecule as a circular RNA (DNA)*
- `#define VRNA_MODEL_DEFAULT_GQUAD 0`
- *Default model behavior regarding the treatment of G-Quadruplexes.*
- `#define VRNA_MODEL_DEFAULT_UNIQ_ML 0`
- *Default behavior of the model regarding unique multi-branch loop decomposition.*
- `#define VRNA_MODEL_DEFAULT_ENERGY_SET 0`
- *Default model behavior on which energy set to use.*
- `#define VRNA_MODEL_DEFAULT_BACKTRACK 1`
- *Default model behavior with regards to backtracking of structures.*
- `#define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'`
- *Default model behavior on what type of backtracking to perform.*
- `#define VRNA_MODEL_DEFAULT_COMPUTE_BPP 1`
- *Default model behavior with regards to computing base pair probabilities.*
- `#define VRNA_MODEL_DEFAULT_MAX_BP_SPAN -1`
- *Default model behavior for the allowed maximum base pair span.*
- `#define VRNA_MODEL_DEFAULT_WINDOW_SIZE -1`
- *Default model behavior for the sliding window approach.*
- `#define VRNA_MODEL_DEFAULT_LOG_ML 0`
- *Default model behavior on how to evaluate the energy contribution of multi-branch loops.*
- `#define VRNA_MODEL_DEFAULT_ALI_OLD_EN 0`
- *Default model behavior for consensus structure energy evaluation.*
- `#define VRNA_MODEL_DEFAULT_ALI_RIBO 0`
- *Default model behavior for consensus structure co-variance contribution assessment.*
- `#define VRNA_MODEL_DEFAULT_ALI_CV_FACT 1.`
- *Default model behavior for weighting the co-variance score in consensus structure prediction.*
- `#define VRNA_MODEL_DEFAULT_ALI_NC_FACT 1.`
- *Default model behavior for weighting the nucleotide conservation? in consensus structure prediction.*
- `#define VRNA_MODEL_DEFAULT_SALT 1.021`
- *Default model salt concentration (M)*
- `#define VRNA_MODEL_DEFAULT_SALTMLLOWER 6`
- *Default model lower bound of multiloop size for salt correction fitting.*
- `#define VRNA_MODEL_DEFAULT_SALTMLUPPER 24`
- *Default model upper bound of multiloop size for salt correction fitting.*
- `#define VRNA_MODEL_DEFAULT_SALDPXINIT 99999`
- *Default model value to turn off user-provided salt correction for duplex initialization.*
- `#define MAXALPHA 20`
- *Maximal length of alphabet.*

Typedefs

- `typedef struct vrna_md_s vrna_md_t`
- *Typename for the model details data structure [vrna_md_s](#).*

Functions

- `void vrna_md_set_default (vrna_md_t *md)`
- *Apply default model details to a provided [vrna_md_t](#) data structure.*
- `void vrna_md_update (vrna_md_t *md)`
- *Update the model details data structure.*
- `vrna_md_t * vrna_md_copy (vrna_md_t *md_to, const vrna_md_t *md_from)`
- *Copy/Clone a [vrna_md_t](#) model.*
- `char * vrna_md_option_string (vrna_md_t *md)`

- Get a corresponding cmdline parameter string of the options in a `vrna_md_t`.

 - void `vrna_md_defaults_reset` (`vrna_md_t *md_p`)

Reset the global default model details to a specific set of parameters, or their initial values.
- void `vrna_md_defaults_temperature` (double T)

Set default temperature for energy evaluation of loops.
- double `vrna_md_defaults_temperature_get` (void)

Get default temperature for energy evaluation of loops.
- void `vrna_md_defaults_betaScale` (double b)

Set default scaling factor of thermodynamic temperature in Boltzmann factors.
- double `vrna_md_defaults_betaScale_get` (void)

Get default scaling factor of thermodynamic temperature in Boltzmann factors.
- void `vrna_md_defaults_dangles` (int d)

Set default dangle model for structure prediction.
- int `vrna_md_defaults_dangles_get` (void)

Get default dangle model for structure prediction.
- void `vrna_md_defaults_special_hp` (int flag)

Set default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
- int `vrna_md_defaults_special_hp_get` (void)

Get default behavior for lookup of tabulated free energies for special hairpin loops, such as Tri-, Tetra-, or Hexa-loops.
- void `vrna_md_defaults_noLP` (int flag)

Set default behavior for prediction of canonical secondary structures.
- int `vrna_md_defaults_noLP_get` (void)

Get default behavior for prediction of canonical secondary structures.
- void `vrna_md_defaults_noGU` (int flag)

Set default behavior for treatment of G-U wobble pairs.
- int `vrna_md_defaults_noGU_get` (void)

Get default behavior for treatment of G-U wobble pairs.
- void `vrna_md_defaults_noGUclosure` (int flag)

Set default behavior for G-U pairs as closing pair for loops.
- int `vrna_md_defaults_noGUclosure_get` (void)

Get default behavior for G-U pairs as closing pair for loops.
- void `vrna_md_defaults_logML` (int flag)

Set default behavior recomputing free energies of multi-branch loops using a logarithmic model.
- int `vrna_md_defaults_logML_get` (void)

Get default behavior recomputing free energies of multi-branch loops using a logarithmic model.
- void `vrna_md_defaults_circ` (int flag)

Set default behavior whether input sequences are circularized.
- int `vrna_md_defaults_circ_get` (void)

Get default behavior whether input sequences are circularized.
- void `vrna_md_defaults_gquad` (int flag)

Set default behavior for treatment of G-Quadruplexes.
- int `vrna_md_defaults_gquad_get` (void)

Get default behavior for treatment of G-Quadruplexes.
- void `vrna_md_defaults_uniq_ML` (int flag)

Set default behavior for creating additional matrix for unique multi-branch loop prediction.
- int `vrna_md_defaults_uniq_ML_get` (void)

Get default behavior for creating additional matrix for unique multi-branch loop prediction.
- void `vrna_md_defaults_energy_set` (int e)

Set default energy set.
- int `vrna_md_defaults_energy_set_get` (void)

Get default energy set.

- void [vrna_md_defaults_backtrack](#) (int flag)
Set default behavior for whether to backtrack secondary structures.
- int [vrna_md_defaults_backtrack_get](#) (void)
Get default behavior for whether to backtrack secondary structures.
- void [vrna_md_defaults_backtrack_type](#) (char t)
Set default backtrack type, i.e. which DP matrix is used.
- char [vrna_md_defaults_backtrack_type_get](#) (void)
Get default backtrack type, i.e. which DP matrix is used.
- void [vrna_md_defaults_compute_bpp](#) (int flag)
Set the default behavior for whether to compute base pair probabilities after partition function computation.
- int [vrna_md_defaults_compute_bpp_get](#) (void)
Get the default behavior for whether to compute base pair probabilities after partition function computation.
- void [vrna_md_defaults_max_bp_span](#) (int span)
Set default maximal base pair span.
- int [vrna_md_defaults_max_bp_span_get](#) (void)
Get default maximal base pair span.
- void [vrna_md_defaults_min_loop_size](#) (int size)
Set default minimal loop size.
- int [vrna_md_defaults_min_loop_size_get](#) (void)
Get default minimal loop size.
- void [vrna_md_defaults_window_size](#) (int size)
Set default window size for sliding window structure prediction approaches.
- int [vrna_md_defaults_window_size_get](#) (void)
Get default window size for sliding window structure prediction approaches.
- void [vrna_md_defaults_oldAliEn](#) (int flag)
Set default behavior for whether to use old energy model for comparative structure prediction.
- int [vrna_md_defaults_oldAliEn_get](#) (void)
Get default behavior for whether to use old energy model for comparative structure prediction.
- void [vrna_md_defaults_ribo](#) (int flag)
Set default behavior for whether to use Ribosum Scoring in comparative structure prediction.
- int [vrna_md_defaults_ribo_get](#) (void)
Get default behavior for whether to use Ribosum Scoring in comparative structure prediction.
- void [vrna_md_defaults_cv_fact](#) (double factor)
Set the default co-variance scaling factor used in comparative structure prediction.
- double [vrna_md_defaults_cv_fact_get](#) (void)
Get the default co-variance scaling factor used in comparative structure prediction.
- void [vrna_md_defaults_nc_fact](#) (double factor)
- double [vrna_md_defaults_nc_fact_get](#) (void)
- void [vrna_md_defaults_sfact](#) (double factor)
Set the default scaling factor used to avoid under-/overflows in partition function computation.
- double [vrna_md_defaults_sfact_get](#) (void)
Get the default scaling factor used to avoid under-/overflows in partition function computation.
- void [vrna_md_defaults_salt](#) (double salt)
Set the default salt concentration.
- double [vrna_md_defaults_salt_get](#) (void)
Get the default salt concentration.
- void [vrna_md_defaults_saltMLLower](#) (int lower)
Set the default multiloop size lower bound for loop salt correction linear fitting.
- int [vrna_md_defaults_saltMLLower_get](#) (void)
Get the default multiloop size lower bound for loop salt correction linear fitting.
- void [vrna_md_defaults_saltMLUpper](#) (int upper)

- *Set the default multiloop size upper bound for loop salt correction linear fitting.*
- int [vrna_md_defaults_saltMLUpper_get](#) (void)
Get the default multiloop size upper bound for loop salt correction linear fitting.
- void [vrna_md_defaults_saltDPXInit](#) (int value)
Set user-provided salt correction for duplex initialization. If value is 99999 the default value from fitting is used.
- int [vrna_md_defaults_saltDPXInit_get](#) (void)
Get user-provided salt correction for duplex initialization. If value is 99999 the default value from fitting is used.
- void [set_model_details](#) ([vrna_md_t](#) *md)
Set default model details.

Variables

- double [temperature](#)
Rescale energy parameters to a temperature in degC.
- double [pf_scale](#)
A scaling factor used by [pf_fold\(\)](#) to avoid overflows.
- int [dangles](#)
Switch the energy model for dangling end contributions (0, 1, 2, 3)
- int [tetra_loop](#)
Include special stabilizing energies for some tri-, tetra- and hexa-loops;.
- int [noLonelyPairs](#)
Global switch to avoid/allow helices of length 1.
- int [noGU](#)
Global switch to forbid/allow GU base pairs at all.
- int [no_closingGU](#)
GU allowed only inside stacks if set to 1.
- int [circ](#)
backward compatibility variable.. this does not effect anything
- int [gquad](#)
Allow G-quadruplex formation.
- int [uniq_ML](#)
do ML decomposition uniquely (for subopt)
- int [energy_set](#)
0 = BP; 1=any with GC; 2=any with AU-parameter
- int [do_backtrack](#)
do backtracking, i.e. compute secondary structures or base pair probabilities
- char [backtrack_type](#)
A backtrack array marker for [inverse_fold\(\)](#)
- char * [nonstandards](#)
contains allowed non standard base pairs
- int [max_bp_span](#)
Maximum allowed base pair span.
- int [oldAlfEn](#)
use old alifold energies (with gaps)
- int [ribo](#)
use ribosum matrices
- int [logML](#)
if nonzero use logarithmic ML energy in [energy_of_struct](#)
- double [salt](#)
salt concentration
- int [saltDPXInit](#)
Salt correction for duplex initialization.

18.139.1 Detailed Description

The model details data structure and its corresponding modifiers.

18.140 model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_MODEL_H
00002 #define VIENNA_RNA_PACKAGE_MODEL_H
00003
00017 #ifndef NBASES
00018 #define NBASES 8
00019 #endif
00020
00022 typedef struct vrna_md_s vrna_md_t;
00023
00030 #define VRNA_MODEL_DEFAULT_TEMPERATURE 37.0
00031
00036 #define VRNA_MODEL_DEFAULT_PF_SCALE -1
00037
00042 #define VRNA_MODEL_DEFAULT_BETA_SCALE 1.
00043
00047 #define VRNA_MODEL_DEFAULT_DANGLES 2
00048
00053 #define VRNA_MODEL_DEFAULT_SPECIAL_HP 1
00054
00059 #define VRNA_MODEL_DEFAULT_NO_LP 0
00060
00065 #define VRNA_MODEL_DEFAULT_NO_GU 0
00066
00071 #define VRNA_MODEL_DEFAULT_NO_GU_CLOSURE 0
00072
00077 #define VRNA_MODEL_DEFAULT_CIRC 0
00078
00083 #define VRNA_MODEL_DEFAULT_GQUAD 0
00084
00089 #define VRNA_MODEL_DEFAULT_UNIQ_ML 0
00090
00095 #define VRNA_MODEL_DEFAULT_ENERGY_SET 0
00096
00101 #define VRNA_MODEL_DEFAULT_BACKTRACK 1
00102
00107 #define VRNA_MODEL_DEFAULT_BACKTRACK_TYPE 'F'
00108
00113 #define VRNA_MODEL_DEFAULT_COMPUTE_BPP 1
00114
00119 #define VRNA_MODEL_DEFAULT_MAX_BP_SPAN -1
00120
00125 #define VRNA_MODEL_DEFAULT_WINDOW_SIZE -1
00126
00131 #define VRNA_MODEL_DEFAULT_LOG_ML 0
00132
00137 #define VRNA_MODEL_DEFAULT_ALI_OLD_EN 0
00138
00143 #define VRNA_MODEL_DEFAULT_ALI_RIBO 0
00144
00149 #define VRNA_MODEL_DEFAULT_ALI_CV_FACT 1.
00150
00154 #define VRNA_MODEL_DEFAULT_ALI_NC_FACT 1.
00155
00156
00157 #define VRNA_MODEL_DEFAULT_PF_SMOOTH 1
00158
00162 #define VRNA_MODEL_DEFAULT_SALT 1.021
00163
00164
00168 #define VRNA_MODEL_DEFAULT_SALTMLOWER 6
00169
00170
00174 #define VRNA_MODEL_DEFAULT_SALTMLUPPER 24
00175
00176
00180 #define VRNA_MODEL_DEFAULT_SALTPXINIT 99999
00181
00182
00183 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00184
00185 #ifndef MAXALPHA
00189 #define MAXALPHA 20
00190 #endif
00191
00192 #endif
00193
00203 struct vrna_md_s {

```

```
00204 double temperature;
00205 double betaScale;
00206 int pf_smooth;
00207 int dangles;
00231 int special_hp;
00232 int noLP;
00233 int noGU;
00234 int noGUclosure;
00235 int logML;
00236 int circ;
00237 int gquad;
00238 int uniq_ML;
00239 int energy_set;
00240 int backtrack;
00241 char backtrack_type;
00242 int compute_bpp;
00243 char nonstandards[64];
00244 int max_bp_span;
00246 int min_loop_size;
00250 int window_size;
00251 int oldAliEn;
00252 int ribo;
00253 double cv_fact;
00254 double nc_fact;
00255 double sfact;
00256 int rtype[8];
00257 short alias[MAXALPHA + 1];
00258 int pair[MAXALPHA + 1][MAXALPHA + 1];
00259 float pair_dist[7][7];
00260 double salt;
00261 int saltMLLower;
00262 int saltMLUpper;
00263 int saltDPXInit;
00267 };
00268
00269
00278 void
00279 vrna_md_set_default(vrna_md_t *md);
00280
00281
00294 void
00295 vrna_md_update(vrna_md_t *md);
00296
00297
00308 vrna_md_t *
00309 vrna_md_copy(vrna_md_t *md_to,
00310             const vrna_md_t *md_from);
00311
00312
00318 char *
00319 vrna_md_option_string(vrna_md_t *md);
00320
00321
00322 void
00323 vrna_md_set_nonstandards(vrna_md_t *md,
00324                         const char *ns_bases);
00325
00326
00344 void
00345 vrna_md_defaults_reset(vrna_md_t *md_p);
00346
00347
00353 void
00354 vrna_md_defaults_temperature(double T);
00355
00356
00362 double
00363 vrna_md_defaults_temperature_get(void);
00364
00365
00373 void
00374 vrna_md_defaults_betaScale(double b);
00375
00376
00383 double
00384 vrna_md_defaults_betaScale_get(void);
00385
00386
00387 void
00388 vrna_md_defaults_pf_smooth(int s);
00389
00390
00391 int
00392 vrna_md_defaults_pf_smooth_get(void);
00393
00394
00400 void
```

```
00401 vrna_md_defaults_dangles(int d);
00402
00403
00409 int
00410 vrna_md_defaults_dangles_get(void);
00411
00412
00418 void
00419 vrna_md_defaults_special_hp(int flag);
00420
00421
00427 int
00428 vrna_md_defaults_special_hp_get(void);
00429
00430
00436 void
00437 vrna_md_defaults_noLP(int flag);
00438
00439
00445 int
00446 vrna_md_defaults_noLP_get(void);
00447
00448
00454 void
00455 vrna_md_defaults_noGU(int flag);
00456
00457
00463 int
00464 vrna_md_defaults_noGU_get(void);
00465
00466
00472 void
00473 vrna_md_defaults_noGUclosure(int flag);
00474
00475
00481 int
00482 vrna_md_defaults_noGUclosure_get(void);
00483
00484
00490 void
00491 vrna_md_defaults_logML(int flag);
00492
00493
00499 int
00500 vrna_md_defaults_logML_get(void);
00501
00502
00508 void
00509 vrna_md_defaults_circ(int flag);
00510
00511
00517 int
00518 vrna_md_defaults_circ_get(void);
00519
00520
00526 void
00527 vrna_md_defaults_gquad(int flag);
00528
00529
00535 int
00536 vrna_md_defaults_gquad_get(void);
00537
00538
00545 void
00546 vrna_md_defaults_uniq_ML(int flag);
00547
00548
00554 int
00555 vrna_md_defaults_uniq_ML_get(void);
00556
00557
00563 void
00564 vrna_md_defaults_energy_set(int e);
00565
00566
00572 int
00573 vrna_md_defaults_energy_set_get(void);
00574
00575
00581 void
00582 vrna_md_defaults_backtrack(int flag);
00583
00584
00590 int
00591 vrna_md_defaults_backtrack_get(void);
00592
00593
```

```
00599 void
00600 vrna_md_defaults_backtrack_type(char t);
00601
00602
00608 char
00609 vrna_md_defaults_backtrack_type_get(void);
00610
00611
00617 void
00618 vrna_md_defaults_compute_bpp(int flag);
00619
00620
00626 int
00627 vrna_md_defaults_compute_bpp_get(void);
00628
00629
00635 void
00636 vrna_md_defaults_max_bp_span(int span);
00637
00638
00644 int
00645 vrna_md_defaults_max_bp_span_get(void);
00646
00647
00653 void
00654 vrna_md_defaults_min_loop_size(int size);
00655
00656
00662 int
00663 vrna_md_defaults_min_loop_size_get(void);
00664
00665
00671 void
00672 vrna_md_defaults_window_size(int size);
00673
00674
00680 int
00681 vrna_md_defaults_window_size_get(void);
00682
00683
00691 void
00692 vrna_md_defaults_oldAliEn(int flag);
00693
00694
00700 int
00701 vrna_md_defaults_oldAliEn_get(void);
00702
00703
00709 void
00710 vrna_md_defaults_ribo(int flag);
00711
00712
00718 int
00719 vrna_md_defaults_ribo_get(void);
00720
00721
00727 void
00728 vrna_md_defaults_cv_fact(double factor);
00729
00730
00736 double
00737 vrna_md_defaults_cv_fact_get(void);
00738
00739
00745 void
00746 vrna_md_defaults_nc_fact(double factor);
00747
00748
00754 double
00755 vrna_md_defaults_nc_fact_get(void);
00756
00757
00763 void
00764 vrna_md_defaults_sfact(double factor);
00765
00766
00772 double
00773 vrna_md_defaults_sfact_get(void);
00774
00775
00781 void
00782 vrna_md_defaults_salt(double salt);
00783
00784
00789 double
00790 vrna_md_defaults_salt_get(void);
00791
```

```

00792
00798 void
00799 vrna_md_defaults_saltMLLower(int lower);
00800
00801
00806 int
00807 vrna_md_defaults_saltMLLower_get(void);
00808
00809
00815 void
00816 vrna_md_defaults_saltMLUpper(int upper);
00817
00818
00823 int
00824 vrna_md_defaults_saltMLUpper_get(void);
00825
00826
00833 void
00834 vrna_md_defaults_saltDPXInit(int value);
00835
00836
00843 int
00844 vrna_md_defaults_saltDPXInit_get(void);
00845
00846 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00847
00848 #define model_detailsT      vrna_md_t          /* restore compatibility of struct rename */
00849
00850 /* BEGIN deprecated global variables: */
00851
00861 extern double temperature;
00862
00874 extern double pf_scale;
00875
00897 extern int dangles;
00898
00904 extern int tetra_loop;
00905
00913 extern int noLonelyPairs;
00914
00918 extern int noGU;
00919
00923 extern int no_closingGU;
00924
00928 extern int circ;
00929
00933 extern int gquad;
00934
00938 extern int uniq_ML;
00939
00947 extern int energy_set;
00948
00955 extern int do_backtrack;
00956
00964 extern char backtrack_type;
00965
00973 extern char *nonstandards;
00974
00980 extern int max_bp_span;
00981
00985 extern int oldAliEn;
00986
00990 extern int ribo;
00991
00992 extern double cv_fact;
00993
00994 extern double nc_fact;
00995
00997 extern int logML;
00998
01000 extern double salt;
01001
01003 extern int saltDPXInit;
01004
01005
01006 /* END deprecated global variables: */
01007
01021 void
01022 set_model_details(vrna_md_t *md);
01023
01024
01025 char *
01026 option_string(void);
01027
01028
01029 #endif
01034 #endif

```

18.141 move_set.h

```

00001 #ifndef __MOVE_SET_H
00002 #define __MOVE_SET_H
00003
00004 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00005
00009 typedef struct _struct_en{
00010     int energy;          /* energy in 10kcal/mol*/
00011     short *structure;    /* structure in energy_of_move format*/
00012 } struct_en;
00013
00014 /* prints structure*/
00015 void print_stren(FILE *out, struct_en *str);
00016 void print_str(FILE *out, short *str);
00017
00018 /* copying functions*/
00019 void copy_arr(short *dest, short *src); /*just copy*/
00020 short *allocopy(short *src);          /*copy and make space*/
00021
00022 enum MOVE_TYPE {GRADIENT, FIRST, ADAPTIVE};
00023
00024 /* walking methods (verbose_lvl 0-2, shifts = use shift moves? noLP = no lone pairs? (not compatible
    with shifts))
00025     input:    seq - sequence
00026              ptable - structure encoded with make_pair_table() from pair_mat.h
00027              s, sl - sequence encoded with encode_sequence from pair_mat.h
00028     methods:  deepest - lowest energy structure is used
00029              first - first found lower energy structure is used
00030              rand - random lower energy structure is used
00031     returns local minima structure in ptable and its energy in 10kcal/mol as output */
00032
00033 int move_gradient( char *seq,
00034                   short *ptable,
00035                   short *s,
00036                   short *sl,
00037                   int verbosity_level,
00038                   int shifts,
00039                   int noLP);
00040 int move_first( char *seq,
00041                short *ptable,
00042                short *s,
00043                short *sl,
00044                int verbosity_level,
00045                int shifts,
00046                int noLP);
00047 int move_adaptive( char *seq,
00048                  short *ptable,
00049                  short *s,
00050                  short *sl,
00051                  int verbosity_level);
00052
00053 /* standardized method that encapsulates above "_pt" methods
00054     input:  seq - sequence
00055            struc - structure in dot-bracket notation
00056            type - type of move selection according to MOVE_TYPE enum
00057     return: energy of LM
00058            structure of LM in struc in bracket-dot notation
00059 */
00060 int move_standard(char *seq,
00061                  char *struc,
00062                  enum MOVE_TYPE type,
00063                  int verbosity_level,
00064                  int shifts,
00065                  int noLP);
00066
00067
00068 /* browse_neighbours and perform funct function on each of them (used mainly for user specified
    flooding)
00069     input:    seq - sequence
00070              ptable - structure encoded with make_pair_table() from pair_mat.h
00071              s, sl - sequence encoded with encode_sequence from pair_mat.h
00072              funct - function (structure from neighbourhood, structure from input) toperform on every
    structure in neighbourhood (if the function returns non-zero, the iteration through neighbourhood
    stops.)
00073     returns energy of the structure funct sets as second argument*/
00074 int browse_neighs_pt( char *seq,
00075                      short *ptable,
00076                      short *s,
00077                      short *sl,
00078                      int verbosity_level,
00079                      int shifts,
00080                      int noLP,
00081                      int (*funct) (struct_en*, struct_en*));
00082
00083 int browse_neighs( char *seq,
00084                   char *struc,

```

```

00085             int verbosity_level,
00086             int shifts,
00087             int noLP,
00088             int (*funct) (struct_en*, struct_en*));
00089
00090 #endif
00091
00092 #endif
00093
00094

```

18.142 ViennaRNA/multibranch_loops.h File Reference

Use [ViennaRNA/loops/multibranch.h](#) instead.

Include dependency graph for multibranch_loops.h:

18.142.1 Detailed Description

Use [ViennaRNA/loops/multibranch.h](#) instead.

Deprecated Use [ViennaRNA/loops/multibranch.h](#) instead

18.143 multibranch_loops.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_MULTIBRANCH_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #  ifdef VRNA_WARN_DEPRECATED
00012 #    warning "Including deprecated header file <ViennaRNA/multibranch_loops.h>! Use
00013 #    <ViennaRNA/loops/multibranch.h> instead!"
00014 #  endif
00015 #  include <ViennaRNA/loops/multibranch.h>
00016 #endif
00017 #endif

```

18.144 ViennaRNA/naview.h File Reference

Use ViennaRNA/plotting/naview/naview.h instead.

18.144.1 Detailed Description

Use ViennaRNA/plotting/naview/naview.h instead.

Deprecated Use ViennaRNA/plotting/naview/naview.h instead

18.145 naview.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_NAVIEW_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_NAVIEW_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #  ifdef VRNA_WARN_DEPRECATED
00012 #    warning "Including deprecated header file <ViennaRNA/naview.h>! Use <ViennaRNA/plotting/naview.h>
00013 #    instead!"
00014 #  endif
00015 #  ifdef VRNA_WITH_NAVIEW_LAYOUT
00016 #    include <ViennaRNA/plotting/naview/naview.h>
00017 #  else
00018 #    warning "Naview Layout algorithm is not available in this version!"
00019 #  endif
00020 #endif
00021 #endif

```


18.146 ViennaRNA/landscape/neighbor.h File Reference

Methods to compute the neighbors of an RNA secondary structure.

Include dependency graph for neighbor.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_NEIGHBOR_CHANGE 1`
State indicator for a neighbor that has been changed.
- `#define VRNA_NEIGHBOR_INVALID 2`
State indicator for a neighbor that has been invalidated.
- `#define VRNA_NEIGHBOR_NEW 3`
State indicator for a neighbor that has become newly available.

Typedefs

- `typedef void(* vrna_move_update_f) (vrna_fold_compound_t *fc, vrna_move_t neighbor, unsigned int state, void *data)`
Prototype of the neighborhood update callback.

Functions

- `void vrna_loopidx_update (int *loopidx, const short *pt, int length, const vrna_move_t *m)`
Alters the loopIndices array that was constructed with vrna_loopidx_from_ptable().
- `vrna_move_t * vrna_neighbors (vrna_fold_compound_t *vc, const short *pt, unsigned int options)`
Generate neighbors of a secondary structure.
- `vrna_move_t * vrna_neighbors_successive (const vrna_fold_compound_t *vc, const vrna_move_t *curr↔_move, const short *prev_pt, const vrna_move_t *prev_neighbors, int size_prev_neighbors, int *size↔_neighbors, unsigned int options)`
Generate neighbors of a secondary structure (the fast way)
- `int vrna_move_neighbor_diff_cb (vrna_fold_compound_t *fc, short *ptable, vrna_move_t move, vrna_move_update_f cb, void *data, unsigned int options)`
Apply a move to a secondary structure and indicate which neighbors have changed consequentially.
- `vrna_move_t * vrna_move_neighbor_diff (vrna_fold_compound_t *fc, short *ptable, vrna_move_t move, vrna_move_t **invalid_moves, unsigned int options)`
Apply a move to a secondary structure and indicate which neighbors have changed consequentially.

18.146.1 Detailed Description

Methods to compute the neighbors of an RNA secondary structure.

18.147 neighbor.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_NEIGHBOR_H
00002 #define VIENNA_RNA_PACKAGE_NEIGHBOR_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)
00009 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00010 # elif defined(__GNUC__)
00011 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00012 # else
00013 #   define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
```

```

00018
00126 #include <ViennaRNA/fold_compound.h>
00127 #include <ViennaRNA/landscape/move.h>
00128
00139 typedef void (*vrna_move_update_f) (vrna_fold_compound_t *fc,
00140                                     vrna_move_t          neighbor,
00141                                     unsigned int          state,
00142                                     void                  *data);
00143
00144 DEPRECATED typedef void (vrna_callback_move_update) (vrna_fold_compound_t *fc,
00145                                                      vrna_move_t          neighbor,
00146                                                      unsigned int          state,
00147                                                      void                  *data),
00148              "Use vrna_move_update_f instead!");
00149
00150
00151
00157 #define VRNA_NEIGHBOR_CHANGE    1
00158
00159
00165 #define VRNA_NEIGHBOR_INVALID   2
00166
00167
00173 #define VRNA_NEIGHBOR_NEW       3
00174
00175
00187 void
00188 vrna_loopidx_update(int          *loopidx,
00189                    const short  *pt,
00190                    int          length,
00191                    const vrna_move_t *m);
00192
00193
00209 vrna_move_t *
00210 vrna_neighbors(vrna_fold_compound_t *vc,
00211               const short            *pt,
00212               unsigned int           options);
00213
00214
00236 vrna_move_t *
00237 vrna_neighbors_successive(const vrna_fold_compound_t *vc,
00238                          const vrna_move_t          *curr_move,
00239                          const short                *prev_pt,
00240                          const vrna_move_t          *prev_neighbors,
00241                          int                        size_prev_neighbors,
00242                          int                        *size_neighbors,
00243                          unsigned int                options);
00244
00245
00267 int
00268 vrna_move_neighbor_diff_cb(vrna_fold_compound_t *fc,
00269                           short                *ptable,
00270                           vrna_move_t          move,
00271                           vrna_move_update_f   cb,
00272                           void                  *data,
00273                           unsigned int          options);
00274
00275
00292 vrna_move_t *
00293 vrna_move_neighbor_diff(vrna_fold_compound_t *fc,
00294                        short                *ptable,
00295                        vrna_move_t          move,
00296                        vrna_move_t          **invalid_moves,
00297                        unsigned int          options);
00298
00299
00303 #endif /* VIENNA_RNA_PACKAGE_NEIGHBOR_H */

```

18.148 ViennaRNA/neighbor.h File Reference

Use [ViennaRNA/landscape/neighbor.h](#) instead.

Include dependency graph for neighbor.h:

18.148.1 Detailed Description

Use [ViennaRNA/landscape/neighbor.h](#) instead.

Deprecated Use [ViennaRNA/landscape/neighbor.h](#) instead

18.149 neighbor.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_NEIGHBOR_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_NEIGHBOR_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/neighbor.h>! Use
    <ViennaRNA/landscape/neighbor.h> instead!"
00013 #   endif
00014 #include <ViennaRNA/landscape/neighbor.h>
00015 #include <ViennaRNA/landscape/move.h>
00016 #endif
00017
00018 #endif
```

18.150 pair_mat.h

```
00001 #ifndef VIENNA_RNA_PACKAGE_PAIR_MAT_H
00002 #define VIENNA_RNA_PACKAGE_PAIR_MAT_H
00003
00004 #include <ctype.h>
00005 #include <ViennaRNA/utils/basic.h>
00006 #include <ViennaRNA/fold_vars.h>
00007
00008 #define NBASES 8
00009 /*@nonnull@*/
00010
00011 #ifndef INLINE
00012 #   ifdef __GNUC__
00013 #       define INLINE inline
00014 #   else
00015 #       define INLINE
00016 #   endif
00017 #endif
00018
00019 static const char Law_and_Order[] = "_ACGUTXKI";
00020 static int BP_pair[NBASES][NBASES] =
00021     /* _ A C G U X K I */
00022     { { 0, 0, 0, 0, 0, 0, 0, 0 },
00023       { 0, 0, 0, 0, 5, 0, 0, 5 },
00024       { 0, 0, 0, 1, 0, 0, 0, 0 },
00025       { 0, 0, 2, 0, 3, 0, 0, 0 },
00026       { 0, 6, 0, 4, 0, 0, 0, 6 },
00027       { 0, 0, 0, 0, 0, 0, 2, 0 },
00028       { 0, 0, 0, 0, 0, 1, 0, 0 },
00029       { 0, 6, 0, 0, 5, 0, 0, 0 } };
00030
00031 #define MAXALPHA 20 /* maximal length of alphabet */
00032
00033 static short alias[MAXALPHA + 1];
00034 static int pair[MAXALPHA + 1][MAXALPHA + 1];
00035 /* rtype[pair[i][j]]:=pair[j][i] */
00036 static int rtype[8] = {
00037     0, 2, 1, 4, 3, 6, 5, 7
00038 };
00039
00040 #ifdef _OPENMP
00041 #pragma omp threadprivate(Law_and_Order, BP_pair, alias, pair, rtype)
00042 #endif
00043
00044 /* for backward compatibility */
00045 #define ENCODE(c) encode_char(c)
00046
00047 static INLINE int
00048 encode_char(char c)
00049 {
00050     /* return numerical representation of base used e.g. in pair[][] */
00051     int code;
00052
00053     c = toupper(c);
00054
00055     if (energy_set > 0) {
00056         code = (int)(c - 'A') + 1;
00057     } else {
00058         const char *pos;
00059         pos = strchr(Law_and_Order, c);
00060         if (pos == NULL)
00061             code = 0;
00062         else
00063             code = (int)(pos - Law_and_Order);
00064
00065         if (code > 5)
```

```

00066     code = 0;
00067
00068     if (code > 4)
00069         code--;          /* make T and U equivalent */
00070 }
00071
00072 return code;
00073 }
00074
00075 /*@+boolint +charint@*/
00076 /*@null@*/
00077 extern char *nonstandards;
00078
00079 static INLINE void
00080 make_pair_matrix(void)
00081 {
00082     int i, j;
00083
00084     if (energy_set == 0) {
00085         for (i = 0; i < 5; i++)
00086             alias[i] = (short)i;
00087         alias[5] = 3; /* X <-> G */
00088         alias[6] = 2; /* K <-> C */
00089         alias[7] = 0; /* I <-> default base '@' */
00090         for (i = 0; i < NBASES; i++)
00091             for (j = 0; j < NBASES; j++)
00092                 pair[i][j] = BP_pair[i][j];
00093         if (noGU)
00094             pair[3][4] = pair[4][3] = 0;
00095
00096         if (nonstandards != NULL) {
00097             /* allow nonstandard bp's */
00098             for (i = 0; i < (int)strlen(nonstandards); i += 2)
00099                 pair[encode_char(nonstandards[i])]
00100                     [encode_char(nonstandards[i + 1])] = 7;
00101         }
00102
00103         for (i = 0; i < NBASES; i++)
00104             for (j = 0; j < NBASES; j++)
00105                 rtype[pair[i][j]] = pair[j][i];
00106     } else {
00107         for (i = 0; i <= MAXALPHA; i++)
00108             for (j = 0; j <= MAXALPHA; j++)
00109                 pair[i][j] = 0;
00110         if (energy_set == 1) {
00111             for (i = 1; i < MAXALPHA; ) {
00112                 alias[i++] = 3; /* A <-> G */
00113                 alias[i++] = 2; /* B <-> C */
00114             }
00115             for (i = 1; i < MAXALPHA; i++) {
00116                 pair[i][i + 1] = 2; /* AB <-> GC */
00117                 i++;
00118                 pair[i][i - 1] = 1; /* BA <-> CG */
00119             }
00120         } else if (energy_set == 2) {
00121             for (i = 1; i < MAXALPHA; ) {
00122                 alias[i++] = 1; /* A <-> A */
00123                 alias[i++] = 4; /* B <-> U */
00124             }
00125             for (i = 1; i < MAXALPHA; i++) {
00126                 pair[i][i + 1] = 5; /* AB <-> AU */
00127                 i++;
00128                 pair[i][i - 1] = 6; /* BA <-> UA */
00129             }
00130         } else if (energy_set == 3) {
00131             for (i = 1; i < MAXALPHA - 2; ) {
00132                 alias[i++] = 3; /* A <-> G */
00133                 alias[i++] = 2; /* B <-> C */
00134                 alias[i++] = 1; /* C <-> A */
00135                 alias[i++] = 4; /* D <-> U */
00136             }
00137             for (i = 1; i < MAXALPHA - 2; i++) {
00138                 pair[i][i + 1] = 2; /* AB <-> GC */
00139                 i++;
00140                 pair[i][i - 1] = 1; /* BA <-> CG */
00141                 i++;
00142                 pair[i][i + 1] = 5; /* CD <-> AU */
00143                 i++;
00144                 pair[i][i - 1] = 6; /* DC <-> UA */
00145             }
00146         } else {
00147             vrna_message_error("What energy_set are YOU using??");
00148         }
00149     }
00150
00151     for (i = 0; i <= MAXALPHA; i++)
00152         for (j = 0; j <= MAXALPHA; j++)

```

```

00153         rtype[pair[i][j]] = pair[j][i];
00154     }
00155 }
00156
00157
00158 static INLINE short *
00159 encode_sequence(const char *sequence,
00160                short      how)
00161 {
00162     unsigned int  i, l = (unsigned int)strlen(sequence);
00163     short         *S = (short *)vrna_alloc(sizeof(short) * (l + 2));
00164
00165     switch (how) {
00166         /* standard encoding as always used for S */
00167         case 0:
00168             for (i = 1; i <= l; i++) /* make numerical encoding of sequence */
00169                 S[i] = (short)encode_char(sequence[i - 1]);
00170             S[l + 1] = S[1];
00171             S[0] = (short)1;
00172             break;
00173         /* encoding for mismatches of nonstandard bases (normally used for S1) */
00174         case 1:
00175             for (i = 1; i <= l; i++)
00176                 S[i] = alias[(short)encode_char(sequence[i - 1])];
00177             S[l + 1] = S[1];
00178             S[0] = S[1];
00179             break;
00180     }
00181
00182     return S;
00183 }
00184
00185
00186 #endif /* VIENNA_RNA_PACKAGE_PAIR_MAT_H */

```

18.151 ViennaRNA/params.h File Reference

Use [ViennaRNA/params/basic.h](#) instead.

Include dependency graph for params.h:

18.151.1 Detailed Description

Use [ViennaRNA/params/basic.h](#) instead.

Deprecated Use [ViennaRNA/params/basic.h](#) instead

18.152 params.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/params.h>! Use <ViennaRNA/params/basic.h>
    instead!"
00013 # endif
00014 #include <ViennaRNA/params/basic.h>
00015 #endif
00016
00017 #endif

```

18.153 ViennaRNA/params/1.8.4_epars.h File Reference

Free energy parameters for parameter file conversion.

18.153.1 Detailed Description

Free energy parameters for parameter file conversion.

This file contains the free energy parameters used in ViennaRNAPackage 1.8.4. They are summarized in:

D.H.Mathews, J. Sabina, M. Zuker, D.H. Turner "Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure" JMB, 288, pp 911-940, 1999

Enthalpies taken from:

A. Walter, D Turner, J Kim, M Lyttle, P M"uller, D Mathews, M Zuker "Coaxial stckaing of helices enhances binding of oligoribonucleotides.." PNAS, 91, pp 9218-9222, 1994 D.H. Turner, N. Sugimoto, and S.M. Freier. "RNA Structure Prediction", Ann. Rev. Biophys. Biophys. Chem. 17, 167-192, 1988. John A.Jaeger, Douglas H.Turner, and Michael Zuker. "Improved predictions of secondary structures for RNA", PNAS, 86, 7706-7710, October 1989. L. He, R. Kierzek, J. SantaLucia, A.E. Walter, D.H. Turner "Nearest-Neughbor Parameters for GU Mismatches...." \leftrightarrow Biochemistry 1991, 30 11124-11132 A.E. Peritz, R. Kierzek, N. Sugimoto, D.H. Turner "Thermodynamic Study of Internal Loops in Oligoribonucleotides..." Biochemistry 1991, 30, 6428-6435

18.154 1.8.4_epars.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_OLD_EPARS__
00002 #define VIENNA_RNA_PACKAGE_OLD_EPARS__
00039 #define K0          273.15
00040 #ifdef INF
00041 #undef INF
00042 #endif
00043 #define INF          1000000
00044 #define NBPAIRS      7
00045 #define NST          0    /* Energy for nonstandard stacked pairs */
00046 #define DEF          -50  /* Default terminal mismatch, used for I */
00047                          /* and any non_pairing bases */
00048 #define NSM          0    /* terminal mismatch for non standard pairs */
00049
00050 PRIVATE double Tmeasure_184 = 37 + K0; /* temperature of param measurements */
00051 PRIVATE double lxc37_184    = 107.856; /* parameter for logarithmic loop
00052                                         energy extrapolation */
00053
00054 PRIVATE int stack37_184[NBPAIRS+1][NBPAIRS+1] =
00055 /*
00056 { { INF,  INF,  INF,  INF,  INF,  INF,  INF,  INF},
00057   { INF,  -240, -330, -210, -140, -210, -210, NST},
00058   { INF,  -330, -340, -250, -150, -220, -240, NST},
00059   { INF,  -210, -250,  130,  -50, -140, -130, NST},
00060   { INF,  -140, -150,  -50,   30,  -60, -100, NST},
00061   { INF,  -210, -220, -140,  -60, -110,  -90, NST},
00062   { INF,  -210, -240, -130, -100,  -90, -130, NST},
00063   { INF,   NST,  NST,  NST,  NST,  NST,  NST, NST}};
00064
00065 /* enthalpies (0.01*kcal/mol at 37 C) for stacked pairs */
00066 /* different from mfold-2.3, which uses values from mfold-2.2 */
00067 PRIVATE int enthalpies_184[NBPAIRS+1][NBPAIRS+1] =
00068 /*
00069 { { INF,  INF,  INF,  INF,  INF,  INF,  INF,  INF},
00070   { INF, -1060, -1340, -1210, -560, -1050, -1040, NST},
00071   { INF, -1340, -1490, -1260, -830, -1140, -1240, NST},
00072   { INF, -1210, -1260, -1460, -1350, -880, -1280, NST},
00073   { INF,  -560, -830, -1350, -930, -320, -700, NST},
00074   { INF, -1050, -1140, -880, -320, -940, -680, NST},
00075   { INF, -1040, -1240, -1280, -700, -680, -770, NST},
00076   { INF,   NST,  NST,  NST,  NST,  NST,  NST, NST}};
00077
00078
00079 /* old values are here just for comparison */
00080 PRIVATE int oldhairpin37_184[31] = { /* from ViennaRNA 1.3 */
00081   INF, INF, INF, 410, 490, 440, 470, 500, 510, 520, 531,
00082   542, 551, 560, 568, 575, 582, 589, 595, 601, 606,
00083   611, 616, 621, 626, 630, 634, 638, 642, 646, 650};
00084
00085 PRIVATE int hairpin37_184[31] = {
00086   INF, INF, INF, 570, 560, 560, 540, 590, 560, 640, 650,
00087   660, 670, 678, 686, 694, 701, 707, 713, 719, 725,
00088   730, 735, 740, 744, 749, 753, 757, 761, 765, 769};
00089
00090 PRIVATE int oldbulge37_184[31] = {
00091   INF, 390, 310, 350, 420, 480, 500, 516, 531, 543, 555,
00092   565, 574, 583, 591, 598, 605, 612, 618, 624, 630,
00093   635, 640, 645, 649, 654, 658, 662, 666, 670, 673};
00094
00095 PRIVATE int bulge37_184[31] = {
00096   INF, 380, 280, 320, 360, 400, 440, 459, 470, 480, 490,
00097   500, 510, 519, 527, 534, 541, 548, 554, 560, 565,
00098   571, 576, 580, 585, 589, 594, 598, 602, 605, 609};
00099
00100 PRIVATE int oldinternal_loop37_184[31] = {
00101   INF, INF, 410, 510, 490, 530, 570, 587, 601, 614, 625,
00102   635, 645, 653, 661, 669, 676, 682, 688, 694, 700,
```

```

00103         705, 710, 715, 720, 724, 728, 732, 736, 740, 744};
00104
00105 PRIVATE int internal_loop37_184[31] = {
00106     INF, INF, 410, 510, 170, 180, 200, 220, 230, 240, 250,
00107     260, 270, 278, 286, 294, 301, 307, 313, 319, 325,
00108     330, 335, 340, 345, 349, 353, 357, 361, 365, 369};
00109
00110 /* terminal mismatches */
00111 /* mismatch free energies for interior loops at 37C */
00112 PRIVATE int mismatchI37_184[NBPAIRS+1][5][5] =
00113 { /* @@ */
00114     {{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}},
00115     { /* CG */
00116         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00117         { 0, 0, 0, -110, 0}, /* A@ AA AC AG AU */
00118         { 0, 0, 0, 0, 0}, /* C@ CA CC CG CU */
00119         { 0, -110, 0, 0, 0}, /* G@ GA GC GG GU */
00120         { 0, 0, 0, 0, -70}}, /* U@ UA UC UG UU */
00121     { /* GC */
00122         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00123         { 0, 0, 0, -110, 0}, /* A@ AA AC AG AU */
00124         { 0, 0, 0, 0, 0}, /* C@ CA CC CG CU */
00125         { 0, -110, 0, 0, 0}, /* G@ GA GC GG GU */
00126         { 0, 0, 0, 0, -70}}, /* U@ UA UC UG UU */
00127     { /* GU */
00128         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00129         { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
00130         { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
00131         { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
00132         { 0, 70, 70, 70, 0}}, /* U@ UA UC UG UU */
00133     { /* UG */
00134         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00135         { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
00136         { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
00137         { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
00138         { 0, 70, 70, 70, 0}}, /* U@ UA UC UG UU */
00139     { /* AU */
00140         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00141         { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
00142         { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
00143         { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
00144         { 0, 70, 70, 70, 0}}, /* U@ UA UC UG UU */
00145     { /* UA */
00146         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00147         { 0, 70, 70, -40, 70}, /* A@ AA AC AG AU */
00148         { 0, 70, 70, 70, 70}, /* C@ CA CC CG CU */
00149         { 0, -40, 70, 70, 70}, /* G@ GA GC GG GU */
00150         { 0, 70, 70, 70, 0}}, /* U@ UA UC UG UU */
00151     { /* @@ */
00152         { 90, 90, 90, 90, 90}, { 90, 90, 90, 90, -20}, { 90, 90, 90, 90, 90},
00153         { 90, -20, 90, 90, 90}, { 90, 90, 90, 90, 20}}
00154 };
00155
00156 /* mismatch free energies for hairpins at 37C */
00157 PRIVATE int mismatchH37_184[NBPAIRS+1][5][5] =
00158 { /* @@ */
00159     {{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}},
00160     { /* CG */
00161         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00162         { -90, -150, -150, -140, -180}, /* A@ AA AC AG AU */
00163         { -90, -100, -90, -290, -80}, /* C@ CA CC CG CU */
00164         { -90, -220, -200, -160, -110}, /* G@ GA GC GG GU */
00165         { -90, -170, -140, -180, -200}}, /* U@ UA UC UG UU */
00166     { /* GC */
00167         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00168         { -70, -110, -150, -130, -210}, /* A@ AA AC AG AU */
00169         { -70, -110, -70, -240, -50}, /* C@ CA CC CG CU */
00170         { -70, -240, -290, -140, -120}, /* G@ GA GC GG GU */
00171         { -70, -190, -100, -220, -150}}, /* U@ UA UC UG UU */
00172     { /* GU */
00173         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00174         { 0, 20, -50, -30, -30}, /* A@ AA AC AG AU */
00175         { 0, -10, -20, -150, -20}, /* C@ CA CC CG CU */
00176         { 0, -90, -110, -30, 0}, /* G@ GA GC GG GU */
00177         { 0, -30, -30, -40, -110}}, /* U@ UA UC UG UU */
00178     { /* UG */
00179         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00180         { 0, -50, -30, -60, -50}, /* A@ AA AC AG AU */
00181         { 0, -20, -10, -170, 0}, /* C@ CA CC CG CU */
00182         { 0, -80, -120, -30, -70}, /* G@ GA GC GG GU */
00183         { 0, -60, -10, -60, -80}}, /* U@ UA UC UG UU */
00184     { /* AU */
00185         { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00186         { 0, -30, -50, -30, -30}, /* A@ AA AC AG AU */
00187         { 0, -10, -20, -150, -20}, /* C@ CA CC CG CU */
00188         { 0, -110, -120, -20, 20}, /* G@ GA GC GG GU */
00189         { 0, -30, -30, -60, -110}}, /* U@ UA UC UG UU */

```

```

00190 { /* UA */
00191 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00192 { 0, -50, -30, -60, -50}, /* A@ AA AC AG AU */
00193 { 0, -20, -10, -120, -0}, /* C@ CA CC CG CU */
00194 { 0, -140, -120, -70, -20}, /* G@ GA GC GG GU */
00195 { 0, -30, -10, -50, -80}, /* U@ UA UC UG UU */
00196 { /* @@ */
00197 { 0, 0, 0, 0, 0}, { 0, 0, 0, 0, 0}, { 0, 0, 0, 0, 0},
00198 { 0, 0, 0, 0, 0}, { 0, 0, 0, 0, 0}
00199 };
00200
00201 /* mismatch energies in multiloops */
00202 PRIVATE int mismatchM37_184[NBPAIRS+1][5][5];
00203
00204 /* these are probably junk */
00205 /* mismatch enthalpies for temperature scaling */
00206 PRIVATE int mism_H_184[NBPAIRS+1][5][5] =
00207 { /* no pair */
00208 {{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}},
00209 { /* CG */
00210 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00211 { DEF, -1030, -950, -1030, -1030}, /* A@ AA AC AG AU */
00212 { DEF, -520, -450, -520, -670}, /* C@ CA CC CG CU */
00213 { DEF, -940, -940, -940, -940}, /* G@ GA GC GG GU */
00214 { DEF, -810, -740, -810, -860}, /* U@ UA UC UG UU */
00215 { /* GC */
00216 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00217 { DEF, -520, -880, -560, -880}, /* A@ AA AC AG AU */
00218 { DEF, -720, -310, -310, -390}, /* C@ CA CC CG CU */
00219 { DEF, -710, -740, -620, -740}, /* G@ GA GC GG GU */
00220 { DEF, -500, -500, -500, -570}, /* U@ UA UC UG UU */
00221 { /* GU */
00222 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00223 { DEF, -430, -600, -600, -600}, /* A@ AA AC AG AU */
00224 { DEF, -260, -240, -240, -240}, /* C@ CA CC CG CU */
00225 { DEF, -340, -690, -690, -690}, /* G@ GA GC GG GU */
00226 { DEF, -330, -330, -330, -330}, /* U@ UA UC UG UU */
00227 { /* UG */
00228 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00229 { DEF, -720, -790, -960, -810}, /* A@ AA AC AG AU */
00230 { DEF, -480, -480, -360, -480}, /* C@ CA CC CG CU */
00231 { DEF, -660, -810, -920, -810}, /* G@ GA GC GG GU */
00232 { DEF, -550, -440, -550, -360}, /* U@ UA UC UG UU */
00233 { /* AU */
00234 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00235 { DEF, -430, -600, -600, -600}, /* A@ AA AC AG AU */
00236 { DEF, -260, -240, -240, -240}, /* C@ CA CC CG CU */
00237 { DEF, -340, -690, -690, -690}, /* G@ GA GC GG GU */
00238 { DEF, -330, -330, -330, -330}, /* U@ UA UC UG UU */
00239 { /* UA */
00240 { 0, 0, 0, 0, 0}, /* @@ @A @C @G @U */
00241 { DEF, -400, -630, -890, -590}, /* A@ AA AC AG AU */
00242 { DEF, -430, -510, -200, -180}, /* C@ CA CC CG CU */
00243 { DEF, -380, -680, -890, -680}, /* G@ GA GC GG GU */
00244 { DEF, -280, -140, -280, -140}, /* U@ UA UC UG UU */
00245 { /* nonstandard pair */
00246 {DEF,DEF,DEF,DEF,DEF},{DEF,DEF,DEF,DEF,DEF},{DEF,DEF,DEF,DEF,DEF},
00247 {DEF,DEF,DEF,DEF,DEF},{DEF,DEF,DEF,DEF,DEF}}
00248 };
00249
00250 /* 5' dangling ends (unpaired base stacks on first paired base) */
00251 PRIVATE int dangle5_37_184[NBPAIRS+1][5]=
00252 { /* @ A C G U */
00253 { INF, INF, INF, INF, INF}, /* no pair */
00254 { INF, -50, -30, -20, -10}, /* CG (stacks on C) */
00255 { INF, -20, -30, -0, -0}, /* GC (stacks on G) */
00256 { INF, -30, -30, -40, -20}, /* GU */
00257 { INF, -30, -10, -20, -20}, /* UG */
00258 { INF, -30, -30, -40, -20}, /* AU */
00259 { INF, -30, -10, -20, -20}, /* UA */
00260 { 0, 0, 0, 0, 0} /* @ */
00261 };
00262
00263 /* 3' dangling ends (unpaired base stacks on second paired base) */
00264 PRIVATE int dangle3_37_184[NBPAIRS+1][5]=
00265 { /* @ A C G U */
00266 { INF, INF, INF, INF, INF}, /* no pair */
00267 { INF, -110, -40, -130, -60}, /* CG (stacks on G) */
00268 { INF, -170, -80, -170, -120}, /* GC */
00269 { INF, -70, -10, -70, -10}, /* GU */
00270 { INF, -80, -50, -80, -60}, /* UG */
00271 { INF, -70, -10, -70, -10}, /* AU */
00272 { INF, -80, -50, -80, -60}, /* UA */
00273 { 0, 0, 0, 0, 0} /* @ */
00274 };
00275
00276 /* enthalpies for temperature scaling */

```



```

00277 PRIVATE int dangle3_H_184[NBPAIRS+1][5] =
00278 { /* @ A C G U */
00279   { INF, INF, INF, INF, INF }, /* no pair */
00280   { 0, -740, -280, -640, -360 },
00281   { 0, -900, -410, -860, -750 },
00282   { 0, -740, -240, -720, -490 },
00283   { 0, -490, -90, -550, -230 },
00284   { 0, -570, -70, -580, -220 },
00285   { 0, -490, -90, -550, -230 },
00286   { 0, 0, 0, 0, 0 }
00287 };
00288
00289 PRIVATE int dangle5_H_184[NBPAIRS+1][5] =
00290 { /* @ A C G U */
00291   { INF, INF, INF, INF, INF }, /* no pair */
00292   { 0, -240, 330, 80, -140 },
00293   { 0, -160, 70, -460, -40 },
00294   { 0, 160, 220, 70, 310 },
00295   { 0, -150, 510, 10, 100 },
00296   { 0, 160, 220, 70, 310 },
00297   { 0, -50, 690, -60, -60 },
00298   { 0, 0, 0, 0, 0 }
00299 };
00300
00301
00302 /* constants for linearly destabilizing contributions for multi-loops
00303 F = ML_closing + ML_intern*k + ML_BASE*u */
00304 /* old versions erroneously used ML_intern*(k-1) */
00305 PRIVATE int ML_BASE37_184 = 0;
00306 PRIVATE int ML_closing37_184 = 340;
00307 PRIVATE int ML_intern37_184 = 40;
00308
00309 /* Ninio-correction for asymmetric internal loops with branches n1 and n2 */
00310 /* ninio_energy = min{max_ninio, |n1-n2|*F_ninio[min{4.0, n1, n2}]} */
00311 PRIVATE int MAX_NINIO_184 = 300; /* maximum correction */
00312 PRIVATE int F_ninio37_184[5] = { 0, 40, 50, 20, 10 }; /* only F[2] used */
00313
00314 /* stabilizing contribution due to special hairpins of size 4 (tetraloops) */
00315
00316 PRIVATE char Tetraloops_184[1400] = /* place for up to 200 tetra loops */
00317 "GGGGAC "
00318 "GGUGAC "
00319 "CGAAAG "
00320 "GGAGAC "
00321 "CGCAAG "
00322 "GGA AAC "
00323 "CGGAAG "
00324 "CUUCGG "
00325 "CGUGAG "
00326 "CGAAGG "
00327 "CUACGG "
00328 "GGCAAC "
00329 "CGCGAG "
00330 "UGAGAG "
00331 "CGAGAG "
00332 "AGAAAU "
00333 "CGUAAG "
00334 "CUAACG "
00335 "UGAAAG "
00336 "GGAAGC "
00337 "GGGAAC "
00338 "UGAAAA "
00339 "AGCAAU "
00340 "AGUAAU "
00341 "CGGGAG "
00342 "AGUGAU "
00343 "GGCGAC "
00344 "GGGAGC "
00345 "GUGAAC "
00346 "UGGAAA "
00347 ;
00348
00349 PRIVATE int TETRA_ENERGY37_184[200] = {
00350 -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300, -300,
00351 -250, -250, -200, -200, -200, -200, -200, -200, -200, -150, -150, -150, -150, -150, -150, -150, -150, -150, -150, -150,
00352 -150, -150, -150, -150, -150, -150};
00353
00354 PRIVATE int TETRA_ENTH37_184 = -400;
00355
00356 PRIVATE char Triloops_184[241] = "";
00357
00358 PRIVATE int Triloop_E37_184[40];
00359
00360 /* penalty for AU (or GU) terminating helix */
00361 /* mismatches already contain these */
00362 PRIVATE int TerminalAU_184 = 50;
00363

```

```

00364 /* penalty for forming a bi-molecular duplex */
00365 PRIVATE int DuplexInit_184 = 410;
00366
00367 #endif

```

18.155 ViennaRNA/params/1.8.4_intloops.h File Reference

Free energy parameters for interior loop contributions needed by the parameter file conversion functions.

18.155.1 Detailed Description

Free energy parameters for interior loop contributions needed by the parameter file conversion functions.

18.156 1.8.4_intloops.h

[Go to the documentation of this file.](#)

```

00001
00007 PRIVATE int int11_37_184[NBPAIRS+1][NBPAIRS+1][5][5] =
00008 { /* noPair */ {{0}},
00009 { /* noPair */ {{0}},
00010 /* CG..CG */
00011 {{ 110, 110, 110, 110, 110},
00012 { 110, 110, 40, 40, 40},
00013 { 110, 40, 40, 40, 40},
00014 { 110, 40, 40, -140, 40},
00015 { 110, 40, 40, 40, 40}
00016 },
00017 /* CG..GC */
00018 {{ 110, 110, 110, 110, 110},
00019 { 110, 40, -40, 40, 40},
00020 { 110, 30, 50, 40, 50},
00021 { 110, -10, 40, -170, 40},
00022 { 110, 40, 0, 40, -30}
00023 },
00024 /* CG..GU */
00025 {{ 110, 110, 110, 110, 110},
00026 { 110, 110, 110, 110, 110},
00027 { 110, 110, 110, 110, 110},
00028 { 110, 110, 110, -100, 110},
00029 { 110, 110, 110, 110, 110}
00030 },
00031 /* CG..UG */
00032 {{ 110, 110, 110, 110, 110},
00033 { 110, 110, 110, 110, 110},
00034 { 110, 110, 110, 110, 110},
00035 { 110, 110, 110, -100, 110},
00036 { 110, 110, 110, 110, 110}
00037 },
00038 /* CG..AU */
00039 {{ 110, 110, 110, 110, 110},
00040 { 110, 110, 110, 110, 110},
00041 { 110, 110, 110, 110, 110},
00042 { 110, 110, 110, -100, 110},
00043 { 110, 110, 110, 110, 110}
00044 },
00045 /* CG..UA */
00046 {{ 110, 110, 110, 110, 110},
00047 { 110, 110, 110, 110, 110},
00048 { 110, 110, 110, 110, 110},
00049 { 110, 110, 110, -100, 110},
00050 { 110, 110, 110, 110, 110}
00051 },
00052 /* CG..?? */
00053 {{ 110, 110, 110, 110, 110},
00054 { 110, 110, 110, 110, 110},
00055 { 110, 110, 110, 110, 110},
00056 { 110, 110, 110, 110, 110},
00057 { 110, 110, 110, 110, 110}
00058 },
00059 },
00060 { /* noPair */ {{0}},
00061 /* GC..CG */
00062 {{ 110, 110, 110, 110, 110},
00063 { 110, 40, 30, -10, 40},
00064 { 110, -40, 50, 40, 0},
00065 { 110, 40, 40, -170, 40},
00066 { 110, 40, 50, 40, -30}
00067 },
00068 /* GC..GC */

```

```

00069 {{ 110, 110, 110, 110, 110},
00070 { 110, 80, 40, 40, 40},
00071 { 110, 40, 40, 40, 40},
00072 { 110, 40, 40, -210, 40},
00073 { 110, 40, 40, 40, -70}
00074 },
00075 /* GC..GU */
00076 {{ 110, 110, 110, 110, 110},
00077 { 110, 110, 110, 110, 110},
00078 { 110, 110, 110, 110, 110},
00079 { 110, 110, 110, -100, 110},
00080 { 110, 110, 110, 110, 110}
00081 },
00082 /* GC..UG */
00083 {{ 110, 110, 110, 110, 110},
00084 { 110, 110, 110, 110, 110},
00085 { 110, 110, 110, 110, 110},
00086 { 110, 110, 110, -100, 110},
00087 { 110, 110, 110, 110, 110}
00088 },
00089 /* GC..AU */
00090 {{ 110, 110, 110, 110, 110},
00091 { 110, 110, 110, 110, 110},
00092 { 110, 110, 110, 110, 110},
00093 { 110, 110, 110, -100, 110},
00094 { 110, 110, 110, 110, 100}
00095 },
00096 /* GC..UA */
00097 {{ 110, 110, 110, 110, 110},
00098 { 110, 110, 110, 110, 110},
00099 { 110, 110, 110, 110, 110},
00100 { 110, 110, 110, -100, 110},
00101 { 110, 110, 110, 110, 110}
00102 },
00103 /* GC..?? */
00104 {{ 110, 110, 110, 110, 110},
00105 { 110, 110, 110, 110, 110},
00106 { 110, 110, 110, 110, 110},
00107 { 110, 110, 110, 110, 110},
00108 { 110, 110, 110, 110, 110}
00109 },
00110 },
00111 { /* noPair */ {{0}},
00112 /* GU..CG */
00113 {{ 110, 110, 110, 110, 110},
00114 { 110, 110, 110, 110, 110},
00115 { 110, 110, 110, 110, 110},
00116 { 110, 110, 110, -100, 110},
00117 { 110, 110, 110, 110, 110}
00118 },
00119 /* GU..GC */
00120 {{ 110, 110, 110, 110, 110},
00121 { 110, 110, 110, 110, 110},
00122 { 110, 110, 110, 110, 110},
00123 { 110, 110, 110, -100, 110},
00124 { 110, 110, 110, 110, 110}
00125 },
00126 /* GU..GU */
00127 {{ 170, 170, 170, 170, 170},
00128 { 170, 170, 170, 170, 170},
00129 { 170, 170, 170, 170, 170},
00130 { 170, 170, 170, -40, 170},
00131 { 170, 170, 170, 170, 170}
00132 },
00133 /* GU..UG */
00134 {{ 170, 170, 170, 170, 170},
00135 { 170, 170, 170, 170, 170},
00136 { 170, 170, 170, 170, 170},
00137 { 170, 170, 170, -40, 170},
00138 { 170, 170, 170, 170, 170}
00139 },
00140 /* GU..AU */
00141 {{ 170, 170, 170, 170, 170},
00142 { 170, 170, 170, 170, 170},
00143 { 170, 170, 170, 170, 170},
00144 { 170, 170, 170, -40, 170},
00145 { 170, 170, 170, 170, 170}
00146 },
00147 /* GU..UA */
00148 {{ 170, 170, 170, 170, 170},
00149 { 170, 170, 170, 170, 170},
00150 { 170, 170, 170, 170, 170},
00151 { 170, 170, 170, -40, 170},
00152 { 170, 170, 170, 170, 170}
00153 },
00154 /* GU..?? */
00155 {{ 170, 170, 170, 170, 170},

```

```
00156 { 170, 170, 170, 170, 170},
00157 { 170, 170, 170, 170, 170},
00158 { 170, 170, 170, 170, 170},
00159 { 170, 170, 170, 170, 170}
00160 }
00161 },
00162 { /* noPair */ {{0}},
00163 /* UG..CG */
00164 {{ 110, 110, 110, 110, 110},
00165 { 110, 110, 110, 110, 110},
00166 { 110, 110, 110, 110, 110},
00167 { 110, 110, 110, -100, 110},
00168 { 110, 110, 110, 110, 110}
00169 },
00170 /* UG..GC */
00171 {{ 110, 110, 110, 110, 110},
00172 { 110, 110, 110, 110, 110},
00173 { 110, 110, 110, 110, 110},
00174 { 110, 110, 110, -100, 110},
00175 { 110, 110, 110, 110, 110}
00176 },
00177 /* UG..GU */
00178 {{ 170, 170, 170, 170, 170},
00179 { 170, 170, 170, 170, 170},
00180 { 170, 170, 170, 170, 170},
00181 { 170, 170, 170, -40, 170},
00182 { 170, 170, 170, 170, 170}
00183 },
00184 /* UG..UG */
00185 {{ 170, 170, 170, 170, 170},
00186 { 170, 170, 170, 170, 170},
00187 { 170, 170, 170, 170, 170},
00188 { 170, 170, 170, -40, 170},
00189 { 170, 170, 170, 170, 170}
00190 },
00191 /* UG..AU */
00192 {{ 170, 170, 170, 170, 170},
00193 { 170, 170, 170, 170, 170},
00194 { 170, 170, 170, 170, 170},
00195 { 170, 170, 170, -40, 170},
00196 { 170, 170, 170, 170, 170}
00197 },
00198 /* UG..UA */
00199 {{ 170, 170, 170, 170, 170},
00200 { 170, 170, 170, 170, 170},
00201 { 170, 170, 170, 170, 170},
00202 { 170, 170, 170, -40, 170},
00203 { 170, 170, 170, 170, 170}
00204 },
00205 /* UG..?? */
00206 {{ 170, 170, 170, 170, 170},
00207 { 170, 170, 170, 170, 170},
00208 { 170, 170, 170, 170, 170},
00209 { 170, 170, 170, 170, 170},
00210 { 170, 170, 170, 170, 170}
00211 }
00212 },
00213 { /* noPair */ {{0}},
00214 /* AU..CG */
00215 {{ 110, 110, 110, 110, 110},
00216 { 110, 110, 110, 110, 110},
00217 { 110, 110, 110, 110, 110},
00218 { 110, 110, 110, -100, 110},
00219 { 110, 110, 110, 110, 110}
00220 },
00221 /* AU..GC */
00222 {{ 110, 110, 110, 110, 110},
00223 { 110, 110, 110, 110, 110},
00224 { 110, 110, 110, 110, 110},
00225 { 110, 110, 110, -100, 110},
00226 { 110, 110, 110, 110, 100}
00227 },
00228 /* AU..GU */
00229 {{ 170, 170, 170, 170, 170},
00230 { 170, 170, 170, 170, 170},
00231 { 170, 170, 170, 170, 170},
00232 { 170, 170, 170, -40, 170},
00233 { 170, 170, 170, 170, 170}
00234 },
00235 /* AU..UG */
00236 {{ 170, 170, 170, 170, 170},
00237 { 170, 170, 170, 170, 170},
00238 { 170, 170, 170, 170, 170},
00239 { 170, 170, 170, -40, 170},
00240 { 170, 170, 170, 170, 170}
00241 },
00242 /* AU..AU */
```

```
00243 {{ 170, 170, 170, 170, 170},
00244 { 170, 170, 170, 170, 170},
00245 { 170, 170, 170, 170, 170},
00246 { 170, 170, 170, -40, 170},
00247 { 170, 170, 170, 170, 120}
00248 },
00249 /* AU..UA */
00250 {{ 170, 170, 170, 170, 170},
00251 { 170, 170, 170, 170, 170},
00252 { 170, 170, 170, 170, 170},
00253 { 170, 170, 170, -40, 170},
00254 { 170, 170, 170, 170, 150}
00255 },
00256 /* AU..?? */
00257 {{ 170, 170, 170, 170, 170},
00258 { 170, 170, 170, 170, 170},
00259 { 170, 170, 170, 170, 170},
00260 { 170, 170, 170, 170, 170},
00261 { 170, 170, 170, 170, 170}
00262 },
00263 },
00264 { /* noPair */ {{0}},
00265 /* UA..CG */
00266 {{ 110, 110, 110, 110, 110},
00267 { 110, 110, 110, 110, 110},
00268 { 110, 110, 110, 110, 110},
00269 { 110, 110, 110, -100, 110},
00270 { 110, 110, 110, 110, 110}
00271 },
00272 /* UA..GC */
00273 {{ 110, 110, 110, 110, 110},
00274 { 110, 110, 110, 110, 110},
00275 { 110, 110, 110, 110, 110},
00276 { 110, 110, 110, -100, 110},
00277 { 110, 110, 110, 110, 110}
00278 },
00279 /* UA..GU */
00280 {{ 170, 170, 170, 170, 170},
00281 { 170, 170, 170, 170, 170},
00282 { 170, 170, 170, 170, 170},
00283 { 170, 170, 170, -40, 170},
00284 { 170, 170, 170, 170, 170}
00285 },
00286 /* UA..UG */
00287 {{ 170, 170, 170, 170, 170},
00288 { 170, 170, 170, 170, 170},
00289 { 170, 170, 170, 170, 170},
00290 { 170, 170, 170, -40, 170},
00291 { 170, 170, 170, 170, 170}
00292 },
00293 /* UA..AU */
00294 {{ 170, 170, 170, 170, 170},
00295 { 170, 170, 170, 170, 170},
00296 { 170, 170, 170, 170, 170},
00297 { 170, 170, 170, -40, 170},
00298 { 170, 170, 170, 170, 150}
00299 },
00300 /* UA..UA */
00301 {{ 170, 170, 170, 170, 170},
00302 { 170, 170, 170, 170, 170},
00303 { 170, 170, 170, 170, 170},
00304 { 170, 170, 170, -40, 170},
00305 { 170, 170, 170, 170, 180}
00306 },
00307 /* UA..?? */
00308 {{ 170, 170, 170, 170, 170},
00309 { 170, 170, 170, 170, 170},
00310 { 170, 170, 170, 170, 170},
00311 { 170, 170, 170, 170, 170},
00312 { 170, 170, 170, 170, 170}
00313 },
00314 },
00315 { /* noPair */ {{0}},
00316 /* ??..CG */
00317 {{ 110, 110, 110, 110, 110},
00318 { 110, 110, 110, 110, 110},
00319 { 110, 110, 110, 110, 110},
00320 { 110, 110, 110, 110, 110},
00321 { 110, 110, 110, 110, 110}
00322 },
00323 /* ??..GC */
00324 {{ 110, 110, 110, 110, 110},
00325 { 110, 110, 110, 110, 110},
00326 { 110, 110, 110, 110, 110},
00327 { 110, 110, 110, 110, 110},
00328 { 110, 110, 110, 110, 110}
00329 },
```

```

00330 /* ??..GU */
00331 {{ 170, 170, 170, 170, 170},
00332 { 170, 170, 170, 170, 170},
00333 { 170, 170, 170, 170, 170},
00334 { 170, 170, 170, 170, 170},
00335 { 170, 170, 170, 170, 170}
00336 },
00337 /* ??..UG */
00338 {{ 170, 170, 170, 170, 170},
00339 { 170, 170, 170, 170, 170},
00340 { 170, 170, 170, 170, 170},
00341 { 170, 170, 170, 170, 170},
00342 { 170, 170, 170, 170, 170}
00343 },
00344 /* ??..AU */
00345 {{ 170, 170, 170, 170, 170},
00346 { 170, 170, 170, 170, 170},
00347 { 170, 170, 170, 170, 170},
00348 { 170, 170, 170, 170, 170},
00349 { 170, 170, 170, 170, 170}
00350 },
00351 /* ??..UA */
00352 {{ 170, 170, 170, 170, 170},
00353 { 170, 170, 170, 170, 170},
00354 { 170, 170, 170, 170, 170},
00355 { 170, 170, 170, 170, 170},
00356 { 170, 170, 170, 170, 170}
00357 },
00358 /* ??..?? */
00359 {{ 170, 170, 170, 170, 170},
00360 { 170, 170, 170, 170, 170},
00361 { 170, 170, 170, 170, 170},
00362 { 170, 170, 170, 170, 170},
00363 { 170, 170, 170, 170, 170}
00364 }
00365 }
00366 };
00367
00368 PRIVATE int int11_H_184[NBPAIRS+1][NBPAIRS+1][5][5] =
00369 /* GC..GC */
00370 { /* noPair */ {{0}},
00371 { /* noPair */ {0}},
00372 { { 0, 0, 0, 0, 0},
00373 { 0, 0, 0, 0, 0},
00374 { 0, 0, 0, 0, 0},
00375 { 0, 0, 0, 0, 0},
00376 { 0, 0, 0, 0, 0}},
00377 /* GC..CG */
00378 { { 0, 0, 0, 0, 0},
00379 { 0, 0, 0, 0, 0},
00380 { 0, 0, 0, 0, 0},
00381 { 0, 0, 0, 0, 0},
00382 { 0, 0, 0, 0, 0}},
00383 /* GC..GU */
00384 { { 0, 0, 0, 0, 0},
00385 { 0, 0, 0, 0, 0},
00386 { 0, 0, 0, 0, 0},
00387 { 0, 0, 0, 0, 0},
00388 { 0, 0, 0, 0, 0}},
00389 /* GC..UG */
00390 { { 0, 0, 0, 0, 0},
00391 { 0, 0, 0, 0, 0},
00392 { 0, 0, 0, 0, 0},
00393 { 0, 0, 0, 0, 0},
00394 { 0, 0, 0, 0, 0}},
00395 /* GC..AU */
00396 { { 0, 0, 0, 0, 0},
00397 { 0, 0, 0, 0, 0},
00398 { 0, 0, 0, 0, 0},
00399 { 0, 0, 0, 0, 0},
00400 { 0, 0, 0, 0, 0}},
00401 /* GC..UA */
00402 { { 0, 0, 0, 0, 0},
00403 { 0, 0, 0, 0, 0},
00404 { 0, 0, 0, 0, 0},
00405 { 0, 0, 0, 0, 0},
00406 { 0, 0, 0, 0, 0}},
00407 /* GC..@ */
00408 { { 0, 0, 0, 0, 0},
00409 { 0, 0, 0, 0, 0},
00410 { 0, 0, 0, 0, 0},
00411 { 0, 0, 0, 0, 0},
00412 { 0, 0, 0, 0, 0}},
00413 /* CG..GC */
00414 { /* noPair */ {{0}},
00415 { { 0, 0, 0, 0, 0},
00416 { 0, 0, 0, 0, 0},

```

```
00417 { 0, 0, 0, 0, 0},
00418 { 0, 0, 0, 0, 0},
00419 { 0, 0, 0, 0, 0}},
00420 /* CG..CG */
00421 { { 0, 0, 0, 0, 0},
00422 { 0, 0, 0, 0, 0},
00423 { 0, 0, 0, 0, 0},
00424 { 0, 0, 0, 0, 0},
00425 { 0, 0, 0, 0, 0}},
00426 /* CG..GU */
00427 { { 0, 0, 0, 0, 0},
00428 { 0, 0, 0, 0, 0},
00429 { 0, 0, 0, 0, 0},
00430 { 0, 0, 0, 0, 0},
00431 { 0, 0, 0, 0, 0}},
00432 /* CG..UG */
00433 { { 0, 0, 0, 0, 0},
00434 { 0, 0, 0, 0, 0},
00435 { 0, 0, 0, 0, 0},
00436 { 0, 0, 0, 0, 0},
00437 { 0, 0, 0, 0, 0}},
00438 /* CG..AU */
00439 { { 0, 0, 0, 0, 0},
00440 { 0, 0, 0, 0, 0},
00441 { 0, 0, 0, 0, 0},
00442 { 0, 0, 0, 0, 0},
00443 { 0, 0, 0, 0, 0}},
00444 /* CG..UA */
00445 { { 0, 0, 0, 0, 0},
00446 { 0, 0, 0, 0, 0},
00447 { 0, 0, 0, 0, 0},
00448 { 0, 0, 0, 0, 0},
00449 { 0, 0, 0, 0, 0}},
00450 /* CG.. @ */
00451 { { 0, 0, 0, 0, 0},
00452 { 0, 0, 0, 0, 0},
00453 { 0, 0, 0, 0, 0},
00454 { 0, 0, 0, 0, 0},
00455 { 0, 0, 0, 0, 0}}},
00456 /* GU..GC */
00457 { /* noPair */ {0}},
00458 { { 0, 0, 0, 0, 0},
00459 { 0, 0, 0, 0, 0},
00460 { 0, 0, 0, 0, 0},
00461 { 0, 0, 0, 0, 0},
00462 { 0, 0, 0, 0, 0}},
00463 /* GU..CG */
00464 { { 0, 0, 0, 0, 0},
00465 { 0, 0, 0, 0, 0},
00466 { 0, 0, 0, 0, 0},
00467 { 0, 0, 0, 0, 0},
00468 { 0, 0, 0, 0, 0}},
00469 /* GU..GU */
00470 { { 0, 0, 0, 0, 0},
00471 { 0, 0, 0, 0, 0},
00472 { 0, 0, 0, 0, 0},
00473 { 0, 0, 0, 0, 0},
00474 { 0, 0, 0, 0, 0}},
00475 /* GU..UG */
00476 { { 0, 0, 0, 0, 0},
00477 { 0, 0, 0, 0, 0},
00478 { 0, 0, 0, 0, 0},
00479 { 0, 0, 0, 0, 0},
00480 { 0, 0, 0, 0, 0}},
00481 /* GU..AU */
00482 { { 0, 0, 0, 0, 0},
00483 { 0, 0, 0, 0, 0},
00484 { 0, 0, 0, 0, 0},
00485 { 0, 0, 0, 0, 0},
00486 { 0, 0, 0, 0, 0}},
00487 /* GU..UA */
00488 { { 0, 0, 0, 0, 0},
00489 { 0, 0, 0, 0, 0},
00490 { 0, 0, 0, 0, 0},
00491 { 0, 0, 0, 0, 0},
00492 { 0, 0, 0, 0, 0}},
00493 /* GU.. @ */
00494 { { 0, 0, 0, 0, 0},
00495 { 0, 0, 0, 0, 0},
00496 { 0, 0, 0, 0, 0},
00497 { 0, 0, 0, 0, 0},
00498 { 0, 0, 0, 0, 0}}},
00499 /* UG..GC */
00500 { /* noPair */ {0}},
00501 { { 0, 0, 0, 0, 0},
00502 { 0, 0, 0, 0, 0},
00503 { 0, 0, 0, 0, 0},
```

```
00504 { 0, 0, 0, 0, 0},
00505 { 0, 0, 0, 0, 0}},
00506 /* UG..CG */
00507 { { 0, 0, 0, 0, 0},
00508 { 0, 0, 0, 0, 0},
00509 { 0, 0, 0, 0, 0},
00510 { 0, 0, 0, 0, 0},
00511 { 0, 0, 0, 0, 0}},
00512 /* UG..GU */
00513 { { 0, 0, 0, 0, 0},
00514 { 0, 0, 0, 0, 0},
00515 { 0, 0, 0, 0, 0},
00516 { 0, 0, 0, 0, 0},
00517 { 0, 0, 0, 0, 0}},
00518 /* UG..UG */
00519 { { 0, 0, 0, 0, 0},
00520 { 0, 0, 0, 0, 0},
00521 { 0, 0, 0, 0, 0},
00522 { 0, 0, 0, 0, 0},
00523 { 0, 0, 0, 0, 0}},
00524 /* UG..AU */
00525 { { 0, 0, 0, 0, 0},
00526 { 0, 0, 0, 0, 0},
00527 { 0, 0, 0, 0, 0},
00528 { 0, 0, 0, 0, 0},
00529 { 0, 0, 0, 0, 0}},
00530 /* UG..UA */
00531 { { 0, 0, 0, 0, 0},
00532 { 0, 0, 0, 0, 0},
00533 { 0, 0, 0, 0, 0},
00534 { 0, 0, 0, 0, 0},
00535 { 0, 0, 0, 0, 0}},
00536 /* UG.. @ */
00537 { { 0, 0, 0, 0, 0},
00538 { 0, 0, 0, 0, 0},
00539 { 0, 0, 0, 0, 0},
00540 { 0, 0, 0, 0, 0},
00541 { 0, 0, 0, 0, 0}}},
00542 /* AU..GC */
00543 { /* noPair */ {0}},
00544 { { 0, 0, 0, 0, 0},
00545 { 0, 0, 0, 0, 0},
00546 { 0, 0, 0, 0, 0},
00547 { 0, 0, 0, 0, 0},
00548 { 0, 0, 0, 0, 0}},
00549 /* AU..CG */
00550 { { 0, 0, 0, 0, 0},
00551 { 0, 0, 0, 0, 0},
00552 { 0, 0, 0, 0, 0},
00553 { 0, 0, 0, 0, 0},
00554 { 0, 0, 0, 0, 0}},
00555 /* AU..GU */
00556 { { 0, 0, 0, 0, 0},
00557 { 0, 0, 0, 0, 0},
00558 { 0, 0, 0, 0, 0},
00559 { 0, 0, 0, 0, 0},
00560 { 0, 0, 0, 0, 0}},
00561 /* AU..UG */
00562 { { 0, 0, 0, 0, 0},
00563 { 0, 0, 0, 0, 0},
00564 { 0, 0, 0, 0, 0},
00565 { 0, 0, 0, 0, 0},
00566 { 0, 0, 0, 0, 0}},
00567 /* AU..AU */
00568 { { 0, 0, 0, 0, 0},
00569 { 0, 0, 0, 0, 0},
00570 { 0, 0, 0, 0, 0},
00571 { 0, 0, 0, 0, 0},
00572 { 0, 0, 0, 0, 0}},
00573 /* AU..UA */
00574 { { 0, 0, 0, 0, 0},
00575 { 0, 0, 0, 0, 0},
00576 { 0, 0, 0, 0, 0},
00577 { 0, 0, 0, 0, 0},
00578 { 0, 0, 0, 0, 0}},
00579 /* AU.. @ */
00580 { { 0, 0, 0, 0, 0},
00581 { 0, 0, 0, 0, 0},
00582 { 0, 0, 0, 0, 0},
00583 { 0, 0, 0, 0, 0},
00584 { 0, 0, 0, 0, 0}}},
00585 /* UA..GC */
00586 { /* noPair */ {0}},
00587 { { 0, 0, 0, 0, 0},
00588 { 0, 0, 0, 0, 0},
00589 { 0, 0, 0, 0, 0},
00590 { 0, 0, 0, 0, 0}},
```



```

00591 { 0, 0, 0, 0, 0, 0}},
00592 /* UA..CG */
00593 { { 0, 0, 0, 0, 0, 0}},
00594 { 0, 0, 0, 0, 0, 0}},
00595 { 0, 0, 0, 0, 0, 0}},
00596 { 0, 0, 0, 0, 0, 0}},
00597 { 0, 0, 0, 0, 0, 0}},
00598 /* UA..GU */
00599 { { 0, 0, 0, 0, 0, 0}},
00600 { 0, 0, 0, 0, 0, 0}},
00601 { 0, 0, 0, 0, 0, 0}},
00602 { 0, 0, 0, 0, 0, 0}},
00603 { 0, 0, 0, 0, 0, 0}},
00604 /* UA..UG */
00605 { { 0, 0, 0, 0, 0, 0}},
00606 { 0, 0, 0, 0, 0, 0}},
00607 { 0, 0, 0, 0, 0, 0}},
00608 { 0, 0, 0, 0, 0, 0}},
00609 { 0, 0, 0, 0, 0, 0}},
00610 /* UA..AU */
00611 { { 0, 0, 0, 0, 0, 0}},
00612 { 0, 0, 0, 0, 0, 0}},
00613 { 0, 0, 0, 0, 0, 0}},
00614 { 0, 0, 0, 0, 0, 0}},
00615 { 0, 0, 0, 0, 0, 0}},
00616 /* UA..UA */
00617 { { 0, 0, 0, 0, 0, 0}},
00618 { 0, 0, 0, 0, 0, 0}},
00619 { 0, 0, 0, 0, 0, 0}},
00620 { 0, 0, 0, 0, 0, 0}},
00621 { 0, 0, 0, 0, 0, 0}},
00622 /* UA.. @ */
00623 { { 0, 0, 0, 0, 0, 0}},
00624 { 0, 0, 0, 0, 0, 0}},
00625 { 0, 0, 0, 0, 0, 0}},
00626 { 0, 0, 0, 0, 0, 0}},
00627 { 0, 0, 0, 0, 0, 0}}},
00628 /* @..GC */
00629 { /* noPair */ {{0}},
00630 { { 0, 0, 0, 0, 0, 0}},
00631 { 0, 0, 0, 0, 0, 0}},
00632 { 0, 0, 0, 0, 0, 0}},
00633 { 0, 0, 0, 0, 0, 0}},
00634 { 0, 0, 0, 0, 0, 0}},
00635 /* @..CG */
00636 { { 0, 0, 0, 0, 0, 0}},
00637 { 0, 0, 0, 0, 0, 0}},
00638 { 0, 0, 0, 0, 0, 0}},
00639 { 0, 0, 0, 0, 0, 0}},
00640 { 0, 0, 0, 0, 0, 0}},
00641 /* @..GU */
00642 { { 0, 0, 0, 0, 0, 0}},
00643 { 0, 0, 0, 0, 0, 0}},
00644 { 0, 0, 0, 0, 0, 0}},
00645 { 0, 0, 0, 0, 0, 0}},
00646 { 0, 0, 0, 0, 0, 0}},
00647 /* @..UG */
00648 { { 0, 0, 0, 0, 0, 0}},
00649 { 0, 0, 0, 0, 0, 0}},
00650 { 0, 0, 0, 0, 0, 0}},
00651 { 0, 0, 0, 0, 0, 0}},
00652 { 0, 0, 0, 0, 0, 0}},
00653 /* @..AU */
00654 { { 0, 0, 0, 0, 0, 0}},
00655 { 0, 0, 0, 0, 0, 0}},
00656 { 0, 0, 0, 0, 0, 0}},
00657 { 0, 0, 0, 0, 0, 0}},
00658 { 0, 0, 0, 0, 0, 0}},
00659 /* @..UA */
00660 { { 0, 0, 0, 0, 0, 0}},
00661 { 0, 0, 0, 0, 0, 0}},
00662 { 0, 0, 0, 0, 0, 0}},
00663 { 0, 0, 0, 0, 0, 0}},
00664 { 0, 0, 0, 0, 0, 0}},
00665 /* @.. @ */
00666 { { 0, 0, 0, 0, 0, 0}},
00667 { 0, 0, 0, 0, 0, 0}},
00668 { 0, 0, 0, 0, 0, 0}},
00669 { 0, 0, 0, 0, 0, 0}},
00670 { 0, 0, 0, 0, 0, 0}}}};
00671
00672 PRIVATE int int21_37_184[NBPAIRS+1][NBPAIRS+1][5][5][5] =
00673 { /* noPair */ {{{0}}}},
00674 { /* noPair */ {{{0}}}},
00675 {
00676 /* CG.@..GC */
00677 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,

```

```
550, 550},{ 550, 550, 550, 550, 550}},
00678 /* CG.A..GC */
00679 {{ 550, 550, 550, 550, 550},{ 550, 240, 220, 160, 400},{ 550, 210, 170, 160, 400},{ 550, 100, 60,
40, 400},{ 550, 400, 400, 400, 400}},
00680 /* CG.C..GC */
00681 {{ 550, 550, 550, 550, 550},{ 550, 230, 220, 400, 220},{ 550, 220, 250, 400, 220},{ 550, 400, 400,
400, 400},{ 550, 250, 190, 400, 220}},
00682 /* CG.G..GC */
00683 {{ 550, 550, 550, 550, 550},{ 550, 170, 400, 80, 400},{ 550, 400, 400, 400, 400},{ 550, 80, 400,
220, 400},{ 550, 400, 400, 400, 400}},
00684 /* CG.U..GC */
00685 {{ 550, 550, 550, 550, 550},{ 550, 400, 400, 400, 400},{ 550, 400, 220, 400, 130},{ 550, 400, 400,
400, 400},{ 550, 400, 170, 400, 120}}
00686 },
00687 {
00688 /* CG.@..CG */
00689 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00690 /* CG.A..CG */
00691 {{ 550, 550, 550, 550, 550},{ 550, 230, 220, 110, 400},{ 550, 210, 170, 160, 400},{ 550, 80, 60,
40, 400},{ 550, 400, 400, 400, 400}},
00692 /* CG.C..CG */
00693 {{ 550, 550, 550, 550, 550},{ 550, 230, 220, 400, 220},{ 550, 220, 250, 400, 220},{ 550, 400, 400,
400, 400},{ 550, 250, 190, 400, 220}},
00694 /* CG.G..CG */
00695 {{ 550, 550, 550, 550, 550},{ 550, 170, 400, 80, 400},{ 550, 400, 400, 400, 400},{ 550, 80, 400,
220, 400},{ 550, 400, 400, 400, 400}},
00696 /* CG.U..CG */
00697 {{ 550, 550, 550, 550, 550},{ 550, 400, 400, 400, 400},{ 550, 400, 220, 400, 150},{ 550, 400, 400,
400, 400},{ 550, 400, 170, 400, 120}}
00698 },
00699 {
00700 /* CG.@..UG */
00701 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00702 /* CG.A..UG */
00703 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
00704 /* CG.C..UG */
00705 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
00706 /* CG.G..UG */
00707 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
00708 /* CG.U..UG */
00709 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
00710 },
00711 {
00712 /* CG.@..GU */
00713 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00714 /* CG.A..GU */
00715 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
00716 /* CG.C..GU */
00717 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
00718 /* CG.G..GU */
00719 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
00720 /* CG.U..GU */
00721 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
00722 },
00723 {
00724 /* CG.@..UA */
00725 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00726 /* CG.A..UA */
00727 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
00728 /* CG.C..UA */
00729 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
00730 /* CG.G..UA */
00731 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
00732 /* CG.U..UA */
00733 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
00734 },
00735 {
00736 /* CG.@..AU */
00737 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00738 /* CG.A..AU */
```

```
00739 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
00740 120, 480},{ 550, 480, 480, 480, 480}},
00741 /* CG.C..AU */
00741 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
00742 480, 480},{ 550, 330, 270, 480, 300}},
00743 /* CG.G..AU */
00743 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
00744 300, 480},{ 550, 480, 480, 480, 480}},
00745 /* CG.U..AU */
00745 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
00746 480, 480},{ 550, 480, 250, 480, 200}}
00746 },
00747 {
00748 /* CG.@..?? */
00749 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00750 550, 550},{ 550, 550, 550, 550, 550}},
00751 /* CG.A..?? */
00751 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00752 550, 550},{ 550, 550, 550, 550, 550}},
00753 /* CG.C..?? */
00753 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00754 550, 550},{ 550, 550, 550, 550, 550}},
00755 /* CG.G..?? */
00755 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00756 550, 550},{ 550, 550, 550, 550, 550}},
00757 /* CG.U..?? */
00757 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00758 550, 550},{ 550, 550, 550, 550, 550}}
00758 },
00759 },
00760 { /* noPair */ {{0}}},
00761 {
00762 /* GC.@..GC */
00763 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00764 550, 550},{ 550, 550, 550, 550, 550}},
00765 /* GC.A..GC */
00765 {{ 550, 550, 550, 550, 550},{ 550, 250, 220, 210, 400},{ 550, 210, 170, 160, 400},{ 550, 120, 60,
00766 40, 400},{ 550, 400, 400, 400, 400}},
00767 /* GC.C..GC */
00767 {{ 550, 550, 550, 550, 550},{ 550, 230, 220, 400, 220},{ 550, 220, 250, 400, 220},{ 550, 400, 400,
00768 400, 400},{ 550, 250, 190, 400, 220}},
00769 /* GC.G..GC */
00769 {{ 550, 550, 550, 550, 550},{ 550, 170, 400, 80, 400},{ 550, 400, 400, 400, 400},{ 550, 80, 400,
00770 220, 400},{ 550, 400, 400, 400, 400}},
00771 /* GC.U..GC */
00771 {{ 550, 550, 550, 550, 550},{ 550, 400, 400, 400, 400},{ 550, 400, 220, 400, 120},{ 550, 400, 400,
00772 400, 400},{ 550, 400, 170, 400, 120}}
00772 },
00773 {
00774 /* GC.@..CG */
00775 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00776 550, 550},{ 550, 550, 550, 550, 550}},
00777 /* GC.A..CG */
00777 {{ 550, 550, 550, 550, 550},{ 550, 240, 220, 160, 400},{ 550, 210, 170, 160, 400},{ 550, 100, 60,
00778 40, 400},{ 550, 400, 400, 400, 400}},
00779 /* GC.C..CG */
00779 {{ 550, 550, 550, 550, 550},{ 550, 230, 220, 400, 220},{ 550, 220, 250, 400, 220},{ 550, 400, 400,
00780 400, 400},{ 550, 250, 190, 400, 220}},
00781 /* GC.G..CG */
00781 {{ 550, 550, 550, 550, 550},{ 550, 170, 400, 80, 400},{ 550, 400, 400, 400, 400},{ 550, 80, 400,
00782 220, 400},{ 550, 400, 400, 400, 400}},
00783 /* GC.U..CG */
00783 {{ 550, 550, 550, 550, 550},{ 550, 400, 400, 400, 400},{ 550, 400, 220, 400, 130},{ 550, 400, 400,
00784 400, 400},{ 550, 400, 170, 400, 120}}
00784 },
00785 {
00786 /* GC.@..UG */
00787 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00788 550, 550},{ 550, 550, 550, 550, 550}},
00789 /* GC.A..UG */
00789 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
00790 120, 480},{ 550, 480, 480, 480, 480}},
00791 /* GC.C..UG */
00791 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
00792 480, 480},{ 550, 330, 270, 480, 300}},
00793 /* GC.G..UG */
00793 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
00794 300, 480},{ 550, 480, 480, 480, 480}},
00795 /* GC.U..UG */
00795 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
00796 480, 480},{ 550, 480, 250, 480, 200}}
00796 },
00797 {
00798 /* GC.@..GU */
00799 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00800 550, 550},{ 550, 550, 550, 550, 550}},
00801 /* GC.A..GU */
```

```
00801 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
00802 /* GC.C..GU */
00803 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
00804 /* GC.G..GU */
00805 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
00806 /* GC.U..GU */
00807 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
00808 },
00809 {
00810 /* GC.@..UA */
00811 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00812 /* GC.A..UA */
00813 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
00814 /* GC.C..UA */
00815 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
00816 /* GC.G..UA */
00817 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
00818 /* GC.U..UA */
00819 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
00820 },
00821 {
00822 /* GC.@..AU */
00823 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00824 /* GC.A..AU */
00825 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
00826 /* GC.C..AU */
00827 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
00828 /* GC.G..AU */
00829 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
00830 /* GC.U..AU */
00831 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
00832 },
00833 {
00834 /* GC.@..?? */
00835 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00836 /* GC.A..?? */
00837 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00838 /* GC.C..?? */
00839 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00840 /* GC.G..?? */
00841 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00842 /* GC.U..?? */
00843 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00844 }
00845 },
00846 { /* noPair */ {{0}}},
00847 {
00848 /* GU.@..GC */
00849 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00850 /* GU.A..GC */
00851 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
00852 /* GU.C..GC */
00853 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
00854 /* GU.G..GC */
00855 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
00856 /* GU.U..GC */
00857 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
00858 },
00859 {
00860 /* GU.@..CG */
00861 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
00862 /* GU.A..CG */
```

```
00863 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
00864 /* GU.C..CG */
00865 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
00866 /* GU.G..CG */
00867 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
00868 /* GU.U..CG */
00869 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
00870 },
00871 {
00872 /* GU.@..UG */
00873 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00874 /* GU.A..UG */
00875 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00876 /* GU.C..UG */
00877 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00878 /* GU.G..UG */
00879 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00880 /* GU.U..UG */
00881 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00882 },
00883 {
00884 /* GU.@..GU */
00885 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00886 /* GU.A..GU */
00887 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00888 /* GU.C..GU */
00889 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00890 /* GU.G..GU */
00891 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00892 /* GU.U..GU */
00893 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00894 },
00895 {
00896 /* GU.@..UA */
00897 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00898 /* GU.A..UA */
00899 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00900 /* GU.C..UA */
00901 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00902 /* GU.G..UA */
00903 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00904 /* GU.U..UA */
00905 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00906 },
00907 {
00908 /* GU.@..AU */
00909 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00910 /* GU.A..AU */
00911 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00912 /* GU.C..AU */
00913 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00914 /* GU.G..AU */
00915 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00916 /* GU.U..AU */
00917 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00918 },
00919 {
00920 /* GU.@..?? */
00921 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00922 /* GU.A..?? */
00923 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
```

```
00924 /* GU.C...?? */
00925 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00926 /* GU.G...?? */
00927 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00928 /* GU.U...?? */
00929 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
00930 }
00931 },
00932 { /* noPair */ {{{0}}},
00933 {
00934 /* UG.@..GC */
00935 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00936 /* UG.A..GC */
00937 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
00938 /* UG.C..GC */
00939 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
00940 /* UG.G..GC */
00941 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
00942 /* UG.U..GC */
00943 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
00944 },
00945 {
00946 /* UG.@..CG */
00947 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00948 /* UG.A..CG */
00949 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
00950 /* UG.C..CG */
00951 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
00952 /* UG.G..CG */
00953 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
00954 /* UG.U..CG */
00955 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
00956 },
00957 {
00958 /* UG.@..UG */
00959 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00960 /* UG.A..UG */
00961 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00962 /* UG.C..UG */
00963 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00964 /* UG.G..UG */
00965 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00966 /* UG.U..UG */
00967 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00968 },
00969 {
00970 /* UG.@..GU */
00971 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00972 /* UG.A..GU */
00973 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00974 /* UG.C..GU */
00975 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00976 /* UG.G..GU */
00977 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00978 /* UG.U..GU */
00979 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00980 },
00981 {
00982 /* UG.@..UA */
00983 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00984 /* UG.A..UA */
00985 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
```

```
00986 /* UG.C..UA */
00987 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
00988 /* UG.G..UA */
00989 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
00990 /* UG.U..UA */
00991 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
00992 },
00993 {
00994 /* UG.@..AU */
00995 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
00996 /* UG.A..AU */
00997 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
00998 /* UG.C..AU */
00999 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01000 /* UG.G..AU */
01001 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01002 /* UG.U..AU */
01003 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01004 },
01005 {
01006 /* UG.@..?? */
01007 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01008 /* UG.A..?? */
01009 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01010 /* UG.C..?? */
01011 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01012 /* UG.G..?? */
01013 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01014 /* UG.U..?? */
01015 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
01016 },
01017 },
01018 { /* noPair */ {{0}}},
01019 {
01020 /* AU.@..GC */
01021 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01022 /* AU.A..GC */
01023 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
01024 /* AU.C..GC */
01025 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
01026 /* AU.G..GC */
01027 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
01028 /* AU.U..GC */
01029 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
01030 },
01031 {
01032 /* AU.@..CG */
01033 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01034 /* AU.A..CG */
01035 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
01036 /* AU.C..CG */
01037 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
01038 /* AU.G..CG */
01039 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
01040 /* AU.U..CG */
01041 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
01042 },
01043 {
01044 /* AU.@..UG */
01045 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01046 /* AU.A..UG */
01047 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
```

```
01048 /* AU.C..UG */
01049 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01050 /* AU.G..UG */
01051 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01052 /* AU.U..UG */
01053 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01054 },
01055 {
01056 /* AU.@..GU */
01057 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01058 /* AU.A..GU */
01059 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01060 /* AU.C..GU */
01061 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01062 /* AU.G..GU */
01063 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01064 /* AU.U..GU */
01065 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01066 },
01067 {
01068 /* AU.@..UA */
01069 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01070 /* AU.A..UA */
01071 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01072 /* AU.C..UA */
01073 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01074 /* AU.G..UA */
01075 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01076 /* AU.U..UA */
01077 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01078 },
01079 {
01080 /* AU.@..AU */
01081 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01082 /* AU.A..AU */
01083 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01084 /* AU.C..AU */
01085 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01086 /* AU.G..AU */
01087 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01088 /* AU.U..AU */
01089 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01090 },
01091 {
01092 /* AU.@..?? */
01093 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01094 /* AU.A..?? */
01095 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01096 /* AU.C..?? */
01097 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01098 /* AU.G..?? */
01099 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01100 /* AU.U..?? */
01101 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
01102 }
01103 },
01104 { /* noPair */ {{0}}},
01105 {
01106 /* UA.@..GC */
01107 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01108 /* UA.A..GC */
01109 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
```



```
01110 /* UA.C..GC */
01111 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
01112 /* UA.G..GC */
01113 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
01114 /* UA.U..GC */
01115 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
01116 },
01117 {
01118 /* UA.@..CG */
01119 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01120 /* UA.A..CG */
01121 {{ 550, 550, 550, 550, 550},{ 550, 320, 300, 240, 480},{ 550, 290, 250, 240, 480},{ 550, 180, 140,
120, 480},{ 550, 480, 480, 480, 480}},
01122 /* UA.C..CG */
01123 {{ 550, 550, 550, 550, 550},{ 550, 310, 300, 480, 300},{ 550, 300, 330, 480, 300},{ 550, 480, 480,
480, 480},{ 550, 330, 270, 480, 300}},
01124 /* UA.G..CG */
01125 {{ 550, 550, 550, 550, 550},{ 550, 250, 480, 160, 480},{ 550, 480, 480, 480, 480},{ 550, 160, 480,
300, 480},{ 550, 480, 480, 480, 480}},
01126 /* UA.U..CG */
01127 {{ 550, 550, 550, 550, 550},{ 550, 480, 480, 480, 480},{ 550, 480, 300, 480, 210},{ 550, 480, 480,
480, 480},{ 550, 480, 250, 480, 200}}
01128 },
01129 {
01130 /* UA.@..UG */
01131 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01132 /* UA.A..UG */
01133 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01134 /* UA.C..UG */
01135 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01136 /* UA.G..UG */
01137 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01138 /* UA.U..UG */
01139 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01140 },
01141 {
01142 /* UA.@..GU */
01143 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01144 /* UA.A..GU */
01145 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01146 /* UA.C..GU */
01147 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01148 /* UA.G..GU */
01149 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01150 /* UA.U..GU */
01151 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01152 },
01153 {
01154 /* UA.@..UA */
01155 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01156 /* UA.A..UA */
01157 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01158 /* UA.C..UA */
01159 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
550, 550},{ 550, 400, 340, 550, 370}},
01160 /* UA.G..UA */
01161 {{ 550, 550, 550, 550, 550},{ 550, 320, 550, 230, 550},{ 550, 550, 550, 550, 550},{ 550, 230, 550,
370, 550},{ 550, 550, 550, 550, 550}},
01162 /* UA.U..UA */
01163 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 370, 550, 280},{ 550, 550, 550,
550, 550},{ 550, 550, 320, 550, 270}}
01164 },
01165 {
01166 /* UA.@..AU */
01167 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01168 /* UA.A..AU */
01169 {{ 550, 550, 550, 550, 550},{ 550, 390, 370, 310, 550},{ 550, 360, 320, 310, 550},{ 550, 250, 210,
190, 550},{ 550, 550, 550, 550, 550}},
01170 /* UA.C..AU */
01171 {{ 550, 550, 550, 550, 550},{ 550, 380, 370, 550, 370},{ 550, 370, 400, 550, 370},{ 550, 550, 550,
```



```

    550, 550},{ 550, 550, 550, 550, 550}},
01234 /* ??..GU */
01235 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01236 /* ??..U..GU */
01237 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
01238 },
01239 {
01240 /* ??..@..UA */
01241 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01242 /* ??..A..UA */
01243 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01244 /* ??..C..UA */
01245 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01246 /* ??..G..UA */
01247 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01248 /* ??..U..UA */
01249 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
01250 },
01251 {
01252 /* ??..@..AU */
01253 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01254 /* ??..A..AU */
01255 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01256 /* ??..C..AU */
01257 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01258 /* ??..G..AU */
01259 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01260 /* ??..U..AU */
01261 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
01262 },
01263 {
01264 /* ??..@..?? */
01265 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01266 /* ??..A..?? */
01267 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01268 /* ??..C..?? */
01269 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01270 /* ??..G..?? */
01271 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}},
01272 /* ??..U..?? */
01273 {{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550, 550, 550},{ 550, 550, 550,
550, 550},{ 550, 550, 550, 550, 550}}
01274 }
01275 }
01276 };
01277
01278 PRIVATE int int21_H_184[NBPAIRS+1][NBPAIRS+1][5][5][5] =
01279 { /* noPair */ {{{0}}},
01280 { /* noPair */ {{{0}}},
01281 {
01282 /* CG..@..CG */
01283 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01284 /* CG..A..CG */
01285 {{ DEF, -1029, -949, -1029, -1029},{ -1079, -2058, -1978, -2058, -2058},{ -569, -1548, -1468, -1548, -1548},{
-989, -1968, -1888, -1968, -1968},{ -859, -1838, -1758, -1838, -1838}},
01286 /* CG..C..CG */
01287 {{ DEF, -519, -449, -519, -669},{ -999, -1468, -1398, -1468, -1618},{ -499, -968, -898, -968, -1118},{
-989, -1458, -1388, -1458, -1608},{ -789, -1258, -1188, -1258, -1408}},
01288 /* CG..G..CG */
01289 {{ DEF, -939, -939, -939, -939},{ -1079, -1968, -1968, -1968, -1968},{ -569, -1458, -1458, -1458, -1458},{
-989, -1878, -1878, -1878, -1878},{ -859, -1748, -1748, -1748, -1748}},
01290 /* CG..U..CG */
01291 {{ DEF, -809, -739, -809, -859},{ -1079, -1838, -1768, -1838, -1888},{ -719, -1478, -1408, -1478, -1528},{
-989, -1748, -1678, -1748, -1798},{ -909, -1668, -1598, -1668, -1718}}
01292 },
01293 {
01294 /* CG..@..GC */
01295 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01296 /* CG..A..GC */

```

```
01297 {{ DEF,-1029, -949,-1029,-1029},{ -569,-1548,-1468,-1548,-1548},{ -769,-1748,-1668,-1748,-1748},{
-759,-1738,-1658,-1738,-1738},{ -549,-1528,-1448,-1528,-1528}},
01298 /* CG.C..GC */
01299 {{ DEF, -519, -449, -519, -669},{ -929,-1398,-1328,-1398,-1548},{ -359, -828, -758, -828, -978},{
-789,-1258,-1188,-1258,-1408},{ -549,-1018, -948,-1018,-1168}},
01300 /* CG.G..GC */
01301 {{ DEF, -939, -939, -939, -939},{ -609,-1498,-1498,-1498,-1498},{ -359,-1248,-1248,-1248,-1248},{
-669,-1558,-1558,-1558,-1558},{ -549,-1438,-1438,-1438,-1438}},
01302 /* CG.U..GC */
01303 {{ DEF, -809, -739, -809, -859},{ -929,-1688,-1618,-1688,-1738},{ -439,-1198,-1128,-1198,-1248},{
-789,-1548,-1478,-1548,-1598},{ -619,-1378,-1308,-1378,-1428}}
01304 },
01305 {
01306 /* CG.@..GU */
01307 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01308 /* CG.A..GU */
01309 {{ DEF,-1029, -949,-1029,-1029},{ -479,-1458,-1378,-1458,-1458},{ -309,-1288,-1208,-1288,-1288},{
-389,-1368,-1288,-1368,-1368},{ -379,-1358,-1278,-1358,-1358}},
01310 /* CG.C..GU */
01311 {{ DEF, -519, -449, -519, -669},{ -649,-1118,-1048,-1118,-1268},{ -289, -758, -688, -758, -908},{
-739,-1208,-1138,-1208,-1358},{ -379, -848, -778, -848, -998}},
01312 /* CG.G..GU */
01313 {{ DEF, -939, -939, -939, -939},{ -649,-1538,-1538,-1538,-1538},{ -289,-1178,-1178,-1178,-1178},{
-739,-1628,-1628,-1628,-1628},{ -379,-1268,-1268,-1268,-1268}},
01314 /* CG.U..GU */
01315 {{ DEF, -809, -739, -809, -859},{ -649,-1408,-1338,-1408,-1458},{ -289,-1048, -978,-1048,-1098},{
-739,-1498,-1428,-1498,-1548},{ -379,-1138,-1068,-1138,-1188}}
01316 },
01317 {
01318 /* CG.@..UG */
01319 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01320 /* CG.A..UG */
01321 {{ DEF,-1029, -949,-1029,-1029},{ -769,-1748,-1668,-1748,-1748},{ -529,-1508,-1428,-1508,-1508},{
-709,-1688,-1608,-1688,-1688},{ -599,-1578,-1498,-1578,-1578}},
01322 /* CG.C..UG */
01323 {{ DEF, -519, -449, -519, -669},{ -839,-1308,-1238,-1308,-1458},{ -529, -998, -928, -998,-1148},{
-859,-1328,-1258,-1328,-1478},{ -489, -958, -888, -958,-1108}},
01324 /* CG.G..UG */
01325 {{ DEF, -939, -939, -939, -939},{ -1009,-1898,-1898,-1898,-1898},{ -409,-1298,-1298,-1298,-1298},{
-969,-1858,-1858,-1858,-1858},{ -599,-1488,-1488,-1488,-1488}},
01326 /* CG.U..UG */
01327 {{ DEF, -809, -739, -809, -859},{ -859,-1618,-1548,-1618,-1668},{ -529,-1288,-1218,-1288,-1338},{
-859,-1618,-1548,-1618,-1668},{ -409,-1168,-1098,-1168,-1218}}
01328 },
01329 {
01330 /* CG.@..AU */
01331 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01332 /* CG.A..AU */
01333 {{ DEF,-1029, -949,-1029,-1029},{ -479,-1458,-1378,-1458,-1458},{ -309,-1288,-1208,-1288,-1288},{
-389,-1368,-1288,-1368,-1368},{ -379,-1358,-1278,-1358,-1358}},
01334 /* CG.C..AU */
01335 {{ DEF, -519, -449, -519, -669},{ -649,-1118,-1048,-1118,-1268},{ -289, -758, -688, -758, -908},{
-739,-1208,-1138,-1208,-1358},{ -379, -848, -778, -848, -998}},
01336 /* CG.G..AU */
01337 {{ DEF, -939, -939, -939, -939},{ -649,-1538,-1538,-1538,-1538},{ -289,-1178,-1178,-1178,-1178},{
-739,-1628,-1628,-1628,-1628},{ -379,-1268,-1268,-1268,-1268}},
01338 /* CG.U..AU */
01339 {{ DEF, -809, -739, -809, -859},{ -649,-1408,-1338,-1408,-1458},{ -289,-1048, -978,-1048,-1098},{
-739,-1498,-1428,-1498,-1548},{ -379,-1138,-1068,-1138,-1188}}
01340 },
01341 {
01342 /* CG.@..UA */
01343 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01344 /* CG.A..UA */
01345 {{ DEF,-1029, -949,-1029,-1029},{ -449,-1428,-1348,-1428,-1428},{ -479,-1458,-1378,-1458,-1458},{
-429,-1408,-1328,-1408,-1408},{ -329,-1308,-1228,-1308,-1308}},
01346 /* CG.C..UA */
01347 {{ DEF, -519, -449, -519, -669},{ -679,-1148,-1078,-1148,-1298},{ -559,-1028, -958,-1028,-1178},{
-729,-1198,-1128,-1198,-1348},{ -189, -658, -588, -658, -808}},
01348 /* CG.G..UA */
01349 {{ DEF, -939, -939, -939, -939},{ -939,-1828,-1828,-1828,-1828},{ -249,-1138,-1138,-1138,-1138},{
-939,-1828,-1828,-1828,-1828},{ -329,-1218,-1218,-1218,-1218}},
01350 /* CG.U..UA */
01351 {{ DEF, -809, -739, -809, -859},{ -639,-1398,-1328,-1398,-1448},{ -229, -988, -918, -988,-1038},{
-729,-1488,-1418,-1488,-1538},{ -190, -949, -879, -949, -999}}
01352 },
01353 {
01354 /* CG.@.. @ */
01355 {{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01356 /* CG.A.. @ */
01357 {{ -100,-1079, -999,-1079,-1079},{ -100,-1079, -999,-1079,-1079},{ -100,-1079, -999,-1079,-1079},{
-100,-1079, -999,-1079,-1079},{ -100,-1079, -999,-1079,-1079}},
```

```

01358 /* CG.C.. @ */
01359 {{ -100, -569, -499, -569, -719},{ -100, -569, -499, -569, -719},{ -100, -569, -499, -569, -719},{
-100, -569, -499, -569, -719},{ -100, -569, -499, -569, -719}},
01360 /* CG.G.. @ */
01361 {{ -100, -989, -989, -989, -989},{ -100, -989, -989, -989, -989},{ -100, -989, -989, -989, -989},{
-100, -989, -989, -989, -989},{ -100, -989, -989, -989, -989}},
01362 /* CG.U.. @ */
01363 {{ -100, -859, -789, -859, -909},{ -100, -859, -789, -859, -909},{ -100, -859, -789, -859, -909},{
-100, -859, -789, -859, -909},{ -100, -859, -789, -859, -909}},
01364 },
01365 },
01366 { /* noPair */ {{0}}},
01367 {
01368 /* GC.@..CG */
01369 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01370 /* GC.A..CG */
01371 {{ DEF, -519, -879, -559, -879},{ -1079, -1548, -1908, -1588, -1908},{ -569, -1038, -1398, -1078, -1398},{
-989, -1458, -1818, -1498, -1818},{ -859, -1328, -1688, -1368, -1688}},
01372 /* GC.C..CG */
01373 {{ DEF, -719, -309, -309, -389},{ -999, -1668, -1258, -1258, -1338},{ -499, -1168, -758, -758, -838},{
-989, -1658, -1248, -1248, -1328},{ -789, -1458, -1048, -1048, -1128}},
01374 /* GC.G..CG */
01375 {{ DEF, -709, -739, -619, -739},{ -1079, -1738, -1768, -1648, -1768},{ -569, -1228, -1258, -1138, -1258},{
-989, -1648, -1678, -1558, -1678},{ -859, -1518, -1548, -1428, -1548}},
01376 /* GC.U..CG */
01377 {{ DEF, -499, -499, -499, -569},{ -1079, -1528, -1528, -1528, -1598},{ -719, -1168, -1168, -1168, -1238},{
-989, -1438, -1438, -1438, -1508},{ -909, -1358, -1358, -1358, -1428}},
01378 },
01379 {
01380 /* GC.@..GC */
01381 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01382 /* GC.A..GC */
01383 {{ DEF, -519, -879, -559, -879},{ -569, -1038, -1398, -1078, -1398},{ -769, -1238, -1598, -1278, -1598},{
-759, -1228, -1588, -1268, -1588},{ -549, -1018, -1378, -1058, -1378}},
01384 /* GC.C..GC */
01385 {{ DEF, -719, -309, -309, -389},{ -929, -1598, -1188, -1188, -1268},{ -359, -1028, -618, -618, -698},{
-789, -1458, -1048, -1048, -1128},{ -549, -1218, -808, -808, -888}},
01386 /* GC.G..GC */
01387 {{ DEF, -709, -739, -619, -739},{ -609, -1268, -1298, -1178, -1298},{ -359, -1018, -1048, -928, -1048},{
-669, -1328, -1358, -1238, -1358},{ -549, -1208, -1238, -1118, -1238}},
01388 /* GC.U..GC */
01389 {{ DEF, -499, -499, -499, -569},{ -929, -1378, -1378, -1378, -1448},{ -439, -888, -888, -888, -958},{
-789, -1238, -1238, -1238, -1308},{ -619, -1068, -1068, -1068, -1138}},
01390 },
01391 {
01392 /* GC.@..GU */
01393 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01394 /* GC.A..GU */
01395 {{ DEF, -519, -879, -559, -879},{ -479, -948, -1308, -988, -1308},{ -309, -778, -1138, -818, -1138},{
-389, -858, -1218, -898, -1218},{ -379, -848, -1208, -888, -1208}},
01396 /* GC.C..GU */
01397 {{ DEF, -719, -309, -309, -389},{ -649, -1318, -908, -908, -988},{ -289, -958, -548, -548, -628},{
-739, -1408, -998, -998, -1078},{ -379, -1048, -638, -638, -718}},
01398 /* GC.G..GU */
01399 {{ DEF, -709, -739, -619, -739},{ -649, -1308, -1338, -1218, -1338},{ -289, -948, -978, -858, -978},{
-739, -1398, -1428, -1308, -1428},{ -379, -1038, -1068, -948, -1068}},
01400 /* GC.U..GU */
01401 {{ DEF, -499, -499, -499, -569},{ -649, -1098, -1098, -1098, -1168},{ -289, -738, -738, -738, -808},{
-739, -1188, -1188, -1188, -1258},{ -379, -828, -828, -828, -898}},
01402 },
01403 {
01404 /* GC.@..UG */
01405 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01406 /* GC.A..UG */
01407 {{ DEF, -519, -879, -559, -879},{ -769, -1238, -1598, -1278, -1598},{ -529, -998, -1358, -1038, -1358},{
-709, -1178, -1538, -1218, -1538},{ -599, -1068, -1428, -1108, -1428}},
01408 /* GC.C..UG */
01409 {{ DEF, -719, -309, -309, -389},{ -839, -1508, -1098, -1098, -1178},{ -529, -1198, -788, -788, -868},{
-859, -1528, -1118, -1118, -1198},{ -489, -1158, -748, -748, -828}},
01410 /* GC.G..UG */
01411 {{ DEF, -709, -739, -619, -739},{ -1009, -1668, -1698, -1578, -1698},{ -409, -1068, -1098, -978, -1098},{
-969, -1628, -1658, -1538, -1658},{ -599, -1258, -1288, -1168, -1288}},
01412 /* GC.U..UG */
01413 {{ DEF, -499, -499, -499, -569},{ -859, -1308, -1308, -1308, -1378},{ -529, -978, -978, -978, -1048},{
-859, -1308, -1308, -1308, -1378},{ -409, -858, -858, -858, -928}},
01414 },
01415 {
01416 /* GC.@..AU */
01417 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01418 /* GC.A..AU */
01419 {{ DEF, -519, -879, -559, -879},{ -479, -948, -1308, -988, -1308},{ -309, -778, -1138, -818, -1138},{
-389, -858, -1218, -898, -1218},{ -379, -848, -1208, -888, -1208}},

```

```
01420 /* GC.C..AU */
01421 {{ DEF, -719, -309, -309, -389},{ -649,-1318, -908, -908, -988},{ -289, -958, -548, -548, -628},{
-739,-1408, -998, -998,-1078},{ -379,-1048, -638, -638, -718}},
01422 /* GC.G..AU */
01423 {{ DEF, -709, -739, -619, -739},{ -649,-1308,-1338,-1218,-1338},{ -289, -948, -978, -858, -978},{
-739,-1398,-1428,-1308,-1428},{ -379,-1038,-1068, -948,-1068}},
01424 /* GC.U..AU */
01425 {{ DEF, -499, -499, -499, -569},{ -649,-1098,-1098,-1098,-1168},{ -289, -738, -738, -738, -808},{
-739,-1188,-1188,-1188,-1258},{ -379, -828, -828, -828, -898}}
01426 },
01427 {
01428 /* GC.@..UA */
01429 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01430 /* GC.A..UA */
01431 {{ DEF, -519, -879, -559, -879},{ -449, -918,-1278, -958,-1278},{ -479, -948,-1308, -988,-1308},{
-429, -898,-1258, -938,-1258},{ -329, -798,-1158, -838,-1158}},
01432 /* GC.C..UA */
01433 {{ DEF, -719, -309, -309, -389},{ -679,-1348, -938, -938,-1018},{ -559,-1228, -818, -818, -898},{
-729,-1398, -988, -988,-1068},{ -189, -858, -448, -448, -528}},
01434 /* GC.G..UA */
01435 {{ DEF, -709, -739, -619, -739},{ -939,-1598,-1628,-1508,-1628},{ -249, -908, -938, -818, -938},{
-939,-1598,-1628,-1508,-1628},{ -329, -988,-1018, -898,-1018}},
01436 /* GC.U..UA */
01437 {{ DEF, -499, -499, -499, -569},{ -639,-1088,-1088,-1088,-1158},{ -229, -678, -678, -678, -748},{
-729,-1178,-1178,-1178,-1248},{ -190, -639, -639, -639, -709}}
01438 },
01439 {
01440 /* GC.@.. @ */
01441 {{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01442 /* GC.A.. @ */
01443 {{ -100, -569, -929, -609, -929},{ -100, -569, -929, -609, -929},{ -100, -569, -929, -609, -929},{
-100, -569, -929, -609, -929},{ -100, -569, -929, -609, -929}},
01444 /* GC.C.. @ */
01445 {{ -100, -769, -359, -359, -439},{ -100, -769, -359, -359, -439},{ -100, -769, -359, -359, -439},{
-100, -769, -359, -359, -439},{ -100, -769, -359, -359, -439}},
01446 /* GC.G.. @ */
01447 {{ -100, -759, -789, -669, -789},{ -100, -759, -789, -669, -789},{ -100, -759, -789, -669, -789},{
-100, -759, -789, -669, -789},{ -100, -759, -789, -669, -789}},
01448 /* GC.U.. @ */
01449 {{ -100, -549, -549, -549, -619},{ -100, -549, -549, -549, -619},{ -100, -549, -549, -549, -619},{
-100, -549, -549, -549, -619},{ -100, -549, -549, -549, -619}}
01450 },
01451 },
01452 { /* noPair */ {{0}}},
01453 {
01454 /* GU.@..CG */
01455 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01456 /* GU.A..CG */
01457 {{ DEF, -429, -599, -599, -599},{ -1079,-1458,-1628,-1628,-1628},{ -569, -948,-1118,-1118,-1118},{
-989,-1368,-1538,-1538,-1538},{ -859,-1238,-1408,-1408,-1408}},
01458 /* GU.C..CG */
01459 {{ DEF, -259, -239, -239, -239},{ -999,-1208,-1188,-1188,-1188},{ -499, -708, -688, -688, -688},{
-989,-1198,-1178,-1178,-1178},{ -789, -998, -978, -978, -978}},
01460 /* GU.G..CG */
01461 {{ DEF, -339, -689, -689, -689},{ -1079,-1368,-1718,-1718,-1718},{ -569, -858,-1208,-1208,-1208},{
-989,-1278,-1628,-1628,-1628},{ -859,-1148,-1498,-1498,-1498}},
01462 /* GU.U..CG */
01463 {{ DEF, -329, -329, -329, -329},{ -1079,-1358,-1358,-1358,-1358},{ -719, -998, -998, -998, -998},{
-989,-1268,-1268,-1268,-1268},{ -909,-1188,-1188,-1188,-1188}}
01464 },
01465 {
01466 /* GU.@..GC */
01467 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01468 /* GU.A..GC */
01469 {{ DEF, -429, -599, -599, -599},{ -569, -948,-1118,-1118,-1118},{ -769,-1148,-1318,-1318,-1318},{
-759,-1138,-1308,-1308,-1308},{ -549, -928,-1098,-1098,-1098}},
01470 /* GU.C..GC */
01471 {{ DEF, -259, -239, -239, -239},{ -929,-1138,-1118,-1118,-1118},{ -359, -568, -548, -548, -548},{
-789, -998, -978, -978, -978},{ -549, -758, -738, -738, -738}},
01472 /* GU.G..GC */
01473 {{ DEF, -339, -689, -689, -689},{ -609, -898,-1248,-1248,-1248},{ -359, -648, -998, -998, -998},{
-669, -958,-1308,-1308,-1308},{ -549, -838,-1188,-1188,-1188}},
01474 /* GU.U..GC */
01475 {{ DEF, -329, -329, -329, -329},{ -929,-1208,-1208,-1208,-1208},{ -439, -718, -718, -718, -718},{
-789,-1068,-1068,-1068,-1068},{ -619, -898, -898, -898, -898}}
01476 },
01477 {
01478 /* GU.@..GU */
01479 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01480 /* GU.A..GU */
01481 {{ DEF, -429, -599, -599, -599},{ -479, -858,-1028,-1028,-1028},{ -309, -688, -858, -858, -858},{
-389, -768, -938, -938, -938},{ -379, -758, -928, -928, -928}},
```

```

01482 /* GU.C..GU */
01483 {{ DEF, -259, -239, -239, -239},{ -649, -858, -838, -838},{ -289, -498, -478, -478},{
-739, -948, -928, -928},{ -379, -588, -568, -568}},
01484 /* GU.G..GU */
01485 {{ DEF, -339, -689, -689, -689},{ -649, -938, -1288, -1288},{ -289, -578, -928, -928},{
-739, -1028, -1378, -1378},{ -379, -668, -1018, -1018}},
01486 /* GU.U..GU */
01487 {{ DEF, -329, -329, -329, -329},{ -649, -928, -928, -928},{ -289, -568, -568, -568},{
-739, -1018, -1018, -1018},{ -379, -658, -658, -658}}
01488 },
01489 {
01490 /* GU.@..UG */
01491 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01492 /* GU.A..UG */
01493 {{ DEF, -429, -599, -599, -599},{ -769, -1148, -1318, -1318},{ -529, -908, -1078, -1078},{
-709, -1088, -1258, -1258},{ -599, -978, -1148, -1148}},
01494 /* GU.C..UG */
01495 {{ DEF, -259, -239, -239, -239},{ -839, -1048, -1028, -1028},{ -529, -738, -718, -718},{
-859, -1068, -1048, -1048},{ -489, -698, -678, -678}},
01496 /* GU.G..UG */
01497 {{ DEF, -339, -689, -689, -689},{ -1009, -1298, -1648, -1648},{ -409, -698, -1048, -1048},{
-969, -1258, -1608, -1608},{ -599, -888, -1238, -1238}},
01498 /* GU.U..UG */
01499 {{ DEF, -329, -329, -329, -329},{ -859, -1138, -1138, -1138},{ -529, -808, -808, -808},{
-859, -1138, -1138, -1138},{ -409, -688, -688, -688}}
01500 },
01501 {
01502 /* GU.@..AU */
01503 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01504 /* GU.A..AU */
01505 {{ DEF, -429, -599, -599, -599},{ -479, -858, -1028, -1028},{ -309, -688, -858, -858},{
-389, -768, -938, -938},{ -379, -758, -928, -928}},
01506 /* GU.C..AU */
01507 {{ DEF, -259, -239, -239, -239},{ -649, -858, -838, -838},{ -289, -498, -478, -478},{
-739, -948, -928, -928},{ -379, -588, -568, -568}},
01508 /* GU.G..AU */
01509 {{ DEF, -339, -689, -689, -689},{ -649, -938, -1288, -1288},{ -289, -578, -928, -928},{
-739, -1028, -1378, -1378},{ -379, -668, -1018, -1018}},
01510 /* GU.U..AU */
01511 {{ DEF, -329, -329, -329, -329},{ -649, -928, -928, -928},{ -289, -568, -568, -568},{
-739, -1018, -1018, -1018},{ -379, -658, -658, -658}}
01512 },
01513 {
01514 /* GU.@..UA */
01515 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01516 /* GU.A..UA */
01517 {{ DEF, -429, -599, -599, -599},{ -449, -828, -998, -998},{ -479, -858, -1028, -1028},{
-429, -808, -978, -978},{ -329, -708, -878, -878}},
01518 /* GU.C..UA */
01519 {{ DEF, -259, -239, -239, -239},{ -679, -888, -868, -868},{ -559, -768, -748, -748},{
-729, -938, -918, -918},{ -189, -398, -378, -378}},
01520 /* GU.G..UA */
01521 {{ DEF, -339, -689, -689, -689},{ -939, -1228, -1578, -1578},{ -249, -538, -888, -888},{
-939, -1228, -1578, -1578},{ -329, -618, -968, -968}},
01522 /* GU.U..UA */
01523 {{ DEF, -329, -329, -329, -329},{ -639, -918, -918, -918},{ -229, -508, -508, -508},{
-729, -1008, -1008, -1008},{ -190, -469, -469, -469}}
01524 },
01525 {
01526 /* GU.@.. @ */
01527 {{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01528 /* GU.A.. @ */
01529 {{ -100, -479, -649, -649, -649},{ -100, -479, -649, -649, -649},{ -100, -479, -649, -649},{
-100, -479, -649, -649, -649},{ -100, -479, -649, -649, -649}},
01530 /* GU.C.. @ */
01531 {{ -100, -309, -289, -289, -289},{ -100, -309, -289, -289, -289},{ -100, -309, -289, -289},{
-100, -309, -289, -289, -289},{ -100, -309, -289, -289, -289}},
01532 /* GU.G.. @ */
01533 {{ -100, -389, -739, -739, -739},{ -100, -389, -739, -739, -739},{ -100, -389, -739, -739},{
-100, -389, -739, -739, -739},{ -100, -389, -739, -739, -739}},
01534 /* GU.U.. @ */
01535 {{ -100, -379, -379, -379, -379},{ -100, -379, -379, -379, -379},{ -100, -379, -379, -379},{
-100, -379, -379, -379, -379},{ -100, -379, -379, -379, -379}}
01536 },
01537 },
01538 { /* noPair */ {{0}}},
01539 {
01540 /* UG.@..CG */
01541 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01542 /* UG.A..CG */
01543 {{ DEF, -719, -789, -959, -809},{ -1079, -1748, -1818, -1988, -1838},{ -569, -1238, -1308, -1478, -1328},{
-989, -1658, -1728, -1898, -1748},{ -859, -1528, -1598, -1768, -1618}},

```

```
01544 /* UG.C..CG */
01545 {{ DEF, -479, -479, -359, -479},{ -999,-1428,-1428,-1308,-1428},{ -499, -928, -928, -808, -928},{
-989,-1418,-1418,-1298,-1418},{ -789,-1218,-1218,-1098,-1218}},
01546 /* UG.G..CG */
01547 {{ DEF, -659, -809, -919, -809},{-1079,-1688,-1838,-1948,-1838},{ -569,-1178,-1328,-1438,-1328},{
-989,-1598,-1748,-1858,-1748},{ -859,-1468,-1618,-1728,-1618}},
01548 /* UG.U..CG */
01549 {{ DEF, -549, -439, -549, -359},{-1079,-1578,-1468,-1578,-1388},{ -719,-1218,-1108,-1218,-1028},{
-989,-1488,-1378,-1488,-1298},{ -909,-1408,-1298,-1408,-1218}}
01550 },
01551 {
01552 /* UG.@..GC */
01553 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01554 /* UG.A..GC */
01555 {{ DEF, -719, -789, -959, -809},{ -569,-1238,-1308,-1478,-1328},{ -769,-1438,-1508,-1678,-1528},{
-759,-1428,-1498,-1668,-1518},{ -549,-1218,-1288,-1458,-1308}},
01556 /* UG.C..GC */
01557 {{ DEF, -479, -479, -359, -479},{ -929,-1358,-1358,-1238,-1358},{ -359, -788, -788, -668, -788},{
-789,-1218,-1218,-1098,-1218},{ -549, -978, -978, -858, -978}},
01558 /* UG.G..GC */
01559 {{ DEF, -659, -809, -919, -809},{ -609,-1218,-1368,-1478,-1368},{ -359, -968,-1118,-1228,-1118},{
-669,-1278,-1428,-1538,-1428},{ -549,-1158,-1308,-1418,-1308}},
01560 /* UG.U..GC */
01561 {{ DEF, -549, -439, -549, -359},{ -929,-1428,-1318,-1428,-1238},{ -439, -938, -828, -938, -748},{
-789,-1288,-1178,-1288,-1098},{ -619,-1118,-1008,-1118, -928}}
01562 },
01563 {
01564 /* UG.@..GU */
01565 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01566 /* UG.A..GU */
01567 {{ DEF, -719, -789, -959, -809},{ -479,-1148,-1218,-1388,-1238},{ -309, -978,-1048,-1218,-1068},{
-389,-1058,-1128,-1298,-1148},{ -379,-1048,-1118,-1288,-1138}},
01568 /* UG.C..GU */
01569 {{ DEF, -479, -479, -359, -479},{ -649,-1078,-1078, -958,-1078},{ -289, -718, -718, -598, -718},{
-739,-1168,-1168,-1048,-1168},{ -379, -808, -808, -688, -808}},
01570 /* UG.G..GU */
01571 {{ DEF, -659, -809, -919, -809},{ -649,-1258,-1408,-1518,-1408},{ -289, -898,-1048,-1158,-1048},{
-739,-1348,-1498,-1608,-1498},{ -379, -988,-1138,-1248,-1138}},
01572 /* UG.U..GU */
01573 {{ DEF, -549, -439, -549, -359},{ -649,-1148,-1038,-1148, -958},{ -289, -788, -678, -788, -598},{
-739,-1238,-1128,-1238,-1048},{ -379, -878, -768, -878, -688}}
01574 },
01575 {
01576 /* UG.@..UG */
01577 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01578 /* UG.A..UG */
01579 {{ DEF, -719, -789, -959, -809},{ -769,-1438,-1508,-1678,-1528},{ -529,-1198,-1268,-1438,-1288},{
-709,-1378,-1448,-1618,-1468},{ -599,-1268,-1338,-1508,-1358}},
01580 /* UG.C..UG */
01581 {{ DEF, -479, -479, -359, -479},{ -839,-1268,-1268,-1148,-1268},{ -529, -958, -958, -838, -958},{
-859,-1288,-1288,-1168,-1288},{ -489, -918, -918, -798, -918}},
01582 /* UG.G..UG */
01583 {{ DEF, -659, -809, -919, -809},{-1009,-1618,-1768,-1878,-1768},{ -409,-1018,-1168,-1278,-1168},{
-969,-1578,-1728,-1838,-1728},{ -599,-1208,-1358,-1468,-1358}},
01584 /* UG.U..UG */
01585 {{ DEF, -549, -439, -549, -359},{ -859,-1358,-1248,-1358,-1168},{ -529,-1028, -918,-1028, -838},{
-859,-1358,-1248,-1358,-1168},{ -409, -908, -798, -908, -718}}
01586 },
01587 {
01588 /* UG.@..AU */
01589 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01590 /* UG.A..AU */
01591 {{ DEF, -719, -789, -959, -809},{ -479,-1148,-1218,-1388,-1238},{ -309, -978,-1048,-1218,-1068},{
-389,-1058,-1128,-1298,-1148},{ -379,-1048,-1118,-1288,-1138}},
01592 /* UG.C..AU */
01593 {{ DEF, -479, -479, -359, -479},{ -649,-1078,-1078, -958,-1078},{ -289, -718, -718, -598, -718},{
-739,-1168,-1168,-1048,-1168},{ -379, -808, -808, -688, -808}},
01594 /* UG.G..AU */
01595 {{ DEF, -659, -809, -919, -809},{ -649,-1258,-1408,-1518,-1408},{ -289, -898,-1048,-1158,-1048},{
-739,-1348,-1498,-1608,-1498},{ -379, -988,-1138,-1248,-1138}},
01596 /* UG.U..AU */
01597 {{ DEF, -549, -439, -549, -359},{ -649,-1148,-1038,-1148, -958},{ -289, -788, -678, -788, -598},{
-739,-1238,-1128,-1238,-1048},{ -379, -878, -768, -878, -688}}
01598 },
01599 {
01600 /* UG.@..UA */
01601 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01602 /* UG.A..UA */
01603 {{ DEF, -719, -789, -959, -809},{ -449,-1118,-1188,-1358,-1208},{ -479,-1148,-1218,-1388,-1238},{
-429,-1098,-1168,-1338,-1188},{ -329, -998,-1068,-1238,-1088}},
01604 /* UG.C..UA */
01605 {{ DEF, -479, -479, -359, -479},{ -679,-1108,-1108, -988,-1108},{ -559, -988, -988, -868, -988},{
```



```

-729,-1158,-1158,-1038,-1158},{ -189, -618, -618, -498, -618}},
01606 /* UG.G..UA */
01607 {{ DEF, -659, -809, -919, -809},{ -939,-1548,-1698,-1808,-1698},{ -249, -858,-1008,-1118,-1008},{
-939,-1548,-1698,-1808,-1698},{ -329, -938,-1088,-1198,-1088}},
01608 /* UG.U..UA */
01609 {{ DEF, -549, -439, -549, -359},{ -639,-1138,-1028,-1138, -948},{ -229, -728, -618, -728, -538},{
-729,-1228,-1118,-1228,-1038},{ -190, -689, -579, -689, -499}}
01610 },
01611 {
01612 /* UG.@.. @ */
01613 {{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01614 /* UG.A.. @ */
01615 {{ -100, -769, -839,-1009, -859},{ -100, -769, -839,-1009, -859},{ -100, -769, -839,-1009, -859},{
-100, -769, -839,-1009, -859},{ -100, -769, -839,-1009, -859}},
01616 /* UG.C.. @ */
01617 {{ -100, -529, -529, -409, -529},{ -100, -529, -529, -409, -529},{ -100, -529, -529, -409, -529},{
-100, -529, -529, -409, -529},{ -100, -529, -529, -409, -529}},
01618 /* UG.G.. @ */
01619 {{ -100, -709, -859, -969, -859},{ -100, -709, -859, -969, -859},{ -100, -709, -859, -969, -859},{
-100, -709, -859, -969, -859},{ -100, -709, -859, -969, -859}},
01620 /* UG.U.. @ */
01621 {{ -100, -599, -489, -599, -409},{ -100, -599, -489, -599, -409},{ -100, -599, -489, -599, -409},{
-100, -599, -489, -599, -409},{ -100, -599, -489, -599, -409}}
01622 },
01623 },
01624 { /* noPair */ {{0}}},
01625 {
01626 /* AU.@..CG */
01627 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01628 /* AU.A..CG */
01629 {{ DEF, -429, -599, -599, -599},{ -1079,-1458,-1628,-1628,-1628},{ -569, -948,-1118,-1118,-1118},{
-989,-1368,-1538,-1538,-1538},{ -859,-1238,-1408,-1408,-1408}},
01630 /* AU.C..CG */
01631 {{ DEF, -259, -239, -239, -239},{ -999,-1208,-1188,-1188,-1188},{ -499, -708, -688, -688, -688},{
-989,-1198,-1178,-1178,-1178},{ -789, -998, -978, -978, -978}},
01632 /* AU.G..CG */
01633 {{ DEF, -339, -689, -689, -689},{ -1079,-1368,-1718,-1718,-1718},{ -569, -858,-1208,-1208,-1208},{
-989,-1278,-1628,-1628,-1628},{ -859,-1148,-1498,-1498,-1498}},
01634 /* AU.U..CG */
01635 {{ DEF, -329, -329, -329, -329},{ -1079,-1358,-1358,-1358,-1358},{ -719, -998, -998, -998, -998},{
-989,-1268,-1268,-1268,-1268},{ -909,-1188,-1188,-1188,-1188}}
01636 },
01637 {
01638 /* AU.@..GC */
01639 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01640 /* AU.A..GC */
01641 {{ DEF, -429, -599, -599, -599},{ -569, -948,-1118,-1118,-1118},{ -769,-1148,-1318,-1318,-1318},{
-759,-1138,-1308,-1308,-1308},{ -549, -928,-1098,-1098,-1098}},
01642 /* AU.C..GC */
01643 {{ DEF, -259, -239, -239, -239},{ -929,-1138,-1118,-1118,-1118},{ -359, -568, -548, -548, -548},{
-789, -998, -978, -978, -978},{ -549, -758, -738, -738, -738}},
01644 /* AU.G..GC */
01645 {{ DEF, -339, -689, -689, -689},{ -609, -898,-1248,-1248,-1248},{ -359, -648, -998, -998, -998},{
-669, -958,-1308,-1308,-1308},{ -549, -838,-1188,-1188,-1188}},
01646 /* AU.U..GC */
01647 {{ DEF, -329, -329, -329, -329},{ -929,-1208,-1208,-1208,-1208},{ -439, -718, -718, -718, -718},{
-789,-1068,-1068,-1068,-1068},{ -619, -898, -898, -898, -898}}
01648 },
01649 {
01650 /* AU.@..GU */
01651 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01652 /* AU.A..GU */
01653 {{ DEF, -429, -599, -599, -599},{ -479, -858,-1028,-1028,-1028},{ -309, -688, -858, -858, -858},{
-389, -768, -938, -938, -938},{ -379, -758, -928, -928, -928}},
01654 /* AU.C..GU */
01655 {{ DEF, -259, -239, -239, -239},{ -649, -858, -838, -838, -838},{ -289, -498, -478, -478, -478},{
-739, -948, -928, -928, -928},{ -379, -588, -568, -568, -568}},
01656 /* AU.G..GU */
01657 {{ DEF, -339, -689, -689, -689},{ -649, -938,-1288,-1288,-1288},{ -289, -578, -928, -928, -928},{
-739,-1028,-1378,-1378,-1378},{ -379, -668,-1018,-1018,-1018}},
01658 /* AU.U..GU */
01659 {{ DEF, -329, -329, -329, -329},{ -649, -928, -928, -928, -928},{ -289, -568, -568, -568, -568},{
-739,-1018,-1018,-1018,-1018},{ -379, -658, -658, -658, -658}}
01660 },
01661 {
01662 /* AU.@..UG */
01663 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01664 /* AU.A..UG */
01665 {{ DEF, -429, -599, -599, -599},{ -769,-1148,-1318,-1318,-1318},{ -529, -908,-1078,-1078,-1078},{
-709,-1088,-1258,-1258,-1258},{ -599, -978,-1148,-1148,-1148}},
01666 /* AU.C..UG */
01667 {{ DEF, -259, -239, -239, -239},{ -839,-1048,-1028,-1028,-1028},{ -529, -738, -718, -718, -718},{

```

```

-859,-1068,-1048,-1048,-1048},{ -489, -698, -678, -678, -678}},
01668 /* AU.G..UG */
01669 {{ DEF, -339, -689, -689, -689, -689},{-1009,-1298,-1648,-1648,-1648},{ -409, -698,-1048,-1048,-1048},{
-969,-1258,-1608,-1608,-1608},{ -599, -888,-1238,-1238,-1238}},
01670 /* AU.U..UG */
01671 {{ DEF, -329, -329, -329, -329},{ -859,-1138,-1138,-1138,-1138},{ -529, -808, -808, -808, -808},{
-859,-1138,-1138,-1138,-1138},{ -409, -688, -688, -688, -688}}
01672 },
01673 {
01674 /* AU.@..AU */
01675 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01676 /* AU.A..AU */
01677 {{ DEF, -429, -599, -599, -599},{ -479, -858,-1028,-1028,-1028},{ -309, -688, -858, -858, -858},{
-389, -768, -938, -938, -938},{ -379, -758, -928, -928, -928}},
01678 /* AU.C..AU */
01679 {{ DEF, -259, -239, -239, -239},{ -649, -858, -838, -838, -838},{ -289, -498, -478, -478, -478},{
-739, -948, -928, -928, -928},{ -379, -588, -568, -568, -568}},
01680 /* AU.G..AU */
01681 {{ DEF, -339, -689, -689, -689},{ -649, -938,-1288,-1288,-1288},{ -289, -578, -928, -928, -928},{
-739,-1028,-1378,-1378,-1378},{ -379, -668,-1018,-1018,-1018}},
01682 /* AU.U..AU */
01683 {{ DEF, -329, -329, -329, -329},{ -649, -928, -928, -928, -928},{ -289, -568, -568, -568, -568},{
-739,-1018,-1018,-1018,-1018},{ -379, -658, -658, -658, -658}}
01684 },
01685 {
01686 /* AU.@..UA */
01687 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01688 /* AU.A..UA */
01689 {{ DEF, -429, -599, -599, -599},{ -449, -828, -998, -998, -998},{ -479, -858,-1028,-1028,-1028},{
-429, -808, -978, -978, -978},{ -329, -708, -878, -878, -878}},
01690 /* AU.C..UA */
01691 {{ DEF, -259, -239, -239, -239},{ -679, -888, -868, -868, -868},{ -559, -768, -748, -748, -748},{
-729, -938, -918, -918, -918},{ -189, -398, -378, -378, -378}},
01692 /* AU.G..UA */
01693 {{ DEF, -339, -689, -689, -689},{ -939,-1228,-1578,-1578,-1578},{ -249, -538, -888, -888, -888},{
-939,-1228,-1578,-1578,-1578},{ -329, -618, -968, -968, -968}},
01694 /* AU.U..UA */
01695 {{ DEF, -329, -329, -329, -329},{ -639, -918, -918, -918, -918},{ -229, -508, -508, -508, -508},{
-729,-1008,-1008,-1008,-1008},{ -190, -469, -469, -469, -469}}
01696 },
01697 {
01698 /* AU.@..@ */
01699 {{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01700 /* AU.A..@ */
01701 {{ -100, -479, -649, -649, -649},{ -100, -479, -649, -649, -649},{ -100, -479, -649, -649, -649},{
-100, -479, -649, -649, -649},{ -100, -479, -649, -649, -649}},
01702 /* AU.C..@ */
01703 {{ -100, -309, -289, -289, -289},{ -100, -309, -289, -289, -289},{ -100, -309, -289, -289, -289},{
-100, -309, -289, -289, -289},{ -100, -309, -289, -289, -289}},
01704 /* AU.G..@ */
01705 {{ -100, -389, -739, -739, -739},{ -100, -389, -739, -739, -739},{ -100, -389, -739, -739, -739},{
-100, -389, -739, -739, -739},{ -100, -389, -739, -739, -739}},
01706 /* AU.U..@ */
01707 {{ -100, -379, -379, -379, -379},{ -100, -379, -379, -379, -379},{ -100, -379, -379, -379, -379},{
-100, -379, -379, -379, -379},{ -100, -379, -379, -379, -379}}
01708 },
01709 },
01710 { /* noPair */ {{0}}},
01711 {
01712 /* UA.@..CG */
01713 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01714 /* UA.A..CG */
01715 {{ DEF, -399, -629, -889, -589},{-1079,-1428,-1658,-1918,-1618},{ -569, -918,-1148,-1408,-1108},{
-989,-1338,-1568,-1828,-1528},{ -859,-1208,-1438,-1698,-1398}},
01716 /* UA.C..CG */
01717 {{ DEF, -429, -509, -199, -179},{ -999,-1378,-1458,-1148,-1128},{ -499, -878, -958, -648, -628},{
-989,-1368,-1448,-1138,-1118},{ -789,-1168,-1248, -938, -918}},
01718 /* UA.G..CG */
01719 {{ DEF, -379, -679, -889, -679},{-1079,-1408,-1708,-1918,-1708},{ -569, -898,-1198,-1408,-1198},{
-989,-1318,-1618,-1828,-1618},{ -859,-1188,-1488,-1698,-1488}},
01720 /* UA.U..CG */
01721 {{ DEF, -279, -139, -279, -140},{-1079,-1308,-1168,-1308,-1169},{ -719, -948, -808, -948, -809},{
-989,-1218,-1078,-1218,-1079},{ -909,-1138, -998,-1138, -999}}
01722 },
01723 {
01724 /* UA.@..GC */
01725 {{ 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01726 /* UA.A..GC */
01727 {{ DEF, -399, -629, -889, -589},{ -569, -918,-1148,-1408,-1108},{ -769,-1118,-1348,-1608,-1308},{
-759,-1108,-1338,-1598,-1298},{ -549, -898,-1128,-1388,-1088}},
01728 /* UA.C..GC */
01729 {{ DEF, -429, -509, -199, -179},{ -929,-1308,-1388,-1078,-1058},{ -359, -738, -818, -508, -488},{

```

```

-789,-1168,-1248, -938, -918},{ -549, -928,-1008, -698, -678}},
01730 /* UA.G..GC */
01731 {{ DEF, -379, -679, -889, -679},{ -609, -938,-1238,-1448,-1238},{ -359, -688, -988,-1198, -988},{
-669, -998,-1298,-1508,-1298},{ -549, -878,-1178,-1388,-1178}},
01732 /* UA.U..GC */
01733 {{ DEF, -279, -139, -279, -140},{ -929,-1158,-1018,-1158,-1019},{ -439, -668, -528, -668, -529},{
-789,-1018, -878,-1018, -879},{ -619, -848, -708, -848, -709}}
01734 },
01735 {
01736 /* UA.@..GU */
01737 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01738 /* UA.A..GU */
01739 {{ DEF, -399, -629, -889, -589},{ -479, -828,-1058,-1318,-1018},{ -309, -658, -888,-1148, -848},{
-389, -738, -968,-1228, -928},{ -379, -728, -958,-1218, -918}},
01740 /* UA.C..GU */
01741 {{ DEF, -429, -509, -199, -179},{ -649,-1028,-1108, -798, -778},{ -289, -668, -748, -438, -418},{
-739,-1118,-1198, -888, -868},{ -379, -758, -838, -528, -508}},
01742 /* UA.G..GU */
01743 {{ DEF, -379, -679, -889, -679},{ -649, -978,-1278,-1488,-1278},{ -289, -618, -918,-1128, -918},{
-739,-1068,-1368,-1578,-1368},{ -379, -708,-1008,-1218,-1008}},
01744 /* UA.U..GU */
01745 {{ DEF, -279, -139, -279, -140},{ -649, -878, -738, -878, -739},{ -289, -518, -378, -518, -379},{
-739, -968, -828, -968, -829},{ -379, -608, -468, -608, -469}}
01746 },
01747 {
01748 /* UA.@..UG */
01749 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01750 /* UA.A..UG */
01751 {{ DEF, -399, -629, -889, -589},{ -769,-1118,-1348,-1608,-1308},{ -529, -878,-1108,-1368,-1068},{
-709,-1058,-1288,-1548,-1248},{ -599, -948,-1178,-1438,-1138}},
01752 /* UA.C..UG */
01753 {{ DEF, -429, -509, -199, -179},{ -839,-1218,-1298, -988, -968},{ -529, -908, -988, -678, -658},{
-859,-1238,-1318,-1008, -988},{ -489, -868, -948, -638, -618}},
01754 /* UA.G..UG */
01755 {{ DEF, -379, -679, -889, -679},{ -1009,-1338,-1638,-1848,-1638},{ -409, -738,-1038,-1248,-1038},{
-969,-1298,-1598,-1808,-1598},{ -599, -928,-1228,-1438,-1228}},
01756 /* UA.U..UG */
01757 {{ DEF, -279, -139, -279, -140},{ -859,-1088, -948,-1088, -949},{ -529, -758, -618, -758, -619},{
-859,-1088, -948,-1088, -949},{ -409, -638, -498, -638, -499}}
01758 },
01759 {
01760 /* UA.@..AU */
01761 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01762 /* UA.A..AU */
01763 {{ DEF, -399, -629, -889, -589},{ -479, -828,-1058,-1318,-1018},{ -309, -658, -888,-1148, -848},{
-389, -738, -968,-1228, -928},{ -379, -728, -958,-1218, -918}},
01764 /* UA.C..AU */
01765 {{ DEF, -429, -509, -199, -179},{ -649,-1028,-1108, -798, -778},{ -289, -668, -748, -438, -418},{
-739,-1118,-1198, -888, -868},{ -379, -758, -838, -528, -508}},
01766 /* UA.G..AU */
01767 {{ DEF, -379, -679, -889, -679},{ -649, -978,-1278,-1488,-1278},{ -289, -618, -918,-1128, -918},{
-739,-1068,-1368,-1578,-1368},{ -379, -708,-1008,-1218,-1008}},
01768 /* UA.U..AU */
01769 {{ DEF, -279, -139, -279, -140},{ -649, -878, -738, -878, -739},{ -289, -518, -378, -518, -379},{
-739, -968, -828, -968, -829},{ -379, -608, -468, -608, -469}}
01770 },
01771 {
01772 /* UA.@..UA */
01773 {{ 0, 0, 0, 0, 0, 0},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01774 /* UA.A..UA */
01775 {{ DEF, -399, -629, -889, -589},{ -449, -798,-1028,-1288, -988},{ -479, -828,-1058,-1318,-1018},{
-429, -778,-1008,-1268, -968},{ -329, -678, -908,-1168, -868}},
01776 /* UA.C..UA */
01777 {{ DEF, -429, -509, -199, -179},{ -679,-1058,-1138, -828, -808},{ -559, -938,-1018, -708, -688},{
-729,-1108,-1188, -878, -858},{ -189, -568, -648, -338, -318}},
01778 /* UA.G..UA */
01779 {{ DEF, -379, -679, -889, -679},{ -939,-1268,-1568,-1778,-1568},{ -249, -578, -878,-1088, -878},{
-939,-1268,-1568,-1778,-1568},{ -329, -658, -958,-1168, -958}},
01780 /* UA.U..UA */
01781 {{ DEF, -279, -139, -279, -140},{ -639, -868, -728, -868, -729},{ -229, -458, -318, -458, -319},{
-729, -958, -818, -958, -819},{ -190, -419, -279, -419, -280}}
01782 },
01783 {
01784 /* UA.@.. @ */
01785 {{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF},{
DEF, DEF, DEF, DEF, DEF},{ DEF, DEF, DEF, DEF, DEF}},
01786 /* UA.A.. @ */
01787 {{ -100, -449, -679, -939, -639},{ -100, -449, -679, -939, -639},{ -100, -449, -679, -939, -639},{
-100, -449, -679, -939, -639},{ -100, -449, -679, -939, -639}},
01788 /* UA.C.. @ */
01789 {{ -100, -479, -559, -249, -229},{ -100, -479, -559, -249, -229},{ -100, -479, -559, -249, -229},{
-100, -479, -559, -249, -229},{ -100, -479, -559, -249, -229}},
01790 /* UA.G.. @ */

```

```
01791 {{ -100, -429, -729, -939, -729},{ -100, -429, -729, -939, -729},{ -100, -429, -729, -939, -729},{
01792 -100, -429, -729, -939, -729},{ -100, -429, -729, -939, -729}},
01793 /* UA.U.. @ */
01793 {{ -100, -329, -189, -329, -190},{ -100, -329, -189, -329, -190},{ -100, -329, -189, -329, -190},{
01794 -100, -329, -189, -329, -190},{ -100, -329, -189, -329, -190}}
01794 }
01795 },
01796 { /* noPair */ {{0}},
01797 {
01798 /* @.U..CG */
01799 {{ DEF, DEF, DEF, DEF, DEF},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
01800 -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01801 /* @.A..CG */
01801 {{ DEF, DEF, DEF, DEF, DEF},{ -1079, -1079, -1079, -1079, -1079},{ -569, -569, -569, -569, -569},{
01802 -989, -989, -989, -989, -989},{ -859, -859, -859, -859, -859}},
01803 /* @.C..CG */
01803 {{ DEF, DEF, DEF, DEF, DEF},{ -999, -999, -999, -999, -999},{ -499, -499, -499, -499, -499},{
01804 -989, -989, -989, -989, -989},{ -789, -789, -789, -789, -789}},
01805 /* @.G..CG */
01805 {{ DEF, DEF, DEF, DEF, DEF},{ -1079, -1079, -1079, -1079, -1079},{ -569, -569, -569, -569, -569},{
01806 -989, -989, -989, -989, -989},{ -859, -859, -859, -859, -859}},
01807 /* @.U..CG */
01807 {{ DEF, DEF, DEF, DEF, DEF},{ -1079, -1079, -1079, -1079, -1079},{ -719, -719, -719, -719, -719},{
01808 -989, -989, -989, -989, -989},{ -909, -909, -909, -909, -909}}
01808 },
01809 {
01810 /* @.U..GC */
01811 {{ DEF, DEF, DEF, DEF, DEF},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
01812 -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01813 /* @.A..GC */
01813 {{ DEF, DEF, DEF, DEF, DEF},{ -569, -569, -569, -569, -569},{ -769, -769, -769, -769, -769},{
01814 -759, -759, -759, -759, -759},{ -549, -549, -549, -549, -549}},
01815 /* @.C..GC */
01815 {{ DEF, DEF, DEF, DEF, DEF},{ -929, -929, -929, -929, -929},{ -359, -359, -359, -359, -359},{
01816 -789, -789, -789, -789, -789},{ -549, -549, -549, -549, -549}},
01817 /* @.G..GC */
01817 {{ DEF, DEF, DEF, DEF, DEF},{ -609, -609, -609, -609, -609},{ -359, -359, -359, -359, -359},{
01818 -669, -669, -669, -669, -669},{ -549, -549, -549, -549, -549}},
01819 /* @.U..GC */
01819 {{ DEF, DEF, DEF, DEF, DEF},{ -929, -929, -929, -929, -929},{ -439, -439, -439, -439, -439},{
01820 -789, -789, -789, -789, -789},{ -619, -619, -619, -619, -619}}
01820 },
01821 {
01822 /* @.U..GU */
01823 {{ DEF, DEF, DEF, DEF, DEF},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
01824 -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01825 /* @.A..GU */
01825 {{ DEF, DEF, DEF, DEF, DEF},{ -479, -479, -479, -479, -479},{ -309, -309, -309, -309, -309},{
01826 -389, -389, -389, -389, -389},{ -379, -379, -379, -379, -379}},
01827 /* @.C..GU */
01827 {{ DEF, DEF, DEF, DEF, DEF},{ -649, -649, -649, -649, -649},{ -289, -289, -289, -289, -289},{
01828 -739, -739, -739, -739, -739},{ -379, -379, -379, -379, -379}},
01829 /* @.G..GU */
01829 {{ DEF, DEF, DEF, DEF, DEF},{ -649, -649, -649, -649, -649},{ -289, -289, -289, -289, -289},{
01830 -739, -739, -739, -739, -739},{ -379, -379, -379, -379, -379}},
01831 /* @.U..GU */
01831 {{ DEF, DEF, DEF, DEF, DEF},{ -649, -649, -649, -649, -649},{ -289, -289, -289, -289, -289},{
01832 -739, -739, -739, -739, -739},{ -379, -379, -379, -379, -379}}
01832 },
01833 {
01834 /* @.U..UG */
01835 {{ DEF, DEF, DEF, DEF, DEF},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
01836 -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01837 /* @.A..UG */
01837 {{ DEF, DEF, DEF, DEF, DEF},{ -769, -769, -769, -769, -769},{ -529, -529, -529, -529, -529},{
01838 -709, -709, -709, -709, -709},{ -599, -599, -599, -599, -599}},
01839 /* @.C..UG */
01839 {{ DEF, DEF, DEF, DEF, DEF},{ -839, -839, -839, -839, -839},{ -529, -529, -529, -529, -529},{
01840 -859, -859, -859, -859, -859},{ -489, -489, -489, -489, -489}},
01841 /* @.G..UG */
01841 {{ DEF, DEF, DEF, DEF, DEF},{ -1009, -1009, -1009, -1009, -1009},{ -409, -409, -409, -409, -409},{
01842 -969, -969, -969, -969, -969},{ -599, -599, -599, -599, -599}},
01843 /* @.U..UG */
01843 {{ DEF, DEF, DEF, DEF, DEF},{ -859, -859, -859, -859, -859},{ -529, -529, -529, -529, -529},{
01844 -859, -859, -859, -859, -859},{ -409, -409, -409, -409, -409}}
01844 },
01845 {
01846 /* @.U..AU */
01847 {{ DEF, DEF, DEF, DEF, DEF},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
01848 -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01849 /* @.A..AU */
01849 {{ DEF, DEF, DEF, DEF, DEF},{ -479, -479, -479, -479, -479},{ -309, -309, -309, -309, -309},{
01850 -389, -389, -389, -389, -389},{ -379, -379, -379, -379, -379}},
01851 /* @.C..AU */
01851 {{ DEF, DEF, DEF, DEF, DEF},{ -649, -649, -649, -649, -649},{ -289, -289, -289, -289, -289},{
01852 -739, -739, -739, -739, -739},{ -379, -379, -379, -379, -379}},
01853 /* @.G..AU */
```

```

01853 {{ DEF, DEF, DEF, DEF, DEF},{ -649, -649, -649, -649, -649},{ -289, -289, -289, -289, -289},{
-739, -739, -739, -739, -739},{ -379, -379, -379, -379, -379}},
01854 /* @.U..AU */
01855 {{ DEF, DEF, DEF, DEF, DEF},{ -649, -649, -649, -649, -649},{ -289, -289, -289, -289, -289},{
-739, -739, -739, -739, -739},{ -379, -379, -379, -379, -379}}
01856 },
01857 {
01858 /* @.@..UA */
01859 {{ DEF, DEF, DEF, DEF, DEF},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
-100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01860 /* @.A..UA */
01861 {{ DEF, DEF, DEF, DEF, DEF},{ -449, -449, -449, -449, -449},{ -479, -479, -479, -479, -479},{
-429, -429, -429, -429, -429},{ -329, -329, -329, -329, -329}},
01862 /* @.C..UA */
01863 {{ DEF, DEF, DEF, DEF, DEF},{ -679, -679, -679, -679, -679},{ -559, -559, -559, -559, -559},{
-729, -729, -729, -729, -729},{ -189, -189, -189, -189, -189}},
01864 /* @.G..UA */
01865 {{ DEF, DEF, DEF, DEF, DEF},{ -939, -939, -939, -939, -939},{ -249, -249, -249, -249, -249},{
-939, -939, -939, -939, -939},{ -329, -329, -329, -329, -329}},
01866 /* @.U..UA */
01867 {{ DEF, DEF, DEF, DEF, DEF},{ -639, -639, -639, -639, -639},{ -229, -229, -229, -229, -229},{
-729, -729, -729, -729, -729},{ -190, -190, -190, -190, -190}}
01868 },
01869 {
01870 /* @.@.. @ */
01871 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
-100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01872 /* @.A.. @ */
01873 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
-100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01874 /* @.C.. @ */
01875 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
-100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01876 /* @.G.. @ */
01877 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
-100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}},
01878 /* @.U.. @ */
01879 {{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100},{
-100, -100, -100, -100, -100},{ -100, -100, -100, -100, -100}}
01880 },
01881 {
01882 };
01883
01884 PRIVATE int int22_37_184[NBPAIRS+1][NBPAIRS+1][5][5][5][5] = {
01885 /* noPair */ {{{{0}}}},
01886 { /* noPair */ {{{{0}}}},
01887 /* CG....CG */
01888 {{
01889 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01890 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01891 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01892 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01893 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
01894 },
01895 {
01896 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01897 {{ 340, 340, 340, 340, 340},{ 340, 120, 150, 20, 200},{ 340, 120, 150, 20, 200},{ 340, 30, 60,
-70, 200},{ 340, 200, 200, 200, 200}},
01898 {{ 340, 340, 340, 340, 340},{ 340, 160, 200, 60, 200},{ 340, 210, 180, 150, 200},{ 340, 200, 200,
200, 200},{ 340, 190, 170, 130, 200}},
01899 {{ 340, 340, 340, 340, 340},{ 340, 30, 60, -70, 200},{ 340, 200, 200, 200, 200},{ 340, 100, 140,
0, 200},{ 340, -40, -110, -60, 200}},
01900 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 190, 170, 130, 200},{ 340, 110, 40,
90, 200},{ 340, 140, 80, 130, 200}}
01901 },
01902 {
01903 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01904 {{ 340, 340, 340, 340, 340},{ 340, 120, 210, 200, 190},{ 340, 110, 140, 200, 120},{ 340, 20, 150,
200, 130},{ 340, 200, 200, 200, 200}},
01905 {{ 340, 340, 340, 340, 340},{ 340, 150, 180, 200, 170},{ 340, 140, 170, 200, 150},{ 340, 200, 200,
200, 200},{ 340, 120, 150, 200, 140}},
01906 {{ 340, 340, 340, 340, 340},{ 340, 20, 150, 200, 130},{ 340, 200, 200, 200, 200},{ 340, 90, 180,
200, 170},{ 340, -150, -20, 200, -40}},
01907 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 120, 150, 200, 140},{ 340, 0, 130,
200, 110},{ 340, 30, 60, 200, 50}}
01908 },
01909 {
01910 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01911 {{ 340, 340, 340, 340, 340},{ 340, 30, 200, 100, 110},{ 340, 20, 200, 90, 0},{ 340, -70, 200,

```

```
0, 90},{ 340, 200, 200, 200, 200}},
01912 {{ 340, 340, 340, 340, 340},{ 340, 60, 200, 140, 40},{ 340, 150, 200, 180, 130},{ 340, 200, 200,
200, 200},{ 340, 130, 200, 170, 110}},
01913 {{ 340, 340, 340, 340, 340},{ 340, -70, 200, 0, 90},{ 340, 200, 200, 200, 200},{ 340, 0, 200,
80, 90},{ 340, -60, 200, -70, -260}},
01914 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 130, 200, 170, 110},{ 340, 90, 200,
90, -110},{ 340, 130, 200, 120, 110}}
01915 },
01916 {
01917 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01918 {{ 340, 340, 340, 340, 340},{ 340, 200, 190, -40, 140},{ 340, 200, 120, -150, 30},{ 340, 200, 130,
-60, 130},{ 340, 200, 200, 200, 200}},
01919 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, -110, 80},{ 340, 200, 150, -20, 60},{ 340, 200, 200,
200, 200},{ 340, 200, 140, -40, 50}},
01920 {{ 340, 340, 340, 340, 340},{ 340, 200, 130, -60, 130},{ 340, 200, 200, 200, 200},{ 340, 200, 170,
-70, 120},{ 340, 200, -40, -420, -50}},
01921 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 140, -40, 50},{ 340, 200, 110,
-260, 110},{ 340, 200, 50, -50, -40}}
01922 }
01923 },
01924 /* CG....GC */
01925 {{
01926 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01927 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01928 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01929 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01930 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
01931 },
01932 {
01933 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01934 {{ 340, 340, 340, 340, 340},{ 340, 50, 60, 0, 200},{ 340, 110, 150, -70, 200},{ 340, -30, 10,
-160, 200},{ 340, 200, 200, 200, 200}},
01935 {{ 340, 340, 340, 340, 340},{ 340, 110, 110, -100, 200},{ 340, 170, 150, -60, 200},{ 340, 200, 200,
200, 200},{ 340, 70, 50, 20, 200}},
01936 {{ 340, 340, 340, 340, 340},{ 340, 40, 50, -70, 200},{ 340, 200, 200, 200, 200},{ 340, 100, 140,
0, 200},{ 340, 10, -70, -80, 200}},
01937 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 180, 150, 120, 200},{ 340, -50, -60,
-60, 200},{ 340, 150, 0, 90, 200}}
01938 },
01939 {
01940 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01941 {{ 340, 340, 340, 340, 340},{ 340, 130, 220, 200, 200},{ 340, 100, 130, 200, 120},{ 340, -70, 70,
200, 40},{ 340, 200, 200, 200, 200}},
01942 {{ 340, 340, 340, 340, 340},{ 340, 100, 190, 200, 110},{ 340, 100, 130, 200, 120},{ 340, 200, 200,
200, 200},{ 340, 0, 30, 200, 170}},
01943 {{ 340, 340, 340, 340, 340},{ 340, 70, 70, 200, 100},{ 340, 200, 200, 200, 200},{ 340, 90, 180,
200, 170},{ 340, -190, -30, 200, -70}},
01944 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 110, 140, 200, 120},{ 340, -150, -20,
200, -30},{ 340, -20, -10, 200, 20}}
01945 },
01946 {
01947 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01948 {{ 340, 340, 340, 340, 340},{ 340, -20, 200, 110, 90},{ 340, -40, 200, 90, 0},{ 340, -170, 200,
-90, 30},{ 340, 200, 200, 200, 200}},
01949 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 80, -10},{ 340, 110, 200, 150, 100},{ 340, 200, 200,
200, 200},{ 340, 20, 200, 50, 0}},
01950 {{ 340, 340, 340, 340, 340},{ 340, -50, 200, -20, 60},{ 340, 200, 200, 200, 200},{ 340, 0, 200,
80, 90},{ 340, -90, 200, -100, -300}},
01951 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 120, 200, 150, 100},{ 340, -130, 200,
-60, -240},{ 340, 90, 200, 110, 60}}
01952 },
01953 {
01954 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01955 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, -10, 140},{ 340, 200, 120, -160, 30},{ 340, 200, 40,
-160, 50},{ 340, 200, 200, 200, 200}},
01956 {{ 340, 340, 340, 340, 340},{ 340, 200, 110, -160, 30},{ 340, 200, 120, -60, 30},{ 340, 200, 200,
200, 200},{ 340, 200, 20, -160, 10}},
01957 {{ 340, 340, 340, 340, 340},{ 340, 200, 50, -60, 140},{ 340, 200, 200, 200, 200},{ 340, 200, 170,
-70, 120},{ 340, 200, -70, -440, -100}},
01958 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 120, -50, 30},{ 340, 200, -10,
-410, 10},{ 340, 200, 40, -100, 60}}
01959 }
01960 },
01961 /* CG....GU */
01962 {{
01963 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
```

```
01964 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01965 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01966 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01967 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01968 },
01969 {
01970 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01971 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 100, 200},{ 340, 180, 210, 80, 200},{ 340, 80, 110,
-20, 200},{ 340, 200, 200, 200, 200}},
01972 {{ 340, 340, 340, 340, 340},{ 340, 190, 220, 90, 200},{ 340, 230, 210, 170, 200},{ 340, 200, 200,
200, 200},{ 340, 230, 210, 170, 200}},
01973 {{ 340, 340, 340, 340, 340},{ 340, 80, 110, -20, 200},{ 340, 200, 200, 200, 200},{ 340, 130, 170,
30, 200},{ 340, 60, 0, 40, 200}},
01974 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 230, 210, 170, 200},{ 340, 160, 90,
140, 200},{ 340, 190, 130, 180, 200}},
01975 },
01976 {
01977 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01978 {{ 340, 340, 340, 340, 340},{ 340, 190, 280, 200, 270},{ 340, 170, 200, 200, 180},{ 340, 70, 200,
200, 180},{ 340, 200, 200, 200, 200}},
01979 {{ 340, 340, 340, 340, 340},{ 340, 180, 210, 200, 190},{ 340, 160, 190, 200, 180},{ 340, 200, 200,
200, 200},{ 340, 160, 190, 200, 180}},
01980 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 200, 180},{ 340, 200, 200, 200, 200},{ 340, 120, 210,
200, 200},{ 340, -50, 80, 200, 70}},
01981 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 190, 200, 180},{ 340, 50, 180,
200, 160},{ 340, 80, 110, 200, 100}},
01982 },
01983 {
01984 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01985 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 180, 180},{ 340, 80, 200, 150, 60},{ 340, -20, 200,
50, 140},{ 340, 200, 200, 200, 200}},
01986 {{ 340, 340, 340, 340, 340},{ 340, 90, 200, 160, 70},{ 340, 170, 200, 210, 150},{ 340, 200, 200,
200, 200},{ 340, 170, 200, 210, 150}},
01987 {{ 340, 340, 340, 340, 340},{ 340, -20, 200, 50, 140},{ 340, 200, 200, 200, 200},{ 340, 30, 200,
110, 110},{ 340, 40, 200, 40, -160}},
01988 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 200, 210, 150},{ 340, 140, 200,
130, -60},{ 340, 180, 200, 170, 160}},
01989 },
01990 {
01991 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
01992 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 30, 220},{ 340, 200, 180, -90, 90},{ 340, 200, 180,
-10, 180},{ 340, 200, 200, 200, 200}},
01993 {{ 340, 340, 340, 340, 340},{ 340, 200, 190, -80, 100},{ 340, 200, 180, 0, 90},{ 340, 200, 200,
200, 200},{ 340, 200, 180, 0, 90}},
01994 {{ 340, 340, 340, 340, 340},{ 340, 200, 180, -10, 180},{ 340, 200, 200, 200, 200},{ 340, 200, 200,
-40, 150},{ 340, 200, 70, -310, 60}},
01995 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 180, 0, 90},{ 340, 200, 160,
-210, 160},{ 340, 200, 100, 0, 10}},
01996 }
01997 },
01998 /* CG....UG */
01999 {{
02000 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02001 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02002 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02003 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02004 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02005 },
02006 {
02007 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02008 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 100, 200},{ 340, 160, 190, 60, 200},{ 340, 100, 130,
0, 200},{ 340, 200, 200, 200, 200}},
02009 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 100, 200},{ 340, 260, 240, 200, 200},{ 340, 200, 200,
200, 200},{ 340, 260, 240, 200, 200}},
02010 {{ 340, 340, 340, 340, 340},{ 340, 100, 130, 0, 200},{ 340, 200, 200, 200, 200},{ 340, 140, 170,
40, 200},{ 340, 20, -40, 0, 200}},
02011 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 230, 210, 170, 200},{ 340, 150, 80,
130, 200},{ 340, 220, 150, 200, 200}},
02012 },
02013 {
02014 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02015 {{ 340, 340, 340, 340, 340},{ 340, 190, 280, 200, 270},{ 340, 150, 180, 200, 160},{ 340, 90, 220,
```

```
200, 200},{ 340, 200, 200, 200, 200}},
02016 {{ 340, 340, 340, 340, 340},{ 340, 190, 220, 200, 210},{ 340, 190, 220, 200, 210},{ 340, 200, 200,
200, 200},{ 340, 190, 220, 200, 210}},
02017 {{ 340, 340, 340, 340, 340},{ 340, 90, 220, 200, 200},{ 340, 200, 200, 200, 200},{ 340, 130, 220,
200, 200},{ 340, -90, 40, 200, 30}},
02018 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 190, 200, 180},{ 340, 40, 170,
200, 150},{ 340, 110, 140, 200, 120}}
02019 },
02020 {
02021 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02022 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 180, 180},{ 340, 60, 200, 130, 40},{ 340, 0, 200,
70, 160},{ 340, 200, 200, 200, 200}},
02023 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 180, 80},{ 340, 200, 200, 240, 180},{ 340, 200, 200,
200, 200},{ 340, 200, 200, 200, 240, 180}},
02024 {{ 340, 340, 340, 340, 340},{ 340, 0, 200, 70, 160},{ 340, 200, 200, 200, 200},{ 340, 40, 200,
110, 120},{ 340, 0, 200, 0, -200}},
02025 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 200, 210, 150},{ 340, 130, 200,
120, -70},{ 340, 200, 200, 190, 180}}
02026 },
02027 {
02028 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02029 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 30, 220},{ 340, 200, 160, -110, 70},{ 340, 200, 200,
10, 190},{ 340, 200, 200, 200, 200}},
02030 {{ 340, 340, 340, 340, 340},{ 340, 200, 210, -70, 120},{ 340, 200, 210, 30, 120},{ 340, 200, 200,
200, 200},{ 340, 200, 210, 30, 120}},
02031 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 10, 190},{ 340, 200, 200, 200, 200},{ 340, 200, 200,
-30, 150},{ 340, 200, 30, -350, 20}},
02032 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 180, 0, 90},{ 340, 200, 150,
-220, 150},{ 340, 200, 120, 30, 30}}
02033 },
02034 },
02035 /* CG....AU */
02036 {{
02037 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02038 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02039 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02040 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02041 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02042 },
02043 {
02044 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02045 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 100, 200},{ 340, 180, 210, 80, 200},{ 340, 80, 110,
-20, 200},{ 340, 200, 200, 200, 200}},
02046 {{ 340, 340, 340, 340, 340},{ 340, 190, 220, 90, 200},{ 340, 230, 210, 170, 200},{ 340, 200, 200,
200, 200},{ 340, 230, 210, 170, 200}},
02047 {{ 340, 340, 340, 340, 340},{ 340, 80, 110, -20, 200},{ 340, 200, 200, 200, 200},{ 340, 130, 170,
30, 200},{ 340, 60, 0, 40, 200}},
02048 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 230, 210, 170, 200},{ 340, 160, 90,
140, 200},{ 340, 190, 130, 180, 200}}
02049 },
02050 {
02051 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02052 {{ 340, 340, 340, 340, 340},{ 340, 190, 280, 200, 270},{ 340, 170, 200, 200, 180},{ 340, 70, 200,
200, 180},{ 340, 200, 200, 200, 200}},
02053 {{ 340, 340, 340, 340, 340},{ 340, 180, 210, 200, 190},{ 340, 160, 190, 200, 180},{ 340, 200, 200,
200, 200},{ 340, 160, 190, 200, 180}},
02054 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 200, 180},{ 340, 200, 200, 200, 200},{ 340, 120, 210,
200, 200},{ 340, -50, 80, 200, 70}},
02055 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 190, 200, 180},{ 340, 50, 180,
200, 160},{ 340, 80, 110, 200, 100}}
02056 },
02057 {
02058 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02059 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 180, 180},{ 340, 80, 200, 150, 60},{ 340, -20, 200,
50, 140},{ 340, 200, 200, 200, 200}},
02060 {{ 340, 340, 340, 340, 340},{ 340, 90, 200, 160, 70},{ 340, 170, 200, 210, 150},{ 340, 200, 200,
200, 200},{ 340, 170, 200, 210, 150}},
02061 {{ 340, 340, 340, 340, 340},{ 340, -20, 200, 50, 140},{ 340, 200, 200, 200, 200},{ 340, 30, 200,
110, 110},{ 340, 40, 200, 40, -160}},
02062 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 200, 210, 150},{ 340, 140, 200,
130, -60},{ 340, 180, 200, 170, 160}}
02063 },
02064 {
02065 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02066 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 30, 220},{ 340, 200, 180, -90, 90},{ 340, 200, 180,
-10, 180},{ 340, 200, 200, 200, 200}},
```



```
02067 {{ 340, 340, 340, 340, 340},{ 340, 200, 190, -80, 100},{ 340, 200, 180, 0, 90},{ 340, 200, 200,
200, 200},{ 340, 200, 180, 0, 90}},
02068 {{ 340, 340, 340, 340, 340},{ 340, 200, 180, -10, 180},{ 340, 200, 200, 200, 200},{ 340, 200, 200,
-40, 150},{ 340, 200, 70, -310, 60}},
02069 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 180, 0, 90},{ 340, 200, 160,
-210, 160},{ 340, 200, 100, 0, 10}}
02070 },
02071 },
02072 /* CG....UA */
02073 {{
02074 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02075 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02076 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02077 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02078 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02079 },
02080 {
02081 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02082 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 100, 200},{ 340, 160, 190, 60, 200},{ 340, 100, 130,
0, 200},{ 340, 200, 200, 200, 200}},
02083 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 100, 200},{ 340, 260, 240, 200, 200},{ 340, 200, 200,
200, 200},{ 340, 260, 240, 200, 200}},
02084 {{ 340, 340, 340, 340, 340},{ 340, 100, 130, 0, 200},{ 340, 200, 200, 200, 200},{ 340, 140, 170,
40, 200},{ 340, 20, -40, 0, 200}},
02085 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 230, 210, 170, 200},{ 340, 150, 80,
130, 200},{ 340, 220, 150, 200, 200}}
02086 },
02087 {
02088 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02089 {{ 340, 340, 340, 340, 340},{ 340, 190, 280, 200, 270},{ 340, 150, 180, 200, 160},{ 340, 90, 220,
200, 200},{ 340, 200, 200, 200, 200}},
02090 {{ 340, 340, 340, 340, 340},{ 340, 190, 220, 200, 210},{ 340, 190, 220, 200, 210},{ 340, 200, 200,
200, 200},{ 340, 190, 220, 200, 210}},
02091 {{ 340, 340, 340, 340, 340},{ 340, 90, 220, 200, 200},{ 340, 200, 200, 200, 200},{ 340, 130, 220,
200, 200},{ 340, -90, 40, 200, 30}},
02092 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 190, 200, 180},{ 340, 40, 170,
200, 150},{ 340, 110, 140, 200, 120}}
02093 },
02094 {
02095 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02096 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 180, 180},{ 340, 60, 200, 130, 40},{ 340, 0, 200,
70, 160},{ 340, 200, 200, 200, 200}},
02097 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 180, 80},{ 340, 200, 200, 240, 180},{ 340, 200, 200,
200, 200},{ 340, 200, 200, 240, 180}},
02098 {{ 340, 340, 340, 340, 340},{ 340, 0, 200, 70, 160},{ 340, 200, 200, 200, 200},{ 340, 40, 200,
110, 120},{ 340, 0, 200, 0, -200}},
02099 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 200, 210, 150},{ 340, 130, 200,
120, -70},{ 340, 200, 200, 190, 180}}
02100 },
02101 {
02102 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02103 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 30, 220},{ 340, 200, 160, -110, 70},{ 340, 200, 200,
10, 190},{ 340, 200, 200, 200, 200}},
02104 {{ 340, 340, 340, 340, 340},{ 340, 200, 210, -70, 120},{ 340, 200, 210, 30, 120},{ 340, 200, 200,
200, 200},{ 340, 200, 210, 30, 120}},
02105 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 10, 190},{ 340, 200, 200, 200, 200},{ 340, 200, 200,
-30, 150},{ 340, 200, 30, -350, 20}},
02106 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 180, 0, 90},{ 340, 200, 150,
-220, 150},{ 340, 200, 120, 30, 30}}
02107 },
02108 },
02109 /* CG....?? */
02110 {{
02111 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02112 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02113 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02114 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02115 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02116 },
02117 {
02118 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02119 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
```



```
02172 {{ 340, 340, 340, 340, 340},{ 340, -30, 200, 100, -50},{ 340, -70, 200, 90, -150},{ 340, -170, 200,
0, -130},{ 340, 200, 200, 200, 200}},
02173 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 140, -60},{ 340, 70, 200, 180, -20},{ 340, 200, 200,
200, 200},{ 340, 40, 200, 170, -10}},
02174 {{ 340, 340, 340, 340, 340},{ 340, -160, 200, 0, -60},{ 340, 200, 200, 200, 200},{ 340, -90, 200,
80, -60},{ 340, -160, 200, -70, -410}},
02175 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 40, 200, 170, -30},{ 340, 30, 200,
90, -240},{ 340, 50, 200, 120, 10}}
02176 },
02177 {
02178 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02179 {{ 340, 340, 340, 340, 340},{ 340, 200, 70, 10, 150},{ 340, 200, 0, -190, -20},{ 340, 200, 20,
-90, 90},{ 340, 200, 200, 200, 200}},
02180 {{ 340, 340, 340, 340, 340},{ 340, 200, 50, -70, 0},{ 340, 200, 30, -30, -10},{ 340, 200, 200,
200, 200},{ 340, 200, 20, -70, 40}},
02181 {{ 340, 340, 340, 340, 340},{ 340, 200, 20, -80, 90},{ 340, 200, 200, 200, 200},{ 340, 200, 50,
-100, 110},{ 340, 200, -160, -440, -100}},
02182 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 170, -70, 20},{ 340, 200, 0,
-300, 60},{ 340, 200, 10, -100, 60}}
02183 }
02184 },
02185 /* GC....GC */
02186 {{
02187 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02188 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02189 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02190 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02191 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02192 },
02193 {
02194 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02195 {{ 340, 340, 340, 340, 340},{ 340, 150, 120, 10, 200},{ 340, 120, 90, -10, 200},{ 340, -50, -80,
-190, 200},{ 340, 200, 200, 200, 200}},
02196 {{ 340, 340, 340, 340, 340},{ 340, 120, 90, -20, 200},{ 340, 180, 90, 90, 200},{ 340, 200, 200,
200, 200},{ 340, 80, 0, -10, 200}},
02197 {{ 340, 340, 340, 340, 340},{ 340, 10, -20, -130, 200},{ 340, 200, 200, 200, 200},{ 340, 110, 80,
-20, 200},{ 340, -70, -200, -130, 200}},
02198 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 190, 100, 90, 200},{ 340, -30, -160,
-90, 200},{ 340, 150, 20, 90, 200}}
02199 },
02200 {
02201 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02202 {{ 340, 340, 340, 340, 340},{ 340, 120, 180, 200, 190},{ 340, 100, 100, 200, 100},{ 340, -80, 20,
200, 30},{ 340, 200, 200, 200, 200}},
02203 {{ 340, 340, 340, 340, 340},{ 340, 90, 90, 200, 100},{ 340, 100, 100, 200, 100},{ 340, 200, 200,
200, 200},{ 340, 0, 0, 200, 0}},
02204 {{ 340, 340, 340, 340, 340},{ 340, -10, 90, 200, 90},{ 340, 200, 200, 200, 200},{ 340, 90, 150,
200, 150},{ 340, -190, -90, 200, -90}},
02205 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 100, 100, 200, 110},{ 340, -150, -50,
200, -50},{ 340, 20, 20, 200, 30}}
02206 },
02207 {
02208 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02209 {{ 340, 340, 340, 340, 340},{ 340, -50, 200, 110, -30},{ 340, -80, 200, 90, -150},{ 340, -260, 200,
-90, -150},{ 340, 200, 200, 200, 200}},
02210 {{ 340, 340, 340, 340, 340},{ 340, -80, 200, 80, -160},{ 340, 20, 200, 150, -50},{ 340, 200, 200,
200, 200},{ 340, -80, 200, 50, -150}},
02211 {{ 340, 340, 340, 340, 340},{ 340, -190, 200, -20, -90},{ 340, 200, 200, 200, 200},{ 340, -90, 200,
80, -60},{ 340, -190, 200, -100, -450}},
02212 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 30, 200, 150, -50},{ 340, -150, 200,
-60, -410},{ 340, 30, 200, 110, -50}}
02213 },
02214 {
02215 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02216 {{ 340, 340, 340, 340, 340},{ 340, 200, 80, -70, 150},{ 340, 200, 0, -190, 20},{ 340, 200, -80,
-190, 30},{ 340, 200, 200, 200, 200}},
02217 {{ 340, 340, 340, 340, 340},{ 340, 200, 0, -200, 20},{ 340, 200, 0, -90, 20},{ 340, 200, 200,
200, 200},{ 340, 200, -100, -190, -70}},
02218 {{ 340, 340, 340, 340, 340},{ 340, 200, -10, -130, 90},{ 340, 200, 200, 200, 200},{ 340, 200, 50,
-100, 110},{ 340, 200, -190, -490, -90}},
02219 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 0, -90, 30},{ 340, 200, -150,
-450, -50},{ 340, 200, -70, -90, -50}}
02220 }
02221 },
02222 /* GC....GU */
02223 {{
02224 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
```

```
340, 340},{ 340, 340, 340, 340, 340}},
02225 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02226 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02227 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02228 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02229 },
02230 {
02231 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02232 {{ 340, 340, 340, 340, 340},{ 340, 210, 180, 70, 200},{ 340, 190, 160, 50, 200},{ 340, 90, 60,
-50, 200},{ 340, 200, 200, 200, 200}},
02233 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, 60, 200},{ 340, 240, 150, 140, 200},{ 340, 200, 200,
200, 200},{ 340, 240, 150, 140, 200}},
02234 {{ 340, 340, 340, 340, 340},{ 340, 90, 60, -50, 200},{ 340, 200, 200, 200, 200},{ 340, 140, 110,
0, 200},{ 340, 70, -60, 10, 200}},
02235 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 150, 140, 200},{ 340, 170, 40,
110, 200},{ 340, 200, 70, 150, 200}},
02236 },
02237 {
02238 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02239 {{ 340, 340, 340, 340, 340},{ 340, 190, 250, 200, 250},{ 340, 160, 160, 200, 170},{ 340, 60, 160,
200, 170},{ 340, 200, 200, 200, 200}},
02240 {{ 340, 340, 340, 340, 340},{ 340, 170, 170, 200, 180},{ 340, 160, 160, 200, 160},{ 340, 200, 200,
200, 200},{ 340, 160, 160, 200, 160}},
02241 {{ 340, 340, 340, 340, 340},{ 340, 60, 160, 200, 170},{ 340, 200, 200, 200, 200},{ 340, 120, 180,
200, 180},{ 340, -50, 50, 200, 50}},
02242 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 160, 200, 160},{ 340, 40, 140,
200, 150},{ 340, 80, 80, 200, 80}},
02243 },
02244 {
02245 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02246 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 180, 40},{ 340, -10, 200, 150, -90},{ 340, -110, 200,
50, -10},{ 340, 200, 200, 200, 200}},
02247 {{ 340, 340, 340, 340, 340},{ 340, 0, 200, 160, -80},{ 340, 80, 200, 210, 10},{ 340, 200, 200,
200, 200},{ 340, 80, 200, 210, 10}},
02248 {{ 340, 340, 340, 340, 340},{ 340, -110, 200, 50, -10},{ 340, 200, 200, 200, 200},{ 340, -60, 200,
110, -30},{ 340, -50, 200, 40, -310}},
02249 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 80, 200, 210, 10},{ 340, 50, 200,
130, -210},{ 340, 80, 200, 170, 10}},
02250 },
02251 {
02252 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02253 {{ 340, 340, 340, 340, 340},{ 340, 200, 150, 0, 210},{ 340, 200, 60, -130, 90},{ 340, 200, 70,
-50, 170},{ 340, 200, 200, 200, 200}},
02254 {{ 340, 340, 340, 340, 340},{ 340, 200, 70, -120, 100},{ 340, 200, 60, -30, 80},{ 340, 200, 200,
200, 200},{ 340, 200, 60, -30, 80}},
02255 {{ 340, 340, 340, 340, 340},{ 340, 200, 70, -50, 170},{ 340, 200, 200, 200, 200},{ 340, 200, 80,
-70, 140},{ 340, 200, -50, -350, 50}},
02256 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 60, -30, 80},{ 340, 200, 50,
-250, 150},{ 340, 200, -20, -30, 0}},
02257 },
02258 },
02259 /* GC....UG */
02260 {{
02261 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02262 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02263 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02264 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02265 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02266 },
02267 {
02268 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02269 {{ 340, 340, 340, 340, 340},{ 340, 210, 180, 70, 200},{ 340, 170, 140, 30, 200},{ 340, 110, 80,
-30, 200},{ 340, 200, 200, 200, 200}},
02270 {{ 340, 340, 340, 340, 340},{ 340, 210, 180, 70, 200},{ 340, 270, 180, 170, 200},{ 340, 200, 200,
200, 200},{ 340, 270, 180, 170, 200}},
02271 {{ 340, 340, 340, 340, 340},{ 340, 110, 80, -30, 200},{ 340, 200, 200, 200, 200},{ 340, 150, 120,
10, 200},{ 340, 30, -100, -30, 200}},
02272 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 150, 140, 200},{ 340, 160, 30,
100, 200},{ 340, 230, 100, 170, 200}},
02273 },
02274 {
02275 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
```

```
02276 {{ 340, 340, 340, 340, 340},{ 340, 190, 250, 200, 250},{ 340, 140, 140, 200, 150},{ 340, 80, 180,
200, 190},{ 340, 200, 200, 200, 200}},
02277 {{ 340, 340, 340, 340, 340},{ 340, 190, 190, 200, 190},{ 340, 190, 190, 200, 190},{ 340, 200, 200,
200, 200},{ 340, 190, 190, 200, 190}},
02278 {{ 340, 340, 340, 340, 340},{ 340, 80, 180, 200, 190},{ 340, 200, 200, 200, 200},{ 340, 120, 180,
200, 190},{ 340, -90, 10, 200, 10}},
02279 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 160, 200, 160},{ 340, 30, 130,
200, 140},{ 340, 100, 100, 200, 110}}
02280 },
02281 {
02282 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02283 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 180, 40},{ 340, -30, 200, 130, -110},{ 340, -90, 200,
70, 10},{ 340, 200, 200, 200, 200}},
02284 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 180, -60},{ 340, 110, 200, 240, 40},{ 340, 200, 200,
200, 200},{ 340, 110, 200, 240, 40}},
02285 {{ 340, 340, 340, 340, 340},{ 340, -90, 200, 70, 10},{ 340, 200, 200, 200, 200},{ 340, -50, 200,
110, -30},{ 340, -90, 200, 0, -350}},
02286 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 80, 200, 210, 10},{ 340, 40, 200,
120, -220},{ 340, 110, 200, 190, 30}}
02287 },
02288 {
02289 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02290 {{ 340, 340, 340, 340, 340},{ 340, 200, 150, 0, 210},{ 340, 200, 40, -150, 70},{ 340, 200, 90,
-30, 190},{ 340, 200, 200, 200, 200}},
02291 {{ 340, 340, 340, 340, 340},{ 340, 200, 90, -100, 110},{ 340, 200, 90, 0, 110},{ 340, 200, 200,
200, 200},{ 340, 200, 90, 0, 110}},
02292 {{ 340, 340, 340, 340, 340},{ 340, 200, 90, -30, 190},{ 340, 200, 200, 200, 200},{ 340, 200, 80,
-70, 150},{ 340, 200, -90, -390, 10}},
02293 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 60, -30, 80},{ 340, 200, 40,
-260, 140},{ 340, 200, 0, -10, 30}}
02294 }
02295 },
02296 /* GC...AU */
02297 {{
02298 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02299 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02300 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02301 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02302 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02303 },
02304 {
02305 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02306 {{ 340, 340, 340, 340, 340},{ 340, 210, 180, 70, 200},{ 340, 190, 160, 50, 200},{ 340, 90, 60,
-50, 200},{ 340, 200, 200, 200, 200}},
02307 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, 60, 200},{ 340, 240, 150, 140, 200},{ 340, 200, 200,
200, 200},{ 340, 240, 150, 140, 200}},
02308 {{ 340, 340, 340, 340, 340},{ 340, 90, 60, -50, 200},{ 340, 200, 200, 200, 200},{ 340, 140, 110,
0, 200},{ 340, 70, -60, 10, 200}},
02309 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 150, 140, 200},{ 340, 170, 40,
110, 200},{ 340, 200, 70, 150, 200}}
02310 },
02311 {
02312 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02313 {{ 340, 340, 340, 340, 340},{ 340, 190, 250, 200, 250},{ 340, 160, 160, 200, 170},{ 340, 60, 160,
200, 170},{ 340, 200, 200, 200, 200}},
02314 {{ 340, 340, 340, 340, 340},{ 340, 170, 170, 200, 180},{ 340, 160, 160, 200, 160},{ 340, 200, 200,
200, 200},{ 340, 160, 160, 200, 160}},
02315 {{ 340, 340, 340, 340, 340},{ 340, 60, 160, 200, 170},{ 340, 200, 200, 200, 200},{ 340, 120, 180,
200, 180},{ 340, -50, 50, 200, 50}},
02316 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 160, 200, 160},{ 340, 40, 140,
200, 150},{ 340, 80, 80, 200, 80}}
02317 },
02318 {
02319 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02320 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 180, 40},{ 340, -10, 200, 150, -90},{ 340, -110, 200,
50, -10},{ 340, 200, 200, 200, 200}},
02321 {{ 340, 340, 340, 340, 340},{ 340, 0, 200, 160, -80},{ 340, 80, 200, 210, 10},{ 340, 200, 200,
200, 200},{ 340, 80, 200, 210, 10}},
02322 {{ 340, 340, 340, 340, 340},{ 340, -110, 200, 50, -10},{ 340, 200, 200, 200, 200},{ 340, -60, 200,
110, -30},{ 340, -50, 200, 40, -310}},
02323 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 80, 200, 210, 10},{ 340, 50, 200,
130, -210},{ 340, 80, 200, 170, 10}}
02324 },
02325 {
02326 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02327 {{ 340, 340, 340, 340, 340},{ 340, 200, 150, 0, 210},{ 340, 200, 60, -130, 90},{ 340, 200, 70,
```

```
-50, 170},{ 340, 200, 200, 200, 200}},
02328 {{ 340, 340, 340, 340, 340},{ 340, 200, 70, -120, 100},{ 340, 200, 60, -30, 80},{ 340, 200, 200,
200, 200},{ 340, 200, 60, -30, 80}},
02329 {{ 340, 340, 340, 340, 340},{ 340, 200, 70, -50, 170},{ 340, 200, 200, 200, 200},{ 340, 200, 80,
-70, 140},{ 340, 200, -50, -350, 50}},
02330 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 60, -30, 80},{ 340, 200, 50,
-250, 150},{ 340, 200, -20, -30, 0}}
02331 }
02332 },
02333 /* GC....UA */
02334 {{
02335 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02336 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02337 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02338 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02339 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02340 },
02341 {
02342 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02343 {{ 340, 340, 340, 340, 340},{ 340, 210, 180, 70, 200},{ 340, 170, 140, 30, 200},{ 340, 110, 80,
-30, 200},{ 340, 200, 200, 200, 200}},
02344 {{ 340, 340, 340, 340, 340},{ 340, 210, 180, 70, 200},{ 340, 270, 180, 170, 200},{ 340, 200, 200,
200, 200},{ 340, 270, 180, 170, 200}},
02345 {{ 340, 340, 340, 340, 340},{ 340, 110, 80, -30, 200},{ 340, 200, 200, 200, 200},{ 340, 150, 120,
10, 200},{ 340, 30, -100, -30, 200}},
02346 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 150, 140, 200},{ 340, 160, 30,
100, 200},{ 340, 230, 100, 170, 200}}
02347 },
02348 {
02349 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02350 {{ 340, 340, 340, 340, 340},{ 340, 190, 250, 200, 250},{ 340, 140, 140, 200, 150},{ 340, 80, 180,
200, 190},{ 340, 200, 200, 200, 200}},
02351 {{ 340, 340, 340, 340, 340},{ 340, 190, 190, 200, 190},{ 340, 190, 190, 200, 190},{ 340, 200, 200,
200, 200},{ 340, 190, 190, 200, 190}},
02352 {{ 340, 340, 340, 340, 340},{ 340, 80, 180, 200, 190},{ 340, 200, 200, 200, 200},{ 340, 120, 180,
200, 190},{ 340, -90, 10, 200, 10}},
02353 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 160, 160, 200, 160},{ 340, 30, 130,
200, 140},{ 340, 100, 100, 200, 110}}
02354 },
02355 {
02356 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02357 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 180, 40},{ 340, -30, 200, 130, -110},{ 340, -90, 200,
70, 10},{ 340, 200, 200, 200, 200}},
02358 {{ 340, 340, 340, 340, 340},{ 340, 10, 200, 180, -60},{ 340, 110, 200, 240, 40},{ 340, 200, 200,
200, 200},{ 340, 110, 200, 240, 40}},
02359 {{ 340, 340, 340, 340, 340},{ 340, -90, 200, 70, 10},{ 340, 200, 200, 200, 200},{ 340, -50, 200,
110, -30},{ 340, -90, 200, 0, -350}},
02360 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 80, 200, 210, 10},{ 340, 40, 200,
120, -220},{ 340, 110, 200, 190, 30}}
02361 },
02362 {
02363 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02364 {{ 340, 340, 340, 340, 340},{ 340, 200, 150, 0, 210},{ 340, 200, 40, -150, 70},{ 340, 200, 90,
-30, 190},{ 340, 200, 200, 200, 200}},
02365 {{ 340, 340, 340, 340, 340},{ 340, 200, 90, -100, 110},{ 340, 200, 90, 0, 110},{ 340, 200, 200,
200, 200},{ 340, 200, 90, 0, 110}},
02366 {{ 340, 340, 340, 340, 340},{ 340, 110, 90, -30, 190},{ 340, 200, 200, 200, 200},{ 340, 200, 80,
-70, 150},{ 340, 200, -90, -390, 10}},
02367 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 60, -30, 80},{ 340, 200, 40,
-260, 140},{ 340, 200, 0, -10, 30}}
02368 }
02369 },
02370 /* GC....?? */
02371 {{
02372 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02373 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02374 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02375 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02376 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02377 },
02378 {
02379 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
```

Generated by Doxygen

```
340, 340},{ 340, 340, 340, 340, 340}},
02433 {{ 340, 340, 340, 340, 340},{ 340, 80, 200, 130, 160},{ 340, 70, 200, 120, 50},{ 340, -20, 200,
30, 140},{ 340, 200, 200, 200, 200}},
02434 {{ 340, 340, 340, 340, 340},{ 340, 110, 200, 170, 90},{ 340, 200, 200, 210, 180},{ 340, 200, 200,
200, 200},{ 340, 180, 200, 200, 160}},
02435 {{ 340, 340, 340, 340, 340},{ 340, -20, 200, 30, 140},{ 340, 200, 200, 200, 200},{ 340, 50, 200,
110, 130},{ 340, -10, 200, -40, -210}},
02436 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 180, 200, 200, 160},{ 340, 140, 200,
110, -60},{ 340, 180, 200, 150, 160}}
02437 },
02438 {
02439 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02440 {{ 340, 340, 340, 340, 340},{ 340, 200, 230, 60, 190},{ 340, 200, 160, -50, 80},{ 340, 200, 170,
40, 180},{ 340, 200, 200, 200, 200}},
02441 {{ 340, 340, 340, 340, 340},{ 340, 200, 210, 0, 130},{ 340, 200, 190, 80, 110},{ 340, 200, 200,
200, 200},{ 340, 200, 180, 70, 100}},
02442 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, 40, 180},{ 340, 200, 200, 200, 200},{ 340, 200, 210,
40, 170},{ 340, 200, 0, -310, 0}},
02443 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 180, 70, 100},{ 340, 200, 150,
-160, 160},{ 340, 200, 90, 60, 10}}
02444 },
02445 },
02446 /* GU....GC */
02447 {{
02448 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02449 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02450 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02451 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02452 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02453 },
02454 {
02455 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02456 {{ 340, 340, 340, 340, 340},{ 340, 210, 200, 90, 200},{ 340, 190, 170, 60, 200},{ 340, 10, 0,
-110, 200},{ 340, 200, 200, 200, 200}},
02457 {{ 340, 340, 340, 340, 340},{ 340, 180, 170, 60, 200},{ 340, 250, 170, 160, 200},{ 340, 200, 200,
200, 200},{ 340, 150, 70, 70, 200}},
02458 {{ 340, 340, 340, 340, 340},{ 340, 70, 60, -50, 200},{ 340, 200, 200, 200, 200},{ 340, 180, 160,
50, 200},{ 340, 0, -120, -50, 200}},
02459 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 250, 180, 170, 200},{ 340, 40, -80,
-10, 200},{ 340, 210, 100, 170, 200}}
02460 },
02461 {
02462 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02463 {{ 340, 340, 340, 340, 340},{ 340, 190, 240, 200, 240},{ 340, 160, 160, 200, 160},{ 340, -10, 80,
200, 80},{ 340, 200, 200, 200, 200}},
02464 {{ 340, 340, 340, 340, 340},{ 340, 160, 150, 200, 150},{ 340, 160, 160, 200, 160},{ 340, 200, 200,
200, 200},{ 340, 60, 60, 200, 60}},
02465 {{ 340, 340, 340, 340, 340},{ 340, 50, 140, 200, 140},{ 340, 200, 200, 200, 200},{ 340, 150, 210,
200, 210},{ 340, -130, -30, 200, -30}},
02466 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 160, 200, 160},{ 340, -90, 10,
200, 10},{ 340, 90, 80, 200, 80}}
02467 },
02468 {
02469 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02470 {{ 340, 340, 340, 340, 340},{ 340, 90, 200, 140, 170},{ 340, 60, 200, 120, 40},{ 340, -110, 200,
-60, 50},{ 340, 200, 200, 200, 200}},
02471 {{ 340, 340, 340, 340, 340},{ 340, 60, 200, 110, 40},{ 340, 160, 200, 180, 140},{ 340, 200, 200,
200, 200},{ 340, 70, 200, 80, 50}},
02472 {{ 340, 340, 340, 340, 340},{ 340, -50, 200, 0, 110},{ 340, 200, 200, 200, 200},{ 340, 50, 200,
110, 130},{ 340, -50, 200, -70, -250}},
02473 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 200, 180, 150},{ 340, -10, 200,
-30, -210},{ 340, 170, 200, 140, 150}}
02474 },
02475 {
02476 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02477 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 70, 200},{ 340, 200, 160, -50, 80},{ 340, 200, 80,
-50, 80},{ 340, 200, 200, 200, 200}},
02478 {{ 340, 340, 340, 340, 340},{ 340, 200, 150, -60, 70},{ 340, 200, 160, 50, 80},{ 340, 200, 200,
200, 200},{ 340, 200, 60, -50, -20}},
02479 {{ 340, 340, 340, 340, 340},{ 340, 200, 140, 10, 150},{ 340, 200, 200, 200, 200},{ 340, 200, 210,
40, 170},{ 340, 200, -30, -350, -30}},
02480 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 160, 50, 80},{ 340, 200, 10,
-310, 10},{ 340, 200, 80, 50, 0}}
02481 },
02482 },
02483 /* GU....GU */
02484 {
```



```
02485 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02486 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02487 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02488 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02489 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02490 },
02491 {
02492 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02493 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 250, 240, 130, 200},{ 340, 150, 140,
30, 200},{ 340, 200, 200, 200, 200}},
02494 {{ 340, 340, 340, 340, 340},{ 340, 260, 250, 140, 200},{ 340, 310, 230, 220, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 230, 220, 200}},
02495 {{ 340, 340, 340, 340, 340},{ 340, 150, 140, 30, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 190,
80, 200},{ 340, 130, 20, 90, 200}},
02496 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 230, 120,
190, 200},{ 340, 270, 150, 220, 200}},
02497 },
02498 {
02499 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02500 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 230, 220, 200, 220},{ 340, 130, 220,
200, 220},{ 340, 200, 200, 200, 200}},
02501 {{ 340, 340, 340, 340, 340},{ 340, 240, 230, 200, 230},{ 340, 220, 220, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 220, 220, 200, 220}},
02502 {{ 340, 340, 340, 340, 340},{ 340, 130, 220, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 180, 240,
200, 240},{ 340, 10, 100, 200, 100}},
02503 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 110, 200,
200, 200},{ 340, 140, 140, 200, 140}},
02504 },
02505 {
02506 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02507 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 130, 200, 180, 110},{ 340, 30, 200,
80, 190},{ 340, 200, 200, 200, 200}},
02508 {{ 340, 340, 340, 340, 340},{ 340, 140, 200, 190, 120},{ 340, 220, 200, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 220, 200, 240, 200}},
02509 {{ 340, 340, 340, 340, 340},{ 340, 30, 200, 80, 190},{ 340, 200, 200, 200, 200},{ 340, 80, 200,
140, 160},{ 340, 90, 200, 70, -110}},
02510 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 190, 200,
160, -10},{ 340, 220, 200, 200, 200}},
02511 },
02512 {
02513 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02514 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 220, 10, 140},{ 340, 200, 220,
90, 220},{ 340, 200, 200, 200, 200}},
02515 {{ 340, 340, 340, 340, 340},{ 340, 200, 230, 20, 150},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
200, 200},{ 340, 200, 220, 100, 140}},
02516 {{ 340, 340, 340, 340, 340},{ 340, 200, 220, 90, 220},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 100, -210, 110}},
02517 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
-110, 200},{ 340, 200, 140, 110, 60}},
02518 },
02519 },
02520 /* GU....UG */
02521 {
02522 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02523 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02524 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02525 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02526 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02527 },
02528 {
02529 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02530 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 230, 220, 110, 200},{ 340, 170, 160,
50, 200},{ 340, 200, 200, 200, 200}},
02531 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 340, 260, 250, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 260, 250, 200}},
02532 {{ 340, 340, 340, 340, 340},{ 340, 170, 160, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 200,
90, 200},{ 340, 100, -20, 50, 200}},
02533 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 220, 110,
180, 200},{ 340, 290, 180, 250, 200}},
02534 },
02535 {
02536 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
```

```
340, 340},{ 340, 340, 340, 340, 340}},
02537 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 210, 200, 200, 200},{ 340, 150, 240,
200, 240},{ 340, 200, 200, 200, 200}},
02538 {{ 340, 340, 340, 340, 340},{ 340, 250, 250, 200, 250},{ 340, 250, 250, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 250, 250, 200, 250}},
02539 {{ 340, 340, 340, 340, 340},{ 340, 150, 240, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 190, 240,
200, 240},{ 340, -30, 70, 200, 70}},
02540 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 100, 190,
200, 190},{ 340, 170, 160, 200, 160}}
02541 },
02542 {
02543 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02544 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 110, 200, 160, 90},{ 340, 50, 200,
100, 210},{ 340, 200, 200, 200, 200}},
02545 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 130},{ 340, 250, 200, 270, 230},{ 340, 200, 200,
200, 200},{ 340, 250, 200, 270, 230}},
02546 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 100, 210},{ 340, 200, 200, 200, 200},{ 340, 90, 200,
140, 170},{ 340, 50, 200, 30, -150}},
02547 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 180, 200,
150, -20},{ 340, 250, 200, 220, 230}}
02548 },
02549 {
02550 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02551 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 200, -10, 120},{ 340, 200, 240,
110, 240},{ 340, 200, 200, 200, 200}},
02552 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 30, 170},{ 340, 200, 250, 130, 170},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 130, 170}},
02553 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 110, 240},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 70, -250, 70}},
02554 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 190,
-120, 190},{ 340, 200, 160, 130, 80}}
02555 },
02556 },
02557 /* GU...AU */
02558 {
02559 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02560 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02561 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02562 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02563 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02564 },
02565 {
02566 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02567 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 250, 240, 130, 200},{ 340, 150, 140,
30, 200},{ 340, 200, 200, 200, 200}},
02568 {{ 340, 340, 340, 340, 340},{ 340, 260, 250, 140, 200},{ 340, 310, 230, 220, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 230, 220, 200}},
02569 {{ 340, 340, 340, 340, 340},{ 340, 150, 140, 30, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 190,
80, 200},{ 340, 130, 20, 90, 200}},
02570 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 230, 120,
190, 200},{ 340, 270, 150, 220, 200}}
02571 },
02572 {
02573 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02574 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 230, 220, 200, 220},{ 340, 130, 220,
200, 220},{ 340, 200, 200, 200, 200}},
02575 {{ 340, 340, 340, 340, 340},{ 340, 240, 230, 200, 230},{ 340, 220, 220, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 220, 220, 200, 220}},
02576 {{ 340, 340, 340, 340, 340},{ 340, 130, 220, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 180, 240,
200, 240},{ 340, 10, 100, 200, 100}},
02577 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 110, 200,
200, 200},{ 340, 140, 140, 200, 140}}
02578 },
02579 {
02580 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02581 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 130, 200, 180, 110},{ 340, 30, 200,
80, 190},{ 340, 200, 200, 200, 200}},
02582 {{ 340, 340, 340, 340, 340},{ 340, 140, 200, 190, 120},{ 340, 220, 200, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 220, 200, 240, 200}},
02583 {{ 340, 340, 340, 340, 340},{ 340, 30, 200, 80, 190},{ 340, 200, 200, 200, 200},{ 340, 80, 200,
140, 160},{ 340, 90, 200, 70, -110}},
02584 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 190, 200,
160, -10},{ 340, 220, 200, 200, 200}}
02585 },
02586 {
02587 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
```

```
02588 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 220, 10, 140},{ 340, 200, 220,
90, 220},{ 340, 200, 200, 200, 200}},
02589 {{ 340, 340, 340, 340, 340},{ 340, 200, 230, 20, 150},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
200, 200},{ 340, 200, 220, 100, 140}},
02590 {{ 340, 340, 340, 340, 340},{ 340, 200, 220, 90, 220},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 100, -210, 110}},
02591 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
-110, 200},{ 340, 200, 140, 110, 60}}
02592 }
02593 },
02594 /* GU....UA */
02595 {{
02596 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02597 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02598 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02599 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02600 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02601 },
02602 {
02603 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02604 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 230, 220, 110, 200},{ 340, 170, 160,
50, 200},{ 340, 200, 200, 200, 200}},
02605 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 340, 260, 250, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 260, 250, 200}},
02606 {{ 340, 340, 340, 340, 340},{ 340, 170, 160, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 200,
90, 200},{ 340, 100, -20, 50, 200}},
02607 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 220, 110,
180, 200},{ 340, 290, 180, 250, 200}}
02608 },
02609 {
02610 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02611 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 210, 200, 200, 200},{ 340, 150, 240,
200, 240},{ 340, 200, 200, 200, 200}},
02612 {{ 340, 340, 340, 340, 340},{ 340, 250, 250, 200, 250},{ 340, 250, 250, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 250, 250, 200, 250}},
02613 {{ 340, 340, 340, 340, 340},{ 340, 150, 240, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 190, 240,
200, 240},{ 340, -30, 70, 200, 70}},
02614 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 100, 190,
200, 190},{ 340, 170, 160, 200, 160}}
02615 },
02616 {
02617 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02618 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 110, 200, 160, 90},{ 340, 50, 200,
100, 210},{ 340, 200, 200, 200, 200}},
02619 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 130},{ 340, 250, 200, 270, 230},{ 340, 200, 200,
200, 200},{ 340, 250, 200, 270, 230}},
02620 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 100, 210},{ 340, 200, 200, 200, 200},{ 340, 90, 200,
140, 170},{ 340, 50, 200, 30, -150}},
02621 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 180, 200,
150, -20},{ 340, 250, 200, 220, 230}}
02622 },
02623 {
02624 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02625 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 200, -10, 120},{ 340, 200, 240,
110, 240},{ 340, 200, 200, 200, 200}},
02626 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 30, 170},{ 340, 200, 250, 130, 170},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 130, 170}},
02627 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 110, 240},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 70, -250, 70}},
02628 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 190,
-120, 190},{ 340, 200, 160, 130, 80}}
02629 }
02630 },
02631 /* GU....?? */
02632 {{
02633 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02634 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02635 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02636 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02637 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02638 },
02639 {
02640 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
```

Generated by Doxygen

```
02693 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02694 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 140, 150},{ 340, 90, 200, 130, 40},{ 340, 0, 200,
40, 130},{ 340, 200, 200, 200, 200}},
02695 {{ 340, 340, 340, 340, 340},{ 340, 130, 200, 170, 80},{ 340, 220, 200, 220, 170},{ 340, 200, 200,
200, 200},{ 340, 200, 200, 200, 150}},
02696 {{ 340, 340, 340, 340, 340},{ 340, 0, 200, 40, 130},{ 340, 200, 200, 200, 200},{ 340, 70, 200,
110, 120},{ 340, 10, 200, -30, -220}},
02697 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 200, 200, 150},{ 340, 160, 200,
120, -70},{ 340, 190, 200, 150, 150}}
02698 },
02699 {
02700 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02701 {{ 340, 340, 340, 340, 340},{ 340, 200, 260, 20, 220},{ 340, 200, 190, -90, 110},{ 340, 200, 200,
0, 200},{ 340, 200, 200, 200, 200}},
02702 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, -40, 150},{ 340, 200, 220, 40, 140},{ 340, 200, 200,
200, 200},{ 340, 200, 210, 30, 120}},
02703 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 0, 200},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
0, 190},{ 340, 200, 30, -350, 30}},
02704 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 210, 30, 120},{ 340, 200, 180,
-200, 180},{ 340, 200, 120, 20, 30}}
02705 }
02706 },
02707 /* UG....GC */
02708 {
02709 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02710 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02711 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02712 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02713 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02714 },
02715 {
02716 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02717 {{ 340, 340, 340, 340, 340},{ 340, 210, 210, 110, 200},{ 340, 190, 190, 80, 200},{ 340, 10, 10,
-90, 200},{ 340, 200, 200, 200, 200}},
02718 {{ 340, 340, 340, 340, 340},{ 340, 180, 180, 80, 200},{ 340, 250, 190, 180, 200},{ 340, 200, 200,
200, 200},{ 340, 150, 90, 90, 200}},
02719 {{ 340, 340, 340, 340, 340},{ 340, 70, 70, -30, 200},{ 340, 200, 200, 200, 200},{ 340, 180, 180,
70, 200},{ 340, 0, -100, -30, 200}},
02720 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 250, 190, 190, 200},{ 340, 40, -60,
10, 200},{ 340, 210, 110, 190, 200}}
02721 },
02722 {
02723 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02724 {{ 340, 340, 340, 340, 340},{ 340, 170, 270, 200, 240},{ 340, 140, 190, 200, 160},{ 340, -30, 110,
200, 80},{ 340, 200, 200, 200, 200}},
02725 {{ 340, 340, 340, 340, 340},{ 340, 140, 180, 200, 150},{ 340, 140, 190, 200, 160},{ 340, 200, 200,
200, 200},{ 340, 40, 90, 200, 60}},
02726 {{ 340, 340, 340, 340, 340},{ 340, 30, 170, 200, 140},{ 340, 200, 200, 200, 200},{ 340, 130, 240,
200, 210},{ 340, -150, 0, 200, -30}},
02727 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 150, 190, 200, 160},{ 340, -110, 40,
200, 10},{ 340, 70, 110, 200, 80}}
02728 },
02729 {
02730 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02731 {{ 340, 340, 340, 340, 340},{ 340, 110, 200, 150, 160},{ 340, 80, 200, 120, 30},{ 340, -90, 200,
-50, 40},{ 340, 200, 200, 200, 200}},
02732 {{ 340, 340, 340, 340, 340},{ 340, 80, 200, 120, 30},{ 340, 180, 200, 180, 130},{ 340, 200, 200,
200, 200},{ 340, 90, 200, 80, 40}},
02733 {{ 340, 340, 340, 340, 340},{ 340, -30, 200, 10, 100},{ 340, 200, 200, 200, 200},{ 340, 70, 200,
110, 120},{ 340, -30, 200, -70, -260}},
02734 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 190, 200, 190, 140},{ 340, 10, 200,
-30, -220},{ 340, 190, 200, 150, 140}}
02735 },
02736 {
02737 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02738 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 30, 230},{ 340, 200, 190, -90, 100},{ 340, 200, 110,
-90, 110},{ 340, 200, 200, 200, 200}},
02739 {{ 340, 340, 340, 340, 340},{ 340, 200, 180, -100, 100},{ 340, 200, 190, 10, 100},{ 340, 200, 200,
200, 200},{ 340, 200, 90, -90, 0}},
02740 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, -30, 170},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
0, 190},{ 340, 200, 0, -390, -10}},
02741 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 190, 10, 110},{ 340, 200, 40,
-350, 30},{ 340, 200, 110, 10, 30}}
02742 },
02743 },
02744 /* UG....GU */
```

```
02745 {{
02746 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02747 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02748 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02749 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02750 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02751 },
02752 {
02753 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02754 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 250, 250, 150, 200},{ 340, 150, 150,
50, 200},{ 340, 200, 200, 200, 200}},
02755 {{ 340, 340, 340, 340, 340},{ 340, 260, 260, 160, 200},{ 340, 310, 250, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 250, 240, 200}},
02756 {{ 340, 340, 340, 340, 340},{ 340, 150, 150, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 210,
100, 200},{ 340, 130, 30, 110, 200}},
02757 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 250, 240, 200},{ 340, 230, 130,
210, 200},{ 340, 270, 170, 240, 200}},
02758 },
02759 {
02760 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02761 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 210, 250, 200, 220},{ 340, 110, 250,
200, 220},{ 340, 200, 200, 200, 200}},
02762 {{ 340, 340, 340, 340, 340},{ 340, 220, 260, 200, 230},{ 340, 200, 250, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 200, 220}},
02763 {{ 340, 340, 340, 340, 340},{ 340, 110, 250, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 160, 270,
200, 240},{ 340, -10, 130, 200, 100}},
02764 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 90, 230,
200, 200},{ 340, 120, 170, 200, 140}},
02765 },
02766 {
02767 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02768 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 150, 200, 190, 100},{ 340, 50, 200,
90, 180},{ 340, 200, 200, 200, 200}},
02769 {{ 340, 340, 340, 340, 340},{ 340, 160, 200, 200, 110},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
200, 200},{ 340, 240, 200, 240, 190}},
02770 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 90, 180},{ 340, 200, 200, 200, 200},{ 340, 100, 200,
140, 150},{ 340, 110, 200, 70, -120}},
02771 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 210, 200,
170, -20},{ 340, 240, 200, 200, 190}},
02772 },
02773 {
02774 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02775 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 250, -30, 170},{ 340, 200, 250,
50, 250},{ 340, 200, 200, 200, 200}},
02776 {{ 340, 340, 340, 340, 340},{ 340, 200, 260, -20, 180},{ 340, 200, 250, 70, 160},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 70, 160}},
02777 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 50, 250},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 220},{ 340, 200, 130, -250, 130}},
02778 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 230,
-150, 230},{ 340, 200, 170, 70, 80}},
02779 }
02780 },
02781 /* UG....UG */
02782 {{
02783 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02784 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02785 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02786 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02787 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02788 },
02789 {
02790 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02791 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 230, 230, 130, 200},{ 340, 170, 170,
70, 200},{ 340, 200, 200, 200, 200}},
02792 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 340, 280, 270, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 280, 270, 200}},
02793 {{ 340, 340, 340, 340, 340},{ 340, 170, 170, 70, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 210,
110, 200},{ 340, 100, 0, 70, 200}},
02794 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 250, 240, 200},{ 340, 220, 120,
200, 200},{ 340, 290, 190, 270, 200}},
02795 },
02796 {
```

```
02797 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02798 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 190, 230, 200, 200},{ 340, 130, 270,
200, 240},{ 340, 200, 200, 200, 200}},
02799 {{ 340, 340, 340, 340, 340},{ 340, 230, 280, 200, 250},{ 340, 230, 280, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 230, 280, 200, 250}},
02800 {{ 340, 340, 340, 340, 340},{ 340, 130, 270, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 170, 270,
200, 240},{ 340, -50, 100, 200, 70}},
02801 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 80, 220,
200, 190},{ 340, 150, 190, 200, 160}}
02802 },
02803 {
02804 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02805 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 130, 200, 170, 80},{ 340, 70, 200,
110, 200},{ 340, 200, 200, 200, 200}},
02806 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 120},{ 340, 270, 200, 270, 220},{ 340, 200, 200,
200, 200},{ 340, 270, 200, 270, 220}},
02807 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 110, 200},{ 340, 200, 200, 200, 200},{ 340, 110, 200,
150, 160},{ 340, 70, 200, 30, -160}},
02808 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
160, -30},{ 340, 270, 200, 230, 220}}
02809 },
02810 {
02811 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02812 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 230, -50, 150},{ 340, 200, 270,
70, 270},{ 340, 200, 200, 200, 200}},
02813 {{ 340, 340, 340, 340, 340},{ 340, 200, 280, 0, 190},{ 340, 200, 280, 100, 190},{ 340, 200, 200,
200, 200},{ 340, 200, 280, 100, 190}},
02814 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 70, 270},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 230},{ 340, 200, 100, -290, 90}},
02815 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 220,
-160, 220},{ 340, 200, 190, 90, 110}}
02816 },
02817 },
02818 /* UG...AU */
02819 {{
02820 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02821 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02822 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02823 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02824 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02825 },
02826 {
02827 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02828 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 250, 250, 150, 200},{ 340, 150, 150,
50, 200},{ 340, 200, 200, 200, 200}},
02829 {{ 340, 340, 340, 340, 340},{ 340, 260, 260, 160, 200},{ 340, 310, 250, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 250, 240, 200}},
02830 {{ 340, 340, 340, 340, 340},{ 340, 150, 150, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 210,
100, 200},{ 340, 130, 30, 110, 200}},
02831 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 250, 240, 200},{ 340, 230, 130,
210, 200},{ 340, 270, 170, 240, 200}}
02832 },
02833 {
02834 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02835 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 210, 250, 200, 220},{ 340, 110, 250,
200, 220},{ 340, 200, 200, 200, 200}},
02836 {{ 340, 340, 340, 340, 340},{ 340, 220, 260, 200, 230},{ 340, 200, 250, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 200, 220}},
02837 {{ 340, 340, 340, 340, 340},{ 340, 110, 250, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 160, 270,
200, 240},{ 340, -10, 130, 200, 100}},
02838 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 90, 230,
200, 200},{ 340, 120, 170, 200, 140}}
02839 },
02840 {
02841 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02842 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 150, 200, 190, 100},{ 340, 50, 200,
90, 180},{ 340, 200, 200, 200, 200}},
02843 {{ 340, 340, 340, 340, 340},{ 340, 160, 200, 200, 110},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
200, 200},{ 340, 240, 200, 200, 190}},
02844 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 90, 180},{ 340, 200, 200, 200, 200},{ 340, 100, 200,
140, 150},{ 340, 110, 200, 70, -120}},
02845 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 210, 200,
170, -20},{ 340, 240, 200, 200, 190}}
02846 },
02847 {
02848 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
```

```
340, 340},{ 340, 340, 340, 340, 340}},
02849 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 250, -30, 170},{ 340, 200, 250,
50, 250},{ 340, 200, 200, 200, 200}},
02850 {{ 340, 340, 340, 340, 340},{ 340, 200, 260, -20, 180},{ 340, 200, 250, 70, 160},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 70, 160}},
02851 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 50, 250},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 220},{ 340, 200, 130, -250, 130}},
02852 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 230,
-150, 230},{ 340, 200, 170, 70, 80}}
02853 }
02854 },
02855 /* UG....UA */
02856 {{
02857 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02858 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02859 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02860 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02861 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02862 },
02863 {
02864 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02865 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 230, 230, 130, 200},{ 340, 170, 170,
70, 200},{ 340, 200, 200, 200, 200}},
02866 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 340, 280, 270, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 280, 270, 200}},
02867 {{ 340, 340, 340, 340, 340},{ 340, 170, 170, 70, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 210,
110, 200},{ 340, 100, 0, 70, 200}},
02868 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 250, 240, 200},{ 340, 220, 120,
200, 200},{ 340, 290, 190, 270, 200}}
02869 },
02870 {
02871 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02872 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 190, 230, 200, 200},{ 340, 130, 270,
200, 240},{ 340, 200, 200, 200, 200}},
02873 {{ 340, 340, 340, 340, 340},{ 340, 230, 280, 200, 250},{ 340, 230, 280, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 230, 280, 200, 250}},
02874 {{ 340, 340, 340, 340, 340},{ 340, 130, 270, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 170, 270,
200, 240},{ 340, -50, 100, 200, 70}},
02875 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 80, 220,
200, 190},{ 340, 150, 190, 200, 160}}
02876 },
02877 {
02878 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02879 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 130, 200, 170, 80},{ 340, 70, 200,
110, 200},{ 340, 200, 200, 200, 200}},
02880 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 120},{ 340, 270, 200, 270, 220},{ 340, 200, 200,
200, 200},{ 340, 270, 200, 270, 220}},
02881 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 110, 200},{ 340, 200, 200, 200, 200},{ 340, 110, 200,
150, 160},{ 340, 70, 200, 30, -160}},
02882 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
160, -30},{ 340, 270, 200, 230, 220}}
02883 },
02884 {
02885 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02886 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 230, -50, 150},{ 340, 200, 270,
70, 270},{ 340, 200, 200, 200, 200}},
02887 {{ 340, 340, 340, 340, 340},{ 340, 200, 280, 0, 190},{ 340, 200, 280, 100, 190},{ 340, 200, 200,
200, 200},{ 340, 200, 280, 100, 190}},
02888 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 70, 270},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 230},{ 340, 200, 100, -290, 90}},
02889 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 220,
-160, 220},{ 340, 200, 190, 90, 110}}
02890 }
02891 },
02892 /* UG....?? */
02893 {{
02894 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02895 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02896 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02897 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02898 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
02899 },
02900 {
```


Generated by Doxygen

```
02953 {
02954 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02955 {{ 340, 340, 340, 340, 340},{ 340, 80, 200, 130, 160},{ 340, 70, 200, 120, 50},{ 340, -20, 200,
30, 140},{ 340, 200, 200, 200, 200}},
02956 {{ 340, 340, 340, 340, 340},{ 340, 110, 200, 170, 90},{ 340, 200, 200, 210, 180},{ 340, 200, 200,
200, 200},{ 340, 180, 200, 200, 160}},
02957 {{ 340, 340, 340, 340, 340},{ 340, -20, 200, 30, 140},{ 340, 200, 200, 200, 200},{ 340, 50, 200,
110, 130},{ 340, -10, 200, -40, -210}},
02958 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 180, 200, 200, 160},{ 340, 140, 200,
110, -60},{ 340, 180, 200, 150, 160}}
02959 },
02960 {
02961 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02962 {{ 340, 340, 340, 340, 340},{ 340, 200, 230, 60, 190},{ 340, 200, 160, -50, 80},{ 340, 200, 170,
40, 180},{ 340, 200, 200, 200, 200}},
02963 {{ 340, 340, 340, 340, 340},{ 340, 200, 210, 0, 130},{ 340, 200, 190, 80, 110},{ 340, 200, 200,
200, 200},{ 340, 200, 180, 70, 100}},
02964 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, 40, 180},{ 340, 200, 200, 200, 200},{ 340, 200, 210,
40, 170},{ 340, 200, 0, -310, 0}},
02965 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 180, 70, 100},{ 340, 200, 150,
-160, 160},{ 340, 200, 90, 60, 10}}
02966 }
02967 },
02968 /* AU....GC */
02969 {{
02970 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02971 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02972 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02973 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02974 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02975 },
02976 {
02977 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02978 {{ 340, 340, 340, 340, 340},{ 340, 210, 200, 90, 200},{ 340, 190, 170, 60, 200},{ 340, 10, 0,
-110, 200},{ 340, 200, 200, 200, 200}},
02979 {{ 340, 340, 340, 340, 340},{ 340, 180, 170, 60, 200},{ 340, 250, 170, 160, 200},{ 340, 200, 200,
200, 200},{ 340, 150, 70, 70, 200}},
02980 {{ 340, 340, 340, 340, 340},{ 340, 70, 60, -50, 200},{ 340, 200, 200, 200, 200},{ 340, 180, 160,
50, 200},{ 340, 0, -120, -50, 200}},
02981 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 250, 180, 170, 200},{ 340, 40, -80,
-10, 200},{ 340, 210, 100, 170, 200}}
02982 },
02983 {
02984 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02985 {{ 340, 340, 340, 340, 340},{ 340, 190, 240, 200, 240},{ 340, 160, 160, 200, 160},{ 340, -10, 80,
200, 80},{ 340, 200, 200, 200, 200}},
02986 {{ 340, 340, 340, 340, 340},{ 340, 160, 150, 200, 150},{ 340, 160, 160, 200, 160},{ 340, 200, 200,
200, 200},{ 340, 60, 60, 200, 60}},
02987 {{ 340, 340, 340, 340, 340},{ 340, 50, 140, 200, 140},{ 340, 200, 200, 200, 200},{ 340, 150, 210,
200, 210},{ 340, -130, -30, 200, -30}},
02988 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 160, 200, 160},{ 340, -90, 10,
200, 10},{ 340, 90, 80, 200, 80}}
02989 },
02990 {
02991 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02992 {{ 340, 340, 340, 340, 340},{ 340, 90, 200, 140, 170},{ 340, 60, 200, 120, 40},{ 340, -110, 200,
-60, 50},{ 340, 200, 200, 200, 200}},
02993 {{ 340, 340, 340, 340, 340},{ 340, 60, 200, 110, 40},{ 340, 160, 200, 180, 140},{ 340, 200, 200,
200, 200},{ 340, 70, 200, 80, 50}},
02994 {{ 340, 340, 340, 340, 340},{ 340, -50, 200, 0, 110},{ 340, 200, 200, 200, 200},{ 340, 50, 200,
110, 130},{ 340, -50, 200, -70, -250}},
02995 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 170, 200, 180, 150},{ 340, -10, 200,
-30, -210},{ 340, 170, 200, 140, 150}}
02996 },
02997 {
02998 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
02999 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 70, 200},{ 340, 200, 160, -50, 80},{ 340, 200, 80,
-50, 80},{ 340, 200, 200, 200, 200}},
03000 {{ 340, 340, 340, 340, 340},{ 340, 200, 150, -60, 70},{ 340, 200, 160, 50, 80},{ 340, 200, 200,
200, 200},{ 340, 200, 60, -50, -20}},
03001 {{ 340, 340, 340, 340, 340},{ 340, 200, 140, 10, 150},{ 340, 200, 200, 200, 200},{ 340, 200, 210,
40, 170},{ 340, 200, -30, -350, -30}},
03002 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 160, 50, 80},{ 340, 200, 10,
-310, 10},{ 340, 200, 80, 50, 0}}
03003 }
03004 },
```

```
03005 /* AU...GU */
03006 {{
03007 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03008 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03009 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03010 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03011 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03012 },
03013 {
03014 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03015 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 250, 240, 130, 200},{ 340, 150, 140,
30, 200},{ 340, 200, 200, 200, 200}},
03016 {{ 340, 340, 340, 340, 340},{ 340, 260, 250, 140, 200},{ 340, 310, 230, 220, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 230, 220, 200}},
03017 {{ 340, 340, 340, 340, 340},{ 340, 150, 140, 30, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 190,
80, 200},{ 340, 130, 20, 90, 200}},
03018 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 230, 120,
190, 200},{ 340, 270, 150, 220, 200}},
03019 },
03020 {
03021 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03022 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 230, 220, 200, 220},{ 340, 130, 220,
200, 220},{ 340, 200, 200, 200, 200}},
03023 {{ 340, 340, 340, 340, 340},{ 340, 240, 230, 200, 230},{ 340, 220, 220, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 220, 220, 200, 220}},
03024 {{ 340, 340, 340, 340, 340},{ 340, 130, 220, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 180, 240,
200, 240},{ 340, 10, 100, 200, 100}},
03025 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 110, 200,
200, 200},{ 340, 140, 140, 200, 140}},
03026 },
03027 {
03028 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03029 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 130, 200, 180, 110},{ 340, 30, 200,
80, 190},{ 340, 200, 200, 200, 200}},
03030 {{ 340, 340, 340, 340, 340},{ 340, 140, 200, 190, 120},{ 340, 220, 200, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 220, 200, 240, 200}},
03031 {{ 340, 340, 340, 340, 340},{ 340, 30, 200, 80, 190},{ 340, 200, 200, 200, 200},{ 340, 80, 200,
140, 160},{ 340, 90, 200, 70, -110}},
03032 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 190, 200,
160, -10},{ 340, 220, 200, 200, 200}},
03033 },
03034 {
03035 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03036 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 220, 10, 140},{ 340, 200, 220,
90, 220},{ 340, 200, 200, 200, 200}},
03037 {{ 340, 340, 340, 340, 340},{ 340, 200, 230, 20, 150},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
200, 200},{ 340, 200, 220, 100, 140}},
03038 {{ 340, 340, 340, 340, 340},{ 340, 200, 220, 90, 220},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 100, -210, 110}},
03039 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
-110, 200},{ 340, 200, 140, 110, 60}},
03040 },
03041 },
03042 /* AU...UG */
03043 {{
03044 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03045 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03046 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340, 340, 340, 340}},
03047 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340, 340, 340, 340}},
03048 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340, 340, 340, 340}},
03049 },
03050 {
03051 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03052 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 230, 220, 110, 200},{ 340, 170, 160,
50, 200},{ 340, 200, 200, 200, 200}},
03053 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 340, 260, 250, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 260, 250, 200}},
03054 {{ 340, 340, 340, 340, 340},{ 340, 170, 160, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 200,
90, 200},{ 340, 100, -20, 50, 200}},
03055 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 220, 110,
180, 200},{ 340, 290, 180, 250, 200}},
03056 },
```

```
03057 {
03058 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03059 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 210, 200, 200, 200},{ 340, 150, 240,
200, 240},{ 340, 200, 200, 200, 200}},
03060 {{ 340, 340, 340, 340, 340},{ 340, 250, 250, 200, 250},{ 340, 250, 250, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 250, 250, 200, 250}},
03061 {{ 340, 340, 340, 340, 340},{ 340, 150, 240, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 190, 240,
200, 240},{ 340, -30, 70, 200, 70}},
03062 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 100, 190,
200, 190},{ 340, 170, 160, 200, 160}}
03063 },
03064 {
03065 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03066 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 110, 200, 160, 90},{ 340, 50, 200,
100, 210},{ 340, 200, 200, 200, 200}},
03067 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 130},{ 340, 250, 200, 270, 230},{ 340, 200, 200,
200, 200},{ 340, 250, 200, 270, 230}},
03068 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 100, 210},{ 340, 200, 200, 200, 200},{ 340, 90, 200,
140, 170},{ 340, 50, 200, 30, -150}},
03069 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 180, 200,
150, -20},{ 340, 250, 200, 220, 230}}
03070 },
03071 {
03072 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03073 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 200, -10, 120},{ 340, 200, 240,
110, 240},{ 340, 200, 200, 200, 200}},
03074 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 30, 170},{ 340, 200, 250, 130, 170},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 130, 170}},
03075 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 110, 240},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 70, -250, 70}},
03076 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 190,
-120, 190},{ 340, 200, 160, 130, 80}}
03077 }
03078 },
03079 /* AU...AU */
03080 {{
03081 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03082 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03083 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03084 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03085 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03086 },
03087 {
03088 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03089 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 250, 240, 130, 200},{ 340, 150, 140,
30, 200},{ 340, 200, 200, 200, 200}},
03090 {{ 340, 340, 340, 340, 340},{ 340, 260, 250, 140, 200},{ 340, 310, 230, 220, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 230, 220, 200}},
03091 {{ 340, 340, 340, 340, 340},{ 340, 150, 140, 30, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 190,
80, 200},{ 340, 130, 20, 90, 200}},
03092 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 230, 120,
190, 200},{ 340, 270, 150, 220, 200}}
03093 },
03094 {
03095 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03096 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 230, 220, 200, 220},{ 340, 130, 220,
200, 220},{ 340, 200, 200, 200, 200}},
03097 {{ 340, 340, 340, 340, 340},{ 340, 240, 230, 200, 230},{ 340, 220, 220, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 220, 220, 200, 220}},
03098 {{ 340, 340, 340, 340, 340},{ 340, 130, 220, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 180, 240,
200, 240},{ 340, 10, 100, 200, 100}},
03099 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 110, 200,
200, 200},{ 340, 140, 140, 200, 140}}
03100 },
03101 {
03102 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03103 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 130, 200, 180, 110},{ 340, 30, 200,
80, 190},{ 340, 200, 200, 200, 200}},
03104 {{ 340, 340, 340, 340, 340},{ 340, 140, 200, 190, 120},{ 340, 220, 200, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 220, 200, 240, 200}},
03105 {{ 340, 340, 340, 340, 340},{ 340, 30, 200, 80, 190},{ 340, 200, 200, 200, 200},{ 340, 80, 200,
140, 160},{ 340, 90, 200, 70, -110}},
03106 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 190, 200,
160, -10},{ 340, 220, 200, 200, 200}}
03107 },
03108 {
```

```
03109 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03110 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 220, 10, 140},{ 340, 200, 220,
90, 220},{ 340, 200, 200, 200, 200}},
03111 {{ 340, 340, 340, 340, 340},{ 340, 200, 230, 20, 150},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
200, 200},{ 340, 200, 220, 100, 140}},
03112 {{ 340, 340, 340, 340, 340},{ 340, 200, 220, 90, 220},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 100, -210, 110}},
03113 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 200,
-110, 200},{ 340, 200, 140, 110, 60}}
03114 }
03115 },
03116 /* AU...UA */
03117 {{
03118 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03119 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03120 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03121 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03122 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
03123 },
03124 {
03125 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03126 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 230, 220, 110, 200},{ 340, 170, 160,
50, 200},{ 340, 200, 200, 200, 200}},
03127 {{ 340, 340, 340, 340, 340},{ 340, 280, 260, 150, 200},{ 340, 340, 260, 250, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 260, 250, 200}},
03128 {{ 340, 340, 340, 340, 340},{ 340, 170, 160, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 200,
90, 200},{ 340, 100, -20, 50, 200}},
03129 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 230, 220, 200},{ 340, 220, 110,
180, 200},{ 340, 290, 180, 250, 200}}
03130 },
03131 {
03132 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03133 {{ 340, 340, 340, 340, 340},{ 340, 250, 310, 200, 310},{ 340, 210, 200, 200, 200},{ 340, 150, 240,
200, 240},{ 340, 200, 200, 200, 200}},
03134 {{ 340, 340, 340, 340, 340},{ 340, 250, 250, 200, 250},{ 340, 250, 250, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 250, 250, 200, 250}},
03135 {{ 340, 340, 340, 340, 340},{ 340, 150, 240, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 190, 240,
200, 240},{ 340, -30, 70, 200, 70}},
03136 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 220, 200, 220},{ 340, 100, 190,
200, 190},{ 340, 170, 160, 200, 160}}
03137 },
03138 {
03139 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03140 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 230},{ 340, 110, 200, 160, 90},{ 340, 50, 200,
100, 210},{ 340, 200, 200, 200, 200}},
03141 {{ 340, 340, 340, 340, 340},{ 340, 150, 200, 210, 130},{ 340, 250, 200, 270, 230},{ 340, 200, 200,
200, 200},{ 340, 250, 200, 270, 230}},
03142 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 100, 210},{ 340, 200, 200, 200, 200},{ 340, 90, 200,
140, 170},{ 340, 50, 200, 30, -150}},
03143 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 220, 200, 240, 200},{ 340, 180, 200,
150, -20},{ 340, 250, 200, 220, 230}}
03144 },
03145 {
03146 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03147 {{ 340, 340, 340, 340, 340},{ 340, 200, 310, 130, 270},{ 340, 200, 200, -10, 120},{ 340, 200, 240,
110, 240},{ 340, 200, 200, 200, 200}},
03148 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 30, 170},{ 340, 200, 250, 130, 170},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 130, 170}},
03149 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, 110, 240},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
70, 200},{ 340, 200, 70, -250, 70}},
03150 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 220, 100, 140},{ 340, 200, 190,
-120, 190},{ 340, 200, 160, 130, 80}}
03151 }
03152 },
03153 /* AU...?? */
03154 {{
03155 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03156 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03157 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03158 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03159 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
03160 },
```

Generated by Doxygen

```
03213 },
03214 {
03215 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03216 {{ 340, 340, 340, 340, 340},{ 340, 100, 200, 140, 150},{ 340, 90, 200, 130, 40},{ 340, 0, 200,
40, 130},{ 340, 200, 200, 200, 200}},
03217 {{ 340, 340, 340, 340, 340},{ 340, 130, 200, 170, 80},{ 340, 220, 200, 220, 170},{ 340, 200, 200,
200, 200},{ 340, 200, 200, 200, 150}},
03218 {{ 340, 340, 340, 340, 340},{ 340, 0, 200, 40, 130},{ 340, 200, 200, 200, 200},{ 340, 70, 200,
110, 120},{ 340, 10, 200, -30, -220}},
03219 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 200, 200, 150},{ 340, 160, 200,
120, -70},{ 340, 190, 200, 150, 150}}
03220 },
03221 {
03222 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03223 {{ 340, 340, 340, 340, 340},{ 340, 200, 260, 20, 220},{ 340, 200, 190, -90, 110},{ 340, 200, 200,
0, 200},{ 340, 200, 200, 200, 200}},
03224 {{ 340, 340, 340, 340, 340},{ 340, 200, 240, -40, 150},{ 340, 200, 220, 40, 140},{ 340, 200, 200,
200, 200},{ 340, 200, 210, 30, 120}},
03225 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 0, 200},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
0, 190},{ 340, 200, 30, -350, 30}},
03226 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 210, 30, 120},{ 340, 200, 180,
-200, 180},{ 340, 200, 120, 20, 30}}
03227 },
03228 },
03229 /* UA....GC */
03230 {{
03231 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03232 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03233 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03234 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03235 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
03236 },
03237 {
03238 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03239 {{ 340, 340, 340, 340, 340},{ 340, 210, 210, 110, 200},{ 340, 190, 190, 80, 200},{ 340, 10, 10,
-90, 200},{ 340, 200, 200, 200, 200}},
03240 {{ 340, 340, 340, 340, 340},{ 340, 180, 180, 80, 200},{ 340, 250, 190, 180, 200},{ 340, 200, 200,
200, 200},{ 340, 150, 90, 90, 200}},
03241 {{ 340, 340, 340, 340, 340},{ 340, 70, 70, -30, 200},{ 340, 200, 200, 200, 200},{ 340, 180, 180,
70, 200},{ 340, 0, -100, -30, 200}},
03242 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 250, 190, 190, 200},{ 340, 40, -60,
10, 200},{ 340, 210, 110, 190, 200}}
03243 },
03244 {
03245 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03246 {{ 340, 340, 340, 340, 340},{ 340, 170, 270, 200, 240},{ 340, 140, 190, 200, 160},{ 340, -30, 110,
200, 80},{ 340, 200, 200, 200, 200}},
03247 {{ 340, 340, 340, 340, 340},{ 340, 140, 180, 200, 150},{ 340, 140, 190, 200, 160},{ 340, 200, 200,
200, 200},{ 340, 40, 90, 200, 60}},
03248 {{ 340, 340, 340, 340, 340},{ 340, 30, 170, 200, 140},{ 340, 200, 200, 200, 200},{ 340, 130, 240,
200, 210},{ 340, -150, 0, 200, -30}},
03249 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 150, 190, 200, 160},{ 340, -110, 40,
200, 10},{ 340, 70, 110, 200, 80}}
03250 },
03251 {
03252 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03253 {{ 340, 340, 340, 340, 340},{ 340, 110, 200, 150, 160},{ 340, 80, 200, 120, 30},{ 340, -90, 200,
-50, 40},{ 340, 200, 200, 200, 200}},
03254 {{ 340, 340, 340, 340, 340},{ 340, 80, 200, 120, 30},{ 340, 180, 200, 180, 130},{ 340, 200, 200,
200, 200},{ 340, 90, 200, 80, 40}},
03255 {{ 340, 340, 340, 340, 340},{ 340, -30, 200, 10, 100},{ 340, 200, 200, 200, 200},{ 340, 70, 200,
110, 120},{ 340, -30, 200, -70, -260}},
03256 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 190, 200, 190, 140},{ 340, 10, 200,
-30, -220},{ 340, 190, 200, 150, 140}}
03257 },
03258 {
03259 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03260 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 30, 230},{ 340, 200, 190, -90, 100},{ 340, 200, 110,
-90, 110},{ 340, 200, 200, 200, 200}},
03261 {{ 340, 340, 340, 340, 340},{ 340, 200, 180, -100, 100},{ 340, 200, 190, 10, 100},{ 340, 200, 200,
200, 200},{ 340, 200, 90, -90, 0}},
03262 {{ 340, 340, 340, 340, 340},{ 340, 200, 170, -30, 170},{ 340, 200, 200, 200, 200},{ 340, 200, 240,
0, 190},{ 340, 200, 0, -390, -10}},
03263 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 190, 10, 110},{ 340, 200, 40,
-350, 30},{ 340, 200, 110, 10, 30}}
03264 }
```

```
03265 },
03266 /* UA....GU */
03267 {{
03268 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03269 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03270 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03271 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03272 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}}
03273 },
03274 {
03275 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03276 {{ 340, 340, 340, 340, 340}},{{ 340, 280, 280, 170, 200}},{{ 340, 250, 250, 150, 200}},{{ 340, 150, 150,
50, 200}},{{ 340, 200, 200, 200, 200}},
03277 {{ 340, 340, 340, 340, 340}},{{ 340, 260, 260, 160, 200}},{{ 340, 310, 250, 240, 200}},{{ 340, 200, 200,
200, 200}},{{ 340, 310, 250, 240, 200}},
03278 {{ 340, 340, 340, 340, 340}},{{ 340, 150, 150, 50, 200}},{{ 340, 200, 200, 200, 200}},{{ 340, 210, 210,
100, 200}},{{ 340, 130, 30, 110, 200}},
03279 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 200, 200, 200}},{{ 340, 310, 250, 240, 200}},{{ 340, 230, 130,
210, 200}},{{ 340, 270, 170, 240, 200}}
03280 },
03281 {
03282 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03283 {{ 340, 340, 340, 340, 340}},{{ 340, 230, 340, 200, 310}},{{ 340, 210, 250, 200, 220}},{{ 340, 110, 250,
200, 220}},{{ 340, 200, 200, 200, 200}},
03284 {{ 340, 340, 340, 340, 340}},{{ 340, 220, 260, 200, 230}},{{ 340, 200, 250, 200, 220}},{{ 340, 200, 200,
200, 200}},{{ 340, 200, 250, 200, 220}},
03285 {{ 340, 340, 340, 340, 340}},{{ 340, 110, 250, 200, 220}},{{ 340, 200, 200, 200, 200}},{{ 340, 160, 270,
200, 240}},{{ 340, -10, 130, 200, 100}},
03286 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 200, 200, 200}},{{ 340, 200, 250, 200, 220}},{{ 340, 90, 230,
200, 200}},{{ 340, 120, 170, 200, 140}}
03287 },
03288 {
03289 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03290 {{ 340, 340, 340, 340, 340}},{{ 340, 170, 200, 210, 220}},{{ 340, 150, 200, 190, 100}},{{ 340, 50, 200,
90, 180}},{{ 340, 200, 200, 200, 200}},
03291 {{ 340, 340, 340, 340, 340}},{{ 340, 160, 200, 200, 110}},{{ 340, 240, 200, 240, 190}},{{ 340, 200, 200,
200, 200}},{{ 340, 240, 200, 240, 190}},
03292 {{ 340, 340, 340, 340, 340}},{{ 340, 50, 200, 90, 180}},{{ 340, 200, 200, 200, 200}},{{ 340, 100, 200,
140, 150}},{{ 340, 110, 200, 70, -120}},
03293 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 200, 200, 200}},{{ 340, 240, 200, 240, 190}},{{ 340, 210, 200,
170, -20}},{{ 340, 240, 200, 200, 190}}
03294 },
03295 {
03296 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03297 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 340, 100, 290}},{{ 340, 200, 250, -30, 170}},{{ 340, 200, 250,
50, 250}},{{ 340, 200, 200, 200, 200}},
03298 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 260, -20, 180}},{{ 340, 200, 250, 70, 160}},{{ 340, 200, 200,
200, 200}},{{ 340, 200, 250, 70, 160}},
03299 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 250, 50, 250}},{{ 340, 200, 200, 200, 200}},{{ 340, 200, 270,
30, 220}},{{ 340, 200, 130, -250, 130}},
03300 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 200, 200, 200}},{{ 340, 200, 250, 70, 160}},{{ 340, 200, 230,
-150, 230}},{{ 340, 200, 170, 70, 80}}
03301 },
03302 },
03303 /* UA....UG */
03304 {{
03305 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03306 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03307 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03308 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03309 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}}
03310 },
03311 {
03312 {{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340, 340, 340}},{{ 340, 340, 340,
340, 340}},{{ 340, 340, 340, 340, 340}},
03313 {{ 340, 340, 340, 340, 340}},{{ 340, 280, 280, 170, 200}},{{ 340, 230, 230, 130, 200}},{{ 340, 170, 170,
70, 200}},{{ 340, 200, 200, 200, 200}},
03314 {{ 340, 340, 340, 340, 340}},{{ 340, 280, 280, 170, 200}},{{ 340, 340, 280, 270, 200}},{{ 340, 200, 200,
200, 200}},{{ 340, 340, 280, 270, 200}},
03315 {{ 340, 340, 340, 340, 340}},{{ 340, 170, 170, 70, 200}},{{ 340, 200, 200, 200, 200}},{{ 340, 210, 210,
110, 200}},{{ 340, 100, 0, 70, 200}},
03316 {{ 340, 340, 340, 340, 340}},{{ 340, 200, 200, 200, 200}},{{ 340, 310, 250, 240, 200}},{{ 340, 220, 120,
200, 200}},{{ 340, 290, 190, 270, 200}}
```



```
03317 },
03318 {
03319 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03320 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 190, 230, 200, 200},{ 340, 130, 270,
200, 240},{ 340, 200, 200, 200, 200}},
03321 {{ 340, 340, 340, 340, 340},{ 340, 230, 280, 200, 250},{ 340, 230, 280, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 230, 280, 200, 250}},
03322 {{ 340, 340, 340, 340, 340},{ 340, 130, 270, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 170, 270,
200, 240},{ 340, -50, 100, 200, 70}},
03323 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 80, 220,
200, 190},{ 340, 150, 190, 200, 160}},
03324 },
03325 {
03326 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03327 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 130, 200, 170, 80},{ 340, 70, 200,
110, 200},{ 340, 200, 200, 200, 200}},
03328 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 120},{ 340, 270, 200, 270, 220},{ 340, 200, 200,
200, 200},{ 340, 270, 200, 270, 220}},
03329 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 110, 200},{ 340, 200, 200, 200, 200},{ 340, 110, 200,
150, 160},{ 340, 70, 200, 30, -160}},
03330 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
160, -30},{ 340, 270, 200, 230, 220}},
03331 },
03332 {
03333 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03334 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 230, -50, 150},{ 340, 200, 270,
70, 270},{ 340, 200, 200, 200, 200}},
03335 {{ 340, 340, 340, 340, 340},{ 340, 200, 280, 0, 190},{ 340, 200, 280, 100, 190},{ 340, 200, 200,
200, 200},{ 340, 200, 280, 100, 190}},
03336 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 70, 270},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 230},{ 340, 200, 100, -290, 90}},
03337 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 220,
-160, 220},{ 340, 200, 190, 90, 110}},
03338 },
03339 },
03340 /* UA...AU */
03341 {{
03342 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03343 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03344 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340, 340, 340, 340, 340},{ 340, 340, 340, 340,
340, 340, 340, 340},{ 340, 340, 340, 340, 340}},
03345 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03346 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03347 },
03348 {
03349 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03350 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 250, 250, 150, 200},{ 340, 150, 150,
50, 200},{ 340, 200, 200, 200, 200}},
03351 {{ 340, 340, 340, 340, 340},{ 340, 260, 260, 160, 200},{ 340, 310, 250, 240, 200},{ 340, 200, 200,
200, 200},{ 340, 310, 250, 240, 200}},
03352 {{ 340, 340, 340, 340, 340},{ 340, 150, 150, 50, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 210,
100, 200},{ 340, 130, 30, 110, 200}},
03353 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 250, 240, 200},{ 340, 230, 130,
210, 200},{ 340, 270, 170, 240, 200}},
03354 },
03355 {
03356 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03357 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 210, 250, 200, 220},{ 340, 110, 250,
200, 220},{ 340, 200, 200, 200, 200}},
03358 {{ 340, 340, 340, 340, 340},{ 340, 220, 260, 200, 230},{ 340, 200, 250, 200, 220},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 200, 220}},
03359 {{ 340, 340, 340, 340, 340},{ 340, 110, 250, 200, 220},{ 340, 200, 200, 200, 200},{ 340, 160, 270,
200, 240},{ 340, -10, 130, 200, 100}},
03360 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 90, 230,
200, 200},{ 340, 120, 170, 200, 140}},
03361 },
03362 {
03363 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03364 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 150, 200, 190, 100},{ 340, 50, 200,
90, 180},{ 340, 200, 200, 200, 200}},
03365 {{ 340, 340, 340, 340, 340},{ 340, 160, 200, 200, 110},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
200, 200},{ 340, 240, 200, 240, 190}},
03366 {{ 340, 340, 340, 340, 340},{ 340, 50, 200, 90, 180},{ 340, 200, 200, 200, 200},{ 340, 100, 200,
140, 150},{ 340, 110, 200, 70, -120}},
03367 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 210, 200,
170, -20},{ 340, 240, 200, 200, 190}},
03368 },
```

```
03369 {
03370 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03371 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 250, -30, 170},{ 340, 200, 250,
50, 250},{ 340, 200, 200, 200, 200}},
03372 {{ 340, 340, 340, 340, 340},{ 340, 200, 260, -20, 180},{ 340, 200, 250, 70, 160},{ 340, 200, 200,
200, 200},{ 340, 200, 250, 70, 160}},
03373 {{ 340, 340, 340, 340, 340},{ 340, 200, 250, 50, 250},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 220},{ 340, 200, 130, -250, 130}},
03374 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 230,
-150, 230},{ 340, 200, 170, 70, 80}}
03375 }
03376 },
03377 /* UA....UA */
03378 {{
03379 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03380 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03381 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03382 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03383 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03384 },
03385 {
03386 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03387 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 230, 230, 130, 200},{ 340, 170, 170,
70, 200},{ 340, 200, 200, 200, 200}},
03388 {{ 340, 340, 340, 340, 340},{ 340, 280, 280, 170, 200},{ 340, 340, 280, 270, 200},{ 340, 200, 200,
200, 200},{ 340, 340, 280, 270, 200}},
03389 {{ 340, 340, 340, 340, 340},{ 340, 170, 170, 70, 200},{ 340, 200, 200, 200, 200},{ 340, 210, 210,
110, 200},{ 340, 100, 0, 70, 200}},
03390 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 310, 250, 240, 200},{ 340, 220, 120,
200, 200},{ 340, 290, 190, 270, 200}}
03391 },
03392 {
03393 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03394 {{ 340, 340, 340, 340, 340},{ 340, 230, 340, 200, 310},{ 340, 190, 230, 200, 200},{ 340, 130, 270,
200, 240},{ 340, 200, 200, 200, 200}},
03395 {{ 340, 340, 340, 340, 340},{ 340, 230, 280, 200, 250},{ 340, 230, 280, 200, 250},{ 340, 200, 200,
200, 200},{ 340, 230, 280, 200, 250}},
03396 {{ 340, 340, 340, 340, 340},{ 340, 130, 270, 200, 240},{ 340, 200, 200, 200, 200},{ 340, 170, 270,
200, 240},{ 340, -50, 100, 200, 70}},
03397 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 200, 220},{ 340, 80, 220,
200, 190},{ 340, 150, 190, 200, 160}}
03398 },
03399 {
03400 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03401 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 220},{ 340, 130, 200, 170, 80},{ 340, 70, 200,
110, 200},{ 340, 200, 200, 200, 200}},
03402 {{ 340, 340, 340, 340, 340},{ 340, 170, 200, 210, 120},{ 340, 270, 200, 270, 220},{ 340, 200, 200,
200, 200},{ 340, 270, 200, 270, 220}},
03403 {{ 340, 340, 340, 340, 340},{ 340, 70, 200, 110, 200},{ 340, 200, 200, 200, 200},{ 340, 110, 200,
150, 160},{ 340, 70, 200, 30, -160}},
03404 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 240, 200, 240, 190},{ 340, 200, 200,
160, -30},{ 340, 270, 200, 230, 220}}
03405 },
03406 {
03407 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03408 {{ 340, 340, 340, 340, 340},{ 340, 200, 340, 100, 290},{ 340, 200, 230, -50, 150},{ 340, 200, 270,
70, 270},{ 340, 200, 200, 200, 200}},
03409 {{ 340, 340, 340, 340, 340},{ 340, 200, 280, 0, 190},{ 340, 200, 280, 100, 190},{ 340, 200, 200,
200, 200},{ 340, 200, 280, 100, 190}},
03410 {{ 340, 340, 340, 340, 340},{ 340, 200, 270, 70, 270},{ 340, 200, 200, 200, 200},{ 340, 200, 270,
30, 230},{ 340, 200, 100, -290, 90}},
03411 {{ 340, 340, 340, 340, 340},{ 340, 200, 200, 200, 200},{ 340, 200, 250, 70, 160},{ 340, 200, 220,
-160, 220},{ 340, 200, 190, 90, 110}}
03412 }
03413 },
03414 /* UA....?? */
03415 {{
03416 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03417 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03418 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03419 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03420 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}}
03421 }}
```

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

```
340, 340},{ 340, 340, 340, 340, 340}}
03682 },
03683 {
03684 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03685 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03686 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03687 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03688 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03689 },
03690 {
03691 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03692 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03693 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03694 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03695 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03696 },
03697 {
03698 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03699 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03700 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03701 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03702 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03703 },
03704 {
03705 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03706 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03707 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03708 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03709 {{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340, 340, 340},{ 340, 340, 340,
340, 340},{ 340, 340, 340, 340, 340}},
03710 }
03711 }
03712 }
03713 };
03714
03715 PRIVATE int int22_H_184[NBPAIRS+1][NBPAIRS+1][5][5][5][5] =
03716 { /* noPair */ {{{{{{0}}}}},
03717 { /* noPair */ {{{{0}}}},
03718 /* CG.@@..CG */
03719 { { { 0, 0, 0, 0, 0},
03720 { DEF, DEF, DEF, DEF, DEF},
03721 { DEF, DEF, DEF, DEF, DEF},
03722 { DEF, DEF, DEF, DEF, DEF},
03723 { DEF, DEF, DEF, DEF, DEF}},
03724 /* CG.@A..CG */
03725 { { 0, 0, 0, 0, 0},
03726 {-1029,-1029,-1029,-1029,-1029},
03727 { -519, -519, -519, -519, -519},
03728 { -939, -939, -939, -939, -939},
03729 { -809, -809, -809, -809, -809}},
03730 /* CG.@C..CG */
03731 { { 0, 0, 0, 0, 0},
03732 { -949, -949, -949, -949, -949},
03733 { -449, -449, -449, -449, -449},
03734 { -939, -939, -939, -939, -939},
03735 { -739, -739, -739, -739, -739}},
03736 /* CG.@G..CG */
03737 { { 0, 0, 0, 0, 0},
03738 {-1029,-1029,-1029,-1029,-1029},
03739 { -519, -519, -519, -519, -519},
03740 { -939, -939, -939, -939, -939},
03741 { -809, -809, -809, -809, -809}},
03742 /* CG.@U..CG */
03743 { { 0, 0, 0, 0, 0},
03744 {-1029,-1029,-1029,-1029,-1029},
03745 { -669, -669, -669, -669, -669},
03746 { -939, -939, -939, -939, -939},
03747 { -859, -859, -859, -859, -859}}},
```



```
03748 /* CG.A@.CG */
03749 {{ DEF, -1029, -949, -1029, -1029 },
03750 { -100, -1079, -999, -1079, -1079 },
03751 { -100, -1079, -999, -1079, -1079 },
03752 { -100, -1079, -999, -1079, -1079 },
03753 { -100, -1079, -999, -1079, -1079 }},
03754 /* CG.AA.CG */
03755 {{ DEF, -1029, -949, -1029, -1029 },
03756 { -1079, -2058, -1978, -2058, -2058 },
03757 { -569, -1548, -1468, -1548, -1548 },
03758 { -989, -1968, -1888, -1968, -1968 },
03759 { -859, -1838, -1758, -1838, -1838 }},
03760 /* CG.AC.CG */
03761 {{ DEF, -1029, -949, -1029, -1029 },
03762 { -999, -1978, -1898, -1978, -1978 },
03763 { -499, -1478, -1398, -1478, -1478 },
03764 { -989, -1968, -1888, -1968, -1968 },
03765 { -789, -1768, -1688, -1768, -1768 }},
03766 /* CG.AG.CG */
03767 {{ DEF, -1029, -949, -1029, -1029 },
03768 { -1079, -2058, -1978, -2058, -2058 },
03769 { -569, -1548, -1468, -1548, -1548 },
03770 { -989, -1968, -1888, -1968, -1968 },
03771 { -859, -1838, -1758, -1838, -1838 }},
03772 /* CG.AU.CG */
03773 {{ DEF, -1029, -949, -1029, -1029 },
03774 { -1079, -2058, -1978, -2058, -2058 },
03775 { -719, -1698, -1618, -1698, -1698 },
03776 { -989, -1968, -1888, -1968, -1968 },
03777 { -909, -1888, -1808, -1888, -1888 }},
03778 /* CG.C@.CG */
03779 {{ DEF, -519, -449, -519, -669 },
03780 { -100, -569, -499, -569, -719 },
03781 { -100, -569, -499, -569, -719 },
03782 { -100, -569, -499, -569, -719 },
03783 { -100, -569, -499, -569, -719 }},
03784 /* CG.CA.CG */
03785 {{ DEF, -519, -449, -519, -669 },
03786 { -1079, -1548, -1478, -1548, -1698 },
03787 { -569, -1038, -968, -1038, -1188 },
03788 { -989, -1458, -1388, -1458, -1608 },
03789 { -859, -1328, -1258, -1328, -1478 }},
03790 /* CG.CC.CG */
03791 {{ DEF, -519, -449, -519, -669 },
03792 { -999, -1468, -1398, -1468, -1618 },
03793 { -499, -968, -898, -968, -1118 },
03794 { -989, -1458, -1388, -1458, -1608 },
03795 { -789, -1258, -1188, -1258, -1408 }},
03796 /* CG.CG.CG */
03797 {{ DEF, -519, -449, -519, -669 },
03798 { -1079, -1548, -1478, -1548, -1698 },
03799 { -569, -1038, -968, -1038, -1188 },
03800 { -989, -1458, -1388, -1458, -1608 },
03801 { -859, -1328, -1258, -1328, -1478 }},
03802 /* CG.CU.CG */
03803 {{ DEF, -519, -449, -519, -669 },
03804 { -1079, -1548, -1478, -1548, -1698 },
03805 { -719, -1188, -1118, -1188, -1338 },
03806 { -989, -1458, -1388, -1458, -1608 },
03807 { -909, -1378, -1308, -1378, -1528 }},
03808 /* CG.G@.CG */
03809 {{ DEF, -939, -939, -939, -939 },
03810 { -100, -989, -989, -989, -989 },
03811 { -100, -989, -989, -989, -989 },
03812 { -100, -989, -989, -989, -989 },
03813 { -100, -989, -989, -989, -989 }},
03814 /* CG.GA.CG */
03815 {{ DEF, -939, -939, -939, -939 },
03816 { -1079, -1968, -1968, -1968, -1968 },
03817 { -569, -1458, -1458, -1458, -1458 },
03818 { -989, -1878, -1878, -1878, -1878 },
03819 { -859, -1748, -1748, -1748, -1748 }},
03820 /* CG.GC.CG */
03821 {{ DEF, -939, -939, -939, -939 },
03822 { -999, -1888, -1888, -1888, -1888 },
03823 { -499, -1388, -1388, -1388, -1388 },
03824 { -989, -1878, -1878, -1878, -1878 },
03825 { -789, -1678, -1678, -1678, -1678 }},
03826 /* CG.GG.CG */
03827 {{ DEF, -939, -939, -939, -939 },
03828 { -1079, -1968, -1968, -1968, -1968 },
03829 { -569, -1458, -1458, -1458, -1458 },
03830 { -989, -1878, -1878, -1878, -1878 },
03831 { -859, -1748, -1748, -1748, -1748 }},
03832 /* CG.GU.CG */
03833 {{ DEF, -939, -939, -939, -939 },
03834 { -1079, -1968, -1968, -1968, -1968 },
```

```

03835 { -719,-1608,-1608,-1608,-1608},
03836 { -989,-1878,-1878,-1878,-1878},
03837 { -909,-1798,-1798,-1798,-1798}},
03838 /* CG.U@.CG */
03839 {{ DEF, -809, -739, -809, -859},
03840 { -100, -859, -789, -859, -909},
03841 { -100, -859, -789, -859, -909},
03842 { -100, -859, -789, -859, -909},
03843 { -100, -859, -789, -859, -909}},
03844 /* CG.UA.CG */
03845 {{ DEF, -809, -739, -809, -859},
03846 {-1079,-1838,-1768,-1838,-1888},
03847 { -569,-1328,-1258,-1328,-1378},
03848 { -989,-1748,-1678,-1748,-1798},
03849 { -859,-1618,-1548,-1618,-1668}},
03850 /* CG.UC.CG */
03851 {{ DEF, -809, -739, -809, -859},
03852 { -999,-1758,-1688,-1758,-1808},
03853 { -499,-1258,-1188,-1258,-1308},
03854 { -989,-1748,-1678,-1748,-1798},
03855 { -789,-1548,-1478,-1548,-1598}},
03856 /* CG.UG.CG */
03857 {{ DEF, -809, -739, -809, -859},
03858 {-1079,-1838,-1768,-1838,-1888},
03859 { -569,-1328,-1258,-1328,-1378},
03860 { -989,-1748,-1678,-1748,-1798},
03861 { -859,-1618,-1548,-1618,-1668}},
03862 /* CG.UU.CG */
03863 {{ DEF, -809, -739, -809, -859},
03864 {-1079,-1838,-1768,-1838,-1888},
03865 { -719,-1478,-1408,-1478,-1528},
03866 { -989,-1748,-1678,-1748,-1798},
03867 { -909,-1668,-1598,-1668,-1718}}}},
03868 /* CG.@@.GC */
03869 {{{ 0, 0, 0, 0, 0},
03870 { DEF, DEF, DEF, DEF, DEF},
03871 { DEF, DEF, DEF, DEF, DEF},
03872 { DEF, DEF, DEF, DEF, DEF},
03873 { DEF, DEF, DEF, DEF, DEF}}},
03874 /* CG.@A.GC */
03875 {{ 0, 0, 0, 0, 0},
03876 { -519, -519, -519, -519, -519},
03877 { -719, -719, -719, -719, -719},
03878 { -709, -709, -709, -709, -709},
03879 { -499, -499, -499, -499, -499}},
03880 /* CG.@C.GC */
03881 {{ 0, 0, 0, 0, 0},
03882 { -879, -879, -879, -879, -879},
03883 { -309, -309, -309, -309, -309},
03884 { -739, -739, -739, -739, -739},
03885 { -499, -499, -499, -499, -499}},
03886 /* CG.@G.GC */
03887 {{ 0, 0, 0, 0, 0},
03888 { -559, -559, -559, -559, -559},
03889 { -309, -309, -309, -309, -309},
03890 { -619, -619, -619, -619, -619},
03891 { -499, -499, -499, -499, -499}},
03892 /* CG.@U.GC */
03893 {{ 0, 0, 0, 0, 0},
03894 { -879, -879, -879, -879, -879},
03895 { -389, -389, -389, -389, -389},
03896 { -739, -739, -739, -739, -739},
03897 { -569, -569, -569, -569, -569}}}},
03898 /* CG.A@.GC */
03899 {{{ DEF,-1029, -949,-1029,-1029},
03900 { -100,-1079, -999,-1079,-1079},
03901 { -100,-1079, -999,-1079,-1079},
03902 { -100,-1079, -999,-1079,-1079},
03903 { -100,-1079, -999,-1079,-1079}},
03904 /* CG.AA.GC */
03905 {{ DEF,-1029, -949,-1029,-1029},
03906 { -569,-1548,-1468,-1548,-1548},
03907 { -769,-1748,-1668,-1748,-1748},
03908 { -759,-1738,-1658,-1738,-1738},
03909 { -549,-1528,-1448,-1528,-1528}},
03910 /* CG.AC.GC */
03911 {{ DEF,-1029, -949,-1029,-1029},
03912 { -929,-1908,-1828,-1908,-1908},
03913 { -359,-1338,-1258,-1338,-1338},
03914 { -789,-1768,-1688,-1768,-1768},
03915 { -549,-1528,-1448,-1528,-1528}},
03916 /* CG.AG.GC */
03917 {{ DEF,-1029, -949,-1029,-1029},
03918 { -609,-1588,-1508,-1588,-1588},
03919 { -359,-1338,-1258,-1338,-1338},
03920 { -669,-1648,-1568,-1648,-1648},
03921 { -549,-1528,-1448,-1528,-1528}},

```

```

03922 /* CG.AU..GC */
03923 {{ DEF, -1029, -949, -1029, -1029},
03924 { -929, -1908, -1828, -1908, -1908},
03925 { -439, -1418, -1338, -1418, -1418},
03926 { -789, -1768, -1688, -1768, -1768},
03927 { -619, -1598, -1518, -1598, -1598}}},
03928 /* CG.C@..GC */
03929 {{ DEF, -519, -449, -519, -669},
03930 { -100, -569, -499, -569, -719},
03931 { -100, -569, -499, -569, -719},
03932 { -100, -569, -499, -569, -719},
03933 { -100, -569, -499, -569, -719}}},
03934 /* CG.CA..GC */
03935 {{ DEF, -519, -449, -519, -669},
03936 { -569, -1038, -968, -1038, -1188},
03937 { -769, -1238, -1168, -1238, -1388},
03938 { -759, -1228, -1158, -1228, -1378},
03939 { -549, -1018, -948, -1018, -1168}}},
03940 /* CG.CC..GC */
03941 {{ DEF, -519, -449, -519, -669},
03942 { -929, -1398, -1328, -1398, -1548},
03943 { -359, -828, -758, -828, -978},
03944 { -789, -1258, -1188, -1258, -1408},
03945 { -549, -1018, -948, -1018, -1168}}},
03946 /* CG.CG..GC */
03947 {{ DEF, -519, -449, -519, -669},
03948 { -609, -1078, -1008, -1078, -1228},
03949 { -359, -828, -758, -828, -978},
03950 { -669, -1138, -1068, -1138, -1288},
03951 { -549, -1018, -948, -1018, -1168}}},
03952 /* CG.CU..GC */
03953 {{ DEF, -519, -449, -519, -669},
03954 { -929, -1398, -1328, -1398, -1548},
03955 { -439, -908, -838, -908, -1058},
03956 { -789, -1258, -1188, -1258, -1408},
03957 { -619, -1088, -1018, -1088, -1238}}},
03958 /* CG.G@..GC */
03959 {{ DEF, -939, -939, -939, -939},
03960 { -100, -989, -989, -989, -989},
03961 { -100, -989, -989, -989, -989},
03962 { -100, -989, -989, -989, -989},
03963 { -100, -989, -989, -989, -989}}},
03964 /* CG.GA..GC */
03965 {{ DEF, -939, -939, -939, -939},
03966 { -569, -1458, -1458, -1458, -1458},
03967 { -769, -1658, -1658, -1658, -1658},
03968 { -759, -1648, -1648, -1648, -1648},
03969 { -549, -1438, -1438, -1438, -1438}}},
03970 /* CG.GC..GC */
03971 {{ DEF, -939, -939, -939, -939},
03972 { -929, -1818, -1818, -1818, -1818},
03973 { -359, -1248, -1248, -1248, -1248},
03974 { -789, -1678, -1678, -1678, -1678},
03975 { -549, -1438, -1438, -1438, -1438}}},
03976 /* CG.GG..GC */
03977 {{ DEF, -939, -939, -939, -939},
03978 { -609, -1498, -1498, -1498, -1498},
03979 { -359, -1248, -1248, -1248, -1248},
03980 { -669, -1558, -1558, -1558, -1558},
03981 { -549, -1438, -1438, -1438, -1438}}},
03982 /* CG.GU..GC */
03983 {{ DEF, -939, -939, -939, -939},
03984 { -929, -1818, -1818, -1818, -1818},
03985 { -439, -1328, -1328, -1328, -1328},
03986 { -789, -1678, -1678, -1678, -3080},
03987 { -619, -1508, -1508, -1508, -1508}}},
03988 /* CG.U@..GC */
03989 {{ DEF, -809, -739, -809, -859},
03990 { -100, -859, -789, -859, -909},
03991 { -100, -859, -789, -859, -909},
03992 { -100, -859, -789, -859, -909},
03993 { -100, -859, -789, -859, -909}}},
03994 /* CG.UA..GC */
03995 {{ DEF, -809, -739, -809, -859},
03996 { -569, -1328, -1258, -1328, -1378},
03997 { -769, -1528, -1458, -1528, -1578},
03998 { -759, -1518, -1448, -1518, -1568},
03999 { -549, -1308, -1238, -1308, -1358}}},
04000 /* CG.UC..GC */
04001 {{ DEF, -809, -739, -809, -859},
04002 { -929, -1688, -1618, -1688, -1738},
04003 { -359, -1118, -1048, -1118, -1168},
04004 { -789, -1548, -1478, -1548, -1598},
04005 { -549, -1308, -1238, -1308, -1358}}},
04006 /* CG.UG..GC */
04007 {{ DEF, -809, -739, -809, -859},
04008 { -609, -1368, -1298, -1368, -1418},

```

```

04009 { -359,-1118,-1048,-1118,-1168},
04010 { -669,-1428,-1358,-1428,-1478},
04011 { -549,-1308,-1238,-1308,-1358}},
04012 /* CG.UU..GC */
04013 {{ DEF, -809, -739, -809, -859},
04014 { -929,-1688,-1618,-1688,-1738},
04015 { -439,-1198,-1128,-1198,-1248},
04016 { -789,-1548,-1478,-1548,-1598},
04017 { -619,-1378,-1308,-1378,-1428}}}},
04018 /* CG.@@..GU */
04019 {{{ 0, 0, 0, 0, 0},
04020 { DEF, DEF, DEF, DEF, DEF},
04021 { DEF, DEF, DEF, DEF, DEF},
04022 { DEF, DEF, DEF, DEF, DEF},
04023 { DEF, DEF, DEF, DEF, DEF}}},
04024 /* CG.@A..GU */
04025 {{ 0, 0, 0, 0, 0},
04026 { -429, -429, -429, -429, -429},
04027 { -259, -259, -259, -259, -259},
04028 { -339, -339, -339, -339, -339},
04029 { -329, -329, -329, -329, -329}},
04030 /* CG.@C..GU */
04031 {{ 0, 0, 0, 0, 0},
04032 { -599, -599, -599, -599, -599},
04033 { -239, -239, -239, -239, -239},
04034 { -689, -689, -689, -689, -689},
04035 { -329, -329, -329, -329, -329}},
04036 /* CG.@G..GU */
04037 {{ 0, 0, 0, 0, 0},
04038 { -599, -599, -599, -599, -599},
04039 { -239, -239, -239, -239, -239},
04040 { -689, -689, -689, -689, -689},
04041 { -329, -329, -329, -329, -329}},
04042 /* CG.@U..GU */
04043 {{ 0, 0, 0, 0, 0},
04044 { -599, -599, -599, -599, -599},
04045 { -239, -239, -239, -239, -239},
04046 { -689, -689, -689, -689, -689},
04047 { -329, -329, -329, -329, -329}}}},
04048 /* CG.A@..GU */
04049 {{{ DEF,-1029, -949,-1029,-1029},
04050 { -100,-1079, -999,-1079,-1079},
04051 { -100,-1079, -999,-1079,-1079},
04052 { -100,-1079, -999,-1079,-1079},
04053 { -100,-1079, -999,-1079,-1079}},
04054 /* CG.AA..GU */
04055 {{ DEF,-1029, -949,-1029,-1029},
04056 { -479,-1458,-1378,-1458,-1458},
04057 { -309,-1288,-1208,-1288,-1288},
04058 { -389,-1368,-1288,-1368,-1368},
04059 { -379,-1358,-1278,-1358,-1358}},
04060 /* CG.AC..GU */
04061 {{ DEF,-1029, -949,-1029,-1029},
04062 { -649,-1628,-1548,-1628,-1628},
04063 { -289,-1268,-1188,-1268,-1268},
04064 { -739,-1718,-1638,-1718,-1718},
04065 { -379,-1358,-1278,-1358,-1358}},
04066 /* CG.AG..GU */
04067 {{ DEF,-1029, -949,-1029,-1029},
04068 { -649,-1628,-1548,-1628,-1628},
04069 { -289,-1268,-1188,-1268,-1268},
04070 { -739,-1718,-1638,-1718,-1718},
04071 { -379,-1358,-1278,-1358,-1358}},
04072 /* CG.AU..GU */
04073 {{ DEF,-1029, -949,-1029,-1029},
04074 { -649,-1628,-1548,-1628,-1628},
04075 { -289,-1268,-1188,-1268,-1268},
04076 { -739,-1718,-1638,-1718,-1718},
04077 { -379,-1358,-1278,-1358,-1358}}}},
04078 /* CG.C@..GU */
04079 {{{ DEF, -519, -449, -519, -669},
04080 { -100, -569, -499, -569, -719},
04081 { -100, -569, -499, -569, -719},
04082 { -100, -569, -499, -569, -719},
04083 { -100, -569, -499, -569, -719}},
04084 /* CG.CA..GU */
04085 {{ DEF, -519, -449, -519, -669},
04086 { -479, -948, -878, -948,-1098},
04087 { -309, -778, -708, -778, -928},
04088 { -389, -858, -788, -858,-1008},
04089 { -379, -848, -778, -848, -998}},
04090 /* CG.CC..GU */
04091 {{ DEF, -519, -449, -519, -669},
04092 { -649,-1118,-1048,-1118,-1268},
04093 { -289, -758, -688, -758, -908},
04094 { -739,-1208,-1138,-1208,-1358},
04095 { -379, -848, -778, -848, -998}},

```

```

04096 /* CG.CG..GU */
04097 {{ DEF, -519, -449, -519, -669},
04098 { -649,-1118,-1048,-1118,-1268},
04099 { -289, -758, -688, -758, -908},
04100 { -739,-1208,-1138,-1208,-1358},
04101 { -379, -848, -778, -848, -998}},
04102 /* CG.CU..GU */
04103 {{ DEF, -519, -449, -519, -669},
04104 { -649,-1118,-1048,-1118,-1268},
04105 { -289, -758, -688, -758, -908},
04106 { -739,-1208,-1138,-1208,-1358},
04107 { -379, -848, -778, -848, -998}}},
04108 /* CG.G@..GU */
04109 {{{ DEF, -939, -939, -939, -939},
04110 { -100, -989, -989, -989, -989},
04111 { -100, -989, -989, -989, -989},
04112 { -100, -989, -989, -989, -989},
04113 { -100, -989, -989, -989, -989}}},
04114 /* CG.GA..GU */
04115 {{ DEF, -939, -939, -939, -939},
04116 { -479,-1368,-1368,-1368,-1368},
04117 { -309,-1198,-1198,-1198,-1198},
04118 { -389,-1278,-1278,-1278,-1278},
04119 { -379,-1268,-1268,-1268,-1268}},
04120 /* CG.GC..GU */
04121 {{ DEF, -939, -939, -939, -939},
04122 { -649,-1538,-1538,-1538,-1538},
04123 { -289,-1178,-1178,-1178,-1178},
04124 { -739,-1628,-1628,-1628,-1628},
04125 { -379,-1268,-1268,-1268,-1268}},
04126 /* CG.GG..GU */
04127 {{ DEF, -939, -939, -939, -939},
04128 { -649,-1538,-1538,-1538,-1538},
04129 { -289,-1178,-1178,-1178,-1178},
04130 { -739,-1628,-1628,-1628,-1628},
04131 { -379,-1268,-1268,-1268,-1268}},
04132 /* CG.GU..GU */
04133 {{ DEF, -939, -939, -939, -939},
04134 { -649,-1538,-1538,-1538,-1538},
04135 { -289,-1178,-1178,-1178,-1178},
04136 { -739,-1628,-1628,-1628,-1628},
04137 { -379,-1268,-1268,-1268,-1268}}},
04138 /* CG.U@..GU */
04139 {{{ DEF, -809, -739, -809, -859},
04140 { -100, -859, -789, -859, -909},
04141 { -100, -859, -789, -859, -909},
04142 { -100, -859, -789, -859, -909},
04143 { -100, -859, -789, -859, -909}}},
04144 /* CG.UA..GU */
04145 {{ DEF, -809, -739, -809, -859},
04146 { -479,-1238,-1168,-1238,-1288},
04147 { -309,-1068, -998,-1068,-1118},
04148 { -389,-1148,-1078,-1148,-1198},
04149 { -379,-1138,-1068,-1138,-1188}},
04150 /* CG.UC..GU */
04151 {{ DEF, -809, -739, -809, -859},
04152 { -649,-1408,-1338,-1408,-1458},
04153 { -289,-1048, -978,-1048,-1098},
04154 { -739,-1498,-1428,-1498,-1548},
04155 { -379,-1138,-1068,-1138,-1188}},
04156 /* CG.UG..GU */
04157 {{ DEF, -809, -739, -809, -859},
04158 { -649,-1408,-1338,-1408,-1458},
04159 { -289,-1048, -978,-1048,-1098},
04160 { -739,-1498,-1428,-1498,-1548},
04161 { -379,-1138,-1068,-1138,-1188}},
04162 /* CG.UU..GU */
04163 {{ DEF, -809, -739, -809, -859},
04164 { -649,-1408,-1338,-1408,-1458},
04165 { -289,-1048, -978,-1048,-1098},
04166 { -739,-1498,-1428,-1498,-1548},
04167 { -379,-1138,-1068,-1138,-1188}}}},
04168 /* CG.@@..UG */
04169 {{{{ 0, 0, 0, 0, 0},
04170 { DEF, DEF, DEF, DEF, DEF},
04171 { DEF, DEF, DEF, DEF, DEF},
04172 { DEF, DEF, DEF, DEF, DEF},
04173 { DEF, DEF, DEF, DEF, DEF}}}},
04174 /* CG.@A..UG */
04175 {{ 0, 0, 0, 0, 0},
04176 { -719, -719, -719, -719, -719},
04177 { -479, -479, -479, -479, -479},
04178 { -659, -659, -659, -659, -659},
04179 { -549, -549, -549, -549, -549}},
04180 /* CG.@C..UG */
04181 {{ 0, 0, 0, 0, 0},
04182 { -789, -789, -789, -789, -789}},

```

```
04183 { -479, -479, -479, -479, -479},
04184 { -809, -809, -809, -809, -809},
04185 { -439, -439, -439, -439, -439}},
04186 /* CG.@G..UG */
04187 {{ 0, 0, 0, 0, 0},
04188 { -959, -959, -959, -959, -959},
04189 { -359, -359, -359, -359, -359},
04190 { -919, -919, -919, -919, -919},
04191 { -549, -549, -549, -549, -549}},
04192 /* CG.@U..UG */
04193 {{ 0, 0, 0, 0, 0},
04194 { -809, -809, -809, -809, -809},
04195 { -479, -479, -479, -479, -479},
04196 { -809, -809, -809, -809, -809},
04197 { -359, -359, -359, -359, -359}}},
04198 /* CG.A@..UG */
04199 {{{ DEF, -1029, -949, -1029, -1029},
04200 { -100, -1079, -999, -1079, -1079},
04201 { -100, -1079, -999, -1079, -1079},
04202 { -100, -1079, -999, -1079, -1079},
04203 { -100, -1079, -999, -1079, -1079}},
04204 /* CG.AA..UG */
04205 {{ DEF, -1029, -949, -1029, -1029},
04206 { -769, -1748, -1668, -1748, -1748},
04207 { -529, -1508, -1428, -1508, -1508},
04208 { -709, -1688, -1608, -1688, -1688},
04209 { -599, -1578, -1498, -1578, -1578}},
04210 /* CG.AC..UG */
04211 {{ DEF, -1029, -949, -1029, -1029},
04212 { -839, -1818, -1738, -1818, -1818},
04213 { -529, -1508, -1428, -1508, -1508},
04214 { -859, -1838, -1758, -1838, -1838},
04215 { -489, -1468, -1388, -1468, -1468}},
04216 /* CG.AG..UG */
04217 {{ DEF, -1029, -949, -1029, -1029},
04218 { -1009, -1988, -1908, -1988, -1988},
04219 { -409, -1388, -1308, -1388, -1388},
04220 { -969, -1948, -1868, -1948, -1948},
04221 { -599, -1578, -1498, -1578, -1578}},
04222 /* CG.AU..UG */
04223 {{ DEF, -1029, -949, -1029, -1029},
04224 { -859, -1838, -1758, -1838, -1838},
04225 { -529, -1508, -1428, -1508, -1508},
04226 { -859, -1838, -1758, -1838, -1838},
04227 { -409, -1388, -1308, -1388, -1388}}},
04228 /* CG.C@..UG */
04229 {{{ DEF, -519, -449, -519, -669},
04230 { -100, -569, -499, -569, -719},
04231 { -100, -569, -499, -569, -719},
04232 { -100, -569, -499, -569, -719},
04233 { -100, -569, -499, -569, -719}},
04234 /* CG.CA..UG */
04235 {{ DEF, -519, -449, -519, -669},
04236 { -769, -1238, -1168, -1238, -1388},
04237 { -529, -998, -928, -998, -1148},
04238 { -709, -1178, -1108, -1178, -1328},
04239 { -599, -1068, -998, -1068, -1218}},
04240 /* CG.CC..UG */
04241 {{ DEF, -519, -449, -519, -669},
04242 { -839, -1308, -1238, -1308, -1458},
04243 { -529, -998, -928, -998, -1148},
04244 { -859, -1328, -1258, -1328, -1478},
04245 { -489, -958, -888, -958, -1108}},
04246 /* CG.CG..UG */
04247 {{ DEF, -519, -449, -519, -669},
04248 { -1009, -1478, -1408, -1478, -1628},
04249 { -409, -878, -808, -878, -1028},
04250 { -969, -1438, -1368, -1438, -1588},
04251 { -599, -1068, -998, -1068, -1218}},
04252 /* CG.CU..UG */
04253 {{ DEF, -519, -449, -519, -669},
04254 { -859, -1328, -1258, -1328, -1478},
04255 { -529, -998, -928, -998, -1148},
04256 { -859, -1328, -1258, -1328, -1478},
04257 { -409, -878, -808, -878, -1028}}},
04258 /* CG.G@..UG */
04259 {{{ DEF, -939, -939, -939, -939},
04260 { -100, -989, -989, -989, -989},
04261 { -100, -989, -989, -989, -989},
04262 { -100, -989, -989, -989, -989},
04263 { -100, -989, -989, -989, -989}},
04264 /* CG.GA..UG */
04265 {{ DEF, -939, -939, -939, -939},
04266 { -769, -1658, -1658, -1658, -1658},
04267 { -529, -1418, -1418, -1418, -1418},
04268 { -709, -1598, -1598, -1598, -1598},
04269 { -599, -1488, -1488, -1488, -1488}},
```

```

04270 /* CG.GC..UG */
04271 {{ DEF, -939, -939, -939, -939},
04272 { -839,-1728,-1728,-1728,-1728},
04273 { -529,-1418,-1418,-1418,-1418},
04274 { -859,-1748,-1748,-1748,-1748},
04275 { -489,-1378,-1378,-1378,-1378}},
04276 /* CG.GG..UG */
04277 {{ DEF, -939, -939, -939, -939},
04278 {-1009,-1898,-1898,-1898,-1898},
04279 {-409,-1298,-1298,-1298,-1298},
04280 {-969,-1858,-1858,-1858,-1858},
04281 {-599,-1488,-1488,-1488,-1488}},
04282 /* CG.GU..UG */
04283 {{ DEF, -939, -939, -939, -939},
04284 { -859,-1748,-1748,-1748,-1748},
04285 { -529,-1418,-1418,-1418,-1418},
04286 { -859,-1748,-1748,-1748,-1748},
04287 { -409,-1298,-1298,-1298,-1298}}},
04288 /* CG.U@..UG */
04289 {{{ DEF, -809, -739, -809, -859},
04290 { -100, -859, -789, -859, -909},
04291 { -100, -859, -789, -859, -909},
04292 { -100, -859, -789, -859, -909},
04293 { -100, -859, -789, -859, -909}}},
04294 /* CG.UA..UG */
04295 {{ DEF, -809, -739, -809, -859},
04296 { -769,-1528,-1458,-1528,-1578},
04297 { -529,-1288,-1218,-1288,-1338},
04298 { -709,-1468,-1398,-1468,-1518},
04299 { -599,-1358,-1288,-1358,-1408}},
04300 /* CG.UC..UG */
04301 {{ DEF, -809, -739, -809, -859},
04302 { -839,-1598,-1528,-1598,-1648},
04303 { -529,-1288,-1218,-1288,-1338},
04304 { -859,-1618,-1548,-1618,-1668},
04305 { -489,-1248,-1178,-1248,-1298}},
04306 /* CG.UG..UG */
04307 {{ DEF, -809, -739, -809, -859},
04308 {-1009,-1768,-1698,-1768,-1818},
04309 {-409,-1168,-1098,-1168,-1218},
04310 {-969,-1728,-1658,-1728,-1778},
04311 {-599,-1358,-1288,-1358,-1408}},
04312 /* CG.UU..UG */
04313 {{{ DEF, -809, -739, -809, -859},
04314 { -859,-1618,-1548,-1618,-1668},
04315 { -529,-1288,-1218,-1288,-1338},
04316 { -859,-1618,-1548,-1618,-1668},
04317 { -409,-1168,-1098,-1168,-1218}}}},
04318 /* CG.@@..AU */
04319 {{{{ 0, 0, 0, 0, 0},
04320 { DEF, DEF, DEF, DEF, DEF},
04321 { DEF, DEF, DEF, DEF, DEF},
04322 { DEF, DEF, DEF, DEF, DEF},
04323 { DEF, DEF, DEF, DEF, DEF}}}},
04324 /* CG.@A..AU */
04325 {{ 0, 0, 0, 0, 0},
04326 { -429, -429, -429, -429, -429},
04327 { -259, -259, -259, -259, -259},
04328 { -339, -339, -339, -339, -339},
04329 { -329, -329, -329, -329, -329}},
04330 /* CG.@C..AU */
04331 {{ 0, 0, 0, 0, 0},
04332 { -599, -599, -599, -599, -599},
04333 { -239, -239, -239, -239, -239},
04334 { -689, -689, -689, -689, -689},
04335 { -329, -329, -329, -329, -329}},
04336 /* CG.@G..AU */
04337 {{ 0, 0, 0, 0, 0},
04338 { -599, -599, -599, -599, -599},
04339 { -239, -239, -239, -239, -239},
04340 { -689, -689, -689, -689, -689},
04341 { -329, -329, -329, -329, -329}},
04342 /* CG.@U..AU */
04343 {{{ 0, 0, 0, 0, 0},
04344 { -599, -599, -599, -599, -599},
04345 { -239, -239, -239, -239, -239},
04346 { -689, -689, -689, -689, -689},
04347 { -329, -329, -329, -329, -329}}}},
04348 /* CG.A@..AU */
04349 {{{ DEF, -1029, -949, -1029, -1029},
04350 { -100, -1079, -999, -1079, -1079},
04351 { -100, -1079, -999, -1079, -1079},
04352 { -100, -1079, -999, -1079, -1079},
04353 { -100, -1079, -999, -1079, -1079}},
04354 /* CG.AA..AU */
04355 {{ DEF, -1029, -949, -1029, -1029},
04356 { -479, -1458, -1378, -1458, -1458},

```

```
04357 { -309,-1288,-1208,-1288,-1288},
04358 { -389,-1368,-1288,-1368,-1368},
04359 { -379,-1358,-1278,-1358,-1358}},
04360 /* CG.AC..AU */
04361 {{ DEF,-1029, -949,-1029,-1029},
04362 { -649,-1628,-1548,-1628,-1628},
04363 { -289,-1268,-1188,-1268,-1268},
04364 { -739,-1718,-1638,-1718,-1718},
04365 { -379,-1358,-1278,-1358,-1358}},
04366 /* CG.AG..AU */
04367 {{ DEF,-1029, -949,-1029,-1029},
04368 { -649,-1628,-1548,-1628,-1628},
04369 { -289,-1268,-1188,-1268,-1268},
04370 { -739,-1718,-1638,-1718,-1718},
04371 { -379,-1358,-1278,-1358,-1358}},
04372 /* CG.AU..AU */
04373 {{ DEF,-1029, -949,-1029,-1029},
04374 { -649,-1628,-1548,-1628,-1628},
04375 { -289,-1268,-1188,-1268,-1268},
04376 { -739,-1718,-1638,-1718,-1718},
04377 { -379,-1358,-1278,-1358,-1358}}},
04378 /* CG.C@..AU */
04379 {{{ DEF, -519, -449, -519, -669},
04380 { -100, -569, -499, -569, -719},
04381 { -100, -569, -499, -569, -719},
04382 { -100, -569, -499, -569, -719},
04383 { -100, -569, -499, -569, -719}},
04384 /* CG.CA..AU */
04385 {{ DEF, -519, -449, -519, -669},
04386 { -479, -948, -878, -948,-1098},
04387 { -309, -778, -708, -778, -928},
04388 { -389, -858, -788, -858,-1008},
04389 { -379, -848, -778, -848, -998}},
04390 /* CG.CC..AU */
04391 {{ DEF, -519, -449, -519, -669},
04392 { -649,-1118,-1048,-1118,-1268},
04393 { -289, -758, -688, -758, -908},
04394 { -739,-1208,-1138,-1208,-1358},
04395 { -379, -848, -778, -848, -998}},
04396 /* CG.CG..AU */
04397 {{ DEF, -519, -449, -519, -669},
04398 { -649,-1118,-1048,-1118,-1268},
04399 { -289, -758, -688, -758, -908},
04400 { -739,-1208,-1138,-1208,-1358},
04401 { -379, -848, -778, -848, -998}},
04402 /* CG.CU..AU */
04403 {{ DEF, -519, -449, -519, -669},
04404 { -649,-1118,-1048,-1118,-1268},
04405 { -289, -758, -688, -758, -908},
04406 { -739,-1208,-1138,-1208,-1358},
04407 { -379, -848, -778, -848, -998}}},
04408 /* CG.G@..AU */
04409 {{{ DEF, -939, -939, -939, -939},
04410 { -100, -989, -989, -989, -989},
04411 { -100, -989, -989, -989, -989},
04412 { -100, -989, -989, -989, -989},
04413 { -100, -989, -989, -989, -989}},
04414 /* CG.GA..AU */
04415 {{ DEF, -939, -939, -939, -939},
04416 { -479,-1368,-1368,-1368,-1368},
04417 { -309,-1198,-1198,-1198,-1198},
04418 { -389,-1278,-1278,-1278,-1278},
04419 { -379,-1268,-1268,-1268,-1268}},
04420 /* CG.GC..AU */
04421 {{ DEF, -939, -939, -939, -939},
04422 { -649,-1538,-1538,-1538,-1538},
04423 { -289,-1178,-1178,-1178,-1178},
04424 { -739,-1628,-1628,-1628,-1628},
04425 { -379,-1268,-1268,-1268,-1268}},
04426 /* CG.GG..AU */
04427 {{ DEF, -939, -939, -939, -939},
04428 { -649,-1538,-1538,-1538,-1538},
04429 { -289,-1178,-1178,-1178,-1178},
04430 { -739,-1628,-1628,-1628,-1628},
04431 { -379,-1268,-1268,-1268,-1268}},
04432 /* CG.GU..AU */
04433 {{ DEF, -939, -939, -939, -939},
04434 { -649,-1538,-1538,-1538,-1538},
04435 { -289,-1178,-1178,-1178,-1178},
04436 { -739,-1628,-1628,-1628,-1628},
04437 { -379,-1268,-1268,-1268,-1268}}},
04438 /* CG.U@..AU */
04439 {{{ DEF, -809, -739, -809, -859},
04440 { -100, -859, -789, -859, -909},
04441 { -100, -859, -789, -859, -909},
04442 { -100, -859, -789, -859, -909},
04443 { -100, -859, -789, -859, -909}},
```



```

04444 /* CG.UA..AU */
04445 {{ DEF, -809, -739, -809, -859},
04446 { -479,-1238,-1168,-1238,-1288},
04447 { -309,-1068, -998,-1068,-1118},
04448 { -389,-1148,-1078,-1148,-1198},
04449 { -379,-1138,-1068,-1138,-1188}},
04450 /* CG.UC..AU */
04451 {{ DEF, -809, -739, -809, -859},
04452 { -649,-1408,-1338,-1408,-1458},
04453 { -289,-1048, -978,-1048,-1098},
04454 { -739,-1498,-1428,-1498,-1548},
04455 { -379,-1138,-1068,-1138,-1188}},
04456 /* CG.UG..AU */
04457 {{ DEF, -809, -739, -809, -859},
04458 { -649,-1408,-1338,-1408,-1458},
04459 { -289,-1048, -978,-1048,-1098},
04460 { -739,-1498,-1428,-1498,-1548},
04461 { -379,-1138,-1068,-1138,-1188}},
04462 /* CG.UU..AU */
04463 {{ DEF, -809, -739, -809, -859},
04464 { -649,-1408,-1338,-1408,-1458},
04465 { -289,-1048, -978,-1048,-1098},
04466 { -739,-1498,-1428,-1498,-1548},
04467 { -379,-1138,-1068,-1138,-1188}}}},
04468 /* CG.@@..UA */
04469 {{{ 0, 0, 0, 0, 0},
04470 { DEF, DEF, DEF, DEF, DEF},
04471 { DEF, DEF, DEF, DEF, DEF},
04472 { DEF, DEF, DEF, DEF, DEF},
04473 { DEF, DEF, DEF, DEF, DEF}}},
04474 /* CG.@A..UA */
04475 {{ 0, 0, 0, 0, 0},
04476 { -399, -399, -399, -399, -399},
04477 { -429, -429, -429, -429, -429},
04478 { -379, -379, -379, -379, -379},
04479 { -279, -279, -279, -279, -279}},
04480 /* CG.@C..UA */
04481 {{ 0, 0, 0, 0, 0},
04482 { -629, -629, -629, -629, -629},
04483 { -509, -509, -509, -509, -509},
04484 { -679, -679, -679, -679, -679},
04485 { -139, -139, -139, -139, -139}},
04486 /* CG.@G..UA */
04487 {{ 0, 0, 0, 0, 0},
04488 { -889, -889, -889, -889, -889},
04489 { -199, -199, -199, -199, -199},
04490 { -889, -889, -889, -889, -889},
04491 { -279, -279, -279, -279, -279}},
04492 /* CG.@U..UA */
04493 {{ 0, 0, 0, 0, 0},
04494 { -589, -589, -589, -589, -589},
04495 { -179, -179, -179, -179, -179},
04496 { -679, -679, -679, -679, -679},
04497 { -140, -140, -140, -140, -140}}}},
04498 /* CG.A@..UA */
04499 {{{ DEF,-1029, -949,-1029,-1029},
04500 { -100,-1079, -999,-1079,-1079},
04501 { -100,-1079, -999,-1079,-1079},
04502 { -100,-1079, -999,-1079,-1079},
04503 { -100,-1079, -999,-1079,-1079}},
04504 /* CG.AA..UA */
04505 {{ DEF,-1029, -949,-1029,-1029},
04506 { -449,-1428,-1348,-1428,-1428},
04507 { -479,-1458,-1378,-1458,-1458},
04508 { -429,-1408,-1328,-1408,-1408},
04509 { -329,-1308,-1228,-1308,-1308}},
04510 /* CG.AC..UA */
04511 {{ DEF,-1029, -949,-1029,-1029},
04512 { -679,-1658,-1578,-1658,-1658},
04513 { -559,-1538,-1458,-1538,-1538},
04514 { -729,-1708,-1628,-1708,-1708},
04515 { -189,-1168,-1088,-1168,-1168}},
04516 /* CG.AG..UA */
04517 {{ DEF,-1029, -949,-1029,-1029},
04518 { -939,-1918,-1838,-1918,-1918},
04519 { -249,-1228,-1148,-1228,-1228},
04520 { -939,-1918,-1838,-1918,-1918},
04521 { -329,-1308,-1228,-1308,-1308}},
04522 /* CG.AU..UA */
04523 {{ DEF,-1029, -949,-1029,-1029},
04524 { -639,-1618,-1538,-1618,-1618},
04525 { -229,-1208,-1128,-1208,-1208},
04526 { -729,-1708,-1628,-1708,-1708},
04527 { -190,-1169,-1089,-1169,-1169}}}},
04528 /* CG.C@..UA */
04529 {{{ DEF, -519, -449, -519, -669},
04530 { -100, -569, -499, -569, -719}},

```

```

04531 { -100, -569, -499, -569, -719},
04532 { -100, -569, -499, -569, -719},
04533 { -100, -569, -499, -569, -719}},
04534 /* CG.CA..UA */
04535 {{ DEF, -519, -449, -519, -669},
04536 { -449, -918, -848, -918, -1068},
04537 { -479, -948, -878, -948, -1098},
04538 { -429, -898, -828, -898, -1048},
04539 { -329, -798, -728, -798, -948}},
04540 /* CG.CC..UA */
04541 {{ DEF, -519, -449, -519, -669},
04542 { -679, -1148, -1078, -1148, -1298},
04543 { -559, -1028, -958, -1028, -1178},
04544 { -729, -1198, -1128, -1198, -1348},
04545 { -189, -658, -588, -658, -808}},
04546 /* CG.CG..UA */
04547 {{ DEF, -519, -449, -519, -669},
04548 { -939, -1408, -1338, -1408, -1558},
04549 { -249, -718, -648, -718, -868},
04550 { -939, -1408, -1338, -1408, -1558},
04551 { -329, -798, -728, -798, -948}},
04552 /* CG.CU..UA */
04553 {{ DEF, -519, -449, -519, -669},
04554 { -639, -1108, -1038, -1108, -1258},
04555 { -229, -698, -628, -698, -848},
04556 { -729, -1198, -1128, -1198, -1348},
04557 { -190, -659, -589, -659, -809}}},
04558 /* CG.G@..UA */
04559 {{{ DEF, -939, -939, -939, -939},
04560 { -100, -989, -989, -989, -989},
04561 { -100, -989, -989, -989, -989},
04562 { -100, -989, -989, -989, -989},
04563 { -100, -989, -989, -989, -989}},
04564 /* CG.GA..UA */
04565 {{ DEF, -939, -939, -939, -939},
04566 { -449, -1338, -1338, -1338, -1338},
04567 { -479, -1368, -1368, -1368, -1368},
04568 { -429, -1318, -1318, -1318, -1318},
04569 { -329, -1218, -1218, -1218, -1218}},
04570 /* CG.GC..UA */
04571 {{ DEF, -939, -939, -939, -939},
04572 { -679, -1568, -1568, -1568, -1568},
04573 { -559, -1448, -1448, -1448, -1448},
04574 { -729, -1618, -1618, -1618, -1618},
04575 { -189, -1078, -1078, -1078, -1078}},
04576 /* CG.GG..UA */
04577 {{ DEF, -939, -939, -939, -939},
04578 { -939, -1828, -1828, -1828, -1828},
04579 { -249, -1138, -1138, -1138, -1138},
04580 { -939, -1828, -1828, -1828, -1828},
04581 { -329, -1218, -1218, -1218, -1218}},
04582 /* CG.GU..UA */
04583 {{ DEF, -939, -939, -939, -939},
04584 { -639, -1528, -1528, -1528, -1528},
04585 { -229, -1118, -1118, -1118, -1118},
04586 { -729, -1618, -1618, -1618, -1618},
04587 { -190, -1079, -1079, -1079, -1079}}},
04588 /* CG.U@..UA */
04589 {{{ DEF, -809, -739, -809, -859},
04590 { -100, -859, -789, -859, -909},
04591 { -100, -859, -789, -859, -909},
04592 { -100, -859, -789, -859, -909},
04593 { -100, -859, -789, -859, -909}},
04594 /* CG.UA..UA */
04595 {{ DEF, -809, -739, -809, -859},
04596 { -449, -1208, -1138, -1208, -1258},
04597 { -479, -1238, -1168, -1238, -1288},
04598 { -429, -1188, -1118, -1188, -1238},
04599 { -329, -1088, -1018, -1088, -1138}},
04600 /* CG.UC..UA */
04601 {{ DEF, -809, -739, -809, -859},
04602 { -679, -1438, -1368, -1438, -1488},
04603 { -559, -1318, -1248, -1318, -1368},
04604 { -729, -1488, -1418, -1488, -1538},
04605 { -189, -948, -878, -948, -998}},
04606 /* CG.UG..UA */
04607 {{ DEF, -809, -739, -809, -859},
04608 { -939, -1698, -1628, -1698, -1748},
04609 { -249, -1008, -938, -1008, -1058},
04610 { -939, -1698, -1628, -1698, -1748},
04611 { -329, -1088, -1018, -1088, -1138}},
04612 /* CG.UU..UA */
04613 {{{ DEF, -809, -739, -809, -859},
04614 { -639, -1398, -1328, -1398, -1448},
04615 { -229, -988, -918, -988, -1038},
04616 { -729, -1488, -1418, -1488, -1538},
04617 { -190, -949, -879, -949, -999}}}},

```

```
04618 /* CG.@@.. @ */
04619 {{{ DEF, DEF, DEF, DEF, DEF},
04620 { DEF, DEF, DEF, DEF, DEF},
04621 { DEF, DEF, DEF, DEF, DEF},
04622 { DEF, DEF, DEF, DEF, DEF},
04623 { DEF, DEF, DEF, DEF, DEF}},
04624 /* CG.@A.. @ */
04625 {{ DEF, DEF, DEF, DEF, DEF},
04626 { DEF, DEF, DEF, DEF, DEF},
04627 { DEF, DEF, DEF, DEF, DEF},
04628 { DEF, DEF, DEF, DEF, DEF},
04629 { DEF, DEF, DEF, DEF, DEF}},
04630 /* CG.@C.. @ */
04631 {{ DEF, DEF, DEF, DEF, DEF},
04632 { DEF, DEF, DEF, DEF, DEF},
04633 { DEF, DEF, DEF, DEF, DEF},
04634 { DEF, DEF, DEF, DEF, DEF},
04635 { DEF, DEF, DEF, DEF, DEF}},
04636 /* CG.@G.. @ */
04637 {{ DEF, DEF, DEF, DEF, DEF},
04638 { DEF, DEF, DEF, DEF, DEF},
04639 { DEF, DEF, DEF, DEF, DEF},
04640 { DEF, DEF, DEF, DEF, DEF},
04641 { DEF, DEF, DEF, DEF, DEF}},
04642 /* CG.@U.. @ */
04643 {{ DEF, DEF, DEF, DEF, DEF},
04644 { DEF, DEF, DEF, DEF, DEF},
04645 { DEF, DEF, DEF, DEF, DEF},
04646 { DEF, DEF, DEF, DEF, DEF},
04647 { DEF, DEF, DEF, DEF, DEF}},
04648 /* CG.@@.. @ */
04649 {{{ -100,-1079, -999,-1079,-1079},
04650 { -100,-1079, -999,-1079,-1079},
04651 { -100,-1079, -999,-1079,-1079},
04652 { -100,-1079, -999,-1079,-1079},
04653 { -100,-1079, -999,-1079,-1079}},
04654 /* CG.@A.. @ */
04655 {{ -100,-1079, -999,-1079,-1079},
04656 { -100,-1079, -999,-1079,-1079},
04657 { -100,-1079, -999,-1079,-1079},
04658 { -100,-1079, -999,-1079,-1079},
04659 { -100,-1079, -999,-1079,-1079}},
04660 /* CG.@C.. @ */
04661 {{{ -100,-1079, -999,-1079,-1079},
04662 { -100,-1079, -999,-1079,-1079},
04663 { -100,-1079, -999,-1079,-1079},
04664 { -100,-1079, -999,-1079,-1079},
04665 { -100,-1079, -999,-1079,-1079}},
04666 /* CG.@G.. @ */
04667 {{{ -100,-1079, -999,-1079,-1079},
04668 { -100,-1079, -999,-1079,-1079},
04669 { -100,-1079, -999,-1079,-1079},
04670 { -100,-1079, -999,-1079,-1079},
04671 { -100,-1079, -999,-1079,-1079}},
04672 /* CG.@U.. @ */
04673 {{{ -100,-1079, -999,-1079,-1079},
04674 { -100,-1079, -999,-1079,-1079},
04675 { -100,-1079, -999,-1079,-1079},
04676 { -100,-1079, -999,-1079,-1079},
04677 { -100,-1079, -999,-1079,-1079}},
04678 /* CG.@@.. @ */
04679 {{{ -100, -569, -499, -569, -719},
04680 { -100, -569, -499, -569, -719},
04681 { -100, -569, -499, -569, -719},
04682 { -100, -569, -499, -569, -719},
04683 { -100, -569, -499, -569, -719}},
04684 /* CG.@A.. @ */
04685 {{ -100, -569, -499, -569, -719},
04686 { -100, -569, -499, -569, -719},
04687 { -100, -569, -499, -569, -719},
04688 { -100, -569, -499, -569, -719},
04689 { -100, -569, -499, -569, -719}},
04690 /* CG.@C.. @ */
04691 {{{ -100, -569, -499, -569, -719},
04692 { -100, -569, -499, -569, -719},
04693 { -100, -569, -499, -569, -719},
04694 { -100, -569, -499, -569, -719},
04695 { -100, -569, -499, -569, -719}},
04696 /* CG.@G.. @ */
04697 {{{ -100, -569, -499, -569, -719},
04698 { -100, -569, -499, -569, -719},
04699 { -100, -569, -499, -569, -719},
04700 { -100, -569, -499, -569, -719},
04701 { -100, -569, -499, -569, -719}},
04702 /* CG.@U.. @ */
04703 {{{ -100, -569, -499, -569, -719},
04704 { -100, -569, -499, -569, -719},
```

```

04705 { -100, -569, -499, -569, -719},
04706 { -100, -569, -499, -569, -719},
04707 { -100, -569, -499, -569, -719}},
04708 /* CG.G@.. @ */
04709 {{ { -100, -989, -989, -989, -989},
04710 { -100, -989, -989, -989, -989},
04711 { -100, -989, -989, -989, -989},
04712 { -100, -989, -989, -989, -989},
04713 { -100, -989, -989, -989, -989}},
04714 /* CG.GA.. @ */
04715 {{ { -100, -989, -989, -989, -989},
04716 { -100, -989, -989, -989, -989},
04717 { -100, -989, -989, -989, -989},
04718 { -100, -989, -989, -989, -989},
04719 { -100, -989, -989, -989, -989}},
04720 /* CG.GC.. @ */
04721 {{ { -100, -989, -989, -989, -989},
04722 { -100, -989, -989, -989, -989},
04723 { -100, -989, -989, -989, -989},
04724 { -100, -989, -989, -989, -989},
04725 { -100, -989, -989, -989, -989}},
04726 /* CG.GG.. @ */
04727 {{ { -100, -989, -989, -989, -989},
04728 { -100, -989, -989, -989, -989},
04729 { -100, -989, -989, -989, -989},
04730 { -100, -989, -989, -989, -989},
04731 { -100, -989, -989, -989, -989}},
04732 /* CG.GU.. @ */
04733 {{ { -100, -989, -989, -989, -989},
04734 { -100, -989, -989, -989, -989},
04735 { -100, -989, -989, -989, -989},
04736 { -100, -989, -989, -989, -989},
04737 { -100, -989, -989, -989, -989}}},
04738 /* CG.U@.. @ */
04739 {{{ { -100, -859, -789, -859, -909},
04740 { -100, -859, -789, -859, -909},
04741 { -100, -859, -789, -859, -909},
04742 { -100, -859, -789, -859, -909},
04743 { -100, -859, -789, -859, -909}},
04744 /* CG.UA.. @ */
04745 {{{ { -100, -859, -789, -859, -909},
04746 { -100, -859, -789, -859, -909},
04747 { -100, -859, -789, -859, -909},
04748 { -100, -859, -789, -859, -909},
04749 { -100, -859, -789, -859, -909}},
04750 /* CG.UC.. @ */
04751 {{{ { -100, -859, -789, -859, -909},
04752 { -100, -859, -789, -859, -909},
04753 { -100, -859, -789, -859, -909},
04754 { -100, -859, -789, -859, -909},
04755 { -100, -859, -789, -859, -909}},
04756 /* CG.UG.. @ */
04757 {{{ { -100, -859, -789, -859, -909},
04758 { -100, -859, -789, -859, -909},
04759 { -100, -859, -789, -859, -909},
04760 { -100, -859, -789, -859, -909},
04761 { -100, -859, -789, -859, -909}},
04762 /* CG.UU.. @ */
04763 {{{ { -100, -859, -789, -859, -909},
04764 { -100, -859, -789, -859, -909},
04765 { -100, -859, -789, -859, -909},
04766 { -100, -859, -789, -859, -909},
04767 { -100, -859, -789, -859, -909}}}}},
04768 { /* noPair */ {{{{0}}}},
04769 /* GC.@@..CG */
04770 {{{ { 0, 0, 0, 0, 0},
04771 { DEF, DEF, DEF, DEF, DEF},
04772 { DEF, DEF, DEF, DEF, DEF},
04773 { DEF, DEF, DEF, DEF, DEF},
04774 { DEF, DEF, DEF, DEF, DEF}},
04775 /* GC.@A..CG */
04776 {{{ { 0, 0, 0, 0, 0},
04777 {-1029,-1029,-1029,-1029,-1029},
04778 { -519, -519, -519, -519, -519},
04779 { -939, -939, -939, -939, -939},
04780 { -809, -809, -809, -809, -809}},
04781 /* GC.@C..CG */
04782 {{{ { 0, 0, 0, 0, 0},
04783 { -949, -949, -949, -949, -949},
04784 { -449, -449, -449, -449, -449},
04785 { -939, -939, -939, -939, -939},
04786 { -739, -739, -739, -739, -739}},
04787 /* GC.@G..CG */
04788 {{{ { 0, 0, 0, 0, 0},
04789 {-1029,-1029,-1029,-1029,-1029},
04790 { -519, -519, -519, -519, -519},
04791 { -939, -939, -939, -939, -939}},

```

```
04792 { -809, -809, -809, -809, -809}},
04793 /* GC.@U..CG */
04794 {{ 0, 0, 0, 0, 0},
04795 {-1029,-1029,-1029,-1029,-1029},
04796 { -669, -669, -669, -669, -669},
04797 { -939, -939, -939, -939, -939},
04798 { -859, -859, -859, -859, -859}}},
04799 /* GC.A@..CG */
04800 {{{ DEF, -519, -879, -559, -879},
04801 { -100, -569, -929, -609, -929},
04802 { -100, -569, -929, -609, -929},
04803 { -100, -569, -929, -609, -929},
04804 { -100, -569, -929, -609, -929}},
04805 /* GC.AA..CG */
04806 {{ DEF, -519, -879, -559, -879},
04807 {-1079,-1548,-1908,-1588,-1908},
04808 { -569,-1038,-1398,-1078,-1398},
04809 { -989,-1458,-1818,-1498,-1818},
04810 { -859,-1328,-1688,-1368,-1688}},
04811 /* GC.AC..CG */
04812 {{ DEF, -519, -879, -559, -879},
04813 { -999,-1468,-1828,-1508,-1828},
04814 { -499, -968,-1328,-1008,-1328},
04815 { -989,-1458,-1818,-1498,-1818},
04816 { -789,-1258,-1618,-1298,-1618}},
04817 /* GC.AG..CG */
04818 {{ DEF, -519, -879, -559, -879},
04819 {-1079,-1548,-1908,-1588,-1908},
04820 { -569,-1038,-1398,-1078,-1398},
04821 { -989,-1458,-1818,-1498,-1818},
04822 { -859,-1328,-1688,-1368,-1688}},
04823 /* GC.AU..CG */
04824 {{ DEF, -519, -879, -559, -879},
04825 {-1079,-1548,-1908,-1588,-1908},
04826 { -719,-1188,-1548,-1228,-1548},
04827 { -989,-1458,-1818,-1498,-1818},
04828 { -909,-1378,-1738,-1418,-1738}}},
04829 /* GC.C@..CG */
04830 {{{ DEF, -719, -309, -309, -389},
04831 { -100, -769, -359, -359, -439},
04832 { -100, -769, -359, -359, -439},
04833 { -100, -769, -359, -359, -439},
04834 { -100, -769, -359, -359, -439}},
04835 /* GC.CA..CG */
04836 {{ DEF, -719, -309, -309, -389},
04837 {-1079,-1748,-1338,-1338,-1418},
04838 { -569,-1238, -828, -828, -908},
04839 { -989,-1658,-1248,-1248,-1328},
04840 { -859,-1528,-1118,-1118,-1198}},
04841 /* GC.CC..CG */
04842 {{ DEF, -719, -309, -309, -389},
04843 { -999,-1668,-1258,-1258,-1338},
04844 { -499,-1168, -758, -758, -838},
04845 { -989,-1658,-1248,-1248,-1328},
04846 { -789,-1458,-1048,-1048,-1128}},
04847 /* GC.CG..CG */
04848 {{ DEF, -719, -309, -309, -389},
04849 {-1079,-1748,-1338,-1338,-1418},
04850 { -569,-1238, -828, -828, -908},
04851 { -989,-1658,-1248,-1248,-1328},
04852 { -859,-1528,-1118,-1118,-1198}},
04853 /* GC.CU..CG */
04854 {{ DEF, -719, -309, -309, -389},
04855 {-1079,-1748,-1338,-1338,-1418},
04856 { -719,-1388, -978, -978,-1058},
04857 { -989,-1658,-1248,-1248,-1328},
04858 { -909,-1578,-1168,-1168,-1248}}},
04859 /* GC.G@..CG */
04860 {{{ DEF, -709, -739, -619, -739},
04861 { -100, -759, -789, -669, -789},
04862 { -100, -759, -789, -669, -789},
04863 { -100, -759, -789, -669, -789},
04864 { -100, -759, -789, -669, -789}},
04865 /* GC.GA..CG */
04866 {{ DEF, -709, -739, -619, -739},
04867 {-1079,-1738,-1768,-1648,-1768},
04868 { -569,-1228,-1258,-1138,-1258},
04869 { -989,-1648,-1678,-1558,-1678},
04870 { -859,-1518,-1548,-1428,-1548}},
04871 /* GC.GC..CG */
04872 {{ DEF, -709, -739, -619, -739},
04873 { -999,-1658,-1688,-1568,-1688},
04874 { -499,-1158,-1188,-1068,-1188},
04875 { -989,-1648,-1678,-1558,-1678},
04876 { -789,-1448,-1478,-1358,-1478}},
04877 /* GC.GG..CG */
04878 {{ DEF, -709, -739, -619, -739},
```

```

04879 {-1079,-1738,-1768,-1648,-1768},
04880 {-569,-1228,-1258,-1138,-1258},
04881 {-989,-1648,-1678,-1558,-1678},
04882 {-859,-1518,-1548,-1428,-1548}},
04883 /* GC.GU..CG */
04884 {{ DEF, -709, -739, -619, -739},
04885 {-1079,-1738,-1768,-1648,-1768},
04886 {-719,-1378,-1408,-1288,-1408},
04887 {-989,-1648,-1678,-1558,-3080},
04888 {-909,-1568,-1598,-1478,-1598}}},
04889 /* GC.U@..CG */
04890 {{{ DEF, -499, -499, -499, -569},
04891 {-100, -549, -549, -549, -619},
04892 {-100, -549, -549, -549, -619},
04893 {-100, -549, -549, -549, -619},
04894 {-100, -549, -549, -549, -619}}},
04895 /* GC.UA..CG */
04896 {{ DEF, -499, -499, -499, -569},
04897 {-1079,-1528,-1528,-1528,-1598},
04898 {-569,-1018,-1018,-1018,-1088},
04899 {-989,-1438,-1438,-1438,-1508},
04900 {-859,-1308,-1308,-1308,-1378}},
04901 /* GC.UC..CG */
04902 {{ DEF, -499, -499, -499, -569},
04903 {-999,-1448,-1448,-1448,-1518},
04904 {-499, -948, -948, -948,-1018},
04905 {-989,-1438,-1438,-1438,-1508},
04906 {-789,-1238,-1238,-1238,-1308}},
04907 /* GC.UG..CG */
04908 {{ DEF, -499, -499, -499, -569},
04909 {-1079,-1528,-1528,-1528,-1598},
04910 {-569,-1018,-1018,-1018,-1088},
04911 {-989,-1438,-1438,-1438,-1508},
04912 {-859,-1308,-1308,-1308,-1378}},
04913 /* GC.UU..CG */
04914 {{ DEF, -499, -499, -499, -569},
04915 {-1079,-1528,-1528,-1528,-1598},
04916 {-719,-1168,-1168,-1168,-1238},
04917 {-989,-1438,-1438,-1438,-1508},
04918 {-909,-1358,-1358,-1358,-1428}}}},
04919 /* GC.@@..GC */
04920 {{{{ 0, 0, 0, 0, 0},
04921 { DEF, DEF, DEF, DEF, DEF},
04922 { DEF, DEF, DEF, DEF, DEF},
04923 { DEF, DEF, DEF, DEF, DEF},
04924 { DEF, DEF, DEF, DEF, DEF}}}},
04925 /* GC.@A..GC */
04926 {{ 0, 0, 0, 0, 0},
04927 {-519, -519, -519, -519, -519},
04928 {-719, -719, -719, -719, -719},
04929 {-709, -709, -709, -709, -709},
04930 {-499, -499, -499, -499, -499}},
04931 /* GC.@C..GC */
04932 {{ 0, 0, 0, 0, 0},
04933 {-879, -879, -879, -879, -879},
04934 {-309, -309, -309, -309, -309},
04935 {-739, -739, -739, -739, -739},
04936 {-499, -499, -499, -499, -499}},
04937 /* GC.@G..GC */
04938 {{ 0, 0, 0, 0, 0},
04939 {-559, -559, -559, -559, -559},
04940 {-309, -309, -309, -309, -309},
04941 {-619, -619, -619, -619, -619},
04942 {-499, -499, -499, -499, -499}},
04943 /* GC.@U..GC */
04944 {{ 0, 0, 0, 0, 0},
04945 {-879, -879, -879, -879, -879},
04946 {-389, -389, -389, -389, -389},
04947 {-739, -739, -739, -739, -739},
04948 {-569, -569, -569, -569, -569}}}},
04949 /* GC.A@..GC */
04950 {{{ DEF, -519, -879, -559, -879},
04951 {-100, -569, -929, -609, -929},
04952 {-100, -569, -929, -609, -929},
04953 {-100, -569, -929, -609, -929},
04954 {-100, -569, -929, -609, -929}}},
04955 /* GC.AA..GC */
04956 {{ DEF, -519, -879, -559, -879},
04957 {-569,-1038,-1398,-1078,-1398},
04958 {-769,-1238,-1598,-1278,-1598},
04959 {-759,-1228,-1588,-1268,-1588},
04960 {-549,-1018,-1378,-1058,-1378}},
04961 /* GC.AC..GC */
04962 {{ DEF, -519, -879, -559, -879},
04963 {-929,-1398,-1758,-1438,-1758},
04964 {-359, -828,-1188, -868,-1188},
04965 {-789,-1258,-1618,-1298,-1618},

```

```
04966 { -549,-1018,-1378,-1058,-1378}},
04967 /* GC.AG..GC */
04968 {{ DEF, -519, -879, -559, -879},
04969 { -609,-1078,-1438,-1118,-1438},
04970 { -359, -828,-1188, -868,-1188},
04971 { -669,-1138,-1498,-1178,-1498},
04972 { -549,-1018,-1378,-1058,-1378}},
04973 /* GC.AU..GC */
04974 {{ DEF, -519, -879, -559, -879},
04975 { -929,-1398,-1758,-1438,-1758},
04976 { -439, -908,-1268, -948,-1268},
04977 { -789,-1258,-1618,-1298,-1618},
04978 { -619,-1088,-1448,-1128,-1448}}},
04979 /* GC.C@..GC */
04980 {{{ DEF, -719, -309, -309, -389},
04981 { -100, -769, -359, -359, -439},
04982 { -100, -769, -359, -359, -439},
04983 { -100, -769, -359, -359, -439},
04984 { -100, -769, -359, -359, -439}},
04985 /* GC.CA..GC */
04986 {{ DEF, -719, -309, -309, -389},
04987 { -569,-1238, -828, -828, -908},
04988 { -769,-1438,-1028,-1028,-1108},
04989 { -759,-1428,-1018,-1018,-1098},
04990 { -549,-1218, -808, -808, -888}},
04991 /* GC.CC..GC */
04992 {{ DEF, -719, -309, -309, -389},
04993 { -929,-1598,-1188,-1188,-1268},
04994 { -359,-1028, -618, -618, -698},
04995 { -789,-1458,-1048,-1048,-1128},
04996 { -549,-1218, -808, -808, -888}},
04997 /* GC.CG..GC */
04998 {{ DEF, -719, -309, -309, -389},
04999 { -609,-1278, -868, -868, -948},
05000 { -359,-1028, -618, -618, -698},
05001 { -669,-1338, -928, -928,-1008},
05002 { -549,-1218, -808, -808, -888}},
05003 /* GC.CU..GC */
05004 {{ DEF, -719, -309, -309, -389},
05005 { -929,-1598,-1188,-1188,-1268},
05006 { -439,-1108, -698, -698, -778},
05007 { -789,-1458,-1048,-1048,-1128},
05008 { -619,-1288, -878, -878, -958}}},
05009 /* GC.G@..GC */
05010 {{{ DEF, -709, -739, -619, -739},
05011 { -100, -759, -789, -669, -789},
05012 { -100, -759, -789, -669, -789},
05013 { -100, -759, -789, -669, -789},
05014 { -100, -759, -789, -669, -789}},
05015 /* GC.GA..GC */
05016 {{ DEF, -709, -739, -619, -739},
05017 { -569,-1228,-1258,-1138,-1258},
05018 { -769,-1428,-1458,-1338,-1458},
05019 { -759,-1418,-1448,-1328,-1448},
05020 { -549,-1208,-1238,-1118,-1238}},
05021 /* GC.GC..GC */
05022 {{ DEF, -709, -739, -619, -739},
05023 { -929,-1588,-1618,-1498,-1618},
05024 { -359,-1018,-1048, -928,-1048},
05025 { -789,-1448,-1478,-1358,-1478},
05026 { -549,-1208,-1238,-1118,-1238}},
05027 /* GC.GG..GC */
05028 {{ DEF, -709, -739, -619, -739},
05029 { -609,-1268,-1298,-1178,-1298},
05030 { -359,-1018,-1048, -928,-1048},
05031 { -669,-1328,-1358,-1238,-1358},
05032 { -549,-1208,-1238,-1118,-1238}},
05033 /* GC.GU..GC */
05034 {{ DEF, -709, -739, -619, -739},
05035 { -929,-1588,-1618,-1498,-1618},
05036 { -439,-1098,-1128,-1008,-1128},
05037 { -789,-1448,-1478,-1358,-3080},
05038 { -619,-1278,-1308,-1188,-1308}}},
05039 /* GC.U@..GC */
05040 {{{ DEF, -499, -499, -499, -569},
05041 { -100, -549, -549, -549, -619},
05042 { -100, -549, -549, -549, -619},
05043 { -100, -549, -549, -549, -619},
05044 { -100, -549, -549, -549, -619}},
05045 /* GC.UA..GC */
05046 {{ DEF, -499, -499, -499, -569},
05047 { -569,-1018,-1018,-1018,-1088},
05048 { -769,-1218,-1218,-1218,-1288},
05049 { -759,-1208,-1208,-1208,-1278},
05050 { -549, -998, -998, -998,-1068}},
05051 /* GC.UC..GC */
05052 {{ DEF, -499, -499, -499, -569},
```

```

05053 { -929,-1378,-1378,-1378,-1448},
05054 { -359, -808, -808, -808, -878},
05055 { -789,-1238,-1238,-1238,-1308},
05056 { -549, -998, -998, -998,-1068}},
05057 /* GC.UG..GC */
05058 {{ DEF, -499, -499, -499, -569},
05059 { -609,-1058,-1058,-1058,-1128},
05060 { -359, -808, -808, -808, -878},
05061 { -669,-1118,-1118,-1118,-1188},
05062 { -549, -998, -998, -998,-1068}},
05063 /* GC.UU..GC */
05064 {{ DEF, -499, -499, -499, -569},
05065 { -929,-1378,-1378,-1378,-1448},
05066 { -439, -888, -888, -888, -958},
05067 { -789,-1238,-1238,-1238,-1308},
05068 { -619,-1068,-1068,-1068,-1138}}}},
05069 /* GC.@@..GU */
05070 {{{ 0, 0, 0, 0, 0},
05071 { DEF, DEF, DEF, DEF, DEF},
05072 { DEF, DEF, DEF, DEF, DEF},
05073 { DEF, DEF, DEF, DEF, DEF},
05074 { DEF, DEF, DEF, DEF, DEF}},
05075 /* GC.@A..GU */
05076 {{ 0, 0, 0, 0, 0},
05077 { -429, -429, -429, -429, -429},
05078 { -259, -259, -259, -259, -259},
05079 { -339, -339, -339, -339, -339},
05080 { -329, -329, -329, -329, -329}},
05081 /* GC.@C..GU */
05082 {{ 0, 0, 0, 0, 0},
05083 { -599, -599, -599, -599, -599},
05084 { -239, -239, -239, -239, -239},
05085 { -689, -689, -689, -689, -689},
05086 { -329, -329, -329, -329, -329}},
05087 /* GC.@G..GU */
05088 {{ 0, 0, 0, 0, 0},
05089 { -599, -599, -599, -599, -599},
05090 { -239, -239, -239, -239, -239},
05091 { -689, -689, -689, -689, -689},
05092 { -329, -329, -329, -329, -329}},
05093 /* GC.@U..GU */
05094 {{ 0, 0, 0, 0, 0},
05095 { -599, -599, -599, -599, -599},
05096 { -239, -239, -239, -239, -239},
05097 { -689, -689, -689, -689, -689},
05098 { -329, -329, -329, -329, -329}},
05099 /* GC.A@..GU */
05100 {{{ DEF, -519, -879, -559, -879},
05101 { -100, -569, -929, -609, -929},
05102 { -100, -569, -929, -609, -929},
05103 { -100, -569, -929, -609, -929},
05104 { -100, -569, -929, -609, -929}},
05105 /* GC.AA..GU */
05106 {{ DEF, -519, -879, -559, -879},
05107 { -479, -948,-1308, -988,-1308},
05108 { -309, -778,-1138, -818,-1138},
05109 { -389, -858,-1218, -898,-1218},
05110 { -379, -848,-1208, -888,-1208}},
05111 /* GC.AC..GU */
05112 {{ DEF, -519, -879, -559, -879},
05113 { -649,-1118,-1478,-1158,-1478},
05114 { -289, -758,-1118, -798,-1118},
05115 { -739,-1208,-1568,-1248,-1568},
05116 { -379, -848,-1208, -888,-1208}},
05117 /* GC.AG..GU */
05118 {{ DEF, -519, -879, -559, -879},
05119 { -649,-1118,-1478,-1158,-1478},
05120 { -289, -758,-1118, -798,-1118},
05121 { -739,-1208,-1568,-1248,-1568},
05122 { -379, -848,-1208, -888,-1208}},
05123 /* GC.AU..GU */
05124 {{ DEF, -519, -879, -559, -879},
05125 { -649,-1118,-1478,-1158,-1478},
05126 { -289, -758,-1118, -798,-1118},
05127 { -739,-1208,-1568,-1248,-1568},
05128 { -379, -848,-1208, -888,-1208}}}},
05129 /* GC.C@..GU */
05130 {{{ DEF, -719, -309, -309, -389},
05131 { -100, -769, -359, -359, -439},
05132 { -100, -769, -359, -359, -439},
05133 { -100, -769, -359, -359, -439},
05134 { -100, -769, -359, -359, -439}},
05135 /* GC.CA..GU */
05136 {{ DEF, -719, -309, -309, -389},
05137 { -479,-1148, -738, -738, -818},
05138 { -309, -978, -568, -568, -648},
05139 { -389,-1058, -648, -648, -728},

```



```

05140 { -379,-1048, -638, -638, -718}},
05141 /* GC.CC..GU */
05142 {{ DEF, -719, -309, -309, -389},
05143 { -649,-1318, -908, -908, -988},
05144 { -289, -958, -548, -548, -628},
05145 { -739,-1408, -998, -998,-1078},
05146 { -379,-1048, -638, -638, -718}},
05147 /* GC.CG..GU */
05148 {{ DEF, -719, -309, -309, -389},
05149 { -649,-1318, -908, -908, -988},
05150 { -289, -958, -548, -548, -628},
05151 { -739,-1408, -998, -998,-1078},
05152 { -379,-1048, -638, -638, -718}},
05153 /* GC.CU..GU */
05154 {{ DEF, -719, -309, -309, -389},
05155 { -649,-1318, -908, -908, -988},
05156 { -289, -958, -548, -548, -628},
05157 { -739,-1408, -998, -998,-1078},
05158 { -379,-1048, -638, -638, -718}}},
05159 /* GC.G@..GU */
05160 {{{ DEF, -709, -739, -619, -739},
05161 { -100, -759, -789, -669, -789},
05162 { -100, -759, -789, -669, -789},
05163 { -100, -759, -789, -669, -789},
05164 { -100, -759, -789, -669, -789}},
05165 /* GC.GA..GU */
05166 {{ DEF, -709, -739, -619, -739},
05167 { -479,-1138,-1168,-1048,-1168},
05168 { -309, -968, -998, -878, -998},
05169 { -389,-1048,-1078, -958,-1078},
05170 { -379,-1038,-1068, -948,-1068}},
05171 /* GC.GC..GU */
05172 {{ DEF, -709, -739, -619, -739},
05173 { -649,-1308,-1338,-1218,-1338},
05174 { -289, -948, -978, -858, -978},
05175 { -739,-1398,-1428,-1308,-1428},
05176 { -379,-1038,-1068, -948,-1068}},
05177 /* GC.GG..GU */
05178 {{ DEF, -709, -739, -619, -739},
05179 { -649,-1308,-1338,-1218,-1338},
05180 { -289, -948, -978, -858, -978},
05181 { -739,-1398,-1428,-1308,-1428},
05182 { -379,-1038,-1068, -948,-1068}},
05183 /* GC.GU..GU */
05184 {{ DEF, -709, -739, -619, -739},
05185 { -649,-1308,-1338,-1218,-1338},
05186 { -289, -948, -978, -858, -978},
05187 { -739,-1398,-1428,-1308,-1428},
05188 { -379,-1038,-1068, -948,-1068}}},
05189 /* GC.U@..GU */
05190 {{{ DEF, -499, -499, -499, -569},
05191 { -100, -549, -549, -549, -619},
05192 { -100, -549, -549, -549, -619},
05193 { -100, -549, -549, -549, -619},
05194 { -100, -549, -549, -549, -619}},
05195 /* GC.UA..GU */
05196 {{ DEF, -499, -499, -499, -569},
05197 { -479, -928, -928, -928, -998},
05198 { -309, -758, -758, -758, -828},
05199 { -389, -838, -838, -838, -908},
05200 { -379, -828, -828, -828, -898}},
05201 /* GC.UC..GU */
05202 {{ DEF, -499, -499, -499, -569},
05203 { -649,-1098,-1098,-1098,-1168},
05204 { -289, -738, -738, -738, -808},
05205 { -739,-1188,-1188,-1188,-1258},
05206 { -379, -828, -828, -828, -898}},
05207 /* GC.UG..GU */
05208 {{ DEF, -499, -499, -499, -569},
05209 { -649,-1098,-1098,-1098,-1168},
05210 { -289, -738, -738, -738, -808},
05211 { -739,-1188,-1188,-1188,-1258},
05212 { -379, -828, -828, -828, -898}},
05213 /* GC.UU..GU */
05214 {{ DEF, -499, -499, -499, -569},
05215 { -649,-1098,-1098,-1098,-1168},
05216 { -289, -738, -738, -738, -808},
05217 { -739,-1188,-1188,-1188,-1258},
05218 { -379, -828, -828, -828, -898}}}},
05219 /* GC.@@..UG */
05220 {{{{ 0, 0, 0, 0, 0},
05221 { DEF, DEF, DEF, DEF, DEF},
05222 { DEF, DEF, DEF, DEF, DEF},
05223 { DEF, DEF, DEF, DEF, DEF},
05224 { DEF, DEF, DEF, DEF, DEF}},
05225 /* GC.@A..UG */
05226 {{ 0, 0, 0, 0, 0},

```

```

05227 { -719, -719, -719, -719, -719},
05228 { -479, -479, -479, -479, -479},
05229 { -659, -659, -659, -659, -659},
05230 { -549, -549, -549, -549, -549}},
05231 /* GC.@C..UG */
05232 {{ 0, 0, 0, 0, 0},
05233 { -789, -789, -789, -789, -789},
05234 { -479, -479, -479, -479, -479},
05235 { -809, -809, -809, -809, -809},
05236 { -439, -439, -439, -439, -439}},
05237 /* GC.@G..UG */
05238 {{ 0, 0, 0, 0, 0},
05239 { -959, -959, -959, -959, -959},
05240 { -359, -359, -359, -359, -359},
05241 { -919, -919, -919, -919, -919},
05242 { -549, -549, -549, -549, -549}},
05243 /* GC.@U..UG */
05244 {{ 0, 0, 0, 0, 0},
05245 { -809, -809, -809, -809, -809},
05246 { -479, -479, -479, -479, -479},
05247 { -809, -809, -809, -809, -809},
05248 { -359, -359, -359, -359, -359}}},
05249 /* GC.A@..UG */
05250 {{{ DEF, -519, -879, -559, -879},
05251 { -100, -569, -929, -609, -929},
05252 { -100, -569, -929, -609, -929},
05253 { -100, -569, -929, -609, -929},
05254 { -100, -569, -929, -609, -929}},
05255 /* GC.AA..UG */
05256 {{ DEF, -519, -879, -559, -879},
05257 { -769, -1238, -1598, -1278, -1598},
05258 { -529, -998, -1358, -1038, -1358},
05259 { -709, -1178, -1538, -1218, -1538},
05260 { -599, -1068, -1428, -1108, -1428}},
05261 /* GC.AC..UG */
05262 {{ DEF, -519, -879, -559, -879},
05263 { -839, -1308, -1668, -1348, -1668},
05264 { -529, -998, -1358, -1038, -1358},
05265 { -859, -1328, -1688, -1368, -1688},
05266 { -489, -958, -1318, -998, -1318}},
05267 /* GC.AG..UG */
05268 {{ DEF, -519, -879, -559, -879},
05269 { -1009, -1478, -1838, -1518, -1838},
05270 { -409, -878, -1238, -918, -1238},
05271 { -969, -1438, -1798, -1478, -1798},
05272 { -599, -1068, -1428, -1108, -1428}},
05273 /* GC.AU..UG */
05274 {{ DEF, -519, -879, -559, -879},
05275 { -859, -1328, -1688, -1368, -1688},
05276 { -529, -998, -1358, -1038, -1358},
05277 { -859, -1328, -1688, -1368, -1688},
05278 { -409, -878, -1238, -918, -1238}}},
05279 /* GC.C@..UG */
05280 {{{ DEF, -719, -309, -309, -389},
05281 { -100, -769, -359, -359, -439},
05282 { -100, -769, -359, -359, -439},
05283 { -100, -769, -359, -359, -439},
05284 { -100, -769, -359, -359, -439}},
05285 /* GC.CA..UG */
05286 {{ DEF, -719, -309, -309, -389},
05287 { -769, -1438, -1028, -1028, -1108},
05288 { -529, -1198, -788, -788, -868},
05289 { -709, -1378, -968, -968, -1048},
05290 { -599, -1268, -858, -858, -938}},
05291 /* GC.CC..UG */
05292 {{ DEF, -719, -309, -309, -389},
05293 { -839, -1508, -1098, -1098, -1178},
05294 { -529, -1198, -788, -788, -868},
05295 { -859, -1528, -1118, -1118, -1198},
05296 { -489, -1158, -748, -748, -828}},
05297 /* GC.CG..UG */
05298 {{ DEF, -719, -309, -309, -389},
05299 { -1009, -1678, -1268, -1268, -1348},
05300 { -409, -1078, -668, -668, -748},
05301 { -969, -1638, -1228, -1228, -1308},
05302 { -599, -1268, -858, -858, -938}},
05303 /* GC.CU..UG */
05304 {{ DEF, -719, -309, -309, -389},
05305 { -859, -1528, -1118, -1118, -1198},
05306 { -529, -1198, -788, -788, -868},
05307 { -859, -1528, -1118, -1118, -1198},
05308 { -409, -1078, -668, -668, -748}}},
05309 /* GC.G@..UG */
05310 {{{ DEF, -709, -739, -619, -739},
05311 { -100, -759, -789, -669, -789},
05312 { -100, -759, -789, -669, -789},
05313 { -100, -759, -789, -669, -789},

```

```

05314 { -100, -759, -789, -669, -789}},
05315 /* GC.GA..UG */
05316 {{ DEF, -709, -739, -619, -739},
05317 { -769, -1428, -1458, -1338, -1458},
05318 { -529, -1188, -1218, -1098, -1218},
05319 { -709, -1368, -1398, -1278, -1398},
05320 { -599, -1258, -1288, -1168, -1288}},
05321 /* GC.GC..UG */
05322 {{ DEF, -709, -739, -619, -739},
05323 { -839, -1498, -1528, -1408, -1528},
05324 { -529, -1188, -1218, -1098, -1218},
05325 { -859, -1518, -1548, -1428, -1548},
05326 { -489, -1148, -1178, -1058, -1178}},
05327 /* GC.GG..UG */
05328 {{ DEF, -709, -739, -619, -739},
05329 {-1009, -1668, -1698, -1578, -1698},
05330 { -409, -1068, -1098, -978, -1098},
05331 { -969, -1628, -1658, -1538, -1658},
05332 { -599, -1258, -1288, -1168, -1288}},
05333 /* GC.GU..UG */
05334 {{ DEF, -709, -739, -619, -739},
05335 { -859, -1518, -1548, -1428, -1548},
05336 { -529, -1188, -1218, -1098, -1218},
05337 { -859, -1518, -1548, -1428, -1548},
05338 { -409, -1068, -1098, -978, -1098}}},
05339 /* GC.U@..UG */
05340 {{{ DEF, -499, -499, -499, -569},
05341 { -100, -549, -549, -549, -619},
05342 { -100, -549, -549, -549, -619},
05343 { -100, -549, -549, -549, -619},
05344 { -100, -549, -549, -549, -619}},
05345 /* GC.UA..UG */
05346 {{ DEF, -499, -499, -499, -569},
05347 { -769, -1218, -1218, -1218, -1288},
05348 { -529, -978, -978, -978, -1048},
05349 { -709, -1158, -1158, -1158, -1228},
05350 { -599, -1048, -1048, -1048, -1118}},
05351 /* GC.UC..UG */
05352 {{ DEF, -499, -499, -499, -569},
05353 { -839, -1288, -1288, -1288, -1358},
05354 { -529, -978, -978, -978, -1048},
05355 { -859, -1308, -1308, -1308, -1378},
05356 { -489, -938, -938, -938, -1008}},
05357 /* GC.UG..UG */
05358 {{ DEF, -499, -499, -499, -569},
05359 {-1009, -1458, -1458, -1458, -1528},
05360 { -409, -858, -858, -858, -928},
05361 { -969, -1418, -1418, -1418, -1488},
05362 { -599, -1048, -1048, -1048, -1118}},
05363 /* GC.UU..UG */
05364 {{ DEF, -499, -499, -499, -569},
05365 { -859, -1308, -1308, -1308, -1378},
05366 { -529, -978, -978, -978, -1048},
05367 { -859, -1308, -1308, -1308, -1378},
05368 { -409, -858, -858, -858, -928}}}},
05369 /* GC.@@..AU */
05370 {{{{ 0, 0, 0, 0, 0},
05371 { DEF, DEF, DEF, DEF, DEF},
05372 { DEF, DEF, DEF, DEF, DEF},
05373 { DEF, DEF, DEF, DEF, DEF},
05374 { DEF, DEF, DEF, DEF, DEF}}},
05375 /* GC.@A..AU */
05376 {{ 0, 0, 0, 0, 0},
05377 { -429, -429, -429, -429, -429},
05378 { -259, -259, -259, -259, -259},
05379 { -339, -339, -339, -339, -339},
05380 { -329, -329, -329, -329, -329}},
05381 /* GC.@C..AU */
05382 {{ 0, 0, 0, 0, 0},
05383 { -599, -599, -599, -599, -599},
05384 { -239, -239, -239, -239, -239},
05385 { -689, -689, -689, -689, -689},
05386 { -329, -329, -329, -329, -329}},
05387 /* GC.@G..AU */
05388 {{ 0, 0, 0, 0, 0},
05389 { -599, -599, -599, -599, -599},
05390 { -239, -239, -239, -239, -239},
05391 { -689, -689, -689, -689, -689},
05392 { -329, -329, -329, -329, -329}},
05393 /* GC.@U..AU */
05394 {{ 0, 0, 0, 0, 0},
05395 { -599, -599, -599, -599, -599},
05396 { -239, -239, -239, -239, -239},
05397 { -689, -689, -689, -689, -689},
05398 { -329, -329, -329, -329, -329}}},
05399 /* GC.A@..AU */
05400 {{{ DEF, -519, -879, -559, -879},

```

```
05401 { -100, -569, -929, -609, -929},
05402 { -100, -569, -929, -609, -929},
05403 { -100, -569, -929, -609, -929},
05404 { -100, -569, -929, -609, -929}},
05405 /* GC.AA..AU */
05406 {{ DEF, -519, -879, -559, -879},
05407 { -479, -948, -1308, -988, -1308},
05408 { -309, -778, -1138, -818, -1138},
05409 { -389, -858, -1218, -898, -1218},
05410 { -379, -848, -1208, -888, -1208}},
05411 /* GC.AC..AU */
05412 {{ DEF, -519, -879, -559, -879},
05413 { -649, -1118, -1478, -1158, -1478},
05414 { -289, -758, -1118, -798, -1118},
05415 { -739, -1208, -1568, -1248, -1568},
05416 { -379, -848, -1208, -888, -1208}},
05417 /* GC.AG..AU */
05418 {{ DEF, -519, -879, -559, -879},
05419 { -649, -1118, -1478, -1158, -1478},
05420 { -289, -758, -1118, -798, -1118},
05421 { -739, -1208, -1568, -1248, -1568},
05422 { -379, -848, -1208, -888, -1208}},
05423 /* GC.AU..AU */
05424 {{ DEF, -519, -879, -559, -879},
05425 { -649, -1118, -1478, -1158, -1478},
05426 { -289, -758, -1118, -798, -1118},
05427 { -739, -1208, -1568, -1248, -1568},
05428 { -379, -848, -1208, -888, -1208}}},
05429 /* GC.C@..AU */
05430 {{{ DEF, -719, -309, -309, -389},
05431 { -100, -769, -359, -359, -439},
05432 { -100, -769, -359, -359, -439},
05433 { -100, -769, -359, -359, -439},
05434 { -100, -769, -359, -359, -439}},
05435 /* GC.CA..AU */
05436 {{ DEF, -719, -309, -309, -389},
05437 { -479, -1148, -738, -738, -818},
05438 { -309, -978, -568, -568, -648},
05439 { -389, -1058, -648, -648, -728},
05440 { -379, -1048, -638, -638, -718}},
05441 /* GC.CC..AU */
05442 {{ DEF, -719, -309, -309, -389},
05443 { -649, -1318, -908, -908, -988},
05444 { -289, -958, -548, -548, -628},
05445 { -739, -1408, -998, -998, -1078},
05446 { -379, -1048, -638, -638, -718}},
05447 /* GC.CG..AU */
05448 {{ DEF, -719, -309, -309, -389},
05449 { -649, -1318, -908, -908, -988},
05450 { -289, -958, -548, -548, -628},
05451 { -739, -1408, -998, -998, -1078},
05452 { -379, -1048, -638, -638, -718}},
05453 /* GC.CU..AU */
05454 {{ DEF, -719, -309, -309, -389},
05455 { -649, -1318, -908, -908, -988},
05456 { -289, -958, -548, -548, -628},
05457 { -739, -1408, -998, -998, -1078},
05458 { -379, -1048, -638, -638, -718}}},
05459 /* GC.G@..AU */
05460 {{{ DEF, -709, -739, -619, -739},
05461 { -100, -759, -789, -669, -789},
05462 { -100, -759, -789, -669, -789},
05463 { -100, -759, -789, -669, -789},
05464 { -100, -759, -789, -669, -789}},
05465 /* GC.GA..AU */
05466 {{ DEF, -709, -739, -619, -739},
05467 { -479, -1138, -1168, -1048, -1168},
05468 { -309, -968, -998, -878, -998},
05469 { -389, -1048, -1078, -958, -1078},
05470 { -379, -1038, -1068, -948, -1068}},
05471 /* GC.GC..AU */
05472 {{ DEF, -709, -739, -619, -739},
05473 { -649, -1308, -1338, -1218, -1338},
05474 { -289, -948, -978, -858, -978},
05475 { -739, -1398, -1428, -1308, -1428},
05476 { -379, -1038, -1068, -948, -1068}},
05477 /* GC.GG..AU */
05478 {{ DEF, -709, -739, -619, -739},
05479 { -649, -1308, -1338, -1218, -1338},
05480 { -289, -948, -978, -858, -978},
05481 { -739, -1398, -1428, -1308, -1428},
05482 { -379, -1038, -1068, -948, -1068}},
05483 /* GC.GU..AU */
05484 {{ DEF, -709, -739, -619, -739},
05485 { -649, -1308, -1338, -1218, -1338},
05486 { -289, -948, -978, -858, -978},
05487 { -739, -1398, -1428, -1308, -1428},
```

```

05488 { -379,-1038,-1068, -948,-1068}},
05489 /* GC.U@.AU */
05490 {{ DEF, -499, -499, -499, -569},
05491 { -100, -549, -549, -549, -619},
05492 { -100, -549, -549, -549, -619},
05493 { -100, -549, -549, -549, -619},
05494 { -100, -549, -549, -549, -619}},
05495 /* GC.UA.AU */
05496 {{ DEF, -499, -499, -499, -569},
05497 { -479, -928, -928, -928, -998},
05498 { -309, -758, -758, -758, -828},
05499 { -389, -838, -838, -838, -908},
05500 { -379, -828, -828, -828, -898}},
05501 /* GC.UC.AU */
05502 {{ DEF, -499, -499, -499, -569},
05503 { -649,-1098,-1098,-1098,-1168},
05504 { -289, -738, -738, -738, -808},
05505 { -739,-1188,-1188,-1188,-1258},
05506 { -379, -828, -828, -828, -898}},
05507 /* GC.UG.AU */
05508 {{ DEF, -499, -499, -499, -569},
05509 { -649,-1098,-1098,-1098,-1168},
05510 { -289, -738, -738, -738, -808},
05511 { -739,-1188,-1188,-1188,-1258},
05512 { -379, -828, -828, -828, -898}},
05513 /* GC.UU.AU */
05514 {{ DEF, -499, -499, -499, -569},
05515 { -649,-1098,-1098,-1098,-1168},
05516 { -289, -738, -738, -738, -808},
05517 { -739,-1188,-1188,-1188,-1258},
05518 { -379, -828, -828, -828, -898}}}},
05519 /* GC.@.UA */
05520 {{{ 0, 0, 0, 0, 0},
05521 { DEF, DEF, DEF, DEF, DEF},
05522 { DEF, DEF, DEF, DEF, DEF},
05523 { DEF, DEF, DEF, DEF, DEF},
05524 { DEF, DEF, DEF, DEF, DEF}},
05525 /* GC.@A.UA */
05526 {{ 0, 0, 0, 0, 0},
05527 { -399, -399, -399, -399, -399},
05528 { -429, -429, -429, -429, -429},
05529 { -379, -379, -379, -379, -379},
05530 { -279, -279, -279, -279, -279}},
05531 /* GC.@C.UA */
05532 {{ 0, 0, 0, 0, 0},
05533 { -629, -629, -629, -629, -629},
05534 { -509, -509, -509, -509, -509},
05535 { -679, -679, -679, -679, -679},
05536 { -139, -139, -139, -139, -139}},
05537 /* GC.@G.UA */
05538 {{ 0, 0, 0, 0, 0},
05539 { -889, -889, -889, -889, -889},
05540 { -199, -199, -199, -199, -199},
05541 { -889, -889, -889, -889, -889},
05542 { -279, -279, -279, -279, -279}},
05543 /* GC.@U.UA */
05544 {{ 0, 0, 0, 0, 0},
05545 { -589, -589, -589, -589, -589},
05546 { -179, -179, -179, -179, -179},
05547 { -679, -679, -679, -679, -679},
05548 { -140, -140, -140, -140, -140}}}},
05549 /* GC.A@.UA */
05550 {{{ DEF, -519, -879, -559, -879},
05551 { -100, -569, -929, -609, -929},
05552 { -100, -569, -929, -609, -929},
05553 { -100, -569, -929, -609, -929},
05554 { -100, -569, -929, -609, -929}},
05555 /* GC.AA.UA */
05556 {{ DEF, -519, -879, -559, -879},
05557 { -449, -918, -1278, -958, -1278},
05558 { -479, -948, -1308, -988, -1308},
05559 { -429, -898, -1258, -938, -1258},
05560 { -329, -798, -1158, -838, -1158}},
05561 /* GC.AC.UA */
05562 {{ DEF, -519, -879, -559, -879},
05563 { -679, -1148, -1508, -1188, -1508},
05564 { -559, -1028, -1388, -1068, -1388},
05565 { -729, -1198, -1558, -1238, -1558},
05566 { -189, -658, -1018, -698, -1018}},
05567 /* GC.AG.UA */
05568 {{ DEF, -519, -879, -559, -879},
05569 { -939, -1408, -1768, -1448, -1768},
05570 { -249, -718, -1078, -758, -1078},
05571 { -939, -1408, -1768, -1448, -1768},
05572 { -329, -798, -1158, -838, -1158}},
05573 /* GC.AU.UA */
05574 {{ DEF, -519, -879, -559, -879},

```

```
05575 { -639,-1108,-1468,-1148,-1468},
05576 { -229, -698,-1058, -738,-1058},
05577 { -729,-1198,-1558,-1238,-1558},
05578 { -190, -659,-1019, -699,-1019}},
05579 /* GC.C@..UA */
05580 {{{ DEF, -719, -309, -309, -389},
05581 { -100, -769, -359, -359, -439},
05582 { -100, -769, -359, -359, -439},
05583 { -100, -769, -359, -359, -439},
05584 { -100, -769, -359, -359, -439}},
05585 /* GC.CA..UA */
05586 {{ DEF, -719, -309, -309, -389},
05587 { -449,-1118, -708, -708, -788},
05588 { -479,-1148, -738, -738, -818},
05589 { -429,-1098, -688, -688, -768},
05590 { -329, -998, -588, -588, -668}},
05591 /* GC.CC..UA */
05592 {{ DEF, -719, -309, -309, -389},
05593 { -679,-1348, -938, -938,-1018},
05594 { -559,-1228, -818, -818, -898},
05595 { -729,-1398, -988, -988,-1068},
05596 { -189, -858, -448, -448, -528}},
05597 /* GC.CG..UA */
05598 {{ DEF, -719, -309, -309, -389},
05599 { -939,-1608,-1198,-1198,-1278},
05600 { -249, -918, -508, -508, -588},
05601 { -939,-1608,-1198,-1198,-1278},
05602 { -329, -998, -588, -588, -668}},
05603 /* GC.CU..UA */
05604 {{ DEF, -719, -309, -309, -389},
05605 { -639,-1308, -898, -898, -978},
05606 { -229, -898, -488, -488, -568},
05607 { -729,-1398, -988, -988,-1068},
05608 { -190, -859, -449, -449, -529}},
05609 /* GC.G@..UA */
05610 {{{ DEF, -709, -739, -619, -739},
05611 { -100, -759, -789, -669, -789},
05612 { -100, -759, -789, -669, -789},
05613 { -100, -759, -789, -669, -789},
05614 { -100, -759, -789, -669, -789}},
05615 /* GC.GA..UA */
05616 {{ DEF, -709, -739, -619, -739},
05617 { -449,-1108,-1138,-1018,-1138},
05618 { -479,-1138,-1168,-1048,-1168},
05619 { -429,-1088,-1118, -998,-1118},
05620 { -329, -988,-1018, -898,-1018}},
05621 /* GC.GC..UA */
05622 {{ DEF, -709, -739, -619, -739},
05623 { -679,-1338,-1368,-1248,-1368},
05624 { -559,-1218,-1248,-1128,-1248},
05625 { -729,-1388,-1418,-1298,-1418},
05626 { -189, -848, -878, -758, -878}},
05627 /* GC.GG..UA */
05628 {{ DEF, -709, -739, -619, -739},
05629 { -939,-1598,-1628,-1508,-1628},
05630 { -249, -908, -938, -818, -938},
05631 { -939,-1598,-1628,-1508,-1628},
05632 { -329, -988,-1018, -898,-1018}},
05633 /* GC.GU..UA */
05634 {{ DEF, -709, -739, -619, -739},
05635 { -639,-1298,-1328,-1208,-1328},
05636 { -229, -888, -918, -798, -918},
05637 { -729,-1388,-1418,-1298,-1418},
05638 { -190, -849, -879, -759, -879}},
05639 /* GC.U@..UA */
05640 {{{ DEF, -499, -499, -499, -569},
05641 { -100, -549, -549, -549, -619},
05642 { -100, -549, -549, -549, -619},
05643 { -100, -549, -549, -549, -619},
05644 { -100, -549, -549, -549, -619}},
05645 /* GC.UA..UA */
05646 {{ DEF, -499, -499, -499, -569},
05647 { -449, -898, -898, -898, -968},
05648 { -479, -928, -928, -928, -998},
05649 { -429, -878, -878, -878, -948},
05650 { -329, -778, -778, -778, -848}},
05651 /* GC.UC..UA */
05652 {{ DEF, -499, -499, -499, -569},
05653 { -679,-1128,-1128,-1128,-1198},
05654 { -559,-1008,-1008,-1008,-1078},
05655 { -729,-1178,-1178,-1178,-1248},
05656 { -189, -638, -638, -638, -708}},
05657 /* GC.UG..UA */
05658 {{ DEF, -499, -499, -499, -569},
05659 { -939,-1388,-1388,-1388,-1458},
05660 { -249, -698, -698, -698, -768},
05661 { -939,-1388,-1388,-1388,-1458},
```

```
05662 { -329, -778, -778, -778, -848}},
05663 /* GC.UU..UA */
05664 {{ DEF, -499, -499, -499, -569},
05665 { -639,-1088,-1088,-1088,-1158},
05666 { -229, -678, -678, -678, -748},
05667 { -729,-1178,-1178,-1178,-1248},
05668 { -190, -639, -639, -639, -709}}}},
05669 /* GC.@@.. @ */
05670 {{{ DEF, DEF, DEF, DEF, DEF},
05671 { DEF, DEF, DEF, DEF, DEF},
05672 { DEF, DEF, DEF, DEF, DEF},
05673 { DEF, DEF, DEF, DEF, DEF},
05674 { DEF, DEF, DEF, DEF, DEF}},
05675 /* GC.AA.. @ */
05676 {{ DEF, DEF, DEF, DEF, DEF},
05677 { DEF, DEF, DEF, DEF, DEF},
05678 { DEF, DEF, DEF, DEF, DEF},
05679 { DEF, DEF, DEF, DEF, DEF},
05680 { DEF, DEF, DEF, DEF, DEF}},
05681 /* GC.CC.. @ */
05682 {{ DEF, DEF, DEF, DEF, DEF},
05683 { DEF, DEF, DEF, DEF, DEF},
05684 { DEF, DEF, DEF, DEF, DEF},
05685 { DEF, DEF, DEF, DEF, DEF},
05686 { DEF, DEF, DEF, DEF, DEF}},
05687 /* GC.GG.. @ */
05688 {{ DEF, DEF, DEF, DEF, DEF},
05689 { DEF, DEF, DEF, DEF, DEF},
05690 { DEF, DEF, DEF, DEF, DEF},
05691 { DEF, DEF, DEF, DEF, DEF},
05692 { DEF, DEF, DEF, DEF, DEF}},
05693 /* GC.UU.. @ */
05694 {{ DEF, DEF, DEF, DEF, DEF},
05695 { DEF, DEF, DEF, DEF, DEF},
05696 { DEF, DEF, DEF, DEF, DEF},
05697 { DEF, DEF, DEF, DEF, DEF},
05698 { DEF, DEF, DEF, DEF, DEF}},
05699 /* GC.AA.. @ */
05700 {{{ -100, -569, -929, -609, -929},
05701 { -100, -569, -929, -609, -929},
05702 { -100, -569, -929, -609, -929},
05703 { -100, -569, -929, -609, -929},
05704 { -100, -569, -929, -609, -929}},
05705 /* GC.AA.. @ */
05706 {{ -100, -569, -929, -609, -929},
05707 { -100, -569, -929, -609, -929},
05708 { -100, -569, -929, -609, -929},
05709 { -100, -569, -929, -609, -929},
05710 { -100, -569, -929, -609, -929}},
05711 /* GC.AC.. @ */
05712 {{ -100, -569, -929, -609, -929},
05713 { -100, -569, -929, -609, -929},
05714 { -100, -569, -929, -609, -929},
05715 { -100, -569, -929, -609, -929},
05716 { -100, -569, -929, -609, -929}},
05717 /* GC.AG.. @ */
05718 {{ -100, -569, -929, -609, -929},
05719 { -100, -569, -929, -609, -929},
05720 { -100, -569, -929, -609, -929},
05721 { -100, -569, -929, -609, -929},
05722 { -100, -569, -929, -609, -929}},
05723 /* GC.AU.. @ */
05724 {{{ -100, -569, -929, -609, -929},
05725 { -100, -569, -929, -609, -929},
05726 { -100, -569, -929, -609, -929},
05727 { -100, -569, -929, -609, -929},
05728 { -100, -569, -929, -609, -929}}}},
05729 /* GC.CC.. @ */
05730 {{{ -100, -769, -359, -359, -439},
05731 { -100, -769, -359, -359, -439},
05732 { -100, -769, -359, -359, -439},
05733 { -100, -769, -359, -359, -439},
05734 { -100, -769, -359, -359, -439}},
05735 /* GC.CA.. @ */
05736 {{{ -100, -769, -359, -359, -439},
05737 { -100, -769, -359, -359, -439},
05738 { -100, -769, -359, -359, -439},
05739 { -100, -769, -359, -359, -439},
05740 { -100, -769, -359, -359, -439}},
05741 /* GC.CC.. @ */
05742 {{{ -100, -769, -359, -359, -439},
05743 { -100, -769, -359, -359, -439},
05744 { -100, -769, -359, -359, -439},
05745 { -100, -769, -359, -359, -439},
05746 { -100, -769, -359, -359, -439}},
05747 /* GC.CG.. @ */
05748 {{ -100, -769, -359, -359, -439},
```

```

05749 { -100, -769, -359, -359, -439},
05750 { -100, -769, -359, -359, -439},
05751 { -100, -769, -359, -359, -439},
05752 { -100, -769, -359, -359, -439}},
05753 /* GC.CU.. @ */
05754 {{ -100, -769, -359, -359, -439},
05755 { -100, -769, -359, -359, -439},
05756 { -100, -769, -359, -359, -439},
05757 { -100, -769, -359, -359, -439},
05758 { -100, -769, -359, -359, -439}}},
05759 /* GC.G@.. @ */
05760 {{{ -100, -759, -789, -669, -789},
05761 { -100, -759, -789, -669, -789},
05762 { -100, -759, -789, -669, -789},
05763 { -100, -759, -789, -669, -789},
05764 { -100, -759, -789, -669, -789}},
05765 /* GC.GA.. @ */
05766 {{ -100, -759, -789, -669, -789},
05767 { -100, -759, -789, -669, -789},
05768 { -100, -759, -789, -669, -789},
05769 { -100, -759, -789, -669, -789},
05770 { -100, -759, -789, -669, -789}},
05771 /* GC.GC.. @ */
05772 {{ -100, -759, -789, -669, -789},
05773 { -100, -759, -789, -669, -789},
05774 { -100, -759, -789, -669, -789},
05775 { -100, -759, -789, -669, -789},
05776 { -100, -759, -789, -669, -789}},
05777 /* GC.GG.. @ */
05778 {{ -100, -759, -789, -669, -789},
05779 { -100, -759, -789, -669, -789},
05780 { -100, -759, -789, -669, -789},
05781 { -100, -759, -789, -669, -789},
05782 { -100, -759, -789, -669, -789}},
05783 /* GC.GU.. @ */
05784 {{ -100, -759, -789, -669, -789},
05785 { -100, -759, -789, -669, -789},
05786 { -100, -759, -789, -669, -789},
05787 { -100, -759, -789, -669, -789},
05788 { -100, -759, -789, -669, -789}}},
05789 /* GC.U@.. @ */
05790 {{{ -100, -549, -549, -549, -619},
05791 { -100, -549, -549, -549, -619},
05792 { -100, -549, -549, -549, -619},
05793 { -100, -549, -549, -549, -619},
05794 { -100, -549, -549, -549, -619}},
05795 /* GC.UA.. @ */
05796 {{ -100, -549, -549, -549, -619},
05797 { -100, -549, -549, -549, -619},
05798 { -100, -549, -549, -549, -619},
05799 { -100, -549, -549, -549, -619},
05800 { -100, -549, -549, -549, -619}},
05801 /* GC.UC.. @ */
05802 {{ -100, -549, -549, -549, -619},
05803 { -100, -549, -549, -549, -619},
05804 { -100, -549, -549, -549, -619},
05805 { -100, -549, -549, -549, -619},
05806 { -100, -549, -549, -549, -619}},
05807 /* GC.UG.. @ */
05808 {{ -100, -549, -549, -549, -619},
05809 { -100, -549, -549, -549, -619},
05810 { -100, -549, -549, -549, -619},
05811 { -100, -549, -549, -549, -619},
05812 { -100, -549, -549, -549, -619}},
05813 /* GC.UU.. @ */
05814 {{ -100, -549, -549, -549, -619},
05815 { -100, -549, -549, -549, -619},
05816 { -100, -549, -549, -549, -619},
05817 { -100, -549, -549, -549, -619},
05818 { -100, -549, -549, -549, -619}}}},
05819 { /* noPair */ {{{{0}}}},
05820 /* GU.@@..CG */
05821 {{{{ 0, 0, 0, 0, 0},
05822 { DEF, DEF, DEF, DEF, DEF},
05823 { DEF, DEF, DEF, DEF, DEF},
05824 { DEF, DEF, DEF, DEF, DEF},
05825 { DEF, DEF, DEF, DEF, DEF}}},
05826 /* GU.@A..CG */
05827 {{ 0, 0, 0, 0, 0},
05828 {-1029, -1029, -1029, -1029, -1029},
05829 {-519, -519, -519, -519, -519},
05830 {-939, -939, -939, -939, -939},
05831 {-809, -809, -809, -809, -809}},
05832 /* GU.@C..CG */
05833 {{ 0, 0, 0, 0, 0},
05834 {-949, -949, -949, -949, -949},
05835 {-449, -449, -449, -449, -449},

```



```
05836 { -939, -939, -939, -939, -939},
05837 { -739, -739, -739, -739, -739}},
05838 /* GU.@G..CG */
05839 {{ 0, 0, 0, 0, 0},
05840 {-1029,-1029,-1029,-1029,-1029},
05841 { -519, -519, -519, -519, -519},
05842 { -939, -939, -939, -939, -939},
05843 { -809, -809, -809, -809, -809}},
05844 /* GU.@U..CG */
05845 {{ 0, 0, 0, 0, 0},
05846 {-1029,-1029,-1029,-1029,-1029},
05847 { -669, -669, -669, -669, -669},
05848 { -939, -939, -939, -939, -939},
05849 { -859, -859, -859, -859, -859}},
05850 /* GU.A@..CG */
05851 {{{ DEF, -429, -599, -599, -599},
05852 { -100, -479, -649, -649, -649},
05853 { -100, -479, -649, -649, -649},
05854 { -100, -479, -649, -649, -649},
05855 { -100, -479, -649, -649, -649}},
05856 /* GU.AA..CG */
05857 {{ DEF, -429, -599, -599, -599},
05858 {-1079,-1458,-1628,-1628,-1628},
05859 { -569, -948,-1118,-1118,-1118},
05860 { -989,-1368,-1538,-1538,-1538},
05861 { -859,-1238,-1408,-1408,-1408}},
05862 /* GU.AC..CG */
05863 {{ DEF, -429, -599, -599, -599},
05864 { -999,-1378,-1548,-1548,-1548},
05865 { -499, -878,-1048,-1048,-1048},
05866 { -989,-1368,-1538,-1538,-1538},
05867 { -789,-1168,-1338,-1338,-1338}},
05868 /* GU.AG..CG */
05869 {{ DEF, -429, -599, -599, -599},
05870 {-1079,-1458,-1628,-1628,-1628},
05871 { -569, -948,-1118,-1118,-1118},
05872 { -989,-1368,-1538,-1538,-1538},
05873 { -859,-1238,-1408,-1408,-1408}},
05874 /* GU.AU..CG */
05875 {{ DEF, -429, -599, -599, -599},
05876 {-1079,-1458,-1628,-1628,-1628},
05877 { -719,-1098,-1268,-1268,-1268},
05878 { -989,-1368,-1538,-1538,-1538},
05879 { -909,-1288,-1458,-1458,-1458}},
05880 /* GU.C@..CG */
05881 {{{ DEF, -259, -239, -239, -239},
05882 { -100, -309, -289, -289, -289},
05883 { -100, -309, -289, -289, -289},
05884 { -100, -309, -289, -289, -289},
05885 { -100, -309, -289, -289, -289}},
05886 /* GU.CA..CG */
05887 {{ DEF, -259, -239, -239, -239},
05888 {-1079,-1288,-1268,-1268,-1268},
05889 { -569, -778, -758, -758, -758},
05890 { -989,-1198,-1178,-1178,-1178},
05891 { -859,-1068,-1048,-1048,-1048}},
05892 /* GU.CC..CG */
05893 {{ DEF, -259, -239, -239, -239},
05894 { -999,-1208,-1188,-1188,-1188},
05895 { -499, -708, -688, -688, -688},
05896 { -989,-1198,-1178,-1178,-1178},
05897 { -789, -998, -978, -978, -978}},
05898 /* GU.CG..CG */
05899 {{ DEF, -259, -239, -239, -239},
05900 {-1079,-1288,-1268,-1268,-1268},
05901 { -569, -778, -758, -758, -758},
05902 { -989,-1198,-1178,-1178,-1178},
05903 { -859,-1068,-1048,-1048,-1048}},
05904 /* GU.CU..CG */
05905 {{ DEF, -259, -239, -239, -239},
05906 {-1079,-1288,-1268,-1268,-1268},
05907 { -719, -928, -908, -908, -908},
05908 { -989,-1198,-1178,-1178,-1178},
05909 { -909,-1118,-1098,-1098,-1098}},
05910 /* GU.G@..CG */
05911 {{{ DEF, -339, -689, -689, -689},
05912 { -100, -389, -739, -739, -739},
05913 { -100, -389, -739, -739, -739},
05914 { -100, -389, -739, -739, -739},
05915 { -100, -389, -739, -739, -739}},
05916 /* GU.GA..CG */
05917 {{ DEF, -339, -689, -689, -689},
05918 {-1079,-1368,-1718,-1718,-1718},
05919 { -569, -858,-1208,-1208,-1208},
05920 { -989,-1278,-1628,-1628,-1628},
05921 { -859,-1148,-1498,-1498,-1498}},
05922 /* GU.GC..CG */
```

```

05923 {{ DEF, -339, -689, -689, -689},
05924 { -999, -1288, -1638, -1638, -1638},
05925 { -499, -788, -1138, -1138, -1138},
05926 { -989, -1278, -1628, -1628, -1628},
05927 { -789, -1078, -1428, -1428, -1428}},
05928 /* GU.GG..CG */
05929 {{ DEF, -339, -689, -689, -689},
05930 {-1079, -1368, -1718, -1718, -1718},
05931 { -569, -858, -1208, -1208, -1208},
05932 { -989, -1278, -1628, -1628, -1628},
05933 { -859, -1148, -1498, -1498, -1498}},
05934 /* GU.GU..CG */
05935 {{ DEF, -339, -689, -689, -689},
05936 {-1079, -1368, -1718, -1718, -1718},
05937 { -719, -1008, -1358, -1358, -1358},
05938 { -989, -1278, -1628, -1628, -1628},
05939 { -909, -1198, -1548, -1548, -1548}}},
05940 /* GU.U@..CG */
05941 {{{ DEF, -329, -329, -329, -329},
05942 { -100, -379, -379, -379, -379},
05943 { -100, -379, -379, -379, -379},
05944 { -100, -379, -379, -379, -379},
05945 { -100, -379, -379, -379, -379}},
05946 /* GU.UA..CG */
05947 {{ DEF, -329, -329, -329, -329},
05948 {-1079, -1358, -1358, -1358, -1358},
05949 { -569, -848, -848, -848, -848},
05950 { -989, -1268, -1268, -1268, -1268},
05951 { -859, -1138, -1138, -1138, -1138}},
05952 /* GU.UC..CG */
05953 {{ DEF, -329, -329, -329, -329},
05954 { -999, -1278, -1278, -1278, -1278},
05955 { -499, -778, -778, -778, -778},
05956 { -989, -1268, -1268, -1268, -1268},
05957 { -789, -1068, -1068, -1068, -1068}},
05958 /* GU.UG..CG */
05959 {{ DEF, -329, -329, -329, -329},
05960 {-1079, -1358, -1358, -1358, -1358},
05961 { -569, -848, -848, -848, -848},
05962 { -989, -1268, -1268, -1268, -1268},
05963 { -859, -1138, -1138, -1138, -1138}},
05964 /* GU.UU..CG */
05965 {{ DEF, -329, -329, -329, -329},
05966 {-1079, -1358, -1358, -1358, -1358},
05967 { -719, -998, -998, -998, -998},
05968 { -989, -1268, -1268, -1268, -1268},
05969 { -909, -1188, -1188, -1188, -1188}}},
05970 /* GU.@@..GC */
05971 {{{ 0, 0, 0, 0, 0},
05972 { DEF, DEF, DEF, DEF, DEF},
05973 { DEF, DEF, DEF, DEF, DEF},
05974 { DEF, DEF, DEF, DEF, DEF},
05975 { DEF, DEF, DEF, DEF, DEF}},
05976 /* GU.@A..GC */
05977 {{ 0, 0, 0, 0, 0},
05978 { -519, -519, -519, -519, -519},
05979 { -719, -719, -719, -719, -719},
05980 { -709, -709, -709, -709, -709},
05981 { -499, -499, -499, -499, -499}},
05982 /* GU.@C..GC */
05983 {{ 0, 0, 0, 0, 0},
05984 { -879, -879, -879, -879, -879},
05985 { -309, -309, -309, -309, -309},
05986 { -739, -739, -739, -739, -739},
05987 { -499, -499, -499, -499, -499}},
05988 /* GU.@G..GC */
05989 {{ 0, 0, 0, 0, 0},
05990 { -559, -559, -559, -559, -559},
05991 { -309, -309, -309, -309, -309},
05992 { -619, -619, -619, -619, -619},
05993 { -499, -499, -499, -499, -499}},
05994 /* GU.@U..GC */
05995 {{ 0, 0, 0, 0, 0},
05996 { -879, -879, -879, -879, -879},
05997 { -389, -389, -389, -389, -389},
05998 { -739, -739, -739, -739, -739},
05999 { -569, -569, -569, -569, -569}}},
06000 /* GU.A@..GC */
06001 {{{ DEF, -429, -599, -599, -599},
06002 { -100, -479, -649, -649, -649},
06003 { -100, -479, -649, -649, -649},
06004 { -100, -479, -649, -649, -649},
06005 { -100, -479, -649, -649, -649}},
06006 /* GU.AA..GC */
06007 {{ DEF, -429, -599, -599, -599},
06008 { -569, -948, -1118, -1118, -1118},
06009 { -769, -1148, -1318, -1318, -1318},

```

```
06010 { -759,-1138,-1308,-1308,-1308},
06011 { -549, -928,-1098,-1098,-1098}},
06012 /* GU.AC..GC */
06013 {{ DEF, -429, -599, -599, -599},
06014 { -929,-1308,-1478,-1478,-1478},
06015 { -359, -738, -908, -908, -908},
06016 { -789,-1168,-1338,-1338,-1338},
06017 { -549, -928,-1098,-1098,-1098}},
06018 /* GU.AG..GC */
06019 {{ DEF, -429, -599, -599, -599},
06020 { -609, -988,-1158,-1158,-1158},
06021 { -359, -738, -908, -908, -908},
06022 { -669,-1048,-1218,-1218,-1218},
06023 { -549, -928,-1098,-1098,-1098}},
06024 /* GU.AU..GC */
06025 {{ DEF, -429, -599, -599, -599},
06026 { -929,-1308,-1478,-1478,-1478},
06027 { -439, -818, -988, -988, -988},
06028 { -789,-1168,-1338,-1338,-1338},
06029 { -619, -998,-1168,-1168,-1168}},
06030 /* GU.C@..GC */
06031 {{{ DEF, -259, -239, -239, -239},
06032 { -100, -309, -289, -289, -289},
06033 { -100, -309, -289, -289, -289},
06034 { -100, -309, -289, -289, -289},
06035 { -100, -309, -289, -289, -289}},
06036 /* GU.CA..GC */
06037 {{ DEF, -259, -239, -239, -239},
06038 { -569, -778, -758, -758, -758},
06039 { -769, -978, -958, -958, -958},
06040 { -759, -968, -948, -948, -948},
06041 { -549, -758, -738, -738, -738}},
06042 /* GU.CC..GC */
06043 {{ DEF, -259, -239, -239, -239},
06044 { -929,-1138,-1118,-1118,-1118},
06045 { -359, -568, -548, -548, -548},
06046 { -789, -998, -978, -978, -978},
06047 { -549, -758, -738, -738, -738}},
06048 /* GU.CG..GC */
06049 {{ DEF, -259, -239, -239, -239},
06050 { -609, -818, -798, -798, -798},
06051 { -359, -568, -548, -548, -548},
06052 { -669, -878, -858, -858, -858},
06053 { -549, -758, -738, -738, -738}},
06054 /* GU.CU..GC */
06055 {{ DEF, -259, -239, -239, -239},
06056 { -929,-1138,-1118,-1118,-1118},
06057 { -439, -648, -628, -628, -628},
06058 { -789, -998, -978, -978, -978},
06059 { -619, -828, -808, -808, -808}},
06060 /* GU.G@..GC */
06061 {{{ DEF, -339, -689, -689, -689},
06062 { -100, -389, -739, -739, -739},
06063 { -100, -389, -739, -739, -739},
06064 { -100, -389, -739, -739, -739},
06065 { -100, -389, -739, -739, -739}},
06066 /* GU.GA..GC */
06067 {{ DEF, -339, -689, -689, -689},
06068 { -569, -858,-1208,-1208,-1208},
06069 { -769,-1058,-1408,-1408,-1408},
06070 { -759,-1048,-1398,-1398,-1398},
06071 { -549, -838,-1188,-1188,-1188}},
06072 /* GU.GC..GC */
06073 {{ DEF, -339, -689, -689, -689},
06074 { -929,-1218,-1568,-1568,-1568},
06075 { -359, -648, -998, -998, -998},
06076 { -789,-1078,-1428,-1428,-1428},
06077 { -549, -838,-1188,-1188,-1188}},
06078 /* GU.GG..GC */
06079 {{ DEF, -339, -689, -689, -689},
06080 { -609, -898,-1248,-1248,-1248},
06081 { -359, -648, -998, -998, -998},
06082 { -669, -958,-1308,-1308,-1308},
06083 { -549, -838,-1188,-1188,-1188}},
06084 /* GU.GU..GC */
06085 {{ DEF, -339, -689, -689, -689},
06086 { -929,-1218,-1568,-1568,-1568},
06087 { -439, -728,-1078,-1078,-1078},
06088 { -789,-1078,-1428,-1428,-1428},
06089 { -619, -908,-1258,-1258,-1258}},
06090 /* GU.U@..GC */
06091 {{{ DEF, -329, -329, -329, -329},
06092 { -100, -379, -379, -379, -379},
06093 { -100, -379, -379, -379, -379},
06094 { -100, -379, -379, -379, -379},
06095 { -100, -379, -379, -379, -379}},
06096 /* GU.UA..GC */
```

```
06097 {{ DEF, -329, -329, -329, -329},
06098 { -569, -848, -848, -848, -848},
06099 { -769, -1048, -1048, -1048, -1048},
06100 { -759, -1038, -1038, -1038, -1038},
06101 { -549, -828, -828, -828, -828}},
06102 /* GU.UC..GC */
06103 {{ DEF, -329, -329, -329, -329},
06104 { -929, -1208, -1208, -1208, -1208},
06105 { -359, -638, -638, -638, -638},
06106 { -789, -1068, -1068, -1068, -1068},
06107 { -549, -828, -828, -828, -828}},
06108 /* GU.UG..GC */
06109 {{ DEF, -329, -329, -329, -329},
06110 { -609, -888, -888, -888, -888},
06111 { -359, -638, -638, -638, -638},
06112 { -669, -948, -948, -948, -948},
06113 { -549, -828, -828, -828, -828}},
06114 /* GU.UU..GC */
06115 {{ DEF, -329, -329, -329, -329},
06116 { -929, -1208, -1208, -1208, -1208},
06117 { -439, -718, -718, -718, -718},
06118 { -789, -1068, -1068, -1068, -1068},
06119 { -619, -898, -898, -898, -898}}}},
06120 /* GU.@@..GU */
06121 {{{ 0, 0, 0, 0, 0},
06122 { DEF, DEF, DEF, DEF, DEF},
06123 { DEF, DEF, DEF, DEF, DEF},
06124 { DEF, DEF, DEF, DEF, DEF},
06125 { DEF, DEF, DEF, DEF, DEF}},
06126 /* GU.@A..GU */
06127 {{ 0, 0, 0, 0, 0},
06128 { -429, -429, -429, -429, -429},
06129 { -259, -259, -259, -259, -259},
06130 { -339, -339, -339, -339, -339},
06131 { -329, -329, -329, -329, -329}},
06132 /* GU.@C..GU */
06133 {{ 0, 0, 0, 0, 0},
06134 { -599, -599, -599, -599, -599},
06135 { -239, -239, -239, -239, -239},
06136 { -689, -689, -689, -689, -689},
06137 { -329, -329, -329, -329, -329}},
06138 /* GU.@G..GU */
06139 {{ 0, 0, 0, 0, 0},
06140 { -599, -599, -599, -599, -599},
06141 { -239, -239, -239, -239, -239},
06142 { -689, -689, -689, -689, -689},
06143 { -329, -329, -329, -329, -329}},
06144 /* GU.@U..GU */
06145 {{ 0, 0, 0, 0, 0},
06146 { -599, -599, -599, -599, -599},
06147 { -239, -239, -239, -239, -239},
06148 { -689, -689, -689, -689, -689},
06149 { -329, -329, -329, -329, -329}},
06150 /* GU.@@..GU */
06151 {{{ DEF, -429, -599, -599, -599},
06152 { -100, -479, -649, -649, -649},
06153 { -100, -479, -649, -649, -649},
06154 { -100, -479, -649, -649, -649},
06155 { -100, -479, -649, -649, -649}},
06156 /* GU.AA..GU */
06157 {{ DEF, -429, -599, -599, -599},
06158 { -479, -858, -1028, -1028, -1028},
06159 { -309, -688, -858, -858, -858},
06160 { -389, -768, -938, -938, -938},
06161 { -379, -758, -928, -928, -928}},
06162 /* GU.AC..GU */
06163 {{ DEF, -429, -599, -599, -599},
06164 { -649, -1028, -1198, -1198, -1198},
06165 { -289, -668, -838, -838, -838},
06166 { -739, -1118, -1288, -1288, -1288},
06167 { -379, -758, -928, -928, -928}},
06168 /* GU.AG..GU */
06169 {{ DEF, -429, -599, -599, -599},
06170 { -649, -1028, -1198, -1198, -1198},
06171 { -289, -668, -838, -838, -838},
06172 { -739, -1118, -1288, -1288, -1288},
06173 { -379, -758, -928, -928, -928}},
06174 /* GU.AU..GU */
06175 {{ DEF, -429, -599, -599, -599},
06176 { -649, -1028, -1198, -1198, -1198},
06177 { -289, -668, -838, -838, -838},
06178 { -739, -1118, -1288, -1288, -1288},
06179 { -379, -758, -928, -928, -928}}}},
06180 /* GU.C@..GU */
06181 {{{ DEF, -259, -239, -239, -239},
06182 { -100, -309, -289, -289, -289},
06183 { -100, -309, -289, -289, -289},
```

```
06184 { -100, -309, -289, -289, -289},
06185 { -100, -309, -289, -289, -289}},
06186 /* GU.CA..GU */
06187 {{ DEF, -259, -239, -239, -239},
06188 { -479, -688, -668, -668, -668},
06189 { -309, -518, -498, -498, -498},
06190 { -389, -598, -578, -578, -578},
06191 { -379, -588, -568, -568, -568}},
06192 /* GU.CC..GU */
06193 {{ DEF, -259, -239, -239, -239},
06194 { -649, -858, -838, -838, -838},
06195 { -289, -498, -478, -478, -478},
06196 { -739, -948, -928, -928, -928},
06197 { -379, -588, -568, -568, -568}},
06198 /* GU.CG..GU */
06199 {{ DEF, -259, -239, -239, -239},
06200 { -649, -858, -838, -838, -838},
06201 { -289, -498, -478, -478, -478},
06202 { -739, -948, -928, -928, -928},
06203 { -379, -588, -568, -568, -568}},
06204 /* GU.CU..GU */
06205 {{ DEF, -259, -239, -239, -239},
06206 { -649, -858, -838, -838, -838},
06207 { -289, -498, -478, -478, -478},
06208 { -739, -948, -928, -928, -928},
06209 { -379, -588, -568, -568, -568}},
06210 /* GU.G@..GU */
06211 {{{ DEF, -339, -689, -689, -689},
06212 { -100, -389, -739, -739, -739},
06213 { -100, -389, -739, -739, -739},
06214 { -100, -389, -739, -739, -739},
06215 { -100, -389, -739, -739, -739}},
06216 /* GU.GA..GU */
06217 {{ DEF, -339, -689, -689, -689},
06218 { -479, -768, -1118, -1118, -1118},
06219 { -309, -598, -948, -948, -948},
06220 { -389, -678, -1028, -1028, -1028},
06221 { -379, -668, -1018, -1018, -1018}},
06222 /* GU.GC..GU */
06223 {{ DEF, -339, -689, -689, -689},
06224 { -649, -938, -1288, -1288, -1288},
06225 { -289, -578, -928, -928, -928},
06226 { -739, -1028, -1378, -1378, -1378},
06227 { -379, -668, -1018, -1018, -1018}},
06228 /* GU.GG..GU */
06229 {{ DEF, -339, -689, -689, -689},
06230 { -649, -938, -1288, -1288, -1288},
06231 { -289, -578, -928, -928, -928},
06232 { -739, -1028, -1378, -1378, -1378},
06233 { -379, -668, -1018, -1018, -1018}},
06234 /* GU.GU..GU */
06235 {{ DEF, -339, -689, -689, -689},
06236 { -649, -938, -1288, -1288, -1288},
06237 { -289, -578, -928, -928, -928},
06238 { -739, -1028, -1378, -1378, -1378},
06239 { -379, -668, -1018, -1018, -1018}},
06240 /* GU.U@..GU */
06241 {{{ DEF, -329, -329, -329, -329},
06242 { -100, -379, -379, -379, -379},
06243 { -100, -379, -379, -379, -379},
06244 { -100, -379, -379, -379, -379},
06245 { -100, -379, -379, -379, -379}},
06246 /* GU.UA..GU */
06247 {{ DEF, -329, -329, -329, -329},
06248 { -479, -758, -758, -758, -758},
06249 { -309, -588, -588, -588, -588},
06250 { -389, -668, -668, -668, -668},
06251 { -379, -658, -658, -658, -658}},
06252 /* GU.UC..GU */
06253 {{ DEF, -329, -329, -329, -329},
06254 { -649, -928, -928, -928, -928},
06255 { -289, -568, -568, -568, -568},
06256 { -739, -1018, -1018, -1018, -1018},
06257 { -379, -658, -658, -658, -658}},
06258 /* GU.UG..GU */
06259 {{ DEF, -329, -329, -329, -329},
06260 { -649, -928, -928, -928, -928},
06261 { -289, -568, -568, -568, -568},
06262 { -739, -1018, -1018, -1018, -1018},
06263 { -379, -658, -658, -658, -658}},
06264 /* GU.UU..GU */
06265 {{ DEF, -329, -329, -329, -329},
06266 { -649, -928, -928, -928, -928},
06267 { -289, -568, -568, -568, -568},
06268 { -739, -1018, -1018, -1018, -1018},
06269 { -379, -658, -658, -658, -658}}}},
06270 /* GU.@@..UG */
```

```

06271 {{{ 0, 0, 0, 0, 0},
06272 { DEF, DEF, DEF, DEF, DEF},
06273 { DEF, DEF, DEF, DEF, DEF},
06274 { DEF, DEF, DEF, DEF, DEF},
06275 { DEF, DEF, DEF, DEF, DEF}},
06276 /* GU.@A..UG */
06277 {{ 0, 0, 0, 0, 0},
06278 { -719, -719, -719, -719, -719},
06279 { -479, -479, -479, -479, -479},
06280 { -659, -659, -659, -659, -659},
06281 { -549, -549, -549, -549, -549}},
06282 /* GU.@C..UG */
06283 {{ 0, 0, 0, 0, 0},
06284 { -789, -789, -789, -789, -789},
06285 { -479, -479, -479, -479, -479},
06286 { -809, -809, -809, -809, -809},
06287 { -439, -439, -439, -439, -439}},
06288 /* GU.@G..UG */
06289 {{ 0, 0, 0, 0, 0},
06290 { -959, -959, -959, -959, -959},
06291 { -359, -359, -359, -359, -359},
06292 { -919, -919, -919, -919, -919},
06293 { -549, -549, -549, -549, -549}},
06294 /* GU.@U..UG */
06295 {{ 0, 0, 0, 0, 0},
06296 { -809, -809, -809, -809, -809},
06297 { -479, -479, -479, -479, -479},
06298 { -809, -809, -809, -809, -809},
06299 { -359, -359, -359, -359, -359}},
06300 /* GU.@..UG */
06301 {{{ DEF, -429, -599, -599, -599},
06302 { -100, -479, -649, -649, -649},
06303 { -100, -479, -649, -649, -649},
06304 { -100, -479, -649, -649, -649},
06305 { -100, -479, -649, -649, -649}},
06306 /* GU.AA..UG */
06307 {{{ DEF, -429, -599, -599, -599},
06308 { -769, -1148, -1318, -1318, -1318},
06309 { -529, -908, -1078, -1078, -1078},
06310 { -709, -1088, -1258, -1258, -1258},
06311 { -599, -978, -1148, -1148, -1148}},
06312 /* GU.AC..UG */
06313 {{{ DEF, -429, -599, -599, -599},
06314 { -839, -1218, -1388, -1388, -1388},
06315 { -529, -908, -1078, -1078, -1078},
06316 { -859, -1238, -1408, -1408, -1408},
06317 { -489, -868, -1038, -1038, -1038}},
06318 /* GU.AG..UG */
06319 {{{ DEF, -429, -599, -599, -599},
06320 { -1009, -1388, -1558, -1558, -1558},
06321 { -409, -788, -958, -958, -958},
06322 { -969, -1348, -1518, -1518, -1518},
06323 { -599, -978, -1148, -1148, -1148}},
06324 /* GU.AU..UG */
06325 {{{ DEF, -429, -599, -599, -599},
06326 { -859, -1238, -1408, -1408, -1408},
06327 { -529, -908, -1078, -1078, -1078},
06328 { -859, -1238, -1408, -1408, -1408},
06329 { -409, -788, -958, -958, -958}},
06330 /* GU.C@..UG */
06331 {{{ DEF, -259, -239, -239, -239},
06332 { -100, -309, -289, -289, -289},
06333 { -100, -309, -289, -289, -289},
06334 { -100, -309, -289, -289, -289},
06335 { -100, -309, -289, -289, -289}},
06336 /* GU.CA..UG */
06337 {{{ DEF, -259, -239, -239, -239},
06338 { -769, -978, -958, -958, -958},
06339 { -529, -738, -718, -718, -718},
06340 { -709, -918, -898, -898, -898},
06341 { -599, -808, -788, -788, -788}},
06342 /* GU.CC..UG */
06343 {{{ DEF, -259, -239, -239, -239},
06344 { -839, -1048, -1028, -1028, -1028},
06345 { -529, -738, -718, -718, -718},
06346 { -859, -1068, -1048, -1048, -1048},
06347 { -489, -698, -678, -678, -678}},
06348 /* GU.CG..UG */
06349 {{{ DEF, -259, -239, -239, -239},
06350 { -1009, -1218, -1198, -1198, -1198},
06351 { -409, -618, -598, -598, -598},
06352 { -969, -1178, -1158, -1158, -1158},
06353 { -599, -808, -788, -788, -788}},
06354 /* GU.CU..UG */
06355 {{{ DEF, -259, -239, -239, -239},
06356 { -859, -1068, -1048, -1048, -1048},
06357 { -529, -738, -718, -718, -718},

```

```

06358 { -859,-1068,-1048,-1048,-1048},
06359 { -409, -618, -598, -598, -598}}},
06360 /* GU.G@..UG */
06361 {{ { DEF, -339, -689, -689, -689},
06362 { -100, -389, -739, -739, -739},
06363 { -100, -389, -739, -739, -739},
06364 { -100, -389, -739, -739, -739},
06365 { -100, -389, -739, -739, -739}},
06366 /* GU.GA..UG */
06367 {{ DEF, -339, -689, -689, -689},
06368 { -769,-1058,-1408,-1408,-1408},
06369 { -529, -818,-1168,-1168,-1168},
06370 { -709, -998,-1348,-1348,-1348},
06371 { -599, -888,-1238,-1238,-1238}},
06372 /* GU.GC..UG */
06373 {{ DEF, -339, -689, -689, -689},
06374 { -839,-1128,-1478,-1478,-1478},
06375 { -529, -818,-1168,-1168,-1168},
06376 { -859,-1148,-1498,-1498,-1498},
06377 { -489, -778,-1128,-1128,-1128}},
06378 /* GU.GG..UG */
06379 {{ DEF, -339, -689, -689, -689},
06380 {-1009,-1298,-1648,-1648,-1648},
06381 { -409, -698,-1048,-1048,-1048},
06382 { -969,-1258,-1608,-1608,-1608},
06383 { -599, -888,-1238,-1238,-1238}},
06384 /* GU.GU..UG */
06385 {{ DEF, -339, -689, -689, -689},
06386 { -859,-1148,-1498,-1498,-1498},
06387 { -529, -818,-1168,-1168,-1168},
06388 { -859,-1148,-1498,-1498,-1498},
06389 { -409, -698,-1048,-1048,-1048}}},
06390 /* GU.U@..UG */
06391 {{ { DEF, -329, -329, -329, -329},
06392 { -100, -379, -379, -379, -379},
06393 { -100, -379, -379, -379, -379},
06394 { -100, -379, -379, -379, -379},
06395 { -100, -379, -379, -379, -379}},
06396 /* GU.UA..UG */
06397 {{ DEF, -329, -329, -329, -329},
06398 { -769,-1048,-1048,-1048,-1048},
06399 { -529, -808, -808, -808, -808},
06400 { -709, -988, -988, -988, -988},
06401 { -599, -878, -878, -878, -878}},
06402 /* GU.UC..UG */
06403 {{ DEF, -329, -329, -329, -329},
06404 { -839,-1118,-1118,-1118,-1118},
06405 { -529, -808, -808, -808, -808},
06406 { -859,-1138,-1138,-1138,-1138},
06407 { -489, -768, -768, -768, -768}},
06408 /* GU.UG..UG */
06409 {{ DEF, -329, -329, -329, -329},
06410 {-1009,-1288,-1288,-1288,-1288},
06411 { -409, -688, -688, -688, -688},
06412 { -969,-1248,-1248,-1248,-1248},
06413 { -599, -878, -878, -878, -878}},
06414 /* GU.UU..UG */
06415 {{ DEF, -329, -329, -329, -329},
06416 { -859,-1138,-1138,-1138,-1138},
06417 { -529, -808, -808, -808, -808},
06418 { -859,-1138,-1138,-1138,-1138},
06419 { -409, -688, -688, -688, -688}}}},
06420 /* GU.@@..AU */
06421 {{{ 0, 0, 0, 0, 0},
06422 { DEF, DEF, DEF, DEF, DEF},
06423 { DEF, DEF, DEF, DEF, DEF},
06424 { DEF, DEF, DEF, DEF, DEF},
06425 { DEF, DEF, DEF, DEF, DEF}},
06426 /* GU.@A..AU */
06427 {{ 0, 0, 0, 0, 0},
06428 { -429, -429, -429, -429, -429},
06429 { -259, -259, -259, -259, -259},
06430 { -339, -339, -339, -339, -339},
06431 { -329, -329, -329, -329, -329}},
06432 /* GU.@C..AU */
06433 {{ 0, 0, 0, 0, 0},
06434 { -599, -599, -599, -599, -599},
06435 { -239, -239, -239, -239, -239},
06436 { -689, -689, -689, -689, -689},
06437 { -329, -329, -329, -329, -329}},
06438 /* GU.@G..AU */
06439 {{ 0, 0, 0, 0, 0},
06440 { -599, -599, -599, -599, -599},
06441 { -239, -239, -239, -239, -239},
06442 { -689, -689, -689, -689, -689},
06443 { -329, -329, -329, -329, -329}},
06444 /* GU.@U..AU */

```

```

06445 {{ 0, 0, 0, 0, 0},
06446 { -599, -599, -599, -599, -599},
06447 { -239, -239, -239, -239, -239},
06448 { -689, -689, -689, -689, -689},
06449 { -329, -329, -329, -329, -329}},
06450 /* GU.A@.AU */
06451 {{{ DEF, -429, -599, -599, -599},
06452 { -100, -479, -649, -649, -649},
06453 { -100, -479, -649, -649, -649},
06454 { -100, -479, -649, -649, -649},
06455 { -100, -479, -649, -649, -649}},
06456 /* GU.AA.AU */
06457 {{ DEF, -429, -599, -599, -599},
06458 { -479, -858, -1028, -1028, -1028},
06459 { -309, -688, -858, -858, -858},
06460 { -389, -768, -938, -938, -938},
06461 { -379, -758, -928, -928, -928}},
06462 /* GU.AC.AU */
06463 {{ DEF, -429, -599, -599, -599},
06464 { -649, -1028, -1198, -1198, -1198},
06465 { -289, -668, -838, -838, -838},
06466 { -739, -1118, -1288, -1288, -1288},
06467 { -379, -758, -928, -928, -928}},
06468 /* GU.AG.AU */
06469 {{ DEF, -429, -599, -599, -599},
06470 { -649, -1028, -1198, -1198, -1198},
06471 { -289, -668, -838, -838, -838},
06472 { -739, -1118, -1288, -1288, -1288},
06473 { -379, -758, -928, -928, -928}},
06474 /* GU.AU.AU */
06475 {{ DEF, -429, -599, -599, -599},
06476 { -649, -1028, -1198, -1198, -1198},
06477 { -289, -668, -838, -838, -838},
06478 { -739, -1118, -1288, -1288, -1288},
06479 { -379, -758, -928, -928, -928}},
06480 /* GU.C@.AU */
06481 {{{ DEF, -259, -239, -239, -239},
06482 { -100, -309, -289, -289, -289},
06483 { -100, -309, -289, -289, -289},
06484 { -100, -309, -289, -289, -289},
06485 { -100, -309, -289, -289, -289}},
06486 /* GU.CA.AU */
06487 {{ DEF, -259, -239, -239, -239},
06488 { -479, -688, -668, -668, -668},
06489 { -309, -518, -498, -498, -498},
06490 { -389, -598, -578, -578, -578},
06491 { -379, -588, -568, -568, -568}},
06492 /* GU.CC.AU */
06493 {{ DEF, -259, -239, -239, -239},
06494 { -649, -858, -838, -838, -838},
06495 { -289, -498, -478, -478, -478},
06496 { -739, -948, -928, -928, -928},
06497 { -379, -588, -568, -568, -568}},
06498 /* GU.CG.AU */
06499 {{ DEF, -259, -239, -239, -239},
06500 { -649, -858, -838, -838, -838},
06501 { -289, -498, -478, -478, -478},
06502 { -739, -948, -928, -928, -928},
06503 { -379, -588, -568, -568, -568}},
06504 /* GU.CU.AU */
06505 {{ DEF, -259, -239, -239, -239},
06506 { -649, -858, -838, -838, -838},
06507 { -289, -498, -478, -478, -478},
06508 { -739, -948, -928, -928, -928},
06509 { -379, -588, -568, -568, -568}},
06510 /* GU.G@.AU */
06511 {{{ DEF, -339, -689, -689, -689},
06512 { -100, -389, -739, -739, -739},
06513 { -100, -389, -739, -739, -739},
06514 { -100, -389, -739, -739, -739},
06515 { -100, -389, -739, -739, -739}},
06516 /* GU.GA.AU */
06517 {{ DEF, -339, -689, -689, -689},
06518 { -479, -768, -1118, -1118, -1118},
06519 { -309, -598, -948, -948, -948},
06520 { -389, -678, -1028, -1028, -1028},
06521 { -379, -668, -1018, -1018, -1018}},
06522 /* GU.GC.AU */
06523 {{ DEF, -339, -689, -689, -689},
06524 { -649, -938, -1288, -1288, -1288},
06525 { -289, -578, -928, -928, -928},
06526 { -739, -1028, -1378, -1378, -1378},
06527 { -379, -668, -1018, -1018, -1018}},
06528 /* GU.GG.AU */
06529 {{ DEF, -339, -689, -689, -689},
06530 { -649, -938, -1288, -1288, -1288},
06531 { -289, -578, -928, -928, -928},

```



```

06532 { -739,-1028,-1378,-1378,-1378},
06533 { -379, -668,-1018,-1018,-1018}},
06534 /* GU.GU..AU */
06535 {{ DEF, -339, -689, -689, -689},
06536 { -649, -938,-1288,-1288,-1288},
06537 { -289, -578, -928, -928, -928},
06538 { -739,-1028,-1378,-1378,-1378},
06539 { -379, -668,-1018,-1018,-1018}}},
06540 /* GU.U@..AU */
06541 {{{ DEF, -329, -329, -329, -329},
06542 { -100, -379, -379, -379, -379},
06543 { -100, -379, -379, -379, -379},
06544 { -100, -379, -379, -379, -379},
06545 { -100, -379, -379, -379, -379}}},
06546 /* GU.UA..AU */
06547 {{ DEF, -329, -329, -329, -329},
06548 { -479, -758, -758, -758, -758},
06549 { -309, -588, -588, -588, -588},
06550 { -389, -668, -668, -668, -668},
06551 { -379, -658, -658, -658, -658}}},
06552 /* GU.UC..AU */
06553 {{ DEF, -329, -329, -329, -329},
06554 { -649, -928, -928, -928, -928},
06555 { -289, -568, -568, -568, -568},
06556 { -739,-1018,-1018,-1018,-1018},
06557 { -379, -658, -658, -658, -658}}},
06558 /* GU.UG..AU */
06559 {{ DEF, -329, -329, -329, -329},
06560 { -649, -928, -928, -928, -928},
06561 { -289, -568, -568, -568, -568},
06562 { -739,-1018,-1018,-1018,-1018},
06563 { -379, -658, -658, -658, -658}}},
06564 /* GU.UU..AU */
06565 {{ DEF, -329, -329, -329, -329},
06566 { -649, -928, -928, -928, -928},
06567 { -289, -568, -568, -568, -568},
06568 { -739,-1018,-1018,-1018,-1018},
06569 { -379, -658, -658, -658, -658}}}},
06570 /* GU.@@..UA */
06571 {{{ 0, 0, 0, 0, 0},
06572 { DEF, DEF, DEF, DEF, DEF},
06573 { DEF, DEF, DEF, DEF, DEF},
06574 { DEF, DEF, DEF, DEF, DEF},
06575 { DEF, DEF, DEF, DEF, DEF}}},
06576 /* GU.@A..UA */
06577 {{ 0, 0, 0, 0, 0},
06578 { -399, -399, -399, -399, -399},
06579 { -429, -429, -429, -429, -429},
06580 { -379, -379, -379, -379, -379},
06581 { -279, -279, -279, -279, -279}}},
06582 /* GU.@C..UA */
06583 {{ 0, 0, 0, 0, 0},
06584 { -629, -629, -629, -629, -629},
06585 { -509, -509, -509, -509, -509},
06586 { -679, -679, -679, -679, -679},
06587 { -139, -139, -139, -139, -139}}},
06588 /* GU.@G..UA */
06589 {{ 0, 0, 0, 0, 0},
06590 { -889, -889, -889, -889, -889},
06591 { -199, -199, -199, -199, -199},
06592 { -889, -889, -889, -889, -889},
06593 { -279, -279, -279, -279, -279}}},
06594 /* GU.@U..UA */
06595 {{ 0, 0, 0, 0, 0},
06596 { -589, -589, -589, -589, -589},
06597 { -179, -179, -179, -179, -179},
06598 { -679, -679, -679, -679, -679},
06599 { -140, -140, -140, -140, -140}}}},
06600 /* GU.A@..UA */
06601 {{{ DEF, -429, -599, -599, -599},
06602 { -100, -479, -649, -649, -649},
06603 { -100, -479, -649, -649, -649},
06604 { -100, -479, -649, -649, -649},
06605 { -100, -479, -649, -649, -649}}},
06606 /* GU.AA..UA */
06607 {{ DEF, -429, -599, -599, -599},
06608 { -449, -828, -998, -998, -998},
06609 { -479, -858,-1028,-1028,-1028},
06610 { -429, -808, -978, -978, -978},
06611 { -329, -708, -878, -878, -878}}},
06612 /* GU.AC..UA */
06613 {{ DEF, -429, -599, -599, -599},
06614 { -679,-1058,-1228,-1228,-1228},
06615 { -559, -938,-1108,-1108,-1108},
06616 { -729,-1108,-1278,-1278,-1278},
06617 { -189, -568, -738, -738, -738}}},
06618 /* GU.AG..UA */

```

```
06619 {{ DEF, -429, -599, -599, -599},
06620 { -939,-1318,-1488,-1488,-1488},
06621 { -249, -628, -798, -798, -798},
06622 { -939,-1318,-1488,-1488,-1488},
06623 { -329, -708, -878, -878, -878}},
06624 /* GU.AU..UA */
06625 {{ DEF, -429, -599, -599, -599},
06626 { -639,-1018,-1188,-1188,-1188},
06627 { -229, -608, -778, -778, -778},
06628 { -729,-1108,-1278,-1278,-1278},
06629 { -190, -569, -739, -739, -739}},
06630 /* GU.C@..UA */
06631 {{{ DEF, -259, -239, -239, -239},
06632 { -100, -309, -289, -289, -289},
06633 { -100, -309, -289, -289, -289},
06634 { -100, -309, -289, -289, -289},
06635 { -100, -309, -289, -289, -289}},
06636 /* GU.CA..UA */
06637 {{ DEF, -259, -239, -239, -239},
06638 { -449, -658, -638, -638, -638},
06639 { -479, -688, -668, -668, -668},
06640 { -429, -638, -618, -618, -618},
06641 { -329, -538, -518, -518, -518}},
06642 /* GU.CC..UA */
06643 {{ DEF, -259, -239, -239, -239},
06644 { -679, -888, -868, -868, -868},
06645 { -559, -768, -748, -748, -748},
06646 { -729, -938, -918, -918, -918},
06647 { -189, -398, -378, -378, -378}},
06648 /* GU.CG..UA */
06649 {{ DEF, -259, -239, -239, -239},
06650 { -939,-1148,-1128,-1128,-1128},
06651 { -249, -458, -438, -438, -438},
06652 { -939,-1148,-1128,-1128,-1128},
06653 { -329, -538, -518, -518, -518}},
06654 /* GU.CU..UA */
06655 {{ DEF, -259, -239, -239, -239},
06656 { -639, -848, -828, -828, -828},
06657 { -229, -438, -418, -418, -418},
06658 { -729, -938, -918, -918, -918},
06659 { -190, -399, -379, -379, -379}},
06660 /* GU.G@..UA */
06661 {{{ DEF, -339, -689, -689, -689},
06662 { -100, -389, -739, -739, -739},
06663 { -100, -389, -739, -739, -739},
06664 { -100, -389, -739, -739, -739},
06665 { -100, -389, -739, -739, -739}},
06666 /* GU.GA..UA */
06667 {{ DEF, -339, -689, -689, -689},
06668 { -449, -738,-1088,-1088,-1088},
06669 { -479, -768,-1118,-1118,-1118},
06670 { -429, -718,-1068,-1068,-1068},
06671 { -329, -618, -968, -968, -968}},
06672 /* GU.GC..UA */
06673 {{ DEF, -339, -689, -689, -689},
06674 { -679, -968,-1318,-1318,-1318},
06675 { -559, -848,-1198,-1198,-1198},
06676 { -729,-1018,-1368,-1368,-1368},
06677 { -189, -478, -828, -828, -828}},
06678 /* GU.GG..UA */
06679 {{ DEF, -339, -689, -689, -689},
06680 { -939,-1228,-1578,-1578,-1578},
06681 { -249, -538, -888, -888, -888},
06682 { -939,-1228,-1578,-1578,-1578},
06683 { -329, -618, -968, -968, -968}},
06684 /* GU.GU..UA */
06685 {{ DEF, -339, -689, -689, -689},
06686 { -639, -928,-1278,-1278,-1278},
06687 { -229, -518, -868, -868, -868},
06688 { -729,-1018,-1368,-1368,-1368},
06689 { -190, -479, -829, -829, -829}},
06690 /* GU.U@..UA */
06691 {{{ DEF, -329, -329, -329, -329},
06692 { -100, -379, -379, -379, -379},
06693 { -100, -379, -379, -379, -379},
06694 { -100, -379, -379, -379, -379},
06695 { -100, -379, -379, -379, -379}},
06696 /* GU.UA..UA */
06697 {{ DEF, -329, -329, -329, -329},
06698 { -449, -728, -728, -728, -728},
06699 { -479, -758, -758, -758, -758},
06700 { -429, -708, -708, -708, -708},
06701 { -329, -608, -608, -608, -608}},
06702 /* GU.UC..UA */
06703 {{ DEF, -329, -329, -329, -329},
06704 { -679, -958, -958, -958, -958},
06705 { -559, -838, -838, -838, -838},
```

```
06706 { -729,-1008,-1008,-1008,-1008},
06707 { -189, -468, -468, -468, -468}},
06708 /* GU.UG..UA */
06709 {{ DEF, -329, -329, -329, -329},
06710 { -939,-1218,-1218,-1218,-1218},
06711 { -249, -528, -528, -528, -528},
06712 { -939,-1218,-1218,-1218,-1218},
06713 { -329, -608, -608, -608, -608}},
06714 /* GU.UU..UA */
06715 {{ DEF, -329, -329, -329, -329},
06716 { -639, -918, -918, -918, -918},
06717 { -229, -508, -508, -508, -508},
06718 { -729,-1008,-1008,-1008,-1008},
06719 { -190, -469, -469, -469, -469}}},
06720 /* GU.@@.. @ */
06721 {{{ DEF, DEF, DEF, DEF, DEF},
06722 { DEF, DEF, DEF, DEF, DEF},
06723 { DEF, DEF, DEF, DEF, DEF},
06724 { DEF, DEF, DEF, DEF, DEF},
06725 { DEF, DEF, DEF, DEF, DEF}},
06726 /* GU.AA.. @ */
06727 {{ DEF, DEF, DEF, DEF, DEF},
06728 { DEF, DEF, DEF, DEF, DEF},
06729 { DEF, DEF, DEF, DEF, DEF},
06730 { DEF, DEF, DEF, DEF, DEF},
06731 { DEF, DEF, DEF, DEF, DEF}},
06732 /* GU.CC.. @ */
06733 {{ DEF, DEF, DEF, DEF, DEF},
06734 { DEF, DEF, DEF, DEF, DEF},
06735 { DEF, DEF, DEF, DEF, DEF},
06736 { DEF, DEF, DEF, DEF, DEF},
06737 { DEF, DEF, DEF, DEF, DEF}},
06738 /* GU.GG.. @ */
06739 {{ DEF, DEF, DEF, DEF, DEF},
06740 { DEF, DEF, DEF, DEF, DEF},
06741 { DEF, DEF, DEF, DEF, DEF},
06742 { DEF, DEF, DEF, DEF, DEF},
06743 { DEF, DEF, DEF, DEF, DEF}},
06744 /* GU.UU.. @ */
06745 {{ DEF, DEF, DEF, DEF, DEF},
06746 { DEF, DEF, DEF, DEF, DEF},
06747 { DEF, DEF, DEF, DEF, DEF},
06748 { DEF, DEF, DEF, DEF, DEF},
06749 { DEF, DEF, DEF, DEF, DEF}},
06750 /* GU.AA.. @ */
06751 {{{ -100, -479, -649, -649, -649},
06752 { -100, -479, -649, -649, -649},
06753 { -100, -479, -649, -649, -649},
06754 { -100, -479, -649, -649, -649},
06755 { -100, -479, -649, -649, -649}},
06756 /* GU.AA.. @ */
06757 {{ -100, -479, -649, -649, -649},
06758 { -100, -479, -649, -649, -649},
06759 { -100, -479, -649, -649, -649},
06760 { -100, -479, -649, -649, -649},
06761 { -100, -479, -649, -649, -649}},
06762 /* GU.AC.. @ */
06763 {{{ -100, -479, -649, -649, -649},
06764 { -100, -479, -649, -649, -649},
06765 { -100, -479, -649, -649, -649},
06766 { -100, -479, -649, -649, -649},
06767 { -100, -479, -649, -649, -649}},
06768 /* GU.AG.. @ */
06769 {{{ -100, -479, -649, -649, -649},
06770 { -100, -479, -649, -649, -649},
06771 { -100, -479, -649, -649, -649},
06772 { -100, -479, -649, -649, -649},
06773 { -100, -479, -649, -649, -649}},
06774 /* GU.AU.. @ */
06775 {{{ -100, -479, -649, -649, -649},
06776 { -100, -479, -649, -649, -649},
06777 { -100, -479, -649, -649, -649},
06778 { -100, -479, -649, -649, -649},
06779 { -100, -479, -649, -649, -649}},
06780 /* GU.CC.. @ */
06781 {{{ -100, -309, -289, -289, -289},
06782 { -100, -309, -289, -289, -289},
06783 { -100, -309, -289, -289, -289},
06784 { -100, -309, -289, -289, -289},
06785 { -100, -309, -289, -289, -289}},
06786 /* GU.CA.. @ */
06787 {{{ -100, -309, -289, -289, -289},
06788 { -100, -309, -289, -289, -289},
06789 { -100, -309, -289, -289, -289},
06790 { -100, -309, -289, -289, -289},
06791 { -100, -309, -289, -289, -289}},
06792 /* GU.CC.. @ */
```

```

06793 {{ -100, -309, -289, -289, -289},
06794 { -100, -309, -289, -289, -289},
06795 { -100, -309, -289, -289, -289},
06796 { -100, -309, -289, -289, -289},
06797 { -100, -309, -289, -289, -289}},
06798 /* GU.CG.. @ */
06799 {{ -100, -309, -289, -289, -289},
06800 { -100, -309, -289, -289, -289},
06801 { -100, -309, -289, -289, -289},
06802 { -100, -309, -289, -289, -289},
06803 { -100, -309, -289, -289, -289}},
06804 /* GU.CU.. @ */
06805 {{ -100, -309, -289, -289, -289},
06806 { -100, -309, -289, -289, -289},
06807 { -100, -309, -289, -289, -289},
06808 { -100, -309, -289, -289, -289},
06809 { -100, -309, -289, -289, -289}}},
06810 /* GU.G@.. @ */
06811 {{{ -100, -389, -739, -739, -739},
06812 { -100, -389, -739, -739, -739},
06813 { -100, -389, -739, -739, -739},
06814 { -100, -389, -739, -739, -739},
06815 { -100, -389, -739, -739, -739}},
06816 /* GU.GA.. @ */
06817 {{ -100, -389, -739, -739, -739},
06818 { -100, -389, -739, -739, -739},
06819 { -100, -389, -739, -739, -739},
06820 { -100, -389, -739, -739, -739},
06821 { -100, -389, -739, -739, -739}},
06822 /* GU.GC.. @ */
06823 {{ -100, -389, -739, -739, -739},
06824 { -100, -389, -739, -739, -739},
06825 { -100, -389, -739, -739, -739},
06826 { -100, -389, -739, -739, -739},
06827 { -100, -389, -739, -739, -739}},
06828 /* GU.GG.. @ */
06829 {{ -100, -389, -739, -739, -739},
06830 { -100, -389, -739, -739, -739},
06831 { -100, -389, -739, -739, -739},
06832 { -100, -389, -739, -739, -739},
06833 { -100, -389, -739, -739, -739}},
06834 /* GU.GU.. @ */
06835 {{ -100, -389, -739, -739, -739},
06836 { -100, -389, -739, -739, -739},
06837 { -100, -389, -739, -739, -739},
06838 { -100, -389, -739, -739, -739},
06839 { -100, -389, -739, -739, -739}}},
06840 /* GU.U@.. @ */
06841 {{{ -100, -379, -379, -379, -379},
06842 { -100, -379, -379, -379, -379},
06843 { -100, -379, -379, -379, -379},
06844 { -100, -379, -379, -379, -379},
06845 { -100, -379, -379, -379, -379}},
06846 /* GU.UA.. @ */
06847 {{ -100, -379, -379, -379, -379},
06848 { -100, -379, -379, -379, -379},
06849 { -100, -379, -379, -379, -379},
06850 { -100, -379, -379, -379, -379},
06851 { -100, -379, -379, -379, -379}},
06852 /* GU.UC.. @ */
06853 {{ -100, -379, -379, -379, -379},
06854 { -100, -379, -379, -379, -379},
06855 { -100, -379, -379, -379, -379},
06856 { -100, -379, -379, -379, -379},
06857 { -100, -379, -379, -379, -379}},
06858 /* GU.UG.. @ */
06859 {{ -100, -379, -379, -379, -379},
06860 { -100, -379, -379, -379, -379},
06861 { -100, -379, -379, -379, -379},
06862 { -100, -379, -379, -379, -379},
06863 { -100, -379, -379, -379, -379}},
06864 /* GU.UU.. @ */
06865 {{ -100, -379, -379, -379, -379},
06866 { -100, -379, -379, -379, -379},
06867 { -100, -379, -379, -379, -379},
06868 { -100, -379, -379, -379, -379},
06869 { -100, -379, -379, -379, -379}}}},
06870 { /* noPair */ {{{{0}}}},
06871 /* UG.@A..CG */
06872 {{{ 0, 0, 0, 0, 0},
06873 { DEF, DEF, DEF, DEF, DEF},
06874 { DEF, DEF, DEF, DEF, DEF},
06875 { DEF, DEF, DEF, DEF, DEF},
06876 { DEF, DEF, DEF, DEF, DEF}},
06877 /* UG.@A..CG */
06878 {{ 0, 0, 0, 0, 0},
06879 {-1029,-1029,-1029,-1029,-1029},

```

```

06880 { -519, -519, -519, -519, -519},
06881 { -939, -939, -939, -939, -939},
06882 { -809, -809, -809, -809, -809}},
06883 /* UG.@C..CG */
06884 {{ 0, 0, 0, 0, 0},
06885 { -949, -949, -949, -949, -949},
06886 { -449, -449, -449, -449, -449},
06887 { -939, -939, -939, -939, -939},
06888 { -739, -739, -739, -739, -739}},
06889 /* UG.@G..CG */
06890 {{ 0, 0, 0, 0, 0},
06891 {-1029, -1029, -1029, -1029, -1029},
06892 { -519, -519, -519, -519, -519},
06893 { -939, -939, -939, -939, -939},
06894 { -809, -809, -809, -809, -809}},
06895 /* UG.@U..CG */
06896 {{ 0, 0, 0, 0, 0},
06897 {-1029, -1029, -1029, -1029, -1029},
06898 { -669, -669, -669, -669, -669},
06899 { -939, -939, -939, -939, -939},
06900 { -859, -859, -859, -859, -859}}},
06901 /* UG.A@..CG */
06902 {{{ DEF, -719, -789, -959, -809},
06903 { -100, -769, -839, -1009, -859},
06904 { -100, -769, -839, -1009, -859},
06905 { -100, -769, -839, -1009, -859},
06906 { -100, -769, -839, -1009, -859}}},
06907 /* UG.AA..CG */
06908 {{{ DEF, -719, -789, -959, -809},
06909 {-1079, -1748, -1818, -1988, -1838},
06910 { -569, -1238, -1308, -1478, -1328},
06911 { -989, -1658, -1728, -1898, -1748},
06912 { -859, -1528, -1598, -1768, -1618}}},
06913 /* UG.AC..CG */
06914 {{{ DEF, -719, -789, -959, -809},
06915 { -999, -1668, -1738, -1908, -1758},
06916 { -499, -1168, -1238, -1408, -1258},
06917 { -989, -1658, -1728, -1898, -1748},
06918 { -789, -1458, -1528, -1698, -1548}}},
06919 /* UG.AG..CG */
06920 {{{ DEF, -719, -789, -959, -809},
06921 {-1079, -1748, -1818, -1988, -1838},
06922 { -569, -1238, -1308, -1478, -1328},
06923 { -989, -1658, -1728, -1898, -1748},
06924 { -859, -1528, -1598, -1768, -1618}}},
06925 /* UG.AU..CG */
06926 {{{ DEF, -719, -789, -959, -809},
06927 {-1079, -1748, -1818, -1988, -1838},
06928 { -719, -1388, -1458, -1628, -1478},
06929 { -989, -1658, -1728, -1898, -1748},
06930 { -909, -1578, -1648, -1818, -1668}}},
06931 /* UG.C@..CG */
06932 {{{ DEF, -479, -479, -359, -479},
06933 { -100, -529, -529, -409, -529},
06934 { -100, -529, -529, -409, -529},
06935 { -100, -529, -529, -409, -529},
06936 { -100, -529, -529, -409, -529}}},
06937 /* UG.CA..CG */
06938 {{{ DEF, -479, -479, -359, -479},
06939 {-1079, -1508, -1508, -1388, -1508},
06940 { -569, -998, -998, -878, -998},
06941 { -989, -1418, -1418, -1298, -1418},
06942 { -859, -1288, -1288, -1168, -1288}}},
06943 /* UG.CC..CG */
06944 {{{ DEF, -479, -479, -359, -479},
06945 { -999, -1428, -1428, -1308, -1428},
06946 { -499, -928, -928, -808, -928},
06947 { -989, -1418, -1418, -1298, -1418},
06948 { -789, -1218, -1218, -1098, -1218}}},
06949 /* UG.CG..CG */
06950 {{{ DEF, -479, -479, -359, -479},
06951 {-1079, -1508, -1508, -1388, -1508},
06952 { -569, -998, -998, -878, -998},
06953 { -989, -1418, -1418, -1298, -1418},
06954 { -859, -1288, -1288, -1168, -1288}}},
06955 /* UG.CU..CG */
06956 {{{ DEF, -479, -479, -359, -479},
06957 {-1079, -1508, -1508, -1388, -1508},
06958 { -719, -1148, -1148, -1028, -1148},
06959 { -989, -1418, -1418, -1298, -1418},
06960 { -909, -1338, -1338, -1218, -1338}}},
06961 /* UG.G@..CG */
06962 {{{ DEF, -659, -809, -919, -809},
06963 { -100, -709, -859, -969, -859},
06964 { -100, -709, -859, -969, -859},
06965 { -100, -709, -859, -969, -859},
06966 { -100, -709, -859, -969, -859}},

```

```

06967 /* UG.GA..CG */
06968 {{ DEF, -659, -809, -919, -809},
06969 {-1079,-1688,-1838,-1948,-1838},
06970 {-569,-1178,-1328,-1438,-1328},
06971 {-989,-1598,-1748,-1858,-1748},
06972 {-859,-1468,-1618,-1728,-1618}},
06973 /* UG.GC..CG */
06974 {{ DEF, -659, -809, -919, -809},
06975 {-999,-1608,-1758,-1868,-1758},
06976 {-499,-1108,-1258,-1368,-1258},
06977 {-989,-1598,-1748,-1858,-1748},
06978 {-789,-1398,-1548,-1658,-1548}},
06979 /* UG.GG..CG */
06980 {{ DEF, -659, -809, -919, -809},
06981 {-1079,-1688,-1838,-1948,-1838},
06982 {-569,-1178,-1328,-1438,-1328},
06983 {-989,-1598,-1748,-1858,-1748},
06984 {-859,-1468,-1618,-1728,-1618}},
06985 /* UG.GU..CG */
06986 {{ DEF, -659, -809, -919, -809},
06987 {-1079,-1688,-1838,-1948,-1838},
06988 {-719,-1328,-1478,-1588,-1478},
06989 {-989,-1598,-1748,-1858,-1748},
06990 {-909,-1518,-1668,-1778,-1668}}},
06991 /* UG.U@..CG */
06992 {{{ DEF, -549, -439, -549, -359},
06993 {-100, -599, -489, -599, -409},
06994 {-100, -599, -489, -599, -409},
06995 {-100, -599, -489, -599, -409},
06996 {-100, -599, -489, -599, -409}}},
06997 /* UG.UA..CG */
06998 {{ DEF, -549, -439, -549, -359},
06999 {-1079,-1578,-1468,-1578,-1388},
07000 {-569,-1068, -958,-1068, -878},
07001 {-989,-1488,-1378,-1488,-1298},
07002 {-859,-1358,-1248,-1358,-1168}},
07003 /* UG.UC..CG */
07004 {{ DEF, -549, -439, -549, -359},
07005 {-999,-1498,-1388,-1498,-1308},
07006 {-499, -998, -888, -998, -808},
07007 {-989,-1488,-1378,-1488,-1298},
07008 {-789,-1288,-1178,-1288,-1098}},
07009 /* UG.UG..CG */
07010 {{ DEF, -549, -439, -549, -359},
07011 {-1079,-1578,-1468,-1578,-1388},
07012 {-569,-1068, -958,-1068, -878},
07013 {-989,-1488,-1378,-1488,-1298},
07014 {-859,-1358,-1248,-1358,-1168}},
07015 /* UG.UU..CG */
07016 {{ DEF, -549, -439, -549, -359},
07017 {-1079,-1578,-1468,-1578,-1388},
07018 {-719,-1218,-1108,-1218,-1028},
07019 {-989,-1488,-1378,-1488,-1298},
07020 {-909,-1408,-1298,-1408,-1218}}}},
07021 /* UG.@@..GC */
07022 {{{ 0, 0, 0, 0, 0},
07023 { DEF, DEF, DEF, DEF, DEF},
07024 { DEF, DEF, DEF, DEF, DEF},
07025 { DEF, DEF, DEF, DEF, DEF},
07026 { DEF, DEF, DEF, DEF, DEF}}},
07027 /* UG.@A..GC */
07028 {{ 0, 0, 0, 0, 0},
07029 {-519, -519, -519, -519, -519},
07030 {-719, -719, -719, -719, -719},
07031 {-709, -709, -709, -709, -709},
07032 {-499, -499, -499, -499, -499}},
07033 /* UG.@C..GC */
07034 {{ 0, 0, 0, 0, 0},
07035 {-879, -879, -879, -879, -879},
07036 {-309, -309, -309, -309, -309},
07037 {-739, -739, -739, -739, -739},
07038 {-499, -499, -499, -499, -499}},
07039 /* UG.@G..GC */
07040 {{ 0, 0, 0, 0, 0},
07041 {-559, -559, -559, -559, -559},
07042 {-309, -309, -309, -309, -309},
07043 {-619, -619, -619, -619, -619},
07044 {-499, -499, -499, -499, -499}},
07045 /* UG.@U..GC */
07046 {{{ 0, 0, 0, 0, 0},
07047 {-879, -879, -879, -879, -879},
07048 {-389, -389, -389, -389, -389},
07049 {-739, -739, -739, -739, -739},
07050 {-569, -569, -569, -569, -569}}},
07051 /* UG.A@..GC */
07052 {{{ DEF, -719, -789, -959, -809},
07053 {-100, -769, -839, -1009, -859},

```

```

07054 { -100, -769, -839, -1009, -859},
07055 { -100, -769, -839, -1009, -859},
07056 { -100, -769, -839, -1009, -859}},
07057 /* UG.AA..GC */
07058 {{ DEF, -719, -789, -959, -809},
07059 { -569, -1238, -1308, -1478, -1328},
07060 { -769, -1438, -1508, -1678, -1528},
07061 { -759, -1428, -1498, -1668, -1518},
07062 { -549, -1218, -1288, -1458, -1308}},
07063 /* UG.AC..GC */
07064 {{ DEF, -719, -789, -959, -809},
07065 { -929, -1598, -1668, -1838, -1688},
07066 { -359, -1028, -1098, -1268, -1118},
07067 { -789, -1458, -1528, -1698, -1548},
07068 { -549, -1218, -1288, -1458, -1308}},
07069 /* UG.AG..GC */
07070 {{ DEF, -719, -789, -959, -809},
07071 { -609, -1278, -1348, -1518, -1368},
07072 { -359, -1028, -1098, -1268, -1118},
07073 { -669, -1338, -1408, -1578, -1428},
07074 { -549, -1218, -1288, -1458, -1308}},
07075 /* UG.AU..GC */
07076 {{ DEF, -719, -789, -959, -809},
07077 { -929, -1598, -1668, -1838, -1688},
07078 { -439, -1108, -1178, -1348, -1198},
07079 { -789, -1458, -1528, -1698, -1548},
07080 { -619, -1288, -1358, -1528, -1378}}},
07081 /* UG.C@..GC */
07082 {{{ DEF, -479, -479, -359, -479},
07083 { -100, -529, -529, -409, -529},
07084 { -100, -529, -529, -409, -529},
07085 { -100, -529, -529, -409, -529},
07086 { -100, -529, -529, -409, -529}},
07087 /* UG.CA..GC */
07088 {{ DEF, -479, -479, -359, -479},
07089 { -569, -998, -998, -878, -998},
07090 { -769, -1198, -1198, -1078, -1198},
07091 { -759, -1188, -1188, -1068, -1188},
07092 { -549, -978, -978, -858, -978}},
07093 /* UG.CC..GC */
07094 {{ DEF, -479, -479, -359, -479},
07095 { -929, -1358, -1358, -1238, -1358},
07096 { -359, -788, -788, -668, -788},
07097 { -789, -1218, -1218, -1098, -1218},
07098 { -549, -978, -978, -858, -978}},
07099 /* UG.CG..GC */
07100 {{ DEF, -479, -479, -359, -479},
07101 { -609, -1038, -1038, -918, -1038},
07102 { -359, -788, -788, -668, -788},
07103 { -669, -1098, -1098, -978, -1098},
07104 { -549, -978, -978, -858, -978}},
07105 /* UG.CU..GC */
07106 {{ DEF, -479, -479, -359, -479},
07107 { -929, -1358, -1358, -1238, -1358},
07108 { -439, -868, -868, -748, -868},
07109 { -789, -1218, -1218, -1098, -1218},
07110 { -619, -1048, -1048, -928, -1048}}},
07111 /* UG.G@..GC */
07112 {{{ DEF, -659, -809, -919, -809},
07113 { -100, -709, -859, -969, -859},
07114 { -100, -709, -859, -969, -859},
07115 { -100, -709, -859, -969, -859},
07116 { -100, -709, -859, -969, -859}},
07117 /* UG.GA..GC */
07118 {{ DEF, -659, -809, -919, -809},
07119 { -569, -1178, -1328, -1438, -1328},
07120 { -769, -1378, -1528, -1638, -1528},
07121 { -759, -1368, -1518, -1628, -1518},
07122 { -549, -1158, -1308, -1418, -1308}},
07123 /* UG.GC..GC */
07124 {{ DEF, -659, -809, -919, -809},
07125 { -929, -1538, -1688, -1798, -1688},
07126 { -359, -968, -1118, -1228, -1118},
07127 { -789, -1398, -1548, -1658, -1548},
07128 { -549, -1158, -1308, -1418, -1308}},
07129 /* UG.GG..GC */
07130 {{ DEF, -659, -809, -919, -809},
07131 { -609, -1218, -1368, -1478, -1368},
07132 { -359, -968, -1118, -1228, -1118},
07133 { -669, -1278, -1428, -1538, -1428},
07134 { -549, -1158, -1308, -1418, -1308}},
07135 /* UG.GU..GC */
07136 {{ DEF, -659, -809, -919, -809},
07137 { -929, -1538, -1688, -1798, -1688},
07138 { -439, -1048, -1198, -1308, -1198},
07139 { -789, -1398, -1548, -1658, -1548},
07140 { -619, -1228, -1378, -1488, -1378}},

```

```

07141 /* UG.U@.GC */
07142 {{ DEF, -549, -439, -549, -359},
07143 { -100, -599, -489, -599, -409},
07144 { -100, -599, -489, -599, -409},
07145 { -100, -599, -489, -599, -409},
07146 { -100, -599, -489, -599, -409}},
07147 /* UG.UA.GC */
07148 {{ DEF, -549, -439, -549, -359},
07149 { -569, -1068, -958, -1068, -878},
07150 { -769, -1268, -1158, -1268, -1078},
07151 { -759, -1258, -1148, -1258, -1068},
07152 { -549, -1048, -938, -1048, -858}},
07153 /* UG.UC.GC */
07154 {{ DEF, -549, -439, -549, -359},
07155 { -929, -1428, -1318, -1428, -1238},
07156 { -359, -858, -748, -858, -668},
07157 { -789, -1288, -1178, -1288, -1098},
07158 { -549, -1048, -938, -1048, -858}},
07159 /* UG.UG.GC */
07160 {{ DEF, -549, -439, -549, -359},
07161 { -609, -1108, -998, -1108, -918},
07162 { -359, -858, -748, -858, -668},
07163 { -669, -1168, -1058, -1168, -978},
07164 { -549, -1048, -938, -1048, -858}},
07165 /* UG.UU.GC */
07166 {{ DEF, -549, -439, -549, -359},
07167 { -929, -1428, -1318, -1428, -1238},
07168 { -439, -938, -828, -938, -748},
07169 { -789, -1288, -1178, -1288, -1098},
07170 { -619, -1118, -1008, -1118, -928}}}},
07171 /* UG.@@.GU */
07172 {{{ 0, 0, 0, 0, 0},
07173 { DEF, DEF, DEF, DEF, DEF},
07174 { DEF, DEF, DEF, DEF, DEF},
07175 { DEF, DEF, DEF, DEF, DEF},
07176 { DEF, DEF, DEF, DEF, DEF}}},
07177 /* UG.@A.GU */
07178 {{ 0, 0, 0, 0, 0},
07179 { -429, -429, -429, -429, -429},
07180 { -259, -259, -259, -259, -259},
07181 { -339, -339, -339, -339, -339},
07182 { -329, -329, -329, -329, -329}},
07183 /* UG.@C.GU */
07184 {{ 0, 0, 0, 0, 0},
07185 { -599, -599, -599, -599, -599},
07186 { -239, -239, -239, -239, -239},
07187 { -689, -689, -689, -689, -689},
07188 { -329, -329, -329, -329, -329}},
07189 /* UG.@G.GU */
07190 {{ 0, 0, 0, 0, 0},
07191 { -599, -599, -599, -599, -599},
07192 { -239, -239, -239, -239, -239},
07193 { -689, -689, -689, -689, -689},
07194 { -329, -329, -329, -329, -329}},
07195 /* UG.@U.GU */
07196 {{ 0, 0, 0, 0, 0},
07197 { -599, -599, -599, -599, -599},
07198 { -239, -239, -239, -239, -239},
07199 { -689, -689, -689, -689, -689},
07200 { -329, -329, -329, -329, -329}}}},
07201 /* UG.A@.GU */
07202 {{{ DEF, -719, -789, -959, -809},
07203 { -100, -769, -839, -1009, -859},
07204 { -100, -769, -839, -1009, -859},
07205 { -100, -769, -839, -1009, -859},
07206 { -100, -769, -839, -1009, -859}},
07207 /* UG.AA.GU */
07208 {{ DEF, -719, -789, -959, -809},
07209 { -479, -1148, -1218, -1388, -1238},
07210 { -309, -978, -1048, -1218, -1068},
07211 { -389, -1058, -1128, -1298, -1148},
07212 { -379, -1048, -1118, -1288, -1138}},
07213 /* UG.AC.GU */
07214 {{ DEF, -719, -789, -959, -809},
07215 { -649, -1318, -1388, -1558, -1408},
07216 { -289, -958, -1028, -1198, -1048},
07217 { -739, -1408, -1478, -1648, -1498},
07218 { -379, -1048, -1118, -1288, -1138}},
07219 /* UG.AG.GU */
07220 {{ DEF, -719, -789, -959, -809},
07221 { -649, -1318, -1388, -1558, -1408},
07222 { -289, -958, -1028, -1198, -1048},
07223 { -739, -1408, -1478, -1648, -1498},
07224 { -379, -1048, -1118, -1288, -1138}},
07225 /* UG.AU.GU */
07226 {{ DEF, -719, -789, -959, -809},
07227 { -649, -1318, -1388, -1558, -1408},

```



```
07228 { -289, -958, -1028, -1198, -1048},
07229 { -739, -1408, -1478, -1648, -1498},
07230 { -379, -1048, -1118, -1288, -1138}},
07231 /* UG.C@.GU */
07232 {{ DEF, -479, -479, -359, -479},
07233 { -100, -529, -529, -409, -529},
07234 { -100, -529, -529, -409, -529},
07235 { -100, -529, -529, -409, -529},
07236 { -100, -529, -529, -409, -529}},
07237 /* UG.CA.GU */
07238 {{ DEF, -479, -479, -359, -479},
07239 { -479, -908, -908, -788, -908},
07240 { -309, -738, -738, -618, -738},
07241 { -389, -818, -818, -698, -818},
07242 { -379, -808, -808, -688, -808}},
07243 /* UG.CC.GU */
07244 {{ DEF, -479, -479, -359, -479},
07245 { -649, -1078, -1078, -958, -1078},
07246 { -289, -718, -718, -598, -718},
07247 { -739, -1168, -1168, -1048, -1168},
07248 { -379, -808, -808, -688, -808}},
07249 /* UG.CG.GU */
07250 {{ DEF, -479, -479, -359, -479},
07251 { -649, -1078, -1078, -958, -1078},
07252 { -289, -718, -718, -598, -718},
07253 { -739, -1168, -1168, -1048, -1168},
07254 { -379, -808, -808, -688, -808}},
07255 /* UG.CU.GU */
07256 {{ DEF, -479, -479, -359, -479},
07257 { -649, -1078, -1078, -958, -1078},
07258 { -289, -718, -718, -598, -718},
07259 { -739, -1168, -1168, -1048, -1168},
07260 { -379, -808, -808, -688, -808}},
07261 /* UG.G@.GU */
07262 {{ DEF, -659, -809, -919, -809},
07263 { -100, -709, -859, -969, -859},
07264 { -100, -709, -859, -969, -859},
07265 { -100, -709, -859, -969, -859},
07266 { -100, -709, -859, -969, -859}},
07267 /* UG.GA.GU */
07268 {{ DEF, -659, -809, -919, -809},
07269 { -479, -1088, -1238, -1348, -1238},
07270 { -309, -918, -1068, -1178, -1068},
07271 { -389, -998, -1148, -1258, -1148},
07272 { -379, -988, -1138, -1248, -1138}},
07273 /* UG.GC.GU */
07274 {{ DEF, -659, -809, -919, -809},
07275 { -649, -1258, -1408, -1518, -1408},
07276 { -289, -898, -1048, -1158, -1048},
07277 { -739, -1348, -1498, -1608, -1498},
07278 { -379, -988, -1138, -1248, -1138}},
07279 /* UG.GG.GU */
07280 {{ DEF, -659, -809, -919, -809},
07281 { -649, -1258, -1408, -1518, -1408},
07282 { -289, -898, -1048, -1158, -1048},
07283 { -739, -1348, -1498, -1608, -1498},
07284 { -379, -988, -1138, -1248, -1138}},
07285 /* UG.GU.GU */
07286 {{ DEF, -659, -809, -919, -809},
07287 { -649, -1258, -1408, -1518, -1408},
07288 { -289, -898, -1048, -1158, -1048},
07289 { -739, -1348, -1498, -1608, -1498},
07290 { -379, -988, -1138, -1248, -1138}},
07291 /* UG.U@.GU */
07292 {{ DEF, -549, -439, -549, -359},
07293 { -100, -599, -489, -599, -409},
07294 { -100, -599, -489, -599, -409},
07295 { -100, -599, -489, -599, -409},
07296 { -100, -599, -489, -599, -409}},
07297 /* UG.UA.GU */
07298 {{ DEF, -549, -439, -549, -359},
07299 { -479, -978, -868, -978, -788},
07300 { -309, -808, -698, -808, -618},
07301 { -389, -888, -778, -888, -698},
07302 { -379, -878, -768, -878, -688}},
07303 /* UG.UC.GU */
07304 {{ DEF, -549, -439, -549, -359},
07305 { -649, -1148, -1038, -1148, -958},
07306 { -289, -788, -678, -788, -598},
07307 { -739, -1238, -1128, -1238, -1048},
07308 { -379, -878, -768, -878, -688}},
07309 /* UG.UG.GU */
07310 {{ DEF, -549, -439, -549, -359},
07311 { -649, -1148, -1038, -1148, -958},
07312 { -289, -788, -678, -788, -598},
07313 { -739, -1238, -1128, -1238, -1048},
07314 { -379, -878, -768, -878, -688}},
```

```

07315 /* UG.UU..GU */
07316 {{ DEF, -549, -439, -549, -359},
07317 { -649,-1148,-1038,-1148, -958},
07318 { -289, -788, -678, -788, -598},
07319 { -739,-1238,-1128,-1238,-1048},
07320 { -379, -878, -768, -878, -688}}},
07321 /* UG.@@..UG */
07322 {{{ 0, 0, 0, 0, 0},
07323 { DEF, DEF, DEF, DEF, DEF},
07324 { DEF, DEF, DEF, DEF, DEF},
07325 { DEF, DEF, DEF, DEF, DEF},
07326 { DEF, DEF, DEF, DEF, DEF}}},
07327 /* UG.@A..UG */
07328 {{ 0, 0, 0, 0, 0},
07329 { -719, -719, -719, -719, -719},
07330 { -479, -479, -479, -479, -479},
07331 { -659, -659, -659, -659, -659},
07332 { -549, -549, -549, -549, -549}},
07333 /* UG.@C..UG */
07334 {{ 0, 0, 0, 0, 0},
07335 { -789, -789, -789, -789, -789},
07336 { -479, -479, -479, -479, -479},
07337 { -809, -809, -809, -809, -809},
07338 { -439, -439, -439, -439, -439}},
07339 /* UG.@G..UG */
07340 {{ 0, 0, 0, 0, 0},
07341 { -959, -959, -959, -959, -959},
07342 { -359, -359, -359, -359, -359},
07343 { -919, -919, -919, -919, -919},
07344 { -549, -549, -549, -549, -549}},
07345 /* UG.@U..UG */
07346 {{ 0, 0, 0, 0, 0},
07347 { -809, -809, -809, -809, -809},
07348 { -479, -479, -479, -479, -479},
07349 { -809, -809, -809, -809, -809},
07350 { -359, -359, -359, -359, -359}}},
07351 /* UG.A@..UG */
07352 {{{ DEF, -719, -789, -959, -809},
07353 { -100, -769, -839,-1009, -859},
07354 { -100, -769, -839,-1009, -859},
07355 { -100, -769, -839,-1009, -859},
07356 { -100, -769, -839,-1009, -859}}},
07357 /* UG.AA..UG */
07358 {{ DEF, -719, -789, -959, -809},
07359 { -769,-1438,-1508,-1678,-1528},
07360 { -529,-1198,-1268,-1438,-1288},
07361 { -709,-1378,-1448,-1618,-1468},
07362 { -599,-1268,-1338,-1508,-1358}},
07363 /* UG.AC..UG */
07364 {{ DEF, -719, -789, -959, -809},
07365 { -839,-1508,-1578,-1748,-1598},
07366 { -529,-1198,-1268,-1438,-1288},
07367 { -859,-1528,-1598,-1768,-1618},
07368 { -489,-1158,-1228,-1398,-1248}},
07369 /* UG.AG..UG */
07370 {{ DEF, -719, -789, -959, -809},
07371 {-1009,-1678,-1748,-1918,-1768},
07372 { -409,-1078,-1148,-1318,-1168},
07373 { -969,-1638,-1708,-1878,-1728},
07374 { -599,-1268,-1338,-1508,-1358}},
07375 /* UG.AU..UG */
07376 {{ DEF, -719, -789, -959, -809},
07377 { -859,-1528,-1598,-1768,-1618},
07378 { -529,-1198,-1268,-1438,-1288},
07379 { -859,-1528,-1598,-1768,-1618},
07380 { -409,-1078,-1148,-1318,-1168}}},
07381 /* UG.C@..UG */
07382 {{{ DEF, -479, -479, -359, -479},
07383 { -100, -529, -529, -409, -529},
07384 { -100, -529, -529, -409, -529},
07385 { -100, -529, -529, -409, -529},
07386 { -100, -529, -529, -409, -529}}},
07387 /* UG.CA..UG */
07388 {{ DEF, -479, -479, -359, -479},
07389 { -769,-1198,-1198,-1078,-1198},
07390 { -529, -958, -958, -838, -958},
07391 { -709,-1138,-1138,-1018,-1138},
07392 { -599,-1028,-1028, -908,-1028}},
07393 /* UG.CC..UG */
07394 {{ DEF, -479, -479, -359, -479},
07395 { -839,-1268,-1268,-1148,-1268},
07396 { -529, -958, -958, -838, -958},
07397 { -859,-1288,-1288,-1168,-1288},
07398 { -489, -918, -918, -798, -918}},
07399 /* UG.CG..UG */
07400 {{ DEF, -479, -479, -359, -479},
07401 {-1009,-1438,-1438,-1318,-1438},

```

```

07402 { -409, -838, -838, -718, -838},
07403 { -969,-1398,-1398,-1278,-1398},
07404 { -599,-1028,-1028, -908,-1028}},
07405 /* UG.CU..UG */
07406 {{ DEF, -479, -479, -359, -479},
07407 { -859,-1288,-1288,-1168,-1288},
07408 { -529, -958, -958, -838, -958},
07409 { -859,-1288,-1288,-1168,-1288}},
07410 { -409, -838, -838, -718, -838}}},
07411 /* UG.G@..UG */
07412 {{{ DEF, -659, -809, -919, -809},
07413 { -100, -709, -859, -969, -859},
07414 { -100, -709, -859, -969, -859},
07415 { -100, -709, -859, -969, -859},
07416 { -100, -709, -859, -969, -859}}},
07417 /* UG.GA..UG */
07418 {{ DEF, -659, -809, -919, -809},
07419 { -769,-1378,-1528,-1638,-1528},
07420 { -529,-1138,-1288,-1398,-1288},
07421 { -709,-1318,-1468,-1578,-1468},
07422 { -599,-1208,-1358,-1468,-1358}},
07423 /* UG.GC..UG */
07424 {{ DEF, -659, -809, -919, -809},
07425 { -839,-1448,-1598,-1708,-1598},
07426 { -529,-1138,-1288,-1398,-1288},
07427 { -859,-1468,-1618,-1728,-1618},
07428 { -489,-1098,-1248,-1358,-1248}}},
07429 /* UG.GG..UG */
07430 {{{ DEF, -659, -809, -919, -809},
07431 {-1009,-1618,-1768,-1878,-1768},
07432 { -409,-1018,-1168,-1278,-1168},
07433 { -969,-1578,-1728,-1838,-1728},
07434 { -599,-1208,-1358,-1468,-1358}},
07435 /* UG.GU..UG */
07436 {{{ DEF, -659, -809, -919, -809},
07437 { -859,-1468,-1618,-1728,-1618},
07438 { -529,-1138,-1288,-1398,-1288},
07439 { -859,-1468,-1618,-1728,-1618},
07440 { -409,-1018,-1168,-1278,-1168}}}},
07441 /* UG.U@..UG */
07442 {{{ DEF, -549, -439, -549, -359},
07443 { -100, -599, -489, -599, -409},
07444 { -100, -599, -489, -599, -409},
07445 { -100, -599, -489, -599, -409},
07446 { -100, -599, -489, -599, -409}}},
07447 /* UG.UA..UG */
07448 {{ DEF, -549, -439, -549, -359},
07449 { -769,-1268,-1158,-1268,-1078},
07450 { -529,-1028, -918,-1028, -838},
07451 { -709,-1208,-1098,-1208,-1018},
07452 { -599,-1098, -988,-1098, -908}},
07453 /* UG.UC..UG */
07454 {{ DEF, -549, -439, -549, -359},
07455 { -839,-1338,-1228,-1338,-1148},
07456 { -529,-1028, -918,-1028, -838},
07457 { -859,-1358,-1248,-1358,-1168},
07458 { -489, -988, -878, -988, -798}},
07459 /* UG.UG..UG */
07460 {{ DEF, -549, -439, -549, -359},
07461 {-1009,-1508,-1398,-1508,-1318},
07462 { -409, -908, -798, -908, -718},
07463 { -969,-1468,-1358,-1468,-1278},
07464 { -599,-1098, -988,-1098, -908}},
07465 /* UG.UU..UG */
07466 {{ DEF, -549, -439, -549, -359},
07467 { -859,-1358,-1248,-1358,-1168},
07468 { -529,-1028, -918,-1028, -838},
07469 { -859,-1358,-1248,-1358,-1168},
07470 { -409, -908, -798, -908, -718}}}},
07471 /* UG.@@..AU */
07472 {{{ 0, 0, 0, 0, 0},
07473 { DEF, DEF, DEF, DEF, DEF},
07474 { DEF, DEF, DEF, DEF, DEF},
07475 { DEF, DEF, DEF, DEF, DEF},
07476 { DEF, DEF, DEF, DEF, DEF}}},
07477 /* UG.@A..AU */
07478 {{ 0, 0, 0, 0, 0},
07479 { -429, -429, -429, -429, -429},
07480 { -259, -259, -259, -259, -259},
07481 { -339, -339, -339, -339, -339},
07482 { -329, -329, -329, -329, -329}},
07483 /* UG.@C..AU */
07484 {{{ 0, 0, 0, 0, 0},
07485 { -599, -599, -599, -599, -599},
07486 { -239, -239, -239, -239, -239},
07487 { -689, -689, -689, -689, -689},
07488 { -329, -329, -329, -329, -329}},

```

```

07489 /* UG.@G..AU */
07490 {{ 0, 0, 0, 0, 0},
07491 { -599, -599, -599, -599, -599},
07492 { -239, -239, -239, -239, -239},
07493 { -689, -689, -689, -689, -689},
07494 { -329, -329, -329, -329, -329}},
07495 /* UG.@U..AU */
07496 {{ 0, 0, 0, 0, 0},
07497 { -599, -599, -599, -599, -599},
07498 { -239, -239, -239, -239, -239},
07499 { -689, -689, -689, -689, -689},
07500 { -329, -329, -329, -329, -329}},
07501 /* UG.A@..AU */
07502 {{{ DEF, -719, -789, -959, -809},
07503 { -100, -769, -839, -1009, -859},
07504 { -100, -769, -839, -1009, -859},
07505 { -100, -769, -839, -1009, -859},
07506 { -100, -769, -839, -1009, -859}}},
07507 /* UG.AA..AU */
07508 {{ DEF, -719, -789, -959, -809},
07509 { -479, -1148, -1218, -1388, -1238},
07510 { -309, -978, -1048, -1218, -1068},
07511 { -389, -1058, -1128, -1298, -1148},
07512 { -379, -1048, -1118, -1288, -1138}},
07513 /* UG.AC..AU */
07514 {{ DEF, -719, -789, -959, -809},
07515 { -649, -1318, -1388, -1558, -1408},
07516 { -289, -958, -1028, -1198, -1048},
07517 { -739, -1408, -1478, -1648, -1498},
07518 { -379, -1048, -1118, -1288, -1138}},
07519 /* UG.AG..AU */
07520 {{ DEF, -719, -789, -959, -809},
07521 { -649, -1318, -1388, -1558, -1408},
07522 { -289, -958, -1028, -1198, -1048},
07523 { -739, -1408, -1478, -1648, -1498},
07524 { -379, -1048, -1118, -1288, -1138}},
07525 /* UG.AU..AU */
07526 {{ DEF, -719, -789, -959, -809},
07527 { -649, -1318, -1388, -1558, -1408},
07528 { -289, -958, -1028, -1198, -1048},
07529 { -739, -1408, -1478, -1648, -1498},
07530 { -379, -1048, -1118, -1288, -1138}},
07531 /* UG.C@..AU */
07532 {{{ DEF, -479, -479, -359, -479},
07533 { -100, -529, -529, -409, -529},
07534 { -100, -529, -529, -409, -529},
07535 { -100, -529, -529, -409, -529},
07536 { -100, -529, -529, -409, -529}}},
07537 /* UG.CA..AU */
07538 {{ DEF, -479, -479, -359, -479},
07539 { -479, -908, -908, -788, -908},
07540 { -309, -738, -738, -618, -738},
07541 { -389, -818, -818, -698, -818},
07542 { -379, -808, -808, -688, -808}},
07543 /* UG.CC..AU */
07544 {{ DEF, -479, -479, -359, -479},
07545 { -649, -1078, -1078, -958, -1078},
07546 { -289, -718, -718, -598, -718},
07547 { -739, -1168, -1168, -1048, -1168},
07548 { -379, -808, -808, -688, -808}},
07549 /* UG.CG..AU */
07550 {{ DEF, -479, -479, -359, -479},
07551 { -649, -1078, -1078, -958, -1078},
07552 { -289, -718, -718, -598, -718},
07553 { -739, -1168, -1168, -1048, -1168},
07554 { -379, -808, -808, -688, -808}},
07555 /* UG.CU..AU */
07556 {{ DEF, -479, -479, -359, -479},
07557 { -649, -1078, -1078, -958, -1078},
07558 { -289, -718, -718, -598, -718},
07559 { -739, -1168, -1168, -1048, -1168},
07560 { -379, -808, -808, -688, -808}},
07561 /* UG.G@..AU */
07562 {{{ DEF, -659, -809, -919, -809},
07563 { -100, -709, -859, -969, -859},
07564 { -100, -709, -859, -969, -859},
07565 { -100, -709, -859, -969, -859},
07566 { -100, -709, -859, -969, -859}}},
07567 /* UG.GA..AU */
07568 {{ DEF, -659, -809, -919, -809},
07569 { -479, -1088, -1238, -1348, -1238},
07570 { -309, -918, -1068, -1178, -1068},
07571 { -389, -998, -1148, -1258, -1148},
07572 { -379, -988, -1138, -1248, -1138}},
07573 /* UG.GC..AU */
07574 {{ DEF, -659, -809, -919, -809},
07575 { -649, -1258, -1408, -1518, -1408},

```

```

07576 { -289, -898, -1048, -1158, -1048},
07577 { -739, -1348, -1498, -1608, -1498},
07578 { -379, -988, -1138, -1248, -1138}},
07579 /* UG.GG..AU */
07580 {{ DEF, -659, -809, -919, -809},
07581 { -649, -1258, -1408, -1518, -1408},
07582 { -289, -898, -1048, -1158, -1048},
07583 { -739, -1348, -1498, -1608, -1498},
07584 { -379, -988, -1138, -1248, -1138}},
07585 /* UG.GU..AU */
07586 {{ DEF, -659, -809, -919, -809},
07587 { -649, -1258, -1408, -1518, -1408},
07588 { -289, -898, -1048, -1158, -1048},
07589 { -739, -1348, -1498, -1608, -1498},
07590 { -379, -988, -1138, -1248, -1138}}},
07591 /* UG.U@..AU */
07592 {{{ DEF, -549, -439, -549, -359},
07593 { -100, -599, -489, -599, -409},
07594 { -100, -599, -489, -599, -409},
07595 { -100, -599, -489, -599, -409},
07596 { -100, -599, -489, -599, -409}}},
07597 /* UG.UA..AU */
07598 {{ DEF, -549, -439, -549, -359},
07599 { -479, -978, -868, -978, -788},
07600 { -309, -808, -698, -808, -618},
07601 { -389, -888, -778, -888, -698},
07602 { -379, -878, -768, -878, -688}},
07603 /* UG.UC..AU */
07604 {{ DEF, -549, -439, -549, -359},
07605 { -649, -1148, -1038, -1148, -958},
07606 { -289, -788, -678, -788, -598},
07607 { -739, -1238, -1128, -1238, -1048},
07608 { -379, -878, -768, -878, -688}},
07609 /* UG.UG..AU */
07610 {{ DEF, -549, -439, -549, -359},
07611 { -649, -1148, -1038, -1148, -958},
07612 { -289, -788, -678, -788, -598},
07613 { -739, -1238, -1128, -1238, -1048},
07614 { -379, -878, -768, -878, -688}},
07615 /* UG.UU..AU */
07616 {{ DEF, -549, -439, -549, -359},
07617 { -649, -1148, -1038, -1148, -958},
07618 { -289, -788, -678, -788, -598},
07619 { -739, -1238, -1128, -1238, -1048},
07620 { -379, -878, -768, -878, -688}}}},
07621 /* UG.@@..UA */
07622 {{{{ 0, 0, 0, 0, 0},
07623 { DEF, DEF, DEF, DEF, DEF},
07624 { DEF, DEF, DEF, DEF, DEF},
07625 { DEF, DEF, DEF, DEF, DEF},
07626 { DEF, DEF, DEF, DEF, DEF}}}},
07627 /* UG.@A..UA */
07628 {{ 0, 0, 0, 0, 0},
07629 { -399, -399, -399, -399, -399},
07630 { -429, -429, -429, -429, -429},
07631 { -379, -379, -379, -379, -379},
07632 { -279, -279, -279, -279, -279}},
07633 /* UG.@C..UA */
07634 {{ 0, 0, 0, 0, 0},
07635 { -629, -629, -629, -629, -629},
07636 { -509, -509, -509, -509, -509},
07637 { -679, -679, -679, -679, -679},
07638 { -139, -139, -139, -139, -139}},
07639 /* UG.@G..UA */
07640 {{ 0, 0, 0, 0, 0},
07641 { -889, -889, -889, -889, -889},
07642 { -199, -199, -199, -199, -199},
07643 { -889, -889, -889, -889, -889},
07644 { -279, -279, -279, -279, -279}},
07645 /* UG.@U..UA */
07646 {{ 0, 0, 0, 0, 0},
07647 { -589, -589, -589, -589, -589},
07648 { -179, -179, -179, -179, -179},
07649 { -679, -679, -679, -679, -679},
07650 { -140, -140, -140, -140, -140}}}},
07651 /* UG.A@..UA */
07652 {{{ DEF, -719, -789, -959, -809},
07653 { -100, -769, -839, -1009, -859},
07654 { -100, -769, -839, -1009, -859},
07655 { -100, -769, -839, -1009, -859},
07656 { -100, -769, -839, -1009, -859}},
07657 /* UG.AA..UA */
07658 {{ DEF, -719, -789, -959, -809},
07659 { -449, -1118, -1188, -1358, -1208},
07660 { -479, -1148, -1218, -1388, -1238},
07661 { -429, -1098, -1168, -1338, -1188},
07662 { -329, -998, -1068, -1238, -1088}},

```

```
07663 /* UG.AC..UA */
07664 {{ DEF, -719, -789, -959, -809},
07665 { -679,-1348,-1418,-1588,-1438},
07666 { -559,-1228,-1298,-1468,-1318},
07667 { -729,-1398,-1468,-1638,-1488},
07668 { -189, -858, -928,-1098, -948}},
07669 /* UG.AG..UA */
07670 {{ DEF, -719, -789, -959, -809},
07671 { -939,-1608,-1678,-1848,-1698},
07672 { -249, -918, -988,-1158,-1008},
07673 { -939,-1608,-1678,-1848,-1698},
07674 { -329, -998,-1068,-1238,-1088}},
07675 /* UG.AU..UA */
07676 {{ DEF, -719, -789, -959, -809},
07677 { -639,-1308,-1378,-1548,-1398},
07678 { -229, -898, -968,-1138, -988},
07679 { -729,-1398,-1468,-1638,-1488},
07680 { -190, -859, -929,-1099, -949}}},
07681 /* UG.C@..UA */
07682 {{{ DEF, -479, -479, -359, -479},
07683 { -100, -529, -529, -409, -529},
07684 { -100, -529, -529, -409, -529},
07685 { -100, -529, -529, -409, -529},
07686 { -100, -529, -529, -409, -529}}},
07687 /* UG.CA..UA */
07688 {{ DEF, -479, -479, -359, -479},
07689 { -449, -878, -878, -758, -878},
07690 { -479, -908, -908, -788, -908},
07691 { -429, -858, -858, -738, -858},
07692 { -329, -758, -758, -638, -758}},
07693 /* UG.CC..UA */
07694 {{ DEF, -479, -479, -359, -479},
07695 { -679,-1108,-1108, -988,-1108},
07696 { -559, -988, -988, -868, -988},
07697 { -729,-1158,-1158,-1038,-1158},
07698 { -189, -618, -618, -498, -618}},
07699 /* UG.CG..UA */
07700 {{ DEF, -479, -479, -359, -479},
07701 { -939,-1368,-1368,-1248,-1368},
07702 { -249, -678, -678, -558, -678},
07703 { -939,-1368,-1368,-1248,-1368},
07704 { -329, -758, -758, -638, -758}},
07705 /* UG.CU..UA */
07706 {{ DEF, -479, -479, -359, -479},
07707 { -639,-1068,-1068, -948,-1068},
07708 { -229, -658, -658, -538, -658},
07709 { -729,-1158,-1158,-1038,-1158},
07710 { -190, -619, -619, -499, -619}}},
07711 /* UG.G@..UA */
07712 {{{ DEF, -659, -809, -919, -809},
07713 { -100, -709, -859, -969, -859},
07714 { -100, -709, -859, -969, -859},
07715 { -100, -709, -859, -969, -859},
07716 { -100, -709, -859, -969, -859}}},
07717 /* UG.GA..UA */
07718 {{ DEF, -659, -809, -919, -809},
07719 { -449,-1058,-1208,-1318,-1208},
07720 { -479,-1088,-1238,-1348,-1238},
07721 { -429,-1038,-1188,-1298,-1188},
07722 { -329, -938,-1088,-1198,-1088}},
07723 /* UG.GC..UA */
07724 {{ DEF, -659, -809, -919, -809},
07725 { -679,-1288,-1438,-1548,-1438},
07726 { -559,-1168,-1318,-1428,-1318},
07727 { -729,-1338,-1488,-1598,-1488},
07728 { -189, -798, -948,-1058,-948}},
07729 /* UG.GG..UA */
07730 {{ DEF, -659, -809, -919, -809},
07731 { -939,-1548,-1698,-1808,-1698},
07732 { -249, -858,-1008,-1118,-1008},
07733 { -939,-1548,-1698,-1808,-1698},
07734 { -329, -938,-1088,-1198,-1088}},
07735 /* UG.GU..UA */
07736 {{ DEF, -659, -809, -919, -809},
07737 { -639,-1248,-1398,-1508,-1398},
07738 { -229, -838, -988,-1098, -988},
07739 { -729,-1338,-1488,-1598,-1488},
07740 { -190, -799, -949,-1059, -949}}},
07741 /* UG.U@..UA */
07742 {{{ DEF, -549, -439, -549, -359},
07743 { -100, -599, -489, -599, -409},
07744 { -100, -599, -489, -599, -409},
07745 { -100, -599, -489, -599, -409},
07746 { -100, -599, -489, -599, -409}}},
07747 /* UG.UA..UA */
07748 {{ DEF, -549, -439, -549, -359},
07749 { -449, -948, -838, -948, -758},
```

```
07750 { -479, -978, -868, -978, -788},
07751 { -429, -928, -818, -928, -738},
07752 { -329, -828, -718, -828, -638}},
07753 /* UG.UC..UA */
07754 {{ DEF, -549, -439, -549, -359},
07755 { -679,-1178,-1068,-1178, -988},
07756 { -559,-1058, -948,-1058, -868},
07757 { -729,-1228,-1118,-1228,-1038},
07758 { -189, -688, -578, -688, -498}},
07759 /* UG.UG..UA */
07760 {{ DEF, -549, -439, -549, -359},
07761 { -939,-1438,-1328,-1438,-1248},
07762 { -249, -748, -638, -748, -558},
07763 { -939,-1438,-1328,-1438,-1248},
07764 { -329, -828, -718, -828, -638}},
07765 /* UG.UU..UA */
07766 {{ DEF, -549, -439, -549, -359},
07767 { -639,-1138,-1028,-1138, -948},
07768 { -229, -728, -618, -728, -538},
07769 { -729,-1228,-1118,-1228,-1038},
07770 { -190, -689, -579, -689, -499}}}},
07771 /* UG.@@.. @ */
07772 {{{ DEF, DEF, DEF, DEF, DEF},
07773 { DEF, DEF, DEF, DEF, DEF},
07774 { DEF, DEF, DEF, DEF, DEF},
07775 { DEF, DEF, DEF, DEF, DEF},
07776 { DEF, DEF, DEF, DEF, DEF}}},
07777 /* UG.@A.. @ */
07778 {{ DEF, DEF, DEF, DEF, DEF},
07779 { DEF, DEF, DEF, DEF, DEF},
07780 { DEF, DEF, DEF, DEF, DEF},
07781 { DEF, DEF, DEF, DEF, DEF},
07782 { DEF, DEF, DEF, DEF, DEF}}},
07783 /* UG.@C.. @ */
07784 {{ DEF, DEF, DEF, DEF, DEF},
07785 { DEF, DEF, DEF, DEF, DEF},
07786 { DEF, DEF, DEF, DEF, DEF},
07787 { DEF, DEF, DEF, DEF, DEF},
07788 { DEF, DEF, DEF, DEF, DEF}}},
07789 /* UG.@G.. @ */
07790 {{ DEF, DEF, DEF, DEF, DEF},
07791 { DEF, DEF, DEF, DEF, DEF},
07792 { DEF, DEF, DEF, DEF, DEF},
07793 { DEF, DEF, DEF, DEF, DEF},
07794 { DEF, DEF, DEF, DEF, DEF}}},
07795 /* UG.@U.. @ */
07796 {{ DEF, DEF, DEF, DEF, DEF},
07797 { DEF, DEF, DEF, DEF, DEF},
07798 { DEF, DEF, DEF, DEF, DEF},
07799 { DEF, DEF, DEF, DEF, DEF},
07800 { DEF, DEF, DEF, DEF, DEF}}}},
07801 /* UG.A@.. @ */
07802 {{{ -100, -769, -839,-1009, -859},
07803 { -100, -769, -839,-1009, -859},
07804 { -100, -769, -839,-1009, -859},
07805 { -100, -769, -839,-1009, -859},
07806 { -100, -769, -839,-1009, -859}},
07807 /* UG.AA.. @ */
07808 {{ -100, -769, -839,-1009, -859},
07809 { -100, -769, -839,-1009, -859},
07810 { -100, -769, -839,-1009, -859},
07811 { -100, -769, -839,-1009, -859},
07812 { -100, -769, -839,-1009, -859}},
07813 /* UG.AC.. @ */
07814 {{ -100, -769, -839,-1009, -859},
07815 { -100, -769, -839,-1009, -859},
07816 { -100, -769, -839,-1009, -859},
07817 { -100, -769, -839,-1009, -859},
07818 { -100, -769, -839,-1009, -859}},
07819 /* UG.AG.. @ */
07820 {{ -100, -769, -839,-1009, -859},
07821 { -100, -769, -839,-1009, -859},
07822 { -100, -769, -839,-1009, -859},
07823 { -100, -769, -839,-1009, -859},
07824 { -100, -769, -839,-1009, -859}},
07825 /* UG.AU.. @ */
07826 {{ -100, -769, -839,-1009, -859},
07827 { -100, -769, -839,-1009, -859},
07828 { -100, -769, -839,-1009, -859},
07829 { -100, -769, -839,-1009, -859},
07830 { -100, -769, -839,-1009, -859}}}},
07831 /* UG.C@.. @ */
07832 {{{ -100, -529, -529, -409, -529},
07833 { -100, -529, -529, -409, -529},
07834 { -100, -529, -529, -409, -529},
07835 { -100, -529, -529, -409, -529},
07836 { -100, -529, -529, -409, -529}},
```

```

07837 /* UG.CA.. @ */
07838 {{ -100, -529, -529, -409, -529},
07839 { -100, -529, -529, -409, -529},
07840 { -100, -529, -529, -409, -529},
07841 { -100, -529, -529, -409, -529},
07842 { -100, -529, -529, -409, -529}},
07843 /* UG.CC.. @ */
07844 {{ -100, -529, -529, -409, -529},
07845 { -100, -529, -529, -409, -529},
07846 { -100, -529, -529, -409, -529},
07847 { -100, -529, -529, -409, -529},
07848 { -100, -529, -529, -409, -529}},
07849 /* UG.CG.. @ */
07850 {{ -100, -529, -529, -409, -529},
07851 { -100, -529, -529, -409, -529},
07852 { -100, -529, -529, -409, -529},
07853 { -100, -529, -529, -409, -529},
07854 { -100, -529, -529, -409, -529}},
07855 /* UG.CU.. @ */
07856 {{ -100, -529, -529, -409, -529},
07857 { -100, -529, -529, -409, -529},
07858 { -100, -529, -529, -409, -529},
07859 { -100, -529, -529, -409, -529},
07860 { -100, -529, -529, -409, -529}}},
07861 /* UG.G@.. @ */
07862 {{{ -100, -709, -859, -969, -859},
07863 { -100, -709, -859, -969, -859},
07864 { -100, -709, -859, -969, -859},
07865 { -100, -709, -859, -969, -859},
07866 { -100, -709, -859, -969, -859}},
07867 /* UG.GA.. @ */
07868 {{ -100, -709, -859, -969, -859},
07869 { -100, -709, -859, -969, -859},
07870 { -100, -709, -859, -969, -859},
07871 { -100, -709, -859, -969, -859},
07872 { -100, -709, -859, -969, -859}},
07873 /* UG.GC.. @ */
07874 {{ -100, -709, -859, -969, -859},
07875 { -100, -709, -859, -969, -859},
07876 { -100, -709, -859, -969, -859},
07877 { -100, -709, -859, -969, -859},
07878 { -100, -709, -859, -969, -859}},
07879 /* UG.GG.. @ */
07880 {{ -100, -709, -859, -969, -859},
07881 { -100, -709, -859, -969, -859},
07882 { -100, -709, -859, -969, -859},
07883 { -100, -709, -859, -969, -859},
07884 { -100, -709, -859, -969, -859}},
07885 /* UG.GU.. @ */
07886 {{ -100, -709, -859, -969, -859},
07887 { -100, -709, -859, -969, -859},
07888 { -100, -709, -859, -969, -859},
07889 { -100, -709, -859, -969, -859},
07890 { -100, -709, -859, -969, -859}}},
07891 /* UG.U@.. @ */
07892 {{{ -100, -599, -489, -599, -409},
07893 { -100, -599, -489, -599, -409},
07894 { -100, -599, -489, -599, -409},
07895 { -100, -599, -489, -599, -409},
07896 { -100, -599, -489, -599, -409}},
07897 /* UG.UA.. @ */
07898 {{ -100, -599, -489, -599, -409},
07899 { -100, -599, -489, -599, -409},
07900 { -100, -599, -489, -599, -409},
07901 { -100, -599, -489, -599, -409},
07902 { -100, -599, -489, -599, -409}},
07903 /* UG.UC.. @ */
07904 {{ -100, -599, -489, -599, -409},
07905 { -100, -599, -489, -599, -409},
07906 { -100, -599, -489, -599, -409},
07907 { -100, -599, -489, -599, -409},
07908 { -100, -599, -489, -599, -409}},
07909 /* UG.UG.. @ */
07910 {{{ -100, -599, -489, -599, -409},
07911 { -100, -599, -489, -599, -409},
07912 { -100, -599, -489, -599, -409},
07913 { -100, -599, -489, -599, -409},
07914 { -100, -599, -489, -599, -409}},
07915 /* UG.UU.. @ */
07916 {{{ -100, -599, -489, -599, -409},
07917 { -100, -599, -489, -599, -409},
07918 { -100, -599, -489, -599, -409},
07919 { -100, -599, -489, -599, -409},
07920 { -100, -599, -489, -599, -409}}}},
07921 { /* noPair */ {{{{0}}}},
07922 /* AU.@@..CG */
07923 {{{{ 0, 0, 0, 0, 0}},

```



```

07924 { DEF, DEF, DEF, DEF, DEF},
07925 { DEF, DEF, DEF, DEF, DEF},
07926 { DEF, DEF, DEF, DEF, DEF},
07927 { DEF, DEF, DEF, DEF, DEF}},
07928 /* AU.@A..CG */
07929 {{ 0, 0, 0, 0, 0},
07930 {-1029,-1029,-1029,-1029,-1029},
07931 {-519, -519, -519, -519, -519},
07932 {-939, -939, -939, -939, -939},
07933 {-809, -809, -809, -809, -809}},
07934 /* AU.@C..CG */
07935 {{ 0, 0, 0, 0, 0},
07936 {-949, -949, -949, -949, -949},
07937 {-449, -449, -449, -449, -449},
07938 {-939, -939, -939, -939, -939},
07939 {-739, -739, -739, -739, -739}},
07940 /* AU.@G..CG */
07941 {{ 0, 0, 0, 0, 0},
07942 {-1029,-1029,-1029,-1029,-1029},
07943 {-519, -519, -519, -519, -519},
07944 {-939, -939, -939, -939, -939},
07945 {-809, -809, -809, -809, -809}},
07946 /* AU.@U..CG */
07947 {{ 0, 0, 0, 0, 0},
07948 {-1029,-1029,-1029,-1029,-1029},
07949 {-669, -669, -669, -669, -669},
07950 {-939, -939, -939, -939, -939},
07951 {-859, -859, -859, -859, -859}}},
07952 /* AU.@..CG */
07953 {{{ DEF, -429, -599, -599, -599},
07954 {-100, -479, -649, -649, -649},
07955 {-100, -479, -649, -649, -649},
07956 {-100, -479, -649, -649, -649},
07957 {-100, -479, -649, -649, -649}},
07958 /* AU.AA..CG */
07959 {{ DEF, -429, -599, -599, -599},
07960 {-1079,-1458,-1628,-1628,-1628},
07961 {-569, -948,-1118,-1118,-1118},
07962 {-989,-1368,-1538,-1538,-1538},
07963 {-859,-1238,-1408,-1408,-1408}},
07964 /* AU.AC..CG */
07965 {{ DEF, -429, -599, -599, -599},
07966 {-999,-1378,-1548,-1548,-1548},
07967 {-499, -878,-1048,-1048,-1048},
07968 {-989,-1368,-1538,-1538,-1538},
07969 {-789,-1168,-1338,-1338,-1338}},
07970 /* AU.AG..CG */
07971 {{ DEF, -429, -599, -599, -599},
07972 {-1079,-1458,-1628,-1628,-1628},
07973 {-569, -948,-1118,-1118,-1118},
07974 {-989,-1368,-1538,-1538,-1538},
07975 {-859,-1238,-1408,-1408,-1408}},
07976 /* AU.AU..CG */
07977 {{ DEF, -429, -599, -599, -599},
07978 {-1079,-1458,-1628,-1628,-1628},
07979 {-719,-1098,-1268,-1268,-1268},
07980 {-989,-1368,-1538,-1538,-1538},
07981 {-909,-1288,-1458,-1458,-1458}}},
07982 /* AU.@..CG */
07983 {{{ DEF, -259, -239, -239, -239},
07984 {-100, -309, -289, -289, -289},
07985 {-100, -309, -289, -289, -289},
07986 {-100, -309, -289, -289, -289},
07987 {-100, -309, -289, -289, -289}},
07988 /* AU.CA..CG */
07989 {{ DEF, -259, -239, -239, -239},
07990 {-1079,-1288,-1268,-1268,-1268},
07991 {-569, -778, -758, -758, -758},
07992 {-989,-1198,-1178,-1178,-1178},
07993 {-859,-1068,-1048,-1048,-1048}},
07994 /* AU.CC..CG */
07995 {{ DEF, -259, -239, -239, -239},
07996 {-999,-1208,-1188,-1188,-1188},
07997 {-499, -708, -688, -688, -688},
07998 {-989,-1198,-1178,-1178,-1178},
07999 {-789, -998, -978, -978, -978}},
08000 /* AU.CG..CG */
08001 {{ DEF, -259, -239, -239, -239},
08002 {-1079,-1288,-1268,-1268,-1268},
08003 {-569, -778, -758, -758, -758},
08004 {-989,-1198,-1178,-1178,-1178},
08005 {-859,-1068,-1048,-1048,-1048}},
08006 /* AU.CU..CG */
08007 {{ DEF, -259, -239, -239, -239},
08008 {-1079,-1288,-1268,-1268,-1268},
08009 {-719, -928, -908, -908, -908},
08010 {-989,-1198,-1178,-1178,-1178},

```

```

08011 { -909,-1118,-1098,-1098,-1098}},
08012 /* AU.G@..CG */
08013 {{{ DEF, -339, -689, -689, -689},
08014 { -100, -389, -739, -739, -739},
08015 { -100, -389, -739, -739, -739},
08016 { -100, -389, -739, -739, -739},
08017 { -100, -389, -739, -739, -739}},
08018 /* AU.GA..CG */
08019 {{ DEF, -339, -689, -689, -689},
08020 {-1079,-1368,-1718,-1718,-1718},
08021 { -569, -858,-1208,-1208,-1208},
08022 { -989,-1278,-1628,-1628,-1628},
08023 { -859,-1148,-1498,-1498,-1498}},
08024 /* AU.GC..CG */
08025 {{ DEF, -339, -689, -689, -689},
08026 { -999,-1288,-1638,-1638,-1638},
08027 { -499, -788,-1138,-1138,-1138},
08028 { -989,-1278,-1628,-1628,-1628},
08029 { -789,-1078,-1428,-1428,-1428}},
08030 /* AU.GG..CG */
08031 {{ DEF, -339, -689, -689, -689},
08032 {-1079,-1368,-1718,-1718,-1718},
08033 { -569, -858,-1208,-1208,-1208},
08034 { -989,-1278,-1628,-1628,-1628},
08035 { -859,-1148,-1498,-1498,-1498}},
08036 /* AU.GU..CG */
08037 {{ DEF, -339, -689, -689, -689},
08038 {-1079,-1368,-1718,-1718,-1718},
08039 { -719,-1008,-1358,-1358,-1358},
08040 { -989,-1278,-1628,-1628,-1628},
08041 { -909,-1198,-1548,-1548,-1548}}},
08042 /* AU.U@..CG */
08043 {{{ DEF, -329, -329, -329, -329},
08044 { -100, -379, -379, -379, -379},
08045 { -100, -379, -379, -379, -379},
08046 { -100, -379, -379, -379, -379},
08047 { -100, -379, -379, -379, -379}},
08048 /* AU.UA..CG */
08049 {{ DEF, -329, -329, -329, -329},
08050 {-1079,-1358,-1358,-1358,-1358},
08051 { -569, -848, -848, -848, -848},
08052 { -989,-1268,-1268,-1268,-1268},
08053 { -859,-1138,-1138,-1138,-1138}},
08054 /* AU.UC..CG */
08055 {{ DEF, -329, -329, -329, -329},
08056 { -999,-1278,-1278,-1278,-1278},
08057 { -499, -778, -778, -778, -778},
08058 { -989,-1268,-1268,-1268,-1268},
08059 { -789,-1068,-1068,-1068,-1068}},
08060 /* AU.UG..CG */
08061 {{ DEF, -329, -329, -329, -329},
08062 {-1079,-1358,-1358,-1358,-1358},
08063 { -569, -848, -848, -848, -848},
08064 { -989,-1268,-1268,-1268,-1268},
08065 { -859,-1138,-1138,-1138,-1138}},
08066 /* AU.UU..CG */
08067 {{ DEF, -329, -329, -329, -329},
08068 {-1079,-1358,-1358,-1358,-1358},
08069 { -719, -998, -998, -998, -998},
08070 { -989,-1268,-1268,-1268,-1268},
08071 { -909,-1188,-1188,-1188,-1188}}}},
08072 /* AU.@@..GC */
08073 {{{{ 0, 0, 0, 0, 0},
08074 { DEF, DEF, DEF, DEF, DEF},
08075 { DEF, DEF, DEF, DEF, DEF},
08076 { DEF, DEF, DEF, DEF, DEF},
08077 { DEF, DEF, DEF, DEF, DEF}}},
08078 /* AU.@A..GC */
08079 {{ 0, 0, 0, 0, 0},
08080 { -519, -519, -519, -519, -519},
08081 { -719, -719, -719, -719, -719},
08082 { -709, -709, -709, -709, -709},
08083 { -499, -499, -499, -499, -499}},
08084 /* AU.@C..GC */
08085 {{ 0, 0, 0, 0, 0},
08086 { -879, -879, -879, -879, -879},
08087 { -309, -309, -309, -309, -309},
08088 { -739, -739, -739, -739, -739},
08089 { -499, -499, -499, -499, -499}},
08090 /* AU.@G..GC */
08091 {{ 0, 0, 0, 0, 0},
08092 { -559, -559, -559, -559, -559},
08093 { -309, -309, -309, -309, -309},
08094 { -619, -619, -619, -619, -619},
08095 { -499, -499, -499, -499, -499}},
08096 /* AU.@U..GC */
08097 {{ 0, 0, 0, 0, 0},

```

```
08098 { -879, -879, -879, -879, -879},
08099 { -389, -389, -389, -389, -389},
08100 { -739, -739, -739, -739, -739},
08101 { -569, -569, -569, -569, -569}},
08102 /* AU.A@.GC */
08103 {{ DEF, -429, -599, -599, -599},
08104 { -100, -479, -649, -649, -649},
08105 { -100, -479, -649, -649, -649},
08106 { -100, -479, -649, -649, -649},
08107 { -100, -479, -649, -649, -649}},
08108 /* AU.AA.GC */
08109 {{ DEF, -429, -599, -599, -599},
08110 { -569, -948, -1118, -1118, -1118},
08111 { -769, -1148, -1318, -1318, -1318},
08112 { -759, -1138, -1308, -1308, -1308},
08113 { -549, -928, -1098, -1098, -1098}},
08114 /* AU.AC.GC */
08115 {{ DEF, -429, -599, -599, -599},
08116 { -929, -1308, -1478, -1478, -1478},
08117 { -359, -738, -908, -908, -908},
08118 { -789, -1168, -1338, -1338, -1338},
08119 { -549, -928, -1098, -1098, -1098}},
08120 /* AU.AG.GC */
08121 {{ DEF, -429, -599, -599, -599},
08122 { -609, -988, -1158, -1158, -1158},
08123 { -359, -738, -908, -908, -908},
08124 { -669, -1048, -1218, -1218, -1218},
08125 { -549, -928, -1098, -1098, -1098}},
08126 /* AU.AU.GC */
08127 {{ DEF, -429, -599, -599, -599},
08128 { -929, -1308, -1478, -1478, -1478},
08129 { -439, -818, -988, -988, -988},
08130 { -789, -1168, -1338, -1338, -1338},
08131 { -619, -998, -1168, -1168, -1168}},
08132 /* AU.C@.GC */
08133 {{ DEF, -259, -239, -239, -239},
08134 { -100, -309, -289, -289, -289},
08135 { -100, -309, -289, -289, -289},
08136 { -100, -309, -289, -289, -289},
08137 { -100, -309, -289, -289, -289}},
08138 /* AU.CA.GC */
08139 {{ DEF, -259, -239, -239, -239},
08140 { -569, -778, -758, -758, -758},
08141 { -769, -978, -958, -958, -958},
08142 { -759, -968, -948, -948, -948},
08143 { -549, -758, -738, -738, -738}},
08144 /* AU.CC.GC */
08145 {{ DEF, -259, -239, -239, -239},
08146 { -929, -1138, -1118, -1118, -1118},
08147 { -359, -568, -548, -548, -548},
08148 { -789, -998, -978, -978, -978},
08149 { -549, -758, -738, -738, -738}},
08150 /* AU.CG.GC */
08151 {{ DEF, -259, -239, -239, -239},
08152 { -609, -818, -798, -798, -798},
08153 { -359, -568, -548, -548, -548},
08154 { -669, -878, -858, -858, -858},
08155 { -549, -758, -738, -738, -738}},
08156 /* AU.CU.GC */
08157 {{ DEF, -259, -239, -239, -239},
08158 { -929, -1138, -1118, -1118, -1118},
08159 { -439, -648, -628, -628, -628},
08160 { -789, -998, -978, -978, -978},
08161 { -619, -828, -808, -808, -808}},
08162 /* AU.G@.GC */
08163 {{ DEF, -339, -689, -689, -689},
08164 { -100, -389, -739, -739, -739},
08165 { -100, -389, -739, -739, -739},
08166 { -100, -389, -739, -739, -739},
08167 { -100, -389, -739, -739, -739}},
08168 /* AU.GA.GC */
08169 {{ DEF, -339, -689, -689, -689},
08170 { -569, -858, -1208, -1208, -1208},
08171 { -769, -1058, -1408, -1408, -1408},
08172 { -759, -1048, -1398, -1398, -1398},
08173 { -549, -838, -1188, -1188, -1188}},
08174 /* AU.GC.GC */
08175 {{ DEF, -339, -689, -689, -689},
08176 { -929, -1218, -1568, -1568, -1568},
08177 { -359, -648, -998, -998, -998},
08178 { -789, -1078, -1428, -1428, -1428},
08179 { -549, -838, -1188, -1188, -1188}},
08180 /* AU.GG.GC */
08181 {{ DEF, -339, -689, -689, -689},
08182 { -609, -898, -1248, -1248, -1248},
08183 { -359, -648, -998, -998, -998},
08184 { -669, -958, -1308, -1308, -1308},
```

```

08185 { -549, -838, -1188, -1188, -1188}},
08186 /* AU.GU..GC */
08187 {{ DEF, -339, -689, -689, -689},
08188 { -929, -1218, -1568, -1568, -1568},
08189 { -439, -728, -1078, -1078, -1078},
08190 { -789, -1078, -1428, -1428, -1428},
08191 { -619, -908, -1258, -1258, -1258}}},
08192 /* AU.U@..GC */
08193 {{{ DEF, -329, -329, -329, -329},
08194 { -100, -379, -379, -379, -379},
08195 { -100, -379, -379, -379, -379},
08196 { -100, -379, -379, -379, -379},
08197 { -100, -379, -379, -379, -379}},
08198 /* AU.UA..GC */
08199 {{ DEF, -329, -329, -329, -329},
08200 { -569, -848, -848, -848, -848},
08201 { -769, -1048, -1048, -1048, -1048},
08202 { -759, -1038, -1038, -1038, -1038},
08203 { -549, -828, -828, -828, -828}},
08204 /* AU.UC..GC */
08205 {{ DEF, -329, -329, -329, -329},
08206 { -929, -1208, -1208, -1208, -1208},
08207 { -359, -638, -638, -638, -638},
08208 { -789, -1068, -1068, -1068, -1068},
08209 { -549, -828, -828, -828, -828}},
08210 /* AU.UG..GC */
08211 {{ DEF, -329, -329, -329, -329},
08212 { -609, -888, -888, -888, -888},
08213 { -359, -638, -638, -638, -638},
08214 { -669, -948, -948, -948, -948},
08215 { -549, -828, -828, -828, -828}},
08216 /* AU.UU..GC */
08217 {{ DEF, -329, -329, -329, -329},
08218 { -929, -1208, -1208, -1208, -1208},
08219 { -439, -718, -718, -718, -718},
08220 { -789, -1068, -1068, -1068, -1068},
08221 { -619, -898, -898, -898, -898}}}},
08222 /* AU.@A..GU */
08223 {{{{ 0, 0, 0, 0, 0},
08224 { DEF, DEF, DEF, DEF, DEF},
08225 { DEF, DEF, DEF, DEF, DEF},
08226 { DEF, DEF, DEF, DEF, DEF},
08227 { DEF, DEF, DEF, DEF, DEF}},
08228 /* AU.@A..GU */
08229 {{ 0, 0, 0, 0, 0},
08230 { -429, -429, -429, -429, -429},
08231 { -259, -259, -259, -259, -259},
08232 { -339, -339, -339, -339, -339},
08233 { -329, -329, -329, -329, -329}},
08234 /* AU.@C..GU */
08235 {{ 0, 0, 0, 0, 0},
08236 { -599, -599, -599, -599, -599},
08237 { -239, -239, -239, -239, -239},
08238 { -689, -689, -689, -689, -689},
08239 { -329, -329, -329, -329, -329}},
08240 /* AU.@G..GU */
08241 {{ 0, 0, 0, 0, 0},
08242 { -599, -599, -599, -599, -599},
08243 { -239, -239, -239, -239, -239},
08244 { -689, -689, -689, -689, -689},
08245 { -329, -329, -329, -329, -329}},
08246 /* AU.@U..GU */
08247 {{ 0, 0, 0, 0, 0},
08248 { -599, -599, -599, -599, -599},
08249 { -239, -239, -239, -239, -239},
08250 { -689, -689, -689, -689, -689},
08251 { -329, -329, -329, -329, -329}}}},
08252 /* AU.A@..GU */
08253 {{{ DEF, -429, -599, -599, -599},
08254 { -100, -479, -649, -649, -649},
08255 { -100, -479, -649, -649, -649},
08256 { -100, -479, -649, -649, -649},
08257 { -100, -479, -649, -649, -649}},
08258 /* AU.AA..GU */
08259 {{ DEF, -429, -599, -599, -599},
08260 { -479, -858, -1028, -1028, -1028},
08261 { -309, -688, -858, -858, -858},
08262 { -389, -768, -938, -938, -938},
08263 { -379, -758, -928, -928, -928}},
08264 /* AU.AC..GU */
08265 {{ DEF, -429, -599, -599, -599},
08266 { -649, -1028, -1198, -1198, -1198},
08267 { -289, -668, -838, -838, -838},
08268 { -739, -1118, -1288, -1288, -1288},
08269 { -379, -758, -928, -928, -928}},
08270 /* AU.AG..GU */
08271 {{ DEF, -429, -599, -599, -599},

```

```
08272 { -649,-1028,-1198,-1198,-1198},
08273 { -289, -668, -838, -838, -838},
08274 { -739,-1118,-1288,-1288,-1288},
08275 { -379, -758, -928, -928, -928}},
08276 /* AU.AU..GU */
08277 {{ DEF, -429, -599, -599, -599},
08278 { -649,-1028,-1198,-1198,-1198},
08279 { -289, -668, -838, -838, -838},
08280 { -739,-1118,-1288,-1288,-1288},
08281 { -379, -758, -928, -928, -928}}},
08282 /* AU.C@..GU */
08283 {{{ DEF, -259, -239, -239, -239},
08284 { -100, -309, -289, -289, -289},
08285 { -100, -309, -289, -289, -289},
08286 { -100, -309, -289, -289, -289},
08287 { -100, -309, -289, -289, -289}},
08288 /* AU.CA..GU */
08289 {{ DEF, -259, -239, -239, -239},
08290 { -479, -688, -668, -668, -668},
08291 { -309, -518, -498, -498, -498},
08292 { -389, -598, -578, -578, -578},
08293 { -379, -588, -568, -568, -568}},
08294 /* AU.CC..GU */
08295 {{ DEF, -259, -239, -239, -239},
08296 { -649, -858, -838, -838, -838},
08297 { -289, -498, -478, -478, -478},
08298 { -739, -948, -928, -928, -928},
08299 { -379, -588, -568, -568, -568}},
08300 /* AU.CG..GU */
08301 {{ DEF, -259, -239, -239, -239},
08302 { -649, -858, -838, -838, -838},
08303 { -289, -498, -478, -478, -478},
08304 { -739, -948, -928, -928, -928},
08305 { -379, -588, -568, -568, -568}},
08306 /* AU.CU..GU */
08307 {{ DEF, -259, -239, -239, -239},
08308 { -649, -858, -838, -838, -838},
08309 { -289, -498, -478, -478, -478},
08310 { -739, -948, -928, -928, -928},
08311 { -379, -588, -568, -568, -568}}},
08312 /* AU.G@..GU */
08313 {{{ DEF, -339, -689, -689, -689},
08314 { -100, -389, -739, -739, -739},
08315 { -100, -389, -739, -739, -739},
08316 { -100, -389, -739, -739, -739},
08317 { -100, -389, -739, -739, -739}},
08318 /* AU.GA..GU */
08319 {{ DEF, -339, -689, -689, -689},
08320 { -479, -768,-1118,-1118,-1118},
08321 { -309, -598, -948, -948, -948},
08322 { -389, -678,-1028,-1028,-1028},
08323 { -379, -668,-1018,-1018,-1018}},
08324 /* AU.GC..GU */
08325 {{ DEF, -339, -689, -689, -689},
08326 { -649, -938,-1288,-1288,-1288},
08327 { -289, -578, -928, -928, -928},
08328 { -739,-1028,-1378,-1378,-1378},
08329 { -379, -668,-1018,-1018,-1018}},
08330 /* AU.GG..GU */
08331 {{ DEF, -339, -689, -689, -689},
08332 { -649, -938,-1288,-1288,-1288},
08333 { -289, -578, -928, -928, -928},
08334 { -739,-1028,-1378,-1378,-1378},
08335 { -379, -668,-1018,-1018,-1018}},
08336 /* AU.GU..GU */
08337 {{ DEF, -339, -689, -689, -689},
08338 { -649, -938,-1288,-1288,-1288},
08339 { -289, -578, -928, -928, -928},
08340 { -739,-1028,-1378,-1378,-1378},
08341 { -379, -668,-1018,-1018,-1018}}},
08342 /* AU.U@..GU */
08343 {{{ DEF, -329, -329, -329, -329},
08344 { -100, -379, -379, -379, -379},
08345 { -100, -379, -379, -379, -379},
08346 { -100, -379, -379, -379, -379},
08347 { -100, -379, -379, -379, -379}},
08348 /* AU.UA..GU */
08349 {{ DEF, -329, -329, -329, -329},
08350 { -479, -758, -758, -758, -758},
08351 { -309, -588, -588, -588, -588},
08352 { -389, -668, -668, -668, -668},
08353 { -379, -658, -658, -658, -658}},
08354 /* AU.UC..GU */
08355 {{ DEF, -329, -329, -329, -329},
08356 { -649, -928, -928, -928, -928},
08357 { -289, -568, -568, -568, -568},
08358 { -739,-1018,-1018,-1018,-1018},
```

```

08359 { -379, -658, -658, -658, -658}},
08360 /* AU.UG..GU */
08361 {{ DEF, -329, -329, -329, -329},
08362 { -649, -928, -928, -928, -928},
08363 { -289, -568, -568, -568, -568},
08364 { -739, -1018, -1018, -1018, -1018},
08365 { -379, -658, -658, -658, -658}},
08366 /* AU.UU..GU */
08367 {{ DEF, -329, -329, -329, -329},
08368 { -649, -928, -928, -928, -928},
08369 { -289, -568, -568, -568, -568},
08370 { -739, -1018, -1018, -1018, -1018},
08371 { -379, -658, -658, -658, -658}}}},
08372 /* AU.@@..UG */
08373 {{{ 0, 0, 0, 0, 0},
08374 { DEF, DEF, DEF, DEF, DEF},
08375 { DEF, DEF, DEF, DEF, DEF},
08376 { DEF, DEF, DEF, DEF, DEF},
08377 { DEF, DEF, DEF, DEF, DEF}},
08378 /* AU.@A..UG */
08379 {{ 0, 0, 0, 0, 0},
08380 { -719, -719, -719, -719, -719},
08381 { -479, -479, -479, -479, -479},
08382 { -659, -659, -659, -659, -659},
08383 { -549, -549, -549, -549, -549}},
08384 /* AU.@C..UG */
08385 {{ 0, 0, 0, 0, 0},
08386 { -789, -789, -789, -789, -789},
08387 { -479, -479, -479, -479, -479},
08388 { -809, -809, -809, -809, -809},
08389 { -439, -439, -439, -439, -439}},
08390 /* AU.@G..UG */
08391 {{ 0, 0, 0, 0, 0},
08392 { -959, -959, -959, -959, -959},
08393 { -359, -359, -359, -359, -359},
08394 { -919, -919, -919, -919, -919},
08395 { -549, -549, -549, -549, -549}},
08396 /* AU.@U..UG */
08397 {{ 0, 0, 0, 0, 0},
08398 { -809, -809, -809, -809, -809},
08399 { -479, -479, -479, -479, -479},
08400 { -809, -809, -809, -809, -809},
08401 { -359, -359, -359, -359, -359}}}},
08402 /* AU.A@..UG */
08403 {{{ DEF, -429, -599, -599, -599},
08404 { -100, -479, -649, -649, -649},
08405 { -100, -479, -649, -649, -649},
08406 { -100, -479, -649, -649, -649},
08407 { -100, -479, -649, -649, -649}},
08408 /* AU.AA..UG */
08409 {{ DEF, -429, -599, -599, -599},
08410 { -769, -1148, -1318, -1318, -1318},
08411 { -529, -908, -1078, -1078, -1078},
08412 { -709, -1088, -1258, -1258, -1258},
08413 { -599, -978, -1148, -1148, -1148}},
08414 /* AU.AC..UG */
08415 {{ DEF, -429, -599, -599, -599},
08416 { -839, -1218, -1388, -1388, -1388},
08417 { -529, -908, -1078, -1078, -1078},
08418 { -859, -1238, -1408, -1408, -1408},
08419 { -489, -868, -1038, -1038, -1038}},
08420 /* AU.AG..UG */
08421 {{ DEF, -429, -599, -599, -599},
08422 { -1009, -1388, -1558, -1558, -1558},
08423 { -409, -788, -958, -958, -958},
08424 { -969, -1348, -1518, -1518, -1518},
08425 { -599, -978, -1148, -1148, -1148}},
08426 /* AU.AU..UG */
08427 {{ DEF, -429, -599, -599, -599},
08428 { -859, -1238, -1408, -1408, -1408},
08429 { -529, -908, -1078, -1078, -1078},
08430 { -859, -1238, -1408, -1408, -1408},
08431 { -409, -788, -958, -958, -958}}}},
08432 /* AU.C@..UG */
08433 {{{ DEF, -259, -239, -239, -239},
08434 { -100, -309, -289, -289, -289},
08435 { -100, -309, -289, -289, -289},
08436 { -100, -309, -289, -289, -289},
08437 { -100, -309, -289, -289, -289}},
08438 /* AU.CA..UG */
08439 {{ DEF, -259, -239, -239, -239},
08440 { -769, -978, -958, -958, -958},
08441 { -529, -738, -718, -718, -718},
08442 { -709, -918, -898, -898, -898},
08443 { -599, -808, -788, -788, -788}},
08444 /* AU.CC..UG */
08445 {{ DEF, -259, -239, -239, -239},

```

```

08446 { -839,-1048,-1028,-1028,-1028},
08447 { -529, -738, -718, -718, -718},
08448 { -859,-1068,-1048,-1048,-1048},
08449 { -489, -698, -678, -678, -678}},
08450 /* AU.CG..UG */
08451 {{ DEF, -259, -239, -239, -239},
08452 {-1009,-1218,-1198,-1198,-1198},
08453 { -409, -618, -598, -598, -598},
08454 { -969,-1178,-1158,-1158,-1158},
08455 { -599, -808, -788, -788, -788}},
08456 /* AU.CU..UG */
08457 {{ DEF, -259, -239, -239, -239},
08458 { -859,-1068,-1048,-1048,-1048},
08459 { -529, -738, -718, -718, -718},
08460 { -859,-1068,-1048,-1048,-1048},
08461 { -409, -618, -598, -598, -598}}},
08462 /* AU.G@..UG */
08463 {{{ DEF, -339, -689, -689, -689},
08464 { -100, -389, -739, -739, -739},
08465 { -100, -389, -739, -739, -739},
08466 { -100, -389, -739, -739, -739},
08467 { -100, -389, -739, -739, -739}}},
08468 /* AU.GA..UG */
08469 {{ DEF, -339, -689, -689, -689},
08470 { -769,-1058,-1408,-1408,-1408},
08471 { -529, -818,-1168,-1168,-1168},
08472 { -709, -998,-1348,-1348,-1348},
08473 { -599, -888,-1238,-1238,-1238}},
08474 /* AU.GC..UG */
08475 {{ DEF, -339, -689, -689, -689},
08476 { -839,-1128,-1478,-1478,-1478},
08477 { -529, -818,-1168,-1168,-1168},
08478 { -859,-1148,-1498,-1498,-1498},
08479 { -489, -778,-1128,-1128,-1128}},
08480 /* AU.GG..UG */
08481 {{ DEF, -339, -689, -689, -689},
08482 {-1009,-1298,-1648,-1648,-1648},
08483 { -409, -698,-1048,-1048,-1048},
08484 { -969,-1258,-1608,-1608,-1608},
08485 { -599, -888,-1238,-1238,-1238}},
08486 /* AU.GU..UG */
08487 {{ DEF, -339, -689, -689, -689},
08488 { -859,-1148,-1498,-1498,-1498},
08489 { -529, -818,-1168,-1168,-1168},
08490 { -859,-1148,-1498,-1498,-1498},
08491 { -409, -698,-1048,-1048,-1048}}},
08492 /* AU.U@..UG */
08493 {{{ DEF, -329, -329, -329, -329},
08494 { -100, -379, -379, -379, -379},
08495 { -100, -379, -379, -379, -379},
08496 { -100, -379, -379, -379, -379},
08497 { -100, -379, -379, -379, -379}}},
08498 /* AU.UA..UG */
08499 {{ DEF, -329, -329, -329, -329},
08500 { -769,-1048,-1048,-1048,-1048},
08501 { -529, -808, -808, -808, -808},
08502 { -709, -988, -988, -988, -988},
08503 { -599, -878, -878, -878, -878}},
08504 /* AU.UC..UG */
08505 {{ DEF, -329, -329, -329, -329},
08506 { -839,-1118,-1118,-1118,-1118},
08507 { -529, -808, -808, -808, -808},
08508 { -859,-1138,-1138,-1138,-1138},
08509 { -489, -768, -768, -768, -768}},
08510 /* AU.UG..UG */
08511 {{ DEF, -329, -329, -329, -329},
08512 {-1009,-1288,-1288,-1288,-1288},
08513 { -409, -688, -688, -688, -688},
08514 { -969,-1248,-1248,-1248,-1248},
08515 { -599, -878, -878, -878, -878}},
08516 /* AU.UU..UG */
08517 {{ DEF, -329, -329, -329, -329},
08518 { -859,-1138,-1138,-1138,-1138},
08519 { -529, -808, -808, -808, -808},
08520 { -859,-1138,-1138,-1138,-1138},
08521 { -409, -688, -688, -688, -688}}}},
08522 /* AU.@@..AU */
08523 {{{{ 0, 0, 0, 0, 0},
08524 { DEF, DEF, DEF, DEF, DEF},
08525 { DEF, DEF, DEF, DEF, DEF},
08526 { DEF, DEF, DEF, DEF, DEF},
08527 { DEF, DEF, DEF, DEF, DEF}},
08528 /* AU.@A..AU */
08529 {{ 0, 0, 0, 0, 0},
08530 { -429, -429, -429, -429, -429},
08531 { -259, -259, -259, -259, -259},
08532 { -339, -339, -339, -339, -339},

```

```

08533 { -329, -329, -329, -329, -329}},
08534 /* AU.@C..AU */
08535 {{ 0, 0, 0, 0, 0},
08536 { -599, -599, -599, -599, -599}},
08537 { -239, -239, -239, -239, -239}},
08538 { -689, -689, -689, -689, -689}},
08539 { -329, -329, -329, -329, -329}},
08540 /* AU.@G..AU */
08541 {{ 0, 0, 0, 0, 0},
08542 { -599, -599, -599, -599, -599}},
08543 { -239, -239, -239, -239, -239}},
08544 { -689, -689, -689, -689, -689}},
08545 { -329, -329, -329, -329, -329}},
08546 /* AU.@U..AU */
08547 {{ 0, 0, 0, 0, 0},
08548 { -599, -599, -599, -599, -599}},
08549 { -239, -239, -239, -239, -239}},
08550 { -689, -689, -689, -689, -689}},
08551 { -329, -329, -329, -329, -329}}},
08552 /* AU.A@..AU */
08553 {{{ DEF, -429, -599, -599, -599}},
08554 { -100, -479, -649, -649, -649}},
08555 { -100, -479, -649, -649, -649}},
08556 { -100, -479, -649, -649, -649}},
08557 { -100, -479, -649, -649, -649}}},
08558 /* AU.AA..AU */
08559 {{ DEF, -429, -599, -599, -599}},
08560 { -479, -858, -1028, -1028, -1028}},
08561 { -309, -688, -858, -858, -858}},
08562 { -389, -768, -938, -938, -938}},
08563 { -379, -758, -928, -928, -928}}},
08564 /* AU.AC..AU */
08565 {{ DEF, -429, -599, -599, -599}},
08566 { -649, -1028, -1198, -1198, -1198}},
08567 { -289, -668, -838, -838, -838}},
08568 { -739, -1118, -1288, -1288, -1288}},
08569 { -379, -758, -928, -928, -928}}},
08570 /* AU.AG..AU */
08571 {{ DEF, -429, -599, -599, -599}},
08572 { -649, -1028, -1198, -1198, -1198}},
08573 { -289, -668, -838, -838, -838}},
08574 { -739, -1118, -1288, -1288, -1288}},
08575 { -379, -758, -928, -928, -928}}},
08576 /* AU.AU..AU */
08577 {{ DEF, -429, -599, -599, -599}},
08578 { -649, -1028, -1198, -1198, -1198}},
08579 { -289, -668, -838, -838, -838}},
08580 { -739, -1118, -1288, -1288, -1288}},
08581 { -379, -758, -928, -928, -928}}},
08582 /* AU.C@..AU */
08583 {{{ DEF, -259, -239, -239, -239}},
08584 { -100, -309, -289, -289, -289}},
08585 { -100, -309, -289, -289, -289}},
08586 { -100, -309, -289, -289, -289}},
08587 { -100, -309, -289, -289, -289}}},
08588 /* AU.CA..AU */
08589 {{ DEF, -259, -239, -239, -239}},
08590 { -479, -688, -668, -668, -668}},
08591 { -309, -518, -498, -498, -498}},
08592 { -389, -598, -578, -578, -578}},
08593 { -379, -588, -568, -568, -568}}},
08594 /* AU.CC..AU */
08595 {{ DEF, -259, -239, -239, -239}},
08596 { -649, -858, -838, -838, -838}},
08597 { -289, -498, -478, -478, -478}},
08598 { -739, -948, -928, -928, -928}},
08599 { -379, -588, -568, -568, -568}}},
08600 /* AU.CG..AU */
08601 {{ DEF, -259, -239, -239, -239}},
08602 { -649, -858, -838, -838, -838}},
08603 { -289, -498, -478, -478, -478}},
08604 { -739, -948, -928, -928, -928}},
08605 { -379, -588, -568, -568, -568}}},
08606 /* AU.CU..AU */
08607 {{ DEF, -259, -239, -239, -239}},
08608 { -649, -858, -838, -838, -838}},
08609 { -289, -498, -478, -478, -478}},
08610 { -739, -948, -928, -928, -928}},
08611 { -379, -588, -568, -568, -568}}},
08612 /* AU.G@..AU */
08613 {{{ DEF, -339, -689, -689, -689}},
08614 { -100, -389, -739, -739, -739}},
08615 { -100, -389, -739, -739, -739}},
08616 { -100, -389, -739, -739, -739}},
08617 { -100, -389, -739, -739, -739}}},
08618 /* AU.GA..AU */
08619 {{ DEF, -339, -689, -689, -689}},

```



```

08620 { -479, -768, -1118, -1118, -1118},
08621 { -309, -598, -948, -948, -948},
08622 { -389, -678, -1028, -1028, -1028},
08623 { -379, -668, -1018, -1018, -1018}},
08624 /* AU.GC..AU */
08625 {{ DEF, -339, -689, -689, -689},
08626 { -649, -938, -1288, -1288, -1288},
08627 { -289, -578, -928, -928, -928},
08628 { -739, -1028, -1378, -1378, -1378},
08629 { -379, -668, -1018, -1018, -1018}},
08630 /* AU.GG..AU */
08631 {{ DEF, -339, -689, -689, -689},
08632 { -649, -938, -1288, -1288, -1288},
08633 { -289, -578, -928, -928, -928},
08634 { -739, -1028, -1378, -1378, -1378},
08635 { -379, -668, -1018, -1018, -1018}},
08636 /* AU.GU..AU */
08637 {{ DEF, -339, -689, -689, -689},
08638 { -649, -938, -1288, -1288, -1288},
08639 { -289, -578, -928, -928, -928},
08640 { -739, -1028, -1378, -1378, -1378},
08641 { -379, -668, -1018, -1018, -1018}}},
08642 /* AU.U@..AU */
08643 {{{ DEF, -329, -329, -329, -329},
08644 { -100, -379, -379, -379, -379},
08645 { -100, -379, -379, -379, -379},
08646 { -100, -379, -379, -379, -379},
08647 { -100, -379, -379, -379, -379}}},
08648 /* AU.UA..AU */
08649 {{ DEF, -329, -329, -329, -329},
08650 { -479, -758, -758, -758, -758},
08651 { -309, -588, -588, -588, -588},
08652 { -389, -668, -668, -668, -668},
08653 { -379, -658, -658, -658, -658}},
08654 /* AU.UC..AU */
08655 {{ DEF, -329, -329, -329, -329},
08656 { -649, -928, -928, -928, -928},
08657 { -289, -568, -568, -568, -568},
08658 { -739, -1018, -1018, -1018, -1018},
08659 { -379, -658, -658, -658, -658}},
08660 /* AU.UG..AU */
08661 {{ DEF, -329, -329, -329, -329},
08662 { -649, -928, -928, -928, -928},
08663 { -289, -568, -568, -568, -568},
08664 { -739, -1018, -1018, -1018, -1018},
08665 { -379, -658, -658, -658, -658}},
08666 /* AU.UU..AU */
08667 {{ DEF, -329, -329, -329, -329},
08668 { -649, -928, -928, -928, -928},
08669 { -289, -568, -568, -568, -568},
08670 { -739, -1018, -1018, -1018, -1018},
08671 { -379, -658, -658, -658, -658}}}},
08672 /* AU.@@..UA */
08673 {{{{ 0, 0, 0, 0, 0},
08674 { DEF, DEF, DEF, DEF, DEF},
08675 { DEF, DEF, DEF, DEF, DEF},
08676 { DEF, DEF, DEF, DEF, DEF},
08677 { DEF, DEF, DEF, DEF, DEF}}}},
08678 /* AU.@A..UA */
08679 {{{ 0, 0, 0, 0, 0},
08680 { -399, -399, -399, -399, -399},
08681 { -429, -429, -429, -429, -429},
08682 { -379, -379, -379, -379, -379},
08683 { -279, -279, -279, -279, -279}}},
08684 /* AU.@C..UA */
08685 {{{ 0, 0, 0, 0, 0},
08686 { -629, -629, -629, -629, -629},
08687 { -509, -509, -509, -509, -509},
08688 { -679, -679, -679, -679, -679},
08689 { -139, -139, -139, -139, -139}}},
08690 /* AU.@G..UA */
08691 {{{ 0, 0, 0, 0, 0},
08692 { -889, -889, -889, -889, -889},
08693 { -199, -199, -199, -199, -199},
08694 { -889, -889, -889, -889, -889},
08695 { -279, -279, -279, -279, -279}}},
08696 /* AU.@U..UA */
08697 {{{ 0, 0, 0, 0, 0},
08698 { -589, -589, -589, -589, -589},
08699 { -179, -179, -179, -179, -179},
08700 { -679, -679, -679, -679, -679},
08701 { -140, -140, -140, -140, -140}}}},
08702 /* AU.A@..UA */
08703 {{{ DEF, -429, -599, -599, -599},
08704 { -100, -479, -649, -649, -649},
08705 { -100, -479, -649, -649, -649},
08706 { -100, -479, -649, -649, -649}},

```

```

08707 { -100, -479, -649, -649, -649}},
08708 /* AU.AA..UA */
08709 {{ DEF, -429, -599, -599, -599},
08710 { -449, -828, -998, -998, -998},
08711 { -479, -858, -1028, -1028, -1028},
08712 { -429, -808, -978, -978, -978},
08713 { -329, -708, -878, -878, -878}},
08714 /* AU.AC..UA */
08715 {{ DEF, -429, -599, -599, -599},
08716 { -679, -1058, -1228, -1228, -1228},
08717 { -559, -938, -1108, -1108, -1108},
08718 { -729, -1108, -1278, -1278, -1278},
08719 { -189, -568, -738, -738, -738}},
08720 /* AU.AG..UA */
08721 {{ DEF, -429, -599, -599, -599},
08722 { -939, -1318, -1488, -1488, -1488},
08723 { -249, -628, -798, -798, -798},
08724 { -939, -1318, -1488, -1488, -1488},
08725 { -329, -708, -878, -878, -878}},
08726 /* AU.AU..UA */
08727 {{ DEF, -429, -599, -599, -599},
08728 { -639, -1018, -1188, -1188, -1188},
08729 { -229, -608, -778, -778, -778},
08730 { -729, -1108, -1278, -1278, -1278},
08731 { -190, -569, -739, -739, -739}},
08732 /* AU.C@..UA */
08733 {{{ DEF, -259, -239, -239, -239},
08734 { -100, -309, -289, -289, -289},
08735 { -100, -309, -289, -289, -289},
08736 { -100, -309, -289, -289, -289},
08737 { -100, -309, -289, -289, -289}},
08738 /* AU.CA..UA */
08739 {{ DEF, -259, -239, -239, -239},
08740 { -449, -658, -638, -638, -638},
08741 { -479, -688, -668, -668, -668},
08742 { -429, -638, -618, -618, -618},
08743 { -329, -538, -518, -518, -518}},
08744 /* AU.CC..UA */
08745 {{ DEF, -259, -239, -239, -239},
08746 { -679, -888, -868, -868, -868},
08747 { -559, -768, -748, -748, -748},
08748 { -729, -938, -918, -918, -918},
08749 { -189, -398, -378, -378, -378}},
08750 /* AU.CG..UA */
08751 {{ DEF, -259, -239, -239, -239},
08752 { -939, -1148, -1128, -1128, -1128},
08753 { -249, -458, -438, -438, -438},
08754 { -939, -1148, -1128, -1128, -1128},
08755 { -329, -538, -518, -518, -518}},
08756 /* AU.CU..UA */
08757 {{ DEF, -259, -239, -239, -239},
08758 { -639, -848, -828, -828, -828},
08759 { -229, -438, -418, -418, -418},
08760 { -729, -938, -918, -918, -918},
08761 { -190, -399, -379, -379, -379}},
08762 /* AU.G@..UA */
08763 {{{ DEF, -339, -689, -689, -689},
08764 { -100, -389, -739, -739, -739},
08765 { -100, -389, -739, -739, -739},
08766 { -100, -389, -739, -739, -739},
08767 { -100, -389, -739, -739, -739}},
08768 /* AU.GA..UA */
08769 {{ DEF, -339, -689, -689, -689},
08770 { -449, -738, -1088, -1088, -1088},
08771 { -479, -768, -1118, -1118, -1118},
08772 { -429, -718, -1068, -1068, -1068},
08773 { -329, -618, -968, -968, -968}},
08774 /* AU.GC..UA */
08775 {{ DEF, -339, -689, -689, -689},
08776 { -679, -968, -1318, -1318, -1318},
08777 { -559, -848, -1198, -1198, -1198},
08778 { -729, -1018, -1368, -1368, -1368},
08779 { -189, -478, -828, -828, -828}},
08780 /* AU.GG..UA */
08781 {{ DEF, -339, -689, -689, -689},
08782 { -939, -1228, -1578, -1578, -1578},
08783 { -249, -538, -888, -888, -888},
08784 { -939, -1228, -1578, -1578, -1578},
08785 { -329, -618, -968, -968, -968}},
08786 /* AU.GU..UA */
08787 {{ DEF, -339, -689, -689, -689},
08788 { -639, -928, -1278, -1278, -1278},
08789 { -229, -518, -868, -868, -868},
08790 { -729, -1018, -1368, -1368, -1368},
08791 { -190, -479, -829, -829, -829}},
08792 /* AU.U@..UA */
08793 {{{ DEF, -329, -329, -329, -329},

```

```
08794 { -100, -379, -379, -379, -379},
08795 { -100, -379, -379, -379, -379},
08796 { -100, -379, -379, -379, -379},
08797 { -100, -379, -379, -379, -379}},
08798 /* AU.UA..UA */
08799 {{ DEF, -329, -329, -329, -329},
08800 { -449, -728, -728, -728, -728},
08801 { -479, -758, -758, -758, -758},
08802 { -429, -708, -708, -708, -708},
08803 { -329, -608, -608, -608, -608}},
08804 /* AU.UC..UA */
08805 {{ DEF, -329, -329, -329, -329},
08806 { -679, -958, -958, -958, -958},
08807 { -559, -838, -838, -838, -838},
08808 { -729, -1008, -1008, -1008, -1008},
08809 { -189, -468, -468, -468, -468}},
08810 /* AU.UG..UA */
08811 {{ DEF, -329, -329, -329, -329},
08812 { -939, -1218, -1218, -1218, -1218},
08813 { -249, -528, -528, -528, -528},
08814 { -939, -1218, -1218, -1218, -1218},
08815 { -329, -608, -608, -608, -608}},
08816 /* AU.UU..UA */
08817 {{ DEF, -329, -329, -329, -329},
08818 { -639, -918, -918, -918, -918},
08819 { -229, -508, -508, -508, -508},
08820 { -729, -1008, -1008, -1008, -1008},
08821 { -190, -469, -469, -469, -469}}}},
08822 /* AU.@@.. @ */
08823 {{{ DEF, DEF, DEF, DEF, DEF},
08824 { DEF, DEF, DEF, DEF, DEF},
08825 { DEF, DEF, DEF, DEF, DEF},
08826 { DEF, DEF, DEF, DEF, DEF},
08827 { DEF, DEF, DEF, DEF, DEF}},
08828 /* AU.@A.. @ */
08829 {{ DEF, DEF, DEF, DEF, DEF},
08830 { DEF, DEF, DEF, DEF, DEF},
08831 { DEF, DEF, DEF, DEF, DEF},
08832 { DEF, DEF, DEF, DEF, DEF},
08833 { DEF, DEF, DEF, DEF, DEF}},
08834 /* AU.@C.. @ */
08835 {{ DEF, DEF, DEF, DEF, DEF},
08836 { DEF, DEF, DEF, DEF, DEF},
08837 { DEF, DEF, DEF, DEF, DEF},
08838 { DEF, DEF, DEF, DEF, DEF},
08839 { DEF, DEF, DEF, DEF, DEF}},
08840 /* AU.@G.. @ */
08841 {{ DEF, DEF, DEF, DEF, DEF},
08842 { DEF, DEF, DEF, DEF, DEF},
08843 { DEF, DEF, DEF, DEF, DEF},
08844 { DEF, DEF, DEF, DEF, DEF},
08845 { DEF, DEF, DEF, DEF, DEF}},
08846 /* AU.@U.. @ */
08847 {{ DEF, DEF, DEF, DEF, DEF},
08848 { DEF, DEF, DEF, DEF, DEF},
08849 { DEF, DEF, DEF, DEF, DEF},
08850 { DEF, DEF, DEF, DEF, DEF},
08851 { DEF, DEF, DEF, DEF, DEF}}}},
08852 /* AU.A@.. @ */
08853 {{{ -100, -479, -649, -649, -649},
08854 { -100, -479, -649, -649, -649},
08855 { -100, -479, -649, -649, -649},
08856 { -100, -479, -649, -649, -649},
08857 { -100, -479, -649, -649, -649}},
08858 /* AU.AA.. @ */
08859 {{ -100, -479, -649, -649, -649},
08860 { -100, -479, -649, -649, -649},
08861 { -100, -479, -649, -649, -649},
08862 { -100, -479, -649, -649, -649},
08863 { -100, -479, -649, -649, -649}},
08864 /* AU.AC.. @ */
08865 {{ -100, -479, -649, -649, -649},
08866 { -100, -479, -649, -649, -649},
08867 { -100, -479, -649, -649, -649},
08868 { -100, -479, -649, -649, -649},
08869 { -100, -479, -649, -649, -649}},
08870 /* AU.AG.. @ */
08871 {{ -100, -479, -649, -649, -649},
08872 { -100, -479, -649, -649, -649},
08873 { -100, -479, -649, -649, -649},
08874 { -100, -479, -649, -649, -649},
08875 { -100, -479, -649, -649, -649}},
08876 /* AU.AU.. @ */
08877 {{ -100, -479, -649, -649, -649},
08878 { -100, -479, -649, -649, -649},
08879 { -100, -479, -649, -649, -649},
08880 { -100, -479, -649, -649, -649},
```

```
08881 { -100, -479, -649, -649, -649}},
08882 /* AU.C@.. @ */
08883 {{{ -100, -309, -289, -289, -289},
08884 { -100, -309, -289, -289, -289},
08885 { -100, -309, -289, -289, -289},
08886 { -100, -309, -289, -289, -289},
08887 { -100, -309, -289, -289, -289}},
08888 /* AU.CA.. @ */
08889 {{{ -100, -309, -289, -289, -289},
08890 { -100, -309, -289, -289, -289},
08891 { -100, -309, -289, -289, -289},
08892 { -100, -309, -289, -289, -289},
08893 { -100, -309, -289, -289, -289}},
08894 /* AU.CC.. @ */
08895 {{{ -100, -309, -289, -289, -289},
08896 { -100, -309, -289, -289, -289},
08897 { -100, -309, -289, -289, -289},
08898 { -100, -309, -289, -289, -289},
08899 { -100, -309, -289, -289, -289}},
08900 /* AU.CG.. @ */
08901 {{{ -100, -309, -289, -289, -289},
08902 { -100, -309, -289, -289, -289},
08903 { -100, -309, -289, -289, -289},
08904 { -100, -309, -289, -289, -289},
08905 { -100, -309, -289, -289, -289}},
08906 /* AU.CU.. @ */
08907 {{{ -100, -309, -289, -289, -289},
08908 { -100, -309, -289, -289, -289},
08909 { -100, -309, -289, -289, -289},
08910 { -100, -309, -289, -289, -289},
08911 { -100, -309, -289, -289, -289}}},
08912 /* AU.G@.. @ */
08913 {{{ -100, -389, -739, -739, -739},
08914 { -100, -389, -739, -739, -739},
08915 { -100, -389, -739, -739, -739},
08916 { -100, -389, -739, -739, -739},
08917 { -100, -389, -739, -739, -739}},
08918 /* AU.GA.. @ */
08919 {{{ -100, -389, -739, -739, -739},
08920 { -100, -389, -739, -739, -739},
08921 { -100, -389, -739, -739, -739},
08922 { -100, -389, -739, -739, -739},
08923 { -100, -389, -739, -739, -739}},
08924 /* AU.GC.. @ */
08925 {{{ -100, -389, -739, -739, -739},
08926 { -100, -389, -739, -739, -739},
08927 { -100, -389, -739, -739, -739},
08928 { -100, -389, -739, -739, -739},
08929 { -100, -389, -739, -739, -739}},
08930 /* AU.GG.. @ */
08931 {{{ -100, -389, -739, -739, -739},
08932 { -100, -389, -739, -739, -739},
08933 { -100, -389, -739, -739, -739},
08934 { -100, -389, -739, -739, -739},
08935 { -100, -389, -739, -739, -739}},
08936 /* AU.GU.. @ */
08937 {{{ -100, -389, -739, -739, -739},
08938 { -100, -389, -739, -739, -739},
08939 { -100, -389, -739, -739, -739},
08940 { -100, -389, -739, -739, -739},
08941 { -100, -389, -739, -739, -739}}},
08942 /* AU.U@.. @ */
08943 {{{ -100, -379, -379, -379, -379},
08944 { -100, -379, -379, -379, -379},
08945 { -100, -379, -379, -379, -379},
08946 { -100, -379, -379, -379, -379},
08947 { -100, -379, -379, -379, -379}},
08948 /* AU.UA.. @ */
08949 {{{ -100, -379, -379, -379, -379},
08950 { -100, -379, -379, -379, -379},
08951 { -100, -379, -379, -379, -379},
08952 { -100, -379, -379, -379, -379},
08953 { -100, -379, -379, -379, -379}},
08954 /* AU.UC.. @ */
08955 {{{ -100, -379, -379, -379, -379},
08956 { -100, -379, -379, -379, -379},
08957 { -100, -379, -379, -379, -379},
08958 { -100, -379, -379, -379, -379},
08959 { -100, -379, -379, -379, -379}},
08960 /* AU.UG.. @ */
08961 {{{ -100, -379, -379, -379, -379},
08962 { -100, -379, -379, -379, -379},
08963 { -100, -379, -379, -379, -379},
08964 { -100, -379, -379, -379, -379},
08965 { -100, -379, -379, -379, -379}},
08966 /* AU.UU.. @ */
08967 {{{ -100, -379, -379, -379, -379},
```

```

08968 { -100, -379, -379, -379, -379},
08969 { -100, -379, -379, -379, -379},
08970 { -100, -379, -379, -379, -379},
08971 { -100, -379, -379, -379, -379}}}},
08972 { /* noPair */ {{{0}}}},
08973 /* UA.@@..CG */
08974 {{{ 0, 0, 0, 0, 0},
08975 { DEF, DEF, DEF, DEF, DEF},
08976 { DEF, DEF, DEF, DEF, DEF},
08977 { DEF, DEF, DEF, DEF, DEF},
08978 { DEF, DEF, DEF, DEF, DEF}},
08979 /* UA.@A..CG */
08980 {{ 0, 0, 0, 0, 0},
08981 {-1029,-1029,-1029,-1029,-1029},
08982 { -519, -519, -519, -519, -519},
08983 { -939, -939, -939, -939, -939},
08984 { -809, -809, -809, -809, -809}},
08985 /* UA.@C..CG */
08986 {{ 0, 0, 0, 0, 0},
08987 { -949, -949, -949, -949, -949},
08988 { -449, -449, -449, -449, -449},
08989 { -939, -939, -939, -939, -939},
08990 { -739, -739, -739, -739, -739}},
08991 /* UA.@G..CG */
08992 {{ 0, 0, 0, 0, 0},
08993 {-1029,-1029,-1029,-1029,-1029},
08994 { -519, -519, -519, -519, -519},
08995 { -939, -939, -939, -939, -939},
08996 { -809, -809, -809, -809, -809}},
08997 /* UA.@U..CG */
08998 {{ 0, 0, 0, 0, 0},
08999 {-1029,-1029,-1029,-1029,-1029},
09000 { -669, -669, -669, -669, -669},
09001 { -939, -939, -939, -939, -939},
09002 { -859, -859, -859, -859, -859}}},
09003 /* UA.A@..CG */
09004 {{{ DEF, -399, -629, -889, -589},
09005 { -100, -449, -679, -939, -639},
09006 { -100, -449, -679, -939, -639},
09007 { -100, -449, -679, -939, -639},
09008 { -100, -449, -679, -939, -639}},
09009 /* UA.AA..CG */
09010 {{ DEF, -399, -629, -889, -589},
09011 {-1079,-1428,-1658,-1918,-1618},
09012 { -569, -918,-1148,-1408,-1108},
09013 { -989,-1338,-1568,-1828,-1528},
09014 { -859,-1208,-1438,-1698,-1398}},
09015 /* UA.AC..CG */
09016 {{ DEF, -399, -629, -889, -589},
09017 { -999,-1348,-1578,-1838,-1538},
09018 { -499, -848,-1078,-1338,-1038},
09019 { -989,-1338,-1568,-1828,-1528},
09020 { -789,-1138,-1368,-1628,-1328}},
09021 /* UA.AG..CG */
09022 {{ DEF, -399, -629, -889, -589},
09023 {-1079,-1428,-1658,-1918,-1618},
09024 { -569, -918,-1148,-1408,-1108},
09025 { -989,-1338,-1568,-1828,-1528},
09026 { -859,-1208,-1438,-1698,-1398}},
09027 /* UA.AU..CG */
09028 {{ DEF, -399, -629, -889, -589},
09029 {-1079,-1428,-1658,-1918,-1618},
09030 { -719,-1068,-1298,-1558,-1258},
09031 { -989,-1338,-1568,-1828,-1528},
09032 { -909,-1258,-1488,-1748,-1448}}},
09033 /* UA.C@..CG */
09034 {{{ DEF, -429, -509, -199, -179},
09035 { -100, -479, -559, -249, -229},
09036 { -100, -479, -559, -249, -229},
09037 { -100, -479, -559, -249, -229},
09038 { -100, -479, -559, -249, -229}},
09039 /* UA.CA..CG */
09040 {{ DEF, -429, -509, -199, -179},
09041 {-1079,-1458,-1538,-1228,-1208},
09042 { -569, -948,-1028, -718, -698},
09043 { -989,-1368,-1448,-1138,-1118},
09044 { -859,-1238,-1318,-1008, -988}},
09045 /* UA.CC..CG */
09046 {{ DEF, -429, -509, -199, -179},
09047 { -999,-1378,-1458,-1148,-1128},
09048 { -499, -878, -958, -648, -628},
09049 { -989,-1368,-1448,-1138,-1118},
09050 { -789,-1168,-1248, -938, -918}},
09051 /* UA.CG..CG */
09052 {{ DEF, -429, -509, -199, -179},
09053 {-1079,-1458,-1538,-1228,-1208},
09054 { -569, -948,-1028, -718, -698},

```

```

09055 { -989,-1368,-1448,-1138,-1118},
09056 { -859,-1238,-1318,-1008, -988}},
09057 /* UA.CU..CG */
09058 {{ DEF, -429, -509, -199, -179},
09059 {-1079,-1458,-1538,-1228,-1208},
09060 { -719,-1098,-1178, -868, -848},
09061 { -989,-1368,-1448,-1138,-1118},
09062 { -909,-1288,-1368,-1058,-1038}}},
09063 /* UA.G@..CG */
09064 {{{ DEF, -379, -679, -889, -679},
09065 { -100, -429, -729, -939, -729},
09066 { -100, -429, -729, -939, -729},
09067 { -100, -429, -729, -939, -729},
09068 { -100, -429, -729, -939, -729}}},
09069 /* UA.GA..CG */
09070 {{ DEF, -379, -679, -889, -679},
09071 {-1079,-1408,-1708,-1918,-1708},
09072 { -569, -898,-1198,-1408,-1198},
09073 { -989,-1318,-1618,-1828,-1618},
09074 { -859,-1188,-1488,-1698,-1488}},
09075 /* UA.GC..CG */
09076 {{ DEF, -379, -679, -889, -679},
09077 { -999,-1328,-1628,-1838,-1628},
09078 { -499, -828,-1128,-1338,-1128},
09079 { -989,-1318,-1618,-1828,-1618},
09080 { -789,-1118,-1418,-1628,-1418}}},
09081 /* UA.GG..CG */
09082 {{ DEF, -379, -679, -889, -679},
09083 {-1079,-1408,-1708,-1918,-1708},
09084 { -569, -898,-1198,-1408,-1198},
09085 { -989,-1318,-1618,-1828,-1618},
09086 { -859,-1188,-1488,-1698,-1488}},
09087 /* UA.GU..CG */
09088 {{ DEF, -379, -679, -889, -679},
09089 {-1079,-1408,-1708,-1918,-1708},
09090 { -719,-1048,-1348,-1558,-1348},
09091 { -989,-1318,-1618,-1828,-1618},
09092 { -909,-1238,-1538,-1748,-1538}}},
09093 /* UA.U@..CG */
09094 {{{ DEF, -279, -139, -279, -140},
09095 { -100, -329, -189, -329, -190},
09096 { -100, -329, -189, -329, -190},
09097 { -100, -329, -189, -329, -190},
09098 { -100, -329, -189, -329, -190}}},
09099 /* UA.UA..CG */
09100 {{ DEF, -279, -139, -279, -140},
09101 {-1079,-1308,-1168,-1308,-1169},
09102 { -569, -798, -658, -798, -659},
09103 { -989,-1218,-1078,-1218,-1079},
09104 { -859,-1088, -948,-1088, -949}},
09105 /* UA.UC..CG */
09106 {{ DEF, -279, -139, -279, -140},
09107 { -999,-1228,-1088,-1228,-1089},
09108 { -499, -728, -588, -728, -589},
09109 { -989,-1218,-1078,-1218,-1079},
09110 { -789,-1018, -878,-1018, -879}}},
09111 /* UA.UG..CG */
09112 {{ DEF, -279, -139, -279, -140},
09113 {-1079,-1308,-1168,-1308,-1169},
09114 { -569, -798, -658, -798, -659},
09115 { -989,-1218,-1078,-1218,-1079},
09116 { -859,-1088, -948,-1088, -949}},
09117 /* UA.UU..CG */
09118 {{ DEF, -279, -139, -279, -140},
09119 {-1079,-1308,-1168,-1308,-1169},
09120 { -719, -948, -808, -948, -809},
09121 { -989,-1218,-1078,-1218,-1079},
09122 { -909,-1138, -998,-1138, -999}}}},
09123 /* UA.@@..GC */
09124 {{{ 0, 0, 0, 0, 0},
09125 { DEF, DEF, DEF, DEF, DEF},
09126 { DEF, DEF, DEF, DEF, DEF},
09127 { DEF, DEF, DEF, DEF, DEF},
09128 { DEF, DEF, DEF, DEF, DEF}}},
09129 /* UA.@A..GC */
09130 {{ 0, 0, 0, 0, 0},
09131 { -519, -519, -519, -519, -519},
09132 { -719, -719, -719, -719, -719},
09133 { -709, -709, -709, -709, -709},
09134 { -499, -499, -499, -499, -499}},
09135 /* UA.@C..GC */
09136 {{ 0, 0, 0, 0, 0},
09137 { -879, -879, -879, -879, -879},
09138 { -309, -309, -309, -309, -309},
09139 { -739, -739, -739, -739, -739},
09140 { -499, -499, -499, -499, -499}},
09141 /* UA.@G..GC */

```

```

09142 {{ 0, 0, 0, 0, 0},
09143 { -559, -559, -559, -559, -559},
09144 { -309, -309, -309, -309, -309},
09145 { -619, -619, -619, -619, -619},
09146 { -499, -499, -499, -499, -499}},
09147 /* UA.@U..GC */
09148 {{ 0, 0, 0, 0, 0},
09149 { -879, -879, -879, -879, -879},
09150 { -389, -389, -389, -389, -389},
09151 { -739, -739, -739, -739, -739},
09152 { -569, -569, -569, -569, -569}},
09153 /* UA.A@..GC */
09154 {{{ DEF, -399, -629, -889, -589},
09155 { -100, -449, -679, -939, -639},
09156 { -100, -449, -679, -939, -639},
09157 { -100, -449, -679, -939, -639},
09158 { -100, -449, -679, -939, -639}},
09159 /* UA.AA..GC */
09160 {{ DEF, -399, -629, -889, -589},
09161 { -569, -918, -1148, -1408, -1108},
09162 { -769, -1118, -1348, -1608, -1308},
09163 { -759, -1108, -1338, -1598, -1298},
09164 { -549, -898, -1128, -1388, -1088}},
09165 /* UA.AC..GC */
09166 {{ DEF, -399, -629, -889, -589},
09167 { -929, -1278, -1508, -1768, -1468},
09168 { -359, -708, -938, -1198, -898},
09169 { -789, -1138, -1368, -1628, -1328},
09170 { -549, -898, -1128, -1388, -1088}},
09171 /* UA.AG..GC */
09172 {{ DEF, -399, -629, -889, -589},
09173 { -609, -958, -1188, -1448, -1148},
09174 { -359, -708, -938, -1198, -898},
09175 { -669, -1018, -1248, -1508, -1208},
09176 { -549, -898, -1128, -1388, -1088}},
09177 /* UA.AU..GC */
09178 {{ DEF, -399, -629, -889, -589},
09179 { -929, -1278, -1508, -1768, -1468},
09180 { -439, -788, -1018, -1278, -978},
09181 { -789, -1138, -1368, -1628, -1328},
09182 { -619, -968, -1198, -1458, -1158}},
09183 /* UA.C@..GC */
09184 {{{ DEF, -429, -509, -199, -179},
09185 { -100, -479, -559, -249, -229},
09186 { -100, -479, -559, -249, -229},
09187 { -100, -479, -559, -249, -229},
09188 { -100, -479, -559, -249, -229}},
09189 /* UA.CA..GC */
09190 {{ DEF, -429, -509, -199, -179},
09191 { -569, -948, -1028, -718, -698},
09192 { -769, -1148, -1228, -918, -898},
09193 { -759, -1138, -1218, -908, -888},
09194 { -549, -928, -1008, -698, -678}},
09195 /* UA.CC..GC */
09196 {{ DEF, -429, -509, -199, -179},
09197 { -929, -1308, -1388, -1078, -1058},
09198 { -359, -738, -818, -508, -488},
09199 { -789, -1168, -1248, -938, -918},
09200 { -549, -928, -1008, -698, -678}},
09201 /* UA.CG..GC */
09202 {{ DEF, -429, -509, -199, -179},
09203 { -609, -988, -1068, -758, -738},
09204 { -359, -738, -818, -508, -488},
09205 { -669, -1048, -1128, -818, -798},
09206 { -549, -928, -1008, -698, -678}},
09207 /* UA.CU..GC */
09208 {{ DEF, -429, -509, -199, -179},
09209 { -929, -1308, -1388, -1078, -1058},
09210 { -439, -818, -898, -588, -568},
09211 { -789, -1168, -1248, -938, -918},
09212 { -619, -998, -1078, -768, -748}},
09213 /* UA.G@..GC */
09214 {{{ DEF, -379, -679, -889, -679},
09215 { -100, -429, -729, -939, -729},
09216 { -100, -429, -729, -939, -729},
09217 { -100, -429, -729, -939, -729},
09218 { -100, -429, -729, -939, -729}},
09219 /* UA.GA..GC */
09220 {{ DEF, -379, -679, -889, -679},
09221 { -569, -898, -1198, -1408, -1198},
09222 { -769, -1098, -1398, -1608, -1398},
09223 { -759, -1088, -1388, -1598, -1388},
09224 { -549, -878, -1178, -1388, -1178}},
09225 /* UA.GC..GC */
09226 {{ DEF, -379, -679, -889, -679},
09227 { -929, -1258, -1558, -1768, -1558},
09228 { -359, -688, -988, -1198, -988},

```

```

09229 { -789,-1118,-1418,-1628,-1418},
09230 { -549, -878,-1178,-1388,-1178}},
09231 /* UA.GG..GC */
09232 {{ DEF, -379, -679, -889, -679},
09233 { -609, -938,-1238,-1448,-1238},
09234 { -359, -688, -988,-1198, -988},
09235 { -669, -998,-1298,-1508,-1298},
09236 { -549, -878,-1178,-1388,-1178}},
09237 /* UA.GU..GC */
09238 {{ DEF, -379, -679, -889, -679},
09239 { -929,-1258,-1558,-1768,-1558},
09240 { -439, -768,-1068,-1278,-1068},
09241 { -789,-1118,-1418,-1628,-1418},
09242 { -619, -948,-1248,-1458,-1248}}},
09243 /* UA.U@..GC */
09244 {{{ DEF, -279, -139, -279, -140},
09245 { -100, -329, -189, -329, -190},
09246 { -100, -329, -189, -329, -190},
09247 { -100, -329, -189, -329, -190},
09248 { -100, -329, -189, -329, -190}},
09249 /* UA.UA..GC */
09250 {{ DEF, -279, -139, -279, -140},
09251 { -569, -798, -658, -798, -659},
09252 { -769, -998, -858, -998, -859},
09253 { -759, -988, -848, -988, -849},
09254 { -549, -778, -638, -778, -639}},
09255 /* UA.UC..GC */
09256 {{ DEF, -279, -139, -279, -140},
09257 { -929,-1158,-1018,-1158,-1019},
09258 { -359, -588, -448, -588, -449},
09259 { -789,-1018, -878,-1018, -879},
09260 { -549, -778, -638, -778, -639}},
09261 /* UA.UG..GC */
09262 {{ DEF, -279, -139, -279, -140},
09263 { -609, -838, -698, -838, -699},
09264 { -359, -588, -448, -588, -449},
09265 { -669, -898, -758, -898, -759},
09266 { -549, -778, -638, -778, -639}},
09267 /* UA.UU..GC */
09268 {{ DEF, -279, -139, -279, -140},
09269 { -929,-1158,-1018,-1158,-1019},
09270 { -439, -668, -528, -668, -529},
09271 { -789,-1018, -878,-1018, -879},
09272 { -619, -848, -708, -848, -709}}}},
09273 /* UA.@@..GU */
09274 {{{ 0, 0, 0, 0, 0},
09275 { DEF, DEF, DEF, DEF, DEF},
09276 { DEF, DEF, DEF, DEF, DEF},
09277 { DEF, DEF, DEF, DEF, DEF},
09278 { DEF, DEF, DEF, DEF, DEF}},
09279 /* UA.@A..GU */
09280 {{ 0, 0, 0, 0, 0},
09281 { -429, -429, -429, -429, -429},
09282 { -259, -259, -259, -259, -259},
09283 { -339, -339, -339, -339, -339},
09284 { -329, -329, -329, -329, -329}},
09285 /* UA.@C..GU */
09286 {{ 0, 0, 0, 0, 0},
09287 { -599, -599, -599, -599, -599},
09288 { -239, -239, -239, -239, -239},
09289 { -689, -689, -689, -689, -689},
09290 { -329, -329, -329, -329, -329}},
09291 /* UA.@G..GU */
09292 {{ 0, 0, 0, 0, 0},
09293 { -599, -599, -599, -599, -599},
09294 { -239, -239, -239, -239, -239},
09295 { -689, -689, -689, -689, -689},
09296 { -329, -329, -329, -329, -329}},
09297 /* UA.@U..GU */
09298 {{ 0, 0, 0, 0, 0},
09299 { -599, -599, -599, -599, -599},
09300 { -239, -239, -239, -239, -239},
09301 { -689, -689, -689, -689, -689},
09302 { -329, -329, -329, -329, -329}}}},
09303 /* UA.A@..GU */
09304 {{{ DEF, -399, -629, -889, -589},
09305 { -100, -449, -679, -939, -639},
09306 { -100, -449, -679, -939, -639},
09307 { -100, -449, -679, -939, -639},
09308 { -100, -449, -679, -939, -639}},
09309 /* UA.AA..GU */
09310 {{ DEF, -399, -629, -889, -589},
09311 { -479, -828,-1058,-1318,-1018},
09312 { -309, -658, -888,-1148, -848},
09313 { -389, -738, -968,-1228, -928},
09314 { -379, -728, -958,-1218, -918}},
09315 /* UA.AC..GU */

```



```
09316 {{ DEF, -399, -629, -889, -589},
09317 { -649, -998,-1228,-1488,-1188},
09318 { -289, -638, -868,-1128, -828},
09319 { -739,-1088,-1318,-1578,-1278},
09320 { -379, -728, -958,-1218, -918}},
09321 /* UA.AG..GU */
09322 {{ DEF, -399, -629, -889, -589},
09323 { -649, -998,-1228,-1488,-1188},
09324 { -289, -638, -868,-1128, -828},
09325 { -739,-1088,-1318,-1578,-1278},
09326 { -379, -728, -958,-1218, -918}},
09327 /* UA.AU..GU */
09328 {{ DEF, -399, -629, -889, -589},
09329 { -649, -998,-1228,-1488,-1188},
09330 { -289, -638, -868,-1128, -828},
09331 { -739,-1088,-1318,-1578,-1278},
09332 { -379, -728, -958,-1218, -918}}},
09333 /* UA.C@..GU */
09334 {{{ DEF, -429, -509, -199, -179},
09335 { -100, -479, -559, -249, -229},
09336 { -100, -479, -559, -249, -229},
09337 { -100, -479, -559, -249, -229},
09338 { -100, -479, -559, -249, -229}},
09339 /* UA.CA..GU */
09340 {{ DEF, -429, -509, -199, -179},
09341 { -479, -858, -938, -628, -608},
09342 { -309, -688, -768, -458, -438},
09343 { -389, -768, -848, -538, -518},
09344 { -379, -758, -838, -528, -508}},
09345 /* UA.CC..GU */
09346 {{ DEF, -429, -509, -199, -179},
09347 { -649,-1028,-1108, -798, -778},
09348 { -289, -668, -748, -438, -418},
09349 { -739,-1118,-1198, -888, -868},
09350 { -379, -758, -838, -528, -508}},
09351 /* UA.CG..GU */
09352 {{ DEF, -429, -509, -199, -179},
09353 { -649,-1028,-1108, -798, -778},
09354 { -289, -668, -748, -438, -418},
09355 { -739,-1118,-1198, -888, -868},
09356 { -379, -758, -838, -528, -508}},
09357 /* UA.CU..GU */
09358 {{ DEF, -429, -509, -199, -179},
09359 { -649,-1028,-1108, -798, -778},
09360 { -289, -668, -748, -438, -418},
09361 { -739,-1118,-1198, -888, -868},
09362 { -379, -758, -838, -528, -508}}},
09363 /* UA.G@..GU */
09364 {{{ DEF, -379, -679, -889, -679},
09365 { -100, -429, -729, -939, -729},
09366 { -100, -429, -729, -939, -729},
09367 { -100, -429, -729, -939, -729},
09368 { -100, -429, -729, -939, -729}},
09369 /* UA.GA..GU */
09370 {{ DEF, -379, -679, -889, -679},
09371 { -479, -808,-1108,-1318,-1108},
09372 { -309, -638, -938,-1148, -938},
09373 { -389, -718,-1018,-1228,-1018},
09374 { -379, -708,-1008,-1218,-1008}},
09375 /* UA.GC..GU */
09376 {{ DEF, -379, -679, -889, -679},
09377 { -649, -978,-1278,-1488,-1278},
09378 { -289, -618, -918,-1128, -918},
09379 { -739,-1068,-1368,-1578,-1368},
09380 { -379, -708,-1008,-1218,-1008}},
09381 /* UA.GG..GU */
09382 {{ DEF, -379, -679, -889, -679},
09383 { -649, -978,-1278,-1488,-1278},
09384 { -289, -618, -918,-1128, -918},
09385 { -739,-1068,-1368,-1578,-1368},
09386 { -379, -708,-1008,-1218,-1008}},
09387 /* UA.GU..GU */
09388 {{ DEF, -379, -679, -889, -679},
09389 { -649, -978,-1278,-1488,-1278},
09390 { -289, -618, -918,-1128, -918},
09391 { -739,-1068,-1368,-1578,-1368},
09392 { -379, -708,-1008,-1218,-1008}}},
09393 /* UA.U@..GU */
09394 {{{ DEF, -279, -139, -279, -140},
09395 { -100, -329, -189, -329, -190},
09396 { -100, -329, -189, -329, -190},
09397 { -100, -329, -189, -329, -190},
09398 { -100, -329, -189, -329, -190}},
09399 /* UA.UA..GU */
09400 {{ DEF, -279, -139, -279, -140},
09401 { -479, -708, -568, -708, -569},
09402 { -309, -538, -398, -538, -399},
```

```

09403 { -389, -618, -478, -618, -479},
09404 { -379, -608, -468, -608, -469}},
09405 /* UA.UC..GU */
09406 {{ DEF, -279, -139, -279, -140},
09407 { -649, -878, -738, -878, -739},
09408 { -289, -518, -378, -518, -379},
09409 { -739, -968, -828, -968, -829},
09410 { -379, -608, -468, -608, -469}},
09411 /* UA.UG..GU */
09412 {{ DEF, -279, -139, -279, -140},
09413 { -649, -878, -738, -878, -739},
09414 { -289, -518, -378, -518, -379},
09415 { -739, -968, -828, -968, -829},
09416 { -379, -608, -468, -608, -469}},
09417 /* UA.UU..GU */
09418 {{ DEF, -279, -139, -279, -140},
09419 { -649, -878, -738, -878, -739},
09420 { -289, -518, -378, -518, -379},
09421 { -739, -968, -828, -968, -829},
09422 { -379, -608, -468, -608, -469}}}},
09423 /* UA.@@..UG */
09424 {{{ 0, 0, 0, 0, 0},
09425 { DEF, DEF, DEF, DEF, DEF},
09426 { DEF, DEF, DEF, DEF, DEF},
09427 { DEF, DEF, DEF, DEF, DEF},
09428 { DEF, DEF, DEF, DEF, DEF}},
09429 /* UA.@A..UG */
09430 {{ 0, 0, 0, 0, 0},
09431 { -719, -719, -719, -719, -719},
09432 { -479, -479, -479, -479, -479},
09433 { -659, -659, -659, -659, -659},
09434 { -549, -549, -549, -549, -549}},
09435 /* UA.@C..UG */
09436 {{ 0, 0, 0, 0, 0},
09437 { -789, -789, -789, -789, -789},
09438 { -479, -479, -479, -479, -479},
09439 { -809, -809, -809, -809, -809},
09440 { -439, -439, -439, -439, -439}},
09441 /* UA.@G..UG */
09442 {{ 0, 0, 0, 0, 0},
09443 { -959, -959, -959, -959, -959},
09444 { -359, -359, -359, -359, -359},
09445 { -919, -919, -919, -919, -919},
09446 { -549, -549, -549, -549, -549}},
09447 /* UA.@U..UG */
09448 {{ 0, 0, 0, 0, 0},
09449 { -809, -809, -809, -809, -809},
09450 { -479, -479, -479, -479, -479},
09451 { -809, -809, -809, -809, -809},
09452 { -359, -359, -359, -359, -359}}}},
09453 /* UA.A@..UG */
09454 {{{ DEF, -399, -629, -889, -589},
09455 { -100, -449, -679, -939, -639},
09456 { -100, -449, -679, -939, -639},
09457 { -100, -449, -679, -939, -639},
09458 { -100, -449, -679, -939, -639}},
09459 /* UA.AA..UG */
09460 {{ DEF, -399, -629, -889, -589},
09461 { -769, -1118, -1348, -1608, -1308},
09462 { -529, -878, -1108, -1368, -1068},
09463 { -709, -1058, -1288, -1548, -1248},
09464 { -599, -948, -1178, -1438, -1138}},
09465 /* UA.AC..UG */
09466 {{ DEF, -399, -629, -889, -589},
09467 { -839, -1188, -1418, -1678, -1378},
09468 { -529, -878, -1108, -1368, -1068},
09469 { -859, -1208, -1438, -1698, -1398},
09470 { -489, -838, -1068, -1328, -1028}},
09471 /* UA.AG..UG */
09472 {{ DEF, -399, -629, -889, -589},
09473 { -1009, -1358, -1588, -1848, -1548},
09474 { -409, -758, -988, -1248, -948},
09475 { -969, -1318, -1548, -1808, -1508},
09476 { -599, -948, -1178, -1438, -1138}},
09477 /* UA.AU..UG */
09478 {{ DEF, -399, -629, -889, -589},
09479 { -859, -1208, -1438, -1698, -1398},
09480 { -529, -878, -1108, -1368, -1068},
09481 { -859, -1208, -1438, -1698, -1398},
09482 { -409, -758, -988, -1248, -948}}}},
09483 /* UA.C@..UG */
09484 {{{ DEF, -429, -509, -199, -179},
09485 { -100, -479, -559, -249, -229},
09486 { -100, -479, -559, -249, -229},
09487 { -100, -479, -559, -249, -229},
09488 { -100, -479, -559, -249, -229}},
09489 /* UA.CA..UG */

```

```
09490 {{ DEF, -429, -509, -199, -179},
09491 { -769,-1148,-1228, -918, -898},
09492 { -529, -908, -988, -678, -658},
09493 { -709,-1088,-1168, -858, -838},
09494 { -599, -978,-1058, -748, -728}},
09495 /* UA.CC..UG */
09496 {{ DEF, -429, -509, -199, -179},
09497 { -839,-1218,-1298, -988, -968},
09498 { -529, -908, -988, -678, -658},
09499 { -859,-1238,-1318,-1008, -988},
09500 { -489, -868, -948, -638, -618}},
09501 /* UA.CG..UG */
09502 {{ DEF, -429, -509, -199, -179},
09503 {-1009,-1388,-1468,-1158,-1138},
09504 { -409, -788, -868, -558, -538},
09505 { -969,-1348,-1428,-1118,-1098},
09506 { -599, -978,-1058, -748, -728}},
09507 /* UA.CU..UG */
09508 {{ DEF, -429, -509, -199, -179},
09509 { -859,-1238,-1318,-1008, -988},
09510 { -529, -908, -988, -678, -658},
09511 { -859,-1238,-1318,-1008, -988},
09512 { -409, -788, -868, -558, -538}}},
09513 /* UA.G@..UG */
09514 {{{ DEF, -379, -679, -889, -679},
09515 { -100, -429, -729, -939, -729},
09516 { -100, -429, -729, -939, -729},
09517 { -100, -429, -729, -939, -729},
09518 { -100, -429, -729, -939, -729}},
09519 /* UA.GA..UG */
09520 {{ DEF, -379, -679, -889, -679},
09521 { -769,-1098,-1398,-1608,-1398},
09522 { -529, -858,-1158,-1368,-1158},
09523 { -709,-1038,-1338,-1548,-1338},
09524 { -599, -928,-1228,-1438,-1228}},
09525 /* UA.GC..UG */
09526 {{ DEF, -379, -679, -889, -679},
09527 { -839,-1168,-1468,-1678,-1468},
09528 { -529, -858,-1158,-1368,-1158},
09529 { -859,-1188,-1488,-1698,-1488},
09530 { -489, -818,-1118,-1328,-1118}},
09531 /* UA.GG..UG */
09532 {{ DEF, -379, -679, -889, -679},
09533 {-1009,-1338,-1638,-1848,-1638},
09534 { -409, -738,-1038,-1248,-1038},
09535 { -969,-1298,-1598,-1808,-1598},
09536 { -599, -928,-1228,-1438,-1228}},
09537 /* UA.GU..UG */
09538 {{ DEF, -379, -679, -889, -679},
09539 { -859,-1188,-1488,-1698,-1488},
09540 { -529, -858,-1158,-1368,-1158},
09541 { -859,-1188,-1488,-1698,-1488},
09542 { -409, -738,-1038,-1248,-1038}}},
09543 /* UA.U@..UG */
09544 {{{ DEF, -279, -139, -279, -140},
09545 { -100, -329, -189, -329, -190},
09546 { -100, -329, -189, -329, -190},
09547 { -100, -329, -189, -329, -190},
09548 { -100, -329, -189, -329, -190}},
09549 /* UA.UA..UG */
09550 {{ DEF, -279, -139, -279, -140},
09551 { -769, -998, -858, -998, -859},
09552 { -529, -758, -618, -758, -619},
09553 { -709, -938, -798, -938, -799},
09554 { -599, -828, -688, -828, -689}},
09555 /* UA.UC..UG */
09556 {{ DEF, -279, -139, -279, -140},
09557 { -839,-1068, -928,-1068, -929},
09558 { -529, -758, -618, -758, -619},
09559 { -859,-1088, -948,-1088, -949},
09560 { -489, -718, -578, -718, -579}},
09561 /* UA.UG..UG */
09562 {{ DEF, -279, -139, -279, -140},
09563 {-1009,-1238,-1098,-1238,-1099},
09564 { -409, -638, -498, -638, -499},
09565 { -969,-1198,-1058,-1198,-1059},
09566 { -599, -828, -688, -828, -689}},
09567 /* UA.UU..UG */
09568 {{ DEF, -279, -139, -279, -140},
09569 { -859,-1088, -948,-1088, -949},
09570 { -529, -758, -618, -758, -619},
09571 { -859,-1088, -948,-1088, -949},
09572 { -409, -638, -498, -638, -499}}}},
09573 /* UA.@@..AU */
09574 {{{ 0, 0, 0, 0, 0},
09575 { DEF, DEF, DEF, DEF, DEF},
09576 { DEF, DEF, DEF, DEF, DEF},
```

```

09577 { DEF, DEF, DEF, DEF, DEF},
09578 { DEF, DEF, DEF, DEF, DEF},
09579 /* UA.@A..AU */
09580 {{ 0, 0, 0, 0, 0},
09581 { -429, -429, -429, -429, -429},
09582 { -259, -259, -259, -259, -259},
09583 { -339, -339, -339, -339, -339},
09584 { -329, -329, -329, -329, -329}},
09585 /* UA.@C..AU */
09586 {{ 0, 0, 0, 0, 0},
09587 { -599, -599, -599, -599, -599},
09588 { -239, -239, -239, -239, -239},
09589 { -689, -689, -689, -689, -689},
09590 { -329, -329, -329, -329, -329}},
09591 /* UA.@G..AU */
09592 {{ 0, 0, 0, 0, 0},
09593 { -599, -599, -599, -599, -599},
09594 { -239, -239, -239, -239, -239},
09595 { -689, -689, -689, -689, -689},
09596 { -329, -329, -329, -329, -329}},
09597 /* UA.@U..AU */
09598 {{ 0, 0, 0, 0, 0},
09599 { -599, -599, -599, -599, -599},
09600 { -239, -239, -239, -239, -239},
09601 { -689, -689, -689, -689, -689},
09602 { -329, -329, -329, -329, -329}}},
09603 /* UA.A@..AU */
09604 {{{ DEF, -399, -629, -889, -589},
09605 { -100, -449, -679, -939, -639},
09606 { -100, -449, -679, -939, -639},
09607 { -100, -449, -679, -939, -639},
09608 { -100, -449, -679, -939, -639}},
09609 /* UA.AA..AU */
09610 {{ DEF, -399, -629, -889, -589},
09611 { -479, -828, -1058, -1318, -1018},
09612 { -309, -658, -888, -1148, -848},
09613 { -389, -738, -968, -1228, -928},
09614 { -379, -728, -958, -1218, -918}},
09615 /* UA.AC..AU */
09616 {{ DEF, -399, -629, -889, -589},
09617 { -649, -998, -1228, -1488, -1188},
09618 { -289, -638, -868, -1128, -828},
09619 { -739, -1088, -1318, -1578, -1278},
09620 { -379, -728, -958, -1218, -918}},
09621 /* UA.AG..AU */
09622 {{ DEF, -399, -629, -889, -589},
09623 { -649, -998, -1228, -1488, -1188},
09624 { -289, -638, -868, -1128, -828},
09625 { -739, -1088, -1318, -1578, -1278},
09626 { -379, -728, -958, -1218, -918}},
09627 /* UA.AU..AU */
09628 {{ DEF, -399, -629, -889, -589},
09629 { -649, -998, -1228, -1488, -1188},
09630 { -289, -638, -868, -1128, -828},
09631 { -739, -1088, -1318, -1578, -1278},
09632 { -379, -728, -958, -1218, -918}}},
09633 /* UA.C@..AU */
09634 {{{ DEF, -429, -509, -199, -179},
09635 { -100, -479, -559, -249, -229},
09636 { -100, -479, -559, -249, -229},
09637 { -100, -479, -559, -249, -229},
09638 { -100, -479, -559, -249, -229}},
09639 /* UA.CA..AU */
09640 {{ DEF, -429, -509, -199, -179},
09641 { -479, -858, -938, -628, -608},
09642 { -309, -688, -768, -458, -438},
09643 { -389, -768, -848, -538, -518},
09644 { -379, -758, -838, -528, -508}},
09645 /* UA.CC..AU */
09646 {{ DEF, -429, -509, -199, -179},
09647 { -649, -1028, -1108, -798, -778},
09648 { -289, -668, -748, -438, -418},
09649 { -739, -1118, -1198, -888, -868},
09650 { -379, -758, -838, -528, -508}},
09651 /* UA.CG..AU */
09652 {{ DEF, -429, -509, -199, -179},
09653 { -649, -1028, -1108, -798, -778},
09654 { -289, -668, -748, -438, -418},
09655 { -739, -1118, -1198, -888, -868},
09656 { -379, -758, -838, -528, -508}},
09657 /* UA.CU..AU */
09658 {{ DEF, -429, -509, -199, -179},
09659 { -649, -1028, -1108, -798, -778},
09660 { -289, -668, -748, -438, -418},
09661 { -739, -1118, -1198, -888, -868},
09662 { -379, -758, -838, -528, -508}}},
09663 /* UA.G@..AU */

```

```
09664 {{ DEF, -379, -679, -889, -679},
09665 { -100, -429, -729, -939, -729},
09666 { -100, -429, -729, -939, -729},
09667 { -100, -429, -729, -939, -729},
09668 { -100, -429, -729, -939, -729}},
09669 /* UA.GA..AU */
09670 {{ DEF, -379, -679, -889, -679},
09671 { -479, -808, -1108, -1318, -1108},
09672 { -309, -638, -938, -1148, -938},
09673 { -389, -718, -1018, -1228, -1018},
09674 { -379, -708, -1008, -1218, -1008}},
09675 /* UA.GC..AU */
09676 {{ DEF, -379, -679, -889, -679},
09677 { -649, -978, -1278, -1488, -1278},
09678 { -289, -618, -918, -1128, -918},
09679 { -739, -1068, -1368, -1578, -1368},
09680 { -379, -708, -1008, -1218, -1008}},
09681 /* UA.GG..AU */
09682 {{ DEF, -379, -679, -889, -679},
09683 { -649, -978, -1278, -1488, -1278},
09684 { -289, -618, -918, -1128, -918},
09685 { -739, -1068, -1368, -1578, -1368},
09686 { -379, -708, -1008, -1218, -1008}},
09687 /* UA.GU..AU */
09688 {{ DEF, -379, -679, -889, -679},
09689 { -649, -978, -1278, -1488, -1278},
09690 { -289, -618, -918, -1128, -918},
09691 { -739, -1068, -1368, -1578, -1368},
09692 { -379, -708, -1008, -1218, -1008}}},
09693 /* UA.U@..AU */
09694 {{ DEF, -279, -139, -279, -140},
09695 { -100, -329, -189, -329, -190},
09696 { -100, -329, -189, -329, -190},
09697 { -100, -329, -189, -329, -190},
09698 { -100, -329, -189, -329, -190}},
09699 /* UA.UA..AU */
09700 {{ DEF, -279, -139, -279, -140},
09701 { -479, -708, -568, -708, -569},
09702 { -309, -538, -398, -538, -399},
09703 { -389, -618, -478, -618, -479},
09704 { -379, -608, -468, -608, -469}},
09705 /* UA.UC..AU */
09706 {{ DEF, -279, -139, -279, -140},
09707 { -649, -878, -738, -878, -739},
09708 { -289, -518, -378, -518, -379},
09709 { -739, -968, -828, -968, -829},
09710 { -379, -608, -468, -608, -469}},
09711 /* UA.UG..AU */
09712 {{ DEF, -279, -139, -279, -140},
09713 { -649, -878, -738, -878, -739},
09714 { -289, -518, -378, -518, -379},
09715 { -739, -968, -828, -968, -829},
09716 { -379, -608, -468, -608, -469}},
09717 /* UA.UU..AU */
09718 {{ DEF, -279, -139, -279, -140},
09719 { -649, -878, -738, -878, -739},
09720 { -289, -518, -378, -518, -379},
09721 { -739, -968, -828, -968, -829},
09722 { -379, -608, -468, -608, -469}}}},
09723 /* UA.@@..UA */
09724 {{{ 0, 0, 0, 0, 0},
09725 { DEF, DEF, DEF, DEF, DEF},
09726 { DEF, DEF, DEF, DEF, DEF},
09727 { DEF, DEF, DEF, DEF, DEF},
09728 { DEF, DEF, DEF, DEF, DEF}},
09729 /* UA.@A..UA */
09730 {{ 0, 0, 0, 0, 0},
09731 { -399, -399, -399, -399, -399},
09732 { -429, -429, -429, -429, -429},
09733 { -379, -379, -379, -379, -379},
09734 { -279, -279, -279, -279, -279}},
09735 /* UA.@C..UA */
09736 {{ 0, 0, 0, 0, 0},
09737 { -629, -629, -629, -629, -629},
09738 { -509, -509, -509, -509, -509},
09739 { -679, -679, -679, -679, -679},
09740 { -139, -139, -139, -139, -139}},
09741 /* UA.@G..UA */
09742 {{ 0, 0, 0, 0, 0},
09743 { -889, -889, -889, -889, -889},
09744 { -199, -199, -199, -199, -199},
09745 { -889, -889, -889, -889, -889},
09746 { -279, -279, -279, -279, -279}},
09747 /* UA.@U..UA */
09748 {{ 0, 0, 0, 0, 0},
09749 { -589, -589, -589, -589, -589},
09750 { -179, -179, -179, -179, -179}},
```

```
09751 { -679, -679, -679, -679, -679},
09752 { -140, -140, -140, -140, -140}},
09753 /* UA.A@.UA */
09754 {{ DEF, -399, -629, -889, -589},
09755 { -100, -449, -679, -939, -639},
09756 { -100, -449, -679, -939, -639},
09757 { -100, -449, -679, -939, -639},
09758 { -100, -449, -679, -939, -639}},
09759 /* UA.AA..UA */
09760 {{ DEF, -399, -629, -889, -589},
09761 { -449, -798, -1028, -1288, -988},
09762 { -479, -828, -1058, -1318, -1018},
09763 { -429, -778, -1008, -1268, -968},
09764 { -329, -678, -908, -1168, -868}},
09765 /* UA.AC..UA */
09766 {{ DEF, -399, -629, -889, -589},
09767 { -679, -1028, -1258, -1518, -1218},
09768 { -559, -908, -1138, -1398, -1098},
09769 { -729, -1078, -1308, -1568, -1268},
09770 { -189, -538, -768, -1028, -728}},
09771 /* UA.AG..UA */
09772 {{ DEF, -399, -629, -889, -589},
09773 { -939, -1288, -1518, -1778, -1478},
09774 { -249, -598, -828, -1088, -788},
09775 { -939, -1288, -1518, -1778, -1478},
09776 { -329, -678, -908, -1168, -868}},
09777 /* UA.AU..UA */
09778 {{ DEF, -399, -629, -889, -589},
09779 { -639, -988, -1218, -1478, -1178},
09780 { -229, -578, -808, -1068, -768},
09781 { -729, -1078, -1308, -1568, -1268},
09782 { -190, -539, -769, -1029, -729}},
09783 /* UA.C@.UA */
09784 {{ DEF, -429, -509, -199, -179},
09785 { -100, -479, -559, -249, -229},
09786 { -100, -479, -559, -249, -229},
09787 { -100, -479, -559, -249, -229},
09788 { -100, -479, -559, -249, -229}},
09789 /* UA.CA..UA */
09790 {{ DEF, -429, -509, -199, -179},
09791 { -449, -828, -908, -598, -578},
09792 { -479, -858, -938, -628, -608},
09793 { -429, -808, -888, -578, -558},
09794 { -329, -708, -788, -478, -458}},
09795 /* UA.CC..UA */
09796 {{ DEF, -429, -509, -199, -179},
09797 { -679, -1058, -1138, -828, -808},
09798 { -559, -938, -1018, -708, -688},
09799 { -729, -1108, -1188, -878, -858},
09800 { -189, -568, -648, -338, -318}},
09801 /* UA.CG..UA */
09802 {{ DEF, -429, -509, -199, -179},
09803 { -939, -1318, -1398, -1088, -1068},
09804 { -249, -628, -708, -398, -378},
09805 { -939, -1318, -1398, -1088, -1068},
09806 { -329, -708, -788, -478, -458}},
09807 /* UA.CU..UA */
09808 {{ DEF, -429, -509, -199, -179},
09809 { -639, -1018, -1098, -788, -768},
09810 { -229, -608, -688, -378, -358},
09811 { -729, -1108, -1188, -878, -858},
09812 { -190, -569, -649, -339, -319}},
09813 /* UA.G@.UA */
09814 {{ DEF, -379, -679, -889, -679},
09815 { -100, -429, -729, -939, -729},
09816 { -100, -429, -729, -939, -729},
09817 { -100, -429, -729, -939, -729},
09818 { -100, -429, -729, -939, -729}},
09819 /* UA.GA..UA */
09820 {{ DEF, -379, -679, -889, -679},
09821 { -449, -778, -1078, -1288, -1078},
09822 { -479, -808, -1108, -1318, -1108},
09823 { -429, -758, -1058, -1268, -1058},
09824 { -329, -658, -958, -1168, -958}},
09825 /* UA.GC..UA */
09826 {{ DEF, -379, -679, -889, -679},
09827 { -679, -1008, -1308, -1518, -1308},
09828 { -559, -888, -1188, -1398, -1188},
09829 { -729, -1058, -1358, -1568, -1358},
09830 { -189, -518, -818, -1028, -818}},
09831 /* UA.GG..UA */
09832 {{ DEF, -379, -679, -889, -679},
09833 { -939, -1268, -1568, -1778, -1568},
09834 { -249, -578, -878, -1088, -878},
09835 { -939, -1268, -1568, -1778, -1568},
09836 { -329, -658, -958, -1168, -958}},
09837 /* UA.GU..UA */
```

```
09838 {{ DEF, -379, -679, -889, -679},
09839 { -639, -968, -1268, -1478, -1268},
09840 { -229, -558, -858, -1068, -858},
09841 { -729, -1058, -1358, -1568, -1358},
09842 { -190, -519, -819, -1029, -819}}},
09843 /* UA.U@..UA */
09844 {{{ DEF, -279, -139, -279, -140},
09845 { -100, -329, -189, -329, -190},
09846 { -100, -329, -189, -329, -190},
09847 { -100, -329, -189, -329, -190},
09848 { -100, -329, -189, -329, -190}},
09849 /* UA.UA..UA */
09850 {{ DEF, -279, -139, -279, -140},
09851 { -449, -678, -538, -678, -539},
09852 { -479, -708, -568, -708, -569},
09853 { -429, -658, -518, -658, -519},
09854 { -329, -558, -418, -558, -419}},
09855 /* UA.UC..UA */
09856 {{ DEF, -279, -139, -279, -140},
09857 { -679, -908, -768, -908, -769},
09858 { -559, -788, -648, -788, -649},
09859 { -729, -958, -818, -958, -819},
09860 { -189, -418, -278, -418, -279}},
09861 /* UA.UG..UA */
09862 {{ DEF, -279, -139, -279, -140},
09863 { -939, -1168, -1028, -1168, -1029},
09864 { -249, -478, -338, -478, -339},
09865 { -939, -1168, -1028, -1168, -1029},
09866 { -329, -558, -418, -558, -419}},
09867 /* UA.UU..UA */
09868 {{ DEF, -279, -139, -279, -140},
09869 { -639, -868, -728, -868, -729},
09870 { -229, -458, -318, -458, -319},
09871 { -729, -958, -818, -958, -819},
09872 { -190, -419, -279, -419, -280}}}},
09873 /* UA.@@.. @ */
09874 {{{ DEF, DEF, DEF, DEF, DEF},
09875 { DEF, DEF, DEF, DEF, DEF},
09876 { DEF, DEF, DEF, DEF, DEF},
09877 { DEF, DEF, DEF, DEF, DEF},
09878 { DEF, DEF, DEF, DEF, DEF}},
09879 /* UA.@A.. @ */
09880 {{ DEF, DEF, DEF, DEF, DEF},
09881 { DEF, DEF, DEF, DEF, DEF},
09882 { DEF, DEF, DEF, DEF, DEF},
09883 { DEF, DEF, DEF, DEF, DEF},
09884 { DEF, DEF, DEF, DEF, DEF}},
09885 /* UA.@C.. @ */
09886 {{ DEF, DEF, DEF, DEF, DEF},
09887 { DEF, DEF, DEF, DEF, DEF},
09888 { DEF, DEF, DEF, DEF, DEF},
09889 { DEF, DEF, DEF, DEF, DEF},
09890 { DEF, DEF, DEF, DEF, DEF}},
09891 /* UA.@G.. @ */
09892 {{ DEF, DEF, DEF, DEF, DEF},
09893 { DEF, DEF, DEF, DEF, DEF},
09894 { DEF, DEF, DEF, DEF, DEF},
09895 { DEF, DEF, DEF, DEF, DEF},
09896 { DEF, DEF, DEF, DEF, DEF}},
09897 /* UA.@U.. @ */
09898 {{ DEF, DEF, DEF, DEF, DEF},
09899 { DEF, DEF, DEF, DEF, DEF},
09900 { DEF, DEF, DEF, DEF, DEF},
09901 { DEF, DEF, DEF, DEF, DEF},
09902 { DEF, DEF, DEF, DEF, DEF}}},
09903 /* UA.A@.. @ */
09904 {{{ -100, -449, -679, -939, -639},
09905 { -100, -449, -679, -939, -639},
09906 { -100, -449, -679, -939, -639},
09907 { -100, -449, -679, -939, -639},
09908 { -100, -449, -679, -939, -639}},
09909 /* UA.AA.. @ */
09910 {{ -100, -449, -679, -939, -639},
09911 { -100, -449, -679, -939, -639},
09912 { -100, -449, -679, -939, -639},
09913 { -100, -449, -679, -939, -639},
09914 { -100, -449, -679, -939, -639}},
09915 /* UA.AC.. @ */
09916 {{ -100, -449, -679, -939, -639},
09917 { -100, -449, -679, -939, -639},
09918 { -100, -449, -679, -939, -639},
09919 { -100, -449, -679, -939, -639},
09920 { -100, -449, -679, -939, -639}},
09921 /* UA.AG.. @ */
09922 {{ -100, -449, -679, -939, -639},
09923 { -100, -449, -679, -939, -639},
09924 { -100, -449, -679, -939, -639},
```

```
09925 { -100, -449, -679, -939, -639},
09926 { -100, -449, -679, -939, -639}},
09927 /* UA.AU.. @ */
09928 {{ -100, -449, -679, -939, -639},
09929 { -100, -449, -679, -939, -639},
09930 { -100, -449, -679, -939, -639},
09931 { -100, -449, -679, -939, -639},
09932 { -100, -449, -679, -939, -639}}},
09933 /* UA.C@.. @ */
09934 {{{ -100, -479, -559, -249, -229},
09935 { -100, -479, -559, -249, -229},
09936 { -100, -479, -559, -249, -229},
09937 { -100, -479, -559, -249, -229},
09938 { -100, -479, -559, -249, -229}}},
09939 /* UA.CA.. @ */
09940 {{ -100, -479, -559, -249, -229},
09941 { -100, -479, -559, -249, -229},
09942 { -100, -479, -559, -249, -229},
09943 { -100, -479, -559, -249, -229},
09944 { -100, -479, -559, -249, -229}}},
09945 /* UA.CC.. @ */
09946 {{ -100, -479, -559, -249, -229},
09947 { -100, -479, -559, -249, -229},
09948 { -100, -479, -559, -249, -229},
09949 { -100, -479, -559, -249, -229},
09950 { -100, -479, -559, -249, -229}}},
09951 /* UA.CG.. @ */
09952 {{ -100, -479, -559, -249, -229},
09953 { -100, -479, -559, -249, -229},
09954 { -100, -479, -559, -249, -229},
09955 { -100, -479, -559, -249, -229},
09956 { -100, -479, -559, -249, -229}}},
09957 /* UA.CU.. @ */
09958 {{ -100, -479, -559, -249, -229},
09959 { -100, -479, -559, -249, -229},
09960 { -100, -479, -559, -249, -229},
09961 { -100, -479, -559, -249, -229},
09962 { -100, -479, -559, -249, -229}}},
09963 /* UA.G@.. @ */
09964 {{{ -100, -429, -729, -939, -729},
09965 { -100, -429, -729, -939, -729},
09966 { -100, -429, -729, -939, -729},
09967 { -100, -429, -729, -939, -729},
09968 { -100, -429, -729, -939, -729}}},
09969 /* UA.GA.. @ */
09970 {{ -100, -429, -729, -939, -729},
09971 { -100, -429, -729, -939, -729},
09972 { -100, -429, -729, -939, -729},
09973 { -100, -429, -729, -939, -729},
09974 { -100, -429, -729, -939, -729}}},
09975 /* UA.GC.. @ */
09976 {{ -100, -429, -729, -939, -729},
09977 { -100, -429, -729, -939, -729},
09978 { -100, -429, -729, -939, -729},
09979 { -100, -429, -729, -939, -729},
09980 { -100, -429, -729, -939, -729}}},
09981 /* UA.GG.. @ */
09982 {{ -100, -429, -729, -939, -729},
09983 { -100, -429, -729, -939, -729},
09984 { -100, -429, -729, -939, -729},
09985 { -100, -429, -729, -939, -729},
09986 { -100, -429, -729, -939, -729}}},
09987 /* UA.GU.. @ */
09988 {{ -100, -429, -729, -939, -729},
09989 { -100, -429, -729, -939, -729},
09990 { -100, -429, -729, -939, -729},
09991 { -100, -429, -729, -939, -729},
09992 { -100, -429, -729, -939, -729}}},
09993 /* UA.U@.. @ */
09994 {{{ -100, -329, -189, -329, -190},
09995 { -100, -329, -189, -329, -190},
09996 { -100, -329, -189, -329, -190},
09997 { -100, -329, -189, -329, -190},
09998 { -100, -329, -189, -329, -190}}},
09999 /* UA.UA.. @ */
10000 {{ -100, -329, -189, -329, -190},
10001 { -100, -329, -189, -329, -190},
10002 { -100, -329, -189, -329, -190},
10003 { -100, -329, -189, -329, -190},
10004 { -100, -329, -189, -329, -190}}},
10005 /* UA.UC.. @ */
10006 {{ -100, -329, -189, -329, -190},
10007 { -100, -329, -189, -329, -190},
10008 { -100, -329, -189, -329, -190},
10009 { -100, -329, -189, -329, -190},
10010 { -100, -329, -189, -329, -190}}},
10011 /* UA.UG.. @ */
```



```
10012 {{ -100, -329, -189, -329, -190},
10013 { -100, -329, -189, -329, -190},
10014 { -100, -329, -189, -329, -190},
10015 { -100, -329, -189, -329, -190},
10016 { -100, -329, -189, -329, -190}},
10017 /* UA.UU.. @ */
10018 {{ -100, -329, -189, -329, -190},
10019 { -100, -329, -189, -329, -190},
10020 { -100, -329, -189, -329, -190},
10021 { -100, -329, -189, -329, -190},
10022 { -100, -329, -189, -329, -190}}}},
10023 { /* noPair */ {{{0}}}},
10024 /* @.@.CG */
10025 {{{ DEF, DEF, DEF, DEF, DEF},
10026 { -100, -100, -100, -100, -100},
10027 { -100, -100, -100, -100, -100},
10028 { -100, -100, -100, -100, -100},
10029 { -100, -100, -100, -100, -100}},
10030 /* @.A.CG */
10031 {{ DEF, DEF, DEF, DEF, DEF},
10032 {-1079,-1079,-1079,-1079,-1079},
10033 { -569, -569, -569, -569, -569},
10034 { -989, -989, -989, -989, -989},
10035 { -859, -859, -859, -859, -859}},
10036 /* @.C.CG */
10037 {{ DEF, DEF, DEF, DEF, DEF},
10038 { -999, -999, -999, -999, -999},
10039 { -499, -499, -499, -499, -499},
10040 { -989, -989, -989, -989, -989},
10041 { -789, -789, -789, -789, -789}},
10042 /* @.G.CG */
10043 {{ DEF, DEF, DEF, DEF, DEF},
10044 {-1079,-1079,-1079,-1079,-1079},
10045 { -569, -569, -569, -569, -569},
10046 { -989, -989, -989, -989, -989},
10047 { -859, -859, -859, -859, -859}},
10048 /* @.U.CG */
10049 {{ DEF, DEF, DEF, DEF, DEF},
10050 {-1079,-1079,-1079,-1079,-1079},
10051 { -719, -719, -719, -719, -719},
10052 { -989, -989, -989, -989, -989},
10053 { -909, -909, -909, -909, -909}}}},
10054 /* @.A@.CG */
10055 {{{ DEF, DEF, DEF, DEF, DEF},
10056 { -100, -100, -100, -100, -100},
10057 { -100, -100, -100, -100, -100},
10058 { -100, -100, -100, -100, -100},
10059 { -100, -100, -100, -100, -100}},
10060 /* @.AA.CG */
10061 {{ DEF, DEF, DEF, DEF, DEF},
10062 {-1079,-1079,-1079,-1079,-1079},
10063 { -569, -569, -569, -569, -569},
10064 { -989, -989, -989, -989, -989},
10065 { -859, -859, -859, -859, -859}},
10066 /* @.AC.CG */
10067 {{ DEF, DEF, DEF, DEF, DEF},
10068 { -999, -999, -999, -999, -999},
10069 { -499, -499, -499, -499, -499},
10070 { -989, -989, -989, -989, -989},
10071 { -789, -789, -789, -789, -789}},
10072 /* @.AG.CG */
10073 {{ DEF, DEF, DEF, DEF, DEF},
10074 {-1079,-1079,-1079,-1079,-1079},
10075 { -569, -569, -569, -569, -569},
10076 { -989, -989, -989, -989, -989},
10077 { -859, -859, -859, -859, -859}},
10078 /* @.AU.CG */
10079 {{ DEF, DEF, DEF, DEF, DEF},
10080 {-1079,-1079,-1079,-1079,-1079},
10081 { -719, -719, -719, -719, -719},
10082 { -989, -989, -989, -989, -989},
10083 { -909, -909, -909, -909, -909}}}},
10084 /* @.C@.CG */
10085 {{{ DEF, DEF, DEF, DEF, DEF},
10086 { -100, -100, -100, -100, -100},
10087 { -100, -100, -100, -100, -100},
10088 { -100, -100, -100, -100, -100},
10089 { -100, -100, -100, -100, -100}},
10090 /* @.CA.CG */
10091 {{ DEF, DEF, DEF, DEF, DEF},
10092 {-1079,-1079,-1079,-1079,-1079},
10093 { -569, -569, -569, -569, -569},
10094 { -989, -989, -989, -989, -989},
10095 { -859, -859, -859, -859, -859}},
10096 /* @.CC.CG */
10097 {{ DEF, DEF, DEF, DEF, DEF},
10098 { -999, -999, -999, -999, -999},
```

```

10099 { -499, -499, -499, -499, -499},
10100 { -989, -989, -989, -989, -989},
10101 { -789, -789, -789, -789, -789}},
10102 /* @.CG..CG */
10103 {{ DEF, DEF, DEF, DEF, DEF},
10104 {-1079,-1079,-1079,-1079,-1079},
10105 { -569, -569, -569, -569, -569},
10106 { -989, -989, -989, -989, -989},
10107 { -859, -859, -859, -859, -859}},
10108 /* @.CU..CG */
10109 {{ DEF, DEF, DEF, DEF, DEF},
10110 {-1079,-1079,-1079,-1079,-1079},
10111 { -719, -719, -719, -719, -719},
10112 { -989, -989, -989, -989, -989},
10113 { -909, -909, -909, -909, -909}}},
10114 /* @.G@..CG */
10115 {{{ DEF, DEF, DEF, DEF, DEF},
10116 { -100, -100, -100, -100, -100},
10117 { -100, -100, -100, -100, -100},
10118 { -100, -100, -100, -100, -100},
10119 { -100, -100, -100, -100, -100}},
10120 /* @.GA..CG */
10121 {{ DEF, DEF, DEF, DEF, DEF},
10122 {-1079,-1079,-1079,-1079,-1079},
10123 { -569, -569, -569, -569, -569},
10124 { -989, -989, -989, -989, -989},
10125 { -859, -859, -859, -859, -859}},
10126 /* @.GC..CG */
10127 {{ DEF, DEF, DEF, DEF, DEF},
10128 { -999, -999, -999, -999, -999},
10129 { -499, -499, -499, -499, -499},
10130 { -989, -989, -989, -989, -989},
10131 { -789, -789, -789, -789, -789}},
10132 /* @.GG..CG */
10133 {{ DEF, DEF, DEF, DEF, DEF},
10134 {-1079,-1079,-1079,-1079,-1079},
10135 { -569, -569, -569, -569, -569},
10136 { -989, -989, -989, -989, -989},
10137 { -859, -859, -859, -859, -859}},
10138 /* @.GU..CG */
10139 {{ DEF, DEF, DEF, DEF, DEF},
10140 {-1079,-1079,-1079,-1079,-1079},
10141 { -719, -719, -719, -719, -719},
10142 { -989, -989, -989, -989, -989},
10143 { -909, -909, -909, -909, -909}}},
10144 /* @.U@..CG */
10145 {{{ DEF, DEF, DEF, DEF, DEF},
10146 { -100, -100, -100, -100, -100},
10147 { -100, -100, -100, -100, -100},
10148 { -100, -100, -100, -100, -100},
10149 { -100, -100, -100, -100, -100}},
10150 /* @.UA..CG */
10151 {{ DEF, DEF, DEF, DEF, DEF},
10152 {-1079,-1079,-1079,-1079,-1079},
10153 { -569, -569, -569, -569, -569},
10154 { -989, -989, -989, -989, -989},
10155 { -859, -859, -859, -859, -859}},
10156 /* @.UC..CG */
10157 {{ DEF, DEF, DEF, DEF, DEF},
10158 { -999, -999, -999, -999, -999},
10159 { -499, -499, -499, -499, -499},
10160 { -989, -989, -989, -989, -989},
10161 { -789, -789, -789, -789, -789}},
10162 /* @.UG..CG */
10163 {{ DEF, DEF, DEF, DEF, DEF},
10164 {-1079,-1079,-1079,-1079,-1079},
10165 { -569, -569, -569, -569, -569},
10166 { -989, -989, -989, -989, -989},
10167 { -859, -859, -859, -859, -859}},
10168 /* @.UU..CG */
10169 {{ DEF, DEF, DEF, DEF, DEF},
10170 {-1079,-1079,-1079,-1079,-1079},
10171 { -719, -719, -719, -719, -719},
10172 { -989, -989, -989, -989, -989},
10173 { -909, -909, -909, -909, -909}}},
10174 /* @.@@..GC */
10175 {{{{ DEF, DEF, DEF, DEF, DEF},
10176 { -100, -100, -100, -100, -100},
10177 { -100, -100, -100, -100, -100},
10178 { -100, -100, -100, -100, -100},
10179 { -100, -100, -100, -100, -100}},
10180 /* @.A@..GC */
10181 {{ DEF, DEF, DEF, DEF, DEF},
10182 { -569, -569, -569, -569, -569},
10183 { -769, -769, -769, -769, -769},
10184 { -759, -759, -759, -759, -759},
10185 { -549, -549, -549, -549, -549}},

```

```

10186 /* @.C..GC */
10187 {{ DEF, DEF, DEF, DEF, DEF},
10188 { -929, -929, -929, -929, -929},
10189 { -359, -359, -359, -359, -359},
10190 { -789, -789, -789, -789, -789},
10191 { -549, -549, -549, -549, -549}},
10192 /* @.G..GC */
10193 {{ DEF, DEF, DEF, DEF, DEF},
10194 { -609, -609, -609, -609, -609},
10195 { -359, -359, -359, -359, -359},
10196 { -669, -669, -669, -669, -669},
10197 { -549, -549, -549, -549, -549}},
10198 /* @.U..GC */
10199 {{ DEF, DEF, DEF, DEF, DEF},
10200 { -929, -929, -929, -929, -929},
10201 { -439, -439, -439, -439, -439},
10202 { -789, -789, -789, -789, -789},
10203 { -619, -619, -619, -619, -619}}},
10204 /* @.A@..GC */
10205 {{{ DEF, DEF, DEF, DEF, DEF},
10206 { -100, -100, -100, -100, -100},
10207 { -100, -100, -100, -100, -100},
10208 { -100, -100, -100, -100, -100},
10209 { -100, -100, -100, -100, -100}},
10210 /* @.AA..GC */
10211 {{ DEF, DEF, DEF, DEF, DEF},
10212 { -569, -569, -569, -569, -569},
10213 { -769, -769, -769, -769, -769},
10214 { -759, -759, -759, -759, -759},
10215 { -549, -549, -549, -549, -549}},
10216 /* @.AC..GC */
10217 {{ DEF, DEF, DEF, DEF, DEF},
10218 { -929, -929, -929, -929, -929},
10219 { -359, -359, -359, -359, -359},
10220 { -789, -789, -789, -789, -789},
10221 { -549, -549, -549, -549, -549}},
10222 /* @.AG..GC */
10223 {{ DEF, DEF, DEF, DEF, DEF},
10224 { -609, -609, -609, -609, -609},
10225 { -359, -359, -359, -359, -359},
10226 { -669, -669, -669, -669, -669},
10227 { -549, -549, -549, -549, -549}},
10228 /* @.AU..GC */
10229 {{ DEF, DEF, DEF, DEF, DEF},
10230 { -929, -929, -929, -929, -929},
10231 { -439, -439, -439, -439, -439},
10232 { -789, -789, -789, -789, -789},
10233 { -619, -619, -619, -619, -619}}},
10234 /* @.C@..GC */
10235 {{{ DEF, DEF, DEF, DEF, DEF},
10236 { -100, -100, -100, -100, -100},
10237 { -100, -100, -100, -100, -100},
10238 { -100, -100, -100, -100, -100},
10239 { -100, -100, -100, -100, -100}},
10240 /* @.CA..GC */
10241 {{ DEF, DEF, DEF, DEF, DEF},
10242 { -569, -569, -569, -569, -569},
10243 { -769, -769, -769, -769, -769},
10244 { -759, -759, -759, -759, -759},
10245 { -549, -549, -549, -549, -549}},
10246 /* @.CC..GC */
10247 {{ DEF, DEF, DEF, DEF, DEF},
10248 { -929, -929, -929, -929, -929},
10249 { -359, -359, -359, -359, -359},
10250 { -789, -789, -789, -789, -789},
10251 { -549, -549, -549, -549, -549}},
10252 /* @.CG..GC */
10253 {{ DEF, DEF, DEF, DEF, DEF},
10254 { -609, -609, -609, -609, -609},
10255 { -359, -359, -359, -359, -359},
10256 { -669, -669, -669, -669, -669},
10257 { -549, -549, -549, -549, -549}},
10258 /* @.CU..GC */
10259 {{ DEF, DEF, DEF, DEF, DEF},
10260 { -929, -929, -929, -929, -929},
10261 { -439, -439, -439, -439, -439},
10262 { -789, -789, -789, -789, -789},
10263 { -619, -619, -619, -619, -619}}},
10264 /* @.G@..GC */
10265 {{{ DEF, DEF, DEF, DEF, DEF},
10266 { -100, -100, -100, -100, -100},
10267 { -100, -100, -100, -100, -100},
10268 { -100, -100, -100, -100, -100},
10269 { -100, -100, -100, -100, -100}},
10270 /* @.GA..GC */
10271 {{ DEF, DEF, DEF, DEF, DEF},
10272 { -569, -569, -569, -569, -569},

```

```
10273 { -769, -769, -769, -769, -769},
10274 { -759, -759, -759, -759, -759},
10275 { -549, -549, -549, -549, -549}},
10276 /* @.GC..GC */
10277 {{ DEF, DEF, DEF, DEF, DEF},
10278 { -929, -929, -929, -929, -929},
10279 { -359, -359, -359, -359, -359},
10280 { -789, -789, -789, -789, -789},
10281 { -549, -549, -549, -549, -549}},
10282 /* @.GG..GC */
10283 {{ DEF, DEF, DEF, DEF, DEF},
10284 { -609, -609, -609, -609, -609},
10285 { -359, -359, -359, -359, -359},
10286 { -669, -669, -669, -669, -669},
10287 { -549, -549, -549, -549, -549}},
10288 /* @.GU..GC */
10289 {{ DEF, DEF, DEF, DEF, DEF},
10290 { -929, -929, -929, -929, -929},
10291 { -439, -439, -439, -439, -439},
10292 { -789, -789, -789, -789, -789},
10293 { -619, -619, -619, -619, -619}}},
10294 /* @.U@..GC */
10295 {{{ DEF, DEF, DEF, DEF, DEF},
10296 { -100, -100, -100, -100, -100},
10297 { -100, -100, -100, -100, -100},
10298 { -100, -100, -100, -100, -100},
10299 { -100, -100, -100, -100, -100}},
10300 /* @.UA..GC */
10301 {{ DEF, DEF, DEF, DEF, DEF},
10302 { -569, -569, -569, -569, -569},
10303 { -769, -769, -769, -769, -769},
10304 { -759, -759, -759, -759, -759},
10305 { -549, -549, -549, -549, -549}},
10306 /* @.UC..GC */
10307 {{ DEF, DEF, DEF, DEF, DEF},
10308 { -929, -929, -929, -929, -929},
10309 { -359, -359, -359, -359, -359},
10310 { -789, -789, -789, -789, -789},
10311 { -549, -549, -549, -549, -549}},
10312 /* @.UG..GC */
10313 {{ DEF, DEF, DEF, DEF, DEF},
10314 { -609, -609, -609, -609, -609},
10315 { -359, -359, -359, -359, -359},
10316 { -669, -669, -669, -669, -669},
10317 { -549, -549, -549, -549, -549}},
10318 /* @.UU..GC */
10319 {{ DEF, DEF, DEF, DEF, DEF},
10320 { -929, -929, -929, -929, -929},
10321 { -439, -439, -439, -439, -439},
10322 { -789, -789, -789, -789, -789},
10323 { -619, -619, -619, -619, -619}}}},
10324 /* @.@@..GU */
10325 {{{{ DEF, DEF, DEF, DEF, DEF},
10326 { -100, -100, -100, -100, -100},
10327 { -100, -100, -100, -100, -100},
10328 { -100, -100, -100, -100, -100},
10329 { -100, -100, -100, -100, -100}},
10330 /* @.@A..GU */
10331 {{ DEF, DEF, DEF, DEF, DEF},
10332 { -479, -479, -479, -479, -479},
10333 { -309, -309, -309, -309, -309},
10334 { -389, -389, -389, -389, -389},
10335 { -379, -379, -379, -379, -379}},
10336 /* @.@C..GU */
10337 {{ DEF, DEF, DEF, DEF, DEF},
10338 { -649, -649, -649, -649, -649},
10339 { -289, -289, -289, -289, -289},
10340 { -739, -739, -739, -739, -739},
10341 { -379, -379, -379, -379, -379}},
10342 /* @.@G..GU */
10343 {{ DEF, DEF, DEF, DEF, DEF},
10344 { -649, -649, -649, -649, -649},
10345 { -289, -289, -289, -289, -289},
10346 { -739, -739, -739, -739, -739},
10347 { -379, -379, -379, -379, -379}},
10348 /* @.@U..GU */
10349 {{ DEF, DEF, DEF, DEF, DEF},
10350 { -649, -649, -649, -649, -649},
10351 { -289, -289, -289, -289, -289},
10352 { -739, -739, -739, -739, -739},
10353 { -379, -379, -379, -379, -379}}}},
10354 /* @.A@..GU */
10355 {{{{ DEF, DEF, DEF, DEF, DEF},
10356 { -100, -100, -100, -100, -100},
10357 { -100, -100, -100, -100, -100},
10358 { -100, -100, -100, -100, -100},
10359 { -100, -100, -100, -100, -100}},
```

```
10360 /* @.AA..GU */
10361 {{ DEF, DEF, DEF, DEF, DEF},
10362 { -479, -479, -479, -479, -479},
10363 { -309, -309, -309, -309, -309},
10364 { -389, -389, -389, -389, -389},
10365 { -379, -379, -379, -379, -379}},
10366 /* @.AC..GU */
10367 {{ DEF, DEF, DEF, DEF, DEF},
10368 { -649, -649, -649, -649, -649},
10369 { -289, -289, -289, -289, -289},
10370 { -739, -739, -739, -739, -739},
10371 { -379, -379, -379, -379, -379}},
10372 /* @.AG..GU */
10373 {{ DEF, DEF, DEF, DEF, DEF},
10374 { -649, -649, -649, -649, -649},
10375 { -289, -289, -289, -289, -289},
10376 { -739, -739, -739, -739, -739},
10377 { -379, -379, -379, -379, -379}},
10378 /* @.AU..GU */
10379 {{ DEF, DEF, DEF, DEF, DEF},
10380 { -649, -649, -649, -649, -649},
10381 { -289, -289, -289, -289, -289},
10382 { -739, -739, -739, -739, -739},
10383 { -379, -379, -379, -379, -379}},
10384 /* @.C@..GU */
10385 {{{ DEF, DEF, DEF, DEF, DEF},
10386 { -100, -100, -100, -100, -100},
10387 { -100, -100, -100, -100, -100},
10388 { -100, -100, -100, -100, -100},
10389 { -100, -100, -100, -100, -100}},
10390 /* @.CA..GU */
10391 {{ DEF, DEF, DEF, DEF, DEF},
10392 { -479, -479, -479, -479, -479},
10393 { -309, -309, -309, -309, -309},
10394 { -389, -389, -389, -389, -389},
10395 { -379, -379, -379, -379, -379}},
10396 /* @.CC..GU */
10397 {{ DEF, DEF, DEF, DEF, DEF},
10398 { -649, -649, -649, -649, -649},
10399 { -289, -289, -289, -289, -289},
10400 { -739, -739, -739, -739, -739},
10401 { -379, -379, -379, -379, -379}},
10402 /* @.CG..GU */
10403 {{ DEF, DEF, DEF, DEF, DEF},
10404 { -649, -649, -649, -649, -649},
10405 { -289, -289, -289, -289, -289},
10406 { -739, -739, -739, -739, -739},
10407 { -379, -379, -379, -379, -379}},
10408 /* @.CU..GU */
10409 {{ DEF, DEF, DEF, DEF, DEF},
10410 { -649, -649, -649, -649, -649},
10411 { -289, -289, -289, -289, -289},
10412 { -739, -739, -739, -739, -739},
10413 { -379, -379, -379, -379, -379}},
10414 /* @.G@..GU */
10415 {{{ DEF, DEF, DEF, DEF, DEF},
10416 { -100, -100, -100, -100, -100},
10417 { -100, -100, -100, -100, -100},
10418 { -100, -100, -100, -100, -100},
10419 { -100, -100, -100, -100, -100}},
10420 /* @.GA..GU */
10421 {{ DEF, DEF, DEF, DEF, DEF},
10422 { -479, -479, -479, -479, -479},
10423 { -309, -309, -309, -309, -309},
10424 { -389, -389, -389, -389, -389},
10425 { -379, -379, -379, -379, -379}},
10426 /* @.GC..GU */
10427 {{ DEF, DEF, DEF, DEF, DEF},
10428 { -649, -649, -649, -649, -649},
10429 { -289, -289, -289, -289, -289},
10430 { -739, -739, -739, -739, -739},
10431 { -379, -379, -379, -379, -379}},
10432 /* @.GG..GU */
10433 {{ DEF, DEF, DEF, DEF, DEF},
10434 { -649, -649, -649, -649, -649},
10435 { -289, -289, -289, -289, -289},
10436 { -739, -739, -739, -739, -739},
10437 { -379, -379, -379, -379, -379}},
10438 /* @.GU..GU */
10439 {{ DEF, DEF, DEF, DEF, DEF},
10440 { -649, -649, -649, -649, -649},
10441 { -289, -289, -289, -289, -289},
10442 { -739, -739, -739, -739, -739},
10443 { -379, -379, -379, -379, -379}},
10444 /* @.U@..GU */
10445 {{{ DEF, DEF, DEF, DEF, DEF},
10446 { -100, -100, -100, -100, -100},
```

```
10447 { -100, -100, -100, -100, -100},
10448 { -100, -100, -100, -100, -100},
10449 { -100, -100, -100, -100, -100}},
10450 /* @.UA..GU */
10451 {{ DEF, DEF, DEF, DEF, DEF},
10452 { -479, -479, -479, -479, -479},
10453 { -309, -309, -309, -309, -309},
10454 { -389, -389, -389, -389, -389},
10455 { -379, -379, -379, -379, -379}},
10456 /* @.UC..GU */
10457 {{ DEF, DEF, DEF, DEF, DEF},
10458 { -649, -649, -649, -649, -649},
10459 { -289, -289, -289, -289, -289},
10460 { -739, -739, -739, -739, -739},
10461 { -379, -379, -379, -379, -379}},
10462 /* @.UG..GU */
10463 {{ DEF, DEF, DEF, DEF, DEF},
10464 { -649, -649, -649, -649, -649},
10465 { -289, -289, -289, -289, -289},
10466 { -739, -739, -739, -739, -739},
10467 { -379, -379, -379, -379, -379}},
10468 /* @.UU..GU */
10469 {{ DEF, DEF, DEF, DEF, DEF},
10470 { -649, -649, -649, -649, -649},
10471 { -289, -289, -289, -289, -289},
10472 { -739, -739, -739, -739, -739},
10473 { -379, -379, -379, -379, -379}}}},
10474 /* @.@@..UG */
10475 {{{ DEF, DEF, DEF, DEF, DEF},
10476 { -100, -100, -100, -100, -100},
10477 { -100, -100, -100, -100, -100},
10478 { -100, -100, -100, -100, -100},
10479 { -100, -100, -100, -100, -100}},
10480 /* @.@A..UG */
10481 {{ DEF, DEF, DEF, DEF, DEF},
10482 { -769, -769, -769, -769, -769},
10483 { -529, -529, -529, -529, -529},
10484 { -709, -709, -709, -709, -709},
10485 { -599, -599, -599, -599, -599}},
10486 /* @.@C..UG */
10487 {{ DEF, DEF, DEF, DEF, DEF},
10488 { -839, -839, -839, -839, -839},
10489 { -529, -529, -529, -529, -529},
10490 { -859, -859, -859, -859, -859},
10491 { -489, -489, -489, -489, -489}},
10492 /* @.@G..UG */
10493 {{ DEF, DEF, DEF, DEF, DEF},
10494 {-1009, -1009, -1009, -1009, -1009},
10495 { -409, -409, -409, -409, -409},
10496 { -969, -969, -969, -969, -969},
10497 { -599, -599, -599, -599, -599}},
10498 /* @.@U..UG */
10499 {{ DEF, DEF, DEF, DEF, DEF},
10500 { -859, -859, -859, -859, -859},
10501 { -529, -529, -529, -529, -529},
10502 { -859, -859, -859, -859, -859},
10503 { -409, -409, -409, -409, -409}}}},
10504 /* @.A@..UG */
10505 {{{ DEF, DEF, DEF, DEF, DEF},
10506 { -100, -100, -100, -100, -100},
10507 { -100, -100, -100, -100, -100},
10508 { -100, -100, -100, -100, -100},
10509 { -100, -100, -100, -100, -100}},
10510 /* @.AA..UG */
10511 {{ DEF, DEF, DEF, DEF, DEF},
10512 { -769, -769, -769, -769, -769},
10513 { -529, -529, -529, -529, -529},
10514 { -709, -709, -709, -709, -709},
10515 { -599, -599, -599, -599, -599}},
10516 /* @.AC..UG */
10517 {{ DEF, DEF, DEF, DEF, DEF},
10518 { -839, -839, -839, -839, -839},
10519 { -529, -529, -529, -529, -529},
10520 { -859, -859, -859, -859, -859},
10521 { -489, -489, -489, -489, -489}},
10522 /* @.AG..UG */
10523 {{ DEF, DEF, DEF, DEF, DEF},
10524 {-1009, -1009, -1009, -1009, -1009},
10525 { -409, -409, -409, -409, -409},
10526 { -969, -969, -969, -969, -969},
10527 { -599, -599, -599, -599, -599}},
10528 /* @.AU..UG */
10529 {{ DEF, DEF, DEF, DEF, DEF},
10530 { -859, -859, -859, -859, -859},
10531 { -529, -529, -529, -529, -529},
10532 { -859, -859, -859, -859, -859},
10533 { -409, -409, -409, -409, -409}}},
```

```

10534 /* @.C@.UG */
10535 {{ DEF, DEF, DEF, DEF, DEF},
10536 { -100, -100, -100, -100, -100},
10537 { -100, -100, -100, -100, -100},
10538 { -100, -100, -100, -100, -100},
10539 { -100, -100, -100, -100, -100}},
10540 /* @.CA@.UG */
10541 {{ DEF, DEF, DEF, DEF, DEF},
10542 { -769, -769, -769, -769, -769},
10543 { -529, -529, -529, -529, -529},
10544 { -709, -709, -709, -709, -709},
10545 { -599, -599, -599, -599, -599}},
10546 /* @.CC@.UG */
10547 {{ DEF, DEF, DEF, DEF, DEF},
10548 { -839, -839, -839, -839, -839},
10549 { -529, -529, -529, -529, -529},
10550 { -859, -859, -859, -859, -859},
10551 { -489, -489, -489, -489, -489}},
10552 /* @.CG@.UG */
10553 {{ DEF, DEF, DEF, DEF, DEF},
10554 { -1009, -1009, -1009, -1009, -1009},
10555 { -409, -409, -409, -409, -409},
10556 { -969, -969, -969, -969, -969},
10557 { -599, -599, -599, -599, -599}},
10558 /* @.CU@.UG */
10559 {{ DEF, DEF, DEF, DEF, DEF},
10560 { -859, -859, -859, -859, -859},
10561 { -529, -529, -529, -529, -529},
10562 { -859, -859, -859, -859, -859},
10563 { -409, -409, -409, -409, -409}},
10564 /* @.G@.UG */
10565 {{ DEF, DEF, DEF, DEF, DEF},
10566 { -100, -100, -100, -100, -100},
10567 { -100, -100, -100, -100, -100},
10568 { -100, -100, -100, -100, -100},
10569 { -100, -100, -100, -100, -100}},
10570 /* @.GA@.UG */
10571 {{ DEF, DEF, DEF, DEF, DEF},
10572 { -769, -769, -769, -769, -769},
10573 { -529, -529, -529, -529, -529},
10574 { -709, -709, -709, -709, -709},
10575 { -599, -599, -599, -599, -599}},
10576 /* @.GC@.UG */
10577 {{ DEF, DEF, DEF, DEF, DEF},
10578 { -839, -839, -839, -839, -839},
10579 { -529, -529, -529, -529, -529},
10580 { -859, -859, -859, -859, -859},
10581 { -489, -489, -489, -489, -489}},
10582 /* @.GG@.UG */
10583 {{ DEF, DEF, DEF, DEF, DEF},
10584 { -1009, -1009, -1009, -1009, -1009},
10585 { -409, -409, -409, -409, -409},
10586 { -969, -969, -969, -969, -969},
10587 { -599, -599, -599, -599, -599}},
10588 /* @.GU@.UG */
10589 {{ DEF, DEF, DEF, DEF, DEF},
10590 { -859, -859, -859, -859, -859},
10591 { -529, -529, -529, -529, -529},
10592 { -859, -859, -859, -859, -859},
10593 { -409, -409, -409, -409, -409}},
10594 /* @.U@.UG */
10595 {{ DEF, DEF, DEF, DEF, DEF},
10596 { -100, -100, -100, -100, -100},
10597 { -100, -100, -100, -100, -100},
10598 { -100, -100, -100, -100, -100},
10599 { -100, -100, -100, -100, -100}},
10600 /* @.UA@.UG */
10601 {{ DEF, DEF, DEF, DEF, DEF},
10602 { -769, -769, -769, -769, -769},
10603 { -529, -529, -529, -529, -529},
10604 { -709, -709, -709, -709, -709},
10605 { -599, -599, -599, -599, -599}},
10606 /* @.UC@.UG */
10607 {{ DEF, DEF, DEF, DEF, DEF},
10608 { -839, -839, -839, -839, -839},
10609 { -529, -529, -529, -529, -529},
10610 { -859, -859, -859, -859, -859},
10611 { -489, -489, -489, -489, -489}},
10612 /* @.UG@.UG */
10613 {{ DEF, DEF, DEF, DEF, DEF},
10614 { -1009, -1009, -1009, -1009, -1009},
10615 { -409, -409, -409, -409, -409},
10616 { -969, -969, -969, -969, -969},
10617 { -599, -599, -599, -599, -599}},
10618 /* @.UU@.UG */
10619 {{ DEF, DEF, DEF, DEF, DEF},
10620 { -859, -859, -859, -859, -859},

```

```

10621 { -529, -529, -529, -529, -529},
10622 { -859, -859, -859, -859, -859},
10623 { -409, -409, -409, -409, -409}}},
10624 /* @.@.AU */
10625 {{{ DEF, DEF, DEF, DEF, DEF},
10626 { -100, -100, -100, -100, -100},
10627 { -100, -100, -100, -100, -100},
10628 { -100, -100, -100, -100, -100},
10629 { -100, -100, -100, -100, -100}},
10630 /* @.A.AU */
10631 {{ DEF, DEF, DEF, DEF, DEF},
10632 { -479, -479, -479, -479, -479},
10633 { -309, -309, -309, -309, -309},
10634 { -389, -389, -389, -389, -389},
10635 { -379, -379, -379, -379, -379}},
10636 /* @.C.AU */
10637 {{ DEF, DEF, DEF, DEF, DEF},
10638 { -649, -649, -649, -649, -649},
10639 { -289, -289, -289, -289, -289},
10640 { -739, -739, -739, -739, -739},
10641 { -379, -379, -379, -379, -379}},
10642 /* @.G.AU */
10643 {{ DEF, DEF, DEF, DEF, DEF},
10644 { -649, -649, -649, -649, -649},
10645 { -289, -289, -289, -289, -289},
10646 { -739, -739, -739, -739, -739},
10647 { -379, -379, -379, -379, -379}},
10648 /* @.U.AU */
10649 {{ DEF, DEF, DEF, DEF, DEF},
10650 { -649, -649, -649, -649, -649},
10651 { -289, -289, -289, -289, -289},
10652 { -739, -739, -739, -739, -739},
10653 { -379, -379, -379, -379, -379}}},
10654 /* @.A.AU */
10655 {{{ DEF, DEF, DEF, DEF, DEF},
10656 { -100, -100, -100, -100, -100},
10657 { -100, -100, -100, -100, -100},
10658 { -100, -100, -100, -100, -100},
10659 { -100, -100, -100, -100, -100}},
10660 /* @.AA.AU */
10661 {{ DEF, DEF, DEF, DEF, DEF},
10662 { -479, -479, -479, -479, -479},
10663 { -309, -309, -309, -309, -309},
10664 { -389, -389, -389, -389, -389},
10665 { -379, -379, -379, -379, -379}},
10666 /* @.AC.AU */
10667 {{ DEF, DEF, DEF, DEF, DEF},
10668 { -649, -649, -649, -649, -649},
10669 { -289, -289, -289, -289, -289},
10670 { -739, -739, -739, -739, -739},
10671 { -379, -379, -379, -379, -379}},
10672 /* @.AG.AU */
10673 {{ DEF, DEF, DEF, DEF, DEF},
10674 { -649, -649, -649, -649, -649},
10675 { -289, -289, -289, -289, -289},
10676 { -739, -739, -739, -739, -739},
10677 { -379, -379, -379, -379, -379}},
10678 /* @.AU.AU */
10679 {{ DEF, DEF, DEF, DEF, DEF},
10680 { -649, -649, -649, -649, -649},
10681 { -289, -289, -289, -289, -289},
10682 { -739, -739, -739, -739, -739},
10683 { -379, -379, -379, -379, -379}}},
10684 /* @.C.AU */
10685 {{{ DEF, DEF, DEF, DEF, DEF},
10686 { -100, -100, -100, -100, -100},
10687 { -100, -100, -100, -100, -100},
10688 { -100, -100, -100, -100, -100},
10689 { -100, -100, -100, -100, -100}},
10690 /* @.CA.AU */
10691 {{ DEF, DEF, DEF, DEF, DEF},
10692 { -479, -479, -479, -479, -479},
10693 { -309, -309, -309, -309, -309},
10694 { -389, -389, -389, -389, -389},
10695 { -379, -379, -379, -379, -379}},
10696 /* @.CC.AU */
10697 {{ DEF, DEF, DEF, DEF, DEF},
10698 { -649, -649, -649, -649, -649},
10699 { -289, -289, -289, -289, -289},
10700 { -739, -739, -739, -739, -739},
10701 { -379, -379, -379, -379, -379}},
10702 /* @.CG.AU */
10703 {{ DEF, DEF, DEF, DEF, DEF},
10704 { -649, -649, -649, -649, -649},
10705 { -289, -289, -289, -289, -289},
10706 { -739, -739, -739, -739, -739},
10707 { -379, -379, -379, -379, -379}},

```



```

10708 /* @.CU..AU */
10709 {{ DEF, DEF, DEF, DEF, DEF},
10710 { -649, -649, -649, -649, -649},
10711 { -289, -289, -289, -289, -289},
10712 { -739, -739, -739, -739, -739},
10713 { -379, -379, -379, -379, -379}}},
10714 /* @.G@..AU */
10715 {{{ DEF, DEF, DEF, DEF, DEF},
10716 { -100, -100, -100, -100, -100},
10717 { -100, -100, -100, -100, -100},
10718 { -100, -100, -100, -100, -100},
10719 { -100, -100, -100, -100, -100}}},
10720 /* @.GA..AU */
10721 {{ DEF, DEF, DEF, DEF, DEF},
10722 { -479, -479, -479, -479, -479},
10723 { -309, -309, -309, -309, -309},
10724 { -389, -389, -389, -389, -389},
10725 { -379, -379, -379, -379, -379}}},
10726 /* @.GC..AU */
10727 {{ DEF, DEF, DEF, DEF, DEF},
10728 { -649, -649, -649, -649, -649},
10729 { -289, -289, -289, -289, -289},
10730 { -739, -739, -739, -739, -739},
10731 { -379, -379, -379, -379, -379}}},
10732 /* @.GG..AU */
10733 {{ DEF, DEF, DEF, DEF, DEF},
10734 { -649, -649, -649, -649, -649},
10735 { -289, -289, -289, -289, -289},
10736 { -739, -739, -739, -739, -739},
10737 { -379, -379, -379, -379, -379}}},
10738 /* @.GU..AU */
10739 {{ DEF, DEF, DEF, DEF, DEF},
10740 { -649, -649, -649, -649, -649},
10741 { -289, -289, -289, -289, -289},
10742 { -739, -739, -739, -739, -739},
10743 { -379, -379, -379, -379, -379}}},
10744 /* @.U@..AU */
10745 {{{ DEF, DEF, DEF, DEF, DEF},
10746 { -100, -100, -100, -100, -100},
10747 { -100, -100, -100, -100, -100},
10748 { -100, -100, -100, -100, -100},
10749 { -100, -100, -100, -100, -100}}},
10750 /* @.UA..AU */
10751 {{ DEF, DEF, DEF, DEF, DEF},
10752 { -479, -479, -479, -479, -479},
10753 { -309, -309, -309, -309, -309},
10754 { -389, -389, -389, -389, -389},
10755 { -379, -379, -379, -379, -379}}},
10756 /* @.UC..AU */
10757 {{ DEF, DEF, DEF, DEF, DEF},
10758 { -649, -649, -649, -649, -649},
10759 { -289, -289, -289, -289, -289},
10760 { -739, -739, -739, -739, -739},
10761 { -379, -379, -379, -379, -379}}},
10762 /* @.UG..AU */
10763 {{ DEF, DEF, DEF, DEF, DEF},
10764 { -649, -649, -649, -649, -649},
10765 { -289, -289, -289, -289, -289},
10766 { -739, -739, -739, -739, -739},
10767 { -379, -379, -379, -379, -379}}},
10768 /* @.UU..AU */
10769 {{ DEF, DEF, DEF, DEF, DEF},
10770 { -649, -649, -649, -649, -649},
10771 { -289, -289, -289, -289, -289},
10772 { -739, -739, -739, -739, -739},
10773 { -379, -379, -379, -379, -379}}}},
10774 /* @.@@..UA */
10775 {{{{ DEF, DEF, DEF, DEF, DEF},
10776 { -100, -100, -100, -100, -100},
10777 { -100, -100, -100, -100, -100},
10778 { -100, -100, -100, -100, -100},
10779 { -100, -100, -100, -100, -100}}}},
10780 /* @.@A..UA */
10781 {{ DEF, DEF, DEF, DEF, DEF},
10782 { -449, -449, -449, -449, -449},
10783 { -479, -479, -479, -479, -479},
10784 { -429, -429, -429, -429, -429},
10785 { -329, -329, -329, -329, -329}}},
10786 /* @.@C..UA */
10787 {{ DEF, DEF, DEF, DEF, DEF},
10788 { -679, -679, -679, -679, -679},
10789 { -559, -559, -559, -559, -559},
10790 { -729, -729, -729, -729, -729},
10791 { -189, -189, -189, -189, -189}}},
10792 /* @.@G..UA */
10793 {{ DEF, DEF, DEF, DEF, DEF},
10794 { -939, -939, -939, -939, -939}},

```

```

10795 { -249, -249, -249, -249, -249},
10796 { -939, -939, -939, -939, -939},
10797 { -329, -329, -329, -329, -329}},
10798 /* @.U.UA */
10799 {{ DEF, DEF, DEF, DEF, DEF},
10800 { -639, -639, -639, -639, -639},
10801 { -229, -229, -229, -229, -229},
10802 { -729, -729, -729, -729, -729},
10803 { -190, -190, -190, -190, -190}}},
10804 /* @.A.UA */
10805 {{{ DEF, DEF, DEF, DEF, DEF},
10806 { -100, -100, -100, -100, -100},
10807 { -100, -100, -100, -100, -100},
10808 { -100, -100, -100, -100, -100},
10809 { -100, -100, -100, -100, -100}}},
10810 /* @.AA.UA */
10811 {{ DEF, DEF, DEF, DEF, DEF},
10812 { -449, -449, -449, -449, -449},
10813 { -479, -479, -479, -479, -479},
10814 { -429, -429, -429, -429, -429},
10815 { -329, -329, -329, -329, -329}},
10816 /* @.AC.UA */
10817 {{ DEF, DEF, DEF, DEF, DEF},
10818 { -679, -679, -679, -679, -679},
10819 { -559, -559, -559, -559, -559},
10820 { -729, -729, -729, -729, -729},
10821 { -189, -189, -189, -189, -189}},
10822 /* @.AG.UA */
10823 {{ DEF, DEF, DEF, DEF, DEF},
10824 { -939, -939, -939, -939, -939},
10825 { -249, -249, -249, -249, -249},
10826 { -939, -939, -939, -939, -939},
10827 { -329, -329, -329, -329, -329}},
10828 /* @.AU.UA */
10829 {{ DEF, DEF, DEF, DEF, DEF},
10830 { -639, -639, -639, -639, -639},
10831 { -229, -229, -229, -229, -229},
10832 { -729, -729, -729, -729, -729},
10833 { -190, -190, -190, -190, -190}}},
10834 /* @.C.UA */
10835 {{{ DEF, DEF, DEF, DEF, DEF},
10836 { -100, -100, -100, -100, -100},
10837 { -100, -100, -100, -100, -100},
10838 { -100, -100, -100, -100, -100},
10839 { -100, -100, -100, -100, -100}}},
10840 /* @.CA.UA */
10841 {{ DEF, DEF, DEF, DEF, DEF},
10842 { -449, -449, -449, -449, -449},
10843 { -479, -479, -479, -479, -479},
10844 { -429, -429, -429, -429, -429},
10845 { -329, -329, -329, -329, -329}},
10846 /* @.CC.UA */
10847 {{ DEF, DEF, DEF, DEF, DEF},
10848 { -679, -679, -679, -679, -679},
10849 { -559, -559, -559, -559, -559},
10850 { -729, -729, -729, -729, -729},
10851 { -189, -189, -189, -189, -189}},
10852 /* @.CG.UA */
10853 {{ DEF, DEF, DEF, DEF, DEF},
10854 { -939, -939, -939, -939, -939},
10855 { -249, -249, -249, -249, -249},
10856 { -939, -939, -939, -939, -939},
10857 { -329, -329, -329, -329, -329}},
10858 /* @.CU.UA */
10859 {{ DEF, DEF, DEF, DEF, DEF},
10860 { -639, -639, -639, -639, -639},
10861 { -229, -229, -229, -229, -229},
10862 { -729, -729, -729, -729, -729},
10863 { -190, -190, -190, -190, -190}}},
10864 /* @.G.UA */
10865 {{{ DEF, DEF, DEF, DEF, DEF},
10866 { -100, -100, -100, -100, -100},
10867 { -100, -100, -100, -100, -100},
10868 { -100, -100, -100, -100, -100},
10869 { -100, -100, -100, -100, -100}}},
10870 /* @.GA.UA */
10871 {{ DEF, DEF, DEF, DEF, DEF},
10872 { -449, -449, -449, -449, -449},
10873 { -479, -479, -479, -479, -479},
10874 { -429, -429, -429, -429, -429},
10875 { -329, -329, -329, -329, -329}},
10876 /* @.GC.UA */
10877 {{ DEF, DEF, DEF, DEF, DEF},
10878 { -679, -679, -679, -679, -679},
10879 { -559, -559, -559, -559, -559},
10880 { -729, -729, -729, -729, -729},
10881 { -189, -189, -189, -189, -189}},

```

```
10882 /* @.GG..UA */
10883 {{ DEF, DEF, DEF, DEF, DEF},
10884 { -939, -939, -939, -939, -939},
10885 { -249, -249, -249, -249, -249},
10886 { -939, -939, -939, -939, -939},
10887 { -329, -329, -329, -329, -329}},
10888 /* @.GU..UA */
10889 {{ DEF, DEF, DEF, DEF, DEF},
10890 { -639, -639, -639, -639, -639},
10891 { -229, -229, -229, -229, -229},
10892 { -729, -729, -729, -729, -729},
10893 { -190, -190, -190, -190, -190}}},
10894 /* @.U@..UA */
10895 {{{ DEF, DEF, DEF, DEF, DEF},
10896 { -100, -100, -100, -100, -100},
10897 { -100, -100, -100, -100, -100},
10898 { -100, -100, -100, -100, -100},
10899 { -100, -100, -100, -100, -100}}},
10900 /* @.UA..UA */
10901 {{ DEF, DEF, DEF, DEF, DEF},
10902 { -449, -449, -449, -449, -449},
10903 { -479, -479, -479, -479, -479},
10904 { -429, -429, -429, -429, -429},
10905 { -329, -329, -329, -329, -329}},
10906 /* @.UC..UA */
10907 {{ DEF, DEF, DEF, DEF, DEF},
10908 { -679, -679, -679, -679, -679},
10909 { -559, -559, -559, -559, -559},
10910 { -729, -729, -729, -729, -729},
10911 { -189, -189, -189, -189, -189}},
10912 /* @.UG..UA */
10913 {{ DEF, DEF, DEF, DEF, DEF},
10914 { -939, -939, -939, -939, -939},
10915 { -249, -249, -249, -249, -249},
10916 { -939, -939, -939, -939, -939},
10917 { -329, -329, -329, -329, -329}},
10918 /* @.UU..UA */
10919 {{ DEF, DEF, DEF, DEF, DEF},
10920 { -639, -639, -639, -639, -639},
10921 { -229, -229, -229, -229, -229},
10922 { -729, -729, -729, -729, -729},
10923 { -190, -190, -190, -190, -190}}}},
10924 /* @.@@.. @ */
10925 {{{{ -100, -100, -100, -100, -100},
10926 { -100, -100, -100, -100, -100},
10927 { -100, -100, -100, -100, -100},
10928 { -100, -100, -100, -100, -100},
10929 { -100, -100, -100, -100, -100}}}},
10930 /* @.@A.. @ */
10931 {{{ -100, -100, -100, -100, -100},
10932 { -100, -100, -100, -100, -100},
10933 { -100, -100, -100, -100, -100},
10934 { -100, -100, -100, -100, -100},
10935 { -100, -100, -100, -100, -100}}},
10936 /* @.@C.. @ */
10937 {{{ -100, -100, -100, -100, -100},
10938 { -100, -100, -100, -100, -100},
10939 { -100, -100, -100, -100, -100},
10940 { -100, -100, -100, -100, -100},
10941 { -100, -100, -100, -100, -100}}},
10942 /* @.@G.. @ */
10943 {{{ -100, -100, -100, -100, -100},
10944 { -100, -100, -100, -100, -100},
10945 { -100, -100, -100, -100, -100},
10946 { -100, -100, -100, -100, -100},
10947 { -100, -100, -100, -100, -100}}},
10948 /* @.@U.. @ */
10949 {{{ -100, -100, -100, -100, -100},
10950 { -100, -100, -100, -100, -100},
10951 { -100, -100, -100, -100, -100},
10952 { -100, -100, -100, -100, -100},
10953 { -100, -100, -100, -100, -100}}}},
10954 /* @.@A.. @ */
10955 {{{{ -100, -100, -100, -100, -100},
10956 { -100, -100, -100, -100, -100},
10957 { -100, -100, -100, -100, -100},
10958 { -100, -100, -100, -100, -100},
10959 { -100, -100, -100, -100, -100}}}},
10960 /* @.@A.. @ */
10961 {{{ -100, -100, -100, -100, -100},
10962 { -100, -100, -100, -100, -100},
10963 { -100, -100, -100, -100, -100},
10964 { -100, -100, -100, -100, -100},
10965 { -100, -100, -100, -100, -100}}},
10966 /* @.@C.. @ */
10967 {{{ -100, -100, -100, -100, -100},
10968 { -100, -100, -100, -100, -100},
```

```

10969 { -100, -100, -100, -100, -100},
10970 { -100, -100, -100, -100, -100},
10971 { -100, -100, -100, -100, -100}},
10972 /* @.AG.. @ */
10973 {{ -100, -100, -100, -100, -100},
10974 { -100, -100, -100, -100, -100},
10975 { -100, -100, -100, -100, -100},
10976 { -100, -100, -100, -100, -100},
10977 { -100, -100, -100, -100, -100}},
10978 /* @.AU.. @ */
10979 {{ -100, -100, -100, -100, -100},
10980 { -100, -100, -100, -100, -100},
10981 { -100, -100, -100, -100, -100},
10982 { -100, -100, -100, -100, -100},
10983 { -100, -100, -100, -100, -100}}},
10984 /* @.C@.. @ */
10985 {{{ -100, -100, -100, -100, -100},
10986 { -100, -100, -100, -100, -100},
10987 { -100, -100, -100, -100, -100},
10988 { -100, -100, -100, -100, -100},
10989 { -100, -100, -100, -100, -100}}},
10990 /* @.CA.. @ */
10991 {{ -100, -100, -100, -100, -100},
10992 { -100, -100, -100, -100, -100},
10993 { -100, -100, -100, -100, -100},
10994 { -100, -100, -100, -100, -100},
10995 { -100, -100, -100, -100, -100}}},
10996 /* @.CC.. @ */
10997 {{ -100, -100, -100, -100, -100},
10998 { -100, -100, -100, -100, -100},
10999 { -100, -100, -100, -100, -100},
11000 { -100, -100, -100, -100, -100},
11001 { -100, -100, -100, -100, -100}},
11002 /* @.CG.. @ */
11003 {{ -100, -100, -100, -100, -100},
11004 { -100, -100, -100, -100, -100},
11005 { -100, -100, -100, -100, -100},
11006 { -100, -100, -100, -100, -100},
11007 { -100, -100, -100, -100, -100}},
11008 /* @.CU.. @ */
11009 {{ -100, -100, -100, -100, -100},
11010 { -100, -100, -100, -100, -100},
11011 { -100, -100, -100, -100, -100},
11012 { -100, -100, -100, -100, -100},
11013 { -100, -100, -100, -100, -100}}},
11014 /* @.G@.. @ */
11015 {{{ -100, -100, -100, -100, -100},
11016 { -100, -100, -100, -100, -100},
11017 { -100, -100, -100, -100, -100},
11018 { -100, -100, -100, -100, -100},
11019 { -100, -100, -100, -100, -100}}},
11020 /* @.GA.. @ */
11021 {{ -100, -100, -100, -100, -100},
11022 { -100, -100, -100, -100, -100},
11023 { -100, -100, -100, -100, -100},
11024 { -100, -100, -100, -100, -100},
11025 { -100, -100, -100, -100, -100}},
11026 /* @.GC.. @ */
11027 {{ -100, -100, -100, -100, -100},
11028 { -100, -100, -100, -100, -100},
11029 { -100, -100, -100, -100, -100},
11030 { -100, -100, -100, -100, -100},
11031 { -100, -100, -100, -100, -100}},
11032 /* @.GG.. @ */
11033 {{ -100, -100, -100, -100, -100},
11034 { -100, -100, -100, -100, -100},
11035 { -100, -100, -100, -100, -100},
11036 { -100, -100, -100, -100, -100},
11037 { -100, -100, -100, -100, -100}},
11038 /* @.GU.. @ */
11039 {{ -100, -100, -100, -100, -100},
11040 { -100, -100, -100, -100, -100},
11041 { -100, -100, -100, -100, -100},
11042 { -100, -100, -100, -100, -100},
11043 { -100, -100, -100, -100, -100}}},
11044 /* @.U@.. @ */
11045 {{{ -100, -100, -100, -100, -100},
11046 { -100, -100, -100, -100, -100},
11047 { -100, -100, -100, -100, -100},
11048 { -100, -100, -100, -100, -100},
11049 { -100, -100, -100, -100, -100}},
11050 /* @.UA.. @ */
11051 {{ -100, -100, -100, -100, -100},
11052 { -100, -100, -100, -100, -100},
11053 { -100, -100, -100, -100, -100},
11054 { -100, -100, -100, -100, -100},
11055 { -100, -100, -100, -100, -100}},

```

```

11056 /* @.UC.. @ */
11057 {{ -100, -100, -100, -100, -100},
11058 { -100, -100, -100, -100, -100},
11059 { -100, -100, -100, -100, -100},
11060 { -100, -100, -100, -100, -100},
11061 { -100, -100, -100, -100, -100}},
11062 /* @.UG.. @ */
11063 {{ -100, -100, -100, -100, -100},
11064 { -100, -100, -100, -100, -100},
11065 { -100, -100, -100, -100, -100},
11066 { -100, -100, -100, -100, -100},
11067 { -100, -100, -100, -100, -100}},
11068 /* @.UU.. @ */
11069 {{ -100, -100, -100, -100, -100},
11070 { -100, -100, -100, -100, -100},
11071 { -100, -100, -100, -100, -100},
11072 { -100, -100, -100, -100, -100},
11073 { -100, -100, -100, -100, -100}}}}};
11074
11075

```

18.157 ViennaRNA/constraints/basic.h File Reference

Functions and data structures for constraining secondary structure predictions and evaluation.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_CONSTRAINT_FILE 0`
Flag for `vrna_constraints_add()` to indicate that constraints are present in a text file.
- `#define VRNA_CONSTRAINT_SOFT_MFE 0`
Indicate generation of constraints for MFE folding.
- `#define VRNA_CONSTRAINT_SOFT_PF VRNA_OPTION_PF`
Indicate generation of constraints for partition function computation.
- `#define VRNA_DECOMP_PAIR_HP (unsigned char)1`
Flag passed to generic softt constraints callback to indicate hairpin loop decomposition step.
- `#define VRNA_DECOMP_PAIR_IL (unsigned char)2`
Indicator for interior loop decomposition step.
- `#define VRNA_DECOMP_PAIR_ML (unsigned char)3`
Indicator for multibranch loop decomposition step.
- `#define VRNA_DECOMP_ML_ML_ML (unsigned char)5`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_ML_STEM (unsigned char)6`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_ML_ML (unsigned char)7`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_ML_UP (unsigned char)8`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_ML_ML_STEM (unsigned char)9`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_ML_COAXIAL (unsigned char)10`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_ML_COAXIAL_ENC (unsigned char)11`
Indicator for decomposition of multibranch loop part.
- `#define VRNA_DECOMP_EXT_EXT (unsigned char)12`
Indicator for decomposition of exterior loop part.
- `#define VRNA_DECOMP_EXT_UP (unsigned char)13`
Indicator for decomposition of exterior loop part.
- `#define VRNA_DECOMP_EXT_STEM (unsigned char)14`

- Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_EXT` (unsigned char)15
- Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM_EXT` (unsigned char)16
- Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_STEM_OUTSIDE` (unsigned char)17
- Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_STEM` (unsigned char)18
- Indicator for decomposition of exterior loop part.*
- `#define VRNA_DECOMP_EXT_EXT_STEM1` (unsigned char)19
- Indicator for decomposition of exterior loop part.*

Functions

- void `vrna_constraints_add` (`vrna_fold_compound_t` *vc, const char *constraint, unsigned int options)
- Add constraints to a `vrna_fold_compound_t` data structure.*

18.157.1 Detailed Description

Functions and data structures for constraining secondary structure predictions and evaluation.

18.158 basic.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_CONSTRAINTS_H
00002 #define VIENNA_RNA_PACKAGE_CONSTRAINTS_H
00003
00004 #include <ViennaRNA/fold_compound.h>
00005
00099 #define VRNA_CONSTRAINT_FILE      0
00100
00107 #define VRNA_CONSTRAINT_SOFT_MFE  0
00108
00115 #define VRNA_CONSTRAINT_SOFT_PF   VRNA_OPTION_PF
00116
00129 #define VRNA_DECOMP_PAIR_HP      (unsigned char)1
00130
00144 #define VRNA_DECOMP_PAIR_IL      (unsigned char)2
00145
00159 #define VRNA_DECOMP_PAIR_ML      (unsigned char)3
00160 #define VRNA_DECOMP_PAIR_ML_EXT  (unsigned char)23
00161
00162 #define VRNA_DECOMP_PAIR_ML_OUTSIDE (unsigned char)4
00176 #define VRNA_DECOMP_ML_ML_ML     (unsigned char)5
00177
00191 #define VRNA_DECOMP_ML_STEM      (unsigned char)6
00192
00206 #define VRNA_DECOMP_ML_ML        (unsigned char)7
00207
00222 #define VRNA_DECOMP_ML_UP        (unsigned char)8
00223
00238 #define VRNA_DECOMP_ML_ML_STEM   (unsigned char)9
00239
00254 #define VRNA_DECOMP_ML_COAXIAL   (unsigned char)10
00255
00270 #define VRNA_DECOMP_ML_COAXIAL_ENC (unsigned char)11
00271
00286 #define VRNA_DECOMP_EXT_EXT      (unsigned char)12
00287
00302 #define VRNA_DECOMP_EXT_UP       (unsigned char)13
00303
00317 #define VRNA_DECOMP_EXT_STEM     (unsigned char)14
00318
00332 #define VRNA_DECOMP_EXT_EXT_EXT  (unsigned char)15
00333
00348 #define VRNA_DECOMP_EXT_STEM_EXT (unsigned char)16
00349
00356 #define VRNA_DECOMP_EXT_STEM_OUTSIDE (unsigned char)17
00357
00372 #define VRNA_DECOMP_EXT_EXT_STEM (unsigned char)18
00373
00389 #define VRNA_DECOMP_EXT_EXT_STEM1 (unsigned char)19
```

```

00390
00391 #define VRNA_DECOMP_EXT_STEM_EXT1 (unsigned char)20
00392
00393 #define VRNA_DECOMP_EXT_L          (unsigned char)21
00394 #define VRNA_DECOMP_EXT_EXT_L     (unsigned char)22
00395
00396 /*
00397  * currently we do not allow for more than 31 different decomposition types
00398  * This must be changed as soon as the above macros turn to values above 32
00399  */
00400 #define VRNA_DECOMP_TYPES_MAX      32
00401
00402
00403 void
00404 vrna_constraints_add(vrna_fold_compound_t *vc,
00405                    const char *constraint,
00406                    unsigned int options);
00407
00408 #endif

```

18.159 ViennaRNA/datastructures/basic.h File Reference

Various data structures and pre-processor macros.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_basepair_s](#)
Base pair data structure used in subopt.c. [More...](#)
- struct [vrna_cpair_s](#)
this datastructure is used as input parameter in functions of PS_dot.c [More...](#)
- struct [vrna_color_s](#)
- struct [vrna_data_linear_s](#)
- struct [vrna_sect_s](#)
Stack of partial structures for backtracking. [More...](#)
- struct [vrna_bp_stack_s](#)
Base pair stack element. [More...](#)
- struct [pu_contrib](#)
contributions to p_u [More...](#)
- struct [interact](#)
interaction data structure for RNAup [More...](#)
- struct [pu_out](#)
Collection of all free_energy of beeing unpaired values for output. [More...](#)
- struct [constrain](#)
constraints for cofolding [More...](#)
- struct [duplexT](#)
Data structure for RNAduplex. [More...](#)
- struct [node](#)
Data structure for RNAsnoop (fold energy list) [More...](#)
- struct [snoopT](#)
Data structure for RNAsnoop. [More...](#)
- struct [dupVar](#)
Data structure used in RNApkplex. [More...](#)

Typedefs

- typedef struct [vrna_basepair_s](#) **vrna_basepair_t**
Typename for the base pair representing data structure [vrna_basepair_s](#).
- typedef struct [vrna_elem_prob_s](#) **vrna_plist_t**
Typename for the base pair list representing data structure [vrna_elem_prob_s](#).
- typedef struct [vrna_bp_stack_s](#) **vrna_bp_stack_t**
Typename for the base pair stack representing data structure [vrna_bp_stack_s](#).
- typedef struct [vrna_cpair_s](#) **vrna_cpair_t**
Typename for data structure [vrna_cpair_s](#).
- typedef struct [vrna_sect_s](#) **vrna_sect_t**
Typename for stack of partial structures [vrna_sect_s](#).
- typedef double **FLT_OR_DBL**
Typename for floating point number in partition function computations.
- typedef struct [vrna_basepair_s](#) **PAIR**
Old typename of [vrna_basepair_s](#).
- typedef struct [vrna_elem_prob_s](#) **plist**
Old typename of [vrna_elem_prob_s](#).
- typedef struct [vrna_cpair_s](#) **cpair**
Old typename of [vrna_cpair_s](#).
- typedef struct [vrna_sect_s](#) **sect**
Old typename of [vrna_sect_s](#).
- typedef struct [vrna_bp_stack_s](#) **bondT**
Old typename of [vrna_bp_stack_s](#).
- typedef struct [pu_contrib](#) **pu_contrib**
contributions to p_u
- typedef struct [interact](#) **interact**
interaction data structure for RNAup
- typedef struct [pu_out](#) **pu_out**
Collection of all free_energy of beeing unpaired values for output.
- typedef struct [constrain](#) **constrain**
constraints for cofolding
- typedef struct [node](#) **folden**
Data structure for RNAsnoop (fold energy list)
- typedef struct [dupVar](#) **dupVar**
Data structure used in RNApkplex.

Functions

- void [vrna_C11_features](#) (void)
Dummy symbol to check whether the library was build using C11/C++11 features.

18.159.1 Detailed Description

Various data structures and pre-processor macros.

18.160 basic.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_H
00002 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_H
00003
00018 /* below are several convenience typedef's we use throughout the ViennaRNA library */
00019
00021 typedef struct vrna_basepair_s vrna_basepair_t;
00022
00024 typedef struct vrna_elem_prob_s vrna_plist_t;
00025
00027 typedef struct vrna_bp_stack_s vrna_bp_stack_t;
00028
00030 typedef struct vrna_cpair_s vrna_cpair_t;
00031
00033 typedef struct vrna_sect_s vrna_sect_t;
00034
00035 typedef struct vrna_data_linear_s vrna_data_lin_t;
00036
00037 typedef struct vrna_color_s vrna_color_t;
00038
00040 #ifdef USE_FLOAT_PF
00041 typedef float FLT_OR_DBL;
00042 #else
00043 typedef double FLT_OR_DBL;
00044 #endif
00045
00046
00047 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00048
00049 /* the following typedefs are for backward compatibility only */
00050
00055 typedef struct vrna_basepair_s PAIR;
00056
00061 typedef struct vrna_elem_prob_s plist;
00066 typedef struct vrna_cpair_s cpair;
00067
00072 typedef struct vrna_sect_s sect;
00073
00078 typedef struct vrna_bp_stack_s bondT;
00079
00080 #endif
00081
00082 #include <ViennaRNA/params/constants.h>
00083 #include <ViennaRNA/fold_compound.h>
00084 #include <ViennaRNA/model.h>
00085 #include <ViennaRNA/params/basic.h>
00086 #include <ViennaRNA/dp_matrices.h>
00087 #include <ViennaRNA/constraints/hard.h>
00088 #include <ViennaRNA/constraints/soft.h>
00089 #include <ViennaRNA/grammar.h>
00090 #include "ViennaRNA/structured_domains.h"
00091 #include "ViennaRNA/unstructured_domains.h"
00092 #include "ViennaRNA/utils/structures.h"
00093
00094 /*
00095  * #####
00096  * Here are the type definitions of various datastructures
00097  * shared among the Vienna RNA Package
00098  * #####
00099  */
00100
00104 struct vrna_basepair_s {
00105     int i;
00106     int j;
00107 };
00108
00112 struct vrna_cpair_s {
00113     int i, j, mfe;
00114     float p, hue, sat;
00115     int type;
00116 };
00117
00118 struct vrna_color_s {
00119     float hue;
00120     float sat;
00121     float bri;
00122 };
00123
00124 struct vrna_data_linear_s {
00125     unsigned int position;
00126     float value;
00127     vrna_color_t color;
00128 };
00129

```

```

00130
00134 struct vrna_sect_s {
00135     int i;
00136     int j;
00137     int ml;
00138 };
00139
00143 struct vrna_bp_stack_s {
00144     unsigned int i;
00145     unsigned int j;
00146 };
00147
00148
00149 /*
00150  * #####
00151  * RNAup data structures
00152  * #####
00153  */
00154
00158 typedef struct pu_contrib {
00159     double **H;
00160     double **I;
00161     double **M;
00162     double **E;
00163     int length;
00164     int w;
00165 } pu_contrib;
00166
00170 typedef struct interact {
00171     double *Pi;
00172     double *Gi;
00173     double Gikjl;
00175     double Gikjl_wo;
00176     int i;
00177     int k;
00178     int j;
00179     int l;
00180     int length;
00181 } interact;
00182
00186 typedef struct pu_out {
00187     int len;
00188     int u_vals;
00189     int contribs;
00190     char **header;
00191     double **u_values;
00192 } pu_out;
00193
00197 typedef struct constrain {
00198     int *indx;
00199     char *ptype;
00200 } constrain;
00201
00202 /*
00203  * #####
00204  * RNAduplex data structures
00205  * #####
00206  */
00207
00211 typedef struct {
00212     int i;
00213     int j;
00214     int end;
00215     char *structure;
00216     double energy;
00217     double energy_backtrack;
00218     double opening_backtrack_x;
00219     double opening_backtrack_y;
00220     int offset;
00221     double dG1;
00222     double dG2;
00223     double ddG;
00224     int tb;
00225     int te;
00226     int qb;
00227     int qe;
00228 } duplexT;
00229
00230 /*
00231  * #####
00232  * RNAsnoop data structures
00233  * #####
00234  */
00235
00239 typedef struct node {
00240     int k;
00241     int energy;

```

```

00242     struct node *next;
00243 } folden;
00244
00248 typedef struct {
00249     int i;
00250     int j;
00251     int u;
00252     char *structure;
00253     float energy;
00254     float Duplex_El;
00255     float Duplex_Er;
00256     float Loop_E;
00257     float Loop_D;
00258     float pscd;
00259     float psct;
00260     float pscg;
00261     float Duplex_Ol;
00262     float Duplex_Or;
00263     float Duplex_Ot;
00264     float fullStemEnergy;
00265 } snoopT;
00266
00267
00268 /*
00269  * #####
00270  * PKplex data structures
00271  * #####
00272  */
00273
00277 typedef struct dupVar {
00278     int i;
00279     int j;
00280     int end;
00281     char *pk_helix;
00282     char *structure;
00283     double energy;
00284     int offset;
00285     double dG1;
00286     double dG2;
00287     double ddG;
00288     int tb;
00289     int te;
00290     int qb;
00291     int qe;
00292     int inactive;
00293     int processed;
00294 } dupVar;
00295
00326 #ifndef VRNA_DISABLE_C11_FEATURES
00327 void vrna_C11_features(void);
00328
00329
00330 #endif
00331
00337 #endif

```

18.161 ViennaRNA/params/basic.h File Reference

Functions to deal with sets of energy parameters.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_param_s](#)
The datastructure that contains temperature scaled energy parameters. [More...](#)
- struct [vrna_exp_param_s](#)
The data structure that contains temperature scaled Boltzmann weights of the energy parameters. [More...](#)

Typedefs

- typedef struct [vrna_param_s](#) [vrna_param_t](#)
Typename for the free energy parameter data structure [vrna_params](#).
- typedef struct [vrna_exp_param_s](#) [vrna_exp_param_t](#)
Typename for the Boltzmann factor data structure [vrna_exp_params](#).

- typedef struct `vrna_param_s` paramT
Old typename of `vrna_param_s`.
- typedef struct `vrna_exp_param_s` pf_paramT
Old typename of `vrna_exp_param_s`.

Functions

- `vrna_param_t * vrna_params` (`vrna_md_t *md`)
Get a data structure containing prescaled free energy parameters.
- `vrna_param_t * vrna_params_copy` (`vrna_param_t *par`)
Get a copy of the provided free energy parameters.
- `vrna_exp_param_t * vrna_exp_params` (`vrna_md_t *md`)
Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors.
- `vrna_exp_param_t * vrna_exp_params_comparative` (unsigned int `n_seq`, `vrna_md_t *md`)
Get a data structure containing prescaled free energy parameters already transformed to Boltzmann factors (alifold version)
- `vrna_exp_param_t * vrna_exp_params_copy` (`vrna_exp_param_t *par`)
Get a copy of the provided free energy parameters (provided as Boltzmann factors)
- void `vrna_params_subst` (`vrna_fold_compound_t *vc`, `vrna_param_t *par`)
Update/Reset energy parameters data structure within a `vrna_fold_compound_t`.
- void `vrna_exp_params_subst` (`vrna_fold_compound_t *vc`, `vrna_exp_param_t *params`)
Update the energy parameters for subsequent partition function computations.
- void `vrna_exp_params_rescale` (`vrna_fold_compound_t *vc`, double `*mfe`)
Rescale Boltzmann factors for partition function computations.
- void `vrna_params_reset` (`vrna_fold_compound_t *vc`, `vrna_md_t *md_p`)
Reset free energy parameters within a `vrna_fold_compound_t` according to provided, or default model details.
- void `vrna_exp_params_reset` (`vrna_fold_compound_t *vc`, `vrna_md_t *md_p`)
Reset Boltzmann factors for partition function computations within a `vrna_fold_compound_t` according to provided, or default model details.
- `vrna_exp_param_t * get_scaled_pf_parameters` (void)
- `vrna_exp_param_t * get_boltzmann_factors` (double `temperature`, double `betaScale`, `vrna_md_t md`, double `pf_scale`)
Get precomputed Boltzmann factors of the loop type dependent energy contributions with independent thermodynamic temperature.
- `vrna_exp_param_t * get_boltzmann_factor_copy` (`vrna_exp_param_t *parameters`)
Get a copy of already precomputed Boltzmann factors.
- `vrna_exp_param_t * get_scaled_alipf_parameters` (unsigned int `n_seq`)
Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant)
- `vrna_exp_param_t * get_boltzmann_factors_ali` (unsigned int `n_seq`, double `temperature`, double `betaScale`, `vrna_md_t md`, double `pf_scale`)
Get precomputed Boltzmann factors of the loop type dependent energy contributions (alifold variant) with independent thermodynamic temperature.
- `vrna_param_t * scale_parameters` (void)
Get precomputed energy contributions for all the known loop types.
- `vrna_param_t * get_scaled_parameters` (double `temperature`, `vrna_md_t md`)
Get precomputed energy contributions for all the known loop types.

18.161.1 Detailed Description

Functions to deal with sets of energy parameters.

18.162 basic.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00036 typedef struct vrna_param_s vrna_param_t;
00038 typedef struct vrna_exp_param_s vrna_exp_param_t;
00039
00040 #include <ViennaRNA/params/constants.h>
00041 #include <ViennaRNA/datastructures/basic.h>
00042 #include <ViennaRNA/fold_compound.h>
00043 #include <ViennaRNA/model.h>
00044
00045 #define VRNA_GQUAD_MAX_STACK_SIZE 7
00046 #define VRNA_GQUAD_MIN_STACK_SIZE 2
00047 #define VRNA_GQUAD_MAX_LINKER_LENGTH 15
00048 #define VRNA_GQUAD_MIN_LINKER_LENGTH 1
00049 #define VRNA_GQUAD_MIN_BOX_SIZE ((4 * VRNA_GQUAD_MIN_STACK_SIZE) + \
00050 (3 * VRNA_GQUAD_MIN_LINKER_LENGTH))
00051 #define VRNA_GQUAD_MAX_BOX_SIZE ((4 * VRNA_GQUAD_MAX_STACK_SIZE) + \
00052 (3 * VRNA_GQUAD_MAX_LINKER_LENGTH))
00053
00057 struct vrna_param_s {
00058     int id;
00059     int stack[NBPAIRS + 1][NBPAIRS + 1];
00060     int hairpin[31];
00061     int bulge[MAXLOOP + 1];
00062     int internal_loop[MAXLOOP + 1];
00063     int mismatchExt[NBPAIRS + 1][5][5];
00064     int mismatchI[NBPAIRS + 1][5][5];
00065     int mismatchlnI[NBPAIRS + 1][5][5];
00066     int mismatch23I[NBPAIRS + 1][5][5];
00067     int mismatchH[NBPAIRS + 1][5][5];
00068     int mismatchM[NBPAIRS + 1][5][5];
00069     int dangle5[NBPAIRS + 1][5];
00070     int dangle3[NBPAIRS + 1][5];
00071     int int11[NBPAIRS + 1][NBPAIRS + 1][5][5];
00072     int int21[NBPAIRS + 1][NBPAIRS + 1][5][5][5];
00073     int int22[NBPAIRS + 1][NBPAIRS + 1][5][5][5][5];
00074     int ninio[5];
00075     double lxc;
00076     int MLbase;
00077     int MLintern[NBPAIRS + 1];
00078     int MLclosing;
00079     int TerminalAU;
00080     int DuplexInit;
00081     int Tetraloop_E[200];
00082     char Tetraloops[1401];
00083     int Triloop_E[40];
00084     char Triloops[241];
00085     int Hexaloop_E[40];
00086     char Hexaloops[1801];
00087     int TripleC;
00088     int MultipleCA;
00089     int MultipleCB;
00090     int gquad[VRNA_GQUAD_MAX_STACK_SIZE + 1][3 * VRNA_GQUAD_MAX_LINKER_LENGTH + 1];
00091     int gquadLayerMismatch;
00092     int gquadLayerMismatchMax;
00093
00094     double temperature;
00096     vrna_md_t model_details;
00097     char param_file[256];
00098     int SaltStack;
00099     int SaltLoop[MAXLOOP + 2];
00100     double SaltLoopDb1[MAXLOOP + 2];
00101     int SaltMLbase;
00102     int SaltMLintern;
00103     int SaltMLclosing;
00104     int SaltDPXInit;
00105 };
00106
00110 struct vrna_exp_param_s {
00111     int id;

```

```

00114 double expstack[NBPAIRS + 1][NBPAIRS + 1];
00115 double exphairpin[31];
00116 double expbulge[MAXLOOP + 1];
00117 double expinternal[MAXLOOP + 1];
00118 double expmismatchExt[NBPAIRS + 1][5][5];
00119 double expmismatchI[NBPAIRS + 1][5][5];
00120 double expmismatch23I[NBPAIRS + 1][5][5];
00121 double expmismatchlnI[NBPAIRS + 1][5][5];
00122 double expmismatchH[NBPAIRS + 1][5][5];
00123 double expmismatchM[NBPAIRS + 1][5][5];
00124 double expdangle5[NBPAIRS + 1][5];
00125 double expdangle3[NBPAIRS + 1][5];
00126 double expint11[NBPAIRS + 1][NBPAIRS + 1][5][5];
00127 double expint21[NBPAIRS + 1][NBPAIRS + 1][5][5][5];
00128 double expint22[NBPAIRS + 1][NBPAIRS + 1][5][5][5][5];
00129 double expninio[5][MAXLOOP + 1];
00130 double lxc;
00131 double expMLbase;
00132 double expMLintern[NBPAIRS + 1];
00133 double expMLclosing;
00134 double expTermAU;
00135 double expDuplexInit;
00136 double expetra[40];
00137 double exptri[40];
00138 double exphex[40];
00139 char Tetraloops[1401];
00140 double expTriloop[40];
00141 char Triloops[241];
00142 char Hexaloops[1801];
00143 double expTripleC;
00144 double expMultipleCA;
00145 double expMultipleCB;
00146 double expgquad[VRNA_GQUAD_MAX_STACK_SIZE + 1][3 * VRNA_GQUAD_MAX_LINKER_LENGTH + 1];
00147 double expgquadLayerMismatch;
00148 int gquadLayerMismatchMax;
00149
00150 double kT;
00151 double pf_scale;
00153 double temperature;
00154 double alpha;
00161 vrna_md_t model_details;
00162 char param_file[256];
00164 double expSaltStack;
00165 double expSaltLoop[MAXLOOP + 2];
00166 double SaltLoopDbl[MAXLOOP + 2];
00167 int SaltMLbase;
00168 int SaltMLintern;
00169 int SaltMLclosing;
00170 int SaltDPXInit;
00171 };
00172
00173
00185 vrna_param_t *
00186 vrna_params(vrna_md_t *md);
00187
00188
00200 vrna_param_t *
00201 vrna_params_copy(vrna_param_t *par);
00202
00203
00226 vrna_exp_param_t *
00227 vrna_exp_params(vrna_md_t *md);
00228
00229
00243 vrna_exp_param_t *
00244 vrna_exp_params_comparative(unsigned int n_seq,
00245                             vrna_md_t *md);
00246
00247
00259 vrna_exp_param_t *
00260 vrna_exp_params_copy(vrna_exp_param_t *par);
00261
00262
00275 void
00276 vrna_params_subst(vrna_fold_compound_t *vc,
00277                  vrna_param_t *par);
00278
00279
00297 void
00298 vrna_exp_params_subst(vrna_fold_compound_t *vc,
00299                      vrna_exp_param_t *params);
00300
00301
00339 void
00340 vrna_exp_params_rescale(vrna_fold_compound_t *vc,
00341                        double *mfe);
00342

```

```

00343
00357 void
00358 vrna_params_reset(vrna_fold_compound_t *vc,
00359                  vrna_md_t *md_p);
00360
00361
00376 void
00377 vrna_exp_params_reset(vrna_fold_compound_t *vc,
00378                      vrna_md_t *md_p);
00379
00380
00381 void
00382 vrna_params_prepare(vrna_fold_compound_t *vc,
00383                    unsigned int options);
00384
00385
00386 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00387
00392 typedef struct vrna_param_s paramT;
00393
00398 typedef struct vrna_exp_param_s pf_paramT;
00399
00400 DEPRECATED(vrna_param_t *get_parameter_copy(vrna_param_t *par),
00401            "Use vrna_params_copy() instead");
00402
00412 DEPRECATED(vrna_exp_param_t *get_scaled_pf_parameters(void),
00413            "Use vrna_params() instead");
00414
00440 DEPRECATED(vrna_exp_param_t *get_boltzmann_factors(double temperature,
00441                                                    double betaScale,
00442                                                    vrna_md_t md,
00443                                                    double pf_scale),
00444            "Use vrna_exp_params() instead");
00445
00456 DEPRECATED(vrna_exp_param_t *get_boltzmann_factor_copy(vrna_exp_param_t *parameters),
00457            "Use vrna_exp_params_copy() instead");
00458
00466 DEPRECATED(vrna_exp_param_t *get_scaled_alipf_parameters(unsigned int n_seq,
00467            "Use vrna_exp_params_comparative() instead");
00468
00477 DEPRECATED(vrna_exp_param_t *get_boltzmann_factors_alipf(unsigned int n_seq,
00478                                                           double temperature,
00479                                                           double betaScale,
00480                                                           vrna_md_t md,
00481                                                           double pf_scale),
00482            "Use vrna_exp_params_comparative() instead");
00483
00495 DEPRECATED(vrna_param_t *scale_parameters(void),
00496            "Use vrna_params() instead");
00497
00514 DEPRECATED(vrna_param_t *get_scaled_parameters(double temperature,
00515            vrna_md_t md),
00516            "Use vrna_params() instead");
00517
00518 DEPRECATED(vrna_param_t *copy_parameters(void), "Use vrna_params_copy() instead");
00519 DEPRECATED(vrna_param_t *set_parameters(vrna_param_t *dest), "Use vrna_params_copy() instead");
00520 DEPRECATED(vrna_exp_param_t *scale_pf_parameters(void), "Use vrna_exp_params() instead");
00521 DEPRECATED(vrna_exp_param_t *copy_pf_param(void), "Use vrna_exp_params_copy() instead");
00522 DEPRECATED(vrna_exp_param_t *set_pf_param(vrna_param_t *dest),
00523            "Use vrna_exp_params_copy() instead");
00524
00525 #endif
00526
00532 #endif

```

18.163 ViennaRNA/utils/basic.h File Reference

General utility- and helper-functions used throughout the *ViennaRNA Package*.

Include dependency graph for basic.h: This graph shows which files directly or indirectly include this file:

Macros

- **#define VRNA_INPUT_ERROR 1U**
Output flag of *get_input_line()*: "An ERROR has occurred, maybe EOF".
- **#define VRNA_INPUT_QUIT 2U**
Output flag of *get_input_line()*: "the user requested quitting the program".
- **#define VRNA_INPUT_MISC 4U**
Output flag of *get_input_line()*: "something was read".

- `#define VRNA_INPUT_FASTA_HEADER 8U`
*Input/Output flag of `get_input_line()`:
if used as input option this tells `get_input_line()` that the data to be read should comply with the FASTA format.*
- `#define VRNA_INPUT_CONSTRAINT 32U`
*Input flag for `get_input_line()`:
Tell `get_input_line()` that we assume to read a structure constraint.*
- `#define VRNA_INPUT_NO_TRUNCATION 256U`
Input switch for `get_input_line()`: "do not truncate the line by eliminating white spaces at end of line".
- `#define VRNA_INPUT_NO_REST 512U`
Input switch for `vrna_file_fasta_read_record()`: "do fill rest array".
- `#define VRNA_INPUT_NO_SPAN 1024U`
Input switch for `vrna_file_fasta_read_record()`: "never allow data to span more than one line".
- `#define VRNA_INPUT_NOSKIP_BLANK_LINES 2048U`
Input switch for `vrna_file_fasta_read_record()`: "do not skip empty lines".
- `#define VRNA_INPUT_BLANK_LINE 4096U`
Output flag for `vrna_file_fasta_read_record()`: "read an empty line".
- `#define VRNA_INPUT_NOSKIP_COMMENTS 128U`
Input switch for `get_input_line()`: "do not skip comment lines".
- `#define VRNA_INPUT_COMMENT 8192U`
Output flag for `vrna_file_fasta_read_record()`: "read a comment".
- `#define MIN2(A, B) ((A) < (B) ? (A) : (B))`
Get the minimum of two comparable values.
- `#define MAX2(A, B) ((A) > (B) ? (A) : (B))`
Get the maximum of two comparable values.
- `#define MIN3(A, B, C) (MIN2((MIN2((A), (B))), (C)))`
Get the minimum of three comparable values.
- `#define MAX3(A, B, C) (MAX2((MAX2((A), (B))), (C)))`
Get the maximum of three comparable values.

Functions

- `void * vrna_alloc` (unsigned size)
Allocate space safely.
- `void * vrna_realloc` (void *p, unsigned size)
Reallocate space safely.
- `void vrna_init_rand` (void)
Initialize seed for random number generator.
- `void vrna_init_rand_seed` (unsigned int seed)
Initialize the random number generator with a pre-defined seed.
- `double vrna_urn` (void)
get a random number from [0..1]
- `int vrna_int_urn` (int from, int to)
Generates a pseudo random integer in a specified range.
- `char * vrna_time_stamp` (void)
Get a timestamp.
- `unsigned int get_input_line` (char **string, unsigned int options)
- `int * vrna_idx_row_wise` (unsigned int length)
Get an index mapper array (iidx) for accessing the energy matrices, e.g. in partition function related functions.
- `int * vrna_idx_col_wise` (unsigned int length)
Get an index mapper array (indx) for accessing the energy matrices, e.g. in MFE related functions.
- `void vrna_message_error` (const char *format,...)

- Print an error message and die.*
- void [vrna_message_verror](#) (const char *format, va_list args)
- Print an error message and die.*
- void [vrna_message_warning](#) (const char *format,...)
- Print a warning message.*
- void [vrna_message_vwarning](#) (const char *format, va_list args)
- Print a warning message.*
- void [vrna_message_info](#) (FILE *fp, const char *format,...)
- Print an info message.*
- void [vrna_message_vinfo](#) (FILE *fp, const char *format, va_list args)
- Print an info message.*
- void [vrna_message_input_seq_simple](#) (void)
- Print a line to stdout that asks for an input sequence.*
- void [vrna_message_input_seq](#) (const char *s)
- Print a line with a user defined string and a ruler to stdout.*
- char * [get_line](#) (FILE *fp)
- Read a line of arbitrary length from a stream.*
- void [print_tty_input_seq](#) (void)
- Print a line to stdout that asks for an input sequence.*
- void [print_tty_input_seq_str](#) (const char *s)
- Print a line with a user defined string and a ruler to stdout.*
- void [warn_user](#) (const char message[])
- Print a warning message.*
- void [nrerror](#) (const char message[])
- Die with an error message.*
- void * [space](#) (unsigned size)
- Allocate space safely.*
- void * [xrealloc](#) (void *p, unsigned size)
- Reallocate space safely.*
- void [init_rand](#) (void)
- Make random number seeds.*
- double [urn](#) (void)
- get a random number from [0..1]*
- int [int_urn](#) (int from, int to)
- Generates a pseudo random integer in a specified range.*
- void [filecopy](#) (FILE *from, FILE *to)
- Inefficient cp*
- char * [time_stamp](#) (void)
- Get a timestamp.*

Variables

- unsigned short [xsubi](#) [3]
- Current 48 bit random number.*

18.163.1 Detailed Description

General utility- and helper-functions used throughout the *ViennaRNA Package*.

18.163.2 Function Documentation

18.163.2.1 get_line()

```
char * get_line (
    FILE * fp )
```

Read a line of arbitrary length from a stream.

Returns a pointer to the resulting string. The necessary memory is allocated and should be released using *free()* when the string is no longer needed.

Deprecated Use [vrna_read_line\(\)](#) as a substitute!

Parameters

<i>fp</i>	A file pointer to the stream where the function should read from
-----------	--

Returns

A pointer to the resulting string

18.163.2.2 print_tty_input_seq()

```
void print_tty_input_seq (
    void )
```

Print a line to *stdout* that asks for an input sequence.

There will also be a ruler (scale line) printed that helps orientation of the sequence positions

Deprecated Use [vrna_message_input_seq_simple\(\)](#) instead!

18.163.2.3 print_tty_input_seq_str()

```
void print_tty_input_seq_str (
    const char * s )
```

Print a line with a user defined string and a ruler to *stdout*.

(usually this is used to ask for user input) There will also be a ruler (scale line) printed that helps orientation of the sequence positions

Deprecated Use [vrna_message_input_seq\(\)](#) instead!

18.163.2.4 warn_user()

```
void warn_user (
    const char message[] )
```

Print a warning message.

Print a warning message to *stderr*

Deprecated Use [vrna_message_warning\(\)](#) instead!

18.163.2.5 nrerror()

```
void nrerror (
    const char message[] )
```

Die with an error message.

Deprecated Use [vrna_message_error\(\)](#) instead!

18.163.2.6 space()

```
void * space (
    unsigned size )
```

Allocate space safely.

Deprecated Use `vrna_alloc()` instead!

18.163.2.7 xrealloc()

```
void * xrealloc (
    void * p,
    unsigned size )
```

Reallocate space safely.

Deprecated Use `vrna_realloc()` instead!

18.163.2.8 init_rand()

```
void init_rand (
    void )
```

Make random number seeds.

Deprecated Use `vrna_init_rand()` instead!

18.163.2.9 urn()

```
double urn (
    void )
```

get a random number from [0..1]

Deprecated Use `vrna_urn()` instead!

18.163.2.10 int_urn()

```
int int_urn (
    int from,
    int to )
```

Generates a pseudo random integer in a specified range.

Deprecated Use `vrna_int_urn()` instead!

18.163.2.11 filecopy()

```
void filecopy (
    FILE * from,
    FILE * to )
```

Inefficient `cp`

Deprecated Use `vrna_file_copy()` instead!

18.163.2.12 time_stamp()

```
char * time_stamp (
    void )
```

Get a timestamp.

Deprecated Use `vrna_time_stamp()` instead!

18.164 basic.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_UTILS_H
00002 #define VIENNA_RNA_PACKAGE_UTILS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00027 /* two helper macros to indicate whether a function should be exported in
00028  * the library or stays hidden */
00029 #define PUBLIC
00030 #define PRIVATE static
00031
00035 #define VRNA_INPUT_ERROR 1U
00039 #define VRNA_INPUT_QUIT 2U
00043 #define VRNA_INPUT_MISC 4U
00044
00052 #define VRNA_INPUT_FASTA_HEADER 8U
00053
00054 /*
00055  * @brief Input flag for get_input_line():\n
00056  * Tell get_input_line() that we assume to read a nucleotide sequence
00057  */
00058 #define VRNA_INPUT_SEQUENCE 16U
00060
00065 #define VRNA_INPUT_CONSTRAINT 32U
00066
00071 #define VRNA_INPUT_NO_TRUNCATION 256U
00072
00076 #define VRNA_INPUT_NO_REST 512U
00077
00081 #define VRNA_INPUT_NO_SPAN 1024U
00082
00086 #define VRNA_INPUT_NOSKIP_BLANK_LINES 2048U
00087
00091 #define VRNA_INPUT_BLANK_LINE 4096U
00092
00096 #define VRNA_INPUT_NOSKIP_COMMENTS 128U
00097
00101 #define VRNA_INPUT_COMMENT 8192U
00102
00106 #define MIN2(A, B) ((A) < (B) ? (A) : (B))
00107
00111 #define MAX2(A, B) ((A) > (B) ? (A) : (B))
00112
00116 #define MIN3(A, B, C) (MIN2((MIN2((A), (B))), (C)))
00117
00121 #define MAX3(A, B, C) (MAX2((MAX2((A), (B))), (C)))
00122
00123 #include <stdio.h>
00124 #include <stdarg.h>
00125
00126 #include <ViennaRNA/datastructures/basic.h>
00127
00128
00129 #ifdef WITH_DMALLOC
00130 /* use dmalloc library to check for memory management bugs */
00131 #include "dmalloc.h"
00132 #define vrna_alloc(S) calloc(1, (S))
00133 #define vrna_realloc(p, S) xrealloc(p, S)
00134 #else
00135
00142 void *
```

```
00143 vrna_alloc(unsigned size);
00144
00145
00153 void *
00154 vrna_realloc(void *p,
00155             unsigned size);
00156
00157
00158 #endif
00159
00165 void
00166 vrna_init_rand(void);
00167
00168
00176 void
00177 vrna_init_rand_seed(unsigned int seed);
00178
00179
00188 extern unsigned short xsubi[3];
00189
00197 double
00198 vrna_urn(void);
00199
00200
00209 int
00210 vrna_int_urn(int from,
00211            int to);
00212
00213
00222 char *
00223 vrna_time_stamp(void);
00224
00225
00246 unsigned int
00247 get_input_line(char **string,
00248               unsigned int options);
00249
00250
00264 int *
00265 vrna_idx_row_wise(unsigned int length);
00266
00267
00282 int *
00283 vrna_idx_col_wise(unsigned int length);
00284
00285
00308 void
00309 vrna_message_error(const char *format,
00310                  ...);
00311
00312
00325 void
00326 vrna_message_verror(const char *format,
00327                   va_list args);
00328
00329
00341 void
00342 vrna_message_warning(const char *format,
00343                   ...);
00344
00345
00357 void
00358 vrna_message_vwarning(const char *format,
00359                   va_list args);
00360
00361
00373 void
00374 vrna_message_info(FILE *fp,
00375                  const char *format,
00376                  ...);
00377
00378
00390 void
00391 vrna_message_vinfo(FILE *fp,
00392                  const char *format,
00393                  va_list args);
00394
00395
00401 void
00402 vrna_message_input_seq_simple(void);
00403
00404
00413 void
00414 vrna_message_input_seq(const char *s);
00415
00416
00417 void
```

```

00418 vrna_message_input_msa(const char *s);
00419
00420
00425 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00426
00427 DEPRECATED(int *get_idx(unsigned int length), "Use vrna_idx_col_wise() instead");
00428
00429 DEPRECATED(int *get_iidx(unsigned int length), "Use vrna_idx_row_wise() instead");
00430
00443 DEPRECATED(char *get_line(FILE *fp), "Use vrna_read_line() instead");
00444
00451 DEPRECATED(void print_tty_input_seq(void), "Use vrna_message_input_seq_simple() instead");
00452
00461 DEPRECATED(void print_tty_input_seq_str(const char *s), "Use vrna_message_input_seq() instead");
00462
00470 DEPRECATED(void warn_user(const char message[]), "Use vrna_message_warning() instead");
00471
00477 DEPRECATED(void nrerror(const char message[]), "Use vrna_message_error() instead()");
00478
00484 DEPRECATED(void *space(unsigned size), "Use vrna_alloc() instead");
00485
00491 DEPRECATED(void *xrealloc(void *p,
00492                             unsigned size), "Use vrna_realloc() instead");
00493
00498 DEPRECATED(void init_rand(void), "Use vrna_init_rand() instead");
00499
00505 DEPRECATED(double urn(void), "Use vrna_urn() instead");
00506
00512 DEPRECATED(int int_urn(int from,
00513                        int to), "Use vrna_int_urn() instead()");
00514
00520 DEPRECATED(void filecopy(FILE *from,
00521                          FILE *to), "Use vrna_file_copy() instead");
00522
00528 DEPRECATED(char *time_stamp(void), "Use vrna_time_stamp() instead");
00529
00530 #endif
00531
00532 #endif

```

18.165 ViennaRNA/params/constants.h File Reference

Energy parameter constants.

Include dependency graph for constants.h: This graph shows which files directly or indirectly include this file:

Macros

- #define **GASCONST** 1.98717 /* in [cal/K] */
- #define **K0** 273.15
- #define **INF** 10000000 /* (INT_MAX/10) */
- #define **FORBIDDEN** 9999
- #define **BONUS** 10000
- #define **NBPAIRS** 7
- #define **TURN** 3
- #define **MAXLOOP** 30

18.165.1 Detailed Description

Energy parameter constants.

18.165.2 Macro Definition Documentation

18.165.2.1 GASCONST

```
#define GASCONST 1.98717 /* in [cal/K] */
```

The gas constant

18.165.2.2 K0

```
#define K0 273.15
0 deg Celsius in Kelvin
```

18.165.2.3 INF

```
#define INF 10000000 /* (INT_MAX/10) */
Infinity as used in minimization routines
```

18.165.2.4 FORBIDDEN

```
#define FORBIDDEN 9999
forbidden
```

18.165.2.5 BONUS

```
#define BONUS 10000
bonus contribution
```

18.165.2.6 NBPAIRS

```
#define NBPAIRS 7
The number of distinguishable base pairs
```

18.165.2.7 TURN

```
#define TURN 3
The minimum loop length
```

18.165.2.8 MAXLOOP

```
#define MAXLOOP 30
The maximum loop length
```

18.166 constants.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_CONSTANTS_H
00003
00004 #include <limits.h>
00005
00013 #define GASCONST 1.98717 /* in [cal/K] */
00015 #define K0 273.15
00017 #define INF 10000000 /* (INT_MAX/10) */
00018
00019 #define EMAX (INF/10)
00021 #define FORBIDDEN 9999
00023 #define BONUS 10000
00025 #define NBPAIRS 7
00027 #define TURN 3
00029 #define MAXLOOP 30
00030
00031 #define UNIT 100
00032
00033 #define MINPScore -2 * UNIT
00034
00035 #endif
```

18.167 ViennaRNA/params/convert.h File Reference

Functions and definitions for energy parameter file format conversion.

This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_CONVERT_OUTPUT_ALL 1U`
- `#define VRNA_CONVERT_OUTPUT_HP 2U`
- `#define VRNA_CONVERT_OUTPUT_STACK 4U`
- `#define VRNA_CONVERT_OUTPUT_MM_HP 8U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT 16U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_1N 32U`
- `#define VRNA_CONVERT_OUTPUT_MM_INT_23 64U`
- `#define VRNA_CONVERT_OUTPUT_MM_MULTI 128U`
- `#define VRNA_CONVERT_OUTPUT_MM_EXT 256U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE5 512U`
- `#define VRNA_CONVERT_OUTPUT_DANGLE3 1024U`
- `#define VRNA_CONVERT_OUTPUT_INT_11 2048U`
- `#define VRNA_CONVERT_OUTPUT_INT_21 4096U`
- `#define VRNA_CONVERT_OUTPUT_INT_22 8192U`
- `#define VRNA_CONVERT_OUTPUT_BULGE 16384U`
- `#define VRNA_CONVERT_OUTPUT_INT 32768U`
- `#define VRNA_CONVERT_OUTPUT_ML 65536U`
- `#define VRNA_CONVERT_OUTPUT_MISC 131072U`
- `#define VRNA_CONVERT_OUTPUT_SPECIAL_HP 262144U`
- `#define VRNA_CONVERT_OUTPUT_VANILLA 524288U`
- `#define VRNA_CONVERT_OUTPUT_NINIO 1048576U`
- `#define VRNA_CONVERT_OUTPUT_DUMP 2097152U`

Functions

- void `convert_parameter_file` (const char *iname, const char *oname, unsigned int options)

18.167.1 Detailed Description

Functions and definitions for energy parameter file format conversion.

18.168 convert.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_CONVERT_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_CONVERT_H
00003
00023 #define VRNA_CONVERT_OUTPUT_ALL          1U
00025 #define VRNA_CONVERT_OUTPUT_HP          2U
00027 #define VRNA_CONVERT_OUTPUT_STACK        4U
00029 #define VRNA_CONVERT_OUTPUT_MM_HP        8U
00031 #define VRNA_CONVERT_OUTPUT_MM_INT       16U
00033 #define VRNA_CONVERT_OUTPUT_MM_INT_1N    32U
00035 #define VRNA_CONVERT_OUTPUT_MM_INT_23    64U
00037 #define VRNA_CONVERT_OUTPUT_MM_MULTI     128U
00039 #define VRNA_CONVERT_OUTPUT_MM_EXT       256U
00041 #define VRNA_CONVERT_OUTPUT_DANGLE5      512U
00043 #define VRNA_CONVERT_OUTPUT_DANGLE3      1024U
00045 #define VRNA_CONVERT_OUTPUT_INT_11       2048U
00047 #define VRNA_CONVERT_OUTPUT_INT_21       4096U
00049 #define VRNA_CONVERT_OUTPUT_INT_22       8192U
00051 #define VRNA_CONVERT_OUTPUT_BULGE        16384U
00053 #define VRNA_CONVERT_OUTPUT_INT          32768U
00055 #define VRNA_CONVERT_OUTPUT_ML           65536U
00057 #define VRNA_CONVERT_OUTPUT_MISC         131072U
00059 #define VRNA_CONVERT_OUTPUT_SPECIAL_HP    262144U
00061 #define VRNA_CONVERT_OUTPUT_VANILLA      524288U
00063 #define VRNA_CONVERT_OUTPUT_NINIO        1048576U
00065 #define VRNA_CONVERT_OUTPUT_DUMP         2097152U
00066
00089 void convert_parameter_file(const char *iname,
00090                             const char *oname,
00091                             unsigned int options);
00092
00093
00097 #endif

```


18.169 default.h

```

00001 /*
00002     prototypes for energy_par.c
00003 */
00004
00005 #ifndef VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_H
00006 #define VIENNA_RNA_PACKAGE_PARAMS_DEFAULT_H
00007
00008 #include <ViennaRNA/params/constants.h>
00009
00010 #define PUBLIC
00011
00012
00013 extern double lxc37; /* parameter for logarithmic loop
00014     energy extrapolation */
00015
00016 extern int stack37[NBPAIRS+1][NBPAIRS+1];
00017 extern int stackdH[NBPAIRS+1][NBPAIRS+1]; /* stack enthalpies */
00018
00019 extern int hairpin37[31];
00020 extern int hairpindH[31];
00021 extern int bulge37[31];
00022 extern int bulgedH[31];
00023 extern int internal_loop37[31];
00024 extern int internal_loophH[31];
00025 extern int mismatchI37[NBPAIRS+1][5][5]; /* interior loop mismatches */
00026 extern int mismatchIdH[NBPAIRS+1][5][5]; /* interior loop mismatches */
00027 extern int mismatchlnI37[NBPAIRS+1][5][5]; /* interior loop mismatches */
00028 extern int mismatch23I37[NBPAIRS+1][5][5]; /* interior loop mismatches */
00029 extern int mismatchlnIdH[NBPAIRS+1][5][5]; /* interior loop mismatches */
00030 extern int mismatch23IdH[NBPAIRS+1][5][5]; /* interior loop mismatches */
00031 extern int mismatchH37[NBPAIRS+1][5][5]; /* same for hairpins */
00032 extern int mismatchM37[NBPAIRS+1][5][5]; /* same for multiloops */
00033 extern int mismatchHdH[NBPAIRS+1][5][5]; /* same for hairpins */
00034 extern int mismatchMdH[NBPAIRS+1][5][5]; /* same for multiloops */
00035 extern int mismatchExt37[NBPAIRS+1][5][5];
00036 extern int mismatchExtIdH[NBPAIRS+1][5][5];
00037
00038 extern int dangle5_37[NBPAIRS+1][5]; /* 5' dangle exterior of pair */
00039 extern int dangle3_37[NBPAIRS+1][5]; /* 3' dangle */
00040 extern int dangle3_dH[NBPAIRS+1][5]; /* corresponding enthalpies */
00041 extern int dangle5_dH[NBPAIRS+1][5];
00042
00043 extern int int11_37[NBPAIRS+1][NBPAIRS+1][5][5]; /* 1x1 interior loops */
00044 extern int int11_dH[NBPAIRS+1][NBPAIRS+1][5][5];
00045
00046 extern int int21_37[NBPAIRS+1][NBPAIRS+1][5][5][5]; /* 2x1 interior loops */
00047 extern int int21_dH[NBPAIRS+1][NBPAIRS+1][5][5][5];
00048
00049 extern int int22_37[NBPAIRS+1][NBPAIRS+1][5][5][5][5]; /* 2x2 interior loops */
00050 extern int int22_dH[NBPAIRS+1][NBPAIRS+1][5][5][5][5];
00051
00052 /* constants for linearly destabilizing contributions for multi-loops
00053     F = ML_closing + ML_intern*(k-1) + ML_BASE*u */
00054 extern int ML_BASE37;
00055 extern int ML_BASEdH;
00056 extern int ML_closing37;
00057 extern int ML_closingdH;
00058 extern int ML_intern37;
00059 extern int ML_interndH;
00060
00061 extern int TripleC37;
00062 extern int TripleCdH;
00063 extern int MultipleCA37;
00064 extern int MultipleCAdH;
00065 extern int MultipleCB37;
00066 extern int MultipleCBdH;
00067
00068 /* Ninio-correction for asymmetric internal loops with branches n1 and n2 */
00069 /*     ninio_energy = min{max_ninio, |n1-n2|*F_ninio[min{4.0, n1, n2}] } */
00070 extern int MAX_NINIO; /* maximum correction */
00071 extern int ninio37;
00072 extern int niniodH;
00073 /* penalty for helices terminated by AU (actually not GC) */
00074 extern int TerminalAU37;
00075 extern int TerminalAUdH;
00076 /* penalty for forming bi-molecular duplex */
00077 extern int DuplexInit37;
00078 extern int DuplexInitdH;
00079 /* stabilizing contribution due to special hairpins of size 4 (tetraloops) */
00080 extern char Tetraloops[281]; /* string containing the special tetraloops */
00081 extern int Tetraloop37[40]; /* Bonus energy for special tetraloops */
00082 extern int TetraloopdH[40];
00083 extern char Triloops[241]; /* string containing the special triloops */
00084 extern int Triloop37[40]; /* Bonus energy for special Triloops */
00085 extern int TriloopdH[40]; /* Bonus energy for special Triloops */

```

```

00086 extern char Hexaloops[361];      /* string containing the special triloops */
00087 extern int Hexaloop37[40]; /* Bonus energy for special Triloops */
00088 extern int HexaloopdH[40]; /* Bonus energy for special Triloops */
00089
00090 extern int GQuadAlpha37;
00091 extern int GQuadAlphadH;
00092 extern int GQuadBeta37;
00093 extern int GQuadBetadH;
00094 extern int GQuadLayerMismatch37; /* penalty per incompatible gquad layer in a sub-alignment (applied
twice for inner layers) */
00095 extern int GQuadLayerMismatchH;
00096 extern int GQuadLayerMismatchMax; /* maximum number of mismatching sequences in the alignment when
gquad should be formed */
00097
00098 extern double Tmeasure; /* temperature of param measurements */
00099
00100 #endif

```

18.170 intl11.h

```

00001 PUBLIC int intl1_37[NBPAIRS+1][NBPAIRS+1][5][5] =
00002 {{{ INF, INF, INF, INF, INF}
00003  ,{ INF, INF, INF, INF, INF}
00004  ,{ INF, INF, INF, INF, INF}
00005  ,{ INF, INF, INF, INF, INF}
00006  ,{ INF, INF, INF, INF, INF}
00007  }
00008 ,{{{ INF, INF, INF, INF, INF}
00009  ,{ INF, INF, INF, INF, INF}
00010  ,{ INF, INF, INF, INF, INF}
00011  ,{ INF, INF, INF, INF, INF}
00012  ,{ INF, INF, INF, INF, INF}
00013  }
00014 ,{{{ INF, INF, INF, INF, INF}
00015  ,{ INF, INF, INF, INF, INF}
00016  ,{ INF, INF, INF, INF, INF}
00017  ,{ INF, INF, INF, INF, INF}
00018  ,{ INF, INF, INF, INF, INF}
00019  }
00020 ,{{{ INF, INF, INF, INF, INF}
00021  ,{ INF, INF, INF, INF, INF}
00022  ,{ INF, INF, INF, INF, INF}
00023  ,{ INF, INF, INF, INF, INF}
00024  ,{ INF, INF, INF, INF, INF}
00025  }
00026 ,{{{ INF, INF, INF, INF, INF}
00027  ,{ INF, INF, INF, INF, INF}
00028  ,{ INF, INF, INF, INF, INF}
00029  ,{ INF, INF, INF, INF, INF}
00030  ,{ INF, INF, INF, INF, INF}
00031  }
00032 ,{{{ INF, INF, INF, INF, INF}
00033  ,{ INF, INF, INF, INF, INF}
00034  ,{ INF, INF, INF, INF, INF}
00035  ,{ INF, INF, INF, INF, INF}
00036  ,{ INF, INF, INF, INF, INF}
00037  }
00038 ,{{{ INF, INF, INF, INF, INF}
00039  ,{ INF, INF, INF, INF, INF}
00040  ,{ INF, INF, INF, INF, INF}
00041  ,{ INF, INF, INF, INF, INF}
00042  ,{ INF, INF, INF, INF, INF}
00043  }
00044 ,{{{ INF, INF, INF, INF, INF}
00045  ,{ INF, INF, INF, INF, INF}
00046  ,{ INF, INF, INF, INF, INF}
00047  ,{ INF, INF, INF, INF, INF}
00048  ,{ INF, INF, INF, INF, INF}
00049  }
00050  }
00051 ,{{{ INF, INF, INF, INF, INF}
00052  ,{ INF, INF, INF, INF, INF}
00053  ,{ INF, INF, INF, INF, INF}
00054  ,{ INF, INF, INF, INF, INF}
00055  ,{ INF, INF, INF, INF, INF}
00056  }
00057 ,{{{ 90, 90, 50, 50, 50}
00058  ,{ 90, 90, 50, 50, 50}
00059  ,{ 50, 50, 50, 50, 50}
00060  ,{ 50, 50, 50, -140, 50}
00061  ,{ 50, 50, 50, 50, 40}
00062  }
00063 ,{{{ 90, 90, 50, 50, 60}
00064  ,{ 90, 90, -40, 50, 50}
00065  ,{ 60, 30, 50, 50, 60}

```

```
00066 , { 50, -10, 50, -220, 50}
00067 , { 50, 50, 0, 50, -10}
00068 }
00069 , { { 120, 120, 120, 120, 120}
00070 , { 120, 60, 50, 120, 120}
00071 , { 120, 120, 120, 120, 120}
00072 , { 120, -20, 120, -140, 120}
00073 , { 120, 120, 100, 120, 110}
00074 }
00075 , { { 220, 220, 170, 120, 120}
00076 , { 220, 220, 130, 120, 120}
00077 , { 170, 120, 170, 120, 120}
00078 , { 120, 120, 120, -140, 120}
00079 , { 120, 120, 120, 120, 110}
00080 }
00081 , { { 120, 120, 120, 120, 120}
00082 , { 120, 120, 120, 120, 120}
00083 , { 120, 120, 120, 120, 120}
00084 , { 120, 120, 120, -140, 120}
00085 , { 120, 120, 120, 120, 80}
00086 }
00087 , { { 120, 120, 120, 120, 120}
00088 , { 120, 120, 120, 120, 120}
00089 , { 120, 120, 120, 120, 120}
00090 , { 120, 120, 120, -140, 120}
00091 , { 120, 120, 120, 120, 120}
00092 }
00093 , { { 220, 220, 170, 120, 120}
00094 , { 220, 220, 130, 120, 120}
00095 , { 170, 120, 170, 120, 120}
00096 , { 120, 120, 120, -140, 120}
00097 , { 120, 120, 120, 120, 120}
00098 }
00099 }
00100 , { { { INF, INF, INF, INF, INF}
00101 , { INF, INF, INF, INF, INF}
00102 , { INF, INF, INF, INF, INF}
00103 , { INF, INF, INF, INF, INF}
00104 , { INF, INF, INF, INF, INF}
00105 }
00106 , { { 90, 90, 60, 50, 50}
00107 , { 90, 90, 30, -10, 50}
00108 , { 50, -40, 50, 50, 0}
00109 , { 50, 50, 50, -220, 50}
00110 , { 60, 50, 60, 50, -10}
00111 }
00112 , { { 80, 80, 50, 50, 50}
00113 , { 80, 80, 50, 50, 50}
00114 , { 50, 50, 50, 50, 50}
00115 , { 50, 50, 50, -230, 50}
00116 , { 50, 50, 50, 50, -60}
00117 }
00118 , { { 190, 190, 120, 150, 150}
00119 , { 190, 190, 120, 150, 120}
00120 , { 120, 120, 120, 120, 120}
00121 , { 120, 120, 120, -140, 120}
00122 , { 150, 120, 120, 120, 150}
00123 }
00124 , { { 160, 160, 120, 120, 120}
00125 , { 160, 160, 120, 100, 120}
00126 , { 120, 120, 120, 120, 120}
00127 , { 120, 120, 120, -140, 120}
00128 , { 120, 120, 120, 120, 70}
00129 }
00130 , { { 120, 120, 120, 120, 120}
00131 , { 120, 120, 120, 120, 120}
00132 , { 120, 120, 120, 120, 120}
00133 , { 120, 120, 120, -140, 120}
00134 , { 120, 120, 120, 120, 80}
00135 }
00136 , { { 120, 120, 120, 120, 120}
00137 , { 120, 120, 120, 120, 120}
00138 , { 120, 120, 120, 120, 120}
00139 , { 120, 120, 120, -140, 120}
00140 , { 120, 120, 120, 120, 120}
00141 }
00142 , { { 190, 190, 120, 150, 150}
00143 , { 190, 190, 120, 150, 120}
00144 , { 120, 120, 120, 120, 120}
00145 , { 120, 120, 120, -140, 120}
00146 , { 150, 120, 120, 120, 150}
00147 }
00148 }
00149 , { { { INF, INF, INF, INF, INF}
00150 , { INF, INF, INF, INF, INF}
00151 , { INF, INF, INF, INF, INF}
00152 , { INF, INF, INF, INF, INF}
```

```
00153 , { INF, INF, INF, INF, INF }
00154 }
00155 , { { 120, 120, 120, 120, 120 }
00156 , { 120, 60, 120, -20, 120 }
00157 , { 120, 50, 120, 120, 100 }
00158 , { 120, 120, 120, -140, 120 }
00159 , { 120, 120, 120, 120, 110 }
00160 }
00161 , { { 190, 190, 120, 120, 150 }
00162 , { 190, 190, 120, 120, 120 }
00163 , { 120, 120, 120, 120, 120 }
00164 , { 150, 150, 120, -140, 120 }
00165 , { 150, 120, 120, 120, 150 }
00166 }
00167 , { { 190, 190, 190, 190, 190 }
00168 , { 190, 190, 190, 190, 190 }
00169 , { 190, 190, 190, 190, 190 }
00170 , { 190, 190, 190, -70, 190 }
00171 , { 190, 190, 190, 190, 120 }
00172 }
00173 , { { 190, 190, 190, 190, 190 }
00174 , { 190, 190, 190, 190, 190 }
00175 , { 190, 190, 190, 190, 190 }
00176 , { 190, 190, 190, -70, 190 }
00177 , { 190, 190, 190, 190, 160 }
00178 }
00179 , { { 190, 190, 190, 190, 190 }
00180 , { 190, 190, 190, 190, 190 }
00181 , { 190, 190, 190, 190, 190 }
00182 , { 190, 190, 190, -70, 190 }
00183 , { 190, 190, 190, 190, 120 }
00184 }
00185 , { { 190, 190, 190, 190, 190 }
00186 , { 190, 190, 190, 190, 190 }
00187 , { 190, 190, 190, 190, 190 }
00188 , { 190, 190, 190, -70, 190 }
00189 , { 190, 190, 190, 190, 160 }
00190 }
00191 , { { 190, 190, 190, 190, 190 }
00192 , { 190, 190, 190, 190, 190 }
00193 , { 190, 190, 190, 190, 190 }
00194 , { 190, 190, 190, -70, 190 }
00195 , { 190, 190, 190, 190, 160 }
00196 }
00197 }
00198 , { { INF, INF, INF, INF, INF }
00199 , { INF, INF, INF, INF, INF }
00200 , { INF, INF, INF, INF, INF }
00201 , { INF, INF, INF, INF, INF }
00202 , { INF, INF, INF, INF, INF }
00203 }
00204 , { { 220, 220, 170, 120, 120 }
00205 , { 220, 220, 120, 120, 120 }
00206 , { 170, 130, 170, 120, 120 }
00207 , { 120, 120, 120, -140, 120 }
00208 , { 120, 120, 120, 120, 110 }
00209 }
00210 , { { 160, 160, 120, 120, 120 }
00211 , { 160, 160, 120, 120, 120 }
00212 , { 120, 120, 120, 120, 120 }
00213 , { 120, 100, 120, -140, 120 }
00214 , { 120, 120, 120, 120, 70 }
00215 }
00216 , { { 190, 190, 190, 190, 190 }
00217 , { 190, 190, 190, 190, 190 }
00218 , { 190, 190, 190, 190, 190 }
00219 , { 190, 190, 190, -70, 190 }
00220 , { 190, 190, 190, 190, 160 }
00221 }
00222 , { { 190, 190, 190, 190, 190 }
00223 , { 190, 190, 190, 190, 190 }
00224 , { 190, 190, 190, 190, 190 }
00225 , { 190, 190, 190, -70, 190 }
00226 , { 190, 190, 190, 190, 190 }
00227 }
00228 , { { 190, 190, 190, 190, 190 }
00229 , { 190, 190, 190, 190, 190 }
00230 , { 190, 190, 190, 190, 190 }
00231 , { 190, 190, 190, -70, 190 }
00232 , { 190, 190, 190, 190, 160 }
00233 }
00234 , { { 190, 190, 190, 190, 190 }
00235 , { 190, 190, 190, 190, 190 }
00236 , { 190, 190, 190, 190, 190 }
00237 , { 190, 190, 190, -70, 190 }
00238 , { 190, 190, 190, 190, 190 }
00239 }
```

```
00240 ,{{ 220, 220, 190, 190, 190}
00241 ,{ 220, 220, 190, 190, 190}
00242 ,{ 190, 190, 190, 190, 190}
00243 ,{ 190, 190, 190, -70, 190}
00244 ,{ 190, 190, 190, 190, 190}
00245 }
00246 }
00247 ,{{{ INF, INF, INF, INF, INF}
00248 ,{ INF, INF, INF, INF, INF}
00249 ,{ INF, INF, INF, INF, INF}
00250 ,{ INF, INF, INF, INF, INF}
00251 ,{ INF, INF, INF, INF, INF}
00252 }
00253 ,{{ 120, 120, 120, 120, 120}
00254 ,{ 120, 120, 120, 120, 120}
00255 ,{ 120, 120, 120, 120, 120}
00256 ,{ 120, 120, 120, -140, 120}
00257 ,{ 120, 120, 120, 120, 80}
00258 }
00259 ,{{ 120, 120, 120, 120, 120}
00260 ,{ 120, 120, 120, 120, 120}
00261 ,{ 120, 120, 120, 120, 120}
00262 ,{ 120, 120, 120, -140, 120}
00263 ,{ 120, 120, 120, 120, 80}
00264 }
00265 ,{{ 190, 190, 190, 190, 190}
00266 ,{ 190, 190, 190, 190, 190}
00267 ,{ 190, 190, 190, 190, 190}
00268 ,{ 190, 190, 190, -70, 190}
00269 ,{ 190, 190, 190, 190, 120}
00270 }
00271 ,{{ 190, 190, 190, 190, 190}
00272 ,{ 190, 190, 190, 190, 190}
00273 ,{ 190, 190, 190, 190, 190}
00274 ,{ 190, 190, 190, -70, 190}
00275 ,{ 190, 190, 190, 190, 160}
00276 }
00277 ,{{ 190, 190, 190, 190, 190}
00278 ,{ 190, 190, 190, 190, 190}
00279 ,{ 190, 190, 190, 190, 190}
00280 ,{ 190, 190, 190, -70, 190}
00281 ,{ 190, 190, 190, 190, 120}
00282 }
00283 ,{{ 190, 190, 190, 190, 190}
00284 ,{ 190, 190, 190, 190, 190}
00285 ,{ 190, 190, 190, 190, 190}
00286 ,{ 190, 190, 190, -70, 190}
00287 ,{ 190, 190, 190, 190, 150}
00288 }
00289 ,{{ 190, 190, 190, 190, 190}
00290 ,{ 190, 190, 190, 190, 190}
00291 ,{ 190, 190, 190, 190, 190}
00292 ,{ 190, 190, 190, -70, 190}
00293 ,{ 190, 190, 190, 190, 160}
00294 }
00295 }
00296 ,{{{ INF, INF, INF, INF, INF}
00297 ,{ INF, INF, INF, INF, INF}
00298 ,{ INF, INF, INF, INF, INF}
00299 ,{ INF, INF, INF, INF, INF}
00300 ,{ INF, INF, INF, INF, INF}
00301 }
00302 ,{{ 120, 120, 120, 120, 120}
00303 ,{ 120, 120, 120, 120, 120}
00304 ,{ 120, 120, 120, 120, 120}
00305 ,{ 120, 120, 120, -140, 120}
00306 ,{ 120, 120, 120, 120, 120}
00307 }
00308 ,{{ 120, 120, 120, 120, 120}
00309 ,{ 120, 120, 120, 120, 120}
00310 ,{ 120, 120, 120, 120, 120}
00311 ,{ 120, 120, 120, -140, 120}
00312 ,{ 120, 120, 120, 120, 120}
00313 }
00314 ,{{ 190, 190, 190, 190, 190}
00315 ,{ 190, 190, 190, 190, 190}
00316 ,{ 190, 190, 190, 190, 190}
00317 ,{ 190, 190, 190, -70, 190}
00318 ,{ 190, 190, 190, 190, 160}
00319 }
00320 ,{{ 190, 190, 190, 190, 190}
00321 ,{ 190, 190, 190, 190, 190}
00322 ,{ 190, 190, 190, 190, 190}
00323 ,{ 190, 190, 190, -70, 190}
00324 ,{ 190, 190, 190, 190, 190}
00325 }
00326 ,{{ 190, 190, 190, 190, 190}
```

```

00327 , { 190, 190, 190, 190, 190}
00328 , { 190, 190, 190, 190, 190}
00329 , { 190, 190, 190, -70, 190}
00330 , { 190, 190, 190, 190, 150}
00331 }
00332 , { { 190, 190, 190, 190, 190}
00333 , { 190, 190, 190, 190, 190}
00334 , { 190, 190, 190, 190, 190}
00335 , { 190, 190, 190, -70, 190}
00336 , { 190, 190, 190, 190, 170}
00337 }
00338 , { { 190, 190, 190, 190, 190}
00339 , { 190, 190, 190, 190, 190}
00340 , { 190, 190, 190, 190, 190}
00341 , { 190, 190, 190, -70, 190}
00342 , { 190, 190, 190, 190, 190}
00343 }
00344 }
00345 , { { { INF, INF, INF, INF, INF}
00346 , { INF, INF, INF, INF, INF}
00347 , { INF, INF, INF, INF, INF}
00348 , { INF, INF, INF, INF, INF}
00349 , { INF, INF, INF, INF, INF}
00350 }
00351 , { { 220, 220, 170, 120, 120}
00352 , { 220, 220, 120, 120, 120}
00353 , { 170, 130, 170, 120, 120}
00354 , { 120, 120, 120, -140, 120}
00355 , { 120, 120, 120, 120, 120}
00356 }
00357 , { { 190, 190, 120, 120, 150}
00358 , { 190, 190, 120, 120, 120}
00359 , { 120, 120, 120, 120, 120}
00360 , { 150, 150, 120, -140, 120}
00361 , { 150, 120, 120, 120, 150}
00362 }
00363 , { { 190, 190, 190, 190, 190}
00364 , { 190, 190, 190, 190, 190}
00365 , { 190, 190, 190, 190, 190}
00366 , { 190, 190, 190, -70, 190}
00367 , { 190, 190, 190, 190, 160}
00368 }
00369 , { { 220, 220, 190, 190, 190}
00370 , { 220, 220, 190, 190, 190}
00371 , { 190, 190, 190, 190, 190}
00372 , { 190, 190, 190, -70, 190}
00373 , { 190, 190, 190, 190, 190}
00374 }
00375 , { { 190, 190, 190, 190, 190}
00376 , { 190, 190, 190, 190, 190}
00377 , { 190, 190, 190, 190, 190}
00378 , { 190, 190, 190, -70, 190}
00379 , { 190, 190, 190, 190, 160}
00380 }
00381 , { { 190, 190, 190, 190, 190}
00382 , { 190, 190, 190, 190, 190}
00383 , { 190, 190, 190, 190, 190}
00384 , { 190, 190, 190, -70, 190}
00385 , { 190, 190, 190, 190, 190}
00386 }
00387 , { { 220, 220, 190, 190, 190}
00388 , { 220, 220, 190, 190, 190}
00389 , { 190, 190, 190, 190, 190}
00390 , { 190, 190, 190, -70, 190}
00391 , { 190, 190, 190, 190, 190}
00392 }
00393 } };
```

18.171 intl11dH.h

```

00001 PUBLIC int intl11_dH[NBPAIRS+1][NBPAIRS+1][5][5] =
00002 { { { { INF, INF, INF, INF, INF}
00003 , { INF, INF, INF, INF, INF}
00004 , { INF, INF, INF, INF, INF}
00005 , { INF, INF, INF, INF, INF}
00006 , { INF, INF, INF, INF, INF}
00007 }
00008 , { { INF, INF, INF, INF, INF}
00009 , { INF, INF, INF, INF, INF}
00010 , { INF, INF, INF, INF, INF}
00011 , { INF, INF, INF, INF, INF}
00012 , { INF, INF, INF, INF, INF}
00013 }
00014 , { { INF, INF, INF, INF, INF}
00015 , { INF, INF, INF, INF, INF}
```

```
00016 , { INF, INF, INF, INF, INF }
00017 , { INF, INF, INF, INF, INF }
00018 , { INF, INF, INF, INF, INF }
00019 }
00020 , { { INF, INF, INF, INF, INF }
00021 , { INF, INF, INF, INF, INF }
00022 , { INF, INF, INF, INF, INF }
00023 , { INF, INF, INF, INF, INF }
00024 , { INF, INF, INF, INF, INF }
00025 }
00026 , { { INF, INF, INF, INF, INF }
00027 , { INF, INF, INF, INF, INF }
00028 , { INF, INF, INF, INF, INF }
00029 , { INF, INF, INF, INF, INF }
00030 , { INF, INF, INF, INF, INF }
00031 }
00032 , { { INF, INF, INF, INF, INF }
00033 , { INF, INF, INF, INF, INF }
00034 , { INF, INF, INF, INF, INF }
00035 , { INF, INF, INF, INF, INF }
00036 , { INF, INF, INF, INF, INF }
00037 }
00038 , { { INF, INF, INF, INF, INF }
00039 , { INF, INF, INF, INF, INF }
00040 , { INF, INF, INF, INF, INF }
00041 , { INF, INF, INF, INF, INF }
00042 , { INF, INF, INF, INF, INF }
00043 }
00044 , { { INF, INF, INF, INF, INF }
00045 , { INF, INF, INF, INF, INF }
00046 , { INF, INF, INF, INF, INF }
00047 , { INF, INF, INF, INF, INF }
00048 , { INF, INF, INF, INF, INF }
00049 }
00050 }
00051 , { { { INF, INF, INF, INF, INF }
00052 , { INF, INF, INF, INF, INF }
00053 , { INF, INF, INF, INF, INF }
00054 , { INF, INF, INF, INF, INF }
00055 , { INF, INF, INF, INF, INF }
00056 }
00057 , { { -1050, -1050, -1050, -1050, -1050 }
00058 , { -1050, -1050, -1050, -1050, -1050 }
00059 , { -1050, -1050, -1050, -1050, -1050 }
00060 , { -1050, -1050, -1050, -1840, -1050 }
00061 , { -1050, -1050, -1050, -1050, -1050 }
00062 }
00063 , { { -1050, -1050, -1050, -1050, -1050 }
00064 , { -1050, -1050, -1050, -1050, -1050 }
00065 , { -1050, -1050, -1050, -1050, -1050 }
00066 , { -1050, -1050, -1050, -1840, -1050 }
00067 , { -1050, -1050, -1050, -1050, -1390 }
00068 }
00069 , { { -550, -550, -550, -550, -550 }
00070 , { -550, -550, -550, -550, -550 }
00071 , { -550, -550, -550, -550, -550 }
00072 , { -550, -550, -550, -1340, -550 }
00073 , { -550, -550, -550, -550, -890 }
00074 }
00075 , { { -550, -550, -550, -550, -550 }
00076 , { -550, -550, -550, -550, -550 }
00077 , { -550, -550, -550, -550, -550 }
00078 , { -550, -550, -550, -1340, -550 }
00079 , { -550, -550, -550, -550, -550 }
00080 }
00081 , { { -550, -550, -550, -550, -550 }
00082 , { -550, -550, -550, -550, -550 }
00083 , { -550, -550, -550, -550, -550 }
00084 , { -550, -550, -550, -1340, -550 }
00085 , { -550, -550, -550, -550, -890 }
00086 }
00087 , { { -550, -550, -550, -550, -550 }
00088 , { -550, -550, -550, -550, -550 }
00089 , { -550, -550, -550, -550, -550 }
00090 , { -550, -550, -550, -1340, -550 }
00091 , { -550, -550, -550, -550, -550 }
00092 }
00093 , { { -550, -550, -550, -550, -550 }
00094 , { -550, -550, -550, -550, -550 }
00095 , { -550, -550, -550, -550, -550 }
00096 , { -550, -550, -550, -1340, -550 }
00097 , { -550, -550, -550, -550, -550 }
00098 }
00099 }
00100 , { { { INF, INF, INF, INF, INF }
00101 , { INF, INF, INF, INF, INF }
00102 , { INF, INF, INF, INF, INF }
```

```
00103 ,{ INF, INF, INF, INF, INF}
00104 ,{ INF, INF, INF, INF, INF}
00105 }
00106 ,{{ -1050, -1050, -1050, -1050, -1050}
00107 ,{ -1050, -1050, -1050, -1050, -1050}
00108 ,{ -1050, -1050, -1050, -1050, -1050}
00109 ,{ -1050, -1050, -1050, -1840, -1050}
00110 ,{ -1050, -1050, -1050, -1050, -1390}
00111 }
00112 ,{{ -1050, -1050, -1050, -1050, -1050}
00113 ,{ -1050, -1050, -1050, -1050, -1050}
00114 ,{ -1050, -1050, -1050, -1050, -1050}
00115 ,{ -1050, -1050, -1050, -1840, -1050}
00116 ,{ -1050, -1050, -1050, -1050, -1730}
00117 }
00118 ,{{ -550, -550, -550, -550, -550}
00119 ,{ -550, -550, -550, -550, -550}
00120 ,{ -550, -550, -550, -550, -550}
00121 ,{ -550, -550, -550, -1340, -550}
00122 ,{ -550, -550, -550, -550, -1230}
00123 }
00124 ,{{ -550, -550, -550, -550, -550}
00125 ,{ -550, -550, -550, -550, -550}
00126 ,{ -550, -550, -550, -550, -550}
00127 ,{ -550, -550, -550, -1340, -550}
00128 ,{ -550, -550, -550, -550, -890}
00129 }
00130 ,{{ -550, -550, -550, -550, -550}
00131 ,{ -550, -550, -550, -550, -550}
00132 ,{ -550, -550, -550, -550, -550}
00133 ,{ -550, -550, -550, -1340, -550}
00134 ,{ -550, -550, -550, -550, -1230}
00135 }
00136 ,{{ -550, -550, -550, -550, -550}
00137 ,{ -550, -550, -550, -550, -550}
00138 ,{ -550, -550, -550, -550, -550}
00139 ,{ -550, -550, -550, -1340, -550}
00140 ,{ -550, -550, -550, -550, -890}
00141 }
00142 ,{{ -550, -550, -550, -550, -550}
00143 ,{ -550, -550, -550, -550, -550}
00144 ,{ -550, -550, -550, -550, -550}
00145 ,{ -550, -550, -550, -1340, -550}
00146 ,{ -550, -550, -550, -550, -890}
00147 }
00148 }
00149 ,{{{ INF, INF, INF, INF, INF}
00150 ,{ INF, INF, INF, INF, INF}
00151 ,{ INF, INF, INF, INF, INF}
00152 ,{ INF, INF, INF, INF, INF}
00153 ,{ INF, INF, INF, INF, INF}
00154 }
00155 ,{{ -550, -550, -550, -550, -550}
00156 ,{ -550, -550, -550, -550, -550}
00157 ,{ -550, -550, -550, -550, -550}
00158 ,{ -550, -550, -550, -1340, -550}
00159 ,{ -550, -550, -550, -550, -890}
00160 }
00161 ,{{ -550, -550, -550, -550, -550}
00162 ,{ -550, -550, -550, -550, -550}
00163 ,{ -550, -550, -550, -550, -550}
00164 ,{ -550, -550, -550, -1340, -550}
00165 ,{ -550, -550, -550, -550, -1230}
00166 }
00167 ,{{ -50, -50, -50, -50, -50}
00168 ,{ -50, -50, -50, -50, -50}
00169 ,{ -50, -50, -50, -50, -50}
00170 ,{ -50, -50, -50, -830, -50}
00171 ,{ -50, -50, -50, -50, -730}
00172 }
00173 ,{{ -50, -50, -50, -50, -50}
00174 ,{ -50, -50, -50, -50, -50}
00175 ,{ -50, -50, -50, -50, -50}
00176 ,{ -50, -50, -50, -830, -50}
00177 ,{ -50, -50, -50, -50, -390}
00178 }
00179 ,{{ -50, -50, -50, -50, -50}
00180 ,{ -50, -50, -50, -50, -50}
00181 ,{ -50, -50, -50, -50, -50}
00182 ,{ -50, -50, -50, -830, -50}
00183 ,{ -50, -50, -50, -50, -730}
00184 }
00185 ,{{ -50, -50, -50, -50, -50}
00186 ,{ -50, -50, -50, -50, -50}
00187 ,{ -50, -50, -50, -50, -50}
00188 ,{ -50, -50, -50, -830, -50}
00189 ,{ -50, -50, -50, -50, -390}
```



```
00190     }
00191     ,{{ -50, -50, -50, -50, -50}
00192     ,{{ -50, -50, -50, -50, -50}
00193     ,{{ -50, -50, -50, -50, -50}
00194     ,{{ -50, -50, -50, -830, -50}
00195     ,{{ -50, -50, -50, -50, -390}
00196     }
00197     }
00198     ,{{{ INF, INF, INF, INF, INF}
00199     ,{{ INF, INF, INF, INF, INF}
00200     ,{{ INF, INF, INF, INF, INF}
00201     ,{{ INF, INF, INF, INF, INF}
00202     ,{{ INF, INF, INF, INF, INF}
00203     }
00204     ,{{{ -550, -550, -550, -550, -550}
00205     ,{{ -550, -550, -550, -550, -550}
00206     ,{{ -550, -550, -550, -550, -550}
00207     ,{{ -550, -550, -550, -1340, -550}
00208     ,{{ -550, -550, -550, -550, -550}
00209     }
00210     ,{{{ -550, -550, -550, -550, -550}
00211     ,{{ -550, -550, -550, -550, -550}
00212     ,{{ -550, -550, -550, -550, -550}
00213     ,{{ -550, -550, -550, -1340, -550}
00214     ,{{ -550, -550, -550, -550, -890}
00215     }
00216     ,{{{ -50, -50, -50, -50, -50}
00217     ,{{ -50, -50, -50, -50, -50}
00218     ,{{ -50, -50, -50, -50, -50}
00219     ,{{ -50, -50, -50, -830, -50}
00220     ,{{ -50, -50, -50, -50, -390}
00221     }
00222     ,{{{ -50, -50, -50, -50, -50}
00223     ,{{ -50, -50, -50, -50, -50}
00224     ,{{ -50, -50, -50, -50, -50}
00225     ,{{ -50, -50, -50, -830, -50}
00226     ,{{ -50, -50, -50, -50, -50}
00227     }
00228     ,{{{ -50, -50, -50, -50, -50}
00229     ,{{ -50, -50, -50, -50, -50}
00230     ,{{ -50, -50, -50, -50, -50}
00231     ,{{ -50, -50, -50, -830, -50}
00232     ,{{ -50, -50, -50, -50, -390}
00233     }
00234     ,{{{ -50, -50, -50, -50, -50}
00235     ,{{ -50, -50, -50, -50, -50}
00236     ,{{ -50, -50, -50, -50, -50}
00237     ,{{ -50, -50, -50, -830, -50}
00238     ,{{ -50, -50, -50, -50, -50}
00239     }
00240     ,{{{ -50, -50, -50, -50, -50}
00241     ,{{ -50, -50, -50, -50, -50}
00242     ,{{ -50, -50, -50, -50, -50}
00243     ,{{ -50, -50, -50, -830, -50}
00244     ,{{ -50, -50, -50, -50, -50}
00245     }
00246     }
00247     ,{{{ INF, INF, INF, INF, INF}
00248     ,{{ INF, INF, INF, INF, INF}
00249     ,{{ INF, INF, INF, INF, INF}
00250     ,{{ INF, INF, INF, INF, INF}
00251     ,{{ INF, INF, INF, INF, INF}
00252     }
00253     ,{{{ -550, -550, -550, -550, -550}
00254     ,{{ -550, -550, -550, -550, -550}
00255     ,{{ -550, -550, -550, -550, -550}
00256     ,{{ -550, -550, -550, -1340, -550}
00257     ,{{ -550, -550, -550, -550, -890}
00258     }
00259     ,{{{ -550, -550, -550, -550, -550}
00260     ,{{ -550, -550, -550, -550, -550}
00261     ,{{ -550, -550, -550, -550, -550}
00262     ,{{ -550, -550, -550, -1340, -550}
00263     ,{{ -550, -550, -550, -550, -1230}
00264     }
00265     ,{{{ -50, -50, -50, -50, -50}
00266     ,{{ -50, -50, -50, -50, -50}
00267     ,{{ -50, -50, -50, -50, -50}
00268     ,{{ -50, -50, -50, -830, -50}
00269     ,{{ -50, -50, -50, -50, -730}
00270     }
00271     ,{{{ -50, -50, -50, -50, -50}
00272     ,{{ -50, -50, -50, -50, -50}
00273     ,{{ -50, -50, -50, -50, -50}
00274     ,{{ -50, -50, -50, -830, -50}
00275     ,{{ -50, -50, -50, -50, -390}
00276     }
```

```
00277 ,{{ -50, -50, -50, -50, -50}
00278 ,{ -50, -50, -50, -50, -50}
00279 ,{ -50, -50, -50, -50, -50}
00280 ,{ -50, -50, -50, -830, -50}
00281 ,{ -50, -50, -50, -50, -730}
00282 }
00283 ,{{ -50, -50, -50, -50, -50}
00284 ,{ -50, -50, -50, -50, -50}
00285 ,{ -50, -50, -50, -50, -50}
00286 ,{ -50, -50, -50, -830, -50}
00287 ,{ -50, -50, -50, -50, -390}
00288 }
00289 ,{{ -50, -50, -50, -50, -50}
00290 ,{ -50, -50, -50, -50, -50}
00291 ,{ -50, -50, -50, -50, -50}
00292 ,{ -50, -50, -50, -830, -50}
00293 ,{ -50, -50, -50, -50, -390}
00294 }
00295 }
00296 ,{{{ INF, INF, INF, INF, INF}
00297 ,{ INF, INF, INF, INF, INF}
00298 ,{ INF, INF, INF, INF, INF}
00299 ,{ INF, INF, INF, INF, INF}
00300 ,{ INF, INF, INF, INF, INF}
00301 }
00302 ,{{ -550, -550, -550, -550, -550}
00303 ,{ -550, -550, -550, -550, -550}
00304 ,{ -550, -550, -550, -550, -550}
00305 ,{ -550, -550, -550, -1340, -550}
00306 ,{ -550, -550, -550, -550, -550}
00307 }
00308 ,{{ -550, -550, -550, -550, -550}
00309 ,{ -550, -550, -550, -550, -550}
00310 ,{ -550, -550, -550, -550, -550}
00311 ,{ -550, -550, -550, -1340, -550}
00312 ,{ -550, -550, -550, -550, -890}
00313 }
00314 ,{{ -50, -50, -50, -50, -50}
00315 ,{ -50, -50, -50, -50, -50}
00316 ,{ -50, -50, -50, -50, -50}
00317 ,{ -50, -50, -50, -830, -50}
00318 ,{ -50, -50, -50, -50, -390}
00319 }
00320 ,{{ -50, -50, -50, -50, -50}
00321 ,{ -50, -50, -50, -50, -50}
00322 ,{ -50, -50, -50, -50, -50}
00323 ,{ -50, -50, -50, -830, -50}
00324 ,{ -50, -50, -50, -50, -50}
00325 }
00326 ,{{ -50, -50, -50, -50, -50}
00327 ,{ -50, -50, -50, -50, -50}
00328 ,{ -50, -50, -50, -50, -50}
00329 ,{ -50, -50, -50, -830, -50}
00330 ,{ -50, -50, -50, -50, -390}
00331 }
00332 ,{{ -50, -50, -50, -50, -50}
00333 ,{ -50, -50, -50, -50, -50}
00334 ,{ -50, -50, -50, -50, -50}
00335 ,{ -50, -50, -50, -830, -50}
00336 ,{ -50, -50, -50, -50, -50}
00337 }
00338 ,{{ -50, -50, -50, -50, -50}
00339 ,{ -50, -50, -50, -50, -50}
00340 ,{ -50, -50, -50, -50, -50}
00341 ,{ -50, -50, -50, -830, -50}
00342 ,{ -50, -50, -50, -50, -50}
00343 }
00344 }
00345 ,{{{ INF, INF, INF, INF, INF}
00346 ,{ INF, INF, INF, INF, INF}
00347 ,{ INF, INF, INF, INF, INF}
00348 ,{ INF, INF, INF, INF, INF}
00349 ,{ INF, INF, INF, INF, INF}
00350 }
00351 ,{{ -550, -550, -550, -550, -550}
00352 ,{ -550, -550, -550, -550, -550}
00353 ,{ -550, -550, -550, -550, -550}
00354 ,{ -550, -550, -550, -1340, -550}
00355 ,{ -550, -550, -550, -550, -550}
00356 }
00357 ,{{ -550, -550, -550, -550, -550}
00358 ,{ -550, -550, -550, -550, -550}
00359 ,{ -550, -550, -550, -550, -550}
00360 ,{ -550, -550, -550, -1340, -550}
00361 ,{ -550, -550, -550, -550, -890}
00362 }
00363 ,{{ -50, -50, -50, -50, -50}
```

```

00364 , { -50, -50, -50, -50, -50}
00365 , { -50, -50, -50, -50, -50}
00366 , { -50, -50, -50, -830, -50}
00367 , { -50, -50, -50, -50, -390}
00368 }
00369 , { { -50, -50, -50, -50, -50}
00370 , { -50, -50, -50, -50, -50}
00371 , { -50, -50, -50, -50, -50}
00372 , { -50, -50, -50, -830, -50}
00373 , { -50, -50, -50, -50, -50}
00374 }
00375 , { { -50, -50, -50, -50, -50}
00376 , { -50, -50, -50, -50, -50}
00377 , { -50, -50, -50, -50, -50}
00378 , { -50, -50, -50, -830, -50}
00379 , { -50, -50, -50, -50, -390}
00380 }
00381 , { { -50, -50, -50, -50, -50}
00382 , { -50, -50, -50, -50, -50}
00383 , { -50, -50, -50, -50, -50}
00384 , { -50, -50, -50, -830, -50}
00385 , { -50, -50, -50, -50, -50}
00386 }
00387 , { { -50, -50, -50, -50, -50}
00388 , { -50, -50, -50, -50, -50}
00389 , { -50, -50, -50, -50, -50}
00390 , { -50, -50, -50, -830, -50}
00391 , { -50, -50, -50, -50, -50}
00392 }
00393 };;

```

18.172 intl21.h

```

00001 PUBLIC int int21_37[NBPAIRS+1][NBPAIRS+1][5][5][5] =
00002 {{{{{ INF, INF, INF, INF, INF}
00003 , { INF, INF, INF, INF, INF}
00004 , { INF, INF, INF, INF, INF}
00005 , { INF, INF, INF, INF, INF}
00006 , { INF, INF, INF, INF, INF}
00007 }
00008 , {{{ INF, INF, INF, INF, INF}
00009 , { INF, INF, INF, INF, INF}
00010 , { INF, INF, INF, INF, INF}
00011 , { INF, INF, INF, INF, INF}
00012 , { INF, INF, INF, INF, INF}
00013 }
00014 , {{{ INF, INF, INF, INF, INF}
00015 , { INF, INF, INF, INF, INF}
00016 , { INF, INF, INF, INF, INF}
00017 , { INF, INF, INF, INF, INF}
00018 , { INF, INF, INF, INF, INF}
00019 }
00020 , {{{ INF, INF, INF, INF, INF}
00021 , { INF, INF, INF, INF, INF}
00022 , { INF, INF, INF, INF, INF}
00023 , { INF, INF, INF, INF, INF}
00024 , { INF, INF, INF, INF, INF}
00025 }
00026 , {{{ INF, INF, INF, INF, INF}
00027 , { INF, INF, INF, INF, INF}
00028 , { INF, INF, INF, INF, INF}
00029 , { INF, INF, INF, INF, INF}
00030 , { INF, INF, INF, INF, INF}
00031 }
00032 }
00033 , {{{ INF, INF, INF, INF, INF}
00034 , { INF, INF, INF, INF, INF}
00035 , { INF, INF, INF, INF, INF}
00036 , { INF, INF, INF, INF, INF}
00037 , { INF, INF, INF, INF, INF}
00038 }
00039 , {{{ INF, INF, INF, INF, INF}
00040 , { INF, INF, INF, INF, INF}
00041 , { INF, INF, INF, INF, INF}
00042 , { INF, INF, INF, INF, INF}
00043 , { INF, INF, INF, INF, INF}
00044 }
00045 , {{{ INF, INF, INF, INF, INF}
00046 , { INF, INF, INF, INF, INF}
00047 , { INF, INF, INF, INF, INF}
00048 , { INF, INF, INF, INF, INF}
00049 , { INF, INF, INF, INF, INF}
00050 }
00051 , {{{ INF, INF, INF, INF, INF}
00052 , { INF, INF, INF, INF, INF}

```

```
00053 , { INF, INF, INF, INF, INF }
00054 , { INF, INF, INF, INF, INF }
00055 , { INF, INF, INF, INF, INF }
00056 }
00057 , { { INF, INF, INF, INF, INF }
00058 , { INF, INF, INF, INF, INF }
00059 , { INF, INF, INF, INF, INF }
00060 , { INF, INF, INF, INF, INF }
00061 , { INF, INF, INF, INF, INF }
00062 }
00063 }
00064 , { { { INF, INF, INF, INF, INF }
00065 , { INF, INF, INF, INF, INF }
00066 , { INF, INF, INF, INF, INF }
00067 , { INF, INF, INF, INF, INF }
00068 , { INF, INF, INF, INF, INF }
00069 }
00070 , { { INF, INF, INF, INF, INF }
00071 , { INF, INF, INF, INF, INF }
00072 , { INF, INF, INF, INF, INF }
00073 , { INF, INF, INF, INF, INF }
00074 , { INF, INF, INF, INF, INF }
00075 }
00076 , { { INF, INF, INF, INF, INF }
00077 , { INF, INF, INF, INF, INF }
00078 , { INF, INF, INF, INF, INF }
00079 , { INF, INF, INF, INF, INF }
00080 , { INF, INF, INF, INF, INF }
00081 }
00082 , { { INF, INF, INF, INF, INF }
00083 , { INF, INF, INF, INF, INF }
00084 , { INF, INF, INF, INF, INF }
00085 , { INF, INF, INF, INF, INF }
00086 , { INF, INF, INF, INF, INF }
00087 }
00088 , { { INF, INF, INF, INF, INF }
00089 , { INF, INF, INF, INF, INF }
00090 , { INF, INF, INF, INF, INF }
00091 , { INF, INF, INF, INF, INF }
00092 , { INF, INF, INF, INF, INF }
00093 }
00094 }
00095 , { { { INF, INF, INF, INF, INF }
00096 , { INF, INF, INF, INF, INF }
00097 , { INF, INF, INF, INF, INF }
00098 , { INF, INF, INF, INF, INF }
00099 , { INF, INF, INF, INF, INF }
00100 }
00101 , { { INF, INF, INF, INF, INF }
00102 , { INF, INF, INF, INF, INF }
00103 , { INF, INF, INF, INF, INF }
00104 , { INF, INF, INF, INF, INF }
00105 , { INF, INF, INF, INF, INF }
00106 }
00107 , { { INF, INF, INF, INF, INF }
00108 , { INF, INF, INF, INF, INF }
00109 , { INF, INF, INF, INF, INF }
00110 , { INF, INF, INF, INF, INF }
00111 , { INF, INF, INF, INF, INF }
00112 }
00113 , { { INF, INF, INF, INF, INF }
00114 , { INF, INF, INF, INF, INF }
00115 , { INF, INF, INF, INF, INF }
00116 , { INF, INF, INF, INF, INF }
00117 , { INF, INF, INF, INF, INF }
00118 }
00119 , { { INF, INF, INF, INF, INF }
00120 , { INF, INF, INF, INF, INF }
00121 , { INF, INF, INF, INF, INF }
00122 , { INF, INF, INF, INF, INF }
00123 , { INF, INF, INF, INF, INF }
00124 }
00125 }
00126 , { { { INF, INF, INF, INF, INF }
00127 , { INF, INF, INF, INF, INF }
00128 , { INF, INF, INF, INF, INF }
00129 , { INF, INF, INF, INF, INF }
00130 , { INF, INF, INF, INF, INF }
00131 }
00132 , { { INF, INF, INF, INF, INF }
00133 , { INF, INF, INF, INF, INF }
00134 , { INF, INF, INF, INF, INF }
00135 , { INF, INF, INF, INF, INF }
00136 , { INF, INF, INF, INF, INF }
00137 }
00138 , { { INF, INF, INF, INF, INF }
00139 , { INF, INF, INF, INF, INF }
```

```
00140 , { INF, INF, INF, INF, INF }
00141 , { INF, INF, INF, INF, INF }
00142 , { INF, INF, INF, INF, INF }
00143 }
00144 , { { INF, INF, INF, INF, INF }
00145 , { INF, INF, INF, INF, INF }
00146 , { INF, INF, INF, INF, INF }
00147 , { INF, INF, INF, INF, INF }
00148 , { INF, INF, INF, INF, INF }
00149 }
00150 , { { INF, INF, INF, INF, INF }
00151 , { INF, INF, INF, INF, INF }
00152 , { INF, INF, INF, INF, INF }
00153 , { INF, INF, INF, INF, INF }
00154 , { INF, INF, INF, INF, INF }
00155 }
00156 }
00157 , { { { INF, INF, INF, INF, INF }
00158 , { INF, INF, INF, INF, INF }
00159 , { INF, INF, INF, INF, INF }
00160 , { INF, INF, INF, INF, INF }
00161 , { INF, INF, INF, INF, INF }
00162 }
00163 , { { INF, INF, INF, INF, INF }
00164 , { INF, INF, INF, INF, INF }
00165 , { INF, INF, INF, INF, INF }
00166 , { INF, INF, INF, INF, INF }
00167 , { INF, INF, INF, INF, INF }
00168 }
00169 , { { INF, INF, INF, INF, INF }
00170 , { INF, INF, INF, INF, INF }
00171 , { INF, INF, INF, INF, INF }
00172 , { INF, INF, INF, INF, INF }
00173 , { INF, INF, INF, INF, INF }
00174 }
00175 , { { INF, INF, INF, INF, INF }
00176 , { INF, INF, INF, INF, INF }
00177 , { INF, INF, INF, INF, INF }
00178 , { INF, INF, INF, INF, INF }
00179 , { INF, INF, INF, INF, INF }
00180 }
00181 , { { INF, INF, INF, INF, INF }
00182 , { INF, INF, INF, INF, INF }
00183 , { INF, INF, INF, INF, INF }
00184 , { INF, INF, INF, INF, INF }
00185 , { INF, INF, INF, INF, INF }
00186 }
00187 }
00188 , { { { INF, INF, INF, INF, INF }
00189 , { INF, INF, INF, INF, INF }
00190 , { INF, INF, INF, INF, INF }
00191 , { INF, INF, INF, INF, INF }
00192 , { INF, INF, INF, INF, INF }
00193 }
00194 , { { INF, INF, INF, INF, INF }
00195 , { INF, INF, INF, INF, INF }
00196 , { INF, INF, INF, INF, INF }
00197 , { INF, INF, INF, INF, INF }
00198 , { INF, INF, INF, INF, INF }
00199 }
00200 , { { INF, INF, INF, INF, INF }
00201 , { INF, INF, INF, INF, INF }
00202 , { INF, INF, INF, INF, INF }
00203 , { INF, INF, INF, INF, INF }
00204 , { INF, INF, INF, INF, INF }
00205 }
00206 , { { INF, INF, INF, INF, INF }
00207 , { INF, INF, INF, INF, INF }
00208 , { INF, INF, INF, INF, INF }
00209 , { INF, INF, INF, INF, INF }
00210 , { INF, INF, INF, INF, INF }
00211 }
00212 , { { INF, INF, INF, INF, INF }
00213 , { INF, INF, INF, INF, INF }
00214 , { INF, INF, INF, INF, INF }
00215 , { INF, INF, INF, INF, INF }
00216 , { INF, INF, INF, INF, INF }
00217 }
00218 }
00219 , { { { INF, INF, INF, INF, INF }
00220 , { INF, INF, INF, INF, INF }
00221 , { INF, INF, INF, INF, INF }
00222 , { INF, INF, INF, INF, INF }
00223 , { INF, INF, INF, INF, INF }
00224 }
00225 , { { INF, INF, INF, INF, INF }
00226 , { INF, INF, INF, INF, INF }
```

```

00227 , { INF, INF, INF, INF, INF }
00228 , { INF, INF, INF, INF, INF }
00229 , { INF, INF, INF, INF, INF }
00230 }
00231 , { { INF, INF, INF, INF, INF }
00232 , { INF, INF, INF, INF, INF }
00233 , { INF, INF, INF, INF, INF }
00234 , { INF, INF, INF, INF, INF }
00235 , { INF, INF, INF, INF, INF }
00236 }
00237 , { { INF, INF, INF, INF, INF }
00238 , { INF, INF, INF, INF, INF }
00239 , { INF, INF, INF, INF, INF }
00240 , { INF, INF, INF, INF, INF }
00241 , { INF, INF, INF, INF, INF }
00242 }
00243 , { { INF, INF, INF, INF, INF }
00244 , { INF, INF, INF, INF, INF }
00245 , { INF, INF, INF, INF, INF }
00246 , { INF, INF, INF, INF, INF }
00247 , { INF, INF, INF, INF, INF }
00248 }
00249 }
00250 }
00251 , { { { INF, INF, INF, INF, INF }
00252 , { INF, INF, INF, INF, INF }
00253 , { INF, INF, INF, INF, INF }
00254 , { INF, INF, INF, INF, INF }
00255 , { INF, INF, INF, INF, INF }
00256 }
00257 , { { INF, INF, INF, INF, INF }
00258 , { INF, INF, INF, INF, INF }
00259 , { INF, INF, INF, INF, INF }
00260 , { INF, INF, INF, INF, INF }
00261 , { INF, INF, INF, INF, INF }
00262 }
00263 , { { INF, INF, INF, INF, INF }
00264 , { INF, INF, INF, INF, INF }
00265 , { INF, INF, INF, INF, INF }
00266 , { INF, INF, INF, INF, INF }
00267 , { INF, INF, INF, INF, INF }
00268 }
00269 , { { INF, INF, INF, INF, INF }
00270 , { INF, INF, INF, INF, INF }
00271 , { INF, INF, INF, INF, INF }
00272 , { INF, INF, INF, INF, INF }
00273 , { INF, INF, INF, INF, INF }
00274 }
00275 , { { INF, INF, INF, INF, INF }
00276 , { INF, INF, INF, INF, INF }
00277 , { INF, INF, INF, INF, INF }
00278 , { INF, INF, INF, INF, INF }
00279 , { INF, INF, INF, INF, INF }
00280 }
00281 }
00282 , { { { 230, 230, 230, 230, 230 }
00283 , { 230, 230, 230, 230, 230 }
00284 , { 230, 230, 230, 230, 230 }
00285 , { 230, 230, 230, 230, 230 }
00286 , { 230, 230, 230, 230, 230 }
00287 }
00288 , { { 230, 230, 230, 110, 230 }
00289 , { 230, 230, 230, 110, 230 }
00290 , { 230, 230, 230, 110, 230 }
00291 , { 110, 110, 110, 110, 110 }
00292 , { 230, 230, 230, 110, 230 }
00293 }
00294 , { { 230, 230, 230, 230, 230 }
00295 , { 230, 230, 230, 230, 230 }
00296 , { 230, 230, 230, 230, 230 }
00297 , { 230, 230, 230, 230, 230 }
00298 , { 230, 230, 230, 230, 230 }
00299 }
00300 , { { 230, 110, 230, 110, 230 }
00301 , { 110, 110, 110, 110, 110 }
00302 , { 230, 110, 230, 110, 230 }
00303 , { 110, 110, 110, 110, 110 }
00304 , { 230, 110, 230, 110, 230 }
00305 }
00306 , { { 230, 230, 230, 230, 150 }
00307 , { 230, 230, 230, 230, 150 }
00308 , { 230, 230, 230, 230, 150 }
00309 , { 230, 230, 230, 230, 150 }
00310 , { 150, 150, 150, 150, 150 }
00311 }
00312 }
00313 , { { { 250, 250, 250, 230, 230 }

```

```
00314 , { 250, 250, 230, 230, 230 }
00315 , { 250, 230, 250, 230, 230 }
00316 , { 230, 230, 230, 230, 230 }
00317 , { 250, 250, 230, 230, 230 }
00318 }
00319 , { { 250, 250, 230, 110, 230 }
00320 , { 250, 250, 230, 110, 230 }
00321 , { 230, 230, 170, 110, 230 }
00322 , { 110, 80, 110, 110, 110 }
00323 , { 230, 230, 230, 110, 230 }
00324 }
00325 , { { 250, 250, 250, 230, 230 }
00326 , { 230, 230, 230, 230, 230 }
00327 , { 250, 230, 250, 230, 230 }
00328 , { 230, 230, 230, 230, 230 }
00329 , { 250, 250, 230, 230, 230 }
00330 }
00331 , { { 230, 170, 230, 110, 230 }
00332 , { 230, 170, 230, 80, 230 }
00333 , { 230, 110, 230, 110, 230 }
00334 , { 120, 120, 110, 110, 110 }
00335 , { 230, 110, 230, 110, 230 }
00336 }
00337 , { { 230, 230, 230, 230, 150 }
00338 , { 230, 230, 230, 230, 150 }
00339 , { 230, 230, 220, 230, 150 }
00340 , { 230, 230, 230, 230, 150 }
00341 , { 170, 150, 170, 150, 140 }
00342 }
00343 }
00344 , { { { 300, 300, 300, 300, 300 }
00345 , { 300, 300, 300, 300, 300 }
00346 , { 300, 300, 300, 300, 300 }
00347 , { 300, 300, 300, 300, 300 }
00348 , { 300, 300, 300, 300, 300 }
00349 }
00350 , { { 300, 300, 300, 190, 300 }
00351 , { 300, 300, 300, 190, 300 }
00352 , { 300, 300, 300, 190, 300 }
00353 , { 190, 190, 190, 190, 190 }
00354 , { 300, 300, 300, 190, 300 }
00355 }
00356 , { { 300, 300, 300, 300, 300 }
00357 , { 300, 300, 300, 300, 300 }
00358 , { 300, 300, 300, 300, 300 }
00359 , { 300, 300, 300, 300, 300 }
00360 , { 300, 300, 300, 300, 300 }
00361 }
00362 , { { 300, 190, 300, 190, 300 }
00363 , { 300, 190, 300, 190, 300 }
00364 , { 300, 190, 300, 190, 300 }
00365 , { 190, 190, 190, 190, 190 }
00366 , { 300, 190, 300, 190, 300 }
00367 }
00368 , { { 300, 300, 300, 300, 220 }
00369 , { 300, 300, 300, 300, 220 }
00370 , { 300, 300, 300, 300, 220 }
00371 , { 300, 300, 300, 300, 220 }
00372 , { 220, 220, 220, 220, 220 }
00373 }
00374 }
00375 , { { { 300, 300, 300, 300, 300 }
00376 , { 300, 300, 300, 300, 300 }
00377 , { 300, 300, 300, 300, 300 }
00378 , { 300, 300, 300, 300, 300 }
00379 , { 300, 300, 300, 300, 300 }
00380 }
00381 , { { 300, 300, 300, 190, 300 }
00382 , { 300, 300, 300, 190, 300 }
00383 , { 300, 300, 300, 190, 300 }
00384 , { 190, 190, 190, 190, 190 }
00385 , { 300, 300, 300, 190, 300 }
00386 }
00387 , { { 300, 300, 300, 300, 300 }
00388 , { 300, 300, 300, 300, 300 }
00389 , { 300, 300, 300, 300, 300 }
00390 , { 300, 300, 300, 300, 300 }
00391 , { 300, 300, 300, 300, 300 }
00392 }
00393 , { { 300, 190, 300, 190, 300 }
00394 , { 190, 190, 190, 190, 190 }
00395 , { 300, 190, 300, 190, 300 }
00396 , { 190, 190, 190, 190, 190 }
00397 , { 300, 190, 300, 190, 300 }
00398 }
00399 , { { 300, 300, 300, 300, 220 }
00400 , { 300, 300, 300, 300, 220 }
```

```
00401 , { 300, 300, 300, 300, 220}
00402 , { 300, 300, 300, 300, 220}
00403 , { 220, 220, 220, 220, 220}
00404 }
00405 }
00406 , { { 300, 300, 300, 300, 300}
00407 , { 300, 300, 300, 300, 300}
00408 , { 300, 300, 300, 300, 300}
00409 , { 300, 300, 300, 300, 300}
00410 , { 300, 300, 300, 300, 300}
00411 }
00412 , { { 300, 300, 300, 190, 300}
00413 , { 300, 300, 300, 190, 300}
00414 , { 300, 300, 300, 190, 300}
00415 , { 190, 190, 190, 190, 190}
00416 , { 300, 300, 300, 190, 300}
00417 }
00418 , { { 300, 300, 300, 300, 300}
00419 , { 300, 300, 300, 300, 300}
00420 , { 300, 300, 300, 300, 300}
00421 , { 300, 300, 300, 300, 300}
00422 , { 300, 300, 300, 300, 300}
00423 }
00424 , { { 300, 190, 300, 190, 300}
00425 , { 300, 190, 300, 190, 300}
00426 , { 300, 190, 300, 190, 300}
00427 , { 190, 190, 190, 190, 190}
00428 , { 300, 190, 300, 190, 300}
00429 }
00430 , { { 300, 300, 300, 300, 220}
00431 , { 300, 300, 300, 300, 220}
00432 , { 300, 300, 300, 300, 220}
00433 , { 300, 300, 300, 300, 220}
00434 , { 220, 220, 220, 220, 220}
00435 }
00436 }
00437 , { { { 300, 300, 300, 300, 300}
00438 , { 300, 300, 300, 300, 300}
00439 , { 300, 300, 300, 300, 300}
00440 , { 300, 300, 300, 300, 300}
00441 , { 300, 300, 300, 300, 300}
00442 }
00443 , { { 300, 300, 300, 190, 300}
00444 , { 300, 300, 300, 190, 300}
00445 , { 300, 300, 300, 190, 300}
00446 , { 190, 190, 190, 190, 190}
00447 , { 300, 300, 300, 190, 300}
00448 }
00449 , { { 300, 300, 300, 300, 300}
00450 , { 300, 300, 300, 300, 300}
00451 , { 300, 300, 300, 300, 300}
00452 , { 300, 300, 300, 300, 300}
00453 , { 300, 300, 300, 300, 300}
00454 }
00455 , { { 300, 190, 300, 190, 300}
00456 , { 190, 190, 190, 190, 190}
00457 , { 300, 190, 300, 190, 300}
00458 , { 190, 190, 190, 190, 190}
00459 , { 300, 190, 300, 190, 300}
00460 }
00461 , { { 300, 300, 300, 300, 220}
00462 , { 300, 300, 300, 300, 220}
00463 , { 300, 300, 300, 300, 220}
00464 , { 300, 300, 300, 300, 220}
00465 , { 220, 220, 220, 220, 220}
00466 }
00467 }
00468 , { { { 300, 300, 300, 300, 300}
00469 , { 300, 300, 300, 300, 300}
00470 , { 300, 300, 300, 300, 300}
00471 , { 300, 300, 300, 300, 300}
00472 , { 300, 300, 300, 300, 300}
00473 }
00474 , { { 300, 300, 300, 190, 300}
00475 , { 300, 300, 300, 190, 300}
00476 , { 300, 300, 300, 190, 300}
00477 , { 190, 190, 190, 190, 190}
00478 , { 300, 300, 300, 190, 300}
00479 }
00480 , { { 300, 300, 300, 300, 300}
00481 , { 300, 300, 300, 300, 300}
00482 , { 300, 300, 300, 300, 300}
00483 , { 300, 300, 300, 300, 300}
00484 , { 300, 300, 300, 300, 300}
00485 }
00486 , { { 300, 190, 300, 190, 300}
00487 , { 300, 190, 300, 190, 300}
```



```
00488     , { 300, 190, 300, 190, 300 }
00489     , { 190, 190, 190, 190, 190 }
00490     , { 300, 190, 300, 190, 300 }
00491     }
00492     , { { 300, 300, 300, 300, 220 }
00493     , { 300, 300, 300, 300, 220 }
00494     , { 300, 300, 300, 300, 220 }
00495     , { 300, 300, 300, 300, 220 }
00496     , { 220, 220, 220, 220, 220 }
00497     }
00498     }
00499     }
00500     , { { { INF, INF, INF, INF, INF }
00501     , { INF, INF, INF, INF, INF }
00502     , { INF, INF, INF, INF, INF }
00503     , { INF, INF, INF, INF, INF }
00504     , { INF, INF, INF, INF, INF }
00505     }
00506     , { { INF, INF, INF, INF, INF }
00507     , { INF, INF, INF, INF, INF }
00508     , { INF, INF, INF, INF, INF }
00509     , { INF, INF, INF, INF, INF }
00510     , { INF, INF, INF, INF, INF }
00511     }
00512     , { { INF, INF, INF, INF, INF }
00513     , { INF, INF, INF, INF, INF }
00514     , { INF, INF, INF, INF, INF }
00515     , { INF, INF, INF, INF, INF }
00516     , { INF, INF, INF, INF, INF }
00517     }
00518     , { { INF, INF, INF, INF, INF }
00519     , { INF, INF, INF, INF, INF }
00520     , { INF, INF, INF, INF, INF }
00521     , { INF, INF, INF, INF, INF }
00522     , { INF, INF, INF, INF, INF }
00523     }
00524     , { { INF, INF, INF, INF, INF }
00525     , { INF, INF, INF, INF, INF }
00526     , { INF, INF, INF, INF, INF }
00527     , { INF, INF, INF, INF, INF }
00528     , { INF, INF, INF, INF, INF }
00529     }
00530     }
00531     , { { { 250, 250, 230, 230, 230 }
00532     , { 250, 250, 230, 230, 230 }
00533     , { 230, 230, 230, 230, 230 }
00534     , { 230, 230, 230, 230, 230 }
00535     , { 230, 230, 230, 230, 230 }
00536     }
00537     , { { 250, 250, 230, 230, 230 }
00538     , { 250, 250, 230, 210, 230 }
00539     , { 230, 230, 230, 230, 230 }
00540     , { 120, 120, 110, 110, 110 }
00541     , { 230, 230, 230, 230, 230 }
00542     }
00543     , { { 230, 230, 230, 230, 230 }
00544     , { 230, 230, 230, 230, 230 }
00545     , { 230, 230, 230, 230, 230 }
00546     , { 230, 230, 230, 230, 230 }
00547     , { 230, 230, 190, 230, 230 }
00548     }
00549     , { { 230, 110, 230, 110, 230 }
00550     , { 110, 110, 110, 110, 110 }
00551     , { 230, 110, 230, 110, 230 }
00552     , { 110, 110, 110, 110, 110 }
00553     , { 230, 110, 230, 110, 230 }
00554     }
00555     , { { 230, 230, 230, 230, 150 }
00556     , { 230, 230, 230, 230, 150 }
00557     , { 230, 230, 230, 230, 150 }
00558     , { 230, 230, 230, 230, 150 }
00559     , { 150, 150, 150, 150, 150 }
00560     }
00561     }
00562     , { { { 230, 230, 230, 230, 230 }
00563     , { 230, 230, 230, 230, 230 }
00564     , { 230, 230, 230, 230, 230 }
00565     , { 230, 230, 230, 230, 230 }
00566     , { 230, 230, 230, 230, 230 }
00567     }
00568     , { { 230, 230, 230, 230, 230 }
00569     , { 230, 230, 230, 230, 230 }
00570     , { 230, 230, 230, 230, 230 }
00571     , { 110, 110, 110, 110, 110 }
00572     , { 230, 230, 230, 230, 230 }
00573     }
00574     , { { 230, 230, 230, 230, 230 }
```

```
00575 , { 230, 230, 230, 230, 230 }
00576 , { 230, 230, 230, 230, 230 }
00577 , { 230, 230, 230, 230, 230 }
00578 , { 230, 230, 230, 230, 230 }
00579 }
00580 , { { 230, 110, 230, 110, 230 }
00581 , { 230, 110, 230, 110, 230 }
00582 , { 230, 110, 230, 110, 230 }
00583 , { 110, 110, 110, 110, 110 }
00584 , { 230, 110, 230, 110, 230 }
00585 }
00586 , { { 230, 230, 230, 230, 150 }
00587 , { 230, 230, 230, 230, 150 }
00588 , { 230, 230, 230, 230, 150 }
00589 , { 230, 230, 230, 230, 150 }
00590 , { 150, 150, 150, 150, 150 }
00591 }
00592 }
00593 , { { { 300, 300, 300, 300, 300 }
00594 , { 300, 300, 300, 300, 300 }
00595 , { 300, 300, 300, 300, 300 }
00596 , { 300, 300, 300, 300, 300 }
00597 , { 300, 300, 300, 300, 300 }
00598 }
00599 , { { 300, 300, 300, 300, 300 }
00600 , { 300, 300, 300, 300, 300 }
00601 , { 300, 300, 300, 300, 300 }
00602 , { 190, 190, 190, 190, 190 }
00603 , { 300, 300, 300, 300, 300 }
00604 }
00605 , { { 300, 300, 300, 300, 300 }
00606 , { 300, 300, 300, 300, 300 }
00607 , { 300, 300, 300, 300, 300 }
00608 , { 300, 300, 300, 300, 300 }
00609 , { 300, 300, 300, 300, 300 }
00610 }
00611 , { { 300, 190, 300, 190, 300 }
00612 , { 300, 190, 300, 190, 300 }
00613 , { 300, 190, 300, 190, 300 }
00614 , { 190, 190, 190, 190, 190 }
00615 , { 300, 190, 300, 190, 300 }
00616 }
00617 , { { 300, 300, 300, 300, 220 }
00618 , { 300, 300, 300, 300, 220 }
00619 , { 300, 300, 300, 300, 220 }
00620 , { 300, 300, 300, 300, 220 }
00621 , { 220, 220, 220, 220, 220 }
00622 }
00623 }
00624 , { { { 300, 300, 300, 300, 300 }
00625 , { 300, 300, 300, 300, 300 }
00626 , { 300, 300, 300, 300, 300 }
00627 , { 300, 300, 300, 300, 300 }
00628 , { 300, 300, 300, 300, 300 }
00629 }
00630 , { { 300, 300, 300, 300, 300 }
00631 , { 300, 250, 300, 210, 300 }
00632 , { 300, 300, 300, 300, 300 }
00633 , { 190, 120, 190, 190, 190 }
00634 , { 300, 300, 300, 300, 300 }
00635 }
00636 , { { 300, 300, 300, 300, 300 }
00637 , { 300, 300, 300, 300, 300 }
00638 , { 300, 300, 300, 300, 300 }
00639 , { 300, 300, 300, 300, 300 }
00640 , { 300, 300, 190, 300, 300 }
00641 }
00642 , { { 300, 190, 300, 190, 300 }
00643 , { 190, 190, 190, 190, 190 }
00644 , { 300, 190, 300, 190, 300 }
00645 , { 190, 190, 190, 190, 190 }
00646 , { 300, 190, 300, 190, 300 }
00647 }
00648 , { { 300, 300, 300, 300, 220 }
00649 , { 300, 300, 300, 300, 220 }
00650 , { 300, 300, 300, 300, 220 }
00651 , { 300, 300, 300, 300, 220 }
00652 , { 220, 220, 220, 220, 220 }
00653 }
00654 }
00655 , { { { 300, 300, 300, 300, 300 }
00656 , { 300, 300, 300, 300, 300 }
00657 , { 300, 300, 300, 300, 300 }
00658 , { 300, 300, 300, 300, 300 }
00659 , { 300, 300, 300, 300, 300 }
00660 }
00661 , { { 300, 300, 300, 300, 300 }
```

```
00662 , { 300, 300, 300, 300, 300}
00663 , { 300, 300, 300, 300, 300}
00664 , { 190, 190, 190, 190, 190}
00665 , { 300, 300, 300, 300, 300}
00666 }
00667 , { { 300, 300, 300, 300, 300}
00668 , { 300, 300, 300, 300, 300}
00669 , { 300, 300, 300, 300, 300}
00670 , { 300, 300, 300, 300, 300}
00671 , { 300, 300, 300, 300, 300}
00672 }
00673 , { { 300, 190, 300, 190, 300}
00674 , { 300, 190, 300, 190, 300}
00675 , { 300, 190, 300, 190, 300}
00676 , { 190, 190, 190, 190, 190}
00677 , { 300, 190, 300, 190, 300}
00678 }
00679 , { { 300, 300, 300, 300, 220}
00680 , { 300, 300, 300, 300, 220}
00681 , { 300, 300, 300, 300, 220}
00682 , { 300, 300, 300, 300, 220}
00683 , { 220, 220, 220, 220, 220}
00684 }
00685 }
00686 , { { { 300, 300, 300, 300, 300}
00687 , { 300, 300, 300, 300, 300}
00688 , { 300, 300, 300, 300, 300}
00689 , { 300, 300, 300, 300, 300}
00690 , { 300, 300, 300, 300, 300}
00691 }
00692 , { { 300, 300, 300, 300, 300}
00693 , { 300, 300, 300, 300, 300}
00694 , { 300, 300, 300, 300, 300}
00695 , { 190, 190, 190, 190, 190}
00696 , { 300, 300, 300, 300, 300}
00697 }
00698 , { { 300, 300, 300, 300, 300}
00699 , { 300, 300, 300, 300, 300}
00700 , { 300, 300, 300, 300, 300}
00701 , { 300, 300, 300, 300, 300}
00702 , { 300, 300, 300, 300, 300}
00703 }
00704 , { { 300, 190, 300, 190, 300}
00705 , { 190, 190, 190, 190, 190}
00706 , { 300, 190, 300, 190, 300}
00707 , { 190, 190, 190, 190, 190}
00708 , { 300, 190, 300, 190, 300}
00709 }
00710 , { { 300, 300, 300, 300, 220}
00711 , { 300, 300, 300, 300, 220}
00712 , { 300, 300, 300, 300, 220}
00713 , { 300, 300, 300, 300, 220}
00714 , { 220, 220, 220, 220, 220}
00715 }
00716 }
00717 , { { { 300, 300, 300, 300, 300}
00718 , { 300, 300, 300, 300, 300}
00719 , { 300, 300, 300, 300, 300}
00720 , { 300, 300, 300, 300, 300}
00721 , { 300, 300, 300, 300, 300}
00722 }
00723 , { { 300, 300, 300, 300, 300}
00724 , { 300, 300, 300, 300, 300}
00725 , { 300, 300, 300, 300, 300}
00726 , { 190, 190, 190, 190, 190}
00727 , { 300, 300, 300, 300, 300}
00728 }
00729 , { { 300, 300, 300, 300, 300}
00730 , { 300, 300, 300, 300, 300}
00731 , { 300, 300, 300, 300, 300}
00732 , { 300, 300, 300, 300, 300}
00733 , { 300, 300, 300, 300, 300}
00734 }
00735 , { { 300, 190, 300, 190, 300}
00736 , { 300, 190, 300, 190, 300}
00737 , { 300, 190, 300, 190, 300}
00738 , { 190, 190, 190, 190, 190}
00739 , { 300, 190, 300, 190, 300}
00740 }
00741 , { { 300, 300, 300, 300, 220}
00742 , { 300, 300, 300, 300, 220}
00743 , { 300, 300, 300, 300, 220}
00744 , { 300, 300, 300, 300, 220}
00745 , { 220, 220, 220, 220, 220}
00746 }
00747 }
00748 }
```

```
00749 ,{{{ INF, INF, INF, INF, INF }
00750 ,{ INF, INF, INF, INF, INF }
00751 ,{ INF, INF, INF, INF, INF }
00752 ,{ INF, INF, INF, INF, INF }
00753 ,{ INF, INF, INF, INF, INF }
00754 }
00755 ,{{{ INF, INF, INF, INF, INF }
00756 ,{ INF, INF, INF, INF, INF }
00757 ,{ INF, INF, INF, INF, INF }
00758 ,{ INF, INF, INF, INF, INF }
00759 ,{ INF, INF, INF, INF, INF }
00760 }
00761 ,{{{ INF, INF, INF, INF, INF }
00762 ,{ INF, INF, INF, INF, INF }
00763 ,{ INF, INF, INF, INF, INF }
00764 ,{ INF, INF, INF, INF, INF }
00765 ,{ INF, INF, INF, INF, INF }
00766 }
00767 ,{{{ INF, INF, INF, INF, INF }
00768 ,{ INF, INF, INF, INF, INF }
00769 ,{ INF, INF, INF, INF, INF }
00770 ,{ INF, INF, INF, INF, INF }
00771 ,{ INF, INF, INF, INF, INF }
00772 }
00773 ,{{{ INF, INF, INF, INF, INF }
00774 ,{ INF, INF, INF, INF, INF }
00775 ,{ INF, INF, INF, INF, INF }
00776 ,{ INF, INF, INF, INF, INF }
00777 ,{ INF, INF, INF, INF, INF }
00778 }
00779 }
00780 ,{{{ 300, 300, 300, 300, 300 }
00781 ,{ 300, 300, 300, 300, 300 }
00782 ,{ 300, 300, 300, 300, 300 }
00783 ,{ 300, 300, 300, 300, 300 }
00784 ,{ 300, 300, 300, 300, 300 }
00785 }
00786 ,{{{ 300, 300, 300, 300, 300 }
00787 ,{ 300, 250, 300, 210, 300 }
00788 ,{ 300, 300, 300, 300, 300 }
00789 ,{ 190, 120, 190, 190, 190 }
00790 ,{ 300, 300, 300, 300, 300 }
00791 }
00792 ,{{{ 300, 300, 300, 300, 300 }
00793 ,{ 300, 300, 300, 300, 300 }
00794 ,{ 300, 300, 300, 300, 300 }
00795 ,{ 300, 300, 300, 300, 300 }
00796 ,{ 300, 300, 190, 300, 300 }
00797 }
00798 ,{{{ 300, 190, 300, 190, 300 }
00799 ,{ 190, 190, 190, 190, 190 }
00800 ,{ 300, 190, 300, 190, 300 }
00801 ,{ 190, 190, 190, 190, 190 }
00802 ,{ 300, 190, 300, 190, 300 }
00803 }
00804 ,{{{ 300, 300, 300, 300, 220 }
00805 ,{ 300, 300, 300, 300, 220 }
00806 ,{ 300, 300, 300, 300, 220 }
00807 ,{ 300, 300, 300, 300, 220 }
00808 ,{ 220, 220, 220, 220, 220 }
00809 }
00810 }
00811 ,{{{ 300, 300, 300, 300, 300 }
00812 ,{ 300, 300, 300, 300, 300 }
00813 ,{ 300, 300, 300, 300, 300 }
00814 ,{ 300, 300, 300, 300, 300 }
00815 ,{ 300, 300, 300, 300, 300 }
00816 }
00817 ,{{{ 300, 300, 300, 300, 300 }
00818 ,{ 300, 300, 300, 300, 300 }
00819 ,{ 300, 300, 300, 300, 300 }
00820 ,{ 190, 190, 190, 190, 190 }
00821 ,{ 300, 300, 300, 300, 300 }
00822 }
00823 ,{{{ 300, 300, 300, 300, 300 }
00824 ,{ 300, 300, 300, 300, 300 }
00825 ,{ 300, 300, 300, 300, 300 }
00826 ,{ 300, 300, 300, 300, 300 }
00827 ,{ 300, 300, 300, 300, 300 }
00828 }
00829 ,{{{ 300, 190, 300, 190, 300 }
00830 ,{ 300, 190, 300, 190, 300 }
00831 ,{ 300, 190, 300, 190, 300 }
00832 ,{ 190, 190, 190, 190, 190 }
00833 ,{ 300, 190, 300, 190, 300 }
00834 }
00835 ,{{{ 300, 300, 300, 300, 220 }
```

```
00836 , { 300, 300, 300, 300, 220 }
00837 , { 300, 300, 300, 300, 220 }
00838 , { 300, 300, 300, 300, 220 }
00839 , { 220, 220, 220, 220, 220 }
00840 }
00841 }
00842 , { { 370, 370, 370, 370, 370 }
00843 , { 370, 370, 370, 370, 370 }
00844 , { 370, 370, 370, 370, 370 }
00845 , { 370, 370, 370, 370, 370 }
00846 , { 370, 370, 370, 370, 370 }
00847 }
00848 , { { 370, 370, 370, 370, 370 }
00849 , { 370, 370, 370, 370, 370 }
00850 , { 370, 370, 370, 370, 370 }
00851 , { 260, 260, 260, 260, 260 }
00852 , { 370, 370, 370, 370, 370 }
00853 }
00854 , { { 370, 370, 370, 370, 370 }
00855 , { 370, 370, 370, 370, 370 }
00856 , { 370, 370, 370, 370, 370 }
00857 , { 370, 370, 370, 370, 370 }
00858 , { 370, 370, 370, 370, 370 }
00859 }
00860 , { { 370, 260, 370, 260, 370 }
00861 , { 370, 260, 370, 260, 370 }
00862 , { 370, 260, 370, 260, 370 }
00863 , { 260, 260, 260, 260, 260 }
00864 , { 370, 260, 370, 260, 370 }
00865 }
00866 , { { 370, 370, 370, 370, 300 }
00867 , { 370, 370, 370, 370, 300 }
00868 , { 370, 370, 370, 370, 300 }
00869 , { 370, 370, 370, 370, 300 }
00870 , { 300, 300, 300, 300, 300 }
00871 }
00872 }
00873 , { { { 370, 370, 370, 370, 370 }
00874 , { 370, 370, 370, 370, 370 }
00875 , { 370, 370, 370, 370, 370 }
00876 , { 370, 370, 370, 370, 370 }
00877 , { 370, 370, 370, 370, 370 }
00878 }
00879 , { { 370, 370, 370, 370, 370 }
00880 , { 370, 250, 370, 210, 370 }
00881 , { 370, 370, 370, 370, 370 }
00882 , { 260, 120, 260, 260, 260 }
00883 , { 370, 370, 370, 370, 370 }
00884 }
00885 , { { 370, 370, 370, 370, 370 }
00886 , { 370, 370, 370, 370, 370 }
00887 , { 370, 370, 370, 370, 370 }
00888 , { 370, 370, 370, 370, 370 }
00889 , { 370, 370, 190, 370, 370 }
00890 }
00891 , { { 370, 260, 370, 260, 370 }
00892 , { 260, 260, 260, 260, 260 }
00893 , { 370, 260, 370, 260, 370 }
00894 , { 260, 260, 260, 260, 260 }
00895 , { 370, 260, 370, 260, 370 }
00896 }
00897 , { { 370, 370, 370, 370, 300 }
00898 , { 370, 370, 370, 370, 300 }
00899 , { 370, 370, 370, 370, 300 }
00900 , { 370, 370, 370, 370, 300 }
00901 , { 300, 300, 300, 300, 300 }
00902 }
00903 }
00904 , { { { 370, 370, 370, 370, 370 }
00905 , { 370, 370, 370, 370, 370 }
00906 , { 370, 370, 370, 370, 370 }
00907 , { 370, 370, 370, 370, 370 }
00908 , { 370, 370, 370, 370, 370 }
00909 }
00910 , { { 370, 370, 370, 370, 370 }
00911 , { 370, 370, 370, 370, 370 }
00912 , { 370, 370, 370, 370, 370 }
00913 , { 260, 260, 260, 260, 260 }
00914 , { 370, 370, 370, 370, 370 }
00915 }
00916 , { { 370, 370, 370, 370, 370 }
00917 , { 370, 370, 370, 370, 370 }
00918 , { 370, 370, 370, 370, 370 }
00919 , { 370, 370, 370, 370, 370 }
00920 , { 370, 370, 370, 370, 370 }
00921 }
00922 , { { 370, 260, 370, 260, 370 }
```

```
00923 , { 370, 260, 370, 260, 370}
00924 , { 370, 260, 370, 260, 370}
00925 , { 260, 260, 260, 260, 260}
00926 , { 370, 260, 370, 260, 370}
00927 }
00928 , { { 370, 370, 370, 370, 300}
00929 , { 370, 370, 370, 370, 300}
00930 , { 370, 370, 370, 370, 300}
00931 , { 370, 370, 370, 370, 300}
00932 , { 300, 300, 300, 300, 300}
00933 }
00934 }
00935 , { { { 370, 370, 370, 370, 370}
00936 , { 370, 370, 370, 370, 370}
00937 , { 370, 370, 370, 370, 370}
00938 , { 370, 370, 370, 370, 370}
00939 , { 370, 370, 370, 370, 370}
00940 }
00941 , { { 370, 370, 370, 370, 370}
00942 , { 370, 370, 370, 370, 370}
00943 , { 370, 370, 370, 370, 370}
00944 , { 260, 260, 260, 260, 260}
00945 , { 370, 370, 370, 370, 370}
00946 }
00947 , { { 370, 370, 370, 370, 370}
00948 , { 370, 370, 370, 370, 370}
00949 , { 370, 370, 370, 370, 370}
00950 , { 370, 370, 370, 370, 370}
00951 , { 370, 370, 370, 370, 370}
00952 }
00953 , { { 370, 260, 370, 260, 370}
00954 , { 260, 260, 260, 260, 260}
00955 , { 370, 260, 370, 260, 370}
00956 , { 260, 260, 260, 260, 260}
00957 , { 370, 260, 370, 260, 370}
00958 }
00959 , { { 370, 370, 370, 370, 300}
00960 , { 370, 370, 370, 370, 300}
00961 , { 370, 370, 370, 370, 300}
00962 , { 370, 370, 370, 370, 300}
00963 , { 300, 300, 300, 300, 300}
00964 }
00965 }
00966 , { { { 370, 370, 370, 370, 370}
00967 , { 370, 370, 370, 370, 370}
00968 , { 370, 370, 370, 370, 370}
00969 , { 370, 370, 370, 370, 370}
00970 , { 370, 370, 370, 370, 370}
00971 }
00972 , { { 370, 370, 370, 370, 370}
00973 , { 370, 370, 370, 370, 370}
00974 , { 370, 370, 370, 370, 370}
00975 , { 260, 260, 260, 260, 260}
00976 , { 370, 370, 370, 370, 370}
00977 }
00978 , { { 370, 370, 370, 370, 370}
00979 , { 370, 370, 370, 370, 370}
00980 , { 370, 370, 370, 370, 370}
00981 , { 370, 370, 370, 370, 370}
00982 , { 370, 370, 370, 370, 370}
00983 }
00984 , { { 370, 260, 370, 260, 370}
00985 , { 370, 260, 370, 260, 370}
00986 , { 370, 260, 370, 260, 370}
00987 , { 260, 260, 260, 260, 260}
00988 , { 370, 260, 370, 260, 370}
00989 }
00990 , { { 370, 370, 370, 370, 300}
00991 , { 370, 370, 370, 370, 300}
00992 , { 370, 370, 370, 370, 300}
00993 , { 370, 370, 370, 370, 300}
00994 , { 300, 300, 300, 300, 300}
00995 }
00996 }
00997 }
00998 , { { { { INF, INF, INF, INF, INF}
00999 , { INF, INF, INF, INF, INF}
01000 , { INF, INF, INF, INF, INF}
01001 , { INF, INF, INF, INF, INF}
01002 , { INF, INF, INF, INF, INF}
01003 }
01004 , { { INF, INF, INF, INF, INF}
01005 , { INF, INF, INF, INF, INF}
01006 , { INF, INF, INF, INF, INF}
01007 , { INF, INF, INF, INF, INF}
01008 , { INF, INF, INF, INF, INF}
01009 }
```

```
01010 ,{{ INF, INF, INF, INF, INF}
01011 ,{ INF, INF, INF, INF, INF}
01012 ,{ INF, INF, INF, INF, INF}
01013 ,{ INF, INF, INF, INF, INF}
01014 ,{ INF, INF, INF, INF, INF}
01015 }
01016 ,{{ INF, INF, INF, INF, INF}
01017 ,{ INF, INF, INF, INF, INF}
01018 ,{ INF, INF, INF, INF, INF}
01019 ,{ INF, INF, INF, INF, INF}
01020 ,{ INF, INF, INF, INF, INF}
01021 }
01022 ,{{ INF, INF, INF, INF, INF}
01023 ,{ INF, INF, INF, INF, INF}
01024 ,{ INF, INF, INF, INF, INF}
01025 ,{ INF, INF, INF, INF, INF}
01026 ,{ INF, INF, INF, INF, INF}
01027 }
01028 }
01029 ,{{{ 300, 300, 300, 300, 300}
01030 ,{ 300, 300, 300, 300, 300}
01031 ,{ 300, 300, 300, 300, 300}
01032 ,{ 300, 300, 300, 300, 300}
01033 ,{ 300, 300, 300, 300, 300}
01034 }
01035 ,{{{ 300, 300, 300, 190, 300}
01036 ,{ 300, 300, 300, 190, 300}
01037 ,{ 300, 300, 300, 190, 300}
01038 ,{ 190, 190, 190, 190, 190}
01039 ,{ 300, 300, 300, 190, 300}
01040 }
01041 ,{{{ 300, 300, 300, 300, 300}
01042 ,{ 300, 300, 300, 300, 300}
01043 ,{ 300, 300, 300, 300, 300}
01044 ,{ 300, 300, 300, 300, 300}
01045 ,{ 300, 300, 300, 300, 300}
01046 }
01047 ,{{{ 300, 190, 300, 190, 300}
01048 ,{ 190, 190, 190, 190, 190}
01049 ,{ 300, 190, 300, 190, 300}
01050 ,{ 190, 190, 190, 190, 190}
01051 ,{ 300, 190, 300, 190, 300}
01052 }
01053 ,{{{ 300, 300, 300, 300, 220}
01054 ,{ 300, 300, 300, 300, 220}
01055 ,{ 300, 300, 300, 300, 220}
01056 ,{ 300, 300, 300, 300, 220}
01057 ,{ 220, 220, 220, 220, 220}
01058 }
01059 }
01060 ,{{{ 300, 300, 300, 300, 300}
01061 ,{ 300, 300, 300, 300, 300}
01062 ,{ 300, 300, 300, 300, 300}
01063 ,{ 300, 300, 300, 300, 300}
01064 ,{ 300, 300, 300, 300, 300}
01065 }
01066 ,{{{ 300, 300, 300, 190, 300}
01067 ,{ 300, 300, 300, 190, 300}
01068 ,{ 300, 300, 300, 190, 300}
01069 ,{ 190, 190, 190, 190, 190}
01070 ,{ 300, 300, 300, 190, 300}
01071 }
01072 ,{{{ 300, 300, 300, 300, 300}
01073 ,{ 300, 300, 300, 300, 300}
01074 ,{ 300, 300, 300, 300, 300}
01075 ,{ 300, 300, 300, 300, 300}
01076 ,{ 300, 300, 300, 300, 300}
01077 }
01078 ,{{{ 300, 190, 300, 190, 300}
01079 ,{ 300, 190, 300, 190, 300}
01080 ,{ 300, 190, 300, 190, 300}
01081 ,{ 190, 190, 190, 190, 190}
01082 ,{ 300, 190, 300, 190, 300}
01083 }
01084 ,{{{ 300, 300, 300, 300, 220}
01085 ,{ 300, 300, 300, 300, 220}
01086 ,{ 300, 300, 300, 300, 220}
01087 ,{ 300, 300, 300, 300, 220}
01088 ,{ 220, 220, 220, 220, 220}
01089 }
01090 }
01091 ,{{{ 370, 370, 370, 370, 370}
01092 ,{ 370, 370, 370, 370, 370}
01093 ,{ 370, 370, 370, 370, 370}
01094 ,{ 370, 370, 370, 370, 370}
01095 ,{ 370, 370, 370, 370, 370}
01096 }
```

```
01097 ,{{ 370, 370, 370, 260, 370}
01098 ,{ 370, 370, 370, 260, 370}
01099 ,{ 370, 370, 370, 260, 370}
01100 ,{ 260, 260, 260, 260, 260}
01101 ,{ 370, 370, 370, 260, 370}
01102 }
01103 ,{{ 370, 370, 370, 370, 370}
01104 ,{ 370, 370, 370, 370, 370}
01105 ,{ 370, 370, 370, 370, 370}
01106 ,{ 370, 370, 370, 370, 370}
01107 ,{ 370, 370, 370, 370, 370}
01108 }
01109 ,{{ 370, 260, 370, 260, 370}
01110 ,{ 370, 260, 370, 260, 370}
01111 ,{ 370, 260, 370, 260, 370}
01112 ,{ 260, 260, 260, 260, 260}
01113 ,{ 370, 260, 370, 260, 370}
01114 }
01115 ,{{ 370, 370, 370, 370, 300}
01116 ,{ 370, 370, 370, 370, 300}
01117 ,{ 370, 370, 370, 370, 300}
01118 ,{ 370, 370, 370, 370, 300}
01119 ,{ 300, 300, 300, 300, 300}
01120 }
01121 }
01122 ,{{{ 370, 370, 370, 370, 370}
01123 ,{ 370, 370, 370, 370, 370}
01124 ,{ 370, 370, 370, 370, 370}
01125 ,{ 370, 370, 370, 370, 370}
01126 ,{ 370, 370, 370, 370, 370}
01127 }
01128 ,{{ 370, 370, 370, 260, 370}
01129 ,{ 370, 370, 370, 260, 370}
01130 ,{ 370, 370, 370, 260, 370}
01131 ,{ 260, 260, 260, 260, 260}
01132 ,{ 370, 370, 370, 260, 370}
01133 }
01134 ,{{{ 370, 370, 370, 370, 370}
01135 ,{ 370, 370, 370, 370, 370}
01136 ,{ 370, 370, 370, 370, 370}
01137 ,{ 370, 370, 370, 370, 370}
01138 ,{ 370, 370, 370, 370, 370}
01139 }
01140 ,{{{ 370, 260, 370, 260, 370}
01141 ,{ 260, 260, 260, 260, 260}
01142 ,{ 370, 260, 370, 260, 370}
01143 ,{ 260, 260, 260, 260, 260}
01144 ,{ 370, 260, 370, 260, 370}
01145 }
01146 ,{{{ 370, 370, 370, 370, 300}
01147 ,{ 370, 370, 370, 370, 300}
01148 ,{ 370, 370, 370, 370, 300}
01149 ,{ 370, 370, 370, 370, 300}
01150 ,{ 300, 300, 300, 300, 300}
01151 }
01152 }
01153 ,{{{ 370, 370, 370, 370, 370}
01154 ,{ 370, 370, 370, 370, 370}
01155 ,{ 370, 370, 370, 370, 370}
01156 ,{ 370, 370, 370, 370, 370}
01157 ,{ 370, 370, 370, 370, 370}
01158 }
01159 ,{{{ 370, 370, 370, 260, 370}
01160 ,{ 370, 370, 370, 260, 370}
01161 ,{ 370, 370, 370, 260, 370}
01162 ,{ 260, 260, 260, 260, 260}
01163 ,{ 370, 370, 370, 260, 370}
01164 }
01165 ,{{{ 370, 370, 370, 370, 370}
01166 ,{ 370, 370, 370, 370, 370}
01167 ,{ 370, 370, 370, 370, 370}
01168 ,{ 370, 370, 370, 370, 370}
01169 ,{ 370, 370, 370, 370, 370}
01170 }
01171 ,{{{ 370, 260, 370, 260, 370}
01172 ,{ 370, 260, 370, 260, 370}
01173 ,{ 370, 260, 370, 260, 370}
01174 ,{ 260, 260, 260, 260, 260}
01175 ,{ 370, 260, 370, 260, 370}
01176 }
01177 ,{{{ 370, 370, 370, 370, 300}
01178 ,{ 370, 370, 370, 370, 300}
01179 ,{ 370, 370, 370, 370, 300}
01180 ,{ 370, 370, 370, 370, 300}
01181 ,{ 300, 300, 300, 300, 300}
01182 }
01183 }
```



```
01184 ,{{{ 370, 370, 370, 370, 370}
01185 ,{ 370, 370, 370, 370, 370}
01186 ,{ 370, 370, 370, 370, 370}
01187 ,{ 370, 370, 370, 370, 370}
01188 ,{ 370, 370, 370, 370, 370}
01189 }
01190 ,{{{ 370, 370, 370, 260, 370}
01191 ,{ 370, 370, 370, 260, 370}
01192 ,{ 370, 370, 370, 260, 370}
01193 ,{ 260, 260, 260, 260, 260}
01194 ,{ 370, 370, 370, 260, 370}
01195 }
01196 ,{{{ 370, 370, 370, 370, 370}
01197 ,{ 370, 370, 370, 370, 370}
01198 ,{ 370, 370, 370, 370, 370}
01199 ,{ 370, 370, 370, 370, 370}
01200 ,{ 370, 370, 370, 370, 370}
01201 }
01202 ,{{{ 370, 260, 370, 260, 370}
01203 ,{ 260, 260, 260, 260, 260}
01204 ,{ 370, 260, 370, 260, 370}
01205 ,{ 260, 260, 260, 260, 260}
01206 ,{ 370, 260, 370, 260, 370}
01207 }
01208 ,{{{ 370, 370, 370, 370, 300}
01209 ,{ 370, 370, 370, 370, 300}
01210 ,{ 370, 370, 370, 370, 300}
01211 ,{ 370, 370, 370, 370, 300}
01212 ,{ 300, 300, 300, 300, 300}
01213 }
01214 }
01215 ,{{{ 370, 370, 370, 370, 370}
01216 ,{ 370, 370, 370, 370, 370}
01217 ,{ 370, 370, 370, 370, 370}
01218 ,{ 370, 370, 370, 370, 370}
01219 ,{ 370, 370, 370, 370, 370}
01220 }
01221 ,{{{ 370, 370, 370, 260, 370}
01222 ,{ 370, 370, 370, 260, 370}
01223 ,{ 370, 370, 370, 260, 370}
01224 ,{ 260, 260, 260, 260, 260}
01225 ,{ 370, 370, 370, 260, 370}
01226 }
01227 ,{{{ 370, 370, 370, 370, 370}
01228 ,{ 370, 370, 370, 370, 370}
01229 ,{ 370, 370, 370, 370, 370}
01230 ,{ 370, 370, 370, 370, 370}
01231 ,{ 370, 370, 370, 370, 370}
01232 }
01233 ,{{{ 370, 260, 370, 260, 370}
01234 ,{ 370, 260, 370, 260, 370}
01235 ,{ 370, 260, 370, 260, 370}
01236 ,{ 260, 260, 260, 260, 260}
01237 ,{ 370, 260, 370, 260, 370}
01238 }
01239 ,{{{ 370, 370, 370, 370, 300}
01240 ,{ 370, 370, 370, 370, 300}
01241 ,{ 370, 370, 370, 370, 300}
01242 ,{ 370, 370, 370, 370, 300}
01243 ,{ 300, 300, 300, 300, 300}
01244 }
01245 }
01246 }
01247 ,{{{ INF, INF, INF, INF, INF}
01248 ,{ INF, INF, INF, INF, INF}
01249 ,{ INF, INF, INF, INF, INF}
01250 ,{ INF, INF, INF, INF, INF}
01251 ,{ INF, INF, INF, INF, INF}
01252 }
01253 ,{{{ INF, INF, INF, INF, INF}
01254 ,{ INF, INF, INF, INF, INF}
01255 ,{ INF, INF, INF, INF, INF}
01256 ,{ INF, INF, INF, INF, INF}
01257 ,{ INF, INF, INF, INF, INF}
01258 }
01259 ,{{{ INF, INF, INF, INF, INF}
01260 ,{ INF, INF, INF, INF, INF}
01261 ,{ INF, INF, INF, INF, INF}
01262 ,{ INF, INF, INF, INF, INF}
01263 ,{ INF, INF, INF, INF, INF}
01264 }
01265 ,{{{ INF, INF, INF, INF, INF}
01266 ,{ INF, INF, INF, INF, INF}
01267 ,{ INF, INF, INF, INF, INF}
01268 ,{ INF, INF, INF, INF, INF}
01269 ,{ INF, INF, INF, INF, INF}
01270 }
```

```
01271 ,{{ INF, INF, INF, INF, INF}
01272 ,{ INF, INF, INF, INF, INF}
01273 ,{ INF, INF, INF, INF, INF}
01274 ,{ INF, INF, INF, INF, INF}
01275 ,{ INF, INF, INF, INF, INF}
01276 }
01277 }
01278 ,{{{ 300, 300, 300, 300, 300}
01279 ,{ 300, 300, 300, 300, 300}
01280 ,{ 300, 300, 300, 300, 300}
01281 ,{ 300, 300, 300, 300, 300}
01282 ,{ 300, 300, 300, 300, 300}
01283 }
01284 ,{{{ 300, 300, 300, 300, 300}
01285 ,{ 300, 300, 300, 300, 300}
01286 ,{ 300, 300, 300, 300, 300}
01287 ,{ 190, 190, 190, 190, 190}
01288 ,{ 300, 300, 300, 300, 300}
01289 }
01290 ,{{{ 300, 300, 300, 300, 300}
01291 ,{ 300, 300, 300, 300, 300}
01292 ,{ 300, 300, 300, 300, 300}
01293 ,{ 300, 300, 300, 300, 300}
01294 ,{ 300, 300, 300, 300, 300}
01295 }
01296 ,{{{ 300, 190, 300, 190, 300}
01297 ,{ 190, 190, 190, 190, 190}
01298 ,{ 300, 190, 300, 190, 300}
01299 ,{ 190, 190, 190, 190, 190}
01300 ,{ 300, 190, 300, 190, 300}
01301 }
01302 ,{{{ 300, 300, 300, 300, 220}
01303 ,{ 300, 300, 300, 300, 220}
01304 ,{ 300, 300, 300, 300, 220}
01305 ,{ 300, 300, 300, 300, 220}
01306 ,{ 220, 220, 220, 220, 220}
01307 }
01308 }
01309 ,{{{ 300, 300, 300, 300, 300}
01310 ,{ 300, 300, 300, 300, 300}
01311 ,{ 300, 300, 300, 300, 300}
01312 ,{ 300, 300, 300, 300, 300}
01313 ,{ 300, 300, 300, 300, 300}
01314 }
01315 ,{{{ 300, 300, 300, 300, 300}
01316 ,{ 300, 300, 300, 300, 300}
01317 ,{ 300, 300, 300, 300, 300}
01318 ,{ 190, 190, 190, 190, 190}
01319 ,{ 300, 300, 300, 300, 300}
01320 }
01321 ,{{{ 300, 300, 300, 300, 300}
01322 ,{ 300, 300, 300, 300, 300}
01323 ,{ 300, 300, 300, 300, 300}
01324 ,{ 300, 300, 300, 300, 300}
01325 ,{ 300, 300, 300, 300, 300}
01326 }
01327 ,{{{ 300, 190, 300, 190, 300}
01328 ,{ 300, 190, 300, 190, 300}
01329 ,{ 300, 190, 300, 190, 300}
01330 ,{ 190, 190, 190, 190, 190}
01331 ,{ 300, 190, 300, 190, 300}
01332 }
01333 ,{{{ 300, 300, 300, 300, 220}
01334 ,{ 300, 300, 300, 300, 220}
01335 ,{ 300, 300, 300, 300, 220}
01336 ,{ 300, 300, 300, 300, 220}
01337 ,{ 220, 220, 220, 220, 220}
01338 }
01339 }
01340 ,{{{ 370, 370, 370, 370, 370}
01341 ,{ 370, 370, 370, 370, 370}
01342 ,{ 370, 370, 370, 370, 370}
01343 ,{ 370, 370, 370, 370, 370}
01344 ,{ 370, 370, 370, 370, 370}
01345 }
01346 ,{{{ 370, 370, 370, 370, 370}
01347 ,{ 370, 370, 370, 370, 370}
01348 ,{ 370, 370, 370, 370, 370}
01349 ,{ 260, 260, 260, 260, 260}
01350 ,{ 370, 370, 370, 370, 370}
01351 }
01352 ,{{{ 370, 370, 370, 370, 370}
01353 ,{ 370, 370, 370, 370, 370}
01354 ,{ 370, 370, 370, 370, 370}
01355 ,{ 370, 370, 370, 370, 370}
01356 ,{ 370, 370, 370, 370, 370}
01357 }
```

```
01358 ,{{ 370, 260, 370, 260, 370}
01359 ,{ 370, 260, 370, 260, 370}
01360 ,{ 370, 260, 370, 260, 370}
01361 ,{ 260, 260, 260, 260, 260}
01362 ,{ 370, 260, 370, 260, 370}
01363 }
01364 ,{{ 370, 370, 370, 370, 300}
01365 ,{ 370, 370, 370, 370, 300}
01366 ,{ 370, 370, 370, 370, 300}
01367 ,{ 370, 370, 370, 370, 300}
01368 ,{ 300, 300, 300, 300, 300}
01369 }
01370 }
01371 ,{{{ 370, 370, 370, 370, 370}
01372 ,{ 370, 370, 370, 370, 370}
01373 ,{ 370, 370, 370, 370, 370}
01374 ,{ 370, 370, 370, 370, 370}
01375 ,{ 370, 370, 370, 370, 370}
01376 }
01377 ,{{{ 370, 370, 370, 370, 370}
01378 ,{ 370, 370, 370, 370, 370}
01379 ,{ 370, 370, 370, 370, 370}
01380 ,{ 260, 260, 260, 260, 260}
01381 ,{ 370, 370, 370, 370, 370}
01382 }
01383 ,{{{ 370, 370, 370, 370, 370}
01384 ,{ 370, 370, 370, 370, 370}
01385 ,{ 370, 370, 370, 370, 370}
01386 ,{ 370, 370, 370, 370, 370}
01387 ,{ 370, 370, 370, 370, 370}
01388 }
01389 ,{{{ 370, 260, 370, 260, 370}
01390 ,{ 260, 260, 260, 260, 260}
01391 ,{ 370, 260, 370, 260, 370}
01392 ,{ 260, 260, 260, 260, 260}
01393 ,{ 370, 260, 370, 260, 370}
01394 }
01395 ,{{{ 370, 370, 370, 370, 300}
01396 ,{ 370, 370, 370, 370, 300}
01397 ,{ 370, 370, 370, 370, 300}
01398 ,{ 370, 370, 370, 370, 300}
01399 ,{ 300, 300, 300, 300, 300}
01400 }
01401 }
01402 ,{{{ 370, 370, 370, 370, 370}
01403 ,{ 370, 370, 370, 370, 370}
01404 ,{ 370, 370, 370, 370, 370}
01405 ,{ 370, 370, 370, 370, 370}
01406 ,{ 370, 370, 370, 370, 370}
01407 }
01408 ,{{{ 370, 370, 370, 370, 370}
01409 ,{ 370, 370, 370, 370, 370}
01410 ,{ 370, 370, 370, 370, 370}
01411 ,{ 260, 260, 260, 260, 260}
01412 ,{ 370, 370, 370, 370, 370}
01413 }
01414 ,{{{ 370, 370, 370, 370, 370}
01415 ,{ 370, 370, 370, 370, 370}
01416 ,{ 370, 370, 370, 370, 370}
01417 ,{ 370, 370, 370, 370, 370}
01418 ,{ 370, 370, 370, 370, 370}
01419 }
01420 ,{{{ 370, 260, 370, 260, 370}
01421 ,{ 370, 260, 370, 260, 370}
01422 ,{ 370, 260, 370, 260, 370}
01423 ,{ 260, 260, 260, 260, 260}
01424 ,{ 370, 260, 370, 260, 370}
01425 }
01426 ,{{{ 370, 370, 370, 370, 300}
01427 ,{ 370, 370, 370, 370, 300}
01428 ,{ 370, 370, 370, 370, 300}
01429 ,{ 370, 370, 370, 370, 300}
01430 ,{ 300, 300, 300, 300, 300}
01431 }
01432 }
01433 ,{{{ 370, 370, 370, 370, 370}
01434 ,{ 370, 370, 370, 370, 370}
01435 ,{ 370, 370, 370, 370, 370}
01436 ,{ 370, 370, 370, 370, 370}
01437 ,{ 370, 370, 370, 370, 370}
01438 }
01439 ,{{{ 370, 370, 370, 370, 370}
01440 ,{ 370, 370, 370, 370, 370}
01441 ,{ 370, 370, 370, 370, 370}
01442 ,{ 260, 260, 260, 260, 260}
01443 ,{ 370, 370, 370, 370, 370}
01444 }
```

```
01445 ,{{ 370, 370, 370, 370, 370}
01446 ,{ 370, 370, 370, 370, 370}
01447 ,{ 370, 370, 370, 370, 370}
01448 ,{ 370, 370, 370, 370, 370}
01449 ,{ 370, 370, 370, 370, 370}
01450 }
01451 ,{{ 370, 260, 370, 260, 370}
01452 ,{ 260, 260, 260, 260, 260}
01453 ,{ 370, 260, 370, 260, 370}
01454 ,{ 260, 260, 260, 260, 260}
01455 ,{ 370, 260, 370, 260, 370}
01456 }
01457 ,{{ 370, 370, 370, 370, 300}
01458 ,{ 370, 370, 370, 370, 300}
01459 ,{ 370, 370, 370, 370, 300}
01460 ,{ 370, 370, 370, 370, 300}
01461 ,{ 300, 300, 300, 300, 300}
01462 }
01463 }
01464 ,{{{ 370, 370, 370, 370, 370}
01465 ,{ 370, 370, 370, 370, 370}
01466 ,{ 370, 370, 370, 370, 370}
01467 ,{ 370, 370, 370, 370, 370}
01468 ,{ 370, 370, 370, 370, 370}
01469 }
01470 ,{{{ 370, 370, 370, 370, 370}
01471 ,{ 370, 370, 370, 370, 370}
01472 ,{ 370, 370, 370, 370, 370}
01473 ,{ 260, 260, 260, 260, 260}
01474 ,{ 370, 370, 370, 370, 370}
01475 }
01476 ,{{{ 370, 370, 370, 370, 370}
01477 ,{ 370, 370, 370, 370, 370}
01478 ,{ 370, 370, 370, 370, 370}
01479 ,{ 370, 370, 370, 370, 370}
01480 ,{ 370, 370, 370, 370, 370}
01481 }
01482 ,{{{ 370, 260, 370, 260, 370}
01483 ,{ 370, 260, 370, 260, 370}
01484 ,{ 370, 260, 370, 260, 370}
01485 ,{ 260, 260, 260, 260, 260}
01486 ,{ 370, 260, 370, 260, 370}
01487 }
01488 ,{{{ 370, 370, 370, 370, 300}
01489 ,{ 370, 370, 370, 370, 300}
01490 ,{ 370, 370, 370, 370, 300}
01491 ,{ 370, 370, 370, 370, 300}
01492 ,{ 300, 300, 300, 300, 300}
01493 }
01494 }
01495 }
01496 ,{{{ INF, INF, INF, INF, INF}
01497 ,{ INF, INF, INF, INF, INF}
01498 ,{ INF, INF, INF, INF, INF}
01499 ,{ INF, INF, INF, INF, INF}
01500 ,{ INF, INF, INF, INF, INF}
01501 }
01502 ,{{{ INF, INF, INF, INF, INF}
01503 ,{ INF, INF, INF, INF, INF}
01504 ,{ INF, INF, INF, INF, INF}
01505 ,{ INF, INF, INF, INF, INF}
01506 ,{ INF, INF, INF, INF, INF}
01507 }
01508 ,{{{ INF, INF, INF, INF, INF}
01509 ,{ INF, INF, INF, INF, INF}
01510 ,{ INF, INF, INF, INF, INF}
01511 ,{ INF, INF, INF, INF, INF}
01512 ,{ INF, INF, INF, INF, INF}
01513 }
01514 ,{{{ INF, INF, INF, INF, INF}
01515 ,{ INF, INF, INF, INF, INF}
01516 ,{ INF, INF, INF, INF, INF}
01517 ,{ INF, INF, INF, INF, INF}
01518 ,{ INF, INF, INF, INF, INF}
01519 }
01520 ,{{{ INF, INF, INF, INF, INF}
01521 ,{ INF, INF, INF, INF, INF}
01522 ,{ INF, INF, INF, INF, INF}
01523 ,{ INF, INF, INF, INF, INF}
01524 ,{ INF, INF, INF, INF, INF}
01525 }
01526 }
01527 ,{{{ 300, 300, 300, 300, 300}
01528 ,{ 300, 300, 300, 300, 300}
01529 ,{ 300, 300, 300, 300, 300}
01530 ,{ 300, 300, 300, 300, 300}
01531 ,{ 300, 300, 300, 300, 300}
```

```
01532     }
01533     ,{{ 300, 300, 300, 190, 300}
01534     ,{{ 300, 300, 300, 190, 300}
01535     ,{{ 300, 300, 300, 190, 300}
01536     ,{{ 190, 190, 190, 190, 190}
01537     ,{{ 300, 300, 300, 190, 300}
01538     }
01539     ,{{ 300, 300, 300, 300, 300}
01540     ,{{ 300, 300, 300, 300, 300}
01541     ,{{ 300, 300, 300, 300, 300}
01542     ,{{ 300, 300, 300, 300, 300}
01543     ,{{ 300, 300, 300, 300, 300}
01544     }
01545     ,{{ 300, 190, 300, 190, 300}
01546     ,{{ 190, 190, 190, 190, 190}
01547     ,{{ 300, 190, 300, 190, 300}
01548     ,{{ 190, 190, 190, 190, 190}
01549     ,{{ 300, 190, 300, 190, 300}
01550     }
01551     ,{{ 300, 300, 300, 300, 220}
01552     ,{{ 300, 300, 300, 300, 220}
01553     ,{{ 300, 300, 300, 300, 220}
01554     ,{{ 300, 300, 300, 300, 220}
01555     ,{{ 220, 220, 220, 220, 220}
01556     }
01557     }
01558     ,{{{ 300, 300, 300, 300, 300}
01559     ,{{ 300, 300, 300, 300, 300}
01560     ,{{ 300, 300, 300, 300, 300}
01561     ,{{ 300, 300, 300, 300, 300}
01562     ,{{ 300, 300, 300, 300, 300}
01563     }
01564     ,{{{ 300, 300, 300, 190, 300}
01565     ,{{ 300, 300, 300, 190, 300}
01566     ,{{ 300, 300, 300, 190, 300}
01567     ,{{ 190, 190, 190, 190, 190}
01568     ,{{ 300, 300, 300, 190, 300}
01569     }
01570     ,{{{ 300, 300, 300, 300, 300}
01571     ,{{ 300, 300, 300, 300, 300}
01572     ,{{ 300, 300, 300, 300, 300}
01573     ,{{ 300, 300, 300, 300, 300}
01574     ,{{ 300, 300, 300, 300, 300}
01575     }
01576     ,{{{ 300, 190, 300, 190, 300}
01577     ,{{ 300, 190, 300, 190, 300}
01578     ,{{ 300, 190, 300, 190, 300}
01579     ,{{ 190, 190, 190, 190, 190}
01580     ,{{ 300, 190, 300, 190, 300}
01581     }
01582     ,{{{ 300, 300, 300, 300, 220}
01583     ,{{ 300, 300, 300, 300, 220}
01584     ,{{ 300, 300, 300, 300, 220}
01585     ,{{ 300, 300, 300, 300, 220}
01586     ,{{ 220, 220, 220, 220, 220}
01587     }
01588     }
01589     ,{{{ 370, 370, 370, 370, 370}
01590     ,{{ 370, 370, 370, 370, 370}
01591     ,{{ 370, 370, 370, 370, 370}
01592     ,{{ 370, 370, 370, 370, 370}
01593     ,{{ 370, 370, 370, 370, 370}
01594     }
01595     ,{{{ 370, 370, 370, 260, 370}
01596     ,{{ 370, 370, 370, 260, 370}
01597     ,{{ 370, 370, 370, 260, 370}
01598     ,{{ 260, 260, 260, 260, 260}
01599     ,{{ 370, 370, 370, 260, 370}
01600     }
01601     ,{{{ 370, 370, 370, 370, 370}
01602     ,{{ 370, 370, 370, 370, 370}
01603     ,{{ 370, 370, 370, 370, 370}
01604     ,{{ 370, 370, 370, 370, 370}
01605     ,{{ 370, 370, 370, 370, 370}
01606     }
01607     ,{{{ 370, 260, 370, 260, 370}
01608     ,{{ 370, 260, 370, 260, 370}
01609     ,{{ 370, 260, 370, 260, 370}
01610     ,{{ 260, 260, 260, 260, 260}
01611     ,{{ 370, 260, 370, 260, 370}
01612     }
01613     ,{{{ 370, 370, 370, 370, 300}
01614     ,{{ 370, 370, 370, 370, 300}
01615     ,{{ 370, 370, 370, 370, 300}
01616     ,{{ 370, 370, 370, 370, 300}
01617     ,{{ 300, 300, 300, 300, 300}
01618     }
```

```
01619    }
01620    ,{{{ 370, 370, 370, 370, 370}
01621    ,{ 370, 370, 370, 370, 370}
01622    ,{ 370, 370, 370, 370, 370}
01623    ,{ 370, 370, 370, 370, 370}
01624    ,{ 370, 370, 370, 370, 370}
01625    }
01626    ,{{{ 370, 370, 370, 260, 370}
01627    ,{ 370, 370, 370, 260, 370}
01628    ,{ 370, 370, 370, 260, 370}
01629    ,{ 260, 260, 260, 260, 260}
01630    ,{ 370, 370, 370, 260, 370}
01631    }
01632    ,{{{ 370, 370, 370, 370, 370}
01633    ,{ 370, 370, 370, 370, 370}
01634    ,{ 370, 370, 370, 370, 370}
01635    ,{ 370, 370, 370, 370, 370}
01636    ,{ 370, 370, 370, 370, 370}
01637    }
01638    ,{{{ 370, 260, 370, 260, 370}
01639    ,{ 260, 260, 260, 260, 260}
01640    ,{ 370, 260, 370, 260, 370}
01641    ,{ 260, 260, 260, 260, 260}
01642    ,{ 370, 260, 370, 260, 370}
01643    }
01644    ,{{{ 370, 370, 370, 370, 300}
01645    ,{ 370, 370, 370, 370, 300}
01646    ,{ 370, 370, 370, 370, 300}
01647    ,{ 370, 370, 370, 370, 300}
01648    ,{ 300, 300, 300, 300, 300}
01649    }
01650    }
01651    ,{{{ 370, 370, 370, 370, 370}
01652    ,{ 370, 370, 370, 370, 370}
01653    ,{ 370, 370, 370, 370, 370}
01654    ,{ 370, 370, 370, 370, 370}
01655    ,{ 370, 370, 370, 370, 370}
01656    }
01657    ,{{{ 370, 370, 370, 260, 370}
01658    ,{ 370, 370, 370, 260, 370}
01659    ,{ 370, 370, 370, 260, 370}
01660    ,{ 260, 260, 260, 260, 260}
01661    ,{ 370, 370, 370, 260, 370}
01662    }
01663    ,{{{ 370, 370, 370, 370, 370}
01664    ,{ 370, 370, 370, 370, 370}
01665    ,{ 370, 370, 370, 370, 370}
01666    ,{ 370, 370, 370, 370, 370}
01667    ,{ 370, 370, 370, 370, 370}
01668    }
01669    ,{{{ 370, 260, 370, 260, 370}
01670    ,{ 370, 260, 370, 260, 370}
01671    ,{ 370, 260, 370, 260, 370}
01672    ,{ 260, 260, 260, 260, 260}
01673    ,{ 370, 260, 370, 260, 370}
01674    }
01675    ,{{{ 370, 370, 370, 370, 300}
01676    ,{ 370, 370, 370, 370, 300}
01677    ,{ 370, 370, 370, 370, 300}
01678    ,{ 370, 370, 370, 370, 300}
01679    ,{ 300, 300, 300, 300, 300}
01680    }
01681    }
01682    ,{{{ 370, 370, 370, 370, 370}
01683    ,{ 370, 370, 370, 370, 370}
01684    ,{ 370, 370, 370, 370, 370}
01685    ,{ 370, 370, 370, 370, 370}
01686    ,{ 370, 370, 370, 370, 370}
01687    }
01688    ,{{{ 370, 370, 370, 260, 370}
01689    ,{ 370, 370, 370, 260, 370}
01690    ,{ 370, 370, 370, 260, 370}
01691    ,{ 260, 260, 260, 260, 260}
01692    ,{ 370, 370, 370, 260, 370}
01693    }
01694    ,{{{ 370, 370, 370, 370, 370}
01695    ,{ 370, 370, 370, 370, 370}
01696    ,{ 370, 370, 370, 370, 370}
01697    ,{ 370, 370, 370, 370, 370}
01698    ,{ 370, 370, 370, 370, 370}
01699    }
01700    ,{{{ 370, 260, 370, 260, 370}
01701    ,{ 260, 260, 260, 260, 260}
01702    ,{ 370, 260, 370, 260, 370}
01703    ,{ 260, 260, 260, 260, 260}
01704    ,{ 370, 260, 370, 260, 370}
01705    }
```

```
01706 ,{{ 370, 370, 370, 370, 300}
01707 ,{ 370, 370, 370, 370, 300}
01708 ,{ 370, 370, 370, 370, 300}
01709 ,{ 370, 370, 370, 370, 300}
01710 ,{ 300, 300, 300, 300, 300}
01711 }
01712 }
01713 ,{{{ 370, 370, 370, 370, 370}
01714 ,{ 370, 370, 370, 370, 370}
01715 ,{ 370, 370, 370, 370, 370}
01716 ,{ 370, 370, 370, 370, 370}
01717 ,{ 370, 370, 370, 370, 370}
01718 }
01719 ,{{{ 370, 370, 370, 260, 370}
01720 ,{ 370, 370, 370, 260, 370}
01721 ,{ 370, 370, 370, 260, 370}
01722 ,{ 260, 260, 260, 260, 260}
01723 ,{ 370, 370, 370, 260, 370}
01724 }
01725 ,{{{ 370, 370, 370, 370, 370}
01726 ,{ 370, 370, 370, 370, 370}
01727 ,{ 370, 370, 370, 370, 370}
01728 ,{ 370, 370, 370, 370, 370}
01729 ,{ 370, 370, 370, 370, 370}
01730 }
01731 ,{{{ 370, 260, 370, 260, 370}
01732 ,{ 370, 260, 370, 260, 370}
01733 ,{ 370, 260, 370, 260, 370}
01734 ,{ 260, 260, 260, 260, 260}
01735 ,{ 370, 260, 370, 260, 370}
01736 }
01737 ,{{{ 370, 370, 370, 370, 300}
01738 ,{ 370, 370, 370, 370, 300}
01739 ,{ 370, 370, 370, 370, 300}
01740 ,{ 370, 370, 370, 370, 300}
01741 ,{ 300, 300, 300, 300, 300}
01742 }
01743 }
01744 }
01745 ,{{{ INF, INF, INF, INF, INF}
01746 ,{ INF, INF, INF, INF, INF}
01747 ,{ INF, INF, INF, INF, INF}
01748 ,{ INF, INF, INF, INF, INF}
01749 ,{ INF, INF, INF, INF, INF}
01750 }
01751 ,{{{ INF, INF, INF, INF, INF}
01752 ,{ INF, INF, INF, INF, INF}
01753 ,{ INF, INF, INF, INF, INF}
01754 ,{ INF, INF, INF, INF, INF}
01755 ,{ INF, INF, INF, INF, INF}
01756 }
01757 ,{{{ INF, INF, INF, INF, INF}
01758 ,{ INF, INF, INF, INF, INF}
01759 ,{ INF, INF, INF, INF, INF}
01760 ,{ INF, INF, INF, INF, INF}
01761 ,{ INF, INF, INF, INF, INF}
01762 }
01763 ,{{{ INF, INF, INF, INF, INF}
01764 ,{ INF, INF, INF, INF, INF}
01765 ,{ INF, INF, INF, INF, INF}
01766 ,{ INF, INF, INF, INF, INF}
01767 ,{ INF, INF, INF, INF, INF}
01768 }
01769 ,{{{ INF, INF, INF, INF, INF}
01770 ,{ INF, INF, INF, INF, INF}
01771 ,{ INF, INF, INF, INF, INF}
01772 ,{ INF, INF, INF, INF, INF}
01773 ,{ INF, INF, INF, INF, INF}
01774 }
01775 }
01776 ,{{{ 300, 300, 300, 300, 300}
01777 ,{ 300, 300, 300, 300, 300}
01778 ,{ 300, 300, 300, 300, 300}
01779 ,{ 300, 300, 300, 300, 300}
01780 ,{ 300, 300, 300, 300, 300}
01781 }
01782 ,{{{ 300, 300, 300, 300, 300}
01783 ,{ 300, 300, 300, 300, 300}
01784 ,{ 300, 300, 300, 300, 300}
01785 ,{ 190, 190, 190, 190, 190}
01786 ,{ 300, 300, 300, 300, 300}
01787 }
01788 ,{{{ 300, 300, 300, 300, 300}
01789 ,{ 300, 300, 300, 300, 300}
01790 ,{ 300, 300, 300, 300, 300}
01791 ,{ 300, 300, 300, 300, 300}
01792 ,{ 300, 300, 300, 300, 300}
```

```
01793     }
01794     ,{{ 300, 190, 300, 190, 300}
01795     ,{{ 190, 190, 190, 190, 190}
01796     ,{{ 300, 190, 300, 190, 300}
01797     ,{{ 190, 190, 190, 190, 190}
01798     ,{{ 300, 190, 300, 190, 300}
01799     }
01800     ,{{ 300, 300, 300, 300, 220}
01801     ,{{ 300, 300, 300, 300, 220}
01802     ,{{ 300, 300, 300, 300, 220}
01803     ,{{ 300, 300, 300, 300, 220}
01804     ,{{ 220, 220, 220, 220, 220}
01805     }
01806     }
01807     ,{{{ 300, 300, 300, 300, 300}
01808     ,{{ 300, 300, 300, 300, 300}
01809     ,{{ 300, 300, 300, 300, 300}
01810     ,{{ 300, 300, 300, 300, 300}
01811     ,{{ 300, 300, 300, 300, 300}
01812     }
01813     ,{{ 300, 300, 300, 300, 300}
01814     ,{{ 300, 300, 300, 300, 300}
01815     ,{{ 300, 300, 300, 300, 300}
01816     ,{{ 190, 190, 190, 190, 190}
01817     ,{{ 300, 300, 300, 300, 300}
01818     }
01819     ,{{{ 300, 300, 300, 300, 300}
01820     ,{{ 300, 300, 300, 300, 300}
01821     ,{{ 300, 300, 300, 300, 300}
01822     ,{{ 300, 300, 300, 300, 300}
01823     ,{{ 300, 300, 300, 300, 300}
01824     }
01825     ,{{{ 300, 190, 300, 190, 300}
01826     ,{{ 300, 190, 300, 190, 300}
01827     ,{{ 300, 190, 300, 190, 300}
01828     ,{{ 190, 190, 190, 190, 190}
01829     ,{{ 300, 190, 300, 190, 300}
01830     }
01831     ,{{{ 300, 300, 300, 300, 220}
01832     ,{{ 300, 300, 300, 300, 220}
01833     ,{{ 300, 300, 300, 300, 220}
01834     ,{{ 300, 300, 300, 300, 220}
01835     ,{{ 220, 220, 220, 220, 220}
01836     }
01837     }
01838     ,{{{ 370, 370, 370, 370, 370}
01839     ,{{ 370, 370, 370, 370, 370}
01840     ,{{ 370, 370, 370, 370, 370}
01841     ,{{ 370, 370, 370, 370, 370}
01842     ,{{ 370, 370, 370, 370, 370}
01843     }
01844     ,{{{ 370, 370, 370, 370, 370}
01845     ,{{ 370, 370, 370, 370, 370}
01846     ,{{ 370, 370, 370, 370, 370}
01847     ,{{ 260, 260, 260, 260, 260}
01848     ,{{ 370, 370, 370, 370, 370}
01849     }
01850     ,{{{ 370, 370, 370, 370, 370}
01851     ,{{ 370, 370, 370, 370, 370}
01852     ,{{ 370, 370, 370, 370, 370}
01853     ,{{ 370, 370, 370, 370, 370}
01854     ,{{ 370, 370, 370, 370, 370}
01855     }
01856     ,{{{ 370, 260, 370, 260, 370}
01857     ,{{ 370, 260, 370, 260, 370}
01858     ,{{ 370, 260, 370, 260, 370}
01859     ,{{ 260, 260, 260, 260, 260}
01860     ,{{ 370, 260, 370, 260, 370}
01861     }
01862     ,{{{ 370, 370, 370, 370, 300}
01863     ,{{ 370, 370, 370, 370, 300}
01864     ,{{ 370, 370, 370, 370, 300}
01865     ,{{ 370, 370, 370, 370, 300}
01866     ,{{ 300, 300, 300, 300, 300}
01867     }
01868     }
01869     ,{{{ 370, 370, 370, 370, 370}
01870     ,{{ 370, 370, 370, 370, 370}
01871     ,{{ 370, 370, 370, 370, 370}
01872     ,{{ 370, 370, 370, 370, 370}
01873     ,{{ 370, 370, 370, 370, 370}
01874     }
01875     ,{{{ 370, 370, 370, 370, 370}
01876     ,{{ 370, 370, 370, 370, 370}
01877     ,{{ 370, 370, 370, 370, 370}
01878     ,{{ 260, 260, 260, 260, 260}
01879     ,{{ 370, 370, 370, 370, 370}
```



```
01880     }
01881     ,{{ 370, 370, 370, 370, 370}
01882     ,{ 370, 370, 370, 370, 370}
01883     ,{ 370, 370, 370, 370, 370}
01884     ,{ 370, 370, 370, 370, 370}
01885     ,{ 370, 370, 370, 370, 370}
01886     }
01887     ,{{ 370, 260, 370, 260, 370}
01888     ,{ 260, 260, 260, 260, 260}
01889     ,{ 370, 260, 370, 260, 370}
01890     ,{ 260, 260, 260, 260, 260}
01891     ,{ 370, 260, 370, 260, 370}
01892     }
01893     ,{{ 370, 370, 370, 370, 300}
01894     ,{ 370, 370, 370, 370, 300}
01895     ,{ 370, 370, 370, 370, 300}
01896     ,{ 370, 370, 370, 370, 300}
01897     ,{ 300, 300, 300, 300, 300}
01898     }
01899     }
01900     ,{{{ 370, 370, 370, 370, 370}
01901     ,{ 370, 370, 370, 370, 370}
01902     ,{ 370, 370, 370, 370, 370}
01903     ,{ 370, 370, 370, 370, 370}
01904     ,{ 370, 370, 370, 370, 370}
01905     }
01906     ,{{{ 370, 370, 370, 370, 370}
01907     ,{ 370, 370, 370, 370, 370}
01908     ,{ 370, 370, 370, 370, 370}
01909     ,{ 260, 260, 260, 260, 260}
01910     ,{ 370, 370, 370, 370, 370}
01911     }
01912     ,{{{ 370, 370, 370, 370, 370}
01913     ,{ 370, 370, 370, 370, 370}
01914     ,{ 370, 370, 370, 370, 370}
01915     ,{ 370, 370, 370, 370, 370}
01916     ,{ 370, 370, 370, 370, 370}
01917     }
01918     ,{{{ 370, 260, 370, 260, 370}
01919     ,{ 370, 260, 370, 260, 370}
01920     ,{ 370, 260, 370, 260, 370}
01921     ,{ 260, 260, 260, 260, 260}
01922     ,{ 370, 260, 370, 260, 370}
01923     }
01924     ,{{{ 370, 370, 370, 370, 300}
01925     ,{ 370, 370, 370, 370, 300}
01926     ,{ 370, 370, 370, 370, 300}
01927     ,{ 370, 370, 370, 370, 300}
01928     ,{ 300, 300, 300, 300, 300}
01929     }
01930     }
01931     ,{{{ 370, 370, 370, 370, 370}
01932     ,{ 370, 370, 370, 370, 370}
01933     ,{ 370, 370, 370, 370, 370}
01934     ,{ 370, 370, 370, 370, 370}
01935     ,{ 370, 370, 370, 370, 370}
01936     }
01937     ,{{{ 370, 370, 370, 370, 370}
01938     ,{ 370, 370, 370, 370, 370}
01939     ,{ 370, 370, 370, 370, 370}
01940     ,{ 260, 260, 260, 260, 260}
01941     ,{ 370, 370, 370, 370, 370}
01942     }
01943     ,{{{ 370, 370, 370, 370, 370}
01944     ,{ 370, 370, 370, 370, 370}
01945     ,{ 370, 370, 370, 370, 370}
01946     ,{ 370, 370, 370, 370, 370}
01947     ,{ 370, 370, 370, 370, 370}
01948     }
01949     ,{{{ 370, 260, 370, 260, 370}
01950     ,{ 260, 260, 260, 260, 260}
01951     ,{ 370, 260, 370, 260, 370}
01952     ,{ 260, 260, 260, 260, 260}
01953     ,{ 370, 260, 370, 260, 370}
01954     }
01955     ,{{{ 370, 370, 370, 370, 300}
01956     ,{ 370, 370, 370, 370, 300}
01957     ,{ 370, 370, 370, 370, 300}
01958     ,{ 370, 370, 370, 370, 300}
01959     ,{ 300, 300, 300, 300, 300}
01960     }
01961     }
01962     ,{{{ 370, 370, 370, 370, 370}
01963     ,{ 370, 370, 370, 370, 370}
01964     ,{ 370, 370, 370, 370, 370}
01965     ,{ 370, 370, 370, 370, 370}
01966     ,{ 370, 370, 370, 370, 370}
```

```

01967     }
01968     ,{{ 370, 370, 370, 370, 370}
01969     ,{{ 370, 370, 370, 370, 370}
01970     ,{{ 370, 370, 370, 370, 370}
01971     ,{{ 260, 260, 260, 260, 260}
01972     ,{{ 370, 370, 370, 370, 370}
01973     }
01974     ,{{ 370, 370, 370, 370, 370}
01975     ,{{ 370, 370, 370, 370, 370}
01976     ,{{ 370, 370, 370, 370, 370}
01977     ,{{ 370, 370, 370, 370, 370}
01978     ,{{ 370, 370, 370, 370, 370}
01979     }
01980     ,{{ 370, 260, 370, 260, 370}
01981     ,{{ 370, 260, 370, 260, 370}
01982     ,{{ 370, 260, 370, 260, 370}
01983     ,{{ 260, 260, 260, 260, 260}
01984     ,{{ 370, 260, 370, 260, 370}
01985     }
01986     ,{{ 370, 370, 370, 370, 300}
01987     ,{{ 370, 370, 370, 370, 300}
01988     ,{{ 370, 370, 370, 370, 300}
01989     ,{{ 370, 370, 370, 370, 300}
01990     ,{{ 300, 300, 300, 300, 300}
01991     }
01992     }
01993     };;

```

18.173 intl21dH.h

```

00001 PUBLIC int int21_dH[NBPAIRS+1][NBPAIRS+1][5][5][5] =
00002 {{{{{ INF, INF, INF, INF, INF}
00003     ,{{ INF, INF, INF, INF, INF}
00004     ,{{ INF, INF, INF, INF, INF}
00005     ,{{ INF, INF, INF, INF, INF}
00006     ,{{ INF, INF, INF, INF, INF}
00007     }
00008     ,{{ INF, INF, INF, INF, INF}
00009     ,{{ INF, INF, INF, INF, INF}
00010     ,{{ INF, INF, INF, INF, INF}
00011     ,{{ INF, INF, INF, INF, INF}
00012     ,{{ INF, INF, INF, INF, INF}
00013     }
00014     ,{{ INF, INF, INF, INF, INF}
00015     ,{{ INF, INF, INF, INF, INF}
00016     ,{{ INF, INF, INF, INF, INF}
00017     ,{{ INF, INF, INF, INF, INF}
00018     ,{{ INF, INF, INF, INF, INF}
00019     }
00020     ,{{ INF, INF, INF, INF, INF}
00021     ,{{ INF, INF, INF, INF, INF}
00022     ,{{ INF, INF, INF, INF, INF}
00023     ,{{ INF, INF, INF, INF, INF}
00024     ,{{ INF, INF, INF, INF, INF}
00025     }
00026     ,{{ INF, INF, INF, INF, INF}
00027     ,{{ INF, INF, INF, INF, INF}
00028     ,{{ INF, INF, INF, INF, INF}
00029     ,{{ INF, INF, INF, INF, INF}
00030     ,{{ INF, INF, INF, INF, INF}
00031     }
00032     }
00033     ,{{{ INF, INF, INF, INF, INF}
00034     ,{{ INF, INF, INF, INF, INF}
00035     ,{{ INF, INF, INF, INF, INF}
00036     ,{{ INF, INF, INF, INF, INF}
00037     ,{{ INF, INF, INF, INF, INF}
00038     }
00039     ,{{ INF, INF, INF, INF, INF}
00040     ,{{ INF, INF, INF, INF, INF}
00041     ,{{ INF, INF, INF, INF, INF}
00042     ,{{ INF, INF, INF, INF, INF}
00043     ,{{ INF, INF, INF, INF, INF}
00044     }
00045     ,{{ INF, INF, INF, INF, INF}
00046     ,{{ INF, INF, INF, INF, INF}
00047     ,{{ INF, INF, INF, INF, INF}
00048     ,{{ INF, INF, INF, INF, INF}
00049     ,{{ INF, INF, INF, INF, INF}
00050     }
00051     ,{{ INF, INF, INF, INF, INF}
00052     ,{{ INF, INF, INF, INF, INF}
00053     ,{{ INF, INF, INF, INF, INF}
00054     ,{{ INF, INF, INF, INF, INF}
00055     ,{{ INF, INF, INF, INF, INF}

```

```
00056     }
00057     ,{{ INF, INF, INF, INF, INF}
00058     ,{ INF, INF, INF, INF, INF}
00059     ,{ INF, INF, INF, INF, INF}
00060     ,{ INF, INF, INF, INF, INF}
00061     ,{ INF, INF, INF, INF, INF}
00062     }
00063     }
00064     ,{{{ INF, INF, INF, INF, INF}
00065     ,{ INF, INF, INF, INF, INF}
00066     ,{ INF, INF, INF, INF, INF}
00067     ,{ INF, INF, INF, INF, INF}
00068     ,{ INF, INF, INF, INF, INF}
00069     }
00070     ,{{{ INF, INF, INF, INF, INF}
00071     ,{ INF, INF, INF, INF, INF}
00072     ,{ INF, INF, INF, INF, INF}
00073     ,{ INF, INF, INF, INF, INF}
00074     ,{ INF, INF, INF, INF, INF}
00075     }
00076     ,{{{ INF, INF, INF, INF, INF}
00077     ,{ INF, INF, INF, INF, INF}
00078     ,{ INF, INF, INF, INF, INF}
00079     ,{ INF, INF, INF, INF, INF}
00080     ,{ INF, INF, INF, INF, INF}
00081     }
00082     ,{{{ INF, INF, INF, INF, INF}
00083     ,{ INF, INF, INF, INF, INF}
00084     ,{ INF, INF, INF, INF, INF}
00085     ,{ INF, INF, INF, INF, INF}
00086     ,{ INF, INF, INF, INF, INF}
00087     }
00088     ,{{{ INF, INF, INF, INF, INF}
00089     ,{ INF, INF, INF, INF, INF}
00090     ,{ INF, INF, INF, INF, INF}
00091     ,{ INF, INF, INF, INF, INF}
00092     ,{ INF, INF, INF, INF, INF}
00093     }
00094     }
00095     ,{{{ INF, INF, INF, INF, INF}
00096     ,{ INF, INF, INF, INF, INF}
00097     ,{ INF, INF, INF, INF, INF}
00098     ,{ INF, INF, INF, INF, INF}
00099     ,{ INF, INF, INF, INF, INF}
00100     }
00101     ,{{{ INF, INF, INF, INF, INF}
00102     ,{ INF, INF, INF, INF, INF}
00103     ,{ INF, INF, INF, INF, INF}
00104     ,{ INF, INF, INF, INF, INF}
00105     ,{ INF, INF, INF, INF, INF}
00106     }
00107     ,{{{ INF, INF, INF, INF, INF}
00108     ,{ INF, INF, INF, INF, INF}
00109     ,{ INF, INF, INF, INF, INF}
00110     ,{ INF, INF, INF, INF, INF}
00111     ,{ INF, INF, INF, INF, INF}
00112     }
00113     ,{{{ INF, INF, INF, INF, INF}
00114     ,{ INF, INF, INF, INF, INF}
00115     ,{ INF, INF, INF, INF, INF}
00116     ,{ INF, INF, INF, INF, INF}
00117     ,{ INF, INF, INF, INF, INF}
00118     }
00119     ,{{{ INF, INF, INF, INF, INF}
00120     ,{ INF, INF, INF, INF, INF}
00121     ,{ INF, INF, INF, INF, INF}
00122     ,{ INF, INF, INF, INF, INF}
00123     ,{ INF, INF, INF, INF, INF}
00124     }
00125     }
00126     ,{{{ INF, INF, INF, INF, INF}
00127     ,{ INF, INF, INF, INF, INF}
00128     ,{ INF, INF, INF, INF, INF}
00129     ,{ INF, INF, INF, INF, INF}
00130     ,{ INF, INF, INF, INF, INF}
00131     }
00132     ,{{{ INF, INF, INF, INF, INF}
00133     ,{ INF, INF, INF, INF, INF}
00134     ,{ INF, INF, INF, INF, INF}
00135     ,{ INF, INF, INF, INF, INF}
00136     ,{ INF, INF, INF, INF, INF}
00137     }
00138     ,{{{ INF, INF, INF, INF, INF}
00139     ,{ INF, INF, INF, INF, INF}
00140     ,{ INF, INF, INF, INF, INF}
00141     ,{ INF, INF, INF, INF, INF}
00142     ,{ INF, INF, INF, INF, INF}
```

```
00143     }
00144     ,{{   INF,   INF,   INF,   INF,   INF}
00145     ,{{   INF,   INF,   INF,   INF,   INF}
00146     ,{{   INF,   INF,   INF,   INF,   INF}
00147     ,{{   INF,   INF,   INF,   INF,   INF}
00148     ,{{   INF,   INF,   INF,   INF,   INF}
00149     }
00150     ,{{   INF,   INF,   INF,   INF,   INF}
00151     ,{{   INF,   INF,   INF,   INF,   INF}
00152     ,{{   INF,   INF,   INF,   INF,   INF}
00153     ,{{   INF,   INF,   INF,   INF,   INF}
00154     ,{{   INF,   INF,   INF,   INF,   INF}
00155     }
00156     }
00157     ,{{{   INF,   INF,   INF,   INF,   INF}
00158     ,{{   INF,   INF,   INF,   INF,   INF}
00159     ,{{   INF,   INF,   INF,   INF,   INF}
00160     ,{{   INF,   INF,   INF,   INF,   INF}
00161     ,{{   INF,   INF,   INF,   INF,   INF}
00162     }
00163     ,{{{   INF,   INF,   INF,   INF,   INF}
00164     ,{{   INF,   INF,   INF,   INF,   INF}
00165     ,{{   INF,   INF,   INF,   INF,   INF}
00166     ,{{   INF,   INF,   INF,   INF,   INF}
00167     ,{{   INF,   INF,   INF,   INF,   INF}
00168     }
00169     ,{{{   INF,   INF,   INF,   INF,   INF}
00170     ,{{   INF,   INF,   INF,   INF,   INF}
00171     ,{{   INF,   INF,   INF,   INF,   INF}
00172     ,{{   INF,   INF,   INF,   INF,   INF}
00173     ,{{   INF,   INF,   INF,   INF,   INF}
00174     }
00175     ,{{{   INF,   INF,   INF,   INF,   INF}
00176     ,{{   INF,   INF,   INF,   INF,   INF}
00177     ,{{   INF,   INF,   INF,   INF,   INF}
00178     ,{{   INF,   INF,   INF,   INF,   INF}
00179     ,{{   INF,   INF,   INF,   INF,   INF}
00180     }
00181     ,{{{   INF,   INF,   INF,   INF,   INF}
00182     ,{{   INF,   INF,   INF,   INF,   INF}
00183     ,{{   INF,   INF,   INF,   INF,   INF}
00184     ,{{   INF,   INF,   INF,   INF,   INF}
00185     ,{{   INF,   INF,   INF,   INF,   INF}
00186     }
00187     }
00188     ,{{{   INF,   INF,   INF,   INF,   INF}
00189     ,{{   INF,   INF,   INF,   INF,   INF}
00190     ,{{   INF,   INF,   INF,   INF,   INF}
00191     ,{{   INF,   INF,   INF,   INF,   INF}
00192     ,{{   INF,   INF,   INF,   INF,   INF}
00193     }
00194     ,{{{   INF,   INF,   INF,   INF,   INF}
00195     ,{{   INF,   INF,   INF,   INF,   INF}
00196     ,{{   INF,   INF,   INF,   INF,   INF}
00197     ,{{   INF,   INF,   INF,   INF,   INF}
00198     ,{{   INF,   INF,   INF,   INF,   INF}
00199     }
00200     ,{{{   INF,   INF,   INF,   INF,   INF}
00201     ,{{   INF,   INF,   INF,   INF,   INF}
00202     ,{{   INF,   INF,   INF,   INF,   INF}
00203     ,{{   INF,   INF,   INF,   INF,   INF}
00204     ,{{   INF,   INF,   INF,   INF,   INF}
00205     }
00206     ,{{{   INF,   INF,   INF,   INF,   INF}
00207     ,{{   INF,   INF,   INF,   INF,   INF}
00208     ,{{   INF,   INF,   INF,   INF,   INF}
00209     ,{{   INF,   INF,   INF,   INF,   INF}
00210     ,{{   INF,   INF,   INF,   INF,   INF}
00211     }
00212     ,{{{   INF,   INF,   INF,   INF,   INF}
00213     ,{{   INF,   INF,   INF,   INF,   INF}
00214     ,{{   INF,   INF,   INF,   INF,   INF}
00215     ,{{   INF,   INF,   INF,   INF,   INF}
00216     ,{{   INF,   INF,   INF,   INF,   INF}
00217     }
00218     }
00219     ,{{{   INF,   INF,   INF,   INF,   INF}
00220     ,{{   INF,   INF,   INF,   INF,   INF}
00221     ,{{   INF,   INF,   INF,   INF,   INF}
00222     ,{{   INF,   INF,   INF,   INF,   INF}
00223     ,{{   INF,   INF,   INF,   INF,   INF}
00224     }
00225     ,{{{   INF,   INF,   INF,   INF,   INF}
00226     ,{{   INF,   INF,   INF,   INF,   INF}
00227     ,{{   INF,   INF,   INF,   INF,   INF}
00228     ,{{   INF,   INF,   INF,   INF,   INF}
00229     ,{{   INF,   INF,   INF,   INF,   INF}
```

```
00230     }
00231     ,{{ INF, INF, INF, INF, INF}
00232     ,{ INF, INF, INF, INF, INF}
00233     ,{ INF, INF, INF, INF, INF}
00234     ,{ INF, INF, INF, INF, INF}
00235     ,{ INF, INF, INF, INF, INF}
00236     }
00237     ,{{ INF, INF, INF, INF, INF}
00238     ,{ INF, INF, INF, INF, INF}
00239     ,{ INF, INF, INF, INF, INF}
00240     ,{ INF, INF, INF, INF, INF}
00241     ,{ INF, INF, INF, INF, INF}
00242     }
00243     ,{{ INF, INF, INF, INF, INF}
00244     ,{ INF, INF, INF, INF, INF}
00245     ,{ INF, INF, INF, INF, INF}
00246     ,{ INF, INF, INF, INF, INF}
00247     ,{ INF, INF, INF, INF, INF}
00248     }
00249     }
00250     }
00251     ,{{{ INF, INF, INF, INF, INF}
00252     ,{ INF, INF, INF, INF, INF}
00253     ,{ INF, INF, INF, INF, INF}
00254     ,{ INF, INF, INF, INF, INF}
00255     ,{ INF, INF, INF, INF, INF}
00256     }
00257     ,{{ INF, INF, INF, INF, INF}
00258     ,{ INF, INF, INF, INF, INF}
00259     ,{ INF, INF, INF, INF, INF}
00260     ,{ INF, INF, INF, INF, INF}
00261     ,{ INF, INF, INF, INF, INF}
00262     }
00263     ,{{ INF, INF, INF, INF, INF}
00264     ,{ INF, INF, INF, INF, INF}
00265     ,{ INF, INF, INF, INF, INF}
00266     ,{ INF, INF, INF, INF, INF}
00267     ,{ INF, INF, INF, INF, INF}
00268     }
00269     ,{{ INF, INF, INF, INF, INF}
00270     ,{ INF, INF, INF, INF, INF}
00271     ,{ INF, INF, INF, INF, INF}
00272     ,{ INF, INF, INF, INF, INF}
00273     ,{ INF, INF, INF, INF, INF}
00274     }
00275     ,{{ INF, INF, INF, INF, INF}
00276     ,{ INF, INF, INF, INF, INF}
00277     ,{ INF, INF, INF, INF, INF}
00278     ,{ INF, INF, INF, INF, INF}
00279     ,{ INF, INF, INF, INF, INF}
00280     }
00281     }
00282     ,{{{ 350, 350, 350, 350, 350}
00283     ,{ 350, 350, 350, 350, 350}
00284     ,{ 350, 350, 350, 350, 350}
00285     ,{ 350, 350, 350, 350, 350}
00286     ,{ 350, 350, 350, 350, 350}
00287     }
00288     ,{{ 350, 350, 350, -230, 350}
00289     ,{ 350, 350, 350, -230, 350}
00290     ,{ 350, 350, 350, -230, 350}
00291     ,{ -230, -230, -230, -230, -230}
00292     ,{ 350, 350, 350, -230, 350}
00293     }
00294     ,{{{ 350, 350, 350, 350, 350}
00295     ,{ 350, 350, 350, 350, 350}
00296     ,{ 350, 350, 350, 350, 350}
00297     ,{ 350, 350, 350, 350, 350}
00298     ,{ 350, 350, 350, 350, 350}
00299     }
00300     ,{{{ 350, -230, 350, -230, 350}
00301     ,{ -230, -230, -230, -230, -230}
00302     ,{ 350, -230, 350, -230, 350}
00303     ,{ -230, -230, -230, -230, -230}
00304     ,{ 350, -230, 350, -230, 350}
00305     }
00306     ,{{{ 350, 350, 350, 350, -670}
00307     ,{ 350, 350, 350, 350, -670}
00308     ,{ 350, 350, 350, 350, -670}
00309     ,{ 350, 350, 350, 350, -670}
00310     ,{ -670, -670, -670, -670, -670}
00311     }
00312     }
00313     ,{{{ 780, 640, 780, 350, 350}
00314     ,{ 350, 350, 350, 350, 350}
00315     ,{ 780, 350, 780, 350, 350}
00316     ,{ 350, 350, 350, 350, 350}
```

```
00317 , { 640, 640, 350, 350, 350 }
00318 }
00319 , { { 350, 350, 350, 250, 350 }
00320 , { 350, 260, 350, 250, 350 }
00321 , { 350, 350, -250, -230, 350 }
00322 , { -230, -230, -230, -230, -230 }
00323 , { 350, 350, 350, -230, 350 }
00324 }
00325 , { { 780, 640, 780, 350, 350 }
00326 , { 350, 160, 350, 350, 350 }
00327 , { 780, 350, 780, 350, 350 }
00328 , { 350, 350, 350, 350, 350 }
00329 , { 640, 640, 350, 350, 350 }
00330 }
00331 , { { 350, -160, 350, -230, 350 }
00332 , { 350, -160, 350, -410, 350 }
00333 , { 350, -230, 350, -230, 350 }
00334 , { -230, -310, -230, -230, -230 }
00335 , { 350, -230, 350, -230, 350 }
00336 }
00337 , { { 580, 350, 580, 350, -580 }
00338 , { 350, 350, 350, 350, -670 }
00339 , { 580, 350, 580, 350, -580 }
00340 , { 350, 350, 350, 350, -670 }
00341 , { -670, -670, -690, -670, -700 }
00342 }
00343 }
00344 , { { { 850, 850, 850, 850, 850 }
00345 , { 850, 850, 850, 850, 850 }
00346 , { 850, 850, 850, 850, 850 }
00347 , { 850, 850, 850, 850, 850 }
00348 , { 850, 850, 850, 850, 850 }
00349 }
00350 , { { 850, 850, 850, 280, 850 }
00351 , { 850, 850, 850, 280, 850 }
00352 , { 850, 850, 850, 280, 850 }
00353 , { 280, 280, 280, 280, 280 }
00354 , { 850, 850, 850, 280, 850 }
00355 }
00356 , { { 850, 850, 850, 850, 850 }
00357 , { 850, 850, 850, 850, 850 }
00358 , { 850, 850, 850, 850, 850 }
00359 , { 850, 850, 850, 850, 850 }
00360 , { 850, 850, 850, 850, 850 }
00361 }
00362 , { { 850, 280, 850, 280, 850 }
00363 , { 850, 280, 850, 280, 850 }
00364 , { 850, 280, 850, 280, 850 }
00365 , { 280, 280, 280, 280, 280 }
00366 , { 850, 280, 850, 280, 850 }
00367 }
00368 , { { 850, 850, 850, 850, -160 }
00369 , { 850, 850, 850, 850, -160 }
00370 , { 850, 850, 850, 850, -160 }
00371 , { 850, 850, 850, 850, -160 }
00372 , { -160, -160, -160, -160, -160 }
00373 }
00374 }
00375 , { { { 850, 850, 850, 850, 850 }
00376 , { 850, 850, 850, 850, 850 }
00377 , { 850, 850, 850, 850, 850 }
00378 , { 850, 850, 850, 850, 850 }
00379 , { 850, 850, 850, 850, 850 }
00380 }
00381 , { { 850, 850, 850, 280, 850 }
00382 , { 850, 850, 850, 280, 850 }
00383 , { 850, 850, 850, 280, 850 }
00384 , { 280, 280, 280, 280, 280 }
00385 , { 850, 850, 850, 280, 850 }
00386 }
00387 , { { 850, 850, 850, 850, 850 }
00388 , { 850, 850, 850, 850, 850 }
00389 , { 850, 850, 850, 850, 850 }
00390 , { 850, 850, 850, 850, 850 }
00391 , { 850, 850, 850, 850, 850 }
00392 }
00393 , { { 850, 280, 850, 280, 850 }
00394 , { 280, 280, 280, 280, 280 }
00395 , { 850, 280, 850, 280, 850 }
00396 , { 280, 280, 280, 280, 280 }
00397 , { 850, 280, 850, 280, 850 }
00398 }
00399 , { { 850, 850, 850, 850, -160 }
00400 , { 850, 850, 850, 850, -160 }
00401 , { 850, 850, 850, 850, -160 }
00402 , { 850, 850, 850, 850, -160 }
00403 , { -160, -160, -160, -160, -160 }
```

```
00404     }
00405     }
00406     ,{{ 850, 850, 850, 850, 850}
00407     ,{ 850, 850, 850, 850, 850}
00408     ,{ 850, 850, 850, 850, 850}
00409     ,{ 850, 850, 850, 850, 850}
00410     ,{ 850, 850, 850, 850, 850}
00411     }
00412     ,{{ 850, 850, 850, 280, 850}
00413     ,{ 850, 850, 850, 280, 850}
00414     ,{ 850, 850, 850, 280, 850}
00415     ,{ 280, 280, 280, 280, 280}
00416     ,{ 850, 850, 850, 280, 850}
00417     }
00418     ,{{ 850, 850, 850, 850, 850}
00419     ,{ 850, 850, 850, 850, 850}
00420     ,{ 850, 850, 850, 850, 850}
00421     ,{ 850, 850, 850, 850, 850}
00422     ,{ 850, 850, 850, 850, 850}
00423     }
00424     ,{{ 850, 280, 850, 280, 850}
00425     ,{ 850, 280, 850, 280, 850}
00426     ,{ 850, 280, 850, 280, 850}
00427     ,{ 280, 280, 280, 280, 280}
00428     ,{ 850, 280, 850, 280, 850}
00429     }
00430     ,{{ 850, 850, 850, 850, -160}
00431     ,{ 850, 850, 850, 850, -160}
00432     ,{ 850, 850, 850, 850, -160}
00433     ,{ 850, 850, 850, 850, -160}
00434     ,{ -160, -160, -160, -160, -160}
00435     }
00436     }
00437     ,{{{ 850, 850, 850, 850, 850}
00438     ,{ 850, 850, 850, 850, 850}
00439     ,{ 850, 850, 850, 850, 850}
00440     ,{ 850, 850, 850, 850, 850}
00441     ,{ 850, 850, 850, 850, 850}
00442     }
00443     ,{{ 850, 850, 850, 280, 850}
00444     ,{ 850, 850, 850, 280, 850}
00445     ,{ 850, 850, 850, 280, 850}
00446     ,{ 280, 280, 280, 280, 280}
00447     ,{ 850, 850, 850, 280, 850}
00448     }
00449     ,{{ 850, 850, 850, 850, 850}
00450     ,{ 850, 850, 850, 850, 850}
00451     ,{ 850, 850, 850, 850, 850}
00452     ,{ 850, 850, 850, 850, 850}
00453     ,{ 850, 850, 850, 850, 850}
00454     }
00455     ,{{ 850, 280, 850, 280, 850}
00456     ,{ 280, 280, 280, 280, 280}
00457     ,{ 850, 280, 850, 280, 850}
00458     ,{ 280, 280, 280, 280, 280}
00459     ,{ 850, 280, 850, 280, 850}
00460     }
00461     ,{{ 850, 850, 850, 850, -160}
00462     ,{ 850, 850, 850, 850, -160}
00463     ,{ 850, 850, 850, 850, -160}
00464     ,{ 850, 850, 850, 850, -160}
00465     ,{ -160, -160, -160, -160, -160}
00466     }
00467     }
00468     ,{{{ 850, 850, 850, 850, 850}
00469     ,{ 850, 850, 850, 850, 850}
00470     ,{ 850, 850, 850, 850, 850}
00471     ,{ 850, 850, 850, 850, 850}
00472     ,{ 850, 850, 850, 850, 850}
00473     }
00474     ,{{ 850, 850, 850, 280, 850}
00475     ,{ 850, 850, 850, 280, 850}
00476     ,{ 850, 850, 850, 280, 850}
00477     ,{ 280, 280, 280, 280, 280}
00478     ,{ 850, 850, 850, 280, 850}
00479     }
00480     ,{{ 850, 850, 850, 850, 850}
00481     ,{ 850, 850, 850, 850, 850}
00482     ,{ 850, 850, 850, 850, 850}
00483     ,{ 850, 850, 850, 850, 850}
00484     ,{ 850, 850, 850, 850, 850}
00485     }
00486     ,{{ 850, 280, 850, 280, 850}
00487     ,{ 850, 280, 850, 280, 850}
00488     ,{ 850, 280, 850, 280, 850}
00489     ,{ 280, 280, 280, 280, 280}
00490     ,{ 850, 280, 850, 280, 850}
```

```
00491     }
00492     ,{{ 850, 850, 850, 850, -160}
00493     ,{ 850, 850, 850, 850, -160}
00494     ,{ 850, 850, 850, 850, -160}
00495     ,{ 850, 850, 850, 850, -160}
00496     ,{ -160, -160, -160, -160, -160}
00497     }
00498     }
00499     }
00500     ,{{{ INF, INF, INF, INF, INF}
00501     ,{ INF, INF, INF, INF, INF}
00502     ,{ INF, INF, INF, INF, INF}
00503     ,{ INF, INF, INF, INF, INF}
00504     ,{ INF, INF, INF, INF, INF}
00505     }
00506     ,{{ INF, INF, INF, INF, INF}
00507     ,{ INF, INF, INF, INF, INF}
00508     ,{ INF, INF, INF, INF, INF}
00509     ,{ INF, INF, INF, INF, INF}
00510     ,{ INF, INF, INF, INF, INF}
00511     }
00512     ,{{ INF, INF, INF, INF, INF}
00513     ,{ INF, INF, INF, INF, INF}
00514     ,{ INF, INF, INF, INF, INF}
00515     ,{ INF, INF, INF, INF, INF}
00516     ,{ INF, INF, INF, INF, INF}
00517     }
00518     ,{{ INF, INF, INF, INF, INF}
00519     ,{ INF, INF, INF, INF, INF}
00520     ,{ INF, INF, INF, INF, INF}
00521     ,{ INF, INF, INF, INF, INF}
00522     ,{ INF, INF, INF, INF, INF}
00523     }
00524     ,{{ INF, INF, INF, INF, INF}
00525     ,{ INF, INF, INF, INF, INF}
00526     ,{ INF, INF, INF, INF, INF}
00527     ,{ INF, INF, INF, INF, INF}
00528     ,{ INF, INF, INF, INF, INF}
00529     }
00530     }
00531     ,{{{ 690, 690, 350, 350, 350}
00532     ,{ 690, 690, 350, 350, 350}
00533     ,{ 350, 350, 350, 350, 350}
00534     ,{ 350, 350, 350, 350, 350}
00535     ,{ 350, 350, 350, 350, 350}
00536     }
00537     ,{{ 690, 690, 350, 350, 350}
00538     ,{ 690, 690, 350, 240, 350}
00539     ,{ 350, 350, 350, 350, 350}
00540     ,{ -230, -500, -230, -230, -230}
00541     ,{ 350, 350, 350, 350, 350}
00542     }
00543     ,{{ 350, 350, 350, 350, 350}
00544     ,{ 350, 350, 350, 350, 350}
00545     ,{ 350, 350, 350, 350, 350}
00546     ,{ 350, 350, 350, 350, 350}
00547     ,{ 350, 350, 130, 350, 350}
00548     }
00549     ,{{ 350, -230, 350, -230, 350}
00550     ,{ -230, -230, -230, -230, -230}
00551     ,{ 350, -230, 350, -230, 350}
00552     ,{ -230, -230, -230, -230, -230}
00553     ,{ 350, -230, 350, -230, 350}
00554     }
00555     ,{{ 350, 350, 350, 350, -670}
00556     ,{ 350, 350, 350, 350, -670}
00557     ,{ 350, 350, 350, 350, -670}
00558     ,{ 350, 350, 350, 350, -670}
00559     ,{ -670, -670, -670, -670, -670}
00560     }
00561     }
00562     ,{{{ 350, 350, 350, 350, 350}
00563     ,{ 350, 350, 350, 350, 350}
00564     ,{ 350, 350, 350, 350, 350}
00565     ,{ 350, 350, 350, 350, 350}
00566     ,{ 350, 350, 350, 350, 350}
00567     }
00568     ,{{ 350, 350, 350, 350, 350}
00569     ,{ 350, 350, 350, 350, 350}
00570     ,{ 350, 350, 350, 350, 350}
00571     ,{ -230, -230, -230, -230, -230}
00572     ,{ 350, 350, 350, 350, 350}
00573     }
00574     ,{{ 350, 350, 350, 350, 350}
00575     ,{ 350, 350, 350, 350, 350}
00576     ,{ 350, 350, 350, 350, 350}
00577     ,{ 350, 350, 350, 350, 350}
```



```
00578 , { 350, 350, 350, 350, 350 }
00579 }
00580 , { { 350, -230, 350, -230, 350 }
00581 , { 350, -230, 350, -230, 350 }
00582 , { 350, -230, 350, -230, 350 }
00583 , { -230, -230, -230, -230, -230 }
00584 , { 350, -230, 350, -230, 350 }
00585 }
00586 , { { 350, 350, 350, 350, -670 }
00587 , { 350, 350, 350, 350, -670 }
00588 , { 350, 350, 350, 350, -670 }
00589 , { 350, 350, 350, 350, -670 }
00590 , { -670, -670, -670, -670, -670 }
00591 }
00592 }
00593 , { { { 850, 850, 850, 850, 850 }
00594 , { 850, 850, 850, 850, 850 }
00595 , { 850, 850, 850, 850, 850 }
00596 , { 850, 850, 850, 850, 850 }
00597 , { 850, 850, 850, 850, 850 }
00598 }
00599 , { { 850, 850, 850, 850, 850 }
00600 , { 850, 850, 850, 850, 850 }
00601 , { 850, 850, 850, 850, 850 }
00602 , { 280, 280, 280, 280, 280 }
00603 , { 850, 850, 850, 850, 850 }
00604 }
00605 , { { 850, 850, 850, 850, 850 }
00606 , { 850, 850, 850, 850, 850 }
00607 , { 850, 850, 850, 850, 850 }
00608 , { 850, 850, 850, 850, 850 }
00609 , { 850, 850, 850, 850, 850 }
00610 }
00611 , { { 850, 280, 850, 280, 850 }
00612 , { 850, 280, 850, 280, 850 }
00613 , { 850, 280, 850, 280, 850 }
00614 , { 280, 280, 280, 280, 280 }
00615 , { 850, 280, 850, 280, 850 }
00616 }
00617 , { { 850, 850, 850, 850, -160 }
00618 , { 850, 850, 850, 850, -160 }
00619 , { 850, 850, 850, 850, -160 }
00620 , { 850, 850, 850, 850, -160 }
00621 , { -160, -160, -160, -160, -160 }
00622 }
00623 }
00624 , { { { 850, 850, 850, 850, 850 }
00625 , { 850, 850, 850, 850, 850 }
00626 , { 850, 850, 850, 850, 850 }
00627 , { 850, 850, 850, 850, 850 }
00628 , { 850, 850, 850, 850, 850 }
00629 }
00630 , { { 850, 850, 850, 850, 850 }
00631 , { 850, 690, 850, 240, 850 }
00632 , { 850, 850, 850, 850, 850 }
00633 , { 280, -500, 280, 280, 280 }
00634 , { 850, 850, 850, 850, 850 }
00635 }
00636 , { { 850, 850, 850, 850, 850 }
00637 , { 850, 850, 850, 850, 850 }
00638 , { 850, 850, 850, 850, 850 }
00639 , { 850, 850, 850, 850, 850 }
00640 , { 850, 850, 130, 850, 850 }
00641 }
00642 , { { 850, 280, 850, 280, 850 }
00643 , { 280, 280, 280, 280, 280 }
00644 , { 850, 280, 850, 280, 850 }
00645 , { 280, 280, 280, 280, 280 }
00646 , { 850, 280, 850, 280, 850 }
00647 }
00648 , { { 850, 850, 850, 850, -160 }
00649 , { 850, 850, 850, 850, -160 }
00650 , { 850, 850, 850, 850, -160 }
00651 , { 850, 850, 850, 850, -160 }
00652 , { -160, -160, -160, -160, -160 }
00653 }
00654 }
00655 , { { { 850, 850, 850, 850, 850 }
00656 , { 850, 850, 850, 850, 850 }
00657 , { 850, 850, 850, 850, 850 }
00658 , { 850, 850, 850, 850, 850 }
00659 , { 850, 850, 850, 850, 850 }
00660 }
00661 , { { 850, 850, 850, 850, 850 }
00662 , { 850, 850, 850, 850, 850 }
00663 , { 850, 850, 850, 850, 850 }
00664 , { 280, 280, 280, 280, 280 }
```

```

00665 , { 850, 850, 850, 850, 850 }
00666 }
00667 , { { 850, 850, 850, 850, 850 }
00668 , { 850, 850, 850, 850, 850 }
00669 , { 850, 850, 850, 850, 850 }
00670 , { 850, 850, 850, 850, 850 }
00671 , { 850, 850, 850, 850, 850 }
00672 }
00673 , { { 850, 280, 850, 280, 850 }
00674 , { 850, 280, 850, 280, 850 }
00675 , { 850, 280, 850, 280, 850 }
00676 , { 280, 280, 280, 280, 280 }
00677 , { 850, 280, 850, 280, 850 }
00678 }
00679 , { { 850, 850, 850, 850, -160 }
00680 , { 850, 850, 850, 850, -160 }
00681 , { 850, 850, 850, 850, -160 }
00682 , { 850, 850, 850, 850, -160 }
00683 , { -160, -160, -160, -160, -160 }
00684 }
00685 }
00686 , { { { 850, 850, 850, 850, 850 }
00687 , { 850, 850, 850, 850, 850 }
00688 , { 850, 850, 850, 850, 850 }
00689 , { 850, 850, 850, 850, 850 }
00690 , { 850, 850, 850, 850, 850 }
00691 }
00692 , { { 850, 850, 850, 850, 850 }
00693 , { 850, 850, 850, 850, 850 }
00694 , { 850, 850, 850, 850, 850 }
00695 , { 280, 280, 280, 280, 280 }
00696 , { 850, 850, 850, 850, 850 }
00697 }
00698 , { { 850, 850, 850, 850, 850 }
00699 , { 850, 850, 850, 850, 850 }
00700 , { 850, 850, 850, 850, 850 }
00701 , { 850, 850, 850, 850, 850 }
00702 , { 850, 850, 850, 850, 850 }
00703 }
00704 , { { 850, 280, 850, 280, 850 }
00705 , { 280, 280, 280, 280, 280 }
00706 , { 850, 280, 850, 280, 850 }
00707 , { 280, 280, 280, 280, 280 }
00708 , { 850, 280, 850, 280, 850 }
00709 }
00710 , { { 850, 850, 850, 850, -160 }
00711 , { 850, 850, 850, 850, -160 }
00712 , { 850, 850, 850, 850, -160 }
00713 , { 850, 850, 850, 850, -160 }
00714 , { -160, -160, -160, -160, -160 }
00715 }
00716 }
00717 , { { { 850, 850, 850, 850, 850 }
00718 , { 850, 850, 850, 850, 850 }
00719 , { 850, 850, 850, 850, 850 }
00720 , { 850, 850, 850, 850, 850 }
00721 , { 850, 850, 850, 850, 850 }
00722 }
00723 , { { 850, 850, 850, 850, 850 }
00724 , { 850, 850, 850, 850, 850 }
00725 , { 850, 850, 850, 850, 850 }
00726 , { 280, 280, 280, 280, 280 }
00727 , { 850, 850, 850, 850, 850 }
00728 }
00729 , { { 850, 850, 850, 850, 850 }
00730 , { 850, 850, 850, 850, 850 }
00731 , { 850, 850, 850, 850, 850 }
00732 , { 850, 850, 850, 850, 850 }
00733 , { 850, 850, 850, 850, 850 }
00734 }
00735 , { { 850, 280, 850, 280, 850 }
00736 , { 850, 280, 850, 280, 850 }
00737 , { 850, 280, 850, 280, 850 }
00738 , { 280, 280, 280, 280, 280 }
00739 , { 850, 280, 850, 280, 850 }
00740 }
00741 , { { 850, 850, 850, 850, -160 }
00742 , { 850, 850, 850, 850, -160 }
00743 , { 850, 850, 850, 850, -160 }
00744 , { 850, 850, 850, 850, -160 }
00745 , { -160, -160, -160, -160, -160 }
00746 }
00747 }
00748 }
00749 , { { { { INF, INF, INF, INF, INF }
00750 , { INF, INF, INF, INF, INF }
00751 , { INF, INF, INF, INF, INF }

```

```
00752 , { INF, INF, INF, INF, INF }
00753 , { INF, INF, INF, INF, INF }
00754 }
00755 , { { INF, INF, INF, INF, INF }
00756 , { INF, INF, INF, INF, INF }
00757 , { INF, INF, INF, INF, INF }
00758 , { INF, INF, INF, INF, INF }
00759 , { INF, INF, INF, INF, INF }
00760 }
00761 , { { INF, INF, INF, INF, INF }
00762 , { INF, INF, INF, INF, INF }
00763 , { INF, INF, INF, INF, INF }
00764 , { INF, INF, INF, INF, INF }
00765 , { INF, INF, INF, INF, INF }
00766 }
00767 , { { INF, INF, INF, INF, INF }
00768 , { INF, INF, INF, INF, INF }
00769 , { INF, INF, INF, INF, INF }
00770 , { INF, INF, INF, INF, INF }
00771 , { INF, INF, INF, INF, INF }
00772 }
00773 , { { INF, INF, INF, INF, INF }
00774 , { INF, INF, INF, INF, INF }
00775 , { INF, INF, INF, INF, INF }
00776 , { INF, INF, INF, INF, INF }
00777 , { INF, INF, INF, INF, INF }
00778 }
00779 }
00780 , { { { 850, 850, 850, 850, 850 }
00781 , { 850, 850, 850, 850, 850 }
00782 , { 850, 850, 850, 850, 850 }
00783 , { 850, 850, 850, 850, 850 }
00784 , { 850, 850, 850, 850, 850 }
00785 }
00786 , { { 850, 850, 850, 850, 850 }
00787 , { 850, 690, 850, 240, 850 }
00788 , { 850, 850, 850, 850, 850 }
00789 , { 280, -500, 280, 280, 280 }
00790 , { 850, 850, 850, 850, 850 }
00791 }
00792 , { { 850, 850, 850, 850, 850 }
00793 , { 850, 850, 850, 850, 850 }
00794 , { 850, 850, 850, 850, 850 }
00795 , { 850, 850, 850, 850, 850 }
00796 , { 850, 850, 130, 850, 850 }
00797 }
00798 , { { 850, 280, 850, 280, 850 }
00799 , { 280, 280, 280, 280, 280 }
00800 , { 850, 280, 850, 280, 850 }
00801 , { 280, 280, 280, 280, 280 }
00802 , { 850, 280, 850, 280, 850 }
00803 }
00804 , { { 850, 850, 850, 850, -160 }
00805 , { 850, 850, 850, 850, -160 }
00806 , { 850, 850, 850, 850, -160 }
00807 , { 850, 850, 850, 850, -160 }
00808 , { -160, -160, -160, -160, -160 }
00809 }
00810 }
00811 , { { { 850, 850, 850, 850, 850 }
00812 , { 850, 850, 850, 850, 850 }
00813 , { 850, 850, 850, 850, 850 }
00814 , { 850, 850, 850, 850, 850 }
00815 , { 850, 850, 850, 850, 850 }
00816 }
00817 , { { 850, 850, 850, 850, 850 }
00818 , { 850, 850, 850, 850, 850 }
00819 , { 850, 850, 850, 850, 850 }
00820 , { 280, 280, 280, 280, 280 }
00821 , { 850, 850, 850, 850, 850 }
00822 }
00823 , { { 850, 850, 850, 850, 850 }
00824 , { 850, 850, 850, 850, 850 }
00825 , { 850, 850, 850, 850, 850 }
00826 , { 850, 850, 850, 850, 850 }
00827 , { 850, 850, 850, 850, 850 }
00828 }
00829 , { { 850, 280, 850, 280, 850 }
00830 , { 850, 280, 850, 280, 850 }
00831 , { 850, 280, 850, 280, 850 }
00832 , { 280, 280, 280, 280, 280 }
00833 , { 850, 280, 850, 280, 850 }
00834 }
00835 , { { 850, 850, 850, 850, -160 }
00836 , { 850, 850, 850, 850, -160 }
00837 , { 850, 850, 850, 850, -160 }
00838 , { 850, 850, 850, 850, -160 }
```

```
00839 , { -160, -160, -160, -160, -160 }
00840 }
00841 }
00842 , { { 1350, 1350, 1350, 1350, 1350 }
00843 , { 1350, 1350, 1350, 1350, 1350 }
00844 , { 1350, 1350, 1350, 1350, 1350 }
00845 , { 1350, 1350, 1350, 1350, 1350 }
00846 , { 1350, 1350, 1350, 1350, 1350 }
00847 }
00848 , { { 1350, 1350, 1350, 1350, 1350 }
00849 , { 1350, 1350, 1350, 1350, 1350 }
00850 , { 1350, 1350, 1350, 1350, 1350 }
00851 , { 780, 780, 780, 780, 780 }
00852 , { 1350, 1350, 1350, 1350, 1350 }
00853 }
00854 , { { 1350, 1350, 1350, 1350, 1350 }
00855 , { 1350, 1350, 1350, 1350, 1350 }
00856 , { 1350, 1350, 1350, 1350, 1350 }
00857 , { 1350, 1350, 1350, 1350, 1350 }
00858 , { 1350, 1350, 1350, 1350, 1350 }
00859 }
00860 , { { 1350, 780, 1350, 780, 1350 }
00861 , { 1350, 780, 1350, 780, 1350 }
00862 , { 1350, 780, 1350, 780, 1350 }
00863 , { 780, 780, 780, 780, 780 }
00864 , { 1350, 780, 1350, 780, 1350 }
00865 }
00866 , { { 1350, 1350, 1350, 1350, 340 }
00867 , { 1350, 1350, 1350, 1350, 340 }
00868 , { 1350, 1350, 1350, 1350, 340 }
00869 , { 1350, 1350, 1350, 1350, 340 }
00870 , { 340, 340, 340, 340, 340 }
00871 }
00872 }
00873 , { { { 1350, 1350, 1350, 1350, 1350 }
00874 , { 1350, 1350, 1350, 1350, 1350 }
00875 , { 1350, 1350, 1350, 1350, 1350 }
00876 , { 1350, 1350, 1350, 1350, 1350 }
00877 , { 1350, 1350, 1350, 1350, 1350 }
00878 }
00879 , { { 1350, 1350, 1350, 1350, 1350 }
00880 , { 1350, 690, 1350, 240, 1350 }
00881 , { 1350, 1350, 1350, 1350, 1350 }
00882 , { 780, -500, 780, 780, 780 }
00883 , { 1350, 1350, 1350, 1350, 1350 }
00884 }
00885 , { { 1350, 1350, 1350, 1350, 1350 }
00886 , { 1350, 1350, 1350, 1350, 1350 }
00887 , { 1350, 1350, 1350, 1350, 1350 }
00888 , { 1350, 1350, 1350, 1350, 1350 }
00889 , { 1350, 1350, 130, 1350, 1350 }
00890 }
00891 , { { 1350, 780, 1350, 780, 1350 }
00892 , { 780, 780, 780, 780, 780 }
00893 , { 1350, 780, 1350, 780, 1350 }
00894 , { 780, 780, 780, 780, 780 }
00895 , { 1350, 780, 1350, 780, 1350 }
00896 }
00897 , { { 1350, 1350, 1350, 1350, 340 }
00898 , { 1350, 1350, 1350, 1350, 340 }
00899 , { 1350, 1350, 1350, 1350, 340 }
00900 , { 1350, 1350, 1350, 1350, 340 }
00901 , { 340, 340, 340, 340, 340 }
00902 }
00903 }
00904 , { { { 1350, 1350, 1350, 1350, 1350 }
00905 , { 1350, 1350, 1350, 1350, 1350 }
00906 , { 1350, 1350, 1350, 1350, 1350 }
00907 , { 1350, 1350, 1350, 1350, 1350 }
00908 , { 1350, 1350, 1350, 1350, 1350 }
00909 }
00910 , { { 1350, 1350, 1350, 1350, 1350 }
00911 , { 1350, 1350, 1350, 1350, 1350 }
00912 , { 1350, 1350, 1350, 1350, 1350 }
00913 , { 780, 780, 780, 780, 780 }
00914 , { 1350, 1350, 1350, 1350, 1350 }
00915 }
00916 , { { 1350, 1350, 1350, 1350, 1350 }
00917 , { 1350, 1350, 1350, 1350, 1350 }
00918 , { 1350, 1350, 1350, 1350, 1350 }
00919 , { 1350, 1350, 1350, 1350, 1350 }
00920 , { 1350, 1350, 1350, 1350, 1350 }
00921 }
00922 , { { 1350, 780, 1350, 780, 1350 }
00923 , { 1350, 780, 1350, 780, 1350 }
00924 , { 1350, 780, 1350, 780, 1350 }
00925 , { 780, 780, 780, 780, 780 }
```

```
00926 , { 1350, 780, 1350, 780, 1350 }
00927 }
00928 , { { 1350, 1350, 1350, 1350, 340 }
00929 , { 1350, 1350, 1350, 1350, 340 }
00930 , { 1350, 1350, 1350, 1350, 340 }
00931 , { 1350, 1350, 1350, 1350, 340 }
00932 , { 340, 340, 340, 340, 340 }
00933 }
00934 }
00935 , { { { 1350, 1350, 1350, 1350, 1350 }
00936 , { 1350, 1350, 1350, 1350, 1350 }
00937 , { 1350, 1350, 1350, 1350, 1350 }
00938 , { 1350, 1350, 1350, 1350, 1350 }
00939 , { 1350, 1350, 1350, 1350, 1350 }
00940 }
00941 , { { 1350, 1350, 1350, 1350, 1350 }
00942 , { 1350, 1350, 1350, 1350, 1350 }
00943 , { 1350, 1350, 1350, 1350, 1350 }
00944 , { 780, 780, 780, 780, 780 }
00945 , { 1350, 1350, 1350, 1350, 1350 }
00946 }
00947 , { { 1350, 1350, 1350, 1350, 1350 }
00948 , { 1350, 1350, 1350, 1350, 1350 }
00949 , { 1350, 1350, 1350, 1350, 1350 }
00950 , { 1350, 1350, 1350, 1350, 1350 }
00951 , { 1350, 1350, 1350, 1350, 1350 }
00952 }
00953 , { { 1350, 780, 1350, 780, 1350 }
00954 , { 780, 780, 780, 780, 780 }
00955 , { 1350, 780, 1350, 780, 1350 }
00956 , { 780, 780, 780, 780, 780 }
00957 , { 1350, 780, 1350, 780, 1350 }
00958 }
00959 , { { 1350, 1350, 1350, 1350, 340 }
00960 , { 1350, 1350, 1350, 1350, 340 }
00961 , { 1350, 1350, 1350, 1350, 340 }
00962 , { 1350, 1350, 1350, 1350, 340 }
00963 , { 340, 340, 340, 340, 340 }
00964 }
00965 }
00966 , { { { 1350, 1350, 1350, 1350, 1350 }
00967 , { 1350, 1350, 1350, 1350, 1350 }
00968 , { 1350, 1350, 1350, 1350, 1350 }
00969 , { 1350, 1350, 1350, 1350, 1350 }
00970 , { 1350, 1350, 1350, 1350, 1350 }
00971 }
00972 , { { 1350, 1350, 1350, 1350, 1350 }
00973 , { 1350, 1350, 1350, 1350, 1350 }
00974 , { 1350, 1350, 1350, 1350, 1350 }
00975 , { 780, 780, 780, 780, 780 }
00976 , { 1350, 1350, 1350, 1350, 1350 }
00977 }
00978 , { { 1350, 1350, 1350, 1350, 1350 }
00979 , { 1350, 1350, 1350, 1350, 1350 }
00980 , { 1350, 1350, 1350, 1350, 1350 }
00981 , { 1350, 1350, 1350, 1350, 1350 }
00982 , { 1350, 1350, 1350, 1350, 1350 }
00983 }
00984 , { { 1350, 780, 1350, 780, 1350 }
00985 , { 1350, 780, 1350, 780, 1350 }
00986 , { 1350, 780, 1350, 780, 1350 }
00987 , { 780, 780, 780, 780, 780 }
00988 , { 1350, 780, 1350, 780, 1350 }
00989 }
00990 , { { 1350, 1350, 1350, 1350, 340 }
00991 , { 1350, 1350, 1350, 1350, 340 }
00992 , { 1350, 1350, 1350, 1350, 340 }
00993 , { 1350, 1350, 1350, 1350, 340 }
00994 , { 340, 340, 340, 340, 340 }
00995 }
00996 }
00997 }
00998 , { { { INF, INF, INF, INF, INF }
00999 , { INF, INF, INF, INF, INF }
01000 , { INF, INF, INF, INF, INF }
01001 , { INF, INF, INF, INF, INF }
01002 , { INF, INF, INF, INF, INF }
01003 }
01004 , { { INF, INF, INF, INF, INF }
01005 , { INF, INF, INF, INF, INF }
01006 , { INF, INF, INF, INF, INF }
01007 , { INF, INF, INF, INF, INF }
01008 , { INF, INF, INF, INF, INF }
01009 }
01010 , { { INF, INF, INF, INF, INF }
01011 , { INF, INF, INF, INF, INF }
01012 , { INF, INF, INF, INF, INF }
```

```
01013 , { INF, INF, INF, INF, INF }
01014 , { INF, INF, INF, INF, INF }
01015 }
01016 , { { INF, INF, INF, INF, INF }
01017 , { INF, INF, INF, INF, INF }
01018 , { INF, INF, INF, INF, INF }
01019 , { INF, INF, INF, INF, INF }
01020 , { INF, INF, INF, INF, INF }
01021 }
01022 , { { INF, INF, INF, INF, INF }
01023 , { INF, INF, INF, INF, INF }
01024 , { INF, INF, INF, INF, INF }
01025 , { INF, INF, INF, INF, INF }
01026 , { INF, INF, INF, INF, INF }
01027 }
01028 }
01029 , { { 850, 850, 850, 850, 850 }
01030 , { 850, 850, 850, 850, 850 }
01031 , { 850, 850, 850, 850, 850 }
01032 , { 850, 850, 850, 850, 850 }
01033 , { 850, 850, 850, 850, 850 }
01034 }
01035 , { { 850, 850, 850, 280, 850 }
01036 , { 850, 850, 850, 280, 850 }
01037 , { 850, 850, 850, 280, 850 }
01038 , { 280, 280, 280, 280, 280 }
01039 , { 850, 850, 850, 280, 850 }
01040 }
01041 , { { 850, 850, 850, 850, 850 }
01042 , { 850, 850, 850, 850, 850 }
01043 , { 850, 850, 850, 850, 850 }
01044 , { 850, 850, 850, 850, 850 }
01045 , { 850, 850, 850, 850, 850 }
01046 }
01047 , { { 850, 280, 850, 280, 850 }
01048 , { 280, 280, 280, 280, 280 }
01049 , { 850, 280, 850, 280, 850 }
01050 , { 280, 280, 280, 280, 280 }
01051 , { 850, 280, 850, 280, 850 }
01052 }
01053 , { { 850, 850, 850, 850, -160 }
01054 , { 850, 850, 850, 850, -160 }
01055 , { 850, 850, 850, 850, -160 }
01056 , { 850, 850, 850, 850, -160 }
01057 , { -160, -160, -160, -160, -160 }
01058 }
01059 }
01060 , { { 850, 850, 850, 850, 850 }
01061 , { 850, 850, 850, 850, 850 }
01062 , { 850, 850, 850, 850, 850 }
01063 , { 850, 850, 850, 850, 850 }
01064 , { 850, 850, 850, 850, 850 }
01065 }
01066 , { { 850, 850, 850, 280, 850 }
01067 , { 850, 850, 850, 280, 850 }
01068 , { 850, 850, 850, 280, 850 }
01069 , { 280, 280, 280, 280, 280 }
01070 , { 850, 850, 850, 280, 850 }
01071 }
01072 , { { 850, 850, 850, 850, 850 }
01073 , { 850, 850, 850, 850, 850 }
01074 , { 850, 850, 850, 850, 850 }
01075 , { 850, 850, 850, 850, 850 }
01076 , { 850, 850, 850, 850, 850 }
01077 }
01078 , { { 850, 280, 850, 280, 850 }
01079 , { 850, 280, 850, 280, 850 }
01080 , { 850, 280, 850, 280, 850 }
01081 , { 280, 280, 280, 280, 280 }
01082 , { 850, 280, 850, 280, 850 }
01083 }
01084 , { { 850, 850, 850, 850, -160 }
01085 , { 850, 850, 850, 850, -160 }
01086 , { 850, 850, 850, 850, -160 }
01087 , { 850, 850, 850, 850, -160 }
01088 , { -160, -160, -160, -160, -160 }
01089 }
01090 }
01091 , { { 1350, 1350, 1350, 1350, 1350 }
01092 , { 1350, 1350, 1350, 1350, 1350 }
01093 , { 1350, 1350, 1350, 1350, 1350 }
01094 , { 1350, 1350, 1350, 1350, 1350 }
01095 , { 1350, 1350, 1350, 1350, 1350 }
01096 }
01097 , { { 1350, 1350, 1350, 780, 1350 }
01098 , { 1350, 1350, 1350, 780, 1350 }
01099 , { 1350, 1350, 1350, 780, 1350 }
```

```
01100 , { 780, 780, 780, 780, 780 }
01101 , { 1350, 1350, 1350, 780, 1350 }
01102 }
01103 , { { 1350, 1350, 1350, 1350, 1350 }
01104 , { 1350, 1350, 1350, 1350, 1350 }
01105 , { 1350, 1350, 1350, 1350, 1350 }
01106 , { 1350, 1350, 1350, 1350, 1350 }
01107 , { 1350, 1350, 1350, 1350, 1350 }
01108 }
01109 , { { 1350, 780, 1350, 780, 1350 }
01110 , { 1350, 780, 1350, 780, 1350 }
01111 , { 1350, 780, 1350, 780, 1350 }
01112 , { 780, 780, 780, 780, 780 }
01113 , { 1350, 780, 1350, 780, 1350 }
01114 }
01115 , { { 1350, 1350, 1350, 1350, 340 }
01116 , { 1350, 1350, 1350, 1350, 340 }
01117 , { 1350, 1350, 1350, 1350, 340 }
01118 , { 1350, 1350, 1350, 1350, 340 }
01119 , { 340, 340, 340, 340, 340 }
01120 }
01121 }
01122 , { { { 1350, 1350, 1350, 1350, 1350 }
01123 , { 1350, 1350, 1350, 1350, 1350 }
01124 , { 1350, 1350, 1350, 1350, 1350 }
01125 , { 1350, 1350, 1350, 1350, 1350 }
01126 , { 1350, 1350, 1350, 1350, 1350 }
01127 }
01128 , { { 1350, 1350, 1350, 780, 1350 }
01129 , { 1350, 1350, 1350, 780, 1350 }
01130 , { 1350, 1350, 1350, 780, 1350 }
01131 , { 780, 780, 780, 780, 780 }
01132 , { 1350, 1350, 1350, 780, 1350 }
01133 }
01134 , { { 1350, 1350, 1350, 1350, 1350 }
01135 , { 1350, 1350, 1350, 1350, 1350 }
01136 , { 1350, 1350, 1350, 1350, 1350 }
01137 , { 1350, 1350, 1350, 1350, 1350 }
01138 , { 1350, 1350, 1350, 1350, 1350 }
01139 }
01140 , { { 1350, 780, 1350, 780, 1350 }
01141 , { 780, 780, 780, 780, 780 }
01142 , { 1350, 780, 1350, 780, 1350 }
01143 , { 780, 780, 780, 780, 780 }
01144 , { 1350, 780, 1350, 780, 1350 }
01145 }
01146 , { { 1350, 1350, 1350, 1350, 340 }
01147 , { 1350, 1350, 1350, 1350, 340 }
01148 , { 1350, 1350, 1350, 1350, 340 }
01149 , { 1350, 1350, 1350, 1350, 340 }
01150 , { 340, 340, 340, 340, 340 }
01151 }
01152 }
01153 , { { { 1350, 1350, 1350, 1350, 1350 }
01154 , { 1350, 1350, 1350, 1350, 1350 }
01155 , { 1350, 1350, 1350, 1350, 1350 }
01156 , { 1350, 1350, 1350, 1350, 1350 }
01157 , { 1350, 1350, 1350, 1350, 1350 }
01158 }
01159 , { { 1350, 1350, 1350, 780, 1350 }
01160 , { 1350, 1350, 1350, 780, 1350 }
01161 , { 1350, 1350, 1350, 780, 1350 }
01162 , { 780, 780, 780, 780, 780 }
01163 , { 1350, 1350, 1350, 780, 1350 }
01164 }
01165 , { { 1350, 1350, 1350, 1350, 1350 }
01166 , { 1350, 1350, 1350, 1350, 1350 }
01167 , { 1350, 1350, 1350, 1350, 1350 }
01168 , { 1350, 1350, 1350, 1350, 1350 }
01169 , { 1350, 1350, 1350, 1350, 1350 }
01170 }
01171 , { { 1350, 780, 1350, 780, 1350 }
01172 , { 1350, 780, 1350, 780, 1350 }
01173 , { 1350, 780, 1350, 780, 1350 }
01174 , { 780, 780, 780, 780, 780 }
01175 , { 1350, 780, 1350, 780, 1350 }
01176 }
01177 , { { 1350, 1350, 1350, 1350, 340 }
01178 , { 1350, 1350, 1350, 1350, 340 }
01179 , { 1350, 1350, 1350, 1350, 340 }
01180 , { 1350, 1350, 1350, 1350, 340 }
01181 , { 340, 340, 340, 340, 340 }
01182 }
01183 }
01184 , { { { 1350, 1350, 1350, 1350, 1350 }
01185 , { 1350, 1350, 1350, 1350, 1350 }
01186 , { 1350, 1350, 1350, 1350, 1350 }
```

```
01187 , { 1350, 1350, 1350, 1350, 1350}
01188 , { 1350, 1350, 1350, 1350, 1350}
01189 }
01190 , { { 1350, 1350, 1350, 780, 1350}
01191 , { 1350, 1350, 1350, 780, 1350}
01192 , { 1350, 1350, 1350, 780, 1350}
01193 , { 780, 780, 780, 780, 780}
01194 , { 1350, 1350, 1350, 780, 1350}
01195 }
01196 , { { 1350, 1350, 1350, 1350, 1350}
01197 , { 1350, 1350, 1350, 1350, 1350}
01198 , { 1350, 1350, 1350, 1350, 1350}
01199 , { 1350, 1350, 1350, 1350, 1350}
01200 , { 1350, 1350, 1350, 1350, 1350}
01201 }
01202 , { { 1350, 780, 1350, 780, 1350}
01203 , { 780, 780, 780, 780, 780}
01204 , { 1350, 780, 1350, 780, 1350}
01205 , { 780, 780, 780, 780, 780}
01206 , { 1350, 780, 1350, 780, 1350}
01207 }
01208 , { { 1350, 1350, 1350, 1350, 340}
01209 , { 1350, 1350, 1350, 1350, 340}
01210 , { 1350, 1350, 1350, 1350, 340}
01211 , { 1350, 1350, 1350, 1350, 340}
01212 , { 340, 340, 340, 340, 340}
01213 }
01214 }
01215 , { { { 1350, 1350, 1350, 1350, 1350}
01216 , { 1350, 1350, 1350, 1350, 1350}
01217 , { 1350, 1350, 1350, 1350, 1350}
01218 , { 1350, 1350, 1350, 1350, 1350}
01219 , { 1350, 1350, 1350, 1350, 1350}
01220 }
01221 , { { 1350, 1350, 1350, 780, 1350}
01222 , { 1350, 1350, 1350, 780, 1350}
01223 , { 1350, 1350, 1350, 780, 1350}
01224 , { 780, 780, 780, 780, 780}
01225 , { 1350, 1350, 1350, 780, 1350}
01226 }
01227 , { { 1350, 1350, 1350, 1350, 1350}
01228 , { 1350, 1350, 1350, 1350, 1350}
01229 , { 1350, 1350, 1350, 1350, 1350}
01230 , { 1350, 1350, 1350, 1350, 1350}
01231 , { 1350, 1350, 1350, 1350, 1350}
01232 }
01233 , { { 1350, 780, 1350, 780, 1350}
01234 , { 1350, 780, 1350, 780, 1350}
01235 , { 1350, 780, 1350, 780, 1350}
01236 , { 780, 780, 780, 780, 780}
01237 , { 1350, 780, 1350, 780, 1350}
01238 }
01239 , { { 1350, 1350, 1350, 1350, 340}
01240 , { 1350, 1350, 1350, 1350, 340}
01241 , { 1350, 1350, 1350, 1350, 340}
01242 , { 1350, 1350, 1350, 1350, 340}
01243 , { 340, 340, 340, 340, 340}
01244 }
01245 }
01246 }
01247 , { { { INF, INF, INF, INF, INF}
01248 , { INF, INF, INF, INF, INF}
01249 , { INF, INF, INF, INF, INF}
01250 , { INF, INF, INF, INF, INF}
01251 , { INF, INF, INF, INF, INF}
01252 }
01253 , { { INF, INF, INF, INF, INF}
01254 , { INF, INF, INF, INF, INF}
01255 , { INF, INF, INF, INF, INF}
01256 , { INF, INF, INF, INF, INF}
01257 , { INF, INF, INF, INF, INF}
01258 }
01259 , { { INF, INF, INF, INF, INF}
01260 , { INF, INF, INF, INF, INF}
01261 , { INF, INF, INF, INF, INF}
01262 , { INF, INF, INF, INF, INF}
01263 , { INF, INF, INF, INF, INF}
01264 }
01265 , { { INF, INF, INF, INF, INF}
01266 , { INF, INF, INF, INF, INF}
01267 , { INF, INF, INF, INF, INF}
01268 , { INF, INF, INF, INF, INF}
01269 , { INF, INF, INF, INF, INF}
01270 }
01271 , { { INF, INF, INF, INF, INF}
01272 , { INF, INF, INF, INF, INF}
01273 , { INF, INF, INF, INF, INF}
```



```
01274 , { INF, INF, INF, INF, INF }
01275 , { INF, INF, INF, INF, INF }
01276 }
01277 }
01278 , { { 850, 850, 850, 850, 850 }
01279 , { 850, 850, 850, 850, 850 }
01280 , { 850, 850, 850, 850, 850 }
01281 , { 850, 850, 850, 850, 850 }
01282 , { 850, 850, 850, 850, 850 }
01283 }
01284 , { { 850, 850, 850, 850, 850 }
01285 , { 850, 850, 850, 850, 850 }
01286 , { 850, 850, 850, 850, 850 }
01287 , { 280, 280, 280, 280, 280 }
01288 , { 850, 850, 850, 850, 850 }
01289 }
01290 , { { 850, 850, 850, 850, 850 }
01291 , { 850, 850, 850, 850, 850 }
01292 , { 850, 850, 850, 850, 850 }
01293 , { 850, 850, 850, 850, 850 }
01294 , { 850, 850, 850, 850, 850 }
01295 }
01296 , { { 850, 280, 850, 280, 850 }
01297 , { 280, 280, 280, 280, 280 }
01298 , { 850, 280, 850, 280, 850 }
01299 , { 280, 280, 280, 280, 280 }
01300 , { 850, 280, 850, 280, 850 }
01301 }
01302 , { { 850, 850, 850, 850, -160 }
01303 , { 850, 850, 850, 850, -160 }
01304 , { 850, 850, 850, 850, -160 }
01305 , { 850, 850, 850, 850, -160 }
01306 , { -160, -160, -160, -160, -160 }
01307 }
01308 }
01309 , { { { 850, 850, 850, 850, 850 }
01310 , { 850, 850, 850, 850, 850 }
01311 , { 850, 850, 850, 850, 850 }
01312 , { 850, 850, 850, 850, 850 }
01313 , { 850, 850, 850, 850, 850 }
01314 }
01315 , { { 850, 850, 850, 850, 850 }
01316 , { 850, 850, 850, 850, 850 }
01317 , { 850, 850, 850, 850, 850 }
01318 , { 280, 280, 280, 280, 280 }
01319 , { 850, 850, 850, 850, 850 }
01320 }
01321 , { { 850, 850, 850, 850, 850 }
01322 , { 850, 850, 850, 850, 850 }
01323 , { 850, 850, 850, 850, 850 }
01324 , { 850, 850, 850, 850, 850 }
01325 , { 850, 850, 850, 850, 850 }
01326 }
01327 , { { 850, 280, 850, 280, 850 }
01328 , { 850, 280, 850, 280, 850 }
01329 , { 850, 280, 850, 280, 850 }
01330 , { 280, 280, 280, 280, 280 }
01331 , { 850, 280, 850, 280, 850 }
01332 }
01333 , { { 850, 850, 850, 850, -160 }
01334 , { 850, 850, 850, 850, -160 }
01335 , { 850, 850, 850, 850, -160 }
01336 , { 850, 850, 850, 850, -160 }
01337 , { -160, -160, -160, -160, -160 }
01338 }
01339 }
01340 , { { { 1350, 1350, 1350, 1350, 1350 }
01341 , { 1350, 1350, 1350, 1350, 1350 }
01342 , { 1350, 1350, 1350, 1350, 1350 }
01343 , { 1350, 1350, 1350, 1350, 1350 }
01344 , { 1350, 1350, 1350, 1350, 1350 }
01345 }
01346 , { { 1350, 1350, 1350, 1350, 1350 }
01347 , { 1350, 1350, 1350, 1350, 1350 }
01348 , { 1350, 1350, 1350, 1350, 1350 }
01349 , { 780, 780, 780, 780, 780 }
01350 , { 1350, 1350, 1350, 1350, 1350 }
01351 }
01352 , { { 1350, 1350, 1350, 1350, 1350 }
01353 , { 1350, 1350, 1350, 1350, 1350 }
01354 , { 1350, 1350, 1350, 1350, 1350 }
01355 , { 1350, 1350, 1350, 1350, 1350 }
01356 , { 1350, 1350, 1350, 1350, 1350 }
01357 }
01358 , { { 1350, 780, 1350, 780, 1350 }
01359 , { 1350, 780, 1350, 780, 1350 }
01360 , { 1350, 780, 1350, 780, 1350 }
```

```
01361 , { 780, 780, 780, 780, 780 }
01362 , { 1350, 780, 1350, 780, 1350 }
01363 }
01364 , { { 1350, 1350, 1350, 1350, 340 }
01365 , { 1350, 1350, 1350, 1350, 340 }
01366 , { 1350, 1350, 1350, 1350, 340 }
01367 , { 1350, 1350, 1350, 1350, 340 }
01368 , { 340, 340, 340, 340, 340 }
01369 }
01370 }
01371 , { { 1350, 1350, 1350, 1350, 1350 }
01372 , { 1350, 1350, 1350, 1350, 1350 }
01373 , { 1350, 1350, 1350, 1350, 1350 }
01374 , { 1350, 1350, 1350, 1350, 1350 }
01375 , { 1350, 1350, 1350, 1350, 1350 }
01376 }
01377 , { { 1350, 1350, 1350, 1350, 1350 }
01378 , { 1350, 1350, 1350, 1350, 1350 }
01379 , { 1350, 1350, 1350, 1350, 1350 }
01380 , { 780, 780, 780, 780, 780 }
01381 , { 1350, 1350, 1350, 1350, 1350 }
01382 }
01383 , { { 1350, 1350, 1350, 1350, 1350 }
01384 , { 1350, 1350, 1350, 1350, 1350 }
01385 , { 1350, 1350, 1350, 1350, 1350 }
01386 , { 1350, 1350, 1350, 1350, 1350 }
01387 , { 1350, 1350, 1350, 1350, 1350 }
01388 }
01389 , { { 1350, 780, 1350, 780, 1350 }
01390 , { 780, 780, 780, 780, 780 }
01391 , { 1350, 780, 1350, 780, 1350 }
01392 , { 780, 780, 780, 780, 780 }
01393 , { 1350, 780, 1350, 780, 1350 }
01394 }
01395 , { { 1350, 1350, 1350, 1350, 340 }
01396 , { 1350, 1350, 1350, 1350, 340 }
01397 , { 1350, 1350, 1350, 1350, 340 }
01398 , { 1350, 1350, 1350, 1350, 340 }
01399 , { 340, 340, 340, 340, 340 }
01400 }
01401 }
01402 , { { { 1350, 1350, 1350, 1350, 1350 }
01403 , { 1350, 1350, 1350, 1350, 1350 }
01404 , { 1350, 1350, 1350, 1350, 1350 }
01405 , { 1350, 1350, 1350, 1350, 1350 }
01406 , { 1350, 1350, 1350, 1350, 1350 }
01407 }
01408 , { { 1350, 1350, 1350, 1350, 1350 }
01409 , { 1350, 1350, 1350, 1350, 1350 }
01410 , { 1350, 1350, 1350, 1350, 1350 }
01411 , { 780, 780, 780, 780, 780 }
01412 , { 1350, 1350, 1350, 1350, 1350 }
01413 }
01414 , { { 1350, 1350, 1350, 1350, 1350 }
01415 , { 1350, 1350, 1350, 1350, 1350 }
01416 , { 1350, 1350, 1350, 1350, 1350 }
01417 , { 1350, 1350, 1350, 1350, 1350 }
01418 , { 1350, 1350, 1350, 1350, 1350 }
01419 }
01420 , { { 1350, 780, 1350, 780, 1350 }
01421 , { 1350, 780, 1350, 780, 1350 }
01422 , { 1350, 780, 1350, 780, 1350 }
01423 , { 780, 780, 780, 780, 780 }
01424 , { 1350, 780, 1350, 780, 1350 }
01425 }
01426 , { { 1350, 1350, 1350, 1350, 340 }
01427 , { 1350, 1350, 1350, 1350, 340 }
01428 , { 1350, 1350, 1350, 1350, 340 }
01429 , { 1350, 1350, 1350, 1350, 340 }
01430 , { 340, 340, 340, 340, 340 }
01431 }
01432 }
01433 , { { { 1350, 1350, 1350, 1350, 1350 }
01434 , { 1350, 1350, 1350, 1350, 1350 }
01435 , { 1350, 1350, 1350, 1350, 1350 }
01436 , { 1350, 1350, 1350, 1350, 1350 }
01437 , { 1350, 1350, 1350, 1350, 1350 }
01438 }
01439 , { { 1350, 1350, 1350, 1350, 1350 }
01440 , { 1350, 1350, 1350, 1350, 1350 }
01441 , { 1350, 1350, 1350, 1350, 1350 }
01442 , { 780, 780, 780, 780, 780 }
01443 , { 1350, 1350, 1350, 1350, 1350 }
01444 }
01445 , { { 1350, 1350, 1350, 1350, 1350 }
01446 , { 1350, 1350, 1350, 1350, 1350 }
01447 , { 1350, 1350, 1350, 1350, 1350 }
```

```
01448 , { 1350, 1350, 1350, 1350, 1350}
01449 , { 1350, 1350, 1350, 1350, 1350}
01450 }
01451 , { { 1350, 780, 1350, 780, 1350}
01452 , { 780, 780, 780, 780, 780}
01453 , { 1350, 780, 1350, 780, 1350}
01454 , { 780, 780, 780, 780, 780}
01455 , { 1350, 780, 1350, 780, 1350}
01456 }
01457 , { { 1350, 1350, 1350, 1350, 340}
01458 , { 1350, 1350, 1350, 1350, 340}
01459 , { 1350, 1350, 1350, 1350, 340}
01460 , { 1350, 1350, 1350, 1350, 340}
01461 , { 340, 340, 340, 340, 340}
01462 }
01463 }
01464 , { { { 1350, 1350, 1350, 1350, 1350}
01465 , { 1350, 1350, 1350, 1350, 1350}
01466 , { 1350, 1350, 1350, 1350, 1350}
01467 , { 1350, 1350, 1350, 1350, 1350}
01468 , { 1350, 1350, 1350, 1350, 1350}
01469 }
01470 , { { 1350, 1350, 1350, 1350, 1350}
01471 , { 1350, 1350, 1350, 1350, 1350}
01472 , { 1350, 1350, 1350, 1350, 1350}
01473 , { 780, 780, 780, 780, 780}
01474 , { 1350, 1350, 1350, 1350, 1350}
01475 }
01476 , { { 1350, 1350, 1350, 1350, 1350}
01477 , { 1350, 1350, 1350, 1350, 1350}
01478 , { 1350, 1350, 1350, 1350, 1350}
01479 , { 1350, 1350, 1350, 1350, 1350}
01480 , { 1350, 1350, 1350, 1350, 1350}
01481 }
01482 , { { 1350, 780, 1350, 780, 1350}
01483 , { 1350, 780, 1350, 780, 1350}
01484 , { 1350, 780, 1350, 780, 1350}
01485 , { 780, 780, 780, 780, 780}
01486 , { 1350, 780, 1350, 780, 1350}
01487 }
01488 , { { 1350, 1350, 1350, 1350, 340}
01489 , { 1350, 1350, 1350, 1350, 340}
01490 , { 1350, 1350, 1350, 1350, 340}
01491 , { 1350, 1350, 1350, 1350, 340}
01492 , { 340, 340, 340, 340, 340}
01493 }
01494 }
01495 }
01496 , { { { INF, INF, INF, INF, INF}
01497 , { INF, INF, INF, INF, INF}
01498 , { INF, INF, INF, INF, INF}
01499 , { INF, INF, INF, INF, INF}
01500 , { INF, INF, INF, INF, INF}
01501 }
01502 , { { INF, INF, INF, INF, INF}
01503 , { INF, INF, INF, INF, INF}
01504 , { INF, INF, INF, INF, INF}
01505 , { INF, INF, INF, INF, INF}
01506 , { INF, INF, INF, INF, INF}
01507 }
01508 , { { INF, INF, INF, INF, INF}
01509 , { INF, INF, INF, INF, INF}
01510 , { INF, INF, INF, INF, INF}
01511 , { INF, INF, INF, INF, INF}
01512 , { INF, INF, INF, INF, INF}
01513 }
01514 , { { INF, INF, INF, INF, INF}
01515 , { INF, INF, INF, INF, INF}
01516 , { INF, INF, INF, INF, INF}
01517 , { INF, INF, INF, INF, INF}
01518 , { INF, INF, INF, INF, INF}
01519 }
01520 , { { INF, INF, INF, INF, INF}
01521 , { INF, INF, INF, INF, INF}
01522 , { INF, INF, INF, INF, INF}
01523 , { INF, INF, INF, INF, INF}
01524 , { INF, INF, INF, INF, INF}
01525 }
01526 }
01527 , { { { 850, 850, 850, 850, 850}
01528 , { 850, 850, 850, 850, 850}
01529 , { 850, 850, 850, 850, 850}
01530 , { 850, 850, 850, 850, 850}
01531 , { 850, 850, 850, 850, 850}
01532 }
01533 , { { 850, 850, 850, 280, 850}
01534 , { 850, 850, 850, 280, 850}
```

```
01535 , { 850, 850, 850, 280, 850}
01536 , { 280, 280, 280, 280, 280}
01537 , { 850, 850, 850, 280, 850}
01538 }
01539 , { { 850, 850, 850, 850, 850}
01540 , { 850, 850, 850, 850, 850}
01541 , { 850, 850, 850, 850, 850}
01542 , { 850, 850, 850, 850, 850}
01543 , { 850, 850, 850, 850, 850}
01544 }
01545 , { { 850, 280, 850, 280, 850}
01546 , { 280, 280, 280, 280, 280}
01547 , { 850, 280, 850, 280, 850}
01548 , { 280, 280, 280, 280, 280}
01549 , { 850, 280, 850, 280, 850}
01550 }
01551 , { { 850, 850, 850, 850, -160}
01552 , { 850, 850, 850, 850, -160}
01553 , { 850, 850, 850, 850, -160}
01554 , { 850, 850, 850, 850, -160}
01555 , { -160, -160, -160, -160, -160}
01556 }
01557 }
01558 , { { { 850, 850, 850, 850, 850}
01559 , { 850, 850, 850, 850, 850}
01560 , { 850, 850, 850, 850, 850}
01561 , { 850, 850, 850, 850, 850}
01562 , { 850, 850, 850, 850, 850}
01563 }
01564 , { { 850, 850, 850, 280, 850}
01565 , { 850, 850, 850, 280, 850}
01566 , { 850, 850, 850, 280, 850}
01567 , { 280, 280, 280, 280, 280}
01568 , { 850, 850, 850, 280, 850}
01569 }
01570 , { { 850, 850, 850, 850, 850}
01571 , { 850, 850, 850, 850, 850}
01572 , { 850, 850, 850, 850, 850}
01573 , { 850, 850, 850, 850, 850}
01574 , { 850, 850, 850, 850, 850}
01575 }
01576 , { { 850, 280, 850, 280, 850}
01577 , { 850, 280, 850, 280, 850}
01578 , { 850, 280, 850, 280, 850}
01579 , { 280, 280, 280, 280, 280}
01580 , { 850, 280, 850, 280, 850}
01581 }
01582 , { { 850, 850, 850, 850, -160}
01583 , { 850, 850, 850, 850, -160}
01584 , { 850, 850, 850, 850, -160}
01585 , { 850, 850, 850, 850, -160}
01586 , { -160, -160, -160, -160, -160}
01587 }
01588 }
01589 , { { { 1350, 1350, 1350, 1350, 1350}
01590 , { 1350, 1350, 1350, 1350, 1350}
01591 , { 1350, 1350, 1350, 1350, 1350}
01592 , { 1350, 1350, 1350, 1350, 1350}
01593 , { 1350, 1350, 1350, 1350, 1350}
01594 }
01595 , { { 1350, 1350, 1350, 780, 1350}
01596 , { 1350, 1350, 1350, 780, 1350}
01597 , { 1350, 1350, 1350, 780, 1350}
01598 , { 780, 780, 780, 780, 780}
01599 , { 1350, 1350, 1350, 780, 1350}
01600 }
01601 , { { 1350, 1350, 1350, 1350, 1350}
01602 , { 1350, 1350, 1350, 1350, 1350}
01603 , { 1350, 1350, 1350, 1350, 1350}
01604 , { 1350, 1350, 1350, 1350, 1350}
01605 , { 1350, 1350, 1350, 1350, 1350}
01606 }
01607 , { { 1350, 780, 1350, 780, 1350}
01608 , { 1350, 780, 1350, 780, 1350}
01609 , { 1350, 780, 1350, 780, 1350}
01610 , { 780, 780, 780, 780, 780}
01611 , { 1350, 780, 1350, 780, 1350}
01612 }
01613 , { { 1350, 1350, 1350, 1350, 340}
01614 , { 1350, 1350, 1350, 1350, 340}
01615 , { 1350, 1350, 1350, 1350, 340}
01616 , { 1350, 1350, 1350, 1350, 340}
01617 , { 340, 340, 340, 340, 340}
01618 }
01619 }
01620 , { { { 1350, 1350, 1350, 1350, 1350}
01621 , { 1350, 1350, 1350, 1350, 1350}
```

```
01622 , { 1350, 1350, 1350, 1350, 1350}
01623 , { 1350, 1350, 1350, 1350, 1350}
01624 , { 1350, 1350, 1350, 1350, 1350}
01625 }
01626 , { { 1350, 1350, 1350, 780, 1350}
01627 , { 1350, 1350, 1350, 780, 1350}
01628 , { 1350, 1350, 1350, 780, 1350}
01629 , { 780, 780, 780, 780, 780}
01630 , { 1350, 1350, 1350, 780, 1350}
01631 }
01632 , { { 1350, 1350, 1350, 1350, 1350}
01633 , { 1350, 1350, 1350, 1350, 1350}
01634 , { 1350, 1350, 1350, 1350, 1350}
01635 , { 1350, 1350, 1350, 1350, 1350}
01636 , { 1350, 1350, 1350, 1350, 1350}
01637 }
01638 , { { 1350, 780, 1350, 780, 1350}
01639 , { 780, 780, 780, 780, 780}
01640 , { 1350, 780, 1350, 780, 1350}
01641 , { 780, 780, 780, 780, 780}
01642 , { 1350, 780, 1350, 780, 1350}
01643 }
01644 , { { 1350, 1350, 1350, 1350, 340}
01645 , { 1350, 1350, 1350, 1350, 340}
01646 , { 1350, 1350, 1350, 1350, 340}
01647 , { 1350, 1350, 1350, 1350, 340}
01648 , { 340, 340, 340, 340, 340}
01649 }
01650 }
01651 , { { { 1350, 1350, 1350, 1350, 1350}
01652 , { 1350, 1350, 1350, 1350, 1350}
01653 , { 1350, 1350, 1350, 1350, 1350}
01654 , { 1350, 1350, 1350, 1350, 1350}
01655 , { 1350, 1350, 1350, 1350, 1350}
01656 }
01657 , { { 1350, 1350, 1350, 780, 1350}
01658 , { 1350, 1350, 1350, 780, 1350}
01659 , { 1350, 1350, 1350, 780, 1350}
01660 , { 780, 780, 780, 780, 780}
01661 , { 1350, 1350, 1350, 780, 1350}
01662 }
01663 , { { 1350, 1350, 1350, 1350, 1350}
01664 , { 1350, 1350, 1350, 1350, 1350}
01665 , { 1350, 1350, 1350, 1350, 1350}
01666 , { 1350, 1350, 1350, 1350, 1350}
01667 , { 1350, 1350, 1350, 1350, 1350}
01668 }
01669 , { { 1350, 780, 1350, 780, 1350}
01670 , { 1350, 780, 1350, 780, 1350}
01671 , { 1350, 780, 1350, 780, 1350}
01672 , { 780, 780, 780, 780, 780}
01673 , { 1350, 780, 1350, 780, 1350}
01674 }
01675 , { { 1350, 1350, 1350, 1350, 340}
01676 , { 1350, 1350, 1350, 1350, 340}
01677 , { 1350, 1350, 1350, 1350, 340}
01678 , { 1350, 1350, 1350, 1350, 340}
01679 , { 340, 340, 340, 340, 340}
01680 }
01681 }
01682 , { { { 1350, 1350, 1350, 1350, 1350}
01683 , { 1350, 1350, 1350, 1350, 1350}
01684 , { 1350, 1350, 1350, 1350, 1350}
01685 , { 1350, 1350, 1350, 1350, 1350}
01686 , { 1350, 1350, 1350, 1350, 1350}
01687 }
01688 , { { 1350, 1350, 1350, 780, 1350}
01689 , { 1350, 1350, 1350, 780, 1350}
01690 , { 1350, 1350, 1350, 780, 1350}
01691 , { 780, 780, 780, 780, 780}
01692 , { 1350, 1350, 1350, 780, 1350}
01693 }
01694 , { { 1350, 1350, 1350, 1350, 1350}
01695 , { 1350, 1350, 1350, 1350, 1350}
01696 , { 1350, 1350, 1350, 1350, 1350}
01697 , { 1350, 1350, 1350, 1350, 1350}
01698 , { 1350, 1350, 1350, 1350, 1350}
01699 }
01700 , { { 1350, 780, 1350, 780, 1350}
01701 , { 780, 780, 780, 780, 780}
01702 , { 1350, 780, 1350, 780, 1350}
01703 , { 780, 780, 780, 780, 780}
01704 , { 1350, 780, 1350, 780, 1350}
01705 }
01706 , { { 1350, 1350, 1350, 1350, 340}
01707 , { 1350, 1350, 1350, 1350, 340}
01708 , { 1350, 1350, 1350, 1350, 340}
```

```
01709 , { 1350, 1350, 1350, 1350, 340}
01710 , { 340, 340, 340, 340, 340}
01711 }
01712 }
01713 , { { 1350, 1350, 1350, 1350, 1350}
01714 , { 1350, 1350, 1350, 1350, 1350}
01715 , { 1350, 1350, 1350, 1350, 1350}
01716 , { 1350, 1350, 1350, 1350, 1350}
01717 , { 1350, 1350, 1350, 1350, 1350}
01718 }
01719 , { { 1350, 1350, 1350, 780, 1350}
01720 , { 1350, 1350, 1350, 780, 1350}
01721 , { 1350, 1350, 1350, 780, 1350}
01722 , { 780, 780, 780, 780, 780}
01723 , { 1350, 1350, 1350, 780, 1350}
01724 }
01725 , { { 1350, 1350, 1350, 1350, 1350}
01726 , { 1350, 1350, 1350, 1350, 1350}
01727 , { 1350, 1350, 1350, 1350, 1350}
01728 , { 1350, 1350, 1350, 1350, 1350}
01729 , { 1350, 1350, 1350, 1350, 1350}
01730 }
01731 , { { 1350, 780, 1350, 780, 1350}
01732 , { 1350, 780, 1350, 780, 1350}
01733 , { 1350, 780, 1350, 780, 1350}
01734 , { 780, 780, 780, 780, 780}
01735 , { 1350, 780, 1350, 780, 1350}
01736 }
01737 , { { 1350, 1350, 1350, 1350, 340}
01738 , { 1350, 1350, 1350, 1350, 340}
01739 , { 1350, 1350, 1350, 1350, 340}
01740 , { 1350, 1350, 1350, 1350, 340}
01741 , { 340, 340, 340, 340, 340}
01742 }
01743 }
01744 }
01745 , { { { INF, INF, INF, INF, INF}
01746 , { INF, INF, INF, INF, INF}
01747 , { INF, INF, INF, INF, INF}
01748 , { INF, INF, INF, INF, INF}
01749 , { INF, INF, INF, INF, INF}
01750 }
01751 , { { INF, INF, INF, INF, INF}
01752 , { INF, INF, INF, INF, INF}
01753 , { INF, INF, INF, INF, INF}
01754 , { INF, INF, INF, INF, INF}
01755 , { INF, INF, INF, INF, INF}
01756 }
01757 , { { INF, INF, INF, INF, INF}
01758 , { INF, INF, INF, INF, INF}
01759 , { INF, INF, INF, INF, INF}
01760 , { INF, INF, INF, INF, INF}
01761 , { INF, INF, INF, INF, INF}
01762 }
01763 , { { INF, INF, INF, INF, INF}
01764 , { INF, INF, INF, INF, INF}
01765 , { INF, INF, INF, INF, INF}
01766 , { INF, INF, INF, INF, INF}
01767 , { INF, INF, INF, INF, INF}
01768 }
01769 , { { INF, INF, INF, INF, INF}
01770 , { INF, INF, INF, INF, INF}
01771 , { INF, INF, INF, INF, INF}
01772 , { INF, INF, INF, INF, INF}
01773 , { INF, INF, INF, INF, INF}
01774 }
01775 }
01776 , { { { 850, 850, 850, 850, 850}
01777 , { 850, 850, 850, 850, 850}
01778 , { 850, 850, 850, 850, 850}
01779 , { 850, 850, 850, 850, 850}
01780 , { 850, 850, 850, 850, 850}
01781 }
01782 , { { 850, 850, 850, 850, 850}
01783 , { 850, 850, 850, 850, 850}
01784 , { 850, 850, 850, 850, 850}
01785 , { 280, 280, 280, 280, 280}
01786 , { 850, 850, 850, 850, 850}
01787 }
01788 , { { 850, 850, 850, 850, 850}
01789 , { 850, 850, 850, 850, 850}
01790 , { 850, 850, 850, 850, 850}
01791 , { 850, 850, 850, 850, 850}
01792 , { 850, 850, 850, 850, 850}
01793 }
01794 , { { 850, 280, 850, 280, 850}
01795 , { 280, 280, 280, 280, 280}
```

```
01796 , { 850, 280, 850, 280, 850}
01797 , { 280, 280, 280, 280, 280}
01798 , { 850, 280, 850, 280, 850}
01799 }
01800 , { { 850, 850, 850, 850, -160}
01801 , { 850, 850, 850, 850, -160}
01802 , { 850, 850, 850, 850, -160}
01803 , { 850, 850, 850, 850, -160}
01804 , { -160, -160, -160, -160, -160}
01805 }
01806 }
01807 , { { { 850, 850, 850, 850, 850}
01808 , { 850, 850, 850, 850, 850}
01809 , { 850, 850, 850, 850, 850}
01810 , { 850, 850, 850, 850, 850}
01811 , { 850, 850, 850, 850, 850}
01812 }
01813 , { { 850, 850, 850, 850, 850}
01814 , { 850, 850, 850, 850, 850}
01815 , { 850, 850, 850, 850, 850}
01816 , { 280, 280, 280, 280, 280}
01817 , { 850, 850, 850, 850, 850}
01818 }
01819 , { { 850, 850, 850, 850, 850}
01820 , { 850, 850, 850, 850, 850}
01821 , { 850, 850, 850, 850, 850}
01822 , { 850, 850, 850, 850, 850}
01823 , { 850, 850, 850, 850, 850}
01824 }
01825 , { { 850, 280, 850, 280, 850}
01826 , { 850, 280, 850, 280, 850}
01827 , { 850, 280, 850, 280, 850}
01828 , { 280, 280, 280, 280, 280}
01829 , { 850, 280, 850, 280, 850}
01830 }
01831 , { { 850, 850, 850, 850, -160}
01832 , { 850, 850, 850, 850, -160}
01833 , { 850, 850, 850, 850, -160}
01834 , { 850, 850, 850, 850, -160}
01835 , { -160, -160, -160, -160, -160}
01836 }
01837 }
01838 , { { { 1350, 1350, 1350, 1350, 1350}
01839 , { 1350, 1350, 1350, 1350, 1350}
01840 , { 1350, 1350, 1350, 1350, 1350}
01841 , { 1350, 1350, 1350, 1350, 1350}
01842 , { 1350, 1350, 1350, 1350, 1350}
01843 }
01844 , { { 1350, 1350, 1350, 1350, 1350}
01845 , { 1350, 1350, 1350, 1350, 1350}
01846 , { 1350, 1350, 1350, 1350, 1350}
01847 , { 780, 780, 780, 780, 780}
01848 , { 1350, 1350, 1350, 1350, 1350}
01849 }
01850 , { { 1350, 1350, 1350, 1350, 1350}
01851 , { 1350, 1350, 1350, 1350, 1350}
01852 , { 1350, 1350, 1350, 1350, 1350}
01853 , { 1350, 1350, 1350, 1350, 1350}
01854 , { 1350, 1350, 1350, 1350, 1350}
01855 }
01856 , { { 1350, 780, 1350, 780, 1350}
01857 , { 1350, 780, 1350, 780, 1350}
01858 , { 1350, 780, 1350, 780, 1350}
01859 , { 780, 780, 780, 780, 780}
01860 , { 1350, 780, 1350, 780, 1350}
01861 }
01862 , { { 1350, 1350, 1350, 1350, 340}
01863 , { 1350, 1350, 1350, 1350, 340}
01864 , { 1350, 1350, 1350, 1350, 340}
01865 , { 1350, 1350, 1350, 1350, 340}
01866 , { 340, 340, 340, 340, 340}
01867 }
01868 }
01869 , { { { 1350, 1350, 1350, 1350, 1350}
01870 , { 1350, 1350, 1350, 1350, 1350}
01871 , { 1350, 1350, 1350, 1350, 1350}
01872 , { 1350, 1350, 1350, 1350, 1350}
01873 , { 1350, 1350, 1350, 1350, 1350}
01874 }
01875 , { { 1350, 1350, 1350, 1350, 1350}
01876 , { 1350, 1350, 1350, 1350, 1350}
01877 , { 1350, 1350, 1350, 1350, 1350}
01878 , { 780, 780, 780, 780, 780}
01879 , { 1350, 1350, 1350, 1350, 1350}
01880 }
01881 , { { 1350, 1350, 1350, 1350, 1350}
01882 , { 1350, 1350, 1350, 1350, 1350}
```

```
01883 , { 1350, 1350, 1350, 1350, 1350}
01884 , { 1350, 1350, 1350, 1350, 1350}
01885 , { 1350, 1350, 1350, 1350, 1350}
01886 }
01887 , { { 1350, 780, 1350, 780, 1350}
01888 , { 780, 780, 780, 780, 780}
01889 , { 1350, 780, 1350, 780, 1350}
01890 , { 780, 780, 780, 780, 780}
01891 , { 1350, 780, 1350, 780, 1350}
01892 }
01893 , { { 1350, 1350, 1350, 1350, 340}
01894 , { 1350, 1350, 1350, 1350, 340}
01895 , { 1350, 1350, 1350, 1350, 340}
01896 , { 1350, 1350, 1350, 1350, 340}
01897 , { 340, 340, 340, 340, 340}
01898 }
01899 }
01900 , { { { 1350, 1350, 1350, 1350, 1350}
01901 , { 1350, 1350, 1350, 1350, 1350}
01902 , { 1350, 1350, 1350, 1350, 1350}
01903 , { 1350, 1350, 1350, 1350, 1350}
01904 , { 1350, 1350, 1350, 1350, 1350}
01905 }
01906 , { { 1350, 1350, 1350, 1350, 1350}
01907 , { 1350, 1350, 1350, 1350, 1350}
01908 , { 1350, 1350, 1350, 1350, 1350}
01909 , { 780, 780, 780, 780, 780}
01910 , { 1350, 1350, 1350, 1350, 1350}
01911 }
01912 , { { 1350, 1350, 1350, 1350, 1350}
01913 , { 1350, 1350, 1350, 1350, 1350}
01914 , { 1350, 1350, 1350, 1350, 1350}
01915 , { 1350, 1350, 1350, 1350, 1350}
01916 , { 1350, 1350, 1350, 1350, 1350}
01917 }
01918 , { { 1350, 780, 1350, 780, 1350}
01919 , { 1350, 780, 1350, 780, 1350}
01920 , { 1350, 780, 1350, 780, 1350}
01921 , { 780, 780, 780, 780, 780}
01922 , { 1350, 780, 1350, 780, 1350}
01923 }
01924 , { { 1350, 1350, 1350, 1350, 340}
01925 , { 1350, 1350, 1350, 1350, 340}
01926 , { 1350, 1350, 1350, 1350, 340}
01927 , { 1350, 1350, 1350, 1350, 340}
01928 , { 340, 340, 340, 340, 340}
01929 }
01930 }
01931 , { { { 1350, 1350, 1350, 1350, 1350}
01932 , { 1350, 1350, 1350, 1350, 1350}
01933 , { 1350, 1350, 1350, 1350, 1350}
01934 , { 1350, 1350, 1350, 1350, 1350}
01935 , { 1350, 1350, 1350, 1350, 1350}
01936 }
01937 , { { 1350, 1350, 1350, 1350, 1350}
01938 , { 1350, 1350, 1350, 1350, 1350}
01939 , { 1350, 1350, 1350, 1350, 1350}
01940 , { 780, 780, 780, 780, 780}
01941 , { 1350, 1350, 1350, 1350, 1350}
01942 }
01943 , { { 1350, 1350, 1350, 1350, 1350}
01944 , { 1350, 1350, 1350, 1350, 1350}
01945 , { 1350, 1350, 1350, 1350, 1350}
01946 , { 1350, 1350, 1350, 1350, 1350}
01947 , { 1350, 1350, 1350, 1350, 1350}
01948 }
01949 , { { 1350, 780, 1350, 780, 1350}
01950 , { 780, 780, 780, 780, 780}
01951 , { 1350, 780, 1350, 780, 1350}
01952 , { 780, 780, 780, 780, 780}
01953 , { 1350, 780, 1350, 780, 1350}
01954 }
01955 , { { 1350, 1350, 1350, 1350, 340}
01956 , { 1350, 1350, 1350, 1350, 340}
01957 , { 1350, 1350, 1350, 1350, 340}
01958 , { 1350, 1350, 1350, 1350, 340}
01959 , { 340, 340, 340, 340, 340}
01960 }
01961 }
01962 , { { { 1350, 1350, 1350, 1350, 1350}
01963 , { 1350, 1350, 1350, 1350, 1350}
01964 , { 1350, 1350, 1350, 1350, 1350}
01965 , { 1350, 1350, 1350, 1350, 1350}
01966 , { 1350, 1350, 1350, 1350, 1350}
01967 }
01968 , { { 1350, 1350, 1350, 1350, 1350}
01969 , { 1350, 1350, 1350, 1350, 1350}
```



```

01970     , { 1350, 1350, 1350, 1350, 1350 }
01971     , { 780, 780, 780, 780, 780 }
01972     , { 1350, 1350, 1350, 1350, 1350 }
01973     }
01974     , { { 1350, 1350, 1350, 1350, 1350 }
01975     , { 1350, 1350, 1350, 1350, 1350 }
01976     , { 1350, 1350, 1350, 1350, 1350 }
01977     , { 1350, 1350, 1350, 1350, 1350 }
01978     , { 1350, 1350, 1350, 1350, 1350 }
01979     }
01980     , { { 1350, 780, 1350, 780, 1350 }
01981     , { 1350, 780, 1350, 780, 1350 }
01982     , { 1350, 780, 1350, 780, 1350 }
01983     , { 780, 780, 780, 780, 780 }
01984     , { 1350, 780, 1350, 780, 1350 }
01985     }
01986     , { { 1350, 1350, 1350, 1350, 340 }
01987     , { 1350, 1350, 1350, 1350, 340 }
01988     , { 1350, 1350, 1350, 1350, 340 }
01989     , { 1350, 1350, 1350, 1350, 340 }
01990     , { 340, 340, 340, 340, 340 }
01991     }
01992     }
01993     };;

```

18.174 intl22.h

```

00001 PUBLIC int int22_37[NBPAIRS+1][NBPAIRS+1][5][5][5][5] =
00002 {{{{{ INF, INF, INF, INF, INF }
00003     , { INF, INF, INF, INF, INF }
00004     , { INF, INF, INF, INF, INF }
00005     , { INF, INF, INF, INF, INF }
00006     , { INF, INF, INF, INF, INF }
00007     }
00008     , {{{ INF, INF, INF, INF, INF }
00009     , { INF, INF, INF, INF, INF }
00010     , { INF, INF, INF, INF, INF }
00011     , { INF, INF, INF, INF, INF }
00012     , { INF, INF, INF, INF, INF }
00013     }
00014     , {{{ INF, INF, INF, INF, INF }
00015     , { INF, INF, INF, INF, INF }
00016     , { INF, INF, INF, INF, INF }
00017     , { INF, INF, INF, INF, INF }
00018     , { INF, INF, INF, INF, INF }
00019     }
00020     , {{{ INF, INF, INF, INF, INF }
00021     , { INF, INF, INF, INF, INF }
00022     , { INF, INF, INF, INF, INF }
00023     , { INF, INF, INF, INF, INF }
00024     , { INF, INF, INF, INF, INF }
00025     }
00026     , {{{ INF, INF, INF, INF, INF }
00027     , { INF, INF, INF, INF, INF }
00028     , { INF, INF, INF, INF, INF }
00029     , { INF, INF, INF, INF, INF }
00030     , { INF, INF, INF, INF, INF }
00031     }
00032     }
00033     , {{{ INF, INF, INF, INF, INF }
00034     , { INF, INF, INF, INF, INF }
00035     , { INF, INF, INF, INF, INF }
00036     , { INF, INF, INF, INF, INF }
00037     , { INF, INF, INF, INF, INF }
00038     }
00039     , {{{ INF, INF, INF, INF, INF }
00040     , { INF, INF, INF, INF, INF }
00041     , { INF, INF, INF, INF, INF }
00042     , { INF, INF, INF, INF, INF }
00043     , { INF, INF, INF, INF, INF }
00044     }
00045     , {{{ INF, INF, INF, INF, INF }
00046     , { INF, INF, INF, INF, INF }
00047     , { INF, INF, INF, INF, INF }
00048     , { INF, INF, INF, INF, INF }
00049     , { INF, INF, INF, INF, INF }
00050     }
00051     , {{{ INF, INF, INF, INF, INF }
00052     , { INF, INF, INF, INF, INF }
00053     , { INF, INF, INF, INF, INF }
00054     , { INF, INF, INF, INF, INF }
00055     , { INF, INF, INF, INF, INF }
00056     }
00057     , {{{ INF, INF, INF, INF, INF }
00058     , { INF, INF, INF, INF, INF }

```

```
00059      , {   INF,   INF,   INF,   INF,   INF }
00060      , {   INF,   INF,   INF,   INF,   INF }
00061      , {   INF,   INF,   INF,   INF,   INF }
00062      }
00063      }
00064      , { {   INF,   INF,   INF,   INF,   INF }
00065      , {   INF,   INF,   INF,   INF,   INF }
00066      , {   INF,   INF,   INF,   INF,   INF }
00067      , {   INF,   INF,   INF,   INF,   INF }
00068      , {   INF,   INF,   INF,   INF,   INF }
00069      }
00070      , { {   INF,   INF,   INF,   INF,   INF }
00071      , {   INF,   INF,   INF,   INF,   INF }
00072      , {   INF,   INF,   INF,   INF,   INF }
00073      , {   INF,   INF,   INF,   INF,   INF }
00074      , {   INF,   INF,   INF,   INF,   INF }
00075      }
00076      , { {   INF,   INF,   INF,   INF,   INF }
00077      , {   INF,   INF,   INF,   INF,   INF }
00078      , {   INF,   INF,   INF,   INF,   INF }
00079      , {   INF,   INF,   INF,   INF,   INF }
00080      , {   INF,   INF,   INF,   INF,   INF }
00081      }
00082      , { {   INF,   INF,   INF,   INF,   INF }
00083      , {   INF,   INF,   INF,   INF,   INF }
00084      , {   INF,   INF,   INF,   INF,   INF }
00085      , {   INF,   INF,   INF,   INF,   INF }
00086      , {   INF,   INF,   INF,   INF,   INF }
00087      }
00088      , { {   INF,   INF,   INF,   INF,   INF }
00089      , {   INF,   INF,   INF,   INF,   INF }
00090      , {   INF,   INF,   INF,   INF,   INF }
00091      , {   INF,   INF,   INF,   INF,   INF }
00092      , {   INF,   INF,   INF,   INF,   INF }
00093      }
00094      }
00095      , { { {   INF,   INF,   INF,   INF,   INF }
00096      , {   INF,   INF,   INF,   INF,   INF }
00097      , {   INF,   INF,   INF,   INF,   INF }
00098      , {   INF,   INF,   INF,   INF,   INF }
00099      , {   INF,   INF,   INF,   INF,   INF }
00100      }
00101      , { {   INF,   INF,   INF,   INF,   INF }
00102      , {   INF,   INF,   INF,   INF,   INF }
00103      , {   INF,   INF,   INF,   INF,   INF }
00104      , {   INF,   INF,   INF,   INF,   INF }
00105      , {   INF,   INF,   INF,   INF,   INF }
00106      }
00107      , { {   INF,   INF,   INF,   INF,   INF }
00108      , {   INF,   INF,   INF,   INF,   INF }
00109      , {   INF,   INF,   INF,   INF,   INF }
00110      , {   INF,   INF,   INF,   INF,   INF }
00111      , {   INF,   INF,   INF,   INF,   INF }
00112      }
00113      , { {   INF,   INF,   INF,   INF,   INF }
00114      , {   INF,   INF,   INF,   INF,   INF }
00115      , {   INF,   INF,   INF,   INF,   INF }
00116      , {   INF,   INF,   INF,   INF,   INF }
00117      , {   INF,   INF,   INF,   INF,   INF }
00118      }
00119      , { {   INF,   INF,   INF,   INF,   INF }
00120      , {   INF,   INF,   INF,   INF,   INF }
00121      , {   INF,   INF,   INF,   INF,   INF }
00122      , {   INF,   INF,   INF,   INF,   INF }
00123      , {   INF,   INF,   INF,   INF,   INF }
00124      }
00125      }
00126      , { { {   INF,   INF,   INF,   INF,   INF }
00127      , {   INF,   INF,   INF,   INF,   INF }
00128      , {   INF,   INF,   INF,   INF,   INF }
00129      , {   INF,   INF,   INF,   INF,   INF }
00130      , {   INF,   INF,   INF,   INF,   INF }
00131      }
00132      , { {   INF,   INF,   INF,   INF,   INF }
00133      , {   INF,   INF,   INF,   INF,   INF }
00134      , {   INF,   INF,   INF,   INF,   INF }
00135      , {   INF,   INF,   INF,   INF,   INF }
00136      , {   INF,   INF,   INF,   INF,   INF }
00137      }
00138      , { {   INF,   INF,   INF,   INF,   INF }
00139      , {   INF,   INF,   INF,   INF,   INF }
00140      , {   INF,   INF,   INF,   INF,   INF }
00141      , {   INF,   INF,   INF,   INF,   INF }
00142      , {   INF,   INF,   INF,   INF,   INF }
00143      }
00144      , { {   INF,   INF,   INF,   INF,   INF }
00145      , {   INF,   INF,   INF,   INF,   INF }
```

```
00146     , {   INF,   INF,   INF,   INF,   INF }
00147     , {   INF,   INF,   INF,   INF,   INF }
00148     , {   INF,   INF,   INF,   INF,   INF }
00149     }
00150     , { {   INF,   INF,   INF,   INF,   INF }
00151     , {   INF,   INF,   INF,   INF,   INF }
00152     , {   INF,   INF,   INF,   INF,   INF }
00153     , {   INF,   INF,   INF,   INF,   INF }
00154     , {   INF,   INF,   INF,   INF,   INF }
00155     }
00156     }
00157     }
00158     , { { {   INF,   INF,   INF,   INF,   INF }
00159     , {   INF,   INF,   INF,   INF,   INF }
00160     , {   INF,   INF,   INF,   INF,   INF }
00161     , {   INF,   INF,   INF,   INF,   INF }
00162     , {   INF,   INF,   INF,   INF,   INF }
00163     }
00164     , { {   INF,   INF,   INF,   INF,   INF }
00165     , {   INF,   INF,   INF,   INF,   INF }
00166     , {   INF,   INF,   INF,   INF,   INF }
00167     , {   INF,   INF,   INF,   INF,   INF }
00168     , {   INF,   INF,   INF,   INF,   INF }
00169     }
00170     , { {   INF,   INF,   INF,   INF,   INF }
00171     , {   INF,   INF,   INF,   INF,   INF }
00172     , {   INF,   INF,   INF,   INF,   INF }
00173     , {   INF,   INF,   INF,   INF,   INF }
00174     , {   INF,   INF,   INF,   INF,   INF }
00175     }
00176     , { {   INF,   INF,   INF,   INF,   INF }
00177     , {   INF,   INF,   INF,   INF,   INF }
00178     , {   INF,   INF,   INF,   INF,   INF }
00179     , {   INF,   INF,   INF,   INF,   INF }
00180     , {   INF,   INF,   INF,   INF,   INF }
00181     }
00182     , { {   INF,   INF,   INF,   INF,   INF }
00183     , {   INF,   INF,   INF,   INF,   INF }
00184     , {   INF,   INF,   INF,   INF,   INF }
00185     , {   INF,   INF,   INF,   INF,   INF }
00186     , {   INF,   INF,   INF,   INF,   INF }
00187     }
00188     }
00189     , { { {   INF,   INF,   INF,   INF,   INF }
00190     , {   INF,   INF,   INF,   INF,   INF }
00191     , {   INF,   INF,   INF,   INF,   INF }
00192     , {   INF,   INF,   INF,   INF,   INF }
00193     , {   INF,   INF,   INF,   INF,   INF }
00194     }
00195     , { {   INF,   INF,   INF,   INF,   INF }
00196     , {   INF,   INF,   INF,   INF,   INF }
00197     , {   INF,   INF,   INF,   INF,   INF }
00198     , {   INF,   INF,   INF,   INF,   INF }
00199     , {   INF,   INF,   INF,   INF,   INF }
00200     }
00201     , { {   INF,   INF,   INF,   INF,   INF }
00202     , {   INF,   INF,   INF,   INF,   INF }
00203     , {   INF,   INF,   INF,   INF,   INF }
00204     , {   INF,   INF,   INF,   INF,   INF }
00205     , {   INF,   INF,   INF,   INF,   INF }
00206     }
00207     , { {   INF,   INF,   INF,   INF,   INF }
00208     , {   INF,   INF,   INF,   INF,   INF }
00209     , {   INF,   INF,   INF,   INF,   INF }
00210     , {   INF,   INF,   INF,   INF,   INF }
00211     , {   INF,   INF,   INF,   INF,   INF }
00212     }
00213     , { {   INF,   INF,   INF,   INF,   INF }
00214     , {   INF,   INF,   INF,   INF,   INF }
00215     , {   INF,   INF,   INF,   INF,   INF }
00216     , {   INF,   INF,   INF,   INF,   INF }
00217     , {   INF,   INF,   INF,   INF,   INF }
00218     }
00219     }
00220     , { { {   INF,   INF,   INF,   INF,   INF }
00221     , {   INF,   INF,   INF,   INF,   INF }
00222     , {   INF,   INF,   INF,   INF,   INF }
00223     , {   INF,   INF,   INF,   INF,   INF }
00224     , {   INF,   INF,   INF,   INF,   INF }
00225     }
00226     , { {   INF,   INF,   INF,   INF,   INF }
00227     , {   INF,   INF,   INF,   INF,   INF }
00228     , {   INF,   INF,   INF,   INF,   INF }
00229     , {   INF,   INF,   INF,   INF,   INF }
00230     , {   INF,   INF,   INF,   INF,   INF }
00231     }
00232     , { {   INF,   INF,   INF,   INF,   INF }
```

```
00233      , {   INF,   INF,   INF,   INF,   INF }
00234      , {   INF,   INF,   INF,   INF,   INF }
00235      , {   INF,   INF,   INF,   INF,   INF }
00236      , {   INF,   INF,   INF,   INF,   INF }
00237      }
00238      , { {   INF,   INF,   INF,   INF,   INF }
00239      , {   INF,   INF,   INF,   INF,   INF }
00240      , {   INF,   INF,   INF,   INF,   INF }
00241      , {   INF,   INF,   INF,   INF,   INF }
00242      , {   INF,   INF,   INF,   INF,   INF }
00243      }
00244      , { {   INF,   INF,   INF,   INF,   INF }
00245      , {   INF,   INF,   INF,   INF,   INF }
00246      , {   INF,   INF,   INF,   INF,   INF }
00247      , {   INF,   INF,   INF,   INF,   INF }
00248      , {   INF,   INF,   INF,   INF,   INF }
00249      }
00250      }
00251      , { { {   INF,   INF,   INF,   INF,   INF }
00252      , {   INF,   INF,   INF,   INF,   INF }
00253      , {   INF,   INF,   INF,   INF,   INF }
00254      , {   INF,   INF,   INF,   INF,   INF }
00255      , {   INF,   INF,   INF,   INF,   INF }
00256      }
00257      , { {   INF,   INF,   INF,   INF,   INF }
00258      , {   INF,   INF,   INF,   INF,   INF }
00259      , {   INF,   INF,   INF,   INF,   INF }
00260      , {   INF,   INF,   INF,   INF,   INF }
00261      , {   INF,   INF,   INF,   INF,   INF }
00262      }
00263      , { {   INF,   INF,   INF,   INF,   INF }
00264      , {   INF,   INF,   INF,   INF,   INF }
00265      , {   INF,   INF,   INF,   INF,   INF }
00266      , {   INF,   INF,   INF,   INF,   INF }
00267      , {   INF,   INF,   INF,   INF,   INF }
00268      }
00269      , { {   INF,   INF,   INF,   INF,   INF }
00270      , {   INF,   INF,   INF,   INF,   INF }
00271      , {   INF,   INF,   INF,   INF,   INF }
00272      , {   INF,   INF,   INF,   INF,   INF }
00273      , {   INF,   INF,   INF,   INF,   INF }
00274      }
00275      , { {   INF,   INF,   INF,   INF,   INF }
00276      , {   INF,   INF,   INF,   INF,   INF }
00277      , {   INF,   INF,   INF,   INF,   INF }
00278      , {   INF,   INF,   INF,   INF,   INF }
00279      , {   INF,   INF,   INF,   INF,   INF }
00280      }
00281      }
00282      , { { {   INF,   INF,   INF,   INF,   INF }
00283      , {   INF,   INF,   INF,   INF,   INF }
00284      , {   INF,   INF,   INF,   INF,   INF }
00285      , {   INF,   INF,   INF,   INF,   INF }
00286      , {   INF,   INF,   INF,   INF,   INF }
00287      }
00288      , { {   INF,   INF,   INF,   INF,   INF }
00289      , {   INF,   INF,   INF,   INF,   INF }
00290      , {   INF,   INF,   INF,   INF,   INF }
00291      , {   INF,   INF,   INF,   INF,   INF }
00292      , {   INF,   INF,   INF,   INF,   INF }
00293      }
00294      , { {   INF,   INF,   INF,   INF,   INF }
00295      , {   INF,   INF,   INF,   INF,   INF }
00296      , {   INF,   INF,   INF,   INF,   INF }
00297      , {   INF,   INF,   INF,   INF,   INF }
00298      , {   INF,   INF,   INF,   INF,   INF }
00299      }
00300      , { {   INF,   INF,   INF,   INF,   INF }
00301      , {   INF,   INF,   INF,   INF,   INF }
00302      , {   INF,   INF,   INF,   INF,   INF }
00303      , {   INF,   INF,   INF,   INF,   INF }
00304      , {   INF,   INF,   INF,   INF,   INF }
00305      }
00306      , { {   INF,   INF,   INF,   INF,   INF }
00307      , {   INF,   INF,   INF,   INF,   INF }
00308      , {   INF,   INF,   INF,   INF,   INF }
00309      , {   INF,   INF,   INF,   INF,   INF }
00310      , {   INF,   INF,   INF,   INF,   INF }
00311      }
00312      }
00313      }
00314      , { { { {   INF,   INF,   INF,   INF,   INF }
00315      , {   INF,   INF,   INF,   INF,   INF }
00316      , {   INF,   INF,   INF,   INF,   INF }
00317      , {   INF,   INF,   INF,   INF,   INF }
00318      , {   INF,   INF,   INF,   INF,   INF }
00319      }
```

```
00320 ,{{ INF, INF, INF, INF, INF}
00321 ,{ INF, INF, INF, INF, INF}
00322 ,{ INF, INF, INF, INF, INF}
00323 ,{ INF, INF, INF, INF, INF}
00324 ,{ INF, INF, INF, INF, INF}
00325 }
00326 ,{{ INF, INF, INF, INF, INF}
00327 ,{ INF, INF, INF, INF, INF}
00328 ,{ INF, INF, INF, INF, INF}
00329 ,{ INF, INF, INF, INF, INF}
00330 ,{ INF, INF, INF, INF, INF}
00331 }
00332 ,{{ INF, INF, INF, INF, INF}
00333 ,{ INF, INF, INF, INF, INF}
00334 ,{ INF, INF, INF, INF, INF}
00335 ,{ INF, INF, INF, INF, INF}
00336 ,{ INF, INF, INF, INF, INF}
00337 }
00338 ,{{ INF, INF, INF, INF, INF}
00339 ,{ INF, INF, INF, INF, INF}
00340 ,{ INF, INF, INF, INF, INF}
00341 ,{ INF, INF, INF, INF, INF}
00342 ,{ INF, INF, INF, INF, INF}
00343 }
00344 }
00345 ,{{{ INF, INF, INF, INF, INF}
00346 ,{ INF, INF, INF, INF, INF}
00347 ,{ INF, INF, INF, INF, INF}
00348 ,{ INF, INF, INF, INF, INF}
00349 ,{ INF, INF, INF, INF, INF}
00350 }
00351 ,{{ INF, INF, INF, INF, INF}
00352 ,{ INF, INF, INF, INF, INF}
00353 ,{ INF, INF, INF, INF, INF}
00354 ,{ INF, INF, INF, INF, INF}
00355 ,{ INF, INF, INF, INF, INF}
00356 }
00357 ,{{ INF, INF, INF, INF, INF}
00358 ,{ INF, INF, INF, INF, INF}
00359 ,{ INF, INF, INF, INF, INF}
00360 ,{ INF, INF, INF, INF, INF}
00361 ,{ INF, INF, INF, INF, INF}
00362 }
00363 ,{{{ INF, INF, INF, INF, INF}
00364 ,{ INF, INF, INF, INF, INF}
00365 ,{ INF, INF, INF, INF, INF}
00366 ,{ INF, INF, INF, INF, INF}
00367 ,{ INF, INF, INF, INF, INF}
00368 }
00369 ,{{{ INF, INF, INF, INF, INF}
00370 ,{ INF, INF, INF, INF, INF}
00371 ,{ INF, INF, INF, INF, INF}
00372 ,{ INF, INF, INF, INF, INF}
00373 ,{ INF, INF, INF, INF, INF}
00374 }
00375 }
00376 ,{{{ INF, INF, INF, INF, INF}
00377 ,{ INF, INF, INF, INF, INF}
00378 ,{ INF, INF, INF, INF, INF}
00379 ,{ INF, INF, INF, INF, INF}
00380 ,{ INF, INF, INF, INF, INF}
00381 }
00382 ,{{{ INF, INF, INF, INF, INF}
00383 ,{ INF, INF, INF, INF, INF}
00384 ,{ INF, INF, INF, INF, INF}
00385 ,{ INF, INF, INF, INF, INF}
00386 ,{ INF, INF, INF, INF, INF}
00387 }
00388 ,{{{ INF, INF, INF, INF, INF}
00389 ,{ INF, INF, INF, INF, INF}
00390 ,{ INF, INF, INF, INF, INF}
00391 ,{ INF, INF, INF, INF, INF}
00392 ,{ INF, INF, INF, INF, INF}
00393 }
00394 ,{{{ INF, INF, INF, INF, INF}
00395 ,{ INF, INF, INF, INF, INF}
00396 ,{ INF, INF, INF, INF, INF}
00397 ,{ INF, INF, INF, INF, INF}
00398 ,{ INF, INF, INF, INF, INF}
00399 }
00400 ,{{{ INF, INF, INF, INF, INF}
00401 ,{ INF, INF, INF, INF, INF}
00402 ,{ INF, INF, INF, INF, INF}
00403 ,{ INF, INF, INF, INF, INF}
00404 ,{ INF, INF, INF, INF, INF}
00405 }
00406 }
```

```
00407 ,{{{ INF, INF, INF, INF, INF}
00408 ,{ INF, INF, INF, INF, INF}
00409 ,{ INF, INF, INF, INF, INF}
00410 ,{ INF, INF, INF, INF, INF}
00411 ,{ INF, INF, INF, INF, INF}
00412 }
00413 ,{{{ INF, INF, INF, INF, INF}
00414 ,{ INF, INF, INF, INF, INF}
00415 ,{ INF, INF, INF, INF, INF}
00416 ,{ INF, INF, INF, INF, INF}
00417 ,{ INF, INF, INF, INF, INF}
00418 }
00419 ,{{{ INF, INF, INF, INF, INF}
00420 ,{ INF, INF, INF, INF, INF}
00421 ,{ INF, INF, INF, INF, INF}
00422 ,{ INF, INF, INF, INF, INF}
00423 ,{ INF, INF, INF, INF, INF}
00424 }
00425 ,{{{ INF, INF, INF, INF, INF}
00426 ,{ INF, INF, INF, INF, INF}
00427 ,{ INF, INF, INF, INF, INF}
00428 ,{ INF, INF, INF, INF, INF}
00429 ,{ INF, INF, INF, INF, INF}
00430 }
00431 ,{{{ INF, INF, INF, INF, INF}
00432 ,{ INF, INF, INF, INF, INF}
00433 ,{ INF, INF, INF, INF, INF}
00434 ,{ INF, INF, INF, INF, INF}
00435 ,{ INF, INF, INF, INF, INF}
00436 }
00437 }
00438 ,{{{ INF, INF, INF, INF, INF}
00439 ,{ INF, INF, INF, INF, INF}
00440 ,{ INF, INF, INF, INF, INF}
00441 ,{ INF, INF, INF, INF, INF}
00442 ,{ INF, INF, INF, INF, INF}
00443 }
00444 ,{{{ INF, INF, INF, INF, INF}
00445 ,{ INF, INF, INF, INF, INF}
00446 ,{ INF, INF, INF, INF, INF}
00447 ,{ INF, INF, INF, INF, INF}
00448 ,{ INF, INF, INF, INF, INF}
00449 }
00450 ,{{{ INF, INF, INF, INF, INF}
00451 ,{ INF, INF, INF, INF, INF}
00452 ,{ INF, INF, INF, INF, INF}
00453 ,{ INF, INF, INF, INF, INF}
00454 ,{ INF, INF, INF, INF, INF}
00455 }
00456 ,{{{ INF, INF, INF, INF, INF}
00457 ,{ INF, INF, INF, INF, INF}
00458 ,{ INF, INF, INF, INF, INF}
00459 ,{ INF, INF, INF, INF, INF}
00460 ,{ INF, INF, INF, INF, INF}
00461 }
00462 ,{{{ INF, INF, INF, INF, INF}
00463 ,{ INF, INF, INF, INF, INF}
00464 ,{ INF, INF, INF, INF, INF}
00465 ,{ INF, INF, INF, INF, INF}
00466 ,{ INF, INF, INF, INF, INF}
00467 }
00468 }
00469 }
00470 ,{{{ INF, INF, INF, INF, INF}
00471 ,{ INF, INF, INF, INF, INF}
00472 ,{ INF, INF, INF, INF, INF}
00473 ,{ INF, INF, INF, INF, INF}
00474 ,{ INF, INF, INF, INF, INF}
00475 }
00476 ,{{{ INF, INF, INF, INF, INF}
00477 ,{ INF, INF, INF, INF, INF}
00478 ,{ INF, INF, INF, INF, INF}
00479 ,{ INF, INF, INF, INF, INF}
00480 ,{ INF, INF, INF, INF, INF}
00481 }
00482 ,{{{ INF, INF, INF, INF, INF}
00483 ,{ INF, INF, INF, INF, INF}
00484 ,{ INF, INF, INF, INF, INF}
00485 ,{ INF, INF, INF, INF, INF}
00486 ,{ INF, INF, INF, INF, INF}
00487 }
00488 ,{{{ INF, INF, INF, INF, INF}
00489 ,{ INF, INF, INF, INF, INF}
00490 ,{ INF, INF, INF, INF, INF}
00491 ,{ INF, INF, INF, INF, INF}
00492 ,{ INF, INF, INF, INF, INF}
00493 }
```

```
00494     , {{    INF,    INF,    INF,    INF,    INF}
00495     , {    INF,    INF,    INF,    INF,    INF}
00496     , {    INF,    INF,    INF,    INF,    INF}
00497     , {    INF,    INF,    INF,    INF,    INF}
00498     , {    INF,    INF,    INF,    INF,    INF}
00499     }
00500 }
00501 , {{ {    INF,    INF,    INF,    INF,    INF}
00502     , {    INF,    INF,    INF,    INF,    INF}
00503     , {    INF,    INF,    INF,    INF,    INF}
00504     , {    INF,    INF,    INF,    INF,    INF}
00505     , {    INF,    INF,    INF,    INF,    INF}
00506     }
00507 , {{ {    INF,    INF,    INF,    INF,    INF}
00508     , {    INF,    INF,    INF,    INF,    INF}
00509     , {    INF,    INF,    INF,    INF,    INF}
00510     , {    INF,    INF,    INF,    INF,    INF}
00511     , {    INF,    INF,    INF,    INF,    INF}
00512     }
00513 , {{ {    INF,    INF,    INF,    INF,    INF}
00514     , {    INF,    INF,    INF,    INF,    INF}
00515     , {    INF,    INF,    INF,    INF,    INF}
00516     , {    INF,    INF,    INF,    INF,    INF}
00517     , {    INF,    INF,    INF,    INF,    INF}
00518     }
00519 , {{ {    INF,    INF,    INF,    INF,    INF}
00520     , {    INF,    INF,    INF,    INF,    INF}
00521     , {    INF,    INF,    INF,    INF,    INF}
00522     , {    INF,    INF,    INF,    INF,    INF}
00523     , {    INF,    INF,    INF,    INF,    INF}
00524     }
00525 , {{ {    INF,    INF,    INF,    INF,    INF}
00526     , {    INF,    INF,    INF,    INF,    INF}
00527     , {    INF,    INF,    INF,    INF,    INF}
00528     , {    INF,    INF,    INF,    INF,    INF}
00529     , {    INF,    INF,    INF,    INF,    INF}
00530     }
00531 }
00532 , {{ {    INF,    INF,    INF,    INF,    INF}
00533     , {    INF,    INF,    INF,    INF,    INF}
00534     , {    INF,    INF,    INF,    INF,    INF}
00535     , {    INF,    INF,    INF,    INF,    INF}
00536     , {    INF,    INF,    INF,    INF,    INF}
00537     }
00538 , {{ {    INF,    INF,    INF,    INF,    INF}
00539     , {    INF,    INF,    INF,    INF,    INF}
00540     , {    INF,    INF,    INF,    INF,    INF}
00541     , {    INF,    INF,    INF,    INF,    INF}
00542     , {    INF,    INF,    INF,    INF,    INF}
00543     }
00544 , {{ {    INF,    INF,    INF,    INF,    INF}
00545     , {    INF,    INF,    INF,    INF,    INF}
00546     , {    INF,    INF,    INF,    INF,    INF}
00547     , {    INF,    INF,    INF,    INF,    INF}
00548     , {    INF,    INF,    INF,    INF,    INF}
00549     }
00550 , {{ {    INF,    INF,    INF,    INF,    INF}
00551     , {    INF,    INF,    INF,    INF,    INF}
00552     , {    INF,    INF,    INF,    INF,    INF}
00553     , {    INF,    INF,    INF,    INF,    INF}
00554     , {    INF,    INF,    INF,    INF,    INF}
00555     }
00556 , {{ {    INF,    INF,    INF,    INF,    INF}
00557     , {    INF,    INF,    INF,    INF,    INF}
00558     , {    INF,    INF,    INF,    INF,    INF}
00559     , {    INF,    INF,    INF,    INF,    INF}
00560     , {    INF,    INF,    INF,    INF,    INF}
00561     }
00562 }
00563 , {{ {    INF,    INF,    INF,    INF,    INF}
00564     , {    INF,    INF,    INF,    INF,    INF}
00565     , {    INF,    INF,    INF,    INF,    INF}
00566     , {    INF,    INF,    INF,    INF,    INF}
00567     , {    INF,    INF,    INF,    INF,    INF}
00568     }
00569 , {{ {    INF,    INF,    INF,    INF,    INF}
00570     , {    INF,    INF,    INF,    INF,    INF}
00571     , {    INF,    INF,    INF,    INF,    INF}
00572     , {    INF,    INF,    INF,    INF,    INF}
00573     , {    INF,    INF,    INF,    INF,    INF}
00574     }
00575 , {{ {    INF,    INF,    INF,    INF,    INF}
00576     , {    INF,    INF,    INF,    INF,    INF}
00577     , {    INF,    INF,    INF,    INF,    INF}
00578     , {    INF,    INF,    INF,    INF,    INF}
00579     , {    INF,    INF,    INF,    INF,    INF}
00580     }
```

```
00581 ,{{ INF, INF, INF, INF, INF}
00582 ,{ INF, INF, INF, INF, INF}
00583 ,{ INF, INF, INF, INF, INF}
00584 ,{ INF, INF, INF, INF, INF}
00585 ,{ INF, INF, INF, INF, INF}
00586 }
00587 ,{{ INF, INF, INF, INF, INF}
00588 ,{ INF, INF, INF, INF, INF}
00589 ,{ INF, INF, INF, INF, INF}
00590 ,{ INF, INF, INF, INF, INF}
00591 ,{ INF, INF, INF, INF, INF}
00592 }
00593 }
00594 ,{{{ INF, INF, INF, INF, INF}
00595 ,{ INF, INF, INF, INF, INF}
00596 ,{ INF, INF, INF, INF, INF}
00597 ,{ INF, INF, INF, INF, INF}
00598 ,{ INF, INF, INF, INF, INF}
00599 }
00600 ,{{ INF, INF, INF, INF, INF}
00601 ,{ INF, INF, INF, INF, INF}
00602 ,{ INF, INF, INF, INF, INF}
00603 ,{ INF, INF, INF, INF, INF}
00604 ,{ INF, INF, INF, INF, INF}
00605 }
00606 ,{{ INF, INF, INF, INF, INF}
00607 ,{ INF, INF, INF, INF, INF}
00608 ,{ INF, INF, INF, INF, INF}
00609 ,{ INF, INF, INF, INF, INF}
00610 ,{ INF, INF, INF, INF, INF}
00611 }
00612 ,{{ INF, INF, INF, INF, INF}
00613 ,{ INF, INF, INF, INF, INF}
00614 ,{ INF, INF, INF, INF, INF}
00615 ,{ INF, INF, INF, INF, INF}
00616 ,{ INF, INF, INF, INF, INF}
00617 }
00618 ,{{ INF, INF, INF, INF, INF}
00619 ,{ INF, INF, INF, INF, INF}
00620 ,{ INF, INF, INF, INF, INF}
00621 ,{ INF, INF, INF, INF, INF}
00622 ,{ INF, INF, INF, INF, INF}
00623 }
00624 }
00625 }
00626 ,{{{ INF, INF, INF, INF, INF}
00627 ,{ INF, INF, INF, INF, INF}
00628 ,{ INF, INF, INF, INF, INF}
00629 ,{ INF, INF, INF, INF, INF}
00630 ,{ INF, INF, INF, INF, INF}
00631 }
00632 ,{{ INF, INF, INF, INF, INF}
00633 ,{ INF, INF, INF, INF, INF}
00634 ,{ INF, INF, INF, INF, INF}
00635 ,{ INF, INF, INF, INF, INF}
00636 ,{ INF, INF, INF, INF, INF}
00637 }
00638 ,{{ INF, INF, INF, INF, INF}
00639 ,{ INF, INF, INF, INF, INF}
00640 ,{ INF, INF, INF, INF, INF}
00641 ,{ INF, INF, INF, INF, INF}
00642 ,{ INF, INF, INF, INF, INF}
00643 }
00644 ,{{ INF, INF, INF, INF, INF}
00645 ,{ INF, INF, INF, INF, INF}
00646 ,{ INF, INF, INF, INF, INF}
00647 ,{ INF, INF, INF, INF, INF}
00648 ,{ INF, INF, INF, INF, INF}
00649 }
00650 ,{{ INF, INF, INF, INF, INF}
00651 ,{ INF, INF, INF, INF, INF}
00652 ,{ INF, INF, INF, INF, INF}
00653 ,{ INF, INF, INF, INF, INF}
00654 ,{ INF, INF, INF, INF, INF}
00655 }
00656 }
00657 ,{{{ INF, INF, INF, INF, INF}
00658 ,{ INF, INF, INF, INF, INF}
00659 ,{ INF, INF, INF, INF, INF}
00660 ,{ INF, INF, INF, INF, INF}
00661 ,{ INF, INF, INF, INF, INF}
00662 }
00663 ,{{ INF, INF, INF, INF, INF}
00664 ,{ INF, INF, INF, INF, INF}
00665 ,{ INF, INF, INF, INF, INF}
00666 ,{ INF, INF, INF, INF, INF}
00667 ,{ INF, INF, INF, INF, INF}
```



```
00668     }
00669     , {{   INF,   INF,   INF,   INF,   INF }
00670     , {   INF,   INF,   INF,   INF,   INF }
00671     , {   INF,   INF,   INF,   INF,   INF }
00672     , {   INF,   INF,   INF,   INF,   INF }
00673     , {   INF,   INF,   INF,   INF,   INF }
00674     }
00675     , {{   INF,   INF,   INF,   INF,   INF }
00676     , {   INF,   INF,   INF,   INF,   INF }
00677     , {   INF,   INF,   INF,   INF,   INF }
00678     , {   INF,   INF,   INF,   INF,   INF }
00679     , {   INF,   INF,   INF,   INF,   INF }
00680     }
00681     , {{   INF,   INF,   INF,   INF,   INF }
00682     , {   INF,   INF,   INF,   INF,   INF }
00683     , {   INF,   INF,   INF,   INF,   INF }
00684     , {   INF,   INF,   INF,   INF,   INF }
00685     , {   INF,   INF,   INF,   INF,   INF }
00686     }
00687     }
00688     , {{{   INF,   INF,   INF,   INF,   INF }
00689     , {   INF,   INF,   INF,   INF,   INF }
00690     , {   INF,   INF,   INF,   INF,   INF }
00691     , {   INF,   INF,   INF,   INF,   INF }
00692     , {   INF,   INF,   INF,   INF,   INF }
00693     }
00694     , {{{   INF,   INF,   INF,   INF,   INF }
00695     , {   INF,   INF,   INF,   INF,   INF }
00696     , {   INF,   INF,   INF,   INF,   INF }
00697     , {   INF,   INF,   INF,   INF,   INF }
00698     , {   INF,   INF,   INF,   INF,   INF }
00699     }
00700     , {{{   INF,   INF,   INF,   INF,   INF }
00701     , {   INF,   INF,   INF,   INF,   INF }
00702     , {   INF,   INF,   INF,   INF,   INF }
00703     , {   INF,   INF,   INF,   INF,   INF }
00704     , {   INF,   INF,   INF,   INF,   INF }
00705     }
00706     , {{{   INF,   INF,   INF,   INF,   INF }
00707     , {   INF,   INF,   INF,   INF,   INF }
00708     , {   INF,   INF,   INF,   INF,   INF }
00709     , {   INF,   INF,   INF,   INF,   INF }
00710     , {   INF,   INF,   INF,   INF,   INF }
00711     }
00712     , {{{   INF,   INF,   INF,   INF,   INF }
00713     , {   INF,   INF,   INF,   INF,   INF }
00714     , {   INF,   INF,   INF,   INF,   INF }
00715     , {   INF,   INF,   INF,   INF,   INF }
00716     , {   INF,   INF,   INF,   INF,   INF }
00717     }
00718     }
00719     , {{{   INF,   INF,   INF,   INF,   INF }
00720     , {   INF,   INF,   INF,   INF,   INF }
00721     , {   INF,   INF,   INF,   INF,   INF }
00722     , {   INF,   INF,   INF,   INF,   INF }
00723     , {   INF,   INF,   INF,   INF,   INF }
00724     }
00725     , {{{   INF,   INF,   INF,   INF,   INF }
00726     , {   INF,   INF,   INF,   INF,   INF }
00727     , {   INF,   INF,   INF,   INF,   INF }
00728     , {   INF,   INF,   INF,   INF,   INF }
00729     , {   INF,   INF,   INF,   INF,   INF }
00730     }
00731     , {{{   INF,   INF,   INF,   INF,   INF }
00732     , {   INF,   INF,   INF,   INF,   INF }
00733     , {   INF,   INF,   INF,   INF,   INF }
00734     , {   INF,   INF,   INF,   INF,   INF }
00735     , {   INF,   INF,   INF,   INF,   INF }
00736     }
00737     , {{{   INF,   INF,   INF,   INF,   INF }
00738     , {   INF,   INF,   INF,   INF,   INF }
00739     , {   INF,   INF,   INF,   INF,   INF }
00740     , {   INF,   INF,   INF,   INF,   INF }
00741     , {   INF,   INF,   INF,   INF,   INF }
00742     }
00743     , {{{   INF,   INF,   INF,   INF,   INF }
00744     , {   INF,   INF,   INF,   INF,   INF }
00745     , {   INF,   INF,   INF,   INF,   INF }
00746     , {   INF,   INF,   INF,   INF,   INF }
00747     , {   INF,   INF,   INF,   INF,   INF }
00748     }
00749     }
00750     , {{{   INF,   INF,   INF,   INF,   INF }
00751     , {   INF,   INF,   INF,   INF,   INF }
00752     , {   INF,   INF,   INF,   INF,   INF }
00753     , {   INF,   INF,   INF,   INF,   INF }
00754     , {   INF,   INF,   INF,   INF,   INF }
```

```
00755     }
00756     ,{{   INF,   INF,   INF,   INF,   INF}
00757     ,{{   INF,   INF,   INF,   INF,   INF}
00758     ,{{   INF,   INF,   INF,   INF,   INF}
00759     ,{{   INF,   INF,   INF,   INF,   INF}
00760     ,{{   INF,   INF,   INF,   INF,   INF}
00761     }
00762     ,{{   INF,   INF,   INF,   INF,   INF}
00763     ,{{   INF,   INF,   INF,   INF,   INF}
00764     ,{{   INF,   INF,   INF,   INF,   INF}
00765     ,{{   INF,   INF,   INF,   INF,   INF}
00766     ,{{   INF,   INF,   INF,   INF,   INF}
00767     }
00768     ,{{   INF,   INF,   INF,   INF,   INF}
00769     ,{{   INF,   INF,   INF,   INF,   INF}
00770     ,{{   INF,   INF,   INF,   INF,   INF}
00771     ,{{   INF,   INF,   INF,   INF,   INF}
00772     ,{{   INF,   INF,   INF,   INF,   INF}
00773     }
00774     ,{{   INF,   INF,   INF,   INF,   INF}
00775     ,{{   INF,   INF,   INF,   INF,   INF}
00776     ,{{   INF,   INF,   INF,   INF,   INF}
00777     ,{{   INF,   INF,   INF,   INF,   INF}
00778     ,{{   INF,   INF,   INF,   INF,   INF}
00779     }
00780     }
00781     }
00782     ,{{{   INF,   INF,   INF,   INF,   INF}
00783     ,{{   INF,   INF,   INF,   INF,   INF}
00784     ,{{   INF,   INF,   INF,   INF,   INF}
00785     ,{{   INF,   INF,   INF,   INF,   INF}
00786     ,{{   INF,   INF,   INF,   INF,   INF}
00787     }
00788     ,{{   INF,   INF,   INF,   INF,   INF}
00789     ,{{   INF,   INF,   INF,   INF,   INF}
00790     ,{{   INF,   INF,   INF,   INF,   INF}
00791     ,{{   INF,   INF,   INF,   INF,   INF}
00792     ,{{   INF,   INF,   INF,   INF,   INF}
00793     }
00794     ,{{   INF,   INF,   INF,   INF,   INF}
00795     ,{{   INF,   INF,   INF,   INF,   INF}
00796     ,{{   INF,   INF,   INF,   INF,   INF}
00797     ,{{   INF,   INF,   INF,   INF,   INF}
00798     ,{{   INF,   INF,   INF,   INF,   INF}
00799     }
00800     ,{{   INF,   INF,   INF,   INF,   INF}
00801     ,{{   INF,   INF,   INF,   INF,   INF}
00802     ,{{   INF,   INF,   INF,   INF,   INF}
00803     ,{{   INF,   INF,   INF,   INF,   INF}
00804     ,{{   INF,   INF,   INF,   INF,   INF}
00805     }
00806     ,{{   INF,   INF,   INF,   INF,   INF}
00807     ,{{   INF,   INF,   INF,   INF,   INF}
00808     ,{{   INF,   INF,   INF,   INF,   INF}
00809     ,{{   INF,   INF,   INF,   INF,   INF}
00810     ,{{   INF,   INF,   INF,   INF,   INF}
00811     }
00812     }
00813     ,{{{   INF,   INF,   INF,   INF,   INF}
00814     ,{{   INF,   INF,   INF,   INF,   INF}
00815     ,{{   INF,   INF,   INF,   INF,   INF}
00816     ,{{   INF,   INF,   INF,   INF,   INF}
00817     ,{{   INF,   INF,   INF,   INF,   INF}
00818     }
00819     ,{{{   INF,   INF,   INF,   INF,   INF}
00820     ,{{   INF,   INF,   INF,   INF,   INF}
00821     ,{{   INF,   INF,   INF,   INF,   INF}
00822     ,{{   INF,   INF,   INF,   INF,   INF}
00823     ,{{   INF,   INF,   INF,   INF,   INF}
00824     }
00825     ,{{{   INF,   INF,   INF,   INF,   INF}
00826     ,{{   INF,   INF,   INF,   INF,   INF}
00827     ,{{   INF,   INF,   INF,   INF,   INF}
00828     ,{{   INF,   INF,   INF,   INF,   INF}
00829     ,{{   INF,   INF,   INF,   INF,   INF}
00830     }
00831     ,{{{   INF,   INF,   INF,   INF,   INF}
00832     ,{{   INF,   INF,   INF,   INF,   INF}
00833     ,{{   INF,   INF,   INF,   INF,   INF}
00834     ,{{   INF,   INF,   INF,   INF,   INF}
00835     ,{{   INF,   INF,   INF,   INF,   INF}
00836     }
00837     ,{{{   INF,   INF,   INF,   INF,   INF}
00838     ,{{   INF,   INF,   INF,   INF,   INF}
00839     ,{{   INF,   INF,   INF,   INF,   INF}
00840     ,{{   INF,   INF,   INF,   INF,   INF}
00841     ,{{   INF,   INF,   INF,   INF,   INF}
```

```
00842     }
00843     }
00844     , {{ { INF, INF, INF, INF, INF }
00845     , { INF, INF, INF, INF, INF }
00846     , { INF, INF, INF, INF, INF }
00847     , { INF, INF, INF, INF, INF }
00848     , { INF, INF, INF, INF, INF }
00849     }
00850     , {{ { INF, INF, INF, INF, INF }
00851     , { INF, INF, INF, INF, INF }
00852     , { INF, INF, INF, INF, INF }
00853     , { INF, INF, INF, INF, INF }
00854     , { INF, INF, INF, INF, INF }
00855     }
00856     , {{ { INF, INF, INF, INF, INF }
00857     , { INF, INF, INF, INF, INF }
00858     , { INF, INF, INF, INF, INF }
00859     , { INF, INF, INF, INF, INF }
00860     , { INF, INF, INF, INF, INF }
00861     }
00862     , {{ { INF, INF, INF, INF, INF }
00863     , { INF, INF, INF, INF, INF }
00864     , { INF, INF, INF, INF, INF }
00865     , { INF, INF, INF, INF, INF }
00866     , { INF, INF, INF, INF, INF }
00867     }
00868     , {{ { INF, INF, INF, INF, INF }
00869     , { INF, INF, INF, INF, INF }
00870     , { INF, INF, INF, INF, INF }
00871     , { INF, INF, INF, INF, INF }
00872     , { INF, INF, INF, INF, INF }
00873     }
00874     }
00875     , {{ { INF, INF, INF, INF, INF }
00876     , { INF, INF, INF, INF, INF }
00877     , { INF, INF, INF, INF, INF }
00878     , { INF, INF, INF, INF, INF }
00879     , { INF, INF, INF, INF, INF }
00880     }
00881     , {{ { INF, INF, INF, INF, INF }
00882     , { INF, INF, INF, INF, INF }
00883     , { INF, INF, INF, INF, INF }
00884     , { INF, INF, INF, INF, INF }
00885     , { INF, INF, INF, INF, INF }
00886     }
00887     , {{ { INF, INF, INF, INF, INF }
00888     , { INF, INF, INF, INF, INF }
00889     , { INF, INF, INF, INF, INF }
00890     , { INF, INF, INF, INF, INF }
00891     , { INF, INF, INF, INF, INF }
00892     }
00893     , {{ { INF, INF, INF, INF, INF }
00894     , { INF, INF, INF, INF, INF }
00895     , { INF, INF, INF, INF, INF }
00896     , { INF, INF, INF, INF, INF }
00897     , { INF, INF, INF, INF, INF }
00898     }
00899     , {{ { INF, INF, INF, INF, INF }
00900     , { INF, INF, INF, INF, INF }
00901     , { INF, INF, INF, INF, INF }
00902     , { INF, INF, INF, INF, INF }
00903     , { INF, INF, INF, INF, INF }
00904     }
00905     }
00906     , {{ { INF, INF, INF, INF, INF }
00907     , { INF, INF, INF, INF, INF }
00908     , { INF, INF, INF, INF, INF }
00909     , { INF, INF, INF, INF, INF }
00910     , { INF, INF, INF, INF, INF }
00911     }
00912     , {{ { INF, INF, INF, INF, INF }
00913     , { INF, INF, INF, INF, INF }
00914     , { INF, INF, INF, INF, INF }
00915     , { INF, INF, INF, INF, INF }
00916     , { INF, INF, INF, INF, INF }
00917     }
00918     , {{ { INF, INF, INF, INF, INF }
00919     , { INF, INF, INF, INF, INF }
00920     , { INF, INF, INF, INF, INF }
00921     , { INF, INF, INF, INF, INF }
00922     , { INF, INF, INF, INF, INF }
00923     }
00924     , {{ { INF, INF, INF, INF, INF }
00925     , { INF, INF, INF, INF, INF }
00926     , { INF, INF, INF, INF, INF }
00927     , { INF, INF, INF, INF, INF }
00928     , { INF, INF, INF, INF, INF }
```

```
00929     }
00930     ,{{   INF,   INF,   INF,   INF,   INF}
00931     ,{{   INF,   INF,   INF,   INF,   INF}
00932     ,{{   INF,   INF,   INF,   INF,   INF}
00933     ,{{   INF,   INF,   INF,   INF,   INF}
00934     ,{{   INF,   INF,   INF,   INF,   INF}
00935     }
00936     }
00937     }
00938     ,{{{   INF,   INF,   INF,   INF,   INF}
00939     ,{{   INF,   INF,   INF,   INF,   INF}
00940     ,{{   INF,   INF,   INF,   INF,   INF}
00941     ,{{   INF,   INF,   INF,   INF,   INF}
00942     ,{{   INF,   INF,   INF,   INF,   INF}
00943     }
00944     ,{{   INF,   INF,   INF,   INF,   INF}
00945     ,{{   INF,   INF,   INF,   INF,   INF}
00946     ,{{   INF,   INF,   INF,   INF,   INF}
00947     ,{{   INF,   INF,   INF,   INF,   INF}
00948     ,{{   INF,   INF,   INF,   INF,   INF}
00949     }
00950     ,{{   INF,   INF,   INF,   INF,   INF}
00951     ,{{   INF,   INF,   INF,   INF,   INF}
00952     ,{{   INF,   INF,   INF,   INF,   INF}
00953     ,{{   INF,   INF,   INF,   INF,   INF}
00954     ,{{   INF,   INF,   INF,   INF,   INF}
00955     }
00956     ,{{   INF,   INF,   INF,   INF,   INF}
00957     ,{{   INF,   INF,   INF,   INF,   INF}
00958     ,{{   INF,   INF,   INF,   INF,   INF}
00959     ,{{   INF,   INF,   INF,   INF,   INF}
00960     ,{{   INF,   INF,   INF,   INF,   INF}
00961     }
00962     ,{{   INF,   INF,   INF,   INF,   INF}
00963     ,{{   INF,   INF,   INF,   INF,   INF}
00964     ,{{   INF,   INF,   INF,   INF,   INF}
00965     ,{{   INF,   INF,   INF,   INF,   INF}
00966     ,{{   INF,   INF,   INF,   INF,   INF}
00967     }
00968     }
00969     ,{{{   INF,   INF,   INF,   INF,   INF}
00970     ,{{   INF,   INF,   INF,   INF,   INF}
00971     ,{{   INF,   INF,   INF,   INF,   INF}
00972     ,{{   INF,   INF,   INF,   INF,   INF}
00973     ,{{   INF,   INF,   INF,   INF,   INF}
00974     }
00975     ,{{{   INF,   INF,   INF,   INF,   INF}
00976     ,{{   INF,   INF,   INF,   INF,   INF}
00977     ,{{   INF,   INF,   INF,   INF,   INF}
00978     ,{{   INF,   INF,   INF,   INF,   INF}
00979     ,{{   INF,   INF,   INF,   INF,   INF}
00980     }
00981     ,{{{   INF,   INF,   INF,   INF,   INF}
00982     ,{{   INF,   INF,   INF,   INF,   INF}
00983     ,{{   INF,   INF,   INF,   INF,   INF}
00984     ,{{   INF,   INF,   INF,   INF,   INF}
00985     ,{{   INF,   INF,   INF,   INF,   INF}
00986     }
00987     ,{{{   INF,   INF,   INF,   INF,   INF}
00988     ,{{   INF,   INF,   INF,   INF,   INF}
00989     ,{{   INF,   INF,   INF,   INF,   INF}
00990     ,{{   INF,   INF,   INF,   INF,   INF}
00991     ,{{   INF,   INF,   INF,   INF,   INF}
00992     }
00993     ,{{{   INF,   INF,   INF,   INF,   INF}
00994     ,{{   INF,   INF,   INF,   INF,   INF}
00995     ,{{   INF,   INF,   INF,   INF,   INF}
00996     ,{{   INF,   INF,   INF,   INF,   INF}
00997     ,{{   INF,   INF,   INF,   INF,   INF}
00998     }
00999     }
01000     ,{{{   INF,   INF,   INF,   INF,   INF}
01001     ,{{   INF,   INF,   INF,   INF,   INF}
01002     ,{{   INF,   INF,   INF,   INF,   INF}
01003     ,{{   INF,   INF,   INF,   INF,   INF}
01004     ,{{   INF,   INF,   INF,   INF,   INF}
01005     }
01006     ,{{{   INF,   INF,   INF,   INF,   INF}
01007     ,{{   INF,   INF,   INF,   INF,   INF}
01008     ,{{   INF,   INF,   INF,   INF,   INF}
01009     ,{{   INF,   INF,   INF,   INF,   INF}
01010     ,{{   INF,   INF,   INF,   INF,   INF}
01011     }
01012     ,{{{   INF,   INF,   INF,   INF,   INF}
01013     ,{{   INF,   INF,   INF,   INF,   INF}
01014     ,{{   INF,   INF,   INF,   INF,   INF}
01015     ,{{   INF,   INF,   INF,   INF,   INF}
```

```
01016     , {   INF,   INF,   INF,   INF,   INF }
01017     }
01018     , { {   INF,   INF,   INF,   INF,   INF }
01019     , {   INF,   INF,   INF,   INF,   INF }
01020     , {   INF,   INF,   INF,   INF,   INF }
01021     , {   INF,   INF,   INF,   INF,   INF }
01022     , {   INF,   INF,   INF,   INF,   INF }
01023     }
01024     , { {   INF,   INF,   INF,   INF,   INF }
01025     , {   INF,   INF,   INF,   INF,   INF }
01026     , {   INF,   INF,   INF,   INF,   INF }
01027     , {   INF,   INF,   INF,   INF,   INF }
01028     , {   INF,   INF,   INF,   INF,   INF }
01029     }
01030     }
01031     , { { {   INF,   INF,   INF,   INF,   INF }
01032     , {   INF,   INF,   INF,   INF,   INF }
01033     , {   INF,   INF,   INF,   INF,   INF }
01034     , {   INF,   INF,   INF,   INF,   INF }
01035     , {   INF,   INF,   INF,   INF,   INF }
01036     }
01037     , { {   INF,   INF,   INF,   INF,   INF }
01038     , {   INF,   INF,   INF,   INF,   INF }
01039     , {   INF,   INF,   INF,   INF,   INF }
01040     , {   INF,   INF,   INF,   INF,   INF }
01041     , {   INF,   INF,   INF,   INF,   INF }
01042     }
01043     , { {   INF,   INF,   INF,   INF,   INF }
01044     , {   INF,   INF,   INF,   INF,   INF }
01045     , {   INF,   INF,   INF,   INF,   INF }
01046     , {   INF,   INF,   INF,   INF,   INF }
01047     , {   INF,   INF,   INF,   INF,   INF }
01048     }
01049     , { {   INF,   INF,   INF,   INF,   INF }
01050     , {   INF,   INF,   INF,   INF,   INF }
01051     , {   INF,   INF,   INF,   INF,   INF }
01052     , {   INF,   INF,   INF,   INF,   INF }
01053     , {   INF,   INF,   INF,   INF,   INF }
01054     }
01055     , { {   INF,   INF,   INF,   INF,   INF }
01056     , {   INF,   INF,   INF,   INF,   INF }
01057     , {   INF,   INF,   INF,   INF,   INF }
01058     , {   INF,   INF,   INF,   INF,   INF }
01059     , {   INF,   INF,   INF,   INF,   INF }
01060     }
01061     }
01062     , { { {   INF,   INF,   INF,   INF,   INF }
01063     , {   INF,   INF,   INF,   INF,   INF }
01064     , {   INF,   INF,   INF,   INF,   INF }
01065     , {   INF,   INF,   INF,   INF,   INF }
01066     , {   INF,   INF,   INF,   INF,   INF }
01067     }
01068     , { {   INF,   INF,   INF,   INF,   INF }
01069     , {   INF,   INF,   INF,   INF,   INF }
01070     , {   INF,   INF,   INF,   INF,   INF }
01071     , {   INF,   INF,   INF,   INF,   INF }
01072     , {   INF,   INF,   INF,   INF,   INF }
01073     }
01074     , { {   INF,   INF,   INF,   INF,   INF }
01075     , {   INF,   INF,   INF,   INF,   INF }
01076     , {   INF,   INF,   INF,   INF,   INF }
01077     , {   INF,   INF,   INF,   INF,   INF }
01078     , {   INF,   INF,   INF,   INF,   INF }
01079     }
01080     , { {   INF,   INF,   INF,   INF,   INF }
01081     , {   INF,   INF,   INF,   INF,   INF }
01082     , {   INF,   INF,   INF,   INF,   INF }
01083     , {   INF,   INF,   INF,   INF,   INF }
01084     , {   INF,   INF,   INF,   INF,   INF }
01085     }
01086     , { {   INF,   INF,   INF,   INF,   INF }
01087     , {   INF,   INF,   INF,   INF,   INF }
01088     , {   INF,   INF,   INF,   INF,   INF }
01089     , {   INF,   INF,   INF,   INF,   INF }
01090     , {   INF,   INF,   INF,   INF,   INF }
01091     }
01092     }
01093     }
01094     , { { { {   INF,   INF,   INF,   INF,   INF }
01095     , {   INF,   INF,   INF,   INF,   INF }
01096     , {   INF,   INF,   INF,   INF,   INF }
01097     , {   INF,   INF,   INF,   INF,   INF }
01098     , {   INF,   INF,   INF,   INF,   INF }
01099     }
01100     , { {   INF,   INF,   INF,   INF,   INF }
01101     , {   INF,   INF,   INF,   INF,   INF }
01102     , {   INF,   INF,   INF,   INF,   INF }
```

```
01103      , {   INF,   INF,   INF,   INF,   INF }
01104      , {   INF,   INF,   INF,   INF,   INF }
01105      }
01106      , { {   INF,   INF,   INF,   INF,   INF }
01107      , {   INF,   INF,   INF,   INF,   INF }
01108      , {   INF,   INF,   INF,   INF,   INF }
01109      , {   INF,   INF,   INF,   INF,   INF }
01110      , {   INF,   INF,   INF,   INF,   INF }
01111      }
01112      , { {   INF,   INF,   INF,   INF,   INF }
01113      , {   INF,   INF,   INF,   INF,   INF }
01114      , {   INF,   INF,   INF,   INF,   INF }
01115      , {   INF,   INF,   INF,   INF,   INF }
01116      , {   INF,   INF,   INF,   INF,   INF }
01117      }
01118      , { {   INF,   INF,   INF,   INF,   INF }
01119      , {   INF,   INF,   INF,   INF,   INF }
01120      , {   INF,   INF,   INF,   INF,   INF }
01121      , {   INF,   INF,   INF,   INF,   INF }
01122      , {   INF,   INF,   INF,   INF,   INF }
01123      }
01124      }
01125      , { { {   INF,   INF,   INF,   INF,   INF }
01126      , {   INF,   INF,   INF,   INF,   INF }
01127      , {   INF,   INF,   INF,   INF,   INF }
01128      , {   INF,   INF,   INF,   INF,   INF }
01129      , {   INF,   INF,   INF,   INF,   INF }
01130      }
01131      , { {   INF,   INF,   INF,   INF,   INF }
01132      , {   INF,   INF,   INF,   INF,   INF }
01133      , {   INF,   INF,   INF,   INF,   INF }
01134      , {   INF,   INF,   INF,   INF,   INF }
01135      , {   INF,   INF,   INF,   INF,   INF }
01136      }
01137      , { {   INF,   INF,   INF,   INF,   INF }
01138      , {   INF,   INF,   INF,   INF,   INF }
01139      , {   INF,   INF,   INF,   INF,   INF }
01140      , {   INF,   INF,   INF,   INF,   INF }
01141      , {   INF,   INF,   INF,   INF,   INF }
01142      }
01143      , { {   INF,   INF,   INF,   INF,   INF }
01144      , {   INF,   INF,   INF,   INF,   INF }
01145      , {   INF,   INF,   INF,   INF,   INF }
01146      , {   INF,   INF,   INF,   INF,   INF }
01147      , {   INF,   INF,   INF,   INF,   INF }
01148      }
01149      , { {   INF,   INF,   INF,   INF,   INF }
01150      , {   INF,   INF,   INF,   INF,   INF }
01151      , {   INF,   INF,   INF,   INF,   INF }
01152      , {   INF,   INF,   INF,   INF,   INF }
01153      , {   INF,   INF,   INF,   INF,   INF }
01154      }
01155      }
01156      , { { {   INF,   INF,   INF,   INF,   INF }
01157      , {   INF,   INF,   INF,   INF,   INF }
01158      , {   INF,   INF,   INF,   INF,   INF }
01159      , {   INF,   INF,   INF,   INF,   INF }
01160      , {   INF,   INF,   INF,   INF,   INF }
01161      }
01162      , { {   INF,   INF,   INF,   INF,   INF }
01163      , {   INF,   INF,   INF,   INF,   INF }
01164      , {   INF,   INF,   INF,   INF,   INF }
01165      , {   INF,   INF,   INF,   INF,   INF }
01166      , {   INF,   INF,   INF,   INF,   INF }
01167      }
01168      , { {   INF,   INF,   INF,   INF,   INF }
01169      , {   INF,   INF,   INF,   INF,   INF }
01170      , {   INF,   INF,   INF,   INF,   INF }
01171      , {   INF,   INF,   INF,   INF,   INF }
01172      , {   INF,   INF,   INF,   INF,   INF }
01173      }
01174      , { {   INF,   INF,   INF,   INF,   INF }
01175      , {   INF,   INF,   INF,   INF,   INF }
01176      , {   INF,   INF,   INF,   INF,   INF }
01177      , {   INF,   INF,   INF,   INF,   INF }
01178      , {   INF,   INF,   INF,   INF,   INF }
01179      }
01180      , { {   INF,   INF,   INF,   INF,   INF }
01181      , {   INF,   INF,   INF,   INF,   INF }
01182      , {   INF,   INF,   INF,   INF,   INF }
01183      , {   INF,   INF,   INF,   INF,   INF }
01184      , {   INF,   INF,   INF,   INF,   INF }
01185      }
01186      }
01187      , { { {   INF,   INF,   INF,   INF,   INF }
01188      , {   INF,   INF,   INF,   INF,   INF }
01189      , {   INF,   INF,   INF,   INF,   INF }
```

```
01190     , {   INF,   INF,   INF,   INF,   INF }
01191     , {   INF,   INF,   INF,   INF,   INF }
01192     }
01193     , { {   INF,   INF,   INF,   INF,   INF }
01194     , {   INF,   INF,   INF,   INF,   INF }
01195     , {   INF,   INF,   INF,   INF,   INF }
01196     , {   INF,   INF,   INF,   INF,   INF }
01197     , {   INF,   INF,   INF,   INF,   INF }
01198     }
01199     , { {   INF,   INF,   INF,   INF,   INF }
01200     , {   INF,   INF,   INF,   INF,   INF }
01201     , {   INF,   INF,   INF,   INF,   INF }
01202     , {   INF,   INF,   INF,   INF,   INF }
01203     , {   INF,   INF,   INF,   INF,   INF }
01204     }
01205     , { {   INF,   INF,   INF,   INF,   INF }
01206     , {   INF,   INF,   INF,   INF,   INF }
01207     , {   INF,   INF,   INF,   INF,   INF }
01208     , {   INF,   INF,   INF,   INF,   INF }
01209     , {   INF,   INF,   INF,   INF,   INF }
01210     }
01211     , { {   INF,   INF,   INF,   INF,   INF }
01212     , {   INF,   INF,   INF,   INF,   INF }
01213     , {   INF,   INF,   INF,   INF,   INF }
01214     , {   INF,   INF,   INF,   INF,   INF }
01215     , {   INF,   INF,   INF,   INF,   INF }
01216     }
01217     }
01218     , { { {   INF,   INF,   INF,   INF,   INF }
01219     , {   INF,   INF,   INF,   INF,   INF }
01220     , {   INF,   INF,   INF,   INF,   INF }
01221     , {   INF,   INF,   INF,   INF,   INF }
01222     , {   INF,   INF,   INF,   INF,   INF }
01223     }
01224     , { {   INF,   INF,   INF,   INF,   INF }
01225     , {   INF,   INF,   INF,   INF,   INF }
01226     , {   INF,   INF,   INF,   INF,   INF }
01227     , {   INF,   INF,   INF,   INF,   INF }
01228     , {   INF,   INF,   INF,   INF,   INF }
01229     }
01230     , { {   INF,   INF,   INF,   INF,   INF }
01231     , {   INF,   INF,   INF,   INF,   INF }
01232     , {   INF,   INF,   INF,   INF,   INF }
01233     , {   INF,   INF,   INF,   INF,   INF }
01234     , {   INF,   INF,   INF,   INF,   INF }
01235     }
01236     , { {   INF,   INF,   INF,   INF,   INF }
01237     , {   INF,   INF,   INF,   INF,   INF }
01238     , {   INF,   INF,   INF,   INF,   INF }
01239     , {   INF,   INF,   INF,   INF,   INF }
01240     , {   INF,   INF,   INF,   INF,   INF }
01241     }
01242     , { {   INF,   INF,   INF,   INF,   INF }
01243     , {   INF,   INF,   INF,   INF,   INF }
01244     , {   INF,   INF,   INF,   INF,   INF }
01245     , {   INF,   INF,   INF,   INF,   INF }
01246     , {   INF,   INF,   INF,   INF,   INF }
01247     }
01248     }
01249     }
01250     }
01251     , { { { {   INF,   INF,   INF,   INF,   INF }
01252     , {   INF,   INF,   INF,   INF,   INF }
01253     , {   INF,   INF,   INF,   INF,   INF }
01254     , {   INF,   INF,   INF,   INF,   INF }
01255     , {   INF,   INF,   INF,   INF,   INF }
01256     }
01257     , { {   INF,   INF,   INF,   INF,   INF }
01258     , {   INF,   INF,   INF,   INF,   INF }
01259     , {   INF,   INF,   INF,   INF,   INF }
01260     , {   INF,   INF,   INF,   INF,   INF }
01261     , {   INF,   INF,   INF,   INF,   INF }
01262     }
01263     , { {   INF,   INF,   INF,   INF,   INF }
01264     , {   INF,   INF,   INF,   INF,   INF }
01265     , {   INF,   INF,   INF,   INF,   INF }
01266     , {   INF,   INF,   INF,   INF,   INF }
01267     , {   INF,   INF,   INF,   INF,   INF }
01268     }
01269     , { {   INF,   INF,   INF,   INF,   INF }
01270     , {   INF,   INF,   INF,   INF,   INF }
01271     , {   INF,   INF,   INF,   INF,   INF }
01272     , {   INF,   INF,   INF,   INF,   INF }
01273     , {   INF,   INF,   INF,   INF,   INF }
01274     }
01275     , { {   INF,   INF,   INF,   INF,   INF }
01276     , {   INF,   INF,   INF,   INF,   INF }
```

```
01277      , {   INF,   INF,   INF,   INF,   INF }
01278      , {   INF,   INF,   INF,   INF,   INF }
01279      , {   INF,   INF,   INF,   INF,   INF }
01280      }
01281      }
01282      , { {   INF,   INF,   INF,   INF,   INF }
01283      , {   INF,   INF,   INF,   INF,   INF }
01284      , {   INF,   INF,   INF,   INF,   INF }
01285      , {   INF,   INF,   INF,   INF,   INF }
01286      , {   INF,   INF,   INF,   INF,   INF }
01287      }
01288      , { {   INF,   INF,   INF,   INF,   INF }
01289      , {   INF,   INF,   INF,   INF,   INF }
01290      , {   INF,   INF,   INF,   INF,   INF }
01291      , {   INF,   INF,   INF,   INF,   INF }
01292      , {   INF,   INF,   INF,   INF,   INF }
01293      }
01294      , { {   INF,   INF,   INF,   INF,   INF }
01295      , {   INF,   INF,   INF,   INF,   INF }
01296      , {   INF,   INF,   INF,   INF,   INF }
01297      , {   INF,   INF,   INF,   INF,   INF }
01298      , {   INF,   INF,   INF,   INF,   INF }
01299      }
01300      , { {   INF,   INF,   INF,   INF,   INF }
01301      , {   INF,   INF,   INF,   INF,   INF }
01302      , {   INF,   INF,   INF,   INF,   INF }
01303      , {   INF,   INF,   INF,   INF,   INF }
01304      , {   INF,   INF,   INF,   INF,   INF }
01305      }
01306      , { {   INF,   INF,   INF,   INF,   INF }
01307      , {   INF,   INF,   INF,   INF,   INF }
01308      , {   INF,   INF,   INF,   INF,   INF }
01309      , {   INF,   INF,   INF,   INF,   INF }
01310      , {   INF,   INF,   INF,   INF,   INF }
01311      }
01312      }
01313      , { { {   INF,   INF,   INF,   INF,   INF }
01314      , {   INF,   INF,   INF,   INF,   INF }
01315      , {   INF,   INF,   INF,   INF,   INF }
01316      , {   INF,   INF,   INF,   INF,   INF }
01317      , {   INF,   INF,   INF,   INF,   INF }
01318      }
01319      , { {   INF,   INF,   INF,   INF,   INF }
01320      , {   INF,   INF,   INF,   INF,   INF }
01321      , {   INF,   INF,   INF,   INF,   INF }
01322      , {   INF,   INF,   INF,   INF,   INF }
01323      , {   INF,   INF,   INF,   INF,   INF }
01324      }
01325      , { {   INF,   INF,   INF,   INF,   INF }
01326      , {   INF,   INF,   INF,   INF,   INF }
01327      , {   INF,   INF,   INF,   INF,   INF }
01328      , {   INF,   INF,   INF,   INF,   INF }
01329      , {   INF,   INF,   INF,   INF,   INF }
01330      }
01331      , { {   INF,   INF,   INF,   INF,   INF }
01332      , {   INF,   INF,   INF,   INF,   INF }
01333      , {   INF,   INF,   INF,   INF,   INF }
01334      , {   INF,   INF,   INF,   INF,   INF }
01335      , {   INF,   INF,   INF,   INF,   INF }
01336      }
01337      , { {   INF,   INF,   INF,   INF,   INF }
01338      , {   INF,   INF,   INF,   INF,   INF }
01339      , {   INF,   INF,   INF,   INF,   INF }
01340      , {   INF,   INF,   INF,   INF,   INF }
01341      , {   INF,   INF,   INF,   INF,   INF }
01342      }
01343      }
01344      , { { {   INF,   INF,   INF,   INF,   INF }
01345      , {   INF,   INF,   INF,   INF,   INF }
01346      , {   INF,   INF,   INF,   INF,   INF }
01347      , {   INF,   INF,   INF,   INF,   INF }
01348      , {   INF,   INF,   INF,   INF,   INF }
01349      }
01350      , { {   INF,   INF,   INF,   INF,   INF }
01351      , {   INF,   INF,   INF,   INF,   INF }
01352      , {   INF,   INF,   INF,   INF,   INF }
01353      , {   INF,   INF,   INF,   INF,   INF }
01354      , {   INF,   INF,   INF,   INF,   INF }
01355      }
01356      , { {   INF,   INF,   INF,   INF,   INF }
01357      , {   INF,   INF,   INF,   INF,   INF }
01358      , {   INF,   INF,   INF,   INF,   INF }
01359      , {   INF,   INF,   INF,   INF,   INF }
01360      , {   INF,   INF,   INF,   INF,   INF }
01361      }
01362      , { {   INF,   INF,   INF,   INF,   INF }
01363      , {   INF,   INF,   INF,   INF,   INF }
```



```

01364     , {   INF,   INF,   INF,   INF,   INF }
01365     , {   INF,   INF,   INF,   INF,   INF }
01366     , {   INF,   INF,   INF,   INF,   INF }
01367     }
01368     , {{   INF,   INF,   INF,   INF,   INF }
01369     , {   INF,   INF,   INF,   INF,   INF }
01370     , {   INF,   INF,   INF,   INF,   INF }
01371     , {   INF,   INF,   INF,   INF,   INF }
01372     , {   INF,   INF,   INF,   INF,   INF }
01373     }
01374     }
01375     , {{{   INF,   INF,   INF,   INF,   INF }
01376     , {   INF,   INF,   INF,   INF,   INF }
01377     , {   INF,   INF,   INF,   INF,   INF }
01378     , {   INF,   INF,   INF,   INF,   INF }
01379     , {   INF,   INF,   INF,   INF,   INF }
01380     }
01381     , {{{   INF,   INF,   INF,   INF,   INF }
01382     , {   INF,   INF,   INF,   INF,   INF }
01383     , {   INF,   INF,   INF,   INF,   INF }
01384     , {   INF,   INF,   INF,   INF,   INF }
01385     , {   INF,   INF,   INF,   INF,   INF }
01386     }
01387     , {{{   INF,   INF,   INF,   INF,   INF }
01388     , {   INF,   INF,   INF,   INF,   INF }
01389     , {   INF,   INF,   INF,   INF,   INF }
01390     , {   INF,   INF,   INF,   INF,   INF }
01391     , {   INF,   INF,   INF,   INF,   INF }
01392     }
01393     , {{{   INF,   INF,   INF,   INF,   INF }
01394     , {   INF,   INF,   INF,   INF,   INF }
01395     , {   INF,   INF,   INF,   INF,   INF }
01396     , {   INF,   INF,   INF,   INF,   INF }
01397     , {   INF,   INF,   INF,   INF,   INF }
01398     }
01399     , {{{   INF,   INF,   INF,   INF,   INF }
01400     , {   INF,   INF,   INF,   INF,   INF }
01401     , {   INF,   INF,   INF,   INF,   INF }
01402     , {   INF,   INF,   INF,   INF,   INF }
01403     , {   INF,   INF,   INF,   INF,   INF }
01404     }
01405     }
01406     }
01407     , {{{ 200,   160,   200,   150,   200 }
01408     , { 200,   160,   200,   150,   200 }
01409     , { 180,   140,   180,   140,   180 }
01410     , { 200,   160,   200,   150,   200 }
01411     , { 170,   130,   170,   120,   170 }
01412     }
01413     , {{{ 160,   120,   160,   110,   160 }
01414     , { 160,   120,   160,   110,   160 }
01415     , { 150,   110,   150,   110,   150 }
01416     , { 110,    20,   110,    20,    90 }
01417     , { 150,   110,   150,   110,   150 }
01418     }
01419     , {{{ 200,   160,   200,   150,   200 }
01420     , { 200,   160,   200,   150,   200 }
01421     , { 180,   140,   180,   140,   180 }
01422     , { 200,   160,   200,   150,   200 }
01423     , { 170,   130,   170,   120,   170 }
01424     }
01425     , {{{ 150,   110,   150,   110,   150 }
01426     , { 110,    20,   110,    20,    90 }
01427     , { 150,   110,   150,   110,   150 }
01428     , {  80,    0,    10,    80,    20 }
01429     , { 150,   110,   150,   110,   150 }
01430     }
01431     , {{{ 200,   160,   200,   150,   200 }
01432     , { 200,   160,   200,   150,   200 }
01433     , { 170,   130,   170,   120,   170 }
01434     , { 200,   160,   200,   150,   200 }
01435     , { 100,   100,    80,    30,    80 }
01436     }
01437     }
01438     , {{{ 200,   160,   200,   110,   200 }
01439     , { 200,   160,   200,    60,   200 }
01440     , { 180,   140,   180,   110,   180 }
01441     , { 200,   160,   200,    60,   200 }
01442     , { 170,   130,   170,    90,   170 }
01443     }
01444     , {{{ 160,   120,   160,    20,   160 }
01445     , { 160,   120,   160,    20,   160 }
01446     , { 150,   110,   150,    20,   150 }
01447     , {  60,    20,    60,   -70,    60 }
01448     , { 150,   110,   150,    20,   150 }
01449     }
01450     , {{{ 200,   160,   200,   110,   200 }

```

```
01451      , { 200, 160, 200, 60, 200}
01452      , { 180, 140, 180, 110, 180}
01453      , { 200, 160, 200, 60, 200}
01454      , { 170, 130, 170, 90, 170}
01455      }
01456      , { { 150, 110, 150, 20, 150}
01457      , { 60, 20, 60, -70, 60}
01458      , { 150, 110, 150, 20, 150}
01459      , { 10, -30, 10, 0, 10}
01460      , { 150, 110, 150, 20, 150}
01461      }
01462      , { { 200, 160, 200, 90, 200}
01463      , { 200, 160, 200, 60, 200}
01464      , { 170, 130, 170, 90, 170}
01465      , { 200, 160, 200, 60, 200}
01466      , { 100, 100, 80, -50, 80}
01467      }
01468      }
01469      , { { { 180, 150, 180, 150, 170}
01470      , { 180, 150, 180, 150, 170}
01471      , { 170, 140, 170, 140, 150}
01472      , { 180, 150, 180, 150, 170}
01473      , { 150, 120, 150, 120, 140}
01474      }
01475      , { { 140, 110, 140, 110, 130}
01476      , { 140, 110, 140, 110, 130}
01477      , { 140, 110, 140, 110, 120}
01478      , { 110, 20, 110, 20, 90}
01479      , { 140, 110, 140, 110, 120}
01480      }
01481      , { { 180, 150, 180, 150, 170}
01482      , { 180, 150, 180, 150, 170}
01483      , { 170, 140, 170, 140, 150}
01484      , { 180, 150, 180, 150, 170}
01485      , { 150, 120, 150, 120, 140}
01486      }
01487      , { { 140, 110, 140, 110, 120}
01488      , { 110, 20, 110, 20, 90}
01489      , { 140, 110, 140, 110, 120}
01490      , { -10, -40, -10, -40, -20}
01491      , { 140, 110, 140, 110, 120}
01492      }
01493      , { { 180, 150, 180, 150, 170}
01494      , { 180, 150, 180, 150, 170}
01495      , { 150, 120, 150, 120, 140}
01496      , { 180, 150, 180, 150, 170}
01497      , { 60, 30, 60, 30, 50}
01498      }
01499      }
01500      , { { { 200, 110, 200, 80, 200}
01501      , { 200, 60, 200, 10, 200}
01502      , { 180, 110, 180, -10, 180}
01503      , { 200, 60, 200, 80, 200}
01504      , { 170, 90, 170, 20, 170}
01505      }
01506      , { { 160, 20, 160, 0, 160}
01507      , { 160, 20, 160, -30, 160}
01508      , { 150, 20, 150, -40, 150}
01509      , { 60, -70, 60, 0, 60}
01510      , { 150, 20, 150, -40, 150}
01511      }
01512      , { { 200, 110, 200, 10, 200}
01513      , { 200, 60, 200, 10, 200}
01514      , { 180, 110, 180, -10, 180}
01515      , { 200, 60, 200, 10, 200}
01516      , { 170, 90, 170, -20, 170}
01517      }
01518      , { { 150, 20, 150, 80, 150}
01519      , { 60, -70, 60, 0, 60}
01520      , { 150, 20, 150, -40, 150}
01521      , { 80, 0, 10, 80, 10}
01522      , { 150, 20, 150, -40, 150}
01523      }
01524      , { { 200, 90, 200, 20, 200}
01525      , { 200, 60, 200, 10, 200}
01526      , { 170, 90, 170, -20, 170}
01527      , { 200, 60, 200, 10, 200}
01528      , { 80, -50, 80, 20, 80}
01529      }
01530      }
01531      , { { { 170, 150, 170, 150, 100}
01532      , { 170, 150, 170, 150, 100}
01533      , { 150, 140, 150, 140, 60}
01534      , { 170, 150, 170, 150, 80}
01535      , { 140, 120, 140, 120, 50}
01536      }
01537      , { { 130, 110, 130, 110, 100}
```

```

01538     , { 130, 110, 130, 110, 100}
01539     , { 120, 110, 120, 110, 30}
01540     , { 90, 20, 90, 20, -50}
01541     , { 120, 110, 120, 110, 30}
01542     }
01543     , { { 170, 150, 170, 150, 80}
01544     , { 170, 150, 170, 150, 80}
01545     , { 150, 140, 150, 140, 60}
01546     , { 170, 150, 170, 150, 80}
01547     , { 140, 120, 140, 120, 50}
01548     }
01549     , { { 120, 110, 120, 110, 30}
01550     , { 90, 20, 90, 20, -50}
01551     , { 120, 110, 120, 110, 30}
01552     , { 20, -40, -20, -40, 20}
01553     , { 120, 110, 120, 110, 30}
01554     }
01555     , { { 170, 150, 170, 150, 80}
01556     , { 170, 150, 170, 150, 80}
01557     , { 140, 120, 140, 120, 50}
01558     , { 170, 150, 170, 150, 80}
01559     , { 50, 30, 50, 30, -40}
01560     }
01561     }
01562     }
01563     , { { { 220, 150, 220, 140, 170}
01564     , { 220, 130, 220, 130, 170}
01565     , { 150, 110, 150, 110, 150}
01566     , { 140, 100, 140, 100, 140}
01567     , { 170, 150, 150, 140, 170}
01568     }
01569     , { { 220, 130, 220, 130, 170}
01570     , { 220, 130, 220, 130, 170}
01571     , { 150, 110, 150, 100, 150}
01572     , { 70, -30, 70, -70, 50}
01573     , { 150, 110, 150, 100, 150}
01574     }
01575     , { { 190, 110, 190, 100, 170}
01576     , { 190, 110, 190, 100, 140}
01577     , { 150, 110, 150, 100, 150}
01578     , { 140, 100, 140, 100, 140}
01579     , { 170, 110, 150, 100, 170}
01580     }
01581     , { { 150, 110, 150, 100, 150}
01582     , { 140, 70, 70, -10, 140}
01583     , { 150, 110, 150, 100, 150}
01584     , { 80, -30, 10, 80, 70}
01585     , { 150, 110, 150, 100, 150}
01586     }
01587     , { { 150, 150, 150, 140, 150}
01588     , { 140, 100, 140, 100, 140}
01589     , { 150, 110, 150, 110, 150}
01590     , { 140, 100, 140, 100, 140}
01591     , { 150, 150, 70, 140, 70}
01592     }
01593     }
01594     , { { { 170, 150, 150, 90, 170}
01595     , { 170, 130, 140, 10, 170}
01596     , { 150, 110, 150, 80, 150}
01597     , { 140, 100, 140, 10, 140}
01598     , { 150, 150, 150, 90, 150}
01599     }
01600     , { { 170, 130, 150, 10, 170}
01601     , { 170, 130, 60, 0, 170}
01602     , { 150, 110, 150, -70, 150}
01603     , { 10, -30, 10, -160, -30}
01604     , { 150, 110, 150, 10, 150}
01605     }
01606     , { { 150, 110, 150, 70, 150}
01607     , { 140, 100, 50, -100, 140}
01608     , { 150, 110, 150, -60, 150}
01609     , { 140, 100, 140, 10, 140}
01610     , { 150, 110, 150, 70, 150}
01611     }
01612     , { { 150, 110, 150, 10, 150}
01613     , { 40, 40, 30, -70, 30}
01614     , { 150, 110, 150, 10, 150}
01615     , { 10, -30, -30, 0, 10}
01616     , { 150, 110, 150, 10, 150}
01617     }
01618     , { { 150, 150, 150, 90, 150}
01619     , { 140, 100, 140, 10, 140}
01620     , { 150, 110, 150, 80, 150}
01621     , { 140, 100, 140, 10, 140}
01622     , { 150, 150, 0, 90, 70}
01623     }
01624     }

```

```
01625 ,{{{ 220, 130, 220, 130, 170}
01626 ,{ 220, 130, 220, 130, 140}
01627 ,{ 140, 110, 140, 110, 120}
01628 ,{ 130, 100, 130, 100, 110}
01629 ,{ 170, 100, 130, 100, 170}
01630 }
01631 ,{{{ 220, 130, 220, 130, 140}
01632 ,{ 220, 130, 220, 130, 140}
01633 ,{ 130, 100, 130, 100, 120}
01634 ,{ 70, -70, 70, -70, 0}
01635 ,{ 130, 100, 130, 100, 120}
01636 }
01637 ,{{{ 190, 110, 190, 100, 170}
01638 ,{ 190, 110, 190, 100, 110}
01639 ,{ 130, 100, 130, 100, 120}
01640 ,{ 130, 100, 130, 100, 110}
01641 ,{ 170, 100, 130, 100, 170}
01642 }
01643 ,{{{ 130, 100, 130, 100, 120}
01644 ,{ 70, 70, 70, -10, 60}
01645 ,{ 130, 100, 130, 100, 120}
01646 ,{ 20, -40, -10, -40, 20}
01647 ,{ 130, 100, 130, 100, 120}
01648 }
01649 ,{{{ 140, 110, 140, 110, 120}
01650 ,{ 130, 100, 130, 100, 110}
01651 ,{ 140, 110, 140, 110, 120}
01652 ,{ 130, 100, 130, 100, 110}
01653 ,{ 30, -20, -10, 30, 20}
01654 }
01655 }
01656 ,{{{ 170, 90, 170, 140, 170}
01657 ,{ 170, 70, 170, -10, 170}
01658 ,{ 150, 80, 150, -40, 150}
01659 ,{ 140, 10, 140, 80, 140}
01660 ,{ 150, 90, 150, 140, 150}
01661 }
01662 ,{{{ 170, 10, 170, -10, 170}
01663 ,{ 170, -20, 170, -10, 170}
01664 ,{ 150, -40, 150, -40, 150}
01665 ,{ -30, -170, -30, -90, -30}
01666 ,{ 150, 10, 150, -40, 150}
01667 }
01668 ,{{{ 150, 70, 150, 20, 150}
01669 ,{ 140, 70, 140, -50, 140}
01670 ,{ 150, 70, 150, -40, 150}
01671 ,{ 140, 10, 140, -50, 140}
01672 ,{ 150, 70, 150, 20, 150}
01673 }
01674 ,{{{ 150, 10, 150, 80, 150}
01675 ,{ 30, -50, 30, -30, 30}
01676 ,{ 150, 10, 150, -40, 150}
01677 ,{ 80, -30, 10, 80, 10}
01678 ,{ 150, 10, 150, -40, 150}
01679 }
01680 ,{{{ 150, 90, 150, 140, 150}
01681 ,{ 140, 10, 140, -50, 140}
01682 ,{ 150, 80, 150, -50, 150}
01683 ,{ 140, 10, 140, -50, 140}
01684 ,{ 140, 90, 70, 140, 70}
01685 }
01686 }
01687 ,{{{ 140, 130, 140, 130, 140}
01688 ,{ 140, 130, 140, 130, 140}
01689 ,{ 120, 110, 120, 110, 30}
01690 ,{ 110, 100, 110, 100, 70}
01691 ,{ 120, 100, 120, 100, 30}
01692 }
01693 ,{{{ 140, 130, 140, 130, 140}
01694 ,{ 140, 130, 140, 130, 140}
01695 ,{ 120, 100, 120, 100, 30}
01696 ,{ 50, -70, 0, -70, 50}
01697 ,{ 120, 100, 120, 100, 30}
01698 }
01699 ,{{{ 120, 100, 120, 100, 30}
01700 ,{ 110, 100, 110, 100, 30}
01701 ,{ 120, 100, 120, 100, 30}
01702 ,{ 110, 100, 110, 100, 20}
01703 ,{ 120, 100, 120, 100, 30}
01704 }
01705 ,{{{ 140, 100, 120, 100, 140}
01706 ,{ 140, -10, 50, -10, 140}
01707 ,{ 120, 100, 120, 100, 30}
01708 ,{ 70, -40, -60, -40, 70}
01709 ,{ 120, 100, 120, 100, 30}
01710 }
01711 ,{{{ 120, 110, 120, 110, 30}
```

```
01712     , { 110, 100, 110, 100, 20}
01713     , { 120, 110, 120, 110, 30}
01714     , { 110, 100, 110, 100, 20}
01715     , { 40, 30, 40, 30, -60}
01716     }
01717     }
01718     }
01719     , {{{ 300, 290, 300, 260, 300}
01720     , { 300, 270, 300, 260, 300}
01721     , { 270, 230, 270, 220, 270}
01722     , { 270, 230, 270, 220, 270}
01723     , { 290, 290, 270, 220, 270}
01724     }
01725     , {{ 300, 270, 300, 260, 300}
01726     , { 300, 270, 300, 260, 300}
01727     , { 270, 230, 270, 220, 270}
01728     , { 230, 150, 230, 140, 220}
01729     , { 270, 230, 270, 220, 270}
01730     }
01731     , {{ 270, 230, 270, 220, 270}
01732     , { 270, 230, 270, 220, 270}
01733     , { 270, 230, 270, 220, 270}
01734     , { 270, 230, 270, 220, 270}
01735     , { 270, 230, 270, 220, 270}
01736     }
01737     , {{ 270, 230, 270, 220, 270}
01738     , { 270, 190, 270, 180, 260}
01739     , { 270, 230, 270, 220, 270}
01740     , { 210, 130, 140, 210, 150}
01741     , { 270, 230, 270, 220, 270}
01742     }
01743     , {{ 290, 290, 270, 220, 270}
01744     , { 270, 230, 270, 220, 270}
01745     , { 270, 230, 270, 220, 270}
01746     , { 270, 230, 270, 220, 270}
01747     , { 290, 290, 270, 220, 270}
01748     }
01749     }
01750     , {{{ 300, 290, 300, 190, 300}
01751     , { 300, 270, 300, 170, 300}
01752     , { 270, 230, 270, 190, 270}
01753     , { 270, 230, 270, 130, 270}
01754     , { 290, 290, 270, 190, 270}
01755     }
01756     , {{{ 300, 270, 300, 170, 300}
01757     , { 300, 270, 300, 170, 300}
01758     , { 270, 230, 270, 130, 270}
01759     , { 190, 150, 190, 50, 190}
01760     , { 270, 230, 270, 130, 270}
01761     }
01762     , {{{ 270, 230, 270, 190, 270}
01763     , { 270, 230, 270, 130, 270}
01764     , { 270, 230, 270, 190, 270}
01765     , { 270, 230, 270, 130, 270}
01766     , { 270, 230, 270, 190, 270}
01767     }
01768     , {{{ 270, 230, 270, 130, 270}
01769     , { 230, 190, 230, 90, 230}
01770     , { 270, 230, 270, 130, 270}
01771     , { 140, 100, 140, 130, 140}
01772     , { 270, 230, 270, 130, 270}
01773     }
01774     , {{{ 290, 290, 270, 190, 270}
01775     , { 270, 230, 270, 130, 270}
01776     , { 270, 230, 270, 190, 270}
01777     , { 270, 230, 270, 130, 270}
01778     , { 290, 290, 270, 130, 270}
01779     }
01780     }
01781     , {{{ 290, 260, 290, 260, 270}
01782     , { 290, 260, 290, 260, 270}
01783     , { 250, 220, 250, 220, 240}
01784     , { 250, 220, 250, 220, 240}
01785     , { 250, 220, 250, 220, 240}
01786     }
01787     , {{{ 290, 260, 290, 260, 270}
01788     , { 290, 260, 290, 260, 270}
01789     , { 250, 220, 250, 220, 240}
01790     , { 230, 140, 230, 140, 220}
01791     , { 250, 220, 250, 220, 240}
01792     }
01793     , {{{ 250, 220, 250, 220, 240}
01794     , { 250, 220, 250, 220, 240}
01795     , { 250, 220, 250, 220, 240}
01796     , { 250, 220, 250, 220, 240}
01797     , { 250, 220, 250, 220, 240}
01798     }
```

```
01799 ,{{ 270, 220, 270, 220, 260}
01800 ,{ 270, 180, 270, 180, 260}
01801 ,{ 250, 220, 250, 220, 240}
01802 ,{ 120, 90, 120, 90, 110}
01803 ,{ 250, 220, 250, 220, 240}
01804 }
01805 ,{{ 250, 220, 250, 220, 240}
01806 ,{ 250, 220, 250, 220, 240}
01807 ,{ 250, 220, 250, 220, 240}
01808 ,{ 250, 220, 250, 220, 240}
01809 ,{ 250, 220, 250, 220, 240}
01810 }
01811 }
01812 ,{{{ 300, 190, 300, 210, 300}
01813 ,{ 300, 170, 300, 170, 300}
01814 ,{ 270, 190, 270, 80, 270}
01815 ,{ 270, 130, 270, 210, 270}
01816 ,{ 270, 190, 270, 210, 270}
01817 }
01818 ,{{ 300, 170, 300, 130, 300}
01819 ,{ 300, 170, 300, 110, 300}
01820 ,{ 270, 130, 270, 80, 270}
01821 ,{ 190, 50, 190, 130, 190}
01822 ,{ 270, 130, 270, 80, 270}
01823 }
01824 ,{{ 270, 190, 270, 80, 270}
01825 ,{ 270, 130, 270, 80, 270}
01826 ,{ 270, 190, 270, 80, 270}
01827 ,{ 270, 130, 270, 80, 270}
01828 ,{ 270, 190, 270, 80, 270}
01829 }
01830 ,{{ 270, 130, 270, 210, 270}
01831 ,{ 230, 90, 230, 170, 230}
01832 ,{ 270, 130, 270, 80, 270}
01833 ,{ 210, 130, 140, 210, 140}
01834 ,{ 270, 130, 270, 80, 270}
01835 }
01836 ,{{ 270, 190, 270, 210, 270}
01837 ,{ 270, 130, 270, 80, 270}
01838 ,{ 270, 190, 270, 80, 270}
01839 ,{ 270, 130, 270, 80, 270}
01840 ,{ 270, 130, 270, 210, 270}
01841 }
01842 }
01843 ,{{{ 270, 260, 270, 260, 240}
01844 ,{ 270, 260, 270, 260, 240}
01845 ,{ 240, 220, 240, 220, 150}
01846 ,{ 240, 220, 240, 220, 150}
01847 ,{ 240, 220, 240, 220, 150}
01848 }
01849 ,{{ 270, 260, 270, 260, 240}
01850 ,{ 270, 260, 270, 260, 240}
01851 ,{ 240, 220, 240, 220, 150}
01852 ,{ 220, 140, 220, 140, 70}
01853 ,{ 240, 220, 240, 220, 150}
01854 }
01855 ,{{ 240, 220, 240, 220, 150}
01856 ,{ 240, 220, 240, 220, 150}
01857 ,{ 240, 220, 240, 220, 150}
01858 ,{ 240, 220, 240, 220, 150}
01859 ,{ 240, 220, 240, 220, 150}
01860 }
01861 ,{{ 260, 220, 260, 220, 150}
01862 ,{ 260, 180, 260, 180, 110}
01863 ,{ 240, 220, 240, 220, 150}
01864 ,{ 150, 90, 110, 90, 150}
01865 ,{ 240, 220, 240, 220, 150}
01866 }
01867 ,{{ 240, 220, 240, 220, 150}
01868 ,{ 240, 220, 240, 220, 150}
01869 ,{ 240, 220, 240, 220, 150}
01870 ,{ 240, 220, 240, 220, 150}
01871 ,{ 240, 220, 240, 220, 150}
01872 }
01873 }
01874 }
01875 ,{{{ 310, 260, 310, 220, 300}
01876 ,{ 310, 230, 310, 220, 300}
01877 ,{ 240, 200, 240, 190, 240}
01878 ,{ 240, 200, 240, 190, 240}
01879 ,{ 260, 260, 240, 190, 240}
01880 }
01881 ,{{ 240, 200, 240, 190, 240}
01882 ,{ 200, 160, 200, 160, 200}
01883 ,{ 240, 200, 240, 190, 240}
01884 ,{ 150, 60, 150, 60, 130}
01885 ,{ 240, 200, 240, 190, 240}
```

```
01886     }
01887     ,{{ 240, 200, 240, 190, 240}
01888     ,{ 240, 200, 240, 190, 240}
01889     ,{ 240, 200, 240, 190, 240}
01890     ,{ 240, 200, 240, 190, 240}
01891     ,{ 240, 200, 240, 190, 240}
01892     }
01893     ,{{ 310, 230, 310, 220, 300}
01894     ,{ 310, 230, 310, 220, 300}
01895     ,{ 240, 200, 240, 190, 240}
01896     ,{ 180, 100, 110, 180, 120}
01897     ,{ 240, 200, 240, 190, 240}
01898     }
01899     ,{{ 260, 260, 240, 190, 240}
01900     ,{ 240, 200, 240, 190, 240}
01901     ,{ 240, 200, 240, 190, 240}
01902     ,{ 240, 200, 240, 190, 240}
01903     ,{ 260, 260, 240, 190, 240}
01904     }
01905     }
01906     ,{{{ 270, 260, 270, 160, 270}
01907     ,{ 270, 230, 270, 130, 270}
01908     ,{ 240, 200, 240, 160, 240}
01909     ,{ 240, 200, 240, 100, 240}
01910     ,{ 260, 260, 240, 160, 240}
01911     }
01912     ,{{ 240, 200, 240, 100, 240}
01913     ,{ 200, 160, 200, 70, 200}
01914     ,{ 240, 200, 240, 100, 240}
01915     ,{ 100, 60, 100, -30, 100}
01916     ,{ 240, 200, 240, 100, 240}
01917     }
01918     ,{{{ 240, 200, 240, 160, 240}
01919     ,{ 240, 200, 240, 100, 240}
01920     ,{ 240, 200, 240, 160, 240}
01921     ,{ 240, 200, 240, 100, 240}
01922     ,{ 240, 200, 240, 160, 240}
01923     }
01924     ,{{{ 270, 230, 270, 130, 270}
01925     ,{ 270, 230, 270, 130, 270}
01926     ,{ 240, 200, 240, 100, 240}
01927     ,{ 110, 70, 110, 100, 110}
01928     ,{ 240, 200, 240, 100, 240}
01929     }
01930     ,{{{ 260, 260, 240, 160, 240}
01931     ,{ 240, 200, 240, 100, 240}
01932     ,{ 240, 200, 240, 160, 240}
01933     ,{ 240, 200, 240, 100, 240}
01934     ,{ 260, 260, 240, 100, 240}
01935     }
01936     }
01937     ,{{{ 310, 220, 310, 220, 300}
01938     ,{ 310, 220, 310, 220, 300}
01939     ,{ 220, 190, 220, 190, 210}
01940     ,{ 220, 190, 220, 190, 210}
01941     ,{ 220, 190, 220, 190, 210}
01942     }
01943     ,{{ 220, 190, 220, 190, 210}
01944     ,{ 190, 160, 190, 160, 170}
01945     ,{ 220, 190, 220, 190, 210}
01946     ,{ 150, 60, 150, 60, 130}
01947     ,{ 220, 190, 220, 190, 210}
01948     }
01949     ,{{ 220, 190, 220, 190, 210}
01950     ,{ 220, 190, 220, 190, 210}
01951     ,{ 220, 190, 220, 190, 210}
01952     ,{ 220, 190, 220, 190, 210}
01953     ,{ 220, 190, 220, 190, 210}
01954     }
01955     ,{{ 310, 220, 310, 220, 300}
01956     ,{ 310, 220, 310, 220, 300}
01957     ,{ 220, 190, 220, 190, 210}
01958     ,{ 90, 60, 90, 60, 80}
01959     ,{ 220, 190, 220, 190, 210}
01960     }
01961     ,{{ 220, 190, 220, 190, 210}
01962     ,{ 220, 190, 220, 190, 210}
01963     ,{ 220, 190, 220, 190, 210}
01964     ,{ 220, 190, 220, 190, 210}
01965     ,{ 220, 190, 220, 190, 210}
01966     }
01967     }
01968     ,{{{ 270, 160, 270, 210, 270}
01969     ,{ 270, 130, 270, 210, 270}
01970     ,{ 240, 160, 240, 50, 240}
01971     ,{ 240, 100, 240, 180, 240}
01972     ,{ 240, 160, 240, 180, 240}
```

```
01973      }
01974      ,{{ 240, 100, 240, 50, 240}
01975      ,{ 200, 70, 200, 10, 200}
01976      ,{ 240, 100, 240, 50, 240}
01977      ,{ 100, -30, 100, 40, 100}
01978      ,{ 240, 100, 240, 50, 240}
01979      }
01980      ,{{ 240, 160, 240, 50, 240}
01981      ,{ 240, 100, 240, 50, 240}
01982      ,{ 240, 160, 240, 50, 240}
01983      ,{ 240, 100, 240, 50, 240}
01984      ,{ 240, 160, 240, 50, 240}
01985      }
01986      ,{{ 270, 130, 270, 210, 270}
01987      ,{ 270, 130, 270, 210, 270}
01988      ,{ 240, 100, 240, 50, 240}
01989      ,{ 180, 100, 110, 180, 110}
01990      ,{ 240, 100, 240, 50, 240}
01991      }
01992      ,{{ 240, 160, 240, 180, 240}
01993      ,{ 240, 100, 240, 50, 240}
01994      ,{ 240, 160, 240, 50, 240}
01995      ,{ 240, 100, 240, 50, 240}
01996      ,{ 240, 100, 240, 180, 240}
01997      }
01998      }
01999      ,{{{ 300, 220, 300, 220, 150}
02000      ,{ 300, 220, 300, 220, 150}
02001      ,{ 210, 190, 210, 190, 120}
02002      ,{ 210, 190, 210, 190, 120}
02003      ,{ 210, 190, 210, 190, 120}
02004      }
02005      ,{{{ 210, 190, 210, 190, 140}
02006      ,{ 170, 160, 170, 160, 140}
02007      ,{ 210, 190, 210, 190, 120}
02008      ,{ 130, 60, 130, 60, -10}
02009      ,{ 210, 190, 210, 190, 120}
02010      }
02011      ,{{{ 210, 190, 210, 190, 120}
02012      ,{ 210, 190, 210, 190, 120}
02013      ,{ 210, 190, 210, 190, 120}
02014      ,{ 210, 190, 210, 190, 120}
02015      ,{ 210, 190, 210, 190, 120}
02016      }
02017      ,{{{ 300, 220, 300, 220, 150}
02018      ,{ 300, 220, 300, 220, 150}
02019      ,{ 210, 190, 210, 190, 120}
02020      ,{ 120, 60, 80, 60, 120}
02021      ,{ 210, 190, 210, 190, 120}
02022      }
02023      ,{{{ 210, 190, 210, 190, 120}
02024      ,{ 210, 190, 210, 190, 120}
02025      ,{ 210, 190, 210, 190, 120}
02026      ,{ 210, 190, 210, 190, 120}
02027      ,{ 210, 190, 210, 190, 120}
02028      }
02029      }
02030      }
02031      ,{{{ 240, 200, 240, 190, 240}
02032      ,{ 240, 200, 240, 190, 240}
02033      ,{ 220, 180, 220, 170, 220}
02034      ,{ 220, 180, 220, 180, 220}
02035      ,{ 220, 180, 220, 170, 220}
02036      }
02037      ,{{{ 240, 200, 240, 190, 240}
02038      ,{ 240, 200, 240, 190, 240}
02039      ,{ 210, 170, 210, 170, 210}
02040      ,{ 160, 70, 160, 70, 140}
02041      ,{ 210, 170, 210, 170, 210}
02042      }
02043      ,{{{ 220, 180, 220, 180, 220}
02044      ,{ 220, 180, 220, 180, 220}
02045      ,{ 220, 180, 220, 170, 220}
02046      ,{ 220, 180, 220, 180, 220}
02047      ,{ 220, 180, 220, 170, 220}
02048      }
02049      ,{{{ 230, 170, 230, 170, 210}
02050      ,{ 230, 140, 230, 140, 210}
02051      ,{ 210, 170, 210, 170, 210}
02052      ,{ 130, 60, 60, 130, 70}
02053      ,{ 210, 170, 210, 170, 210}
02054      }
02055      ,{{{ 220, 180, 220, 180, 220}
02056      ,{ 220, 180, 220, 180, 220}
02057      ,{ 220, 180, 220, 170, 220}
02058      ,{ 220, 180, 220, 180, 220}
02059      ,{ 150, 150, 130, 80, 130}
```



```
02060     }
02061     }
02062     ,{{{ 240, 200, 240, 140, 240}
02063     ,{ 240, 200, 240, 100, 240}
02064     ,{ 220, 180, 220, 140, 220}
02065     ,{ 220, 180, 220, 90, 220}
02066     ,{ 220, 180, 220, 140, 220}
02067     }
02068     ,{{{ 240, 200, 240, 100, 240}
02069     ,{ 240, 200, 240, 100, 240}
02070     ,{ 210, 170, 210, 80, 210}
02071     ,{ 110, 70, 110, -20, 110}
02072     ,{ 210, 170, 210, 80, 210}
02073     }
02074     ,{{{ 220, 180, 220, 140, 220}
02075     ,{ 220, 180, 220, 90, 220}
02076     ,{ 220, 180, 220, 140, 220}
02077     ,{ 220, 180, 220, 90, 220}
02078     ,{ 220, 180, 220, 140, 220}
02079     }
02080     ,{{{ 210, 170, 210, 80, 210}
02081     ,{ 180, 140, 180, 50, 180}
02082     ,{ 210, 170, 210, 80, 210}
02083     ,{ 60, 20, 60, 60, 60}
02084     ,{ 210, 170, 210, 80, 210}
02085     }
02086     ,{{{ 220, 180, 220, 140, 220}
02087     ,{ 220, 180, 220, 90, 220}
02088     ,{ 220, 180, 220, 140, 220}
02089     ,{ 220, 180, 220, 90, 220}
02090     ,{ 150, 150, 130, 0, 130}
02091     }
02092     }
02093     ,{{{ 230, 190, 230, 190, 210}
02094     ,{ 230, 190, 230, 190, 210}
02095     ,{ 200, 170, 200, 170, 190}
02096     ,{ 210, 180, 210, 180, 190}
02097     ,{ 200, 170, 200, 170, 190}
02098     }
02099     ,{{{ 220, 190, 220, 190, 210}
02100     ,{ 220, 190, 220, 190, 210}
02101     ,{ 200, 170, 200, 170, 180}
02102     ,{ 160, 70, 160, 70, 140}
02103     ,{ 200, 170, 200, 170, 180}
02104     }
02105     ,{{{ 210, 180, 210, 180, 190}
02106     ,{ 210, 180, 210, 180, 190}
02107     ,{ 200, 170, 200, 170, 190}
02108     ,{ 210, 180, 210, 180, 190}
02109     ,{ 200, 170, 200, 170, 190}
02110     }
02111     ,{{{ 230, 170, 230, 170, 210}
02112     ,{ 230, 140, 230, 140, 210}
02113     ,{ 200, 170, 200, 170, 180}
02114     ,{ 50, 20, 50, 20, 30}
02115     ,{ 200, 170, 200, 170, 180}
02116     }
02117     ,{{{ 210, 180, 210, 180, 190}
02118     ,{ 210, 180, 210, 180, 190}
02119     ,{ 200, 170, 200, 170, 190}
02120     ,{ 210, 180, 210, 180, 190}
02121     ,{ 110, 80, 110, 80, 100}
02122     }
02123     }
02124     ,{{{ 240, 140, 240, 130, 240}
02125     ,{ 240, 100, 240, 120, 240}
02126     ,{ 220, 140, 220, 30, 220}
02127     ,{ 220, 90, 220, 130, 220}
02128     ,{ 220, 140, 220, 70, 220}
02129     }
02130     ,{{{ 240, 100, 240, 50, 240}
02131     ,{ 240, 100, 240, 50, 240}
02132     ,{ 210, 80, 210, 20, 210}
02133     ,{ 110, -20, 110, 50, 110}
02134     ,{ 210, 80, 210, 20, 210}
02135     }
02136     ,{{{ 220, 140, 220, 30, 220}
02137     ,{ 220, 90, 220, 30, 220}
02138     ,{ 220, 140, 220, 30, 220}
02139     ,{ 220, 90, 220, 30, 220}
02140     ,{ 220, 140, 220, 30, 220}
02141     }
02142     ,{{{ 210, 80, 210, 130, 210}
02143     ,{ 180, 50, 180, 120, 180}
02144     ,{ 210, 80, 210, 20, 210}
02145     ,{ 130, 60, 130, 60, 130}
02146     ,{ 210, 80, 210, 20, 210}
```

```
02147     }
02148     ,{{ 220, 140, 220, 70, 220}
02149     ,{ 220, 90, 220, 30, 220}
02150     ,{ 220, 140, 220, 30, 220}
02151     ,{ 220, 90, 220, 30, 220}
02152     ,{ 130, 0, 130, 70, 130}
02153     }
02154     }
02155     ,{{{ 210, 190, 210, 190, 180}
02156     ,{ 210, 190, 210, 190, 180}
02157     ,{ 190, 170, 190, 170, 100}
02158     ,{ 190, 180, 190, 180, 100}
02159     ,{ 190, 170, 190, 170, 100}
02160     }
02161     ,{{{ 210, 190, 210, 190, 180}
02162     ,{ 210, 190, 210, 190, 180}
02163     ,{ 180, 170, 180, 170, 90}
02164     ,{ 140, 70, 140, 70, 0}
02165     ,{ 180, 170, 180, 170, 90}
02166     }
02167     ,{{{ 190, 180, 190, 180, 100}
02168     ,{ 190, 180, 190, 180, 100}
02169     ,{ 190, 170, 190, 170, 100}
02170     ,{ 190, 180, 190, 180, 100}
02171     ,{ 190, 170, 190, 170, 100}
02172     }
02173     ,{{{ 210, 170, 210, 170, 90}
02174     ,{ 210, 140, 210, 140, 60}
02175     ,{ 180, 170, 180, 170, 90}
02176     ,{ 70, 20, 30, 20, 70}
02177     ,{ 180, 170, 180, 170, 90}
02178     }
02179     ,{{{ 190, 180, 190, 180, 100}
02180     ,{ 190, 180, 190, 180, 100}
02181     ,{ 190, 170, 190, 170, 100}
02182     ,{ 190, 180, 190, 180, 100}
02183     ,{ 100, 80, 100, 80, 10}
02184     }
02185     }
02186     }
02187     ,{{{ 240, 200, 240, 190, 240}
02188     ,{ 240, 200, 240, 190, 240}
02189     ,{ 240, 200, 240, 190, 240}
02190     ,{ 240, 200, 240, 190, 240}
02191     ,{ 240, 200, 240, 190, 240}
02192     }
02193     ,{{{ 240, 200, 240, 190, 240}
02194     ,{ 240, 200, 240, 190, 240}
02195     ,{ 190, 150, 190, 150, 190}
02196     ,{ 180, 90, 180, 90, 160}
02197     ,{ 190, 150, 190, 150, 190}
02198     }
02199     ,{{{ 240, 200, 240, 190, 240}
02200     ,{ 240, 200, 240, 190, 240}
02201     ,{ 240, 200, 240, 190, 240}
02202     ,{ 240, 200, 240, 190, 240}
02203     ,{ 240, 200, 240, 190, 240}
02204     }
02205     ,{{{ 190, 150, 190, 150, 190}
02206     ,{ 190, 100, 190, 100, 170}
02207     ,{ 190, 150, 190, 150, 190}
02208     ,{ 150, 80, 150, 150, 90}
02209     ,{ 190, 150, 190, 150, 190}
02210     }
02211     ,{{{ 240, 200, 240, 190, 240}
02212     ,{ 240, 200, 240, 190, 240}
02213     ,{ 210, 170, 210, 160, 210}
02214     ,{ 240, 200, 240, 190, 240}
02215     ,{ 170, 170, 150, 110, 150}
02216     }
02217     }
02218     ,{{{ 240, 200, 240, 160, 240}
02219     ,{ 240, 200, 240, 100, 240}
02220     ,{ 240, 200, 240, 160, 240}
02221     ,{ 240, 200, 240, 100, 240}
02222     ,{ 240, 200, 240, 160, 240}
02223     }
02224     ,{{{ 240, 200, 240, 100, 240}
02225     ,{ 240, 200, 240, 100, 240}
02226     ,{ 190, 150, 190, 60, 190}
02227     ,{ 130, 90, 130, 0, 130}
02228     ,{ 190, 150, 190, 60, 190}
02229     }
02230     ,{{{ 240, 200, 240, 160, 240}
02231     ,{ 240, 200, 240, 100, 240}
02232     ,{ 240, 200, 240, 160, 240}
02233     ,{ 240, 200, 240, 100, 240}
```

```

02234     , { 240, 200, 240, 160, 240}
02235     }
02236     , { { 190, 150, 190, 80, 190}
02237     , { 140, 100, 140, 10, 140}
02238     , { 190, 150, 190, 60, 190}
02239     , { 80, 40, 80, 80, 80}
02240     , { 190, 150, 190, 60, 190}
02241     }
02242     , { { 240, 200, 240, 130, 240}
02243     , { 240, 200, 240, 100, 240}
02244     , { 210, 170, 210, 130, 210}
02245     , { 240, 200, 240, 100, 240}
02246     , { 170, 170, 150, 20, 150}
02247     }
02248     }
02249     , { { { 220, 190, 220, 190, 210}
02250     , { 220, 190, 220, 190, 210}
02251     , { 220, 190, 220, 190, 210}
02252     , { 220, 190, 220, 190, 210}
02253     , { 220, 190, 220, 190, 210}
02254     }
02255     , { { 220, 190, 220, 190, 210}
02256     , { 220, 190, 220, 190, 210}
02257     , { 180, 150, 180, 150, 160}
02258     , { 180, 90, 180, 90, 160}
02259     , { 180, 150, 180, 150, 160}
02260     }
02261     , { { 220, 190, 220, 190, 210}
02262     , { 220, 190, 220, 190, 210}
02263     , { 220, 190, 220, 190, 210}
02264     , { 220, 190, 220, 190, 210}
02265     , { 220, 190, 220, 190, 210}
02266     }
02267     , { { 190, 150, 190, 150, 170}
02268     , { 190, 100, 190, 100, 170}
02269     , { 180, 150, 180, 150, 160}
02270     , { 70, 40, 70, 40, 50}
02271     , { 180, 150, 180, 150, 160}
02272     }
02273     , { { 220, 190, 220, 190, 210}
02274     , { 220, 190, 220, 190, 210}
02275     , { 190, 160, 190, 160, 180}
02276     , { 220, 190, 220, 190, 210}
02277     , { 140, 110, 140, 110, 120}
02278     }
02279     }
02280     , { { { 240, 160, 240, 150, 240}
02281     , { 240, 100, 240, 80, 240}
02282     , { 240, 160, 240, 50, 240}
02283     , { 240, 100, 240, 150, 240}
02284     , { 240, 160, 240, 90, 240}
02285     }
02286     , { { 240, 100, 240, 70, 240}
02287     , { 240, 100, 240, 50, 240}
02288     , { 190, 60, 190, 0, 190}
02289     , { 130, 0, 130, 70, 130}
02290     , { 190, 60, 190, 0, 190}
02291     }
02292     , { { 240, 160, 240, 50, 240}
02293     , { 240, 100, 240, 50, 240}
02294     , { 240, 160, 240, 50, 240}
02295     , { 240, 100, 240, 50, 240}
02296     , { 240, 160, 240, 50, 240}
02297     }
02298     , { { 190, 80, 190, 150, 190}
02299     , { 140, 10, 140, 80, 140}
02300     , { 190, 60, 190, 0, 190}
02301     , { 150, 80, 150, 150, 80}
02302     , { 190, 60, 190, 0, 190}
02303     }
02304     , { { 240, 130, 240, 90, 240}
02305     , { 240, 100, 240, 50, 240}
02306     , { 210, 130, 210, 20, 210}
02307     , { 240, 100, 240, 50, 240}
02308     , { 150, 20, 150, 90, 150}
02309     }
02310     }
02311     , { { { 210, 190, 210, 190, 180}
02312     , { 210, 190, 210, 190, 180}
02313     , { 210, 190, 210, 190, 120}
02314     , { 210, 190, 210, 190, 120}
02315     , { 210, 190, 210, 190, 120}
02316     }
02317     , { { 210, 190, 210, 190, 180}
02318     , { 210, 190, 210, 190, 180}
02319     , { 160, 150, 160, 150, 70}
02320     , { 160, 90, 160, 90, 10}

```

```
02321      , { 160, 150, 160, 150, 70}
02322      }
02323      , {{ 210, 190, 210, 190, 120}
02324      , { 210, 190, 210, 190, 120}
02325      , { 210, 190, 210, 190, 120}
02326      , { 210, 190, 210, 190, 120}
02327      , { 210, 190, 210, 190, 120}
02328      }
02329      , {{ 170, 150, 170, 150, 90}
02330      , { 170, 100, 170, 100, 20}
02331      , { 160, 150, 160, 150, 70}
02332      , { 90, 40, 50, 40, 90}
02333      , { 160, 150, 160, 150, 70}
02334      }
02335      , {{ 210, 190, 210, 190, 120}
02336      , { 210, 190, 210, 190, 120}
02337      , { 180, 160, 180, 160, 90}
02338      , { 210, 190, 210, 190, 120}
02339      , { 120, 110, 120, 110, 30}
02340      }
02341      }
02342      }
02343      , {{{ 310, 290, 310, 260, 300}
02344      , { 310, 270, 310, 260, 300}
02345      , { 270, 230, 270, 220, 270}
02346      , { 270, 230, 270, 220, 270}
02347      , { 290, 290, 270, 220, 270}
02348      }
02349      , {{ 300, 270, 300, 260, 300}
02350      , { 300, 270, 300, 260, 300}
02351      , { 270, 230, 270, 220, 270}
02352      , { 230, 150, 230, 140, 220}
02353      , { 270, 230, 270, 220, 270}
02354      }
02355      , {{ 270, 230, 270, 220, 270}
02356      , { 270, 230, 270, 220, 270}
02357      , { 270, 230, 270, 220, 270}
02358      , { 270, 230, 270, 220, 270}
02359      , { 270, 230, 270, 220, 270}
02360      }
02361      , {{ 310, 230, 310, 220, 300}
02362      , { 310, 230, 310, 220, 300}
02363      , { 270, 230, 270, 220, 270}
02364      , { 210, 130, 140, 210, 150}
02365      , { 270, 230, 270, 220, 270}
02366      }
02367      , {{{ 290, 290, 270, 220, 270}
02368      , { 270, 230, 270, 220, 270}
02369      , { 270, 230, 270, 220, 270}
02370      , { 270, 230, 270, 220, 270}
02371      , { 290, 290, 270, 220, 270}
02372      }
02373      }
02374      , {{{ 300, 290, 300, 190, 300}
02375      , { 300, 270, 300, 170, 300}
02376      , { 270, 230, 270, 190, 270}
02377      , { 270, 230, 270, 130, 270}
02378      , { 290, 290, 270, 190, 270}
02379      }
02380      , {{{ 300, 270, 300, 170, 300}
02381      , { 300, 270, 300, 170, 300}
02382      , { 270, 230, 270, 130, 270}
02383      , { 190, 150, 190, 50, 190}
02384      , { 270, 230, 270, 130, 270}
02385      }
02386      , {{{ 270, 230, 270, 190, 270}
02387      , { 270, 230, 270, 130, 270}
02388      , { 270, 230, 270, 190, 270}
02389      , { 270, 230, 270, 130, 270}
02390      , { 270, 230, 270, 190, 270}
02391      }
02392      , {{{ 270, 230, 270, 130, 270}
02393      , { 270, 230, 270, 130, 270}
02394      , { 270, 230, 270, 130, 270}
02395      , { 140, 100, 140, 130, 140}
02396      , { 270, 230, 270, 130, 270}
02397      }
02398      , {{{ 290, 290, 270, 190, 270}
02399      , { 270, 230, 270, 130, 270}
02400      , { 270, 230, 270, 190, 270}
02401      , { 270, 230, 270, 130, 270}
02402      , { 290, 290, 270, 130, 270}
02403      }
02404      }
02405      , {{{ 310, 260, 310, 260, 300}
02406      , { 310, 260, 310, 260, 300}
02407      , { 250, 220, 250, 220, 240}
```

```
02408     , { 250, 220, 250, 220, 240}
02409     , { 250, 220, 250, 220, 240}
02410     }
02411     , { { 290, 260, 290, 260, 270}
02412     , { 290, 260, 290, 260, 270}
02413     , { 250, 220, 250, 220, 240}
02414     , { 230, 140, 230, 140, 220}
02415     , { 250, 220, 250, 220, 240}
02416     }
02417     , { { 250, 220, 250, 220, 240}
02418     , { 250, 220, 250, 220, 240}
02419     , { 250, 220, 250, 220, 240}
02420     , { 250, 220, 250, 220, 240}
02421     , { 250, 220, 250, 220, 240}
02422     }
02423     , { { 310, 220, 310, 220, 300}
02424     , { 310, 220, 310, 220, 300}
02425     , { 250, 220, 250, 220, 240}
02426     , { 120, 90, 120, 90, 110}
02427     , { 250, 220, 250, 220, 240}
02428     }
02429     , { { 250, 220, 250, 220, 240}
02430     , { 250, 220, 250, 220, 240}
02431     , { 250, 220, 250, 220, 240}
02432     , { 250, 220, 250, 220, 240}
02433     , { 250, 220, 250, 220, 240}
02434     }
02435     }
02436     , { { { 300, 190, 300, 210, 300}
02437     , { 300, 170, 300, 210, 300}
02438     , { 270, 190, 270, 80, 270}
02439     , { 270, 130, 270, 210, 270}
02440     , { 270, 190, 270, 210, 270}
02441     }
02442     , { { 300, 170, 300, 130, 300}
02443     , { 300, 170, 300, 110, 300}
02444     , { 270, 130, 270, 80, 270}
02445     , { 190, 50, 190, 130, 190}
02446     , { 270, 130, 270, 80, 270}
02447     }
02448     , { { 270, 190, 270, 80, 270}
02449     , { 270, 130, 270, 80, 270}
02450     , { 270, 190, 270, 80, 270}
02451     , { 270, 130, 270, 80, 270}
02452     , { 270, 190, 270, 80, 270}
02453     }
02454     , { { 270, 130, 270, 210, 270}
02455     , { 270, 130, 270, 210, 270}
02456     , { 270, 130, 270, 80, 270}
02457     , { 210, 130, 140, 210, 140}
02458     , { 270, 130, 270, 80, 270}
02459     }
02460     , { { 270, 190, 270, 210, 270}
02461     , { 270, 130, 270, 80, 270}
02462     , { 270, 190, 270, 80, 270}
02463     , { 270, 130, 270, 80, 270}
02464     , { 270, 130, 270, 210, 270}
02465     }
02466     }
02467     , { { { 300, 260, 300, 260, 240}
02468     , { 300, 260, 300, 260, 240}
02469     , { 240, 220, 240, 220, 150}
02470     , { 240, 220, 240, 220, 150}
02471     , { 240, 220, 240, 220, 150}
02472     }
02473     , { { 270, 260, 270, 260, 240}
02474     , { 270, 260, 270, 260, 240}
02475     , { 240, 220, 240, 220, 150}
02476     , { 220, 140, 220, 140, 70}
02477     , { 240, 220, 240, 220, 150}
02478     }
02479     , { { 240, 220, 240, 220, 150}
02480     , { 240, 220, 240, 220, 150}
02481     , { 240, 220, 240, 220, 150}
02482     , { 240, 220, 240, 220, 150}
02483     , { 240, 220, 240, 220, 150}
02484     }
02485     , { { 300, 220, 300, 220, 150}
02486     , { 300, 220, 300, 220, 150}
02487     , { 240, 220, 240, 220, 150}
02488     , { 150, 90, 110, 90, 150}
02489     , { 240, 220, 240, 220, 150}
02490     }
02491     , { { 240, 220, 240, 220, 150}
02492     , { 240, 220, 240, 220, 150}
02493     , { 240, 220, 240, 220, 150}
02494     , { 240, 220, 240, 220, 150}
```

```
02495      , { 240, 220, 240, 220, 150}
02496      }
02497      }
02498      }
02499      }
02500      ,{{{ INF, INF, INF, INF, INF}
02501      , { INF, INF, INF, INF, INF}
02502      , { INF, INF, INF, INF, INF}
02503      , { INF, INF, INF, INF, INF}
02504      , { INF, INF, INF, INF, INF}
02505      }
02506      ,{{{ INF, INF, INF, INF, INF}
02507      , { INF, INF, INF, INF, INF}
02508      , { INF, INF, INF, INF, INF}
02509      , { INF, INF, INF, INF, INF}
02510      , { INF, INF, INF, INF, INF}
02511      }
02512      ,{{{ INF, INF, INF, INF, INF}
02513      , { INF, INF, INF, INF, INF}
02514      , { INF, INF, INF, INF, INF}
02515      , { INF, INF, INF, INF, INF}
02516      , { INF, INF, INF, INF, INF}
02517      }
02518      ,{{{ INF, INF, INF, INF, INF}
02519      , { INF, INF, INF, INF, INF}
02520      , { INF, INF, INF, INF, INF}
02521      , { INF, INF, INF, INF, INF}
02522      , { INF, INF, INF, INF, INF}
02523      }
02524      ,{{{ INF, INF, INF, INF, INF}
02525      , { INF, INF, INF, INF, INF}
02526      , { INF, INF, INF, INF, INF}
02527      , { INF, INF, INF, INF, INF}
02528      , { INF, INF, INF, INF, INF}
02529      }
02530      }
02531      ,{{{ INF, INF, INF, INF, INF}
02532      , { INF, INF, INF, INF, INF}
02533      , { INF, INF, INF, INF, INF}
02534      , { INF, INF, INF, INF, INF}
02535      , { INF, INF, INF, INF, INF}
02536      }
02537      ,{{{ INF, INF, INF, INF, INF}
02538      , { INF, INF, INF, INF, INF}
02539      , { INF, INF, INF, INF, INF}
02540      , { INF, INF, INF, INF, INF}
02541      , { INF, INF, INF, INF, INF}
02542      }
02543      ,{{{ INF, INF, INF, INF, INF}
02544      , { INF, INF, INF, INF, INF}
02545      , { INF, INF, INF, INF, INF}
02546      , { INF, INF, INF, INF, INF}
02547      , { INF, INF, INF, INF, INF}
02548      }
02549      ,{{{ INF, INF, INF, INF, INF}
02550      , { INF, INF, INF, INF, INF}
02551      , { INF, INF, INF, INF, INF}
02552      , { INF, INF, INF, INF, INF}
02553      , { INF, INF, INF, INF, INF}
02554      }
02555      ,{{{ INF, INF, INF, INF, INF}
02556      , { INF, INF, INF, INF, INF}
02557      , { INF, INF, INF, INF, INF}
02558      , { INF, INF, INF, INF, INF}
02559      , { INF, INF, INF, INF, INF}
02560      }
02561      }
02562      ,{{{ INF, INF, INF, INF, INF}
02563      , { INF, INF, INF, INF, INF}
02564      , { INF, INF, INF, INF, INF}
02565      , { INF, INF, INF, INF, INF}
02566      , { INF, INF, INF, INF, INF}
02567      }
02568      ,{{{ INF, INF, INF, INF, INF}
02569      , { INF, INF, INF, INF, INF}
02570      , { INF, INF, INF, INF, INF}
02571      , { INF, INF, INF, INF, INF}
02572      , { INF, INF, INF, INF, INF}
02573      }
02574      ,{{{ INF, INF, INF, INF, INF}
02575      , { INF, INF, INF, INF, INF}
02576      , { INF, INF, INF, INF, INF}
02577      , { INF, INF, INF, INF, INF}
02578      , { INF, INF, INF, INF, INF}
02579      }
02580      ,{{{ INF, INF, INF, INF, INF}
02581      , { INF, INF, INF, INF, INF}
```

```
02582     , {   INF,   INF,   INF,   INF,   INF }
02583     , {   INF,   INF,   INF,   INF,   INF }
02584     , {   INF,   INF,   INF,   INF,   INF }
02585     }
02586     , { {   INF,   INF,   INF,   INF,   INF }
02587     , {   INF,   INF,   INF,   INF,   INF }
02588     , {   INF,   INF,   INF,   INF,   INF }
02589     , {   INF,   INF,   INF,   INF,   INF }
02590     , {   INF,   INF,   INF,   INF,   INF }
02591     }
02592     }
02593     , { { {   INF,   INF,   INF,   INF,   INF }
02594     , {   INF,   INF,   INF,   INF,   INF }
02595     , {   INF,   INF,   INF,   INF,   INF }
02596     , {   INF,   INF,   INF,   INF,   INF }
02597     , {   INF,   INF,   INF,   INF,   INF }
02598     }
02599     , { {   INF,   INF,   INF,   INF,   INF }
02600     , {   INF,   INF,   INF,   INF,   INF }
02601     , {   INF,   INF,   INF,   INF,   INF }
02602     , {   INF,   INF,   INF,   INF,   INF }
02603     , {   INF,   INF,   INF,   INF,   INF }
02604     }
02605     , { {   INF,   INF,   INF,   INF,   INF }
02606     , {   INF,   INF,   INF,   INF,   INF }
02607     , {   INF,   INF,   INF,   INF,   INF }
02608     , {   INF,   INF,   INF,   INF,   INF }
02609     , {   INF,   INF,   INF,   INF,   INF }
02610     }
02611     , { {   INF,   INF,   INF,   INF,   INF }
02612     , {   INF,   INF,   INF,   INF,   INF }
02613     , {   INF,   INF,   INF,   INF,   INF }
02614     , {   INF,   INF,   INF,   INF,   INF }
02615     , {   INF,   INF,   INF,   INF,   INF }
02616     }
02617     , { {   INF,   INF,   INF,   INF,   INF }
02618     , {   INF,   INF,   INF,   INF,   INF }
02619     , {   INF,   INF,   INF,   INF,   INF }
02620     , {   INF,   INF,   INF,   INF,   INF }
02621     , {   INF,   INF,   INF,   INF,   INF }
02622     }
02623     }
02624     , { { {   INF,   INF,   INF,   INF,   INF }
02625     , {   INF,   INF,   INF,   INF,   INF }
02626     , {   INF,   INF,   INF,   INF,   INF }
02627     , {   INF,   INF,   INF,   INF,   INF }
02628     , {   INF,   INF,   INF,   INF,   INF }
02629     }
02630     , { {   INF,   INF,   INF,   INF,   INF }
02631     , {   INF,   INF,   INF,   INF,   INF }
02632     , {   INF,   INF,   INF,   INF,   INF }
02633     , {   INF,   INF,   INF,   INF,   INF }
02634     , {   INF,   INF,   INF,   INF,   INF }
02635     }
02636     , { {   INF,   INF,   INF,   INF,   INF }
02637     , {   INF,   INF,   INF,   INF,   INF }
02638     , {   INF,   INF,   INF,   INF,   INF }
02639     , {   INF,   INF,   INF,   INF,   INF }
02640     , {   INF,   INF,   INF,   INF,   INF }
02641     }
02642     , { {   INF,   INF,   INF,   INF,   INF }
02643     , {   INF,   INF,   INF,   INF,   INF }
02644     , {   INF,   INF,   INF,   INF,   INF }
02645     , {   INF,   INF,   INF,   INF,   INF }
02646     , {   INF,   INF,   INF,   INF,   INF }
02647     }
02648     , { {   INF,   INF,   INF,   INF,   INF }
02649     , {   INF,   INF,   INF,   INF,   INF }
02650     , {   INF,   INF,   INF,   INF,   INF }
02651     , {   INF,   INF,   INF,   INF,   INF }
02652     , {   INF,   INF,   INF,   INF,   INF }
02653     }
02654     }
02655     }
02656     , { { { {   220,   220,   190,   150,   150 }
02657     , {   170,   170,   150,   150,   150 }
02658     , {   220,   220,   190,   130,   140 }
02659     , {   170,   170,   150,   150,   150 }
02660     , {   140,   140,   120,   140,   120 }
02661     }
02662     , { {   150,   130,   110,   110,   150 }
02663     , {   150,   130,   110,   110,   150 }
02664     , {   130,   130,   110,   100,   110 }
02665     , {    90,    10,    70,    10,    90 }
02666     , {   130,   130,   100,   100,   110 }
02667     }
02668     , { {   220,   220,   190,   150,   150 }
```

```
02669      , { 150, 150, 150, 150, 150}
02670      , { 220, 220, 190, 130, 140}
02671      , { 170, 170, 150, 150, 150}
02672      , { 140, 140, 120, 120, 120}
02673      }
02674      , { { 140, 130, 100, 100, 140}
02675      , { 90, 10, 70, 10, 90}
02676      , { 130, 130, 100, 100, 110}
02677      , { 140, -10, 20, 80, 140}
02678      , { 130, 130, 100, 100, 110}
02679      }
02680      , { { 170, 170, 170, 150, 150}
02681      , { 170, 170, 150, 150, 150}
02682      , { 170, 140, 170, 120, 120}
02683      , { 170, 170, 150, 150, 150}
02684      , { 140, 140, 30, 140, 30}
02685      }
02686      }
02687      , { { { 220, 220, 190, 140, 140}
02688      , { 170, 170, 140, 40, 140}
02689      , { 220, 220, 190, 70, 130}
02690      , { 170, 170, 140, 30, 140}
02691      , { 140, 140, 110, 140, 110}
02692      }
02693      , { { 130, 130, 110, 70, 100}
02694      , { 130, 130, 100, 40, 100}
02695      , { 130, 130, 110, 70, 100}
02696      , { 70, -20, 70, -50, 10}
02697      , { 130, 130, 100, -10, 100}
02698      }
02699      , { { 220, 220, 190, 70, 140}
02700      , { 140, 60, 50, 30, 140}
02701      , { 220, 220, 190, 70, 130}
02702      , { 170, 170, 140, 30, 140}
02703      , { 140, 140, 110, 50, 110}
02704      }
02705      , { { 130, 130, 100, -10, 100}
02706      , { 10, 0, -100, -70, 10}
02707      , { 130, 130, 100, -10, 100}
02708      , { -10, -10, -50, -30, -50}
02709      , { 130, 130, 100, -10, 100}
02710      }
02711      , { { 170, 170, 140, 140, 140}
02712      , { 170, 170, 140, 30, 140}
02713      , { 140, 140, 110, 60, 110}
02714      , { 170, 170, 140, 30, 140}
02715      , { 140, 140, 30, 140, 20}
02716      }
02717      }
02718      , { { { 150, 150, 150, 150, 150}
02719      , { 150, 150, 150, 150, 150}
02720      , { 140, 130, 130, 130, 140}
02721      , { 150, 150, 150, 150, 150}
02722      , { 120, 120, 120, 120, 120}
02723      }
02724      , { { 110, 110, 110, 110, 110}
02725      , { 110, 110, 110, 110, 110}
02726      , { 110, 100, 100, 100, 110}
02727      , { 80, -40, 70, 10, 80}
02728      , { 110, 100, 100, 100, 110}
02729      }
02730      , { { 150, 150, 150, 150, 150}
02731      , { 150, 150, 150, 150, 150}
02732      , { 140, 130, 130, 130, 140}
02733      , { 150, 150, 150, 150, 150}
02734      , { 120, 120, 120, 120, 120}
02735      }
02736      , { { 110, 100, 100, 100, 110}
02737      , { 80, -70, -60, 10, 80}
02738      , { 110, 100, 100, 100, 110}
02739      , { -40, -40, -40, -40, -50}
02740      , { 110, 100, 100, 100, 110}
02741      }
02742      , { { 150, 150, 150, 150, 150}
02743      , { 150, 150, 150, 150, 150}
02744      , { 120, 120, 120, 120, 120}
02745      , { 150, 150, 150, 150, 150}
02746      , { 30, 30, 30, 30, 30}
02747      }
02748      }
02749      , { { { 140, 70, 140, 80, 140}
02750      , { 140, 10, 140, 10, 140}
02751      , { 130, 70, 130, 20, 130}
02752      , { 140, -30, 140, 80, 140}
02753      , { 110, 50, 110, 70, 110}
02754      }
02755      , { { 100, -30, 100, -30, 100}
```



```
02756     , { 100, -30, 100, -30, 100}
02757     , { 100, -70, 100, -40, 100}
02758     , { 10, -170, 10, -30, 10}
02759     , { 100, -70, 100, -40, 100}
02760     }
02761     , { { 140, 70, 140, 10, 140}
02762     , { 140, 10, 140, -30, 140}
02763     , { 130, 70, 130, -10, 130}
02764     , { 140, -30, 140, 10, 140}
02765     , { 110, 0, 110, -60, 110}
02766     }
02767     , { { 100, -70, 100, 80, 100}
02768     , { 10, -160, 10, 0, 10}
02769     , { 100, -70, 100, -40, 100}
02770     , { 80, -90, -50, 80, -50}
02771     , { 100, -70, 100, -40, 100}
02772     }
02773     , { { 140, 50, 140, 70, 140}
02774     , { 140, -30, 140, 10, 140}
02775     , { 110, 0, 110, 20, 110}
02776     , { 140, -30, 140, 10, 140}
02777     , { 70, 50, 20, 70, 20}
02778     }
02779     }
02780     , { { { 170, 150, 170, 150, 150}
02781     , { 150, 150, 150, 150, 150}
02782     , { 170, 130, 170, 130, 30}
02783     , { 150, 150, 150, 150, 140}
02784     , { 120, 120, 120, 120, 40}
02785     }
02786     , { { 150, 110, 110, 110, 150}
02787     , { 150, 110, 110, 110, 150}
02788     , { 100, 100, 100, 100, -20}
02789     , { 90, 10, 70, 10, 90}
02790     , { 100, 100, 100, 100, 30}
02791     }
02792     , { { 150, 150, 150, 150, 70}
02793     , { 150, 150, 150, 150, 0}
02794     , { 130, 130, 130, 130, -10}
02795     , { 150, 150, 150, 150, 70}
02796     , { 120, 120, 120, 120, 40}
02797     }
02798     , { { 140, 100, 100, 100, 140}
02799     , { 90, 10, 70, 10, 90}
02800     , { 100, 100, 100, 100, 30}
02801     , { 140, -40, 20, -40, 140}
02802     , { 100, 100, 100, 100, 30}
02803     }
02804     , { { 170, 150, 170, 150, 70}
02805     , { 150, 150, 150, 150, 70}
02806     , { 170, 120, 170, 120, 20}
02807     , { 150, 150, 150, 150, 70}
02808     , { 30, 30, 30, 30, -60}
02809     }
02810     }
02811     }
02812     , { { { { 150, 150, 120, 120, 130}
02813     , { 150, 150, 120, 120, 130}
02814     , { 130, 130, 100, 100, 110}
02815     , { 120, 120, 90, 90, 100}
02816     , { 120, 120, 100, 100, 100}
02817     }
02818     , { { { 150, 150, 120, 120, 130}
02819     , { 150, 150, 120, 120, 130}
02820     , { 120, 120, 100, 100, 100}
02821     , { -10, -50, -20, -80, -10}
02822     , { 120, 120, 100, 100, 100}
02823     }
02824     , { { { 120, 120, 100, 100, 100}
02825     , { 120, 120, 90, 90, 100}
02826     , { 120, 120, 100, 100, 100}
02827     , { 120, 120, 90, 90, 100}
02828     , { 120, 120, 100, 100, 100}
02829     }
02830     , { { { 120, 120, 100, 100, 100}
02831     , { 50, 10, 50, -10, 50}
02832     , { 120, 120, 100, 100, 100}
02833     , { 80, -20, -40, 80, 10}
02834     , { 120, 120, 100, 100, 100}
02835     }
02836     , { { { 130, 130, 100, 100, 110}
02837     , { 120, 120, 90, 90, 100}
02838     , { 130, 130, 100, 100, 110}
02839     , { 120, 120, 90, 90, 100}
02840     , { 110, 110, 20, 20, 30}
02841     }
02842     }
```

```
02843 ,{{{ 150, 150, 120, 50, 120}
02844 ,{ 150, 150, 120, 10, 120}
02845 ,{ 130, 130, 100, 50, 100}
02846 ,{ 120, 120, 90, -20, 90}
02847 ,{ 120, 120, 90, 50, 90}
02848 }
02849 ,{{{ 150, 150, 120, 10, 120}
02850 ,{ 150, 150, 120, 10, 120}
02851 ,{ 120, 120, 90, -10, 90}
02852 ,{ -50, -50, -80, -190, -80}
02853 ,{ 120, 120, 90, -10, 90}
02854 }
02855 ,{{{ 120, 120, 90, 50, 90}
02856 ,{ 120, 120, 90, -20, 90}
02857 ,{ 120, 120, 90, 50, 90}
02858 ,{ 120, 120, 90, -20, 90}
02859 ,{ 120, 120, 90, 50, 90}
02860 }
02861 ,{{{ 120, 120, 90, -10, 90}
02862 ,{ 10, 10, -20, -130, -20}
02863 ,{ 120, 120, 90, -10, 90}
02864 ,{ -20, -20, -50, -20, -50}
02865 ,{ 120, 120, 90, -10, 90}
02866 }
02867 ,{{{ 130, 130, 100, 50, 100}
02868 ,{ 120, 120, 90, -20, 90}
02869 ,{ 130, 130, 100, 50, 100}
02870 ,{ 120, 120, 90, -20, 90}
02871 ,{ 110, 110, 20, -90, 20}
02872 }
02873 }
02874 ,{{{ 130, 120, 120, 120, 130}
02875 ,{ 130, 120, 120, 120, 130}
02876 ,{ 110, 100, 100, 100, 110}
02877 ,{ 100, 90, 90, 90, 100}
02878 ,{ 100, 100, 100, 100, 100}
02879 }
02880 ,{{{ 130, 120, 120, 120, 130}
02881 ,{ 130, 120, 120, 120, 130}
02882 ,{ 100, 100, 100, 100, 100}
02883 ,{ -10, -80, -20, -80, -10}
02884 ,{ 100, 100, 100, 100, 100}
02885 }
02886 ,{{{ 100, 100, 100, 100, 100}
02887 ,{ 100, 90, 90, 90, 100}
02888 ,{ 100, 100, 100, 100, 100}
02889 ,{ 100, 90, 90, 90, 100}
02890 ,{ 100, 100, 100, 100, 100}
02891 }
02892 ,{{{ 100, 100, 100, 100, 100}
02893 ,{ 50, -10, 50, -10, 50}
02894 ,{ 100, 100, 100, 100, 100}
02895 ,{ -40, -40, -40, -40, -40}
02896 ,{ 100, 100, 100, 100, 100}
02897 }
02898 ,{{{ 110, 100, 100, 100, 110}
02899 ,{ 100, 90, 90, 90, 100}
02900 ,{ 110, 100, 100, 100, 110}
02901 ,{ 100, 90, 90, 90, 100}
02902 ,{ 30, 20, 20, 20, 30}
02903 }
02904 }
02905 ,{{{ 120, -10, 120, 80, 120}
02906 ,{ 120, -50, 120, -20, 120}
02907 ,{ 100, -10, 100, -40, 100}
02908 ,{ 90, -80, 90, 80, 90}
02909 ,{ 90, -20, 90, 10, 90}
02910 }
02911 ,{{{ 120, -50, 120, -20, 120}
02912 ,{ 120, -50, 120, -20, 120}
02913 ,{ 90, -80, 90, -40, 90}
02914 ,{ -80, -260, -80, -90, -80}
02915 ,{ 90, -80, 90, -40, 90}
02916 }
02917 ,{{{ 90, -20, 90, -40, 90}
02918 ,{ 90, -80, 90, -50, 90}
02919 ,{ 90, -20, 90, -40, 90}
02920 ,{ 90, -80, 90, -50, 90}
02921 ,{ 90, -20, 90, -40, 90}
02922 }
02923 ,{{{ 90, -80, 90, 80, 90}
02924 ,{ -20, -190, -20, -20, -20}
02925 ,{ 90, -80, 90, -40, 90}
02926 ,{ 80, -90, -50, 80, -50}
02927 ,{ 90, -80, 90, -40, 90}
02928 }
02929 ,{{{ 100, -10, 100, 10, 100}
```

```
02930     , {    90,   -80,    90,   -50,    90 }
02931     , {   100,   -10,   100,   -40,   100 }
02932     , {    90,   -80,    90,   -50,    90 }
02933     , {    20, -150,    20,    10,    20 }
02934     }
02935 }
02936 , {{{   120,   120,   120,   120,   110 }
02937     , {   120,   120,   120,   120,   110 }
02938     , {   100,   100,   100,   100,    30 }
02939     , {    90,    90,    90,    90,    20 }
02940     , {   100,   100,   100,   100,    20 }
02941     }
02942     , {{{   120,   120,   120,   120,   110 }
02943     , {   120,   120,   120,   120,   110 }
02944     , {   100,   100,   100,   100,    20 }
02945     , {   -20,   -80,   -20,   -80, -150 }
02946     , {   100,   100,   100,   100,    20 }
02947     }
02948     , {{{   100,   100,   100,   100,    20 }
02949     , {    90,    90,    90,    90,    20 }
02950     , {   100,   100,   100,   100,    20 }
02951     , {    90,    90,    90,    90,    20 }
02952     , {   100,   100,   100,   100,    20 }
02953     }
02954     , {{{   100,   100,   100,   100,    20 }
02955     , {    50,   -10,    50,   -10,   -90 }
02956     , {   100,   100,   100,   100,    20 }
02957     , {    10,   -40,   -40,   -40,    10 }
02958     , {   100,   100,   100,   100,    20 }
02959     }
02960     , {{{   100,   100,   100,   100,    30 }
02961     , {    90,    90,    90,    90,    20 }
02962     , {   100,   100,   100,   100,    30 }
02963     , {    90,    90,    90,    90,    20 }
02964     , {    20,    20,    20,    20,   -50 }
02965     }
02966     }
02967 }
02968 , {{{   300,   300,   250,   250,   260 }
02969     , {   280,   280,   250,   250,   260 }
02970     , {   240,   240,   220,   220,   220 }
02971     , {   240,   240,   220,   220,   220 }
02972     , {   300,   300,   220,   220,   220 }
02973     }
02974     , {{{   280,   280,   250,   250,   260 }
02975     , {   280,   280,   250,   250,   260 }
02976     , {   240,   240,   220,   220,   220 }
02977     , {   200,   160,   200,   140,   200 }
02978     , {   240,   240,   220,   220,   220 }
02979     }
02980     , {{{   240,   240,   220,   220,   220 }
02981     , {   240,   240,   220,   220,   220 }
02982     , {   240,   240,   220,   220,   220 }
02983     , {   240,   240,   220,   220,   220 }
02984     , {   240,   240,   220,   220,   220 }
02985     }
02986     , {{{   240,   240,   240,   220,   240 }
02987     , {   240,   200,   240,   180,   240 }
02988     , {   240,   240,   220,   220,   220 }
02989     , {   210,   110,    90,   210,   140 }
02990     , {   240,   240,   220,   220,   220 }
02991     }
02992     , {{{   300,   300,   220,   220,   220 }
02993     , {   240,   240,   220,   220,   220 }
02994     , {   240,   240,   220,   220,   220 }
02995     , {   240,   240,   220,   220,   220 }
02996     , {   300,   300,   220,   220,   220 }
02997     }
02998 }
02999 , {{{   300,   300,   250,   160,   250 }
03000     , {   280,   280,   250,   140,   250 }
03001     , {   240,   240,   210,   160,   210 }
03002     , {   240,   240,   210,   100,   210 }
03003     , {   300,   300,   210,   160,   210 }
03004     }
03005     , {{{   280,   280,   250,   140,   250 }
03006     , {   280,   280,   250,   140,   250 }
03007     , {   240,   240,   210,   100,   210 }
03008     , {   160,   160,   130,    20,   130 }
03009     , {   240,   240,   210,   100,   210 }
03010     }
03011     , {{{   240,   240,   210,   160,   210 }
03012     , {   240,   240,   210,   100,   210 }
03013     , {   240,   240,   210,   160,   210 }
03014     , {   240,   240,   210,   100,   210 }
03015     , {   240,   240,   210,   160,   210 }
03016     }
```

```
03017 ,{{ 240, 240, 210, 100, 210}
03018 ,{ 200, 200, 170, 60, 170}
03019 ,{ 240, 240, 210, 100, 210}
03020 ,{ 110, 110, 80, 100, 80}
03021 ,{ 240, 240, 210, 100, 210}
03022 }
03023 ,{{ 300, 300, 210, 160, 210}
03024 ,{ 240, 240, 210, 100, 210}
03025 ,{ 240, 240, 210, 160, 210}
03026 ,{ 240, 240, 210, 100, 210}
03027 ,{ 300, 300, 210, 100, 210}
03028 }
03029 }
03030 ,{{{ 260, 250, 250, 250, 260}
03031 ,{ 260, 250, 250, 250, 260}
03032 ,{ 220, 220, 220, 220, 220}
03033 ,{ 220, 220, 220, 220, 220}
03034 ,{ 220, 220, 220, 220, 220}
03035 }
03036 ,{{ 260, 250, 250, 250, 260}
03037 ,{ 260, 250, 250, 250, 260}
03038 ,{ 220, 220, 220, 220, 220}
03039 ,{ 200, 140, 200, 140, 200}
03040 ,{ 220, 220, 220, 220, 220}
03041 }
03042 ,{{ 220, 220, 220, 220, 220}
03043 ,{ 220, 220, 220, 220, 220}
03044 ,{ 220, 220, 220, 220, 220}
03045 ,{ 220, 220, 220, 220, 220}
03046 ,{ 220, 220, 220, 220, 220}
03047 }
03048 ,{{ 240, 220, 240, 220, 240}
03049 ,{ 240, 180, 240, 180, 240}
03050 ,{ 220, 220, 220, 220, 220}
03051 ,{ 90, 90, 90, 90, 90}
03052 ,{ 220, 220, 220, 220, 220}
03053 }
03054 ,{{ 220, 220, 220, 220, 220}
03055 ,{ 220, 220, 220, 220, 220}
03056 ,{ 220, 220, 220, 220, 220}
03057 ,{ 220, 220, 220, 220, 220}
03058 ,{ 220, 220, 220, 220, 220}
03059 }
03060 }
03061 ,{{{ 250, 100, 250, 210, 250}
03062 ,{ 250, 70, 250, 170, 250}
03063 ,{ 210, 100, 210, 80, 210}
03064 ,{ 210, 40, 210, 210, 210}
03065 ,{ 210, 100, 210, 210, 210}
03066 }
03067 ,{{{ 250, 70, 250, 130, 250}
03068 ,{ 250, 70, 250, 110, 250}
03069 ,{ 210, 40, 210, 80, 210}
03070 ,{ 130, -40, 130, 130, 130}
03071 ,{ 210, 40, 210, 80, 210}
03072 }
03073 ,{{{ 210, 100, 210, 80, 210}
03074 ,{ 210, 40, 210, 80, 210}
03075 ,{ 210, 100, 210, 80, 210}
03076 ,{ 210, 40, 210, 80, 210}
03077 ,{ 210, 100, 210, 80, 210}
03078 }
03079 ,{{{ 210, 40, 210, 210, 210}
03080 ,{ 170, 0, 170, 170, 170}
03081 ,{ 210, 40, 210, 80, 210}
03082 ,{ 210, 40, 80, 210, 80}
03083 ,{ 210, 40, 210, 80, 210}
03084 }
03085 ,{{{ 210, 100, 210, 210, 210}
03086 ,{ 210, 40, 210, 80, 210}
03087 ,{ 210, 100, 210, 80, 210}
03088 ,{ 210, 40, 210, 80, 210}
03089 ,{ 210, 40, 210, 210, 210}
03090 }
03091 }
03092 ,{{{ 250, 250, 250, 250, 240}
03093 ,{ 250, 250, 250, 250, 240}
03094 ,{ 220, 220, 220, 220, 140}
03095 ,{ 220, 220, 220, 220, 140}
03096 ,{ 220, 220, 220, 220, 140}
03097 }
03098 ,{{ 250, 250, 250, 250, 240}
03099 ,{ 250, 250, 250, 250, 240}
03100 ,{ 220, 220, 220, 220, 140}
03101 ,{ 200, 140, 200, 140, 60}
03102 ,{ 220, 220, 220, 220, 140}
03103 }
```

```

03104     ,{{ 220, 220, 220, 220, 140}
03105     ,{ 220, 220, 220, 220, 140}
03106     ,{ 220, 220, 220, 220, 140}
03107     ,{ 220, 220, 220, 220, 140}
03108     ,{ 220, 220, 220, 220, 140}
03109     }
03110     ,{{ 240, 220, 240, 220, 140}
03111     ,{ 240, 180, 240, 180, 100}
03112     ,{ 220, 220, 220, 220, 140}
03113     ,{ 140, 90, 90, 90, 140}
03114     ,{ 220, 220, 220, 220, 140}
03115     }
03116     ,{{ 220, 220, 220, 220, 140}
03117     ,{ 220, 220, 220, 220, 140}
03118     ,{ 220, 220, 220, 220, 140}
03119     ,{ 220, 220, 220, 220, 140}
03120     ,{ 220, 220, 220, 220, 140}
03121     }
03122     }
03123     }
03124     ,{{{ 280, 270, 280, 220, 280}
03125     ,{ 280, 240, 280, 220, 280}
03126     ,{ 210, 210, 190, 190, 190}
03127     ,{ 210, 210, 190, 190, 190}
03128     ,{ 270, 270, 190, 190, 190}
03129     }
03130     ,{{ 210, 210, 190, 190, 190}
03131     ,{ 190, 190, 150, 150, 160}
03132     ,{ 210, 210, 190, 190, 190}
03133     ,{ 120, 80, 110, 50, 120}
03134     ,{ 210, 210, 190, 190, 190}
03135     }
03136     ,{{ 210, 210, 190, 190, 190}
03137     ,{ 210, 210, 190, 190, 190}
03138     ,{ 210, 210, 190, 190, 190}
03139     ,{ 210, 210, 190, 190, 190}
03140     ,{ 210, 210, 190, 190, 190}
03141     }
03142     ,{{ 280, 240, 280, 220, 280}
03143     ,{ 280, 240, 280, 220, 280}
03144     ,{ 210, 210, 190, 190, 190}
03145     ,{ 180, 80, 60, 180, 110}
03146     ,{ 210, 210, 190, 190, 190}
03147     }
03148     ,{{{ 270, 270, 190, 190, 190}
03149     ,{ 210, 210, 190, 190, 190}
03150     ,{ 210, 210, 190, 190, 190}
03151     ,{ 210, 210, 190, 190, 190}
03152     ,{ 270, 270, 190, 190, 190}
03153     }
03154     }
03155     ,{{{ 270, 270, 210, 130, 210}
03156     ,{ 240, 240, 210, 100, 210}
03157     ,{ 210, 210, 180, 130, 180}
03158     ,{ 210, 210, 180, 70, 180}
03159     ,{ 270, 270, 180, 130, 180}
03160     }
03161     ,{{ 210, 210, 180, 70, 180}
03162     ,{ 190, 190, 150, 40, 150}
03163     ,{ 210, 210, 180, 70, 180}
03164     ,{ 80, 80, 50, -60, 50}
03165     ,{ 210, 210, 180, 70, 180}
03166     }
03167     ,{{ 210, 210, 180, 130, 180}
03168     ,{ 210, 210, 180, 70, 180}
03169     ,{ 210, 210, 180, 130, 180}
03170     ,{ 210, 210, 180, 70, 180}
03171     ,{ 210, 210, 180, 130, 180}
03172     }
03173     ,{{ 240, 240, 210, 100, 210}
03174     ,{ 240, 240, 210, 100, 210}
03175     ,{ 210, 210, 180, 70, 180}
03176     ,{ 80, 80, 50, 70, 50}
03177     ,{ 210, 210, 180, 70, 180}
03178     }
03179     ,{{{ 270, 270, 180, 130, 180}
03180     ,{ 210, 210, 180, 70, 180}
03181     ,{ 210, 210, 180, 130, 180}
03182     ,{ 210, 210, 180, 70, 180}
03183     ,{ 270, 270, 180, 70, 180}
03184     }
03185     }
03186     ,{{{ 280, 220, 280, 220, 280}
03187     ,{ 280, 220, 280, 220, 280}
03188     ,{ 190, 190, 190, 190, 190}
03189     ,{ 190, 190, 190, 190, 190}
03190     ,{ 190, 190, 190, 190, 190}

```

```
03191     }
03192     ,{{ 190, 190, 190, 190, 190}
03193     ,{ 160, 150, 150, 150, 160}
03194     ,{ 190, 190, 190, 190, 190}
03195     ,{ 120, 50, 110, 50, 120}
03196     ,{ 190, 190, 190, 190, 190}
03197     }
03198     ,{{ 190, 190, 190, 190, 190}
03199     ,{ 190, 190, 190, 190, 190}
03200     ,{ 190, 190, 190, 190, 190}
03201     ,{ 190, 190, 190, 190, 190}
03202     ,{ 190, 190, 190, 190, 190}
03203     }
03204     ,{{ 280, 220, 280, 220, 280}
03205     ,{ 280, 220, 280, 220, 280}
03206     ,{ 190, 190, 190, 190, 190}
03207     ,{ 60, 60, 60, 60, 60}
03208     ,{ 190, 190, 190, 190, 190}
03209     }
03210     ,{{ 190, 190, 190, 190, 190}
03211     ,{ 190, 190, 190, 190, 190}
03212     ,{ 190, 190, 190, 190, 190}
03213     ,{ 190, 190, 190, 190, 190}
03214     ,{ 190, 190, 190, 190, 190}
03215     }
03216     }
03217     ,{{{ 210, 70, 210, 210, 210}
03218     ,{ 210, 40, 210, 210, 210}
03219     ,{ 180, 70, 180, 50, 180}
03220     ,{ 180, 10, 180, 180, 180}
03221     ,{ 180, 70, 180, 180, 180}
03222     }
03223     ,{{{ 180, 10, 180, 50, 180}
03224     ,{ 150, -20, 150, 10, 150}
03225     ,{ 180, 10, 180, 50, 180}
03226     ,{ 50, -120, 50, 40, 50}
03227     ,{ 180, 10, 180, 50, 180}
03228     }
03229     ,{{{ 180, 70, 180, 50, 180}
03230     ,{ 180, 10, 180, 50, 180}
03231     ,{ 180, 70, 180, 50, 180}
03232     ,{ 180, 10, 180, 50, 180}
03233     ,{ 180, 70, 180, 50, 180}
03234     }
03235     ,{{{ 210, 40, 210, 210, 210}
03236     ,{ 210, 40, 210, 210, 210}
03237     ,{ 180, 10, 180, 50, 180}
03238     ,{ 180, 10, 50, 180, 50}
03239     ,{ 180, 10, 180, 50, 180}
03240     }
03241     ,{{{ 180, 70, 180, 180, 180}
03242     ,{ 180, 10, 180, 50, 180}
03243     ,{ 180, 70, 180, 50, 180}
03244     ,{ 180, 10, 180, 50, 180}
03245     ,{ 180, 10, 180, 180, 180}
03246     }
03247     }
03248     ,{{{ 280, 220, 280, 220, 140}
03249     ,{ 280, 220, 280, 220, 140}
03250     ,{ 190, 190, 190, 190, 110}
03251     ,{ 190, 190, 190, 190, 110}
03252     ,{ 190, 190, 190, 190, 110}
03253     }
03254     ,{{ 190, 190, 190, 190, 140}
03255     ,{ 150, 150, 150, 150, 140}
03256     ,{ 190, 190, 190, 190, 110}
03257     ,{ 110, 50, 110, 50, -20}
03258     ,{ 190, 190, 190, 190, 110}
03259     }
03260     ,{{ 190, 190, 190, 190, 110}
03261     ,{ 190, 190, 190, 190, 110}
03262     ,{ 190, 190, 190, 190, 110}
03263     ,{ 190, 190, 190, 190, 110}
03264     ,{ 190, 190, 190, 190, 110}
03265     }
03266     ,{{ 280, 220, 280, 220, 140}
03267     ,{ 280, 220, 280, 220, 140}
03268     ,{ 190, 190, 190, 190, 110}
03269     ,{ 110, 60, 60, 60, 110}
03270     ,{ 190, 190, 190, 190, 110}
03271     }
03272     ,{{ 190, 190, 190, 190, 110}
03273     ,{ 190, 190, 190, 190, 110}
03274     ,{ 190, 190, 190, 190, 110}
03275     ,{ 190, 190, 190, 190, 110}
03276     ,{ 190, 190, 190, 190, 110}
03277     }
```

```
03278     }
03279     }
03280     ,{{{ 210, 210, 190, 190, 200}
03281     ,{ 210, 210, 190, 190, 200}
03282     ,{ 190, 190, 170, 170, 170}
03283     ,{ 200, 200, 170, 170, 180}
03284     ,{ 190, 190, 170, 170, 170}
03285     }
03286     ,{{{ 210, 210, 190, 190, 190}
03287     ,{ 210, 210, 190, 190, 190}
03288     ,{ 190, 190, 160, 160, 170}
03289     ,{ 130, 90, 120, 60, 130}
03290     ,{ 190, 190, 160, 160, 170}
03291     }
03292     ,{{{ 200, 200, 170, 170, 180}
03293     ,{ 200, 200, 170, 170, 180}
03294     ,{ 190, 190, 170, 170, 170}
03295     ,{ 200, 200, 170, 170, 180}
03296     ,{ 190, 190, 170, 170, 170}
03297     }
03298     ,{{{ 200, 190, 190, 160, 200}
03299     ,{ 200, 160, 190, 130, 200}
03300     ,{ 190, 190, 160, 160, 170}
03301     ,{ 130, 40, 10, 130, 70}
03302     ,{ 190, 190, 160, 160, 170}
03303     }
03304     ,{{{ 200, 200, 170, 170, 180}
03305     ,{ 200, 200, 170, 170, 180}
03306     ,{ 190, 190, 170, 170, 170}
03307     ,{ 200, 200, 170, 170, 180}
03308     ,{ 160, 160, 80, 80, 80}
03309     }
03310     }
03311     ,{{{ 210, 210, 180, 110, 180}
03312     ,{ 210, 210, 180, 70, 180}
03313     ,{ 190, 190, 160, 110, 160}
03314     ,{ 200, 200, 170, 60, 170}
03315     ,{ 190, 190, 160, 110, 160}
03316     }
03317     ,{{{ 210, 210, 180, 70, 180}
03318     ,{ 210, 210, 180, 70, 180}
03319     ,{ 190, 190, 160, 50, 160}
03320     ,{ 90, 90, 60, -50, 60}
03321     ,{ 190, 190, 160, 50, 160}
03322     }
03323     ,{{{ 200, 200, 170, 110, 170}
03324     ,{ 200, 200, 170, 60, 170}
03325     ,{ 190, 190, 160, 110, 160}
03326     ,{ 200, 200, 170, 60, 170}
03327     ,{ 190, 190, 160, 110, 160}
03328     }
03329     ,{{{ 190, 190, 160, 50, 160}
03330     ,{ 160, 160, 130, 20, 130}
03331     ,{ 190, 190, 160, 50, 160}
03332     ,{ 40, 40, 10, 30, 10}
03333     ,{ 190, 190, 160, 50, 160}
03334     }
03335     ,{{{ 200, 200, 170, 110, 170}
03336     ,{ 200, 200, 170, 60, 170}
03337     ,{ 190, 190, 160, 110, 160}
03338     ,{ 200, 200, 170, 60, 170}
03339     ,{ 160, 160, 70, -30, 70}
03340     }
03341     }
03342     ,{{{ 200, 190, 190, 190, 200}
03343     ,{ 200, 190, 190, 190, 200}
03344     ,{ 170, 170, 170, 170, 170}
03345     ,{ 180, 170, 170, 170, 180}
03346     ,{ 170, 170, 170, 170, 170}
03347     }
03348     ,{{{ 190, 190, 190, 190, 190}
03349     ,{ 190, 190, 190, 190, 190}
03350     ,{ 170, 160, 160, 160, 170}
03351     ,{ 130, 60, 120, 60, 130}
03352     ,{ 170, 160, 160, 160, 170}
03353     }
03354     ,{{{ 180, 170, 170, 170, 180}
03355     ,{ 180, 170, 170, 170, 180}
03356     ,{ 170, 170, 170, 170, 170}
03357     ,{ 180, 170, 170, 170, 180}
03358     ,{ 170, 170, 170, 170, 170}
03359     }
03360     ,{{{ 200, 160, 190, 160, 200}
03361     ,{ 200, 130, 190, 130, 200}
03362     ,{ 170, 160, 160, 160, 170}
03363     ,{ 20, 10, 10, 10, 20}
03364     ,{ 170, 160, 160, 160, 170}
```

```
03365     }
03366     ,{{ 180, 170, 170, 170, 180}
03367     ,{ 180, 170, 170, 170, 180}
03368     ,{ 170, 170, 170, 170, 170}
03369     ,{ 180, 170, 170, 170, 180}
03370     ,{ 80, 80, 80, 80, 80}
03371     }
03372     }
03373     ,{{{ 180, 50, 180, 130, 180}
03374     ,{ 180, 10, 180, 120, 180}
03375     ,{ 160, 50, 160, 30, 160}
03376     ,{ 170, 0, 170, 130, 170}
03377     ,{ 160, 50, 160, 70, 160}
03378     }
03379     ,{{{ 180, 10, 180, 50, 180}
03380     ,{ 180, 10, 180, 50, 180}
03381     ,{ 160, -10, 160, 20, 160}
03382     ,{ 60, -110, 60, 50, 60}
03383     ,{ 160, -10, 160, 20, 160}
03384     }
03385     ,{{{ 170, 50, 170, 30, 170}
03386     ,{ 170, 0, 170, 30, 170}
03387     ,{ 160, 50, 160, 30, 160}
03388     ,{ 170, 0, 170, 30, 170}
03389     ,{ 160, 50, 160, 30, 160}
03390     }
03391     ,{{{ 160, -10, 160, 130, 160}
03392     ,{ 130, -40, 130, 120, 130}
03393     ,{ 160, -10, 160, 20, 160}
03394     ,{ 130, -30, 10, 130, 10}
03395     ,{ 160, -10, 160, 20, 160}
03396     }
03397     ,{{{ 170, 50, 170, 70, 170}
03398     ,{ 170, 0, 170, 30, 170}
03399     ,{ 160, 50, 160, 30, 160}
03400     ,{ 170, 0, 170, 30, 170}
03401     ,{ 70, -100, 70, 70, 70}
03402     }
03403     }
03404     ,{{{ 190, 190, 190, 190, 170}
03405     ,{ 190, 190, 190, 190, 170}
03406     ,{ 170, 170, 170, 170, 90}
03407     ,{ 170, 170, 170, 170, 100}
03408     ,{ 170, 170, 170, 170, 90}
03409     }
03410     ,{{{ 190, 190, 190, 190, 170}
03411     ,{ 190, 190, 190, 190, 170}
03412     ,{ 160, 160, 160, 160, 90}
03413     ,{ 120, 60, 120, 60, -10}
03414     ,{ 160, 160, 160, 160, 90}
03415     }
03416     ,{{{ 170, 170, 170, 170, 100}
03417     ,{ 170, 170, 170, 170, 100}
03418     ,{ 170, 170, 170, 170, 90}
03419     ,{ 170, 170, 170, 170, 100}
03420     ,{ 170, 170, 170, 170, 90}
03421     }
03422     ,{{{ 190, 160, 190, 160, 90}
03423     ,{ 190, 130, 190, 130, 60}
03424     ,{ 160, 160, 160, 160, 90}
03425     ,{ 70, 10, 10, 10, 70}
03426     ,{ 160, 160, 160, 160, 90}
03427     }
03428     ,{{{ 170, 170, 170, 170, 100}
03429     ,{ 170, 170, 170, 170, 100}
03430     ,{ 170, 170, 170, 170, 90}
03431     ,{ 170, 170, 170, 170, 100}
03432     ,{ 80, 80, 80, 80, 0}
03433     }
03434     }
03435     }
03436     ,{{{ 210, 210, 190, 190, 190}
03437     ,{ 210, 210, 190, 190, 190}
03438     ,{ 210, 210, 190, 190, 190}
03439     ,{ 210, 210, 190, 190, 190}
03440     ,{ 210, 210, 190, 190, 190}
03441     }
03442     ,{{{ 210, 210, 190, 190, 190}
03443     ,{ 210, 210, 190, 190, 190}
03444     ,{ 170, 170, 140, 140, 150}
03445     ,{ 150, 110, 140, 80, 150}
03446     ,{ 170, 170, 140, 140, 150}
03447     }
03448     ,{{{ 210, 210, 190, 190, 190}
03449     ,{ 210, 210, 190, 190, 190}
03450     ,{ 210, 210, 190, 190, 190}
03451     ,{ 210, 210, 190, 190, 190}
```



```

03452     , { 210, 210, 190, 190, 190}
03453     }
03454     , { { 170, 170, 150, 150, 160}
03455     , { 160, 120, 150, 90, 160}
03456     , { 170, 170, 140, 140, 150}
03457     , { 150, 60, 30, 150, 90}
03458     , { 170, 170, 140, 140, 150}
03459     }
03460     , { { 210, 210, 190, 190, 190}
03461     , { 210, 210, 190, 190, 190}
03462     , { 180, 180, 160, 160, 160}
03463     , { 210, 210, 190, 190, 190}
03464     , { 190, 190, 100, 100, 110}
03465     }
03466     }
03467     , { { { 210, 210, 180, 130, 180}
03468     , { 210, 210, 180, 70, 180}
03469     , { 210, 210, 180, 130, 180}
03470     , { 210, 210, 180, 70, 180}
03471     , { 210, 210, 180, 130, 180}
03472     }
03473     , { { 210, 210, 180, 70, 180}
03474     , { 210, 210, 180, 70, 180}
03475     , { 170, 170, 140, 30, 140}
03476     , { 110, 110, 80, -30, 80}
03477     , { 170, 170, 140, 30, 140}
03478     }
03479     , { { 210, 210, 180, 130, 180}
03480     , { 210, 210, 180, 70, 180}
03481     , { 210, 210, 180, 130, 180}
03482     , { 210, 210, 180, 70, 180}
03483     , { 210, 210, 180, 130, 180}
03484     }
03485     , { { 170, 170, 140, 50, 140}
03486     , { 120, 120, 90, -20, 90}
03487     , { 170, 170, 140, 30, 140}
03488     , { 60, 60, 30, 50, 30}
03489     , { 170, 170, 140, 30, 140}
03490     }
03491     , { { 210, 210, 180, 100, 180}
03492     , { 210, 210, 180, 70, 180}
03493     , { 180, 180, 150, 100, 150}
03494     , { 210, 210, 180, 70, 180}
03495     , { 190, 190, 100, -10, 100}
03496     }
03497     }
03498     , { { { 190, 190, 190, 190, 190}
03499     , { 190, 190, 190, 190, 190}
03500     , { 190, 190, 190, 190, 190}
03501     , { 190, 190, 190, 190, 190}
03502     , { 190, 190, 190, 190, 190}
03503     }
03504     , { { 190, 190, 190, 190, 190}
03505     , { 190, 190, 190, 190, 190}
03506     , { 150, 140, 140, 140, 150}
03507     , { 150, 80, 140, 80, 150}
03508     , { 150, 140, 140, 140, 150}
03509     }
03510     , { { 190, 190, 190, 190, 190}
03511     , { 190, 190, 190, 190, 190}
03512     , { 190, 190, 190, 190, 190}
03513     , { 190, 190, 190, 190, 190}
03514     , { 190, 190, 190, 190, 190}
03515     }
03516     , { { 160, 140, 150, 140, 160}
03517     , { 160, 90, 150, 90, 160}
03518     , { 150, 140, 140, 140, 150}
03519     , { 40, 30, 30, 30, 40}
03520     , { 150, 140, 140, 140, 150}
03521     }
03522     , { { 190, 190, 190, 190, 190}
03523     , { 190, 190, 190, 190, 190}
03524     , { 160, 160, 160, 160, 160}
03525     , { 190, 190, 190, 190, 190}
03526     , { 110, 100, 100, 100, 110}
03527     }
03528     }
03529     , { { { 180, 70, 180, 150, 180}
03530     , { 180, 10, 180, 80, 180}
03531     , { 180, 70, 180, 50, 180}
03532     , { 180, 10, 180, 150, 180}
03533     , { 180, 70, 180, 90, 180}
03534     }
03535     , { { 180, 10, 180, 70, 180}
03536     , { 180, 10, 180, 50, 180}
03537     , { 140, -30, 140, 0, 140}
03538     , { 80, -90, 80, 70, 80}

```

```
03539      , { 140, -30, 140, 0, 140}
03540      }
03541      , { { 180, 70, 180, 50, 180}
03542      , { 180, 10, 180, 50, 180}
03543      , { 180, 70, 180, 50, 180}
03544      , { 180, 10, 180, 50, 180}
03545      , { 180, 70, 180, 50, 180}
03546      }
03547      , { { 150, -10, 140, 150, 140}
03548      , { 90, -80, 90, 80, 90}
03549      , { 140, -30, 140, 0, 140}
03550      , { 150, -10, 30, 150, 30}
03551      , { 140, -30, 140, 0, 140}
03552      }
03553      , { { 180, 40, 180, 90, 180}
03554      , { 180, 10, 180, 50, 180}
03555      , { 150, 40, 150, 20, 150}
03556      , { 180, 10, 180, 50, 180}
03557      , { 100, -70, 100, 90, 100}
03558      }
03559      }
03560      , { { { 190, 190, 190, 190, 170}
03561      , { 190, 190, 190, 190, 170}
03562      , { 190, 190, 190, 190, 110}
03563      , { 190, 190, 190, 190, 110}
03564      , { 190, 190, 190, 190, 110}
03565      }
03566      , { { 190, 190, 190, 190, 170}
03567      , { 190, 190, 190, 190, 170}
03568      , { 140, 140, 140, 140, 70}
03569      , { 140, 80, 140, 80, 10}
03570      , { 140, 140, 140, 140, 70}
03571      }
03572      , { { 190, 190, 190, 190, 110}
03573      , { 190, 190, 190, 190, 110}
03574      , { 190, 190, 190, 190, 110}
03575      , { 190, 190, 190, 190, 110}
03576      , { 190, 190, 190, 190, 110}
03577      }
03578      , { { 150, 140, 150, 140, 90}
03579      , { 150, 90, 150, 90, 20}
03580      , { 140, 140, 140, 140, 70}
03581      , { 90, 30, 30, 30, 90}
03582      , { 140, 140, 140, 140, 70}
03583      }
03584      , { { 190, 190, 190, 190, 110}
03585      , { 190, 190, 190, 190, 110}
03586      , { 160, 160, 160, 160, 80}
03587      , { 190, 190, 190, 190, 110}
03588      , { 100, 100, 100, 100, 30}
03589      }
03590      }
03591      }
03592      , { { { { 300, 300, 280, 250, 280}
03593      , { 280, 280, 280, 250, 280}
03594      , { 240, 240, 220, 220, 220}
03595      , { 240, 240, 220, 220, 220}
03596      , { 300, 300, 220, 220, 220}
03597      }
03598      , { { { 280, 280, 250, 250, 260}
03599      , { 280, 280, 250, 250, 260}
03600      , { 240, 240, 220, 220, 220}
03601      , { 200, 160, 200, 140, 200}
03602      , { 240, 240, 220, 220, 220}
03603      }
03604      , { { { 240, 240, 220, 220, 220}
03605      , { 240, 240, 220, 220, 220}
03606      , { 240, 240, 220, 220, 220}
03607      , { 240, 240, 220, 220, 220}
03608      , { 240, 240, 220, 220, 220}
03609      }
03610      , { { { 280, 240, 280, 220, 280}
03611      , { 280, 240, 280, 220, 280}
03612      , { 240, 240, 220, 220, 220}
03613      , { 210, 110, 90, 210, 140}
03614      , { 240, 240, 220, 220, 220}
03615      }
03616      , { { { 300, 300, 220, 220, 220}
03617      , { 240, 240, 220, 220, 220}
03618      , { 240, 240, 220, 220, 220}
03619      , { 240, 240, 220, 220, 220}
03620      , { 300, 300, 220, 220, 220}
03621      }
03622      }
03623      , { { { { 300, 300, 250, 160, 250}
03624      , { 280, 280, 250, 140, 250}
03625      , { 240, 240, 210, 160, 210}
```

```
03626     , { 240, 240, 210, 100, 210}
03627     , { 300, 300, 210, 160, 210}
03628     }
03629     , {{ 280, 280, 250, 140, 250}
03630     , { 280, 280, 250, 140, 250}
03631     , { 240, 240, 210, 100, 210}
03632     , { 160, 160, 130, 20, 130}
03633     , { 240, 240, 210, 100, 210}
03634     }
03635     , {{ 240, 240, 210, 160, 210}
03636     , { 240, 240, 210, 100, 210}
03637     , { 240, 240, 210, 160, 210}
03638     , { 240, 240, 210, 100, 210}
03639     , { 240, 240, 210, 160, 210}
03640     }
03641     , {{ 240, 240, 210, 100, 210}
03642     , { 240, 240, 210, 100, 210}
03643     , { 240, 240, 210, 100, 210}
03644     , { 110, 110, 80, 100, 80}
03645     , { 240, 240, 210, 100, 210}
03646     }
03647     , {{ 300, 300, 210, 160, 210}
03648     , { 240, 240, 210, 100, 210}
03649     , { 240, 240, 210, 160, 210}
03650     , { 240, 240, 210, 100, 210}
03651     , { 300, 300, 210, 140, 210}
03652     }
03653     }
03654     , {{{ 280, 250, 280, 250, 280}
03655     , { 280, 250, 280, 250, 280}
03656     , { 220, 220, 220, 220, 220}
03657     , { 220, 220, 220, 220, 220}
03658     , { 220, 220, 220, 220, 220}
03659     }
03660     , {{ 260, 250, 250, 250, 260}
03661     , { 260, 250, 250, 250, 260}
03662     , { 220, 220, 220, 220, 220}
03663     , { 200, 140, 200, 140, 200}
03664     , { 220, 220, 220, 220, 220}
03665     }
03666     , {{ 220, 220, 220, 220, 220}
03667     , { 220, 220, 220, 220, 220}
03668     , { 220, 220, 220, 220, 220}
03669     , { 220, 220, 220, 220, 220}
03670     , { 220, 220, 220, 220, 220}
03671     }
03672     , {{{ 280, 220, 280, 220, 280}
03673     , { 280, 220, 280, 220, 280}
03674     , { 220, 220, 220, 220, 220}
03675     , { 90, 90, 90, 90, 90}
03676     , { 220, 220, 220, 220, 220}
03677     }
03678     , {{{ 220, 220, 220, 220, 220}
03679     , { 220, 220, 220, 220, 220}
03680     , { 220, 220, 220, 220, 220}
03681     , { 220, 220, 220, 220, 220}
03682     , { 220, 220, 220, 220, 220}
03683     }
03684     }
03685     , {{{ 250, 100, 250, 210, 250}
03686     , { 250, 70, 250, 210, 250}
03687     , { 210, 100, 210, 80, 210}
03688     , { 210, 40, 210, 210, 210}
03689     , { 210, 100, 210, 210, 210}
03690     }
03691     , {{ 250, 70, 250, 130, 250}
03692     , { 250, 70, 250, 110, 250}
03693     , { 210, 40, 210, 80, 210}
03694     , { 130, -40, 130, 130, 130}
03695     , { 210, 40, 210, 80, 210}
03696     }
03697     , {{{ 210, 100, 210, 80, 210}
03698     , { 210, 40, 210, 80, 210}
03699     , { 210, 100, 210, 80, 210}
03700     , { 210, 40, 210, 80, 210}
03701     , { 210, 100, 210, 80, 210}
03702     }
03703     , {{{ 210, 40, 210, 210, 210}
03704     , { 210, 40, 210, 210, 210}
03705     , { 210, 40, 210, 80, 210}
03706     , { 210, 40, 80, 210, 80}
03707     , { 210, 40, 210, 80, 210}
03708     }
03709     , {{{ 210, 100, 210, 210, 210}
03710     , { 210, 40, 210, 80, 210}
03711     , { 210, 100, 210, 80, 210}
03712     , { 210, 40, 210, 80, 210}
```

```
03713      , { 210, 50, 210, 210, 210}
03714      }
03715      }
03716      , {{{ 280, 250, 280, 250, 240}
03717      , { 280, 250, 280, 250, 240}
03718      , { 220, 220, 220, 220, 140}
03719      , { 220, 220, 220, 220, 140}
03720      , { 220, 220, 220, 220, 140}
03721      }
03722      , {{{ 250, 250, 250, 250, 240}
03723      , { 250, 250, 250, 250, 240}
03724      , { 220, 220, 220, 220, 140}
03725      , { 200, 140, 200, 140, 90}
03726      , { 220, 220, 220, 220, 140}
03727      }
03728      , {{{ 220, 220, 220, 220, 140}
03729      , { 220, 220, 220, 220, 140}
03730      , { 220, 220, 220, 220, 140}
03731      , { 220, 220, 220, 220, 140}
03732      , { 220, 220, 220, 220, 140}
03733      }
03734      , {{{ 280, 220, 280, 220, 140}
03735      , { 280, 220, 280, 220, 140}
03736      , { 220, 220, 220, 220, 140}
03737      , { 140, 90, 90, 90, 140}
03738      , { 220, 220, 220, 220, 140}
03739      }
03740      , {{{ 220, 220, 220, 220, 140}
03741      , { 220, 220, 220, 220, 140}
03742      , { 220, 220, 220, 220, 140}
03743      , { 220, 220, 220, 220, 140}
03744      , { 220, 220, 220, 220, 140}
03745      }
03746      }
03747      }
03748      }
03749      , {{{ { INF, INF, INF, INF, INF}
03750      , { INF, INF, INF, INF, INF}
03751      , { INF, INF, INF, INF, INF}
03752      , { INF, INF, INF, INF, INF}
03753      , { INF, INF, INF, INF, INF}
03754      }
03755      , {{{ INF, INF, INF, INF, INF}
03756      , { INF, INF, INF, INF, INF}
03757      , { INF, INF, INF, INF, INF}
03758      , { INF, INF, INF, INF, INF}
03759      , { INF, INF, INF, INF, INF}
03760      }
03761      , {{{ INF, INF, INF, INF, INF}
03762      , { INF, INF, INF, INF, INF}
03763      , { INF, INF, INF, INF, INF}
03764      , { INF, INF, INF, INF, INF}
03765      , { INF, INF, INF, INF, INF}
03766      }
03767      , {{{ INF, INF, INF, INF, INF}
03768      , { INF, INF, INF, INF, INF}
03769      , { INF, INF, INF, INF, INF}
03770      , { INF, INF, INF, INF, INF}
03771      , { INF, INF, INF, INF, INF}
03772      }
03773      , {{{ INF, INF, INF, INF, INF}
03774      , { INF, INF, INF, INF, INF}
03775      , { INF, INF, INF, INF, INF}
03776      , { INF, INF, INF, INF, INF}
03777      , { INF, INF, INF, INF, INF}
03778      }
03779      }
03780      , {{{ INF, INF, INF, INF, INF}
03781      , { INF, INF, INF, INF, INF}
03782      , { INF, INF, INF, INF, INF}
03783      , { INF, INF, INF, INF, INF}
03784      , { INF, INF, INF, INF, INF}
03785      }
03786      , {{{ INF, INF, INF, INF, INF}
03787      , { INF, INF, INF, INF, INF}
03788      , { INF, INF, INF, INF, INF}
03789      , { INF, INF, INF, INF, INF}
03790      , { INF, INF, INF, INF, INF}
03791      }
03792      , {{{ INF, INF, INF, INF, INF}
03793      , { INF, INF, INF, INF, INF}
03794      , { INF, INF, INF, INF, INF}
03795      , { INF, INF, INF, INF, INF}
03796      , { INF, INF, INF, INF, INF}
03797      }
03798      , {{{ INF, INF, INF, INF, INF}
03799      , { INF, INF, INF, INF, INF}
```

```
03800     , {   INF,   INF,   INF,   INF,   INF }
03801     , {   INF,   INF,   INF,   INF,   INF }
03802     , {   INF,   INF,   INF,   INF,   INF }
03803     }
03804     , { {   INF,   INF,   INF,   INF,   INF }
03805     , {   INF,   INF,   INF,   INF,   INF }
03806     , {   INF,   INF,   INF,   INF,   INF }
03807     , {   INF,   INF,   INF,   INF,   INF }
03808     , {   INF,   INF,   INF,   INF,   INF }
03809     }
03810     }
03811     , { { {   INF,   INF,   INF,   INF,   INF }
03812     , {   INF,   INF,   INF,   INF,   INF }
03813     , {   INF,   INF,   INF,   INF,   INF }
03814     , {   INF,   INF,   INF,   INF,   INF }
03815     , {   INF,   INF,   INF,   INF,   INF }
03816     }
03817     , { { {   INF,   INF,   INF,   INF,   INF }
03818     , {   INF,   INF,   INF,   INF,   INF }
03819     , {   INF,   INF,   INF,   INF,   INF }
03820     , {   INF,   INF,   INF,   INF,   INF }
03821     , {   INF,   INF,   INF,   INF,   INF }
03822     }
03823     , { { {   INF,   INF,   INF,   INF,   INF }
03824     , {   INF,   INF,   INF,   INF,   INF }
03825     , {   INF,   INF,   INF,   INF,   INF }
03826     , {   INF,   INF,   INF,   INF,   INF }
03827     , {   INF,   INF,   INF,   INF,   INF }
03828     }
03829     , { { {   INF,   INF,   INF,   INF,   INF }
03830     , {   INF,   INF,   INF,   INF,   INF }
03831     , {   INF,   INF,   INF,   INF,   INF }
03832     , {   INF,   INF,   INF,   INF,   INF }
03833     , {   INF,   INF,   INF,   INF,   INF }
03834     }
03835     , { { {   INF,   INF,   INF,   INF,   INF }
03836     , {   INF,   INF,   INF,   INF,   INF }
03837     , {   INF,   INF,   INF,   INF,   INF }
03838     , {   INF,   INF,   INF,   INF,   INF }
03839     , {   INF,   INF,   INF,   INF,   INF }
03840     }
03841     }
03842     , { { { {   INF,   INF,   INF,   INF,   INF }
03843     , {   INF,   INF,   INF,   INF,   INF }
03844     , {   INF,   INF,   INF,   INF,   INF }
03845     , {   INF,   INF,   INF,   INF,   INF }
03846     , {   INF,   INF,   INF,   INF,   INF }
03847     }
03848     , { { {   INF,   INF,   INF,   INF,   INF }
03849     , {   INF,   INF,   INF,   INF,   INF }
03850     , {   INF,   INF,   INF,   INF,   INF }
03851     , {   INF,   INF,   INF,   INF,   INF }
03852     , {   INF,   INF,   INF,   INF,   INF }
03853     }
03854     , { { {   INF,   INF,   INF,   INF,   INF }
03855     , {   INF,   INF,   INF,   INF,   INF }
03856     , {   INF,   INF,   INF,   INF,   INF }
03857     , {   INF,   INF,   INF,   INF,   INF }
03858     , {   INF,   INF,   INF,   INF,   INF }
03859     }
03860     , { { {   INF,   INF,   INF,   INF,   INF }
03861     , {   INF,   INF,   INF,   INF,   INF }
03862     , {   INF,   INF,   INF,   INF,   INF }
03863     , {   INF,   INF,   INF,   INF,   INF }
03864     , {   INF,   INF,   INF,   INF,   INF }
03865     }
03866     , { { {   INF,   INF,   INF,   INF,   INF }
03867     , {   INF,   INF,   INF,   INF,   INF }
03868     , {   INF,   INF,   INF,   INF,   INF }
03869     , {   INF,   INF,   INF,   INF,   INF }
03870     , {   INF,   INF,   INF,   INF,   INF }
03871     }
03872     }
03873     , { { { {   INF,   INF,   INF,   INF,   INF }
03874     , {   INF,   INF,   INF,   INF,   INF }
03875     , {   INF,   INF,   INF,   INF,   INF }
03876     , {   INF,   INF,   INF,   INF,   INF }
03877     , {   INF,   INF,   INF,   INF,   INF }
03878     }
03879     , { { {   INF,   INF,   INF,   INF,   INF }
03880     , {   INF,   INF,   INF,   INF,   INF }
03881     , {   INF,   INF,   INF,   INF,   INF }
03882     , {   INF,   INF,   INF,   INF,   INF }
03883     , {   INF,   INF,   INF,   INF,   INF }
03884     }
03885     , { { {   INF,   INF,   INF,   INF,   INF }
03886     , {   INF,   INF,   INF,   INF,   INF }
```

```
03887      , {   INF,   INF,   INF,   INF,   INF }
03888      , {   INF,   INF,   INF,   INF,   INF }
03889      , {   INF,   INF,   INF,   INF,   INF }
03890      }
03891      , { {   INF,   INF,   INF,   INF,   INF }
03892      , {   INF,   INF,   INF,   INF,   INF }
03893      , {   INF,   INF,   INF,   INF,   INF }
03894      , {   INF,   INF,   INF,   INF,   INF }
03895      , {   INF,   INF,   INF,   INF,   INF }
03896      }
03897      , { {   INF,   INF,   INF,   INF,   INF }
03898      , {   INF,   INF,   INF,   INF,   INF }
03899      , {   INF,   INF,   INF,   INF,   INF }
03900      , {   INF,   INF,   INF,   INF,   INF }
03901      , {   INF,   INF,   INF,   INF,   INF }
03902      }
03903      }
03904      }
03905      , { { { 300,   300,   270,   270,   290 }
03906      , { 300,   300,   270,   270,   290 }
03907      , { 290,   290,   250,   270,   250 }
03908      , { 300,   300,   270,   270,   270 }
03909      , { 270,   270,   240,   260,   240 }
03910      }
03911      , { { 290,   270,   230,   230,   290 }
03912      , { 290,   270,   230,   230,   290 }
03913      , { 260,   260,   220,   220,   220 }
03914      , { 190,   170,   190,   130,   190 }
03915      , { 260,   260,   220,   220,   220 }
03916      }
03917      , { { 300,   300,   270,   270,   270 }
03918      , { 300,   300,   270,   270,   270 }
03919      , { 290,   290,   250,   270,   250 }
03920      , { 300,   300,   270,   270,   270 }
03921      , { 270,   270,   240,   260,   240 }
03922      }
03923      , { { 260,   260,   220,   220,   220 }
03924      , { 190,   170,   190,   130,   190 }
03925      , { 260,   260,   220,   220,   220 }
03926      , { 210,   130,    80,   210,   210 }
03927      , { 260,   260,   220,   220,   220 }
03928      }
03929      , { { 300,   300,   270,   270,   270 }
03930      , { 300,   300,   270,   270,   270 }
03931      , { 270,   270,   240,   260,   240 }
03932      , { 300,   300,   270,   270,   270 }
03933      , { 240,   240,   150,   150,   150 }
03934      }
03935      }
03936      , { { { 300,   300,   270,   270,   270 }
03937      , { 300,   300,   270,   230,   270 }
03938      , { 290,   290,   250,   270,   250 }
03939      , { 300,   300,   270,   230,   270 }
03940      , { 270,   270,   240,   260,   240 }
03941      }
03942      , { { 270,   270,   230,   190,   230 }
03943      , { 270,   270,   230,   190,   230 }
03944      , { 260,   260,   220,   180,   220 }
03945      , { 170,   170,   130,    90,   130 }
03946      , { 260,   260,   220,   180,   220 }
03947      }
03948      , { { 300,   300,   270,   270,   270 }
03949      , { 300,   300,   270,   230,   270 }
03950      , { 290,   290,   250,   270,   250 }
03951      , { 300,   300,   270,   230,   270 }
03952      , { 270,   270,   240,   260,   240 }
03953      }
03954      , { { 260,   260,   220,   180,   220 }
03955      , { 170,   170,   130,    90,   130 }
03956      , { 260,   260,   220,   180,   220 }
03957      , { 170,   110,    80,   170,    80 }
03958      , { 260,   260,   220,   180,   220 }
03959      }
03960      , { { 300,   300,   270,   260,   270 }
03961      , { 300,   300,   270,   230,   270 }
03962      , { 270,   270,   240,   260,   240 }
03963      , { 300,   300,   270,   230,   270 }
03964      , { 240,   240,   150,   110,   150 }
03965      }
03966      }
03967      , { { { 270,   270,   270,   270,   270 }
03968      , { 270,   270,   270,   270,   270 }
03969      , { 250,   250,   250,   250,   250 }
03970      , { 270,   270,   270,   270,   270 }
03971      , { 240,   240,   240,   240,   240 }
03972      }
03973      , { { 230,   230,   230,   230,   230 }
```

```

03974     , { 230, 230, 230, 230, 230 }
03975     , { 220, 220, 220, 220, 220 }
03976     , { 190, 130, 190, 130, 190 }
03977     , { 220, 220, 220, 220, 220 }
03978     }
03979     , { { 270, 270, 270, 270, 270 }
03980     , { 270, 270, 270, 270, 270 }
03981     , { 250, 250, 250, 250, 250 }
03982     , { 270, 270, 270, 270, 270 }
03983     , { 240, 240, 240, 240, 240 }
03984     }
03985     , { { 220, 220, 220, 220, 220 }
03986     , { 190, 130, 190, 130, 190 }
03987     , { 220, 220, 220, 220, 220 }
03988     , { 80, 80, 80, 80, 80 }
03989     , { 220, 220, 220, 220, 220 }
03990     }
03991     , { { 270, 270, 270, 270, 270 }
03992     , { 270, 270, 270, 270, 270 }
03993     , { 240, 240, 240, 240, 240 }
03994     , { 270, 270, 270, 270, 270 }
03995     , { 150, 150, 150, 150, 150 }
03996     }
03997     }
03998     , { { { 270, 230, 270, 210, 270 }
03999     , { 270, 190, 270, 140, 270 }
04000     , { 250, 230, 250, 120, 250 }
04001     , { 270, 190, 270, 210, 270 }
04002     , { 240, 220, 240, 150, 240 }
04003     }
04004     , { { 230, 150, 230, 130, 230 }
04005     , { 230, 150, 230, 100, 230 }
04006     , { 220, 140, 220, 90, 220 }
04007     , { 130, 50, 130, 130, 130 }
04008     , { 220, 140, 220, 90, 220 }
04009     }
04010     , { { 270, 230, 270, 140, 270 }
04011     , { 270, 190, 270, 140, 270 }
04012     , { 250, 230, 250, 120, 250 }
04013     , { 270, 190, 270, 140, 270 }
04014     , { 240, 220, 240, 110, 240 }
04015     }
04016     , { { 220, 140, 220, 210, 220 }
04017     , { 130, 50, 130, 130, 130 }
04018     , { 220, 140, 220, 90, 220 }
04019     , { 210, 130, 80, 210, 80 }
04020     , { 220, 140, 220, 90, 220 }
04021     }
04022     , { { 270, 220, 270, 150, 270 }
04023     , { 270, 190, 270, 140, 270 }
04024     , { 240, 220, 240, 110, 240 }
04025     , { 270, 190, 270, 140, 270 }
04026     , { 150, 70, 150, 150, 150 }
04027     }
04028     }
04029     , { { { 290, 270, 270, 270, 290 }
04030     , { 290, 270, 270, 270, 290 }
04031     , { 250, 250, 250, 250, 250 }
04032     , { 270, 270, 270, 270, 270 }
04033     , { 240, 240, 240, 240, 240 }
04034     }
04035     , { { 290, 230, 230, 230, 290 }
04036     , { 290, 230, 230, 230, 290 }
04037     , { 220, 220, 220, 220, 220 }
04038     , { 190, 130, 190, 130, 130 }
04039     , { 220, 220, 220, 220, 220 }
04040     }
04041     , { { 270, 270, 270, 270, 270 }
04042     , { 270, 270, 270, 270, 270 }
04043     , { 250, 250, 250, 250, 250 }
04044     , { 270, 270, 270, 270, 270 }
04045     , { 240, 240, 240, 240, 240 }
04046     }
04047     , { { 220, 220, 220, 220, 220 }
04048     , { 190, 130, 190, 130, 130 }
04049     , { 220, 220, 220, 220, 220 }
04050     , { 210, 80, 80, 80, 210 }
04051     , { 220, 220, 220, 220, 220 }
04052     }
04053     , { { 270, 270, 270, 270, 270 }
04054     , { 270, 270, 270, 270, 270 }
04055     , { 240, 240, 240, 240, 240 }
04056     , { 270, 270, 270, 270, 270 }
04057     , { 150, 150, 150, 150, 150 }
04058     }
04059     }
04060     }

```

```
04061 ,{{{ 300, 280, 240, 240, 300}
04062 ,{ 300, 280, 240, 240, 300}
04063 ,{ 260, 260, 220, 240, 220}
04064 ,{ 250, 250, 210, 210, 210}
04065 ,{ 250, 250, 220, 240, 220}
04066 }
04067 ,{{{ 300, 280, 240, 240, 300}
04068 ,{ 300, 280, 240, 240, 300}
04069 ,{ 250, 250, 220, 220, 220}
04070 ,{ 100, 70, 100, 40, 100}
04071 ,{ 250, 250, 220, 220, 220}
04072 }
04073 ,{{{ 250, 250, 220, 240, 220}
04074 ,{ 250, 250, 210, 210, 210}
04075 ,{ 250, 250, 220, 240, 220}
04076 ,{ 250, 250, 210, 210, 210}
04077 ,{ 250, 250, 220, 240, 220}
04078 }
04079 ,{{{ 250, 250, 220, 220, 220}
04080 ,{ 160, 140, 160, 100, 160}
04081 ,{ 250, 250, 220, 220, 220}
04082 ,{ 210, 130, 80, 210, 210}
04083 ,{ 250, 250, 220, 220, 220}
04084 }
04085 ,{{{ 260, 260, 220, 240, 220}
04086 ,{ 250, 250, 210, 210, 210}
04087 ,{ 260, 260, 220, 240, 220}
04088 ,{ 250, 250, 210, 210, 210}
04089 ,{ 240, 240, 140, 140, 140}
04090 }
04091 }
04092 ,{{{ 280, 280, 240, 240, 240}
04093 ,{ 280, 280, 240, 200, 240}
04094 ,{ 260, 260, 220, 240, 220}
04095 ,{ 250, 250, 210, 170, 210}
04096 ,{ 250, 250, 220, 240, 220}
04097 }
04098 ,{{{ 280, 280, 240, 200, 240}
04099 ,{ 280, 280, 240, 200, 240}
04100 ,{ 250, 250, 220, 180, 220}
04101 ,{ 70, 70, 40, 0, 40}
04102 ,{ 250, 250, 220, 180, 220}
04103 }
04104 ,{{{ 250, 250, 220, 240, 220}
04105 ,{ 250, 250, 210, 170, 210}
04106 ,{ 250, 250, 220, 240, 220}
04107 ,{ 250, 250, 210, 170, 210}
04108 ,{ 250, 250, 220, 240, 220}
04109 }
04110 ,{{{ 250, 250, 220, 180, 220}
04111 ,{ 140, 140, 100, 60, 100}
04112 ,{ 250, 250, 220, 180, 220}
04113 ,{ 170, 110, 80, 170, 80}
04114 ,{ 250, 250, 220, 180, 220}
04115 }
04116 ,{{{ 260, 260, 220, 240, 220}
04117 ,{ 250, 250, 210, 170, 210}
04118 ,{ 260, 260, 220, 240, 220}
04119 ,{ 250, 250, 210, 170, 210}
04120 ,{ 240, 240, 140, 100, 140}
04121 }
04122 }
04123 ,{{{ 240, 240, 240, 240, 240}
04124 ,{ 240, 240, 240, 240, 240}
04125 ,{ 220, 220, 220, 220, 220}
04126 ,{ 210, 210, 210, 210, 210}
04127 ,{ 220, 220, 220, 220, 220}
04128 }
04129 ,{{{ 240, 240, 240, 240, 240}
04130 ,{ 240, 240, 240, 240, 240}
04131 ,{ 220, 220, 220, 220, 220}
04132 ,{ 100, 40, 100, 40, 100}
04133 ,{ 220, 220, 220, 220, 220}
04134 }
04135 ,{{{ 220, 220, 220, 220, 220}
04136 ,{ 210, 210, 210, 210, 210}
04137 ,{ 220, 220, 220, 220, 220}
04138 ,{ 210, 210, 210, 210, 210}
04139 ,{ 220, 220, 220, 220, 220}
04140 }
04141 ,{{{ 220, 220, 220, 220, 220}
04142 ,{ 160, 100, 160, 100, 160}
04143 ,{ 220, 220, 220, 220, 220}
04144 ,{ 80, 80, 80, 80, 80}
04145 ,{ 220, 220, 220, 220, 220}
04146 }
04147 ,{{{ 220, 220, 220, 220, 220}
```



```
04148     , { 210, 210, 210, 210, 210}
04149     , { 220, 220, 220, 220, 220}
04150     , { 210, 210, 210, 210, 210}
04151     , { 140, 140, 140, 140, 140}
04152     }
04153     }
04154     , {{{ 240, 200, 240, 210, 240}
04155     , { 240, 160, 240, 110, 240}
04156     , { 220, 200, 220, 90, 220}
04157     , { 210, 130, 210, 210, 210}
04158     , { 220, 200, 220, 140, 220}
04159     }
04160     , {{{ 240, 160, 240, 110, 240}
04161     , { 240, 160, 240, 110, 240}
04162     , { 220, 140, 220, 90, 220}
04163     , { 40, -40, 40, 40, 40}
04164     , { 220, 140, 220, 90, 220}
04165     }
04166     , {{{ 220, 200, 220, 90, 220}
04167     , { 210, 130, 210, 80, 210}
04168     , { 220, 200, 220, 90, 220}
04169     , { 210, 130, 210, 80, 210}
04170     , { 220, 200, 220, 90, 220}
04171     }
04172     , {{{ 220, 140, 220, 210, 220}
04173     , { 100, 20, 100, 100, 100}
04174     , { 220, 140, 220, 90, 220}
04175     , { 210, 130, 80, 210, 80}
04176     , { 220, 140, 220, 90, 220}
04177     }
04178     , {{{ 220, 200, 220, 140, 220}
04179     , { 210, 130, 210, 80, 210}
04180     , { 220, 200, 220, 90, 220}
04181     , { 210, 130, 210, 80, 210}
04182     , { 140, 60, 140, 140, 140}
04183     }
04184     }
04185     , {{{ 300, 240, 240, 240, 300}
04186     , { 300, 240, 240, 240, 300}
04187     , { 220, 220, 220, 220, 220}
04188     , { 210, 210, 210, 210, 210}
04189     , { 220, 220, 220, 220, 220}
04190     }
04191     , {{{ 300, 240, 240, 240, 300}
04192     , { 300, 240, 240, 240, 300}
04193     , { 220, 220, 220, 220, 220}
04194     , { 100, 40, 100, 40, 40}
04195     , { 220, 220, 220, 220, 220}
04196     }
04197     , {{{ 220, 220, 220, 220, 220}
04198     , { 210, 210, 210, 210, 210}
04199     , { 220, 220, 220, 220, 220}
04200     , { 210, 210, 210, 210, 210}
04201     , { 220, 220, 220, 220, 220}
04202     }
04203     , {{{ 220, 220, 220, 220, 220}
04204     , { 160, 100, 160, 100, 100}
04205     , { 220, 220, 220, 220, 220}
04206     , { 210, 80, 80, 80, 210}
04207     , { 220, 220, 220, 220, 220}
04208     }
04209     , {{{ 220, 220, 220, 220, 220}
04210     , { 210, 210, 210, 210, 210}
04211     , { 220, 220, 220, 220, 220}
04212     , { 210, 210, 210, 210, 210}
04213     , { 140, 140, 140, 140, 140}
04214     }
04215     }
04216     }
04217     , {{{ 430, 430, 370, 370, 430}
04218     , { 430, 410, 370, 370, 430}
04219     , { 370, 370, 340, 360, 340}
04220     , { 370, 370, 340, 340, 340}
04221     , { 430, 430, 340, 360, 340}
04222     }
04223     , {{{ 430, 410, 370, 370, 430}
04224     , { 430, 410, 370, 370, 430}
04225     , { 370, 370, 340, 340, 340}
04226     , { 320, 290, 320, 260, 320}
04227     , { 370, 370, 340, 340, 340}
04228     }
04229     , {{{ 370, 370, 340, 360, 340}
04230     , { 370, 370, 340, 340, 340}
04231     , { 370, 370, 340, 360, 340}
04232     , { 370, 370, 340, 340, 340}
04233     , { 370, 370, 340, 360, 340}
04234     }
```

```
04235 ,{{ 370, 370, 360, 340, 360}
04236 ,{ 360, 330, 360, 300, 360}
04237 ,{ 370, 370, 340, 340, 340}
04238 ,{ 340, 260, 210, 340, 340}
04239 ,{ 370, 370, 340, 340, 340}
04240 }
04241 ,{{ 430, 430, 340, 360, 340}
04242 ,{ 370, 370, 340, 340, 340}
04243 ,{ 370, 370, 340, 360, 340}
04244 ,{ 370, 370, 340, 340, 340}
04245 ,{ 430, 430, 340, 340, 340}
04246 }
04247 }
04248 ,{{{ 430, 430, 370, 360, 370}
04249 ,{ 410, 410, 370, 330, 370}
04250 ,{ 370, 370, 340, 360, 340}
04251 ,{ 370, 370, 340, 300, 340}
04252 ,{ 430, 430, 340, 360, 340}
04253 }
04254 ,{{ 410, 410, 370, 330, 370}
04255 ,{ 410, 410, 370, 330, 370}
04256 ,{ 370, 370, 340, 300, 340}
04257 ,{ 290, 290, 260, 220, 260}
04258 ,{ 370, 370, 340, 300, 340}
04259 }
04260 ,{{ 370, 370, 340, 360, 340}
04261 ,{ 370, 370, 340, 300, 340}
04262 ,{ 370, 370, 340, 360, 340}
04263 ,{ 370, 370, 340, 300, 340}
04264 ,{ 370, 370, 340, 360, 340}
04265 }
04266 ,{{ 370, 370, 340, 300, 340}
04267 ,{ 330, 330, 300, 260, 300}
04268 ,{ 370, 370, 340, 300, 340}
04269 ,{ 300, 240, 210, 300, 210}
04270 ,{ 370, 370, 340, 300, 340}
04271 }
04272 ,{{{ 430, 430, 340, 360, 340}
04273 ,{ 370, 370, 340, 300, 340}
04274 ,{ 370, 370, 340, 360, 340}
04275 ,{ 370, 370, 340, 300, 340}
04276 ,{ 430, 430, 340, 300, 340}
04277 }
04278 }
04279 ,{{{ 370, 370, 370, 370, 370}
04280 ,{ 370, 370, 370, 370, 370}
04281 ,{ 340, 340, 340, 340, 340}
04282 ,{ 340, 340, 340, 340, 340}
04283 ,{ 340, 340, 340, 340, 340}
04284 }
04285 ,{{{ 370, 370, 370, 370, 370}
04286 ,{ 370, 370, 370, 370, 370}
04287 ,{ 340, 340, 340, 340, 340}
04288 ,{ 320, 260, 320, 260, 320}
04289 ,{ 340, 340, 340, 340, 340}
04290 }
04291 ,{{{ 340, 340, 340, 340, 340}
04292 ,{ 340, 340, 340, 340, 340}
04293 ,{ 340, 340, 340, 340, 340}
04294 ,{ 340, 340, 340, 340, 340}
04295 ,{ 340, 340, 340, 340, 340}
04296 }
04297 ,{{{ 360, 340, 360, 340, 360}
04298 ,{ 360, 300, 360, 300, 360}
04299 ,{ 340, 340, 340, 340, 340}
04300 ,{ 210, 210, 210, 210, 210}
04301 ,{ 340, 340, 340, 340, 340}
04302 }
04303 ,{{{ 340, 340, 340, 340, 340}
04304 ,{ 340, 340, 340, 340, 340}
04305 ,{ 340, 340, 340, 340, 340}
04306 ,{ 340, 340, 340, 340, 340}
04307 ,{ 340, 340, 340, 340, 340}
04308 }
04309 }
04310 ,{{{ 370, 320, 370, 340, 370}
04311 ,{ 370, 290, 370, 300, 370}
04312 ,{ 340, 320, 340, 210, 340}
04313 ,{ 340, 260, 340, 340, 340}
04314 ,{ 340, 320, 340, 340, 340}
04315 }
04316 ,{{ 370, 290, 370, 260, 370}
04317 ,{ 370, 290, 370, 240, 370}
04318 ,{ 340, 260, 340, 210, 340}
04319 ,{ 260, 180, 260, 260, 260}
04320 ,{ 340, 260, 340, 210, 340}
04321 }
```

```
04322     ,{{ 340, 320, 340, 210, 340}
04323     ,{ 340, 260, 340, 210, 340}
04324     ,{ 340, 320, 340, 210, 340}
04325     ,{ 340, 260, 340, 210, 340}
04326     ,{ 340, 320, 340, 210, 340}
04327     }
04328     ,{{ 340, 260, 340, 340, 340}
04329     ,{ 300, 220, 300, 300, 300}
04330     ,{ 340, 260, 340, 210, 340}
04331     ,{ 340, 260, 210, 340, 210}
04332     ,{ 340, 260, 340, 210, 340}
04333     }
04334     ,{{ 340, 320, 340, 340, 340}
04335     ,{ 340, 260, 340, 210, 340}
04336     ,{ 340, 320, 340, 210, 340}
04337     ,{ 340, 260, 340, 210, 340}
04338     ,{ 340, 260, 340, 340, 340}
04339     }
04340     }
04341     ,{{{ 430, 370, 370, 370, 430}
04342     ,{ 430, 370, 370, 370, 430}
04343     ,{ 340, 340, 340, 340, 340}
04344     ,{ 340, 340, 340, 340, 340}
04345     ,{ 340, 340, 340, 340, 340}
04346     }
04347     ,{{{ 430, 370, 370, 370, 430}
04348     ,{ 430, 370, 370, 370, 430}
04349     ,{ 340, 340, 340, 340, 340}
04350     ,{ 320, 260, 320, 260, 260}
04351     ,{ 340, 340, 340, 340, 340}
04352     }
04353     ,{{{ 340, 340, 340, 340, 340}
04354     ,{ 340, 340, 340, 340, 340}
04355     ,{ 340, 340, 340, 340, 340}
04356     ,{ 340, 340, 340, 340, 340}
04357     ,{ 340, 340, 340, 340, 340}
04358     }
04359     ,{{{ 360, 340, 360, 340, 340}
04360     ,{ 360, 300, 360, 300, 300}
04361     ,{ 340, 340, 340, 340, 340}
04362     ,{ 340, 210, 210, 210, 340}
04363     ,{ 340, 340, 340, 340, 340}
04364     }
04365     ,{{{ 340, 340, 340, 340, 340}
04366     ,{ 340, 340, 340, 340, 340}
04367     ,{ 340, 340, 340, 340, 340}
04368     ,{ 340, 340, 340, 340, 340}
04369     ,{ 340, 340, 340, 340, 340}
04370     }
04371     }
04372     }
04373     ,{{{ 400, 400, 400, 360, 400}
04374     ,{ 400, 370, 400, 360, 400}
04375     ,{ 340, 340, 310, 330, 310}
04376     ,{ 340, 340, 310, 310, 310}
04377     ,{ 400, 400, 310, 330, 310}
04378     }
04379     ,{{ 360, 360, 310, 360, 330}
04380     ,{ 360, 360, 270, 360, 330}
04381     ,{ 340, 340, 310, 310, 310}
04382     ,{ 230, 220, 230, 170, 230}
04383     ,{ 340, 340, 310, 310, 310}
04384     }
04385     ,{{ 340, 340, 310, 330, 310}
04386     ,{ 340, 340, 310, 310, 310}
04387     ,{ 340, 340, 310, 330, 310}
04388     ,{ 340, 340, 310, 310, 310}
04389     ,{ 340, 340, 310, 330, 310}
04390     }
04391     ,{{ 400, 370, 400, 340, 400}
04392     ,{ 400, 370, 400, 340, 400}
04393     ,{ 340, 340, 310, 310, 310}
04394     ,{ 310, 230, 180, 310, 310}
04395     ,{ 340, 340, 310, 310, 310}
04396     }
04397     ,{{ 400, 400, 310, 330, 310}
04398     ,{ 340, 340, 310, 310, 310}
04399     ,{ 340, 340, 310, 330, 310}
04400     ,{ 340, 340, 310, 310, 310}
04401     ,{ 400, 400, 310, 310, 310}
04402     }
04403     }
04404     ,{{{ 400, 400, 340, 360, 340}
04405     ,{ 370, 370, 340, 360, 340}
04406     ,{ 340, 340, 310, 330, 310}
04407     ,{ 340, 340, 310, 270, 310}
04408     ,{ 400, 400, 310, 330, 310}
```

```
04409      }
04410      ,{{ 360, 360, 310, 360, 310}
04411      ,{ 360, 360, 270, 360, 270}
04412      ,{ 340, 340, 310, 270, 310}
04413      ,{ 220, 220, 170, 130, 170}
04414      ,{ 340, 340, 310, 270, 310}
04415      }
04416      ,{{ 340, 340, 310, 330, 310}
04417      ,{ 340, 340, 310, 270, 310}
04418      ,{ 340, 340, 310, 330, 310}
04419      ,{ 340, 340, 310, 270, 310}
04420      ,{ 340, 340, 310, 330, 310}
04421      }
04422      ,{{ 370, 370, 340, 300, 340}
04423      ,{ 370, 370, 340, 300, 340}
04424      ,{ 340, 340, 310, 270, 310}
04425      ,{ 270, 210, 180, 270, 180}
04426      ,{ 340, 340, 310, 270, 310}
04427      }
04428      ,{{ 400, 400, 310, 330, 310}
04429      ,{ 340, 340, 310, 270, 310}
04430      ,{ 340, 340, 310, 330, 310}
04431      ,{ 340, 340, 310, 270, 310}
04432      ,{ 400, 400, 310, 270, 310}
04433      }
04434      }
04435      ,{{{ 400, 340, 400, 340, 400}
04436      ,{ 400, 340, 400, 340, 400}
04437      ,{ 310, 310, 310, 310, 310}
04438      ,{ 310, 310, 310, 310, 310}
04439      ,{ 310, 310, 310, 310, 310}
04440      }
04441      ,{{{ 310, 310, 310, 310, 310}
04442      ,{ 270, 270, 270, 270, 270}
04443      ,{ 310, 310, 310, 310, 310}
04444      ,{ 230, 170, 230, 170, 230}
04445      ,{ 310, 310, 310, 310, 310}
04446      }
04447      ,{{{ 310, 310, 310, 310, 310}
04448      ,{ 310, 310, 310, 310, 310}
04449      ,{ 310, 310, 310, 310, 310}
04450      ,{ 310, 310, 310, 310, 310}
04451      ,{ 310, 310, 310, 310, 310}
04452      }
04453      ,{{{ 400, 340, 400, 340, 400}
04454      ,{ 400, 340, 400, 340, 400}
04455      ,{ 310, 310, 310, 310, 310}
04456      ,{ 180, 180, 180, 180, 180}
04457      ,{ 310, 310, 310, 310, 310}
04458      }
04459      ,{{{ 310, 310, 310, 310, 310}
04460      ,{ 310, 310, 310, 310, 310}
04461      ,{ 310, 310, 310, 310, 310}
04462      ,{ 310, 310, 310, 310, 310}
04463      ,{ 310, 310, 310, 310, 310}
04464      }
04465      }
04466      ,{{{ 340, 290, 340, 340, 340}
04467      ,{ 340, 260, 340, 340, 340}
04468      ,{ 310, 290, 310, 180, 310}
04469      ,{ 310, 230, 310, 310, 310}
04470      ,{ 310, 290, 310, 310, 310}
04471      }
04472      ,{{ 310, 230, 310, 180, 310}
04473      ,{ 270, 190, 270, 140, 270}
04474      ,{ 310, 230, 310, 180, 310}
04475      ,{ 170, 20, 170, 170, 170}
04476      ,{ 310, 230, 310, 180, 310}
04477      }
04478      ,{{ 310, 290, 310, 180, 310}
04479      ,{ 310, 230, 310, 180, 310}
04480      ,{ 310, 290, 310, 180, 310}
04481      ,{ 310, 230, 310, 180, 310}
04482      ,{ 310, 290, 310, 180, 310}
04483      }
04484      ,{{ 340, 260, 340, 340, 340}
04485      ,{ 340, 260, 340, 340, 340}
04486      ,{ 310, 230, 310, 180, 310}
04487      ,{ 310, 230, 180, 310, 180}
04488      ,{ 310, 230, 310, 180, 310}
04489      }
04490      ,{{ 310, 290, 310, 310, 310}
04491      ,{ 310, 230, 310, 180, 310}
04492      ,{ 310, 290, 310, 180, 310}
04493      ,{ 310, 230, 310, 180, 310}
04494      ,{ 310, 230, 310, 310, 310}
04495      }
```

```
04496     }
04497     , {{ { 400, 340, 400, 340, 340}
04498     , { 400, 340, 400, 340, 340}
04499     , { 310, 310, 310, 310, 310}
04500     , { 310, 310, 310, 310, 310}
04501     , { 310, 310, 310, 310, 310}
04502     }
04503     , {{ { 330, 310, 310, 310, 330}
04504     , { 330, 270, 270, 270, 330}
04505     , { 310, 310, 310, 310, 310}
04506     , { 230, 170, 230, 170, 170}
04507     , { 310, 310, 310, 310, 310}
04508     }
04509     , {{ { 310, 310, 310, 310, 310}
04510     , { 310, 310, 310, 310, 310}
04511     , { 310, 310, 310, 310, 310}
04512     , { 310, 310, 310, 310, 310}
04513     , { 310, 310, 310, 310, 310}
04514     }
04515     , {{ { 400, 340, 400, 340, 340}
04516     , { 400, 340, 400, 340, 340}
04517     , { 310, 310, 310, 310, 310}
04518     , { 310, 180, 180, 180, 310}
04519     , { 310, 310, 310, 310, 310}
04520     }
04521     , {{ { 310, 310, 310, 310, 310}
04522     , { 310, 310, 310, 310, 310}
04523     , { 310, 310, 310, 310, 310}
04524     , { 310, 310, 310, 310, 310}
04525     , { 310, 310, 310, 310, 310}
04526     }
04527     }
04528     }
04529     , {{{ { 370, 340, 310, 310, 370}
04530     , { 370, 340, 310, 310, 370}
04531     , { 320, 320, 290, 310, 290}
04532     , { 330, 330, 290, 290, 290}
04533     , { 320, 320, 290, 310, 290}
04534     }
04535     , {{ { 370, 340, 310, 310, 370}
04536     , { 370, 340, 310, 310, 370}
04537     , { 320, 320, 280, 280, 280}
04538     , { 240, 220, 240, 180, 240}
04539     , { 320, 320, 280, 280, 280}
04540     }
04541     , {{ { 330, 330, 290, 310, 290}
04542     , { 330, 330, 290, 290, 290}
04543     , { 320, 320, 290, 310, 290}
04544     , { 330, 330, 290, 290, 290}
04545     , { 320, 320, 290, 310, 290}
04546     }
04547     , {{ { 320, 320, 310, 280, 310}
04548     , { 310, 290, 310, 250, 310}
04549     , { 320, 320, 280, 280, 280}
04550     , { 260, 180, 130, 260, 260}
04551     , { 320, 320, 280, 280, 280}
04552     }
04553     , {{ { 330, 330, 290, 310, 290}
04554     , { 330, 330, 290, 290, 290}
04555     , { 320, 320, 290, 310, 290}
04556     , { 330, 330, 290, 290, 290}
04557     , { 290, 290, 200, 200, 200}
04558     }
04559     }
04560     , {{{ { 340, 340, 310, 310, 310}
04561     , { 340, 340, 310, 270, 310}
04562     , { 320, 320, 290, 310, 290}
04563     , { 330, 330, 290, 250, 290}
04564     , { 320, 320, 290, 310, 290}
04565     }
04566     , {{ { 340, 340, 310, 270, 310}
04567     , { 340, 340, 310, 270, 310}
04568     , { 320, 320, 280, 240, 280}
04569     , { 220, 220, 180, 140, 180}
04570     , { 320, 320, 280, 240, 280}
04571     }
04572     , {{ { 330, 330, 290, 310, 290}
04573     , { 330, 330, 290, 250, 290}
04574     , { 320, 320, 290, 310, 290}
04575     , { 330, 330, 290, 250, 290}
04576     , { 320, 320, 290, 310, 290}
04577     }
04578     , {{ { 320, 320, 280, 240, 280}
04579     , { 290, 290, 250, 210, 250}
04580     , { 320, 320, 280, 240, 280}
04581     , { 220, 170, 130, 220, 130}
04582     , { 320, 320, 280, 240, 280}
```

```
04583     }
04584     ,{{ 330, 330, 290, 310, 290}
04585     ,{ 330, 330, 290, 250, 290}
04586     ,{ 320, 320, 290, 310, 290}
04587     ,{ 330, 330, 290, 250, 290}
04588     ,{ 290, 290, 200, 160, 200}
04589     }
04590     }
04591     ,{{{ 310, 310, 310, 310, 310}
04592     ,{ 310, 310, 310, 310, 310}
04593     ,{ 290, 290, 290, 290, 290}
04594     ,{ 290, 290, 290, 290, 290}
04595     ,{ 290, 290, 290, 290, 290}
04596     }
04597     ,{{{ 310, 310, 310, 310, 310}
04598     ,{ 310, 310, 310, 310, 310}
04599     ,{ 280, 280, 280, 280, 280}
04600     ,{ 240, 180, 240, 180, 240}
04601     ,{ 280, 280, 280, 280, 280}
04602     }
04603     ,{{{ 290, 290, 290, 290, 290}
04604     ,{ 290, 290, 290, 290, 290}
04605     ,{ 290, 290, 290, 290, 290}
04606     ,{ 290, 290, 290, 290, 290}
04607     ,{ 290, 290, 290, 290, 290}
04608     }
04609     ,{{{ 310, 280, 310, 280, 310}
04610     ,{ 310, 250, 310, 250, 310}
04611     ,{ 280, 280, 280, 280, 280}
04612     ,{ 130, 130, 130, 130, 130}
04613     ,{ 280, 280, 280, 280, 280}
04614     }
04615     ,{{{ 290, 290, 290, 290, 290}
04616     ,{ 290, 290, 290, 290, 290}
04617     ,{ 290, 290, 290, 290, 290}
04618     ,{ 290, 290, 290, 290, 290}
04619     ,{ 200, 200, 200, 200, 200}
04620     }
04621     }
04622     ,{{{ 310, 270, 310, 260, 310}
04623     ,{ 310, 230, 310, 250, 310}
04624     ,{ 290, 270, 290, 160, 290}
04625     ,{ 290, 210, 290, 260, 290}
04626     ,{ 290, 270, 290, 200, 290}
04627     }
04628     ,{{{ 310, 230, 310, 180, 310}
04629     ,{ 310, 230, 310, 180, 310}
04630     ,{ 280, 200, 280, 150, 280}
04631     ,{ 180, 100, 180, 180, 180}
04632     ,{ 280, 200, 280, 150, 280}
04633     }
04634     ,{{{ 290, 270, 290, 160, 290}
04635     ,{ 290, 210, 290, 160, 290}
04636     ,{ 290, 270, 290, 160, 290}
04637     ,{ 290, 210, 290, 160, 290}
04638     ,{ 290, 270, 290, 160, 290}
04639     }
04640     ,{{{ 280, 200, 280, 260, 280}
04641     ,{ 250, 170, 250, 250, 250}
04642     ,{ 280, 200, 280, 150, 280}
04643     ,{ 260, 180, 130, 260, 130}
04644     ,{ 280, 200, 280, 150, 280}
04645     }
04646     ,{{{ 290, 270, 290, 200, 290}
04647     ,{ 290, 210, 290, 160, 290}
04648     ,{ 290, 270, 290, 160, 290}
04649     ,{ 290, 210, 290, 160, 290}
04650     ,{ 200, 120, 200, 200, 200}
04651     }
04652     }
04653     ,{{{ 370, 310, 310, 310, 370}
04654     ,{ 370, 310, 310, 310, 370}
04655     ,{ 290, 290, 290, 290, 290}
04656     ,{ 290, 290, 290, 290, 290}
04657     ,{ 290, 290, 290, 290, 290}
04658     }
04659     ,{{{ 370, 310, 310, 310, 370}
04660     ,{ 370, 310, 310, 310, 370}
04661     ,{ 280, 280, 280, 280, 280}
04662     ,{ 240, 180, 240, 180, 180}
04663     ,{ 280, 280, 280, 280, 280}
04664     }
04665     ,{{{ 290, 290, 290, 290, 290}
04666     ,{ 290, 290, 290, 290, 290}
04667     ,{ 290, 290, 290, 290, 290}
04668     ,{ 290, 290, 290, 290, 290}
04669     ,{ 290, 290, 290, 290, 290}
```

```
04670     }
04671     ,{{ 310, 280, 310, 280, 280}
04672     ,{ 310, 250, 310, 250, 250}
04673     ,{ 280, 280, 280, 280, 280}
04674     ,{ 260, 130, 130, 130, 260}
04675     ,{ 280, 280, 280, 280, 280}
04676     }
04677     ,{{ 290, 290, 290, 290, 290}
04678     ,{ 290, 290, 290, 290, 290}
04679     ,{ 290, 290, 290, 290, 290}
04680     ,{ 290, 290, 290, 290, 290}
04681     ,{ 200, 200, 200, 200, 200}
04682     }
04683     }
04684     }
04685     ,{{{ 370, 340, 310, 330, 370}
04686     ,{ 370, 340, 310, 310, 370}
04687     ,{ 340, 340, 310, 330, 310}
04688     ,{ 340, 340, 310, 310, 310}
04689     ,{ 340, 340, 310, 330, 310}
04690     }
04691     ,{{ 370, 340, 310, 310, 370}
04692     ,{ 370, 340, 310, 310, 370}
04693     ,{ 300, 300, 260, 260, 260}
04694     ,{ 260, 240, 260, 200, 260}
04695     ,{ 300, 300, 260, 260, 260}
04696     }
04697     ,{{ 340, 340, 310, 330, 310}
04698     ,{ 340, 340, 310, 310, 310}
04699     ,{ 340, 340, 310, 330, 310}
04700     ,{ 340, 340, 310, 310, 310}
04701     ,{ 340, 340, 310, 330, 310}
04702     }
04703     ,{{ 300, 300, 270, 280, 280}
04704     ,{ 270, 250, 270, 210, 270}
04705     ,{ 300, 300, 260, 260, 260}
04706     ,{ 280, 200, 150, 280, 280}
04707     ,{ 300, 300, 260, 260, 260}
04708     }
04709     ,{{ 340, 340, 310, 310, 310}
04710     ,{ 340, 340, 310, 310, 310}
04711     ,{ 310, 310, 280, 300, 280}
04712     ,{ 340, 340, 310, 310, 310}
04713     ,{ 320, 320, 220, 220, 220}
04714     }
04715     }
04716     ,{{{ 340, 340, 310, 330, 310}
04717     ,{ 340, 340, 310, 270, 310}
04718     ,{ 340, 340, 310, 330, 310}
04719     ,{ 340, 340, 310, 270, 310}
04720     ,{ 340, 340, 310, 330, 310}
04721     }
04722     ,{{ 340, 340, 310, 270, 310}
04723     ,{ 340, 340, 310, 270, 310}
04724     ,{ 300, 300, 260, 220, 260}
04725     ,{ 240, 240, 200, 160, 200}
04726     ,{ 300, 300, 260, 220, 260}
04727     }
04728     ,{{ 340, 340, 310, 330, 310}
04729     ,{ 340, 340, 310, 270, 310}
04730     ,{ 340, 340, 310, 330, 310}
04731     ,{ 340, 340, 310, 270, 310}
04732     ,{ 340, 340, 310, 330, 310}
04733     }
04734     ,{{ 300, 300, 260, 240, 260}
04735     ,{ 250, 250, 210, 170, 210}
04736     ,{ 300, 300, 260, 220, 260}
04737     ,{ 240, 190, 150, 240, 150}
04738     ,{ 300, 300, 260, 220, 260}
04739     }
04740     ,{{ 340, 340, 310, 300, 310}
04741     ,{ 340, 340, 310, 270, 310}
04742     ,{ 310, 310, 280, 300, 280}
04743     ,{ 340, 340, 310, 270, 310}
04744     ,{ 320, 320, 220, 180, 220}
04745     }
04746     }
04747     ,{{{ 310, 310, 310, 310, 310}
04748     ,{ 310, 310, 310, 310, 310}
04749     ,{ 310, 310, 310, 310, 310}
04750     ,{ 310, 310, 310, 310, 310}
04751     ,{ 310, 310, 310, 310, 310}
04752     }
04753     ,{{ 310, 310, 310, 310, 310}
04754     ,{ 310, 310, 310, 310, 310}
04755     ,{ 260, 260, 260, 260, 260}
04756     ,{ 260, 200, 260, 200, 260}
```

```
04757      , { 260, 260, 260, 260, 260}
04758      }
04759      , {{ 310, 310, 310, 310, 310}
04760      , { 310, 310, 310, 310, 310}
04761      , { 310, 310, 310, 310, 310}
04762      , { 310, 310, 310, 310, 310}
04763      , { 310, 310, 310, 310, 310}
04764      }
04765      , {{ 270, 260, 270, 260, 270}
04766      , { 270, 210, 270, 210, 270}
04767      , { 260, 260, 260, 260, 260}
04768      , { 150, 150, 150, 150, 150}
04769      , { 260, 260, 260, 260, 260}
04770      }
04771      , {{ 310, 310, 310, 310, 310}
04772      , { 310, 310, 310, 310, 310}
04773      , { 280, 280, 280, 280, 280}
04774      , { 310, 310, 310, 310, 310}
04775      , { 220, 220, 220, 220, 220}
04776      }
04777      }
04778      , {{{ 310, 290, 310, 280, 310}
04779      , { 310, 230, 310, 210, 310}
04780      , { 310, 290, 310, 180, 310}
04781      , { 310, 230, 310, 280, 310}
04782      , { 310, 290, 310, 220, 310}
04783      }
04784      , {{{ 310, 230, 310, 200, 310}
04785      , { 310, 230, 310, 180, 310}
04786      , { 260, 180, 260, 130, 260}
04787      , { 200, 120, 200, 200, 200}
04788      , { 260, 180, 260, 130, 260}
04789      }
04790      , {{{ 310, 290, 310, 180, 310}
04791      , { 310, 230, 310, 180, 310}
04792      , { 310, 290, 310, 180, 310}
04793      , { 310, 230, 310, 180, 310}
04794      , { 310, 290, 310, 180, 310}
04795      }
04796      , {{{ 280, 200, 260, 280, 260}
04797      , { 210, 130, 210, 210, 210}
04798      , { 260, 180, 260, 130, 260}
04799      , { 280, 200, 150, 280, 150}
04800      , { 260, 180, 260, 130, 260}
04801      }
04802      , {{{ 310, 260, 310, 220, 310}
04803      , { 310, 230, 310, 180, 310}
04804      , { 280, 260, 280, 150, 280}
04805      , { 310, 230, 310, 180, 310}
04806      , { 220, 140, 220, 220, 220}
04807      }
04808      }
04809      , {{{ 370, 310, 310, 310, 370}
04810      , { 370, 310, 310, 310, 370}
04811      , { 310, 310, 310, 310, 310}
04812      , { 310, 310, 310, 310, 310}
04813      , { 310, 310, 310, 310, 310}
04814      }
04815      , {{{ 370, 310, 310, 310, 370}
04816      , { 370, 310, 310, 310, 370}
04817      , { 260, 260, 260, 260, 260}
04818      , { 260, 200, 260, 200, 200}
04819      , { 260, 260, 260, 260, 260}
04820      }
04821      , {{{ 310, 310, 310, 310, 310}
04822      , { 310, 310, 310, 310, 310}
04823      , { 310, 310, 310, 310, 310}
04824      , { 310, 310, 310, 310, 310}
04825      , { 310, 310, 310, 310, 310}
04826      }
04827      , {{{ 280, 260, 270, 260, 280}
04828      , { 270, 210, 270, 210, 210}
04829      , { 260, 260, 260, 260, 260}
04830      , { 280, 150, 150, 150, 280}
04831      , { 260, 260, 260, 260, 260}
04832      }
04833      , {{{ 310, 310, 310, 310, 310}
04834      , { 310, 310, 310, 310, 310}
04835      , { 280, 280, 280, 280, 280}
04836      , { 310, 310, 310, 310, 310}
04837      , { 220, 220, 220, 220, 220}
04838      }
04839      }
04840      }
04841      , {{{ 430, 430, 400, 370, 430}
04842      , { 430, 410, 400, 370, 430}
04843      , { 370, 370, 340, 360, 340}
```



```
04844     , { 370, 370, 340, 340, 340}
04845     , { 430, 430, 340, 360, 340}
04846     }
04847     , {{ 430, 410, 370, 370, 430}
04848     , { 430, 410, 370, 370, 430}
04849     , { 370, 370, 340, 340, 340}
04850     , { 320, 290, 320, 260, 320}
04851     , { 370, 370, 340, 340, 340}
04852     }
04853     , {{ 370, 370, 340, 360, 340}
04854     , { 370, 370, 340, 340, 340}
04855     , { 370, 370, 340, 360, 340}
04856     , { 370, 370, 340, 340, 340}
04857     , { 370, 370, 340, 360, 340}
04858     }
04859     , {{ 400, 370, 400, 340, 400}
04860     , { 400, 370, 400, 340, 400}
04861     , { 370, 370, 340, 340, 340}
04862     , { 340, 260, 210, 340, 340}
04863     , { 370, 370, 340, 340, 340}
04864     }
04865     , {{ 430, 430, 340, 360, 340}
04866     , { 370, 370, 340, 340, 340}
04867     , { 370, 370, 340, 360, 340}
04868     , { 370, 370, 340, 340, 340}
04869     , { 430, 430, 340, 340, 340}
04870     }
04871     }
04872     , {{{ 430, 430, 370, 360, 370}
04873     , { 410, 410, 370, 360, 370}
04874     , { 370, 370, 340, 360, 340}
04875     , { 370, 370, 340, 300, 340}
04876     , { 430, 430, 340, 360, 340}
04877     }
04878     , {{ 410, 410, 370, 360, 370}
04879     , { 410, 410, 370, 360, 370}
04880     , { 370, 370, 340, 300, 340}
04881     , { 290, 290, 260, 220, 260}
04882     , { 370, 370, 340, 300, 340}
04883     }
04884     , {{{ 370, 370, 340, 360, 340}
04885     , { 370, 370, 340, 300, 340}
04886     , { 370, 370, 340, 360, 340}
04887     , { 370, 370, 340, 300, 340}
04888     , { 370, 370, 340, 360, 340}
04889     }
04890     , {{{ 370, 370, 340, 300, 340}
04891     , { 370, 370, 340, 300, 340}
04892     , { 370, 370, 340, 300, 340}
04893     , { 300, 240, 210, 300, 210}
04894     , { 370, 370, 340, 300, 340}
04895     }
04896     , {{{ 430, 430, 340, 360, 340}
04897     , { 370, 370, 340, 300, 340}
04898     , { 370, 370, 340, 360, 340}
04899     , { 370, 370, 340, 300, 340}
04900     , { 430, 430, 340, 300, 340}
04901     }
04902     }
04903     , {{{ 400, 370, 400, 370, 400}
04904     , { 400, 370, 400, 370, 400}
04905     , { 340, 340, 340, 340, 340}
04906     , { 340, 340, 340, 340, 340}
04907     , { 340, 340, 340, 340, 340}
04908     }
04909     , {{ 370, 370, 370, 370, 370}
04910     , { 370, 370, 370, 370, 370}
04911     , { 340, 340, 340, 340, 340}
04912     , { 320, 260, 320, 260, 320}
04913     , { 340, 340, 340, 340, 340}
04914     }
04915     , {{{ 340, 340, 340, 340, 340}
04916     , { 340, 340, 340, 340, 340}
04917     , { 340, 340, 340, 340, 340}
04918     , { 340, 340, 340, 340, 340}
04919     , { 340, 340, 340, 340, 340}
04920     }
04921     , {{ 400, 340, 400, 340, 400}
04922     , { 400, 340, 400, 340, 400}
04923     , { 340, 340, 340, 340, 340}
04924     , { 210, 210, 210, 210, 210}
04925     , { 340, 340, 340, 340, 340}
04926     }
04927     , {{{ 340, 340, 340, 340, 340}
04928     , { 340, 340, 340, 340, 340}
04929     , { 340, 340, 340, 340, 340}
04930     , { 340, 340, 340, 340, 340}
```

```
04931      , { 340, 340, 340, 340, 340}
04932      }
04933      }
04934      ,{{{ 370, 320, 370, 340, 370}
04935      , { 370, 290, 370, 340, 370}
04936      , { 340, 320, 340, 210, 340}
04937      , { 340, 260, 340, 340, 340}
04938      , { 340, 320, 340, 340, 340}
04939      }
04940      ,{{{ 370, 290, 370, 260, 370}
04941      , { 370, 290, 370, 240, 370}
04942      , { 340, 260, 340, 210, 340}
04943      , { 260, 180, 260, 260, 260}
04944      , { 340, 260, 340, 210, 340}
04945      }
04946      ,{{{ 340, 320, 340, 210, 340}
04947      , { 340, 260, 340, 210, 340}
04948      , { 340, 320, 340, 210, 340}
04949      , { 340, 260, 340, 210, 340}
04950      , { 340, 320, 340, 210, 340}
04951      }
04952      ,{{{ 340, 260, 340, 340, 340}
04953      , { 340, 260, 340, 340, 340}
04954      , { 340, 260, 340, 210, 340}
04955      , { 340, 260, 210, 340, 210}
04956      , { 340, 260, 340, 210, 340}
04957      }
04958      ,{{{ 340, 320, 340, 340, 340}
04959      , { 340, 260, 340, 210, 340}
04960      , { 340, 320, 340, 210, 340}
04961      , { 340, 260, 340, 210, 340}
04962      , { 340, 260, 340, 340, 340}
04963      }
04964      }
04965      ,{{{ 430, 370, 400, 370, 430}
04966      , { 430, 370, 400, 370, 430}
04967      , { 340, 340, 340, 340, 340}
04968      , { 340, 340, 340, 340, 340}
04969      , { 340, 340, 340, 340, 340}
04970      }
04971      ,{{{ 430, 370, 370, 370, 430}
04972      , { 430, 370, 370, 370, 430}
04973      , { 340, 340, 340, 340, 340}
04974      , { 320, 260, 320, 260, 260}
04975      , { 340, 340, 340, 340, 340}
04976      }
04977      ,{{{ 340, 340, 340, 340, 340}
04978      , { 340, 340, 340, 340, 340}
04979      , { 340, 340, 340, 340, 340}
04980      , { 340, 340, 340, 340, 340}
04981      , { 340, 340, 340, 340, 340}
04982      }
04983      ,{{{ 400, 340, 400, 340, 340}
04984      , { 400, 340, 400, 340, 340}
04985      , { 340, 340, 340, 340, 340}
04986      , { 340, 210, 210, 210, 340}
04987      , { 340, 340, 340, 340, 340}
04988      }
04989      ,{{{ 340, 340, 340, 340, 340}
04990      , { 340, 340, 340, 340, 340}
04991      , { 340, 340, 340, 340, 340}
04992      , { 340, 340, 340, 340, 340}
04993      , { 340, 340, 340, 340, 340}
04994      }
04995      }
04996      }
04997      }
04998      ,{{{ INF, INF, INF, INF, INF}
04999      , { INF, INF, INF, INF, INF}
05000      , { INF, INF, INF, INF, INF}
05001      , { INF, INF, INF, INF, INF}
05002      , { INF, INF, INF, INF, INF}
05003      }
05004      ,{{{ INF, INF, INF, INF, INF}
05005      , { INF, INF, INF, INF, INF}
05006      , { INF, INF, INF, INF, INF}
05007      , { INF, INF, INF, INF, INF}
05008      , { INF, INF, INF, INF, INF}
05009      }
05010      ,{{{ INF, INF, INF, INF, INF}
05011      , { INF, INF, INF, INF, INF}
05012      , { INF, INF, INF, INF, INF}
05013      , { INF, INF, INF, INF, INF}
05014      , { INF, INF, INF, INF, INF}
05015      }
05016      ,{{{ INF, INF, INF, INF, INF}
05017      , { INF, INF, INF, INF, INF}
```

```
05018     , {   INF,   INF,   INF,   INF,   INF }
05019     , {   INF,   INF,   INF,   INF,   INF }
05020     , {   INF,   INF,   INF,   INF,   INF }
05021     }
05022     , { {   INF,   INF,   INF,   INF,   INF }
05023     , {   INF,   INF,   INF,   INF,   INF }
05024     , {   INF,   INF,   INF,   INF,   INF }
05025     , {   INF,   INF,   INF,   INF,   INF }
05026     , {   INF,   INF,   INF,   INF,   INF }
05027     }
05028     }
05029     , { { {   INF,   INF,   INF,   INF,   INF }
05030     , {   INF,   INF,   INF,   INF,   INF }
05031     , {   INF,   INF,   INF,   INF,   INF }
05032     , {   INF,   INF,   INF,   INF,   INF }
05033     , {   INF,   INF,   INF,   INF,   INF }
05034     }
05035     , { {   INF,   INF,   INF,   INF,   INF }
05036     , {   INF,   INF,   INF,   INF,   INF }
05037     , {   INF,   INF,   INF,   INF,   INF }
05038     , {   INF,   INF,   INF,   INF,   INF }
05039     , {   INF,   INF,   INF,   INF,   INF }
05040     }
05041     , { {   INF,   INF,   INF,   INF,   INF }
05042     , {   INF,   INF,   INF,   INF,   INF }
05043     , {   INF,   INF,   INF,   INF,   INF }
05044     , {   INF,   INF,   INF,   INF,   INF }
05045     , {   INF,   INF,   INF,   INF,   INF }
05046     }
05047     , { {   INF,   INF,   INF,   INF,   INF }
05048     , {   INF,   INF,   INF,   INF,   INF }
05049     , {   INF,   INF,   INF,   INF,   INF }
05050     , {   INF,   INF,   INF,   INF,   INF }
05051     , {   INF,   INF,   INF,   INF,   INF }
05052     }
05053     , { {   INF,   INF,   INF,   INF,   INF }
05054     , {   INF,   INF,   INF,   INF,   INF }
05055     , {   INF,   INF,   INF,   INF,   INF }
05056     , {   INF,   INF,   INF,   INF,   INF }
05057     , {   INF,   INF,   INF,   INF,   INF }
05058     }
05059     }
05060     , { { {   INF,   INF,   INF,   INF,   INF }
05061     , {   INF,   INF,   INF,   INF,   INF }
05062     , {   INF,   INF,   INF,   INF,   INF }
05063     , {   INF,   INF,   INF,   INF,   INF }
05064     , {   INF,   INF,   INF,   INF,   INF }
05065     }
05066     , { {   INF,   INF,   INF,   INF,   INF }
05067     , {   INF,   INF,   INF,   INF,   INF }
05068     , {   INF,   INF,   INF,   INF,   INF }
05069     , {   INF,   INF,   INF,   INF,   INF }
05070     , {   INF,   INF,   INF,   INF,   INF }
05071     }
05072     , { {   INF,   INF,   INF,   INF,   INF }
05073     , {   INF,   INF,   INF,   INF,   INF }
05074     , {   INF,   INF,   INF,   INF,   INF }
05075     , {   INF,   INF,   INF,   INF,   INF }
05076     , {   INF,   INF,   INF,   INF,   INF }
05077     }
05078     , { {   INF,   INF,   INF,   INF,   INF }
05079     , {   INF,   INF,   INF,   INF,   INF }
05080     , {   INF,   INF,   INF,   INF,   INF }
05081     , {   INF,   INF,   INF,   INF,   INF }
05082     , {   INF,   INF,   INF,   INF,   INF }
05083     }
05084     , { {   INF,   INF,   INF,   INF,   INF }
05085     , {   INF,   INF,   INF,   INF,   INF }
05086     , {   INF,   INF,   INF,   INF,   INF }
05087     , {   INF,   INF,   INF,   INF,   INF }
05088     , {   INF,   INF,   INF,   INF,   INF }
05089     }
05090     }
05091     , { { {   INF,   INF,   INF,   INF,   INF }
05092     , {   INF,   INF,   INF,   INF,   INF }
05093     , {   INF,   INF,   INF,   INF,   INF }
05094     , {   INF,   INF,   INF,   INF,   INF }
05095     , {   INF,   INF,   INF,   INF,   INF }
05096     }
05097     , { {   INF,   INF,   INF,   INF,   INF }
05098     , {   INF,   INF,   INF,   INF,   INF }
05099     , {   INF,   INF,   INF,   INF,   INF }
05100     , {   INF,   INF,   INF,   INF,   INF }
05101     , {   INF,   INF,   INF,   INF,   INF }
05102     }
05103     , { {   INF,   INF,   INF,   INF,   INF }
05104     , {   INF,   INF,   INF,   INF,   INF }
```

```
05105      , {   INF,   INF,   INF,   INF,   INF }
05106      , {   INF,   INF,   INF,   INF,   INF }
05107      , {   INF,   INF,   INF,   INF,   INF }
05108      }
05109      , { {   INF,   INF,   INF,   INF,   INF }
05110      , {   INF,   INF,   INF,   INF,   INF }
05111      , {   INF,   INF,   INF,   INF,   INF }
05112      , {   INF,   INF,   INF,   INF,   INF }
05113      , {   INF,   INF,   INF,   INF,   INF }
05114      }
05115      , { {   INF,   INF,   INF,   INF,   INF }
05116      , {   INF,   INF,   INF,   INF,   INF }
05117      , {   INF,   INF,   INF,   INF,   INF }
05118      , {   INF,   INF,   INF,   INF,   INF }
05119      , {   INF,   INF,   INF,   INF,   INF }
05120      }
05121      }
05122      , { { {   INF,   INF,   INF,   INF,   INF }
05123      , {   INF,   INF,   INF,   INF,   INF }
05124      , {   INF,   INF,   INF,   INF,   INF }
05125      , {   INF,   INF,   INF,   INF,   INF }
05126      , {   INF,   INF,   INF,   INF,   INF }
05127      }
05128      , { {   INF,   INF,   INF,   INF,   INF }
05129      , {   INF,   INF,   INF,   INF,   INF }
05130      , {   INF,   INF,   INF,   INF,   INF }
05131      , {   INF,   INF,   INF,   INF,   INF }
05132      , {   INF,   INF,   INF,   INF,   INF }
05133      }
05134      , { {   INF,   INF,   INF,   INF,   INF }
05135      , {   INF,   INF,   INF,   INF,   INF }
05136      , {   INF,   INF,   INF,   INF,   INF }
05137      , {   INF,   INF,   INF,   INF,   INF }
05138      , {   INF,   INF,   INF,   INF,   INF }
05139      }
05140      , { {   INF,   INF,   INF,   INF,   INF }
05141      , {   INF,   INF,   INF,   INF,   INF }
05142      , {   INF,   INF,   INF,   INF,   INF }
05143      , {   INF,   INF,   INF,   INF,   INF }
05144      , {   INF,   INF,   INF,   INF,   INF }
05145      }
05146      , { {   INF,   INF,   INF,   INF,   INF }
05147      , {   INF,   INF,   INF,   INF,   INF }
05148      , {   INF,   INF,   INF,   INF,   INF }
05149      , {   INF,   INF,   INF,   INF,   INF }
05150      , {   INF,   INF,   INF,   INF,   INF }
05151      }
05152      }
05153      }
05154      , { { { {   310,   240,   240,   310,   260 }
05155      , {   270,   240,   240,   270,   260 }
05156      , {   310,   220,   220,   310,   220 }
05157      , {   270,   240,   240,   270,   240 }
05158      , {   300,   210,   210,   300,   210 }
05159      }
05160      , { {   260,   200,   200,   230,   260 }
05161      , {   260,   200,   200,   230,   260 }
05162      , {   220,   190,   190,   220,   190 }
05163      , {   160,   100,   160,   130,   160 }
05164      , {   220,   190,   190,   220,   190 }
05165      }
05166      , { {   310,   240,   240,   310,   240 }
05167      , {   270,   240,   240,   270,   240 }
05168      , {   310,   220,   220,   310,   220 }
05169      , {   270,   240,   240,   270,   240 }
05170      , {   300,   210,   210,   300,   210 }
05171      }
05172      , { {   220,   190,   190,   220,   190 }
05173      , {   160,   100,   160,   130,   160 }
05174      , {   220,   190,   190,   220,   190 }
05175      , {   210,    50,    50,   210,   180 }
05176      , {   220,   190,   190,   220,   190 }
05177      }
05178      , { {   300,   240,   240,   300,   240 }
05179      , {   270,   240,   240,   270,   240 }
05180      , {   300,   210,   210,   300,   210 }
05181      , {   270,   240,   240,   270,   240 }
05182      , {   150,   140,   120,   150,   120 }
05183      }
05184      }
05185      , { { {   310,   200,   240,   310,   240 }
05186      , {   270,   200,   240,   270,   240 }
05187      , {   310,   190,   220,   310,   220 }
05188      , {   270,   200,   240,   270,   240 }
05189      , {   300,   170,   210,   300,   210 }
05190      }
05191      , { {   230,   160,   200,   230,   200 }
```

```
05192     , { 230, 160, 200, 230, 200}
05193     , { 220, 160, 190, 220, 190}
05194     , { 130, 70, 100, 130, 100}
05195     , { 220, 160, 190, 220, 190}
05196     }
05197     , { { 310, 200, 240, 310, 240}
05198     , { 270, 200, 240, 270, 240}
05199     , { 310, 190, 220, 310, 220}
05200     , { 270, 200, 240, 270, 240}
05201     , { 300, 170, 210, 300, 210}
05202     }
05203     , { { 220, 160, 190, 220, 190}
05204     , { 130, 70, 100, 130, 100}
05205     , { 220, 160, 190, 220, 190}
05206     , { 210, 10, 50, 210, 50}
05207     , { 220, 160, 190, 220, 190}
05208     }
05209     , { { 300, 200, 240, 300, 240}
05210     , { 270, 200, 240, 270, 240}
05211     , { 300, 170, 210, 300, 210}
05212     , { 270, 200, 240, 270, 240}
05213     , { 150, 140, 120, 150, 120}
05214     }
05215     }
05216     , { { { 240, 240, 240, 240, 240}
05217     , { 240, 240, 240, 240, 240}
05218     , { 220, 220, 220, 220, 220}
05219     , { 240, 240, 240, 240, 240}
05220     , { 210, 210, 210, 210, 210}
05221     }
05222     , { { 200, 200, 200, 200, 200}
05223     , { 200, 200, 200, 200, 200}
05224     , { 190, 190, 190, 190, 190}
05225     , { 160, 100, 160, 100, 160}
05226     , { 190, 190, 190, 190, 190}
05227     }
05228     , { { 240, 240, 240, 240, 240}
05229     , { 240, 240, 240, 240, 240}
05230     , { 220, 220, 220, 220, 220}
05231     , { 240, 240, 240, 240, 240}
05232     , { 210, 210, 210, 210, 210}
05233     }
05234     , { { 190, 190, 190, 190, 190}
05235     , { 160, 100, 160, 100, 160}
05236     , { 190, 190, 190, 190, 190}
05237     , { 50, 50, 50, 50, 50}
05238     , { 190, 190, 190, 190, 190}
05239     }
05240     , { { 240, 240, 240, 240, 240}
05241     , { 240, 240, 240, 240, 240}
05242     , { 210, 210, 210, 210, 210}
05243     , { 240, 240, 240, 240, 240}
05244     , { 120, 120, 120, 120, 120}
05245     }
05246     }
05247     , { { { 240, 150, 240, 180, 240}
05248     , { 240, 100, 240, 110, 240}
05249     , { 220, 150, 220, 90, 220}
05250     , { 240, 100, 240, 180, 240}
05251     , { 210, 130, 210, 120, 210}
05252     }
05253     , { { 200, 60, 200, 100, 200}
05254     , { 200, 60, 200, 70, 200}
05255     , { 190, 60, 190, 60, 190}
05256     , { 100, -30, 100, 100, 100}
05257     , { 190, 60, 190, 60, 190}
05258     }
05259     , { { 240, 150, 240, 110, 240}
05260     , { 240, 100, 240, 110, 240}
05261     , { 220, 150, 220, 90, 220}
05262     , { 240, 100, 240, 110, 240}
05263     , { 210, 130, 210, 80, 210}
05264     }
05265     , { { 190, 60, 190, 180, 190}
05266     , { 100, -30, 100, 100, 100}
05267     , { 190, 60, 190, 60, 190}
05268     , { 180, 40, 50, 180, 50}
05269     , { 190, 60, 190, 60, 190}
05270     }
05271     , { { 240, 130, 240, 120, 240}
05272     , { 240, 100, 240, 110, 240}
05273     , { 210, 130, 210, 80, 210}
05274     , { 240, 100, 240, 110, 240}
05275     , { 120, -10, 120, 120, 120}
05276     }
05277     }
05278     , { { { 260, 240, 240, 240, 260}
```

```

05279      , { 260, 240, 240, 240, 260}
05280      , { 220, 220, 220, 220, 220}
05281      , { 240, 240, 240, 240, 240}
05282      , { 210, 210, 210, 210, 210}
05283      }
05284      , { { 260, 200, 200, 200, 260}
05285      , { 260, 200, 200, 200, 260}
05286      , { 190, 190, 190, 190, 190}
05287      , { 160, 100, 160, 100, 100}
05288      , { 190, 190, 190, 190, 190}
05289      }
05290      , { { 240, 240, 240, 240, 240}
05291      , { 240, 240, 240, 240, 240}
05292      , { 220, 220, 220, 220, 220}
05293      , { 240, 240, 240, 240, 240}
05294      , { 210, 210, 210, 210, 210}
05295      }
05296      , { { 190, 190, 190, 190, 190}
05297      , { 160, 100, 160, 100, 100}
05298      , { 190, 190, 190, 190, 190}
05299      , { 180, 50, 50, 50, 180}
05300      , { 190, 190, 190, 190, 190}
05301      }
05302      , { { 240, 240, 240, 240, 240}
05303      , { 240, 240, 240, 240, 240}
05304      , { 210, 210, 210, 210, 210}
05305      , { 240, 240, 240, 240, 240}
05306      , { 120, 120, 120, 120, 120}
05307      }
05308      }
05309      }
05310      , { { { 280, 210, 210, 280, 270}
05311      , { 270, 210, 210, 240, 270}
05312      , { 280, 190, 190, 280, 190}
05313      , { 210, 180, 180, 210, 180}
05314      , { 280, 190, 190, 280, 190}
05315      }
05316      , { { 270, 210, 210, 240, 270}
05317      , { 270, 210, 210, 240, 270}
05318      , { 220, 190, 190, 220, 190}
05319      , { 70, 10, 70, 40, 70}
05320      , { 220, 190, 190, 220, 190}
05321      }
05322      , { { 280, 190, 190, 280, 190}
05323      , { 210, 180, 180, 210, 180}
05324      , { 280, 190, 190, 280, 190}
05325      , { 210, 180, 180, 210, 180}
05326      , { 280, 190, 190, 280, 190}
05327      }
05328      , { { 220, 190, 190, 220, 190}
05329      , { 130, 70, 130, 100, 130}
05330      , { 220, 190, 190, 220, 190}
05331      , { 210, 50, 50, 210, 180}
05332      , { 220, 190, 190, 220, 190}
05333      }
05334      , { { 280, 190, 190, 280, 190}
05335      , { 210, 180, 180, 210, 180}
05336      , { 280, 190, 190, 280, 190}
05337      , { 210, 180, 180, 210, 180}
05338      , { 140, 140, 110, 140, 110}
05339      }
05340      }
05341      , { { { 280, 190, 210, 280, 210}
05342      , { 240, 190, 210, 240, 210}
05343      , { 280, 160, 190, 280, 190}
05344      , { 210, 150, 180, 210, 180}
05345      , { 280, 150, 190, 280, 190}
05346      }
05347      , { { 240, 190, 210, 240, 210}
05348      , { 240, 190, 210, 240, 210}
05349      , { 220, 150, 190, 220, 190}
05350      , { 40, -20, 10, 40, 10}
05351      , { 220, 150, 190, 220, 190}
05352      }
05353      , { { 280, 150, 190, 280, 190}
05354      , { 210, 150, 180, 210, 180}
05355      , { 280, 150, 190, 280, 190}
05356      , { 210, 150, 180, 210, 180}
05357      , { 280, 150, 190, 280, 190}
05358      }
05359      , { { 220, 150, 190, 220, 190}
05360      , { 100, 40, 70, 100, 70}
05361      , { 220, 150, 190, 220, 190}
05362      , { 210, 10, 50, 210, 50}
05363      , { 220, 150, 190, 220, 190}
05364      }
05365      , { { 280, 160, 190, 280, 190}

```

```
05366     , { 210, 150, 180, 210, 180}
05367     , { 280, 160, 190, 280, 190}
05368     , { 210, 150, 180, 210, 180}
05369     , { 140, 140, 110, 140, 110}
05370     }
05371     }
05372     , {{{ 210, 210, 210, 210, 210}
05373     , { 210, 210, 210, 210, 210}
05374     , { 190, 190, 190, 190, 190}
05375     , { 180, 180, 180, 180, 180}
05376     , { 190, 190, 190, 190, 190}
05377     }
05378     , {{{ 210, 210, 210, 210, 210}
05379     , { 210, 210, 210, 210, 210}
05380     , { 190, 190, 190, 190, 190}
05381     , { 70, 10, 70, 10, 70}
05382     , { 190, 190, 190, 190, 190}
05383     }
05384     , {{{ 190, 190, 190, 190, 190}
05385     , { 180, 180, 180, 180, 180}
05386     , { 190, 190, 190, 190, 190}
05387     , { 180, 180, 180, 180, 180}
05388     , { 190, 190, 190, 190, 190}
05389     }
05390     , {{{ 190, 190, 190, 190, 190}
05391     , { 130, 70, 130, 70, 130}
05392     , { 190, 190, 190, 190, 190}
05393     , { 50, 50, 50, 50, 50}
05394     , { 190, 190, 190, 190, 190}
05395     }
05396     , {{{ 190, 190, 190, 190, 190}
05397     , { 180, 180, 180, 180, 180}
05398     , { 190, 190, 190, 190, 190}
05399     , { 180, 180, 180, 180, 180}
05400     , { 110, 110, 110, 110, 110}
05401     }
05402     }
05403     , {{{ 210, 120, 210, 180, 210}
05404     , { 210, 80, 210, 80, 210}
05405     , { 190, 120, 190, 60, 190}
05406     , { 180, 50, 180, 180, 180}
05407     , { 190, 110, 190, 110, 190}
05408     }
05409     , {{{ 210, 80, 210, 80, 210}
05410     , { 210, 80, 210, 80, 210}
05411     , { 190, 50, 190, 60, 190}
05412     , { 10, -120, 10, 10, 10}
05413     , { 190, 50, 190, 60, 190}
05414     }
05415     , {{{ 190, 110, 190, 60, 190}
05416     , { 180, 50, 180, 50, 180}
05417     , { 190, 110, 190, 60, 190}
05418     , { 180, 50, 180, 50, 180}
05419     , { 190, 110, 190, 60, 190}
05420     }
05421     , {{{ 190, 50, 190, 180, 190}
05422     , { 70, -60, 70, 70, 70}
05423     , { 190, 50, 190, 60, 190}
05424     , { 180, 40, 50, 180, 50}
05425     , { 190, 50, 190, 60, 190}
05426     }
05427     , {{{ 190, 120, 190, 110, 190}
05428     , { 180, 50, 180, 50, 180}
05429     , { 190, 120, 190, 60, 190}
05430     , { 180, 50, 180, 50, 180}
05431     , { 110, -20, 110, 110, 110}
05432     }
05433     }
05434     , {{{ 270, 210, 210, 210, 270}
05435     , { 270, 210, 210, 210, 270}
05436     , { 190, 190, 190, 190, 190}
05437     , { 180, 180, 180, 180, 180}
05438     , { 190, 190, 190, 190, 190}
05439     }
05440     , {{{ 270, 210, 210, 210, 270}
05441     , { 270, 210, 210, 210, 270}
05442     , { 190, 190, 190, 190, 190}
05443     , { 70, 10, 70, 10, 10}
05444     , { 190, 190, 190, 190, 190}
05445     }
05446     , {{{ 190, 190, 190, 190, 190}
05447     , { 180, 180, 180, 180, 180}
05448     , { 190, 190, 190, 190, 190}
05449     , { 180, 180, 180, 180, 180}
05450     , { 190, 190, 190, 190, 190}
05451     }
05452     , {{{ 190, 190, 190, 190, 190}
```

```
05453      , { 130, 70, 130, 70, 70}
05454      , { 190, 190, 190, 190, 190}
05455      , { 180, 50, 50, 50, 180}
05456      , { 190, 190, 190, 190, 190}
05457      }
05458      , { { 190, 190, 190, 190, 190}
05459      , { 180, 180, 180, 180, 180}
05460      , { 190, 190, 190, 190, 190}
05461      , { 180, 180, 180, 180, 180}
05462      , { 110, 110, 110, 110, 110}
05463      }
05464      }
05465      }
05466      , { { { 400, 360, 340, 400, 400}
05467      , { 400, 360, 340, 370, 400}
05468      , { 400, 310, 310, 400, 310}
05469      , { 340, 310, 310, 340, 310}
05470      , { 400, 330, 310, 400, 310}
05471      }
05472      , { { 400, 360, 340, 370, 400}
05473      , { 400, 360, 340, 370, 400}
05474      , { 340, 310, 310, 340, 310}
05475      , { 290, 230, 290, 260, 290}
05476      , { 340, 310, 310, 340, 310}
05477      }
05478      , { { 400, 310, 310, 400, 310}
05479      , { 340, 310, 310, 340, 310}
05480      , { 400, 310, 310, 400, 310}
05481      , { 340, 310, 310, 340, 310}
05482      , { 400, 310, 310, 400, 310}
05483      }
05484      , { { 360, 360, 330, 340, 330}
05485      , { 360, 360, 330, 300, 330}
05486      , { 340, 310, 310, 340, 310}
05487      , { 340, 180, 180, 340, 310}
05488      , { 340, 310, 310, 340, 310}
05489      }
05490      , { { 400, 330, 310, 400, 310}
05491      , { 340, 310, 310, 340, 310}
05492      , { 400, 310, 310, 400, 310}
05493      , { 340, 310, 310, 340, 310}
05494      , { 340, 330, 310, 340, 310}
05495      }
05496      }
05497      , { { { 400, 360, 340, 400, 340}
05498      , { 370, 360, 340, 370, 340}
05499      , { 400, 270, 310, 400, 310}
05500      , { 340, 270, 310, 340, 310}
05501      , { 400, 330, 310, 400, 310}
05502      }
05503      , { { 370, 360, 340, 370, 340}
05504      , { 370, 360, 340, 370, 340}
05505      , { 340, 270, 310, 340, 310}
05506      , { 260, 190, 230, 260, 230}
05507      , { 340, 270, 310, 340, 310}
05508      }
05509      , { { 400, 270, 310, 400, 310}
05510      , { 340, 270, 310, 340, 310}
05511      , { 400, 270, 310, 400, 310}
05512      , { 340, 270, 310, 340, 310}
05513      , { 400, 270, 310, 400, 310}
05514      }
05515      , { { 360, 360, 310, 340, 310}
05516      , { 360, 360, 270, 300, 270}
05517      , { 340, 270, 310, 340, 310}
05518      , { 340, 140, 180, 340, 180}
05519      , { 340, 270, 310, 340, 310}
05520      }
05521      , { { 400, 330, 310, 400, 310}
05522      , { 340, 270, 310, 340, 310}
05523      , { 400, 270, 310, 400, 310}
05524      , { 340, 270, 310, 340, 310}
05525      , { 340, 330, 310, 340, 310}
05526      }
05527      }
05528      , { { { 340, 340, 340, 340, 340}
05529      , { 340, 340, 340, 340, 340}
05530      , { 310, 310, 310, 310, 310}
05531      , { 310, 310, 310, 310, 310}
05532      , { 310, 310, 310, 310, 310}
05533      }
05534      , { { 340, 340, 340, 340, 340}
05535      , { 340, 340, 340, 340, 340}
05536      , { 310, 310, 310, 310, 310}
05537      , { 290, 230, 290, 230, 290}
05538      , { 310, 310, 310, 310, 310}
05539      }
```



```
05540 ,{{ 310, 310, 310, 310, 310}
05541 ,{ 310, 310, 310, 310, 310}
05542 ,{ 310, 310, 310, 310, 310}
05543 ,{ 310, 310, 310, 310, 310}
05544 ,{ 310, 310, 310, 310, 310}
05545 }
05546 ,{{ 330, 310, 330, 310, 330}
05547 ,{ 330, 270, 330, 270, 330}
05548 ,{ 310, 310, 310, 310, 310}
05549 ,{ 180, 180, 180, 180, 180}
05550 ,{ 310, 310, 310, 310, 310}
05551 }
05552 ,{{ 310, 310, 310, 310, 310}
05553 ,{ 310, 310, 310, 310, 310}
05554 ,{ 310, 310, 310, 310, 310}
05555 ,{ 310, 310, 310, 310, 310}
05556 ,{ 310, 310, 310, 310, 310}
05557 }
05558 }
05559 ,{{{ 340, 230, 340, 310, 340}
05560 ,{ 340, 220, 340, 270, 340}
05561 ,{ 310, 230, 310, 180, 310}
05562 ,{ 310, 170, 310, 310, 310}
05563 ,{ 310, 230, 310, 310, 310}
05564 }
05565 ,{{ 340, 220, 340, 230, 340}
05566 ,{ 340, 220, 340, 210, 340}
05567 ,{ 310, 170, 310, 180, 310}
05568 ,{ 230, 20, 230, 230, 230}
05569 ,{ 310, 170, 310, 180, 310}
05570 }
05571 ,{{ 310, 230, 310, 180, 310}
05572 ,{ 310, 170, 310, 180, 310}
05573 ,{ 310, 230, 310, 180, 310}
05574 ,{ 310, 170, 310, 180, 310}
05575 ,{ 310, 230, 310, 180, 310}
05576 }
05577 ,{{ 310, 170, 310, 310, 310}
05578 ,{ 270, 130, 270, 270, 270}
05579 ,{ 310, 170, 310, 180, 310}
05580 ,{ 310, 170, 180, 310, 180}
05581 ,{ 310, 170, 310, 180, 310}
05582 }
05583 ,{{ 310, 230, 310, 310, 310}
05584 ,{ 310, 170, 310, 180, 310}
05585 ,{ 310, 230, 310, 180, 310}
05586 ,{ 310, 170, 310, 180, 310}
05587 ,{ 310, 170, 310, 310, 310}
05588 }
05589 }
05590 ,{{{ 400, 340, 340, 340, 400}
05591 ,{ 400, 340, 340, 340, 400}
05592 ,{ 310, 310, 310, 310, 310}
05593 ,{ 310, 310, 310, 310, 310}
05594 ,{ 310, 310, 310, 310, 310}
05595 }
05596 ,{{ 400, 340, 340, 340, 400}
05597 ,{ 400, 340, 340, 340, 400}
05598 ,{ 310, 310, 310, 310, 310}
05599 ,{ 290, 230, 290, 230, 230}
05600 ,{ 310, 310, 310, 310, 310}
05601 }
05602 ,{{ 310, 310, 310, 310, 310}
05603 ,{ 310, 310, 310, 310, 310}
05604 ,{ 310, 310, 310, 310, 310}
05605 ,{ 310, 310, 310, 310, 310}
05606 ,{ 310, 310, 310, 310, 310}
05607 }
05608 ,{{ 330, 310, 330, 310, 310}
05609 ,{ 330, 270, 330, 270, 270}
05610 ,{ 310, 310, 310, 310, 310}
05611 ,{ 310, 180, 180, 180, 310}
05612 ,{ 310, 310, 310, 310, 310}
05613 }
05614 ,{{ 310, 310, 310, 310, 310}
05615 ,{ 310, 310, 310, 310, 310}
05616 ,{ 310, 310, 310, 310, 310}
05617 ,{ 310, 310, 310, 310, 310}
05618 ,{ 310, 310, 310, 310, 310}
05619 }
05620 }
05621 }
05622 ,{{{ 370, 310, 370, 370, 370}
05623 ,{ 370, 310, 370, 340, 370}
05624 ,{ 370, 280, 280, 370, 280}
05625 ,{ 310, 280, 280, 310, 280}
05626 ,{ 370, 300, 280, 370, 280}
```

```
05627      }
05628      ,{{ 310, 280, 280, 310, 300}
05629      ,{ 300, 240, 240, 270, 300}
05630      ,{ 310, 280, 280, 310, 280}
05631      ,{ 200, 140, 200, 170, 200}
05632      ,{ 310, 280, 280, 310, 280}
05633      }
05634      ,{{ 370, 280, 280, 370, 280}
05635      ,{ 310, 280, 280, 310, 280}
05636      ,{ 370, 280, 280, 370, 280}
05637      ,{ 310, 280, 280, 310, 280}
05638      ,{ 370, 280, 280, 370, 280}
05639      }
05640      ,{{ 370, 310, 370, 340, 370}
05641      ,{ 370, 310, 370, 340, 370}
05642      ,{ 310, 280, 280, 310, 280}
05643      ,{ 310, 150, 150, 310, 280}
05644      ,{ 310, 280, 280, 310, 280}
05645      }
05646      ,{{ 370, 300, 280, 370, 280}
05647      ,{ 310, 280, 280, 310, 280}
05648      ,{ 370, 280, 280, 370, 280}
05649      ,{ 310, 280, 280, 310, 280}
05650      ,{ 310, 300, 280, 310, 280}
05651      }
05652      }
05653      ,{{{ 370, 300, 310, 370, 310}
05654      ,{ 340, 270, 310, 340, 310}
05655      ,{ 370, 240, 280, 370, 280}
05656      ,{ 310, 240, 280, 310, 280}
05657      ,{ 370, 300, 280, 370, 280}
05658      }
05659      ,{{{ 310, 240, 280, 310, 280}
05660      ,{ 270, 210, 240, 270, 240}
05661      ,{ 310, 240, 280, 310, 280}
05662      ,{ 170, 110, 140, 170, 140}
05663      ,{ 310, 240, 280, 310, 280}
05664      }
05665      ,{{{ 370, 240, 280, 370, 280}
05666      ,{ 310, 240, 280, 310, 280}
05667      ,{ 370, 240, 280, 370, 280}
05668      ,{ 310, 240, 280, 310, 280}
05669      ,{ 370, 240, 280, 370, 280}
05670      }
05671      ,{{{ 340, 270, 310, 340, 310}
05672      ,{ 340, 270, 310, 340, 310}
05673      ,{ 310, 240, 280, 310, 280}
05674      ,{ 310, 110, 150, 310, 150}
05675      ,{ 310, 240, 280, 310, 280}
05676      }
05677      ,{{{ 370, 300, 280, 370, 280}
05678      ,{ 310, 240, 280, 310, 280}
05679      ,{ 370, 240, 280, 370, 280}
05680      ,{ 310, 240, 280, 310, 280}
05681      ,{ 310, 300, 280, 310, 280}
05682      }
05683      }
05684      ,{{{ 370, 310, 370, 310, 370}
05685      ,{ 370, 310, 370, 310, 370}
05686      ,{ 280, 280, 280, 280, 280}
05687      ,{ 280, 280, 280, 280, 280}
05688      ,{ 280, 280, 280, 280, 280}
05689      }
05690      ,{{ 280, 280, 280, 280, 280}
05691      ,{ 240, 240, 240, 240, 240}
05692      ,{ 280, 280, 280, 280, 280}
05693      ,{ 200, 140, 200, 140, 200}
05694      ,{ 280, 280, 280, 280, 280}
05695      }
05696      ,{{ 280, 280, 280, 280, 280}
05697      ,{ 280, 280, 280, 280, 280}
05698      ,{ 280, 280, 280, 280, 280}
05699      ,{ 280, 280, 280, 280, 280}
05700      ,{ 280, 280, 280, 280, 280}
05701      }
05702      ,{{ 370, 310, 370, 310, 370}
05703      ,{ 370, 310, 370, 310, 370}
05704      ,{ 280, 280, 280, 280, 280}
05705      ,{ 150, 150, 150, 150, 150}
05706      ,{ 280, 280, 280, 280, 280}
05707      }
05708      ,{{ 280, 280, 280, 280, 280}
05709      ,{ 280, 280, 280, 280, 280}
05710      ,{ 280, 280, 280, 280, 280}
05711      ,{ 280, 280, 280, 280, 280}
05712      ,{ 280, 280, 280, 280, 280}
05713      }
```

```
05714     }
05715     , {{{ 310, 200, 310, 310, 310}
05716     , { 310, 170, 310, 310, 310}
05717     , { 280, 200, 280, 150, 280}
05718     , { 280, 140, 280, 280, 280}
05719     , { 280, 200, 280, 280, 280}
05720     }
05721     , {{{ 280, 140, 280, 150, 280}
05722     , { 240, 110, 240, 110, 240}
05723     , { 280, 140, 280, 150, 280}
05724     , { 140, 10, 140, 140, 140}
05725     , { 280, 140, 280, 150, 280}
05726     }
05727     , {{{ 280, 200, 280, 150, 280}
05728     , { 280, 140, 280, 150, 280}
05729     , { 280, 200, 280, 150, 280}
05730     , { 280, 140, 280, 150, 280}
05731     , { 280, 200, 280, 150, 280}
05732     }
05733     , {{{ 310, 170, 310, 310, 310}
05734     , { 310, 170, 310, 310, 310}
05735     , { 280, 140, 280, 150, 280}
05736     , { 280, 140, 150, 280, 150}
05737     , { 280, 140, 280, 150, 280}
05738     }
05739     , {{{ 280, 200, 280, 280, 280}
05740     , { 280, 140, 280, 150, 280}
05741     , { 280, 200, 280, 150, 280}
05742     , { 280, 140, 280, 150, 280}
05743     , { 280, 140, 280, 280, 280}
05744     }
05745     }
05746     , {{{ 370, 310, 370, 310, 310}
05747     , { 370, 310, 370, 310, 310}
05748     , { 280, 280, 280, 280, 280}
05749     , { 280, 280, 280, 280, 280}
05750     , { 280, 280, 280, 280, 280}
05751     }
05752     , {{{ 300, 280, 280, 280, 300}
05753     , { 300, 240, 240, 240, 300}
05754     , { 280, 280, 280, 280, 280}
05755     , { 200, 140, 200, 140, 140}
05756     , { 280, 280, 280, 280, 280}
05757     }
05758     , {{{ 280, 280, 280, 280, 280}
05759     , { 280, 280, 280, 280, 280}
05760     , { 280, 280, 280, 280, 280}
05761     , { 280, 280, 280, 280, 280}
05762     , { 280, 280, 280, 280, 280}
05763     }
05764     , {{{ 370, 310, 370, 310, 310}
05765     , { 370, 310, 370, 310, 310}
05766     , { 280, 280, 280, 280, 280}
05767     , { 280, 150, 150, 150, 280}
05768     , { 280, 280, 280, 280, 280}
05769     }
05770     , {{{ 280, 280, 280, 280, 280}
05771     , { 280, 280, 280, 280, 280}
05772     , { 280, 280, 280, 280, 280}
05773     , { 280, 280, 280, 280, 280}
05774     , { 280, 280, 280, 280, 280}
05775     }
05776     }
05777     }
05778     , {{{ 350, 280, 280, 350, 340}
05779     , { 340, 280, 280, 310, 340}
05780     , { 350, 260, 260, 350, 260}
05781     , { 290, 260, 260, 290, 260}
05782     , { 350, 260, 260, 350, 260}
05783     }
05784     , {{{ 340, 280, 280, 310, 340}
05785     , { 340, 280, 280, 310, 340}
05786     , { 280, 250, 250, 280, 250}
05787     , { 210, 150, 210, 180, 210}
05788     , { 280, 250, 250, 280, 250}
05789     }
05790     , {{{ 350, 260, 260, 350, 260}
05791     , { 290, 260, 260, 290, 260}
05792     , { 350, 260, 260, 350, 260}
05793     , { 290, 260, 260, 290, 260}
05794     , { 350, 260, 260, 350, 260}
05795     }
05796     , {{{ 280, 250, 280, 280, 280}
05797     , { 280, 220, 280, 250, 280}
05798     , { 280, 250, 250, 280, 250}
05799     , { 260, 100, 100, 260, 230}
05800     , { 280, 250, 250, 280, 250}
```

```
05801     }
05802     ,{{ 350, 260, 260, 350, 260}
05803     ,{ 290, 260, 260, 290, 260}
05804     ,{ 350, 260, 260, 350, 260}
05805     ,{ 290, 260, 260, 290, 260}
05806     ,{ 200, 190, 170, 200, 170}
05807     }
05808     }
05809     ,{{{ 350, 240, 280, 350, 280}
05810     ,{ 310, 240, 280, 310, 280}
05811     ,{ 350, 220, 260, 350, 260}
05812     ,{ 290, 230, 260, 290, 260}
05813     ,{ 350, 220, 260, 350, 260}
05814     }
05815     ,{{{ 310, 240, 280, 310, 280}
05816     ,{ 310, 240, 280, 310, 280}
05817     ,{ 280, 220, 250, 280, 250}
05818     ,{ 180, 120, 150, 180, 150}
05819     ,{ 280, 220, 250, 280, 250}
05820     }
05821     ,{{{ 350, 230, 260, 350, 260}
05822     ,{ 290, 230, 260, 290, 260}
05823     ,{ 350, 220, 260, 350, 260}
05824     ,{ 290, 230, 260, 290, 260}
05825     ,{ 350, 220, 260, 350, 260}
05826     }
05827     ,{{{ 280, 220, 250, 280, 250}
05828     ,{ 250, 190, 220, 250, 220}
05829     ,{ 280, 220, 250, 280, 250}
05830     ,{ 260, 70, 100, 260, 100}
05831     ,{ 280, 220, 250, 280, 250}
05832     }
05833     ,{{{ 350, 230, 260, 350, 260}
05834     ,{ 290, 230, 260, 290, 260}
05835     ,{ 350, 220, 260, 350, 260}
05836     ,{ 290, 230, 260, 290, 260}
05837     ,{ 200, 190, 170, 200, 170}
05838     }
05839     }
05840     ,{{{ 280, 280, 280, 280, 280}
05841     ,{ 280, 280, 280, 280, 280}
05842     ,{ 260, 260, 260, 260, 260}
05843     ,{ 260, 260, 260, 260, 260}
05844     ,{ 260, 260, 260, 260, 260}
05845     }
05846     ,{{{ 280, 280, 280, 280, 280}
05847     ,{ 280, 280, 280, 280, 280}
05848     ,{ 250, 250, 250, 250, 250}
05849     ,{ 210, 150, 210, 150, 210}
05850     ,{ 250, 250, 250, 250, 250}
05851     }
05852     ,{{{ 260, 260, 260, 260, 260}
05853     ,{ 260, 260, 260, 260, 260}
05854     ,{ 260, 260, 260, 260, 260}
05855     ,{ 260, 260, 260, 260, 260}
05856     ,{ 260, 260, 260, 260, 260}
05857     }
05858     ,{{{ 280, 250, 280, 250, 280}
05859     ,{ 280, 220, 280, 220, 280}
05860     ,{ 250, 250, 250, 250, 250}
05861     ,{ 100, 100, 100, 100, 100}
05862     ,{ 250, 250, 250, 250, 250}
05863     }
05864     ,{{{ 260, 260, 260, 260, 260}
05865     ,{ 260, 260, 260, 260, 260}
05866     ,{ 260, 260, 260, 260, 260}
05867     ,{ 260, 260, 260, 260, 260}
05868     ,{ 170, 170, 170, 170, 170}
05869     }
05870     }
05871     ,{{{ 280, 180, 280, 230, 280}
05872     ,{ 280, 140, 280, 220, 280}
05873     ,{ 260, 180, 260, 130, 260}
05874     ,{ 260, 130, 260, 230, 260}
05875     ,{ 260, 180, 260, 170, 260}
05876     }
05877     ,{{{ 280, 140, 280, 150, 280}
05878     ,{ 280, 140, 280, 150, 280}
05879     ,{ 250, 120, 250, 120, 250}
05880     ,{ 150, 20, 150, 150, 150}
05881     ,{ 250, 120, 250, 120, 250}
05882     }
05883     ,{{{ 260, 180, 260, 130, 260}
05884     ,{ 260, 130, 260, 130, 260}
05885     ,{ 260, 180, 260, 130, 260}
05886     ,{ 260, 130, 260, 130, 260}
05887     ,{ 260, 180, 260, 130, 260}
```

```
05888     }
05889     ,{{ 250, 120, 250, 230, 250}
05890     ,{ 220, 90, 220, 220, 220}
05891     ,{ 250, 120, 250, 120, 250}
05892     ,{ 230, 100, 100, 230, 100}
05893     ,{ 250, 120, 250, 120, 250}
05894     }
05895     ,{{ 260, 180, 260, 170, 260}
05896     ,{ 260, 130, 260, 130, 260}
05897     ,{ 260, 180, 260, 130, 260}
05898     ,{ 260, 130, 260, 130, 260}
05899     ,{ 170, 30, 170, 170, 170}
05900     }
05901     }
05902     ,{{{ 340, 280, 280, 280, 340}
05903     ,{ 340, 280, 280, 280, 340}
05904     ,{ 260, 260, 260, 260, 260}
05905     ,{ 260, 260, 260, 260, 260}
05906     ,{ 260, 260, 260, 260, 260}
05907     }
05908     ,{{{ 340, 280, 280, 280, 340}
05909     ,{ 340, 280, 280, 280, 340}
05910     ,{ 250, 250, 250, 250, 250}
05911     ,{ 210, 150, 210, 150, 150}
05912     ,{ 250, 250, 250, 250, 250}
05913     }
05914     ,{{{ 260, 260, 260, 260, 260}
05915     ,{ 260, 260, 260, 260, 260}
05916     ,{ 260, 260, 260, 260, 260}
05917     ,{ 260, 260, 260, 260, 260}
05918     ,{ 260, 260, 260, 260, 260}
05919     }
05920     ,{{{ 280, 250, 280, 250, 250}
05921     ,{ 280, 220, 280, 220, 220}
05922     ,{ 250, 250, 250, 250, 250}
05923     ,{ 230, 100, 100, 100, 230}
05924     ,{ 250, 250, 250, 250, 250}
05925     }
05926     ,{{{ 260, 260, 260, 260, 260}
05927     ,{ 260, 260, 260, 260, 260}
05928     ,{ 260, 260, 260, 260, 260}
05929     ,{ 260, 260, 260, 260, 260}
05930     ,{ 170, 170, 170, 170, 170}
05931     }
05932     }
05933     }
05934     ,{{{ 370, 280, 280, 370, 340}
05935     ,{ 340, 280, 280, 310, 340}
05936     ,{ 370, 280, 280, 370, 280}
05937     ,{ 310, 280, 280, 310, 280}
05938     ,{ 370, 280, 280, 370, 280}
05939     }
05940     ,{{{ 340, 280, 280, 310, 340}
05941     ,{ 340, 280, 280, 310, 340}
05942     ,{ 260, 230, 230, 260, 230}
05943     ,{ 230, 170, 230, 200, 230}
05944     ,{ 260, 230, 230, 260, 230}
05945     }
05946     ,{{{ 370, 280, 280, 370, 280}
05947     ,{ 310, 280, 280, 310, 280}
05948     ,{ 370, 280, 280, 370, 280}
05949     ,{ 310, 280, 280, 310, 280}
05950     ,{ 370, 280, 280, 370, 280}
05951     }
05952     ,{{{ 280, 230, 240, 280, 250}
05953     ,{ 240, 180, 240, 210, 240}
05954     ,{ 260, 230, 230, 260, 230}
05955     ,{ 280, 120, 120, 280, 250}
05956     ,{ 260, 230, 230, 260, 230}
05957     }
05958     ,{{{ 340, 280, 280, 340, 280}
05959     ,{ 310, 280, 280, 310, 280}
05960     ,{ 340, 250, 250, 340, 250}
05961     ,{ 310, 280, 280, 310, 280}
05962     ,{ 220, 220, 190, 220, 190}
05963     }
05964     }
05965     ,{{{ 370, 240, 280, 370, 280}
05966     ,{ 310, 240, 280, 310, 280}
05967     ,{ 370, 240, 280, 370, 280}
05968     ,{ 310, 240, 280, 310, 280}
05969     ,{ 370, 240, 280, 370, 280}
05970     }
05971     ,{{{ 310, 240, 280, 310, 280}
05972     ,{ 310, 240, 280, 310, 280}
05973     ,{ 260, 200, 230, 260, 230}
05974     ,{ 200, 140, 170, 200, 170}
```

```
05975      , { 260, 200, 230, 260, 230}
05976      }
05977      , { { 370, 240, 280, 370, 280}
05978      , { 310, 240, 280, 310, 280}
05979      , { 370, 240, 280, 370, 280}
05980      , { 310, 240, 280, 310, 280}
05981      , { 370, 240, 280, 370, 280}
05982      }
05983      , { { 280, 200, 230, 280, 230}
05984      , { 210, 150, 180, 210, 180}
05985      , { 260, 200, 230, 260, 230}
05986      , { 280, 90, 120, 280, 120}
05987      , { 260, 200, 230, 260, 230}
05988      }
05989      , { { 340, 240, 280, 340, 280}
05990      , { 310, 240, 280, 310, 280}
05991      , { 340, 210, 250, 340, 250}
05992      , { 310, 240, 280, 310, 280}
05993      , { 220, 220, 190, 220, 190}
05994      }
05995      }
05996      , { { { 280, 280, 280, 280, 280}
05997      , { 280, 280, 280, 280, 280}
05998      , { 280, 280, 280, 280, 280}
05999      , { 280, 280, 280, 280, 280}
06000      , { 280, 280, 280, 280, 280}
06001      }
06002      , { { 280, 280, 280, 280, 280}
06003      , { 280, 280, 280, 280, 280}
06004      , { 230, 230, 230, 230, 230}
06005      , { 230, 170, 230, 170, 230}
06006      , { 230, 230, 230, 230, 230}
06007      }
06008      , { { 280, 280, 280, 280, 280}
06009      , { 280, 280, 280, 280, 280}
06010      , { 280, 280, 280, 280, 280}
06011      , { 280, 280, 280, 280, 280}
06012      , { 280, 280, 280, 280, 280}
06013      }
06014      , { { 240, 230, 240, 230, 240}
06015      , { 240, 180, 240, 180, 240}
06016      , { 230, 230, 230, 230, 230}
06017      , { 120, 120, 120, 120, 120}
06018      , { 230, 230, 230, 230, 230}
06019      }
06020      , { { 280, 280, 280, 280, 280}
06021      , { 280, 280, 280, 280, 280}
06022      , { 250, 250, 250, 250, 250}
06023      , { 280, 280, 280, 280, 280}
06024      , { 190, 190, 190, 190, 190}
06025      }
06026      }
06027      , { { { 280, 200, 280, 250, 280}
06028      , { 280, 140, 280, 180, 280}
06029      , { 280, 200, 280, 150, 280}
06030      , { 280, 140, 280, 250, 280}
06031      , { 280, 200, 280, 190, 280}
06032      }
06033      , { { 280, 140, 280, 170, 280}
06034      , { 280, 140, 280, 150, 280}
06035      , { 230, 100, 230, 100, 230}
06036      , { 170, 40, 170, 170, 170}
06037      , { 230, 100, 230, 100, 230}
06038      }
06039      , { { 280, 200, 280, 150, 280}
06040      , { 280, 140, 280, 150, 280}
06041      , { 280, 200, 280, 150, 280}
06042      , { 280, 140, 280, 150, 280}
06043      , { 280, 200, 280, 150, 280}
06044      }
06045      , { { 250, 120, 230, 250, 230}
06046      , { 180, 50, 180, 180, 180}
06047      , { 230, 100, 230, 100, 230}
06048      , { 250, 120, 120, 250, 120}
06049      , { 230, 100, 230, 100, 230}
06050      }
06051      , { { 280, 170, 280, 190, 280}
06052      , { 280, 140, 280, 150, 280}
06053      , { 250, 170, 250, 120, 250}
06054      , { 280, 140, 280, 150, 280}
06055      , { 190, 60, 190, 190, 190}
06056      }
06057      }
06058      , { { { 340, 280, 280, 280, 340}
06059      , { 340, 280, 280, 280, 340}
06060      , { 280, 280, 280, 280, 280}
06061      , { 280, 280, 280, 280, 280}
```

```
06062     , { 280, 280, 280, 280, 280}
06063     }
06064     , { { 340, 280, 280, 280, 340}
06065     , { 340, 280, 280, 280, 340}
06066     , { 230, 230, 230, 230, 230}
06067     , { 230, 170, 230, 170, 170}
06068     , { 230, 230, 230, 230, 230}
06069     }
06070     , { { 280, 280, 280, 280, 280}
06071     , { 280, 280, 280, 280, 280}
06072     , { 280, 280, 280, 280, 280}
06073     , { 280, 280, 280, 280, 280}
06074     , { 280, 280, 280, 280, 280}
06075     }
06076     , { { 250, 230, 240, 230, 250}
06077     , { 240, 180, 240, 180, 180}
06078     , { 230, 230, 230, 230, 230}
06079     , { 250, 120, 120, 120, 250}
06080     , { 230, 230, 230, 230, 230}
06081     }
06082     , { { 280, 280, 280, 280, 280}
06083     , { 280, 280, 280, 280, 280}
06084     , { 250, 250, 250, 250, 250}
06085     , { 280, 280, 280, 280, 280}
06086     , { 190, 190, 190, 190, 190}
06087     }
06088     }
06089     }
06090     , { { { 400, 360, 370, 400, 400}
06091     , { 400, 360, 370, 370, 400}
06092     , { 400, 310, 310, 400, 310}
06093     , { 340, 310, 310, 340, 310}
06094     , { 400, 330, 310, 400, 310}
06095     }
06096     , { { 400, 360, 340, 370, 400}
06097     , { 400, 360, 340, 370, 400}
06098     , { 340, 310, 310, 340, 310}
06099     , { 290, 230, 290, 260, 290}
06100     , { 340, 310, 310, 340, 310}
06101     }
06102     , { { 400, 310, 310, 400, 310}
06103     , { 340, 310, 310, 340, 310}
06104     , { 400, 310, 310, 400, 310}
06105     , { 340, 310, 310, 340, 310}
06106     , { 400, 310, 310, 400, 310}
06107     }
06108     , { { 370, 360, 370, 340, 370}
06109     , { 370, 360, 370, 340, 370}
06110     , { 340, 310, 310, 340, 310}
06111     , { 340, 180, 180, 340, 310}
06112     , { 340, 310, 310, 340, 310}
06113     }
06114     , { { 400, 330, 310, 400, 310}
06115     , { 340, 310, 310, 340, 310}
06116     , { 400, 310, 310, 400, 310}
06117     , { 340, 310, 310, 340, 310}
06118     , { 340, 330, 310, 340, 310}
06119     }
06120     }
06121     , { { { 400, 360, 340, 400, 340}
06122     , { 370, 360, 340, 370, 340}
06123     , { 400, 270, 310, 400, 310}
06124     , { 340, 270, 310, 340, 310}
06125     , { 400, 330, 310, 400, 310}
06126     }
06127     , { { 370, 360, 340, 370, 340}
06128     , { 370, 360, 340, 370, 340}
06129     , { 340, 270, 310, 340, 310}
06130     , { 260, 190, 230, 260, 230}
06131     , { 340, 270, 310, 340, 310}
06132     }
06133     , { { 400, 270, 310, 400, 310}
06134     , { 340, 270, 310, 340, 310}
06135     , { 400, 270, 310, 400, 310}
06136     , { 340, 270, 310, 340, 310}
06137     , { 400, 270, 310, 400, 310}
06138     }
06139     , { { 360, 360, 310, 340, 310}
06140     , { 360, 360, 310, 340, 310}
06141     , { 340, 270, 310, 340, 310}
06142     , { 340, 140, 180, 340, 180}
06143     , { 340, 270, 310, 340, 310}
06144     }
06145     , { { 400, 330, 310, 400, 310}
06146     , { 340, 270, 310, 340, 310}
06147     , { 400, 270, 310, 400, 310}
06148     , { 340, 270, 310, 340, 310}
```

```
06149      , { 340, 330, 310, 340, 310}
06150      }
06151      }
06152      ,{{{ 370, 340, 370, 340, 370}
06153      , { 370, 340, 370, 340, 370}
06154      , { 310, 310, 310, 310, 310}
06155      , { 310, 310, 310, 310, 310}
06156      , { 310, 310, 310, 310, 310}
06157      }
06158      ,{{{ 340, 340, 340, 340, 340}
06159      , { 340, 340, 340, 340, 340}
06160      , { 310, 310, 310, 310, 310}
06161      , { 290, 230, 290, 230, 290}
06162      , { 310, 310, 310, 310, 310}
06163      }
06164      ,{{{ 310, 310, 310, 310, 310}
06165      , { 310, 310, 310, 310, 310}
06166      , { 310, 310, 310, 310, 310}
06167      , { 310, 310, 310, 310, 310}
06168      , { 310, 310, 310, 310, 310}
06169      }
06170      ,{{{ 370, 310, 370, 310, 370}
06171      , { 370, 310, 370, 310, 370}
06172      , { 310, 310, 310, 310, 310}
06173      , { 180, 180, 180, 180, 180}
06174      , { 310, 310, 310, 310, 310}
06175      }
06176      ,{{{ 310, 310, 310, 310, 310}
06177      , { 310, 310, 310, 310, 310}
06178      , { 310, 310, 310, 310, 310}
06179      , { 310, 310, 310, 310, 310}
06180      , { 310, 310, 310, 310, 310}
06181      }
06182      }
06183      ,{{{ 340, 230, 340, 310, 340}
06184      , { 340, 220, 340, 310, 340}
06185      , { 310, 230, 310, 180, 310}
06186      , { 310, 170, 310, 310, 310}
06187      , { 310, 230, 310, 310, 310}
06188      }
06189      ,{{{ 340, 220, 340, 230, 340}
06190      , { 340, 220, 340, 210, 340}
06191      , { 310, 170, 310, 180, 310}
06192      , { 230, 40, 230, 230, 230}
06193      , { 310, 170, 310, 180, 310}
06194      }
06195      ,{{{ 310, 230, 310, 180, 310}
06196      , { 310, 170, 310, 180, 310}
06197      , { 310, 230, 310, 180, 310}
06198      , { 310, 170, 310, 180, 310}
06199      , { 310, 230, 310, 180, 310}
06200      }
06201      ,{{{ 310, 170, 310, 310, 310}
06202      , { 310, 170, 310, 310, 310}
06203      , { 310, 170, 310, 180, 310}
06204      , { 310, 170, 180, 310, 180}
06205      , { 310, 170, 310, 180, 310}
06206      }
06207      ,{{{ 310, 230, 310, 310, 310}
06208      , { 310, 170, 310, 180, 310}
06209      , { 310, 230, 310, 180, 310}
06210      , { 310, 170, 310, 180, 310}
06211      , { 310, 170, 310, 310, 310}
06212      }
06213      }
06214      ,{{{ 400, 340, 370, 340, 400}
06215      , { 400, 340, 370, 340, 400}
06216      , { 310, 310, 310, 310, 310}
06217      , { 310, 310, 310, 310, 310}
06218      , { 310, 310, 310, 310, 310}
06219      }
06220      ,{{{ 400, 340, 340, 340, 400}
06221      , { 400, 340, 340, 340, 400}
06222      , { 310, 310, 310, 310, 310}
06223      , { 290, 230, 290, 230, 230}
06224      , { 310, 310, 310, 310, 310}
06225      }
06226      ,{{{ 310, 310, 310, 310, 310}
06227      , { 310, 310, 310, 310, 310}
06228      , { 310, 310, 310, 310, 310}
06229      , { 310, 310, 310, 310, 310}
06230      , { 310, 310, 310, 310, 310}
06231      }
06232      ,{{{ 370, 310, 370, 310, 310}
06233      , { 370, 310, 370, 310, 310}
06234      , { 310, 310, 310, 310, 310}
06235      , { 310, 180, 180, 180, 310}
```



```
06236     , { 310, 310, 310, 310, 310}
06237     }
06238     , {{ 310, 310, 310, 310, 310}
06239     , { 310, 310, 310, 310, 310}
06240     , { 310, 310, 310, 310, 310}
06241     , { 310, 310, 310, 310, 310}
06242     , { 310, 310, 310, 310, 310}
06243     }
06244     }
06245     }
06246     }
06247     , {{{ INF, INF, INF, INF, INF}
06248     , { INF, INF, INF, INF, INF}
06249     , { INF, INF, INF, INF, INF}
06250     , { INF, INF, INF, INF, INF}
06251     , { INF, INF, INF, INF, INF}
06252     }
06253     , {{ INF, INF, INF, INF, INF}
06254     , { INF, INF, INF, INF, INF}
06255     , { INF, INF, INF, INF, INF}
06256     , { INF, INF, INF, INF, INF}
06257     , { INF, INF, INF, INF, INF}
06258     }
06259     , {{ INF, INF, INF, INF, INF}
06260     , { INF, INF, INF, INF, INF}
06261     , { INF, INF, INF, INF, INF}
06262     , { INF, INF, INF, INF, INF}
06263     , { INF, INF, INF, INF, INF}
06264     }
06265     , {{ INF, INF, INF, INF, INF}
06266     , { INF, INF, INF, INF, INF}
06267     , { INF, INF, INF, INF, INF}
06268     , { INF, INF, INF, INF, INF}
06269     , { INF, INF, INF, INF, INF}
06270     }
06271     , {{ INF, INF, INF, INF, INF}
06272     , { INF, INF, INF, INF, INF}
06273     , { INF, INF, INF, INF, INF}
06274     , { INF, INF, INF, INF, INF}
06275     , { INF, INF, INF, INF, INF}
06276     }
06277     }
06278     , {{{ INF, INF, INF, INF, INF}
06279     , { INF, INF, INF, INF, INF}
06280     , { INF, INF, INF, INF, INF}
06281     , { INF, INF, INF, INF, INF}
06282     , { INF, INF, INF, INF, INF}
06283     }
06284     , {{ INF, INF, INF, INF, INF}
06285     , { INF, INF, INF, INF, INF}
06286     , { INF, INF, INF, INF, INF}
06287     , { INF, INF, INF, INF, INF}
06288     , { INF, INF, INF, INF, INF}
06289     }
06290     , {{ INF, INF, INF, INF, INF}
06291     , { INF, INF, INF, INF, INF}
06292     , { INF, INF, INF, INF, INF}
06293     , { INF, INF, INF, INF, INF}
06294     , { INF, INF, INF, INF, INF}
06295     }
06296     , {{ INF, INF, INF, INF, INF}
06297     , { INF, INF, INF, INF, INF}
06298     , { INF, INF, INF, INF, INF}
06299     , { INF, INF, INF, INF, INF}
06300     , { INF, INF, INF, INF, INF}
06301     }
06302     , {{ INF, INF, INF, INF, INF}
06303     , { INF, INF, INF, INF, INF}
06304     , { INF, INF, INF, INF, INF}
06305     , { INF, INF, INF, INF, INF}
06306     , { INF, INF, INF, INF, INF}
06307     }
06308     }
06309     , {{{ INF, INF, INF, INF, INF}
06310     , { INF, INF, INF, INF, INF}
06311     , { INF, INF, INF, INF, INF}
06312     , { INF, INF, INF, INF, INF}
06313     , { INF, INF, INF, INF, INF}
06314     }
06315     , {{ INF, INF, INF, INF, INF}
06316     , { INF, INF, INF, INF, INF}
06317     , { INF, INF, INF, INF, INF}
06318     , { INF, INF, INF, INF, INF}
06319     , { INF, INF, INF, INF, INF}
06320     }
06321     , {{ INF, INF, INF, INF, INF}
06322     , { INF, INF, INF, INF, INF}
```

```

06323      , {   INF,   INF,   INF,   INF,   INF }
06324      , {   INF,   INF,   INF,   INF,   INF }
06325      , {   INF,   INF,   INF,   INF,   INF }
06326      }
06327      , { {   INF,   INF,   INF,   INF,   INF }
06328      , {   INF,   INF,   INF,   INF,   INF }
06329      , {   INF,   INF,   INF,   INF,   INF }
06330      , {   INF,   INF,   INF,   INF,   INF }
06331      , {   INF,   INF,   INF,   INF,   INF }
06332      }
06333      , { {   INF,   INF,   INF,   INF,   INF }
06334      , {   INF,   INF,   INF,   INF,   INF }
06335      , {   INF,   INF,   INF,   INF,   INF }
06336      , {   INF,   INF,   INF,   INF,   INF }
06337      , {   INF,   INF,   INF,   INF,   INF }
06338      }
06339      }
06340      , { { {   INF,   INF,   INF,   INF,   INF }
06341      , {   INF,   INF,   INF,   INF,   INF }
06342      , {   INF,   INF,   INF,   INF,   INF }
06343      , {   INF,   INF,   INF,   INF,   INF }
06344      , {   INF,   INF,   INF,   INF,   INF }
06345      }
06346      , { {   INF,   INF,   INF,   INF,   INF }
06347      , {   INF,   INF,   INF,   INF,   INF }
06348      , {   INF,   INF,   INF,   INF,   INF }
06349      , {   INF,   INF,   INF,   INF,   INF }
06350      , {   INF,   INF,   INF,   INF,   INF }
06351      }
06352      , { {   INF,   INF,   INF,   INF,   INF }
06353      , {   INF,   INF,   INF,   INF,   INF }
06354      , {   INF,   INF,   INF,   INF,   INF }
06355      , {   INF,   INF,   INF,   INF,   INF }
06356      , {   INF,   INF,   INF,   INF,   INF }
06357      }
06358      , { {   INF,   INF,   INF,   INF,   INF }
06359      , {   INF,   INF,   INF,   INF,   INF }
06360      , {   INF,   INF,   INF,   INF,   INF }
06361      , {   INF,   INF,   INF,   INF,   INF }
06362      , {   INF,   INF,   INF,   INF,   INF }
06363      }
06364      , { {   INF,   INF,   INF,   INF,   INF }
06365      , {   INF,   INF,   INF,   INF,   INF }
06366      , {   INF,   INF,   INF,   INF,   INF }
06367      , {   INF,   INF,   INF,   INF,   INF }
06368      , {   INF,   INF,   INF,   INF,   INF }
06369      }
06370      }
06371      , { { {   INF,   INF,   INF,   INF,   INF }
06372      , {   INF,   INF,   INF,   INF,   INF }
06373      , {   INF,   INF,   INF,   INF,   INF }
06374      , {   INF,   INF,   INF,   INF,   INF }
06375      , {   INF,   INF,   INF,   INF,   INF }
06376      }
06377      , { {   INF,   INF,   INF,   INF,   INF }
06378      , {   INF,   INF,   INF,   INF,   INF }
06379      , {   INF,   INF,   INF,   INF,   INF }
06380      , {   INF,   INF,   INF,   INF,   INF }
06381      , {   INF,   INF,   INF,   INF,   INF }
06382      }
06383      , { {   INF,   INF,   INF,   INF,   INF }
06384      , {   INF,   INF,   INF,   INF,   INF }
06385      , {   INF,   INF,   INF,   INF,   INF }
06386      , {   INF,   INF,   INF,   INF,   INF }
06387      , {   INF,   INF,   INF,   INF,   INF }
06388      }
06389      , { {   INF,   INF,   INF,   INF,   INF }
06390      , {   INF,   INF,   INF,   INF,   INF }
06391      , {   INF,   INF,   INF,   INF,   INF }
06392      , {   INF,   INF,   INF,   INF,   INF }
06393      , {   INF,   INF,   INF,   INF,   INF }
06394      }
06395      , { {   INF,   INF,   INF,   INF,   INF }
06396      , {   INF,   INF,   INF,   INF,   INF }
06397      , {   INF,   INF,   INF,   INF,   INF }
06398      , {   INF,   INF,   INF,   INF,   INF }
06399      , {   INF,   INF,   INF,   INF,   INF }
06400      }
06401      }
06402      }
06403      , { { { {   240,   240,   220,   230,   220 }
06404      , {   240,   240,   220,   210,   220 }
06405      , {   230,   220,   210,   230,   210 }
06406      , {   240,   240,   220,   210,   220 }
06407      , {   210,   210,   190,   210,   190 }
06408      }
06409      , { {   200,   200,   180,   170,   180 }

```

```
06410      , { 200, 200, 180, 170, 180}
06411      , { 190, 190, 180, 170, 180}
06412      , { 140, 100, 140, 80, 140}
06413      , { 190, 190, 180, 170, 180}
06414      }
06415      , { { 240, 240, 220, 230, 220}
06416      , { 240, 240, 220, 210, 220}
06417      , { 230, 220, 210, 230, 210}
06418      , { 240, 240, 220, 210, 220}
06419      , { 210, 210, 190, 210, 190}
06420      }
06421      , { { 190, 190, 180, 170, 180}
06422      , { 140, 100, 140, 80, 140}
06423      , { 190, 190, 180, 170, 180}
06424      , { 130, 50, 30, 130, 70}
06425      , { 190, 190, 180, 170, 180}
06426      }
06427      , { { 240, 240, 220, 210, 220}
06428      , { 240, 240, 220, 210, 220}
06429      , { 210, 210, 190, 210, 190}
06430      , { 240, 240, 220, 210, 220}
06431      , { 180, 180, 100, 90, 100}
06432      }
06433      }
06434      , { { { 240, 240, 220, 230, 220}
06435      , { 240, 240, 220, 180, 220}
06436      , { 230, 220, 210, 230, 210}
06437      , { 240, 240, 220, 180, 220}
06438      , { 210, 210, 190, 210, 190}
06439      }
06440      , { { 200, 200, 180, 140, 180}
06441      , { 200, 200, 180, 140, 180}
06442      , { 190, 190, 180, 140, 180}
06443      , { 100, 100, 90, 50, 90}
06444      , { 190, 190, 180, 140, 180}
06445      }
06446      , { { 240, 240, 220, 230, 220}
06447      , { 240, 240, 220, 180, 220}
06448      , { 230, 220, 210, 230, 210}
06449      , { 240, 240, 220, 180, 220}
06450      , { 210, 210, 190, 210, 190}
06451      }
06452      , { { 190, 190, 180, 140, 180}
06453      , { 100, 100, 90, 50, 90}
06454      , { 190, 190, 180, 140, 180}
06455      , { 120, 50, 30, 120, 30}
06456      , { 190, 190, 180, 140, 180}
06457      }
06458      , { { 240, 240, 220, 210, 220}
06459      , { 240, 240, 220, 180, 220}
06460      , { 210, 210, 190, 210, 190}
06461      , { 240, 240, 220, 180, 220}
06462      , { 180, 180, 100, 60, 100}
06463      }
06464      }
06465      , { { { 220, 210, 220, 210, 220}
06466      , { 220, 210, 220, 210, 220}
06467      , { 200, 200, 200, 200, 200}
06468      , { 220, 210, 220, 210, 220}
06469      , { 190, 180, 190, 180, 190}
06470      }
06471      , { { 180, 170, 180, 170, 180}
06472      , { 180, 170, 180, 170, 180}
06473      , { 170, 170, 170, 170, 170}
06474      , { 140, 80, 140, 80, 140}
06475      , { 170, 170, 170, 170, 170}
06476      }
06477      , { { 220, 210, 220, 210, 220}
06478      , { 220, 210, 220, 210, 220}
06479      , { 200, 200, 200, 200, 200}
06480      , { 220, 210, 220, 210, 220}
06481      , { 190, 180, 190, 180, 190}
06482      }
06483      , { { 170, 170, 170, 170, 170}
06484      , { 140, 80, 140, 80, 140}
06485      , { 170, 170, 170, 170, 170}
06486      , { 30, 20, 30, 20, 30}
06487      , { 170, 170, 170, 170, 170}
06488      }
06489      , { { 220, 210, 220, 210, 220}
06490      , { 220, 210, 220, 210, 220}
06491      , { 190, 180, 190, 180, 190}
06492      , { 220, 210, 220, 210, 220}
06493      , { 100, 90, 100, 90, 100}
06494      }
06495      }
06496      , { { { 220, 160, 220, 130, 220}
```

```

06497      , { 220, 110, 220, 60, 220}
06498      , { 210, 160, 210, 50, 210}
06499      , { 220, 110, 220, 130, 220}
06500      , { 190, 140, 190, 70, 190}
06501      }
06502      , { { 180, 70, 180, 60, 180}
06503      , { 180, 70, 180, 20, 180}
06504      , { 180, 70, 180, 20, 180}
06505      , { 90, -20, 90, 60, 90}
06506      , { 180, 70, 180, 20, 180}
06507      }
06508      , { { 220, 160, 220, 60, 220}
06509      , { 220, 110, 220, 60, 220}
06510      , { 210, 160, 210, 50, 210}
06511      , { 220, 110, 220, 60, 220}
06512      , { 190, 140, 190, 30, 190}
06513      }
06514      , { { 180, 70, 180, 130, 180}
06515      , { 90, -20, 90, 60, 90}
06516      , { 180, 70, 180, 20, 180}
06517      , { 130, 50, 30, 130, 30}
06518      , { 180, 70, 180, 20, 180}
06519      }
06520      , { { 220, 140, 220, 70, 220}
06521      , { 220, 110, 220, 60, 220}
06522      , { 190, 140, 190, 30, 190}
06523      , { 220, 110, 220, 60, 220}
06524      , { 100, 0, 100, 70, 100}
06525      }
06526      }
06527      , { { { 220, 210, 220, 210, 150}
06528      , { 220, 210, 220, 210, 150}
06529      , { 200, 200, 200, 200, 110}
06530      , { 220, 210, 220, 210, 130}
06531      , { 190, 180, 190, 180, 100}
06532      }
06533      , { { 180, 170, 180, 170, 150}
06534      , { 180, 170, 180, 170, 150}
06535      , { 170, 170, 170, 170, 80}
06536      , { 140, 80, 140, 80, 0}
06537      , { 170, 170, 170, 170, 80}
06538      }
06539      , { { 220, 210, 220, 210, 130}
06540      , { 220, 210, 220, 210, 130}
06541      , { 200, 200, 200, 200, 110}
06542      , { 220, 210, 220, 210, 130}
06543      , { 190, 180, 190, 180, 100}
06544      }
06545      , { { 170, 170, 170, 170, 80}
06546      , { 140, 80, 140, 80, 0}
06547      , { 170, 170, 170, 170, 80}
06548      , { 70, 20, 30, 20, 70}
06549      , { 170, 170, 170, 170, 80}
06550      }
06551      , { { 220, 210, 220, 210, 130}
06552      , { 220, 210, 220, 210, 130}
06553      , { 190, 180, 190, 180, 100}
06554      , { 220, 210, 220, 210, 130}
06555      , { 100, 90, 100, 90, 10}
06556      }
06557      }
06558      }
06559      , { { { { 210, 210, 200, 200, 200}
06560      , { 210, 210, 200, 190, 200}
06561      , { 200, 190, 180, 200, 180}
06562      , { 180, 180, 170, 160, 170}
06563      , { 190, 190, 170, 190, 170}
06564      }
06565      , { { { 210, 210, 200, 190, 200}
06566      , { 210, 210, 200, 190, 200}
06567      , { 190, 190, 170, 160, 170}
06568      , { 50, 10, 50, -10, 50}
06569      , { 190, 190, 170, 160, 170}
06570      }
06571      , { { { 190, 190, 170, 190, 170}
06572      , { 180, 180, 170, 160, 170}
06573      , { 190, 190, 170, 190, 170}
06574      , { 180, 180, 170, 160, 170}
06575      , { 190, 190, 170, 190, 170}
06576      }
06577      , { { { 190, 190, 170, 160, 170}
06578      , { 110, 70, 110, 50, 110}
06579      , { 190, 190, 170, 160, 170}
06580      , { 130, 50, 30, 130, 70}
06581      , { 190, 190, 170, 160, 170}
06582      }
06583      , { { { 200, 190, 180, 200, 180}

```

```
06584     , { 180, 180, 170, 160, 170}
06585     , { 200, 190, 180, 200, 180}
06586     , { 180, 180, 170, 160, 170}
06587     , { 170, 170, 100, 90, 100}
06588     }
06589     }
06590     , {{{ 210, 210, 200, 200, 200}
06591     , { 210, 210, 200, 160, 200}
06592     , { 200, 190, 180, 200, 180}
06593     , { 180, 180, 170, 130, 170}
06594     , { 190, 190, 170, 190, 170}
06595     }
06596     , {{{ 210, 210, 200, 160, 200}
06597     , { 210, 210, 200, 160, 200}
06598     , { 190, 190, 170, 130, 170}
06599     , { 10, 10, 0, -40, 0}
06600     , { 190, 190, 170, 130, 170}
06601     }
06602     , {{{ 190, 190, 170, 190, 170}
06603     , { 180, 180, 170, 130, 170}
06604     , { 190, 190, 170, 190, 170}
06605     , { 180, 180, 170, 130, 170}
06606     , { 190, 190, 170, 190, 170}
06607     }
06608     , {{{ 190, 190, 170, 130, 170}
06609     , { 70, 70, 60, 20, 60}
06610     , { 190, 190, 170, 130, 170}
06611     , { 120, 50, 30, 120, 30}
06612     , { 190, 190, 170, 130, 170}
06613     }
06614     , {{{ 200, 190, 180, 200, 180}
06615     , { 180, 180, 170, 130, 170}
06616     , { 200, 190, 180, 200, 180}
06617     , { 180, 180, 170, 130, 170}
06618     , { 170, 170, 100, 60, 100}
06619     }
06620     }
06621     , {{{ 190, 190, 190, 190, 190}
06622     , { 190, 190, 190, 190, 190}
06623     , { 170, 170, 170, 170, 170}
06624     , { 160, 160, 160, 160, 160}
06625     , { 170, 160, 170, 160, 170}
06626     }
06627     , {{{ 190, 190, 190, 190, 190}
06628     , { 190, 190, 190, 190, 190}
06629     , { 170, 160, 170, 160, 170}
06630     , { 50, -10, 50, -10, 50}
06631     , { 170, 160, 170, 160, 170}
06632     }
06633     , {{{ 170, 160, 170, 160, 170}
06634     , { 160, 160, 160, 160, 160}
06635     , { 170, 160, 170, 160, 170}
06636     , { 160, 160, 160, 160, 160}
06637     , { 170, 160, 170, 160, 170}
06638     }
06639     , {{{ 170, 160, 170, 160, 170}
06640     , { 110, 50, 110, 50, 110}
06641     , { 170, 160, 170, 160, 170}
06642     , { 30, 20, 30, 20, 30}
06643     , { 170, 160, 170, 160, 170}
06644     }
06645     , {{{ 170, 170, 170, 170, 170}
06646     , { 160, 160, 160, 160, 160}
06647     , { 170, 170, 170, 170, 170}
06648     , { 160, 160, 160, 160, 160}
06649     , { 90, 90, 90, 90, 90}
06650     }
06651     }
06652     , {{{ 200, 130, 200, 130, 200}
06653     , { 200, 90, 200, 40, 200}
06654     , { 180, 130, 180, 20, 180}
06655     , { 170, 60, 170, 130, 170}
06656     , { 170, 120, 170, 70, 170}
06657     }
06658     , {{{ 200, 90, 200, 40, 200}
06659     , { 200, 90, 200, 40, 200}
06660     , { 170, 60, 170, 10, 170}
06661     , { 0, -110, 0, -30, 0}
06662     , { 170, 60, 170, 10, 170}
06663     }
06664     , {{{ 170, 120, 170, 10, 170}
06665     , { 170, 60, 170, 10, 170}
06666     , { 170, 120, 170, 10, 170}
06667     , { 170, 60, 170, 10, 170}
06668     , { 170, 120, 170, 10, 170}
06669     }
06670     , {{{ 170, 60, 170, 130, 170}
```

```
06671      , {      60,    -50,    60,    30,    60}
06672      , {    170,    60,   170,    10,   170}
06673      , {    130,    50,    30,   130,    30}
06674      , {    170,    60,   170,    10,   170}
06675      }
06676      , { {    180,   130,   180,    70,   180}
06677      , {    170,    60,   170,    10,   170}
06678      , {    180,   130,   180,    20,   180}
06679      , {    170,    60,   170,    10,   170}
06680      , {    100,   -10,   100,    70,   100}
06681      }
06682      }
06683      , { { {    190,   190,   190,   190,   160}
06684      , {    190,   190,   190,   190,   160}
06685      , {    170,   170,   170,   170,    80}
06686      , {    160,   160,   160,   160,    70}
06687      , {    170,   160,   170,   160,    80}
06688      }
06689      , { {    190,   190,   190,   190,   160}
06690      , {    190,   190,   190,   190,   160}
06691      , {    170,   160,   170,   160,    80}
06692      , {     50,   -10,    50,   -10,  -100}
06693      , {    170,   160,   170,   160,    80}
06694      }
06695      , { {    170,   160,   170,   160,    80}
06696      , {    160,   160,   160,   160,    70}
06697      , {    170,   160,   170,   160,    80}
06698      , {    160,   160,   160,   160,    70}
06699      , {    170,   160,   170,   160,    80}
06700      }
06701      , { {    170,   160,   170,   160,    80}
06702      , {    110,    50,   110,    50,   -30}
06703      , {    170,   160,   170,   160,    80}
06704      , {     70,    20,    30,    20,    70}
06705      , {    170,   160,   170,   160,    80}
06706      }
06707      , { {    170,   170,   170,   170,    80}
06708      , {    160,   160,   160,   160,    70}
06709      , {    170,   170,   170,   170,    80}
06710      , {    160,   160,   160,   160,    70}
06711      , {     90,    90,    90,    90,     0}
06712      }
06713      }
06714      }
06715      , { { { {    370,   370,   330,   320,   330}
06716      , {    340,   340,   330,   320,   330}
06717      , {    310,   310,   290,   310,   290}
06718      , {    310,   310,   290,   280,   290}
06719      , {    370,   370,   290,   310,   290}
06720      }
06721      , { { {    340,   340,   330,   320,   330}
06722      , {    340,   340,   330,   320,   330}
06723      , {    310,   310,   290,   280,   290}
06724      , {    270,   230,   270,   200,   270}
06725      , {    310,   310,   290,   280,   290}
06726      }
06727      , { { {    310,   310,   290,   310,   290}
06728      , {    310,   310,   290,   280,   290}
06729      , {    310,   310,   290,   310,   290}
06730      , {    310,   310,   290,   280,   290}
06731      , {    310,   310,   290,   310,   290}
06732      }
06733      , { { {    310,   310,   310,   280,   310}
06734      , {    310,   270,   310,   240,   310}
06735      , {    310,   310,   290,   280,   290}
06736      , {    260,   180,   160,   260,   200}
06737      , {    310,   310,   290,   280,   290}
06738      }
06739      , { { {    370,   370,   290,   310,   290}
06740      , {    310,   310,   290,   280,   290}
06741      , {    310,   310,   290,   310,   290}
06742      , {    310,   310,   290,   280,   290}
06743      , {    370,   370,   290,   280,   290}
06744      }
06745      }
06746      , { { { {    370,   370,   330,   310,   330}
06747      , {    340,   340,   330,   290,   330}
06748      , {    310,   310,   290,   310,   290}
06749      , {    310,   310,   290,   250,   290}
06750      , {    370,   370,   290,   310,   290}
06751      }
06752      , { { {    340,   340,   330,   290,   330}
06753      , {    340,   340,   330,   290,   330}
06754      , {    310,   310,   290,   250,   290}
06755      , {    230,   230,   210,   170,   210}
06756      , {    310,   310,   290,   250,   290}
06757      }
}
```

```
06758 ,{{ 310, 310, 290, 310, 290}
06759 ,{ 310, 310, 290, 250, 290}
06760 ,{ 310, 310, 290, 310, 290}
06761 ,{ 310, 310, 290, 250, 290}
06762 ,{ 310, 310, 290, 310, 290}
06763 }
06764 ,{{ 310, 310, 290, 250, 290}
06765 ,{ 270, 270, 250, 210, 250}
06766 ,{ 310, 310, 290, 250, 290}
06767 ,{ 250, 180, 160, 250, 160}
06768 ,{ 310, 310, 290, 250, 290}
06769 }
06770 ,{{ 370, 370, 290, 310, 290}
06771 ,{ 310, 310, 290, 250, 290}
06772 ,{ 310, 310, 290, 310, 290}
06773 ,{ 310, 310, 290, 250, 290}
06774 ,{ 370, 370, 290, 250, 290}
06775 }
06776 }
06777 ,{{{ 320, 320, 320, 320, 320}
06778 ,{ 320, 320, 320, 320, 320}
06779 ,{ 290, 280, 290, 280, 290}
06780 ,{ 290, 280, 290, 280, 290}
06781 ,{ 290, 280, 290, 280, 290}
06782 }
06783 ,{{{ 320, 320, 320, 320, 320}
06784 ,{ 320, 320, 320, 320, 320}
06785 ,{ 290, 280, 290, 280, 290}
06786 ,{ 270, 200, 270, 200, 270}
06787 ,{ 290, 280, 290, 280, 290}
06788 }
06789 ,{{{ 290, 280, 290, 280, 290}
06790 ,{ 290, 280, 290, 280, 290}
06791 ,{ 290, 280, 290, 280, 290}
06792 ,{ 290, 280, 290, 280, 290}
06793 ,{ 290, 280, 290, 280, 290}
06794 }
06795 ,{{{ 310, 280, 310, 280, 310}
06796 ,{ 310, 240, 310, 240, 310}
06797 ,{ 290, 280, 290, 280, 290}
06798 ,{ 160, 150, 160, 150, 160}
06799 ,{ 290, 280, 290, 280, 290}
06800 }
06801 ,{{{ 290, 280, 290, 280, 290}
06802 ,{ 290, 280, 290, 280, 290}
06803 ,{ 290, 280, 290, 280, 290}
06804 ,{ 290, 280, 290, 280, 290}
06805 ,{ 290, 280, 290, 280, 290}
06806 }
06807 }
06808 ,{{{ 330, 240, 330, 260, 330}
06809 ,{ 330, 220, 330, 220, 330}
06810 ,{ 290, 240, 290, 130, 290}
06811 ,{ 290, 180, 290, 260, 290}
06812 ,{ 290, 240, 290, 260, 290}
06813 }
06814 ,{{{ 330, 220, 330, 180, 330}
06815 ,{ 330, 220, 330, 170, 330}
06816 ,{ 290, 180, 290, 130, 290}
06817 ,{ 210, 100, 210, 180, 210}
06818 ,{ 290, 180, 290, 130, 290}
06819 }
06820 ,{{{ 290, 240, 290, 130, 290}
06821 ,{ 290, 180, 290, 130, 290}
06822 ,{ 290, 240, 290, 130, 290}
06823 ,{ 290, 180, 290, 130, 290}
06824 ,{ 290, 240, 290, 130, 290}
06825 }
06826 ,{{{ 290, 180, 290, 260, 290}
06827 ,{ 250, 140, 250, 220, 250}
06828 ,{ 290, 180, 290, 130, 290}
06829 ,{ 260, 180, 160, 260, 160}
06830 ,{ 290, 180, 290, 130, 290}
06831 }
06832 ,{{{ 290, 240, 290, 260, 290}
06833 ,{ 290, 180, 290, 130, 290}
06834 ,{ 290, 240, 290, 130, 290}
06835 ,{ 290, 180, 290, 130, 290}
06836 ,{ 290, 180, 290, 260, 290}
06837 }
06838 }
06839 ,{{{ 320, 320, 320, 320, 290}
06840 ,{ 320, 320, 320, 320, 290}
06841 ,{ 290, 280, 290, 280, 200}
06842 ,{ 290, 280, 290, 280, 200}
06843 ,{ 290, 280, 290, 280, 200}
06844 }
```

```
06845 ,{{ 320, 320, 320, 320, 290}
06846 ,{ 320, 320, 320, 320, 290}
06847 ,{ 290, 280, 290, 280, 200}
06848 ,{ 270, 200, 270, 200, 120}
06849 ,{ 290, 280, 290, 280, 200}
06850 }
06851 ,{{ 290, 280, 290, 280, 200}
06852 ,{ 290, 280, 290, 280, 200}
06853 ,{ 290, 280, 290, 280, 200}
06854 ,{ 290, 280, 290, 280, 200}
06855 ,{ 290, 280, 290, 280, 200}
06856 }
06857 ,{{ 310, 280, 310, 280, 200}
06858 ,{ 310, 240, 310, 240, 160}
06859 ,{ 290, 280, 290, 280, 200}
06860 ,{ 200, 150, 160, 150, 200}
06861 ,{ 290, 280, 290, 280, 200}
06862 }
06863 ,{{ 290, 280, 290, 280, 200}
06864 ,{ 290, 280, 290, 280, 200}
06865 ,{ 290, 280, 290, 280, 200}
06866 ,{ 290, 280, 290, 280, 200}
06867 ,{ 290, 280, 290, 280, 200}
06868 }
06869 }
06870 }
06871 ,{{{ 350, 340, 350, 280, 350}
06872 ,{ 350, 310, 350, 280, 350}
06873 ,{ 280, 280, 260, 280, 260}
06874 ,{ 280, 280, 260, 250, 260}
06875 ,{ 340, 340, 260, 280, 260}
06876 }
06877 ,{{ 280, 280, 260, 250, 260}
06878 ,{ 240, 240, 230, 220, 230}
06879 ,{ 280, 280, 260, 250, 260}
06880 ,{ 180, 140, 180, 120, 180}
06881 ,{ 280, 280, 260, 250, 260}
06882 }
06883 ,{{ 280, 280, 260, 280, 260}
06884 ,{ 280, 280, 260, 250, 260}
06885 ,{ 280, 280, 260, 280, 260}
06886 ,{ 280, 280, 260, 250, 260}
06887 ,{ 280, 280, 260, 280, 260}
06888 }
06889 ,{{ 350, 310, 350, 280, 350}
06890 ,{ 350, 310, 350, 280, 350}
06891 ,{ 280, 280, 260, 250, 260}
06892 ,{ 230, 150, 130, 230, 170}
06893 ,{ 280, 280, 260, 250, 260}
06894 }
06895 ,{{ 340, 340, 260, 280, 260}
06896 ,{ 280, 280, 260, 250, 260}
06897 ,{ 280, 280, 260, 280, 260}
06898 ,{ 280, 280, 260, 250, 260}
06899 ,{ 340, 340, 260, 250, 260}
06900 }
06901 }
06902 ,{{{ 340, 340, 290, 280, 290}
06903 ,{ 310, 310, 290, 250, 290}
06904 ,{ 280, 280, 260, 280, 260}
06905 ,{ 280, 280, 260, 220, 260}
06906 ,{ 340, 340, 260, 280, 260}
06907 }
06908 ,{{ 280, 280, 260, 220, 260}
06909 ,{ 240, 240, 230, 190, 230}
06910 ,{ 280, 280, 260, 220, 260}
06911 ,{ 140, 140, 130, 90, 130}
06912 ,{ 280, 280, 260, 220, 260}
06913 }
06914 ,{{ 280, 280, 260, 280, 260}
06915 ,{ 280, 280, 260, 220, 260}
06916 ,{ 280, 280, 260, 280, 260}
06917 ,{ 280, 280, 260, 220, 260}
06918 ,{ 280, 280, 260, 280, 260}
06919 }
06920 ,{{ 310, 310, 290, 250, 290}
06921 ,{ 310, 310, 290, 250, 290}
06922 ,{ 280, 280, 260, 220, 260}
06923 ,{ 220, 150, 130, 220, 130}
06924 ,{ 280, 280, 260, 220, 260}
06925 }
06926 ,{{ 340, 340, 260, 280, 260}
06927 ,{ 280, 280, 260, 220, 260}
06928 ,{ 280, 280, 260, 280, 260}
06929 ,{ 280, 280, 260, 220, 260}
06930 ,{ 340, 340, 260, 220, 260}
06931 }
```



```
06932     }
06933     , {{ { 350, 280, 350, 280, 350}
06934     , { 350, 280, 350, 280, 350}
06935     , { 260, 250, 260, 250, 260}
06936     , { 260, 250, 260, 250, 260}
06937     , { 260, 250, 260, 250, 260}
06938     }
06939     , {{ { 260, 250, 260, 250, 260}
06940     , { 220, 220, 220, 220, 220}
06941     , { 260, 250, 260, 250, 260}
06942     , { 180, 120, 180, 120, 180}
06943     , { 260, 250, 260, 250, 260}
06944     }
06945     , {{ { 260, 250, 260, 250, 260}
06946     , { 260, 250, 260, 250, 260}
06947     , { 260, 250, 260, 250, 260}
06948     , { 260, 250, 260, 250, 260}
06949     , { 260, 250, 260, 250, 260}
06950     }
06951     , {{ { 350, 280, 350, 280, 350}
06952     , { 350, 280, 350, 280, 350}
06953     , { 260, 250, 260, 250, 260}
06954     , { 130, 120, 130, 120, 130}
06955     , { 260, 250, 260, 250, 260}
06956     }
06957     , {{ { 260, 250, 260, 250, 260}
06958     , { 260, 250, 260, 250, 260}
06959     , { 260, 250, 260, 250, 260}
06960     , { 260, 250, 260, 250, 260}
06961     , { 260, 250, 260, 250, 260}
06962     }
06963     }
06964     , {{ { 290, 210, 290, 260, 290}
06965     , { 290, 180, 290, 260, 290}
06966     , { 260, 210, 260, 100, 260}
06967     , { 260, 150, 260, 230, 260}
06968     , { 260, 210, 260, 230, 260}
06969     }
06970     , {{ { 260, 150, 260, 100, 260}
06971     , { 230, 120, 230, 70, 230}
06972     , { 260, 150, 260, 100, 260}
06973     , { 130, 20, 130, 100, 130}
06974     , { 260, 150, 260, 100, 260}
06975     }
06976     , {{ { 260, 210, 260, 100, 260}
06977     , { 260, 150, 260, 100, 260}
06978     , { 260, 210, 260, 100, 260}
06979     , { 260, 150, 260, 100, 260}
06980     , { 260, 210, 260, 100, 260}
06981     }
06982     , {{ { 290, 180, 290, 260, 290}
06983     , { 290, 180, 290, 260, 290}
06984     , { 260, 150, 260, 100, 260}
06985     , { 230, 150, 130, 230, 130}
06986     , { 260, 150, 260, 100, 260}
06987     }
06988     , {{ { 260, 210, 260, 230, 260}
06989     , { 260, 150, 260, 100, 260}
06990     , { 260, 210, 260, 100, 260}
06991     , { 260, 150, 260, 100, 260}
06992     , { 260, 150, 260, 230, 260}
06993     }
06994     }
06995     , {{ { 350, 280, 350, 280, 200}
06996     , { 350, 280, 350, 280, 200}
06997     , { 260, 250, 260, 250, 170}
06998     , { 260, 250, 260, 250, 170}
06999     , { 260, 250, 260, 250, 170}
07000     }
07001     , {{ { 260, 250, 260, 250, 190}
07002     , { 220, 220, 220, 220, 190}
07003     , { 260, 250, 260, 250, 170}
07004     , { 180, 120, 180, 120, 30}
07005     , { 260, 250, 260, 250, 170}
07006     }
07007     , {{ { 260, 250, 260, 250, 170}
07008     , { 260, 250, 260, 250, 170}
07009     , { 260, 250, 260, 250, 170}
07010     , { 260, 250, 260, 250, 170}
07011     , { 260, 250, 260, 250, 170}
07012     }
07013     , {{ { 350, 280, 350, 280, 200}
07014     , { 350, 280, 350, 280, 200}
07015     , { 260, 250, 260, 250, 170}
07016     , { 170, 120, 130, 120, 170}
07017     , { 260, 250, 260, 250, 170}
07018     }
```

```
07019 ,{{ 260, 250, 260, 250, 170}
07020 ,{ 260, 250, 260, 250, 170}
07021 ,{ 260, 250, 260, 250, 170}
07022 ,{ 260, 250, 260, 250, 170}
07023 ,{ 260, 250, 260, 250, 170}
07024 }
07025 }
07026 }
07027 ,{{{ 280, 280, 260, 260, 260}
07028 ,{ 280, 280, 260, 250, 260}
07029 ,{ 260, 260, 240, 260, 240}
07030 ,{ 260, 260, 250, 240, 250}
07031 ,{ 260, 260, 240, 260, 240}
07032 }
07033 ,{{ 280, 280, 260, 250, 260}
07034 ,{ 280, 280, 260, 250, 260}
07035 ,{ 250, 250, 240, 230, 240}
07036 ,{ 190, 150, 190, 130, 190}
07037 ,{ 250, 250, 240, 230, 240}
07038 }
07039 ,{{ 260, 260, 250, 260, 250}
07040 ,{ 260, 260, 250, 240, 250}
07041 ,{ 260, 260, 240, 260, 240}
07042 ,{ 260, 260, 250, 240, 250}
07043 ,{ 260, 260, 240, 260, 240}
07044 }
07045 ,{{ 260, 250, 260, 230, 260}
07046 ,{ 260, 220, 260, 200, 260}
07047 ,{ 250, 250, 240, 230, 240}
07048 ,{ 190, 110, 90, 190, 120}
07049 ,{ 250, 250, 240, 230, 240}
07050 }
07051 ,{{ 260, 260, 250, 260, 250}
07052 ,{ 260, 260, 250, 240, 250}
07053 ,{ 260, 260, 240, 260, 240}
07054 ,{ 260, 260, 250, 240, 250}
07055 ,{ 230, 230, 150, 140, 150}
07056 }
07057 }
07058 ,{{{ 280, 280, 260, 260, 260}
07059 ,{ 280, 280, 260, 220, 260}
07060 ,{ 260, 260, 240, 260, 240}
07061 ,{ 260, 260, 250, 210, 250}
07062 ,{ 260, 260, 240, 260, 240}
07063 }
07064 ,{{ 280, 280, 260, 220, 260}
07065 ,{ 280, 280, 260, 220, 260}
07066 ,{ 250, 250, 240, 200, 240}
07067 ,{ 150, 150, 140, 100, 140}
07068 ,{ 250, 250, 240, 200, 240}
07069 }
07070 ,{{ 260, 260, 250, 260, 250}
07071 ,{ 260, 260, 250, 210, 250}
07072 ,{ 260, 260, 240, 260, 240}
07073 ,{ 260, 260, 250, 210, 250}
07074 ,{ 260, 260, 240, 260, 240}
07075 }
07076 ,{{ 250, 250, 240, 200, 240}
07077 ,{ 220, 220, 210, 170, 210}
07078 ,{ 250, 250, 240, 200, 240}
07079 ,{ 180, 100, 90, 180, 90}
07080 ,{ 250, 250, 240, 200, 240}
07081 }
07082 ,{{ 260, 260, 250, 260, 250}
07083 ,{ 260, 260, 250, 210, 250}
07084 ,{ 260, 260, 240, 260, 240}
07085 ,{ 260, 260, 250, 210, 250}
07086 ,{ 230, 230, 150, 110, 150}
07087 }
07088 }
07089 ,{{{ 260, 250, 260, 250, 260}
07090 ,{ 260, 250, 260, 250, 260}
07091 ,{ 240, 230, 240, 230, 240}
07092 ,{ 240, 240, 240, 240, 240}
07093 ,{ 240, 230, 240, 230, 240}
07094 }
07095 ,{{ 260, 250, 260, 250, 260}
07096 ,{ 260, 250, 260, 250, 260}
07097 ,{ 230, 230, 230, 230, 230}
07098 ,{ 190, 130, 190, 130, 190}
07099 ,{ 230, 230, 230, 230, 230}
07100 }
07101 ,{{ 240, 240, 240, 240, 240}
07102 ,{ 240, 240, 240, 240, 240}
07103 ,{ 240, 230, 240, 230, 240}
07104 ,{ 240, 240, 240, 240, 240}
07105 ,{ 240, 230, 240, 230, 240}
```

```
07106     }
07107     ,{{ 260, 230, 260, 230, 260}
07108     ,{ 260, 200, 260, 200, 260}
07109     ,{ 230, 230, 230, 230, 230}
07110     ,{ 80, 80, 80, 80, 80}
07111     ,{ 230, 230, 230, 230, 230}
07112     }
07113     ,{{ 240, 240, 240, 240, 240}
07114     ,{ 240, 240, 240, 240, 240}
07115     ,{ 240, 230, 240, 230, 240}
07116     ,{ 240, 240, 240, 240, 240}
07117     ,{ 150, 140, 150, 140, 150}
07118     }
07119     }
07120     ,{{{ 260, 190, 260, 190, 260}
07121     ,{ 260, 150, 260, 180, 260}
07122     ,{ 240, 190, 240, 80, 240}
07123     ,{ 250, 140, 250, 190, 250}
07124     ,{ 240, 190, 240, 120, 240}
07125     }
07126     ,{{{ 260, 150, 260, 110, 260}
07127     ,{ 260, 150, 260, 100, 260}
07128     ,{ 240, 130, 240, 80, 240}
07129     ,{ 140, 30, 140, 110, 140}
07130     ,{ 240, 130, 240, 80, 240}
07131     }
07132     ,{{{ 250, 190, 250, 90, 250}
07133     ,{ 250, 140, 250, 90, 250}
07134     ,{ 240, 190, 240, 80, 240}
07135     ,{ 250, 140, 250, 90, 250}
07136     ,{ 240, 190, 240, 80, 240}
07137     }
07138     ,{{{ 240, 130, 240, 190, 240}
07139     ,{ 210, 100, 210, 180, 210}
07140     ,{ 240, 130, 240, 80, 240}
07141     ,{ 190, 110, 90, 190, 90}
07142     ,{ 240, 130, 240, 80, 240}
07143     }
07144     ,{{{ 250, 190, 250, 120, 250}
07145     ,{ 250, 140, 250, 90, 250}
07146     ,{ 240, 190, 240, 80, 240}
07147     ,{ 250, 140, 250, 90, 250}
07148     ,{ 150, 40, 150, 120, 150}
07149     }
07150     }
07151     ,{{{ 260, 250, 260, 250, 230}
07152     ,{ 260, 250, 260, 250, 230}
07153     ,{ 240, 230, 240, 230, 150}
07154     ,{ 240, 240, 240, 240, 150}
07155     ,{ 240, 230, 240, 230, 150}
07156     }
07157     ,{{{ 260, 250, 260, 250, 230}
07158     ,{ 260, 250, 260, 250, 230}
07159     ,{ 230, 230, 230, 230, 140}
07160     ,{ 190, 130, 190, 130, 40}
07161     ,{ 230, 230, 230, 230, 140}
07162     }
07163     ,{{{ 240, 240, 240, 240, 150}
07164     ,{ 240, 240, 240, 240, 150}
07165     ,{ 240, 230, 240, 230, 150}
07166     ,{ 240, 240, 240, 240, 150}
07167     ,{ 240, 230, 240, 230, 150}
07168     }
07169     ,{{{ 260, 230, 260, 230, 140}
07170     ,{ 260, 200, 260, 200, 110}
07171     ,{ 230, 230, 230, 230, 140}
07172     ,{ 120, 80, 80, 80, 120}
07173     ,{ 230, 230, 230, 230, 140}
07174     }
07175     ,{{{ 240, 240, 240, 240, 150}
07176     ,{ 240, 240, 240, 240, 150}
07177     ,{ 240, 230, 240, 230, 150}
07178     ,{ 240, 240, 240, 240, 150}
07179     ,{ 150, 140, 150, 140, 60}
07180     }
07181     }
07182     }
07183     ,{{{ 280, 280, 260, 280, 260}
07184     ,{ 280, 280, 260, 250, 260}
07185     ,{ 280, 280, 260, 280, 260}
07186     ,{ 280, 280, 260, 250, 260}
07187     ,{ 280, 280, 260, 280, 260}
07188     }
07189     ,{{{ 280, 280, 260, 250, 260}
07190     ,{ 280, 280, 260, 250, 260}
07191     ,{ 230, 230, 220, 210, 220}
07192     ,{ 210, 170, 210, 150, 210}
```

```
07193      , { 230, 230, 220, 210, 220}
07194      }
07195      , { { 280, 280, 260, 280, 260}
07196      , { 280, 280, 260, 250, 260}
07197      , { 280, 280, 260, 280, 260}
07198      , { 280, 280, 260, 250, 260}
07199      , { 280, 280, 260, 280, 260}
07200      }
07201      , { { 230, 230, 220, 210, 220}
07202      , { 220, 180, 220, 160, 220}
07203      , { 230, 230, 220, 210, 220}
07204      , { 210, 130, 110, 210, 140}
07205      , { 230, 230, 220, 210, 220}
07206      }
07207      , { { 280, 280, 260, 250, 260}
07208      , { 280, 280, 260, 250, 260}
07209      , { 250, 250, 230, 250, 230}
07210      , { 280, 280, 260, 250, 260}
07211      , { 250, 250, 180, 170, 180}
07212      }
07213      }
07214      , { { { 280, 280, 260, 280, 260}
07215      , { 280, 280, 260, 220, 260}
07216      , { 280, 280, 260, 280, 260}
07217      , { 280, 280, 260, 220, 260}
07218      , { 280, 280, 260, 280, 260}
07219      }
07220      , { { 280, 280, 260, 220, 260}
07221      , { 280, 280, 260, 220, 260}
07222      , { 230, 230, 220, 180, 220}
07223      , { 170, 170, 160, 120, 160}
07224      , { 230, 230, 220, 180, 220}
07225      }
07226      , { { 280, 280, 260, 280, 260}
07227      , { 280, 280, 260, 220, 260}
07228      , { 280, 280, 260, 280, 260}
07229      , { 280, 280, 260, 220, 260}
07230      , { 280, 280, 260, 280, 260}
07231      }
07232      , { { 230, 230, 220, 200, 220}
07233      , { 180, 180, 170, 130, 170}
07234      , { 230, 230, 220, 180, 220}
07235      , { 200, 120, 110, 200, 110}
07236      , { 230, 230, 220, 180, 220}
07237      }
07238      , { { 280, 280, 260, 250, 260}
07239      , { 280, 280, 260, 220, 260}
07240      , { 250, 250, 230, 250, 230}
07241      , { 280, 280, 260, 220, 260}
07242      , { 250, 250, 180, 140, 180}
07243      }
07244      }
07245      , { { { 260, 250, 260, 250, 260}
07246      , { 260, 250, 260, 250, 260}
07247      , { 260, 250, 260, 250, 260}
07248      , { 260, 250, 260, 250, 260}
07249      , { 260, 250, 260, 250, 260}
07250      }
07251      , { { 260, 250, 260, 250, 260}
07252      , { 260, 250, 260, 250, 260}
07253      , { 210, 210, 210, 210, 210}
07254      , { 210, 150, 210, 150, 210}
07255      , { 210, 210, 210, 210, 210}
07256      }
07257      , { { 260, 250, 260, 250, 260}
07258      , { 260, 250, 260, 250, 260}
07259      , { 260, 250, 260, 250, 260}
07260      , { 260, 250, 260, 250, 260}
07261      , { 260, 250, 260, 250, 260}
07262      }
07263      , { { 220, 210, 220, 210, 220}
07264      , { 220, 160, 220, 160, 220}
07265      , { 210, 210, 210, 210, 210}
07266      , { 100, 100, 100, 100, 100}
07267      , { 210, 210, 210, 210, 210}
07268      }
07269      , { { 260, 250, 260, 250, 260}
07270      , { 260, 250, 260, 250, 260}
07271      , { 230, 220, 230, 220, 230}
07272      , { 260, 250, 260, 250, 260}
07273      , { 170, 170, 170, 170, 170}
07274      }
07275      }
07276      , { { { 260, 210, 260, 210, 260}
07277      , { 260, 150, 260, 140, 260}
07278      , { 260, 210, 260, 100, 260}
07279      , { 260, 150, 260, 210, 260}
```

```

07280     , { 260, 210, 260, 150, 260}
07281     }
07282     , {{ 260, 150, 260, 130, 260}
07283     , { 260, 150, 260, 100, 260}
07284     , { 220, 110, 220, 60, 220}
07285     , { 160, 50, 160, 130, 160}
07286     , { 220, 110, 220, 60, 220}
07287     }
07288     , {{ 260, 210, 260, 100, 260}
07289     , { 260, 150, 260, 100, 260}
07290     , { 260, 210, 260, 100, 260}
07291     , { 260, 150, 260, 100, 260}
07292     , { 260, 210, 260, 100, 260}
07293     }
07294     , {{ 220, 130, 220, 210, 220}
07295     , { 170, 60, 170, 140, 170}
07296     , { 220, 110, 220, 60, 220}
07297     , { 210, 130, 110, 210, 110}
07298     , { 220, 110, 220, 60, 220}
07299     }
07300     , {{ 260, 180, 260, 150, 260}
07301     , { 260, 150, 260, 100, 260}
07302     , { 230, 180, 230, 70, 230}
07303     , { 260, 150, 260, 100, 260}
07304     , { 180, 70, 180, 150, 180}
07305     }
07306     }
07307     , {{{ 260, 250, 260, 250, 230}
07308     , { 260, 250, 260, 250, 230}
07309     , { 260, 250, 260, 250, 170}
07310     , { 260, 250, 260, 250, 170}
07311     , { 260, 250, 260, 250, 170}
07312     }
07313     , {{ 260, 250, 260, 250, 230}
07314     , { 260, 250, 260, 250, 230}
07315     , { 210, 210, 210, 210, 120}
07316     , { 210, 150, 210, 150, 60}
07317     , { 210, 210, 210, 210, 120}
07318     }
07319     , {{ 260, 250, 260, 250, 170}
07320     , { 260, 250, 260, 250, 170}
07321     , { 260, 250, 260, 250, 170}
07322     , { 260, 250, 260, 250, 170}
07323     , { 260, 250, 260, 250, 170}
07324     }
07325     , {{ 220, 210, 220, 210, 140}
07326     , { 220, 160, 220, 160, 70}
07327     , { 210, 210, 210, 210, 120}
07328     , { 140, 100, 100, 100, 140}
07329     , { 210, 210, 210, 210, 120}
07330     }
07331     , {{ 260, 250, 260, 250, 170}
07332     , { 260, 250, 260, 250, 170}
07333     , { 230, 220, 230, 220, 140}
07334     , { 260, 250, 260, 250, 170}
07335     , { 170, 170, 170, 170, 80}
07336     }
07337     }
07338     }
07339     , {{{ 370, 370, 350, 320, 350}
07340     , { 350, 340, 350, 320, 350}
07341     , { 310, 310, 290, 310, 290}
07342     , { 310, 310, 290, 280, 290}
07343     , { 370, 370, 290, 310, 290}
07344     }
07345     , {{ 340, 340, 330, 320, 330}
07346     , { 340, 340, 330, 320, 330}
07347     , { 310, 310, 290, 280, 290}
07348     , { 270, 230, 270, 200, 270}
07349     , { 310, 310, 290, 280, 290}
07350     }
07351     , {{ 310, 310, 290, 310, 290}
07352     , { 310, 310, 290, 280, 290}
07353     , { 310, 310, 290, 310, 290}
07354     , { 310, 310, 290, 280, 290}
07355     , { 310, 310, 290, 310, 290}
07356     }
07357     , {{ 350, 310, 350, 280, 350}
07358     , { 350, 310, 350, 280, 350}
07359     , { 310, 310, 290, 280, 290}
07360     , { 260, 180, 160, 260, 200}
07361     , { 310, 310, 290, 280, 290}
07362     }
07363     , {{ 370, 370, 290, 310, 290}
07364     , { 310, 310, 290, 280, 290}
07365     , { 310, 310, 290, 310, 290}
07366     , { 310, 310, 290, 280, 290}

```

```
07367      , { 370, 370, 290, 280, 290}
07368      }
07369      }
07370      , {{{ 370, 370, 330, 310, 330}
07371      , { 340, 340, 330, 290, 330}
07372      , { 310, 310, 290, 310, 290}
07373      , { 310, 310, 290, 250, 290}
07374      , { 370, 370, 290, 310, 290}
07375      }
07376      , {{{ 340, 340, 330, 290, 330}
07377      , { 340, 340, 330, 290, 330}
07378      , { 310, 310, 290, 250, 290}
07379      , { 230, 230, 210, 170, 210}
07380      , { 310, 310, 290, 250, 290}
07381      }
07382      , {{{ 310, 310, 290, 310, 290}
07383      , { 310, 310, 290, 250, 290}
07384      , { 310, 310, 290, 310, 290}
07385      , { 310, 310, 290, 250, 290}
07386      , { 310, 310, 290, 310, 290}
07387      }
07388      , {{{ 310, 310, 290, 250, 290}
07389      , { 310, 310, 290, 250, 290}
07390      , { 310, 310, 290, 250, 290}
07391      , { 250, 180, 160, 250, 160}
07392      , { 310, 310, 290, 250, 290}
07393      }
07394      , {{{ 370, 370, 290, 310, 290}
07395      , { 310, 310, 290, 250, 290}
07396      , { 310, 310, 290, 310, 290}
07397      , { 310, 310, 290, 250, 290}
07398      , { 370, 370, 290, 250, 290}
07399      }
07400      }
07401      , {{{ 350, 320, 350, 320, 350}
07402      , { 350, 320, 350, 320, 350}
07403      , { 290, 280, 290, 280, 290}
07404      , { 290, 280, 290, 280, 290}
07405      , { 290, 280, 290, 280, 290}
07406      }
07407      , {{{ 320, 320, 320, 320, 320}
07408      , { 320, 320, 320, 320, 320}
07409      , { 290, 280, 290, 280, 290}
07410      , { 270, 200, 270, 200, 270}
07411      , { 290, 280, 290, 280, 290}
07412      }
07413      , {{{ 290, 280, 290, 280, 290}
07414      , { 290, 280, 290, 280, 290}
07415      , { 290, 280, 290, 280, 290}
07416      , { 290, 280, 290, 280, 290}
07417      , { 290, 280, 290, 280, 290}
07418      }
07419      , {{{ 350, 280, 350, 280, 350}
07420      , { 350, 280, 350, 280, 350}
07421      , { 290, 280, 290, 280, 290}
07422      , { 160, 150, 160, 150, 160}
07423      , { 290, 280, 290, 280, 290}
07424      }
07425      , {{{ 290, 280, 290, 280, 290}
07426      , { 290, 280, 290, 280, 290}
07427      , { 290, 280, 290, 280, 290}
07428      , { 290, 280, 290, 280, 290}
07429      , { 290, 280, 290, 280, 290}
07430      }
07431      }
07432      , {{{ 330, 240, 330, 260, 330}
07433      , { 330, 220, 330, 260, 330}
07434      , { 290, 240, 290, 130, 290}
07435      , { 290, 180, 290, 260, 290}
07436      , { 290, 240, 290, 260, 290}
07437      }
07438      , {{{ 330, 220, 330, 180, 330}
07439      , { 330, 220, 330, 170, 330}
07440      , { 290, 180, 290, 130, 290}
07441      , { 210, 100, 210, 180, 210}
07442      , { 290, 180, 290, 130, 290}
07443      }
07444      , {{{ 290, 240, 290, 130, 290}
07445      , { 290, 180, 290, 130, 290}
07446      , { 290, 240, 290, 130, 290}
07447      , { 290, 180, 290, 130, 290}
07448      , { 290, 240, 290, 130, 290}
07449      }
07450      , {{{ 290, 180, 290, 260, 290}
07451      , { 290, 180, 290, 260, 290}
07452      , { 290, 180, 290, 130, 290}
07453      , { 260, 180, 160, 260, 160}
```

```
07454     , { 290, 180, 290, 130, 290}
07455     }
07456     , {{ 290, 240, 290, 260, 290}
07457     , { 290, 180, 290, 130, 290}
07458     , { 290, 240, 290, 130, 290}
07459     , { 290, 180, 290, 130, 290}
07460     , { 290, 180, 290, 260, 290}
07461     }
07462     }
07463     , {{{ 350, 320, 350, 320, 290}
07464     , { 350, 320, 350, 320, 290}
07465     , { 290, 280, 290, 280, 200}
07466     , { 290, 280, 290, 280, 200}
07467     , { 290, 280, 290, 280, 200}
07468     }
07469     , {{ 320, 320, 320, 320, 290}
07470     , { 320, 320, 320, 320, 290}
07471     , { 290, 280, 290, 280, 200}
07472     , { 270, 200, 270, 200, 120}
07473     , { 290, 280, 290, 280, 200}
07474     }
07475     , {{ 290, 280, 290, 280, 200}
07476     , { 290, 280, 290, 280, 200}
07477     , { 290, 280, 290, 280, 200}
07478     , { 290, 280, 290, 280, 200}
07479     , { 290, 280, 290, 280, 200}
07480     }
07481     , {{ 350, 280, 350, 280, 200}
07482     , { 350, 280, 350, 280, 200}
07483     , { 290, 280, 290, 280, 200}
07484     , { 200, 150, 160, 150, 200}
07485     , { 290, 280, 290, 280, 200}
07486     }
07487     , {{ 290, 280, 290, 280, 200}
07488     , { 290, 280, 290, 280, 200}
07489     , { 290, 280, 290, 280, 200}
07490     , { 290, 280, 290, 280, 200}
07491     , { 290, 280, 290, 280, 200}
07492     }
07493     }
07494     }
07495     }
07496     , {{{ { INF, INF, INF, INF, INF}
07497     , { INF, INF, INF, INF, INF}
07498     , { INF, INF, INF, INF, INF}
07499     , { INF, INF, INF, INF, INF}
07500     , { INF, INF, INF, INF, INF}
07501     }
07502     , {{ { INF, INF, INF, INF, INF}
07503     , { INF, INF, INF, INF, INF}
07504     , { INF, INF, INF, INF, INF}
07505     , { INF, INF, INF, INF, INF}
07506     , { INF, INF, INF, INF, INF}
07507     }
07508     , {{ { INF, INF, INF, INF, INF}
07509     , { INF, INF, INF, INF, INF}
07510     , { INF, INF, INF, INF, INF}
07511     , { INF, INF, INF, INF, INF}
07512     , { INF, INF, INF, INF, INF}
07513     }
07514     , {{ { INF, INF, INF, INF, INF}
07515     , { INF, INF, INF, INF, INF}
07516     , { INF, INF, INF, INF, INF}
07517     , { INF, INF, INF, INF, INF}
07518     , { INF, INF, INF, INF, INF}
07519     }
07520     , {{ { INF, INF, INF, INF, INF}
07521     , { INF, INF, INF, INF, INF}
07522     , { INF, INF, INF, INF, INF}
07523     , { INF, INF, INF, INF, INF}
07524     , { INF, INF, INF, INF, INF}
07525     }
07526     }
07527     , {{{ { INF, INF, INF, INF, INF}
07528     , { INF, INF, INF, INF, INF}
07529     , { INF, INF, INF, INF, INF}
07530     , { INF, INF, INF, INF, INF}
07531     , { INF, INF, INF, INF, INF}
07532     }
07533     , {{ { INF, INF, INF, INF, INF}
07534     , { INF, INF, INF, INF, INF}
07535     , { INF, INF, INF, INF, INF}
07536     , { INF, INF, INF, INF, INF}
07537     , { INF, INF, INF, INF, INF}
07538     }
07539     , {{ { INF, INF, INF, INF, INF}
07540     , { INF, INF, INF, INF, INF}
```

```
07541      , {   INF,   INF,   INF,   INF,   INF }
07542      , {   INF,   INF,   INF,   INF,   INF }
07543      , {   INF,   INF,   INF,   INF,   INF }
07544      }
07545      , { {   INF,   INF,   INF,   INF,   INF }
07546      , {   INF,   INF,   INF,   INF,   INF }
07547      , {   INF,   INF,   INF,   INF,   INF }
07548      , {   INF,   INF,   INF,   INF,   INF }
07549      , {   INF,   INF,   INF,   INF,   INF }
07550      }
07551      , { {   INF,   INF,   INF,   INF,   INF }
07552      , {   INF,   INF,   INF,   INF,   INF }
07553      , {   INF,   INF,   INF,   INF,   INF }
07554      , {   INF,   INF,   INF,   INF,   INF }
07555      , {   INF,   INF,   INF,   INF,   INF }
07556      }
07557      }
07558      , { { {   INF,   INF,   INF,   INF,   INF }
07559      , {   INF,   INF,   INF,   INF,   INF }
07560      , {   INF,   INF,   INF,   INF,   INF }
07561      , {   INF,   INF,   INF,   INF,   INF }
07562      , {   INF,   INF,   INF,   INF,   INF }
07563      }
07564      , { {   INF,   INF,   INF,   INF,   INF }
07565      , {   INF,   INF,   INF,   INF,   INF }
07566      , {   INF,   INF,   INF,   INF,   INF }
07567      , {   INF,   INF,   INF,   INF,   INF }
07568      , {   INF,   INF,   INF,   INF,   INF }
07569      }
07570      , { {   INF,   INF,   INF,   INF,   INF }
07571      , {   INF,   INF,   INF,   INF,   INF }
07572      , {   INF,   INF,   INF,   INF,   INF }
07573      , {   INF,   INF,   INF,   INF,   INF }
07574      , {   INF,   INF,   INF,   INF,   INF }
07575      }
07576      , { {   INF,   INF,   INF,   INF,   INF }
07577      , {   INF,   INF,   INF,   INF,   INF }
07578      , {   INF,   INF,   INF,   INF,   INF }
07579      , {   INF,   INF,   INF,   INF,   INF }
07580      , {   INF,   INF,   INF,   INF,   INF }
07581      }
07582      , { {   INF,   INF,   INF,   INF,   INF }
07583      , {   INF,   INF,   INF,   INF,   INF }
07584      , {   INF,   INF,   INF,   INF,   INF }
07585      , {   INF,   INF,   INF,   INF,   INF }
07586      , {   INF,   INF,   INF,   INF,   INF }
07587      }
07588      }
07589      , { { {   INF,   INF,   INF,   INF,   INF }
07590      , {   INF,   INF,   INF,   INF,   INF }
07591      , {   INF,   INF,   INF,   INF,   INF }
07592      , {   INF,   INF,   INF,   INF,   INF }
07593      , {   INF,   INF,   INF,   INF,   INF }
07594      }
07595      , { {   INF,   INF,   INF,   INF,   INF }
07596      , {   INF,   INF,   INF,   INF,   INF }
07597      , {   INF,   INF,   INF,   INF,   INF }
07598      , {   INF,   INF,   INF,   INF,   INF }
07599      , {   INF,   INF,   INF,   INF,   INF }
07600      }
07601      , { {   INF,   INF,   INF,   INF,   INF }
07602      , {   INF,   INF,   INF,   INF,   INF }
07603      , {   INF,   INF,   INF,   INF,   INF }
07604      , {   INF,   INF,   INF,   INF,   INF }
07605      , {   INF,   INF,   INF,   INF,   INF }
07606      }
07607      , { {   INF,   INF,   INF,   INF,   INF }
07608      , {   INF,   INF,   INF,   INF,   INF }
07609      , {   INF,   INF,   INF,   INF,   INF }
07610      , {   INF,   INF,   INF,   INF,   INF }
07611      , {   INF,   INF,   INF,   INF,   INF }
07612      }
07613      , { {   INF,   INF,   INF,   INF,   INF }
07614      , {   INF,   INF,   INF,   INF,   INF }
07615      , {   INF,   INF,   INF,   INF,   INF }
07616      , {   INF,   INF,   INF,   INF,   INF }
07617      , {   INF,   INF,   INF,   INF,   INF }
07618      }
07619      }
07620      , { { {   INF,   INF,   INF,   INF,   INF }
07621      , {   INF,   INF,   INF,   INF,   INF }
07622      , {   INF,   INF,   INF,   INF,   INF }
07623      , {   INF,   INF,   INF,   INF,   INF }
07624      , {   INF,   INF,   INF,   INF,   INF }
07625      }
07626      , { {   INF,   INF,   INF,   INF,   INF }
07627      , {   INF,   INF,   INF,   INF,   INF }
```



```
07628     , {   INF,   INF,   INF,   INF,   INF }
07629     , {   INF,   INF,   INF,   INF,   INF }
07630     , {   INF,   INF,   INF,   INF,   INF }
07631     }
07632     , { {   INF,   INF,   INF,   INF,   INF }
07633     , {   INF,   INF,   INF,   INF,   INF }
07634     , {   INF,   INF,   INF,   INF,   INF }
07635     , {   INF,   INF,   INF,   INF,   INF }
07636     , {   INF,   INF,   INF,   INF,   INF }
07637     }
07638     , { {   INF,   INF,   INF,   INF,   INF }
07639     , {   INF,   INF,   INF,   INF,   INF }
07640     , {   INF,   INF,   INF,   INF,   INF }
07641     , {   INF,   INF,   INF,   INF,   INF }
07642     , {   INF,   INF,   INF,   INF,   INF }
07643     }
07644     , { {   INF,   INF,   INF,   INF,   INF }
07645     , {   INF,   INF,   INF,   INF,   INF }
07646     , {   INF,   INF,   INF,   INF,   INF }
07647     , {   INF,   INF,   INF,   INF,   INF }
07648     , {   INF,   INF,   INF,   INF,   INF }
07649     }
07650     }
07651     }
07652     , { { { 240,   240,   240,   190,   240 }
07653     , { 240,   240,   240,   190,   240 }
07654     , { 220,   220,   220,   190,   220 }
07655     , { 240,   240,   240,   190,   240 }
07656     , { 210,   210,   210,   170,   210 }
07657     }
07658     , { { 200,   200,   200,   150,   200 }
07659     , { 200,   200,   200,   150,   200 }
07660     , { 190,   190,   190,   150,   190 }
07661     , { 160,   100,   160,    80,   130 }
07662     , { 190,   190,   190,   150,   190 }
07663     }
07664     , { { 240,   240,   240,   190,   240 }
07665     , { 240,   240,   240,   190,   240 }
07666     , { 220,   220,   220,   190,   220 }
07667     , { 240,   240,   240,   190,   240 }
07668     , { 210,   210,   210,   170,   210 }
07669     }
07670     , { { 190,   190,   190,   150,   190 }
07671     , { 160,   100,   160,    80,   130 }
07672     , { 190,   190,   190,   150,   190 }
07673     , { 150,    70,    50,   150,    90 }
07674     , { 190,   190,   190,   150,   190 }
07675     }
07676     , { { 240,   240,   240,   190,   240 }
07677     , { 240,   240,   240,   190,   240 }
07678     , { 210,   210,   210,   170,   210 }
07679     , { 240,   240,   240,   190,   240 }
07680     , { 180,   180,   120,    90,   120 }
07681     }
07682     }
07683     , { { { 240,   240,   240,   190,   240 }
07684     , { 240,   240,   240,   140,   240 }
07685     , { 220,   220,   220,   190,   220 }
07686     , { 240,   240,   240,   140,   240 }
07687     , { 210,   210,   210,   170,   210 }
07688     }
07689     , { { 200,   200,   200,   100,   200 }
07690     , { 200,   200,   200,   100,   200 }
07691     , { 190,   190,   190,   100,   190 }
07692     , { 100,   100,   100,    10,   100 }
07693     , { 190,   190,   190,   100,   190 }
07694     }
07695     , { { 240,   240,   240,   190,   240 }
07696     , { 240,   240,   240,   140,   240 }
07697     , { 220,   220,   220,   190,   220 }
07698     , { 240,   240,   240,   140,   240 }
07699     , { 210,   210,   210,   170,   210 }
07700     }
07701     , { { 190,   190,   190,   100,   190 }
07702     , { 100,   100,   100,    10,   100 }
07703     , { 190,   190,   190,   100,   190 }
07704     , {  80,    50,    50,    80,    50 }
07705     , { 190,   190,   190,   100,   190 }
07706     }
07707     , { { 240,   240,   240,   170,   240 }
07708     , { 240,   240,   240,   140,   240 }
07709     , { 210,   210,   210,   170,   210 }
07710     , { 240,   240,   240,   140,   240 }
07711     , { 180,   180,   120,    20,   120 }
07712     }
07713     }
07714     , { { { 240,   190,   240,   190,   210 }
```

```
07715      , { 240, 190, 240, 190, 210}
07716      , { 220, 180, 220, 180, 190}
07717      , { 240, 190, 240, 190, 210}
07718      , { 210, 160, 210, 160, 180}
07719      }
07720      , { { 200, 150, 200, 150, 170}
07721      , { 200, 150, 200, 150, 170}
07722      , { 190, 150, 190, 150, 160}
07723      , { 160, 60, 160, 60, 130}
07724      , { 190, 150, 190, 150, 160}
07725      }
07726      , { { 240, 190, 240, 190, 210}
07727      , { 240, 190, 240, 190, 210}
07728      , { 220, 180, 220, 180, 190}
07729      , { 240, 190, 240, 190, 210}
07730      , { 210, 160, 210, 160, 180}
07731      }
07732      , { { 190, 150, 190, 150, 160}
07733      , { 160, 60, 160, 60, 130}
07734      , { 190, 150, 190, 150, 160}
07735      , { 50, 0, 50, 0, 20}
07736      , { 190, 150, 190, 150, 160}
07737      }
07738      , { { 240, 190, 240, 190, 210}
07739      , { 240, 190, 240, 190, 210}
07740      , { 210, 160, 210, 160, 180}
07741      , { 240, 190, 240, 190, 210}
07742      , { 120, 70, 120, 70, 90}
07743      }
07744      }
07745      , { { { 240, 180, 240, 150, 240}
07746      , { 240, 130, 240, 80, 240}
07747      , { 220, 180, 220, 70, 220}
07748      , { 240, 130, 240, 150, 240}
07749      , { 210, 160, 210, 90, 210}
07750      }
07751      , { { 200, 90, 200, 80, 200}
07752      , { 200, 90, 200, 40, 200}
07753      , { 190, 90, 190, 40, 190}
07754      , { 100, 0, 100, 80, 100}
07755      , { 190, 90, 190, 40, 190}
07756      }
07757      , { { 240, 180, 240, 80, 240}
07758      , { 240, 130, 240, 80, 240}
07759      , { 220, 180, 220, 70, 220}
07760      , { 240, 130, 240, 80, 240}
07761      , { 210, 160, 210, 50, 210}
07762      }
07763      , { { 190, 90, 190, 150, 190}
07764      , { 100, 0, 100, 80, 100}
07765      , { 190, 90, 190, 40, 190}
07766      , { 150, 70, 50, 150, 50}
07767      , { 190, 90, 190, 40, 190}
07768      }
07769      , { { 240, 160, 240, 90, 240}
07770      , { 240, 130, 240, 80, 240}
07771      , { 210, 160, 210, 50, 210}
07772      , { 240, 130, 240, 80, 240}
07773      , { 120, 10, 120, 90, 120}
07774      }
07775      }
07776      , { { { 240, 190, 240, 190, 170}
07777      , { 240, 190, 240, 190, 170}
07778      , { 220, 180, 220, 180, 140}
07779      , { 240, 190, 240, 190, 150}
07780      , { 210, 160, 210, 160, 120}
07781      }
07782      , { { 200, 150, 200, 150, 170}
07783      , { 200, 150, 200, 150, 170}
07784      , { 190, 150, 190, 150, 110}
07785      , { 160, 60, 160, 60, 20}
07786      , { 190, 150, 190, 150, 110}
07787      }
07788      , { { 240, 190, 240, 190, 150}
07789      , { 240, 190, 240, 190, 150}
07790      , { 220, 180, 220, 180, 140}
07791      , { 240, 190, 240, 190, 150}
07792      , { 210, 160, 210, 160, 120}
07793      }
07794      , { { 190, 150, 190, 150, 110}
07795      , { 160, 60, 160, 60, 20}
07796      , { 190, 150, 190, 150, 110}
07797      , { 90, 0, 50, 0, 90}
07798      , { 190, 150, 190, 150, 110}
07799      }
07800      , { { 240, 190, 240, 190, 150}
07801      , { 240, 190, 240, 190, 150}
```

```
07802     , { 210, 160, 210, 160, 120}
07803     , { 240, 190, 240, 190, 150}
07804     , { 120, 70, 120, 70, 30}
07805     }
07806     }
07807     }
07808     , {{{ 210, 210, 210, 170, 210}
07809     , { 210, 210, 210, 170, 210}
07810     , { 190, 190, 190, 160, 190}
07811     , { 180, 180, 180, 150, 180}
07812     , { 190, 190, 190, 150, 190}
07813     }
07814     , {{ 210, 210, 210, 170, 210}
07815     , { 210, 210, 210, 170, 210}
07816     , { 190, 190, 190, 140, 190}
07817     , { 70, 10, 70, -10, 40}
07818     , { 190, 190, 190, 140, 190}
07819     }
07820     , {{ 190, 190, 190, 150, 190}
07821     , { 180, 180, 180, 140, 180}
07822     , { 190, 190, 190, 150, 190}
07823     , { 180, 180, 180, 140, 180}
07824     , { 190, 190, 190, 150, 190}
07825     }
07826     , {{ 190, 190, 190, 150, 190}
07827     , { 130, 70, 130, 50, 100}
07828     , { 190, 190, 190, 140, 190}
07829     , { 150, 70, 50, 150, 90}
07830     , { 190, 190, 190, 140, 190}
07831     }
07832     , {{ 190, 190, 190, 160, 190}
07833     , { 180, 180, 180, 140, 180}
07834     , { 190, 190, 190, 160, 190}
07835     , { 180, 180, 180, 140, 180}
07836     , { 170, 170, 110, 90, 110}
07837     }
07838     }
07839     , {{{ 210, 210, 210, 160, 210}
07840     , { 210, 210, 210, 120, 210}
07841     , { 190, 190, 190, 160, 190}
07842     , { 180, 180, 180, 90, 180}
07843     , { 190, 190, 190, 150, 190}
07844     }
07845     , {{ 210, 210, 210, 120, 210}
07846     , { 210, 210, 210, 120, 210}
07847     , { 190, 190, 190, 90, 190}
07848     , { 10, 10, 10, -80, 10}
07849     , { 190, 190, 190, 90, 190}
07850     }
07851     , {{ 190, 190, 190, 150, 190}
07852     , { 180, 180, 180, 90, 180}
07853     , { 190, 190, 190, 150, 190}
07854     , { 180, 180, 180, 90, 180}
07855     , { 190, 190, 190, 150, 190}
07856     }
07857     , {{ 190, 190, 190, 90, 190}
07858     , { 70, 70, 70, -20, 70}
07859     , { 190, 190, 190, 90, 190}
07860     , { 80, 50, 50, 80, 50}
07861     , { 190, 190, 190, 90, 190}
07862     }
07863     , {{ 190, 190, 190, 160, 190}
07864     , { 180, 180, 180, 90, 180}
07865     , { 190, 190, 190, 160, 190}
07866     , { 180, 180, 180, 90, 180}
07867     , { 170, 170, 110, 20, 110}
07868     }
07869     }
07870     , {{{ 210, 170, 210, 170, 180}
07871     , { 210, 170, 210, 170, 180}
07872     , { 190, 150, 190, 150, 160}
07873     , { 180, 140, 180, 140, 150}
07874     , { 190, 140, 190, 140, 160}
07875     }
07876     , {{ 210, 170, 210, 170, 180}
07877     , { 210, 170, 210, 170, 180}
07878     , { 190, 140, 190, 140, 160}
07879     , { 70, -30, 70, -30, 40}
07880     , { 190, 140, 190, 140, 160}
07881     }
07882     , {{ 190, 140, 190, 140, 160}
07883     , { 180, 140, 180, 140, 150}
07884     , { 190, 140, 190, 140, 160}
07885     , { 180, 140, 180, 140, 150}
07886     , { 190, 140, 190, 140, 160}
07887     }
07888     , {{ 190, 140, 190, 140, 160}
```

```
07889      , { 130, 30, 130, 30, 100}
07890      , { 190, 140, 190, 140, 160}
07891      , { 50, 0, 50, 0, 20}
07892      , { 190, 140, 190, 140, 160}
07893      }
07894      , { { 190, 150, 190, 150, 160}
07895      , { 180, 140, 180, 140, 150}
07896      , { 190, 150, 190, 150, 160}
07897      , { 180, 140, 180, 140, 150}
07898      , { 110, 70, 110, 70, 80}
07899      }
07900      }
07901      , { { { 210, 150, 210, 150, 210}
07902      , { 210, 110, 210, 60, 210}
07903      , { 190, 150, 190, 40, 190}
07904      , { 180, 80, 180, 150, 180}
07905      , { 190, 140, 190, 90, 190}
07906      }
07907      , { { 210, 110, 210, 60, 210}
07908      , { 210, 110, 210, 60, 210}
07909      , { 190, 80, 190, 30, 190}
07910      , { 10, -90, 10, -10, 10}
07911      , { 190, 80, 190, 30, 190}
07912      }
07913      , { { 190, 140, 190, 30, 190}
07914      , { 180, 80, 180, 30, 180}
07915      , { 190, 140, 190, 30, 190}
07916      , { 180, 80, 180, 30, 180}
07917      , { 190, 140, 190, 30, 190}
07918      }
07919      , { { 190, 80, 190, 150, 190}
07920      , { 70, -30, 70, 50, 70}
07921      , { 190, 80, 190, 30, 190}
07922      , { 150, 70, 50, 150, 50}
07923      , { 190, 80, 190, 30, 190}
07924      }
07925      , { { 190, 150, 190, 90, 190}
07926      , { 180, 80, 180, 30, 180}
07927      , { 190, 150, 190, 40, 190}
07928      , { 180, 80, 180, 30, 180}
07929      , { 110, 10, 110, 90, 110}
07930      }
07931      }
07932      , { { { 210, 170, 210, 170, 190}
07933      , { 210, 170, 210, 170, 190}
07934      , { 190, 150, 190, 150, 110}
07935      , { 180, 140, 180, 140, 100}
07936      , { 190, 140, 190, 140, 100}
07937      }
07938      , { { 210, 170, 210, 170, 190}
07939      , { 210, 170, 210, 170, 190}
07940      , { 190, 140, 190, 140, 100}
07941      , { 70, -30, 70, -30, -70}
07942      , { 190, 140, 190, 140, 100}
07943      }
07944      , { { 190, 140, 190, 140, 100}
07945      , { 180, 140, 180, 140, 100}
07946      , { 190, 140, 190, 140, 100}
07947      , { 180, 140, 180, 140, 100}
07948      , { 190, 140, 190, 140, 100}
07949      }
07950      , { { 190, 140, 190, 140, 100}
07951      , { 130, 30, 130, 30, -10}
07952      , { 190, 140, 190, 140, 100}
07953      , { 90, 0, 50, 0, 90}
07954      , { 190, 140, 190, 140, 100}
07955      }
07956      , { { 190, 150, 190, 150, 110}
07957      , { 180, 140, 180, 140, 100}
07958      , { 190, 150, 190, 150, 110}
07959      , { 180, 140, 180, 140, 100}
07960      , { 110, 70, 110, 70, 30}
07961      }
07962      }
07963      }
07964      , { { { { 370, 370, 340, 300, 340}
07965      , { 340, 340, 340, 300, 340}
07966      , { 310, 310, 310, 270, 310}
07967      , { 310, 310, 310, 280, 310}
07968      , { 370, 370, 310, 280, 310}
07969      }
07970      , { { 340, 340, 340, 300, 340}
07971      , { 340, 340, 340, 300, 340}
07972      , { 310, 310, 310, 260, 310}
07973      , { 290, 230, 290, 200, 260}
07974      , { 310, 310, 310, 260, 310}
07975      }
```

```
07976 ,{{ 310, 310, 310, 270, 310}
07977 ,{ 310, 310, 310, 260, 310}
07978 ,{ 310, 310, 310, 270, 310}
07979 ,{ 310, 310, 310, 260, 310}
07980 ,{ 310, 310, 310, 270, 310}
07981 }
07982 ,{{ 330, 310, 330, 280, 310}
07983 ,{ 330, 270, 330, 240, 300}
07984 ,{ 310, 310, 310, 260, 310}
07985 ,{ 280, 200, 180, 280, 220}
07986 ,{ 310, 310, 310, 260, 310}
07987 }
07988 ,{{ 370, 370, 310, 280, 310}
07989 ,{ 310, 310, 310, 260, 310}
07990 ,{ 310, 310, 310, 270, 310}
07991 ,{ 310, 310, 310, 260, 310}
07992 ,{ 370, 370, 310, 280, 310}
07993 }
07994 }
07995 ,{{{ 370, 370, 340, 270, 340}
07996 ,{ 340, 340, 340, 250, 340}
07997 ,{ 310, 310, 310, 270, 310}
07998 ,{ 310, 310, 310, 210, 310}
07999 ,{ 370, 370, 310, 270, 310}
08000 }
08001 ,{{ 340, 340, 340, 250, 340}
08002 ,{ 340, 340, 340, 250, 340}
08003 ,{ 310, 310, 310, 210, 310}
08004 ,{ 230, 230, 230, 130, 230}
08005 ,{ 310, 310, 310, 210, 310}
08006 }
08007 ,{{ 310, 310, 310, 270, 310}
08008 ,{ 310, 310, 310, 210, 310}
08009 ,{ 310, 310, 310, 270, 310}
08010 ,{ 310, 310, 310, 210, 310}
08011 ,{ 310, 310, 310, 270, 310}
08012 }
08013 ,{{ 310, 310, 310, 210, 310}
08014 ,{ 270, 270, 270, 170, 270}
08015 ,{ 310, 310, 310, 210, 310}
08016 ,{ 210, 180, 180, 210, 180}
08017 ,{ 310, 310, 310, 210, 310}
08018 }
08019 ,{{ 370, 370, 310, 270, 310}
08020 ,{ 310, 310, 310, 210, 310}
08021 ,{ 310, 310, 310, 270, 310}
08022 ,{ 310, 310, 310, 210, 310}
08023 ,{ 370, 370, 310, 210, 310}
08024 }
08025 }
08026 ,{{{ 340, 300, 340, 300, 310}
08027 ,{ 340, 300, 340, 300, 310}
08028 ,{ 310, 260, 310, 260, 280}
08029 ,{ 310, 260, 310, 260, 280}
08030 ,{ 310, 260, 310, 260, 280}
08031 }
08032 ,{{ 340, 300, 340, 300, 310}
08033 ,{ 340, 300, 340, 300, 310}
08034 ,{ 310, 260, 310, 260, 280}
08035 ,{ 290, 180, 290, 180, 260}
08036 ,{ 310, 260, 310, 260, 280}
08037 }
08038 ,{{ 310, 260, 310, 260, 280}
08039 ,{ 310, 260, 310, 260, 280}
08040 ,{ 310, 260, 310, 260, 280}
08041 ,{ 310, 260, 310, 260, 280}
08042 ,{ 310, 260, 310, 260, 280}
08043 }
08044 ,{{ 330, 260, 330, 260, 300}
08045 ,{ 330, 220, 330, 220, 300}
08046 ,{ 310, 260, 310, 260, 280}
08047 ,{ 180, 130, 180, 130, 150}
08048 ,{ 310, 260, 310, 260, 280}
08049 }
08050 ,{{ 310, 260, 310, 260, 280}
08051 ,{ 310, 260, 310, 260, 280}
08052 ,{ 310, 260, 310, 260, 280}
08053 ,{ 310, 260, 310, 260, 280}
08054 ,{ 310, 260, 310, 260, 280}
08055 }
08056 }
08057 ,{{{ 340, 260, 340, 280, 340}
08058 ,{ 340, 240, 340, 240, 340}
08059 ,{ 310, 260, 310, 150, 310}
08060 ,{ 310, 200, 310, 280, 310}
08061 ,{ 310, 260, 310, 280, 310}
08062 }
```

```
08063 ,{{ 340, 240, 340, 200, 340}
08064 ,{ 340, 240, 340, 190, 340}
08065 ,{ 310, 200, 310, 150, 310}
08066 ,{ 230, 120, 230, 200, 230}
08067 ,{ 310, 200, 310, 150, 310}
08068 }
08069 ,{{ 310, 260, 310, 150, 310}
08070 ,{ 310, 200, 310, 150, 310}
08071 ,{ 310, 260, 310, 150, 310}
08072 ,{ 310, 200, 310, 150, 310}
08073 ,{ 310, 260, 310, 150, 310}
08074 }
08075 ,{{ 310, 200, 310, 280, 310}
08076 ,{ 270, 160, 270, 240, 270}
08077 ,{ 310, 200, 310, 150, 310}
08078 ,{ 280, 200, 180, 280, 180}
08079 ,{ 310, 200, 310, 150, 310}
08080 }
08081 ,{{ 310, 260, 310, 280, 310}
08082 ,{ 310, 200, 310, 150, 310}
08083 ,{ 310, 260, 310, 150, 310}
08084 ,{ 310, 200, 310, 150, 310}
08085 ,{ 310, 200, 310, 280, 310}
08086 }
08087 }
08088 ,{{{ 340, 300, 340, 300, 320}
08089 ,{ 340, 300, 340, 300, 320}
08090 ,{ 310, 260, 310, 260, 220}
08091 ,{ 310, 260, 310, 260, 220}
08092 ,{ 310, 260, 310, 260, 220}
08093 }
08094 ,{{ 340, 300, 340, 300, 320}
08095 ,{ 340, 300, 340, 300, 320}
08096 ,{ 310, 260, 310, 260, 220}
08097 ,{ 290, 180, 290, 180, 140}
08098 ,{ 310, 260, 310, 260, 220}
08099 }
08100 ,{{ 310, 260, 310, 260, 220}
08101 ,{ 310, 260, 310, 260, 220}
08102 ,{ 310, 260, 310, 260, 220}
08103 ,{ 310, 260, 310, 260, 220}
08104 ,{ 310, 260, 310, 260, 220}
08105 }
08106 ,{{ 330, 260, 330, 260, 220}
08107 ,{ 330, 220, 330, 220, 180}
08108 ,{ 310, 260, 310, 260, 220}
08109 ,{ 220, 130, 180, 130, 220}
08110 ,{ 310, 260, 310, 260, 220}
08111 }
08112 ,{{ 310, 260, 310, 260, 220}
08113 ,{ 310, 260, 310, 260, 220}
08114 ,{ 310, 260, 310, 260, 220}
08115 ,{ 310, 260, 310, 260, 220}
08116 ,{ 310, 260, 310, 260, 220}
08117 }
08118 }
08119 }
08120 ,{{{ 370, 340, 370, 280, 340}
08121 ,{ 370, 310, 370, 280, 340}
08122 ,{ 280, 280, 280, 240, 280}
08123 ,{ 280, 280, 280, 250, 280}
08124 ,{ 340, 340, 280, 250, 280}
08125 }
08126 ,{{ 280, 280, 280, 230, 280}
08127 ,{ 240, 240, 240, 200, 240}
08128 ,{ 280, 280, 280, 230, 280}
08129 ,{ 200, 140, 200, 120, 170}
08130 ,{ 280, 280, 280, 230, 280}
08131 }
08132 ,{{ 280, 280, 280, 240, 280}
08133 ,{ 280, 280, 280, 230, 280}
08134 ,{ 280, 280, 280, 240, 280}
08135 ,{ 280, 280, 280, 230, 280}
08136 ,{ 280, 280, 280, 240, 280}
08137 }
08138 ,{{ 370, 310, 370, 280, 340}
08139 ,{ 370, 310, 370, 280, 340}
08140 ,{ 280, 280, 280, 230, 280}
08141 ,{ 250, 170, 150, 250, 190}
08142 ,{ 280, 280, 280, 230, 280}
08143 }
08144 ,{{ 340, 340, 280, 250, 280}
08145 ,{ 280, 280, 280, 230, 280}
08146 ,{ 280, 280, 280, 240, 280}
08147 ,{ 280, 280, 280, 230, 280}
08148 ,{ 340, 340, 280, 250, 280}
08149 }
```

```
08150     }
08151     , {{ { 340, 340, 310, 240, 310}
08152     , { 310, 310, 310, 210, 310}
08153     , { 280, 280, 280, 240, 280}
08154     , { 280, 280, 280, 180, 280}
08155     , { 340, 340, 280, 240, 280}
08156     }
08157     , {{ { 280, 280, 280, 180, 280}
08158     , { 240, 240, 240, 150, 240}
08159     , { 280, 280, 280, 180, 280}
08160     , { 140, 140, 140, 50, 140}
08161     , { 280, 280, 280, 180, 280}
08162     }
08163     , {{ { 280, 280, 280, 240, 280}
08164     , { 280, 280, 280, 180, 280}
08165     , { 280, 280, 280, 240, 280}
08166     , { 280, 280, 280, 180, 280}
08167     , { 280, 280, 280, 240, 280}
08168     }
08169     , {{ { 310, 310, 310, 210, 310}
08170     , { 310, 310, 310, 210, 310}
08171     , { 280, 280, 280, 180, 280}
08172     , { 180, 150, 150, 180, 150}
08173     , { 280, 280, 280, 180, 280}
08174     }
08175     , {{ { 340, 340, 280, 240, 280}
08176     , { 280, 280, 280, 180, 280}
08177     , { 280, 280, 280, 240, 280}
08178     , { 280, 280, 280, 180, 280}
08179     , { 340, 340, 280, 180, 280}
08180     }
08181     }
08182     , {{ { 370, 260, 370, 260, 340}
08183     , { 370, 260, 370, 260, 340}
08184     , { 280, 230, 280, 230, 250}
08185     , { 280, 230, 280, 230, 250}
08186     , { 280, 230, 280, 230, 250}
08187     }
08188     , {{ { 280, 230, 280, 230, 250}
08189     , { 240, 200, 240, 200, 210}
08190     , { 280, 230, 280, 230, 250}
08191     , { 200, 100, 200, 100, 170}
08192     , { 280, 230, 280, 230, 250}
08193     }
08194     , {{ { 280, 230, 280, 230, 250}
08195     , { 280, 230, 280, 230, 250}
08196     , { 280, 230, 280, 230, 250}
08197     , { 280, 230, 280, 230, 250}
08198     , { 280, 230, 280, 230, 250}
08199     }
08200     , {{ { 370, 260, 370, 260, 340}
08201     , { 370, 260, 370, 260, 340}
08202     , { 280, 230, 280, 230, 250}
08203     , { 150, 100, 150, 100, 120}
08204     , { 280, 230, 280, 230, 250}
08205     }
08206     , {{ { 280, 230, 280, 230, 250}
08207     , { 280, 230, 280, 230, 250}
08208     , { 280, 230, 280, 230, 250}
08209     , { 280, 230, 280, 230, 250}
08210     , { 280, 230, 280, 230, 250}
08211     }
08212     }
08213     , {{ { 310, 230, 310, 280, 310}
08214     , { 310, 200, 310, 280, 310}
08215     , { 280, 230, 280, 120, 280}
08216     , { 280, 170, 280, 250, 280}
08217     , { 280, 230, 280, 250, 280}
08218     }
08219     , {{ { 280, 170, 280, 120, 280}
08220     , { 240, 140, 240, 90, 240}
08221     , { 280, 170, 280, 120, 280}
08222     , { 140, 40, 140, 120, 140}
08223     , { 280, 170, 280, 120, 280}
08224     }
08225     , {{ { 280, 230, 280, 120, 280}
08226     , { 280, 170, 280, 120, 280}
08227     , { 280, 230, 280, 120, 280}
08228     , { 280, 170, 280, 120, 280}
08229     , { 280, 230, 280, 120, 280}
08230     }
08231     , {{ { 310, 200, 310, 280, 310}
08232     , { 310, 200, 310, 280, 310}
08233     , { 280, 170, 280, 120, 280}
08234     , { 250, 170, 150, 250, 150}
08235     , { 280, 170, 280, 120, 280}
08236     }
```

```
08237 ,{{ 280, 230, 280, 250, 280}
08238 ,{ 280, 170, 280, 120, 280}
08239 ,{ 280, 230, 280, 120, 280}
08240 ,{ 280, 170, 280, 120, 280}
08241 ,{ 280, 170, 280, 250, 280}
08242 }
08243 }
08244 ,{{{ 370, 260, 370, 260, 220}
08245 ,{ 370, 260, 370, 260, 220}
08246 ,{ 280, 230, 280, 230, 190}
08247 ,{ 280, 230, 280, 230, 190}
08248 ,{ 280, 230, 280, 230, 190}
08249 }
08250 ,{{ 280, 230, 280, 230, 220}
08251 ,{ 240, 200, 240, 200, 220}
08252 ,{ 280, 230, 280, 230, 190}
08253 ,{ 200, 100, 200, 100, 60}
08254 ,{ 280, 230, 280, 230, 190}
08255 }
08256 ,{{ 280, 230, 280, 230, 190}
08257 ,{ 280, 230, 280, 230, 190}
08258 ,{ 280, 230, 280, 230, 190}
08259 ,{ 280, 230, 280, 230, 190}
08260 ,{ 280, 230, 280, 230, 190}
08261 }
08262 ,{{{ 370, 260, 370, 260, 220}
08263 ,{ 370, 260, 370, 260, 220}
08264 ,{ 280, 230, 280, 230, 190}
08265 ,{ 190, 100, 150, 100, 190}
08266 ,{ 280, 230, 280, 230, 190}
08267 }
08268 ,{{{ 280, 230, 280, 230, 190}
08269 ,{ 280, 230, 280, 230, 190}
08270 ,{ 280, 230, 280, 230, 190}
08271 ,{ 280, 230, 280, 230, 190}
08272 ,{ 280, 230, 280, 230, 190}
08273 }
08274 }
08275 }
08276 ,{{{ 280, 280, 280, 230, 280}
08277 ,{ 280, 280, 280, 230, 280}
08278 ,{ 260, 260, 260, 220, 260}
08279 ,{ 260, 260, 260, 220, 260}
08280 ,{ 260, 260, 260, 220, 260}
08281 }
08282 ,{{ 280, 280, 280, 230, 280}
08283 ,{ 280, 280, 280, 230, 280}
08284 ,{ 250, 250, 250, 210, 250}
08285 ,{ 210, 150, 210, 130, 180}
08286 ,{ 250, 250, 250, 210, 250}
08287 }
08288 ,{{{ 260, 260, 260, 220, 260}
08289 ,{ 260, 260, 260, 220, 260}
08290 ,{ 260, 260, 260, 220, 260}
08291 ,{ 260, 260, 260, 220, 260}
08292 ,{ 260, 260, 260, 220, 260}
08293 }
08294 ,{{ 280, 250, 280, 210, 250}
08295 ,{ 280, 220, 280, 200, 250}
08296 ,{ 250, 250, 250, 210, 250}
08297 ,{ 210, 130, 100, 210, 150}
08298 ,{ 250, 250, 250, 210, 250}
08299 }
08300 ,{{ 260, 260, 260, 220, 260}
08301 ,{ 260, 260, 260, 220, 260}
08302 ,{ 260, 260, 260, 220, 260}
08303 ,{ 260, 260, 260, 220, 260}
08304 ,{ 230, 230, 170, 140, 170}
08305 }
08306 }
08307 ,{{{ 280, 280, 280, 220, 280}
08308 ,{ 280, 280, 280, 180, 280}
08309 ,{ 260, 260, 260, 220, 260}
08310 ,{ 260, 260, 260, 170, 260}
08311 ,{ 260, 260, 260, 220, 260}
08312 }
08313 ,{{ 280, 280, 280, 180, 280}
08314 ,{ 280, 280, 280, 180, 280}
08315 ,{ 250, 250, 250, 160, 250}
08316 ,{ 150, 150, 150, 60, 150}
08317 ,{ 250, 250, 250, 160, 250}
08318 }
08319 ,{{ 260, 260, 260, 220, 260}
08320 ,{ 260, 260, 260, 170, 260}
08321 ,{ 260, 260, 260, 220, 260}
08322 ,{ 260, 260, 260, 170, 260}
08323 ,{ 260, 260, 260, 220, 260}
```



```

08324     }
08325     , {{ 250, 250, 250, 160, 250}
08326     , { 220, 220, 220, 130, 220}
08327     , { 250, 250, 250, 160, 250}
08328     , { 140, 100, 100, 140, 100}
08329     , { 250, 250, 250, 160, 250}
08330     }
08331     , {{ 260, 260, 260, 220, 260}
08332     , { 260, 260, 260, 170, 260}
08333     , { 260, 260, 260, 220, 260}
08334     , { 260, 260, 260, 170, 260}
08335     , { 230, 230, 170, 70, 170}
08336     }
08337     }
08338     , {{{ 280, 230, 280, 230, 250}
08339     , { 280, 230, 280, 230, 250}
08340     , { 260, 210, 260, 210, 230}
08341     , { 260, 220, 260, 220, 230}
08342     , { 260, 210, 260, 210, 230}
08343     }
08344     , {{{ 280, 230, 280, 230, 250}
08345     , { 280, 230, 280, 230, 250}
08346     , { 250, 210, 250, 210, 220}
08347     , { 210, 110, 210, 110, 180}
08348     , { 250, 210, 250, 210, 220}
08349     }
08350     , {{{ 260, 220, 260, 220, 230}
08351     , { 260, 220, 260, 220, 230}
08352     , { 260, 210, 260, 210, 230}
08353     , { 260, 220, 260, 220, 230}
08354     , { 260, 210, 260, 210, 230}
08355     }
08356     , {{{ 280, 210, 280, 210, 250}
08357     , { 280, 180, 280, 180, 250}
08358     , { 250, 210, 250, 210, 220}
08359     , { 100, 60, 100, 60, 70}
08360     , { 250, 210, 250, 210, 220}
08361     }
08362     , {{{ 260, 220, 260, 220, 230}
08363     , { 260, 220, 260, 220, 230}
08364     , { 260, 210, 260, 210, 230}
08365     , { 260, 220, 260, 220, 230}
08366     , { 170, 120, 170, 120, 140}
08367     }
08368     }
08369     , {{{ 280, 210, 280, 210, 280}
08370     , { 280, 170, 280, 200, 280}
08371     , { 260, 210, 260, 100, 260}
08372     , { 260, 160, 260, 210, 260}
08373     , { 260, 210, 260, 140, 260}
08374     }
08375     , {{{ 280, 170, 280, 130, 280}
08376     , { 280, 170, 280, 120, 280}
08377     , { 250, 150, 250, 100, 250}
08378     , { 150, 50, 150, 130, 150}
08379     , { 250, 150, 250, 100, 250}
08380     }
08381     , {{{ 260, 210, 260, 110, 260}
08382     , { 260, 160, 260, 110, 260}
08383     , { 260, 210, 260, 100, 260}
08384     , { 260, 160, 260, 110, 260}
08385     , { 260, 210, 260, 100, 260}
08386     }
08387     , {{{ 250, 150, 250, 210, 250}
08388     , { 220, 120, 220, 200, 220}
08389     , { 250, 150, 250, 100, 250}
08390     , { 210, 130, 100, 210, 100}
08391     , { 250, 150, 250, 100, 250}
08392     }
08393     , {{{ 260, 210, 260, 140, 260}
08394     , { 260, 160, 260, 110, 260}
08395     , { 260, 210, 260, 100, 260}
08396     , { 260, 160, 260, 110, 260}
08397     , { 170, 60, 170, 140, 170}
08398     }
08399     }
08400     , {{{ 280, 230, 280, 230, 250}
08401     , { 280, 230, 280, 230, 250}
08402     , { 260, 210, 260, 210, 170}
08403     , { 260, 220, 260, 220, 180}
08404     , { 260, 210, 260, 210, 170}
08405     }
08406     , {{{ 280, 230, 280, 230, 250}
08407     , { 280, 230, 280, 230, 250}
08408     , { 250, 210, 250, 210, 170}
08409     , { 210, 110, 210, 110, 70}
08410     , { 250, 210, 250, 210, 170}

```

```
08411      }
08412      ,{{ 260, 220, 260, 220, 180}
08413      ,{ 260, 220, 260, 220, 180}
08414      ,{ 260, 210, 260, 210, 170}
08415      ,{ 260, 220, 260, 220, 180}
08416      ,{ 260, 210, 260, 210, 170}
08417      }
08418      ,{{ 280, 210, 280, 210, 170}
08419      ,{ 280, 180, 280, 180, 140}
08420      ,{ 250, 210, 250, 210, 170}
08421      ,{ 150, 60, 100, 60, 150}
08422      ,{ 250, 210, 250, 210, 170}
08423      }
08424      ,{{ 260, 220, 260, 220, 180}
08425      ,{ 260, 220, 260, 220, 180}
08426      ,{ 260, 210, 260, 210, 170}
08427      ,{ 260, 220, 260, 220, 180}
08428      ,{ 170, 120, 170, 120, 80}
08429      }
08430      }
08431      }
08432      ,{{{ 280, 280, 280, 240, 280}
08433      ,{ 280, 280, 280, 230, 280}
08434      ,{ 280, 280, 280, 240, 280}
08435      ,{ 280, 280, 280, 230, 280}
08436      ,{ 280, 280, 280, 240, 280}
08437      }
08438      ,{{ 280, 280, 280, 230, 280}
08439      ,{ 280, 280, 280, 230, 280}
08440      ,{ 230, 230, 230, 190, 230}
08441      ,{ 230, 170, 230, 150, 200}
08442      ,{ 230, 230, 230, 190, 230}
08443      }
08444      ,{{ 280, 280, 280, 240, 280}
08445      ,{ 280, 280, 280, 230, 280}
08446      ,{ 280, 280, 280, 240, 280}
08447      ,{ 280, 280, 280, 230, 280}
08448      ,{ 280, 280, 280, 240, 280}
08449      }
08450      ,{{ 240, 230, 240, 230, 230}
08451      ,{ 240, 180, 240, 160, 210}
08452      ,{ 230, 230, 230, 190, 230}
08453      ,{ 230, 150, 120, 230, 170}
08454      ,{ 230, 230, 230, 190, 230}
08455      }
08456      ,{{ 280, 280, 280, 230, 280}
08457      ,{ 280, 280, 280, 230, 280}
08458      ,{ 250, 250, 250, 210, 250}
08459      ,{ 280, 280, 280, 230, 280}
08460      ,{ 250, 250, 190, 170, 190}
08461      }
08462      }
08463      ,{{{ 280, 280, 280, 240, 280}
08464      ,{ 280, 280, 280, 180, 280}
08465      ,{ 280, 280, 280, 240, 280}
08466      ,{ 280, 280, 280, 180, 280}
08467      ,{ 280, 280, 280, 240, 280}
08468      }
08469      ,{{ 280, 280, 280, 180, 280}
08470      ,{ 280, 280, 280, 180, 280}
08471      ,{ 230, 230, 230, 140, 230}
08472      ,{ 170, 170, 170, 80, 170}
08473      ,{ 230, 230, 230, 140, 230}
08474      }
08475      ,{{ 280, 280, 280, 240, 280}
08476      ,{ 280, 280, 280, 180, 280}
08477      ,{ 280, 280, 280, 240, 280}
08478      ,{ 280, 280, 280, 180, 280}
08479      ,{ 280, 280, 280, 240, 280}
08480      }
08481      ,{{ 230, 230, 230, 160, 230}
08482      ,{ 180, 180, 180, 90, 180}
08483      ,{ 230, 230, 230, 140, 230}
08484      ,{ 160, 120, 120, 160, 120}
08485      ,{ 230, 230, 230, 140, 230}
08486      }
08487      ,{{ 280, 280, 280, 210, 280}
08488      ,{ 280, 280, 280, 180, 280}
08489      ,{ 250, 250, 250, 210, 250}
08490      ,{ 280, 280, 280, 180, 280}
08491      ,{ 250, 250, 190, 100, 190}
08492      }
08493      }
08494      ,{{{ 280, 230, 280, 230, 250}
08495      ,{ 280, 230, 280, 230, 250}
08496      ,{ 280, 230, 280, 230, 250}
08497      ,{ 280, 230, 280, 230, 250}
```

```
08498     , { 280, 230, 280, 230, 250}
08499     }
08500     , { { 280, 230, 280, 230, 250}
08501     , { 280, 230, 280, 230, 250}
08502     , { 230, 190, 230, 190, 200}
08503     , { 230, 130, 230, 130, 200}
08504     , { 230, 190, 230, 190, 200}
08505     }
08506     , { { 280, 230, 280, 230, 250}
08507     , { 280, 230, 280, 230, 250}
08508     , { 280, 230, 280, 230, 250}
08509     , { 280, 230, 280, 230, 250}
08510     , { 280, 230, 280, 230, 250}
08511     }
08512     , { { 240, 190, 240, 190, 210}
08513     , { 240, 140, 240, 140, 210}
08514     , { 230, 190, 230, 190, 200}
08515     , { 120, 80, 120, 80, 90}
08516     , { 230, 190, 230, 190, 200}
08517     }
08518     , { { 280, 230, 280, 230, 250}
08519     , { 280, 230, 280, 230, 250}
08520     , { 250, 200, 250, 200, 220}
08521     , { 280, 230, 280, 230, 250}
08522     , { 190, 150, 190, 150, 160}
08523     }
08524     }
08525     , { { { 280, 230, 280, 230, 280}
08526     , { 280, 170, 280, 160, 280}
08527     , { 280, 230, 280, 120, 280}
08528     , { 280, 170, 280, 230, 280}
08529     , { 280, 230, 280, 170, 280}
08530     }
08531     , { { 280, 170, 280, 150, 280}
08532     , { 280, 170, 280, 120, 280}
08533     , { 230, 130, 230, 80, 230}
08534     , { 170, 70, 170, 150, 170}
08535     , { 230, 130, 230, 80, 230}
08536     }
08537     , { { 280, 230, 280, 120, 280}
08538     , { 280, 170, 280, 120, 280}
08539     , { 280, 230, 280, 120, 280}
08540     , { 280, 170, 280, 120, 280}
08541     , { 280, 230, 280, 120, 280}
08542     }
08543     , { { 230, 150, 230, 230, 230}
08544     , { 180, 80, 180, 160, 180}
08545     , { 230, 130, 230, 80, 230}
08546     , { 230, 150, 120, 230, 120}
08547     , { 230, 130, 230, 80, 230}
08548     }
08549     , { { 280, 200, 280, 170, 280}
08550     , { 280, 170, 280, 120, 280}
08551     , { 250, 200, 250, 90, 250}
08552     , { 280, 170, 280, 120, 280}
08553     , { 190, 90, 190, 170, 190}
08554     }
08555     }
08556     , { { { 280, 230, 280, 230, 250}
08557     , { 280, 230, 280, 230, 250}
08558     , { 280, 230, 280, 230, 190}
08559     , { 280, 230, 280, 230, 190}
08560     , { 280, 230, 280, 230, 190}
08561     }
08562     , { { 280, 230, 280, 230, 250}
08563     , { 280, 230, 280, 230, 250}
08564     , { 230, 190, 230, 190, 150}
08565     , { 230, 130, 230, 130, 90}
08566     , { 230, 190, 230, 190, 150}
08567     }
08568     , { { 280, 230, 280, 230, 190}
08569     , { 280, 230, 280, 230, 190}
08570     , { 280, 230, 280, 230, 190}
08571     , { 280, 230, 280, 230, 190}
08572     , { 280, 230, 280, 230, 190}
08573     }
08574     , { { 240, 190, 240, 190, 170}
08575     , { 240, 140, 240, 140, 100}
08576     , { 230, 190, 230, 190, 150}
08577     , { 170, 80, 120, 80, 170}
08578     , { 230, 190, 230, 190, 150}
08579     }
08580     , { { 280, 230, 280, 230, 190}
08581     , { 280, 230, 280, 230, 190}
08582     , { 250, 200, 250, 200, 160}
08583     , { 280, 230, 280, 230, 190}
08584     , { 190, 150, 190, 150, 110}
```

```
08585     }
08586     }
08587     }
08588     ,{{{ 370, 370, 370, 300, 340}
08589     ,{ 370, 340, 370, 300, 340}
08590     ,{ 310, 310, 310, 270, 310}
08591     ,{ 310, 310, 310, 280, 310}
08592     ,{ 370, 370, 310, 280, 310}
08593     }
08594     ,{{ 340, 340, 340, 300, 340}
08595     ,{ 340, 340, 340, 300, 340}
08596     ,{ 310, 310, 310, 260, 310}
08597     ,{ 290, 230, 290, 200, 260}
08598     ,{ 310, 310, 310, 260, 310}
08599     }
08600     ,{{ 310, 310, 310, 270, 310}
08601     ,{ 310, 310, 310, 260, 310}
08602     ,{ 310, 310, 310, 270, 310}
08603     ,{ 310, 310, 310, 260, 310}
08604     ,{ 310, 310, 310, 270, 310}
08605     }
08606     ,{{ 370, 310, 370, 280, 340}
08607     ,{ 370, 310, 370, 280, 340}
08608     ,{ 310, 310, 310, 260, 310}
08609     ,{ 280, 200, 180, 280, 220}
08610     ,{ 310, 310, 310, 260, 310}
08611     }
08612     ,{{ 370, 370, 310, 280, 310}
08613     ,{ 310, 310, 310, 260, 310}
08614     ,{ 310, 310, 310, 270, 310}
08615     ,{ 310, 310, 310, 260, 310}
08616     ,{ 370, 370, 310, 280, 310}
08617     }
08618     }
08619     ,{{{ 370, 370, 340, 270, 340}
08620     ,{ 340, 340, 340, 250, 340}
08621     ,{ 310, 310, 310, 270, 310}
08622     ,{ 310, 310, 310, 210, 310}
08623     ,{ 370, 370, 310, 270, 310}
08624     }
08625     ,{{{ 340, 340, 340, 250, 340}
08626     ,{ 340, 340, 340, 250, 340}
08627     ,{ 310, 310, 310, 210, 310}
08628     ,{ 230, 230, 230, 130, 230}
08629     ,{ 310, 310, 310, 210, 310}
08630     }
08631     ,{{{ 310, 310, 310, 270, 310}
08632     ,{ 310, 310, 310, 210, 310}
08633     ,{ 310, 310, 310, 270, 310}
08634     ,{ 310, 310, 310, 210, 310}
08635     ,{ 310, 310, 310, 270, 310}
08636     }
08637     ,{{{ 310, 310, 310, 210, 310}
08638     ,{ 310, 310, 310, 210, 310}
08639     ,{ 310, 310, 310, 210, 310}
08640     ,{ 210, 180, 180, 210, 180}
08641     ,{ 310, 310, 310, 210, 310}
08642     }
08643     ,{{{ 370, 370, 310, 270, 310}
08644     ,{ 310, 310, 310, 210, 310}
08645     ,{ 310, 310, 310, 270, 310}
08646     ,{ 310, 310, 310, 210, 310}
08647     ,{ 370, 370, 310, 210, 310}
08648     }
08649     }
08650     ,{{{ 370, 300, 370, 300, 340}
08651     ,{ 370, 300, 370, 300, 340}
08652     ,{ 310, 260, 310, 260, 280}
08653     ,{ 310, 260, 310, 260, 280}
08654     ,{ 310, 260, 310, 260, 280}
08655     }
08656     ,{{{ 340, 300, 340, 300, 310}
08657     ,{ 340, 300, 340, 300, 310}
08658     ,{ 310, 260, 310, 260, 280}
08659     ,{ 290, 180, 290, 180, 260}
08660     ,{ 310, 260, 310, 260, 280}
08661     }
08662     ,{{{ 310, 260, 310, 260, 280}
08663     ,{ 310, 260, 310, 260, 280}
08664     ,{ 310, 260, 310, 260, 280}
08665     ,{ 310, 260, 310, 260, 280}
08666     ,{ 310, 260, 310, 260, 280}
08667     }
08668     ,{{{ 370, 260, 370, 260, 340}
08669     ,{ 370, 260, 370, 260, 340}
08670     ,{ 310, 260, 310, 260, 280}
08671     ,{ 180, 130, 180, 130, 150}
```

```
08672     , { 310, 260, 310, 260, 280}
08673     }
08674     , {{ 310, 260, 310, 260, 280}
08675     , { 310, 260, 310, 260, 280}
08676     , { 310, 260, 310, 260, 280}
08677     , { 310, 260, 310, 260, 280}
08678     , { 310, 260, 310, 260, 280}
08679     }
08680     }
08681     , {{{ 340, 260, 340, 280, 340}
08682     , { 340, 240, 340, 280, 340}
08683     , { 310, 260, 310, 150, 310}
08684     , { 310, 200, 310, 280, 310}
08685     , { 310, 260, 310, 280, 310}
08686     }
08687     , {{ 340, 240, 340, 200, 340}
08688     , { 340, 240, 340, 190, 340}
08689     , { 310, 200, 310, 150, 310}
08690     , { 230, 120, 230, 200, 230}
08691     , { 310, 200, 310, 150, 310}
08692     }
08693     , {{ 310, 260, 310, 150, 310}
08694     , { 310, 200, 310, 150, 310}
08695     , { 310, 260, 310, 150, 310}
08696     , { 310, 200, 310, 150, 310}
08697     , { 310, 260, 310, 150, 310}
08698     }
08699     , {{ 310, 200, 310, 280, 310}
08700     , { 310, 200, 310, 280, 310}
08701     , { 310, 200, 310, 150, 310}
08702     , { 280, 200, 180, 280, 180}
08703     , { 310, 200, 310, 150, 310}
08704     }
08705     , {{ 310, 260, 310, 280, 310}
08706     , { 310, 200, 310, 150, 310}
08707     , { 310, 260, 310, 150, 310}
08708     , { 310, 200, 310, 150, 310}
08709     , { 310, 200, 310, 280, 310}
08710     }
08711     }
08712     , {{{ 370, 300, 370, 300, 320}
08713     , { 370, 300, 370, 300, 320}
08714     , { 310, 260, 310, 260, 220}
08715     , { 310, 260, 310, 260, 220}
08716     , { 310, 260, 310, 260, 220}
08717     }
08718     , {{{ 340, 300, 340, 300, 320}
08719     , { 340, 300, 340, 300, 320}
08720     , { 310, 260, 310, 260, 220}
08721     , { 290, 180, 290, 180, 140}
08722     , { 310, 260, 310, 260, 220}
08723     }
08724     , {{{ 310, 260, 310, 260, 220}
08725     , { 310, 260, 310, 260, 220}
08726     , { 310, 260, 310, 260, 220}
08727     , { 310, 260, 310, 260, 220}
08728     , { 310, 260, 310, 260, 220}
08729     }
08730     , {{{ 370, 260, 370, 260, 220}
08731     , { 370, 260, 370, 260, 220}
08732     , { 310, 260, 310, 260, 220}
08733     , { 220, 130, 180, 130, 220}
08734     , { 310, 260, 310, 260, 220}
08735     }
08736     , {{{ 310, 260, 310, 260, 220}
08737     , { 310, 260, 310, 260, 220}
08738     , { 310, 260, 310, 260, 220}
08739     , { 310, 260, 310, 260, 220}
08740     , { 310, 260, 310, 260, 220}
08741     }
08742     }
08743     }
08744     }
08745     , {{{ { INF, INF, INF, INF, INF}
08746     , { INF, INF, INF, INF, INF}
08747     , { INF, INF, INF, INF, INF}
08748     , { INF, INF, INF, INF, INF}
08749     , { INF, INF, INF, INF, INF}
08750     }
08751     , {{{ INF, INF, INF, INF, INF}
08752     , { INF, INF, INF, INF, INF}
08753     , { INF, INF, INF, INF, INF}
08754     , { INF, INF, INF, INF, INF}
08755     , { INF, INF, INF, INF, INF}
08756     }
08757     , {{{ INF, INF, INF, INF, INF}
08758     , { INF, INF, INF, INF, INF}
```

```
08759      , {   INF,   INF,   INF,   INF,   INF }
08760      , {   INF,   INF,   INF,   INF,   INF }
08761      , {   INF,   INF,   INF,   INF,   INF }
08762      }
08763      , { {   INF,   INF,   INF,   INF,   INF }
08764      , {   INF,   INF,   INF,   INF,   INF }
08765      , {   INF,   INF,   INF,   INF,   INF }
08766      , {   INF,   INF,   INF,   INF,   INF }
08767      , {   INF,   INF,   INF,   INF,   INF }
08768      }
08769      , { {   INF,   INF,   INF,   INF,   INF }
08770      , {   INF,   INF,   INF,   INF,   INF }
08771      , {   INF,   INF,   INF,   INF,   INF }
08772      , {   INF,   INF,   INF,   INF,   INF }
08773      , {   INF,   INF,   INF,   INF,   INF }
08774      }
08775      }
08776      , { { {   INF,   INF,   INF,   INF,   INF }
08777      , {   INF,   INF,   INF,   INF,   INF }
08778      , {   INF,   INF,   INF,   INF,   INF }
08779      , {   INF,   INF,   INF,   INF,   INF }
08780      , {   INF,   INF,   INF,   INF,   INF }
08781      }
08782      , { {   INF,   INF,   INF,   INF,   INF }
08783      , {   INF,   INF,   INF,   INF,   INF }
08784      , {   INF,   INF,   INF,   INF,   INF }
08785      , {   INF,   INF,   INF,   INF,   INF }
08786      , {   INF,   INF,   INF,   INF,   INF }
08787      }
08788      , { {   INF,   INF,   INF,   INF,   INF }
08789      , {   INF,   INF,   INF,   INF,   INF }
08790      , {   INF,   INF,   INF,   INF,   INF }
08791      , {   INF,   INF,   INF,   INF,   INF }
08792      , {   INF,   INF,   INF,   INF,   INF }
08793      }
08794      , { {   INF,   INF,   INF,   INF,   INF }
08795      , {   INF,   INF,   INF,   INF,   INF }
08796      , {   INF,   INF,   INF,   INF,   INF }
08797      , {   INF,   INF,   INF,   INF,   INF }
08798      , {   INF,   INF,   INF,   INF,   INF }
08799      }
08800      , { {   INF,   INF,   INF,   INF,   INF }
08801      , {   INF,   INF,   INF,   INF,   INF }
08802      , {   INF,   INF,   INF,   INF,   INF }
08803      , {   INF,   INF,   INF,   INF,   INF }
08804      , {   INF,   INF,   INF,   INF,   INF }
08805      }
08806      }
08807      , { { {   INF,   INF,   INF,   INF,   INF }
08808      , {   INF,   INF,   INF,   INF,   INF }
08809      , {   INF,   INF,   INF,   INF,   INF }
08810      , {   INF,   INF,   INF,   INF,   INF }
08811      , {   INF,   INF,   INF,   INF,   INF }
08812      }
08813      , { {   INF,   INF,   INF,   INF,   INF }
08814      , {   INF,   INF,   INF,   INF,   INF }
08815      , {   INF,   INF,   INF,   INF,   INF }
08816      , {   INF,   INF,   INF,   INF,   INF }
08817      , {   INF,   INF,   INF,   INF,   INF }
08818      }
08819      , { {   INF,   INF,   INF,   INF,   INF }
08820      , {   INF,   INF,   INF,   INF,   INF }
08821      , {   INF,   INF,   INF,   INF,   INF }
08822      , {   INF,   INF,   INF,   INF,   INF }
08823      , {   INF,   INF,   INF,   INF,   INF }
08824      }
08825      , { {   INF,   INF,   INF,   INF,   INF }
08826      , {   INF,   INF,   INF,   INF,   INF }
08827      , {   INF,   INF,   INF,   INF,   INF }
08828      , {   INF,   INF,   INF,   INF,   INF }
08829      , {   INF,   INF,   INF,   INF,   INF }
08830      }
08831      , { {   INF,   INF,   INF,   INF,   INF }
08832      , {   INF,   INF,   INF,   INF,   INF }
08833      , {   INF,   INF,   INF,   INF,   INF }
08834      , {   INF,   INF,   INF,   INF,   INF }
08835      , {   INF,   INF,   INF,   INF,   INF }
08836      }
08837      }
08838      , { { {   INF,   INF,   INF,   INF,   INF }
08839      , {   INF,   INF,   INF,   INF,   INF }
08840      , {   INF,   INF,   INF,   INF,   INF }
08841      , {   INF,   INF,   INF,   INF,   INF }
08842      , {   INF,   INF,   INF,   INF,   INF }
08843      }
08844      , { {   INF,   INF,   INF,   INF,   INF }
08845      , {   INF,   INF,   INF,   INF,   INF }
```

```
08846     , {   INF,   INF,   INF,   INF,   INF }
08847     , {   INF,   INF,   INF,   INF,   INF }
08848     , {   INF,   INF,   INF,   INF,   INF }
08849     }
08850     , { {   INF,   INF,   INF,   INF,   INF }
08851     , {   INF,   INF,   INF,   INF,   INF }
08852     , {   INF,   INF,   INF,   INF,   INF }
08853     , {   INF,   INF,   INF,   INF,   INF }
08854     , {   INF,   INF,   INF,   INF,   INF }
08855     }
08856     , { {   INF,   INF,   INF,   INF,   INF }
08857     , {   INF,   INF,   INF,   INF,   INF }
08858     , {   INF,   INF,   INF,   INF,   INF }
08859     , {   INF,   INF,   INF,   INF,   INF }
08860     , {   INF,   INF,   INF,   INF,   INF }
08861     }
08862     , { {   INF,   INF,   INF,   INF,   INF }
08863     , {   INF,   INF,   INF,   INF,   INF }
08864     , {   INF,   INF,   INF,   INF,   INF }
08865     , {   INF,   INF,   INF,   INF,   INF }
08866     , {   INF,   INF,   INF,   INF,   INF }
08867     }
08868     }
08869     , { { {   INF,   INF,   INF,   INF,   INF }
08870     , {   INF,   INF,   INF,   INF,   INF }
08871     , {   INF,   INF,   INF,   INF,   INF }
08872     , {   INF,   INF,   INF,   INF,   INF }
08873     , {   INF,   INF,   INF,   INF,   INF }
08874     }
08875     , { {   INF,   INF,   INF,   INF,   INF }
08876     , {   INF,   INF,   INF,   INF,   INF }
08877     , {   INF,   INF,   INF,   INF,   INF }
08878     , {   INF,   INF,   INF,   INF,   INF }
08879     , {   INF,   INF,   INF,   INF,   INF }
08880     }
08881     , { {   INF,   INF,   INF,   INF,   INF }
08882     , {   INF,   INF,   INF,   INF,   INF }
08883     , {   INF,   INF,   INF,   INF,   INF }
08884     , {   INF,   INF,   INF,   INF,   INF }
08885     , {   INF,   INF,   INF,   INF,   INF }
08886     }
08887     , { {   INF,   INF,   INF,   INF,   INF }
08888     , {   INF,   INF,   INF,   INF,   INF }
08889     , {   INF,   INF,   INF,   INF,   INF }
08890     , {   INF,   INF,   INF,   INF,   INF }
08891     , {   INF,   INF,   INF,   INF,   INF }
08892     }
08893     , { {   INF,   INF,   INF,   INF,   INF }
08894     , {   INF,   INF,   INF,   INF,   INF }
08895     , {   INF,   INF,   INF,   INF,   INF }
08896     , {   INF,   INF,   INF,   INF,   INF }
08897     , {   INF,   INF,   INF,   INF,   INF }
08898     }
08899     }
08900     }
08901     , { { { {   310,   300,   270,   310,   290 }
08902     , {   300,   300,   270,   270,   290 }
08903     , {   310,   290,   250,   310,   250 }
08904     , {   300,   300,   270,   270,   270 }
08905     , {   300,   270,   240,   300,   240 }
08906     }
08907     , { {   290,   270,   230,   230,   290 }
08908     , {   290,   270,   230,   230,   290 }
08909     , {   260,   260,   220,   220,   220 }
08910     , {   190,   170,   190,   130,   190 }
08911     , {   260,   260,   220,   220,   220 }
08912     }
08913     , { {   310,   300,   270,   310,   270 }
08914     , {   300,   300,   270,   270,   270 }
08915     , {   310,   290,   250,   310,   250 }
08916     , {   300,   300,   270,   270,   270 }
08917     , {   300,   270,   240,   300,   240 }
08918     }
08919     , { {   260,   260,   220,   220,   220 }
08920     , {   190,   170,   190,   130,   190 }
08921     , {   260,   260,   220,   220,   220 }
08922     , {   210,   130,   80,   210,   210 }
08923     , {   260,   260,   220,   220,   220 }
08924     }
08925     , { {   300,   300,   270,   300,   270 }
08926     , {   300,   300,   270,   270,   270 }
08927     , {   300,   270,   240,   300,   240 }
08928     , {   300,   300,   270,   270,   270 }
08929     , {   240,   240,   150,   150,   150 }
08930     }
08931     }
08932     , { { {   310,   300,   270,   310,   270 }
```

```
08933      , { 300, 300, 270, 270, 270}
08934      , { 310, 290, 250, 310, 250}
08935      , { 300, 300, 270, 270, 270}
08936      , { 300, 270, 240, 300, 240}
08937      }
08938      , { { 270, 270, 230, 230, 230}
08939      , { 270, 270, 230, 230, 230}
08940      , { 260, 260, 220, 220, 220}
08941      , { 170, 170, 130, 130, 130}
08942      , { 260, 260, 220, 220, 220}
08943      }
08944      , { { 310, 300, 270, 310, 270}
08945      , { 300, 300, 270, 270, 270}
08946      , { 310, 290, 250, 310, 250}
08947      , { 300, 300, 270, 270, 270}
08948      , { 300, 270, 240, 300, 240}
08949      }
08950      , { { 260, 260, 220, 220, 220}
08951      , { 170, 170, 130, 130, 130}
08952      , { 260, 260, 220, 220, 220}
08953      , { 210, 110, 80, 210, 80}
08954      , { 260, 260, 220, 220, 220}
08955      }
08956      , { { 300, 300, 270, 300, 270}
08957      , { 300, 300, 270, 270, 270}
08958      , { 300, 270, 240, 300, 240}
08959      , { 300, 300, 270, 270, 270}
08960      , { 240, 240, 150, 150, 150}
08961      }
08962      }
08963      , { { { 270, 270, 270, 270, 270}
08964      , { 270, 270, 270, 270, 270}
08965      , { 250, 250, 250, 250, 250}
08966      , { 270, 270, 270, 270, 270}
08967      , { 240, 240, 240, 240, 240}
08968      }
08969      , { { 230, 230, 230, 230, 230}
08970      , { 230, 230, 230, 230, 230}
08971      , { 220, 220, 220, 220, 220}
08972      , { 190, 130, 190, 130, 190}
08973      , { 220, 220, 220, 220, 220}
08974      }
08975      , { { 270, 270, 270, 270, 270}
08976      , { 270, 270, 270, 270, 270}
08977      , { 250, 250, 250, 250, 250}
08978      , { 270, 270, 270, 270, 270}
08979      , { 240, 240, 240, 240, 240}
08980      }
08981      , { { 220, 220, 220, 220, 220}
08982      , { 190, 130, 190, 130, 190}
08983      , { 220, 220, 220, 220, 220}
08984      , { 80, 80, 80, 80, 80}
08985      , { 220, 220, 220, 220, 220}
08986      }
08987      , { { 270, 270, 270, 270, 270}
08988      , { 270, 270, 270, 270, 270}
08989      , { 240, 240, 240, 240, 240}
08990      , { 270, 270, 270, 270, 270}
08991      , { 150, 150, 150, 150, 150}
08992      }
08993      }
08994      , { { { 270, 230, 270, 210, 270}
08995      , { 270, 190, 270, 140, 270}
08996      , { 250, 230, 250, 120, 250}
08997      , { 270, 190, 270, 210, 270}
08998      , { 240, 220, 240, 150, 240}
08999      }
09000      , { { 230, 150, 230, 130, 230}
09001      , { 230, 150, 230, 100, 230}
09002      , { 220, 140, 220, 90, 220}
09003      , { 130, 50, 130, 130, 130}
09004      , { 220, 140, 220, 90, 220}
09005      }
09006      , { { 270, 230, 270, 140, 270}
09007      , { 270, 190, 270, 140, 270}
09008      , { 250, 230, 250, 120, 250}
09009      , { 270, 190, 270, 140, 270}
09010      , { 240, 220, 240, 110, 240}
09011      }
09012      , { { 220, 140, 220, 210, 220}
09013      , { 130, 50, 130, 130, 130}
09014      , { 220, 140, 220, 90, 220}
09015      , { 210, 130, 80, 210, 80}
09016      , { 220, 140, 220, 90, 220}
09017      }
09018      , { { 270, 220, 270, 150, 270}
09019      , { 270, 190, 270, 140, 270}
```



```
09020     , { 240, 220, 240, 110, 240}
09021     , { 270, 190, 270, 140, 270}
09022     , { 150, 70, 150, 150, 150}
09023     }
09024     }
09025     , {{ { 290, 270, 270, 270, 290}
09026     , { 290, 270, 270, 270, 290}
09027     , { 250, 250, 250, 250, 250}
09028     , { 270, 270, 270, 270, 270}
09029     , { 240, 240, 240, 240, 240}
09030     }
09031     , {{ { 290, 230, 230, 230, 290}
09032     , { 290, 230, 230, 230, 290}
09033     , { 220, 220, 220, 220, 220}
09034     , { 190, 130, 190, 130, 130}
09035     , { 220, 220, 220, 220, 220}
09036     }
09037     , {{ { 270, 270, 270, 270, 270}
09038     , { 270, 270, 270, 270, 270}
09039     , { 250, 250, 250, 250, 250}
09040     , { 270, 270, 270, 270, 270}
09041     , { 240, 240, 240, 240, 240}
09042     }
09043     , {{ { 220, 220, 220, 220, 220}
09044     , { 190, 130, 190, 130, 130}
09045     , { 220, 220, 220, 220, 220}
09046     , { 210, 80, 80, 80, 210}
09047     , { 220, 220, 220, 220, 220}
09048     }
09049     , {{ { 270, 270, 270, 270, 270}
09050     , { 270, 270, 270, 270, 270}
09051     , { 240, 240, 240, 240, 240}
09052     , { 270, 270, 270, 270, 270}
09053     , { 150, 150, 150, 150, 150}
09054     }
09055     }
09056     }
09057     , {{{ { 300, 280, 240, 280, 300}
09058     , { 300, 280, 240, 240, 300}
09059     , { 280, 260, 220, 280, 220}
09060     , { 250, 250, 210, 210, 210}
09061     , { 280, 250, 220, 280, 220}
09062     }
09063     , {{ { 300, 280, 240, 240, 300}
09064     , { 300, 280, 240, 240, 300}
09065     , { 250, 250, 220, 220, 220}
09066     , { 100, 70, 100, 40, 100}
09067     , { 250, 250, 220, 220, 220}
09068     }
09069     , {{ { 280, 250, 220, 280, 220}
09070     , { 250, 250, 210, 210, 210}
09071     , { 280, 250, 220, 280, 220}
09072     , { 250, 250, 210, 210, 210}
09073     , { 280, 250, 220, 280, 220}
09074     }
09075     , {{ { 250, 250, 220, 220, 220}
09076     , { 160, 140, 160, 100, 160}
09077     , { 250, 250, 220, 220, 220}
09078     , { 210, 130, 80, 210, 210}
09079     , { 250, 250, 220, 220, 220}
09080     }
09081     , {{ { 280, 260, 220, 280, 220}
09082     , { 250, 250, 210, 210, 210}
09083     , { 280, 260, 220, 280, 220}
09084     , { 250, 250, 210, 210, 210}
09085     , { 240, 240, 140, 140, 140}
09086     }
09087     }
09088     , {{{ { 280, 280, 240, 280, 240}
09089     , { 280, 280, 240, 240, 240}
09090     , { 280, 260, 220, 280, 220}
09091     , { 250, 250, 210, 210, 210}
09092     , { 280, 250, 220, 280, 220}
09093     }
09094     , {{ { 280, 280, 240, 240, 240}
09095     , { 280, 280, 240, 240, 240}
09096     , { 250, 250, 220, 220, 220}
09097     , { 70, 70, 40, 40, 40}
09098     , { 250, 250, 220, 220, 220}
09099     }
09100     , {{ { 280, 250, 220, 280, 220}
09101     , { 250, 250, 210, 210, 210}
09102     , { 280, 250, 220, 280, 220}
09103     , { 250, 250, 210, 210, 210}
09104     , { 280, 250, 220, 280, 220}
09105     }
09106     , {{ { 250, 250, 220, 220, 220}
```

```
09107      , { 140, 140, 100, 100, 100}
09108      , { 250, 250, 220, 220, 220}
09109      , { 210, 110, 80, 210, 80}
09110      , { 250, 250, 220, 220, 220}
09111      }
09112      , { { 280, 260, 220, 280, 220}
09113      , { 250, 250, 210, 210, 210}
09114      , { 280, 260, 220, 280, 220}
09115      , { 250, 250, 210, 210, 210}
09116      , { 240, 240, 140, 140, 140}
09117      }
09118      }
09119      , { { { 240, 240, 240, 240, 240}
09120      , { 240, 240, 240, 240, 240}
09121      , { 220, 220, 220, 220, 220}
09122      , { 210, 210, 210, 210, 210}
09123      , { 220, 220, 220, 220, 220}
09124      }
09125      , { { 240, 240, 240, 240, 240}
09126      , { 240, 240, 240, 240, 240}
09127      , { 220, 220, 220, 220, 220}
09128      , { 100, 40, 100, 40, 100}
09129      , { 220, 220, 220, 220, 220}
09130      }
09131      , { { 220, 220, 220, 220, 220}
09132      , { 210, 210, 210, 210, 210}
09133      , { 220, 220, 220, 220, 220}
09134      , { 210, 210, 210, 210, 210}
09135      , { 220, 220, 220, 220, 220}
09136      }
09137      , { { 220, 220, 220, 220, 220}
09138      , { 160, 100, 160, 100, 160}
09139      , { 220, 220, 220, 220, 220}
09140      , { 80, 80, 80, 80, 80}
09141      , { 220, 220, 220, 220, 220}
09142      }
09143      , { { 220, 220, 220, 220, 220}
09144      , { 210, 210, 210, 210, 210}
09145      , { 220, 220, 220, 220, 220}
09146      , { 210, 210, 210, 210, 210}
09147      , { 140, 140, 140, 140, 140}
09148      }
09149      }
09150      , { { { 240, 200, 240, 210, 240}
09151      , { 240, 160, 240, 110, 240}
09152      , { 220, 200, 220, 90, 220}
09153      , { 210, 130, 210, 210, 210}
09154      , { 220, 200, 220, 140, 220}
09155      }
09156      , { { 240, 160, 240, 110, 240}
09157      , { 240, 160, 240, 110, 240}
09158      , { 220, 140, 220, 90, 220}
09159      , { 40, -40, 40, 40, 40}
09160      , { 220, 140, 220, 90, 220}
09161      }
09162      , { { 220, 200, 220, 90, 220}
09163      , { 210, 130, 210, 80, 210}
09164      , { 220, 200, 220, 90, 220}
09165      , { 210, 130, 210, 80, 210}
09166      , { 220, 200, 220, 90, 220}
09167      }
09168      , { { 220, 140, 220, 210, 220}
09169      , { 100, 20, 100, 100, 100}
09170      , { 220, 140, 220, 90, 220}
09171      , { 210, 130, 80, 210, 80}
09172      , { 220, 140, 220, 90, 220}
09173      }
09174      , { { 220, 200, 220, 140, 220}
09175      , { 210, 130, 210, 80, 210}
09176      , { 220, 200, 220, 90, 220}
09177      , { 210, 130, 210, 80, 210}
09178      , { 140, 90, 140, 140, 140}
09179      }
09180      }
09181      , { { { 300, 240, 240, 240, 300}
09182      , { 300, 240, 240, 240, 300}
09183      , { 220, 220, 220, 220, 220}
09184      , { 210, 210, 210, 210, 210}
09185      , { 220, 220, 220, 220, 220}
09186      }
09187      , { { 300, 240, 240, 240, 300}
09188      , { 300, 240, 240, 240, 300}
09189      , { 220, 220, 220, 220, 220}
09190      , { 100, 40, 100, 40, 50}
09191      , { 220, 220, 220, 220, 220}
09192      }
09193      , { { 220, 220, 220, 220, 220}
```

```
09194     , { 210, 210, 210, 210, 210}
09195     , { 220, 220, 220, 220, 220}
09196     , { 210, 210, 210, 210, 210}
09197     , { 220, 220, 220, 220, 220}
09198     }
09199     , { { 220, 220, 220, 220, 220}
09200     , { 160, 100, 160, 100, 140}
09201     , { 220, 220, 220, 220, 220}
09202     , { 210, 80, 80, 80, 210}
09203     , { 220, 220, 220, 220, 220}
09204     }
09205     , { { 220, 220, 220, 220, 220}
09206     , { 210, 210, 210, 210, 210}
09207     , { 220, 220, 220, 220, 220}
09208     , { 210, 210, 210, 210, 210}
09209     , { 140, 140, 140, 140, 140}
09210     }
09211     }
09212     }
09213     , { { { 430, 430, 370, 400, 430}
09214     , { 430, 410, 370, 370, 430}
09215     , { 400, 370, 340, 400, 340}
09216     , { 370, 370, 340, 340, 340}
09217     , { 430, 430, 340, 400, 340}
09218     }
09219     , { { 430, 410, 370, 370, 430}
09220     , { 430, 410, 370, 370, 430}
09221     , { 370, 370, 340, 340, 340}
09222     , { 320, 290, 320, 260, 320}
09223     , { 370, 370, 340, 340, 340}
09224     }
09225     , { { 400, 370, 340, 400, 340}
09226     , { 370, 370, 340, 340, 340}
09227     , { 400, 370, 340, 400, 340}
09228     , { 370, 370, 340, 340, 340}
09229     , { 400, 370, 340, 400, 340}
09230     }
09231     , { { 370, 370, 360, 340, 360}
09232     , { 360, 360, 360, 300, 360}
09233     , { 370, 370, 340, 340, 340}
09234     , { 340, 260, 210, 340, 340}
09235     , { 370, 370, 340, 340, 340}
09236     }
09237     , { { 430, 430, 340, 400, 340}
09238     , { 370, 370, 340, 340, 340}
09239     , { 400, 370, 340, 400, 340}
09240     , { 370, 370, 340, 340, 340}
09241     , { 430, 430, 340, 340, 340}
09242     }
09243     }
09244     , { { { 430, 430, 370, 400, 370}
09245     , { 410, 410, 370, 370, 370}
09246     , { 400, 370, 340, 400, 340}
09247     , { 370, 370, 340, 340, 340}
09248     , { 430, 430, 340, 400, 340}
09249     }
09250     , { { 410, 410, 370, 370, 370}
09251     , { 410, 410, 370, 370, 370}
09252     , { 370, 370, 340, 340, 340}
09253     , { 290, 290, 260, 260, 260}
09254     , { 370, 370, 340, 340, 340}
09255     }
09256     , { { 400, 370, 340, 400, 340}
09257     , { 370, 370, 340, 340, 340}
09258     , { 400, 370, 340, 400, 340}
09259     , { 370, 370, 340, 340, 340}
09260     , { 400, 370, 340, 400, 340}
09261     }
09262     , { { 370, 370, 340, 340, 340}
09263     , { 360, 360, 300, 300, 300}
09264     , { 370, 370, 340, 340, 340}
09265     , { 340, 240, 210, 340, 210}
09266     , { 370, 370, 340, 340, 340}
09267     }
09268     , { { 430, 430, 340, 400, 340}
09269     , { 370, 370, 340, 340, 340}
09270     , { 400, 370, 340, 400, 340}
09271     , { 370, 370, 340, 340, 340}
09272     , { 430, 430, 340, 340, 340}
09273     }
09274     }
09275     , { { { 370, 370, 370, 370, 370}
09276     , { 370, 370, 370, 370, 370}
09277     , { 340, 340, 340, 340, 340}
09278     , { 340, 340, 340, 340, 340}
09279     , { 340, 340, 340, 340, 340}
09280     }
```

```
09281 ,{{ 370, 370, 370, 370, 370}
09282 ,{ 370, 370, 370, 370, 370}
09283 ,{ 340, 340, 340, 340, 340}
09284 ,{ 320, 260, 320, 260, 320}
09285 ,{ 340, 340, 340, 340, 340}
09286 }
09287 ,{{ 340, 340, 340, 340, 340}
09288 ,{ 340, 340, 340, 340, 340}
09289 ,{ 340, 340, 340, 340, 340}
09290 ,{ 340, 340, 340, 340, 340}
09291 ,{ 340, 340, 340, 340, 340}
09292 }
09293 ,{{ 360, 340, 360, 340, 360}
09294 ,{ 360, 300, 360, 300, 360}
09295 ,{ 340, 340, 340, 340, 340}
09296 ,{ 210, 210, 210, 210, 210}
09297 ,{ 340, 340, 340, 340, 340}
09298 }
09299 ,{{ 340, 340, 340, 340, 340}
09300 ,{ 340, 340, 340, 340, 340}
09301 ,{ 340, 340, 340, 340, 340}
09302 ,{ 340, 340, 340, 340, 340}
09303 ,{ 340, 340, 340, 340, 340}
09304 }
09305 }
09306 ,{{{ 370, 320, 370, 340, 370}
09307 ,{ 370, 290, 370, 300, 370}
09308 ,{ 340, 320, 340, 210, 340}
09309 ,{ 340, 260, 340, 340, 340}
09310 ,{ 340, 320, 340, 340, 340}
09311 }
09312 ,{{ 370, 290, 370, 260, 370}
09313 ,{ 370, 290, 370, 240, 370}
09314 ,{ 340, 260, 340, 210, 340}
09315 ,{ 260, 180, 260, 260, 260}
09316 ,{ 340, 260, 340, 210, 340}
09317 }
09318 ,{{ 340, 320, 340, 210, 340}
09319 ,{ 340, 260, 340, 210, 340}
09320 ,{ 340, 320, 340, 210, 340}
09321 ,{ 340, 260, 340, 210, 340}
09322 ,{ 340, 320, 340, 210, 340}
09323 }
09324 ,{{ 340, 260, 340, 340, 340}
09325 ,{ 300, 220, 300, 300, 300}
09326 ,{ 340, 260, 340, 210, 340}
09327 ,{ 340, 260, 210, 340, 210}
09328 ,{ 340, 260, 340, 210, 340}
09329 }
09330 ,{{ 340, 320, 340, 340, 340}
09331 ,{ 340, 260, 340, 210, 340}
09332 ,{ 340, 320, 340, 210, 340}
09333 ,{ 340, 260, 340, 210, 340}
09334 ,{ 340, 260, 340, 340, 340}
09335 }
09336 }
09337 ,{{{ 430, 370, 370, 370, 430}
09338 ,{ 430, 370, 370, 370, 430}
09339 ,{ 340, 340, 340, 340, 340}
09340 ,{ 340, 340, 340, 340, 340}
09341 ,{ 340, 340, 340, 340, 340}
09342 }
09343 ,{{ 430, 370, 370, 370, 430}
09344 ,{ 430, 370, 370, 370, 430}
09345 ,{ 340, 340, 340, 340, 340}
09346 ,{ 320, 260, 320, 260, 260}
09347 ,{ 340, 340, 340, 340, 340}
09348 }
09349 ,{{ 340, 340, 340, 340, 340}
09350 ,{ 340, 340, 340, 340, 340}
09351 ,{ 340, 340, 340, 340, 340}
09352 ,{ 340, 340, 340, 340, 340}
09353 ,{ 340, 340, 340, 340, 340}
09354 }
09355 ,{{ 360, 340, 360, 340, 340}
09356 ,{ 360, 300, 360, 300, 300}
09357 ,{ 340, 340, 340, 340, 340}
09358 ,{ 340, 210, 210, 210, 340}
09359 ,{ 340, 340, 340, 340, 340}
09360 }
09361 ,{{ 340, 340, 340, 340, 340}
09362 ,{ 340, 340, 340, 340, 340}
09363 ,{ 340, 340, 340, 340, 340}
09364 ,{ 340, 340, 340, 340, 340}
09365 ,{ 340, 340, 340, 340, 340}
09366 }
09367 }
```

```
09368     }
09369     , {{{ 400, 400, 400, 370, 400}
09370         , { 400, 370, 400, 360, 400}
09371         , { 370, 340, 310, 370, 310}
09372         , { 340, 340, 310, 310, 310}
09373         , { 400, 400, 310, 370, 310}
09374     }
09375     , {{{ 360, 360, 310, 360, 330}
09376         , { 360, 360, 270, 360, 330}
09377         , { 340, 340, 310, 310, 310}
09378         , { 230, 220, 230, 170, 230}
09379         , { 340, 340, 310, 310, 310}
09380     }
09381     , {{{ 370, 340, 310, 370, 310}
09382         , { 340, 340, 310, 310, 310}
09383         , { 370, 340, 310, 370, 310}
09384         , { 340, 340, 310, 310, 310}
09385         , { 370, 340, 310, 370, 310}
09386     }
09387     , {{{ 400, 370, 400, 340, 400}
09388         , { 400, 370, 400, 340, 400}
09389         , { 340, 340, 310, 310, 310}
09390         , { 310, 230, 180, 310, 310}
09391         , { 340, 340, 310, 310, 310}
09392     }
09393     , {{{ 400, 400, 310, 370, 310}
09394         , { 340, 340, 310, 310, 310}
09395         , { 370, 340, 310, 370, 310}
09396         , { 340, 340, 310, 310, 310}
09397         , { 400, 400, 310, 310, 310}
09398     }
09399     }
09400     , {{{ 400, 400, 340, 370, 340}
09401         , { 370, 370, 340, 360, 340}
09402         , { 370, 340, 310, 370, 310}
09403         , { 340, 340, 310, 310, 310}
09404         , { 400, 400, 310, 370, 310}
09405     }
09406     , {{{ 360, 360, 310, 360, 310}
09407         , { 360, 360, 270, 360, 270}
09408         , { 340, 340, 310, 310, 310}
09409         , { 220, 220, 170, 170, 170}
09410         , { 340, 340, 310, 310, 310}
09411     }
09412     , {{{ 370, 340, 310, 370, 310}
09413         , { 340, 340, 310, 310, 310}
09414         , { 370, 340, 310, 370, 310}
09415         , { 340, 340, 310, 310, 310}
09416         , { 370, 340, 310, 370, 310}
09417     }
09418     , {{{ 370, 370, 340, 340, 340}
09419         , { 370, 370, 340, 340, 340}
09420         , { 340, 340, 310, 310, 310}
09421         , { 310, 210, 180, 310, 180}
09422         , { 340, 340, 310, 310, 310}
09423     }
09424     , {{{ 400, 400, 310, 370, 310}
09425         , { 340, 340, 310, 310, 310}
09426         , { 370, 340, 310, 370, 310}
09427         , { 340, 340, 310, 310, 310}
09428         , { 400, 400, 310, 310, 310}
09429     }
09430     }
09431     , {{{ 400, 340, 400, 340, 400}
09432         , { 400, 340, 400, 340, 400}
09433         , { 310, 310, 310, 310, 310}
09434         , { 310, 310, 310, 310, 310}
09435         , { 310, 310, 310, 310, 310}
09436     }
09437     , {{{ 310, 310, 310, 310, 310}
09438         , { 270, 270, 270, 270, 270}
09439         , { 310, 310, 310, 310, 310}
09440         , { 230, 170, 230, 170, 230}
09441         , { 310, 310, 310, 310, 310}
09442     }
09443     , {{{ 310, 310, 310, 310, 310}
09444         , { 310, 310, 310, 310, 310}
09445         , { 310, 310, 310, 310, 310}
09446         , { 310, 310, 310, 310, 310}
09447         , { 310, 310, 310, 310, 310}
09448     }
09449     , {{{ 400, 340, 400, 340, 400}
09450         , { 400, 340, 400, 340, 400}
09451         , { 310, 310, 310, 310, 310}
09452         , { 180, 180, 180, 180, 180}
09453         , { 310, 310, 310, 310, 310}
09454     }
```

```
09455 ,{{ 310, 310, 310, 310, 310}
09456 ,{ 310, 310, 310, 310, 310}
09457 ,{ 310, 310, 310, 310, 310}
09458 ,{ 310, 310, 310, 310, 310}
09459 ,{ 310, 310, 310, 310, 310}
09460 }
09461 }
09462 ,{{{ 340, 290, 340, 340, 340}
09463 ,{ 340, 260, 340, 340, 340}
09464 ,{ 310, 290, 310, 180, 310}
09465 ,{ 310, 230, 310, 310, 310}
09466 ,{ 310, 290, 310, 310, 310}
09467 }
09468 ,{{ 310, 230, 310, 180, 310}
09469 ,{ 270, 190, 270, 140, 270}
09470 ,{ 310, 230, 310, 180, 310}
09471 ,{ 170, 40, 170, 170, 170}
09472 ,{ 310, 230, 310, 180, 310}
09473 }
09474 ,{{ 310, 290, 310, 180, 310}
09475 ,{ 310, 230, 310, 180, 310}
09476 ,{ 310, 290, 310, 180, 310}
09477 ,{ 310, 230, 310, 180, 310}
09478 ,{ 310, 290, 310, 180, 310}
09479 }
09480 ,{{{ 340, 260, 340, 340, 340}
09481 ,{ 340, 260, 340, 340, 340}
09482 ,{ 310, 230, 310, 180, 310}
09483 ,{ 310, 230, 180, 310, 180}
09484 ,{ 310, 230, 310, 180, 310}
09485 }
09486 ,{{{ 310, 290, 310, 310, 310}
09487 ,{ 310, 230, 310, 180, 310}
09488 ,{ 310, 290, 310, 180, 310}
09489 ,{ 310, 230, 310, 180, 310}
09490 ,{ 310, 230, 310, 310, 310}
09491 }
09492 }
09493 ,{{{ 400, 340, 400, 340, 340}
09494 ,{ 400, 340, 400, 340, 340}
09495 ,{ 310, 310, 310, 310, 310}
09496 ,{ 310, 310, 310, 310, 310}
09497 ,{ 310, 310, 310, 310, 310}
09498 }
09499 ,{{{ 330, 310, 310, 310, 330}
09500 ,{ 330, 270, 270, 270, 330}
09501 ,{ 310, 310, 310, 310, 310}
09502 ,{ 230, 170, 230, 170, 170}
09503 ,{ 310, 310, 310, 310, 310}
09504 }
09505 ,{{{ 310, 310, 310, 310, 310}
09506 ,{ 310, 310, 310, 310, 310}
09507 ,{ 310, 310, 310, 310, 310}
09508 ,{ 310, 310, 310, 310, 310}
09509 ,{ 310, 310, 310, 310, 310}
09510 }
09511 ,{{{ 400, 340, 400, 340, 340}
09512 ,{ 400, 340, 400, 340, 340}
09513 ,{ 310, 310, 310, 310, 310}
09514 ,{ 310, 180, 180, 180, 310}
09515 ,{ 310, 310, 310, 310, 310}
09516 }
09517 ,{{{ 310, 310, 310, 310, 310}
09518 ,{ 310, 310, 310, 310, 310}
09519 ,{ 310, 310, 310, 310, 310}
09520 ,{ 310, 310, 310, 310, 310}
09521 ,{ 310, 310, 310, 310, 310}
09522 }
09523 }
09524 }
09525 ,{{{ 370, 340, 310, 350, 370}
09526 ,{ 370, 340, 310, 310, 370}
09527 ,{ 350, 320, 290, 350, 290}
09528 ,{ 330, 330, 290, 290, 290}
09529 ,{ 350, 320, 290, 350, 290}
09530 }
09531 ,{{{ 370, 340, 310, 310, 370}
09532 ,{ 370, 340, 310, 310, 370}
09533 ,{ 320, 320, 280, 280, 280}
09534 ,{ 240, 220, 240, 180, 240}
09535 ,{ 320, 320, 280, 280, 280}
09536 }
09537 ,{{{ 350, 330, 290, 350, 290}
09538 ,{ 330, 330, 290, 290, 290}
09539 ,{ 350, 320, 290, 350, 290}
09540 ,{ 330, 330, 290, 290, 290}
09541 ,{ 350, 320, 290, 350, 290}
```

```
09542     }
09543     ,{{ 320, 320, 310, 280, 310}
09544     ,{ 310, 290, 310, 250, 310}
09545     ,{ 320, 320, 280, 280, 280}
09546     ,{ 260, 180, 130, 260, 260}
09547     ,{ 320, 320, 280, 280, 280}
09548     }
09549     ,{{ 350, 330, 290, 350, 290}
09550     ,{ 330, 330, 290, 290, 290}
09551     ,{ 350, 320, 290, 350, 290}
09552     ,{ 330, 330, 290, 290, 290}
09553     ,{ 290, 290, 200, 200, 200}
09554     }
09555     }
09556     ,{{{ 350, 340, 310, 350, 310}
09557     ,{ 340, 340, 310, 310, 310}
09558     ,{ 350, 320, 290, 350, 290}
09559     ,{ 330, 330, 290, 290, 290}
09560     ,{ 350, 320, 290, 350, 290}
09561     }
09562     ,{{{ 340, 340, 310, 310, 310}
09563     ,{ 340, 340, 310, 310, 310}
09564     ,{ 320, 320, 280, 280, 280}
09565     ,{ 220, 220, 180, 180, 180}
09566     ,{ 320, 320, 280, 280, 280}
09567     }
09568     ,{{{ 350, 330, 290, 350, 290}
09569     ,{ 330, 330, 290, 290, 290}
09570     ,{ 350, 320, 290, 350, 290}
09571     ,{ 330, 330, 290, 290, 290}
09572     ,{ 350, 320, 290, 350, 290}
09573     }
09574     ,{{{ 320, 320, 280, 280, 280}
09575     ,{ 290, 290, 250, 250, 250}
09576     ,{ 320, 320, 280, 280, 280}
09577     ,{ 260, 170, 130, 260, 130}
09578     ,{ 320, 320, 280, 280, 280}
09579     }
09580     ,{{{ 350, 330, 290, 350, 290}
09581     ,{ 330, 330, 290, 290, 290}
09582     ,{ 350, 320, 290, 350, 290}
09583     ,{ 330, 330, 290, 290, 290}
09584     ,{ 290, 290, 200, 200, 200}
09585     }
09586     }
09587     ,{{{ 310, 310, 310, 310, 310}
09588     ,{ 310, 310, 310, 310, 310}
09589     ,{ 290, 290, 290, 290, 290}
09590     ,{ 290, 290, 290, 290, 290}
09591     ,{ 290, 290, 290, 290, 290}
09592     }
09593     ,{{{ 310, 310, 310, 310, 310}
09594     ,{ 310, 310, 310, 310, 310}
09595     ,{ 280, 280, 280, 280, 280}
09596     ,{ 240, 180, 240, 180, 240}
09597     ,{ 280, 280, 280, 280, 280}
09598     }
09599     ,{{{ 290, 290, 290, 290, 290}
09600     ,{ 290, 290, 290, 290, 290}
09601     ,{ 290, 290, 290, 290, 290}
09602     ,{ 290, 290, 290, 290, 290}
09603     ,{ 290, 290, 290, 290, 290}
09604     }
09605     ,{{{ 310, 280, 310, 280, 310}
09606     ,{ 310, 250, 310, 250, 310}
09607     ,{ 280, 280, 280, 280, 280}
09608     ,{ 130, 130, 130, 130, 130}
09609     ,{ 280, 280, 280, 280, 280}
09610     }
09611     ,{{{ 290, 290, 290, 290, 290}
09612     ,{ 290, 290, 290, 290, 290}
09613     ,{ 290, 290, 290, 290, 290}
09614     ,{ 290, 290, 290, 290, 290}
09615     ,{ 200, 200, 200, 200, 200}
09616     }
09617     }
09618     ,{{{ 310, 270, 310, 260, 310}
09619     ,{ 310, 230, 310, 250, 310}
09620     ,{ 290, 270, 290, 160, 290}
09621     ,{ 290, 210, 290, 260, 290}
09622     ,{ 290, 270, 290, 200, 290}
09623     }
09624     ,{{{ 310, 230, 310, 180, 310}
09625     ,{ 310, 230, 310, 180, 310}
09626     ,{ 280, 200, 280, 150, 280}
09627     ,{ 180, 100, 180, 180, 180}
09628     ,{ 280, 200, 280, 150, 280}
```

```
09629     }
09630     ,{{ 290, 270, 290, 160, 290}
09631     ,{ 290, 210, 290, 160, 290}
09632     ,{ 290, 270, 290, 160, 290}
09633     ,{ 290, 210, 290, 160, 290}
09634     ,{ 290, 270, 290, 160, 290}
09635     }
09636     ,{{ 280, 200, 280, 260, 280}
09637     ,{ 250, 170, 250, 250, 250}
09638     ,{ 280, 200, 280, 150, 280}
09639     ,{ 260, 180, 130, 260, 130}
09640     ,{ 280, 200, 280, 150, 280}
09641     }
09642     ,{{ 290, 270, 290, 200, 290}
09643     ,{ 290, 210, 290, 160, 290}
09644     ,{ 290, 270, 290, 160, 290}
09645     ,{ 290, 210, 290, 160, 290}
09646     ,{ 200, 120, 200, 200, 200}
09647     }
09648     }
09649     ,{{{ 370, 310, 310, 310, 370}
09650     ,{ 370, 310, 310, 310, 370}
09651     ,{ 290, 290, 290, 290, 290}
09652     ,{ 290, 290, 290, 290, 290}
09653     ,{ 290, 290, 290, 290, 290}
09654     }
09655     ,{{{ 370, 310, 310, 310, 370}
09656     ,{ 370, 310, 310, 310, 370}
09657     ,{ 280, 280, 280, 280, 280}
09658     ,{ 240, 180, 240, 180, 180}
09659     ,{ 280, 280, 280, 280, 280}
09660     }
09661     ,{{{ 290, 290, 290, 290, 290}
09662     ,{ 290, 290, 290, 290, 290}
09663     ,{ 290, 290, 290, 290, 290}
09664     ,{ 290, 290, 290, 290, 290}
09665     ,{ 290, 290, 290, 290, 290}
09666     }
09667     ,{{{ 310, 280, 310, 280, 280}
09668     ,{ 310, 250, 310, 250, 250}
09669     ,{ 280, 280, 280, 280, 280}
09670     ,{ 260, 130, 130, 130, 260}
09671     ,{ 280, 280, 280, 280, 280}
09672     }
09673     ,{{{ 290, 290, 290, 290, 290}
09674     ,{ 290, 290, 290, 290, 290}
09675     ,{ 290, 290, 290, 290, 290}
09676     ,{ 290, 290, 290, 290, 290}
09677     ,{ 200, 200, 200, 200, 200}
09678     }
09679     }
09680     }
09681     ,{{{ 370, 340, 310, 370, 370}
09682     ,{ 370, 340, 310, 310, 370}
09683     ,{ 370, 340, 310, 370, 310}
09684     ,{ 340, 340, 310, 310, 310}
09685     ,{ 370, 340, 310, 370, 310}
09686     }
09687     ,{{{ 370, 340, 310, 310, 370}
09688     ,{ 370, 340, 310, 310, 370}
09689     ,{ 300, 300, 260, 260, 260}
09690     ,{ 260, 240, 260, 200, 260}
09691     ,{ 300, 300, 260, 260, 260}
09692     }
09693     ,{{{ 370, 340, 310, 370, 310}
09694     ,{ 340, 340, 310, 310, 310}
09695     ,{ 370, 340, 310, 370, 310}
09696     ,{ 340, 340, 310, 310, 310}
09697     ,{ 370, 340, 310, 370, 310}
09698     }
09699     ,{{{ 300, 300, 270, 280, 280}
09700     ,{ 270, 250, 270, 210, 270}
09701     ,{ 300, 300, 260, 260, 260}
09702     ,{ 280, 200, 150, 280, 280}
09703     ,{ 300, 300, 260, 260, 260}
09704     }
09705     ,{{{ 340, 340, 310, 340, 310}
09706     ,{ 340, 340, 310, 310, 310}
09707     ,{ 340, 310, 280, 340, 280}
09708     ,{ 340, 340, 310, 310, 310}
09709     ,{ 320, 320, 220, 220, 220}
09710     }
09711     }
09712     ,{{{ 370, 340, 310, 370, 310}
09713     ,{ 340, 340, 310, 310, 310}
09714     ,{ 370, 340, 310, 370, 310}
09715     ,{ 340, 340, 310, 310, 310}
```



```
09716     , { 370, 340, 310, 370, 310}
09717     }
09718     , { { 340, 340, 310, 310, 310}
09719     , { 340, 340, 310, 310, 310}
09720     , { 300, 300, 260, 260, 260}
09721     , { 240, 240, 200, 200, 200}
09722     , { 300, 300, 260, 260, 260}
09723     }
09724     , { { 370, 340, 310, 370, 310}
09725     , { 340, 340, 310, 310, 310}
09726     , { 370, 340, 310, 370, 310}
09727     , { 340, 340, 310, 310, 310}
09728     , { 370, 340, 310, 370, 310}
09729     }
09730     , { { 300, 300, 260, 280, 260}
09731     , { 250, 250, 210, 210, 210}
09732     , { 300, 300, 260, 260, 260}
09733     , { 280, 190, 150, 280, 150}
09734     , { 300, 300, 260, 260, 260}
09735     }
09736     , { { 340, 340, 310, 340, 310}
09737     , { 340, 340, 310, 310, 310}
09738     , { 340, 310, 280, 340, 280}
09739     , { 340, 340, 310, 310, 310}
09740     , { 320, 320, 220, 220, 220}
09741     }
09742     }
09743     , { { { 310, 310, 310, 310, 310}
09744     , { 310, 310, 310, 310, 310}
09745     , { 310, 310, 310, 310, 310}
09746     , { 310, 310, 310, 310, 310}
09747     , { 310, 310, 310, 310, 310}
09748     }
09749     , { { 310, 310, 310, 310, 310}
09750     , { 310, 310, 310, 310, 310}
09751     , { 260, 260, 260, 260, 260}
09752     , { 260, 200, 260, 200, 260}
09753     , { 260, 260, 260, 260, 260}
09754     }
09755     , { { 310, 310, 310, 310, 310}
09756     , { 310, 310, 310, 310, 310}
09757     , { 310, 310, 310, 310, 310}
09758     , { 310, 310, 310, 310, 310}
09759     , { 310, 310, 310, 310, 310}
09760     }
09761     , { { 270, 260, 270, 260, 270}
09762     , { 270, 210, 270, 210, 270}
09763     , { 260, 260, 260, 260, 260}
09764     , { 150, 150, 150, 150, 150}
09765     , { 260, 260, 260, 260, 260}
09766     }
09767     , { { 310, 310, 310, 310, 310}
09768     , { 310, 310, 310, 310, 310}
09769     , { 280, 280, 280, 280, 280}
09770     , { 310, 310, 310, 310, 310}
09771     , { 220, 220, 220, 220, 220}
09772     }
09773     }
09774     , { { { 310, 290, 310, 280, 310}
09775     , { 310, 230, 310, 210, 310}
09776     , { 310, 290, 310, 180, 310}
09777     , { 310, 230, 310, 280, 310}
09778     , { 310, 290, 310, 220, 310}
09779     }
09780     , { { 310, 230, 310, 200, 310}
09781     , { 310, 230, 310, 180, 310}
09782     , { 260, 180, 260, 130, 260}
09783     , { 200, 120, 200, 200, 200}
09784     , { 260, 180, 260, 130, 260}
09785     }
09786     , { { 310, 290, 310, 180, 310}
09787     , { 310, 230, 310, 180, 310}
09788     , { 310, 290, 310, 180, 310}
09789     , { 310, 230, 310, 180, 310}
09790     , { 310, 290, 310, 180, 310}
09791     }
09792     , { { 280, 200, 260, 280, 260}
09793     , { 210, 130, 210, 210, 210}
09794     , { 260, 180, 260, 130, 260}
09795     , { 280, 200, 150, 280, 150}
09796     , { 260, 180, 260, 130, 260}
09797     }
09798     , { { 310, 260, 310, 220, 310}
09799     , { 310, 230, 310, 180, 310}
09800     , { 280, 260, 280, 150, 280}
09801     , { 310, 230, 310, 180, 310}
09802     , { 220, 140, 220, 220, 220}
```

```
09803     }
09804     }
09805     ,{{{ 370, 310, 310, 310, 370}
09806     ,{ 370, 310, 310, 310, 370}
09807     ,{ 310, 310, 310, 310, 310}
09808     ,{ 310, 310, 310, 310, 310}
09809     ,{ 310, 310, 310, 310, 310}
09810     }
09811     ,{{{ 370, 310, 310, 310, 370}
09812     ,{ 370, 310, 310, 310, 370}
09813     ,{ 260, 260, 260, 260, 260}
09814     ,{ 260, 200, 260, 200, 200}
09815     ,{ 260, 260, 260, 260, 260}
09816     }
09817     ,{{{ 310, 310, 310, 310, 310}
09818     ,{ 310, 310, 310, 310, 310}
09819     ,{ 310, 310, 310, 310, 310}
09820     ,{ 310, 310, 310, 310, 310}
09821     ,{ 310, 310, 310, 310, 310}
09822     }
09823     ,{{{ 280, 260, 270, 260, 280}
09824     ,{ 270, 210, 270, 210, 210}
09825     ,{ 260, 260, 260, 260, 260}
09826     ,{ 280, 150, 150, 150, 280}
09827     ,{ 260, 260, 260, 260, 260}
09828     }
09829     ,{{{ 310, 310, 310, 310, 310}
09830     ,{ 310, 310, 310, 310, 310}
09831     ,{ 280, 280, 280, 280, 280}
09832     ,{ 310, 310, 310, 310, 310}
09833     ,{ 220, 220, 220, 220, 220}
09834     }
09835     }
09836     }
09837     ,{{{ 430, 430, 400, 400, 430}
09838     ,{ 430, 410, 400, 370, 430}
09839     ,{ 400, 370, 340, 400, 340}
09840     ,{ 370, 370, 340, 340, 340}
09841     ,{ 430, 430, 340, 400, 340}
09842     }
09843     ,{{{ 430, 410, 370, 370, 430}
09844     ,{ 430, 410, 370, 370, 430}
09845     ,{ 370, 370, 340, 340, 340}
09846     ,{ 320, 290, 320, 260, 320}
09847     ,{ 370, 370, 340, 340, 340}
09848     }
09849     ,{{{ 400, 370, 340, 400, 340}
09850     ,{ 370, 370, 340, 340, 340}
09851     ,{ 400, 370, 340, 400, 340}
09852     ,{ 370, 370, 340, 340, 340}
09853     ,{ 400, 370, 340, 400, 340}
09854     }
09855     ,{{{ 400, 370, 400, 340, 400}
09856     ,{ 400, 370, 400, 340, 400}
09857     ,{ 370, 370, 340, 340, 340}
09858     ,{ 340, 260, 210, 340, 340}
09859     ,{ 370, 370, 340, 340, 340}
09860     }
09861     ,{{{ 430, 430, 340, 400, 340}
09862     ,{ 370, 370, 340, 340, 340}
09863     ,{ 400, 370, 340, 400, 340}
09864     ,{ 370, 370, 340, 340, 340}
09865     ,{ 430, 430, 340, 340, 340}
09866     }
09867     }
09868     ,{{{ 430, 430, 370, 400, 370}
09869     ,{ 410, 410, 370, 370, 370}
09870     ,{ 400, 370, 340, 400, 340}
09871     ,{ 370, 370, 340, 340, 340}
09872     ,{ 430, 430, 340, 400, 340}
09873     }
09874     ,{{{ 410, 410, 370, 370, 370}
09875     ,{ 410, 410, 370, 370, 370}
09876     ,{ 370, 370, 340, 340, 340}
09877     ,{ 290, 290, 260, 260, 260}
09878     ,{ 370, 370, 340, 340, 340}
09879     }
09880     ,{{{ 400, 370, 340, 400, 340}
09881     ,{ 370, 370, 340, 340, 340}
09882     ,{ 400, 370, 340, 400, 340}
09883     ,{ 370, 370, 340, 340, 340}
09884     ,{ 400, 370, 340, 400, 340}
09885     }
09886     ,{{{ 370, 370, 340, 340, 340}
09887     ,{ 370, 370, 340, 340, 340}
09888     ,{ 370, 370, 340, 340, 340}
09889     ,{ 340, 240, 210, 340, 210}
```

```
09890     , { 370, 370, 340, 340, 340}
09891     }
09892     , { { 430, 430, 340, 400, 340}
09893     , { 370, 370, 340, 340, 340}
09894     , { 400, 370, 340, 400, 340}
09895     , { 370, 370, 340, 340, 340}
09896     , { 430, 430, 340, 340, 340}
09897     }
09898     }
09899     , { { { 400, 370, 400, 370, 400}
09900     , { 400, 370, 400, 370, 400}
09901     , { 340, 340, 340, 340, 340}
09902     , { 340, 340, 340, 340, 340}
09903     , { 340, 340, 340, 340, 340}
09904     }
09905     , { { 370, 370, 370, 370, 370}
09906     , { 370, 370, 370, 370, 370}
09907     , { 340, 340, 340, 340, 340}
09908     , { 320, 260, 320, 260, 320}
09909     , { 340, 340, 340, 340, 340}
09910     }
09911     , { { 340, 340, 340, 340, 340}
09912     , { 340, 340, 340, 340, 340}
09913     , { 340, 340, 340, 340, 340}
09914     , { 340, 340, 340, 340, 340}
09915     , { 340, 340, 340, 340, 340}
09916     }
09917     , { { 400, 340, 400, 340, 400}
09918     , { 400, 340, 400, 340, 400}
09919     , { 340, 340, 340, 340, 340}
09920     , { 210, 210, 210, 210, 210}
09921     , { 340, 340, 340, 340, 340}
09922     }
09923     , { { 340, 340, 340, 340, 340}
09924     , { 340, 340, 340, 340, 340}
09925     , { 340, 340, 340, 340, 340}
09926     , { 340, 340, 340, 340, 340}
09927     , { 340, 340, 340, 340, 340}
09928     }
09929     }
09930     , { { { 370, 320, 370, 340, 370}
09931     , { 370, 290, 370, 340, 370}
09932     , { 340, 320, 340, 210, 340}
09933     , { 340, 260, 340, 340, 340}
09934     , { 340, 320, 340, 340, 340}
09935     }
09936     , { { 370, 290, 370, 260, 370}
09937     , { 370, 290, 370, 240, 370}
09938     , { 340, 260, 340, 210, 340}
09939     , { 260, 180, 260, 260, 260}
09940     , { 340, 260, 340, 210, 340}
09941     }
09942     , { { 340, 320, 340, 210, 340}
09943     , { 340, 260, 340, 210, 340}
09944     , { 340, 320, 340, 210, 340}
09945     , { 340, 260, 340, 210, 340}
09946     , { 340, 320, 340, 210, 340}
09947     }
09948     , { { 340, 260, 340, 340, 340}
09949     , { 340, 260, 340, 340, 340}
09950     , { 340, 260, 340, 210, 340}
09951     , { 340, 260, 210, 340, 210}
09952     , { 340, 260, 340, 210, 340}
09953     }
09954     , { { 340, 320, 340, 340, 340}
09955     , { 340, 260, 340, 210, 340}
09956     , { 340, 320, 340, 210, 340}
09957     , { 340, 260, 340, 210, 340}
09958     , { 340, 260, 340, 340, 340}
09959     }
09960     }
09961     , { { { 430, 370, 400, 370, 430}
09962     , { 430, 370, 400, 370, 430}
09963     , { 340, 340, 340, 340, 340}
09964     , { 340, 340, 340, 340, 340}
09965     , { 340, 340, 340, 340, 340}
09966     }
09967     , { { 430, 370, 370, 370, 430}
09968     , { 430, 370, 370, 370, 430}
09969     , { 340, 340, 340, 340, 340}
09970     , { 320, 260, 320, 260, 260}
09971     , { 340, 340, 340, 340, 340}
09972     }
09973     , { { 340, 340, 340, 340, 340}
09974     , { 340, 340, 340, 340, 340}
09975     , { 340, 340, 340, 340, 340}
09976     , { 340, 340, 340, 340, 340}
```

```

09977     , { 340, 340, 340, 340, 340}
09978     }
09979     , { { 400, 340, 400, 340, 340}
09980     , { 400, 340, 400, 340, 340}
09981     , { 340, 340, 340, 340, 340}
09982     , { 340, 210, 210, 210, 340}
09983     , { 340, 340, 340, 340, 340}
09984     }
09985     , { { 340, 340, 340, 340, 340}
09986     , { 340, 340, 340, 340, 340}
09987     , { 340, 340, 340, 340, 340}
09988     , { 340, 340, 340, 340, 340}
09989     , { 340, 340, 340, 340, 340}
09990     }
09991     }
09992     }
09993     };;

```

18.175 intI22dH.h

```

00001 PUBLIC int int22_dH[NBPAIRS+1][NBPAIRS+1][5][5][5] =
00002 {{{{{ INF, INF, INF, INF, INF}
00003     , { INF, INF, INF, INF, INF}
00004     , { INF, INF, INF, INF, INF}
00005     , { INF, INF, INF, INF, INF}
00006     , { INF, INF, INF, INF, INF}
00007     }
00008     , { { INF, INF, INF, INF, INF}
00009     , { INF, INF, INF, INF, INF}
00010     , { INF, INF, INF, INF, INF}
00011     , { INF, INF, INF, INF, INF}
00012     , { INF, INF, INF, INF, INF}
00013     }
00014     , { { INF, INF, INF, INF, INF}
00015     , { INF, INF, INF, INF, INF}
00016     , { INF, INF, INF, INF, INF}
00017     , { INF, INF, INF, INF, INF}
00018     , { INF, INF, INF, INF, INF}
00019     }
00020     , { { INF, INF, INF, INF, INF}
00021     , { INF, INF, INF, INF, INF}
00022     , { INF, INF, INF, INF, INF}
00023     , { INF, INF, INF, INF, INF}
00024     , { INF, INF, INF, INF, INF}
00025     }
00026     , { { INF, INF, INF, INF, INF}
00027     , { INF, INF, INF, INF, INF}
00028     , { INF, INF, INF, INF, INF}
00029     , { INF, INF, INF, INF, INF}
00030     , { INF, INF, INF, INF, INF}
00031     }
00032     }
00033     , {{{ INF, INF, INF, INF, INF}
00034     , { INF, INF, INF, INF, INF}
00035     , { INF, INF, INF, INF, INF}
00036     , { INF, INF, INF, INF, INF}
00037     , { INF, INF, INF, INF, INF}
00038     }
00039     , {{{ INF, INF, INF, INF, INF}
00040     , { INF, INF, INF, INF, INF}
00041     , { INF, INF, INF, INF, INF}
00042     , { INF, INF, INF, INF, INF}
00043     , { INF, INF, INF, INF, INF}
00044     }
00045     , {{{ INF, INF, INF, INF, INF}
00046     , { INF, INF, INF, INF, INF}
00047     , { INF, INF, INF, INF, INF}
00048     , { INF, INF, INF, INF, INF}
00049     , { INF, INF, INF, INF, INF}
00050     }
00051     , {{{ INF, INF, INF, INF, INF}
00052     , { INF, INF, INF, INF, INF}
00053     , { INF, INF, INF, INF, INF}
00054     , { INF, INF, INF, INF, INF}
00055     , { INF, INF, INF, INF, INF}
00056     }
00057     , {{{ INF, INF, INF, INF, INF}
00058     , { INF, INF, INF, INF, INF}
00059     , { INF, INF, INF, INF, INF}
00060     , { INF, INF, INF, INF, INF}
00061     , { INF, INF, INF, INF, INF}
00062     }
00063     }
00064     , {{{ INF, INF, INF, INF, INF}
00065     , { INF, INF, INF, INF, INF}

```

```
00066     , {   INF,   INF,   INF,   INF,   INF }
00067     , {   INF,   INF,   INF,   INF,   INF }
00068     , {   INF,   INF,   INF,   INF,   INF }
00069     }
00070     , { {   INF,   INF,   INF,   INF,   INF }
00071     , {   INF,   INF,   INF,   INF,   INF }
00072     , {   INF,   INF,   INF,   INF,   INF }
00073     , {   INF,   INF,   INF,   INF,   INF }
00074     , {   INF,   INF,   INF,   INF,   INF }
00075     }
00076     , { {   INF,   INF,   INF,   INF,   INF }
00077     , {   INF,   INF,   INF,   INF,   INF }
00078     , {   INF,   INF,   INF,   INF,   INF }
00079     , {   INF,   INF,   INF,   INF,   INF }
00080     , {   INF,   INF,   INF,   INF,   INF }
00081     }
00082     , { {   INF,   INF,   INF,   INF,   INF }
00083     , {   INF,   INF,   INF,   INF,   INF }
00084     , {   INF,   INF,   INF,   INF,   INF }
00085     , {   INF,   INF,   INF,   INF,   INF }
00086     , {   INF,   INF,   INF,   INF,   INF }
00087     }
00088     , { {   INF,   INF,   INF,   INF,   INF }
00089     , {   INF,   INF,   INF,   INF,   INF }
00090     , {   INF,   INF,   INF,   INF,   INF }
00091     , {   INF,   INF,   INF,   INF,   INF }
00092     , {   INF,   INF,   INF,   INF,   INF }
00093     }
00094     }
00095     , { { {   INF,   INF,   INF,   INF,   INF }
00096     , {   INF,   INF,   INF,   INF,   INF }
00097     , {   INF,   INF,   INF,   INF,   INF }
00098     , {   INF,   INF,   INF,   INF,   INF }
00099     , {   INF,   INF,   INF,   INF,   INF }
00100     }
00101     , { {   INF,   INF,   INF,   INF,   INF }
00102     , {   INF,   INF,   INF,   INF,   INF }
00103     , {   INF,   INF,   INF,   INF,   INF }
00104     , {   INF,   INF,   INF,   INF,   INF }
00105     , {   INF,   INF,   INF,   INF,   INF }
00106     }
00107     , { {   INF,   INF,   INF,   INF,   INF }
00108     , {   INF,   INF,   INF,   INF,   INF }
00109     , {   INF,   INF,   INF,   INF,   INF }
00110     , {   INF,   INF,   INF,   INF,   INF }
00111     , {   INF,   INF,   INF,   INF,   INF }
00112     }
00113     , { {   INF,   INF,   INF,   INF,   INF }
00114     , {   INF,   INF,   INF,   INF,   INF }
00115     , {   INF,   INF,   INF,   INF,   INF }
00116     , {   INF,   INF,   INF,   INF,   INF }
00117     , {   INF,   INF,   INF,   INF,   INF }
00118     }
00119     , { {   INF,   INF,   INF,   INF,   INF }
00120     , {   INF,   INF,   INF,   INF,   INF }
00121     , {   INF,   INF,   INF,   INF,   INF }
00122     , {   INF,   INF,   INF,   INF,   INF }
00123     , {   INF,   INF,   INF,   INF,   INF }
00124     }
00125     }
00126     , { { {   INF,   INF,   INF,   INF,   INF }
00127     , {   INF,   INF,   INF,   INF,   INF }
00128     , {   INF,   INF,   INF,   INF,   INF }
00129     , {   INF,   INF,   INF,   INF,   INF }
00130     , {   INF,   INF,   INF,   INF,   INF }
00131     }
00132     , { {   INF,   INF,   INF,   INF,   INF }
00133     , {   INF,   INF,   INF,   INF,   INF }
00134     , {   INF,   INF,   INF,   INF,   INF }
00135     , {   INF,   INF,   INF,   INF,   INF }
00136     , {   INF,   INF,   INF,   INF,   INF }
00137     }
00138     , { {   INF,   INF,   INF,   INF,   INF }
00139     , {   INF,   INF,   INF,   INF,   INF }
00140     , {   INF,   INF,   INF,   INF,   INF }
00141     , {   INF,   INF,   INF,   INF,   INF }
00142     , {   INF,   INF,   INF,   INF,   INF }
00143     }
00144     , { {   INF,   INF,   INF,   INF,   INF }
00145     , {   INF,   INF,   INF,   INF,   INF }
00146     , {   INF,   INF,   INF,   INF,   INF }
00147     , {   INF,   INF,   INF,   INF,   INF }
00148     , {   INF,   INF,   INF,   INF,   INF }
00149     }
00150     , { {   INF,   INF,   INF,   INF,   INF }
00151     , {   INF,   INF,   INF,   INF,   INF }
00152     , {   INF,   INF,   INF,   INF,   INF }
```

```
00153      , {   INF,   INF,   INF,   INF,   INF }
00154      , {   INF,   INF,   INF,   INF,   INF }
00155      }
00156  }
00157  }
00158  , { { {   INF,   INF,   INF,   INF,   INF }
00159      , {   INF,   INF,   INF,   INF,   INF }
00160      , {   INF,   INF,   INF,   INF,   INF }
00161      , {   INF,   INF,   INF,   INF,   INF }
00162      , {   INF,   INF,   INF,   INF,   INF }
00163      }
00164  , { {   INF,   INF,   INF,   INF,   INF }
00165      , {   INF,   INF,   INF,   INF,   INF }
00166      , {   INF,   INF,   INF,   INF,   INF }
00167      , {   INF,   INF,   INF,   INF,   INF }
00168      , {   INF,   INF,   INF,   INF,   INF }
00169      }
00170  , { {   INF,   INF,   INF,   INF,   INF }
00171      , {   INF,   INF,   INF,   INF,   INF }
00172      , {   INF,   INF,   INF,   INF,   INF }
00173      , {   INF,   INF,   INF,   INF,   INF }
00174      , {   INF,   INF,   INF,   INF,   INF }
00175      }
00176  , { {   INF,   INF,   INF,   INF,   INF }
00177      , {   INF,   INF,   INF,   INF,   INF }
00178      , {   INF,   INF,   INF,   INF,   INF }
00179      , {   INF,   INF,   INF,   INF,   INF }
00180      , {   INF,   INF,   INF,   INF,   INF }
00181      }
00182  , { {   INF,   INF,   INF,   INF,   INF }
00183      , {   INF,   INF,   INF,   INF,   INF }
00184      , {   INF,   INF,   INF,   INF,   INF }
00185      , {   INF,   INF,   INF,   INF,   INF }
00186      , {   INF,   INF,   INF,   INF,   INF }
00187      }
00188  }
00189  , { { {   INF,   INF,   INF,   INF,   INF }
00190      , {   INF,   INF,   INF,   INF,   INF }
00191      , {   INF,   INF,   INF,   INF,   INF }
00192      , {   INF,   INF,   INF,   INF,   INF }
00193      , {   INF,   INF,   INF,   INF,   INF }
00194      }
00195  , { {   INF,   INF,   INF,   INF,   INF }
00196      , {   INF,   INF,   INF,   INF,   INF }
00197      , {   INF,   INF,   INF,   INF,   INF }
00198      , {   INF,   INF,   INF,   INF,   INF }
00199      , {   INF,   INF,   INF,   INF,   INF }
00200      }
00201  , { {   INF,   INF,   INF,   INF,   INF }
00202      , {   INF,   INF,   INF,   INF,   INF }
00203      , {   INF,   INF,   INF,   INF,   INF }
00204      , {   INF,   INF,   INF,   INF,   INF }
00205      , {   INF,   INF,   INF,   INF,   INF }
00206      }
00207  , { {   INF,   INF,   INF,   INF,   INF }
00208      , {   INF,   INF,   INF,   INF,   INF }
00209      , {   INF,   INF,   INF,   INF,   INF }
00210      , {   INF,   INF,   INF,   INF,   INF }
00211      , {   INF,   INF,   INF,   INF,   INF }
00212      }
00213  , { {   INF,   INF,   INF,   INF,   INF }
00214      , {   INF,   INF,   INF,   INF,   INF }
00215      , {   INF,   INF,   INF,   INF,   INF }
00216      , {   INF,   INF,   INF,   INF,   INF }
00217      , {   INF,   INF,   INF,   INF,   INF }
00218      }
00219  }
00220  , { { {   INF,   INF,   INF,   INF,   INF }
00221      , {   INF,   INF,   INF,   INF,   INF }
00222      , {   INF,   INF,   INF,   INF,   INF }
00223      , {   INF,   INF,   INF,   INF,   INF }
00224      , {   INF,   INF,   INF,   INF,   INF }
00225      }
00226  , { {   INF,   INF,   INF,   INF,   INF }
00227      , {   INF,   INF,   INF,   INF,   INF }
00228      , {   INF,   INF,   INF,   INF,   INF }
00229      , {   INF,   INF,   INF,   INF,   INF }
00230      , {   INF,   INF,   INF,   INF,   INF }
00231      }
00232  , { {   INF,   INF,   INF,   INF,   INF }
00233      , {   INF,   INF,   INF,   INF,   INF }
00234      , {   INF,   INF,   INF,   INF,   INF }
00235      , {   INF,   INF,   INF,   INF,   INF }
00236      , {   INF,   INF,   INF,   INF,   INF }
00237      }
00238  , { {   INF,   INF,   INF,   INF,   INF }
00239      , {   INF,   INF,   INF,   INF,   INF }
```

```
00240     , {   INF,   INF,   INF,   INF,   INF }
00241     , {   INF,   INF,   INF,   INF,   INF }
00242     , {   INF,   INF,   INF,   INF,   INF }
00243     }
00244     , { {   INF,   INF,   INF,   INF,   INF }
00245     , {   INF,   INF,   INF,   INF,   INF }
00246     , {   INF,   INF,   INF,   INF,   INF }
00247     , {   INF,   INF,   INF,   INF,   INF }
00248     , {   INF,   INF,   INF,   INF,   INF }
00249     }
00250     }
00251     , { { {   INF,   INF,   INF,   INF,   INF }
00252     , {   INF,   INF,   INF,   INF,   INF }
00253     , {   INF,   INF,   INF,   INF,   INF }
00254     , {   INF,   INF,   INF,   INF,   INF }
00255     , {   INF,   INF,   INF,   INF,   INF }
00256     }
00257     , { { {   INF,   INF,   INF,   INF,   INF }
00258     , {   INF,   INF,   INF,   INF,   INF }
00259     , {   INF,   INF,   INF,   INF,   INF }
00260     , {   INF,   INF,   INF,   INF,   INF }
00261     , {   INF,   INF,   INF,   INF,   INF }
00262     }
00263     , { { {   INF,   INF,   INF,   INF,   INF }
00264     , {   INF,   INF,   INF,   INF,   INF }
00265     , {   INF,   INF,   INF,   INF,   INF }
00266     , {   INF,   INF,   INF,   INF,   INF }
00267     , {   INF,   INF,   INF,   INF,   INF }
00268     }
00269     , { { {   INF,   INF,   INF,   INF,   INF }
00270     , {   INF,   INF,   INF,   INF,   INF }
00271     , {   INF,   INF,   INF,   INF,   INF }
00272     , {   INF,   INF,   INF,   INF,   INF }
00273     , {   INF,   INF,   INF,   INF,   INF }
00274     }
00275     , { { {   INF,   INF,   INF,   INF,   INF }
00276     , {   INF,   INF,   INF,   INF,   INF }
00277     , {   INF,   INF,   INF,   INF,   INF }
00278     , {   INF,   INF,   INF,   INF,   INF }
00279     , {   INF,   INF,   INF,   INF,   INF }
00280     }
00281     }
00282     , { { { {   INF,   INF,   INF,   INF,   INF }
00283     , {   INF,   INF,   INF,   INF,   INF }
00284     , {   INF,   INF,   INF,   INF,   INF }
00285     , {   INF,   INF,   INF,   INF,   INF }
00286     , {   INF,   INF,   INF,   INF,   INF }
00287     }
00288     , { { {   INF,   INF,   INF,   INF,   INF }
00289     , {   INF,   INF,   INF,   INF,   INF }
00290     , {   INF,   INF,   INF,   INF,   INF }
00291     , {   INF,   INF,   INF,   INF,   INF }
00292     , {   INF,   INF,   INF,   INF,   INF }
00293     }
00294     , { { {   INF,   INF,   INF,   INF,   INF }
00295     , {   INF,   INF,   INF,   INF,   INF }
00296     , {   INF,   INF,   INF,   INF,   INF }
00297     , {   INF,   INF,   INF,   INF,   INF }
00298     , {   INF,   INF,   INF,   INF,   INF }
00299     }
00300     , { { {   INF,   INF,   INF,   INF,   INF }
00301     , {   INF,   INF,   INF,   INF,   INF }
00302     , {   INF,   INF,   INF,   INF,   INF }
00303     , {   INF,   INF,   INF,   INF,   INF }
00304     , {   INF,   INF,   INF,   INF,   INF }
00305     }
00306     , { { {   INF,   INF,   INF,   INF,   INF }
00307     , {   INF,   INF,   INF,   INF,   INF }
00308     , {   INF,   INF,   INF,   INF,   INF }
00309     , {   INF,   INF,   INF,   INF,   INF }
00310     , {   INF,   INF,   INF,   INF,   INF }
00311     }
00312     }
00313     }
00314     , { { { { {   INF,   INF,   INF,   INF,   INF }
00315     , {   INF,   INF,   INF,   INF,   INF }
00316     , {   INF,   INF,   INF,   INF,   INF }
00317     , {   INF,   INF,   INF,   INF,   INF }
00318     , {   INF,   INF,   INF,   INF,   INF }
00319     }
00320     , { { {   INF,   INF,   INF,   INF,   INF }
00321     , {   INF,   INF,   INF,   INF,   INF }
00322     , {   INF,   INF,   INF,   INF,   INF }
00323     , {   INF,   INF,   INF,   INF,   INF }
00324     , {   INF,   INF,   INF,   INF,   INF }
00325     }
00326     , { { {   INF,   INF,   INF,   INF,   INF }
```

```
00327 , { INF, INF, INF, INF, INF }
00328 , { INF, INF, INF, INF, INF }
00329 , { INF, INF, INF, INF, INF }
00330 , { INF, INF, INF, INF, INF }
00331 }
00332 , { { INF, INF, INF, INF, INF }
00333 , { INF, INF, INF, INF, INF }
00334 , { INF, INF, INF, INF, INF }
00335 , { INF, INF, INF, INF, INF }
00336 , { INF, INF, INF, INF, INF }
00337 }
00338 , { { INF, INF, INF, INF, INF }
00339 , { INF, INF, INF, INF, INF }
00340 , { INF, INF, INF, INF, INF }
00341 , { INF, INF, INF, INF, INF }
00342 , { INF, INF, INF, INF, INF }
00343 }
00344 }
00345 , { { { INF, INF, INF, INF, INF }
00346 , { INF, INF, INF, INF, INF }
00347 , { INF, INF, INF, INF, INF }
00348 , { INF, INF, INF, INF, INF }
00349 , { INF, INF, INF, INF, INF }
00350 }
00351 , { { INF, INF, INF, INF, INF }
00352 , { INF, INF, INF, INF, INF }
00353 , { INF, INF, INF, INF, INF }
00354 , { INF, INF, INF, INF, INF }
00355 , { INF, INF, INF, INF, INF }
00356 }
00357 , { { INF, INF, INF, INF, INF }
00358 , { INF, INF, INF, INF, INF }
00359 , { INF, INF, INF, INF, INF }
00360 , { INF, INF, INF, INF, INF }
00361 , { INF, INF, INF, INF, INF }
00362 }
00363 , { { INF, INF, INF, INF, INF }
00364 , { INF, INF, INF, INF, INF }
00365 , { INF, INF, INF, INF, INF }
00366 , { INF, INF, INF, INF, INF }
00367 , { INF, INF, INF, INF, INF }
00368 }
00369 , { { INF, INF, INF, INF, INF }
00370 , { INF, INF, INF, INF, INF }
00371 , { INF, INF, INF, INF, INF }
00372 , { INF, INF, INF, INF, INF }
00373 , { INF, INF, INF, INF, INF }
00374 }
00375 }
00376 , { { { INF, INF, INF, INF, INF }
00377 , { INF, INF, INF, INF, INF }
00378 , { INF, INF, INF, INF, INF }
00379 , { INF, INF, INF, INF, INF }
00380 , { INF, INF, INF, INF, INF }
00381 }
00382 , { { INF, INF, INF, INF, INF }
00383 , { INF, INF, INF, INF, INF }
00384 , { INF, INF, INF, INF, INF }
00385 , { INF, INF, INF, INF, INF }
00386 , { INF, INF, INF, INF, INF }
00387 }
00388 , { { INF, INF, INF, INF, INF }
00389 , { INF, INF, INF, INF, INF }
00390 , { INF, INF, INF, INF, INF }
00391 , { INF, INF, INF, INF, INF }
00392 , { INF, INF, INF, INF, INF }
00393 }
00394 , { { INF, INF, INF, INF, INF }
00395 , { INF, INF, INF, INF, INF }
00396 , { INF, INF, INF, INF, INF }
00397 , { INF, INF, INF, INF, INF }
00398 , { INF, INF, INF, INF, INF }
00399 }
00400 , { { INF, INF, INF, INF, INF }
00401 , { INF, INF, INF, INF, INF }
00402 , { INF, INF, INF, INF, INF }
00403 , { INF, INF, INF, INF, INF }
00404 , { INF, INF, INF, INF, INF }
00405 }
00406 }
00407 , { { { INF, INF, INF, INF, INF }
00408 , { INF, INF, INF, INF, INF }
00409 , { INF, INF, INF, INF, INF }
00410 , { INF, INF, INF, INF, INF }
00411 , { INF, INF, INF, INF, INF }
00412 }
00413 , { { INF, INF, INF, INF, INF }
```



```
00414     , {   INF,   INF,   INF,   INF,   INF }
00415     , {   INF,   INF,   INF,   INF,   INF }
00416     , {   INF,   INF,   INF,   INF,   INF }
00417     , {   INF,   INF,   INF,   INF,   INF }
00418     }
00419     , { {   INF,   INF,   INF,   INF,   INF }
00420     , {   INF,   INF,   INF,   INF,   INF }
00421     , {   INF,   INF,   INF,   INF,   INF }
00422     , {   INF,   INF,   INF,   INF,   INF }
00423     , {   INF,   INF,   INF,   INF,   INF }
00424     }
00425     , { {   INF,   INF,   INF,   INF,   INF }
00426     , {   INF,   INF,   INF,   INF,   INF }
00427     , {   INF,   INF,   INF,   INF,   INF }
00428     , {   INF,   INF,   INF,   INF,   INF }
00429     , {   INF,   INF,   INF,   INF,   INF }
00430     }
00431     , { {   INF,   INF,   INF,   INF,   INF }
00432     , {   INF,   INF,   INF,   INF,   INF }
00433     , {   INF,   INF,   INF,   INF,   INF }
00434     , {   INF,   INF,   INF,   INF,   INF }
00435     , {   INF,   INF,   INF,   INF,   INF }
00436     }
00437     }
00438     , { { {   INF,   INF,   INF,   INF,   INF }
00439     , {   INF,   INF,   INF,   INF,   INF }
00440     , {   INF,   INF,   INF,   INF,   INF }
00441     , {   INF,   INF,   INF,   INF,   INF }
00442     , {   INF,   INF,   INF,   INF,   INF }
00443     }
00444     , { {   INF,   INF,   INF,   INF,   INF }
00445     , {   INF,   INF,   INF,   INF,   INF }
00446     , {   INF,   INF,   INF,   INF,   INF }
00447     , {   INF,   INF,   INF,   INF,   INF }
00448     , {   INF,   INF,   INF,   INF,   INF }
00449     }
00450     , { {   INF,   INF,   INF,   INF,   INF }
00451     , {   INF,   INF,   INF,   INF,   INF }
00452     , {   INF,   INF,   INF,   INF,   INF }
00453     , {   INF,   INF,   INF,   INF,   INF }
00454     , {   INF,   INF,   INF,   INF,   INF }
00455     }
00456     , { {   INF,   INF,   INF,   INF,   INF }
00457     , {   INF,   INF,   INF,   INF,   INF }
00458     , {   INF,   INF,   INF,   INF,   INF }
00459     , {   INF,   INF,   INF,   INF,   INF }
00460     , {   INF,   INF,   INF,   INF,   INF }
00461     }
00462     , { {   INF,   INF,   INF,   INF,   INF }
00463     , {   INF,   INF,   INF,   INF,   INF }
00464     , {   INF,   INF,   INF,   INF,   INF }
00465     , {   INF,   INF,   INF,   INF,   INF }
00466     , {   INF,   INF,   INF,   INF,   INF }
00467     }
00468     }
00469     }
00470     , { { { {   INF,   INF,   INF,   INF,   INF }
00471     , {   INF,   INF,   INF,   INF,   INF }
00472     , {   INF,   INF,   INF,   INF,   INF }
00473     , {   INF,   INF,   INF,   INF,   INF }
00474     , {   INF,   INF,   INF,   INF,   INF }
00475     }
00476     , { { {   INF,   INF,   INF,   INF,   INF }
00477     , {   INF,   INF,   INF,   INF,   INF }
00478     , {   INF,   INF,   INF,   INF,   INF }
00479     , {   INF,   INF,   INF,   INF,   INF }
00480     , {   INF,   INF,   INF,   INF,   INF }
00481     }
00482     , { { {   INF,   INF,   INF,   INF,   INF }
00483     , {   INF,   INF,   INF,   INF,   INF }
00484     , {   INF,   INF,   INF,   INF,   INF }
00485     , {   INF,   INF,   INF,   INF,   INF }
00486     , {   INF,   INF,   INF,   INF,   INF }
00487     }
00488     , { { {   INF,   INF,   INF,   INF,   INF }
00489     , {   INF,   INF,   INF,   INF,   INF }
00490     , {   INF,   INF,   INF,   INF,   INF }
00491     , {   INF,   INF,   INF,   INF,   INF }
00492     , {   INF,   INF,   INF,   INF,   INF }
00493     }
00494     , { { {   INF,   INF,   INF,   INF,   INF }
00495     , {   INF,   INF,   INF,   INF,   INF }
00496     , {   INF,   INF,   INF,   INF,   INF }
00497     , {   INF,   INF,   INF,   INF,   INF }
00498     , {   INF,   INF,   INF,   INF,   INF }
00499     }
00500     }
```

```
00501 ,{{{ INF, INF, INF, INF, INF}
00502 ,{ INF, INF, INF, INF, INF}
00503 ,{ INF, INF, INF, INF, INF}
00504 ,{ INF, INF, INF, INF, INF}
00505 ,{ INF, INF, INF, INF, INF}
00506 }
00507 ,{{{ INF, INF, INF, INF, INF}
00508 ,{ INF, INF, INF, INF, INF}
00509 ,{ INF, INF, INF, INF, INF}
00510 ,{ INF, INF, INF, INF, INF}
00511 ,{ INF, INF, INF, INF, INF}
00512 }
00513 ,{{{ INF, INF, INF, INF, INF}
00514 ,{ INF, INF, INF, INF, INF}
00515 ,{ INF, INF, INF, INF, INF}
00516 ,{ INF, INF, INF, INF, INF}
00517 ,{ INF, INF, INF, INF, INF}
00518 }
00519 ,{{{ INF, INF, INF, INF, INF}
00520 ,{ INF, INF, INF, INF, INF}
00521 ,{ INF, INF, INF, INF, INF}
00522 ,{ INF, INF, INF, INF, INF}
00523 ,{ INF, INF, INF, INF, INF}
00524 }
00525 ,{{{ INF, INF, INF, INF, INF}
00526 ,{ INF, INF, INF, INF, INF}
00527 ,{ INF, INF, INF, INF, INF}
00528 ,{ INF, INF, INF, INF, INF}
00529 ,{ INF, INF, INF, INF, INF}
00530 }
00531 }
00532 ,{{{ INF, INF, INF, INF, INF}
00533 ,{ INF, INF, INF, INF, INF}
00534 ,{ INF, INF, INF, INF, INF}
00535 ,{ INF, INF, INF, INF, INF}
00536 ,{ INF, INF, INF, INF, INF}
00537 }
00538 ,{{{ INF, INF, INF, INF, INF}
00539 ,{ INF, INF, INF, INF, INF}
00540 ,{ INF, INF, INF, INF, INF}
00541 ,{ INF, INF, INF, INF, INF}
00542 ,{ INF, INF, INF, INF, INF}
00543 }
00544 ,{{{ INF, INF, INF, INF, INF}
00545 ,{ INF, INF, INF, INF, INF}
00546 ,{ INF, INF, INF, INF, INF}
00547 ,{ INF, INF, INF, INF, INF}
00548 ,{ INF, INF, INF, INF, INF}
00549 }
00550 ,{{{ INF, INF, INF, INF, INF}
00551 ,{ INF, INF, INF, INF, INF}
00552 ,{ INF, INF, INF, INF, INF}
00553 ,{ INF, INF, INF, INF, INF}
00554 ,{ INF, INF, INF, INF, INF}
00555 }
00556 ,{{{ INF, INF, INF, INF, INF}
00557 ,{ INF, INF, INF, INF, INF}
00558 ,{ INF, INF, INF, INF, INF}
00559 ,{ INF, INF, INF, INF, INF}
00560 ,{ INF, INF, INF, INF, INF}
00561 }
00562 }
00563 ,{{{ INF, INF, INF, INF, INF}
00564 ,{ INF, INF, INF, INF, INF}
00565 ,{ INF, INF, INF, INF, INF}
00566 ,{ INF, INF, INF, INF, INF}
00567 ,{ INF, INF, INF, INF, INF}
00568 }
00569 ,{{{ INF, INF, INF, INF, INF}
00570 ,{ INF, INF, INF, INF, INF}
00571 ,{ INF, INF, INF, INF, INF}
00572 ,{ INF, INF, INF, INF, INF}
00573 ,{ INF, INF, INF, INF, INF}
00574 }
00575 ,{{{ INF, INF, INF, INF, INF}
00576 ,{ INF, INF, INF, INF, INF}
00577 ,{ INF, INF, INF, INF, INF}
00578 ,{ INF, INF, INF, INF, INF}
00579 ,{ INF, INF, INF, INF, INF}
00580 }
00581 ,{{{ INF, INF, INF, INF, INF}
00582 ,{ INF, INF, INF, INF, INF}
00583 ,{ INF, INF, INF, INF, INF}
00584 ,{ INF, INF, INF, INF, INF}
00585 ,{ INF, INF, INF, INF, INF}
00586 }
00587 ,{{{ INF, INF, INF, INF, INF}
```

```
00588     , {   INF,   INF,   INF,   INF,   INF }
00589     , {   INF,   INF,   INF,   INF,   INF }
00590     , {   INF,   INF,   INF,   INF,   INF }
00591     , {   INF,   INF,   INF,   INF,   INF }
00592     }
00593 }
00594 , { { {   INF,   INF,   INF,   INF,   INF }
00595     , {   INF,   INF,   INF,   INF,   INF }
00596     , {   INF,   INF,   INF,   INF,   INF }
00597     , {   INF,   INF,   INF,   INF,   INF }
00598     , {   INF,   INF,   INF,   INF,   INF }
00599     }
00600 , { { {   INF,   INF,   INF,   INF,   INF }
00601     , {   INF,   INF,   INF,   INF,   INF }
00602     , {   INF,   INF,   INF,   INF,   INF }
00603     , {   INF,   INF,   INF,   INF,   INF }
00604     , {   INF,   INF,   INF,   INF,   INF }
00605     }
00606 , { { {   INF,   INF,   INF,   INF,   INF }
00607     , {   INF,   INF,   INF,   INF,   INF }
00608     , {   INF,   INF,   INF,   INF,   INF }
00609     , {   INF,   INF,   INF,   INF,   INF }
00610     , {   INF,   INF,   INF,   INF,   INF }
00611     }
00612 , { { {   INF,   INF,   INF,   INF,   INF }
00613     , {   INF,   INF,   INF,   INF,   INF }
00614     , {   INF,   INF,   INF,   INF,   INF }
00615     , {   INF,   INF,   INF,   INF,   INF }
00616     , {   INF,   INF,   INF,   INF,   INF }
00617     }
00618 , { { {   INF,   INF,   INF,   INF,   INF }
00619     , {   INF,   INF,   INF,   INF,   INF }
00620     , {   INF,   INF,   INF,   INF,   INF }
00621     , {   INF,   INF,   INF,   INF,   INF }
00622     , {   INF,   INF,   INF,   INF,   INF }
00623     }
00624 }
00625 }
00626 , { { { {   INF,   INF,   INF,   INF,   INF }
00627     , {   INF,   INF,   INF,   INF,   INF }
00628     , {   INF,   INF,   INF,   INF,   INF }
00629     , {   INF,   INF,   INF,   INF,   INF }
00630     , {   INF,   INF,   INF,   INF,   INF }
00631     }
00632 , { { {   INF,   INF,   INF,   INF,   INF }
00633     , {   INF,   INF,   INF,   INF,   INF }
00634     , {   INF,   INF,   INF,   INF,   INF }
00635     , {   INF,   INF,   INF,   INF,   INF }
00636     , {   INF,   INF,   INF,   INF,   INF }
00637     }
00638 , { { {   INF,   INF,   INF,   INF,   INF }
00639     , {   INF,   INF,   INF,   INF,   INF }
00640     , {   INF,   INF,   INF,   INF,   INF }
00641     , {   INF,   INF,   INF,   INF,   INF }
00642     , {   INF,   INF,   INF,   INF,   INF }
00643     }
00644 , { { {   INF,   INF,   INF,   INF,   INF }
00645     , {   INF,   INF,   INF,   INF,   INF }
00646     , {   INF,   INF,   INF,   INF,   INF }
00647     , {   INF,   INF,   INF,   INF,   INF }
00648     , {   INF,   INF,   INF,   INF,   INF }
00649     }
00650 , { { {   INF,   INF,   INF,   INF,   INF }
00651     , {   INF,   INF,   INF,   INF,   INF }
00652     , {   INF,   INF,   INF,   INF,   INF }
00653     , {   INF,   INF,   INF,   INF,   INF }
00654     , {   INF,   INF,   INF,   INF,   INF }
00655     }
00656 }
00657 , { { { {   INF,   INF,   INF,   INF,   INF }
00658     , {   INF,   INF,   INF,   INF,   INF }
00659     , {   INF,   INF,   INF,   INF,   INF }
00660     , {   INF,   INF,   INF,   INF,   INF }
00661     , {   INF,   INF,   INF,   INF,   INF }
00662     }
00663 , { { {   INF,   INF,   INF,   INF,   INF }
00664     , {   INF,   INF,   INF,   INF,   INF }
00665     , {   INF,   INF,   INF,   INF,   INF }
00666     , {   INF,   INF,   INF,   INF,   INF }
00667     , {   INF,   INF,   INF,   INF,   INF }
00668     }
00669 , { { {   INF,   INF,   INF,   INF,   INF }
00670     , {   INF,   INF,   INF,   INF,   INF }
00671     , {   INF,   INF,   INF,   INF,   INF }
00672     , {   INF,   INF,   INF,   INF,   INF }
00673     , {   INF,   INF,   INF,   INF,   INF }
00674     }
```

```
00675 ,{{ INF, INF, INF, INF, INF}
00676 ,{ INF, INF, INF, INF, INF}
00677 ,{ INF, INF, INF, INF, INF}
00678 ,{ INF, INF, INF, INF, INF}
00679 ,{ INF, INF, INF, INF, INF}
00680 }
00681 ,{{ INF, INF, INF, INF, INF}
00682 ,{ INF, INF, INF, INF, INF}
00683 ,{ INF, INF, INF, INF, INF}
00684 ,{ INF, INF, INF, INF, INF}
00685 ,{ INF, INF, INF, INF, INF}
00686 }
00687 }
00688 ,{{{ INF, INF, INF, INF, INF}
00689 ,{ INF, INF, INF, INF, INF}
00690 ,{ INF, INF, INF, INF, INF}
00691 ,{ INF, INF, INF, INF, INF}
00692 ,{ INF, INF, INF, INF, INF}
00693 }
00694 ,{{ INF, INF, INF, INF, INF}
00695 ,{ INF, INF, INF, INF, INF}
00696 ,{ INF, INF, INF, INF, INF}
00697 ,{ INF, INF, INF, INF, INF}
00698 ,{ INF, INF, INF, INF, INF}
00699 }
00700 ,{{ INF, INF, INF, INF, INF}
00701 ,{ INF, INF, INF, INF, INF}
00702 ,{ INF, INF, INF, INF, INF}
00703 ,{ INF, INF, INF, INF, INF}
00704 ,{ INF, INF, INF, INF, INF}
00705 }
00706 ,{{ INF, INF, INF, INF, INF}
00707 ,{ INF, INF, INF, INF, INF}
00708 ,{ INF, INF, INF, INF, INF}
00709 ,{ INF, INF, INF, INF, INF}
00710 ,{ INF, INF, INF, INF, INF}
00711 }
00712 ,{{{ INF, INF, INF, INF, INF}
00713 ,{ INF, INF, INF, INF, INF}
00714 ,{ INF, INF, INF, INF, INF}
00715 ,{ INF, INF, INF, INF, INF}
00716 ,{ INF, INF, INF, INF, INF}
00717 }
00718 }
00719 ,{{{ INF, INF, INF, INF, INF}
00720 ,{ INF, INF, INF, INF, INF}
00721 ,{ INF, INF, INF, INF, INF}
00722 ,{ INF, INF, INF, INF, INF}
00723 ,{ INF, INF, INF, INF, INF}
00724 }
00725 ,{{{ INF, INF, INF, INF, INF}
00726 ,{ INF, INF, INF, INF, INF}
00727 ,{ INF, INF, INF, INF, INF}
00728 ,{ INF, INF, INF, INF, INF}
00729 ,{ INF, INF, INF, INF, INF}
00730 }
00731 ,{{{ INF, INF, INF, INF, INF}
00732 ,{ INF, INF, INF, INF, INF}
00733 ,{ INF, INF, INF, INF, INF}
00734 ,{ INF, INF, INF, INF, INF}
00735 ,{ INF, INF, INF, INF, INF}
00736 }
00737 ,{{{ INF, INF, INF, INF, INF}
00738 ,{ INF, INF, INF, INF, INF}
00739 ,{ INF, INF, INF, INF, INF}
00740 ,{ INF, INF, INF, INF, INF}
00741 ,{ INF, INF, INF, INF, INF}
00742 }
00743 ,{{{ INF, INF, INF, INF, INF}
00744 ,{ INF, INF, INF, INF, INF}
00745 ,{ INF, INF, INF, INF, INF}
00746 ,{ INF, INF, INF, INF, INF}
00747 ,{ INF, INF, INF, INF, INF}
00748 }
00749 }
00750 ,{{{ INF, INF, INF, INF, INF}
00751 ,{ INF, INF, INF, INF, INF}
00752 ,{ INF, INF, INF, INF, INF}
00753 ,{ INF, INF, INF, INF, INF}
00754 ,{ INF, INF, INF, INF, INF}
00755 }
00756 ,{{ INF, INF, INF, INF, INF}
00757 ,{ INF, INF, INF, INF, INF}
00758 ,{ INF, INF, INF, INF, INF}
00759 ,{ INF, INF, INF, INF, INF}
00760 ,{ INF, INF, INF, INF, INF}
00761 }
```

```
00762 ,{{ INF, INF, INF, INF, INF}
00763 ,{ INF, INF, INF, INF, INF}
00764 ,{ INF, INF, INF, INF, INF}
00765 ,{ INF, INF, INF, INF, INF}
00766 ,{ INF, INF, INF, INF, INF}
00767 }
00768 ,{{ INF, INF, INF, INF, INF}
00769 ,{ INF, INF, INF, INF, INF}
00770 ,{ INF, INF, INF, INF, INF}
00771 ,{ INF, INF, INF, INF, INF}
00772 ,{ INF, INF, INF, INF, INF}
00773 }
00774 ,{{ INF, INF, INF, INF, INF}
00775 ,{ INF, INF, INF, INF, INF}
00776 ,{ INF, INF, INF, INF, INF}
00777 ,{ INF, INF, INF, INF, INF}
00778 ,{ INF, INF, INF, INF, INF}
00779 }
00780 }
00781 }
00782 ,{{{ INF, INF, INF, INF, INF}
00783 ,{ INF, INF, INF, INF, INF}
00784 ,{ INF, INF, INF, INF, INF}
00785 ,{ INF, INF, INF, INF, INF}
00786 ,{ INF, INF, INF, INF, INF}
00787 }
00788 ,{{{ INF, INF, INF, INF, INF}
00789 ,{ INF, INF, INF, INF, INF}
00790 ,{ INF, INF, INF, INF, INF}
00791 ,{ INF, INF, INF, INF, INF}
00792 ,{ INF, INF, INF, INF, INF}
00793 }
00794 ,{{{ INF, INF, INF, INF, INF}
00795 ,{ INF, INF, INF, INF, INF}
00796 ,{ INF, INF, INF, INF, INF}
00797 ,{ INF, INF, INF, INF, INF}
00798 ,{ INF, INF, INF, INF, INF}
00799 }
00800 ,{{{ INF, INF, INF, INF, INF}
00801 ,{ INF, INF, INF, INF, INF}
00802 ,{ INF, INF, INF, INF, INF}
00803 ,{ INF, INF, INF, INF, INF}
00804 ,{ INF, INF, INF, INF, INF}
00805 }
00806 ,{{{ INF, INF, INF, INF, INF}
00807 ,{ INF, INF, INF, INF, INF}
00808 ,{ INF, INF, INF, INF, INF}
00809 ,{ INF, INF, INF, INF, INF}
00810 ,{ INF, INF, INF, INF, INF}
00811 }
00812 }
00813 ,{{{ INF, INF, INF, INF, INF}
00814 ,{ INF, INF, INF, INF, INF}
00815 ,{ INF, INF, INF, INF, INF}
00816 ,{ INF, INF, INF, INF, INF}
00817 ,{ INF, INF, INF, INF, INF}
00818 }
00819 ,{{{ INF, INF, INF, INF, INF}
00820 ,{ INF, INF, INF, INF, INF}
00821 ,{ INF, INF, INF, INF, INF}
00822 ,{ INF, INF, INF, INF, INF}
00823 ,{ INF, INF, INF, INF, INF}
00824 }
00825 ,{{{ INF, INF, INF, INF, INF}
00826 ,{ INF, INF, INF, INF, INF}
00827 ,{ INF, INF, INF, INF, INF}
00828 ,{ INF, INF, INF, INF, INF}
00829 ,{ INF, INF, INF, INF, INF}
00830 }
00831 ,{{{ INF, INF, INF, INF, INF}
00832 ,{ INF, INF, INF, INF, INF}
00833 ,{ INF, INF, INF, INF, INF}
00834 ,{ INF, INF, INF, INF, INF}
00835 ,{ INF, INF, INF, INF, INF}
00836 }
00837 ,{{{ INF, INF, INF, INF, INF}
00838 ,{ INF, INF, INF, INF, INF}
00839 ,{ INF, INF, INF, INF, INF}
00840 ,{ INF, INF, INF, INF, INF}
00841 ,{ INF, INF, INF, INF, INF}
00842 }
00843 }
00844 ,{{{ INF, INF, INF, INF, INF}
00845 ,{ INF, INF, INF, INF, INF}
00846 ,{ INF, INF, INF, INF, INF}
00847 ,{ INF, INF, INF, INF, INF}
00848 ,{ INF, INF, INF, INF, INF}
```

```
00849     }
00850     ,{{   INF,   INF,   INF,   INF,   INF}
00851     ,{{   INF,   INF,   INF,   INF,   INF}
00852     ,{{   INF,   INF,   INF,   INF,   INF}
00853     ,{{   INF,   INF,   INF,   INF,   INF}
00854     ,{{   INF,   INF,   INF,   INF,   INF}
00855     }
00856     ,{{   INF,   INF,   INF,   INF,   INF}
00857     ,{{   INF,   INF,   INF,   INF,   INF}
00858     ,{{   INF,   INF,   INF,   INF,   INF}
00859     ,{{   INF,   INF,   INF,   INF,   INF}
00860     ,{{   INF,   INF,   INF,   INF,   INF}
00861     }
00862     ,{{   INF,   INF,   INF,   INF,   INF}
00863     ,{{   INF,   INF,   INF,   INF,   INF}
00864     ,{{   INF,   INF,   INF,   INF,   INF}
00865     ,{{   INF,   INF,   INF,   INF,   INF}
00866     ,{{   INF,   INF,   INF,   INF,   INF}
00867     }
00868     ,{{   INF,   INF,   INF,   INF,   INF}
00869     ,{{   INF,   INF,   INF,   INF,   INF}
00870     ,{{   INF,   INF,   INF,   INF,   INF}
00871     ,{{   INF,   INF,   INF,   INF,   INF}
00872     ,{{   INF,   INF,   INF,   INF,   INF}
00873     }
00874     }
00875     ,{{{   INF,   INF,   INF,   INF,   INF}
00876     ,{{   INF,   INF,   INF,   INF,   INF}
00877     ,{{   INF,   INF,   INF,   INF,   INF}
00878     ,{{   INF,   INF,   INF,   INF,   INF}
00879     ,{{   INF,   INF,   INF,   INF,   INF}
00880     }
00881     ,{{{   INF,   INF,   INF,   INF,   INF}
00882     ,{{   INF,   INF,   INF,   INF,   INF}
00883     ,{{   INF,   INF,   INF,   INF,   INF}
00884     ,{{   INF,   INF,   INF,   INF,   INF}
00885     ,{{   INF,   INF,   INF,   INF,   INF}
00886     }
00887     ,{{{   INF,   INF,   INF,   INF,   INF}
00888     ,{{   INF,   INF,   INF,   INF,   INF}
00889     ,{{   INF,   INF,   INF,   INF,   INF}
00890     ,{{   INF,   INF,   INF,   INF,   INF}
00891     ,{{   INF,   INF,   INF,   INF,   INF}
00892     }
00893     ,{{{   INF,   INF,   INF,   INF,   INF}
00894     ,{{   INF,   INF,   INF,   INF,   INF}
00895     ,{{   INF,   INF,   INF,   INF,   INF}
00896     ,{{   INF,   INF,   INF,   INF,   INF}
00897     ,{{   INF,   INF,   INF,   INF,   INF}
00898     }
00899     ,{{{   INF,   INF,   INF,   INF,   INF}
00900     ,{{   INF,   INF,   INF,   INF,   INF}
00901     ,{{   INF,   INF,   INF,   INF,   INF}
00902     ,{{   INF,   INF,   INF,   INF,   INF}
00903     ,{{   INF,   INF,   INF,   INF,   INF}
00904     }
00905     }
00906     ,{{{   INF,   INF,   INF,   INF,   INF}
00907     ,{{   INF,   INF,   INF,   INF,   INF}
00908     ,{{   INF,   INF,   INF,   INF,   INF}
00909     ,{{   INF,   INF,   INF,   INF,   INF}
00910     ,{{   INF,   INF,   INF,   INF,   INF}
00911     }
00912     ,{{{   INF,   INF,   INF,   INF,   INF}
00913     ,{{   INF,   INF,   INF,   INF,   INF}
00914     ,{{   INF,   INF,   INF,   INF,   INF}
00915     ,{{   INF,   INF,   INF,   INF,   INF}
00916     ,{{   INF,   INF,   INF,   INF,   INF}
00917     }
00918     ,{{{   INF,   INF,   INF,   INF,   INF}
00919     ,{{   INF,   INF,   INF,   INF,   INF}
00920     ,{{   INF,   INF,   INF,   INF,   INF}
00921     ,{{   INF,   INF,   INF,   INF,   INF}
00922     ,{{   INF,   INF,   INF,   INF,   INF}
00923     }
00924     ,{{{   INF,   INF,   INF,   INF,   INF}
00925     ,{{   INF,   INF,   INF,   INF,   INF}
00926     ,{{   INF,   INF,   INF,   INF,   INF}
00927     ,{{   INF,   INF,   INF,   INF,   INF}
00928     ,{{   INF,   INF,   INF,   INF,   INF}
00929     }
00930     ,{{{   INF,   INF,   INF,   INF,   INF}
00931     ,{{   INF,   INF,   INF,   INF,   INF}
00932     ,{{   INF,   INF,   INF,   INF,   INF}
00933     ,{{   INF,   INF,   INF,   INF,   INF}
00934     ,{{   INF,   INF,   INF,   INF,   INF}
00935     }
```

```
00936     }
00937     }
00938     ,{{{ INF, INF, INF, INF, INF}
00939     ,{ INF, INF, INF, INF, INF}
00940     ,{ INF, INF, INF, INF, INF}
00941     ,{ INF, INF, INF, INF, INF}
00942     ,{ INF, INF, INF, INF, INF}
00943     }
00944     ,{{{ INF, INF, INF, INF, INF}
00945     ,{ INF, INF, INF, INF, INF}
00946     ,{ INF, INF, INF, INF, INF}
00947     ,{ INF, INF, INF, INF, INF}
00948     ,{ INF, INF, INF, INF, INF}
00949     }
00950     ,{{{ INF, INF, INF, INF, INF}
00951     ,{ INF, INF, INF, INF, INF}
00952     ,{ INF, INF, INF, INF, INF}
00953     ,{ INF, INF, INF, INF, INF}
00954     ,{ INF, INF, INF, INF, INF}
00955     }
00956     ,{{{ INF, INF, INF, INF, INF}
00957     ,{ INF, INF, INF, INF, INF}
00958     ,{ INF, INF, INF, INF, INF}
00959     ,{ INF, INF, INF, INF, INF}
00960     ,{ INF, INF, INF, INF, INF}
00961     }
00962     ,{{{ INF, INF, INF, INF, INF}
00963     ,{ INF, INF, INF, INF, INF}
00964     ,{ INF, INF, INF, INF, INF}
00965     ,{ INF, INF, INF, INF, INF}
00966     ,{ INF, INF, INF, INF, INF}
00967     }
00968     }
00969     ,{{{ INF, INF, INF, INF, INF}
00970     ,{ INF, INF, INF, INF, INF}
00971     ,{ INF, INF, INF, INF, INF}
00972     ,{ INF, INF, INF, INF, INF}
00973     ,{ INF, INF, INF, INF, INF}
00974     }
00975     ,{{{ INF, INF, INF, INF, INF}
00976     ,{ INF, INF, INF, INF, INF}
00977     ,{ INF, INF, INF, INF, INF}
00978     ,{ INF, INF, INF, INF, INF}
00979     ,{ INF, INF, INF, INF, INF}
00980     }
00981     ,{{{ INF, INF, INF, INF, INF}
00982     ,{ INF, INF, INF, INF, INF}
00983     ,{ INF, INF, INF, INF, INF}
00984     ,{ INF, INF, INF, INF, INF}
00985     ,{ INF, INF, INF, INF, INF}
00986     }
00987     ,{{{ INF, INF, INF, INF, INF}
00988     ,{ INF, INF, INF, INF, INF}
00989     ,{ INF, INF, INF, INF, INF}
00990     ,{ INF, INF, INF, INF, INF}
00991     ,{ INF, INF, INF, INF, INF}
00992     }
00993     ,{{{ INF, INF, INF, INF, INF}
00994     ,{ INF, INF, INF, INF, INF}
00995     ,{ INF, INF, INF, INF, INF}
00996     ,{ INF, INF, INF, INF, INF}
00997     ,{ INF, INF, INF, INF, INF}
00998     }
00999     }
01000     ,{{{ INF, INF, INF, INF, INF}
01001     ,{ INF, INF, INF, INF, INF}
01002     ,{ INF, INF, INF, INF, INF}
01003     ,{ INF, INF, INF, INF, INF}
01004     ,{ INF, INF, INF, INF, INF}
01005     }
01006     ,{{{ INF, INF, INF, INF, INF}
01007     ,{ INF, INF, INF, INF, INF}
01008     ,{ INF, INF, INF, INF, INF}
01009     ,{ INF, INF, INF, INF, INF}
01010     ,{ INF, INF, INF, INF, INF}
01011     }
01012     ,{{{ INF, INF, INF, INF, INF}
01013     ,{ INF, INF, INF, INF, INF}
01014     ,{ INF, INF, INF, INF, INF}
01015     ,{ INF, INF, INF, INF, INF}
01016     ,{ INF, INF, INF, INF, INF}
01017     }
01018     ,{{{ INF, INF, INF, INF, INF}
01019     ,{ INF, INF, INF, INF, INF}
01020     ,{ INF, INF, INF, INF, INF}
01021     ,{ INF, INF, INF, INF, INF}
01022     ,{ INF, INF, INF, INF, INF}
```

```
01023     }
01024     ,{{   INF,   INF,   INF,   INF,   INF}
01025     ,{{   INF,   INF,   INF,   INF,   INF}
01026     ,{{   INF,   INF,   INF,   INF,   INF}
01027     ,{{   INF,   INF,   INF,   INF,   INF}
01028     ,{{   INF,   INF,   INF,   INF,   INF}
01029     }
01030     }
01031     ,{{{   INF,   INF,   INF,   INF,   INF}
01032     ,{{   INF,   INF,   INF,   INF,   INF}
01033     ,{{   INF,   INF,   INF,   INF,   INF}
01034     ,{{   INF,   INF,   INF,   INF,   INF}
01035     ,{{   INF,   INF,   INF,   INF,   INF}
01036     }
01037     ,{{{   INF,   INF,   INF,   INF,   INF}
01038     ,{{   INF,   INF,   INF,   INF,   INF}
01039     ,{{   INF,   INF,   INF,   INF,   INF}
01040     ,{{   INF,   INF,   INF,   INF,   INF}
01041     ,{{   INF,   INF,   INF,   INF,   INF}
01042     }
01043     ,{{{   INF,   INF,   INF,   INF,   INF}
01044     ,{{   INF,   INF,   INF,   INF,   INF}
01045     ,{{   INF,   INF,   INF,   INF,   INF}
01046     ,{{   INF,   INF,   INF,   INF,   INF}
01047     ,{{   INF,   INF,   INF,   INF,   INF}
01048     }
01049     ,{{{   INF,   INF,   INF,   INF,   INF}
01050     ,{{   INF,   INF,   INF,   INF,   INF}
01051     ,{{   INF,   INF,   INF,   INF,   INF}
01052     ,{{   INF,   INF,   INF,   INF,   INF}
01053     ,{{   INF,   INF,   INF,   INF,   INF}
01054     }
01055     ,{{{   INF,   INF,   INF,   INF,   INF}
01056     ,{{   INF,   INF,   INF,   INF,   INF}
01057     ,{{   INF,   INF,   INF,   INF,   INF}
01058     ,{{   INF,   INF,   INF,   INF,   INF}
01059     ,{{   INF,   INF,   INF,   INF,   INF}
01060     }
01061     }
01062     ,{{{   INF,   INF,   INF,   INF,   INF}
01063     ,{{   INF,   INF,   INF,   INF,   INF}
01064     ,{{   INF,   INF,   INF,   INF,   INF}
01065     ,{{   INF,   INF,   INF,   INF,   INF}
01066     ,{{   INF,   INF,   INF,   INF,   INF}
01067     }
01068     ,{{{   INF,   INF,   INF,   INF,   INF}
01069     ,{{   INF,   INF,   INF,   INF,   INF}
01070     ,{{   INF,   INF,   INF,   INF,   INF}
01071     ,{{   INF,   INF,   INF,   INF,   INF}
01072     ,{{   INF,   INF,   INF,   INF,   INF}
01073     }
01074     ,{{{   INF,   INF,   INF,   INF,   INF}
01075     ,{{   INF,   INF,   INF,   INF,   INF}
01076     ,{{   INF,   INF,   INF,   INF,   INF}
01077     ,{{   INF,   INF,   INF,   INF,   INF}
01078     ,{{   INF,   INF,   INF,   INF,   INF}
01079     }
01080     ,{{{   INF,   INF,   INF,   INF,   INF}
01081     ,{{   INF,   INF,   INF,   INF,   INF}
01082     ,{{   INF,   INF,   INF,   INF,   INF}
01083     ,{{   INF,   INF,   INF,   INF,   INF}
01084     ,{{   INF,   INF,   INF,   INF,   INF}
01085     }
01086     ,{{{   INF,   INF,   INF,   INF,   INF}
01087     ,{{   INF,   INF,   INF,   INF,   INF}
01088     ,{{   INF,   INF,   INF,   INF,   INF}
01089     ,{{   INF,   INF,   INF,   INF,   INF}
01090     ,{{   INF,   INF,   INF,   INF,   INF}
01091     }
01092     }
01093     }
01094     ,{{{   INF,   INF,   INF,   INF,   INF}
01095     ,{{   INF,   INF,   INF,   INF,   INF}
01096     ,{{   INF,   INF,   INF,   INF,   INF}
01097     ,{{   INF,   INF,   INF,   INF,   INF}
01098     ,{{   INF,   INF,   INF,   INF,   INF}
01099     }
01100     ,{{{   INF,   INF,   INF,   INF,   INF}
01101     ,{{   INF,   INF,   INF,   INF,   INF}
01102     ,{{   INF,   INF,   INF,   INF,   INF}
01103     ,{{   INF,   INF,   INF,   INF,   INF}
01104     ,{{   INF,   INF,   INF,   INF,   INF}
01105     }
01106     ,{{{   INF,   INF,   INF,   INF,   INF}
01107     ,{{   INF,   INF,   INF,   INF,   INF}
01108     ,{{   INF,   INF,   INF,   INF,   INF}
01109     ,{{   INF,   INF,   INF,   INF,   INF}
```



```
01110     , {   INF,   INF,   INF,   INF,   INF }
01111     }
01112     , { {   INF,   INF,   INF,   INF,   INF }
01113     , {   INF,   INF,   INF,   INF,   INF }
01114     , {   INF,   INF,   INF,   INF,   INF }
01115     , {   INF,   INF,   INF,   INF,   INF }
01116     , {   INF,   INF,   INF,   INF,   INF }
01117     }
01118     , { {   INF,   INF,   INF,   INF,   INF }
01119     , {   INF,   INF,   INF,   INF,   INF }
01120     , {   INF,   INF,   INF,   INF,   INF }
01121     , {   INF,   INF,   INF,   INF,   INF }
01122     , {   INF,   INF,   INF,   INF,   INF }
01123     }
01124     }
01125     , { { {   INF,   INF,   INF,   INF,   INF }
01126     , {   INF,   INF,   INF,   INF,   INF }
01127     , {   INF,   INF,   INF,   INF,   INF }
01128     , {   INF,   INF,   INF,   INF,   INF }
01129     , {   INF,   INF,   INF,   INF,   INF }
01130     }
01131     , { {   INF,   INF,   INF,   INF,   INF }
01132     , {   INF,   INF,   INF,   INF,   INF }
01133     , {   INF,   INF,   INF,   INF,   INF }
01134     , {   INF,   INF,   INF,   INF,   INF }
01135     , {   INF,   INF,   INF,   INF,   INF }
01136     }
01137     , { {   INF,   INF,   INF,   INF,   INF }
01138     , {   INF,   INF,   INF,   INF,   INF }
01139     , {   INF,   INF,   INF,   INF,   INF }
01140     , {   INF,   INF,   INF,   INF,   INF }
01141     , {   INF,   INF,   INF,   INF,   INF }
01142     }
01143     , { {   INF,   INF,   INF,   INF,   INF }
01144     , {   INF,   INF,   INF,   INF,   INF }
01145     , {   INF,   INF,   INF,   INF,   INF }
01146     , {   INF,   INF,   INF,   INF,   INF }
01147     , {   INF,   INF,   INF,   INF,   INF }
01148     }
01149     , { {   INF,   INF,   INF,   INF,   INF }
01150     , {   INF,   INF,   INF,   INF,   INF }
01151     , {   INF,   INF,   INF,   INF,   INF }
01152     , {   INF,   INF,   INF,   INF,   INF }
01153     , {   INF,   INF,   INF,   INF,   INF }
01154     }
01155     }
01156     , { { {   INF,   INF,   INF,   INF,   INF }
01157     , {   INF,   INF,   INF,   INF,   INF }
01158     , {   INF,   INF,   INF,   INF,   INF }
01159     , {   INF,   INF,   INF,   INF,   INF }
01160     , {   INF,   INF,   INF,   INF,   INF }
01161     }
01162     , { {   INF,   INF,   INF,   INF,   INF }
01163     , {   INF,   INF,   INF,   INF,   INF }
01164     , {   INF,   INF,   INF,   INF,   INF }
01165     , {   INF,   INF,   INF,   INF,   INF }
01166     , {   INF,   INF,   INF,   INF,   INF }
01167     }
01168     , { {   INF,   INF,   INF,   INF,   INF }
01169     , {   INF,   INF,   INF,   INF,   INF }
01170     , {   INF,   INF,   INF,   INF,   INF }
01171     , {   INF,   INF,   INF,   INF,   INF }
01172     , {   INF,   INF,   INF,   INF,   INF }
01173     }
01174     , { {   INF,   INF,   INF,   INF,   INF }
01175     , {   INF,   INF,   INF,   INF,   INF }
01176     , {   INF,   INF,   INF,   INF,   INF }
01177     , {   INF,   INF,   INF,   INF,   INF }
01178     , {   INF,   INF,   INF,   INF,   INF }
01179     }
01180     , { {   INF,   INF,   INF,   INF,   INF }
01181     , {   INF,   INF,   INF,   INF,   INF }
01182     , {   INF,   INF,   INF,   INF,   INF }
01183     , {   INF,   INF,   INF,   INF,   INF }
01184     , {   INF,   INF,   INF,   INF,   INF }
01185     }
01186     }
01187     , { { {   INF,   INF,   INF,   INF,   INF }
01188     , {   INF,   INF,   INF,   INF,   INF }
01189     , {   INF,   INF,   INF,   INF,   INF }
01190     , {   INF,   INF,   INF,   INF,   INF }
01191     , {   INF,   INF,   INF,   INF,   INF }
01192     }
01193     , { {   INF,   INF,   INF,   INF,   INF }
01194     , {   INF,   INF,   INF,   INF,   INF }
01195     , {   INF,   INF,   INF,   INF,   INF }
01196     , {   INF,   INF,   INF,   INF,   INF }
```

```
01197 , { INF, INF, INF, INF, INF }
01198 }
01199 , { { INF, INF, INF, INF, INF }
01200 , { INF, INF, INF, INF, INF }
01201 , { INF, INF, INF, INF, INF }
01202 , { INF, INF, INF, INF, INF }
01203 , { INF, INF, INF, INF, INF }
01204 }
01205 , { { INF, INF, INF, INF, INF }
01206 , { INF, INF, INF, INF, INF }
01207 , { INF, INF, INF, INF, INF }
01208 , { INF, INF, INF, INF, INF }
01209 , { INF, INF, INF, INF, INF }
01210 }
01211 , { { INF, INF, INF, INF, INF }
01212 , { INF, INF, INF, INF, INF }
01213 , { INF, INF, INF, INF, INF }
01214 , { INF, INF, INF, INF, INF }
01215 , { INF, INF, INF, INF, INF }
01216 }
01217 }
01218 , { { { INF, INF, INF, INF, INF }
01219 , { INF, INF, INF, INF, INF }
01220 , { INF, INF, INF, INF, INF }
01221 , { INF, INF, INF, INF, INF }
01222 , { INF, INF, INF, INF, INF }
01223 }
01224 , { { INF, INF, INF, INF, INF }
01225 , { INF, INF, INF, INF, INF }
01226 , { INF, INF, INF, INF, INF }
01227 , { INF, INF, INF, INF, INF }
01228 , { INF, INF, INF, INF, INF }
01229 }
01230 , { { INF, INF, INF, INF, INF }
01231 , { INF, INF, INF, INF, INF }
01232 , { INF, INF, INF, INF, INF }
01233 , { INF, INF, INF, INF, INF }
01234 , { INF, INF, INF, INF, INF }
01235 }
01236 , { { INF, INF, INF, INF, INF }
01237 , { INF, INF, INF, INF, INF }
01238 , { INF, INF, INF, INF, INF }
01239 , { INF, INF, INF, INF, INF }
01240 , { INF, INF, INF, INF, INF }
01241 }
01242 , { { INF, INF, INF, INF, INF }
01243 , { INF, INF, INF, INF, INF }
01244 , { INF, INF, INF, INF, INF }
01245 , { INF, INF, INF, INF, INF }
01246 , { INF, INF, INF, INF, INF }
01247 }
01248 }
01249 }
01250 }
01251 , { { { { { INF, INF, INF, INF, INF }
01252 , { INF, INF, INF, INF, INF }
01253 , { INF, INF, INF, INF, INF }
01254 , { INF, INF, INF, INF, INF }
01255 , { INF, INF, INF, INF, INF }
01256 }
01257 , { { INF, INF, INF, INF, INF }
01258 , { INF, INF, INF, INF, INF }
01259 , { INF, INF, INF, INF, INF }
01260 , { INF, INF, INF, INF, INF }
01261 , { INF, INF, INF, INF, INF }
01262 }
01263 , { { INF, INF, INF, INF, INF }
01264 , { INF, INF, INF, INF, INF }
01265 , { INF, INF, INF, INF, INF }
01266 , { INF, INF, INF, INF, INF }
01267 , { INF, INF, INF, INF, INF }
01268 }
01269 , { { INF, INF, INF, INF, INF }
01270 , { INF, INF, INF, INF, INF }
01271 , { INF, INF, INF, INF, INF }
01272 , { INF, INF, INF, INF, INF }
01273 , { INF, INF, INF, INF, INF }
01274 }
01275 , { { INF, INF, INF, INF, INF }
01276 , { INF, INF, INF, INF, INF }
01277 , { INF, INF, INF, INF, INF }
01278 , { INF, INF, INF, INF, INF }
01279 , { INF, INF, INF, INF, INF }
01280 }
01281 }
01282 , { { { INF, INF, INF, INF, INF }
01283 , { INF, INF, INF, INF, INF }
```

```
01284     , {   INF,   INF,   INF,   INF,   INF }
01285     , {   INF,   INF,   INF,   INF,   INF }
01286     , {   INF,   INF,   INF,   INF,   INF }
01287     }
01288     , { {   INF,   INF,   INF,   INF,   INF }
01289     , {   INF,   INF,   INF,   INF,   INF }
01290     , {   INF,   INF,   INF,   INF,   INF }
01291     , {   INF,   INF,   INF,   INF,   INF }
01292     , {   INF,   INF,   INF,   INF,   INF }
01293     }
01294     , { {   INF,   INF,   INF,   INF,   INF }
01295     , {   INF,   INF,   INF,   INF,   INF }
01296     , {   INF,   INF,   INF,   INF,   INF }
01297     , {   INF,   INF,   INF,   INF,   INF }
01298     , {   INF,   INF,   INF,   INF,   INF }
01299     }
01300     , { {   INF,   INF,   INF,   INF,   INF }
01301     , {   INF,   INF,   INF,   INF,   INF }
01302     , {   INF,   INF,   INF,   INF,   INF }
01303     , {   INF,   INF,   INF,   INF,   INF }
01304     , {   INF,   INF,   INF,   INF,   INF }
01305     }
01306     , { {   INF,   INF,   INF,   INF,   INF }
01307     , {   INF,   INF,   INF,   INF,   INF }
01308     , {   INF,   INF,   INF,   INF,   INF }
01309     , {   INF,   INF,   INF,   INF,   INF }
01310     , {   INF,   INF,   INF,   INF,   INF }
01311     }
01312     }
01313     , { { {   INF,   INF,   INF,   INF,   INF }
01314     , {   INF,   INF,   INF,   INF,   INF }
01315     , {   INF,   INF,   INF,   INF,   INF }
01316     , {   INF,   INF,   INF,   INF,   INF }
01317     , {   INF,   INF,   INF,   INF,   INF }
01318     }
01319     , { {   INF,   INF,   INF,   INF,   INF }
01320     , {   INF,   INF,   INF,   INF,   INF }
01321     , {   INF,   INF,   INF,   INF,   INF }
01322     , {   INF,   INF,   INF,   INF,   INF }
01323     , {   INF,   INF,   INF,   INF,   INF }
01324     }
01325     , { {   INF,   INF,   INF,   INF,   INF }
01326     , {   INF,   INF,   INF,   INF,   INF }
01327     , {   INF,   INF,   INF,   INF,   INF }
01328     , {   INF,   INF,   INF,   INF,   INF }
01329     , {   INF,   INF,   INF,   INF,   INF }
01330     }
01331     , { {   INF,   INF,   INF,   INF,   INF }
01332     , {   INF,   INF,   INF,   INF,   INF }
01333     , {   INF,   INF,   INF,   INF,   INF }
01334     , {   INF,   INF,   INF,   INF,   INF }
01335     , {   INF,   INF,   INF,   INF,   INF }
01336     }
01337     , { {   INF,   INF,   INF,   INF,   INF }
01338     , {   INF,   INF,   INF,   INF,   INF }
01339     , {   INF,   INF,   INF,   INF,   INF }
01340     , {   INF,   INF,   INF,   INF,   INF }
01341     , {   INF,   INF,   INF,   INF,   INF }
01342     }
01343     }
01344     , { { {   INF,   INF,   INF,   INF,   INF }
01345     , {   INF,   INF,   INF,   INF,   INF }
01346     , {   INF,   INF,   INF,   INF,   INF }
01347     , {   INF,   INF,   INF,   INF,   INF }
01348     , {   INF,   INF,   INF,   INF,   INF }
01349     }
01350     , { {   INF,   INF,   INF,   INF,   INF }
01351     , {   INF,   INF,   INF,   INF,   INF }
01352     , {   INF,   INF,   INF,   INF,   INF }
01353     , {   INF,   INF,   INF,   INF,   INF }
01354     , {   INF,   INF,   INF,   INF,   INF }
01355     }
01356     , { {   INF,   INF,   INF,   INF,   INF }
01357     , {   INF,   INF,   INF,   INF,   INF }
01358     , {   INF,   INF,   INF,   INF,   INF }
01359     , {   INF,   INF,   INF,   INF,   INF }
01360     , {   INF,   INF,   INF,   INF,   INF }
01361     }
01362     , { {   INF,   INF,   INF,   INF,   INF }
01363     , {   INF,   INF,   INF,   INF,   INF }
01364     , {   INF,   INF,   INF,   INF,   INF }
01365     , {   INF,   INF,   INF,   INF,   INF }
01366     , {   INF,   INF,   INF,   INF,   INF }
01367     }
01368     , { {   INF,   INF,   INF,   INF,   INF }
01369     , {   INF,   INF,   INF,   INF,   INF }
01370     , {   INF,   INF,   INF,   INF,   INF }
```

```

01371      , {   INF,   INF,   INF,   INF,   INF }
01372      , {   INF,   INF,   INF,   INF,   INF }
01373      }
01374      }
01375      , {{ {   INF,   INF,   INF,   INF,   INF }
01376      , {   INF,   INF,   INF,   INF,   INF }
01377      , {   INF,   INF,   INF,   INF,   INF }
01378      , {   INF,   INF,   INF,   INF,   INF }
01379      , {   INF,   INF,   INF,   INF,   INF }
01380      }
01381      , {{ {   INF,   INF,   INF,   INF,   INF }
01382      , {   INF,   INF,   INF,   INF,   INF }
01383      , {   INF,   INF,   INF,   INF,   INF }
01384      , {   INF,   INF,   INF,   INF,   INF }
01385      , {   INF,   INF,   INF,   INF,   INF }
01386      }
01387      , {{ {   INF,   INF,   INF,   INF,   INF }
01388      , {   INF,   INF,   INF,   INF,   INF }
01389      , {   INF,   INF,   INF,   INF,   INF }
01390      , {   INF,   INF,   INF,   INF,   INF }
01391      , {   INF,   INF,   INF,   INF,   INF }
01392      }
01393      , {{ {   INF,   INF,   INF,   INF,   INF }
01394      , {   INF,   INF,   INF,   INF,   INF }
01395      , {   INF,   INF,   INF,   INF,   INF }
01396      , {   INF,   INF,   INF,   INF,   INF }
01397      , {   INF,   INF,   INF,   INF,   INF }
01398      }
01399      , {{ {   INF,   INF,   INF,   INF,   INF }
01400      , {   INF,   INF,   INF,   INF,   INF }
01401      , {   INF,   INF,   INF,   INF,   INF }
01402      , {   INF,   INF,   INF,   INF,   INF }
01403      , {   INF,   INF,   INF,   INF,   INF }
01404      }
01405      }
01406      }
01407      , {{{ {   80,  -120,   30,   80,   80 }
01408      , {   30,  -310,  -170,   30,  -110 }
01409      , {   80,  -230,  -110,   80,  -60 }
01410      , {   80,  -120,   30,   30,   80 }
01411      , {  -30,  -340,  -220,  -30,  -170 }
01412      }
01413      , {{ { -120,  -460,  -290,  -120,  -230 }
01414      , { -120,  -460,  -310,  -120,  -260 }
01415      , { -430,  -770,  -620,  -430,  -570 }
01416      , { -230,  -670,  -290,  -980,  -230 }
01417      , { -430,  -770,  -620,  -430,  -570 }
01418      }
01419      , {{ {   30,  -290,  -170,   30,  -110 }
01420      , {   30,  -310,  -170,   30,  -110 }
01421      , {   20,  -290,  -170,   20,  -120 }
01422      , {   30,  -310,  -170,   30,  -110 }
01423      , {  -30,  -340,  -220,  -30,  -170 }
01424      }
01425      , {{ {   80,  -120,   30,  -430,   80 }
01426      , { -520,  -960,  -580, -1270,  -520 }
01427      , { -430,  -770,  -620,  -430,  -570 }
01428      , {   80,  -120,   30,  -430,   80 }
01429      , { -430,  -770,  -620,  -430,  -570 }
01430      }
01431      , {{ {   80,  -230,  -110,   80,  -60 }
01432      , {   30,  -310,  -170,   30,  -110 }
01433      , {   80,  -230,  -110,   80,  -60 }
01434      , {   30,  -310,  -170,   30,  -110 }
01435      , {  -860,  -860,  -960, -1410,  -900 }
01436      }
01437      }
01438      , {{{ {   30,  -120,   30,  -520,   30 }
01439      , { -170,  -310,  -170,  -810,  -170 }
01440      , { -110,  -260,  -110,  -520,  -110 }
01441      , {   30,  -120,   30,  -810,   30 }
01442      , { -220,  -370,  -220,  -630,  -220 }
01443      }
01444      , {{ { -310,  -460,  -310,  -960,  -310 }
01445      , { -310,  -460,  -310,  -960,  -310 }
01446      , { -620,  -770,  -620, -1270,  -620 }
01447      , { -530,  -670,  -530, -1170,  -530 }
01448      , { -620,  -770,  -620, -1270,  -620 }
01449      }
01450      , {{ { -170,  -310,  -170,  -580,  -170 }
01451      , { -170,  -310,  -170,  -810,  -170 }
01452      , { -170,  -320,  -170,  -580,  -170 }
01453      , { -170,  -310,  -170,  -810,  -170 }
01454      , { -220,  -370,  -220,  -630,  -220 }
01455      }
01456      , {{ {   30,  -120,   30, -1270,   30 }
01457      , {  -810,  -960,  -810, -1460,  -810 }

```

```
01458 , { -620, -770, -620, -1270, -620}
01459 , { 30, -120, 30, -1870, 30}
01460 , { -620, -770, -620, -1270, -620}
01461 }
01462 , { { -110, -260, -110, -520, -110}
01463 , { -170, -310, -170, -810, -170}
01464 , { -110, -260, -110, -520, -110}
01465 , { -170, -310, -170, -810, -170}
01466 , { -860, -860, -960, -1600, -960}
01467 }
01468 }
01469 , { { { 80, -430, 20, -430, 80}
01470 , { -110, -620, -170, -620, -110}
01471 , { -60, -570, -120, -570, -60}
01472 , { 80, -430, 20, -430, 80}
01473 , { -170, -680, -230, -680, -170}
01474 }
01475 , { { -230, -770, -290, -770, -230}
01476 , { -260, -770, -320, -770, -260}
01477 , { -570, -1080, -630, -1080, -570}
01478 , { -230, -980, -290, -980, -230}
01479 , { -570, -1080, -630, -1080, -570}
01480 }
01481 , { { -110, -620, -170, -620, -110}
01482 , { -110, -620, -170, -620, -110}
01483 , { -120, -630, -180, -630, -120}
01484 , { -110, -620, -170, -620, -110}
01485 , { -170, -680, -230, -680, -170}
01486 }
01487 , { { { 80, -430, 20, -430, 80}
01488 , { -520, -1270, -580, -1270, -520}
01489 , { -570, -1080, -630, -1080, -570}
01490 , { 80, -430, 20, -430, 80}
01491 , { -570, -1080, -630, -1080, -570}
01492 }
01493 , { { -60, -570, -120, -570, -60}
01494 , { -110, -620, -170, -620, -110}
01495 , { -60, -570, -120, -570, -60}
01496 , { -110, -620, -170, -620, -110}
01497 , { -900, -1410, -960, -1410, -900}
01498 }
01499 }
01500 , { { { { 80, -230, 30, 80, 30}
01501 , { 30, -530, -170, 30, -170}
01502 , { 80, -230, -110, 80, -110}
01503 , { 30, -530, 30, 30, 30}
01504 , { -30, -340, -220, -30, -220}
01505 }
01506 , { { -120, -670, -310, -120, -310}
01507 , { -120, -670, -310, -120, -310}
01508 , { -430, -980, -620, -430, -620}
01509 , { -530, -890, -530, -1580, -530}
01510 , { -430, -980, -620, -430, -620}
01511 }
01512 , { { 30, -290, -170, 30, -170}
01513 , { 30, -530, -170, 30, -170}
01514 , { 20, -290, -170, 20, -170}
01515 , { 30, -530, -170, 30, -170}
01516 , { -30, -340, -220, -30, -220}
01517 }
01518 , { { 30, -980, 30, -430, 30}
01519 , { -810, -1170, -810, -1870, -810}
01520 , { -430, -980, -620, -430, -620}
01521 , { 30, -1580, 30, -2280, 30}
01522 , { -430, -980, -620, -430, -620}
01523 }
01524 , { { { 80, -230, -110, 80, -110}
01525 , { 30, -530, -170, 30, -170}
01526 , { 80, -230, -110, 80, -110}
01527 , { 30, -530, -170, 30, -170}
01528 , { -960, -1320, -960, -2010, -960}
01529 }
01530 }
01531 , { { { { -30, -430, -30, -430, -860}
01532 , { -220, -620, -220, -620, -860}
01533 , { -170, -570, -170, -570, -900}
01534 , { -30, -430, -30, -430, -960}
01535 , { -280, -680, -280, -680, -1010}
01536 }
01537 , { { { -340, -770, -340, -770, -860}
01538 , { -370, -770, -370, -770, -860}
01539 , { -680, -1080, -680, -1080, -1410}
01540 , { -340, -980, -340, -980, -1320}
01541 , { -680, -1080, -680, -1080, -1410}
01542 }
01543 , { { { -220, -620, -220, -620, -960}
01544 , { -220, -620, -220, -620, -960}
```

```

01545     , { -230, -630, -230, -630, -960}
01546     , { -220, -620, -220, -620, -960}
01547     , { -280, -680, -280, -680, -1010}
01548     }
01549     , { { -30, -430, -30, -430, -1410}
01550     , { -630, -1270, -630, -1270, -1600}
01551     , { -680, -1080, -680, -1080, -1410}
01552     , { -30, -430, -30, -430, -2010}
01553     , { -680, -1080, -680, -1080, -1410}
01554     }
01555     , { { -170, -570, -170, -570, -900}
01556     , { -220, -620, -220, -620, -960}
01557     , { -170, -570, -170, -570, -900}
01558     , { -220, -620, -220, -620, -960}
01559     , { -1010, -1410, -1010, -1410, -1750}
01560     }
01561     }
01562     }
01563     , { { { 540, 180, 30, 540, 180}
01564     , { 10, -580, -150, 10, -90}
01565     , { 540, -350, -600, 540, -540}
01566     , { 180, 180, 30, -320, 180}
01567     , { -90, -740, -90, -260, -540}
01568     }
01569     , { { -90, -350, -150, -100, -90}
01570     , { -90, -580, -150, -200, -90}
01571     , { -100, -350, -600, -100, -540}
01572     , { -630, -1790, -630, -1790, -1040}
01573     , { -400, -740, -600, -400, -540}
01574     }
01575     , { { 540, -660, -510, 540, -400}
01576     , { 10, -660, -510, 10, -400}
01577     , { 540, -940, -820, 540, -760}
01578     , { -320, -660, -510, -320, -460}
01579     , { -260, -940, -820, -260, -550}
01580     }
01581     , { { 180, 180, 30, -400, 180}
01582     , { -500, -1070, -500, -1080, -570}
01583     , { -400, -740, -600, -400, -540}
01584     , { 180, 180, 30, -430, 180}
01585     , { -400, -740, -600, -400, -540}
01586     }
01587     , { { -90, -660, -90, -210, -460}
01588     , { -320, -660, -510, -320, -460}
01589     , { -210, -1250, -1130, -210, -1070}
01590     , { -320, -660, -510, -320, -460}
01591     , { -90, -830, -90, -810, -800}
01592     }
01593     }
01594     , { { { 540, 180, -90, 540, 30}
01595     , { 10, -580, -220, 10, -150}
01596     , { 540, -740, -600, 540, -600}
01597     , { 180, 180, -390, -1160, 30}
01598     , { -90, -740, -90, -810, -600}
01599     }
01600     , { { -100, -580, -220, -100, -150}
01601     , { -150, -580, -220, -970, -150}
01602     , { -100, -740, -600, -100, -600}
01603     , { -1340, -2010, -1650, -1980, -1340}
01604     , { -600, -740, -600, -1240, -600}
01605     }
01606     , { { 540, -660, -510, 540, -510}
01607     , { 10, -660, -1150, 10, -510}
01608     , { 540, -960, -820, 540, -820}
01609     , { -510, -660, -510, -1160, -510}
01610     , { -820, -960, -820, -1220, -820}
01611     }
01612     , { { 180, 180, -390, -1240, 30}
01613     , { -860, -1340, -860, -2450, -860}
01614     , { -600, -740, -600, -1240, -600}
01615     , { 180, 180, -390, -1870, 30}
01616     , { -600, -740, -600, -1240, -600}
01617     }
01618     , { { -90, -660, -90, -810, -510}
01619     , { -510, -660, -510, -1160, -510}
01620     , { -1130, -1270, -1130, -1530, -1130}
01621     , { -510, -660, -510, -1160, -510}
01622     , { -90, -1240, -90, -810, -800}
01623     }
01624     }
01625     , { { { 180, -430, 20, -430, 180}
01626     , { -90, -600, -500, -600, -90}
01627     , { -540, -1050, -600, -1050, -540}
01628     , { 180, -430, 20, -430, 180}
01629     , { -540, -830, -600, -1050, -540}
01630     }
01631     , { { -90, -600, -600, -600, -90}

```

```
01632     , { -90, -600, -1070, -600, -90 }
01633     , { -540, -1050, -600, -1050, -540 }
01634     , { -630, -1790, -630, -1790, -1040 }
01635     , { -540, -1050, -600, -1050, -540 }
01636     }
01637     , { { -460, -970, -520, -970, -460 }
01638     , { -460, -970, -750, -970, -460 }
01639     , { -760, -1270, -820, -1270, -760 }
01640     , { -460, -970, -520, -970, -460 }
01641     , { -550, -1270, -820, -1270, -550 }
01642     }
01643     , { { 180, -430, 20, -430, 180 }
01644     , { -500, -1070, -500, -1320, -570 }
01645     , { -540, -1050, -600, -1050, -540 }
01646     , { 180, -430, 20, -430, 180 }
01647     , { -540, -1050, -600, -1050, -540 }
01648     }
01649     , { { -460, -830, -520, -970, -460 }
01650     , { -460, -970, -520, -970, -460 }
01651     , { -1070, -1580, -1130, -1580, -1070 }
01652     , { -460, -970, -520, -970, -460 }
01653     , { -830, -830, -1710, -1260, -1460 }
01654     }
01655     }
01656     , { { { 30, -350, 30, -200, 30 }
01657     , { -150, -870, -150, -200, -150 }
01658     , { -210, -350, -600, -210, -600 }
01659     , { 30, -870, 30, -320, 30 }
01660     , { -260, -940, -600, -260, -600 }
01661     }
01662     , { { -150, -350, -150, -200, -150 }
01663     , { -150, -1600, -150, -200, -150 }
01664     , { -350, -350, -600, -440, -600 }
01665     , { -1340, -3070, -1340, -2390, -1340 }
01666     , { -400, -960, -600, -400, -600 }
01667     }
01668     , { { -260, -870, -510, -260, -510 }
01669     , { -320, -1110, -510, -320, -510 }
01670     , { -620, -940, -820, -620, -820 }
01671     , { -320, -870, -510, -320, -510 }
01672     , { -260, -940, -820, -260, -820 }
01673     }
01674     , { { 30, -960, 30, -400, 30 }
01675     , { -860, -1880, -860, -1080, -860 }
01676     , { -400, -960, -600, -400, -600 }
01677     , { 30, -1370, 30, -2280, 30 }
01678     , { -400, -960, -600, -400, -600 }
01679     }
01680     , { { -210, -870, -510, -210, -510 }
01681     , { -320, -870, -510, -320, -510 }
01682     , { -210, -1250, -1130, -210, -1130 }
01683     , { -320, -870, -510, -320, -510 }
01684     , { -800, -1360, -800, -1550, -800 }
01685     }
01686     }
01687     , { { { -200, -430, -200, -430, -230 }
01688     , { -200, -600, -200, -600, -400 }
01689     , { -650, -1050, -650, -1050, -1390 }
01690     , { -230, -430, -570, -430, -230 }
01691     , { -650, -1050, -650, -1050, -1390 }
01692     }
01693     , { { -200, -600, -200, -600, -1390 }
01694     , { -200, -600, -200, -600, -1490 }
01695     , { -650, -1050, -650, -1050, -1390 }
01696     , { -1150, -1790, -1150, -1790, -1520 }
01697     , { -650, -1050, -650, -1050, -1390 }
01698     }
01699     , { { -400, -970, -570, -970, -400 }
01700     , { -400, -970, -570, -970, -400 }
01701     , { -870, -1270, -870, -1270, -1610 }
01702     , { -570, -970, -570, -970, -1300 }
01703     , { -870, -1270, -870, -1270, -1610 }
01704     }
01705     , { { -230, -430, -650, -430, -230 }
01706     , { -1300, -1320, -1750, -1320, -1300 }
01707     , { -650, -1050, -650, -1050, -1390 }
01708     , { -230, -430, -880, -430, -230 }
01709     , { -650, -1050, -650, -1050, -1390 }
01710     }
01711     , { { -570, -970, -570, -970, -1300 }
01712     , { -570, -970, -570, -970, -1300 }
01713     , { -1180, -1580, -1180, -1580, -1920 }
01714     , { -570, -970, -570, -970, -1300 }
01715     , { -860, -1260, -860, -1260, -2350 }
01716     }
01717     }
01718 }
```

```
01719 ,{{{ 240, 40, 190, -270, 240}
01720 ,{ -590, -1030, -650, -870, -590}
01721 ,{ -870, -1180, -1060, -870, -1010}
01722 ,{ 240, 40, 190, -270, 240}
01723 ,{ -870, -970, -1060, -870, -1010}
01724 }
01725 ,{{{ -780, -1210, -840, -870, -780}
01726 ,{ -1050, -1370, -1240, -1050, -1190}
01727 ,{ -870, -1210, -1060, -870, -1010}
01728 ,{ -780, -1220, -840, -1530, -780}
01729 ,{ -870, -1210, -1060, -870, -1010}
01730 }
01731 ,{{{ -870, -1180, -1060, -870, -1010}
01732 ,{ -870, -1210, -1060, -870, -1010}
01733 ,{ -870, -1180, -1060, -870, -1010}
01734 ,{ -870, -1210, -1060, -870, -1010}
01735 ,{ -870, -1180, -1060, -870, -1010}
01736 }
01737 ,{{{ 240, 40, 190, -270, 240}
01738 ,{ -590, -1030, -650, -1340, -590}
01739 ,{ -870, -1210, -1060, -870, -1010}
01740 ,{ 240, 40, 190, -270, 240}
01741 ,{ -870, -1210, -1060, -870, -1010}
01742 }
01743 ,{{{ -870, -970, -1060, -870, -1010}
01744 ,{ -870, -1210, -1060, -870, -1010}
01745 ,{ -870, -1180, -1060, -870, -1010}
01746 ,{ -870, -1210, -1060, -870, -1010}
01747 ,{ -970, -970, -1060, -1520, -1010}
01748 }
01749 }
01750 ,{{{ 190, 40, 190, -1470, 190}
01751 ,{ -890, -1030, -890, -1530, -890}
01752 ,{ -1060, -1210, -1060, -1470, -1060}
01753 ,{ 190, 40, 190, -1710, 190}
01754 ,{ -970, -970, -1060, -1470, -1060}
01755 }
01756 ,{{{ -1060, -1210, -1060, -1710, -1060}
01757 ,{ -1240, -1370, -1240, -1890, -1240}
01758 ,{ -1060, -1210, -1060, -1710, -1060}
01759 ,{ -1080, -1220, -1080, -1720, -1080}
01760 ,{ -1060, -1210, -1060, -1710, -1060}
01761 }
01762 ,{{{ -1060, -1210, -1060, -1470, -1060}
01763 ,{ -1060, -1210, -1060, -1710, -1060}
01764 ,{ -1060, -1210, -1060, -1470, -1060}
01765 ,{ -1060, -1210, -1060, -1710, -1060}
01766 ,{ -1060, -1210, -1060, -1470, -1060}
01767 }
01768 ,{{{ 190, 40, 190, -1530, 190}
01769 ,{ -890, -1030, -890, -1530, -890}
01770 ,{ -1060, -1210, -1060, -1710, -1060}
01771 ,{ 190, 40, 190, -1710, 190}
01772 ,{ -1060, -1210, -1060, -1710, -1060}
01773 }
01774 ,{{{ -970, -970, -1060, -1470, -1060}
01775 ,{ -1060, -1210, -1060, -1710, -1060}
01776 ,{ -1060, -1210, -1060, -1470, -1060}
01777 ,{ -1060, -1210, -1060, -1710, -1060}
01778 ,{ -970, -970, -1060, -1710, -1060}
01779 }
01780 }
01781 ,{{{ 240, -270, 180, -270, 240}
01782 ,{ -590, -1340, -650, -1340, -590}
01783 ,{ -1010, -1520, -1070, -1520, -1010}
01784 ,{ 240, -270, 180, -270, 240}
01785 ,{ -1010, -1520, -1070, -1520, -1010}
01786 }
01787 ,{{{ -780, -1520, -840, -1520, -780}
01788 ,{ -1190, -1700, -1250, -1700, -1190}
01789 ,{ -1010, -1520, -1070, -1520, -1010}
01790 ,{ -780, -1530, -840, -1530, -780}
01791 ,{ -1010, -1520, -1070, -1520, -1010}
01792 }
01793 ,{{{ -1010, -1520, -1070, -1520, -1010}
01794 ,{ -1010, -1520, -1070, -1520, -1010}
01795 ,{ -1010, -1520, -1070, -1520, -1010}
01796 ,{ -1010, -1520, -1070, -1520, -1010}
01797 ,{ -1010, -1520, -1070, -1520, -1010}
01798 }
01799 ,{{{ 240, -270, 180, -270, 240}
01800 ,{ -590, -1340, -650, -1340, -590}
01801 ,{ -1010, -1520, -1070, -1520, -1010}
01802 ,{ 240, -270, 180, -270, 240}
01803 ,{ -1010, -1520, -1070, -1520, -1010}
01804 }
01805 ,{{{ -1010, -1520, -1070, -1520, -1010}
```



```
01806     , { -1010, -1520, -1070, -1520, -1010}
01807     , { -1010, -1520, -1070, -1520, -1010}
01808     , { -1010, -1520, -1070, -1520, -1010}
01809     , { -1010, -1520, -1070, -1520, -1010}
01810     }
01811     }
01812     , {{{ 190, -1180, 190, -870, 190}
01813     , { -870, -1250, -890, -870, -890}
01814     , { -870, -1180, -1060, -870, -1060}
01815     , { 190, -1420, 190, -870, 190}
01816     , { -870, -1180, -1060, -870, -1060}
01817     }
01818     , {{{ -870, -1420, -1060, -870, -1060}
01819     , { -1050, -1600, -1240, -1050, -1240}
01820     , { -870, -1420, -1060, -870, -1060}
01821     , { -1080, -1440, -1080, -2130, -1080}
01822     , { -870, -1420, -1060, -870, -1060}
01823     }
01824     , {{{ -870, -1180, -1060, -870, -1060}
01825     , { -870, -1420, -1060, -870, -1060}
01826     , { -870, -1180, -1060, -870, -1060}
01827     , { -870, -1420, -1060, -870, -1060}
01828     , { -870, -1180, -1060, -870, -1060}
01829     }
01830     , {{{ 190, -1250, 190, -870, 190}
01831     , { -890, -1250, -890, -1940, -890}
01832     , { -870, -1420, -1060, -870, -1060}
01833     , { 190, -1420, 190, -2120, 190}
01834     , { -870, -1420, -1060, -870, -1060}
01835     }
01836     , {{{ -870, -1180, -1060, -870, -1060}
01837     , { -870, -1420, -1060, -870, -1060}
01838     , { -870, -1180, -1060, -870, -1060}
01839     , { -870, -1420, -1060, -870, -1060}
01840     , { -1060, -1420, -1060, -2120, -1060}
01841     }
01842     }
01843     , {{{ 130, -270, 130, -270, -1680}
01844     , { -700, -1340, -700, -1340, -1680}
01845     , { -1120, -1520, -1120, -1520, -1850}
01846     , { 130, -270, 130, -270, -1850}
01847     , { -1120, -1520, -1120, -1520, -1850}
01848     }
01849     , {{{ -890, -1520, -890, -1520, -1790}
01850     , { -1300, -1700, -1300, -1700, -1790}
01851     , { -1120, -1520, -1120, -1520, -1850}
01852     , { -890, -1530, -890, -1530, -1870}
01853     , { -1120, -1520, -1120, -1520, -1850}
01854     }
01855     , {{{ -1120, -1520, -1120, -1520, -1850}
01856     , { -1120, -1520, -1120, -1520, -1850}
01857     , { -1120, -1520, -1120, -1520, -1850}
01858     , { -1120, -1520, -1120, -1520, -1850}
01859     , { -1120, -1520, -1120, -1520, -1850}
01860     }
01861     , {{{ 130, -270, 130, -270, -1680}
01862     , { -700, -1340, -700, -1340, -1680}
01863     , { -1120, -1520, -1120, -1520, -1850}
01864     , { 130, -270, 130, -270, -1850}
01865     , { -1120, -1520, -1120, -1520, -1850}
01866     }
01867     , {{{ -1120, -1520, -1120, -1520, -1850}
01868     , { -1120, -1520, -1120, -1520, -1850}
01869     , { -1120, -1520, -1120, -1520, -1850}
01870     , { -1120, -1520, -1120, -1520, -1850}
01871     , { -1120, -1520, -1120, -1520, -1850}
01872     }
01873     }
01874     }
01875     , {{{ 800, 600, 740, 290, 800}
01876     , { 200, -140, 0, 200, 50}
01877     , { -310, -630, -510, -310, -450}
01878     , { 800, 600, 740, 290, 800}
01879     , { -310, -410, -510, -310, -450}
01880     }
01881     , {{{ 200, -140, 0, 200, 50}
01882     , { 200, -140, 0, 200, 50}
01883     , { -310, -650, -510, -310, -450}
01884     , { -550, -990, -610, -1300, -550}
01885     , { -310, -650, -510, -310, -450}
01886     }
01887     , {{{ -310, -630, -510, -310, -450}
01888     , { -310, -650, -510, -310, -450}
01889     , { -310, -630, -510, -310, -450}
01890     , { -310, -650, -510, -310, -450}
01891     , { -310, -630, -510, -310, -450}
01892     }
```

```
01893 ,{{ 800, 600, 740, 290, 800}
01894 ,{ -720, -1160, -780, -1470, -720}
01895 ,{ -310, -650, -510, -310, -450}
01896 ,{ 800, 600, 740, 290, 800}
01897 ,{ -310, -650, -510, -310, -450}
01898 }
01899 ,{{ -310, -410, -510, -310, -450}
01900 ,{ -310, -650, -510, -310, -450}
01901 ,{ -310, -630, -510, -310, -450}
01902 ,{ -310, -650, -510, -310, -450}
01903 ,{ -410, -410, -510, -960, -450}
01904 }
01905 }
01906 ,{{{ 740, 600, 740, -640, 740}
01907 ,{ 0, -140, 0, -640, 0}
01908 ,{ -510, -650, -510, -910, -510}
01909 ,{ 740, 600, 740, -1150, 740}
01910 ,{ -410, -410, -510, -910, -510}
01911 }
01912 ,{{ 0, -140, 0, -640, 0}
01913 ,{ 0, -140, 0, -640, 0}
01914 ,{ -510, -650, -510, -1150, -510}
01915 ,{ -850, -990, -850, -1490, -850}
01916 ,{ -510, -650, -510, -1150, -510}
01917 }
01918 ,{{ -510, -650, -510, -910, -510}
01919 ,{ -510, -650, -510, -1150, -510}
01920 ,{ -510, -650, -510, -910, -510}
01921 ,{ -510, -650, -510, -1150, -510}
01922 ,{ -510, -650, -510, -910, -510}
01923 }
01924 ,{{{ 740, 600, 740, -1150, 740}
01925 ,{ -1020, -1160, -1020, -1660, -1020}
01926 ,{ -510, -650, -510, -1150, -510}
01927 ,{ 740, 600, 740, -1150, 740}
01928 ,{ -510, -650, -510, -1150, -510}
01929 }
01930 ,{{ -410, -410, -510, -910, -510}
01931 ,{ -510, -650, -510, -1150, -510}
01932 ,{ -510, -650, -510, -910, -510}
01933 ,{ -510, -650, -510, -1150, -510}
01934 ,{ -410, -410, -510, -1150, -510}
01935 }
01936 }
01937 ,{{{ 800, 290, 740, 290, 800}
01938 ,{ 50, -450, 0, -450, 50}
01939 ,{ -450, -960, -510, -960, -450}
01940 ,{ 800, 290, 740, 290, 800}
01941 ,{ -450, -960, -510, -960, -450}
01942 }
01943 ,{{{ 50, -450, 0, -450, 50}
01944 ,{ 50, -450, 0, -450, 50}
01945 ,{ -450, -960, -510, -960, -450}
01946 ,{ -550, -1300, -610, -1300, -550}
01947 ,{ -450, -960, -510, -960, -450}
01948 }
01949 ,{{{ -450, -960, -510, -960, -450}
01950 ,{ -450, -960, -510, -960, -450}
01951 ,{ -450, -960, -510, -960, -450}
01952 ,{ -450, -960, -510, -960, -450}
01953 ,{ -450, -960, -510, -960, -450}
01954 }
01955 ,{{{ 800, 290, 740, 290, 800}
01956 ,{ -720, -1470, -780, -1470, -720}
01957 ,{ -450, -960, -510, -960, -450}
01958 ,{ 800, 290, 740, 290, 800}
01959 ,{ -450, -960, -510, -960, -450}
01960 }
01961 ,{{{ -450, -960, -510, -960, -450}
01962 ,{ -450, -960, -510, -960, -450}
01963 ,{ -450, -960, -510, -960, -450}
01964 ,{ -450, -960, -510, -960, -450}
01965 ,{ -450, -960, -510, -960, -450}
01966 }
01967 }
01968 ,{{{ 740, -360, 740, 200, 740}
01969 ,{ 200, -360, 0, 200, 0}
01970 ,{ -310, -630, -510, -310, -510}
01971 ,{ 740, -870, 740, -310, 740}
01972 ,{ -310, -630, -510, -310, -510}
01973 }
01974 ,{{ 200, -360, 0, 200, 0}
01975 ,{ 200, -360, 0, 200, 0}
01976 ,{ -310, -870, -510, -310, -510}
01977 ,{ -850, -1210, -850, -1900, -850}
01978 ,{ -310, -870, -510, -310, -510}
01979 }
```

```
01980 ,{{ -310, -630, -510, -310, -510}
01981 ,{ -310, -870, -510, -310, -510}
01982 ,{ -310, -630, -510, -310, -510}
01983 ,{ -310, -870, -510, -310, -510}
01984 ,{ -310, -630, -510, -310, -510}
01985 }
01986 ,{{ 740, -870, 740, -310, 740}
01987 ,{ -1020, -1380, -1020, -2070, -1020}
01988 ,{ -310, -870, -510, -310, -510}
01989 ,{ 740, -870, 740, -1560, 740}
01990 ,{ -310, -870, -510, -310, -510}
01991 }
01992 ,{{ -310, -630, -510, -310, -510}
01993 ,{ -310, -870, -510, -310, -510}
01994 ,{ -310, -630, -510, -310, -510}
01995 ,{ -310, -870, -510, -310, -510}
01996 ,{ -510, -870, -510, -1560, -510}
01997 }
01998 }
01999 ,{{{ 690, 290, 690, 290, -550}
02000 ,{ -50, -450, -50, -450, -550}
02001 ,{ -560, -960, -560, -960, -1300}
02002 ,{ 690, 290, 690, 290, -1300}
02003 ,{ -560, -960, -560, -960, -1300}
02004 }
02005 ,{{ -50, -450, -50, -450, -550}
02006 ,{ -50, -450, -50, -450, -550}
02007 ,{ -560, -960, -560, -960, -1300}
02008 ,{ -660, -1300, -660, -1300, -1640}
02009 ,{ -560, -960, -560, -960, -1300}
02010 }
02011 ,{{ -560, -960, -560, -960, -1300}
02012 ,{ -560, -960, -560, -960, -1300}
02013 ,{ -560, -960, -560, -960, -1300}
02014 ,{ -560, -960, -560, -960, -1300}
02015 ,{ -560, -960, -560, -960, -1300}
02016 }
02017 ,{{{ 690, 290, 690, 290, -1300}
02018 ,{ -830, -1470, -830, -1470, -1810}
02019 ,{ -560, -960, -560, -960, -1300}
02020 ,{ 690, 290, 690, 290, -1300}
02021 ,{ -560, -960, -560, -960, -1300}
02022 }
02023 ,{{ -560, -960, -560, -960, -1300}
02024 ,{ -560, -960, -560, -960, -1300}
02025 ,{ -560, -960, -560, -960, -1300}
02026 ,{ -560, -960, -560, -960, -1300}
02027 ,{ -560, -960, -560, -960, -1300}
02028 }
02029 }
02030 }
02031 ,{{{ 1170, 970, 1120, 780, 1170}
02032 ,{ 780, 440, 580, 780, 640}
02033 ,{ 480, 170, 280, 480, 340}
02034 ,{ 1170, 970, 1120, 660, 1170}
02035 ,{ 480, 170, 280, 480, 340}
02036 }
02037 ,{{ 780, 440, 580, 780, 640}
02038 ,{ 780, 440, 580, 780, 640}
02039 ,{ 470, 130, 270, 470, 330}
02040 ,{ -510, -950, -570, -1260, -510}
02041 ,{ 470, 130, 270, 470, 330}
02042 }
02043 ,{{ 490, 170, 290, 490, 340}
02044 ,{ 490, 140, 290, 490, 340}
02045 ,{ 480, 170, 280, 480, 340}
02046 ,{ 490, 140, 290, 490, 340}
02047 ,{ 480, 170, 280, 480, 340}
02048 }
02049 ,{{ 1170, 970, 1120, 660, 1170}
02050 ,{ -330, -770, -390, -1080, -330}
02051 ,{ 470, 130, 270, 470, 330}
02052 ,{ 1170, 970, 1120, 660, 1170}
02053 ,{ 470, 130, 270, 470, 330}
02054 }
02055 ,{{ 490, 170, 290, 490, 340}
02056 ,{ 490, 140, 290, 490, 340}
02057 ,{ 480, 170, 280, 480, 340}
02058 ,{ 490, 140, 290, 490, 340}
02059 ,{ -600, -600, -690, -1150, -640}
02060 }
02061 }
02062 ,{{{ 1120, 970, 1120, -60, 1120}
02063 ,{ 580, 440, 580, -60, 580}
02064 ,{ 280, 140, 280, -120, 280}
02065 ,{ 1120, 970, 1120, -350, 1120}
02066 ,{ 280, 140, 280, -120, 280}
```

```
02067      }
02068      ,{{ 580, 440, 580, -60, 580}
02069      ,{ 580, 440, 580, -60, 580}
02070      ,{ 270, 130, 270, -370, 270}
02071      ,{ -800, -950, -800, -1450, -800}
02072      ,{ 270, 130, 270, -370, 270}
02073      }
02074      ,{{ 290, 140, 290, -120, 290}
02075      ,{ 290, 140, 290, -350, 290}
02076      ,{ 280, 140, 280, -120, 280}
02077      ,{ 290, 140, 290, -350, 290}
02078      ,{ 280, 140, 280, -120, 280}
02079      }
02080      ,{{ 1120, 970, 1120, -370, 1120}
02081      ,{ -620, -770, -620, -1270, -620}
02082      ,{ 270, 130, 270, -370, 270}
02083      ,{ 1120, 970, 1120, -780, 1120}
02084      ,{ 270, 130, 270, -370, 270}
02085      }
02086      ,{{ 290, 140, 290, -120, 290}
02087      ,{ 290, 140, 290, -350, 290}
02088      ,{ 280, 140, 280, -120, 280}
02089      ,{ 290, 140, 290, -350, 290}
02090      ,{ -600, -600, -690, -1340, -690}
02091      }
02092      }
02093      ,{{{ 1170, 660, 1110, 660, 1170}
02094      ,{ 640, 130, 580, 130, 640}
02095      ,{ 340, -170, 280, -170, 340}
02096      ,{ 1170, 660, 1110, 660, 1170}
02097      ,{ 340, -170, 280, -170, 340}
02098      }
02099      ,{{ 640, 130, 580, 130, 640}
02100      ,{ 640, 130, 580, 130, 640}
02101      ,{ 330, -180, 270, -180, 330}
02102      ,{ -510, -1260, -570, -1260, -510}
02103      ,{ 330, -180, 270, -180, 330}
02104      }
02105      ,{{ 340, -160, 280, -160, 340}
02106      ,{ 340, -160, 280, -160, 340}
02107      ,{ 340, -170, 280, -170, 340}
02108      ,{ 340, -160, 280, -160, 340}
02109      ,{ 340, -170, 280, -170, 340}
02110      }
02111      ,{{{ 1170, 660, 1110, 660, 1170}
02112      ,{ -330, -1080, -390, -1080, -330}
02113      ,{ 330, -180, 270, -180, 330}
02114      ,{ 1170, 660, 1110, 660, 1170}
02115      ,{ 330, -180, 270, -180, 330}
02116      }
02117      ,{{{ 340, -160, 280, -160, 340}
02118      ,{ 340, -160, 280, -160, 340}
02119      ,{ 340, -170, 280, -170, 340}
02120      ,{ 340, -160, 280, -160, 340}
02121      ,{ -640, -1150, -700, -1150, -640}
02122      }
02123      }
02124      ,{{{ 1120, 220, 1120, 780, 1120}
02125      ,{ 780, 220, 580, 780, 580}
02126      ,{ 480, 170, 280, 480, 280}
02127      ,{ 1120, -70, 1120, 490, 1120}
02128      ,{ 480, 170, 280, 480, 280}
02129      }
02130      ,{{ 780, 220, 580, 780, 580}
02131      ,{ 780, 220, 580, 780, 580}
02132      ,{ 470, -80, 270, 470, 270}
02133      ,{ -800, -1160, -800, -1860, -800}
02134      ,{ 470, -80, 270, 470, 270}
02135      }
02136      ,{{ 490, 170, 290, 490, 290}
02137      ,{ 490, -70, 290, 490, 290}
02138      ,{ 480, 170, 280, 480, 280}
02139      ,{ 490, -70, 290, 490, 290}
02140      ,{ 480, 170, 280, 480, 280}
02141      }
02142      ,{{ 1120, -80, 1120, 470, 1120}
02143      ,{ -620, -980, -620, -1680, -620}
02144      ,{ 470, -80, 270, 470, 270}
02145      ,{ 1120, -490, 1120, -1190, 1120}
02146      ,{ 470, -80, 270, 470, 270}
02147      }
02148      ,{{ 490, 170, 290, 490, 290}
02149      ,{ 490, -70, 290, 490, 290}
02150      ,{ 480, 170, 280, 480, 280}
02151      ,{ 490, -70, 290, 490, 290}
02152      ,{ -690, -1050, -690, -1750, -690}
02153      }
```

```

02154     }
02155     ,{{ { 1060, 660, 1060, 660, 40}
02156     , { 530, 130, 530, 130, 40}
02157     , { 230, -170, 230, -170, -500}
02158     , { 1060, 660, 1060, 660, -500}
02159     , { 230, -170, 230, -170, -500}
02160     }
02161     ,{{ { 530, 130, 530, 130, 40}
02162     , { 530, 130, 530, 130, 40}
02163     , { 220, -180, 220, -180, -510}
02164     , { -620, -1260, -620, -1260, -1590}
02165     , { 220, -180, 220, -180, -510}
02166     }
02167     ,{{ { 230, -160, 230, -160, -500}
02168     , { 230, -160, 230, -160, -500}
02169     , { 230, -170, 230, -170, -500}
02170     , { 230, -160, 230, -160, -500}
02171     , { 230, -170, 230, -170, -500}
02172     }
02173     ,{{ { 1060, 660, 1060, 660, -510}
02174     , { -440, -1080, -440, -1080, -1410}
02175     , { 220, -180, 220, -180, -510}
02176     , { 1060, 660, 1060, 660, -920}
02177     , { 220, -180, 220, -180, -510}
02178     }
02179     ,{{ { 230, -160, 230, -160, -500}
02180     , { 230, -160, 230, -160, -500}
02181     , { 230, -170, 230, -170, -500}
02182     , { 230, -160, 230, -160, -500}
02183     , { -750, -1150, -750, -1150, -1480}
02184     }
02185     }
02186     }
02187     ,{{{ { 1350, 1160, 1300, 850, 1350}
02188     , { 850, 500, 650, 850, 700}
02189     , { 720, 400, 520, 720, 570}
02190     , { 1350, 1160, 1300, 850, 1350}
02191     , { 590, 270, 390, 590, 440}
02192     }
02193     ,{{ { 850, 500, 650, 850, 700}
02194     , { 850, 500, 650, 850, 700}
02195     , { 570, 220, 370, 570, 420}
02196     , { -460, -900, -520, -1210, -460}
02197     , { 570, 220, 370, 570, 420}
02198     }
02199     ,{{ { 720, 400, 520, 720, 570}
02200     , { 720, 370, 520, 720, 570}
02201     , { 720, 400, 520, 720, 570}
02202     , { 720, 370, 520, 720, 570}
02203     , { 590, 270, 390, 590, 440}
02204     }
02205     ,{{ { 1350, 1160, 1300, 850, 1350}
02206     , { -760, -1200, -820, -1510, -760}
02207     , { 570, 220, 370, 570, 420}
02208     , { 1350, 1160, 1300, 850, 1350}
02209     , { 570, 220, 370, 570, 420}
02210     }
02211     ,{{ { 720, 370, 520, 720, 570}
02212     , { 720, 370, 520, 720, 570}
02213     , { 280, -40, 80, 280, 130}
02214     , { 720, 370, 520, 720, 570}
02215     , { -320, -320, -420, -870, -360}
02216     }
02217     }
02218     ,{{{ { 1300, 1160, 1300, 120, 1300}
02219     , { 650, 500, 650, 0, 650}
02220     , { 520, 370, 520, 120, 520}
02221     , { 1300, 1160, 1300, -120, 1300}
02222     , { 390, 240, 390, -10, 390}
02223     }
02224     ,{{ { 650, 500, 650, 0, 650}
02225     , { 650, 500, 650, 0, 650}
02226     , { 370, 220, 370, -270, 370}
02227     , { -750, -900, -750, -1400, -750}
02228     , { 370, 220, 370, -270, 370}
02229     }
02230     ,{{ { 520, 370, 520, 120, 520}
02231     , { 520, 370, 520, -120, 520}
02232     , { 520, 370, 520, 120, 520}
02233     , { 520, 370, 520, -120, 520}
02234     , { 390, 240, 390, -10, 390}
02235     }
02236     ,{{ { 1300, 1160, 1300, -270, 1300}
02237     , { -1050, -1200, -1050, -1700, -1050}
02238     , { 370, 220, 370, -270, 370}
02239     , { 1300, 1160, 1300, -590, 1300}
02240     , { 370, 220, 370, -270, 370}

```

```

02241     }
02242     ,{{ 520, 370, 520, -120, 520}
02243     ,{ 520, 370, 520, -120, 520}
02244     ,{ 80, -60, 80, -320, 80}
02245     ,{ 520, 370, 520, -120, 520}
02246     ,{ -320, -320, -420, -1060, -420}
02247     }
02248     }
02249     ,{{{ 1350, 850, 1290, 850, 1350}
02250     ,{ 700, 190, 640, 190, 700}
02251     ,{ 570, 60, 510, 60, 570}
02252     ,{ 1350, 850, 1290, 850, 1350}
02253     ,{ 440, -60, 380, -60, 440}
02254     }
02255     ,{{{ 700, 190, 640, 190, 700}
02256     ,{ 700, 190, 640, 190, 700}
02257     ,{ 420, -80, 360, -80, 420}
02258     ,{ -460, -1210, -520, -1210, -460}
02259     ,{ 420, -80, 360, -80, 420}
02260     }
02261     ,{{{ 570, 60, 510, 60, 570}
02262     ,{ 570, 60, 510, 60, 570}
02263     ,{ 570, 60, 510, 60, 570}
02264     ,{ 570, 60, 510, 60, 570}
02265     ,{ 440, -60, 380, -60, 440}
02266     }
02267     ,{{{ 1350, 850, 1290, 850, 1350}
02268     ,{ -760, -1510, -820, -1510, -760}
02269     ,{ 420, -80, 360, -80, 420}
02270     ,{ 1350, 850, 1290, 850, 1350}
02271     ,{ 420, -80, 360, -80, 420}
02272     }
02273     ,{{{ 570, 60, 510, 60, 570}
02274     ,{ 570, 60, 510, 60, 570}
02275     ,{ 130, -370, 70, -370, 130}
02276     ,{ 570, 60, 510, 60, 570}
02277     ,{ -360, -870, -420, -870, -360}
02278     }
02279     }
02280     ,{{{ 1300, 400, 1300, 850, 1300}
02281     ,{ 850, 290, 650, 850, 650}
02282     ,{ 720, 400, 520, 720, 520}
02283     ,{ 1300, 160, 1300, 720, 1300}
02284     ,{ 590, 270, 390, 590, 390}
02285     }
02286     ,{{{ 850, 290, 650, 850, 650}
02287     ,{ 850, 290, 650, 850, 650}
02288     ,{ 570, 10, 370, 570, 370}
02289     ,{ -750, -1110, -750, -1810, -750}
02290     ,{ 570, 10, 370, 570, 370}
02291     }
02292     ,{{{ 720, 400, 520, 720, 520}
02293     ,{ 720, 160, 520, 720, 520}
02294     ,{ 720, 400, 520, 720, 520}
02295     ,{ 720, 160, 520, 720, 520}
02296     ,{ 590, 270, 390, 590, 390}
02297     }
02298     ,{{{ 1300, 10, 1300, 570, 1300}
02299     ,{ -1050, -1410, -1050, -2110, -1050}
02300     ,{ 570, 10, 370, 570, 370}
02301     ,{ 1300, -310, 1300, -1000, 1300}
02302     ,{ 570, 10, 370, 570, 370}
02303     }
02304     ,{{{ 720, 160, 520, 720, 520}
02305     ,{ 720, 160, 520, 720, 520}
02306     ,{ 280, -40, 80, 280, 80}
02307     ,{ 720, 160, 520, 720, 520}
02308     ,{ -420, -780, -420, -1470, -420}
02309     }
02310     }
02311     ,{{{ 1250, 850, 1250, 850, 100}
02312     ,{ 590, 190, 590, 190, 100}
02313     ,{ 460, 60, 460, 60, -270}
02314     ,{ 1250, 850, 1250, 850, -270}
02315     ,{ 330, -60, 330, -60, -400}
02316     }
02317     ,{{{ 590, 190, 590, 190, 100}
02318     ,{ 590, 190, 590, 190, 100}
02319     ,{ 310, -80, 310, -80, -420}
02320     ,{ -570, -1210, -570, -1210, -1540}
02321     ,{ 310, -80, 310, -80, -420}
02322     }
02323     ,{{{ 460, 60, 460, 60, -270}
02324     ,{ 460, 60, 460, 60, -270}
02325     ,{ 460, 60, 460, 60, -270}
02326     ,{ 460, 60, 460, 60, -270}
02327     ,{ 330, -60, 330, -60, -400}

```

```
02328     }
02329     ,{{ 1250, 850, 1250, 850, -420}
02330     ,{ -870, -1510, -870, -1510, -1840}
02331     ,{ 310, -80, 310, -80, -420}
02332     ,{ 1250, 850, 1250, 850, -740}
02333     ,{ 310, -80, 310, -80, -420}
02334     }
02335     ,{{ 460, 60, 460, 60, -270}
02336     ,{ 460, 60, 460, 60, -270}
02337     ,{ 20, -370, 20, -370, -710}
02338     ,{ 460, 60, 460, 60, -270}
02339     ,{ -470, -870, -470, -870, -1210}
02340     }
02341     }
02342     }
02343     ,{{{ 1350, 1160, 1300, 850, 1350}
02344     ,{ 850, 500, 650, 850, 700}
02345     ,{ 720, 400, 520, 720, 570}
02346     ,{ 1350, 1160, 1300, 850, 1350}
02347     ,{ 590, 270, 390, 590, 440}
02348     }
02349     ,{{ 850, 500, 650, 850, 700}
02350     ,{ 850, 500, 650, 850, 700}
02351     ,{ 570, 220, 370, 570, 420}
02352     ,{ -230, -670, -290, -980, -230}
02353     ,{ 570, 220, 370, 570, 420}
02354     }
02355     ,{{ 720, 400, 520, 720, 570}
02356     ,{ 720, 370, 520, 720, 570}
02357     ,{ 720, 400, 520, 720, 570}
02358     ,{ 720, 370, 520, 720, 570}
02359     ,{ 590, 270, 390, 590, 440}
02360     }
02361     ,{{ 1350, 1160, 1300, 850, 1350}
02362     ,{ -330, -770, -390, -1080, -330}
02363     ,{ 570, 220, 370, 570, 420}
02364     ,{ 1350, 1160, 1300, 850, 1350}
02365     ,{ 570, 220, 370, 570, 420}
02366     }
02367     ,{{ 720, 370, 520, 720, 570}
02368     ,{ 720, 370, 520, 720, 570}
02369     ,{ 480, 170, 280, 480, 340}
02370     ,{ 720, 370, 520, 720, 570}
02371     ,{ -90, -320, -90, -810, -360}
02372     }
02373     }
02374     ,{{{ 1300, 1160, 1300, 540, 1300}
02375     ,{ 650, 500, 650, 10, 650}
02376     ,{ 540, 370, 520, 540, 520}
02377     ,{ 1300, 1160, 1300, -120, 1300}
02378     ,{ 390, 240, 390, -10, 390}
02379     }
02380     ,{{ 650, 500, 650, 0, 650}
02381     ,{ 650, 500, 650, 0, 650}
02382     ,{ 370, 220, 370, -100, 370}
02383     ,{ -530, -670, -530, -1170, -530}
02384     ,{ 370, 220, 370, -270, 370}
02385     }
02386     ,{{ 540, 370, 520, 540, 520}
02387     ,{ 520, 370, 520, 10, 520}
02388     ,{ 540, 370, 520, 540, 520}
02389     ,{ 520, 370, 520, -120, 520}
02390     ,{ 390, 240, 390, -10, 390}
02391     }
02392     ,{{ 1300, 1160, 1300, -270, 1300}
02393     ,{ -620, -770, -620, -1270, -620}
02394     ,{ 370, 220, 370, -270, 370}
02395     ,{ 1300, 1160, 1300, -590, 1300}
02396     ,{ 370, 220, 370, -270, 370}
02397     }
02398     ,{{ 520, 370, 520, -120, 520}
02399     ,{ 520, 370, 520, -120, 520}
02400     ,{ 280, 140, 280, -120, 280}
02401     ,{ 520, 370, 520, -120, 520}
02402     ,{ -90, -320, -90, -810, -420}
02403     }
02404     }
02405     ,{{{ 1350, 850, 1290, 850, 1350}
02406     ,{ 700, 190, 640, 190, 700}
02407     ,{ 570, 60, 510, 60, 570}
02408     ,{ 1350, 850, 1290, 850, 1350}
02409     ,{ 440, -60, 380, -60, 440}
02410     }
02411     ,{{ 700, 190, 640, 190, 700}
02412     ,{ 700, 190, 640, 190, 700}
02413     ,{ 420, -80, 360, -80, 420}
02414     ,{ -230, -980, -290, -980, -230}
```

```

02415     , { 420, -80, 360, -80, 420}
02416     }
02417     , { { 570, 60, 510, 60, 570}
02418     , { 570, 60, 510, 60, 570}
02419     , { 570, 60, 510, 60, 570}
02420     , { 570, 60, 510, 60, 570}
02421     , { 440, -60, 380, -60, 440}
02422     }
02423     , { { 1350, 850, 1290, 850, 1350}
02424     , { -330, -1070, -390, -1080, -330}
02425     , { 420, -80, 360, -80, 420}
02426     , { 1350, 850, 1290, 850, 1350}
02427     , { 420, -80, 360, -80, 420}
02428     }
02429     , { { 570, 60, 510, 60, 570}
02430     , { 570, 60, 510, 60, 570}
02431     , { 340, -170, 280, -170, 340}
02432     , { 570, 60, 510, 60, 570}
02433     , { -360, -830, -420, -870, -360}
02434     }
02435     }
02436     , { { { 1300, 400, 1300, 850, 1300}
02437     , { 850, 290, 650, 850, 650}
02438     , { 720, 400, 520, 720, 520}
02439     , { 1300, 160, 1300, 720, 1300}
02440     , { 590, 270, 390, 590, 390}
02441     }
02442     , { { 850, 290, 650, 850, 650}
02443     , { 850, 290, 650, 850, 650}
02444     , { 570, 10, 370, 570, 370}
02445     , { -530, -890, -530, -1580, -530}
02446     , { 570, 10, 370, 570, 370}
02447     }
02448     , { { 720, 400, 520, 720, 520}
02449     , { 720, 160, 520, 720, 520}
02450     , { 720, 400, 520, 720, 520}
02451     , { 720, 160, 520, 720, 520}
02452     , { 590, 270, 390, 590, 390}
02453     }
02454     , { { 1300, 10, 1300, 570, 1300}
02455     , { -620, -980, -620, -1080, -620}
02456     , { 570, 10, 370, 570, 370}
02457     , { 1300, -310, 1300, -1000, 1300}
02458     , { 570, 10, 370, 570, 370}
02459     }
02460     , { { 720, 170, 520, 720, 520}
02461     , { 720, 160, 520, 720, 520}
02462     , { 480, 170, 280, 480, 280}
02463     , { 720, 160, 520, 720, 520}
02464     , { -420, -780, -420, -1470, -420}
02465     }
02466     }
02467     , { { { 1250, 850, 1250, 850, 100}
02468     , { 590, 190, 590, 190, 100}
02469     , { 460, 60, 460, 60, -270}
02470     , { 1250, 850, 1250, 850, -230}
02471     , { 330, -60, 330, -60, -400}
02472     }
02473     , { { 590, 190, 590, 190, 100}
02474     , { 590, 190, 590, 190, 100}
02475     , { 310, -80, 310, -80, -420}
02476     , { -340, -980, -340, -980, -1320}
02477     , { 310, -80, 310, -80, -420}
02478     }
02479     , { { 460, 60, 460, 60, -270}
02480     , { 460, 60, 460, 60, -270}
02481     , { 460, 60, 460, 60, -270}
02482     , { 460, 60, 460, 60, -270}
02483     , { 330, -60, 330, -60, -400}
02484     }
02485     , { { 1250, 850, 1250, 850, -230}
02486     , { -440, -1080, -440, -1080, -1300}
02487     , { 310, -80, 310, -80, -420}
02488     , { 1250, 850, 1250, 850, -230}
02489     , { 310, -80, 310, -80, -420}
02490     }
02491     , { { 460, 60, 460, 60, -270}
02492     , { 460, 60, 460, 60, -270}
02493     , { 230, -170, 230, -170, -500}
02494     , { 460, 60, 460, 60, -270}
02495     , { -470, -870, -470, -870, -1210}
02496     }
02497     }
02498     }
02499     }
02500     , { { { { INF, INF, INF, INF, INF}
02501     , { INF, INF, INF, INF, INF}

```



```
02502     , {   INF,   INF,   INF,   INF,   INF }
02503     , {   INF,   INF,   INF,   INF,   INF }
02504     , {   INF,   INF,   INF,   INF,   INF }
02505     }
02506     , { {   INF,   INF,   INF,   INF,   INF }
02507     , {   INF,   INF,   INF,   INF,   INF }
02508     , {   INF,   INF,   INF,   INF,   INF }
02509     , {   INF,   INF,   INF,   INF,   INF }
02510     , {   INF,   INF,   INF,   INF,   INF }
02511     }
02512     , { {   INF,   INF,   INF,   INF,   INF }
02513     , {   INF,   INF,   INF,   INF,   INF }
02514     , {   INF,   INF,   INF,   INF,   INF }
02515     , {   INF,   INF,   INF,   INF,   INF }
02516     , {   INF,   INF,   INF,   INF,   INF }
02517     }
02518     , { {   INF,   INF,   INF,   INF,   INF }
02519     , {   INF,   INF,   INF,   INF,   INF }
02520     , {   INF,   INF,   INF,   INF,   INF }
02521     , {   INF,   INF,   INF,   INF,   INF }
02522     , {   INF,   INF,   INF,   INF,   INF }
02523     }
02524     , { {   INF,   INF,   INF,   INF,   INF }
02525     , {   INF,   INF,   INF,   INF,   INF }
02526     , {   INF,   INF,   INF,   INF,   INF }
02527     , {   INF,   INF,   INF,   INF,   INF }
02528     , {   INF,   INF,   INF,   INF,   INF }
02529     }
02530     }
02531     , { { {   INF,   INF,   INF,   INF,   INF }
02532     , {   INF,   INF,   INF,   INF,   INF }
02533     , {   INF,   INF,   INF,   INF,   INF }
02534     , {   INF,   INF,   INF,   INF,   INF }
02535     , {   INF,   INF,   INF,   INF,   INF }
02536     }
02537     , { {   INF,   INF,   INF,   INF,   INF }
02538     , {   INF,   INF,   INF,   INF,   INF }
02539     , {   INF,   INF,   INF,   INF,   INF }
02540     , {   INF,   INF,   INF,   INF,   INF }
02541     , {   INF,   INF,   INF,   INF,   INF }
02542     }
02543     , { {   INF,   INF,   INF,   INF,   INF }
02544     , {   INF,   INF,   INF,   INF,   INF }
02545     , {   INF,   INF,   INF,   INF,   INF }
02546     , {   INF,   INF,   INF,   INF,   INF }
02547     , {   INF,   INF,   INF,   INF,   INF }
02548     }
02549     , { {   INF,   INF,   INF,   INF,   INF }
02550     , {   INF,   INF,   INF,   INF,   INF }
02551     , {   INF,   INF,   INF,   INF,   INF }
02552     , {   INF,   INF,   INF,   INF,   INF }
02553     , {   INF,   INF,   INF,   INF,   INF }
02554     }
02555     , { {   INF,   INF,   INF,   INF,   INF }
02556     , {   INF,   INF,   INF,   INF,   INF }
02557     , {   INF,   INF,   INF,   INF,   INF }
02558     , {   INF,   INF,   INF,   INF,   INF }
02559     , {   INF,   INF,   INF,   INF,   INF }
02560     }
02561     }
02562     , { { {   INF,   INF,   INF,   INF,   INF }
02563     , {   INF,   INF,   INF,   INF,   INF }
02564     , {   INF,   INF,   INF,   INF,   INF }
02565     , {   INF,   INF,   INF,   INF,   INF }
02566     , {   INF,   INF,   INF,   INF,   INF }
02567     }
02568     , { {   INF,   INF,   INF,   INF,   INF }
02569     , {   INF,   INF,   INF,   INF,   INF }
02570     , {   INF,   INF,   INF,   INF,   INF }
02571     , {   INF,   INF,   INF,   INF,   INF }
02572     , {   INF,   INF,   INF,   INF,   INF }
02573     }
02574     , { {   INF,   INF,   INF,   INF,   INF }
02575     , {   INF,   INF,   INF,   INF,   INF }
02576     , {   INF,   INF,   INF,   INF,   INF }
02577     , {   INF,   INF,   INF,   INF,   INF }
02578     , {   INF,   INF,   INF,   INF,   INF }
02579     }
02580     , { {   INF,   INF,   INF,   INF,   INF }
02581     , {   INF,   INF,   INF,   INF,   INF }
02582     , {   INF,   INF,   INF,   INF,   INF }
02583     , {   INF,   INF,   INF,   INF,   INF }
02584     , {   INF,   INF,   INF,   INF,   INF }
02585     }
02586     , { {   INF,   INF,   INF,   INF,   INF }
02587     , {   INF,   INF,   INF,   INF,   INF }
02588     , {   INF,   INF,   INF,   INF,   INF }
```

```

02589     , {   INF,   INF,   INF,   INF,   INF }
02590     , {   INF,   INF,   INF,   INF,   INF }
02591     }
02592 }
02593 , { {   INF,   INF,   INF,   INF,   INF }
02594     , {   INF,   INF,   INF,   INF,   INF }
02595     , {   INF,   INF,   INF,   INF,   INF }
02596     , {   INF,   INF,   INF,   INF,   INF }
02597     , {   INF,   INF,   INF,   INF,   INF }
02598     }
02599     , { {   INF,   INF,   INF,   INF,   INF }
02600     , {   INF,   INF,   INF,   INF,   INF }
02601     , {   INF,   INF,   INF,   INF,   INF }
02602     , {   INF,   INF,   INF,   INF,   INF }
02603     , {   INF,   INF,   INF,   INF,   INF }
02604     }
02605     , { {   INF,   INF,   INF,   INF,   INF }
02606     , {   INF,   INF,   INF,   INF,   INF }
02607     , {   INF,   INF,   INF,   INF,   INF }
02608     , {   INF,   INF,   INF,   INF,   INF }
02609     , {   INF,   INF,   INF,   INF,   INF }
02610     }
02611     , { {   INF,   INF,   INF,   INF,   INF }
02612     , {   INF,   INF,   INF,   INF,   INF }
02613     , {   INF,   INF,   INF,   INF,   INF }
02614     , {   INF,   INF,   INF,   INF,   INF }
02615     , {   INF,   INF,   INF,   INF,   INF }
02616     }
02617     , { {   INF,   INF,   INF,   INF,   INF }
02618     , {   INF,   INF,   INF,   INF,   INF }
02619     , {   INF,   INF,   INF,   INF,   INF }
02620     , {   INF,   INF,   INF,   INF,   INF }
02621     , {   INF,   INF,   INF,   INF,   INF }
02622     }
02623     }
02624     , { { {   INF,   INF,   INF,   INF,   INF }
02625         , {   INF,   INF,   INF,   INF,   INF }
02626         , {   INF,   INF,   INF,   INF,   INF }
02627         , {   INF,   INF,   INF,   INF,   INF }
02628         , {   INF,   INF,   INF,   INF,   INF }
02629         }
02630         , { {   INF,   INF,   INF,   INF,   INF }
02631         , {   INF,   INF,   INF,   INF,   INF }
02632         , {   INF,   INF,   INF,   INF,   INF }
02633         , {   INF,   INF,   INF,   INF,   INF }
02634         , {   INF,   INF,   INF,   INF,   INF }
02635         }
02636         , { {   INF,   INF,   INF,   INF,   INF }
02637         , {   INF,   INF,   INF,   INF,   INF }
02638         , {   INF,   INF,   INF,   INF,   INF }
02639         , {   INF,   INF,   INF,   INF,   INF }
02640         , {   INF,   INF,   INF,   INF,   INF }
02641         }
02642         , { {   INF,   INF,   INF,   INF,   INF }
02643         , {   INF,   INF,   INF,   INF,   INF }
02644         , {   INF,   INF,   INF,   INF,   INF }
02645         , {   INF,   INF,   INF,   INF,   INF }
02646         , {   INF,   INF,   INF,   INF,   INF }
02647         }
02648         , { {   INF,   INF,   INF,   INF,   INF }
02649         , {   INF,   INF,   INF,   INF,   INF }
02650         , {   INF,   INF,   INF,   INF,   INF }
02651         , {   INF,   INF,   INF,   INF,   INF }
02652         , {   INF,   INF,   INF,   INF,   INF }
02653         }
02654         }
02655     }
02656     , { { { {   540,   -90,   540,   180,   -90 }
02657         , {   540,  -100,   540,   180,   -90 }
02658         , {   180,   -90,  -460,   180,  -460 }
02659         , {    30,  -150,  -260,    30,  -210 }
02660         , {  -200,  -200,  -400,  -230,  -570 }
02661         }
02662         , { {   180,  -350,  -660,   180,  -660 }
02663         , {   180,  -580,  -660,   180,  -660 }
02664         , {  -430,  -600,  -970,  -430,  -830 }
02665         , {  -350,  -350,  -870,  -960,  -870 }
02666         , {  -430,  -600,  -970,  -430,  -970 }
02667         }
02668         , { {    30,  -150,  -510,    30,   -90 }
02669         , {   -90,  -220,  -510,  -390,   -90 }
02670         , {    20,  -600,  -520,    20,  -520 }
02671         , {    30,  -150,  -510,    30,  -510 }
02672         , {  -200,  -200,  -570,  -650,  -570 }
02673         }
02674         , { {   540,  -100,   540,  -400,  -210 }
02675         , {   540,  -100,   540, -1240,  -810 }

```

```
02676     , { -430, -600, -970, -430, -970}
02677     , { -200, -200, -260, -400, -210}
02678     , { -430, -600, -970, -430, -970}
02679     }
02680     , { { 180, -90, -400, 180, -460}
02681     , { 30, -150, -510, 30, -510}
02682     , { 180, -90, -460, 180, -460}
02683     , { 30, -150, -510, 30, -510}
02684     , { -230, -1390, -400, -230, -1300}
02685     }
02686     }
02687     , { { { 10, -90, 10, -500, -320}
02688     , { 10, -150, 10, -860, -510}
02689     , { -90, -90, -460, -500, -460}
02690     , { -150, -150, -320, -860, -320}
02691     , { -200, -200, -400, -1300, -570}
02692     }
02693     , { { -580, -580, -660, -1070, -660}
02694     , { -580, -580, -660, -1340, -660}
02695     , { -600, -600, -970, -1070, -970}
02696     , { -870, -1600, -1110, -1880, -870}
02697     , { -600, -600, -970, -1320, -970}
02698     }
02699     , { { -150, -150, -510, -500, -510}
02700     , { -220, -220, -1150, -860, -510}
02701     , { -500, -1070, -750, -500, -520}
02702     , { -150, -150, -510, -860, -510}
02703     , { -200, -200, -570, -1750, -570}
02704     }
02705     , { { 10, -200, 10, -1080, -320}
02706     , { 10, -970, 10, -2450, -1160}
02707     , { -600, -600, -970, -1320, -970}
02708     , { -200, -200, -320, -1080, -320}
02709     , { -600, -600, -970, -1320, -970}
02710     }
02711     , { { -90, -90, -400, -570, -460}
02712     , { -150, -150, -510, -860, -510}
02713     , { -90, -90, -460, -570, -460}
02714     , { -150, -150, -510, -860, -510}
02715     , { -400, -1490, -400, -1300, -1300}
02716     }
02717     }
02718     , { { { 540, -100, 540, -400, -210}
02719     , { 540, -100, 540, -600, -1130}
02720     , { -540, -540, -760, -540, -1070}
02721     , { -210, -350, -620, -400, -210}
02722     , { -650, -650, -870, -650, -1180}
02723     }
02724     , { { -350, -350, -940, -740, -1250}
02725     , { -740, -740, -960, -740, -1270}
02726     , { -1050, -1050, -1270, -1050, -1580}
02727     , { -350, -350, -940, -960, -1250}
02728     , { -1050, -1050, -1270, -1050, -1580}
02729     }
02730     , { { -600, -600, -820, -600, -1130}
02731     , { -600, -600, -820, -600, -1130}
02732     , { -600, -600, -820, -600, -1130}
02733     , { -600, -600, -820, -600, -1130}
02734     , { -650, -650, -870, -650, -1180}
02735     }
02736     , { { 540, -100, 540, -400, -210}
02737     , { 540, -100, 540, -1240, -1530}
02738     , { -1050, -1050, -1270, -1050, -1580}
02739     , { -210, -440, -620, -400, -210}
02740     , { -1050, -1050, -1270, -1050, -1580}
02741     }
02742     , { { -540, -540, -760, -540, -1070}
02743     , { -600, -600, -820, -600, -1130}
02744     , { -540, -540, -760, -540, -1070}
02745     , { -600, -600, -820, -600, -1130}
02746     , { -1390, -1390, -1610, -1390, -1920}
02747     }
02748     }
02749     , { { { 180, -630, -320, 180, -320}
02750     , { 180, -1340, -510, 180, -510}
02751     , { 180, -630, -460, 180, -460}
02752     , { 30, -1340, -320, 30, -320}
02753     , { -230, -1150, -570, -230, -570}
02754     }
02755     , { { 180, -1790, -660, 180, -660}
02756     , { 180, -2010, -660, 180, -660}
02757     , { -430, -1790, -970, -430, -970}
02758     , { -870, -3070, -870, -1370, -870}
02759     , { -430, -1790, -970, -430, -970}
02760     }
02761     , { { 30, -630, -510, 30, -510}
02762     , { -390, -1650, -510, -390, -510}
```

```

02763      , {      20,   -630,   -520,    20,   -520}
02764      , {      30,  -1340,   -510,    30,   -510}
02765      , {   -570, -1150,   -570,   -880,   -570}
02766      }
02767      , {{   -320, -1790,   -320,   -430,   -320}
02768      , { -1160, -1980, -1160, -1870, -1160}
02769      , {   -430, -1790,   -970,   -430,   -970}
02770      , {   -320, -2390,   -320, -2280,   -320}
02771      , {   -430, -1790,   -970,   -430,   -970}
02772      }
02773      , {{    180, -1040,   -460,    180,   -460}
02774      , {     30,  -1340,   -510,     30,   -510}
02775      , {    180, -1040,   -460,    180,   -460}
02776      , {     30,  -1340,   -510,     30,   -510}
02777      , {   -230, -1520, -1300,   -230, -1300}
02778      }
02779      }
02780      , {{{   -90,   -400,   -260,   -400,   -90}
02781      , {   -90,   -600,   -820,   -600,   -90}
02782      , {  -540,   -540,   -550,   -540,   -830}
02783      , {  -260,   -400,   -260,   -400,   -800}
02784      , {  -650,   -650,   -870,   -650,   -860}
02785      }
02786      , {{{  -740,   -740,   -940,   -740,   -830}
02787      , {  -740,   -740,   -960,   -740, -1240}
02788      , {  -830, -1050, -1270, -1050,   -830}
02789      , {  -940,   -960,   -940,   -960, -1360}
02790      , { -1050, -1050, -1270, -1050, -1260}
02791      }
02792      , {{{   -90,   -600,   -820,   -600,   -90}
02793      , {   -90,   -600,   -820,   -600,   -90}
02794      , {  -600,   -600,   -820,   -600, -1710}
02795      , {  -600,   -600,   -820,   -600,   -800}
02796      , {  -650,   -650,   -870,   -650,   -860}
02797      }
02798      , {{{  -260,   -400,   -260,   -400,   -810}
02799      , {  -810, -1240, -1220, -1240,   -810}
02800      , { -1050, -1050, -1270, -1050, -1260}
02801      , {  -260,   -400,   -260,   -400, -1550}
02802      , { -1050, -1050, -1270, -1050, -1260}
02803      }
02804      , {{{  -540,   -540,   -550,   -540,   -800}
02805      , {  -600,   -600,   -820,   -600,   -800}
02806      , {  -540,   -540,   -550,   -540, -1460}
02807      , {  -600,   -600,   -820,   -600,   -800}
02808      , { -1390, -1390, -1610, -1390, -2350}
02809      }
02810      }
02811      }
02812      , {{{    50,     50,   -320,     50,   -320}
02813      , {     50,   -130,   -490,     50,   -490}
02814      , {  -400,   -580,   -940,   -400,   -940}
02815      , {     50,     50,   -320,   -320,   -320}
02816      , {  -400,   -540,   -940,   -400,   -940}
02817      }
02818      , {{{    50,   -130,   -490,     50,   -490}
02819      , {     50,   -130,   -490,     50,   -490}
02820      , {  -400,   -580,   -940,   -400,   -940}
02821      , { -1320, -1320, -1680, -1770, -1680}
02822      , {  -400,   -580,   -940,   -400,   -940}
02823      }
02824      , {{{  -320,   -490,   -860,   -320,   -860}
02825      , {  -320,   -490,   -860,   -320,   -860}
02826      , {  -620,   -800, -1160,   -620, -1160}
02827      , {  -320,   -490,   -860,   -320,   -860}
02828      , {  -620,   -800, -1160,   -620, -1160}
02829      }
02830      , {{{    50,     50,   -320,   -400,   -320}
02831      , {  -840,   -840, -1210, -1290, -1210}
02832      , {  -400,   -580,   -940,   -400,   -940}
02833      , {     50,     50,   -320,   -400,   -320}
02834      , {  -400,   -580,   -940,   -400,   -940}
02835      }
02836      , {{{  -320,   -490,   -860,   -320,   -860}
02837      , {  -320,   -490,   -860,   -320,   -860}
02838      , {  -930, -1110, -1470,   -930, -1470}
02839      , {  -320,   -490,   -860,   -320,   -860}
02840      , {  -540,   -540, -1150, -1230, -1150}
02841      }
02842      }
02843      , {{{    50,     50,   -320,   -840,   -320}
02844      , {   -130,   -130,   -490,   -840,   -490}
02845      , {  -580,   -580,   -940, -1270,   -940}
02846      , {     50,     50,   -320, -1210,   -320}
02847      , {  -540,   -540,   -940, -1270,   -940}
02848      }
02849      , {{{  -130,   -130,   -490,   -840,   -490}

```

```
02850     , { -130, -130, -490, -840, -490 }
02851     , { -580, -580, -940, -1290, -940 }
02852     , { -1320, -1320, -1680, -2030, -1680 }
02853     , { -580, -580, -940, -1290, -940 }
02854     }
02855     , { { -490, -490, -860, -1210, -860 }
02856     , { -490, -490, -860, -1210, -860 }
02857     , { -800, -800, -1160, -1270, -1160 }
02858     , { -490, -490, -860, -1210, -860 }
02859     , { -800, -800, -1160, -1270, -1160 }
02860     }
02861     , { { 50, 50, -320, -1290, -320 }
02862     , { -840, -840, -1210, -1560, -1210 }
02863     , { -580, -580, -940, -1290, -940 }
02864     , { 50, 50, -320, -1920, -320 }
02865     , { -580, -580, -940, -1290, -940 }
02866     }
02867     , { { -490, -490, -860, -1210, -860 }
02868     , { -490, -490, -860, -1210, -860 }
02869     , { -1110, -1110, -1470, -1580, -1470 }
02870     , { -490, -490, -860, -1210, -860 }
02871     , { -540, -540, -1150, -1500, -1150 }
02872     }
02873     }
02874     , { { { -400, -400, -620, -400, -930 }
02875     , { -580, -580, -800, -580, -1110 }
02876     , { -1030, -1030, -1250, -1030, -1560 }
02877     , { -400, -400, -620, -400, -930 }
02878     , { -1030, -1030, -1250, -1030, -1560 }
02879     }
02880     , { { -580, -580, -800, -580, -1110 }
02881     , { -580, -580, -800, -580, -1110 }
02882     , { -1030, -1030, -1250, -1030, -1560 }
02883     , { -1750, -1770, -1750, -1770, -2060 }
02884     , { -1030, -1030, -1250, -1030, -1560 }
02885     }
02886     , { { -940, -940, -1160, -940, -1470 }
02887     , { -940, -940, -1160, -940, -1470 }
02888     , { -1250, -1250, -1470, -1250, -1780 }
02889     , { -940, -940, -1160, -940, -1470 }
02890     , { -1250, -1250, -1470, -1250, -1780 }
02891     }
02892     , { { -400, -400, -620, -400, -930 }
02893     , { -1270, -1290, -1270, -1290, -1580 }
02894     , { -1030, -1030, -1250, -1030, -1560 }
02895     , { -400, -400, -620, -400, -930 }
02896     , { -1030, -1030, -1250, -1030, -1560 }
02897     }
02898     , { { -940, -940, -1160, -940, -1470 }
02899     , { -940, -940, -1160, -940, -1470 }
02900     , { -1560, -1560, -1780, -1560, -2090 }
02901     , { -940, -940, -1160, -940, -1470 }
02902     , { -1230, -1230, -1450, -1230, -1760 }
02903     }
02904     }
02905     , { { { 50, -1320, -320, 50, -320 }
02906     , { 50, -1320, -490, 50, -490 }
02907     , { -400, -1750, -940, -400, -940 }
02908     , { -320, -1680, -320, -320, -320 }
02909     , { -400, -1750, -940, -400, -940 }
02910     }
02911     , { { 50, -1320, -490, 50, -490 }
02912     , { 50, -1320, -490, 50, -490 }
02913     , { -400, -1770, -940, -400, -940 }
02914     , { -1680, -2510, -1680, -2390, -1680 }
02915     , { -400, -1770, -940, -400, -940 }
02916     }
02917     , { { -320, -1680, -860, -320, -860 }
02918     , { -320, -1680, -860, -320, -860 }
02919     , { -620, -1750, -1160, -620, -1160 }
02920     , { -320, -1680, -860, -320, -860 }
02921     , { -620, -1750, -1160, -620, -1160 }
02922     }
02923     , { { -320, -1770, -320, -400, -320 }
02924     , { -1210, -2030, -1210, -1920, -1210 }
02925     , { -400, -1770, -940, -400, -940 }
02926     , { -320, -2390, -320, -2280, -320 }
02927     , { -400, -1770, -940, -400, -940 }
02928     }
02929     , { { -320, -1680, -860, -320, -860 }
02930     , { -320, -1680, -860, -320, -860 }
02931     , { -930, -2060, -1470, -930, -1470 }
02932     , { -320, -1680, -860, -320, -860 }
02933     , { -1150, -1970, -1150, -1860, -1150 }
02934     }
02935     }
02936     , { { { -400, -400, -620, -400, -540 }
```

```
02937 , { -540, -580, -800, -580, -540 }
02938 , { -1030, -1030, -1250, -1030, -1230 }
02939 , { -400, -400, -620, -400, -1150 }
02940 , { -1030, -1030, -1250, -1030, -1230 }
02941 }
02942 , { { -540, -580, -800, -580, -540 }
02943 , { -540, -580, -800, -580, -540 }
02944 , { -1030, -1030, -1250, -1030, -1230 }
02945 , { -1750, -1770, -1750, -1770, -1970 }
02946 , { -1030, -1030, -1250, -1030, -1230 }
02947 }
02948 , { { -940, -940, -1160, -940, -1150 }
02949 , { -940, -940, -1160, -940, -1150 }
02950 , { -1250, -1250, -1470, -1250, -1450 }
02951 , { -940, -940, -1160, -940, -1150 }
02952 , { -1250, -1250, -1470, -1250, -1450 }
02953 }
02954 , { { -400, -400, -620, -400, -1230 }
02955 , { -1270, -1290, -1270, -1290, -1500 }
02956 , { -1030, -1030, -1250, -1030, -1230 }
02957 , { -400, -400, -620, -400, -1860 }
02958 , { -1030, -1030, -1250, -1030, -1230 }
02959 }
02960 , { { -940, -940, -1160, -940, -1150 }
02961 , { -940, -940, -1160, -940, -1150 }
02962 , { -1560, -1560, -1780, -1560, -1760 }
02963 , { -940, -940, -1160, -940, -1150 }
02964 , { -1230, -1230, -1450, -1230, -1440 }
02965 }
02966 }
02967 }
02968 , { { { 210, 210, -160, -240, -160 }
02969 , { -870, -870, -1230, -870, -1230 }
02970 , { -870, -1040, -1410, -870, -1410 }
02971 , { 210, 210, -160, -240, -160 }
02972 , { -800, -800, -1410, -870, -1410 }
02973 }
02974 , { { -870, -1040, -1410, -870, -1410 }
02975 , { -1050, -1220, -1590, -1050, -1590 }
02976 , { -870, -1040, -1410, -870, -1410 }
02977 , { -1060, -1060, -1420, -1510, -1420 }
02978 , { -870, -1040, -1410, -870, -1410 }
02979 }
02980 , { { -870, -1040, -1410, -870, -1410 }
02981 , { -870, -1040, -1410, -870, -1410 }
02982 , { -870, -1040, -1410, -870, -1410 }
02983 , { -870, -1040, -1410, -870, -1410 }
02984 , { -870, -1040, -1410, -870, -1410 }
02985 }
02986 , { { 210, 210, -160, -240, -160 }
02987 , { -870, -870, -1230, -1320, -1230 }
02988 , { -870, -1040, -1410, -870, -1410 }
02989 , { 210, 210, -160, -240, -160 }
02990 , { -870, -1040, -1410, -870, -1410 }
02991 }
02992 , { { -800, -800, -1410, -870, -1410 }
02993 , { -870, -1040, -1410, -870, -1410 }
02994 , { -870, -1040, -1410, -870, -1410 }
02995 , { -870, -1040, -1410, -870, -1410 }
02996 , { -800, -800, -1410, -1490, -1410 }
02997 }
02998 }
02999 , { { { 210, 210, -160, -1520, -160 }
03000 , { -870, -870, -1230, -1580, -1230 }
03001 , { -1040, -1040, -1410, -1520, -1410 }
03002 , { 210, 210, -160, -1760, -160 }
03003 , { -800, -800, -1410, -1520, -1410 }
03004 }
03005 , { { -1040, -1040, -1410, -1760, -1410 }
03006 , { -1220, -1220, -1590, -1940, -1590 }
03007 , { -1040, -1040, -1410, -1760, -1410 }
03008 , { -1060, -1060, -1420, -1770, -1420 }
03009 , { -1040, -1040, -1410, -1760, -1410 }
03010 }
03011 , { { -1040, -1040, -1410, -1520, -1410 }
03012 , { -1040, -1040, -1410, -1760, -1410 }
03013 , { -1040, -1040, -1410, -1520, -1410 }
03014 , { -1040, -1040, -1410, -1760, -1410 }
03015 , { -1040, -1040, -1410, -1520, -1410 }
03016 }
03017 , { { 210, 210, -160, -1580, -160 }
03018 , { -870, -870, -1230, -1580, -1230 }
03019 , { -1040, -1040, -1410, -1760, -1410 }
03020 , { 210, 210, -160, -1760, -160 }
03021 , { -1040, -1040, -1410, -1760, -1410 }
03022 }
03023 , { { -800, -800, -1410, -1520, -1410 }
```

```
03024     , { -1040, -1040, -1410, -1760, -1410}
03025     , { -1040, -1040, -1410, -1520, -1410}
03026     , { -1040, -1040, -1410, -1760, -1410}
03027     , { -800, -800, -1410, -1760, -1410}
03028     }
03029     }
03030     , {{{ -240, -240, -460, -240, -770}
03031     , { -1300, -1320, -1300, -1320, -1610}
03032     , { -1490, -1490, -1710, -1490, -2020}
03033     , { -240, -240, -460, -240, -770}
03034     , { -1490, -1490, -1710, -1490, -2020}
03035     }
03036     , {{{ -1490, -1490, -1490, -1490, -1800}
03037     , { -1670, -1670, -1890, -1670, -2200}
03038     , { -1490, -1490, -1710, -1490, -2020}
03039     , { -1490, -1510, -1490, -1510, -1800}
03040     , { -1490, -1490, -1710, -1490, -2020}
03041     }
03042     , {{{ -1490, -1490, -1710, -1490, -2020}
03043     , { -1490, -1490, -1710, -1490, -2020}
03044     , { -1490, -1490, -1710, -1490, -2020}
03045     , { -1490, -1490, -1710, -1490, -2020}
03046     , { -1490, -1490, -1710, -1490, -2020}
03047     }
03048     , {{{ -240, -240, -460, -240, -770}
03049     , { -1300, -1320, -1300, -1320, -1610}
03050     , { -1490, -1490, -1710, -1490, -2020}
03051     , { -240, -240, -460, -240, -770}
03052     , { -1490, -1490, -1710, -1490, -2020}
03053     }
03054     , {{{ -1490, -1490, -1710, -1490, -2020}
03055     , { -1490, -1490, -1710, -1490, -2020}
03056     , { -1490, -1490, -1710, -1490, -2020}
03057     , { -1490, -1490, -1710, -1490, -2020}
03058     , { -1490, -1490, -1710, -1490, -2020}
03059     }
03060     }
03061     , {{{ -160, -1990, -160, -870, -160}
03062     , { -870, -2060, -1230, -870, -1230}
03063     , { -870, -1990, -1410, -870, -1410}
03064     , { -160, -2230, -160, -870, -160}
03065     , { -870, -1990, -1410, -870, -1410}
03066     }
03067     , {{{ -870, -2230, -1410, -870, -1410}
03068     , { -1050, -2410, -1590, -1050, -1590}
03069     , { -870, -2230, -1410, -870, -1410}
03070     , { -1420, -2250, -1420, -2130, -1420}
03071     , { -870, -2230, -1410, -870, -1410}
03072     }
03073     , {{{ -870, -1990, -1410, -870, -1410}
03074     , { -870, -2230, -1410, -870, -1410}
03075     , { -870, -1990, -1410, -870, -1410}
03076     , { -870, -2230, -1410, -870, -1410}
03077     , { -870, -1990, -1410, -870, -1410}
03078     }
03079     , {{{ -160, -2060, -160, -870, -160}
03080     , { -1230, -2060, -1230, -1940, -1230}
03081     , { -870, -2230, -1410, -870, -1410}
03082     , { -160, -2230, -160, -2120, -160}
03083     , { -870, -2230, -1410, -870, -1410}
03084     }
03085     , {{{ -870, -1990, -1410, -870, -1410}
03086     , { -870, -2230, -1410, -870, -1410}
03087     , { -870, -1990, -1410, -870, -1410}
03088     , { -870, -2230, -1410, -870, -1410}
03089     , { -1410, -2230, -1410, -2120, -1410}
03090     }
03091     }
03092     , {{{ -240, -240, -460, -240, -1520}
03093     , { -1300, -1320, -1300, -1320, -1520}
03094     , { -1490, -1490, -1710, -1490, -1700}
03095     , { -240, -240, -460, -240, -1700}
03096     , { -1490, -1490, -1710, -1490, -1700}
03097     }
03098     , {{{ -1490, -1490, -1490, -1490, -1640}
03099     , { -1640, -1670, -1890, -1670, -1640}
03100     , { -1490, -1490, -1710, -1490, -1700}
03101     , { -1490, -1510, -1490, -1510, -1710}
03102     , { -1490, -1490, -1710, -1490, -1700}
03103     }
03104     , {{{ -1490, -1490, -1710, -1490, -1700}
03105     , { -1490, -1490, -1710, -1490, -1700}
03106     , { -1490, -1490, -1710, -1490, -1700}
03107     , { -1490, -1490, -1710, -1490, -1700}
03108     , { -1490, -1490, -1710, -1490, -1700}
03109     }
03110     , {{{ -240, -240, -460, -240, -1520}
```

```

03111      , { -1300, -1320, -1300, -1320, -1520}
03112      , { -1490, -1490, -1710, -1490, -1700}
03113      , { -240, -240, -460, -240, -1700}
03114      , { -1490, -1490, -1710, -1490, -1700}
03115      }
03116      , { { -1490, -1490, -1710, -1490, -1700}
03117      , { -1490, -1490, -1710, -1490, -1700}
03118      , { -1490, -1490, -1710, -1490, -1700}
03119      , { -1490, -1490, -1710, -1490, -1700}
03120      , { -1490, -1490, -1710, -1490, -1700}
03121      }
03122      }
03123      }
03124      , { { { 760, 760, 400, 310, 400}
03125      , { 200, -430, -340, 200, -340}
03126      , { -310, -490, -850, -310, -850}
03127      , { 760, 760, 400, 310, 400}
03128      , { -250, -250, -850, -310, -850}
03129      }
03130      , { { 200, -430, -340, 200, -340}
03131      , { 200, -430, -340, 200, -340}
03132      , { -310, -490, -850, -310, -850}
03133      , { -830, -830, -1190, -1280, -1190}
03134      , { -310, -490, -850, -310, -850}
03135      }
03136      , { { -310, -490, -850, -310, -850}
03137      , { -310, -490, -850, -310, -850}
03138      , { -310, -490, -850, -310, -850}
03139      , { -310, -490, -850, -310, -850}
03140      , { -310, -490, -850, -310, -850}
03141      }
03142      , { { 760, 760, 400, 310, 400}
03143      , { -1000, -1000, -1360, -1450, -1360}
03144      , { -310, -490, -850, -310, -850}
03145      , { 760, 760, 400, 310, 400}
03146      , { -310, -490, -850, -310, -850}
03147      }
03148      , { { -250, -250, -850, -310, -850}
03149      , { -310, -490, -850, -310, -850}
03150      , { -310, -490, -850, -310, -850}
03151      , { -310, -490, -850, -310, -850}
03152      , { -250, -250, -850, -940, -850}
03153      }
03154      }
03155      , { { { 760, 760, 400, -690, 400}
03156      , { -340, -490, -340, -690, -340}
03157      , { -490, -490, -850, -960, -850}
03158      , { 760, 760, 400, -1200, 400}
03159      , { -250, -250, -850, -960, -850}
03160      }
03161      , { { -340, -490, -340, -690, -340}
03162      , { -340, -2040, -340, -690, -340}
03163      , { -490, -490, -850, -1200, -850}
03164      , { -830, -830, -1190, -1540, -1190}
03165      , { -490, -490, -850, -1200, -850}
03166      }
03167      , { { -490, -490, -850, -960, -850}
03168      , { -490, -490, -850, -1200, -850}
03169      , { -490, -490, -850, -960, -850}
03170      , { -490, -490, -850, -1200, -850}
03171      , { -490, -490, -850, -960, -850}
03172      }
03173      , { { 760, 760, 400, -1200, 400}
03174      , { -1000, -1000, -1360, -1710, -1360}
03175      , { -490, -490, -850, -1200, -850}
03176      , { 760, 760, 400, -1200, 400}
03177      , { -490, -490, -850, -1200, -850}
03178      }
03179      , { { -250, -250, -850, -960, -850}
03180      , { -490, -490, -850, -1200, -850}
03181      , { -490, -490, -850, -960, -850}
03182      , { -490, -490, -850, -1200, -850}
03183      , { -250, -250, -850, -1200, -850}
03184      }
03185      }
03186      , { { { 310, 310, 90, 310, -220}
03187      , { -430, -430, -650, -430, -960}
03188      , { -940, -940, -1160, -940, -1470}
03189      , { 310, 310, 90, 310, -220}
03190      , { -940, -940, -1160, -940, -1470}
03191      }
03192      , { { -430, -430, -650, -430, -960}
03193      , { -430, -430, -650, -430, -960}
03194      , { -940, -940, -1160, -940, -1470}
03195      , { -1260, -1280, -1260, -1280, -1570}
03196      , { -940, -940, -1160, -940, -1470}
03197      }

```



```
03198 ,{{ -940, -940, -1160, -940, -1470}
03199 ,{ -940, -940, -1160, -940, -1470}
03200 ,{ -940, -940, -1160, -940, -1470}
03201 ,{ -940, -940, -1160, -940, -1470}
03202 ,{ -940, -940, -1160, -940, -1470}
03203 }
03204 ,{{ 310, 310, 90, 310, -220}
03205 ,{ -1430, -1450, -1430, -1450, -1740}
03206 ,{ -940, -940, -1160, -940, -1470}
03207 ,{ 310, 310, 90, 310, -220}
03208 ,{ -940, -940, -1160, -940, -1470}
03209 }
03210 ,{{ -940, -940, -1160, -940, -1470}
03211 ,{ -940, -940, -1160, -940, -1470}
03212 ,{ -940, -940, -1160, -940, -1470}
03213 ,{ -940, -940, -1160, -940, -1470}
03214 ,{ -940, -940, -1160, -940, -1470}
03215 }
03216 }
03217 ,{{{ 400, -1170, 400, 200, 400}
03218 ,{ 200, -1170, -340, 200, -340}
03219 ,{ -310, -1440, -850, -310, -850}
03220 ,{ 400, -1680, 400, -310, 400}
03221 ,{ -310, -1440, -850, -310, -850}
03222 }
03223 ,{{ 200, -1170, -340, 200, -340}
03224 ,{ 200, -1170, -340, 200, -340}
03225 ,{ -310, -1680, -850, -310, -850}
03226 ,{ -1190, -2020, -1190, -1900, -1190}
03227 ,{ -310, -1680, -850, -310, -850}
03228 }
03229 ,{{ -310, -1440, -850, -310, -850}
03230 ,{ -310, -1680, -850, -310, -850}
03231 ,{ -310, -1440, -850, -310, -850}
03232 ,{ -310, -1680, -850, -310, -850}
03233 ,{ -310, -1440, -850, -310, -850}
03234 }
03235 ,{{{ 400, -1680, 400, -310, 400}
03236 ,{ -1360, -2190, -1360, -2070, -1360}
03237 ,{ -310, -1680, -850, -310, -850}
03238 ,{ 400, -1680, 400, -1560, 400}
03239 ,{ -310, -1680, -850, -310, -850}
03240 }
03241 ,{{ -310, -1440, -850, -310, -850}
03242 ,{ -310, -1680, -850, -310, -850}
03243 ,{ -310, -1440, -850, -310, -850}
03244 ,{ -310, -1680, -850, -310, -850}
03245 ,{ -850, -1680, -850, -1560, -850}
03246 }
03247 }
03248 ,{{{ 310, 310, 90, 310, -390}
03249 ,{ -390, -430, -650, -430, -390}
03250 ,{ -940, -940, -1160, -940, -1140}
03251 ,{ 310, 310, 90, 310, -1140}
03252 ,{ -940, -940, -1160, -940, -1140}
03253 }
03254 ,{{ -390, -430, -650, -430, -390}
03255 ,{ -390, -430, -650, -430, -390}
03256 ,{ -940, -940, -1160, -940, -1140}
03257 ,{ -1260, -1280, -1260, -1280, -1480}
03258 ,{ -940, -940, -1160, -940, -1140}
03259 }
03260 ,{{ -940, -940, -1160, -940, -1140}
03261 ,{ -940, -940, -1160, -940, -1140}
03262 ,{ -940, -940, -1160, -940, -1140}
03263 ,{ -940, -940, -1160, -940, -1140}
03264 ,{ -940, -940, -1160, -940, -1140}
03265 }
03266 ,{{{ 310, 310, 90, 310, -1140}
03267 ,{ -1430, -1450, -1430, -1450, -1650}
03268 ,{ -940, -940, -1160, -940, -1140}
03269 ,{ 310, 310, 90, 310, -1140}
03270 ,{ -940, -940, -1160, -940, -1140}
03271 }
03272 ,{{ -940, -940, -1160, -940, -1140}
03273 ,{ -940, -940, -1160, -940, -1140}
03274 ,{ -940, -940, -1160, -940, -1140}
03275 ,{ -940, -940, -1160, -940, -1140}
03276 ,{ -940, -940, -1160, -940, -1140}
03277 }
03278 }
03279 }
03280 ,{{{ 1140, 1140, 770, 780, 770}
03281 ,{ 780, 600, 240, 780, 240}
03282 ,{ 480, 300, -60, 480, -60}
03283 ,{ 1140, 1140, 770, 690, 770}
03284 ,{ 480, 300, -60, 480, -60}
```

```

03285     }
03286     ,{{ 780, 600, 240, 780, 240}
03287     ,{ 780, 600, 240, 780, 240}
03288     ,{ 470, 290, -70, 470, -70}
03289     ,{ -780, -780, -1150, -1230, -1150}
03290     ,{ 470, 290, -70, 470, -70}
03291     }
03292     ,{{ 490, 310, -50, 490, -50}
03293     ,{ 490, 310, -50, 490, -50}
03294     ,{ 480, 300, -60, 480, -60}
03295     ,{ 490, 310, -50, 490, -50}
03296     ,{ 480, 300, -60, 480, -60}
03297     }
03298     ,{{ 1140, 1140, 770, 690, 770}
03299     ,{ -600, -600, -970, -1050, -970}
03300     ,{ 470, 290, -70, 470, -70}
03301     ,{ 1140, 1140, 770, 690, 770}
03302     ,{ 470, 290, -70, 470, -70}
03303     }
03304     ,{{ 490, 310, -50, 490, -50}
03305     ,{ 490, 310, -50, 490, -50}
03306     ,{ 480, 300, -60, 480, -60}
03307     ,{ 490, 310, -50, 490, -50}
03308     ,{ -430, -430, -1040, -1120, -1040}
03309     }
03310     }
03311     ,{{{ 1140, 1140, 770, -110, 770}
03312     ,{ 600, 600, 240, -110, 240}
03313     ,{ 300, 300, -60, -170, -60}
03314     ,{ 1140, 1140, 770, -400, 770}
03315     ,{ 300, 300, -60, -170, -60}
03316     }
03317     ,{{ 600, 600, 240, -110, 240}
03318     ,{ 600, 600, 240, -110, 240}
03319     ,{ 290, 290, -70, -420, -70}
03320     ,{ -780, -780, -1150, -1500, -1150}
03321     ,{ 290, 290, -70, -420, -70}
03322     }
03323     ,{{ 310, 310, -50, -170, -50}
03324     ,{ 310, 310, -50, -400, -50}
03325     ,{ 300, 300, -60, -170, -60}
03326     ,{ 310, 310, -50, -400, -50}
03327     ,{ 300, 300, -60, -170, -60}
03328     }
03329     ,{{ 1140, 1140, 770, -420, 770}
03330     ,{ -600, -600, -970, -1320, -970}
03331     ,{ 290, 290, -70, -420, -70}
03332     ,{ 1140, 1140, 770, -830, 770}
03333     ,{ 290, 290, -70, -420, -70}
03334     }
03335     ,{{{ 310, 310, -50, -170, -50}
03336     ,{ 310, 310, -50, -400, -50}
03337     ,{ 300, 300, -60, -170, -60}
03338     ,{ 310, 310, -50, -400, -50}
03339     ,{ -430, -430, -1040, -1390, -1040}
03340     }
03341     }
03342     ,{{{ 690, 690, 470, 690, 160}
03343     ,{ 150, 150, -60, 150, -370}
03344     ,{ -140, -140, -360, -140, -670}
03345     ,{ 690, 690, 470, 690, 160}
03346     ,{ -140, -140, -360, -140, -670}
03347     }
03348     ,{{ 150, 150, -60, 150, -370}
03349     ,{ 150, 150, -60, 150, -370}
03350     ,{ -150, -150, -370, -150, -680}
03351     ,{ -1210, -1230, -1210, -1230, -1520}
03352     ,{ -150, -150, -370, -150, -680}
03353     }
03354     ,{{ -140, -140, -360, -140, -670}
03355     ,{ -140, -140, -360, -140, -670}
03356     ,{ -140, -140, -360, -140, -670}
03357     ,{ -140, -140, -360, -140, -670}
03358     ,{ -140, -140, -360, -140, -670}
03359     }
03360     ,{{ 690, 690, 470, 690, 160}
03361     ,{ -1030, -1050, -1030, -1050, -1340}
03362     ,{ -150, -150, -370, -150, -680}
03363     ,{ 690, 690, 470, 690, 160}
03364     ,{ -150, -150, -370, -150, -680}
03365     }
03366     ,{{ -140, -140, -360, -140, -670}
03367     ,{ -140, -140, -360, -140, -670}
03368     ,{ -140, -140, -360, -140, -670}
03369     ,{ -140, -140, -360, -140, -670}
03370     ,{ -1120, -1120, -1340, -1120, -1650}
03371     }

```

```
03372     }
03373     ,{{{ 780, -580, 770, 780, 770}
03374     ,{ 780, -580, 240, 780, 240}
03375     ,{ 480, -640, -60, 480, -60}
03376     ,{ 770, -880, 770, 490, 770}
03377     ,{ 480, -640, -60, 480, -60}
03378     }
03379     ,{{{ 780, -580, 240, 780, 240}
03380     ,{ 780, -580, 240, 780, 240}
03381     ,{ 470, -890, -70, 470, -70}
03382     ,{ -1150, -1970, -1150, -1860, -1150}
03383     ,{ 470, -890, -70, 470, -70}
03384     }
03385     ,{{{ 490, -640, -50, 490, -50}
03386     ,{ 490, -880, -50, 490, -50}
03387     ,{ 480, -640, -60, 480, -60}
03388     ,{ 490, -880, -50, 490, -50}
03389     ,{ 480, -640, -60, 480, -60}
03390     }
03391     ,{{{ 770, -890, 770, 470, 770}
03392     ,{ -970, -1790, -970, -1680, -970}
03393     ,{ 470, -890, -70, 470, -70}
03394     ,{ 770, -1300, 770, -1190, 770}
03395     ,{ 470, -890, -70, 470, -70}
03396     }
03397     ,{{{ 490, -640, -50, 490, -50}
03398     ,{ 490, -880, -50, 490, -50}
03399     ,{ 480, -640, -60, 480, -60}
03400     ,{ 490, -880, -50, 490, -50}
03401     ,{ -1040, -1860, -1040, -1750, -1040}
03402     }
03403     }
03404     ,{{{ 690, 690, 470, 690, 190}
03405     ,{ 190, 150, -60, 150, 190}
03406     ,{ -140, -140, -360, -140, -350}
03407     ,{ 690, 690, 470, 690, -340}
03408     ,{ -140, -140, -360, -140, -350}
03409     }
03410     ,{{{ 190, 150, -60, 150, 190}
03411     ,{ 190, 150, -60, 150, 190}
03412     ,{ -150, -150, -370, -150, -360}
03413     ,{ -1210, -1230, -1210, -1230, -1440}
03414     ,{ -150, -150, -370, -150, -360}
03415     }
03416     ,{{{ -140, -140, -360, -140, -340}
03417     ,{ -140, -140, -360, -140, -340}
03418     ,{ -140, -140, -360, -140, -350}
03419     ,{ -140, -140, -360, -140, -340}
03420     ,{ -140, -140, -360, -140, -350}
03421     }
03422     ,{{{ 690, 690, 470, 690, -360}
03423     ,{ -1030, -1050, -1030, -1050, -1260}
03424     ,{ -150, -150, -370, -150, -360}
03425     ,{ 690, 690, 470, 690, -770}
03426     ,{ -150, -150, -370, -150, -360}
03427     }
03428     ,{{{ -140, -140, -360, -140, -340}
03429     ,{ -140, -140, -360, -140, -340}
03430     ,{ -140, -140, -360, -140, -350}
03431     ,{ -140, -140, -360, -140, -340}
03432     ,{ -1120, -1120, -1340, -1120, -1330}
03433     }
03434     }
03435     }
03436     ,{{{ 1320, 1320, 960, 870, 960}
03437     ,{ 850, 670, 300, 850, 300}
03438     ,{ 720, 540, 170, 720, 170}
03439     ,{ 1320, 1320, 960, 870, 960}
03440     ,{ 590, 410, 40, 590, 40}
03441     }
03442     ,{{{ 850, 670, 300, 850, 300}
03443     ,{ 850, 670, 300, 850, 300}
03444     ,{ 570, 390, 20, 570, 20}
03445     ,{ -730, -730, -1100, -1180, -1100}
03446     ,{ 570, 390, 20, 570, 20}
03447     }
03448     ,{{{ 720, 540, 170, 720, 170}
03449     ,{ 720, 540, 170, 720, 170}
03450     ,{ 720, 540, 170, 720, 170}
03451     ,{ 720, 540, 170, 720, 170}
03452     ,{ 590, 410, 40, 590, 40}
03453     }
03454     ,{{{ 1320, 1320, 960, 870, 960}
03455     ,{ -1030, -1030, -1400, -1480, -1400}
03456     ,{ 570, 390, 20, 570, 20}
03457     ,{ 1320, 1320, 960, 870, 960}
03458     ,{ 570, 390, 20, 570, 20}
```

```
03459     }
03460     ,{{ 720, 540, 170, 720, 170}
03461     ,{ 720, 540, 170, 720, 170}
03462     ,{ 280, 100, -260, 280, -260}
03463     ,{ 720, 540, 170, 720, 170}
03464     ,{ -160, -160, -760, -850, -760}
03465     }
03466     }
03467     ,{{{ 1320, 1320, 960, 70, 960}
03468     ,{ 670, 670, 300, -40, 300}
03469     ,{ 540, 540, 170, 70, 170}
03470     ,{ 1320, 1320, 960, -170, 960}
03471     ,{ 410, 410, 40, -60, 40}
03472     }
03473     ,{{{ 670, 670, 300, -40, 300}
03474     ,{ 670, 670, 300, -40, 300}
03475     ,{ 390, 390, 20, -320, 20}
03476     ,{ -730, -730, -1100, -1450, -1100}
03477     ,{ 390, 390, 20, -320, 20}
03478     }
03479     ,{{{ 540, 540, 170, 70, 170}
03480     ,{ 540, 540, 170, -170, 170}
03481     ,{ 540, 540, 170, 70, 170}
03482     ,{ 540, 540, 170, -170, 170}
03483     ,{ 410, 410, 40, -60, 40}
03484     }
03485     ,{{{ 1320, 1320, 960, -320, 960}
03486     ,{ -1030, -1030, -1400, -1750, -1400}
03487     ,{ 390, 390, 20, -320, 20}
03488     ,{ 1320, 1320, 960, -640, 960}
03489     ,{ 390, 390, 20, -320, 20}
03490     }
03491     ,{{{ 540, 540, 170, -170, 170}
03492     ,{ 540, 540, 170, -170, 170}
03493     ,{ 100, 100, -260, -370, -260}
03494     ,{ 540, 540, 170, -170, 170}
03495     ,{ -160, -160, -760, -1110, -760}
03496     }
03497     }
03498     ,{{{ 870, 870, 650, 870, 340}
03499     ,{ 220, 220, 0, 220, -310}
03500     ,{ 90, 90, -130, 90, -440}
03501     ,{ 870, 870, 650, 870, 340}
03502     ,{ -40, -40, -260, -40, -570}
03503     }
03504     ,{{{ 220, 220, 0, 220, -310}
03505     ,{ 220, 220, 0, 220, -310}
03506     ,{ -60, -60, -280, -60, -590}
03507     ,{ -1160, -1180, -1160, -1180, -1470}
03508     ,{ -60, -60, -280, -60, -590}
03509     }
03510     ,{{{ 90, 90, -130, 90, -440}
03511     ,{ 90, 90, -130, 90, -440}
03512     ,{ 90, 90, -130, 90, -440}
03513     ,{ 90, 90, -130, 90, -440}
03514     ,{ -40, -40, -260, -40, -570}
03515     }
03516     ,{{{ 870, 870, 650, 870, 340}
03517     ,{ -1460, -1480, -1460, -1480, -1770}
03518     ,{ -60, -60, -280, -60, -590}
03519     ,{ 870, 870, 650, 870, 340}
03520     ,{ -60, -60, -280, -60, -590}
03521     }
03522     ,{{{ 90, 90, -130, 90, -440}
03523     ,{ 90, 90, -130, 90, -440}
03524     ,{ -350, -350, -570, -350, -880}
03525     ,{ 90, 90, -130, 90, -440}
03526     ,{ -850, -850, -1070, -850, -1380}
03527     }
03528     }
03529     ,{{{ 960, -410, 960, 850, 960}
03530     ,{ 850, -520, 300, 850, 300}
03531     ,{ 720, -410, 170, 720, 170}
03532     ,{ 960, -650, 960, 720, 960}
03533     ,{ 590, -540, 40, 590, 40}
03534     }
03535     ,{{{ 850, -520, 300, 850, 300}
03536     ,{ 850, -520, 300, 850, 300}
03537     ,{ 570, -800, 20, 570, 20}
03538     ,{ -1100, -1920, -1100, -1810, -1100}
03539     ,{ 570, -800, 20, 570, 20}
03540     }
03541     ,{{{ 720, -410, 170, 720, 170}
03542     ,{ 720, -650, 170, 720, 170}
03543     ,{ 720, -410, 170, 720, 170}
03544     ,{ 720, -650, 170, 720, 170}
03545     ,{ 590, -540, 40, 590, 40}
```

```
03546     }
03547     ,{{ 960, -800, 960, 570, 960}
03548     ,{ -1400, -2220, -1400, -2110, -1400}
03549     ,{ 570, -800, 20, 570, 20}
03550     ,{ 960, -1120, 960, -1000, 960}
03551     ,{ 570, -800, 20, 570, 20}
03552     }
03553     ,{{ 720, -650, 170, 720, 170}
03554     ,{ 720, -650, 170, 720, 170}
03555     ,{ 280, -850, -260, 280, -260}
03556     ,{ 720, -650, 170, 720, 170}
03557     ,{ -760, -1590, -760, -1470, -760}
03558     }
03559     }
03560     ,{{{ 870, 870, 650, 870, 250}
03561     ,{ 250, 220, 0, 220, 250}
03562     ,{ 90, 90, -130, 90, -110}
03563     ,{ 870, 870, 650, 870, -110}
03564     ,{ -40, -40, -260, -40, -240}
03565     }
03566     ,{{{ 250, 220, 0, 220, 250}
03567     ,{ 250, 220, 0, 220, 250}
03568     ,{ -60, -60, -280, -60, -260}
03569     ,{ -1160, -1180, -1160, -1180, -1390}
03570     ,{ -60, -60, -280, -60, -260}
03571     }
03572     ,{{{ 90, 90, -130, 90, -110}
03573     ,{ 90, 90, -130, 90, -110}
03574     ,{ 90, 90, -130, 90, -110}
03575     ,{ 90, 90, -130, 90, -110}
03576     ,{ -40, -40, -260, -40, -240}
03577     }
03578     ,{{{ 870, 870, 650, 870, -260}
03579     ,{ -1460, -1480, -1460, -1480, -1690}
03580     ,{ -60, -60, -280, -60, -260}
03581     ,{ 870, 870, 650, 870, -580}
03582     ,{ -60, -60, -280, -60, -260}
03583     }
03584     ,{{{ 90, 90, -130, 90, -110}
03585     ,{ 90, 90, -130, 90, -110}
03586     ,{ -350, -350, -570, -350, -550}
03587     ,{ 90, 90, -130, 90, -110}
03588     ,{ -850, -850, -1070, -850, -1050}
03589     }
03590     }
03591     }
03592     ,{{{ 1320, 1320, 960, 870, 960}
03593     ,{ 850, 670, 540, 850, 300}
03594     ,{ 720, 540, 170, 720, 170}
03595     ,{ 1320, 1320, 960, 870, 960}
03596     ,{ 590, 410, 40, 590, 40}
03597     }
03598     ,{{{ 850, 670, 300, 850, 300}
03599     ,{ 850, 670, 300, 850, 300}
03600     ,{ 570, 390, 20, 570, 20}
03601     ,{ -350, -350, -870, -960, -870}
03602     ,{ 570, 390, 20, 570, 20}
03603     }
03604     ,{{{ 720, 540, 170, 720, 170}
03605     ,{ 720, 540, 170, 720, 170}
03606     ,{ 720, 540, 170, 720, 170}
03607     ,{ 720, 540, 170, 720, 170}
03608     ,{ 590, 410, 40, 590, 40}
03609     }
03610     ,{{{ 1320, 1320, 960, 870, 960}
03611     ,{ 540, -100, 540, -1050, -810}
03612     ,{ 570, 390, 20, 570, 20}
03613     ,{ 1320, 1320, 960, 870, 960}
03614     ,{ 570, 390, 20, 570, 20}
03615     }
03616     ,{{{ 720, 540, 170, 720, 170}
03617     ,{ 720, 540, 170, 720, 170}
03618     ,{ 480, 300, -60, 480, -60}
03619     ,{ 720, 540, 170, 720, 170}
03620     ,{ -160, -160, -400, -230, -760}
03621     }
03622     }
03623     ,{{{ 1320, 1320, 960, 70, 960}
03624     ,{ 670, 670, 300, -40, 300}
03625     ,{ 540, 540, 170, 70, 170}
03626     ,{ 1320, 1320, 960, -170, 960}
03627     ,{ 410, 410, 40, -60, 40}
03628     }
03629     ,{{{ 670, 670, 300, -40, 300}
03630     ,{ 670, 670, 300, -40, 300}
03631     ,{ 390, 390, 20, -320, 20}
03632     ,{ -730, -730, -1100, -1450, -870}
```

```

03633      , { 390, 390, 20, -320, 20}
03634      }
03635      , { { 540, 540, 170, 70, 170}
03636      , { 540, 540, 170, -170, 170}
03637      , { 540, 540, 170, 70, 170}
03638      , { 540, 540, 170, -170, 170}
03639      , { 410, 410, 40, -60, 40}
03640      }
03641      , { { 1320, 1320, 960, -320, 960}
03642      , { 10, -600, 10, -1320, -970}
03643      , { 390, 390, 20, -320, 20}
03644      , { 1320, 1320, 960, -640, 960}
03645      , { 390, 390, 20, -320, 20}
03646      }
03647      , { { 540, 540, 170, -170, 170}
03648      , { 540, 540, 170, -170, 170}
03649      , { 300, 300, -60, -170, -60}
03650      , { 540, 540, 170, -170, 170}
03651      , { -160, -160, -400, -1110, -760}
03652      }
03653      }
03654      , { { { 870, 870, 650, 870, 340}
03655      , { 540, 220, 540, 220, -310}
03656      , { 90, 90, -130, 90, -440}
03657      , { 870, 870, 650, 870, 340}
03658      , { -40, -40, -260, -40, -570}
03659      }
03660      , { { 220, 220, 0, 220, -310}
03661      , { 220, 220, 0, 220, -310}
03662      , { -60, -60, -280, -60, -590}
03663      , { -350, -350, -940, -960, -1250}
03664      , { -60, -60, -280, -60, -590}
03665      }
03666      , { { 90, 90, -130, 90, -440}
03667      , { 90, 90, -130, 90, -440}
03668      , { 90, 90, -130, 90, -440}
03669      , { 90, 90, -130, 90, -440}
03670      , { -40, -40, -260, -40, -570}
03671      }
03672      , { { 870, 870, 650, 870, 340}
03673      , { 540, -100, 540, -1050, -1340}
03674      , { -60, -60, -280, -60, -590}
03675      , { 870, 870, 650, 870, 340}
03676      , { -60, -60, -280, -60, -590}
03677      }
03678      , { { 90, 90, -130, 90, -440}
03679      , { 90, 90, -130, 90, -440}
03680      , { -140, -140, -360, -140, -670}
03681      , { 90, 90, -130, 90, -440}
03682      , { -850, -850, -1070, -850, -1380}
03683      }
03684      }
03685      , { { { 960, -410, 960, 850, 960}
03686      , { 850, -520, 300, 850, 300}
03687      , { 720, -410, 170, 720, 170}
03688      , { 960, -650, 960, 720, 960}
03689      , { 590, -540, 40, 590, 40}
03690      }
03691      , { { 850, -520, 300, 850, 300}
03692      , { 850, -520, 300, 850, 300}
03693      , { 570, -800, 20, 570, 20}
03694      , { -870, -1920, -870, -1370, -870}
03695      , { 570, -800, 20, 570, 20}
03696      }
03697      , { { 720, -410, 170, 720, 170}
03698      , { 720, -650, 170, 720, 170}
03699      , { 720, -410, 170, 720, 170}
03700      , { 720, -650, 170, 720, 170}
03701      , { 590, -540, 40, 590, 40}
03702      }
03703      , { { 960, -800, 960, 570, 960}
03704      , { -970, -1790, -970, -1680, -970}
03705      , { 570, -800, 20, 570, 20}
03706      , { 960, -1120, 960, -1000, 960}
03707      , { 570, -800, 20, 570, 20}
03708      }
03709      , { { 720, -640, 170, 720, 170}
03710      , { 720, -650, 170, 720, 170}
03711      , { 480, -640, -60, 480, -60}
03712      , { 720, -650, 170, 720, 170}
03713      , { -230, -1520, -760, -230, -760}
03714      }
03715      }
03716      , { { { 870, 870, 650, 870, 250}
03717      , { 250, 220, 0, 220, 250}
03718      , { 90, 90, -130, 90, -110}
03719      , { 870, 870, 650, 870, -110}

```

```
03720     , {   -40,   -40,  -260,   -40,  -240 }
03721     }
03722     , { {   250,   220,    0,   220,   250 }
03723     , {   250,   220,    0,   220,   250 }
03724     , {   -60,   -60,  -280,   -60,  -260 }
03725     , {  -940,  -960,  -940,  -960, -1360 }
03726     , {   -60,   -60,  -280,   -60,  -260 }
03727     }
03728     , { {    90,    90,  -130,    90,   -90 }
03729     , {    90,    90,  -130,    90,   -90 }
03730     , {    90,    90,  -130,    90,  -110 }
03731     , {    90,    90,  -130,    90,  -110 }
03732     , {   -40,   -40,  -260,   -40,  -240 }
03733     }
03734     , { {   870,   870,   650,   870,  -260 }
03735     , {  -810, -1050, -1030, -1050,  -810 }
03736     , {   -60,   -60,  -280,   -60,  -260 }
03737     , {   870,   870,   650,   870,  -580 }
03738     , {   -60,   -60,  -280,   -60,  -260 }
03739     }
03740     , { {    90,    90,  -130,    90,  -110 }
03741     , {    90,    90,  -130,    90,  -110 }
03742     , {  -140,  -140,  -360,  -140,  -350 }
03743     , {    90,    90,  -130,    90,  -110 }
03744     , {  -850,  -850, -1070,  -850, -1050 }
03745     }
03746     }
03747     }
03748     }
03749     , { { { INF,   INF,   INF,   INF,   INF }
03750     , {   INF,   INF,   INF,   INF,   INF }
03751     , {   INF,   INF,   INF,   INF,   INF }
03752     , {   INF,   INF,   INF,   INF,   INF }
03753     , {   INF,   INF,   INF,   INF,   INF }
03754     }
03755     , { {   INF,   INF,   INF,   INF,   INF }
03756     , {   INF,   INF,   INF,   INF,   INF }
03757     , {   INF,   INF,   INF,   INF,   INF }
03758     , {   INF,   INF,   INF,   INF,   INF }
03759     , {   INF,   INF,   INF,   INF,   INF }
03760     }
03761     , { {   INF,   INF,   INF,   INF,   INF }
03762     , {   INF,   INF,   INF,   INF,   INF }
03763     , {   INF,   INF,   INF,   INF,   INF }
03764     , {   INF,   INF,   INF,   INF,   INF }
03765     , {   INF,   INF,   INF,   INF,   INF }
03766     }
03767     , { {   INF,   INF,   INF,   INF,   INF }
03768     , {   INF,   INF,   INF,   INF,   INF }
03769     , {   INF,   INF,   INF,   INF,   INF }
03770     , {   INF,   INF,   INF,   INF,   INF }
03771     , {   INF,   INF,   INF,   INF,   INF }
03772     }
03773     , { {   INF,   INF,   INF,   INF,   INF }
03774     , {   INF,   INF,   INF,   INF,   INF }
03775     , {   INF,   INF,   INF,   INF,   INF }
03776     , {   INF,   INF,   INF,   INF,   INF }
03777     , {   INF,   INF,   INF,   INF,   INF }
03778     }
03779     }
03780     , { { { INF,   INF,   INF,   INF,   INF }
03781     , {   INF,   INF,   INF,   INF,   INF }
03782     , {   INF,   INF,   INF,   INF,   INF }
03783     , {   INF,   INF,   INF,   INF,   INF }
03784     , {   INF,   INF,   INF,   INF,   INF }
03785     }
03786     , { {   INF,   INF,   INF,   INF,   INF }
03787     , {   INF,   INF,   INF,   INF,   INF }
03788     , {   INF,   INF,   INF,   INF,   INF }
03789     , {   INF,   INF,   INF,   INF,   INF }
03790     , {   INF,   INF,   INF,   INF,   INF }
03791     }
03792     , { {   INF,   INF,   INF,   INF,   INF }
03793     , {   INF,   INF,   INF,   INF,   INF }
03794     , {   INF,   INF,   INF,   INF,   INF }
03795     , {   INF,   INF,   INF,   INF,   INF }
03796     , {   INF,   INF,   INF,   INF,   INF }
03797     }
03798     , { {   INF,   INF,   INF,   INF,   INF }
03799     , {   INF,   INF,   INF,   INF,   INF }
03800     , {   INF,   INF,   INF,   INF,   INF }
03801     , {   INF,   INF,   INF,   INF,   INF }
03802     , {   INF,   INF,   INF,   INF,   INF }
03803     }
03804     , { {   INF,   INF,   INF,   INF,   INF }
03805     , {   INF,   INF,   INF,   INF,   INF }
03806     , {   INF,   INF,   INF,   INF,   INF }
```

```
03807      , {   INF,   INF,   INF,   INF,   INF }
03808      , {   INF,   INF,   INF,   INF,   INF }
03809      }
03810    }
03811  , { {   INF,   INF,   INF,   INF,   INF }
03812    , {   INF,   INF,   INF,   INF,   INF }
03813    , {   INF,   INF,   INF,   INF,   INF }
03814    , {   INF,   INF,   INF,   INF,   INF }
03815    , {   INF,   INF,   INF,   INF,   INF }
03816    }
03817  , { {   INF,   INF,   INF,   INF,   INF }
03818    , {   INF,   INF,   INF,   INF,   INF }
03819    , {   INF,   INF,   INF,   INF,   INF }
03820    , {   INF,   INF,   INF,   INF,   INF }
03821    , {   INF,   INF,   INF,   INF,   INF }
03822    }
03823  , { {   INF,   INF,   INF,   INF,   INF }
03824    , {   INF,   INF,   INF,   INF,   INF }
03825    , {   INF,   INF,   INF,   INF,   INF }
03826    , {   INF,   INF,   INF,   INF,   INF }
03827    , {   INF,   INF,   INF,   INF,   INF }
03828    }
03829  , { {   INF,   INF,   INF,   INF,   INF }
03830    , {   INF,   INF,   INF,   INF,   INF }
03831    , {   INF,   INF,   INF,   INF,   INF }
03832    , {   INF,   INF,   INF,   INF,   INF }
03833    , {   INF,   INF,   INF,   INF,   INF }
03834    }
03835  , { {   INF,   INF,   INF,   INF,   INF }
03836    , {   INF,   INF,   INF,   INF,   INF }
03837    , {   INF,   INF,   INF,   INF,   INF }
03838    , {   INF,   INF,   INF,   INF,   INF }
03839    , {   INF,   INF,   INF,   INF,   INF }
03840    }
03841  }
03842  , { { {   INF,   INF,   INF,   INF,   INF }
03843    , {   INF,   INF,   INF,   INF,   INF }
03844    , {   INF,   INF,   INF,   INF,   INF }
03845    , {   INF,   INF,   INF,   INF,   INF }
03846    , {   INF,   INF,   INF,   INF,   INF }
03847    }
03848  , { { {   INF,   INF,   INF,   INF,   INF }
03849    , {   INF,   INF,   INF,   INF,   INF }
03850    , {   INF,   INF,   INF,   INF,   INF }
03851    , {   INF,   INF,   INF,   INF,   INF }
03852    , {   INF,   INF,   INF,   INF,   INF }
03853    }
03854  , { { {   INF,   INF,   INF,   INF,   INF }
03855    , {   INF,   INF,   INF,   INF,   INF }
03856    , {   INF,   INF,   INF,   INF,   INF }
03857    , {   INF,   INF,   INF,   INF,   INF }
03858    , {   INF,   INF,   INF,   INF,   INF }
03859    }
03860  , { { {   INF,   INF,   INF,   INF,   INF }
03861    , {   INF,   INF,   INF,   INF,   INF }
03862    , {   INF,   INF,   INF,   INF,   INF }
03863    , {   INF,   INF,   INF,   INF,   INF }
03864    , {   INF,   INF,   INF,   INF,   INF }
03865    }
03866  , { { {   INF,   INF,   INF,   INF,   INF }
03867    , {   INF,   INF,   INF,   INF,   INF }
03868    , {   INF,   INF,   INF,   INF,   INF }
03869    , {   INF,   INF,   INF,   INF,   INF }
03870    , {   INF,   INF,   INF,   INF,   INF }
03871    }
03872  }
03873  , { { {   INF,   INF,   INF,   INF,   INF }
03874    , {   INF,   INF,   INF,   INF,   INF }
03875    , {   INF,   INF,   INF,   INF,   INF }
03876    , {   INF,   INF,   INF,   INF,   INF }
03877    , {   INF,   INF,   INF,   INF,   INF }
03878    }
03879  , { {   INF,   INF,   INF,   INF,   INF }
03880    , {   INF,   INF,   INF,   INF,   INF }
03881    , {   INF,   INF,   INF,   INF,   INF }
03882    , {   INF,   INF,   INF,   INF,   INF }
03883    , {   INF,   INF,   INF,   INF,   INF }
03884    }
03885  , { {   INF,   INF,   INF,   INF,   INF }
03886    , {   INF,   INF,   INF,   INF,   INF }
03887    , {   INF,   INF,   INF,   INF,   INF }
03888    , {   INF,   INF,   INF,   INF,   INF }
03889    , {   INF,   INF,   INF,   INF,   INF }
03890    }
03891  , { {   INF,   INF,   INF,   INF,   INF }
03892    , {   INF,   INF,   INF,   INF,   INF }
03893    , {   INF,   INF,   INF,   INF,   INF }
```



```
03894     , {   INF,   INF,   INF,   INF,   INF }
03895     , {   INF,   INF,   INF,   INF,   INF }
03896     }
03897     , { {   INF,   INF,   INF,   INF,   INF }
03898     , {   INF,   INF,   INF,   INF,   INF }
03899     , {   INF,   INF,   INF,   INF,   INF }
03900     , {   INF,   INF,   INF,   INF,   INF }
03901     , {   INF,   INF,   INF,   INF,   INF }
03902     }
03903     }
03904     }
03905     , { { { 240, -780, -870, 240, -870 }
03906     , { 190, -1060, -1060, 190, -970 }
03907     , { 240, -780, -1010, 240, -1010 }
03908     , { 190, -870, -870, 190, -870 }
03909     , { 130, -890, -1120, 130, -1120 }
03910     }
03911     , { { 40, -1210, -1180, 40, -970 }
03912     , { 40, -1210, -1210, 40, -970 }
03913     , { -270, -1520, -1520, -270, -1520 }
03914     , { -1180, -1420, -1180, -1250, -1180 }
03915     , { -270, -1520, -1520, -270, -1520 }
03916     }
03917     , { { 190, -840, -1060, 190, -1060 }
03918     , { 190, -1060, -1060, 190, -1060 }
03919     , { 180, -840, -1070, 180, -1070 }
03920     , { 190, -1060, -1060, 190, -1060 }
03921     , { 130, -890, -1120, 130, -1120 }
03922     }
03923     , { { -270, -870, -870, -270, -870 }
03924     , { -1470, -1710, -1470, -1530, -1470 }
03925     , { -270, -1520, -1520, -270, -1520 }
03926     , { -870, -870, -870, -870, -870 }
03927     , { -270, -1520, -1520, -270, -1520 }
03928     }
03929     , { { 240, -780, -1010, 240, -1010 }
03930     , { 190, -1060, -1060, 190, -1060 }
03931     , { 240, -780, -1010, 240, -1010 }
03932     , { 190, -1060, -1060, 190, -1060 }
03933     , { -1680, -1790, -1850, -1680, -1850 }
03934     }
03935     }
03936     , { { { -590, -1050, -870, -590, -870 }
03937     , { -890, -1240, -1060, -890, -1060 }
03938     , { -590, -1190, -1010, -590, -1010 }
03939     , { -870, -1050, -870, -890, -870 }
03940     , { -700, -1300, -1120, -700, -1120 }
03941     }
03942     , { { -1030, -1370, -1210, -1030, -1210 }
03943     , { -1030, -1370, -1210, -1030, -1210 }
03944     , { -1340, -1700, -1520, -1340, -1520 }
03945     , { -1250, -1600, -1420, -1250, -1420 }
03946     , { -1340, -1700, -1520, -1340, -1520 }
03947     }
03948     , { { -650, -1240, -1060, -650, -1060 }
03949     , { -890, -1240, -1060, -890, -1060 }
03950     , { -650, -1250, -1070, -650, -1070 }
03951     , { -890, -1240, -1060, -890, -1060 }
03952     , { -700, -1300, -1120, -700, -1120 }
03953     }
03954     , { { -870, -1050, -870, -1340, -870 }
03955     , { -1530, -1890, -1710, -1530, -1710 }
03956     , { -1340, -1700, -1520, -1340, -1520 }
03957     , { -870, -1050, -870, -1940, -870 }
03958     , { -1340, -1700, -1520, -1340, -1520 }
03959     }
03960     , { { -590, -1190, -1010, -590, -1010 }
03961     , { -890, -1240, -1060, -890, -1060 }
03962     , { -590, -1190, -1010, -590, -1010 }
03963     , { -890, -1240, -1060, -890, -1060 }
03964     , { -1680, -1790, -1850, -1680, -1850 }
03965     }
03966     }
03967     , { { { -870, -870, -870, -870, -870 }
03968     , { -1060, -1060, -1060, -1060, -1060 }
03969     , { -1010, -1010, -1010, -1010, -1010 }
03970     , { -870, -870, -870, -870, -870 }
03971     , { -1120, -1120, -1120, -1120, -1120 }
03972     }
03973     , { { -1180, -1210, -1180, -1210, -1180 }
03974     , { -1210, -1210, -1210, -1210, -1210 }
03975     , { -1520, -1520, -1520, -1520, -1520 }
03976     , { -1180, -1420, -1180, -1420, -1180 }
03977     , { -1520, -1520, -1520, -1520, -1520 }
03978     }
03979     , { { -1060, -1060, -1060, -1060, -1060 }
03980     , { -1060, -1060, -1060, -1060, -1060 }
```

```

03981      , { -1070, -1070, -1070, -1070, -1070}
03982      , { -1060, -1060, -1060, -1060, -1060}
03983      , { -1120, -1120, -1120, -1120, -1120}
03984      }
03985      , { { -870, -870, -870, -870, -870}
03986      , { -1470, -1710, -1470, -1710, -1470}
03987      , { -1520, -1520, -1520, -1520, -1520}
03988      , { -870, -870, -870, -870, -870}
03989      , { -1520, -1520, -1520, -1520, -1520}
03990      }
03991      , { { -1010, -1010, -1010, -1010, -1010}
03992      , { -1060, -1060, -1060, -1060, -1060}
03993      , { -1010, -1010, -1010, -1010, -1010}
03994      , { -1060, -1060, -1060, -1060, -1060}
03995      , { -1850, -1850, -1850, -1850, -1850}
03996      }
03997      }
03998      , { { { 240, -780, -870, 240, -870}
03999      , { 190, -1080, -1060, 190, -1060}
04000      , { 240, -780, -1010, 240, -1010}
04001      , { 190, -1080, -870, 190, -870}
04002      , { 130, -890, -1120, 130, -1120}
04003      }
04004      , { { 40, -1220, -1210, 40, -1210}
04005      , { 40, -1220, -1210, 40, -1210}
04006      , { -270, -1530, -1520, -270, -1520}
04007      , { -1420, -1440, -1420, -1420, -1420}
04008      , { -270, -1530, -1520, -270, -1520}
04009      }
04010      , { { 190, -840, -1060, 190, -1060}
04011      , { 190, -1080, -1060, 190, -1060}
04012      , { 180, -840, -1070, 180, -1070}
04013      , { 190, -1080, -1060, 190, -1060}
04014      , { 130, -890, -1120, 130, -1120}
04015      }
04016      , { { -270, -1530, -870, -270, -870}
04017      , { -1710, -1720, -1710, -1710, -1710}
04018      , { -270, -1530, -1520, -270, -1520}
04019      , { -870, -2130, -870, -2120, -870}
04020      , { -270, -1530, -1520, -270, -1520}
04021      }
04022      , { { { 240, -780, -1010, 240, -1010}
04023      , { 190, -1080, -1060, 190, -1060}
04024      , { 240, -780, -1010, 240, -1010}
04025      , { 190, -1080, -1060, 190, -1060}
04026      , { -1850, -1870, -1850, -1850, -1850}
04027      }
04028      }
04029      , { { { -870, -870, -870, -870, -970}
04030      , { -970, -1060, -1060, -1060, -970}
04031      , { -1010, -1010, -1010, -1010, -1010}
04032      , { -870, -870, -870, -870, -1060}
04033      , { -1120, -1120, -1120, -1120, -1120}
04034      }
04035      , { { -970, -1210, -1180, -1210, -970}
04036      , { -970, -1210, -1210, -1210, -970}
04037      , { -1520, -1520, -1520, -1520, -1520}
04038      , { -1180, -1420, -1180, -1420, -1420}
04039      , { -1520, -1520, -1520, -1520, -1520}
04040      }
04041      , { { -1060, -1060, -1060, -1060, -1060}
04042      , { -1060, -1060, -1060, -1060, -1060}
04043      , { -1070, -1070, -1070, -1070, -1070}
04044      , { -1060, -1060, -1060, -1060, -1060}
04045      , { -1120, -1120, -1120, -1120, -1120}
04046      }
04047      , { { -870, -870, -870, -870, -1520}
04048      , { -1470, -1710, -1470, -1710, -1710}
04049      , { -1520, -1520, -1520, -1520, -1520}
04050      , { -870, -870, -870, -870, -2120}
04051      , { -1520, -1520, -1520, -1520, -1520}
04052      }
04053      , { { -1010, -1010, -1010, -1010, -1010}
04054      , { -1060, -1060, -1060, -1060, -1060}
04055      , { -1010, -1010, -1010, -1010, -1010}
04056      , { -1060, -1060, -1060, -1060, -1060}
04057      , { -1850, -1850, -1850, -1850, -1850}
04058      }
04059      }
04060      }
04061      , { { { { 210, -870, -870, 210, -800}
04062      , { 210, -1040, -1040, 210, -800}
04063      , { -240, -1490, -1490, -240, -1490}
04064      , { -160, -870, -870, -160, -870}
04065      , { -240, -1490, -1490, -240, -1490}
04066      }
04067      , { { 210, -1040, -1040, 210, -800}

```

```
04068     , {    210, -1040, -1040,    210,   -800}
04069     , {   -240, -1490, -1490,   -240, -1490}
04070     , { -1990, -2230, -1990, -2060, -1990}
04071     , {   -240, -1490, -1490,   -240, -1490}
04072     }
04073     , { {   -160, -1410, -1410,   -160, -1410}
04074     , {   -160, -1410, -1410,   -160, -1410}
04075     , {   -460, -1490, -1710,   -460, -1710}
04076     , {   -160, -1410, -1410,   -160, -1410}
04077     , {   -460, -1490, -1710,   -460, -1710}
04078     }
04079     , { {   -240,   -870,   -870,   -240,   -870}
04080     , { -1520, -1760, -1520, -1580, -1520}
04081     , {   -240, -1490, -1490,   -240, -1490}
04082     , {   -870,   -870,   -870,   -870,   -870}
04083     , {   -240, -1490, -1490,   -240, -1490}
04084     }
04085     , { {   -160, -1410, -1410,   -160, -1410}
04086     , {   -160, -1410, -1410,   -160, -1410}
04087     , {   -770, -1800, -2020,   -770, -2020}
04088     , {   -160, -1410, -1410,   -160, -1410}
04089     , { -1520, -1640, -1700, -1520, -1700}
04090     }
04091     }
04092     , { { {   -870, -1050,   -870,   -870,   -870}
04093     , {   -870, -1220, -1040,   -870, -1040}
04094     , { -1300, -1670, -1490, -1300, -1490}
04095     , {   -870, -1050,   -870, -1230,   -870}
04096     , { -1300, -1640, -1490, -1300, -1490}
04097     }
04098     , { {   -870, -1220, -1040,   -870, -1040}
04099     , {   -870, -1220, -1040,   -870, -1040}
04100     , { -1320, -1670, -1490, -1320, -1490}
04101     , { -2060, -2410, -2230, -2060, -2230}
04102     , { -1320, -1670, -1490, -1320, -1490}
04103     }
04104     , { { -1230, -1590, -1410, -1230, -1410}
04105     , { -1230, -1590, -1410, -1230, -1410}
04106     , { -1300, -1890, -1710, -1300, -1710}
04107     , { -1230, -1590, -1410, -1230, -1410}
04108     , { -1300, -1890, -1710, -1300, -1710}
04109     }
04110     , { {   -870, -1050,   -870, -1320,   -870}
04111     , { -1580, -1940, -1760, -1580, -1760}
04112     , { -1320, -1670, -1490, -1320, -1490}
04113     , {   -870, -1050,   -870, -1940,   -870}
04114     , { -1320, -1670, -1490, -1320, -1490}
04115     }
04116     , { { -1230, -1590, -1410, -1230, -1410}
04117     , { -1230, -1590, -1410, -1230, -1410}
04118     , { -1610, -2200, -2020, -1610, -2020}
04119     , { -1230, -1590, -1410, -1230, -1410}
04120     , { -1520, -1640, -1700, -1520, -1700}
04121     }
04122     }
04123     , { { {   -870,   -870,   -870,   -870,   -870}
04124     , { -1040, -1040, -1040, -1040, -1040}
04125     , { -1490, -1490, -1490, -1490, -1490}
04126     , {   -870,   -870,   -870,   -870,   -870}
04127     , { -1490, -1490, -1490, -1490, -1490}
04128     }
04129     , { { -1040, -1040, -1040, -1040, -1040}
04130     , { -1040, -1040, -1040, -1040, -1040}
04131     , { -1490, -1490, -1490, -1490, -1490}
04132     , { -1990, -2230, -1990, -2230, -1990}
04133     , { -1490, -1490, -1490, -1490, -1490}
04134     }
04135     , { { -1410, -1410, -1410, -1410, -1410}
04136     , { -1410, -1410, -1410, -1410, -1410}
04137     , { -1710, -1710, -1710, -1710, -1710}
04138     , { -1410, -1410, -1410, -1410, -1410}
04139     , { -1710, -1710, -1710, -1710, -1710}
04140     }
04141     , { {   -870,   -870,   -870,   -870,   -870}
04142     , { -1520, -1760, -1520, -1760, -1520}
04143     , { -1490, -1490, -1490, -1490, -1490}
04144     , {   -870,   -870,   -870,   -870,   -870}
04145     , { -1490, -1490, -1490, -1490, -1490}
04146     }
04147     , { { -1410, -1410, -1410, -1410, -1410}
04148     , { -1410, -1410, -1410, -1410, -1410}
04149     , { -2020, -2020, -2020, -2020, -2020}
04150     , { -1410, -1410, -1410, -1410, -1410}
04151     , { -1700, -1700, -1700, -1700, -1700}
04152     }
04153     }
04154     , { { {    210, -1060,   -870,    210,   -870}
```

```
04155 , { 210, -1060, -1040, 210, -1040}
04156 , { -240, -1490, -1490, -240, -1490}
04157 , { -160, -1420, -870, -160, -870}
04158 , { -240, -1490, -1490, -240, -1490}
04159 }
04160 , { { 210, -1060, -1040, 210, -1040}
04161 , { 210, -1060, -1040, 210, -1040}
04162 , { -240, -1510, -1490, -240, -1490}
04163 , { -2230, -2250, -2230, -2230, -2230}
04164 , { -240, -1510, -1490, -240, -1490}
04165 }
04166 , { { -160, -1420, -1410, -160, -1410}
04167 , { -160, -1420, -1410, -160, -1410}
04168 , { -460, -1490, -1710, -460, -1710}
04169 , { -160, -1420, -1410, -160, -1410}
04170 , { -460, -1490, -1710, -460, -1710}
04171 }
04172 , { { -240, -1510, -870, -240, -870}
04173 , { -1760, -1770, -1760, -1760, -1760}
04174 , { -240, -1510, -1490, -240, -1490}
04175 , { -870, -2130, -870, -2120, -870}
04176 , { -240, -1510, -1490, -240, -1490}
04177 }
04178 , { { -160, -1420, -1410, -160, -1410}
04179 , { -160, -1420, -1410, -160, -1410}
04180 , { -770, -1800, -2020, -770, -2020}
04181 , { -160, -1420, -1410, -160, -1410}
04182 , { -1700, -1710, -1700, -1700, -1700}
04183 }
04184 }
04185 , { { { -800, -870, -870, -870, -800}
04186 , { -800, -1040, -1040, -1040, -800}
04187 , { -1490, -1490, -1490, -1490, -1490}
04188 , { -870, -870, -870, -870, -1410}
04189 , { -1490, -1490, -1490, -1490, -1490}
04190 }
04191 , { { -800, -1040, -1040, -1040, -800}
04192 , { -800, -1040, -1040, -1040, -800}
04193 , { -1490, -1490, -1490, -1490, -1490}
04194 , { -1990, -2230, -1990, -2230, -2230}
04195 , { -1490, -1490, -1490, -1490, -1490}
04196 }
04197 , { { -1410, -1410, -1410, -1410, -1410}
04198 , { -1410, -1410, -1410, -1410, -1410}
04199 , { -1710, -1710, -1710, -1710, -1710}
04200 , { -1410, -1410, -1410, -1410, -1410}
04201 , { -1710, -1710, -1710, -1710, -1710}
04202 }
04203 , { { -870, -870, -870, -870, -1490}
04204 , { -1520, -1760, -1520, -1760, -1760}
04205 , { -1490, -1490, -1490, -1490, -1490}
04206 , { -870, -870, -870, -870, -2120}
04207 , { -1490, -1490, -1490, -1490, -1490}
04208 }
04209 , { { -1410, -1410, -1410, -1410, -1410}
04210 , { -1410, -1410, -1410, -1410, -1410}
04211 , { -2020, -2020, -2020, -2020, -2020}
04212 , { -1410, -1410, -1410, -1410, -1410}
04213 , { -1700, -1700, -1700, -1700, -1700}
04214 }
04215 }
04216 }
04217 , { { { { -710, -710, -710, -710, -710}
04218 , { -710, -1780, -1540, -710, -1540}
04219 , { -710, -1730, -1960, -710, -1960}
04220 , { -710, -710, -710, -710, -710}
04221 , { -710, -1730, -1960, -710, -1960}
04222 }
04223 , { { -710, -1960, -1730, -710, -1730}
04224 , { -890, -2140, -2140, -890, -1900}
04225 , { -710, -1960, -1960, -710, -1960}
04226 , { -1730, -1970, -1730, -1800, -1730}
04227 , { -710, -1960, -1960, -710, -1960}
04228 }
04229 , { { -710, -1730, -1960, -710, -1960}
04230 , { -710, -1960, -1960, -710, -1960}
04231 , { -710, -1730, -1960, -710, -1960}
04232 , { -710, -1960, -1960, -710, -1960}
04233 , { -710, -1730, -1960, -710, -1960}
04234 }
04235 , { { -710, -710, -710, -710, -710}
04236 , { -1540, -1780, -1540, -1610, -1540}
04237 , { -710, -1960, -1960, -710, -1960}
04238 , { -710, -710, -710, -710, -710}
04239 , { -710, -1960, -1960, -710, -1960}
04240 }
04241 , { { -710, -1730, -1960, -710, -1960}
```

```
04242     , { -710, -1960, -1960, -710, -1960}
04243     , { -710, -1730, -1960, -710, -1960}
04244     , { -710, -1960, -1960, -710, -1960}
04245     , { -1780, -1900, -1960, -1780, -1960}
04246     }
04247     }
04248     , {{{ -710, -890, -710, -1540, -710}
04249     , { -1610, -1960, -1780, -1610, -1780}
04250     , { -1540, -2140, -1960, -1540, -1960}
04251     , { -710, -890, -710, -1780, -710}
04252     , { -1540, -1900, -1960, -1540, -1960}
04253     }
04254     , {{{ -1780, -2140, -1960, -1780, -1960}
04255     , { -1960, -2320, -2140, -1960, -2140}
04256     , { -1780, -2140, -1960, -1780, -1960}
04257     , { -1800, -2150, -1970, -1800, -1970}
04258     , { -1780, -2140, -1960, -1780, -1960}
04259     }
04260     , {{{ -1540, -2140, -1960, -1540, -1960}
04261     , { -1780, -2140, -1960, -1780, -1960}
04262     , { -1540, -2140, -1960, -1540, -1960}
04263     , { -1780, -2140, -1960, -1780, -1960}
04264     , { -1540, -2140, -1960, -1540, -1960}
04265     }
04266     , {{{ -710, -890, -710, -1610, -710}
04267     , { -1610, -1960, -1780, -1610, -1780}
04268     , { -1780, -2140, -1960, -1780, -1960}
04269     , { -710, -890, -710, -1780, -710}
04270     , { -1780, -2140, -1960, -1780, -1960}
04271     }
04272     , {{{ -1540, -1900, -1960, -1540, -1960}
04273     , { -1780, -2140, -1960, -1780, -1960}
04274     , { -1540, -2140, -1960, -1540, -1960}
04275     , { -1780, -2140, -1960, -1780, -1960}
04276     , { -1780, -1900, -1960, -1780, -1960}
04277     }
04278     }
04279     , {{{ -710, -710, -710, -710, -710}
04280     , { -1540, -1780, -1540, -1780, -1540}
04281     , { -1960, -1960, -1960, -1960, -1960}
04282     , { -710, -710, -710, -710, -710}
04283     , { -1960, -1960, -1960, -1960, -1960}
04284     }
04285     , {{{ -1730, -1960, -1730, -1960, -1730}
04286     , { -2140, -2140, -2140, -2140, -2140}
04287     , { -1960, -1960, -1960, -1960, -1960}
04288     , { -1730, -1970, -1730, -1970, -1730}
04289     , { -1960, -1960, -1960, -1960, -1960}
04290     }
04291     , {{{ -1960, -1960, -1960, -1960, -1960}
04292     , { -1960, -1960, -1960, -1960, -1960}
04293     , { -1960, -1960, -1960, -1960, -1960}
04294     , { -1960, -1960, -1960, -1960, -1960}
04295     , { -1960, -1960, -1960, -1960, -1960}
04296     }
04297     , {{{ -710, -710, -710, -710, -710}
04298     , { -1540, -1780, -1540, -1780, -1540}
04299     , { -1960, -1960, -1960, -1960, -1960}
04300     , { -710, -710, -710, -710, -710}
04301     , { -1960, -1960, -1960, -1960, -1960}
04302     }
04303     , {{{ -1960, -1960, -1960, -1960, -1960}
04304     , { -1960, -1960, -1960, -1960, -1960}
04305     , { -1960, -1960, -1960, -1960, -1960}
04306     , { -1960, -1960, -1960, -1960, -1960}
04307     , { -1960, -1960, -1960, -1960, -1960}
04308     }
04309     }
04310     , {{{ -710, -1730, -710, -710, -710}
04311     , { -710, -1800, -1780, -710, -1780}
04312     , { -710, -1730, -1960, -710, -1960}
04313     , { -710, -1970, -710, -710, -710}
04314     , { -710, -1730, -1960, -710, -1960}
04315     }
04316     , {{{ -710, -1970, -1960, -710, -1960}
04317     , { -890, -2150, -2140, -890, -2140}
04318     , { -710, -1970, -1960, -710, -1960}
04319     , { -1970, -1990, -1970, -1970, -1970}
04320     , { -710, -1970, -1960, -710, -1960}
04321     }
04322     , {{{ -710, -1730, -1960, -710, -1960}
04323     , { -710, -1970, -1960, -710, -1960}
04324     , { -710, -1730, -1960, -710, -1960}
04325     , { -710, -1970, -1960, -710, -1960}
04326     , { -710, -1730, -1960, -710, -1960}
04327     }
04328     , {{{ -710, -1800, -710, -710, -710}
```

```
04329      , { -1780, -1800, -1780, -1780, -1780 }
04330      , { -710, -1970, -1960, -710, -1960 }
04331      , { -710, -1970, -710, -1960, -710 }
04332      , { -710, -1970, -1960, -710, -1960 }
04333      }
04334      , { { -710, -1730, -1960, -710, -1960 }
04335      , { -710, -1970, -1960, -710, -1960 }
04336      , { -710, -1730, -1960, -710, -1960 }
04337      , { -710, -1970, -1960, -710, -1960 }
04338      , { -1960, -1970, -1960, -1960, -1960 }
04339      }
04340      }
04341      , { { { -710, -710, -710, -710, -1780 }
04342      , { -1540, -1780, -1540, -1780, -1780 }
04343      , { -1960, -1960, -1960, -1960, -1960 }
04344      , { -710, -710, -710, -710, -1960 }
04345      , { -1960, -1960, -1960, -1960, -1960 }
04346      }
04347      , { { -1730, -1960, -1730, -1960, -1900 }
04348      , { -1900, -2140, -2140, -2140, -1900 }
04349      , { -1960, -1960, -1960, -1960, -1960 }
04350      , { -1730, -1970, -1730, -1970, -1970 }
04351      , { -1960, -1960, -1960, -1960, -1960 }
04352      }
04353      , { { -1960, -1960, -1960, -1960, -1960 }
04354      , { -1960, -1960, -1960, -1960, -1960 }
04355      , { -1960, -1960, -1960, -1960, -1960 }
04356      , { -1960, -1960, -1960, -1960, -1960 }
04357      , { -1960, -1960, -1960, -1960, -1960 }
04358      }
04359      , { { -710, -710, -710, -710, -1780 }
04360      , { -1540, -1780, -1540, -1780, -1780 }
04361      , { -1960, -1960, -1960, -1960, -1960 }
04362      , { -710, -710, -710, -710, -1960 }
04363      , { -1960, -1960, -1960, -1960, -1960 }
04364      }
04365      , { { -1960, -1960, -1960, -1960, -1960 }
04366      , { -1960, -1960, -1960, -1960, -1960 }
04367      , { -1960, -1960, -1960, -1960, -1960 }
04368      , { -1960, -1960, -1960, -1960, -1960 }
04369      , { -1960, -1960, -1960, -1960, -1960 }
04370      }
04371      }
04372      }
04373      , { { { { 360, -70, -150, 360, -150 }
04374      , { 360, -70, -890, 360, -650 }
04375      , { -150, -1180, -1400, -150, -1400 }
04376      , { -150, -150, -150, -150, -150 }
04377      , { -150, -1180, -1400, -150, -1400 }
04378      }
04379      , { { { 360, -70, -890, 360, -650 }
04380      , { 360, -70, -890, 360, -650 }
04381      , { -150, -1400, -1400, -150, -1400 }
04382      , { -1500, -1600, -1500, -1570, -1500 }
04383      , { -150, -1400, -1400, -150, -1400 }
04384      }
04385      , { { { -150, -1180, -1400, -150, -1400 }
04386      , { -150, -1400, -1400, -150, -1400 }
04387      , { -150, -1180, -1400, -150, -1400 }
04388      , { -150, -1400, -1400, -150, -1400 }
04389      , { -150, -1180, -1400, -150, -1400 }
04390      }
04391      , { { { -150, -150, -150, -150, -150 }
04392      , { -1670, -1910, -1670, -1740, -1670 }
04393      , { -150, -1400, -1400, -150, -1400 }
04394      , { -150, -150, -150, -150, -150 }
04395      , { -150, -1400, -1400, -150, -1400 }
04396      }
04397      , { { { -150, -1180, -1400, -150, -1400 }
04398      , { -150, -1400, -1400, -150, -1400 }
04399      , { -150, -1180, -1400, -150, -1400 }
04400      , { -150, -1400, -1400, -150, -1400 }
04401      , { -1230, -1340, -1400, -1230, -1400 }
04402      }
04403      }
04404      , { { { { -30, -70, -150, -30, -150 }
04405      , { -30, -70, -890, -30, -890 }
04406      , { -990, -1580, -1400, -990, -1400 }
04407      , { -150, -330, -150, -1230, -150 }
04408      , { -990, -1340, -1400, -990, -1400 }
04409      }
04410      , { { { -30, -70, -890, -30, -890 }
04411      , { -30, -70, -890, -30, -890 }
04412      , { -1230, -1580, -1400, -1230, -1400 }
04413      , { -1570, -1600, -1740, -1570, -1740 }
04414      , { -1230, -1580, -1400, -1230, -1400 }
04415      }
```

```
04416 ,{{ -990, -1580, -1400, -990, -1400}
04417 ,{ -1230, -1580, -1400, -1230, -1400}
04418 ,{ -990, -1580, -1400, -990, -1400}
04419 ,{ -1230, -1580, -1400, -1230, -1400}
04420 ,{ -990, -1580, -1400, -990, -1400}
04421 }
04422 ,{{ -150, -330, -150, -1230, -150}
04423 ,{ -1740, -2090, -1910, -1740, -1910}
04424 ,{ -1230, -1580, -1400, -1230, -1400}
04425 ,{ -150, -330, -150, -1230, -150}
04426 ,{ -1230, -1580, -1400, -1230, -1400}
04427 }
04428 ,{{ -990, -1340, -1400, -990, -1400}
04429 ,{ -1230, -1580, -1400, -1230, -1400}
04430 ,{ -990, -1580, -1400, -990, -1400}
04431 ,{ -1230, -1580, -1400, -1230, -1400}
04432 ,{ -1230, -1340, -1400, -1230, -1400}
04433 }
04434 }
04435 ,{{{ -150, -150, -150, -150, -150}
04436 ,{ -890, -890, -890, -890, -890}
04437 ,{ -1400, -1400, -1400, -1400, -1400}
04438 ,{ -150, -150, -150, -150, -150}
04439 ,{ -1400, -1400, -1400, -1400, -1400}
04440 }
04441 ,{{{ -890, -890, -890, -890, -890}
04442 ,{ -890, -890, -890, -890, -890}
04443 ,{ -1400, -1400, -1400, -1400, -1400}
04444 ,{ -1500, -1740, -1500, -1740, -1500}
04445 ,{ -1400, -1400, -1400, -1400, -1400}
04446 }
04447 ,{{{ -1400, -1400, -1400, -1400, -1400}
04448 ,{ -1400, -1400, -1400, -1400, -1400}
04449 ,{ -1400, -1400, -1400, -1400, -1400}
04450 ,{ -1400, -1400, -1400, -1400, -1400}
04451 ,{ -1400, -1400, -1400, -1400, -1400}
04452 }
04453 ,{{{ -150, -150, -150, -150, -150}
04454 ,{ -1670, -1910, -1670, -1910, -1670}
04455 ,{ -1400, -1400, -1400, -1400, -1400}
04456 ,{ -150, -150, -150, -150, -150}
04457 ,{ -1400, -1400, -1400, -1400, -1400}
04458 }
04459 ,{{{ -1400, -1400, -1400, -1400, -1400}
04460 ,{ -1400, -1400, -1400, -1400, -1400}
04461 ,{ -1400, -1400, -1400, -1400, -1400}
04462 ,{ -1400, -1400, -1400, -1400, -1400}
04463 ,{ -1400, -1400, -1400, -1400, -1400}
04464 }
04465 }
04466 ,{{{ 360, -910, -150, 360, -150}
04467 ,{ 360, -910, -890, 360, -890}
04468 ,{ -150, -1180, -1400, -150, -1400}
04469 ,{ -150, -1420, -150, -150, -150}
04470 ,{ -150, -1180, -1400, -150, -1400}
04471 }
04472 ,{{{ 360, -910, -890, 360, -890}
04473 ,{ 360, -910, -890, 360, -890}
04474 ,{ -150, -1420, -1400, -150, -1400}
04475 ,{ -1740, -3040, -1740, -1740, -1740}
04476 ,{ -150, -1420, -1400, -150, -1400}
04477 }
04478 ,{{{ -150, -1180, -1400, -150, -1400}
04479 ,{ -150, -1420, -1400, -150, -1400}
04480 ,{ -150, -1180, -1400, -150, -1400}
04481 ,{ -150, -1420, -1400, -150, -1400}
04482 ,{ -150, -1180, -1400, -150, -1400}
04483 }
04484 ,{{{ -150, -1420, -150, -150, -150}
04485 ,{ -1910, -1930, -1910, -1910, -1910}
04486 ,{ -150, -1420, -1400, -150, -1400}
04487 ,{ -150, -1420, -150, -1400, -150}
04488 ,{ -150, -1420, -1400, -150, -1400}
04489 }
04490 ,{{{ -150, -1180, -1400, -150, -1400}
04491 ,{ -150, -1420, -1400, -150, -1400}
04492 ,{ -150, -1180, -1400, -150, -1400}
04493 ,{ -150, -1420, -1400, -150, -1400}
04494 ,{ -1400, -1420, -1400, -1400, -1400}
04495 }
04496 }
04497 ,{{{ -150, -150, -150, -150, -650}
04498 ,{ -650, -890, -890, -890, -650}
04499 ,{ -1400, -1400, -1400, -1400, -1400}
04500 ,{ -150, -150, -150, -150, -1400}
04501 ,{ -1400, -1400, -1400, -1400, -1400}
04502 }
```

```
04503 ,{{ -650, -890, -890, -890, -650}
04504 ,{ -650, -890, -890, -890, -650}
04505 ,{ -1400, -1400, -1400, -1400, -1400}
04506 ,{ -1500, -1740, -1500, -1740, -1740}
04507 ,{ -1400, -1400, -1400, -1400, -1400}
04508 }
04509 ,{{ -1400, -1400, -1400, -1400, -1400}
04510 ,{ -1400, -1400, -1400, -1400, -1400}
04511 ,{ -1400, -1400, -1400, -1400, -1400}
04512 ,{ -1400, -1400, -1400, -1400, -1400}
04513 ,{ -1400, -1400, -1400, -1400, -1400}
04514 }
04515 ,{{ -150, -150, -150, -150, -1400}
04516 ,{ -1670, -1910, -1670, -1910, -1910}
04517 ,{ -1400, -1400, -1400, -1400, -1400}
04518 ,{ -150, -150, -150, -150, -1400}
04519 ,{ -1400, -1400, -1400, -1400, -1400}
04520 }
04521 ,{{ -1400, -1400, -1400, -1400, -1400}
04522 ,{ -1400, -1400, -1400, -1400, -1400}
04523 ,{ -1400, -1400, -1400, -1400, -1400}
04524 ,{ -1400, -1400, -1400, -1400, -1400}
04525 ,{ -1400, -1400, -1400, -1400, -1400}
04526 }
04527 }
04528 }
04529 ,{{{ 940, 220, 220, 940, 220}
04530 ,{ 940, -310, -310, 940, -70}
04531 ,{ 640, -380, -610, 640, -610}
04532 ,{ 650, 220, 220, 650, 220}
04533 ,{ 640, -380, -610, 640, -610}
04534 }
04535 ,{{ 940, -310, -310, 940, -70}
04536 ,{ 940, -310, -310, 940, -70}
04537 ,{ 630, -620, -620, 630, -620}
04538 ,{ -1460, -1700, -1460, -1520, -1460}
04539 ,{ 630, -620, -620, 630, -620}
04540 }
04541 ,{{ 650, -380, -600, 650, -600}
04542 ,{ 650, -600, -600, 650, -600}
04543 ,{ 640, -380, -610, 640, -610}
04544 ,{ 650, -600, -600, 650, -600}
04545 ,{ 640, -380, -610, 640, -610}
04546 }
04547 ,{{ 630, 220, 220, 630, 220}
04548 ,{ -1280, -1520, -1280, -1340, -1280}
04549 ,{ 630, -620, -620, 630, -620}
04550 ,{ 220, 220, 220, 220, 220}
04551 ,{ 630, -620, -620, 630, -620}
04552 }
04553 ,{{ 650, -380, -600, 650, -600}
04554 ,{ 650, -600, -600, 650, -600}
04555 ,{ 640, -380, -610, 640, -610}
04556 ,{ 650, -600, -600, 650, -600}
04557 ,{ -1410, -1530, -1590, -1410, -1590}
04558 }
04559 }
04560 ,{{{ 220, 40, 220, -130, 220}
04561 ,{ -130, -490, -310, -130, -310}
04562 ,{ -190, -790, -610, -190, -610}
04563 ,{ 220, 40, 220, -430, 220}
04564 ,{ -190, -790, -610, -190, -610}
04565 }
04566 ,{{ -130, -490, -310, -130, -310}
04567 ,{ -130, -490, -310, -130, -310}
04568 ,{ -440, -800, -620, -440, -620}
04569 ,{ -1520, -1880, -1700, -1520, -1700}
04570 ,{ -440, -800, -620, -440, -620}
04571 }
04572 ,{{ -190, -780, -600, -190, -600}
04573 ,{ -430, -780, -600, -430, -600}
04574 ,{ -190, -790, -610, -190, -610}
04575 ,{ -430, -780, -600, -430, -600}
04576 ,{ -190, -790, -610, -190, -610}
04577 }
04578 ,{{ 220, 40, 220, -440, 220}
04579 ,{ -1340, -1700, -1520, -1340, -1520}
04580 ,{ -440, -800, -620, -440, -620}
04581 ,{ 220, 40, 220, -850, 220}
04582 ,{ -440, -800, -620, -440, -620}
04583 }
04584 ,{{ -190, -780, -600, -190, -600}
04585 ,{ -430, -780, -600, -430, -600}
04586 ,{ -190, -790, -610, -190, -610}
04587 ,{ -430, -780, -600, -430, -600}
04588 ,{ -1410, -1530, -1590, -1410, -1590}
04589 }
```



```
04590     }
04591     , {{ { 220, 220, 220, 220, 220 }
04592     , { -310, -310, -310, -310, -310 }
04593     , { -610, -610, -610, -610, -610 }
04594     , { 220, 220, 220, 220, 220 }
04595     , { -610, -610, -610, -610, -610 }
04596     }
04597     , {{ { -310, -310, -310, -310, -310 }
04598     , { -310, -310, -310, -310, -310 }
04599     , { -620, -620, -620, -620, -620 }
04600     , { -1460, -1700, -1460, -1700, -1460 }
04601     , { -620, -620, -620, -620, -620 }
04602     }
04603     , {{ { -600, -600, -600, -600, -600 }
04604     , { -600, -600, -600, -600, -600 }
04605     , { -610, -610, -610, -610, -610 }
04606     , { -600, -600, -600, -600, -600 }
04607     , { -610, -610, -610, -610, -610 }
04608     }
04609     , {{ { 220, 220, 220, 220, 220 }
04610     , { -1280, -1520, -1280, -1520, -1280 }
04611     , { -620, -620, -620, -620, -620 }
04612     , { 220, 220, 220, 220, 220 }
04613     , { -620, -620, -620, -620, -620 }
04614     }
04615     , {{ { -600, -600, -600, -600, -600 }
04616     , { -600, -600, -600, -600, -600 }
04617     , { -610, -610, -610, -610, -610 }
04618     , { -600, -600, -600, -600, -600 }
04619     , { -1590, -1590, -1590, -1590, -1590 }
04620     }
04621     }
04622     , {{ { 940, -320, 220, 940, 220 }
04623     , { 940, -320, -310, 940, -310 }
04624     , { 640, -380, -610, 640, -610 }
04625     , { 650, -620, 220, 650, 220 }
04626     , { 640, -380, -610, 640, -610 }
04627     }
04628     , {{ { 940, -320, -310, 940, -310 }
04629     , { 940, -320, -310, 940, -310 }
04630     , { 630, -630, -620, 630, -620 }
04631     , { -1700, -1710, -1700, -1700, -1700 }
04632     , { 630, -630, -620, 630, -620 }
04633     }
04634     , {{ { 650, -380, -600, 650, -600 }
04635     , { 650, -620, -600, 650, -600 }
04636     , { 640, -380, -610, 640, -610 }
04637     , { 650, -620, -600, 650, -600 }
04638     , { 640, -380, -610, 640, -610 }
04639     }
04640     , {{ { 630, -630, 220, 630, 220 }
04641     , { -1520, -1530, -1520, -1520, -1520 }
04642     , { 630, -630, -620, 630, -620 }
04643     , { 220, -1040, 220, -1030, 220 }
04644     , { 630, -630, -620, 630, -620 }
04645     }
04646     , {{ { 650, -380, -600, 650, -600 }
04647     , { 650, -620, -600, 650, -600 }
04648     , { 640, -380, -610, 640, -610 }
04649     , { 650, -620, -600, 650, -600 }
04650     , { -1590, -1600, -1590, -1590, -1590 }
04651     }
04652     }
04653     , {{ { 220, 220, 220, 220, -70 }
04654     , { -70, -310, -310, -310, -70 }
04655     , { -610, -610, -610, -610, -610 }
04656     , { 220, 220, 220, 220, -600 }
04657     , { -610, -610, -610, -610, -610 }
04658     }
04659     , {{ { -70, -310, -310, -310, -70 }
04660     , { -70, -310, -310, -310, -70 }
04661     , { -620, -620, -620, -620, -620 }
04662     , { -1460, -1700, -1460, -1700, -1700 }
04663     , { -620, -620, -620, -620, -620 }
04664     }
04665     , {{ { -600, -600, -600, -600, -600 }
04666     , { -600, -600, -600, -600, -600 }
04667     , { -610, -610, -610, -610, -610 }
04668     , { -600, -600, -600, -600, -600 }
04669     , { -610, -610, -610, -610, -610 }
04670     }
04671     , {{ { 220, 220, 220, 220, -620 }
04672     , { -1280, -1520, -1280, -1520, -1520 }
04673     , { -620, -620, -620, -620, -620 }
04674     , { 220, 220, 220, 220, -1030 }
04675     , { -620, -620, -620, -620, -620 }
04676     }
```

```
04677 ,{{ -600, -600, -600, -600, -600}
04678 ,{ -600, -600, -600, -600, -600}
04679 ,{ -610, -610, -610, -610, -610}
04680 ,{ -600, -600, -600, -600, -600}
04681 ,{ -1590, -1590, -1590, -1590, -1590}
04682 }
04683 }
04684 }
04685 ,{{{ 1010, 410, 410, 1010, 410}
04686 ,{ 1010, -240, -240, 1010, 0}
04687 ,{ 880, -150, -370, 880, -370}
04688 ,{ 880, 410, 410, 880, 410}
04689 ,{ 750, -280, -500, 750, -500}
04690 }
04691 ,{{ 1010, -240, -240, 1010, 0}
04692 ,{ 1010, -240, -240, 1010, 0}
04693 ,{ 730, -520, -520, 730, -520}
04694 ,{ -1410, -1650, -1410, -1470, -1410}
04695 ,{ 730, -520, -520, 730, -520}
04696 }
04697 ,{{{ 880, -150, -370, 880, -370}
04698 ,{ 880, -370, -370, 880, -370}
04699 ,{ 880, -150, -370, 880, -370}
04700 ,{ 880, -370, -370, 880, -370}
04701 ,{ 750, -280, -500, 750, -500}
04702 }
04703 ,{{{ 730, 410, 410, 730, 410}
04704 ,{ -1710, -1950, -1710, -1770, -1710}
04705 ,{ 730, -520, -520, 730, -520}
04706 ,{ 410, 410, 410, 410, 410}
04707 ,{ 730, -520, -520, 730, -520}
04708 }
04709 ,{{{ 880, -370, -370, 880, -370}
04710 ,{ 880, -370, -370, 880, -370}
04711 ,{ 440, -590, -810, 440, -810}
04712 ,{ 880, -370, -370, 880, -370}
04713 ,{ -1140, -1250, -1310, -1140, -1310}
04714 }
04715 }
04716 ,{{{ 410, 230, 410, 40, 410}
04717 ,{ -70, -420, -240, -70, -240}
04718 ,{ 40, -550, -370, 40, -370}
04719 ,{ 410, 230, 410, -200, 410}
04720 ,{ -90, -680, -500, -90, -500}
04721 }
04722 ,{{{ -70, -420, -240, -70, -240}
04723 ,{ -70, -420, -240, -70, -240}
04724 ,{ -350, -700, -520, -350, -520}
04725 ,{ -1470, -1830, -1650, -1470, -1650}
04726 ,{ -350, -700, -520, -350, -520}
04727 }
04728 ,{{{ 40, -550, -370, 40, -370}
04729 ,{ -200, -550, -370, -200, -370}
04730 ,{ 40, -550, -370, 40, -370}
04731 ,{ -200, -550, -370, -200, -370}
04732 ,{ -90, -680, -500, -90, -500}
04733 }
04734 ,{{{ 410, 230, 410, -350, 410}
04735 ,{ -1770, -2130, -1950, -1770, -1950}
04736 ,{ -350, -700, -520, -350, -520}
04737 ,{ 410, 230, 410, -670, 410}
04738 ,{ -350, -700, -520, -350, -520}
04739 }
04740 ,{{{ -200, -550, -370, -200, -370}
04741 ,{ -200, -550, -370, -200, -370}
04742 ,{ -400, -990, -810, -400, -810}
04743 ,{ -200, -550, -370, -200, -370}
04744 ,{ -1140, -1250, -1310, -1140, -1310}
04745 }
04746 }
04747 ,{{{ 410, 410, 410, 410, 410}
04748 ,{ -240, -240, -240, -240, -240}
04749 ,{ -370, -370, -370, -370, -370}
04750 ,{ 410, 410, 410, 410, 410}
04751 ,{ -500, -500, -500, -500, -500}
04752 }
04753 ,{{{ -240, -240, -240, -240, -240}
04754 ,{ -240, -240, -240, -240, -240}
04755 ,{ -520, -520, -520, -520, -520}
04756 ,{ -1410, -1650, -1410, -1650, -1410}
04757 ,{ -520, -520, -520, -520, -520}
04758 }
04759 ,{{{ -370, -370, -370, -370, -370}
04760 ,{ -370, -370, -370, -370, -370}
04761 ,{ -370, -370, -370, -370, -370}
04762 ,{ -370, -370, -370, -370, -370}
04763 ,{ -500, -500, -500, -500, -500}
```

```
04764     }
04765     ,{{ 410, 410, 410, 410, 410}
04766     ,{{ -1710, -1950, -1710, -1950, -1710}
04767     ,{{ -520, -520, -520, -520, -520}
04768     ,{{ 410, 410, 410, 410, 410}
04769     ,{{ -520, -520, -520, -520, -520}
04770     }
04771     ,{{ -370, -370, -370, -370, -370}
04772     ,{{ -370, -370, -370, -370, -370}
04773     ,{{ -810, -810, -810, -810, -810}
04774     ,{{ -370, -370, -370, -370, -370}
04775     ,{{ -1310, -1310, -1310, -1310, -1310}
04776     }
04777     }
04778     ,{{{ 1010, -150, 410, 1010, 410}
04779     ,{{ 1010, -260, -240, 1010, -240}
04780     ,{{ 880, -150, -370, 880, -370}
04781     ,{{ 880, -390, 410, 880, 410}
04782     ,{{ 750, -280, -500, 750, -500}
04783     }
04784     ,{{{ 1010, -260, -240, 1010, -240}
04785     ,{{ 1010, -260, -240, 1010, -240}
04786     ,{{ 730, -540, -520, 730, -520}
04787     ,{{ -1650, -1660, -1650, -1650, -1650}
04788     ,{{ 730, -540, -520, 730, -520}
04789     }
04790     ,{{{ 880, -150, -370, 880, -370}
04791     ,{{ 880, -390, -370, 880, -370}
04792     ,{{ 880, -150, -370, 880, -370}
04793     ,{{ 880, -390, -370, 880, -370}
04794     ,{{ 750, -280, -500, 750, -500}
04795     }
04796     ,{{{ 730, -540, 410, 730, 410}
04797     ,{{ -1950, -1960, -1950, -1950, -1950}
04798     ,{{ 730, -540, -520, 730, -520}
04799     ,{{ 410, -860, 410, -840, 410}
04800     ,{{ 730, -540, -520, 730, -520}
04801     }
04802     ,{{{ 880, -390, -370, 880, -370}
04803     ,{{ 880, -390, -370, 880, -370}
04804     ,{{ 440, -590, -810, 440, -810}
04805     ,{{ 880, -390, -370, 880, -370}
04806     ,{{ -1310, -1330, -1310, -1310, -1310}
04807     }
04808     }
04809     ,{{{ 410, 410, 410, 410, 0}
04810     ,{{ 0, -240, -240, -240, 0}
04811     ,{{ -370, -370, -370, -370, -370}
04812     ,{{ 410, 410, 410, 410, -370}
04813     ,{{ -500, -500, -500, -500, -500}
04814     }
04815     ,{{{ 0, -240, -240, -240, 0}
04816     ,{{ 0, -240, -240, -240, 0}
04817     ,{{ -520, -520, -520, -520, -520}
04818     ,{{ -1410, -1650, -1410, -1650, -1650}
04819     ,{{ -520, -520, -520, -520, -520}
04820     }
04821     ,{{{ -370, -370, -370, -370, -370}
04822     ,{{ -370, -370, -370, -370, -370}
04823     ,{{ -370, -370, -370, -370, -370}
04824     ,{{ -370, -370, -370, -370, -370}
04825     ,{{ -500, -500, -500, -500, -500}
04826     }
04827     ,{{{ 410, 410, 410, 410, -520}
04828     ,{{ -1710, -1950, -1710, -1950, -1950}
04829     ,{{ -520, -520, -520, -520, -520}
04830     ,{{ 410, 410, 410, 410, -840}
04831     ,{{ -520, -520, -520, -520, -520}
04832     }
04833     ,{{{ -370, -370, -370, -370, -370}
04834     ,{{ -370, -370, -370, -370, -370}
04835     ,{{ -810, -810, -810, -810, -810}
04836     ,{{ -370, -370, -370, -370, -370}
04837     ,{{ -1310, -1310, -1310, -1310, -1310}
04838     }
04839     }
04840     }
04841     ,{{{ 1010, 410, 410, 1010, 410}
04842     ,{{ 1010, -70, -240, 1010, 0}
04843     ,{{ 880, -150, -370, 880, -370}
04844     ,{{ 880, 410, 410, 880, 410}
04845     ,{{ 750, -280, -500, 750, -500}
04846     }
04847     ,{{{ 1010, -70, -240, 1010, 0}
04848     ,{{ 1010, -70, -240, 1010, 0}
04849     ,{{ 730, -520, -520, 730, -520}
04850     ,{{ -1180, -1420, -1180, -1250, -1180}
```

```
04851      , { 730, -520, -520, 730, -520}
04852      }
04853      , { { 880, -150, -370, 880, -370}
04854      , { 880, -370, -370, 880, -370}
04855      , { 880, -150, -370, 880, -370}
04856      , { 880, -370, -370, 880, -370}
04857      , { 750, -280, -500, 750, -500}
04858      }
04859      , { { 730, 410, 410, 730, 410}
04860      , { -1280, -1520, -1280, -1340, -1280}
04861      , { 730, -520, -520, 730, -520}
04862      , { 410, 410, 410, 410, 410}
04863      , { 730, -520, -520, 730, -520}
04864      }
04865      , { { 880, -370, -370, 880, -370}
04866      , { 880, -370, -370, 880, -370}
04867      , { 640, -380, -610, 640, -610}
04868      , { 880, -370, -370, 880, -370}
04869      , { -1140, -1250, -1310, -1140, -1310}
04870      }
04871      }
04872      , { { { 410, 230, 410, 40, 410}
04873      , { -30, -70, -240, -30, -240}
04874      , { 40, -550, -370, 40, -370}
04875      , { 410, 230, 410, -200, 410}
04876      , { -90, -680, -500, -90, -500}
04877      }
04878      , { { -30, -70, -240, -30, -240}
04879      , { -30, -70, -240, -30, -240}
04880      , { -350, -700, -520, -350, -520}
04881      , { -1250, -1600, -1420, -1250, -1420}
04882      , { -350, -700, -520, -350, -520}
04883      }
04884      , { { 40, -550, -370, 40, -370}
04885      , { -200, -550, -370, -200, -370}
04886      , { 40, -550, -370, 40, -370}
04887      , { -200, -550, -370, -200, -370}
04888      , { -90, -680, -500, -90, -500}
04889      }
04890      , { { 410, 230, 410, -350, 410}
04891      , { -1340, -1700, -1520, -1340, -1520}
04892      , { -350, -700, -520, -350, -520}
04893      , { 410, 230, 410, -670, 410}
04894      , { -350, -700, -520, -350, -520}
04895      }
04896      , { { -190, -550, -370, -190, -370}
04897      , { -200, -550, -370, -200, -370}
04898      , { -190, -790, -610, -190, -610}
04899      , { -200, -550, -370, -200, -370}
04900      , { -1140, -1250, -1310, -1140, -1310}
04901      }
04902      }
04903      , { { { 410, 410, 410, 410, 410}
04904      , { -240, -240, -240, -240, -240}
04905      , { -370, -370, -370, -370, -370}
04906      , { 410, 410, 410, 410, 410}
04907      , { -500, -500, -500, -500, -500}
04908      }
04909      , { { -240, -240, -240, -240, -240}
04910      , { -240, -240, -240, -240, -240}
04911      , { -520, -520, -520, -520, -520}
04912      , { -1180, -1420, -1180, -1420, -1180}
04913      , { -520, -520, -520, -520, -520}
04914      }
04915      , { { -370, -370, -370, -370, -370}
04916      , { -370, -370, -370, -370, -370}
04917      , { -370, -370, -370, -370, -370}
04918      , { -370, -370, -370, -370, -370}
04919      , { -500, -500, -500, -500, -500}
04920      }
04921      , { { 410, 410, 410, 410, 410}
04922      , { -1280, -1520, -1280, -1520, -1280}
04923      , { -520, -520, -520, -520, -520}
04924      , { 410, 410, 410, 410, 410}
04925      , { -520, -520, -520, -520, -520}
04926      }
04927      , { { -370, -370, -370, -370, -370}
04928      , { -370, -370, -370, -370, -370}
04929      , { -610, -610, -610, -610, -610}
04930      , { -370, -370, -370, -370, -370}
04931      , { -1310, -1310, -1310, -1310, -1310}
04932      }
04933      }
04934      , { { { 1010, -150, 410, 1010, 410}
04935      , { 1010, -260, -240, 1010, -240}
04936      , { 880, -150, -370, 880, -370}
04937      , { 880, -390, 410, 880, 410}
```

```
04938     , { 750, -280, -500, 750, -500}
04939     }
04940     , { { 1010, -260, -240, 1010, -240}
04941     , { 1010, -260, -240, 1010, -240}
04942     , { 730, -540, -520, 730, -520}
04943     , { -1420, -1440, -1420, -1420, -1420}
04944     , { 730, -540, -520, 730, -520}
04945     }
04946     , { { 880, -150, -370, 880, -370}
04947     , { 880, -390, -370, 880, -370}
04948     , { 880, -150, -370, 880, -370}
04949     , { 880, -390, -370, 880, -370}
04950     , { 750, -280, -500, 750, -500}
04951     }
04952     , { { 730, -540, 410, 730, 410}
04953     , { -1520, -1530, -1520, -1520, -1520}
04954     , { 730, -540, -520, 730, -520}
04955     , { 410, -860, 410, -840, 410}
04956     , { 730, -540, -520, 730, -520}
04957     }
04958     , { { 880, -380, -370, 880, -370}
04959     , { 880, -390, -370, 880, -370}
04960     , { 640, -380, -610, 640, -610}
04961     , { 880, -390, -370, 880, -370}
04962     , { -1310, -1330, -1310, -1310, -1310}
04963     }
04964     }
04965     , { { { 410, 410, 410, 410, 0}
04966     , { 0, -240, -240, -240, 0}
04967     , { -370, -370, -370, -370, -370}
04968     , { 410, 410, 410, 410, -370}
04969     , { -500, -500, -500, -500, -500}
04970     }
04971     , { { 0, -240, -240, -240, 0}
04972     , { 0, -240, -240, -240, 0}
04973     , { -520, -520, -520, -520, -520}
04974     , { -1180, -1420, -1180, -1420, -1420}
04975     , { -520, -520, -520, -520, -520}
04976     }
04977     , { { -370, -370, -370, -370, -370}
04978     , { -370, -370, -370, -370, -370}
04979     , { -370, -370, -370, -370, -370}
04980     , { -370, -370, -370, -370, -370}
04981     , { -500, -500, -500, -500, -500}
04982     }
04983     , { { 410, 410, 410, 410, -520}
04984     , { -1280, -1520, -1280, -1520, -1520}
04985     , { -520, -520, -520, -520, -520}
04986     , { 410, 410, 410, 410, -840}
04987     , { -520, -520, -520, -520, -520}
04988     }
04989     , { { -370, -370, -370, -370, -370}
04990     , { -370, -370, -370, -370, -370}
04991     , { -610, -610, -610, -610, -610}
04992     , { -370, -370, -370, -370, -370}
04993     , { -1310, -1310, -1310, -1310, -1310}
04994     }
04995     }
04996     }
04997     }
04998     , { { { { INF, INF, INF, INF, INF}
04999     , { INF, INF, INF, INF, INF}
05000     , { INF, INF, INF, INF, INF}
05001     , { INF, INF, INF, INF, INF}
05002     , { INF, INF, INF, INF, INF}
05003     }
05004     , { { INF, INF, INF, INF, INF}
05005     , { INF, INF, INF, INF, INF}
05006     , { INF, INF, INF, INF, INF}
05007     , { INF, INF, INF, INF, INF}
05008     , { INF, INF, INF, INF, INF}
05009     }
05010     , { { INF, INF, INF, INF, INF}
05011     , { INF, INF, INF, INF, INF}
05012     , { INF, INF, INF, INF, INF}
05013     , { INF, INF, INF, INF, INF}
05014     , { INF, INF, INF, INF, INF}
05015     }
05016     , { { INF, INF, INF, INF, INF}
05017     , { INF, INF, INF, INF, INF}
05018     , { INF, INF, INF, INF, INF}
05019     , { INF, INF, INF, INF, INF}
05020     , { INF, INF, INF, INF, INF}
05021     }
05022     , { { INF, INF, INF, INF, INF}
05023     , { INF, INF, INF, INF, INF}
05024     , { INF, INF, INF, INF, INF}
```

```
05025      , {   INF,   INF,   INF,   INF,   INF }
05026      , {   INF,   INF,   INF,   INF,   INF }
05027      }
05028    }
05029    , { {   INF,   INF,   INF,   INF,   INF }
05030      , {   INF,   INF,   INF,   INF,   INF }
05031      , {   INF,   INF,   INF,   INF,   INF }
05032      , {   INF,   INF,   INF,   INF,   INF }
05033      , {   INF,   INF,   INF,   INF,   INF }
05034      }
05035    , { {   INF,   INF,   INF,   INF,   INF }
05036      , {   INF,   INF,   INF,   INF,   INF }
05037      , {   INF,   INF,   INF,   INF,   INF }
05038      , {   INF,   INF,   INF,   INF,   INF }
05039      , {   INF,   INF,   INF,   INF,   INF }
05040      }
05041    , { {   INF,   INF,   INF,   INF,   INF }
05042      , {   INF,   INF,   INF,   INF,   INF }
05043      , {   INF,   INF,   INF,   INF,   INF }
05044      , {   INF,   INF,   INF,   INF,   INF }
05045      , {   INF,   INF,   INF,   INF,   INF }
05046      }
05047    , { {   INF,   INF,   INF,   INF,   INF }
05048      , {   INF,   INF,   INF,   INF,   INF }
05049      , {   INF,   INF,   INF,   INF,   INF }
05050      , {   INF,   INF,   INF,   INF,   INF }
05051      , {   INF,   INF,   INF,   INF,   INF }
05052      }
05053    , { {   INF,   INF,   INF,   INF,   INF }
05054      , {   INF,   INF,   INF,   INF,   INF }
05055      , {   INF,   INF,   INF,   INF,   INF }
05056      , {   INF,   INF,   INF,   INF,   INF }
05057      , {   INF,   INF,   INF,   INF,   INF }
05058      }
05059    }
05060    , { { {   INF,   INF,   INF,   INF,   INF }
05061      , {   INF,   INF,   INF,   INF,   INF }
05062      , {   INF,   INF,   INF,   INF,   INF }
05063      , {   INF,   INF,   INF,   INF,   INF }
05064      , {   INF,   INF,   INF,   INF,   INF }
05065      }
05066    , { { {   INF,   INF,   INF,   INF,   INF }
05067      , {   INF,   INF,   INF,   INF,   INF }
05068      , {   INF,   INF,   INF,   INF,   INF }
05069      , {   INF,   INF,   INF,   INF,   INF }
05070      , {   INF,   INF,   INF,   INF,   INF }
05071      }
05072    , { { {   INF,   INF,   INF,   INF,   INF }
05073      , {   INF,   INF,   INF,   INF,   INF }
05074      , {   INF,   INF,   INF,   INF,   INF }
05075      , {   INF,   INF,   INF,   INF,   INF }
05076      , {   INF,   INF,   INF,   INF,   INF }
05077      }
05078    , { { {   INF,   INF,   INF,   INF,   INF }
05079      , {   INF,   INF,   INF,   INF,   INF }
05080      , {   INF,   INF,   INF,   INF,   INF }
05081      , {   INF,   INF,   INF,   INF,   INF }
05082      , {   INF,   INF,   INF,   INF,   INF }
05083      }
05084    , { { {   INF,   INF,   INF,   INF,   INF }
05085      , {   INF,   INF,   INF,   INF,   INF }
05086      , {   INF,   INF,   INF,   INF,   INF }
05087      , {   INF,   INF,   INF,   INF,   INF }
05088      , {   INF,   INF,   INF,   INF,   INF }
05089      }
05090    }
05091    , { { {   INF,   INF,   INF,   INF,   INF }
05092      , {   INF,   INF,   INF,   INF,   INF }
05093      , {   INF,   INF,   INF,   INF,   INF }
05094      , {   INF,   INF,   INF,   INF,   INF }
05095      , {   INF,   INF,   INF,   INF,   INF }
05096      }
05097    , { {   INF,   INF,   INF,   INF,   INF }
05098      , {   INF,   INF,   INF,   INF,   INF }
05099      , {   INF,   INF,   INF,   INF,   INF }
05100      , {   INF,   INF,   INF,   INF,   INF }
05101      , {   INF,   INF,   INF,   INF,   INF }
05102      }
05103    , { {   INF,   INF,   INF,   INF,   INF }
05104      , {   INF,   INF,   INF,   INF,   INF }
05105      , {   INF,   INF,   INF,   INF,   INF }
05106      , {   INF,   INF,   INF,   INF,   INF }
05107      , {   INF,   INF,   INF,   INF,   INF }
05108      }
05109    , { {   INF,   INF,   INF,   INF,   INF }
05110      , {   INF,   INF,   INF,   INF,   INF }
05111      , {   INF,   INF,   INF,   INF,   INF }
```

```

05112     , {   INF,   INF,   INF,   INF,   INF }
05113     , {   INF,   INF,   INF,   INF,   INF }
05114     }
05115     , { {   INF,   INF,   INF,   INF,   INF }
05116     , {   INF,   INF,   INF,   INF,   INF }
05117     , {   INF,   INF,   INF,   INF,   INF }
05118     , {   INF,   INF,   INF,   INF,   INF }
05119     , {   INF,   INF,   INF,   INF,   INF }
05120     }
05121     }
05122     , { { {   INF,   INF,   INF,   INF,   INF }
05123     , {   INF,   INF,   INF,   INF,   INF }
05124     , {   INF,   INF,   INF,   INF,   INF }
05125     , {   INF,   INF,   INF,   INF,   INF }
05126     , {   INF,   INF,   INF,   INF,   INF }
05127     }
05128     , { {   INF,   INF,   INF,   INF,   INF }
05129     , {   INF,   INF,   INF,   INF,   INF }
05130     , {   INF,   INF,   INF,   INF,   INF }
05131     , {   INF,   INF,   INF,   INF,   INF }
05132     , {   INF,   INF,   INF,   INF,   INF }
05133     }
05134     , { {   INF,   INF,   INF,   INF,   INF }
05135     , {   INF,   INF,   INF,   INF,   INF }
05136     , {   INF,   INF,   INF,   INF,   INF }
05137     , {   INF,   INF,   INF,   INF,   INF }
05138     , {   INF,   INF,   INF,   INF,   INF }
05139     }
05140     , { {   INF,   INF,   INF,   INF,   INF }
05141     , {   INF,   INF,   INF,   INF,   INF }
05142     , {   INF,   INF,   INF,   INF,   INF }
05143     , {   INF,   INF,   INF,   INF,   INF }
05144     , {   INF,   INF,   INF,   INF,   INF }
05145     }
05146     , { {   INF,   INF,   INF,   INF,   INF }
05147     , {   INF,   INF,   INF,   INF,   INF }
05148     , {   INF,   INF,   INF,   INF,   INF }
05149     , {   INF,   INF,   INF,   INF,   INF }
05150     , {   INF,   INF,   INF,   INF,   INF }
05151     }
05152     }
05153     }
05154     , { { { {   800,    200,   -310,    800,   -310 }
05155     , {   740,      0,   -510,    740,   -410 }
05156     , {   800,     50,   -450,    800,   -450 }
05157     , {   740,    200,   -310,    740,   -310 }
05158     , {   690,    -50,   -560,    690,   -560 }
05159     }
05160     , { {   600,   -140,   -630,    600,   -410 }
05161     , {   600,   -140,   -650,    600,   -410 }
05162     , {   290,   -450,   -960,    290,   -960 }
05163     , {  -360,   -360,   -630,   -870,   -630 }
05164     , {   290,   -450,   -960,    290,   -960 }
05165     }
05166     , { {   740,      0,   -510,    740,   -510 }
05167     , {   740,      0,   -510,    740,   -510 }
05168     , {   740,      0,   -510,    740,   -510 }
05169     , {   740,      0,   -510,    740,   -510 }
05170     , {   690,    -50,   -560,    690,   -560 }
05171     }
05172     , { {   290,    200,   -310,    290,   -310 }
05173     , {  -640,   -640,   -910,  -1150,   -910 }
05174     , {   290,   -450,   -960,    290,   -960 }
05175     , {   200,    200,   -310,   -310,   -310 }
05176     , {   290,   -450,   -960,    290,   -960 }
05177     }
05178     , { {   800,     50,   -450,    800,   -450 }
05179     , {   740,      0,   -510,    740,   -510 }
05180     , {   800,     50,   -450,    800,   -450 }
05181     , {   740,      0,   -510,    740,   -510 }
05182     , {  -550,   -550,  -1300,  -1300,  -1300 }
05183     }
05184     }
05185     , { { {   200,    200,   -310,   -720,   -310 }
05186     , {      0,      0,   -510,  -1020,   -510 }
05187     , {     50,     50,   -450,   -720,   -450 }
05188     , {   200,    200,   -310,  -1020,   -310 }
05189     , {    -50,    -50,   -560,   -830,   -560 }
05190     }
05191     , { {  -140,   -140,   -650,  -1160,   -650 }
05192     , {  -140,   -140,   -650,  -1160,   -650 }
05193     , {  -450,   -450,   -960,  -1470,   -960 }
05194     , {  -360,   -360,   -870,  -1380,   -870 }
05195     , {  -450,   -450,   -960,  -1470,   -960 }
05196     }
05197     , { {      0,      0,   -510,   -780,   -510 }
05198     , {      0,      0,   -510,  -1020,   -510 }

```

```

05199     , {      0,      0, -510, -780, -510}
05200     , {      0,      0, -510, -1020, -510}
05201     , {    -50,    -50, -560, -830, -560}
05202     }
05203     , {{    200,    200, -310, -1470, -310}
05204     , {   -640,   -640, -1150, -1660, -1150}
05205     , {   -450,   -450, -960, -1470, -960}
05206     , {    200,    200, -310, -2070, -310}
05207     , {   -450,   -450, -960, -1470, -960}
05208     }
05209     , {{     50,     50, -450, -720, -450}
05210     , {      0,      0, -510, -1020, -510}
05211     , {     50,     50, -450, -720, -450}
05212     , {      0,      0, -510, -1020, -510}
05213     , {   -550,   -550, -1300, -1810, -1300}
05214     }
05215     }
05216     , {{{ -310, -310, -310, -310, -310}
05217     , {   -510,   -510, -510, -510, -510}
05218     , {   -450,   -450, -450, -450, -450}
05219     , {   -310,   -310, -310, -310, -310}
05220     , {   -560,   -560, -560, -560, -560}
05221     }
05222     , {{ -630, -650, -630, -650, -630}
05223     , {   -650,   -650, -650, -650, -650}
05224     , {   -960,   -960, -960, -960, -960}
05225     , {   -630,   -870, -630, -870, -630}
05226     , {   -960,   -960, -960, -960, -960}
05227     }
05228     , {{{ -510, -510, -510, -510, -510}
05229     , {   -510,   -510, -510, -510, -510}
05230     , {   -510,   -510, -510, -510, -510}
05231     , {   -510,   -510, -510, -510, -510}
05232     , {   -560,   -560, -560, -560, -560}
05233     }
05234     , {{{ -310, -310, -310, -310, -310}
05235     , {   -910, -1150, -910, -1150, -910}
05236     , {   -960,   -960, -960, -960, -960}
05237     , {   -310,   -310, -310, -310, -310}
05238     , {   -960,   -960, -960, -960, -960}
05239     }
05240     , {{{ -450, -450, -450, -450, -450}
05241     , {   -510,   -510, -510, -510, -510}
05242     , {   -450,   -450, -450, -450, -450}
05243     , {   -510,   -510, -510, -510, -510}
05244     , { -1300, -1300, -1300, -1300, -1300}
05245     }
05246     }
05247     , {{{  800,   -550, -310,  800, -310}
05248     , {   740,   -850, -510,  740, -510}
05249     , {   800,   -550, -450,  800, -450}
05250     , {   740,   -850, -310,  740, -310}
05251     , {   690,   -660, -560,  690, -560}
05252     }
05253     , {{  600,   -990, -650,  600, -650}
05254     , {  600,   -990, -650,  600, -650}
05255     , {   290, -1300, -960,  290, -960}
05256     , {  -870, -1210, -870, -870, -870}
05257     , {   290, -1300, -960,  290, -960}
05258     }
05259     , {{  740,   -610, -510,  740, -510}
05260     , {  740,   -850, -510,  740, -510}
05261     , {  740,   -610, -510,  740, -510}
05262     , {  740,   -850, -510,  740, -510}
05263     , {   690,   -660, -560,  690, -560}
05264     }
05265     , {{  290, -1300, -310,  290, -310}
05266     , { -1150, -1490, -1150, -1150, -1150}
05267     , {  290, -1300, -960,  290, -960}
05268     , {  -310, -1900, -310, -1560, -310}
05269     , {  290, -1300, -960,  290, -960}
05270     }
05271     , {{  800,   -550, -450,  800, -450}
05272     , {  740,   -850, -510,  740, -510}
05273     , {  800,   -550, -450,  800, -450}
05274     , {  740,   -850, -510,  740, -510}
05275     , { -1300, -1640, -1300, -1300, -1300}
05276     }
05277     }
05278     , {{{ -310, -310, -310, -310, -410}
05279     , {  -410,   -510, -510, -510, -410}
05280     , {  -450,   -450, -450, -450, -450}
05281     , {  -310,   -310, -310, -310, -510}
05282     , {  -560,   -560, -560, -560, -560}
05283     }
05284     , {{{ -410,   -650, -630,   -650, -410}
05285     , {  -410,   -650, -650,   -650, -410}

```



```
05286     , { -960, -960, -960, -960, -960 }
05287     , { -630, -870, -630, -870, -870 }
05288     , { -960, -960, -960, -960, -960 }
05289     }
05290     , { { -510, -510, -510, -510, -510 }
05291     , { -510, -510, -510, -510, -510 }
05292     , { -510, -510, -510, -510, -510 }
05293     , { -510, -510, -510, -510, -510 }
05294     , { -560, -560, -560, -560, -560 }
05295     }
05296     , { { -310, -310, -310, -310, -960 }
05297     , { -910, -1150, -910, -1150, -1150 }
05298     , { -960, -960, -960, -960, -960 }
05299     , { -310, -310, -310, -310, -1560 }
05300     , { -960, -960, -960, -960, -960 }
05301     }
05302     , { { -450, -450, -450, -450, -450 }
05303     , { -510, -510, -510, -510, -510 }
05304     , { -450, -450, -450, -450, -450 }
05305     , { -510, -510, -510, -510, -510 }
05306     , { -1300, -1300, -1300, -1300, -1300 }
05307     }
05308     }
05309     }
05310     , { { { 760, 200, -310, 760, -250 }
05311     , { 760, -340, -490, 760, -250 }
05312     , { 310, -430, -940, 310, -940 }
05313     , { 400, 200, -310, 400, -310 }
05314     , { 310, -390, -940, 310, -940 }
05315     }
05316     , { { 760, -430, -490, 760, -250 }
05317     , { 760, -490, -490, 760, -250 }
05318     , { 310, -430, -940, 310, -940 }
05319     , { -1170, -1170, -1440, -1680, -1440 }
05320     , { 310, -430, -940, 310, -940 }
05321     }
05322     , { { 400, -340, -850, 400, -850 }
05323     , { 400, -340, -850, 400, -850 }
05324     , { 90, -650, -1160, 90, -1160 }
05325     , { 400, -340, -850, 400, -850 }
05326     , { 90, -650, -1160, 90, -1160 }
05327     }
05328     , { { 310, 200, -310, 310, -310 }
05329     , { -690, -690, -960, -1200, -960 }
05330     , { 310, -430, -940, 310, -940 }
05331     , { 200, 200, -310, -310, -310 }
05332     , { 310, -430, -940, 310, -940 }
05333     }
05334     , { { 400, -340, -850, 400, -850 }
05335     , { 400, -340, -850, 400, -850 }
05336     , { -220, -960, -1470, -220, -1470 }
05337     , { 400, -340, -850, 400, -850 }
05338     , { -390, -390, -1140, -1140, -1140 }
05339     }
05340     }
05341     , { { { 200, 200, -310, -1000, -310 }
05342     , { -340, -340, -490, -1000, -490 }
05343     , { -430, -430, -940, -1430, -940 }
05344     , { 200, 200, -310, -1360, -310 }
05345     , { -390, -390, -940, -1430, -940 }
05346     }
05347     , { { -430, -430, -490, -1000, -490 }
05348     , { -490, -2040, -490, -1000, -490 }
05349     , { -430, -430, -940, -1450, -940 }
05350     , { -1170, -1170, -1680, -2190, -1680 }
05351     , { -430, -430, -940, -1450, -940 }
05352     }
05353     , { { -340, -340, -850, -1360, -850 }
05354     , { -340, -340, -850, -1360, -850 }
05355     , { -650, -650, -1160, -1430, -1160 }
05356     , { -340, -340, -850, -1360, -850 }
05357     , { -650, -650, -1160, -1430, -1160 }
05358     }
05359     , { { 200, 200, -310, -1450, -310 }
05360     , { -690, -690, -1200, -1710, -1200 }
05361     , { -430, -430, -940, -1450, -940 }
05362     , { 200, 200, -310, -2070, -310 }
05363     , { -430, -430, -940, -1450, -940 }
05364     }
05365     , { { -340, -340, -850, -1360, -850 }
05366     , { -340, -340, -850, -1360, -850 }
05367     , { -960, -960, -1470, -1740, -1470 }
05368     , { -340, -340, -850, -1360, -850 }
05369     , { -390, -390, -1140, -1650, -1140 }
05370     }
05371     }
05372     , { { { -310, -310, -310, -310, -310 }
```

```
05373 , { -490, -490, -490, -490, -490 }
05374 , { -940, -940, -940, -940, -940 }
05375 , { -310, -310, -310, -310, -310 }
05376 , { -940, -940, -940, -940, -940 }
05377 }
05378 , { { -490, -490, -490, -490, -490 }
05379 , { -490, -490, -490, -490, -490 }
05380 , { -940, -940, -940, -940, -940 }
05381 , { -1440, -1680, -1440, -1680, -1440 }
05382 , { -940, -940, -940, -940, -940 }
05383 }
05384 , { { -850, -850, -850, -850, -850 }
05385 , { -850, -850, -850, -850, -850 }
05386 , { -1160, -1160, -1160, -1160, -1160 }
05387 , { -850, -850, -850, -850, -850 }
05388 , { -1160, -1160, -1160, -1160, -1160 }
05389 }
05390 , { { -310, -310, -310, -310, -310 }
05391 , { -960, -1200, -960, -1200, -960 }
05392 , { -940, -940, -940, -940, -940 }
05393 , { -310, -310, -310, -310, -310 }
05394 , { -940, -940, -940, -940, -940 }
05395 }
05396 , { { -850, -850, -850, -850, -850 }
05397 , { -850, -850, -850, -850, -850 }
05398 , { -1470, -1470, -1470, -1470, -1470 }
05399 , { -850, -850, -850, -850, -850 }
05400 , { -1140, -1140, -1140, -1140, -1140 }
05401 }
05402 }
05403 , { { { 760, -830, -310, 760, -310 }
05404 , { 760, -830, -490, 760, -490 }
05405 , { 310, -1260, -940, 310, -940 }
05406 , { 400, -1190, -310, 400, -310 }
05407 , { 310, -1260, -940, 310, -940 }
05408 }
05409 , { { 760, -830, -490, 760, -490 }
05410 , { 760, -830, -490, 760, -490 }
05411 , { 310, -1280, -940, 310, -940 }
05412 , { -1680, -2020, -1680, -1680, -1680 }
05413 , { 310, -1280, -940, 310, -940 }
05414 }
05415 , { { 400, -1190, -850, 400, -850 }
05416 , { 400, -1190, -850, 400, -850 }
05417 , { 90, -1260, -1160, 90, -1160 }
05418 , { 400, -1190, -850, 400, -850 }
05419 , { 90, -1260, -1160, 90, -1160 }
05420 }
05421 , { { 310, -1280, -310, 310, -310 }
05422 , { -1200, -1540, -1200, -1200, -1200 }
05423 , { 310, -1280, -940, 310, -940 }
05424 , { -310, -1900, -310, -1560, -310 }
05425 , { 310, -1280, -940, 310, -940 }
05426 }
05427 , { { 400, -1190, -850, 400, -850 }
05428 , { 400, -1190, -850, 400, -850 }
05429 , { -220, -1570, -1470, -220, -1470 }
05430 , { 400, -1190, -850, 400, -850 }
05431 , { -1140, -1480, -1140, -1140, -1140 }
05432 }
05433 }
05434 , { { { -250, -310, -310, -310, -250 }
05435 , { -250, -490, -490, -490, -250 }
05436 , { -940, -940, -940, -940, -940 }
05437 , { -310, -310, -310, -310, -850 }
05438 , { -940, -940, -940, -940, -940 }
05439 }
05440 , { { -250, -490, -490, -490, -250 }
05441 , { -250, -490, -490, -490, -250 }
05442 , { -940, -940, -940, -940, -940 }
05443 , { -1440, -1680, -1440, -1680, -1680 }
05444 , { -940, -940, -940, -940, -940 }
05445 }
05446 , { { -850, -850, -850, -850, -850 }
05447 , { -850, -850, -850, -850, -850 }
05448 , { -1160, -1160, -1160, -1160, -1160 }
05449 , { -850, -850, -850, -850, -850 }
05450 , { -1160, -1160, -1160, -1160, -1160 }
05451 }
05452 , { { -310, -310, -310, -310, -940 }
05453 , { -960, -1200, -960, -1200, -1200 }
05454 , { -940, -940, -940, -940, -940 }
05455 , { -310, -310, -310, -310, -1560 }
05456 , { -940, -940, -940, -940, -940 }
05457 }
05458 , { { -850, -850, -850, -850, -850 }
05459 , { -850, -850, -850, -850, -850 }
```

```
05460     , { -1470, -1470, -1470, -1470, -1470}
05461     , {  -850,  -850,  -850,  -850,  -850}
05462     , { -1140, -1140, -1140, -1140, -1140}
05463     }
05464     }
05465     }
05466     , {{{ 360, 360, -150, -150, -150}
05467     , {  -30,  -30, -990, -150, -990}
05468     , { -150, -890, -1400, -150, -1400}
05469     , { 360, 360, -150, -150, -150}
05470     , { -150, -650, -1400, -150, -1400}
05471     }
05472     , {{ -70, -70, -1180, -150, -1180}
05473     , {  -70,  -70, -1580, -330, -1340}
05474     , { -150, -890, -1400, -150, -1400}
05475     , { -910, -910, -1180, -1420, -1180}
05476     , { -150, -890, -1400, -150, -1400}
05477     }
05478     , {{ -150, -890, -1400, -150, -1400}
05479     , { -150, -890, -1400, -150, -1400}
05480     , { -150, -890, -1400, -150, -1400}
05481     , { -150, -890, -1400, -150, -1400}
05482     , { -150, -890, -1400, -150, -1400}
05483     }
05484     , {{ 360, 360, -150, -150, -150}
05485     , {  -30,  -30, -990, -1230, -990}
05486     , { -150, -890, -1400, -150, -1400}
05487     , { 360, 360, -150, -150, -150}
05488     , { -150, -890, -1400, -150, -1400}
05489     }
05490     , {{ -150, -650, -1400, -150, -1400}
05491     , { -150, -890, -1400, -150, -1400}
05492     , { -150, -890, -1400, -150, -1400}
05493     , { -150, -890, -1400, -150, -1400}
05494     , { -650, -650, -1400, -1400, -1400}
05495     }
05496     }
05497     , {{{ 360, 360, -150, -1670, -150}
05498     , {  -30,  -30, -1230, -1740, -1230}
05499     , { -890, -890, -1400, -1670, -1400}
05500     , { 360, 360, -150, -1910, -150}
05501     , { -650, -650, -1400, -1670, -1400}
05502     }
05503     , {{ -70, -70, -1400, -1910, -1400}
05504     , {  -70,  -70, -1580, -2090, -1580}
05505     , { -890, -890, -1400, -1910, -1400}
05506     , { -910, -910, -1420, -1930, -1420}
05507     , { -890, -890, -1400, -1910, -1400}
05508     }
05509     , {{ -890, -890, -1400, -1670, -1400}
05510     , { -890, -890, -1400, -1910, -1400}
05511     , { -890, -890, -1400, -1670, -1400}
05512     , { -890, -890, -1400, -1910, -1400}
05513     , { -890, -890, -1400, -1670, -1400}
05514     }
05515     , {{{ 360, 360, -150, -1740, -150}
05516     , {  -30,  -30, -1230, -1740, -1230}
05517     , { -890, -890, -1400, -1910, -1400}
05518     , { 360, 360, -150, -1910, -150}
05519     , { -890, -890, -1400, -1910, -1400}
05520     }
05521     , {{{ -650, -650, -1400, -1670, -1400}
05522     , { -890, -890, -1400, -1910, -1400}
05523     , { -890, -890, -1400, -1670, -1400}
05524     , { -890, -890, -1400, -1910, -1400}
05525     , { -650, -650, -1400, -1910, -1400}
05526     }
05527     }
05528     , {{{ -150, -150, -150, -150, -150}
05529     , { -990, -1230, -990, -1230, -990}
05530     , { -1400, -1400, -1400, -1400, -1400}
05531     , { -150, -150, -150, -150, -150}
05532     , { -1400, -1400, -1400, -1400, -1400}
05533     }
05534     , {{ -1180, -1400, -1180, -1400, -1180}
05535     , { -1580, -1580, -1580, -1580, -1580}
05536     , { -1400, -1400, -1400, -1400, -1400}
05537     , { -1180, -1420, -1180, -1420, -1180}
05538     , { -1400, -1400, -1400, -1400, -1400}
05539     }
05540     , {{ -1400, -1400, -1400, -1400, -1400}
05541     , { -1400, -1400, -1400, -1400, -1400}
05542     , { -1400, -1400, -1400, -1400, -1400}
05543     , { -1400, -1400, -1400, -1400, -1400}
05544     , { -1400, -1400, -1400, -1400, -1400}
05545     }
05546     , {{ -150, -150, -150, -150, -150}
```

```
05547 , { -990, -1230, -990, -1230, -990}
05548 , { -1400, -1400, -1400, -1400, -1400}
05549 , { -150, -150, -150, -150, -150}
05550 , { -1400, -1400, -1400, -1400, -1400}
05551 }
05552 , { { -1400, -1400, -1400, -1400, -1400}
05553 , { -1400, -1400, -1400, -1400, -1400}
05554 , { -1400, -1400, -1400, -1400, -1400}
05555 , { -1400, -1400, -1400, -1400, -1400}
05556 , { -1400, -1400, -1400, -1400, -1400}
05557 }
05558 }
05559 , { { { -150, -1500, -150, -150, -150}
05560 , { -150, -1570, -1230, -150, -1230}
05561 , { -150, -1500, -1400, -150, -1400}
05562 , { -150, -1740, -150, -150, -150}
05563 , { -150, -1500, -1400, -150, -1400}
05564 }
05565 , { { -150, -1600, -1400, -150, -1400}
05566 , { -330, -1600, -1580, -330, -1580}
05567 , { -150, -1740, -1400, -150, -1400}
05568 , { -1420, -3040, -1420, -1420, -1420}
05569 , { -150, -1740, -1400, -150, -1400}
05570 }
05571 , { { -150, -1500, -1400, -150, -1400}
05572 , { -150, -1740, -1400, -150, -1400}
05573 , { -150, -1500, -1400, -150, -1400}
05574 , { -150, -1740, -1400, -150, -1400}
05575 , { -150, -1500, -1400, -150, -1400}
05576 }
05577 , { { -150, -1570, -150, -150, -150}
05578 , { -1230, -1570, -1230, -1230, -1230}
05579 , { -150, -1740, -1400, -150, -1400}
05580 , { -150, -1740, -150, -1400, -150}
05581 , { -150, -1740, -1400, -150, -1400}
05582 }
05583 , { { -150, -1500, -1400, -150, -1400}
05584 , { -150, -1740, -1400, -150, -1400}
05585 , { -150, -1500, -1400, -150, -1400}
05586 , { -150, -1740, -1400, -150, -1400}
05587 , { -1400, -1740, -1400, -1400, -1400}
05588 }
05589 }
05590 , { { { -150, -150, -150, -150, -1230}
05591 , { -990, -1230, -990, -1230, -1230}
05592 , { -1400, -1400, -1400, -1400, -1400}
05593 , { -150, -150, -150, -150, -1400}
05594 , { -1400, -1400, -1400, -1400, -1400}
05595 }
05596 , { { -1180, -1400, -1180, -1400, -1340}
05597 , { -1340, -1580, -1580, -1580, -1340}
05598 , { -1400, -1400, -1400, -1400, -1400}
05599 , { -1180, -1420, -1180, -1420, -1420}
05600 , { -1400, -1400, -1400, -1400, -1400}
05601 }
05602 , { { -1400, -1400, -1400, -1400, -1400}
05603 , { -1400, -1400, -1400, -1400, -1400}
05604 , { -1400, -1400, -1400, -1400, -1400}
05605 , { -1400, -1400, -1400, -1400, -1400}
05606 , { -1400, -1400, -1400, -1400, -1400}
05607 }
05608 , { { -150, -150, -150, -150, -1230}
05609 , { -990, -1230, -990, -1230, -1230}
05610 , { -1400, -1400, -1400, -1400, -1400}
05611 , { -150, -150, -150, -150, -1400}
05612 , { -1400, -1400, -1400, -1400, -1400}
05613 }
05614 , { { -1400, -1400, -1400, -1400, -1400}
05615 , { -1400, -1400, -1400, -1400, -1400}
05616 , { -1400, -1400, -1400, -1400, -1400}
05617 , { -1400, -1400, -1400, -1400, -1400}
05618 , { -1400, -1400, -1400, -1400, -1400}
05619 }
05620 }
05621 }
05622 , { { { { 910, 910, 400, 910, 400}
05623 , { 910, 170, -340, 910, -100}
05624 , { 400, -340, -850, 400, -850}
05625 , { 910, 910, 400, 400, 400}
05626 , { 400, -100, -850, 400, -850}
05627 }
05628 , { { 910, 170, -340, 910, -100}
05629 , { 910, 170, -340, 910, -100}
05630 , { 400, -340, -850, 400, -850}
05631 , { -680, -680, -950, -1190, -950}
05632 , { 400, -340, -850, 400, -850}
05633 }
```

```
05634 ,{{ 400, -340, -850, 400, -850}
05635 ,{ 400, -340, -850, 400, -850}
05636 ,{ 400, -340, -850, 400, -850}
05637 ,{ 400, -340, -850, 400, -850}
05638 ,{ 400, -340, -850, 400, -850}
05639 }
05640 ,{{ 910, 910, 400, 400, 400}
05641 ,{ -850, -850, -1120, -1360, -1120}
05642 ,{ 400, -340, -850, 400, -850}
05643 ,{ 910, 910, 400, 400, 400}
05644 ,{ 400, -340, -850, 400, -850}
05645 }
05646 ,{{ 400, -100, -850, 400, -850}
05647 ,{ 400, -340, -850, 400, -850}
05648 ,{ 400, -340, -850, 400, -850}
05649 ,{ 400, -340, -850, 400, -850}
05650 ,{ -100, -100, -850, -850, -850}
05651 }
05652 }
05653 ,{{{ 910, 910, 400, -850, 400}
05654 ,{ 170, 170, -340, -850, -340}
05655 ,{ -340, -340, -850, -1120, -850}
05656 ,{ 910, 910, 400, -1360, 400}
05657 ,{ -100, -100, -850, -1120, -850}
05658 }
05659 ,{{ 170, 170, -340, -850, -340}
05660 ,{ 170, 170, -340, -850, -340}
05661 ,{ -340, -340, -850, -1360, -850}
05662 ,{ -680, -680, -1190, -1700, -1190}
05663 ,{ -340, -340, -850, -1360, -850}
05664 }
05665 ,{{ -340, -340, -850, -1120, -850}
05666 ,{ -340, -340, -850, -1360, -850}
05667 ,{ -340, -340, -850, -1120, -850}
05668 ,{ -340, -340, -850, -1360, -850}
05669 ,{ -340, -340, -850, -1120, -850}
05670 }
05671 ,{{{ 910, 910, 400, -1360, 400}
05672 ,{ -850, -850, -1360, -1870, -1360}
05673 ,{ -340, -340, -850, -1360, -850}
05674 ,{ 910, 910, 400, -1360, 400}
05675 ,{ -340, -340, -850, -1360, -850}
05676 }
05677 ,{{{ -100, -100, -850, -1120, -850}
05678 ,{ -340, -340, -850, -1360, -850}
05679 ,{ -340, -340, -850, -1120, -850}
05680 ,{ -340, -340, -850, -1360, -850}
05681 ,{ -100, -100, -850, -1360, -850}
05682 }
05683 }
05684 ,{{{ 400, 400, 400, 400, 400}
05685 ,{ -340, -340, -340, -340, -340}
05686 ,{ -850, -850, -850, -850, -850}
05687 ,{ 400, 400, 400, 400, 400}
05688 ,{ -850, -850, -850, -850, -850}
05689 }
05690 ,{{{ -340, -340, -340, -340, -340}
05691 ,{ -340, -340, -340, -340, -340}
05692 ,{ -850, -850, -850, -850, -850}
05693 ,{ -950, -1190, -950, -1190, -950}
05694 ,{ -850, -850, -850, -850, -850}
05695 }
05696 ,{{{ -850, -850, -850, -850, -850}
05697 ,{ -850, -850, -850, -850, -850}
05698 ,{ -850, -850, -850, -850, -850}
05699 ,{ -850, -850, -850, -850, -850}
05700 ,{ -850, -850, -850, -850, -850}
05701 }
05702 ,{{{ 400, 400, 400, 400, 400}
05703 ,{ -1120, -1360, -1120, -1360, -1120}
05704 ,{ -850, -850, -850, -850, -850}
05705 ,{ 400, 400, 400, 400, 400}
05706 ,{ -850, -850, -850, -850, -850}
05707 }
05708 ,{{{ -850, -850, -850, -850, -850}
05709 ,{ -850, -850, -850, -850, -850}
05710 ,{ -850, -850, -850, -850, -850}
05711 ,{ -850, -850, -850, -850, -850}
05712 ,{ -850, -850, -850, -850, -850}
05713 }
05714 }
05715 ,{{{ 910, -680, 400, 910, 400}
05716 ,{ 910, -680, -340, 910, -340}
05717 ,{ 400, -950, -850, 400, -850}
05718 ,{ 400, -1190, 400, 400, 400}
05719 ,{ 400, -950, -850, 400, -850}
05720 }
```

```
05721 ,{{ 910, -680, -340, 910, -340}
05722 ,{ 910, -680, -340, 910, -340}
05723 ,{ 400, -1190, -850, 400, -850}
05724 ,{ -1190, -1530, -1190, -1190, -1190}
05725 ,{ 400, -1190, -850, 400, -850}
05726 }
05727 ,{{ 400, -950, -850, 400, -850}
05728 ,{ 400, -1190, -850, 400, -850}
05729 ,{ 400, -950, -850, 400, -850}
05730 ,{ 400, -1190, -850, 400, -850}
05731 ,{ 400, -950, -850, 400, -850}
05732 }
05733 ,{{ 400, -1190, 400, 400, 400}
05734 ,{ -1360, -1700, -1360, -1360, -1360}
05735 ,{ 400, -1190, -850, 400, -850}
05736 ,{ 400, -1190, 400, -850, 400}
05737 ,{ 400, -1190, -850, 400, -850}
05738 }
05739 ,{{ 400, -950, -850, 400, -850}
05740 ,{ 400, -1190, -850, 400, -850}
05741 ,{ 400, -950, -850, 400, -850}
05742 ,{ 400, -1190, -850, 400, -850}
05743 ,{ -850, -1190, -850, -850, -850}
05744 }
05745 }
05746 ,{{{ 400, 400, 400, 400, -100}
05747 ,{ -100, -340, -340, -340, -100}
05748 ,{ -850, -850, -850, -850, -850}
05749 ,{ 400, 400, 400, 400, -850}
05750 ,{ -850, -850, -850, -850, -850}
05751 }
05752 ,{{ -100, -340, -340, -340, -100}
05753 ,{ -100, -340, -340, -340, -100}
05754 ,{ -850, -850, -850, -850, -850}
05755 ,{ -950, -1190, -950, -1190, -1190}
05756 ,{ -850, -850, -850, -850, -850}
05757 }
05758 ,{{ -850, -850, -850, -850, -850}
05759 ,{ -850, -850, -850, -850, -850}
05760 ,{ -850, -850, -850, -850, -850}
05761 ,{ -850, -850, -850, -850, -850}
05762 ,{ -850, -850, -850, -850, -850}
05763 }
05764 ,{{{ 400, 400, 400, 400, -850}
05765 ,{ -1120, -1360, -1120, -1360, -1360}
05766 ,{ -850, -850, -850, -850, -850}
05767 ,{ 400, 400, 400, 400, -850}
05768 ,{ -850, -850, -850, -850, -850}
05769 }
05770 ,{{ -850, -850, -850, -850, -850}
05771 ,{ -850, -850, -850, -850, -850}
05772 ,{ -850, -850, -850, -850, -850}
05773 ,{ -850, -850, -850, -850, -850}
05774 ,{ -850, -850, -850, -850, -850}
05775 }
05776 }
05777 }
05778 ,{{{ 1490, 1280, 780, 1490, 780}
05779 ,{ 1490, 750, 240, 1490, 480}
05780 ,{ 1200, 450, -50, 1200, -50}
05781 ,{ 1280, 1280, 780, 1200, 780}
05782 ,{ 1200, 450, -50, 1200, -50}
05783 }
05784 ,{{ 1490, 750, 240, 1490, 480}
05785 ,{ 1490, 750, 240, 1490, 480}
05786 ,{ 1190, 440, -60, 1190, -60}
05787 ,{ -630, -630, -900, -1140, -900}
05788 ,{ 1190, 440, -60, 1190, -60}
05789 }
05790 ,{{ 1200, 460, -50, 1200, -50}
05791 ,{ 1200, 460, -50, 1200, -50}
05792 ,{ 1200, 450, -50, 1200, -50}
05793 ,{ 1200, 460, -50, 1200, -50}
05794 ,{ 1200, 450, -50, 1200, -50}
05795 }
05796 ,{{ 1280, 1280, 780, 1190, 780}
05797 ,{ -450, -450, -720, -960, -720}
05798 ,{ 1190, 440, -60, 1190, -60}
05799 ,{ 1280, 1280, 780, 780, 780}
05800 ,{ 1190, 440, -60, 1190, -60}
05801 }
05802 ,{{ 1200, 460, -50, 1200, -50}
05803 ,{ 1200, 460, -50, 1200, -50}
05804 ,{ 1200, 450, -50, 1200, -50}
05805 ,{ 1200, 460, -50, 1200, -50}
05806 ,{ -280, -280, -1030, -1030, -1030}
05807 }
```

```
05808     }
05809     , {{ { 1280, 1280, 780, -260, 780}
05810         , { 750, 750, 240, -260, 240}
05811         , { 450, 450, -50, -320, -50}
05812         , { 1280, 1280, 780, -560, 780}
05813         , { 450, 450, -50, -320, -50}
05814     }
05815     , {{ { 750, 750, 240, -260, 240}
05816         , { 750, 750, 240, -260, 240}
05817         , { 440, 440, -60, -570, -60}
05818         , { -630, -630, -1140, -1650, -1140}
05819         , { 440, 440, -60, -570, -60}
05820     }
05821     , {{ { 460, 460, -50, -320, -50}
05822         , { 460, 460, -50, -560, -50}
05823         , { 450, 450, -50, -320, -50}
05824         , { 460, 460, -50, -560, -50}
05825         , { 450, 450, -50, -320, -50}
05826     }
05827     , {{ { 1280, 1280, 780, -570, 780}
05828         , { -450, -450, -960, -1470, -960}
05829         , { 440, 440, -60, -570, -60}
05830         , { 1280, 1280, 780, -980, 780}
05831         , { 440, 440, -60, -570, -60}
05832     }
05833     , {{ { 460, 460, -50, -320, -50}
05834         , { 460, 460, -50, -560, -50}
05835         , { 450, 450, -50, -320, -50}
05836         , { 460, 460, -50, -560, -50}
05837         , { -280, -280, -1030, -1540, -1030}
05838     }
05839     }
05840     , {{ { 780, 780, 780, 780, 780}
05841         , { 240, 240, 240, 240, 240}
05842         , { -50, -50, -50, -50, -50}
05843         , { 780, 780, 780, 780, 780}
05844         , { -50, -50, -50, -50, -50}
05845     }
05846     , {{ { 240, 240, 240, 240, 240}
05847         , { 240, 240, 240, 240, 240}
05848         , { -60, -60, -60, -60, -60}
05849         , { -900, -1140, -900, -1140, -900}
05850         , { -60, -60, -60, -60, -60}
05851     }
05852     , {{ { -50, -50, -50, -50, -50}
05853         , { -50, -50, -50, -50, -50}
05854         , { -50, -50, -50, -50, -50}
05855         , { -50, -50, -50, -50, -50}
05856         , { -50, -50, -50, -50, -50}
05857     }
05858     , {{ { 780, 780, 780, 780, 780}
05859         , { -720, -960, -720, -960, -720}
05860         , { -60, -60, -60, -60, -60}
05861         , { 780, 780, 780, 780, 780}
05862         , { -60, -60, -60, -60, -60}
05863     }
05864     , {{ { -50, -50, -50, -50, -50}
05865         , { -50, -50, -50, -50, -50}
05866         , { -50, -50, -50, -50, -50}
05867         , { -50, -50, -50, -50, -50}
05868         , { -1030, -1030, -1030, -1030, -1030}
05869     }
05870     }
05871     , {{ { 1490, -90, 780, 1490, 780}
05872         , { 1490, -90, 240, 1490, 240}
05873         , { 1200, -150, -50, 1200, -50}
05874         , { 1200, -390, 780, 1200, 780}
05875         , { 1200, -150, -50, 1200, -50}
05876     }
05877     , {{ { 1490, -90, 240, 1490, 240}
05878         , { 1490, -90, 240, 1490, 240}
05879         , { 1190, -400, -60, 1190, -60}
05880         , { -1140, -1480, -1140, -1140, -1140}
05881         , { 1190, -400, -60, 1190, -60}
05882     }
05883     , {{ { 1200, -150, -50, 1200, -50}
05884         , { 1200, -390, -50, 1200, -50}
05885         , { 1200, -150, -50, 1200, -50}
05886         , { 1200, -390, -50, 1200, -50}
05887         , { 1200, -150, -50, 1200, -50}
05888     }
05889     , {{ { 1190, -400, 780, 1190, 780}
05890         , { -960, -1300, -960, -960, -960}
05891         , { 1190, -400, -60, 1190, -60}
05892         , { 780, -810, 780, -470, 780}
05893         , { 1190, -400, -60, 1190, -60}
05894     }
```

```

05895 ,{{ 1200, -150, -50, 1200, -50}
05896 ,{ 1200, -390, -50, 1200, -50}
05897 ,{ 1200, -150, -50, 1200, -50}
05898 ,{ 1200, -390, -50, 1200, -50}
05899 ,{ -1030, -1370, -1030, -1030, -1030}
05900 }
05901 }
05902 ,{{{ 780, 780, 780, 780, 480}
05903 ,{ 480, 240, 240, 240, 480}
05904 ,{ -50, -50, -50, -50, -50}
05905 ,{ 780, 780, 780, 780, -50}
05906 ,{ -50, -50, -50, -50, -50}
05907 }
05908 ,{{{ 480, 240, 240, 240, 480}
05909 ,{ 480, 240, 240, 240, 480}
05910 ,{ -60, -60, -60, -60, -60}
05911 ,{ -900, -1140, -900, -1140, -1140}
05912 ,{ -60, -60, -60, -60, -60}
05913 }
05914 ,{{{ -50, -50, -50, -50, -50}
05915 ,{ -50, -50, -50, -50, -50}
05916 ,{ -50, -50, -50, -50, -50}
05917 ,{ -50, -50, -50, -50, -50}
05918 ,{ -50, -50, -50, -50, -50}
05919 }
05920 ,{{{ 780, 780, 780, 780, -60}
05921 ,{ -720, -960, -720, -960, -960}
05922 ,{ -60, -60, -60, -60, -60}
05923 ,{ 780, 780, 780, 780, -470}
05924 ,{ -60, -60, -60, -60, -60}
05925 }
05926 ,{{{ -50, -50, -50, -50, -50}
05927 ,{ -50, -50, -50, -50, -50}
05928 ,{ -50, -50, -50, -50, -50}
05929 ,{ -50, -50, -50, -50, -50}
05930 ,{ -1030, -1030, -1030, -1030, -1030}
05931 }
05932 }
05933 }
05934 ,{{{ 1560, 1470, 960, 1560, 960}
05935 ,{ 1560, 820, 310, 1560, 550}
05936 ,{ 1430, 690, 180, 1430, 180}
05937 ,{ 1470, 1470, 960, 1430, 960}
05938 ,{ 1300, 560, 50, 1300, 50}
05939 }
05940 ,{{{ 1560, 820, 310, 1560, 550}
05941 ,{ 1560, 820, 310, 1560, 550}
05942 ,{ 1280, 540, 30, 1280, 30}
05943 ,{ -580, -580, -850, -1090, -850}
05944 ,{ 1280, 540, 30, 1280, 30}
05945 }
05946 ,{{{ 1430, 690, 180, 1430, 180}
05947 ,{ 1430, 690, 180, 1430, 180}
05948 ,{ 1430, 690, 180, 1430, 180}
05949 ,{ 1430, 690, 180, 1430, 180}
05950 ,{ 1300, 560, 50, 1300, 50}
05951 }
05952 ,{{{ 1470, 1470, 960, 1280, 960}
05953 ,{ -880, -880, -1150, -1390, -1150}
05954 ,{ 1280, 540, 30, 1280, 30}
05955 ,{ 1470, 1470, 960, 960, 960}
05956 ,{ 1280, 540, 30, 1280, 30}
05957 }
05958 ,{{{ 1430, 690, 180, 1430, 180}
05959 ,{ 1430, 690, 180, 1430, 180}
05960 ,{ 990, 250, -260, 990, -260}
05961 ,{ 1430, 690, 180, 1430, 180}
05962 ,{ -10, -10, -760, -760, -760}
05963 }
05964 }
05965 ,{{{ 1470, 1470, 960, -90, 960}
05966 ,{ 820, 820, 310, -200, 310}
05967 ,{ 690, 690, 180, -90, 180}
05968 ,{ 1470, 1470, 960, -330, 960}
05969 ,{ 560, 560, 50, -220, 50}
05970 }
05971 ,{{{ 820, 820, 310, -200, 310}
05972 ,{ 820, 820, 310, -200, 310}
05973 ,{ 540, 540, 30, -480, 30}
05974 ,{ -580, -580, -1090, -1600, -1090}
05975 ,{ 540, 540, 30, -480, 30}
05976 }
05977 ,{{{ 690, 690, 180, -90, 180}
05978 ,{ 690, 690, 180, -330, 180}
05979 ,{ 690, 690, 180, -90, 180}
05980 ,{ 690, 690, 180, -330, 180}
05981 ,{ 560, 560, 50, -220, 50}

```



```
05982     }
05983     ,{{ 1470, 1470, 960, -480, 960}
05984     ,{ -880, -880, -1390, -1900, -1390}
05985     ,{ 540, 540, 30, -480, 30}
05986     ,{ 1470, 1470, 960, -800, 960}
05987     ,{ 540, 540, 30, -480, 30}
05988     }
05989     ,{{ 690, 690, 180, -330, 180}
05990     ,{ 690, 690, 180, -330, 180}
05991     ,{ 250, 250, -260, -530, -260}
05992     ,{ 690, 690, 180, -330, 180}
05993     ,{ -10, -10, -760, -1270, -760}
05994     }
05995     }
05996     ,{{{ 960, 960, 960, 960, 960}
05997     ,{ 310, 310, 310, 310, 310}
05998     ,{ 180, 180, 180, 180, 180}
05999     ,{ 960, 960, 960, 960, 960}
06000     ,{ 50, 50, 50, 50, 50}
06001     }
06002     ,{{{ 310, 310, 310, 310, 310}
06003     ,{ 310, 310, 310, 310, 310}
06004     ,{ 30, 30, 30, 30, 30}
06005     ,{ -850, -1090, -850, -1090, -850}
06006     ,{ 30, 30, 30, 30, 30}
06007     }
06008     ,{{{ 180, 180, 180, 180, 180}
06009     ,{ 180, 180, 180, 180, 180}
06010     ,{ 180, 180, 180, 180, 180}
06011     ,{ 180, 180, 180, 180, 180}
06012     ,{ 50, 50, 50, 50, 50}
06013     }
06014     ,{{{ 960, 960, 960, 960, 960}
06015     ,{ -1150, -1390, -1150, -1390, -1150}
06016     ,{ 30, 30, 30, 30, 30}
06017     ,{ 960, 960, 960, 960, 960}
06018     ,{ 30, 30, 30, 30, 30}
06019     }
06020     ,{{{ 180, 180, 180, 180, 180}
06021     ,{ 180, 180, 180, 180, 180}
06022     ,{ -260, -260, -260, -260, -260}
06023     ,{ 180, 180, 180, 180, 180}
06024     ,{ -760, -760, -760, -760, -760}
06025     }
06026     }
06027     ,{{{ 1560, 80, 960, 1560, 960}
06028     ,{ 1560, -30, 310, 1560, 310}
06029     ,{ 1430, 80, 180, 1430, 180}
06030     ,{ 1430, -160, 960, 1430, 960}
06031     ,{ 1300, -50, 50, 1300, 50}
06032     }
06033     ,{{{ 1560, -30, 310, 1560, 310}
06034     ,{ 1560, -30, 310, 1560, 310}
06035     ,{ 1280, -310, 30, 1280, 30}
06036     ,{ -1090, -1430, -1090, -1090, -1090}
06037     ,{ 1280, -310, 30, 1280, 30}
06038     }
06039     ,{{{ 1430, 80, 180, 1430, 180}
06040     ,{ 1430, -160, 180, 1430, 180}
06041     ,{ 1430, 80, 180, 1430, 180}
06042     ,{ 1430, -160, 180, 1430, 180}
06043     ,{ 1300, -50, 50, 1300, 50}
06044     }
06045     ,{{{ 1280, -310, 960, 1280, 960}
06046     ,{ -1390, -1730, -1390, -1390, -1390}
06047     ,{ 1280, -310, 30, 1280, 30}
06048     ,{ 960, -630, 960, -290, 960}
06049     ,{ 1280, -310, 30, 1280, 30}
06050     }
06051     ,{{{ 1430, -160, 180, 1430, 180}
06052     ,{ 1430, -160, 180, 1430, 180}
06053     ,{ 990, -360, -260, 990, -260}
06054     ,{ 1430, -160, 180, 1430, 180}
06055     ,{ -760, -1100, -760, -760, -760}
06056     }
06057     }
06058     ,{{{ 960, 960, 960, 960, 550}
06059     ,{ 550, 310, 310, 310, 550}
06060     ,{ 180, 180, 180, 180, 180}
06061     ,{ 960, 960, 960, 960, 180}
06062     ,{ 50, 50, 50, 50, 50}
06063     }
06064     ,{{{ 550, 310, 310, 310, 550}
06065     ,{ 550, 310, 310, 310, 550}
06066     ,{ 30, 30, 30, 30, 30}
06067     ,{ -850, -1090, -850, -1090, -1090}
06068     ,{ 30, 30, 30, 30, 30}
```

```

06069      }
06070      ,{{ 180, 180, 180, 180, 180}
06071      ,{ 180, 180, 180, 180, 180}
06072      ,{ 180, 180, 180, 180, 180}
06073      ,{ 180, 180, 180, 180, 180}
06074      ,{ 50, 50, 50, 50, 50}
06075      }
06076      ,{{ 960, 960, 960, 960, 30}
06077      ,{ -1150, -1390, -1150, -1390, -1390}
06078      ,{ 30, 30, 30, 30, 30}
06079      ,{ 960, 960, 960, 960, -290}
06080      ,{ 30, 30, 30, 30, 30}
06081      }
06082      ,{{ 180, 180, 180, 180, 180}
06083      ,{ 180, 180, 180, 180, 180}
06084      ,{ -260, -260, -260, -260, -260}
06085      ,{ 180, 180, 180, 180, 180}
06086      ,{ -760, -760, -760, -760, -760}
06087      }
06088      }
06089      }
06090      ,{{{ 1560, 1470, 960, 1560, 960}
06091      ,{ 1560, 820, 310, 1560, 550}
06092      ,{ 1430, 690, 180, 1430, 180}
06093      ,{ 1470, 1470, 960, 1430, 960}
06094      ,{ 1300, 560, 50, 1300, 50}
06095      }
06096      ,{{ 1560, 820, 310, 1560, 550}
06097      ,{ 1560, 820, 310, 1560, 550}
06098      ,{ 1280, 540, 30, 1280, 30}
06099      ,{ -360, -360, -630, -870, -630}
06100      ,{ 1280, 540, 30, 1280, 30}
06101      }
06102      ,{{ 1430, 690, 180, 1430, 180}
06103      ,{ 1430, 690, 180, 1430, 180}
06104      ,{ 1430, 690, 180, 1430, 180}
06105      ,{ 1430, 690, 180, 1430, 180}
06106      ,{ 1300, 560, 50, 1300, 50}
06107      }
06108      ,{{ 1470, 1470, 960, 1280, 960}
06109      ,{ -30, -30, -720, -960, -720}
06110      ,{ 1280, 540, 30, 1280, 30}
06111      ,{ 1470, 1470, 960, 960, 960}
06112      ,{ 1280, 540, 30, 1280, 30}
06113      }
06114      ,{{ 1430, 690, 180, 1430, 180}
06115      ,{ 1430, 690, 180, 1430, 180}
06116      ,{ 1200, 450, -50, 1200, -50}
06117      ,{ 1430, 690, 180, 1430, 180}
06118      ,{ -10, -10, -760, -760, -760}
06119      }
06120      }
06121      ,{{{ 1470, 1470, 960, -90, 960}
06122      ,{ 820, 820, 310, -200, 310}
06123      ,{ 690, 690, 180, -90, 180}
06124      ,{ 1470, 1470, 960, -330, 960}
06125      ,{ 560, 560, 50, -220, 50}
06126      }
06127      ,{{{ 820, 820, 310, -200, 310}
06128      ,{ 820, 820, 310, -200, 310}
06129      ,{ 540, 540, 30, -480, 30}
06130      ,{ -360, -360, -870, -1380, -870}
06131      ,{ 540, 540, 30, -480, 30}
06132      }
06133      ,{{{ 690, 690, 180, -90, 180}
06134      ,{ 690, 690, 180, -330, 180}
06135      ,{ 690, 690, 180, -90, 180}
06136      ,{ 690, 690, 180, -330, 180}
06137      ,{ 560, 560, 50, -220, 50}
06138      }
06139      ,{{ 1470, 1470, 960, -480, 960}
06140      ,{ -30, -30, -960, -1470, -960}
06141      ,{ 540, 540, 30, -480, 30}
06142      ,{ 1470, 1470, 960, -800, 960}
06143      ,{ 540, 540, 30, -480, 30}
06144      }
06145      ,{{{ 690, 690, 180, -320, 180}
06146      ,{ 690, 690, 180, -330, 180}
06147      ,{ 450, 450, -50, -320, -50}
06148      ,{ 690, 690, 180, -330, 180}
06149      ,{ -10, -10, -760, -1270, -760}
06150      }
06151      }
06152      ,{{{ 960, 960, 960, 960, 960}
06153      ,{ 310, 310, 310, 310, 310}
06154      ,{ 180, 180, 180, 180, 180}
06155      ,{ 960, 960, 960, 960, 960}

```

```
06156     , {      50,      50,      50,      50,      50}
06157     }
06158     , { {      310,      310,      310,      310,      310}
06159     , {      310,      310,      310,      310,      310}
06160     , {        30,        30,        30,        30,        30}
06161     , {     -630,     -870,     -630,     -870,     -630}
06162     , {        30,        30,        30,        30,        30}
06163     }
06164     , { {      180,      180,      180,      180,      180}
06165     , {      180,      180,      180,      180,      180}
06166     , {      180,      180,      180,      180,      180}
06167     , {      180,      180,      180,      180,      180}
06168     , {        50,        50,        50,        50,        50}
06169     }
06170     , { {      960,      960,      960,      960,      960}
06171     , {     -720,     -960,     -720,     -960,     -720}
06172     , {        30,        30,        30,        30,        30}
06173     , {      960,      960,      960,      960,      960}
06174     , {        30,        30,        30,        30,        30}
06175     }
06176     , { {      180,      180,      180,      180,      180}
06177     , {      180,      180,      180,      180,      180}
06178     , {     -50,     -50,     -50,     -50,     -50}
06179     , {      180,      180,      180,      180,      180}
06180     , {     -760,     -760,     -760,     -760,     -760}
06181     }
06182     }
06183     , { { {      1560,        80,      960,      1560,      960}
06184     , {      1560,     -30,      310,      1560,      310}
06185     , {      1430,        80,      180,      1430,      180}
06186     , {      1430,    -160,      960,      1430,      960}
06187     , {      1300,     -50,        50,      1300,        50}
06188     }
06189     , { {      1560,     -30,      310,      1560,      310}
06190     , {      1560,     -30,      310,      1560,      310}
06191     , {      1280,    -310,        30,      1280,        30}
06192     , {     -870,   -1210,     -870,     -870,     -870}
06193     , {      1280,    -310,        30,      1280,        30}
06194     }
06195     , { {      1430,        80,      180,      1430,      180}
06196     , {      1430,   -160,      180,      1430,      180}
06197     , {      1430,        80,      180,      1430,      180}
06198     , {      1430,   -160,      180,      1430,      180}
06199     , {      1300,     -50,        50,      1300,        50}
06200     }
06201     , { {      1280,    -310,      960,      1280,      960}
06202     , {     -960,   -1300,     -960,     -960,     -960}
06203     , {      1280,    -310,        30,      1280,        30}
06204     , {        960,     -630,      960,     -290,      960}
06205     , {      1280,    -310,        30,      1280,        30}
06206     }
06207     , { {      1430,   -150,      180,      1430,      180}
06208     , {      1430,   -160,      180,      1430,      180}
06209     , {      1200,   -150,     -50,      1200,     -50}
06210     , {      1430,   -160,      180,      1430,      180}
06211     , {     -760,   -1100,     -760,     -760,     -760}
06212     }
06213     }
06214     , { { {      960,      960,      960,      960,      550}
06215     , {      550,      310,      310,      310,      550}
06216     , {      180,      180,      180,      180,      180}
06217     , {      960,      960,      960,      960,      180}
06218     , {        50,        50,        50,        50,        50}
06219     }
06220     , { {      550,      310,      310,      310,      550}
06221     , {      550,      310,      310,      310,      550}
06222     , {        30,        30,        30,        30,        30}
06223     , {     -630,     -870,     -630,     -870,     -870}
06224     , {        30,        30,        30,        30,        30}
06225     }
06226     , { {      180,      180,      180,      180,      180}
06227     , {      180,      180,      180,      180,      180}
06228     , {      180,      180,      180,      180,      180}
06229     , {      180,      180,      180,      180,      180}
06230     , {        50,        50,        50,        50,        50}
06231     }
06232     , { {      960,      960,      960,      960,        30}
06233     , {     -720,     -960,     -720,     -960,     -960}
06234     , {        30,        30,        30,        30,        30}
06235     , {      960,      960,      960,      960,     -290}
06236     , {        30,        30,        30,        30,        30}
06237     }
06238     , { {      180,      180,      180,      180,      180}
06239     , {      180,      180,      180,      180,      180}
06240     , {     -50,     -50,     -50,     -50,     -50}
06241     , {      180,      180,      180,      180,      180}
06242     , {     -760,     -760,     -760,     -760,     -760}
```

```
06243     }
06244     }
06245     }
06246     }
06247     ,{{{ INF, INF, INF, INF, INF}
06248     ,{ INF, INF, INF, INF, INF}
06249     ,{ INF, INF, INF, INF, INF}
06250     ,{ INF, INF, INF, INF, INF}
06251     ,{ INF, INF, INF, INF, INF}
06252     }
06253     ,{{ INF, INF, INF, INF, INF}
06254     ,{ INF, INF, INF, INF, INF}
06255     ,{ INF, INF, INF, INF, INF}
06256     ,{ INF, INF, INF, INF, INF}
06257     ,{ INF, INF, INF, INF, INF}
06258     }
06259     ,{{ INF, INF, INF, INF, INF}
06260     ,{ INF, INF, INF, INF, INF}
06261     ,{ INF, INF, INF, INF, INF}
06262     ,{ INF, INF, INF, INF, INF}
06263     ,{ INF, INF, INF, INF, INF}
06264     }
06265     ,{{ INF, INF, INF, INF, INF}
06266     ,{ INF, INF, INF, INF, INF}
06267     ,{ INF, INF, INF, INF, INF}
06268     ,{ INF, INF, INF, INF, INF}
06269     ,{ INF, INF, INF, INF, INF}
06270     }
06271     ,{{ INF, INF, INF, INF, INF}
06272     ,{ INF, INF, INF, INF, INF}
06273     ,{ INF, INF, INF, INF, INF}
06274     ,{ INF, INF, INF, INF, INF}
06275     ,{ INF, INF, INF, INF, INF}
06276     }
06277     }
06278     ,{{{ INF, INF, INF, INF, INF}
06279     ,{ INF, INF, INF, INF, INF}
06280     ,{ INF, INF, INF, INF, INF}
06281     ,{ INF, INF, INF, INF, INF}
06282     ,{ INF, INF, INF, INF, INF}
06283     }
06284     ,{{{ INF, INF, INF, INF, INF}
06285     ,{ INF, INF, INF, INF, INF}
06286     ,{ INF, INF, INF, INF, INF}
06287     ,{ INF, INF, INF, INF, INF}
06288     ,{ INF, INF, INF, INF, INF}
06289     }
06290     ,{{{ INF, INF, INF, INF, INF}
06291     ,{ INF, INF, INF, INF, INF}
06292     ,{ INF, INF, INF, INF, INF}
06293     ,{ INF, INF, INF, INF, INF}
06294     ,{ INF, INF, INF, INF, INF}
06295     }
06296     ,{{{ INF, INF, INF, INF, INF}
06297     ,{ INF, INF, INF, INF, INF}
06298     ,{ INF, INF, INF, INF, INF}
06299     ,{ INF, INF, INF, INF, INF}
06300     ,{ INF, INF, INF, INF, INF}
06301     }
06302     ,{{{ INF, INF, INF, INF, INF}
06303     ,{ INF, INF, INF, INF, INF}
06304     ,{ INF, INF, INF, INF, INF}
06305     ,{ INF, INF, INF, INF, INF}
06306     ,{ INF, INF, INF, INF, INF}
06307     }
06308     }
06309     ,{{{ INF, INF, INF, INF, INF}
06310     ,{ INF, INF, INF, INF, INF}
06311     ,{ INF, INF, INF, INF, INF}
06312     ,{ INF, INF, INF, INF, INF}
06313     ,{ INF, INF, INF, INF, INF}
06314     }
06315     ,{{ INF, INF, INF, INF, INF}
06316     ,{ INF, INF, INF, INF, INF}
06317     ,{ INF, INF, INF, INF, INF}
06318     ,{ INF, INF, INF, INF, INF}
06319     ,{ INF, INF, INF, INF, INF}
06320     }
06321     ,{{ INF, INF, INF, INF, INF}
06322     ,{ INF, INF, INF, INF, INF}
06323     ,{ INF, INF, INF, INF, INF}
06324     ,{ INF, INF, INF, INF, INF}
06325     ,{ INF, INF, INF, INF, INF}
06326     }
06327     ,{{ INF, INF, INF, INF, INF}
06328     ,{ INF, INF, INF, INF, INF}
06329     ,{ INF, INF, INF, INF, INF}
```

```
06330     , {   INF,   INF,   INF,   INF,   INF }
06331     , {   INF,   INF,   INF,   INF,   INF }
06332     }
06333     , { {   INF,   INF,   INF,   INF,   INF }
06334     , {   INF,   INF,   INF,   INF,   INF }
06335     , {   INF,   INF,   INF,   INF,   INF }
06336     , {   INF,   INF,   INF,   INF,   INF }
06337     , {   INF,   INF,   INF,   INF,   INF }
06338     }
06339     }
06340     , { { {   INF,   INF,   INF,   INF,   INF }
06341     , {   INF,   INF,   INF,   INF,   INF }
06342     , {   INF,   INF,   INF,   INF,   INF }
06343     , {   INF,   INF,   INF,   INF,   INF }
06344     , {   INF,   INF,   INF,   INF,   INF }
06345     }
06346     , { {   INF,   INF,   INF,   INF,   INF }
06347     , {   INF,   INF,   INF,   INF,   INF }
06348     , {   INF,   INF,   INF,   INF,   INF }
06349     , {   INF,   INF,   INF,   INF,   INF }
06350     , {   INF,   INF,   INF,   INF,   INF }
06351     }
06352     , { {   INF,   INF,   INF,   INF,   INF }
06353     , {   INF,   INF,   INF,   INF,   INF }
06354     , {   INF,   INF,   INF,   INF,   INF }
06355     , {   INF,   INF,   INF,   INF,   INF }
06356     , {   INF,   INF,   INF,   INF,   INF }
06357     }
06358     , { {   INF,   INF,   INF,   INF,   INF }
06359     , {   INF,   INF,   INF,   INF,   INF }
06360     , {   INF,   INF,   INF,   INF,   INF }
06361     , {   INF,   INF,   INF,   INF,   INF }
06362     , {   INF,   INF,   INF,   INF,   INF }
06363     }
06364     , { {   INF,   INF,   INF,   INF,   INF }
06365     , {   INF,   INF,   INF,   INF,   INF }
06366     , {   INF,   INF,   INF,   INF,   INF }
06367     , {   INF,   INF,   INF,   INF,   INF }
06368     , {   INF,   INF,   INF,   INF,   INF }
06369     }
06370     }
06371     , { { {   INF,   INF,   INF,   INF,   INF }
06372     , {   INF,   INF,   INF,   INF,   INF }
06373     , {   INF,   INF,   INF,   INF,   INF }
06374     , {   INF,   INF,   INF,   INF,   INF }
06375     , {   INF,   INF,   INF,   INF,   INF }
06376     }
06377     , { {   INF,   INF,   INF,   INF,   INF }
06378     , {   INF,   INF,   INF,   INF,   INF }
06379     , {   INF,   INF,   INF,   INF,   INF }
06380     , {   INF,   INF,   INF,   INF,   INF }
06381     , {   INF,   INF,   INF,   INF,   INF }
06382     }
06383     , { {   INF,   INF,   INF,   INF,   INF }
06384     , {   INF,   INF,   INF,   INF,   INF }
06385     , {   INF,   INF,   INF,   INF,   INF }
06386     , {   INF,   INF,   INF,   INF,   INF }
06387     , {   INF,   INF,   INF,   INF,   INF }
06388     }
06389     , { {   INF,   INF,   INF,   INF,   INF }
06390     , {   INF,   INF,   INF,   INF,   INF }
06391     , {   INF,   INF,   INF,   INF,   INF }
06392     , {   INF,   INF,   INF,   INF,   INF }
06393     , {   INF,   INF,   INF,   INF,   INF }
06394     }
06395     , { {   INF,   INF,   INF,   INF,   INF }
06396     , {   INF,   INF,   INF,   INF,   INF }
06397     , {   INF,   INF,   INF,   INF,   INF }
06398     , {   INF,   INF,   INF,   INF,   INF }
06399     , {   INF,   INF,   INF,   INF,   INF }
06400     }
06401     }
06402     }
06403     , { { { {   1170,   780,   490,   1170,   490 }
06404     , {   1120,   580,   290,   1120,   290 }
06405     , {   1170,   640,   340,   1170,   340 }
06406     , {   1120,   780,   490,   1120,   490 }
06407     , {   1060,   530,   230,   1060,   230 }
06408     }
06409     , { {   970,   440,   170,   970,   170 }
06410     , {   970,   440,   140,   970,   140 }
06411     , {   660,   130,  -160,   660,  -160 }
06412     , {   220,   220,   170,   -80,   170 }
06413     , {   660,   130,  -160,   660,  -160 }
06414     }
06415     , { {   1120,   580,   290,   1120,   290 }
06416     , {   1120,   580,   290,   1120,   290 }
```

```
06417      , { 1110, 580, 280, 1110, 280}
06418      , { 1120, 580, 290, 1120, 290}
06419      , { 1060, 530, 230, 1060, 230}
06420      }
06421      , { { 780, 780, 490, 660, 490}
06422      , { -60, -60, -120, -370, -120}
06423      , { 660, 130, -160, 660, -160}
06424      , { 780, 780, 490, 470, 490}
06425      , { 660, 130, -160, 660, -160}
06426      }
06427      , { { 1170, 640, 340, 1170, 340}
06428      , { 1120, 580, 290, 1120, 290}
06429      , { 1170, 640, 340, 1170, 340}
06430      , { 1120, 580, 290, 1120, 290}
06431      , { 40, 40, -500, -510, -500}
06432      }
06433      }
06434      , { { { 780, 780, 490, -330, 490}
06435      , { 580, 580, 290, -620, 290}
06436      , { 640, 640, 340, -330, 340}
06437      , { 780, 780, 490, -620, 490}
06438      , { 530, 530, 230, -440, 230}
06439      }
06440      , { { 440, 440, 140, -770, 140}
06441      , { 440, 440, 140, -770, 140}
06442      , { 130, 130, -160, -1080, -160}
06443      , { 220, 220, -70, -980, -70}
06444      , { 130, 130, -160, -1080, -160}
06445      }
06446      , { { 580, 580, 290, -390, 290}
06447      , { 580, 580, 290, -620, 290}
06448      , { 580, 580, 280, -390, 280}
06449      , { 580, 580, 290, -620, 290}
06450      , { 530, 530, 230, -440, 230}
06451      }
06452      , { { 780, 780, 490, -1080, 490}
06453      , { -60, -60, -350, -1270, -350}
06454      , { 130, 130, -160, -1080, -160}
06455      , { 780, 780, 490, -1680, 490}
06456      , { 130, 130, -160, -1080, -160}
06457      }
06458      , { { 640, 640, 340, -330, 340}
06459      , { 580, 580, 290, -620, 290}
06460      , { 640, 640, 340, -330, 340}
06461      , { 580, 580, 290, -620, 290}
06462      , { 40, 40, -500, -1410, -500}
06463      }
06464      }
06465      , { { { 480, 470, 480, 470, 480}
06466      , { 280, 270, 280, 270, 280}
06467      , { 340, 330, 340, 330, 340}
06468      , { 480, 470, 480, 470, 480}
06469      , { 230, 220, 230, 220, 230}
06470      }
06471      , { { 170, 130, 170, 130, 170}
06472      , { 140, 130, 140, 130, 140}
06473      , { -170, -180, -170, -180, -170}
06474      , { 170, -80, 170, -80, 170}
06475      , { -170, -180, -170, -180, -170}
06476      }
06477      , { { 280, 270, 280, 270, 280}
06478      , { 280, 270, 280, 270, 280}
06479      , { 280, 270, 280, 270, 280}
06480      , { 280, 270, 280, 270, 280}
06481      , { 230, 220, 230, 220, 230}
06482      }
06483      , { { 480, 470, 480, 470, 480}
06484      , { -120, -370, -120, -370, -120}
06485      , { -170, -180, -170, -180, -170}
06486      , { 480, 470, 480, 470, 480}
06487      , { -170, -180, -170, -180, -170}
06488      }
06489      , { { 340, 330, 340, 330, 340}
06490      , { 280, 270, 280, 270, 280}
06491      , { 340, 330, 340, 330, 340}
06492      , { 280, 270, 280, 270, 280}
06493      , { -500, -510, -500, -510, -500}
06494      }
06495      }
06496      , { { { 1170, -510, 490, 1170, 490}
06497      , { 1120, -800, 290, 1120, 290}
06498      , { 1170, -510, 340, 1170, 340}
06499      , { 1120, -800, 490, 1120, 490}
06500      , { 1060, -620, 230, 1060, 230}
06501      }
06502      , { { 970, -950, 140, 970, 140}
06503      , { 970, -950, 140, 970, 140}
```

```
06504      , { 660, -1260, -160, 660, -160}
06505      , { -70, -1160, -70, -490, -70}
06506      , { 660, -1260, -160, 660, -160}
06507      }
06508      , { { 1120, -570, 290, 1120, 290}
06509      , { 1120, -800, 290, 1120, 290}
06510      , { 1110, -570, 280, 1110, 280}
06511      , { 1120, -800, 290, 1120, 290}
06512      , { 1060, -620, 230, 1060, 230}
06513      }
06514      , { { 660, -1260, 490, 660, 490}
06515      , { -350, -1450, -350, -780, -350}
06516      , { 660, -1260, -160, 660, -160}
06517      , { 490, -1860, 490, -1190, 490}
06518      , { 660, -1260, -160, 660, -160}
06519      }
06520      , { { 1170, -510, 340, 1170, 340}
06521      , { 1120, -800, 290, 1120, 290}
06522      , { 1170, -510, 340, 1170, 340}
06523      , { 1120, -800, 290, 1120, 290}
06524      , { -500, -1590, -500, -920, -500}
06525      }
06526      }
06527      , { { { 480, 470, 480, 470, -600}
06528      , { 280, 270, 280, 270, -600}
06529      , { 340, 330, 340, 330, -640}
06530      , { 480, 470, 480, 470, -690}
06531      , { 230, 220, 230, 220, -750}
06532      }
06533      , { { 170, 130, 170, 130, -600}
06534      , { 140, 130, 140, 130, -600}
06535      , { -170, -180, -170, -180, -1150}
06536      , { 170, -80, 170, -80, -1050}
06537      , { -170, -180, -170, -180, -1150}
06538      }
06539      , { { 280, 270, 280, 270, -690}
06540      , { 280, 270, 280, 270, -690}
06541      , { 280, 270, 280, 270, -700}
06542      , { 280, 270, 280, 270, -690}
06543      , { 230, 220, 230, 220, -750}
06544      }
06545      , { { 480, 470, 480, 470, -1150}
06546      , { -120, -370, -120, -370, -1340}
06547      , { -170, -180, -170, -180, -1150}
06548      , { 480, 470, 480, 470, -1750}
06549      , { -170, -180, -170, -180, -1150}
06550      }
06551      , { { 340, 330, 340, 330, -640}
06552      , { 280, 270, 280, 270, -690}
06553      , { 340, 330, 340, 330, -640}
06554      , { 280, 270, 280, 270, -690}
06555      , { -500, -510, -500, -510, -1480}
06556      }
06557      }
06558      }
06559      , { { { { 1140, 780, 490, 1140, 490}
06560      , { 1140, 600, 310, 1140, 310}
06561      , { 690, 150, -140, 690, -140}
06562      , { 780, 780, 490, 770, 490}
06563      , { 690, 190, -140, 690, -140}
06564      }
06565      , { { 1140, 600, 310, 1140, 310}
06566      , { 1140, 600, 310, 1140, 310}
06567      , { 690, 150, -140, 690, -140}
06568      , { -580, -580, -640, -890, -640}
06569      , { 690, 150, -140, 690, -140}
06570      }
06571      , { { 770, 240, -50, 770, -50}
06572      , { 770, 240, -50, 770, -50}
06573      , { 470, -60, -360, 470, -360}
06574      , { 770, 240, -50, 770, -50}
06575      , { 470, -60, -360, 470, -360}
06576      }
06577      , { { 780, 780, 490, 690, 490}
06578      , { -110, -110, -170, -420, -170}
06579      , { 690, 150, -140, 690, -140}
06580      , { 780, 780, 490, 470, 490}
06581      , { 690, 150, -140, 690, -140}
06582      }
06583      , { { 770, 240, -50, 770, -50}
06584      , { 770, 240, -50, 770, -50}
06585      , { 160, -370, -670, 160, -670}
06586      , { 770, 240, -50, 770, -50}
06587      , { 190, 190, -340, -360, -340}
06588      }
06589      }
06590      , { { { 780, 780, 490, -600, 490}
```

```

06591      , { 600, 600, 310, -600, 310}
06592      , { 150, 150, -140, -1030, -140}
06593      , { 780, 780, 490, -970, 490}
06594      , { 190, 190, -140, -1030, -140}
06595      }
06596      , { { 600, 600, 310, -600, 310}
06597      , { 600, 600, 310, -600, 310}
06598      , { 150, 150, -140, -1050, -140}
06599      , { -580, -580, -880, -1790, -880}
06600      , { 150, 150, -140, -1050, -140}
06601      }
06602      , { { 240, 240, -50, -970, -50}
06603      , { 240, 240, -50, -970, -50}
06604      , { -60, -60, -360, -1030, -360}
06605      , { 240, 240, -50, -970, -50}
06606      , { -60, -60, -360, -1030, -360}
06607      }
06608      , { { 780, 780, 490, -1050, 490}
06609      , { -110, -110, -400, -1320, -400}
06610      , { 150, 150, -140, -1050, -140}
06611      , { 780, 780, 490, -1680, 490}
06612      , { 150, 150, -140, -1050, -140}
06613      }
06614      , { { 240, 240, -50, -970, -50}
06615      , { 240, 240, -50, -970, -50}
06616      , { -370, -370, -670, -1340, -670}
06617      , { 240, 240, -50, -970, -50}
06618      , { 190, 190, -340, -1260, -340}
06619      }
06620      }
06621      , { { { 480, 470, 480, 470, 480}
06622      , { 300, 290, 300, 290, 300}
06623      , { -140, -150, -140, -150, -140}
06624      , { 480, 470, 480, 470, 480}
06625      , { -140, -150, -140, -150, -140}
06626      }
06627      , { { 300, 290, 300, 290, 300}
06628      , { 300, 290, 300, 290, 300}
06629      , { -140, -150, -140, -150, -140}
06630      , { -640, -890, -640, -890, -640}
06631      , { -140, -150, -140, -150, -140}
06632      }
06633      , { { -60, -70, -60, -70, -60}
06634      , { -60, -70, -60, -70, -60}
06635      , { -360, -370, -360, -370, -360}
06636      , { -60, -70, -60, -70, -60}
06637      , { -360, -370, -360, -370, -360}
06638      }
06639      , { { 480, 470, 480, 470, 480}
06640      , { -170, -420, -170, -420, -170}
06641      , { -140, -150, -140, -150, -140}
06642      , { 480, 470, 480, 470, 480}
06643      , { -140, -150, -140, -150, -140}
06644      }
06645      , { { -60, -70, -60, -70, -60}
06646      , { -60, -70, -60, -70, -60}
06647      , { -670, -680, -670, -680, -670}
06648      , { -60, -70, -60, -70, -60}
06649      , { -350, -360, -350, -360, -350}
06650      }
06651      }
06652      , { { { 1140, -780, 490, 1140, 490}
06653      , { 1140, -780, 310, 1140, 310}
06654      , { 690, -1210, -140, 690, -140}
06655      , { 770, -1150, 490, 770, 490}
06656      , { 690, -1210, -140, 690, -140}
06657      }
06658      , { { 1140, -780, 310, 1140, 310}
06659      , { 1140, -780, 310, 1140, 310}
06660      , { 690, -1230, -140, 690, -140}
06661      , { -880, -1970, -880, -1300, -880}
06662      , { 690, -1230, -140, 690, -140}
06663      }
06664      , { { 770, -1150, -50, 770, -50}
06665      , { 770, -1150, -50, 770, -50}
06666      , { 470, -1210, -360, 470, -360}
06667      , { 770, -1150, -50, 770, -50}
06668      , { 470, -1210, -360, 470, -360}
06669      }
06670      , { { 690, -1230, 490, 690, 490}
06671      , { -400, -1500, -400, -830, -400}
06672      , { 690, -1230, -140, 690, -140}
06673      , { 490, -1860, 490, -1190, 490}
06674      , { 690, -1230, -140, 690, -140}
06675      }
06676      , { { 770, -1150, -50, 770, -50}
06677      , { 770, -1150, -50, 770, -50}

```



```
06678     , { 160, -1520, -670, 160, -670}
06679     , { 770, -1150, -50, 770, -50}
06680     , { -340, -1440, -340, -770, -340}
06681     }
06682     }
06683     , {{ { 480, 470, 480, 470, -430}
06684     , { 300, 290, 300, 290, -430}
06685     , { -140, -150, -140, -150, -1120}
06686     , { 480, 470, 480, 470, -1040}
06687     , { -140, -150, -140, -150, -1120}
06688     }
06689     , {{ { 300, 290, 300, 290, -430}
06690     , { 300, 290, 300, 290, -430}
06691     , { -140, -150, -140, -150, -1120}
06692     , { -640, -890, -640, -890, -1860}
06693     , { -140, -150, -140, -150, -1120}
06694     }
06695     , {{ { -60, -70, -60, -70, -1040}
06696     , { -60, -70, -60, -70, -1040}
06697     , { -360, -370, -360, -370, -1340}
06698     , { -60, -70, -60, -70, -1040}
06699     , { -360, -370, -360, -370, -1340}
06700     }
06701     , {{ { 480, 470, 480, 470, -1120}
06702     , { -170, -420, -170, -420, -1390}
06703     , { -140, -150, -140, -150, -1120}
06704     , { 480, 470, 480, 470, -1750}
06705     , { -140, -150, -140, -150, -1120}
06706     }
06707     , {{ { -60, -70, -60, -70, -1040}
06708     , { -60, -70, -60, -70, -1040}
06709     , { -670, -680, -670, -680, -1650}
06710     , { -60, -70, -60, -70, -1040}
06711     , { -350, -360, -350, -360, -1330}
06712     }
06713     }
06714     }
06715     , {{{ { 940, 940, 650, 630, 650}
06716     , { 220, -130, -190, 220, -190}
06717     , { 220, -310, -600, 220, -600}
06718     , { 940, 940, 650, 630, 650}
06719     , { 220, -70, -600, 220, -600}
06720     }
06721     , {{ { 220, -310, -380, 220, -380}
06722     , { 40, -490, -780, 40, -780}
06723     , { 220, -310, -600, 220, -600}
06724     , { -320, -320, -380, -630, -380}
06725     , { 220, -310, -600, 220, -600}
06726     }
06727     , {{ { 220, -310, -600, 220, -600}
06728     , { 220, -310, -600, 220, -600}
06729     , { 220, -310, -600, 220, -600}
06730     , { 220, -310, -600, 220, -600}
06731     , { 220, -310, -600, 220, -600}
06732     }
06733     , {{ { 940, 940, 650, 630, 650}
06734     , { -130, -130, -190, -440, -190}
06735     , { 220, -310, -600, 220, -600}
06736     , { 940, 940, 650, 630, 650}
06737     , { 220, -310, -600, 220, -600}
06738     }
06739     , {{ { 220, -70, -600, 220, -600}
06740     , { 220, -310, -600, 220, -600}
06741     , { 220, -310, -600, 220, -600}
06742     , { 220, -310, -600, 220, -600}
06743     , { -70, -70, -600, -620, -600}
06744     }
06745     }
06746     , {{{ { 940, 940, 650, -1280, 650}
06747     , { -130, -130, -430, -1340, -430}
06748     , { -310, -310, -600, -1280, -600}
06749     , { 940, 940, 650, -1520, 650}
06750     , { -70, -70, -600, -1280, -600}
06751     }
06752     , {{ { -310, -310, -600, -1520, -600}
06753     , { -490, -490, -780, -1700, -780}
06754     , { -310, -310, -600, -1520, -600}
06755     , { -320, -320, -620, -1530, -620}
06756     , { -310, -310, -600, -1520, -600}
06757     }
06758     , {{ { -310, -310, -600, -1280, -600}
06759     , { -310, -310, -600, -1520, -600}
06760     , { -310, -310, -600, -1280, -600}
06761     , { -310, -310, -600, -1520, -600}
06762     , { -310, -310, -600, -1280, -600}
06763     }
06764     , {{ { 940, 940, 650, -1340, 650}
```

```

06765      , { -130, -130, -430, -1340, -430}
06766      , { -310, -310, -600, -1520, -600}
06767      , {  940,  940,  650, -1520,  650}
06768      , { -310, -310, -600, -1520, -600}
06769      }
06770      , { { -70, -70, -600, -1280, -600}
06771      , { -310, -310, -600, -1520, -600}
06772      , { -310, -310, -600, -1280, -600}
06773      , { -310, -310, -600, -1520, -600}
06774      , { -70, -70, -600, -1520, -600}
06775      }
06776      }
06777      , { { { 640, 630, 640, 630, 640}
06778      , { -190, -440, -190, -440, -190}
06779      , { -610, -620, -610, -620, -610}
06780      , { 640, 630, 640, 630, 640}
06781      , { -610, -620, -610, -620, -610}
06782      }
06783      , { { -380, -620, -380, -620, -380}
06784      , { -790, -800, -790, -800, -790}
06785      , { -610, -620, -610, -620, -610}
06786      , { -380, -630, -380, -630, -380}
06787      , { -610, -620, -610, -620, -610}
06788      }
06789      , { { -610, -620, -610, -620, -610}
06790      , { -610, -620, -610, -620, -610}
06791      , { -610, -620, -610, -620, -610}
06792      , { -610, -620, -610, -620, -610}
06793      , { -610, -620, -610, -620, -610}
06794      }
06795      , { { 640, 630, 640, 630, 640}
06796      , { -190, -440, -190, -440, -190}
06797      , { -610, -620, -610, -620, -610}
06798      , { 640, 630, 640, 630, 640}
06799      , { -610, -620, -610, -620, -610}
06800      }
06801      , { { -610, -620, -610, -620, -610}
06802      , { -610, -620, -610, -620, -610}
06803      , { -610, -620, -610, -620, -610}
06804      , { -610, -620, -610, -620, -610}
06805      , { -610, -620, -610, -620, -610}
06806      }
06807      }
06808      , { { { 650, -1460, 650, 220, 650}
06809      , { 220, -1520, -430, 220, -430}
06810      , { 220, -1460, -600, 220, -600}
06811      , { 650, -1700, 650, 220, 650}
06812      , { 220, -1460, -600, 220, -600}
06813      }
06814      , { { 220, -1700, -600, 220, -600}
06815      , { 40, -1880, -780, 40, -780}
06816      , { 220, -1700, -600, 220, -600}
06817      , { -620, -1710, -620, -1040, -620}
06818      , { 220, -1700, -600, 220, -600}
06819      }
06820      , { { 220, -1460, -600, 220, -600}
06821      , { 220, -1700, -600, 220, -600}
06822      , { 220, -1460, -600, 220, -600}
06823      , { 220, -1700, -600, 220, -600}
06824      , { 220, -1460, -600, 220, -600}
06825      }
06826      , { { 650, -1520, 650, 220, 650}
06827      , { -430, -1520, -430, -850, -430}
06828      , { 220, -1700, -600, 220, -600}
06829      , { 650, -1700, 650, -1030, 650}
06830      , { 220, -1700, -600, 220, -600}
06831      }
06832      , { { 220, -1460, -600, 220, -600}
06833      , { 220, -1700, -600, 220, -600}
06834      , { 220, -1460, -600, 220, -600}
06835      , { 220, -1700, -600, 220, -600}
06836      , { -600, -1700, -600, -1030, -600}
06837      }
06838      }
06839      , { { { 640, 630, 640, 630, -1410}
06840      , { -190, -440, -190, -440, -1410}
06841      , { -610, -620, -610, -620, -1590}
06842      , { 640, 630, 640, 630, -1590}
06843      , { -610, -620, -610, -620, -1590}
06844      }
06845      , { { -380, -620, -380, -620, -1530}
06846      , { -790, -800, -790, -800, -1530}
06847      , { -610, -620, -610, -620, -1590}
06848      , { -380, -630, -380, -630, -1600}
06849      , { -610, -620, -610, -620, -1590}
06850      }
06851      , { { -610, -620, -610, -620, -1590}

```

```
06852     , { -610, -620, -610, -620, -1590}
06853     , { -610, -620, -610, -620, -1590}
06854     , { -610, -620, -610, -620, -1590}
06855     , { -610, -620, -610, -620, -1590}
06856     }
06857     , { { 640, 630, 640, 630, -1410}
06858     , { -190, -440, -190, -440, -1410}
06859     , { -610, -620, -610, -620, -1590}
06860     , { 640, 630, 640, 630, -1590}
06861     , { -610, -620, -610, -620, -1590}
06862     }
06863     , { { -610, -620, -610, -620, -1590}
06864     , { -610, -620, -610, -620, -1590}
06865     , { -610, -620, -610, -620, -1590}
06866     , { -610, -620, -610, -620, -1590}
06867     , { -610, -620, -610, -620, -1590}
06868     }
06869     }
06870     }
06871     , { { { 1490, 1490, 1200, 1280, 1200}
06872     , { 1280, 750, 460, 1280, 460}
06873     , { 780, 240, -50, 780, -50}
06874     , { 1490, 1490, 1200, 1190, 1200}
06875     , { 780, 480, -50, 780, -50}
06876     }
06877     , { { 1280, 750, 460, 1280, 460}
06878     , { 1280, 750, 460, 1280, 460}
06879     , { 780, 240, -50, 780, -50}
06880     , { -90, -90, -150, -400, -150}
06881     , { 780, 240, -50, 780, -50}
06882     }
06883     , { { 780, 240, -50, 780, -50}
06884     , { 780, 240, -50, 780, -50}
06885     , { 780, 240, -50, 780, -50}
06886     , { 780, 240, -50, 780, -50}
06887     , { 780, 240, -50, 780, -50}
06888     }
06889     , { { 1490, 1490, 1200, 1190, 1200}
06890     , { -260, -260, -320, -570, -320}
06891     , { 780, 240, -50, 780, -50}
06892     , { 1490, 1490, 1200, 1190, 1200}
06893     , { 780, 240, -50, 780, -50}
06894     }
06895     , { { 780, 480, -50, 780, -50}
06896     , { 780, 240, -50, 780, -50}
06897     , { 780, 240, -50, 780, -50}
06898     , { 780, 240, -50, 780, -50}
06899     , { 480, 480, -50, -60, -50}
06900     }
06901     }
06902     , { { { 1490, 1490, 1200, -450, 1200}
06903     , { 750, 750, 460, -450, 460}
06904     , { 240, 240, -50, -720, -50}
06905     , { 1490, 1490, 1200, -960, 1200}
06906     , { 480, 480, -50, -720, -50}
06907     }
06908     , { { 750, 750, 460, -450, 460}
06909     , { 750, 750, 460, -450, 460}
06910     , { 240, 240, -50, -960, -50}
06911     , { -90, -90, -390, -1300, -390}
06912     , { 240, 240, -50, -960, -50}
06913     }
06914     , { { 240, 240, -50, -720, -50}
06915     , { 240, 240, -50, -960, -50}
06916     , { 240, 240, -50, -720, -50}
06917     , { 240, 240, -50, -960, -50}
06918     , { 240, 240, -50, -720, -50}
06919     }
06920     , { { 1490, 1490, 1200, -960, 1200}
06921     , { -260, -260, -560, -1470, -560}
06922     , { 240, 240, -50, -960, -50}
06923     , { 1490, 1490, 1200, -960, 1200}
06924     , { 240, 240, -50, -960, -50}
06925     }
06926     , { { 480, 480, -50, -720, -50}
06927     , { 240, 240, -50, -960, -50}
06928     , { 240, 240, -50, -720, -50}
06929     , { 240, 240, -50, -960, -50}
06930     , { 480, 480, -50, -960, -50}
06931     }
06932     }
06933     , { { { 1200, 1190, 1200, 1190, 1200}
06934     , { 450, 440, 450, 440, 450}
06935     , { -50, -60, -50, -60, -50}
06936     , { 1200, 1190, 1200, 1190, 1200}
06937     , { -50, -60, -50, -60, -50}
06938     }
```

```
06939 ,{{ 450, 440, 450, 440, 450}
06940 ,{ 450, 440, 450, 440, 450}
06941 ,{ -50, -60, -50, -60, -50}
06942 ,{ -150, -400, -150, -400, -150}
06943 ,{ -50, -60, -50, -60, -50}
06944 }
06945 ,{{ -50, -60, -50, -60, -50}
06946 ,{ -50, -60, -50, -60, -50}
06947 ,{ -50, -60, -50, -60, -50}
06948 ,{ -50, -60, -50, -60, -50}
06949 ,{ -50, -60, -50, -60, -50}
06950 }
06951 ,{{ 1200, 1190, 1200, 1190, 1200}
06952 ,{ -320, -570, -320, -570, -320}
06953 ,{ -50, -60, -50, -60, -50}
06954 ,{ 1200, 1190, 1200, 1190, 1200}
06955 ,{ -50, -60, -50, -60, -50}
06956 }
06957 ,{{ -50, -60, -50, -60, -50}
06958 ,{ -50, -60, -50, -60, -50}
06959 ,{ -50, -60, -50, -60, -50}
06960 ,{ -50, -60, -50, -60, -50}
06961 ,{ -50, -60, -50, -60, -50}
06962 }
06963 }
06964 ,{{{ 1280, -630, 1200, 1280, 1200}
06965 ,{ 1280, -630, 460, 1280, 460}
06966 ,{ 780, -900, -50, 780, -50}
06967 ,{ 1200, -1140, 1200, 780, 1200}
06968 ,{ 780, -900, -50, 780, -50}
06969 }
06970 ,{{ 1280, -630, 460, 1280, 460}
06971 ,{ 1280, -630, 460, 1280, 460}
06972 ,{ 780, -1140, -50, 780, -50}
06973 ,{ -390, -1480, -390, -810, -390}
06974 ,{ 780, -1140, -50, 780, -50}
06975 }
06976 ,{{ 780, -900, -50, 780, -50}
06977 ,{ 780, -1140, -50, 780, -50}
06978 ,{ 780, -900, -50, 780, -50}
06979 ,{ 780, -1140, -50, 780, -50}
06980 ,{ 780, -900, -50, 780, -50}
06981 }
06982 ,{{ 1200, -1140, 1200, 780, 1200}
06983 ,{ -560, -1650, -560, -980, -560}
06984 ,{ 780, -1140, -50, 780, -50}
06985 ,{ 1200, -1140, 1200, -470, 1200}
06986 ,{ 780, -1140, -50, 780, -50}
06987 }
06988 ,{{ 780, -900, -50, 780, -50}
06989 ,{ 780, -1140, -50, 780, -50}
06990 ,{ 780, -900, -50, 780, -50}
06991 ,{ 780, -1140, -50, 780, -50}
06992 ,{ -50, -1140, -50, -470, -50}
06993 }
06994 }
06995 ,{{{ 1200, 1190, 1200, 1190, -280}
06996 ,{ 450, 440, 450, 440, -280}
06997 ,{ -50, -60, -50, -60, -1030}
06998 ,{ 1200, 1190, 1200, 1190, -1030}
06999 ,{ -50, -60, -50, -60, -1030}
07000 }
07001 ,{{ 450, 440, 450, 440, -280}
07002 ,{ 450, 440, 450, 440, -280}
07003 ,{ -50, -60, -50, -60, -1030}
07004 ,{ -150, -400, -150, -400, -1370}
07005 ,{ -50, -60, -50, -60, -1030}
07006 }
07007 ,{{ -50, -60, -50, -60, -1030}
07008 ,{ -50, -60, -50, -60, -1030}
07009 ,{ -50, -60, -50, -60, -1030}
07010 ,{ -50, -60, -50, -60, -1030}
07011 ,{ -50, -60, -50, -60, -1030}
07012 }
07013 ,{{ 1200, 1190, 1200, 1190, -1030}
07014 ,{ -320, -570, -320, -570, -1540}
07015 ,{ -50, -60, -50, -60, -1030}
07016 ,{ 1200, 1190, 1200, 1190, -1030}
07017 ,{ -50, -60, -50, -60, -1030}
07018 }
07019 ,{{ -50, -60, -50, -60, -1030}
07020 ,{ -50, -60, -50, -60, -1030}
07021 ,{ -50, -60, -50, -60, -1030}
07022 ,{ -50, -60, -50, -60, -1030}
07023 ,{ -50, -60, -50, -60, -1030}
07024 }
07025 }
```

```
07026     }
07027     ,{{{ 1870, 1870, 1570, 1870, 1570}
07028         ,{ 1870, 1340, 1040, 1870, 1040}
07029         ,{ 1570, 1040, 740, 1570, 740}
07030         ,{ 1870, 1870, 1570, 1570, 1570}
07031         ,{ 1570, 1040, 740, 1570, 740}
07032     }
07033     ,{{{ 1870, 1340, 1040, 1870, 1040}
07034         ,{ 1870, 1340, 1040, 1870, 1040}
07035         ,{ 1560, 1030, 730, 1560, 730}
07036         ,{ -50, -50, -110, -360, -110}
07037         ,{ 1560, 1030, 730, 1560, 730}
07038     }
07039     ,{{{ 1570, 1040, 750, 1570, 750}
07040         ,{ 1570, 1040, 750, 1570, 750}
07041         ,{ 1570, 1040, 740, 1570, 740}
07042         ,{ 1570, 1040, 750, 1570, 750}
07043         ,{ 1570, 1040, 740, 1570, 740}
07044     }
07045     ,{{{ 1870, 1870, 1570, 1560, 1570}
07046         ,{ 130, 130, 70, -180, 70}
07047         ,{ 1560, 1030, 730, 1560, 730}
07048         ,{ 1870, 1870, 1570, 1560, 1570}
07049         ,{ 1560, 1030, 730, 1560, 730}
07050     }
07051     ,{{{ 1570, 1040, 750, 1570, 750}
07052         ,{ 1570, 1040, 750, 1570, 750}
07053         ,{ 1570, 1040, 740, 1570, 740}
07054         ,{ 1570, 1040, 750, 1570, 750}
07055         ,{ 300, 300, -230, -250, -230}
07056     }
07057     }
07058     ,{{{ 1870, 1870, 1570, 130, 1570}
07059         ,{ 1340, 1340, 1040, 130, 1040}
07060         ,{ 1040, 1040, 740, 70, 740}
07061         ,{ 1870, 1870, 1570, -160, 1570}
07062         ,{ 1040, 1040, 740, 70, 740}
07063     }
07064     ,{{{ 1340, 1340, 1040, 130, 1040}
07065         ,{ 1340, 1340, 1040, 130, 1040}
07066         ,{ 1030, 1030, 730, -180, 730}
07067         ,{ -50, -50, -340, -1260, -340}
07068         ,{ 1030, 1030, 730, -180, 730}
07069     }
07070     ,{{{ 1040, 1040, 750, 70, 750}
07071         ,{ 1040, 1040, 750, -160, 750}
07072         ,{ 1040, 1040, 740, 70, 740}
07073         ,{ 1040, 1040, 750, -160, 750}
07074         ,{ 1040, 1040, 740, 70, 740}
07075     }
07076     ,{{{ 1870, 1870, 1570, -180, 1570}
07077         ,{ 130, 130, -160, -1080, -160}
07078         ,{ 1030, 1030, 730, -180, 730}
07079         ,{ 1870, 1870, 1570, -590, 1570}
07080         ,{ 1030, 1030, 730, -180, 730}
07081     }
07082     ,{{{ 1040, 1040, 750, 70, 750}
07083         ,{ 1040, 1040, 750, -160, 750}
07084         ,{ 1040, 1040, 740, 70, 740}
07085         ,{ 1040, 1040, 750, -160, 750}
07086         ,{ 300, 300, -230, -1150, -230}
07087     }
07088     }
07089     ,{{{ 1570, 1560, 1570, 1560, 1570}
07090         ,{ 1040, 1030, 1040, 1030, 1040}
07091         ,{ 740, 730, 740, 730, 740}
07092         ,{ 1570, 1560, 1570, 1560, 1570}
07093         ,{ 740, 730, 740, 730, 740}
07094     }
07095     ,{{{ 1040, 1030, 1040, 1030, 1040}
07096         ,{ 1040, 1030, 1040, 1030, 1040}
07097         ,{ 730, 720, 730, 720, 730}
07098         ,{ -110, -360, -110, -360, -110}
07099         ,{ 730, 720, 730, 720, 730}
07100     }
07101     ,{{{ 740, 730, 740, 730, 740}
07102         ,{ 740, 730, 740, 730, 740}
07103         ,{ 740, 730, 740, 730, 740}
07104         ,{ 740, 730, 740, 730, 740}
07105         ,{ 740, 730, 740, 730, 740}
07106     }
07107     ,{{{ 1570, 1560, 1570, 1560, 1570}
07108         ,{ 70, -180, 70, -180, 70}
07109         ,{ 730, 720, 730, 720, 730}
07110         ,{ 1570, 1560, 1570, 1560, 1570}
07111         ,{ 730, 720, 730, 720, 730}
07112     }
```

```
07113 ,{{ 740, 730, 740, 730, 740}
07114 ,{ 740, 730, 740, 730, 740}
07115 ,{ 740, 730, 740, 730, 740}
07116 ,{ 740, 730, 740, 730, 740}
07117 ,{ -240, -250, -240, -250, -240}
07118 }
07119 }
07120 ,{{{ 1870, -50, 1570, 1870, 1570}
07121 ,{ 1870, -50, 1040, 1870, 1040}
07122 ,{ 1570, -110, 740, 1570, 740}
07123 ,{ 1570, -340, 1570, 1570, 1570}
07124 ,{ 1570, -110, 740, 1570, 740}
07125 }
07126 ,{{{ 1870, -50, 1040, 1870, 1040}
07127 ,{ 1870, -50, 1040, 1870, 1040}
07128 ,{ 1560, -360, 730, 1560, 730}
07129 ,{ -340, -1440, -340, -770, -340}
07130 ,{ 1560, -360, 730, 1560, 730}
07131 }
07132 ,{{{ 1570, -110, 750, 1570, 750}
07133 ,{ 1570, -340, 750, 1570, 750}
07134 ,{ 1570, -110, 740, 1570, 740}
07135 ,{ 1570, -340, 750, 1570, 750}
07136 ,{ 1570, -110, 740, 1570, 740}
07137 }
07138 ,{{{ 1570, -360, 1570, 1560, 1570}
07139 ,{ -160, -1260, -160, -590, -160}
07140 ,{ 1560, -360, 730, 1560, 730}
07141 ,{ 1570, -770, 1570, -100, 1570}
07142 ,{ 1560, -360, 730, 1560, 730}
07143 }
07144 ,{{{ 1570, -110, 750, 1570, 750}
07145 ,{ 1570, -340, 750, 1570, 750}
07146 ,{ 1570, -110, 740, 1570, 740}
07147 ,{ 1570, -340, 750, 1570, 750}
07148 ,{ -230, -1330, -230, -660, -230}
07149 }
07150 }
07151 ,{{{ 1570, 1560, 1570, 1560, 300}
07152 ,{ 1040, 1030, 1040, 1030, 300}
07153 ,{ 740, 730, 740, 730, -240}
07154 ,{ 1570, 1560, 1570, 1560, -230}
07155 ,{ 740, 730, 740, 730, -240}
07156 }
07157 ,{{{ 1040, 1030, 1040, 1030, 300}
07158 ,{ 1040, 1030, 1040, 1030, 300}
07159 ,{ 730, 720, 730, 720, -250}
07160 ,{ -110, -360, -110, -360, -1330}
07161 ,{ 730, 720, 730, 720, -250}
07162 }
07163 ,{{{ 740, 730, 740, 730, -230}
07164 ,{ 740, 730, 740, 730, -230}
07165 ,{ 740, 730, 740, 730, -240}
07166 ,{ 740, 730, 740, 730, -230}
07167 ,{ 740, 730, 740, 730, -240}
07168 }
07169 ,{{{ 1570, 1560, 1570, 1560, -250}
07170 ,{ 70, -180, 70, -180, -1150}
07171 ,{ 730, 720, 730, 720, -250}
07172 ,{ 1570, 1560, 1570, 1560, -660}
07173 ,{ 730, 720, 730, 720, -250}
07174 }
07175 ,{{{ 740, 730, 740, 730, -230}
07176 ,{ 740, 730, 740, 730, -230}
07177 ,{ 740, 730, 740, 730, -240}
07178 ,{ 740, 730, 740, 730, -230}
07179 ,{ -240, -250, -240, -250, -1220}
07180 }
07181 }
07182 }
07183 ,{{{ 2050, 2050, 1760, 1930, 1760}
07184 ,{ 1930, 1400, 1110, 1930, 1110}
07185 ,{ 1800, 1270, 980, 1800, 980}
07186 ,{ 2050, 2050, 1760, 1800, 1760}
07187 ,{ 1670, 1140, 850, 1670, 850}
07188 }
07189 ,{{{ 1930, 1400, 1110, 1930, 1110}
07190 ,{ 1930, 1400, 1110, 1930, 1110}
07191 ,{ 1650, 1120, 830, 1650, 830}
07192 ,{ 0, 0, -60, -310, -60}
07193 ,{ 1650, 1120, 830, 1650, 830}
07194 }
07195 ,{{{ 1800, 1270, 980, 1800, 980}
07196 ,{ 1800, 1270, 980, 1800, 980}
07197 ,{ 1800, 1270, 980, 1800, 980}
07198 ,{ 1800, 1270, 980, 1800, 980}
07199 ,{ 1670, 1140, 850, 1670, 850}
```

```
07200     }
07201     ,{{ 2050, 2050, 1760, 1740, 1760}
07202     ,{ -300, -300, -360, -610, -360}
07203     ,{ 1650, 1120, 830, 1650, 830}
07204     ,{ 2050, 2050, 1760, 1740, 1760}
07205     ,{ 1650, 1120, 830, 1650, 830}
07206     }
07207     ,{{ 1800, 1270, 980, 1800, 980}
07208     ,{ 1800, 1270, 980, 1800, 980}
07209     ,{ 1360, 830, 540, 1360, 540}
07210     ,{ 1800, 1270, 980, 1800, 980}
07211     ,{ 570, 570, 40, 20, 40}
07212     }
07213     }
07214     ,{{{ 2050, 2050, 1760, 300, 1760}
07215     ,{ 1400, 1400, 1110, 190, 1110}
07216     ,{ 1270, 1270, 980, 300, 980}
07217     ,{ 2050, 2050, 1760, 60, 1760}
07218     ,{ 1140, 1140, 850, 180, 850}
07219     }
07220     ,{{{ 1400, 1400, 1110, 190, 1110}
07221     ,{ 1400, 1400, 1110, 190, 1110}
07222     ,{ 1120, 1120, 830, -80, 830}
07223     ,{ 0, 0, -290, -1210, -290}
07224     ,{ 1120, 1120, 830, -80, 830}
07225     }
07226     ,{{{ 1270, 1270, 980, 300, 980}
07227     ,{ 1270, 1270, 980, 60, 980}
07228     ,{ 1270, 1270, 980, 300, 980}
07229     ,{ 1270, 1270, 980, 60, 980}
07230     ,{ 1140, 1140, 850, 180, 850}
07231     }
07232     ,{{{ 2050, 2050, 1760, -80, 1760}
07233     ,{ -300, -300, -590, -1510, -590}
07234     ,{ 1120, 1120, 830, -80, 830}
07235     ,{ 2050, 2050, 1760, -400, 1760}
07236     ,{ 1120, 1120, 830, -80, 830}
07237     }
07238     ,{{{ 1270, 1270, 980, 60, 980}
07239     ,{ 1270, 1270, 980, 60, 980}
07240     ,{ 830, 830, 540, -130, 540}
07241     ,{ 1270, 1270, 980, 60, 980}
07242     ,{ 570, 570, 40, -870, 40}
07243     }
07244     }
07245     ,{{{ 1750, 1740, 1750, 1740, 1750}
07246     ,{ 1100, 1090, 1100, 1090, 1100}
07247     ,{ 970, 960, 970, 960, 970}
07248     ,{ 1750, 1740, 1750, 1740, 1750}
07249     ,{ 840, 830, 840, 830, 840}
07250     }
07251     ,{{{ 1100, 1090, 1100, 1090, 1100}
07252     ,{ 1100, 1090, 1100, 1090, 1100}
07253     ,{ 820, 810, 820, 810, 820}
07254     ,{ -60, -310, -60, -310, -60}
07255     ,{ 820, 810, 820, 810, 820}
07256     }
07257     ,{{{ 970, 960, 970, 960, 970}
07258     ,{ 970, 960, 970, 960, 970}
07259     ,{ 970, 960, 970, 960, 970}
07260     ,{ 970, 960, 970, 960, 970}
07261     ,{ 840, 830, 840, 830, 840}
07262     }
07263     ,{{{ 1750, 1740, 1750, 1740, 1750}
07264     ,{ -360, -610, -360, -610, -360}
07265     ,{ 820, 810, 820, 810, 820}
07266     ,{ 1750, 1740, 1750, 1740, 1750}
07267     ,{ 820, 810, 820, 810, 820}
07268     }
07269     ,{{{ 970, 960, 970, 960, 970}
07270     ,{ 970, 960, 970, 960, 970}
07271     ,{ 530, 520, 530, 520, 530}
07272     ,{ 970, 960, 970, 960, 970}
07273     ,{ 30, 20, 30, 20, 30}
07274     }
07275     }
07276     ,{{{ 1930, 130, 1760, 1930, 1760}
07277     ,{ 1930, 10, 1110, 1930, 1110}
07278     ,{ 1800, 130, 980, 1800, 980}
07279     ,{ 1800, -110, 1760, 1800, 1760}
07280     ,{ 1670, 0, 850, 1670, 850}
07281     }
07282     ,{{{ 1930, 10, 1110, 1930, 1110}
07283     ,{ 1930, 10, 1110, 1930, 1110}
07284     ,{ 1650, -260, 830, 1650, 830}
07285     ,{ -290, -1390, -290, -720, -290}
07286     ,{ 1650, -260, 830, 1650, 830}
```

```

07287     }
07288     ,{{ 1800, 130, 980, 1800, 980}
07289     ,{ 1800, -110, 980, 1800, 980}
07290     ,{ 1800, 130, 980, 1800, 980}
07291     ,{ 1800, -110, 980, 1800, 980}
07292     ,{ 1670, 0, 850, 1670, 850}
07293     }
07294     ,{{ 1760, -260, 1760, 1650, 1760}
07295     ,{ -590, -1690, -590, -1020, -590}
07296     ,{ 1650, -260, 830, 1650, 830}
07297     ,{ 1760, -580, 1760, 80, 1760}
07298     ,{ 1650, -260, 830, 1650, 830}
07299     }
07300     ,{{ 1800, -110, 980, 1800, 980}
07301     ,{ 1800, -110, 980, 1800, 980}
07302     ,{ 1360, -310, 540, 1360, 540}
07303     ,{ 1800, -110, 980, 1800, 980}
07304     ,{ 40, -1050, 40, -380, 40}
07305     }
07306     }
07307     ,{{{ 1750, 1740, 1750, 1740, 360}
07308     ,{ 1100, 1090, 1100, 1090, 360}
07309     ,{ 970, 960, 970, 960, 0}
07310     ,{ 1750, 1740, 1750, 1740, 0}
07311     ,{ 840, 830, 840, 830, -130}
07312     }
07313     ,{{{ 1100, 1090, 1100, 1090, 360}
07314     ,{ 1100, 1090, 1100, 1090, 360}
07315     ,{ 820, 810, 820, 810, -150}
07316     ,{ -60, -310, -60, -310, -1280}
07317     ,{ 820, 810, 820, 810, -150}
07318     }
07319     ,{{{ 970, 960, 970, 960, 0}
07320     ,{ 970, 960, 970, 960, 0}
07321     ,{ 970, 960, 970, 960, 0}
07322     ,{ 970, 960, 970, 960, 0}
07323     ,{ 840, 830, 840, 830, -130}
07324     }
07325     ,{{{ 1750, 1740, 1750, 1740, -150}
07326     ,{ -360, -610, -360, -610, -1580}
07327     ,{ 820, 810, 820, 810, -150}
07328     ,{ 1750, 1740, 1750, 1740, -470}
07329     ,{ 820, 810, 820, 810, -150}
07330     }
07331     ,{{{ 970, 960, 970, 960, 0}
07332     ,{ 970, 960, 970, 960, 0}
07333     ,{ 530, 520, 530, 520, -440}
07334     ,{ 970, 960, 970, 960, 0}
07335     ,{ 30, 20, 30, 20, -940}
07336     }
07337     }
07338     }
07339     ,{{{ 2050, 2050, 1760, 1930, 1760}
07340     ,{ 1930, 1400, 1110, 1930, 1110}
07341     ,{ 1800, 1270, 980, 1800, 980}
07342     ,{ 2050, 2050, 1760, 1800, 1760}
07343     ,{ 1670, 1140, 850, 1670, 850}
07344     }
07345     ,{{{ 1930, 1400, 1110, 1930, 1110}
07346     ,{ 1930, 1400, 1110, 1930, 1110}
07347     ,{ 1650, 1120, 830, 1650, 830}
07348     ,{ 220, 220, 170, -80, 170}
07349     ,{ 1650, 1120, 830, 1650, 830}
07350     }
07351     ,{{{ 1800, 1270, 980, 1800, 980}
07352     ,{ 1800, 1270, 980, 1800, 980}
07353     ,{ 1800, 1270, 980, 1800, 980}
07354     ,{ 1800, 1270, 980, 1800, 980}
07355     ,{ 1670, 1140, 850, 1670, 850}
07356     }
07357     ,{{{ 2050, 2050, 1760, 1740, 1760}
07358     ,{ 130, 130, 70, -180, 70}
07359     ,{ 1650, 1120, 830, 1650, 830}
07360     ,{ 2050, 2050, 1760, 1740, 1760}
07361     ,{ 1650, 1120, 830, 1650, 830}
07362     }
07363     ,{{{ 1800, 1270, 980, 1800, 980}
07364     ,{ 1800, 1270, 980, 1800, 980}
07365     ,{ 1570, 1040, 740, 1570, 740}
07366     ,{ 1800, 1270, 980, 1800, 980}
07367     ,{ 570, 570, 40, 20, 40}
07368     }
07369     }
07370     ,{{{ 2050, 2050, 1760, 300, 1760}
07371     ,{ 1400, 1400, 1110, 190, 1110}
07372     ,{ 1270, 1270, 980, 300, 980}
07373     ,{ 2050, 2050, 1760, 60, 1760}

```



```
07374 , { 1140, 1140, 850, 180, 850}
07375 }
07376 , { { 1400, 1400, 1110, 190, 1110}
07377 , { 1400, 1400, 1110, 190, 1110}
07378 , { 1120, 1120, 830, -80, 830}
07379 , { 220, 220, -70, -980, -70}
07380 , { 1120, 1120, 830, -80, 830}
07381 }
07382 , { { 1270, 1270, 980, 300, 980}
07383 , { 1270, 1270, 980, 60, 980}
07384 , { 1270, 1270, 980, 300, 980}
07385 , { 1270, 1270, 980, 60, 980}
07386 , { 1140, 1140, 850, 180, 850}
07387 }
07388 , { { 2050, 2050, 1760, -80, 1760}
07389 , { 130, 130, -160, -1080, -160}
07390 , { 1120, 1120, 830, -80, 830}
07391 , { 2050, 2050, 1760, -400, 1760}
07392 , { 1120, 1120, 830, -80, 830}
07393 }
07394 , { { 1270, 1270, 980, 70, 980}
07395 , { 1270, 1270, 980, 60, 980}
07396 , { 1040, 1040, 740, 70, 740}
07397 , { 1270, 1270, 980, 60, 980}
07398 , { 570, 570, 40, -870, 40}
07399 }
07400 }
07401 , { { { 1750, 1740, 1750, 1740, 1750}
07402 , { 1100, 1090, 1100, 1090, 1100}
07403 , { 970, 960, 970, 960, 970}
07404 , { 1750, 1740, 1750, 1740, 1750}
07405 , { 840, 830, 840, 830, 840}
07406 }
07407 , { { 1100, 1090, 1100, 1090, 1100}
07408 , { 1100, 1090, 1100, 1090, 1100}
07409 , { 820, 810, 820, 810, 820}
07410 , { 170, -80, 170, -80, 170}
07411 , { 820, 810, 820, 810, 820}
07412 }
07413 , { { 970, 960, 970, 960, 970}
07414 , { 970, 960, 970, 960, 970}
07415 , { 970, 960, 970, 960, 970}
07416 , { 970, 960, 970, 960, 970}
07417 , { 840, 830, 840, 830, 840}
07418 }
07419 , { { 1750, 1740, 1750, 1740, 1750}
07420 , { 70, -180, 70, -180, 70}
07421 , { 820, 810, 820, 810, 820}
07422 , { 1750, 1740, 1750, 1740, 1750}
07423 , { 820, 810, 820, 810, 820}
07424 }
07425 , { { 970, 960, 970, 960, 970}
07426 , { 970, 960, 970, 960, 970}
07427 , { 740, 730, 740, 730, 740}
07428 , { 970, 960, 970, 960, 970}
07429 , { 30, 20, 30, 20, 30}
07430 }
07431 }
07432 , { { { 1930, 130, 1760, 1930, 1760}
07433 , { 1930, 10, 1110, 1930, 1110}
07434 , { 1800, 130, 980, 1800, 980}
07435 , { 1800, -110, 1760, 1800, 1760}
07436 , { 1670, 0, 850, 1670, 850}
07437 }
07438 , { { 1930, 10, 1110, 1930, 1110}
07439 , { 1930, 10, 1110, 1930, 1110}
07440 , { 1650, -260, 830, 1650, 830}
07441 , { -70, -1160, -70, -490, -70}
07442 , { 1650, -260, 830, 1650, 830}
07443 }
07444 , { { 1800, 130, 980, 1800, 980}
07445 , { 1800, -110, 980, 1800, 980}
07446 , { 1800, 130, 980, 1800, 980}
07447 , { 1800, -110, 980, 1800, 980}
07448 , { 1670, 0, 850, 1670, 850}
07449 }
07450 , { { 1760, -260, 1760, 1650, 1760}
07451 , { -160, -1260, -160, -590, -160}
07452 , { 1650, -260, 830, 1650, 830}
07453 , { 1760, -580, 1760, 80, 1760}
07454 , { 1650, -260, 830, 1650, 830}
07455 }
07456 , { { 1800, -110, 980, 1800, 980}
07457 , { 1800, -110, 980, 1800, 980}
07458 , { 1570, -110, 740, 1570, 740}
07459 , { 1800, -110, 980, 1800, 980}
07460 , { 40, -1050, 40, -380, 40}
```

```
07461     }
07462     }
07463     ,{{{ 1750, 1740, 1750, 1740, 360}
07464     ,{ 1100, 1090, 1100, 1090, 360}
07465     ,{ 970, 960, 970, 960, 0}
07466     ,{ 1750, 1740, 1750, 1740, 0}
07467     ,{ 840, 830, 840, 830, -130}
07468     }
07469     ,{{{ 1100, 1090, 1100, 1090, 360}
07470     ,{ 1100, 1090, 1100, 1090, 360}
07471     ,{ 820, 810, 820, 810, -150}
07472     ,{ 170, -80, 170, -80, -1050}
07473     ,{ 820, 810, 820, 810, -150}
07474     }
07475     ,{{{ 970, 960, 970, 960, 0}
07476     ,{ 970, 960, 970, 960, 0}
07477     ,{ 970, 960, 970, 960, 0}
07478     ,{ 970, 960, 970, 960, 0}
07479     ,{ 840, 830, 840, 830, -130}
07480     }
07481     ,{{{ 1750, 1740, 1750, 1740, -150}
07482     ,{ 70, -180, 70, -180, -1150}
07483     ,{ 820, 810, 820, 810, -150}
07484     ,{ 1750, 1740, 1750, 1740, -470}
07485     ,{ 820, 810, 820, 810, -150}
07486     }
07487     ,{{{ 970, 960, 970, 960, 0}
07488     ,{ 970, 960, 970, 960, 0}
07489     ,{ 740, 730, 740, 730, -240}
07490     ,{ 970, 960, 970, 960, 0}
07491     ,{ 30, 20, 30, 20, -940}
07492     }
07493     }
07494     }
07495     }
07496     ,{{{ INF, INF, INF, INF, INF}
07497     ,{ INF, INF, INF, INF, INF}
07498     ,{ INF, INF, INF, INF, INF}
07499     ,{ INF, INF, INF, INF, INF}
07500     ,{ INF, INF, INF, INF, INF}
07501     }
07502     ,{{{ INF, INF, INF, INF, INF}
07503     ,{ INF, INF, INF, INF, INF}
07504     ,{ INF, INF, INF, INF, INF}
07505     ,{ INF, INF, INF, INF, INF}
07506     ,{ INF, INF, INF, INF, INF}
07507     }
07508     ,{{{ INF, INF, INF, INF, INF}
07509     ,{ INF, INF, INF, INF, INF}
07510     ,{ INF, INF, INF, INF, INF}
07511     ,{ INF, INF, INF, INF, INF}
07512     ,{ INF, INF, INF, INF, INF}
07513     }
07514     ,{{{ INF, INF, INF, INF, INF}
07515     ,{ INF, INF, INF, INF, INF}
07516     ,{ INF, INF, INF, INF, INF}
07517     ,{ INF, INF, INF, INF, INF}
07518     ,{ INF, INF, INF, INF, INF}
07519     }
07520     ,{{{ INF, INF, INF, INF, INF}
07521     ,{ INF, INF, INF, INF, INF}
07522     ,{ INF, INF, INF, INF, INF}
07523     ,{ INF, INF, INF, INF, INF}
07524     ,{ INF, INF, INF, INF, INF}
07525     }
07526     }
07527     ,{{{ INF, INF, INF, INF, INF}
07528     ,{ INF, INF, INF, INF, INF}
07529     ,{ INF, INF, INF, INF, INF}
07530     ,{ INF, INF, INF, INF, INF}
07531     ,{ INF, INF, INF, INF, INF}
07532     }
07533     ,{{{ INF, INF, INF, INF, INF}
07534     ,{ INF, INF, INF, INF, INF}
07535     ,{ INF, INF, INF, INF, INF}
07536     ,{ INF, INF, INF, INF, INF}
07537     ,{ INF, INF, INF, INF, INF}
07538     }
07539     ,{{{ INF, INF, INF, INF, INF}
07540     ,{ INF, INF, INF, INF, INF}
07541     ,{ INF, INF, INF, INF, INF}
07542     ,{ INF, INF, INF, INF, INF}
07543     ,{ INF, INF, INF, INF, INF}
07544     }
07545     ,{{{ INF, INF, INF, INF, INF}
07546     ,{ INF, INF, INF, INF, INF}
07547     ,{ INF, INF, INF, INF, INF}
```

```
07548     , {   INF,   INF,   INF,   INF,   INF }
07549     , {   INF,   INF,   INF,   INF,   INF }
07550     }
07551     , { {   INF,   INF,   INF,   INF,   INF }
07552     , {   INF,   INF,   INF,   INF,   INF }
07553     , {   INF,   INF,   INF,   INF,   INF }
07554     , {   INF,   INF,   INF,   INF,   INF }
07555     , {   INF,   INF,   INF,   INF,   INF }
07556     }
07557     }
07558     , { { {   INF,   INF,   INF,   INF,   INF }
07559     , {   INF,   INF,   INF,   INF,   INF }
07560     , {   INF,   INF,   INF,   INF,   INF }
07561     , {   INF,   INF,   INF,   INF,   INF }
07562     , {   INF,   INF,   INF,   INF,   INF }
07563     }
07564     , { {   INF,   INF,   INF,   INF,   INF }
07565     , {   INF,   INF,   INF,   INF,   INF }
07566     , {   INF,   INF,   INF,   INF,   INF }
07567     , {   INF,   INF,   INF,   INF,   INF }
07568     , {   INF,   INF,   INF,   INF,   INF }
07569     }
07570     , { {   INF,   INF,   INF,   INF,   INF }
07571     , {   INF,   INF,   INF,   INF,   INF }
07572     , {   INF,   INF,   INF,   INF,   INF }
07573     , {   INF,   INF,   INF,   INF,   INF }
07574     , {   INF,   INF,   INF,   INF,   INF }
07575     }
07576     , { {   INF,   INF,   INF,   INF,   INF }
07577     , {   INF,   INF,   INF,   INF,   INF }
07578     , {   INF,   INF,   INF,   INF,   INF }
07579     , {   INF,   INF,   INF,   INF,   INF }
07580     , {   INF,   INF,   INF,   INF,   INF }
07581     }
07582     , { {   INF,   INF,   INF,   INF,   INF }
07583     , {   INF,   INF,   INF,   INF,   INF }
07584     , {   INF,   INF,   INF,   INF,   INF }
07585     , {   INF,   INF,   INF,   INF,   INF }
07586     , {   INF,   INF,   INF,   INF,   INF }
07587     }
07588     }
07589     , { { {   INF,   INF,   INF,   INF,   INF }
07590     , {   INF,   INF,   INF,   INF,   INF }
07591     , {   INF,   INF,   INF,   INF,   INF }
07592     , {   INF,   INF,   INF,   INF,   INF }
07593     , {   INF,   INF,   INF,   INF,   INF }
07594     }
07595     , { { {   INF,   INF,   INF,   INF,   INF }
07596     , {   INF,   INF,   INF,   INF,   INF }
07597     , {   INF,   INF,   INF,   INF,   INF }
07598     , {   INF,   INF,   INF,   INF,   INF }
07599     , {   INF,   INF,   INF,   INF,   INF }
07600     }
07601     , { { {   INF,   INF,   INF,   INF,   INF }
07602     , {   INF,   INF,   INF,   INF,   INF }
07603     , {   INF,   INF,   INF,   INF,   INF }
07604     , {   INF,   INF,   INF,   INF,   INF }
07605     , {   INF,   INF,   INF,   INF,   INF }
07606     }
07607     , { { {   INF,   INF,   INF,   INF,   INF }
07608     , {   INF,   INF,   INF,   INF,   INF }
07609     , {   INF,   INF,   INF,   INF,   INF }
07610     , {   INF,   INF,   INF,   INF,   INF }
07611     , {   INF,   INF,   INF,   INF,   INF }
07612     }
07613     , { { {   INF,   INF,   INF,   INF,   INF }
07614     , {   INF,   INF,   INF,   INF,   INF }
07615     , {   INF,   INF,   INF,   INF,   INF }
07616     , {   INF,   INF,   INF,   INF,   INF }
07617     , {   INF,   INF,   INF,   INF,   INF }
07618     }
07619     }
07620     , { { {   INF,   INF,   INF,   INF,   INF }
07621     , {   INF,   INF,   INF,   INF,   INF }
07622     , {   INF,   INF,   INF,   INF,   INF }
07623     , {   INF,   INF,   INF,   INF,   INF }
07624     , {   INF,   INF,   INF,   INF,   INF }
07625     }
07626     , { { {   INF,   INF,   INF,   INF,   INF }
07627     , {   INF,   INF,   INF,   INF,   INF }
07628     , {   INF,   INF,   INF,   INF,   INF }
07629     , {   INF,   INF,   INF,   INF,   INF }
07630     , {   INF,   INF,   INF,   INF,   INF }
07631     }
07632     , { { {   INF,   INF,   INF,   INF,   INF }
07633     , {   INF,   INF,   INF,   INF,   INF }
07634     , {   INF,   INF,   INF,   INF,   INF }
```

```

07635     , {   INF,   INF,   INF,   INF,   INF }
07636     , {   INF,   INF,   INF,   INF,   INF }
07637     }
07638     , { {   INF,   INF,   INF,   INF,   INF }
07639     , {   INF,   INF,   INF,   INF,   INF }
07640     , {   INF,   INF,   INF,   INF,   INF }
07641     , {   INF,   INF,   INF,   INF,   INF }
07642     , {   INF,   INF,   INF,   INF,   INF }
07643     }
07644     , { {   INF,   INF,   INF,   INF,   INF }
07645     , {   INF,   INF,   INF,   INF,   INF }
07646     , {   INF,   INF,   INF,   INF,   INF }
07647     , {   INF,   INF,   INF,   INF,   INF }
07648     , {   INF,   INF,   INF,   INF,   INF }
07649     }
07650     }
07651     }
07652     , { { { 1350,   850,   720, 1350,   720 }
07653     , { 1300,   650,   520, 1300,   520 }
07654     , { 1350,   700,   570, 1350,   570 }
07655     , { 1300,   850,   720, 1300,   720 }
07656     , { 1250,   590,   460, 1250,   460 }
07657     }
07658     , { { 1160,   500,   400, 1160,   370 }
07659     , { 1160,   500,   370, 1160,   370 }
07660     , {   850,   190,    60,   850,    60 }
07661     , {   400,   290,   400,    10,   160 }
07662     , {   850,   190,    60,   850,    60 }
07663     }
07664     , { { 1300,   650,   520, 1300,   520 }
07665     , { 1300,   650,   520, 1300,   520 }
07666     , { 1290,   640,   510, 1290,   510 }
07667     , { 1300,   650,   520, 1300,   520 }
07668     , { 1250,   590,   460, 1250,   460 }
07669     }
07670     , { {   850,   850,   720,   850,   720 }
07671     , {   120,    0,   120,  -270,  -120 }
07672     , {   850,   190,    60,   850,    60 }
07673     , {   850,   850,   720,   570,   720 }
07674     , {   850,   190,    60,   850,    60 }
07675     }
07676     , { { 1350,   700,   570, 1350,   570 }
07677     , { 1300,   650,   520, 1300,   520 }
07678     , { 1350,   700,   570, 1350,   570 }
07679     , { 1300,   650,   520, 1300,   520 }
07680     , {   100,   100,  -270,  -420,  -270 }
07681     }
07682     }
07683     , { { {   850,   850,   720,  -760,   720 }
07684     , {   650,   650,   520, -1050,   520 }
07685     , {   700,   700,   570,  -760,   570 }
07686     , {   850,   850,   720, -1050,   720 }
07687     , {   590,   590,   460,  -870,   460 }
07688     }
07689     , { {   500,   500,   370, -1200,   370 }
07690     , {   500,   500,   370, -1200,   370 }
07691     , {   190,   190,    60, -1510,    60 }
07692     , {   290,   290,   160, -1410,   160 }
07693     , {   190,   190,    60, -1510,    60 }
07694     }
07695     , { {   650,   650,   520,  -820,   520 }
07696     , {   650,   650,   520, -1050,   520 }
07697     , {   640,   640,   510,  -820,   510 }
07698     , {   650,   650,   520, -1050,   520 }
07699     , {   590,   590,   460,  -870,   460 }
07700     }
07701     , { {   850,   850,   720, -1510,   720 }
07702     , {    0,    0,  -120, -1700,  -120 }
07703     , {   190,   190,    60, -1510,    60 }
07704     , {   850,   850,   720, -2110,   720 }
07705     , {   190,   190,    60, -1510,    60 }
07706     }
07707     , { {   700,   700,   570,  -760,   570 }
07708     , {   650,   650,   520, -1050,   520 }
07709     , {   700,   700,   570,  -760,   570 }
07710     , {   650,   650,   520, -1050,   520 }
07711     , {   100,   100,  -270, -1840,  -270 }
07712     }
07713     }
07714     , { { { 720,   570,   720,   570,   280 }
07715     , { 520,   370,   520,   370,    80 }
07716     , { 570,   420,   570,   420,   130 }
07717     , { 720,   570,   720,   570,   280 }
07718     , { 460,   310,   460,   310,    20 }
07719     }
07720     , { { 400,   220,   400,   220,   -40 }
07721     , { 370,   220,   370,   220,   -60 }

```

```
07722 , { 60, -80, 60, -80, -370}
07723 , { 400, 10, 400, 10, -40}
07724 , { 60, -80, 60, -80, -370}
07725 }
07726 , { { 520, 370, 520, 370, 80}
07727 , { 520, 370, 520, 370, 80}
07728 , { 510, 360, 510, 360, 70}
07729 , { 520, 370, 520, 370, 80}
07730 , { 460, 310, 460, 310, 20}
07731 }
07732 , { { 720, 570, 720, 570, 280}
07733 , { 120, -270, 120, -270, -320}
07734 , { 60, -80, 60, -80, -370}
07735 , { 720, 570, 720, 570, 280}
07736 , { 60, -80, 60, -80, -370}
07737 }
07738 , { { 570, 420, 570, 420, 130}
07739 , { 520, 370, 520, 370, 80}
07740 , { 570, 420, 570, 420, 130}
07741 , { 520, 370, 520, 370, 80}
07742 , { -270, -420, -270, -420, -710}
07743 }
07744 }
07745 , { { { 1350, -460, 720, 1350, 720}
07746 , { 1300, -750, 520, 1300, 520}
07747 , { 1350, -460, 570, 1350, 570}
07748 , { 1300, -750, 720, 1300, 720}
07749 , { 1250, -570, 460, 1250, 460}
07750 }
07751 , { { 1160, -900, 370, 1160, 370}
07752 , { 1160, -900, 370, 1160, 370}
07753 , { 850, -1210, 60, 850, 60}
07754 , { 160, -1110, 160, -310, 160}
07755 , { 850, -1210, 60, 850, 60}
07756 }
07757 , { { 1300, -520, 520, 1300, 520}
07758 , { 1300, -750, 520, 1300, 520}
07759 , { 1290, -520, 510, 1290, 510}
07760 , { 1300, -750, 520, 1300, 520}
07761 , { 1250, -570, 460, 1250, 460}
07762 }
07763 , { { 850, -1210, 720, 850, 720}
07764 , { -120, -1400, -120, -590, -120}
07765 , { 850, -1210, 60, 850, 60}
07766 , { 720, -1810, 720, -1000, 720}
07767 , { 850, -1210, 60, 850, 60}
07768 }
07769 , { { 1350, -460, 570, 1350, 570}
07770 , { 1300, -750, 520, 1300, 520}
07771 , { 1350, -460, 570, 1350, 570}
07772 , { 1300, -750, 520, 1300, 520}
07773 , { -270, -1540, -270, -740, -270}
07774 }
07775 }
07776 , { { { 590, 570, 590, 570, -320}
07777 , { 390, 370, 390, 370, -320}
07778 , { 440, 420, 440, 420, -360}
07779 , { 590, 570, 590, 570, -420}
07780 , { 330, 310, 330, 310, -470}
07781 }
07782 , { { 270, 220, 270, 220, -320}
07783 , { 240, 220, 240, 220, -320}
07784 , { -60, -80, -60, -80, -870}
07785 , { 270, 10, 270, 10, -780}
07786 , { -60, -80, -60, -80, -870}
07787 }
07788 , { { 390, 370, 390, 370, -420}
07789 , { 390, 370, 390, 370, -420}
07790 , { 380, 360, 380, 360, -420}
07791 , { 390, 370, 390, 370, -420}
07792 , { 330, 310, 330, 310, -470}
07793 }
07794 , { { 590, 570, 590, 570, -870}
07795 , { -10, -270, -10, -270, -1060}
07796 , { -60, -80, -60, -80, -870}
07797 , { 590, 570, 590, 570, -1470}
07798 , { -60, -80, -60, -80, -870}
07799 }
07800 , { { 440, 420, 440, 420, -360}
07801 , { 390, 370, 390, 370, -420}
07802 , { 440, 420, 440, 420, -360}
07803 , { 390, 370, 390, 370, -420}
07804 , { -400, -420, -400, -420, -1210}
07805 }
07806 }
07807 }
07808 , { { { { 1320, 850, 720, 1320, 720}
```

```

07809      , { 1320, 670, 540, 1320, 540}
07810      , { 870, 220, 90, 870, 90}
07811      , { 960, 850, 720, 960, 720}
07812      , { 870, 250, 90, 870, 90}
07813      }
07814      , { { 1320, 670, 540, 1320, 540}
07815      , { 1320, 670, 540, 1320, 540}
07816      , { 870, 220, 90, 870, 90}
07817      , { -410, -520, -410, -800, -650}
07818      , { 870, 220, 90, 870, 90}
07819      }
07820      , { { 960, 300, 170, 960, 170}
07821      , { 960, 300, 170, 960, 170}
07822      , { 650, 0, -130, 650, -130}
07823      , { 960, 300, 170, 960, 170}
07824      , { 650, 0, -130, 650, -130}
07825      }
07826      , { { 870, 850, 720, 870, 720}
07827      , { 70, -40, 70, -320, -170}
07828      , { 870, 220, 90, 870, 90}
07829      , { 850, 850, 720, 570, 720}
07830      , { 870, 220, 90, 870, 90}
07831      }
07832      , { { 960, 300, 170, 960, 170}
07833      , { 960, 300, 170, 960, 170}
07834      , { 340, -310, -440, 340, -440}
07835      , { 960, 300, 170, 960, 170}
07836      , { 250, 250, -110, -260, -110}
07837      }
07838      }
07839      , { { { 850, 850, 720, -1030, 720}
07840      , { 670, 670, 540, -1030, 540}
07841      , { 220, 220, 90, -1460, 90}
07842      , { 850, 850, 720, -1400, 720}
07843      , { 250, 250, 90, -1460, 90}
07844      }
07845      , { { 670, 670, 540, -1030, 540}
07846      , { 670, 670, 540, -1030, 540}
07847      , { 220, 220, 90, -1480, 90}
07848      , { -520, -520, -650, -2220, -650}
07849      , { 220, 220, 90, -1480, 90}
07850      }
07851      , { { 300, 300, 170, -1400, 170}
07852      , { 300, 300, 170, -1400, 170}
07853      , { 0, 0, -130, -1460, -130}
07854      , { 300, 300, 170, -1400, 170}
07855      , { 0, 0, -130, -1460, -130}
07856      }
07857      , { { 850, 850, 720, -1480, 720}
07858      , { -40, -40, -170, -1750, -170}
07859      , { 220, 220, 90, -1480, 90}
07860      , { 850, 850, 720, -2110, 720}
07861      , { 220, 220, 90, -1480, 90}
07862      }
07863      , { { 300, 300, 170, -1400, 170}
07864      , { 300, 300, 170, -1400, 170}
07865      , { -310, -310, -440, -1770, -440}
07866      , { 300, 300, 170, -1400, 170}
07867      , { 250, 250, -110, -1690, -110}
07868      }
07869      }
07870      , { { { 720, 570, 720, 570, 280}
07871      , { 540, 390, 540, 390, 100}
07872      , { 90, -60, 90, -60, -350}
07873      , { 720, 570, 720, 570, 280}
07874      , { 90, -60, 90, -60, -350}
07875      }
07876      , { { 540, 390, 540, 390, 100}
07877      , { 540, 390, 540, 390, 100}
07878      , { 90, -60, 90, -60, -350}
07879      , { -410, -800, -410, -800, -850}
07880      , { 90, -60, 90, -60, -350}
07881      }
07882      , { { 170, 20, 170, 20, -260}
07883      , { 170, 20, 170, 20, -260}
07884      , { -130, -280, -130, -280, -570}
07885      , { 170, 20, 170, 20, -260}
07886      , { -130, -280, -130, -280, -570}
07887      }
07888      , { { 720, 570, 720, 570, 280}
07889      , { 70, -320, 70, -320, -370}
07890      , { 90, -60, 90, -60, -350}
07891      , { 720, 570, 720, 570, 280}
07892      , { 90, -60, 90, -60, -350}
07893      }
07894      , { { 170, 20, 170, 20, -260}
07895      , { 170, 20, 170, 20, -260}

```

```
07896     , { -440, -590, -440, -590, -880}
07897     , {  170,   20,  170,   20, -260}
07898     , { -110, -260, -110, -260, -550}
07899     }
07900     }
07901     , { { 1320, -730,  720, 1320,  720}
07902     , { 1320, -730,  540, 1320,  540}
07903     , {  870, -1160,   90,  870,   90}
07904     , {  960, -1100,  720,  960,  720}
07905     , {  870, -1160,   90,  870,   90}
07906     }
07907     , { { 1320, -730,  540, 1320,  540}
07908     , { 1320, -730,  540, 1320,  540}
07909     , {  870, -1180,   90,  870,   90}
07910     , { -650, -1920, -650, -1120, -650}
07911     , {  870, -1180,   90,  870,   90}
07912     }
07913     , { {  960, -1100,  170,  960,  170}
07914     , {  960, -1100,  170,  960,  170}
07915     , {  650, -1160, -130,  650, -130}
07916     , {  960, -1100,  170,  960,  170}
07917     , {  650, -1160, -130,  650, -130}
07918     }
07919     , { {  870, -1180,  720,  870,  720}
07920     , { -170, -1450, -170, -640, -170}
07921     , {  870, -1180,   90,  870,   90}
07922     , {  720, -1810,  720, -1000,  720}
07923     , {  870, -1180,   90,  870,   90}
07924     }
07925     , { {  960, -1100,  170,  960,  170}
07926     , {  960, -1100,  170,  960,  170}
07927     , {  340, -1470, -440,  340, -440}
07928     , {  960, -1100,  170,  960,  170}
07929     , { -110, -1390, -110, -580, -110}
07930     }
07931     }
07932     , { { {  590,  570,  590,  570, -160}
07933     , {  410,  390,  410,  390, -160}
07934     , { -40, -60, -40, -60, -850}
07935     , {  590,  570,  590,  570, -760}
07936     , { -40, -60, -40, -60, -850}
07937     }
07938     , { {  410,  390,  410,  390, -160}
07939     , {  410,  390,  410,  390, -160}
07940     , { -40, -60, -40, -60, -850}
07941     , { -540, -800, -540, -800, -1590}
07942     , { -40, -60, -40, -60, -850}
07943     }
07944     , { {  40,  20,  40,  20, -760}
07945     , {  40,  20,  40,  20, -760}
07946     , { -260, -280, -260, -280, -1070}
07947     , {  40,  20,  40,  20, -760}
07948     , { -260, -280, -260, -280, -1070}
07949     }
07950     , { {  590,  570,  590,  570, -850}
07951     , { -60, -320, -60, -320, -1110}
07952     , { -40, -60, -40, -60, -850}
07953     , {  590,  570,  590,  570, -1470}
07954     , { -40, -60, -40, -60, -850}
07955     }
07956     , { {  40,  20,  40,  20, -760}
07957     , {  40,  20,  40,  20, -760}
07958     , { -570, -590, -570, -590, -1380}
07959     , {  40,  20,  40,  20, -760}
07960     , { -240, -260, -240, -260, -1050}
07961     }
07962     }
07963     }
07964     , { { { { 1010, 1010,  880,  730,  880}
07965     , {  410, -70,  40,  410, -200}
07966     , {  410, -240, -370,  410, -370}
07967     , { 1010, 1010,  880,  730,  880}
07968     , {  410,  0, -370,  410, -370}
07969     }
07970     , { {  410, -240, -150,  410, -370}
07971     , {  230, -420, -550,  230, -550}
07972     , {  410, -240, -370,  410, -370}
07973     , { -150, -260, -150, -540, -390}
07974     , {  410, -240, -370,  410, -370}
07975     }
07976     , { {  410, -240, -370,  410, -370}
07977     , {  410, -240, -370,  410, -370}
07978     , {  410, -240, -370,  410, -370}
07979     , {  410, -240, -370,  410, -370}
07980     , {  410, -240, -370,  410, -370}
07981     }
07982     , { { 1010, 1010,  880,  730,  880}
```

```
07983      , {      40,      -70,      40,    -350,    -200}
07984      , {     410,    -240,    -370,     410,    -370}
07985      , {    1010,    1010,     880,     730,     880}
07986      , {     410,    -240,    -370,     410,    -370}
07987      }
07988      , { {     410,        0,    -370,     410,    -370}
07989      , {     410,    -240,    -370,     410,    -370}
07990      , {     410,    -240,    -370,     410,    -370}
07991      , {     410,    -240,    -370,     410,    -370}
07992      , {        0,        0,    -370,    -520,    -370}
07993      }
07994      }
07995      , { { {    1010,    1010,     880,   -1710,     880}
07996      , {      -70,      -70,    -200,   -1770,    -200}
07997      , {    -240,    -240,    -370,   -1710,    -370}
07998      , {    1010,    1010,     880,   -1950,     880}
07999      , {        0,        0,    -370,   -1710,    -370}
08000      }
08001      , { {    -240,    -240,    -370,   -1950,    -370}
08002      , {    -420,    -420,   -550,   -2130,   -550}
08003      , {    -240,    -240,    -370,   -1950,    -370}
08004      , {    -260,    -260,   -390,   -1960,   -390}
08005      , {    -240,    -240,    -370,   -1950,    -370}
08006      }
08007      , { {    -240,    -240,    -370,   -1710,    -370}
08008      , {    -240,    -240,    -370,   -1950,    -370}
08009      , {    -240,    -240,    -370,   -1710,    -370}
08010      , {    -240,    -240,    -370,   -1950,    -370}
08011      , {    -240,    -240,    -370,   -1710,    -370}
08012      }
08013      , { {    1010,    1010,     880,   -1770,     880}
08014      , {      -70,      -70,    -200,   -1770,    -200}
08015      , {    -240,    -240,    -370,   -1950,    -370}
08016      , {    1010,    1010,     880,   -1950,     880}
08017      , {    -240,    -240,    -370,   -1950,    -370}
08018      }
08019      , { {        0,        0,    -370,   -1710,    -370}
08020      , {    -240,    -240,    -370,   -1950,    -370}
08021      , {    -240,    -240,    -370,   -1710,    -370}
08022      , {    -240,    -240,    -370,   -1950,    -370}
08023      , {        0,        0,    -370,   -1950,    -370}
08024      }
08025      }
08026      , { { {     880,     730,     880,     730,     440}
08027      , {      40,    -350,      40,    -350,    -400}
08028      , {    -370,    -520,    -370,    -520,    -810}
08029      , {     880,     730,     880,     730,     440}
08030      , {    -370,    -520,    -370,    -520,    -810}
08031      }
08032      , { {    -150,    -520,    -150,    -520,    -590}
08033      , {    -550,    -700,    -550,    -700,    -990}
08034      , {    -370,    -520,    -370,    -520,    -810}
08035      , {    -150,    -540,    -150,    -540,    -590}
08036      , {    -370,    -520,    -370,    -520,    -810}
08037      }
08038      , { {    -370,    -520,    -370,    -520,    -810}
08039      , {    -370,    -520,    -370,    -520,    -810}
08040      , {    -370,    -520,    -370,    -520,    -810}
08041      , {    -370,    -520,    -370,    -520,    -810}
08042      , {    -370,    -520,    -370,    -520,    -810}
08043      }
08044      , { { {     880,     730,     880,     730,     440}
08045      , {      40,    -350,      40,    -350,    -400}
08046      , {    -370,    -520,    -370,    -520,    -810}
08047      , {     880,     730,     880,     730,     440}
08048      , {    -370,    -520,    -370,    -520,    -810}
08049      }
08050      , { {    -370,    -520,    -370,    -520,    -810}
08051      , {    -370,    -520,    -370,    -520,    -810}
08052      , {    -370,    -520,    -370,    -520,    -810}
08053      , {    -370,    -520,    -370,    -520,    -810}
08054      , {    -370,    -520,    -370,    -520,    -810}
08055      }
08056      }
08057      , { { {     880,   -1410,     880,     410,     880}
08058      , {     410,   -1470,    -200,     410,    -200}
08059      , {     410,   -1410,    -370,     410,    -370}
08060      , {     880,   -1650,     880,     410,     880}
08061      , {     410,   -1410,    -370,     410,    -370}
08062      }
08063      , { {     410,   -1650,    -370,     410,    -370}
08064      , {     230,   -1830,   -550,     230,   -550}
08065      , {     410,   -1650,    -370,     410,    -370}
08066      , {    -390,   -1660,   -390,    -860,   -390}
08067      , {     410,   -1650,    -370,     410,    -370}
08068      }
08069      , { {     410,   -1410,    -370,     410,    -370}
```



```
08070      , { 410, -1650, -370, 410, -370}
08071      , { 410, -1410, -370, 410, -370}
08072      , { 410, -1650, -370, 410, -370}
08073      , { 410, -1410, -370, 410, -370}
08074      }
08075      , { { 880, -1470, 880, 410, 880}
08076      , { -200, -1470, -200, -670, -200}
08077      , { 410, -1650, -370, 410, -370}
08078      , { 880, -1650, 880, -840, 880}
08079      , { 410, -1650, -370, 410, -370}
08080      }
08081      , { { 410, -1410, -370, 410, -370}
08082      , { 410, -1650, -370, 410, -370}
08083      , { 410, -1410, -370, 410, -370}
08084      , { 410, -1650, -370, 410, -370}
08085      , { -370, -1650, -370, -840, -370}
08086      }
08087      }
08088      , { { { 750, 730, 750, 730, -1140}
08089      , { -90, -350, -90, -350, -1140}
08090      , { -500, -520, -500, -520, -1310}
08091      , { 750, 730, 750, 730, -1310}
08092      , { -500, -520, -500, -520, -1310}
08093      }
08094      , { { -280, -520, -280, -520, -1250}
08095      , { -680, -700, -680, -700, -1250}
08096      , { -500, -520, -500, -520, -1310}
08097      , { -280, -540, -280, -540, -1330}
08098      , { -500, -520, -500, -520, -1310}
08099      }
08100      , { { -500, -520, -500, -520, -1310}
08101      , { -500, -520, -500, -520, -1310}
08102      , { -500, -520, -500, -520, -1310}
08103      , { -500, -520, -500, -520, -1310}
08104      , { -500, -520, -500, -520, -1310}
08105      }
08106      , { { 750, 730, 750, 730, -1140}
08107      , { -90, -350, -90, -350, -1140}
08108      , { -500, -520, -500, -520, -1310}
08109      , { 750, 730, 750, 730, -1310}
08110      , { -500, -520, -500, -520, -1310}
08111      }
08112      , { { -500, -520, -500, -520, -1310}
08113      , { -500, -520, -500, -520, -1310}
08114      , { -500, -520, -500, -520, -1310}
08115      , { -500, -520, -500, -520, -1310}
08116      , { -500, -520, -500, -520, -1310}
08117      }
08118      }
08119      }
08120      , { { { { 1560, 1560, 1430, 1470, 1430}
08121      , { 1470, 820, 690, 1470, 690}
08122      , { 960, 310, 180, 960, 180}
08123      , { 1560, 1560, 1430, 1280, 1430}
08124      , { 960, 550, 180, 960, 180}
08125      }
08126      , { { { 1470, 820, 690, 1470, 690}
08127      , { 1470, 820, 690, 1470, 690}
08128      , { 960, 310, 180, 960, 180}
08129      , { 80, -30, 80, -310, -160}
08130      , { 960, 310, 180, 960, 180}
08131      }
08132      , { { { 960, 310, 180, 960, 180}
08133      , { 960, 310, 180, 960, 180}
08134      , { 960, 310, 180, 960, 180}
08135      , { 960, 310, 180, 960, 180}
08136      , { 960, 310, 180, 960, 180}
08137      }
08138      , { { { 1560, 1560, 1430, 1280, 1430}
08139      , { -90, -200, -90, -480, -330}
08140      , { 960, 310, 180, 960, 180}
08141      , { 1560, 1560, 1430, 1280, 1430}
08142      , { 960, 310, 180, 960, 180}
08143      }
08144      , { { { 960, 550, 180, 960, 180}
08145      , { 960, 310, 180, 960, 180}
08146      , { 960, 310, 180, 960, 180}
08147      , { 960, 310, 180, 960, 180}
08148      , { 550, 550, 180, 30, 180}
08149      }
08150      }
08151      , { { { { 1560, 1560, 1430, -880, 1430}
08152      , { 820, 820, 690, -880, 690}
08153      , { 310, 310, 180, -1150, 180}
08154      , { 1560, 1560, 1430, -1390, 1430}
08155      , { 550, 550, 180, -1150, 180}
08156      }
```

```
08157 ,{{ 820, 820, 690, -880, 690}
08158 ,{ 820, 820, 690, -880, 690}
08159 ,{ 310, 310, 180, -1390, 180}
08160 ,{ -30, -30, -160, -1730, -160}
08161 ,{ 310, 310, 180, -1390, 180}
08162 }
08163 ,{{ 310, 310, 180, -1150, 180}
08164 ,{ 310, 310, 180, -1390, 180}
08165 ,{ 310, 310, 180, -1150, 180}
08166 ,{ 310, 310, 180, -1390, 180}
08167 ,{ 310, 310, 180, -1150, 180}
08168 }
08169 ,{{ 1560, 1560, 1430, -1390, 1430}
08170 ,{ -200, -200, -330, -1900, -330}
08171 ,{ 310, 310, 180, -1390, 180}
08172 ,{ 1560, 1560, 1430, -1390, 1430}
08173 ,{ 310, 310, 180, -1390, 180}
08174 }
08175 ,{{ 550, 550, 180, -1150, 180}
08176 ,{ 310, 310, 180, -1390, 180}
08177 ,{ 310, 310, 180, -1150, 180}
08178 ,{ 310, 310, 180, -1390, 180}
08179 ,{ 550, 550, 180, -1390, 180}
08180 }
08181 }
08182 ,{{{ 1430, 1280, 1430, 1280, 990}
08183 ,{ 690, 540, 690, 540, 250}
08184 ,{ 180, 30, 180, 30, -260}
08185 ,{ 1430, 1280, 1430, 1280, 990}
08186 ,{ 180, 30, 180, 30, -260}
08187 }
08188 ,{{ 690, 540, 690, 540, 250}
08189 ,{ 690, 540, 690, 540, 250}
08190 ,{ 180, 30, 180, 30, -260}
08191 ,{ 80, -310, 80, -310, -360}
08192 ,{ 180, 30, 180, 30, -260}
08193 }
08194 ,{{{ 180, 30, 180, 30, -260}
08195 ,{ 180, 30, 180, 30, -260}
08196 ,{ 180, 30, 180, 30, -260}
08197 ,{ 180, 30, 180, 30, -260}
08198 ,{ 180, 30, 180, 30, -260}
08199 }
08200 ,{{{ 1430, 1280, 1430, 1280, 990}
08201 ,{ -90, -480, -90, -480, -530}
08202 ,{ 180, 30, 180, 30, -260}
08203 ,{ 1430, 1280, 1430, 1280, 990}
08204 ,{ 180, 30, 180, 30, -260}
08205 }
08206 ,{{{ 180, 30, 180, 30, -260}
08207 ,{ 180, 30, 180, 30, -260}
08208 ,{ 180, 30, 180, 30, -260}
08209 ,{ 180, 30, 180, 30, -260}
08210 ,{ 180, 30, 180, 30, -260}
08211 }
08212 }
08213 ,{{{ 1470, -580, 1430, 1470, 1430}
08214 ,{ 1470, -580, 690, 1470, 690}
08215 ,{ 960, -850, 180, 960, 180}
08216 ,{ 1430, -1090, 1430, 960, 1430}
08217 ,{ 960, -850, 180, 960, 180}
08218 }
08219 ,{{{ 1470, -580, 690, 1470, 690}
08220 ,{ 1470, -580, 690, 1470, 690}
08221 ,{ 960, -1090, 180, 960, 180}
08222 ,{ -160, -1430, -160, -630, -160}
08223 ,{ 960, -1090, 180, 960, 180}
08224 }
08225 ,{{{ 960, -850, 180, 960, 180}
08226 ,{ 960, -1090, 180, 960, 180}
08227 ,{ 960, -850, 180, 960, 180}
08228 ,{ 960, -1090, 180, 960, 180}
08229 ,{ 960, -850, 180, 960, 180}
08230 }
08231 ,{{{ 1430, -1090, 1430, 960, 1430}
08232 ,{ -330, -1600, -330, -800, -330}
08233 ,{ 960, -1090, 180, 960, 180}
08234 ,{ 1430, -1090, 1430, -290, 1430}
08235 ,{ 960, -1090, 180, 960, 180}
08236 }
08237 ,{{{ 960, -850, 180, 960, 180}
08238 ,{ 960, -1090, 180, 960, 180}
08239 ,{ 960, -850, 180, 960, 180}
08240 ,{ 960, -1090, 180, 960, 180}
08241 ,{ 180, -1090, 180, -290, 180}
08242 }
08243 }
```

```
08244 ,{{{ 1300, 1280, 1300, 1280, -10}
08245 ,{ 560, 540, 560, 540, -10}
08246 ,{ 50, 30, 50, 30, -760}
08247 ,{ 1300, 1280, 1300, 1280, -760}
08248 ,{ 50, 30, 50, 30, -760}
08249 }
08250 ,{{{ 560, 540, 560, 540, -10}
08251 ,{ 560, 540, 560, 540, -10}
08252 ,{ 50, 30, 50, 30, -760}
08253 ,{ -50, -310, -50, -310, -1100}
08254 ,{ 50, 30, 50, 30, -760}
08255 }
08256 ,{{{ 50, 30, 50, 30, -760}
08257 ,{ 50, 30, 50, 30, -760}
08258 ,{ 50, 30, 50, 30, -760}
08259 ,{ 50, 30, 50, 30, -760}
08260 ,{ 50, 30, 50, 30, -760}
08261 }
08262 ,{{{ 1300, 1280, 1300, 1280, -760}
08263 ,{ -220, -480, -220, -480, -1270}
08264 ,{ 50, 30, 50, 30, -760}
08265 ,{ 1300, 1280, 1300, 1280, -760}
08266 ,{ 50, 30, 50, 30, -760}
08267 }
08268 ,{{{ 50, 30, 50, 30, -760}
08269 ,{ 50, 30, 50, 30, -760}
08270 ,{ 50, 30, 50, 30, -760}
08271 ,{ 50, 30, 50, 30, -760}
08272 ,{ 50, 30, 50, 30, -760}
08273 }
08274 }
08275 }
08276 ,{{{ 2050, 1930, 1800, 2050, 1800}
08277 ,{ 2050, 1400, 1270, 2050, 1270}
08278 ,{ 1750, 1100, 970, 1750, 970}
08279 ,{ 1930, 1930, 1800, 1760, 1800}
08280 ,{ 1750, 1100, 970, 1750, 970}
08281 }
08282 ,{{{ 2050, 1400, 1270, 2050, 1270}
08283 ,{ 2050, 1400, 1270, 2050, 1270}
08284 ,{ 1740, 1090, 960, 1740, 960}
08285 ,{ 130, 10, 130, -260, -110}
08286 ,{ 1740, 1090, 960, 1740, 960}
08287 }
08288 ,{{{ 1760, 1110, 980, 1760, 980}
08289 ,{ 1760, 1110, 980, 1760, 980}
08290 ,{ 1750, 1100, 970, 1750, 970}
08291 ,{ 1760, 1110, 980, 1760, 980}
08292 ,{ 1750, 1100, 970, 1750, 970}
08293 }
08294 ,{{{ 1930, 1930, 1800, 1740, 1800}
08295 ,{ 300, 190, 300, -80, 60}
08296 ,{ 1740, 1090, 960, 1740, 960}
08297 ,{ 1930, 1930, 1800, 1650, 1800}
08298 ,{ 1740, 1090, 960, 1740, 960}
08299 }
08300 ,{{{ 1760, 1110, 980, 1760, 980}
08301 ,{ 1760, 1110, 980, 1760, 980}
08302 ,{ 1750, 1100, 970, 1750, 970}
08303 ,{ 1760, 1110, 980, 1760, 980}
08304 ,{ 360, 360, 0, -150, 0}
08305 }
08306 }
08307 ,{{{ 1930, 1930, 1800, -300, 1800}
08308 ,{ 1400, 1400, 1270, -300, 1270}
08309 ,{ 1100, 1100, 970, -360, 970}
08310 ,{ 1930, 1930, 1800, -590, 1800}
08311 ,{ 1100, 1100, 970, -360, 970}
08312 }
08313 ,{{{ 1400, 1400, 1270, -300, 1270}
08314 ,{ 1400, 1400, 1270, -300, 1270}
08315 ,{ 1090, 1090, 960, -610, 960}
08316 ,{ 10, 10, -110, -1690, -110}
08317 ,{ 1090, 1090, 960, -610, 960}
08318 }
08319 ,{{{ 1110, 1110, 980, -360, 980}
08320 ,{ 1110, 1110, 980, -590, 980}
08321 ,{ 1100, 1100, 970, -360, 970}
08322 ,{ 1110, 1110, 980, -590, 980}
08323 ,{ 1100, 1100, 970, -360, 970}
08324 }
08325 ,{{{ 1930, 1930, 1800, -610, 1800}
08326 ,{ 190, 190, 60, -1510, 60}
08327 ,{ 1090, 1090, 960, -610, 960}
08328 ,{ 1930, 1930, 1800, -1020, 1800}
08329 ,{ 1090, 1090, 960, -610, 960}
08330 }
```

```
08331 ,{{ 1110, 1110, 980, -360, 980}
08332 ,{ 1110, 1110, 980, -590, 980}
08333 ,{ 1100, 1100, 970, -360, 970}
08334 ,{ 1110, 1110, 980, -590, 980}
08335 ,{ 360, 360, 0, -1580, 0}
08336 }
08337 }
08338 ,{{{ 1800, 1650, 1800, 1650, 1360}
08339 ,{ 1270, 1120, 1270, 1120, 830}
08340 ,{ 970, 820, 970, 820, 530}
08341 ,{ 1800, 1650, 1800, 1650, 1360}
08342 ,{ 970, 820, 970, 820, 530}
08343 }
08344 ,{{{ 1270, 1120, 1270, 1120, 830}
08345 ,{ 1270, 1120, 1270, 1120, 830}
08346 ,{ 960, 810, 960, 810, 520}
08347 ,{ 130, -260, 130, -260, -310}
08348 ,{ 960, 810, 960, 810, 520}
08349 }
08350 ,{{{ 980, 830, 980, 830, 540}
08351 ,{ 980, 830, 980, 830, 540}
08352 ,{ 970, 820, 970, 820, 530}
08353 ,{ 980, 830, 980, 830, 540}
08354 ,{ 970, 820, 970, 820, 530}
08355 }
08356 ,{{{ 1800, 1650, 1800, 1650, 1360}
08357 ,{ 300, -80, 300, -80, -130}
08358 ,{ 960, 810, 960, 810, 520}
08359 ,{ 1800, 1650, 1800, 1650, 1360}
08360 ,{ 960, 810, 960, 810, 520}
08361 }
08362 ,{{{ 980, 830, 980, 830, 540}
08363 ,{ 980, 830, 980, 830, 540}
08364 ,{ 970, 820, 970, 820, 530}
08365 ,{ 980, 830, 980, 830, 540}
08366 ,{ 0, -150, 0, -150, -440}
08367 }
08368 }
08369 ,{{{ 2050, 0, 1800, 2050, 1800}
08370 ,{ 2050, 0, 1270, 2050, 1270}
08371 ,{ 1750, -60, 970, 1750, 970}
08372 ,{ 1800, -290, 1800, 1760, 1800}
08373 ,{ 1750, -60, 970, 1750, 970}
08374 }
08375 ,{{{ 2050, 0, 1270, 2050, 1270}
08376 ,{ 2050, 0, 1270, 2050, 1270}
08377 ,{ 1740, -310, 960, 1740, 960}
08378 ,{ -110, -1390, -110, -580, -110}
08379 ,{ 1740, -310, 960, 1740, 960}
08380 }
08381 ,{{{ 1760, -60, 980, 1760, 980}
08382 ,{ 1760, -290, 980, 1760, 980}
08383 ,{ 1750, -60, 970, 1750, 970}
08384 ,{ 1760, -290, 980, 1760, 980}
08385 ,{ 1750, -60, 970, 1750, 970}
08386 }
08387 ,{{{ 1800, -310, 1800, 1740, 1800}
08388 ,{ 60, -1210, 60, -400, 60}
08389 ,{ 1740, -310, 960, 1740, 960}
08390 ,{ 1800, -720, 1800, 80, 1800}
08391 ,{ 1740, -310, 960, 1740, 960}
08392 }
08393 ,{{{ 1760, -60, 980, 1760, 980}
08394 ,{ 1760, -290, 980, 1760, 980}
08395 ,{ 1750, -60, 970, 1750, 970}
08396 ,{ 1760, -290, 980, 1760, 980}
08397 ,{ 0, -1280, 0, -470, 0}
08398 }
08399 }
08400 ,{{{ 1670, 1650, 1670, 1650, 570}
08401 ,{ 1140, 1120, 1140, 1120, 570}
08402 ,{ 840, 820, 840, 820, 30}
08403 ,{ 1670, 1650, 1670, 1650, 40}
08404 ,{ 840, 820, 840, 820, 30}
08405 }
08406 ,{{{ 1140, 1120, 1140, 1120, 570}
08407 ,{ 1140, 1120, 1140, 1120, 570}
08408 ,{ 830, 810, 830, 810, 20}
08409 ,{ 0, -260, 0, -260, -1050}
08410 ,{ 830, 810, 830, 810, 20}
08411 }
08412 ,{{{ 850, 830, 850, 830, 40}
08413 ,{ 850, 830, 850, 830, 40}
08414 ,{ 840, 820, 840, 820, 30}
08415 ,{ 850, 830, 850, 830, 40}
08416 ,{ 840, 820, 840, 820, 30}
08417 }
```

```
08418 ,{{ 1670, 1650, 1670, 1650, 20}
08419 ,{ 180, -80, 180, -80, -870}
08420 ,{ 830, 810, 830, 810, 20}
08421 ,{ 1670, 1650, 1670, 1650, -380}
08422 ,{ 830, 810, 830, 810, 20}
08423 }
08424 ,{{ 850, 830, 850, 830, 40}
08425 ,{ 850, 830, 850, 830, 40}
08426 ,{ 840, 820, 840, 820, 30}
08427 ,{ 850, 830, 850, 830, 40}
08428 ,{ -130, -150, -130, -150, -940}
08429 }
08430 }
08431 }
08432 ,{{{ 2120, 2120, 1990, 2120, 1990}
08433 ,{ 2120, 1470, 1340, 2120, 1340}
08434 ,{ 1990, 1340, 1210, 1990, 1210}
08435 ,{ 2120, 2120, 1990, 1990, 1990}
08436 ,{ 1860, 1210, 1080, 1860, 1080}
08437 }
08438 ,{{ 2120, 1470, 1340, 2120, 1340}
08439 ,{ 2120, 1470, 1340, 2120, 1340}
08440 ,{ 1840, 1190, 1060, 1840, 1060}
08441 ,{ 180, 60, 180, -210, -60}
08442 ,{ 1840, 1190, 1060, 1840, 1060}
08443 }
08444 ,{{{ 1990, 1340, 1210, 1990, 1210}
08445 ,{ 1990, 1340, 1210, 1990, 1210}
08446 ,{ 1990, 1340, 1210, 1990, 1210}
08447 ,{ 1990, 1340, 1210, 1990, 1210}
08448 ,{ 1860, 1210, 1080, 1860, 1080}
08449 }
08450 ,{{ 2120, 2120, 1990, 1840, 1990}
08451 ,{ -120, -230, -120, -510, -360}
08452 ,{ 1840, 1190, 1060, 1840, 1060}
08453 ,{ 2120, 2120, 1990, 1840, 1990}
08454 ,{ 1840, 1190, 1060, 1840, 1060}
08455 }
08456 ,{{{ 1990, 1340, 1210, 1990, 1210}
08457 ,{ 1990, 1340, 1210, 1990, 1210}
08458 ,{ 1550, 900, 770, 1550, 770}
08459 ,{ 1990, 1340, 1210, 1990, 1210}
08460 ,{ 640, 640, 270, 120, 270}
08461 }
08462 }
08463 ,{{{ 2120, 2120, 1990, -120, 1990}
08464 ,{ 1470, 1470, 1340, -230, 1340}
08465 ,{ 1340, 1340, 1210, -120, 1210}
08466 ,{ 2120, 2120, 1990, -360, 1990}
08467 ,{ 1210, 1210, 1080, -250, 1080}
08468 }
08469 ,{{ 1470, 1470, 1340, -230, 1340}
08470 ,{ 1470, 1470, 1340, -230, 1340}
08471 ,{ 1190, 1190, 1060, -510, 1060}
08472 ,{ 60, 60, -60, -1640, -60}
08473 ,{ 1190, 1190, 1060, -510, 1060}
08474 }
08475 ,{{ 1340, 1340, 1210, -120, 1210}
08476 ,{ 1340, 1340, 1210, -360, 1210}
08477 ,{ 1340, 1340, 1210, -120, 1210}
08478 ,{ 1340, 1340, 1210, -360, 1210}
08479 ,{ 1210, 1210, 1080, -250, 1080}
08480 }
08481 ,{{ 2120, 2120, 1990, -510, 1990}
08482 ,{ -230, -230, -360, -1940, -360}
08483 ,{ 1190, 1190, 1060, -510, 1060}
08484 ,{ 2120, 2120, 1990, -830, 1990}
08485 ,{ 1190, 1190, 1060, -510, 1060}
08486 }
08487 ,{{ 1340, 1340, 1210, -360, 1210}
08488 ,{ 1340, 1340, 1210, -360, 1210}
08489 ,{ 900, 900, 770, -560, 770}
08490 ,{ 1340, 1340, 1210, -360, 1210}
08491 ,{ 640, 640, 270, -1300, 270}
08492 }
08493 }
08494 ,{{{ 1990, 1840, 1990, 1840, 1550}
08495 ,{ 1340, 1190, 1340, 1190, 900}
08496 ,{ 1210, 1060, 1210, 1060, 770}
08497 ,{ 1990, 1840, 1990, 1840, 1550}
08498 ,{ 1080, 930, 1080, 930, 640}
08499 }
08500 ,{{ 1340, 1190, 1340, 1190, 900}
08501 ,{ 1340, 1190, 1340, 1190, 900}
08502 ,{ 1060, 910, 1060, 910, 620}
08503 ,{ 180, -210, 180, -210, -260}
08504 ,{ 1060, 910, 1060, 910, 620}
```

```
08505      }
08506      ,{{ 1210, 1060, 1210, 1060, 770}
08507      ,{ 1210, 1060, 1210, 1060, 770}
08508      ,{ 1210, 1060, 1210, 1060, 770}
08509      ,{ 1210, 1060, 1210, 1060, 770}
08510      ,{ 1080, 930, 1080, 930, 640}
08511      }
08512      ,{{ 1990, 1840, 1990, 1840, 1550}
08513      ,{ -120, -510, -120, -510, -560}
08514      ,{ 1060, 910, 1060, 910, 620}
08515      ,{ 1990, 1840, 1990, 1840, 1550}
08516      ,{ 1060, 910, 1060, 910, 620}
08517      }
08518      ,{{ 1210, 1060, 1210, 1060, 770}
08519      ,{ 1210, 1060, 1210, 1060, 770}
08520      ,{ 770, 620, 770, 620, 330}
08521      ,{ 1210, 1060, 1210, 1060, 770}
08522      ,{ 270, 120, 270, 120, -170}
08523      }
08524      }
08525      ,{{{ 2120, 180, 1990, 2120, 1990}
08526      ,{ 2120, 60, 1340, 2120, 1340}
08527      ,{ 1990, 180, 1210, 1990, 1210}
08528      ,{ 1990, -60, 1990, 1990, 1990}
08529      ,{ 1860, 50, 1080, 1860, 1080}
08530      }
08531      ,{{{ 2120, 60, 1340, 2120, 1340}
08532      ,{ 2120, 60, 1340, 2120, 1340}
08533      ,{ 1840, -210, 1060, 1840, 1060}
08534      ,{ -60, -1340, -60, -530, -60}
08535      ,{ 1840, -210, 1060, 1840, 1060}
08536      }
08537      ,{{{ 1990, 180, 1210, 1990, 1210}
08538      ,{ 1990, -60, 1210, 1990, 1210}
08539      ,{ 1990, 180, 1210, 1990, 1210}
08540      ,{ 1990, -60, 1210, 1990, 1210}
08541      ,{ 1860, 50, 1080, 1860, 1080}
08542      }
08543      ,{{{ 1990, -210, 1990, 1840, 1990}
08544      ,{ -360, -1640, -360, -830, -360}
08545      ,{ 1840, -210, 1060, 1840, 1060}
08546      ,{ 1990, -530, 1990, 270, 1990}
08547      ,{ 1840, -210, 1060, 1840, 1060}
08548      }
08549      ,{{{ 1990, -60, 1210, 1990, 1210}
08550      ,{ 1990, -60, 1210, 1990, 1210}
08551      ,{ 1550, -260, 770, 1550, 770}
08552      ,{ 1990, -60, 1210, 1990, 1210}
08553      ,{ 270, -1000, 270, -200, 270}
08554      }
08555      }
08556      ,{{{ 1860, 1840, 1860, 1840, 640}
08557      ,{ 1210, 1190, 1210, 1190, 640}
08558      ,{ 1080, 1060, 1080, 1060, 270}
08559      ,{ 1860, 1840, 1860, 1840, 270}
08560      ,{ 950, 930, 950, 930, 140}
08561      }
08562      ,{{ 1210, 1190, 1210, 1190, 640}
08563      ,{ 1210, 1190, 1210, 1190, 640}
08564      ,{ 930, 910, 930, 910, 120}
08565      ,{ 50, -210, 50, -210, -1000}
08566      ,{ 930, 910, 930, 910, 120}
08567      }
08568      ,{{ 1080, 1060, 1080, 1060, 270}
08569      ,{ 1080, 1060, 1080, 1060, 270}
08570      ,{ 1080, 1060, 1080, 1060, 270}
08571      ,{ 1080, 1060, 1080, 1060, 270}
08572      ,{ 950, 930, 950, 930, 140}
08573      }
08574      ,{{ 1860, 1840, 1860, 1840, 120}
08575      ,{ -250, -510, -250, -510, -1300}
08576      ,{ 930, 910, 930, 910, 120}
08577      ,{ 1860, 1840, 1860, 1840, -200}
08578      ,{ 930, 910, 930, 910, 120}
08579      }
08580      ,{{ 1080, 1060, 1080, 1060, 270}
08581      ,{ 1080, 1060, 1080, 1060, 270}
08582      ,{ 640, 620, 640, 620, -170}
08583      ,{ 1080, 1060, 1080, 1060, 270}
08584      ,{ 140, 120, 140, 120, -670}
08585      }
08586      }
08587      }
08588      ,{{{ 2120, 2120, 1990, 2120, 1990}
08589      ,{ 2120, 1470, 1340, 2120, 1340}
08590      ,{ 1990, 1340, 1210, 1990, 1210}
08591      ,{ 2120, 2120, 1990, 1990, 1990}
```

```
08592     , { 1860, 1210, 1080, 1860, 1080}
08593     }
08594     , { { 2120, 1470, 1340, 2120, 1340}
08595     , { 2120, 1470, 1340, 2120, 1340}
08596     , { 1840, 1190, 1060, 1840, 1060}
08597     , { 400, 290, 400, 10, 160}
08598     , { 1840, 1190, 1060, 1840, 1060}
08599     }
08600     , { { 1990, 1340, 1210, 1990, 1210}
08601     , { 1990, 1340, 1210, 1990, 1210}
08602     , { 1990, 1340, 1210, 1990, 1210}
08603     , { 1990, 1340, 1210, 1990, 1210}
08604     , { 1860, 1210, 1080, 1860, 1080}
08605     }
08606     , { { 2120, 2120, 1990, 1840, 1990}
08607     , { 300, 190, 300, -80, 60}
08608     , { 1840, 1190, 1060, 1840, 1060}
08609     , { 2120, 2120, 1990, 1840, 1990}
08610     , { 1840, 1190, 1060, 1840, 1060}
08611     }
08612     , { { 1990, 1340, 1210, 1990, 1210}
08613     , { 1990, 1340, 1210, 1990, 1210}
08614     , { 1750, 1100, 970, 1750, 970}
08615     , { 1990, 1340, 1210, 1990, 1210}
08616     , { 640, 640, 270, 120, 270}
08617     }
08618     }
08619     , { { { 2120, 2120, 1990, -120, 1990}
08620     , { 1470, 1470, 1340, -230, 1340}
08621     , { 1340, 1340, 1210, -120, 1210}
08622     , { 2120, 2120, 1990, -360, 1990}
08623     , { 1210, 1210, 1080, -250, 1080}
08624     }
08625     , { { 1470, 1470, 1340, -230, 1340}
08626     , { 1470, 1470, 1340, -230, 1340}
08627     , { 1190, 1190, 1060, -510, 1060}
08628     , { 290, 290, 160, -1410, 160}
08629     , { 1190, 1190, 1060, -510, 1060}
08630     }
08631     , { { 1340, 1340, 1210, -120, 1210}
08632     , { 1340, 1340, 1210, -360, 1210}
08633     , { 1340, 1340, 1210, -120, 1210}
08634     , { 1340, 1340, 1210, -360, 1210}
08635     , { 1210, 1210, 1080, -250, 1080}
08636     }
08637     , { { 2120, 2120, 1990, -510, 1990}
08638     , { 190, 190, 60, -1510, 60}
08639     , { 1190, 1190, 1060, -510, 1060}
08640     , { 2120, 2120, 1990, -830, 1990}
08641     , { 1190, 1190, 1060, -510, 1060}
08642     }
08643     , { { 1340, 1340, 1210, -360, 1210}
08644     , { 1340, 1340, 1210, -360, 1210}
08645     , { 1100, 1100, 970, -360, 970}
08646     , { 1340, 1340, 1210, -360, 1210}
08647     , { 640, 640, 270, -1300, 270}
08648     }
08649     }
08650     , { { { 1990, 1840, 1990, 1840, 1550}
08651     , { 1340, 1190, 1340, 1190, 900}
08652     , { 1210, 1060, 1210, 1060, 770}
08653     , { 1990, 1840, 1990, 1840, 1550}
08654     , { 1080, 930, 1080, 930, 640}
08655     }
08656     , { { 1340, 1190, 1340, 1190, 900}
08657     , { 1340, 1190, 1340, 1190, 900}
08658     , { 1060, 910, 1060, 910, 620}
08659     , { 400, 10, 400, 10, -40}
08660     , { 1060, 910, 1060, 910, 620}
08661     }
08662     , { { 1210, 1060, 1210, 1060, 770}
08663     , { 1210, 1060, 1210, 1060, 770}
08664     , { 1210, 1060, 1210, 1060, 770}
08665     , { 1210, 1060, 1210, 1060, 770}
08666     , { 1080, 930, 1080, 930, 640}
08667     }
08668     , { { 1990, 1840, 1990, 1840, 1550}
08669     , { 300, -80, 300, -80, -130}
08670     , { 1060, 910, 1060, 910, 620}
08671     , { 1990, 1840, 1990, 1840, 1550}
08672     , { 1060, 910, 1060, 910, 620}
08673     }
08674     , { { 1210, 1060, 1210, 1060, 770}
08675     , { 1210, 1060, 1210, 1060, 770}
08676     , { 970, 820, 970, 820, 530}
08677     , { 1210, 1060, 1210, 1060, 770}
08678     , { 270, 120, 270, 120, -170}
```

```
08679     }
08680     }
08681     ,{{{ 2120, 180, 1990, 2120, 1990}
08682     ,{ 2120, 60, 1340, 2120, 1340}
08683     ,{ 1990, 180, 1210, 1990, 1210}
08684     ,{ 1990, -60, 1990, 1990, 1990}
08685     ,{ 1860, 50, 1080, 1860, 1080}
08686     }
08687     ,{{{ 2120, 60, 1340, 2120, 1340}
08688     ,{ 2120, 60, 1340, 2120, 1340}
08689     ,{ 1840, -210, 1060, 1840, 1060}
08690     ,{ 160, -1110, 160, -310, 160}
08691     ,{ 1840, -210, 1060, 1840, 1060}
08692     }
08693     ,{{{ 1990, 180, 1210, 1990, 1210}
08694     ,{ 1990, -60, 1210, 1990, 1210}
08695     ,{ 1990, 180, 1210, 1990, 1210}
08696     ,{ 1990, -60, 1210, 1990, 1210}
08697     ,{ 1860, 50, 1080, 1860, 1080}
08698     }
08699     ,{{{ 1990, -210, 1990, 1840, 1990}
08700     ,{ 60, -1210, 60, -400, 60}
08701     ,{ 1840, -210, 1060, 1840, 1060}
08702     ,{ 1990, -530, 1990, 270, 1990}
08703     ,{ 1840, -210, 1060, 1840, 1060}
08704     }
08705     ,{{{ 1990, -60, 1210, 1990, 1210}
08706     ,{ 1990, -60, 1210, 1990, 1210}
08707     ,{ 1750, -60, 970, 1750, 970}
08708     ,{ 1990, -60, 1210, 1990, 1210}
08709     ,{ 270, -1000, 270, -200, 270}
08710     }
08711     }
08712     ,{{{ 1860, 1840, 1860, 1840, 640}
08713     ,{ 1210, 1190, 1210, 1190, 640}
08714     ,{ 1080, 1060, 1080, 1060, 270}
08715     ,{ 1860, 1840, 1860, 1840, 270}
08716     ,{ 950, 930, 950, 930, 140}
08717     }
08718     ,{{{ 1210, 1190, 1210, 1190, 640}
08719     ,{ 1210, 1190, 1210, 1190, 640}
08720     ,{ 930, 910, 930, 910, 120}
08721     ,{ 270, 10, 270, 10, -780}
08722     ,{ 930, 910, 930, 910, 120}
08723     }
08724     ,{{{ 1080, 1060, 1080, 1060, 270}
08725     ,{ 1080, 1060, 1080, 1060, 270}
08726     ,{ 1080, 1060, 1080, 1060, 270}
08727     ,{ 1080, 1060, 1080, 1060, 270}
08728     ,{ 950, 930, 950, 930, 140}
08729     }
08730     ,{{{ 1860, 1840, 1860, 1840, 120}
08731     ,{ 180, -80, 180, -80, -870}
08732     ,{ 930, 910, 930, 910, 120}
08733     ,{ 1860, 1840, 1860, 1840, -200}
08734     ,{ 930, 910, 930, 910, 120}
08735     }
08736     ,{{{ 1080, 1060, 1080, 1060, 270}
08737     ,{ 1080, 1060, 1080, 1060, 270}
08738     ,{ 840, 820, 840, 820, 30}
08739     ,{ 1080, 1060, 1080, 1060, 270}
08740     ,{ 140, 120, 140, 120, -670}
08741     }
08742     }
08743     }
08744     }
08745     ,{{{ { INF, INF, INF, INF, INF}
08746     ,{ INF, INF, INF, INF, INF}
08747     ,{ INF, INF, INF, INF, INF}
08748     ,{ INF, INF, INF, INF, INF}
08749     ,{ INF, INF, INF, INF, INF}
08750     }
08751     ,{{{ INF, INF, INF, INF, INF}
08752     ,{ INF, INF, INF, INF, INF}
08753     ,{ INF, INF, INF, INF, INF}
08754     ,{ INF, INF, INF, INF, INF}
08755     ,{ INF, INF, INF, INF, INF}
08756     }
08757     ,{{{ INF, INF, INF, INF, INF}
08758     ,{ INF, INF, INF, INF, INF}
08759     ,{ INF, INF, INF, INF, INF}
08760     ,{ INF, INF, INF, INF, INF}
08761     ,{ INF, INF, INF, INF, INF}
08762     }
08763     ,{{{ INF, INF, INF, INF, INF}
08764     ,{ INF, INF, INF, INF, INF}
08765     ,{ INF, INF, INF, INF, INF}
```



```
08766     , {   INF,   INF,   INF,   INF,   INF }
08767     , {   INF,   INF,   INF,   INF,   INF }
08768     }
08769     , { {   INF,   INF,   INF,   INF,   INF }
08770     , {   INF,   INF,   INF,   INF,   INF }
08771     , {   INF,   INF,   INF,   INF,   INF }
08772     , {   INF,   INF,   INF,   INF,   INF }
08773     , {   INF,   INF,   INF,   INF,   INF }
08774     }
08775     }
08776     , { { {   INF,   INF,   INF,   INF,   INF }
08777     , {   INF,   INF,   INF,   INF,   INF }
08778     , {   INF,   INF,   INF,   INF,   INF }
08779     , {   INF,   INF,   INF,   INF,   INF }
08780     , {   INF,   INF,   INF,   INF,   INF }
08781     }
08782     , { {   INF,   INF,   INF,   INF,   INF }
08783     , {   INF,   INF,   INF,   INF,   INF }
08784     , {   INF,   INF,   INF,   INF,   INF }
08785     , {   INF,   INF,   INF,   INF,   INF }
08786     , {   INF,   INF,   INF,   INF,   INF }
08787     }
08788     , { {   INF,   INF,   INF,   INF,   INF }
08789     , {   INF,   INF,   INF,   INF,   INF }
08790     , {   INF,   INF,   INF,   INF,   INF }
08791     , {   INF,   INF,   INF,   INF,   INF }
08792     , {   INF,   INF,   INF,   INF,   INF }
08793     }
08794     , { {   INF,   INF,   INF,   INF,   INF }
08795     , {   INF,   INF,   INF,   INF,   INF }
08796     , {   INF,   INF,   INF,   INF,   INF }
08797     , {   INF,   INF,   INF,   INF,   INF }
08798     , {   INF,   INF,   INF,   INF,   INF }
08799     }
08800     , { {   INF,   INF,   INF,   INF,   INF }
08801     , {   INF,   INF,   INF,   INF,   INF }
08802     , {   INF,   INF,   INF,   INF,   INF }
08803     , {   INF,   INF,   INF,   INF,   INF }
08804     , {   INF,   INF,   INF,   INF,   INF }
08805     }
08806     }
08807     , { { {   INF,   INF,   INF,   INF,   INF }
08808     , {   INF,   INF,   INF,   INF,   INF }
08809     , {   INF,   INF,   INF,   INF,   INF }
08810     , {   INF,   INF,   INF,   INF,   INF }
08811     , {   INF,   INF,   INF,   INF,   INF }
08812     }
08813     , { { {   INF,   INF,   INF,   INF,   INF }
08814     , {   INF,   INF,   INF,   INF,   INF }
08815     , {   INF,   INF,   INF,   INF,   INF }
08816     , {   INF,   INF,   INF,   INF,   INF }
08817     , {   INF,   INF,   INF,   INF,   INF }
08818     }
08819     , { { {   INF,   INF,   INF,   INF,   INF }
08820     , {   INF,   INF,   INF,   INF,   INF }
08821     , {   INF,   INF,   INF,   INF,   INF }
08822     , {   INF,   INF,   INF,   INF,   INF }
08823     , {   INF,   INF,   INF,   INF,   INF }
08824     }
08825     , { { {   INF,   INF,   INF,   INF,   INF }
08826     , {   INF,   INF,   INF,   INF,   INF }
08827     , {   INF,   INF,   INF,   INF,   INF }
08828     , {   INF,   INF,   INF,   INF,   INF }
08829     , {   INF,   INF,   INF,   INF,   INF }
08830     }
08831     , { { {   INF,   INF,   INF,   INF,   INF }
08832     , {   INF,   INF,   INF,   INF,   INF }
08833     , {   INF,   INF,   INF,   INF,   INF }
08834     , {   INF,   INF,   INF,   INF,   INF }
08835     , {   INF,   INF,   INF,   INF,   INF }
08836     }
08837     }
08838     , { { {   INF,   INF,   INF,   INF,   INF }
08839     , {   INF,   INF,   INF,   INF,   INF }
08840     , {   INF,   INF,   INF,   INF,   INF }
08841     , {   INF,   INF,   INF,   INF,   INF }
08842     , {   INF,   INF,   INF,   INF,   INF }
08843     }
08844     , { { {   INF,   INF,   INF,   INF,   INF }
08845     , {   INF,   INF,   INF,   INF,   INF }
08846     , {   INF,   INF,   INF,   INF,   INF }
08847     , {   INF,   INF,   INF,   INF,   INF }
08848     , {   INF,   INF,   INF,   INF,   INF }
08849     }
08850     , { { {   INF,   INF,   INF,   INF,   INF }
08851     , {   INF,   INF,   INF,   INF,   INF }
08852     , {   INF,   INF,   INF,   INF,   INF }
```

```

08853      , {   INF,   INF,   INF,   INF,   INF }
08854      , {   INF,   INF,   INF,   INF,   INF }
08855      }
08856      , { {   INF,   INF,   INF,   INF,   INF }
08857      , {   INF,   INF,   INF,   INF,   INF }
08858      , {   INF,   INF,   INF,   INF,   INF }
08859      , {   INF,   INF,   INF,   INF,   INF }
08860      , {   INF,   INF,   INF,   INF,   INF }
08861      }
08862      , { {   INF,   INF,   INF,   INF,   INF }
08863      , {   INF,   INF,   INF,   INF,   INF }
08864      , {   INF,   INF,   INF,   INF,   INF }
08865      , {   INF,   INF,   INF,   INF,   INF }
08866      , {   INF,   INF,   INF,   INF,   INF }
08867      }
08868      }
08869      , { { {   INF,   INF,   INF,   INF,   INF }
08870      , {   INF,   INF,   INF,   INF,   INF }
08871      , {   INF,   INF,   INF,   INF,   INF }
08872      , {   INF,   INF,   INF,   INF,   INF }
08873      , {   INF,   INF,   INF,   INF,   INF }
08874      }
08875      , { {   INF,   INF,   INF,   INF,   INF }
08876      , {   INF,   INF,   INF,   INF,   INF }
08877      , {   INF,   INF,   INF,   INF,   INF }
08878      , {   INF,   INF,   INF,   INF,   INF }
08879      , {   INF,   INF,   INF,   INF,   INF }
08880      }
08881      , { {   INF,   INF,   INF,   INF,   INF }
08882      , {   INF,   INF,   INF,   INF,   INF }
08883      , {   INF,   INF,   INF,   INF,   INF }
08884      , {   INF,   INF,   INF,   INF,   INF }
08885      , {   INF,   INF,   INF,   INF,   INF }
08886      }
08887      , { {   INF,   INF,   INF,   INF,   INF }
08888      , {   INF,   INF,   INF,   INF,   INF }
08889      , {   INF,   INF,   INF,   INF,   INF }
08890      , {   INF,   INF,   INF,   INF,   INF }
08891      , {   INF,   INF,   INF,   INF,   INF }
08892      }
08893      , { {   INF,   INF,   INF,   INF,   INF }
08894      , {   INF,   INF,   INF,   INF,   INF }
08895      , {   INF,   INF,   INF,   INF,   INF }
08896      , {   INF,   INF,   INF,   INF,   INF }
08897      , {   INF,   INF,   INF,   INF,   INF }
08898      }
08899      }
08900      }
08901      , { { { {   1350,   850,   720,   1350,   720 }
08902      , {   1300,   650,   540,   1300,   520 }
08903      , {   1350,   700,   570,   1350,   570 }
08904      , {   1300,   850,   720,   1300,   720 }
08905      , {   1250,   590,   460,   1250,   460 }
08906      }
08907      , { {   1160,   500,   400,   1160,   370 }
08908      , {   1160,   500,   370,   1160,   370 }
08909      , {   850,   190,   60,   850,   60 }
08910      , {   400,   290,   400,   10,   170 }
08911      , {   850,   190,   60,   850,   60 }
08912      }
08913      , { {   1300,   650,   520,   1300,   520 }
08914      , {   1300,   650,   520,   1300,   520 }
08915      , {   1290,   640,   510,   1290,   510 }
08916      , {   1300,   650,   520,   1300,   520 }
08917      , {   1250,   590,   460,   1250,   460 }
08918      }
08919      , { {   850,   850,   720,   850,   720 }
08920      , {   540,   0,   540,   -270,   -120 }
08921      , {   850,   190,   60,   850,   60 }
08922      , {   850,   850,   720,   570,   720 }
08923      , {   850,   190,   60,   850,   60 }
08924      }
08925      , { {   1350,   700,   570,   1350,   570 }
08926      , {   1300,   650,   520,   1300,   520 }
08927      , {   1350,   700,   570,   1350,   570 }
08928      , {   1300,   650,   520,   1300,   520 }
08929      , {   100,   100,   -270,   -230,   -270 }
08930      }
08931      }
08932      , { { {   850,   850,   720,   -330,   720 }
08933      , {   650,   650,   520,   -620,   520 }
08934      , {   700,   700,   570,   -330,   570 }
08935      , {   850,   850,   720,   -620,   720 }
08936      , {   590,   590,   460,   -440,   460 }
08937      }
08938      , { {   500,   500,   370,   -770,   370 }
08939      , {   500,   500,   370,   -770,   370 }

```

```
08940 , { 190, 190, 60, -1070, 60}
08941 , { 290, 290, 160, -980, 160}
08942 , { 190, 190, 60, -1080, 60}
08943 }
08944 , { { 650, 650, 520, -390, 520}
08945 , { 650, 650, 520, -620, 520}
08946 , { 640, 640, 510, -390, 510}
08947 , { 650, 650, 520, -620, 520}
08948 , { 590, 590, 460, -440, 460}
08949 }
08950 , { { 850, 850, 720, -1080, 720}
08951 , { 10, 0, 10, -1270, -120}
08952 , { 190, 190, 60, -1080, 60}
08953 , { 850, 850, 720, -1080, 720}
08954 , { 190, 190, 60, -1080, 60}
08955 }
08956 , { { 700, 700, 570, -330, 570}
08957 , { 650, 650, 520, -620, 520}
08958 , { 700, 700, 570, -330, 570}
08959 , { 650, 650, 520, -620, 520}
08960 , { 100, 100, -270, -1300, -270}
08961 }
08962 }
08963 , { { { 720, 570, 720, 570, 480}
08964 , { 540, 370, 540, 370, 280}
08965 , { 570, 420, 570, 420, 340}
08966 , { 720, 570, 720, 570, 480}
08967 , { 460, 310, 460, 310, 230}
08968 }
08969 , { { 400, 220, 400, 220, 170}
08970 , { 370, 220, 370, 220, 140}
08971 , { 60, -80, 60, -80, -170}
08972 , { 400, 10, 400, 10, 170}
08973 , { 60, -80, 60, -80, -170}
08974 }
08975 , { { 520, 370, 520, 370, 280}
08976 , { 520, 370, 520, 370, 280}
08977 , { 510, 360, 510, 360, 280}
08978 , { 520, 370, 520, 370, 280}
08979 , { 460, 310, 460, 310, 230}
08980 }
08981 , { { 720, 570, 720, 570, 480}
08982 , { 540, -100, 540, -270, -120}
08983 , { 60, -80, 60, -80, -170}
08984 , { 720, 570, 720, 570, 480}
08985 , { 60, -80, 60, -80, -170}
08986 }
08987 , { { 570, 420, 570, 420, 340}
08988 , { 520, 370, 520, 370, 280}
08989 , { 570, 420, 570, 420, 340}
08990 , { 520, 370, 520, 370, 280}
08991 , { -270, -420, -270, -420, -500}
08992 }
08993 }
08994 , { { { 1350, -230, 720, 1350, 720}
08995 , { 1300, -530, 520, 1300, 520}
08996 , { 1350, -230, 570, 1350, 570}
08997 , { 1300, -530, 720, 1300, 720}
08998 , { 1250, -340, 460, 1250, 460}
08999 }
09000 , { { 1160, -670, 370, 1160, 370}
09001 , { 1160, -670, 370, 1160, 370}
09002 , { 850, -980, 60, 850, 60}
09003 , { 160, -890, 160, -310, 160}
09004 , { 850, -980, 60, 850, 60}
09005 }
09006 , { { 1300, -290, 520, 1300, 520}
09007 , { 1300, -530, 520, 1300, 520}
09008 , { 1290, -290, 510, 1290, 510}
09009 , { 1300, -530, 520, 1300, 520}
09010 , { 1250, -340, 460, 1250, 460}
09011 }
09012 , { { 850, -980, 720, 850, 720}
09013 , { -120, -1170, -120, -590, -120}
09014 , { 850, -980, 60, 850, 60}
09015 , { 720, -1580, 720, -1000, 720}
09016 , { 850, -980, 60, 850, 60}
09017 }
09018 , { { 1350, -230, 570, 1350, 570}
09019 , { 1300, -530, 520, 1300, 520}
09020 , { 1350, -230, 570, 1350, 570}
09021 , { 1300, -530, 520, 1300, 520}
09022 , { -230, -1320, -270, -230, -270}
09023 }
09024 }
09025 , { { { 590, 570, 590, 570, -90}
09026 , { 390, 370, 390, 370, -90}
```

```
09027      , { 440, 420, 440, 420, -360}
09028      , { 590, 570, 590, 570, -420}
09029      , { 330, 310, 330, 310, -470}
09030      }
09031      , {{ 270, 220, 270, 220, -320}
09032      , { 240, 220, 240, 220, -320}
09033      , { -60, -80, -60, -80, -830}
09034      , { 270, 10, 270, 10, -780}
09035      , { -60, -80, -60, -80, -870}
09036      }
09037      , {{ 390, 370, 390, 370, -90}
09038      , { 390, 370, 390, 370, -90}
09039      , { 380, 360, 380, 360, -420}
09040      , { 390, 370, 390, 370, -420}
09041      , { 330, 310, 330, 310, -470}
09042      }
09043      , {{ 590, 570, 590, 570, -810}
09044      , { -10, -270, -10, -270, -810}
09045      , { -60, -80, -60, -80, -870}
09046      , { 590, 570, 590, 570, -1470}
09047      , { -60, -80, -60, -80, -870}
09048      }
09049      , {{ 440, 420, 440, 420, -360}
09050      , { 390, 370, 390, 370, -420}
09051      , { 440, 420, 440, 420, -360}
09052      , { 390, 370, 390, 370, -420}
09053      , { -400, -420, -400, -420, -1210}
09054      }
09055      }
09056      }
09057      , {{{ 1320, 850, 720, 1320, 720}
09058      , { 1320, 670, 540, 1320, 540}
09059      , { 870, 220, 90, 870, 90}
09060      , { 960, 850, 720, 960, 720}
09061      , { 870, 250, 90, 870, 90}
09062      }
09063      , {{ 1320, 670, 540, 1320, 540}
09064      , { 1320, 670, 540, 1320, 540}
09065      , { 870, 220, 90, 870, 90}
09066      , { -410, -520, -410, -800, -640}
09067      , { 870, 220, 90, 870, 90}
09068      }
09069      , {{ 960, 300, 170, 960, 170}
09070      , { 960, 300, 170, 960, 170}
09071      , { 650, 0, -130, 650, -130}
09072      , { 960, 300, 170, 960, 170}
09073      , { 650, 0, -130, 650, -130}
09074      }
09075      , {{ 870, 850, 720, 870, 720}
09076      , { 70, -40, 70, -320, -170}
09077      , { 870, 220, 90, 870, 90}
09078      , { 850, 850, 720, 570, 720}
09079      , { 870, 220, 90, 870, 90}
09080      }
09081      , {{ 960, 300, 170, 960, 170}
09082      , { 960, 300, 170, 960, 170}
09083      , { 340, -310, -440, 340, -440}
09084      , { 960, 300, 170, 960, 170}
09085      , { 250, 250, -90, -260, -110}
09086      }
09087      }
09088      , {{{ 850, 850, 720, 540, 720}
09089      , { 670, 670, 540, 10, 540}
09090      , { 540, 220, 90, 540, 90}
09091      , { 850, 850, 720, -970, 720}
09092      , { 250, 250, 90, -810, 90}
09093      }
09094      , {{ 670, 670, 540, -100, 540}
09095      , { 670, 670, 540, -600, 540}
09096      , { 220, 220, 90, -100, 90}
09097      , { -520, -520, -650, -1790, -650}
09098      , { 220, 220, 90, -1050, 90}
09099      }
09100      , {{ 540, 300, 170, 540, 170}
09101      , { 300, 300, 170, 10, 170}
09102      , { 540, 0, -130, 540, -130}
09103      , { 300, 300, 170, -970, 170}
09104      , { 0, 0, -130, -1030, -130}
09105      }
09106      , {{ 850, 850, 720, -1050, 720}
09107      , { -40, -40, -170, -1320, -170}
09108      , { 220, 220, 90, -1050, 90}
09109      , { 850, 850, 720, -1680, 720}
09110      , { 220, 220, 90, -1050, 90}
09111      }
09112      , {{ 300, 300, 170, -810, 170}
09113      , { 300, 300, 170, -970, 170}
```

```
09114      , { -310, -310, -440, -1340, -440}
09115      , { 300, 300, 170, -970, 170}
09116      , { 250, 250, -90, -810, -110}
09117      }
09118      }
09119      , { { 720, 570, 720, 570, 480}
09120      , { 540, 390, 540, 390, 300}
09121      , { 90, -60, 90, -60, -140}
09122      , { 720, 570, 720, 570, 480}
09123      , { 90, -60, 90, -60, -140}
09124      }
09125      , { { 540, 390, 540, 390, 300}
09126      , { 540, 390, 540, 390, 300}
09127      , { 90, -60, 90, -60, -140}
09128      , { -410, -800, -410, -800, -640}
09129      , { 90, -60, 90, -60, -140}
09130      }
09131      , { { 170, 20, 170, 20, -60}
09132      , { 170, 20, 170, 20, -60}
09133      , { -130, -280, -130, -280, -360}
09134      , { 170, 20, 170, 20, -60}
09135      , { -130, -280, -130, -280, -360}
09136      }
09137      , { { 720, 570, 720, 570, 480}
09138      , { 70, -320, 70, -320, -170}
09139      , { 90, -60, 90, -60, -140}
09140      , { 720, 570, 720, 570, 480}
09141      , { 90, -60, 90, -60, -140}
09142      }
09143      , { { 170, 20, 170, 20, -60}
09144      , { 170, 20, 170, 20, -60}
09145      , { -440, -590, -440, -590, -670}
09146      , { 170, 20, 170, 20, -60}
09147      , { -110, -260, -110, -260, -350}
09148      }
09149      }
09150      , { { { 1320, -350, 720, 1320, 720}
09151      , { 1320, -730, 540, 1320, 540}
09152      , { 870, -350, 90, 870, 90}
09153      , { 960, -870, 720, 960, 720}
09154      , { 870, -940, 90, 870, 90}
09155      }
09156      , { { 1320, -350, 540, 1320, 540}
09157      , { 1320, -730, 540, 1320, 540}
09158      , { 870, -350, 90, 870, 90}
09159      , { -650, -1920, -650, -1120, -650}
09160      , { 870, -960, 90, 870, 90}
09161      }
09162      , { { 960, -870, 170, 960, 170}
09163      , { 960, -1100, 170, 960, 170}
09164      , { 650, -940, -130, 650, -130}
09165      , { 960, -870, 170, 960, 170}
09166      , { 650, -940, -130, 650, -130}
09167      }
09168      , { { 870, -960, 720, 870, 720}
09169      , { -170, -1450, -170, -640, -170}
09170      , { 870, -960, 90, 870, 90}
09171      , { 720, -1370, 720, -1000, 720}
09172      , { 870, -960, 90, 870, 90}
09173      }
09174      , { { 960, -870, 170, 960, 170}
09175      , { 960, -870, 170, 960, 170}
09176      , { 340, -1250, -440, 340, -440}
09177      , { 960, -870, 170, 960, 170}
09178      , { -110, -1360, -110, -580, -110}
09179      }
09180      }
09181      , { { { 590, 570, 590, 570, -160}
09182      , { 410, 390, 410, 390, -160}
09183      , { -40, -60, -40, -60, -850}
09184      , { 590, 570, 590, 570, -230}
09185      , { -40, -60, -40, -60, -850}
09186      }
09187      , { { 410, 390, 410, 390, -160}
09188      , { 410, 390, 410, 390, -160}
09189      , { -40, -60, -40, -60, -850}
09190      , { -540, -800, -540, -800, -1520}
09191      , { -40, -60, -40, -60, -850}
09192      }
09193      , { { 40, 20, 40, 20, -400}
09194      , { 40, 20, 40, 20, -400}
09195      , { -260, -280, -260, -280, -1070}
09196      , { 40, 20, 40, 20, -760}
09197      , { -260, -280, -260, -280, -1070}
09198      }
09199      , { { 590, 570, 590, 570, -230}
09200      , { -60, -320, -60, -320, -1110}
```

```
09201      , {    -40,    -60,    -40,    -60,   -850}
09202      , {    590,    570,    590,    570,   -230}
09203      , {   -40,   -60,   -40,   -60,   -850}
09204      }
09205      , { {     40,     20,     40,     20,   -760}
09206      , {     40,     20,     40,     20,   -760}
09207      , {   -570,   -590,   -570,   -590,  -1380}
09208      , {     40,     20,     40,     20,   -760}
09209      , {   -240,   -260,   -240,   -260,  -1050}
09210      }
09211      }
09212      }
09213      , { { {  1010,  1010,   880,   730,   880}
09214      , {    410,   -30,     40,   410,   -190}
09215      , {    410,  -240,   -370,   410,   -370}
09216      , {  1010,  1010,   880,   730,   880}
09217      , {    410,     0,   -370,   410,   -370}
09218      }
09219      , { {    410,   -70,  -150,   410,   -370}
09220      , {    230,   -70,  -550,   230,  -550}
09221      , {    410,  -240,   -370,   410,   -370}
09222      , {   -150,  -260,  -150,  -540,  -380}
09223      , {    410,  -240,   -370,   410,   -370}
09224      }
09225      , { {    410,  -240,   -370,   410,   -370}
09226      , {    410,  -240,   -370,   410,   -370}
09227      , {    410,  -240,   -370,   410,   -370}
09228      , {    410,  -240,   -370,   410,   -370}
09229      , {    410,  -240,   -370,   410,   -370}
09230      }
09231      , { {  1010,  1010,   880,   730,   880}
09232      , {     40,   -30,     40,  -350,  -190}
09233      , {    410,  -240,   -370,   410,   -370}
09234      , {  1010,  1010,   880,   730,   880}
09235      , {    410,  -240,   -370,   410,   -370}
09236      }
09237      , { {    410,     0,   -370,   410,   -370}
09238      , {    410,  -240,   -370,   410,   -370}
09239      , {    410,  -240,   -370,   410,   -370}
09240      , {    410,  -240,   -370,   410,   -370}
09241      , {     0,     0,   -370,  -520,  -370}
09242      }
09243      }
09244      , { { {  1010,  1010,   880, -1280,   880}
09245      , {    -30,   -30,   -200, -1340,  -200}
09246      , {   -240,  -240,   -370, -1280,  -370}
09247      , {  1010,  1010,   880, -1520,   880}
09248      , {     0,     0,   -370, -1280,  -370}
09249      }
09250      , { {    -70,   -70,   -370, -1520,  -370}
09251      , {    -70,   -70,  -550, -1700,  -550}
09252      , {   -240,  -240,   -370, -1520,  -370}
09253      , {   -260,  -260,  -390, -1530,  -390}
09254      , {   -240,  -240,   -370, -1520,  -370}
09255      }
09256      , { {   -240,  -240,   -370, -1280,  -370}
09257      , {   -240,  -240,   -370, -1520,  -370}
09258      , {   -240,  -240,   -370, -1280,  -370}
09259      , {   -240,  -240,   -370, -1520,  -370}
09260      , {   -240,  -240,   -370, -1280,  -370}
09261      }
09262      , { {  1010,  1010,   880, -1340,   880}
09263      , {    -30,   -30,   -200, -1340,  -200}
09264      , {   -240,  -240,   -370, -1520,  -370}
09265      , {  1010,  1010,   880, -1520,   880}
09266      , {   -240,  -240,   -370, -1520,  -370}
09267      }
09268      , { {     0,     0,   -370, -1280,  -370}
09269      , {   -240,  -240,   -370, -1520,  -370}
09270      , {   -240,  -240,   -370, -1280,  -370}
09271      , {   -240,  -240,   -370, -1520,  -370}
09272      , {     0,     0,   -370, -1520,  -370}
09273      }
09274      }
09275      , { { {   880,   730,   880,   730,   640}
09276      , {     40,  -350,     40,  -350,  -190}
09277      , {   -370,  -520,   -370,  -520,  -610}
09278      , {   880,   730,   880,   730,   640}
09279      , {   -370,  -520,   -370,  -520,  -610}
09280      }
09281      , { {   -150,  -520,  -150,  -520,  -380}
09282      , {   -550,  -700,  -550,  -700,  -790}
09283      , {   -370,  -520,   -370,  -520,  -610}
09284      , {   -150,  -540,  -150,  -540,  -380}
09285      , {   -370,  -520,   -370,  -520,  -610}
09286      }
09287      , { {   -370,  -520,   -370,  -520,  -610}
```

```
09288 , { -370, -520, -370, -520, -610}
09289 , { -370, -520, -370, -520, -610}
09290 , { -370, -520, -370, -520, -610}
09291 , { -370, -520, -370, -520, -610}
09292 }
09293 , { { 880, 730, 880, 730, 640}
09294 , { 40, -350, 40, -350, -190}
09295 , { -370, -520, -370, -520, -610}
09296 , { 880, 730, 880, 730, 640}
09297 , { -370, -520, -370, -520, -610}
09298 }
09299 , { { -370, -520, -370, -520, -610}
09300 , { -370, -520, -370, -520, -610}
09301 , { -370, -520, -370, -520, -610}
09302 , { -370, -520, -370, -520, -610}
09303 , { -370, -520, -370, -520, -610}
09304 }
09305 }
09306 , { { { 880, -1180, 880, 410, 880}
09307 , { 410, -1250, -200, 410, -200}
09308 , { 410, -1180, -370, 410, -370}
09309 , { 880, -1420, 880, 410, 880}
09310 , { 410, -1180, -370, 410, -370}
09311 }
09312 , { { 410, -1420, -370, 410, -370}
09313 , { 230, -1600, -550, 230, -550}
09314 , { 410, -1420, -370, 410, -370}
09315 , { -390, -1440, -390, -860, -390}
09316 , { 410, -1420, -370, 410, -370}
09317 }
09318 , { { 410, -1180, -370, 410, -370}
09319 , { 410, -1420, -370, 410, -370}
09320 , { 410, -1180, -370, 410, -370}
09321 , { 410, -1420, -370, 410, -370}
09322 , { 410, -1180, -370, 410, -370}
09323 }
09324 , { { 880, -1250, 880, 410, 880}
09325 , { -200, -1250, -200, -670, -200}
09326 , { 410, -1420, -370, 410, -370}
09327 , { 880, -1420, 880, -840, 880}
09328 , { 410, -1420, -370, 410, -370}
09329 }
09330 , { { 410, -1180, -370, 410, -370}
09331 , { 410, -1420, -370, 410, -370}
09332 , { 410, -1180, -370, 410, -370}
09333 , { 410, -1420, -370, 410, -370}
09334 , { -370, -1420, -370, -840, -370}
09335 }
09336 }
09337 , { { { 750, 730, 750, 730, -1140}
09338 , { -90, -350, -90, -350, -1140}
09339 , { -500, -520, -500, -520, -1310}
09340 , { 750, 730, 750, 730, -1310}
09341 , { -500, -520, -500, -520, -1310}
09342 }
09343 , { { -280, -520, -280, -520, -1250}
09344 , { -680, -700, -680, -700, -1250}
09345 , { -500, -520, -500, -520, -1310}
09346 , { -280, -540, -280, -540, -1330}
09347 , { -500, -520, -500, -520, -1310}
09348 }
09349 , { { -500, -520, -500, -520, -1310}
09350 , { -500, -520, -500, -520, -1310}
09351 , { -500, -520, -500, -520, -1310}
09352 , { -500, -520, -500, -520, -1310}
09353 , { -500, -520, -500, -520, -1310}
09354 }
09355 , { { 750, 730, 750, 730, -1140}
09356 , { -90, -350, -90, -350, -1140}
09357 , { -500, -520, -500, -520, -1310}
09358 , { 750, 730, 750, 730, -1310}
09359 , { -500, -520, -500, -520, -1310}
09360 }
09361 , { { -500, -520, -500, -520, -1310}
09362 , { -500, -520, -500, -520, -1310}
09363 , { -500, -520, -500, -520, -1310}
09364 , { -500, -520, -500, -520, -1310}
09365 , { -500, -520, -500, -520, -1310}
09366 }
09367 }
09368 }
09369 , { { { { 1560, 1560, 1430, 1470, 1430}
09370 , { 1470, 820, 690, 1470, 690}
09371 , { 960, 310, 180, 960, 180}
09372 , { 1560, 1560, 1430, 1280, 1430}
09373 , { 960, 550, 180, 960, 180}
09374 }
```

```

09375 ,{{ 1470, 820, 690, 1470, 690}
09376 ,{ 1470, 820, 690, 1470, 690}
09377 ,{ 960, 310, 180, 960, 180}
09378 ,{ 80, -30, 80, -310, -150}
09379 ,{ 960, 310, 180, 960, 180}
09380 }
09381 ,{{ 960, 310, 180, 960, 180}
09382 ,{ 960, 310, 180, 960, 180}
09383 ,{ 960, 310, 180, 960, 180}
09384 ,{ 960, 310, 180, 960, 180}
09385 ,{ 960, 310, 180, 960, 180}
09386 }
09387 ,{{ 1560, 1560, 1430, 1280, 1430}
09388 ,{ -90, -200, -90, -480, -320}
09389 ,{ 960, 310, 180, 960, 180}
09390 ,{ 1560, 1560, 1430, 1280, 1430}
09391 ,{ 960, 310, 180, 960, 180}
09392 }
09393 ,{{ 960, 550, 180, 960, 180}
09394 ,{ 960, 310, 180, 960, 180}
09395 ,{ 960, 310, 180, 960, 180}
09396 ,{ 960, 310, 180, 960, 180}
09397 ,{ 550, 550, 180, 30, 180}
09398 }
09399 }
09400 ,{{{ 1560, 1560, 1430, -30, 1430}
09401 ,{ 820, 820, 690, -30, 690}
09402 ,{ 310, 310, 180, -720, 180}
09403 ,{ 1560, 1560, 1430, -960, 1430}
09404 ,{ 550, 550, 180, -720, 180}
09405 }
09406 ,{{ 820, 820, 690, -30, 690}
09407 ,{ 820, 820, 690, -30, 690}
09408 ,{ 310, 310, 180, -960, 180}
09409 ,{ -30, -30, -160, -1300, -160}
09410 ,{ 310, 310, 180, -960, 180}
09411 }
09412 ,{{ 310, 310, 180, -720, 180}
09413 ,{ 310, 310, 180, -960, 180}
09414 ,{ 310, 310, 180, -720, 180}
09415 ,{ 310, 310, 180, -960, 180}
09416 ,{ 310, 310, 180, -720, 180}
09417 }
09418 ,{{ 1560, 1560, 1430, -960, 1430}
09419 ,{ -200, -200, -330, -1470, -330}
09420 ,{ 310, 310, 180, -960, 180}
09421 ,{ 1560, 1560, 1430, -960, 1430}
09422 ,{ 310, 310, 180, -960, 180}
09423 }
09424 ,{{ 550, 550, 180, -720, 180}
09425 ,{ 310, 310, 180, -960, 180}
09426 ,{ 310, 310, 180, -720, 180}
09427 ,{ 310, 310, 180, -960, 180}
09428 ,{ 550, 550, 180, -960, 180}
09429 }
09430 }
09431 ,{{{ 1430, 1280, 1430, 1280, 1200}
09432 ,{ 690, 540, 690, 540, 450}
09433 ,{ 180, 30, 180, 30, -50}
09434 ,{ 1430, 1280, 1430, 1280, 1200}
09435 ,{ 180, 30, 180, 30, -50}
09436 }
09437 ,{{ 690, 540, 690, 540, 450}
09438 ,{ 690, 540, 690, 540, 450}
09439 ,{ 180, 30, 180, 30, -50}
09440 ,{ 80, -310, 80, -310, -150}
09441 ,{ 180, 30, 180, 30, -50}
09442 }
09443 ,{{ 180, 30, 180, 30, -50}
09444 ,{ 180, 30, 180, 30, -50}
09445 ,{ 180, 30, 180, 30, -50}
09446 ,{ 180, 30, 180, 30, -50}
09447 ,{ 180, 30, 180, 30, -50}
09448 }
09449 ,{{ 1430, 1280, 1430, 1280, 1200}
09450 ,{ -90, -480, -90, -480, -320}
09451 ,{ 180, 30, 180, 30, -50}
09452 ,{ 1430, 1280, 1430, 1280, 1200}
09453 ,{ 180, 30, 180, 30, -50}
09454 }
09455 ,{{ 180, 30, 180, 30, -50}
09456 ,{ 180, 30, 180, 30, -50}
09457 ,{ 180, 30, 180, 30, -50}
09458 ,{ 180, 30, 180, 30, -50}
09459 ,{ 180, 30, 180, 30, -50}
09460 }
09461 }

```



```
09462 ,{{{ 1470, -360, 1430, 1470, 1430}
09463 ,{ 1470, -360, 690, 1470, 690}
09464 ,{ 960, -630, 180, 960, 180}
09465 ,{ 1430, -870, 1430, 960, 1430}
09466 ,{ 960, -630, 180, 960, 180}
09467 }
09468 ,{{{ 1470, -360, 690, 1470, 690}
09469 ,{ 1470, -360, 690, 1470, 690}
09470 ,{ 960, -870, 180, 960, 180}
09471 ,{ -160, -1210, -160, -630, -160}
09472 ,{ 960, -870, 180, 960, 180}
09473 }
09474 ,{{{ 960, -630, 180, 960, 180}
09475 ,{ 960, -870, 180, 960, 180}
09476 ,{ 960, -630, 180, 960, 180}
09477 ,{ 960, -870, 180, 960, 180}
09478 ,{ 960, -630, 180, 960, 180}
09479 }
09480 ,{{{ 1430, -870, 1430, 960, 1430}
09481 ,{ -330, -1380, -330, -800, -330}
09482 ,{ 960, -870, 180, 960, 180}
09483 ,{ 1430, -870, 1430, -290, 1430}
09484 ,{ 960, -870, 180, 960, 180}
09485 }
09486 ,{{{ 960, -630, 180, 960, 180}
09487 ,{ 960, -870, 180, 960, 180}
09488 ,{ 960, -630, 180, 960, 180}
09489 ,{ 960, -870, 180, 960, 180}
09490 ,{ 180, -870, 180, -290, 180}
09491 }
09492 }
09493 ,{{{ 1300, 1280, 1300, 1280, -10}
09494 ,{ 560, 540, 560, 540, -10}
09495 ,{ 50, 30, 50, 30, -760}
09496 ,{ 1300, 1280, 1300, 1280, -760}
09497 ,{ 50, 30, 50, 30, -760}
09498 }
09499 ,{{{ 560, 540, 560, 540, -10}
09500 ,{ 560, 540, 560, 540, -10}
09501 ,{ 50, 30, 50, 30, -760}
09502 ,{ -50, -310, -50, -310, -1100}
09503 ,{ 50, 30, 50, 30, -760}
09504 }
09505 ,{{{ 50, 30, 50, 30, -760}
09506 ,{ 50, 30, 50, 30, -760}
09507 ,{ 50, 30, 50, 30, -760}
09508 ,{ 50, 30, 50, 30, -760}
09509 ,{ 50, 30, 50, 30, -760}
09510 }
09511 ,{{{ 1300, 1280, 1300, 1280, -760}
09512 ,{ -220, -480, -220, -480, -1270}
09513 ,{ 50, 30, 50, 30, -760}
09514 ,{ 1300, 1280, 1300, 1280, -760}
09515 ,{ 50, 30, 50, 30, -760}
09516 }
09517 ,{{{ 50, 30, 50, 30, -760}
09518 ,{ 50, 30, 50, 30, -760}
09519 ,{ 50, 30, 50, 30, -760}
09520 ,{ 50, 30, 50, 30, -760}
09521 ,{ 50, 30, 50, 30, -760}
09522 }
09523 }
09524 }
09525 ,{{{ 2050, 1930, 1800, 2050, 1800}
09526 ,{ 2050, 1400, 1270, 2050, 1270}
09527 ,{ 1750, 1100, 970, 1750, 970}
09528 ,{ 1930, 1930, 1800, 1760, 1800}
09529 ,{ 1750, 1100, 970, 1750, 970}
09530 }
09531 ,{{{ 2050, 1400, 1270, 2050, 1270}
09532 ,{ 2050, 1400, 1270, 2050, 1270}
09533 ,{ 1740, 1090, 960, 1740, 960}
09534 ,{ 130, 10, 130, -260, -110}
09535 ,{ 1740, 1090, 960, 1740, 960}
09536 }
09537 ,{{{ 1760, 1110, 980, 1760, 980}
09538 ,{ 1760, 1110, 980, 1760, 980}
09539 ,{ 1750, 1100, 970, 1750, 970}
09540 ,{ 1760, 1110, 980, 1760, 980}
09541 ,{ 1750, 1100, 970, 1750, 970}
09542 }
09543 ,{{{ 1930, 1930, 1800, 1740, 1800}
09544 ,{ 300, 190, 300, -80, 70}
09545 ,{ 1740, 1090, 960, 1740, 960}
09546 ,{ 1930, 1930, 1800, 1650, 1800}
09547 ,{ 1740, 1090, 960, 1740, 960}
09548 }
```

```

09549 ,{{ 1760, 1110, 980, 1760, 980}
09550 ,{ 1760, 1110, 980, 1760, 980}
09551 ,{ 1750, 1100, 970, 1750, 970}
09552 ,{ 1760, 1110, 980, 1760, 980}
09553 ,{ 360, 360, 0, -150, 0}
09554 }
09555 }
09556 ,{{{ 1930, 1930, 1800, 130, 1800}
09557 ,{ 1400, 1400, 1270, 130, 1270}
09558 ,{ 1100, 1100, 970, 70, 970}
09559 ,{ 1930, 1930, 1800, -160, 1800}
09560 ,{ 1100, 1100, 970, 70, 970}
09561 }
09562 ,{{ 1400, 1400, 1270, 130, 1270}
09563 ,{ 1400, 1400, 1270, 130, 1270}
09564 ,{ 1090, 1090, 960, -180, 960}
09565 ,{ 10, 10, -110, -1260, -110}
09566 ,{ 1090, 1090, 960, -180, 960}
09567 }
09568 ,{{ 1110, 1110, 980, 70, 980}
09569 ,{ 1110, 1110, 980, -160, 980}
09570 ,{ 1100, 1100, 970, 70, 970}
09571 ,{ 1110, 1110, 980, -160, 980}
09572 ,{ 1100, 1100, 970, 70, 970}
09573 }
09574 ,{{{ 1930, 1930, 1800, -180, 1800}
09575 ,{ 190, 190, 60, -1080, 60}
09576 ,{ 1090, 1090, 960, -180, 960}
09577 ,{ 1930, 1930, 1800, -590, 1800}
09578 ,{ 1090, 1090, 960, -180, 960}
09579 }
09580 ,{{{ 1110, 1110, 980, 70, 980}
09581 ,{ 1110, 1110, 980, -160, 980}
09582 ,{ 1100, 1100, 970, 70, 970}
09583 ,{ 1110, 1110, 980, -160, 980}
09584 ,{ 360, 360, 0, -1150, 0}
09585 }
09586 }
09587 ,{{{ 1800, 1650, 1800, 1650, 1570}
09588 ,{ 1270, 1120, 1270, 1120, 1040}
09589 ,{ 970, 820, 970, 820, 740}
09590 ,{ 1800, 1650, 1800, 1650, 1570}
09591 ,{ 970, 820, 970, 820, 740}
09592 }
09593 ,{{{ 1270, 1120, 1270, 1120, 1040}
09594 ,{ 1270, 1120, 1270, 1120, 1040}
09595 ,{ 960, 810, 960, 810, 730}
09596 ,{ 130, -260, 130, -260, -110}
09597 ,{ 960, 810, 960, 810, 730}
09598 }
09599 ,{{{ 980, 830, 980, 830, 740}
09600 ,{ 980, 830, 980, 830, 740}
09601 ,{ 970, 820, 970, 820, 740}
09602 ,{ 980, 830, 980, 830, 740}
09603 ,{ 970, 820, 970, 820, 740}
09604 }
09605 ,{{{ 1800, 1650, 1800, 1650, 1570}
09606 ,{ 300, -80, 300, -80, 70}
09607 ,{ 960, 810, 960, 810, 730}
09608 ,{ 1800, 1650, 1800, 1650, 1570}
09609 ,{ 960, 810, 960, 810, 730}
09610 }
09611 ,{{{ 980, 830, 980, 830, 740}
09612 ,{ 980, 830, 980, 830, 740}
09613 ,{ 970, 820, 970, 820, 740}
09614 ,{ 980, 830, 980, 830, 740}
09615 ,{ 0, -150, 0, -150, -240}
09616 }
09617 }
09618 ,{{{ 2050, 220, 1800, 2050, 1800}
09619 ,{ 2050, 220, 1270, 2050, 1270}
09620 ,{ 1750, 170, 970, 1750, 970}
09621 ,{ 1800, -70, 1800, 1760, 1800}
09622 ,{ 1750, 170, 970, 1750, 970}
09623 }
09624 ,{{{ 2050, 220, 1270, 2050, 1270}
09625 ,{ 2050, 220, 1270, 2050, 1270}
09626 ,{ 1740, -80, 960, 1740, 960}
09627 ,{ -110, -1160, -110, -580, -110}
09628 ,{ 1740, -80, 960, 1740, 960}
09629 }
09630 ,{{{ 1760, 170, 980, 1760, 980}
09631 ,{ 1760, -70, 980, 1760, 980}
09632 ,{ 1750, 170, 970, 1750, 970}
09633 ,{ 1760, -70, 980, 1760, 980}
09634 ,{ 1750, 170, 970, 1750, 970}
09635 }

```

```
09636 ,{{ 1800, -80, 1800, 1740, 1800}
09637 ,{ 60, -980, 60, -400, 60}
09638 ,{ 1740, -80, 960, 1740, 960}
09639 ,{ 1800, -490, 1800, 80, 1800}
09640 ,{ 1740, -80, 960, 1740, 960}
09641 }
09642 ,{{ 1760, 170, 980, 1760, 980}
09643 ,{ 1760, -70, 980, 1760, 980}
09644 ,{ 1750, 170, 970, 1750, 970}
09645 ,{ 1760, -70, 980, 1760, 980}
09646 ,{ 0, -1050, 0, -470, 0}
09647 }
09648 }
09649 ,{{{ 1670, 1650, 1670, 1650, 570}
09650 ,{ 1140, 1120, 1140, 1120, 570}
09651 ,{ 840, 820, 840, 820, 30}
09652 ,{ 1670, 1650, 1670, 1650, 40}
09653 ,{ 840, 820, 840, 820, 30}
09654 }
09655 ,{{ 1140, 1120, 1140, 1120, 570}
09656 ,{ 1140, 1120, 1140, 1120, 570}
09657 ,{ 830, 810, 830, 810, 20}
09658 ,{ 0, -260, 0, -260, -1050}
09659 ,{ 830, 810, 830, 810, 20}
09660 }
09661 ,{{ 850, 830, 850, 830, 40}
09662 ,{ 850, 830, 850, 830, 40}
09663 ,{ 840, 820, 840, 820, 30}
09664 ,{ 850, 830, 850, 830, 40}
09665 ,{ 840, 820, 840, 820, 30}
09666 }
09667 ,{{{ 1670, 1650, 1670, 1650, 20}
09668 ,{ 180, -80, 180, -80, -870}
09669 ,{ 830, 810, 830, 810, 20}
09670 ,{ 1670, 1650, 1670, 1650, -380}
09671 ,{ 830, 810, 830, 810, 20}
09672 }
09673 ,{{{ 850, 830, 850, 830, 40}
09674 ,{ 850, 830, 850, 830, 40}
09675 ,{ 840, 820, 840, 820, 30}
09676 ,{ 850, 830, 850, 830, 40}
09677 ,{ -130, -150, -130, -150, -940}
09678 }
09679 }
09680 }
09681 ,{{{ 2120, 2120, 1990, 2120, 1990}
09682 ,{ 2120, 1470, 1340, 2120, 1340}
09683 ,{ 1990, 1340, 1210, 1990, 1210}
09684 ,{ 2120, 2120, 1990, 1990, 1990}
09685 ,{ 1860, 1210, 1080, 1860, 1080}
09686 }
09687 ,{{ 2120, 1470, 1340, 2120, 1340}
09688 ,{ 2120, 1470, 1340, 2120, 1340}
09689 ,{ 1840, 1190, 1060, 1840, 1060}
09690 ,{ 180, 60, 180, -210, -60}
09691 ,{ 1840, 1190, 1060, 1840, 1060}
09692 }
09693 ,{{ 1990, 1340, 1210, 1990, 1210}
09694 ,{ 1990, 1340, 1210, 1990, 1210}
09695 ,{ 1990, 1340, 1210, 1990, 1210}
09696 ,{ 1990, 1340, 1210, 1990, 1210}
09697 ,{ 1860, 1210, 1080, 1860, 1080}
09698 }
09699 ,{{ 2120, 2120, 1990, 1840, 1990}
09700 ,{ -120, -230, -120, -510, -360}
09701 ,{ 1840, 1190, 1060, 1840, 1060}
09702 ,{ 2120, 2120, 1990, 1840, 1990}
09703 ,{ 1840, 1190, 1060, 1840, 1060}
09704 }
09705 ,{{ 1990, 1340, 1210, 1990, 1210}
09706 ,{ 1990, 1340, 1210, 1990, 1210}
09707 ,{ 1550, 900, 770, 1550, 770}
09708 ,{ 1990, 1340, 1210, 1990, 1210}
09709 ,{ 640, 640, 270, 120, 270}
09710 }
09711 }
09712 ,{{{ 2120, 2120, 1990, 300, 1990}
09713 ,{ 1470, 1470, 1340, 190, 1340}
09714 ,{ 1340, 1340, 1210, 300, 1210}
09715 ,{ 2120, 2120, 1990, 60, 1990}
09716 ,{ 1210, 1210, 1080, 180, 1080}
09717 }
09718 ,{{{ 1470, 1470, 1340, 190, 1340}
09719 ,{ 1470, 1470, 1340, 190, 1340}
09720 ,{ 1190, 1190, 1060, -80, 1060}
09721 ,{ 60, 60, -60, -1210, -60}
09722 ,{ 1190, 1190, 1060, -80, 1060}
```

```
09723      }
09724      ,{{ 1340, 1340, 1210, 300, 1210}
09725      ,{ 1340, 1340, 1210, 60, 1210}
09726      ,{ 1340, 1340, 1210, 300, 1210}
09727      ,{ 1340, 1340, 1210, 60, 1210}
09728      ,{ 1210, 1210, 1080, 180, 1080}
09729      }
09730      ,{{ 2120, 2120, 1990, -80, 1990}
09731      ,{ -230, -230, -360, -1510, -360}
09732      ,{ 1190, 1190, 1060, -80, 1060}
09733      ,{ 2120, 2120, 1990, -400, 1990}
09734      ,{ 1190, 1190, 1060, -80, 1060}
09735      }
09736      ,{{ 1340, 1340, 1210, 60, 1210}
09737      ,{ 1340, 1340, 1210, 60, 1210}
09738      ,{ 900, 900, 770, -130, 770}
09739      ,{ 1340, 1340, 1210, 60, 1210}
09740      ,{ 640, 640, 270, -870, 270}
09741      }
09742      }
09743      ,{{{ 1990, 1840, 1990, 1840, 1750}
09744      ,{ 1340, 1190, 1340, 1190, 1100}
09745      ,{ 1210, 1060, 1210, 1060, 970}
09746      ,{ 1990, 1840, 1990, 1840, 1750}
09747      ,{ 1080, 930, 1080, 930, 840}
09748      }
09749      ,{{{ 1340, 1190, 1340, 1190, 1100}
09750      ,{ 1340, 1190, 1340, 1190, 1100}
09751      ,{ 1060, 910, 1060, 910, 820}
09752      ,{ 180, -210, 180, -210, -60}
09753      ,{ 1060, 910, 1060, 910, 820}
09754      }
09755      ,{{{ 1210, 1060, 1210, 1060, 970}
09756      ,{ 1210, 1060, 1210, 1060, 970}
09757      ,{ 1210, 1060, 1210, 1060, 970}
09758      ,{ 1210, 1060, 1210, 1060, 970}
09759      ,{ 1080, 930, 1080, 930, 840}
09760      }
09761      ,{{{ 1990, 1840, 1990, 1840, 1750}
09762      ,{ -120, -510, -120, -510, -360}
09763      ,{ 1060, 910, 1060, 910, 820}
09764      ,{ 1990, 1840, 1990, 1840, 1750}
09765      ,{ 1060, 910, 1060, 910, 820}
09766      }
09767      ,{{{ 1210, 1060, 1210, 1060, 970}
09768      ,{ 1210, 1060, 1210, 1060, 970}
09769      ,{ 770, 620, 770, 620, 530}
09770      ,{ 1210, 1060, 1210, 1060, 970}
09771      ,{ 270, 120, 270, 120, 30}
09772      }
09773      }
09774      ,{{{ 2120, 400, 1990, 2120, 1990}
09775      ,{ 2120, 290, 1340, 2120, 1340}
09776      ,{ 1990, 400, 1210, 1990, 1210}
09777      ,{ 1990, 160, 1990, 1990, 1990}
09778      ,{ 1860, 270, 1080, 1860, 1080}
09779      }
09780      ,{{ 2120, 290, 1340, 2120, 1340}
09781      ,{ 2120, 290, 1340, 2120, 1340}
09782      ,{ 1840, 10, 1060, 1840, 1060}
09783      ,{ -60, -1110, -60, -530, -60}
09784      ,{ 1840, 10, 1060, 1840, 1060}
09785      }
09786      ,{{ 1990, 400, 1210, 1990, 1210}
09787      ,{ 1990, 160, 1210, 1990, 1210}
09788      ,{ 1990, 400, 1210, 1990, 1210}
09789      ,{ 1990, 160, 1210, 1990, 1210}
09790      ,{ 1860, 270, 1080, 1860, 1080}
09791      }
09792      ,{{ 1990, 10, 1990, 1840, 1990}
09793      ,{ -360, -1410, -360, -830, -360}
09794      ,{ 1840, 10, 1060, 1840, 1060}
09795      ,{ 1990, -310, 1990, 270, 1990}
09796      ,{ 1840, 10, 1060, 1840, 1060}
09797      }
09798      ,{{ 1990, 160, 1210, 1990, 1210}
09799      ,{ 1990, 160, 1210, 1990, 1210}
09800      ,{ 1550, -40, 770, 1550, 770}
09801      ,{ 1990, 160, 1210, 1990, 1210}
09802      ,{ 270, -780, 270, -200, 270}
09803      }
09804      }
09805      ,{{{ 1860, 1840, 1860, 1840, 640}
09806      ,{ 1210, 1190, 1210, 1190, 640}
09807      ,{ 1080, 1060, 1080, 1060, 270}
09808      ,{ 1860, 1840, 1860, 1840, 270}
09809      ,{ 950, 930, 950, 930, 140}
```

```
09810     }
09811     ,{{ 1210, 1190, 1210, 1190, 640}
09812     ,{ 1210, 1190, 1210, 1190, 640}
09813     ,{ 930, 910, 930, 910, 120}
09814     ,{ 50, -210, 50, -210, -1000}
09815     ,{ 930, 910, 930, 910, 120}
09816     }
09817     ,{{ 1080, 1060, 1080, 1060, 270}
09818     ,{ 1080, 1060, 1080, 1060, 270}
09819     ,{ 1080, 1060, 1080, 1060, 270}
09820     ,{ 1080, 1060, 1080, 1060, 270}
09821     ,{ 950, 930, 950, 930, 140}
09822     }
09823     ,{{ 1860, 1840, 1860, 1840, 120}
09824     ,{ -250, -510, -250, -510, -1300}
09825     ,{ 930, 910, 930, 910, 120}
09826     ,{ 1860, 1840, 1860, 1840, -200}
09827     ,{ 930, 910, 930, 910, 120}
09828     }
09829     ,{{ 1080, 1060, 1080, 1060, 270}
09830     ,{ 1080, 1060, 1080, 1060, 270}
09831     ,{ 640, 620, 640, 620, -170}
09832     ,{ 1080, 1060, 1080, 1060, 270}
09833     ,{ 140, 120, 140, 120, -670}
09834     }
09835     }
09836     }
09837     ,{{{ 2120, 2120, 1990, 2120, 1990}
09838     ,{ 2120, 1470, 1340, 2120, 1340}
09839     ,{ 1990, 1340, 1210, 1990, 1210}
09840     ,{ 2120, 2120, 1990, 1990, 1990}
09841     ,{ 1860, 1210, 1080, 1860, 1080}
09842     }
09843     ,{{ 2120, 1470, 1340, 2120, 1340}
09844     ,{ 2120, 1470, 1340, 2120, 1340}
09845     ,{ 1840, 1190, 1060, 1840, 1060}
09846     ,{ 400, 290, 400, 10, 170}
09847     ,{ 1840, 1190, 1060, 1840, 1060}
09848     }
09849     ,{{ 1990, 1340, 1210, 1990, 1210}
09850     ,{ 1990, 1340, 1210, 1990, 1210}
09851     ,{ 1990, 1340, 1210, 1990, 1210}
09852     ,{ 1990, 1340, 1210, 1990, 1210}
09853     ,{ 1860, 1210, 1080, 1860, 1080}
09854     }
09855     ,{{ 2120, 2120, 1990, 1840, 1990}
09856     ,{ 540, 190, 540, -80, 70}
09857     ,{ 1840, 1190, 1060, 1840, 1060}
09858     ,{ 2120, 2120, 1990, 1840, 1990}
09859     ,{ 1840, 1190, 1060, 1840, 1060}
09860     }
09861     ,{{ 1990, 1340, 1210, 1990, 1210}
09862     ,{ 1990, 1340, 1210, 1990, 1210}
09863     ,{ 1750, 1100, 970, 1750, 970}
09864     ,{ 1990, 1340, 1210, 1990, 1210}
09865     ,{ 640, 640, 270, 120, 270}
09866     }
09867     }
09868     ,{{{ 2120, 2120, 1990, 540, 1990}
09869     ,{ 1470, 1470, 1340, 190, 1340}
09870     ,{ 1340, 1340, 1210, 540, 1210}
09871     ,{ 2120, 2120, 1990, 60, 1990}
09872     ,{ 1210, 1210, 1080, 180, 1080}
09873     }
09874     ,{{{ 1470, 1470, 1340, 190, 1340}
09875     ,{ 1470, 1470, 1340, 190, 1340}
09876     ,{ 1190, 1190, 1060, -80, 1060}
09877     ,{ 290, 290, 160, -980, 160}
09878     ,{ 1190, 1190, 1060, -80, 1060}
09879     }
09880     ,{{{ 1340, 1340, 1210, 540, 1210}
09881     ,{ 1340, 1340, 1210, 60, 1210}
09882     ,{ 1340, 1340, 1210, 540, 1210}
09883     ,{ 1340, 1340, 1210, 60, 1210}
09884     ,{ 1210, 1210, 1080, 180, 1080}
09885     }
09886     ,{{ 2120, 2120, 1990, -80, 1990}
09887     ,{ 190, 190, 60, -1080, 60}
09888     ,{ 1190, 1190, 1060, -80, 1060}
09889     ,{ 2120, 2120, 1990, -400, 1990}
09890     ,{ 1190, 1190, 1060, -80, 1060}
09891     }
09892     ,{{{ 1340, 1340, 1210, 70, 1210}
09893     ,{ 1340, 1340, 1210, 60, 1210}
09894     ,{ 1100, 1100, 970, 70, 970}
09895     ,{ 1340, 1340, 1210, 60, 1210}
09896     ,{ 640, 640, 270, -810, 270}
```

```

09897     }
09898     }
09899     ,{{{ 1990, 1840, 1990, 1840, 1750}
09900     ,{ 1340, 1190, 1340, 1190, 1100}
09901     ,{ 1210, 1060, 1210, 1060, 970}
09902     ,{ 1990, 1840, 1990, 1840, 1750}
09903     ,{ 1080, 930, 1080, 930, 840}
09904     }
09905     ,{{{ 1340, 1190, 1340, 1190, 1100}
09906     ,{ 1340, 1190, 1340, 1190, 1100}
09907     ,{ 1060, 910, 1060, 910, 820}
09908     ,{ 400, 10, 400, 10, 170}
09909     ,{ 1060, 910, 1060, 910, 820}
09910     }
09911     ,{{{ 1210, 1060, 1210, 1060, 970}
09912     ,{ 1210, 1060, 1210, 1060, 970}
09913     ,{ 1210, 1060, 1210, 1060, 970}
09914     ,{ 1210, 1060, 1210, 1060, 970}
09915     ,{ 1080, 930, 1080, 930, 840}
09916     }
09917     ,{{{ 1990, 1840, 1990, 1840, 1750}
09918     ,{ 540, -80, 540, -80, 70}
09919     ,{ 1060, 910, 1060, 910, 820}
09920     ,{ 1990, 1840, 1990, 1840, 1750}
09921     ,{ 1060, 910, 1060, 910, 820}
09922     }
09923     ,{{{ 1210, 1060, 1210, 1060, 970}
09924     ,{ 1210, 1060, 1210, 1060, 970}
09925     ,{ 970, 820, 970, 820, 740}
09926     ,{ 1210, 1060, 1210, 1060, 970}
09927     ,{ 270, 120, 270, 120, 30}
09928     }
09929     }
09930     ,{{{ 2120, 400, 1990, 2120, 1990}
09931     ,{ 2120, 290, 1340, 2120, 1340}
09932     ,{ 1990, 400, 1210, 1990, 1210}
09933     ,{ 1990, 160, 1990, 1990, 1990}
09934     ,{ 1860, 270, 1080, 1860, 1080}
09935     }
09936     ,{{{ 2120, 290, 1340, 2120, 1340}
09937     ,{ 2120, 290, 1340, 2120, 1340}
09938     ,{ 1840, 10, 1060, 1840, 1060}
09939     ,{ 160, -890, 160, -310, 160}
09940     ,{ 1840, 10, 1060, 1840, 1060}
09941     }
09942     ,{{{ 1990, 400, 1210, 1990, 1210}
09943     ,{ 1990, 160, 1210, 1990, 1210}
09944     ,{ 1990, 400, 1210, 1990, 1210}
09945     ,{ 1990, 160, 1210, 1990, 1210}
09946     ,{ 1860, 270, 1080, 1860, 1080}
09947     }
09948     ,{{{ 1990, 10, 1990, 1840, 1990}
09949     ,{ 60, -980, 60, -400, 60}
09950     ,{ 1840, 10, 1060, 1840, 1060}
09951     ,{ 1990, -310, 1990, 270, 1990}
09952     ,{ 1840, 10, 1060, 1840, 1060}
09953     }
09954     ,{{{ 1990, 170, 1210, 1990, 1210}
09955     ,{ 1990, 160, 1210, 1990, 1210}
09956     ,{ 1750, 170, 970, 1750, 970}
09957     ,{ 1990, 160, 1210, 1990, 1210}
09958     ,{ 270, -780, 270, -200, 270}
09959     }
09960     }
09961     ,{{{ 1860, 1840, 1860, 1840, 640}
09962     ,{ 1210, 1190, 1210, 1190, 640}
09963     ,{ 1080, 1060, 1080, 1060, 270}
09964     ,{ 1860, 1840, 1860, 1840, 270}
09965     ,{ 950, 930, 950, 930, 140}
09966     }
09967     ,{{{ 1210, 1190, 1210, 1190, 640}
09968     ,{ 1210, 1190, 1210, 1190, 640}
09969     ,{ 930, 910, 930, 910, 120}
09970     ,{ 270, 10, 270, 10, -780}
09971     ,{ 930, 910, 930, 910, 120}
09972     }
09973     ,{{{ 1080, 1060, 1080, 1060, 270}
09974     ,{ 1080, 1060, 1080, 1060, 270}
09975     ,{ 1080, 1060, 1080, 1060, 270}
09976     ,{ 1080, 1060, 1080, 1060, 270}
09977     ,{ 950, 930, 950, 930, 140}
09978     }
09979     ,{{{ 1860, 1840, 1860, 1840, 120}
09980     ,{ 180, -80, 180, -80, -810}
09981     ,{ 930, 910, 930, 910, 120}
09982     ,{ 1860, 1840, 1860, 1840, -200}
09983     ,{ 930, 910, 930, 910, 120}

```

```

09984     }
09985     , { { 1080, 1060, 1080, 1060, 270 }
09986     , { 1080, 1060, 1080, 1060, 270 }
09987     , { 840, 820, 840, 820, 30 }
09988     , { 1080, 1060, 1080, 1060, 270 }
09989     , { 140, 120, 140, 120, -670 }
09990     }
09991   }
09992 }
09993 };
```

18.176 ViennaRNA/params/io.h File Reference

Read and write energy parameter files.

This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_PARAMETER_FORMAT_DEFAULT 0`
Default Energy Parameter File format.

Functions

- `int vrna_params_load` (const char fname[], unsigned int options)
Load energy parameters from a file.
- `int vrna_params_save` (const char fname[], unsigned int options)
Save energy parameters to a file.
- `int vrna_params_load_from_string` (const char *string, const char *name, unsigned int options)
Load energy parameters from string.
- `int vrna_params_load_defaults` (void)
Load default RNA energy parameter set.
- `int vrna_params_load_RNA_Turner2004` (void)
Load Turner 2004 RNA energy parameter set.
- `int vrna_params_load_RNA_Turner1999` (void)
Load Turner 1999 RNA energy parameter set.
- `int vrna_params_load_RNA_Andronescu2007` (void)
Load Andronescu 2007 RNA energy parameter set.
- `int vrna_params_load_RNA_Langdon2018` (void)
Load Langdon 2018 RNA energy parameter set.
- `int vrna_params_load_RNA_misc_special_hairpins` (void)
Load Misc Special Hairpin RNA energy parameter set.
- `int vrna_params_load_DNA_Mathews2004` (void)
Load Mathews 2004 DNA energy parameter set.
- `int vrna_params_load_DNA_Mathews1999` (void)
Load Mathews 1999 DNA energy parameter set.
- `const char * last_parameter_file` (void)
Get the file name of the parameter file that was most recently loaded.
- `void read_parameter_file` (const char fname[])
Read energy parameters from a file.
- `void write_parameter_file` (const char fname[])
Write energy parameters to a file.

18.176.1 Detailed Description

Read and write energy parameter files.

18.177 io.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_IO_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_IO_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00035 #define VRNA_PARAMETER_FORMAT_DEFAULT 0
00036
00037
00051 int
00052 vrna_params_load(const char  fname[],
00053                 unsigned int options);
00054
00055
00065 int
00066 vrna_params_save(const char  fname[],
00067                 unsigned int options);
00068
00069
00089 int
00090 vrna_params_load_from_string(const char  *string,
00091                             const char  *name,
00092                             unsigned int options);
00093
00094
00110 int
00111 vrna_params_load_defaults(void);
00112
00113
00126 int
00127 vrna_params_load_RNA_Turner2004(void);
00128
00129
00142 int
00143 vrna_params_load_RNA_Turner1999(void);
00144
00145
00158 int
00159 vrna_params_load_RNA_Andronesco2007(void);
00160
00161
00174 int
00175 vrna_params_load_RNA_Langdon2018(void);
00176
00177
00190 int
00191 vrna_params_load_RNA_misc_special_hairpins(void);
00192
00193
00206 int
00207 vrna_params_load_DNA_Mathews2004(void);
00208
00209
00222 int
00223 vrna_params_load_DNA_Mathews1999(void);
00224
00225
00226 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00227
00232 enum parset {
00233     UNKNOWN= -1, QUIT,
00234     S, S_H, HP, HP_H, B, B_H, IL, IL_H, MMH, MMH_H, MMI, MMI_H,
00235     MMI1N, MMI1N_H, MMI23, MMI23_H, MMM, MMM_H, MME, MME_H, D5, D5_H, D3, D3_H,
00236     INT11, INT11_H, INT21, INT21_H, INT22, INT22_H, ML, TL,
00237     TRI, HEX, NIN, MISC
00238 };
00239
00240
00246 const char *
00247 last_parameter_file(void);
00248
00249
00256 DEPRECATED(void
00257             read_parameter_file(const char fname[]),

```



```

00258         "Use vrna_params_load() instead!");
00259
00260
00267 DEPRECATED(void
00268     write_parameter_file(const char fname[]),
00269     "Use vrna_params_save() instead!");
00270
00271
00276 enum parset
00277 gettype(const char *ident);
00278
00279
00284 char *
00285 settype(enum parset s);
00286
00287
00292 #endif
00293
00294 #endif

```

18.178 ViennaRNA/params/salt.h File Reference

Functions to compute salt correction.

Include dependency graph for salt.h: This graph shows which files directly or indirectly include this file:

Functions

- double [vrna_salt_loop](#) (int L, double [salt](#), double T)
Get salt correction for a loop at a given salt concentration and temperature.
- int [vrna_salt_loop_int](#) (int L, double [salt](#), double T)
Get salt correction for a loop at a given salt concentration and temperature.
- int [vrna_salt_stack](#) (double [salt](#), double T)
Get salt correction for a stack at a given salt concentration and temperature.

18.178.1 Detailed Description

Functions to compute salt correction.

18.179 salt.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_LOOPS_SALT_H
00002 #define VIENNA_RNA_PACKAGE_LOOPS_SALT_H
00003
00020 #include <math.h>
00021 #include "ViennaRNA/utils/basic.h"
00022
00023 #ifdef __GNUC__
00024 # define INLINE inline
00025 #else
00026 # define INLINE
00027 #endif
00028
00038 double
00039 vrna_salt_loop(int L, double salt, double T);
00040
00041
00055 int
00056 vrna_salt_loop_int(int L, double salt, double T);
00057
00058
00067 int
00068 vrna_salt_stack(double salt, double T);
00069
00070
00085 void
00086 vrna_salt_ml(double saltLoop[], int lower, int upper, int *m, int *b);
00087
00088
00095 int
00096 vrna_salt_duplex_init(double salt);
00097
00102 #endif

```

18.180 ViennaRNA/part_func.h File Reference

Partition function implementations.

Include dependency graph for part_func.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_dimer_pf_s](#)
Data structure returned by [vrna_pf_dimer\(\)](#) [More...](#)
- struct [vrna_multimer_pf_s](#)

Typedefs

- typedef struct [vrna_dimer_pf_s](#) [vrna_dimer_pf_t](#)
Typename for the data structure that stores the dimer partition functions, [vrna_dimer_pf_s](#), as returned by [vrna_pf_dimer\(\)](#)
- typedef struct [vrna_dimer_pf_s](#) [cofoldF](#)
Backward compatibility typedef for [vrna_dimer_pf_s](#).

Functions

- int [vrna_pf_float_precision](#) (void)
Find out whether partition function computations are using single precision floating points.
- float [pf_fold_par](#) (const char *sequence, char *structure, [vrna_exp_param_t](#) *parameters, int calculate_bp, int is_constrained, int is_circular)
Compute the partition function Q for a given RNA sequence.
- float [pf_fold](#) (const char *sequence, char *structure)
Compute the partition function Q of an RNA sequence.
- float [pf_circ_fold](#) (const char *sequence, char *structure)
Compute the partition function of a circular RNA sequence.
- char * [pbacktrack](#) (char *sequence)
Sample a secondary structure from the Boltzmann ensemble according its probability.
- char * [pbacktrack5](#) (char *sequence, int length)
Sample a sub-structure from the Boltzmann ensemble according its probability.
- char * [pbacktrack_circ](#) (char *sequence)
Sample a secondary structure of a circular RNA from the Boltzmann ensemble according its probability.
- void [free_pf_arrays](#) (void)
Free arrays for the partition function recursions.
- void [update_pf_params](#) (int length)
Recalculate energy parameters.
- void [update_pf_params_par](#) (int length, [vrna_exp_param_t](#) *parameters)
Recalculate energy parameters.
- [FLT_OR_DBL](#) * [export_bppm](#) (void)
Get a pointer to the base pair probability array.
- int [get_pf_arrays](#) (short **S_p, short **S1_p, char **ptype_p, [FLT_OR_DBL](#) **qb_p, [FLT_OR_DBL](#) **qm_p, [FLT_OR_DBL](#) **q1k_p, [FLT_OR_DBL](#) **qln_p)
Get the pointers to (almost) all relevant computation arrays used in partition function computation.
- double [get_subseq_F](#) (int i, int j)
Get the free energy of a subsequence from the $q[]$ array.
- double [mean_bp_distance](#) (int length)
Get the mean base pair distance of the last partition function computation.
- double [mean_bp_distance_pr](#) (int length, [FLT_OR_DBL](#) *pr)
Get the mean base pair distance in the thermodynamic ensemble.

- [vrna_ep_t](#) * [stackProb](#) (double cutoff)
Get the probability of stacks.
- void [init_pf_fold](#) (int length)
Allocate space for [pf_fold\(\)](#)
- char * [centroid](#) (int length, double *dist)
- char * [get_centroid_struct_gquad_pr](#) (int length, double *dist)
- double [mean_bp_dist](#) (int length)
- double [expLoopEnergy](#) (int u1, int u2, int type, int type2, short si1, short sj1, short sp1, short sq1)
- double [expHairpinEnergy](#) (int u, int type, short si1, short sj1, const char *string)

Basic global partition function interface

- [FLT_OR_DBL](#) [vrna_pf](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Compute the partition function Q for a given RNA sequence, or sequence alignment.
- [vrna_dimer_pf_t](#) [vrna_pf_dimer](#) ([vrna_fold_compound_t](#) *vc, char *structure)
Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.
- [FLT_OR_DBL](#) * [vrna_pf_substrands](#) ([vrna_fold_compound_t](#) *fc, [size_t](#) complex_size)
- [FLT_OR_DBL](#) [vrna_pf_add](#) ([FLT_OR_DBL](#) dG1, [FLT_OR_DBL](#) dG2, double kT)

Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- float [vrna_pf_fold](#) (const char *sequence, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for an RNA sequence using a comparative method.
- float [vrna_pf_circfold](#) (const char *sequence, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for a circular RNA sequences using a comparative method.
- float [vrna_pf_alifold](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for an RNA sequence alignment using a comparative method.
- float [vrna_pf_circalifold](#) (const char **sequences, char *structure, [vrna_ep_t](#) **pl)
Compute Partition function Q (and base pair probabilities) for an alignment of circular RNA sequences using a comparative method.
- [vrna_dimer_pf_t](#) [vrna_pf_co_fold](#) (const char *seq, char *structure, [vrna_ep_t](#) **pl)
Calculate partition function and base pair probabilities of nucleic acid/nucleic acid dimers.

Variables

- int [st_back](#)
Flag indicating that auxiliary arrays are needed throughout the computations. This is essential for stochastic backtracking.

18.180.1 Detailed Description

Partition function implementations.

,
This file includes (almost) all function declarations within the **RNAlib** that are related to Partion function computations

18.180.2 Function Documentation

18.180.2.1 centroid()

```
char * centroid (
    int length,
    double * dist )
```

Deprecated This function is deprecated and should not be used anymore as it is not threadsafe!

See also

[get_centroid_struct_pl\(\)](#), [get_centroid_struct_pr\(\)](#)

18.180.2.2 `get_centroid_struct_gquad_pr()`

```
char * get_centroid_struct_gquad_pr (
    int length,
    double * dist )
```

Deprecated This function is deprecated and should not be used anymore as it is not threadsafe!

See also

[vrna_centroid\(\)](#), [vrna_centroid_from_probs\(\)](#), [vrna_centroid_from_plist\(\)](#)

18.180.2.3 `mean_bp_dist()`

```
double mean_bp_dist (
    int length )
```

get the mean pair distance of ensemble

Deprecated This function is not threadsafe and should not be used anymore. Use [mean_bp_distance\(\)](#) instead!

18.180.2.4 `expLoopEnergy()`

```
double expLoopEnergy (
    int u1,
    int u2,
    int type,
    int type2,
    short sil,
    short sj1,
    short sp1,
    short sq1 )
```

Deprecated Use [exp_E_IntLoop\(\)](#) from [loop_energies.h](#) instead

18.180.2.5 `expHairpinEnergy()`

```
double expHairpinEnergy (
    int u,
    int type,
    short sil,
    short sj1,
    const char * string )
```

Deprecated Use [exp_E_Hairpin\(\)](#) from [loop_energies.h](#) instead

18.181 part_func.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_H
00002 #define VIENNA_RNA_PACKAGE_PART_FUNC_H
00003
00008 typedef struct vrna_dimer_pf_s vrna_dimer_pf_t;
00009
00010 typedef struct vrna_multimer_pf_s vrna_multimer_pf_t;
00011
00012 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00013
00018 typedef struct vrna_dimer_pf_s cofoldF;
00019
00020 #endif
00021
00022
00023 #include <ViennaRNA/datastructures/basic.h>
00024 #include <ViennaRNA/fold_compound.h>
00025 #include <ViennaRNA/utils/structures.h>
00026 #include <ViennaRNA/params/basic.h>
00027 #include <ViennaRNA/centroid.h>
00028 #include <ViennaRNA/equilibrium_probs.h>
00029 #include <ViennaRNA/boltzmann_sampling.h>
00030
00031 #ifdef VRNA_WARN_DEPRECATED
00032 # if defined(__clang__)
00033 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00034 # elif defined(__GNUC__)
00035 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00036 # else
00037 #  define DEPRECATED(func, msg) func
00038 # endif
00039 #else
00040 # define DEPRECATED(func, msg) func
00041 #endif
00042
00052 /*
00053 #####
00054 # PARTITION FUNCTION COMPUTATION
00055 #####
00056 */
00057
00098 struct vrna_dimer_pf_s {
00099     /* free energies for: */
00100     double FOAB;
00101     double FAB;
00102     double FcAB;
00103     double FA;
00104     double FB;
00105 };
00106
00107 struct vrna_multimer_pf_s {
00108     /* free energies for: */
00109     double F_connected;
00110     double *F_monomers;
00111     size_t num_monomers;
00112 };
00113
00147 FLT_OR_DBL
00148 vrna_pf(vrna_fold_compound_t *vc,
00149         char *structure);
00150
00151
00170 vrna_dimer_pf_t
00171 vrna_pf_dimer(vrna_fold_compound_t *vc,
00172              char *structure);
00173
00174
00175 FLT_OR_DBL *
00176 vrna_pf_substrands(vrna_fold_compound_t *fc,
00177                   size_t complex_size);
00178
00179 FLT_OR_DBL
00180 vrna_pf_add(FLT_OR_DBL dG1,
00181            FLT_OR_DBL dG2,
00182            double kT);
00183
00184 /* End basic global interface */
00213 float
00214 vrna_pf_fold(const char *sequence,
00215             char *structure,
00216             vrna_ep_t **pl);
00217
00218
00243 float

```

```

00244 vrna_pf_circfold(const char *sequence,
00245                  char *structure,
00246                  vrna_ep_t **pl);
00247
00248
00270 float
00271 vrna_pf_alifold(const char **sequences,
00272                 char *structure,
00273                 vrna_ep_t **pl);
00274
00275
00300 float
00301 vrna_pf_circalifold(const char **sequences,
00302                     char *structure,
00303                     vrna_ep_t **pl);
00304
00305
00332 vrna_dimer_pf_t
00333 vrna_pf_co_fold(const char *seq,
00334                 char *structure,
00335                 vrna_ep_t **pl);
00336
00337
00338 /* End simplified global interface */
00343 /*
00344 #####
00345 # OTHER PARTITION FUNCTION RELATED DECLARATIONS #
00346 #####
00347 */
00348
00358 int
00359 vrna_pf_float_precision(void);
00360
00361
00362 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00363
00364 /*
00365 #####
00366 # DEPRECATED FUNCTIONS #
00367 #####
00368 */
00369
00382 extern int st_back;
00383
00424 DEPRECATED(float
00425             pf_fold_par(const char *sequence,
00426                         char *structure,
00427                         vrna_exp_param_t *parameters,
00428                         int calculate_bppm,
00429                         int is_constrained,
00430                         int is_circular),
00431             "Use the new API and vrna_pf() instead");
00432
00472 DEPRECATED(float
00473             pf_fold(const char *sequence,
00474                     char *structure),
00475             "Use vrna_pf_fold() or vrna_pf() instead");
00476
00503 DEPRECATED(float
00504             pf_circ_fold(const char *sequence,
00505                          char *structure),
00506             "Use vrna_pf_circfold() or vrna_pf() instead");
00507
00519 DEPRECATED(char *pbacktrack(char *sequence), "Use vrna_pbacktrack() instead");
00520
00526 DEPRECATED(char *pbacktrack5(char *sequence,
00527                               int length), "Use vrna_pbacktrack5() instead");
00528
00544 DEPRECATED(char *pbacktrack_circ(char *sequence), "Use vrna_pbacktrack() instead");
00545
00564 DEPRECATED(void
00565             free_pf_arrays(void), "This function is obsolete");
00566
00578 DEPRECATED(void
00579             update_pf_params(int length), "This function is obsolete");
00580
00589 DEPRECATED(void
00590             update_pf_params_par(int length,
00591                                 vrna_exp_param_t *parameters),
00592             "Use the new API with vrna_fold_compound_t instead");
00593
00611 DEPRECATED(FLT_OR_DBL * export_bppm(void),
00612             "Use the new API with vrna_fold_compound_t instead");
00613
00614
00631 DEPRECATED(int
00632             get_pf_arrays(short **S_p,

```

```

00633             short      **sl_p,
00634             char       **ptype_p,
00635             FLT_OR_DBL **qb_p,
00636             FLT_OR_DBL **qm_p,
00637             FLT_OR_DBL **qlk_p,
00638             FLT_OR_DBL **qln_p),
00639     "Use the new API with vrna_fold_compound_t instead");
00640
00646 DEPRECATED(double
00647     get_subseq_F(int i,
00648                 int j),
00649     "Use the new API with vrna_fold_compound_t instead");
00650
00651
00663 DEPRECATED(double
00664     mean_bp_distance(int length),
00665     "Use vrna_mean_bp_distance() or vrna_mean_bp_distance_pr() instead");
00666
00684 DEPRECATED(double
00685     mean_bp_distance_pr(int length,
00686                        FLT_OR_DBL *pr),
00687     "Use vrna_mean_bp_distance() or vrna_mean_bp_distance_pr() instead");
00688
00696 DEPRECATED(vrna_ep_t * stackProb(double cutoff), "Use vrna_stack_prob() instead");
00697
00698
00706 DEPRECATED(void
00707     init_pf_fold(int length), "This function is obsolete");
00708
00713 DEPRECATED(char *centroid(int length,
00714                          double *dist),
00715     "Use vrna_centroid() instead");
00716
00721 DEPRECATED(char *get_centroid_struct_gquad_pr(int length,
00722                                               double *dist),
00723     "Use vrna_centroid() instead");
00724
00730 DEPRECATED(double
00731     mean_bp_dist(int length),
00732     "Use vrna_mean_bp_distance() or vrna_mean_bp_distance_pr() instead");
00733
00737 DEPRECATED(double
00738     expLoopEnergy(int u1,
00739                  int u2,
00740                  int type,
00741                  int type2,
00742                  short sil,
00743                  short sj1,
00744                  short spl,
00745                  short sql),
00746     "");
00747
00751 DEPRECATED(double
00752     expHairpinEnergy(int u,
00753                    int type,
00754                    short sil,
00755                    short sj1,
00756                    const char *string),
00757     "");
00758
00759 /* this doesn't work if free_pf_arrays() is called before */
00760 DEPRECATED(void
00761     assign_plist_gquad_from_pr(vrna_ep_t **pl,
00762                              int length,
00763                              double cut_off),
00764     "Use vrna_plist_from_probs() instead");
00765
00766 #endif
00767
00768 #endif

```

18.182 ViennaRNA/part_func_co.h File Reference

Partition function for two RNA sequences.

Include dependency graph for part_func_co.h:

Functions

- [vrna_dimer_pf_t co_pf_fold](#) (char *sequence, char *structure)

Calculate partition function and base pair probabilities.

- [vrna_dimer_pf_t co_pf_fold_par](#) (char *sequence, char *structure, [vrna_exp_param_t](#) *parameters, int calculate_bppm, int is_constrained)
Calculate partition function and base pair probabilities.
- [vrna_ep_t * get_plist](#) ([vrna_ep_t](#) *pl, int length, double cut_off)
- void [compute_probabilities](#) (double FAB, double FEA, double FEB, [vrna_ep_t](#) *prAB, [vrna_ep_t](#) *prA, [vrna_ep_t](#) *prB, int Alength)
Compute Boltzmann probabilities of dimerization without homodimers.
- void [init_co_pf_fold](#) (int length)
- [FLT_OR_DBL](#) * [export_co_bppm](#) (void)
Get a pointer to the base pair probability array.
- void [free_co_pf_arrays](#) (void)
Free the memory occupied by [co_pf_fold\(\)](#)
- void [update_co_pf_params](#) (int length)
Recalculate energy parameters.
- void [update_co_pf_params_par](#) (int length, [vrna_exp_param_t](#) *parameters)
Recalculate energy parameters.

Variables

- int **mirnatog**
Toggles no intrabp in 2nd mol.
- double **F_monomer** [2]
Free energies of the two monomers.

18.182.1 Detailed Description

Partition function for two RNA sequences.

18.182.2 Function Documentation

18.182.2.1 [get_plist\(\)](#)

```
vrna_ep_t * get_plist (
    vrna_ep_t * pl,
    int length,
    double cut_off )
```

DO NOT USE THIS FUNCTION ANYMORE

Deprecated { This function is deprecated and will be removed soon!} use [assign_plist_from_pr\(\)](#) instead!

18.183 [part_func_co.h](#)

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_CO_H
00002 #define VIENNA_RNA_PACKAGE_PART_FUNC_CO_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00052 #include <ViennaRNA/datastructures/basic.h>
```



```

00053 #include <ViennaRNA/params/basic.h>
00054 #include <ViennaRNA/part_func.h>
00055 #include <ViennaRNA/equilibrium_probs.h>
00056 #include <ViennaRNA/concentrations.h>
00057 #include <ViennaRNA/utils/structures.h>
00058
00062 extern int    mirnatog;
00063
00067 extern double F_monomer[2];
00068
00073 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00074
00075 /*
00076 #####
00077 # DEPRECATED FUNCTIONS
00078 #####
00079 */
00080
00101 DEPRECATED(vrna_dimer_pf_t co_pf_fold(char *sequence,
00102                                     char *structure),
00103 "Use vrna_pf_co_fold() or vrna_pf_dimer() instead");
00104
00126 DEPRECATED(vrna_dimer_pf_t co_pf_fold_par(char *sequence,
00127                                     char *structure,
00128                                     vrna_exp_param_t *parameters,
00129                                     int calculate_bppm,
00130                                     int is_constrained),
00131 "Use the new API and vrna_pf_dimer() instead");
00132
00138 DEPRECATED(vrna_ep_t *get_plist(vrna_ep_t *pl,
00139                                int length,
00140                                double cut_off),
00141 "Use vrna_plist() and vrna_plist_from_probs() instead");
00142
00164 DEPRECATED(void compute_probabilities(double FAB,
00165                                     double FEA,
00166                                     double FEB,
00167                                     vrna_ep_t *prAB,
00168                                     vrna_ep_t *prA,
00169                                     vrna_ep_t *prB,
00170                                     int Alength),
00171 "Use vrna_pf_dimer_probs() instead");
00172
00178 DEPRECATED(void init_co_pf_fold(int length),
00179 "This function is obsolete");
00180
00196 DEPRECATED(FLT_OR_DBL *export_co_bppm(void),
00197 "Use the new API with vrna_fold_compound_t instead");
00198
00207 DEPRECATED(void free_co_pf_arrays(void),
00208 "This function is obsolete");
00209
00222 DEPRECATED(void update_co_pf_params(int length),
00223 "This function is obsolete");
00224
00246 DEPRECATED(void update_co_pf_params_par(int length,
00247                                     vrna_exp_param_t *parameters),
00248 "Use the new API with vrna_fold_compound_t instead");
00249
00250 #endif
00251
00252 #endif

```

18.184 ViennaRNA/part_func_up.h File Reference

Implementations for accessibility and RNA-RNA interaction as a stepwise process.

Include dependency graph for part_func_up.h:

Functions

- [pu_contrib](#) * [pf_unstru](#) (char *sequence, int max_w)
Calculate the partition function over all unpaired regions of a maximal length.
- [interact](#) * [pf_interact](#) (const char *s1, const char *s2, [pu_contrib](#) *p_c, [pu_contrib](#) *p_c2, int max_w, char *cstruc, int incr3, int incr5)
Calculates the probability of a local interaction between two sequences.
- void [free_interact](#) ([interact](#) *pin)
Frees the output of function [pf_interact\(\)](#).

- `void free_pu_contrib_struct (pu_contrib *pu)`
Frees the output of function `pf_unstru()`.

18.184.1 Detailed Description

Implementations for accessibility and RNA-RNA interaction as a stepwise process.

18.185 part_func_up.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_UP_H
00002 #define VIENNA_RNA_PACKAGE_PART_FUNC_UP_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005
00006 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00007
00008 #define RNA_UP_MODE_1 1U
00009 #define RNA_UP_MODE_2 2U
00010 #define RNA_UP_MODE_3 4U
00011
00061 pu_contrib *pf_unstru(char *sequence,
00062                      int max_w);
00063
00106 interact *pf_interact(const char *s1,
00107                      const char *s2,
00108                      pu_contrib *p_c,
00109                      pu_contrib *p_c2,
00110                      int max_w,
00111                      char *cstruc,
00112                      int incr3,
00113                      int incr5);
00114
00118 void free_interact(interact *pin);
00119
00123 int Up_plot(pu_contrib *p_c,
00124            pu_contrib *p_c_sh,
00125            interact *pint,
00126            char *ofile,
00127            int **unpaired_values,
00128            char *select_contrib,
00129            char *head,
00130            unsigned int mode);
00131
00135 pu_contrib *get_pu_contrib_struct( unsigned int n,
00136                                   unsigned int w);
00137
00141 void free_pu_contrib_struct(pu_contrib *pu);
00142
00143 void
00144 free_pu_contrib(pu_contrib *pu);
00145
00150 #endif
00151
00152 #endif
```

18.186 ViennaRNA/part_func_window.h File Reference

Partition function and equilibrium probability implementation for the sliding window algorithm.

Include dependency graph for `part_func_window.h`: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_EXT_LOOP 1U`
Exterior loop.
- `#define VRNA_HP_LOOP 2U`
Hairpin loop.
- `#define VRNA_INT_LOOP 4U`
Internal loop.
- `#define VRNA_MB_LOOP 8U`

- Multibranch loop.*
- `#define VRNA_ANY_LOOP (VRNA_EXT_LOOP | VRNA_HP_LOOP | VRNA_INT_LOOP | VRNA_MB_LOOP)`
- Any loop.*
- `#define VRNA_PROBS_WINDOW_BPP 4096U`
- Trigger base pairing probabilities.*
- `#define VRNA_PROBS_WINDOW_UP 8192U`
- Trigger unpaired probabilities.*
- `#define VRNA_PROBS_WINDOW_STACKP 16384U`
- Trigger base pair stack probabilities.*
- `#define VRNA_PROBS_WINDOW_UP_SPLIT 32768U`
- Trigger detailed unpaired probabilities split up into different loop type contexts.*
- `#define VRNA_PROBS_WINDOW_PF 65536U`
- Trigger partition function.*

Typedefs

- `typedef void(* vrna_probs_window_f) (FLT_OR_DBL *pr, int pr_size, int i, int max, unsigned int type, void *data)`
- Sliding window probability computation callback.*

Functions

Basic local partition function interface

- `int vrna_probs_window (vrna_fold_compound_t *fc, int ulength, unsigned int options, vrna_probs_window_f cb, void *data)`
- Compute various equilibrium probabilities under a sliding window approach.*

Simplified global partition function computation using sequence(s) or multiple sequence alignment(s)

- `vrna_ep_t * vrna_pfl_fold (const char *sequence, int window_size, int max_bp_span, float cutoff)`
- Compute base pair probabilities using a sliding-window approach.*
- `int vrna_pfl_fold_cb (const char *sequence, int window_size, int max_bp_span, vrna_probs_window_f cb, void *data)`
- Compute base pair probabilities using a sliding-window approach (callback version)*
- `double ** vrna_pfl_fold_up (const char *sequence, int ulength, int window_size, int max_bp_span)`
- Compute probability of contiguous unpaired segments.*
- `int vrna_pfl_fold_up_cb (const char *sequence, int ulength, int window_size, int max_bp_span, vrna_probs_window_f cb, void *data)`
- Compute probability of contiguous unpaired segments.*

18.186.1 Detailed Description

Partition function and equilibrium probability implementation for the sliding window algorithm.

This file contains the implementation for sliding window partition function and equilibrium probabilities. It also provides the unpaired probability implementation from Bernhart et al. 2011 [4]

18.187 part_func_window.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_WINDOW_H
00002 #define VIENNA_RNA_PACKAGE_PART_FUNC_WINDOW_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)
```

```

00009 # define DEPRECATED(func, msg) func __attribute__ ((deprecated("", msg)))
00010 # elif defined(__GNUC__)
00011 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00012 # else
00013 # define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
00018
00030 #include <ViennaRNA/datastructures/basic.h>
00031
00078 typedef void (*vrna_probs_window_f) (FLT_OR_DBL *pr,
00079                                     int pr_size,
00080                                     int i,
00081                                     int max,
00082                                     unsigned int type,
00083                                     void *data);
00084
00085 DEPRECATED(typedef void (vrna_probs_window_callback) (FLT_OR_DBL *pr,
00086                                                       int pr_size,
00087                                                       int i,
00088                                                       int max,
00089                                                       unsigned int type,
00090                                                       void *data),
00091           "Use vrna_probs_window_f instead!");
00092
00093
00094 #include <ViennaRNA/fold_compound.h>
00095 #include <ViennaRNA/utils/structures.h>
00096
00100 #define VRNA_EXT_LOOP 1U
00101
00105 #define VRNA_HP_LOOP 2U
00106
00110 #define VRNA_INT_LOOP 4U
00111
00115 #define VRNA_MB_LOOP 8U
00116
00120 #define VRNA_ANY_LOOP (VRNA_EXT_LOOP | VRNA_HP_LOOP | VRNA_INT_LOOP | VRNA_MB_LOOP)
00121
00122
00135 #define VRNA_PROBS_WINDOW_BPP 4096U
00136
00149 #define VRNA_PROBS_WINDOW_UP 8192U
00150
00163 #define VRNA_PROBS_WINDOW_STACKP 16384U
00164
00182 #define VRNA_PROBS_WINDOW_UP_SPLIT 32768U
00183
00184
00198 #define VRNA_PROBS_WINDOW_PF 65536U
00199
00231 int
00232 vrna_probs_window(vrna_fold_compound_t *fc,
00233                  int ulength,
00234                  unsigned int options,
00235                  vrna_probs_window_f cb,
00236                  void *data);
00237
00238 /* End basic interface */
00267 vrna_ep_t *
00268 vrna_pfl_fold(const char *sequence,
00269              int window_size,
00270              int max_bp_span,
00271              float cutoff);
00272
00273
00296 int
00297 vrna_pfl_fold_cb(const char *sequence,
00298                 int window_size,
00299                 int max_bp_span,
00300                 vrna_probs_window_f cb,
00301                 void *data);
00302
00303
00326 double **
00327 vrna_pfl_fold_up(const char *sequence,
00328                 int ulength,
00329                 int window_size,
00330                 int max_bp_span);
00331
00332
00356 int
00357 vrna_pfl_fold_up_cb(const char *sequence,
00358                    int ulength,
00359                    int window_size,

```

```

00360             int                max_bp_span,
00361             vrna_probs_window_f cb,
00362             void                *data);
00363
00364
00365 /* End simplified interface */
00371 #endif

```

18.188 ViennaRNA/perturbation_fold.h File Reference

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

Include dependency graph for perturbation_fold.h:

Macros

- `#define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0`
Use the sum of squared aberrations as objective function.
- `#define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1`
Use the sum of absolute aberrations as objective function.
- `#define VRNA_MINIMIZER_DEFAULT 0`
Use a custom implementation of the gradient descent algorithm to minimize the objective function.
- `#define VRNA_MINIMIZER_CONJUGATE_FR 1`
Use the GNU Scientific Library implementation of the Fletcher-Reeves conjugate gradient algorithm to minimize the objective function.
- `#define VRNA_MINIMIZER_CONJUGATE_PR 2`
Use the GNU Scientific Library implementation of the Polak-Ribiere conjugate gradient algorithm to minimize the objective function.
- `#define VRNA_MINIMIZER_VECTOR_BFGS 3`
Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.
- `#define VRNA_MINIMIZER_VECTOR_BFGS2 4`
Use the GNU Scientific Library implementation of the vector Broyden-Fletcher-Goldfarb-Shanno algorithm to minimize the objective function.
- `#define VRNA_MINIMIZER_STEEPEST_DESCENT 5`
Use the GNU Scientific Library implementation of the steepest descent algorithm to minimize the objective function.

Typedefs

- `typedef void(* progress_callback) (int iteration, double score, double *epsilon)`
Callback for following the progress of the minimization process.

Functions

- `void vrna_sc_minimize_pertubation (vrna_fold_compound_t *vc, const double *q_prob_unpaired, int objective_function, double sigma_squared, double tau_squared, int algorithm, int sample_size, double *epsilon, double initialStepSize, double minStepSize, double minImprovement, double minimizerTolerance, progress_callback callback)`
Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

18.188.1 Detailed Description

Find a vector of perturbation energies that minimizes the discrepancies between predicted and observed pairing probabilities and the amount of necessary adjustments.

18.189 perturbation_fold.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PERTURBATION_FOLD_H
00002 #define VIENNA_RNA_PACKAGE_PERTURBATION_FOLD_H
00003
00004 #include <ViennaRNA/fold_compound.h>
00005
00024 #define VRNA_OBJECTIVE_FUNCTION_QUADRATIC 0
00025
00033 #define VRNA_OBJECTIVE_FUNCTION_ABSOLUTE 1
00034
00040 #define VRNA_MINIMIZER_DEFAULT 0
00041
00049 #define VRNA_MINIMIZER_CONJUGATE_FR 1
00050
00058 #define VRNA_MINIMIZER_CONJUGATE_PR 2
00059
00067 #define VRNA_MINIMIZER_VECTOR_BFGS 3
00068
00076 #define VRNA_MINIMIZER_VECTOR_BFGS2 4
00077
00085 #define VRNA_MINIMIZER_STEEPEST_DESCENT 5
00086
00096 typedef void (*progress_callback)(int iteration,
00097                                   double score,
00098                                   double *epsilon);
00099
00139 void vrna_sc_minimize_pertubation(vrna_fold_compound_t *vc,
00140                                  const double *q_prob_unpaired,
00141                                  int objective_function,
00142                                  double sigma_squared,
00143                                  double tau_squared,
00144                                  int algorithm,
00145                                  int sample_size,
00146                                  double *epsilon,
00147                                  double initialStepSize,
00148                                  double minStepSize,
00149                                  double minImprovement,
00150                                  double minimizerTolerance,
00151                                  progress_callback callback);
00152
00153
00154 #endif
```

18.190 pf_multifold.h

```
00001 #ifndef VIENNA_RNA_PACKAGE_PART_FUNC_MULTIFOLD_H
00002 #define VIENNA_RNA_PACKAGE_PART_FUNC_MULTIFOLD_H
00003
00004 #include "ViennaRNA/fold_compound.h"
00005
00006 int
00007 vrna_pf_multifold_prepare(vrna_fold_compound_t *fc);
00008
00009
00010 #endif
```

18.191 ViennaRNA/pk_plex.h File Reference

Heuristics for two-step pseudoknot forming interaction predictions.

Include dependency graph for pk_plex.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_pk_plex_result_s](#)
A result of the RNA PKplex interaction prediction. [More...](#)

Typedefs

- typedef int(* [vrna_pk_plex_score_f](#)) (const short *pt, int start_5, int end_5, int start_3, int end_3, void *data)
Pseudoknot loop scoring function prototype.
- typedef struct vrna_pk_plex_option_s * [vrna_pk_plex_opt_t](#)

RNA PKplex options object.

- typedef struct [vrna_pk_plex_result_s](#) [vrna_pk_plex_t](#)

Convenience typedef for results of the RNA PKplex prediction.

Functions

- [vrna_pk_plex_t](#) * [vrna_pk_plex](#) ([vrna_fold_compound_t](#) *fc, const int **accessibility, [vrna_pk_plex_opt_t](#) options)

Predict Pseudoknot interactions in terms of a two-step folding process.

- int ** [vrna_pk_plex_accessibility](#) (const char *sequence, unsigned int [unpaired](#), double cutoff)

Obtain a list of opening energies suitable for PKplex computations.

- [vrna_pk_plex_opt_t](#) [vrna_pk_plex_opt_defaults](#) (void)

Default options for PKplex algorithm.

- [vrna_pk_plex_opt_t](#) [vrna_pk_plex_opt](#) (unsigned int delta, unsigned int max_interaction_length, int pk_penalty)

Simple options for PKplex algorithm.

- [vrna_pk_plex_opt_t](#) [vrna_pk_plex_opt_fun](#) (unsigned int delta, unsigned int max_interaction_length, [vrna_pk_plex_score_f](#) scoring_function, void *scoring_data)

Simple options for PKplex algorithm.

18.191.1 Detailed Description

Heuristics for two-step pseudoknot forming interaction predictions.

18.192 pk_plex.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PK_PLEX_H
00002 #define VIENNA_RNA_PACKAGE_PK_PLEX_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)
00009 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00010 # elif defined(__GNUC__)
00011 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00012 # else
00013 #   define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
00018
00060 typedef int (*vrna_pk_plex_score_f)(const short *pt,
00061                                     int start_5,
00062                                     int end_5,
00063                                     int start_3,
00064                                     int end_3,
00065                                     void *data);
00066
00067 DEPRECATED(typedef int (vrna_callback_pk_plex_score)(const short *pt,
00068                                                       int start_5,
00069                                                       int end_5,
00070                                                       int start_3,
00071                                                       int end_3,
00072                                                       void *data),
00073            "Use vrna_pk_plex_score_f instead!");
00074
00075
00082 typedef struct vrna_pk_plex_option_s *vrna_pk_plex_opt_t;
00083
00089 typedef struct vrna_pk_plex_result_s vrna_pk_plex_t;
00090
00091 #include <ViennaRNA/fold_compound.h>
00092
00098 struct vrna_pk_plex_result_s {
00099     char *structure;
00100     double energy;
```

```

00101     double          dGpk;
00102     double          dGint;
00103     double          dG1;
00104     double          dG2;
00105     unsigned int     start_5;
00106     unsigned int     end_5;
00107     unsigned int     start_3;
00108     unsigned int     end_3;
00109 };
00110
00140 vrna_pk_plex_t *
00141 vrna_pk_plex(vrna_fold_compound_t *fc,
00142             const int **accessibility,
00143             vrna_pk_plex_opt_t options);
00144
00145
00156 int **
00157 vrna_pk_plex_accessibility(const char *sequence,
00158                          unsigned int unpaired,
00159                          double cutoff);
00160
00161
00169 vrna_pk_plex_opt_t
00170 vrna_pk_plex_opt_defaults(void);
00171
00172
00183 vrna_pk_plex_opt_t
00184 vrna_pk_plex_opt(unsigned int delta,
00185                 unsigned int max_interaction_length,
00186                 int pk_penalty);
00187
00188
00200 vrna_pk_plex_opt_t
00201 vrna_pk_plex_opt_fun(unsigned int delta,
00202                    unsigned int max_interaction_length,
00203                    vrna_pk_plex_score_f scoring_function,
00204                    void *scoring_data);
00205
00206
00211 #endif

```

18.193 PKplex.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_PKPLEX_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PKPLEX_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/PKplex.h>! Use <ViennaRNA/pk_plex.h> instead!"
00013 # endif
00014
00015 #ifdef VRNA_WARN_DEPRECATED
00016 # if defined(__clang__)
00017 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00018 # elif defined(__GNUC__)
00019 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00020 # else
00021 #   define DEPRECATED(func, msg) func
00022 # endif
00023 #else
00024 # define DEPRECATED(func, msg) func
00025 #endif
00026
00027 #include <ViennaRNA/datastructures/basic.h>
00028
00029
00030 DEPRECATED(dupVar *
00031 PKLduplexfold_XS(const char *s1,
00032                 const int **access_s1,
00033                 int penalty,
00034                 int max_interaction_length,
00035                 int delta),
00036 "Use vrna_pk_plex() instead!");
00037
00038 #include <ViennaRNA/pk_plex.h>
00039
00040 #endif
00041
00042 #endif

```


18.194 plex.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLEX_H
00002 #define VIENNA_RNA_PACKAGE_PLEX_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005
00006 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00007
00008 extern int subopt_sorted;
00009
00013 duplexT** Lduplexfold(const char *s1,
00014                      const char *s2,
00015                      const int threshold,
00016                      const int extension_cost,
00017                      const int alignment_length,
00018                      const int delta,
00019                      const int fast,
00020                      const int il_a,
00021                      const int il_b,
00022                      const int b_a,
00023                      const int b_b);
00024
00028 duplexT** Lduplexfold_XS( const char*s1,
00029                          const char* s2,
00030                          const int **access_s1,
00031                          const int **access_s2,
00032                          const int threshold,
00033                          const int delta,
00034                          const int alignment_length,
00035                          const int fast,
00036                          const int il_a,
00037                          const int il_b,
00038                          const int b_a,
00039                          const int b_b); /* , const int target_dead, const int query_dead); */
00040
00044 duplexT** Lduplexfold_C(const char *s1,
00045                        const char *s2,
00046                        const int threshold,
00047                        const int extension_cost,
00048                        const int alignment_length,
00049                        const int delta,
00050                        const int fast,
00051                        const char* structure,
00052                        const int il_a,
00053                        const int il_b,
00054                        const int b_a,
00055                        const int b_b);
00056
00061 duplexT** Lduplexfold_CXS(const char*s1,
00062                          const char* s2,
00063                          const int **access_s1,
00064                          const int **access_s2,
00065                          const int threshold,
00066                          const int delta,
00067                          const int alignment_length,
00068                          const int fast,
00069                          const char* structure,
00070                          const int il_a,
00071                          const int il_b,
00072                          const int b_a,
00073                          const int b_b); /* , const int target_dead, const int query_dead); */
00074
00075
00076
00077
00078 int      arraySize(duplexT** array);
00079 void     freeDuplexT(duplexT** array);
00080
00081 #endif
00082
00083 #endif

```

18.195 ViennaRNA/plot_aln.h File Reference

Use [ViennaRNA/plotting/alignments.h](#) instead.

Include dependency graph for plot_aln.h:

18.195.1 Detailed Description

Use [ViennaRNA/plotting/alignments.h](#) instead.

Deprecated Use [ViennaRNA/plotting/alignments.h](#) instead

18.196 plot_aln.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_ALN_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_ALN_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/plot_aln.h>! Use
    <ViennaRNA/plotting/alignments.h> instead!"
00013 #   endif
00014 #include <ViennaRNA/plotting/alignments.h>
00015 #endif
00016
00017 #endif
```

18.197 ViennaRNA/plot_layouts.h File Reference

Use [ViennaRNA/plotting/layouts.h](#) instead.

Include dependency graph for plot_layouts.h:

18.197.1 Detailed Description

Use [ViennaRNA/plotting/layouts.h](#) instead.

Deprecated Use [ViennaRNA/plotting/layouts.h](#) instead

18.198 plot_layouts.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/plot_layouts.h>! Use
    <ViennaRNA/plotting/layouts.h> instead!"
00013 #   endif
00014 #include <ViennaRNA/plotting/layouts.h>
00015 #endif
00016
00017 #endif
```

18.199 ViennaRNA/plot_structure.h File Reference

Use [ViennaRNA/plotting/structures.h](#) instead.

Include dependency graph for plot_structure.h:

18.199.1 Detailed Description

Use [ViennaRNA/plotting/structures.h](#) instead.

Deprecated Use [ViennaRNA/plotting/structures.h](#) instead

18.200 plot_structure.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
```

```

00012 #warning "Including deprecated header file <ViennaRNA/plot_structure.h>! Use
      <ViennaRNA/plotting/structures.h> instead!"
00013 # endif
00014 #include <ViennaRNA/plotting/structures.h>
00015 #endif
00016
00017 #endif

```

18.201 ViennaRNA/plot_utils.h File Reference

Use [ViennaRNA/plotting/utlis.h](#) instead.

Include dependency graph for plot_utils.h:

18.201.1 Detailed Description

Use [ViennaRNA/plotting/utlis.h](#) instead.

Deprecated Use [ViennaRNA/plotting/utlis.h](#) instead

18.202 plot_utils.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_UTILS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_UTILS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/plot_utils.h>! Use <ViennaRNA/plotting/utlis.h>
      instead!"
00013 # endif
00014 #include <ViennaRNA/plotting/utlis.h>
00015 #endif
00016
00017 #endif

```

18.203 ViennaRNA/plotting/alignments.h File Reference

Various functions for plotting Sequence / Structure Alignments.

This graph shows which files directly or indirectly include this file:

Functions

- int [vrna_file_PS_aln](#) (const char *filename, const char **seqs, const char **names, const char *structure, unsigned int columns)
Create an annotated PostScript alignment plot.
- int [vrna_file_PS_aln_slice](#) (const char *filename, const char **seqs, const char **names, const char *structure, unsigned int start, unsigned int end, int offset, unsigned int columns)
Create an annotated PostScript alignment plot.
- int [PS_color_aln](#) (const char *structure, const char *filename, const char *seqs[], const char *names[])
Produce PostScript sequence alignment color-annotated by consensus structure.
- int [aliPS_color_aln](#) (const char *structure, const char *filename, const char *seqs[], const char *names[])
PS_color_aln for duplexes.

18.203.1 Detailed Description

Various functions for plotting Sequence / Structure Alignments.

18.204 alignments.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_ALN_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_ALN_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00039 int
00040 vrna_file_PS_aln(const char *filename,
00041                 const char **seqs,
00042                 const char **names,
00043                 const char *structure,
00044                 unsigned int columns);
00045
00046
00065 int
00066 vrna_file_PS_aln_slice(const char *filename,
00067                       const char **seqs,
00068                       const char **names,
00069                       const char *structure,
00070                       unsigned int start,
00071                       unsigned int end,
00072                       int offset,
00073                       unsigned int columns);
00074
00075
00080 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00081
00089 DEPRECATED(int PS_color_aln(const char *structure,
00090                             const char *filename,
00091                             const char *seqs[],
00092                             const char *names[]),
00093            "Use vrna_file_PS_aln() instead!");
00094
00095
00102 DEPRECATED(int aliPS_color_aln(const char *structure,
00103                                const char *filename,
00104                                const char *seqs[],
00105                                const char *names[]),
00106            "Use vrna_file_PS_aln() instead!");
00107
00108 #endif
00109
00110 #endif

```

18.205 ViennaRNA/utils/alignments.h File Reference

Various utility- and helper-functions for sequence alignments and comparative structure prediction.

Include dependency graph for alignments.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_pinfo_s](#)
A base pair info structure. [More...](#)

Macros

- #define **VRNA_ALN_DEFAULT** 0U
Use default alignment settings.
- #define **VRNA_ALN_RNA** 1U
Convert to RNA alphabet.
- #define **VRNA_ALN_DNA** 2U
Convert to DNA alphabet.

- `#define VRNA_ALN_UPPERCASE 4U`
Convert to uppercase nucleotide letters.
- `#define VRNA_ALN_LOWERCASE 8U`
Convert to lowercase nucleotide letters.
- `#define VRNA_MEASURE_SHANNON_ENTROPY 1U`
Flag indicating Shannon Entropy measure.

Typedefs

- `typedef struct vrna_pinfo_s vrna_pinfo_t`
Typename for the base pair info representing data structure `vrna_pinfo_s`.
- `typedef struct vrna_pinfo_s pair_info`
Old typename of `vrna_pinfo_s`.

Functions

- `int vrna_aln_mpi (const char **alignment)`
Get the mean pairwise identity in steps from ?to?(ident)
- `vrna_pinfo_t * vrna_aln_pinfo (vrna_fold_compound_t *vc, const char *structure, double threshold)`
Retrieve an array of `vrna_pinfo_t` structures from precomputed pair probabilities.
- `char ** vrna_aln_slice (const char **alignment, unsigned int i, unsigned int j)`
Slice out a subalignment from a larger alignment.
- `void vrna_aln_free (char **alignment)`
Free memory occupied by a set of aligned sequences.
- `char ** vrna_aln_uppercase (const char **alignment)`
Create a copy of an alignment with only uppercase letters in the sequences.
- `char ** vrna_aln_toRNA (const char **alignment)`
Create a copy of an alignment where DNA alphabet is replaced by RNA alphabet.
- `char ** vrna_aln_copy (const char **alignment, unsigned int options)`
Make a copy of a multiple sequence alignment.
- `float * vrna_aln_conservation_struct (const char **alignment, const char *structure, const vrna_md_t *md)`
Compute base pair conservation of a consensus structure.
- `float * vrna_aln_conservation_col (const char **alignment, const vrna_md_t *md_p, unsigned int options)`
Compute nucleotide conservation in an alignment.
- `char * vrna_aln_consensus_sequence (const char **alignment, const vrna_md_t *md_p)`
Compute the consensus sequence for a given multiple sequence alignment.
- `char * vrna_aln_consensus_mis (const char **alignment, const vrna_md_t *md_p)`
Compute the Most Informative Sequence (MIS) for a given multiple sequence alignment.
- `int get_mpi (char *Aseq[], int n_seq, int length, int *mini)`
Get the mean pairwise identity in steps from ?to?(ident)
- `void encode_aln_sequence (const char *sequence, short *S, short *s5, short *s3, char *ss, unsigned short *as, int circ)`
Get arrays with encoded sequence of the alignment.
- `void alloc_sequence_arrays (const char **sequences, short ***S, short ***S5, short ***S3, unsigned short ***a2s, char ***Ss, int circ)`
Allocate memory for sequence array used to deal with aligned sequences.
- `void free_sequence_arrays (unsigned int n_seq, short ***S, short ***S5, short ***S3, unsigned short ***a2s, char ***Ss)`
Free the memory of the sequence arrays used to deal with aligned sequences.

18.205.1 Detailed Description

Various utility- and helper-functions for sequence alignments and comparative structure prediction.

18.206 alignments.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_ALN_UTIL_H
00002 #define VIENNA_RNA_PACKAGE_ALN_UTIL_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00029 typedef struct vrna_pinfo_s vrna_pinfo_t;
00030
00031
00035 #define VRNA_ALN_DEFAULT      0U
00036
00037
00041 #define VRNA_ALN_RNA          1U
00042
00043
00047 #define VRNA_ALN_DNA          2U
00048
00049
00053 #define VRNA_ALN_UPPERCASE    4U
00054
00055
00059 #define VRNA_ALN_LOWERCASE     8U
00060
00066 #define VRNA_MEASURE_SHANNON_ENTROPY 1U
00067
00068 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00069
00070 /* the following typedefs are for backward compatibility only */
00071
00077 typedef struct vrna_pinfo_s pair_info;
00078
00079 #endif
00080
00081 #include <ViennaRNA/fold_compound.h>
00082 #include <ViennaRNA/model.h>
00083
00094 struct vrna_pinfo_s {
00095     unsigned i;
00096     unsigned j;
00097     float p;
00098     float ent;
00099     short bp[8];
00100     char comp;
00101 };
00102
00103
00110 int
00111 vrna_aln_mpi(const char **alignment);
00112
00113
00127 vrna_pinfo_t *
00128 vrna_aln_pinfo(vrna_fold_compound_t *vc,
00129               const char *structure,
00130               double threshold);
00131
00132
00133 int *
00134 vrna_aln_pscore(const char **alignment,
00135                 vrna_md_t *md);
00136
00137
00138 int
00139 vrna_pscore(vrna_fold_compound_t *fc,
00140             unsigned int i,
00141             unsigned int j);
00142
```

```
00143
00144 int
00145 vrna_pscore_freq(vrna_fold_compound_t *fc,
00146                 const unsigned int *frequencies,
00147                 unsigned int pairs);
00148
00162 char **
00163 vrna_aln_slice(const char **alignment,
00164               unsigned int i,
00165               unsigned int j);
00166
00167
00173 void
00174 vrna_aln_free(char **alignment);
00175
00176
00185 char **
00186 vrna_aln_uppercase(const char **alignment);
00187
00188
00197 char **
00198 vrna_aln_toRNA(const char **alignment);
00199
00200
00214 char **
00215 vrna_aln_copy(const char **alignment,
00216               unsigned int options);
00217
00218
00233 float *
00234 vrna_aln_conservation_struct(const char **alignment,
00235                             const char *structure,
00236                             const vrna_md_t *md);
00237
00238
00256 float *
00257 vrna_aln_conservation_col(const char **alignment,
00258                           const vrna_md_t *md_p,
00259                           unsigned int options);
00260
00261
00269 char *
00270 vrna_aln_consensus_sequence(const char **alignment,
00271                             const vrna_md_t *md_p);
00272
00273
00284 char *
00285 vrna_aln_consensus_mis(const char **alignment,
00286                       const vrna_md_t *md_p);
00287
00288
00289 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00290
00291 #include <stdio.h>
00295 DEPRECATED(int read_clustal(FILE *clust,
00296                             char *AlignedSeqs[],
00297                             char *names[]),
00298             "Use vrna_file_msa_read() and vrna_file_msa_read_record() instead");
00299
00300
00304 DEPRECATED(char *consensus(const char *AS[]),
00305             "Use vrna_aln_consensus_sequence() instead!");
00306
00307
00311 DEPRECATED(char *consens_mis(const char *AS[]),
00312             "Use vrna_aln_consensus_mis() instead!");
00313
00314
00318 DEPRECATED(char *get_ungapped_sequence(const char *seq),
00319             "Use vrna_seq_ungapped() instead!");
00320
00321
00333 DEPRECATED(int get_mpi(char *Aseq[],
00334                        int n_seq,
00335                        int length,
00336                        int *mini),
00337             "Use vrna_aln_mpi() instead");
00338
00339 /*
00340 #####
00341 # some helper functions that might be useful in the library #
00342 #####
00343 */
00344
00360 DEPRECATED(void encode_ali_sequence(const char *sequence,
00361                                     short *S,
00362                                     short *s5,
00363                                     short *s3,
```

```

00364             char          *ss,
00365             unsigned short *as,
00366             int            circ),
00367     "This function is obsolete");
00368
00369
00386 DEPRECATED(void alloc_sequence_arrays(const char **sequences,
00387             short ***S,
00388             short ***S5,
00389             short ***S3,
00390             unsigned short ***a2s,
00391             char ***Ss,
00392             int circ),
00393     "This function is obsolete");
00394
00395
00411 DEPRECATED(void free_sequence_arrays(unsigned int n_seq,
00412             short ***S,
00413             short ***S5,
00414             short ***S3,
00415             unsigned short ***a2s,
00416             char ***Ss),
00417     "This fuction is obsolete");
00418
00419 #endif
00420
00426 #endif

```

18.207 ViennaRNA/plotting/layouts.h File Reference

Secondary structure plot layout algorithms.

Include dependency graph for layouts.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_plot_layout_s](#)
- struct [COORDINATE](#)

this is a workaround for the SWIG Perl Wrapper RNA plot function that returns an array of type [COORDINATE](#) More...

Macros

- #define [VRNA_PLOT_TYPE_SIMPLE](#) 0
Definition of Plot type simple
- #define [VRNA_PLOT_TYPE_NAVIEW](#) 1
Definition of Plot type Naview
- #define [VRNA_PLOT_TYPE_CIRCULAR](#) 2
Definition of Plot type Circular
- #define [VRNA_PLOT_TYPE_TURTLE](#) 3
Definition of Plot type Turtle [30].
- #define [VRNA_PLOT_TYPE_PUZZLER](#) 4
Definition of Plot type RNApuzzler [30].

Typedefs

- typedef struct [vrna_plot_layout_s](#) [vrna_plot_layout_t](#)
RNA secondary structure figure layout.

Functions

- [vrna_plot_layout_t](#) * [vrna_plot_layout](#) (const char *structure, unsigned int plot_type)
Create a layout (coordinates, etc.) for a secondary structure plot.
- [vrna_plot_layout_t](#) * [vrna_plot_layout_simple](#) (const char *structure)
Create a layout (coordinates, etc.) for a simple secondary structure plot.

- `vrna_plot_layout_t * vrna_plot_layout_circular` (const char *structure)
Create a layout (coordinates, etc.) for a circular secondary structure plot.
- `vrna_plot_layout_t * vrna_plot_layout_turtle` (const char *structure)
Create a layout (coordinates, etc.) for a secondary structure plot using the Turtle Algorithm [30].
- `vrna_plot_layout_t * vrna_plot_layout_puzzler` (const char *structure, `vrna_plot_options_puzzler_t` *options)
Create a layout (coordinates, etc.) for a secondary structure plot using the RNApuzzler Algorithm [30].
- `void vrna_plot_layout_free` (`vrna_plot_layout_t` *layout)
Free memory occupied by a figure layout data structure.
- `int vrna_plot_coords` (const char *structure, float **x, float **y, int plot_type)
Compute nucleotide coordinates for secondary structure plot.
- `int vrna_plot_coords_pt` (const short *pt, float **x, float **y, int plot_type)
Compute nucleotide coordinates for secondary structure plot.
- `int vrna_plot_coords_simple` (const char *structure, float **x, float **y)
Compute nucleotide coordinates for secondary structure plot the Simple way
- `int vrna_plot_coords_simple_pt` (const short *pt, float **x, float **y)
Compute nucleotide coordinates for secondary structure plot the Simple way
- `int vrna_plot_coords_circular` (const char *structure, float **x, float **y)
Compute coordinates of nucleotides mapped in equal distances onto a unit circle.
- `int vrna_plot_coords_circular_pt` (const short *pt, float **x, float **y)
Compute nucleotide coordinates for a Circular Plot
- `int simple_xy_coordinates` (short *pair_table, float *X, float *Y)
Calculate nucleotide coordinates for secondary structure plot the Simple way
- `int simple_circplot_coordinates` (short *pair_table, float *x, float *y)
Calculate nucleotide coordinates for Circular Plot

Variables

- `int rna_plot_type`
Switch for changing the secondary structure layout algorithm.

18.207.1 Detailed Description

Secondary structure plot layout algorithms.

18.208 layouts.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_LAYOUTS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00036 typedef struct vrna_plot_layout_s vrna_plot_layout_t;
00037
00038
00039 #include <ViennaRNA/datastructures/basic.h>
00040
00041 #ifdef VRNA_WITH_NAVIEW_LAYOUT
00042 #include <ViennaRNA/plotting/naview/naview.h>
00043 #endif
00044
00045 #include "ViennaRNA/plotting/RNApuzzler/RNAturtle.h"
```

```
00046 #include "ViennaRNA/plotting/RNApuzzler/RNApuzzler.h"
00047
00048
00057 #define VRNA_PLOT_TYPE_SIMPLE      0
00058
00067 #define VRNA_PLOT_TYPE_NAVIEW      1
00068
00077 #define VRNA_PLOT_TYPE_CIRCULAR    2
00078
00083 #define VRNA_PLOT_TYPE_TURTLE      3
00084
00089 #define VRNA_PLOT_TYPE_PUZZLER     4
00090
00091 #ifndef VRNA_WITH_NAVIEW_LAYOUT
00092 # define VRNA_PLOT_TYPE_DEFAULT    VRNA_PLOT_TYPE_NAVIEW
00093 #else
00094 # define VRNA_PLOT_TYPE_DEFAULT    VRNA_PLOT_TYPE_PUZZLER
00095 #endif
00096
00097
00098 struct vrna_plot_layout_s {
00099     unsigned int    length;
00100     float           **x;
00101     float           **y;
00102     double          *arcs;
00103     int             bbox[4];
00104 };
00105
00106
00133 vrna_plot_layout_t *
00134 vrna_plot_layout(const char *structure,
00135                 unsigned int plot_type);
00136
00137
00153 vrna_plot_layout_t *
00154 vrna_plot_layout_simple(const char *structure);
00155
00156
00157 #ifdef VRNA_WITH_NAVIEW_LAYOUT
00173 vrna_plot_layout_t *
00174 vrna_plot_layout_naview(const char *structure);
00175 #endif
00176
00192 vrna_plot_layout_t *
00193 vrna_plot_layout_circular(const char *structure);
00194
00195
00211 vrna_plot_layout_t *
00212 vrna_plot_layout_turtle(const char *structure);
00213
00214
00230 vrna_plot_layout_t *
00231 vrna_plot_layout_puzzler(const char *structure,
00232                         vrna_plot_options_puzzler_t *options);
00233
00234
00244 void
00245 vrna_plot_layout_free(vrna_plot_layout_t *layout);
00246
00247
00291 int
00292 vrna_plot_coords(const char *structure,
00293                 float **x,
00294                 float **y,
00295                 int plot_type);
00296
00297
00317 int
00318 vrna_plot_coords_pt(const short *pt,
00319                   float **x,
00320                   float **y,
00321                   int plot_type);
00322
00323
00356 int
00357 vrna_plot_coords_simple(const char *structure,
00358                       float **x,
00359                       float **y);
00360
00361
00380 int
00381 vrna_plot_coords_simple_pt(const short *pt,
00382                          float **x,
00383                          float **y);
00384
00385
00416 int
```

```

00417 vrna_plot_coords_circular(const char *structure,
00418                             float **x,
00419                             float **y);
00420
00421
00440 int
00441 vrna_plot_coords_circular_pt(const short *pt,
00442                             float **x,
00443                             float **y);
00444
00445
00451 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00452
00463 typedef struct {
00464     float X; /* X coords */
00465     float Y; /* Y coords */
00466 } COORDINATE;
00467
00468
00481 extern int rna_plot_type;
00482
00483
00497 DEPRECATED(int
00498             simple_xy_coordinates(short *pair_table,
00499                                 float *X,
00500                                 float *Y),
00501             "Use vrna_plot_coords_simple_pt() instead!");
00502
00503
00526 DEPRECATED(int
00527             simple_circplot_coordinates(short *pair_table,
00528                                         float *x,
00529                                         float *y),
00530             "Use vrna_plot_coords_circular_pt() instead!");
00531
00532
00537 #endif
00538
00539
00540 #endif

```

18.209 ViennaRNA/plotting/probabilities.h File Reference

Various functions for plotting RNA secondary structures, dot-plots and other visualizations.

Include dependency graph for probabilities.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_dotplot_auxdata_t](#)

Functions

- int [PS_dot_plot_list](#) (char *seq, char *filename, [vrna_ep_t](#) *pl, [vrna_ep_t](#) *mf, char *comment)
Produce a postscript dot-plot from two pair lists.
- int [PS_dot_plot](#) (char *string, char *file)
Produce postscript dot-plot.

18.209.1 Detailed Description

Various functions for plotting RNA secondary structures, dot-plots and other visualizations.

18.210 probabilities.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_H
00003
00004
00005 #include <ViennaRNA/datastructures/basic.h>
00006 #include <ViennaRNA/utils/structures.h>
00007
00008 #ifdef VRNA_WARN_DEPRECATED

```

```

00009 # if defined(__clang__)
00010 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(" ", msg)))
00011 #   elif defined(__GNUC__)
00012 #   define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00013 #   else
00014 #   define DEPRECATED(func, msg) func
00015 #   endif
00016 #else
00017 #   define DEPRECATED(func, msg) func
00018 #endif
00019
00032 #define VRNA_PLOT_PROBABILITIES_BP      1U
00033 #define VRNA_PLOT_PROBABILITIES_ACC     2U
00034
00035 #define VRNA_PLOT_PROBABILITIES_UD      4U
00036 #define VRNA_PLOT_PROBABILITIES_UD_LIN  8U
00037
00038 #define VRNA_PLOT_PROBABILITIES_SD      16U
00039
00040 #define VRNA_PLOT_PROBABILITIES_SC_MOTIF 32U
00041 #define VRNA_PLOT_PROBABILITIES_SC_UP    64U
00042 #define VRNA_PLOT_PROBABILITIES_SC_BP    128U
00043
00044 #define VRNA_PLOT_PROBABILITIES_DEFAULT (VRNA_PLOT_PROBABILITIES_BP \
00045                                         | VRNA_PLOT_PROBABILITIES_SD \
00046                                         | VRNA_PLOT_PROBABILITIES_SC_MOTIF \
00047                                         | VRNA_PLOT_PROBABILITIES_UD_LIN)
00048 typedef struct {
00049     char          *comment;
00050     char          *title;
00051
00052     vrna_data_lin_t **top;
00053     char          **top_title;
00054
00055     vrna_data_lin_t **bottom;
00056     char          **bottom_title;
00057
00058     vrna_data_lin_t **left;
00059     char          **left_title;
00060
00061     vrna_data_lin_t **right;
00062     char          **right_title;
00063 } vrna_dotplot_auxdata_t;
00064
00065
00066 int
00067 vrna_plot_dp_EPS(const char          *filename,
00068                 const char          *sequence,
00069                 vrna_ep_t           *upper,
00070                 vrna_ep_t           *lower,
00071                 vrna_dotplot_auxdata_t *auxdata,
00072                 unsigned int         options);
00073
00074
00075 int
00076 vrna_plot_dp_PS_list(char          *seq,
00077                     int           cp,
00078                     char          *wastlfile,
00079                     vrna_ep_t     *pl,
00080                     vrna_ep_t     *mf,
00081                     char          *comment);
00082
00083
00084 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00085
00086 int
00087 PS_color_dot_plot(char          *string,
00088                  vrna_cpair_t   *pi,
00089                  char          *filename);
00090
00091
00092 int
00093 PS_color_dot_plot_turn(char          *seq,
00094                      vrna_cpair_t   *pi,
00095                      char          *filename,
00096                      int           winSize);
00097
00098
00099 int
00100 PS_dot_plot_turn(char          *seq,
00101                 vrna_ep_t     *pl,
00102                 char          *filename,
00103                 int           winSize);
00104
00105
00125 int PS_dot_plot_list(char          *seq,
00126                    char          *filename,

```

```

00127         vrna_ep_t    *pl,
00128         vrna_ep_t    *mf,
00129         char          *comment);
00130
00131
00147 DEPRECATED (int PS_dot_plot(char *string,
00148                             char *file),
00149             "Use vrna_plot_dp_EPS() instead");
00150
00151 #endif
00152
00157 #endif

```

18.211 ViennaRNA/plotting/RNAPuzzler/RNAPuzzler.h File Reference

Implementation of the RNAPuzzler RNA secondary structure layout algorithm [30].

This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_plot_options_puzzler_t](#)
Options data structure for RNAPuzzler algorithm implementation. [More...](#)

Functions

- int [vrna_plot_coords_puzzler](#) (const char *structure, float **x, float **y, double **arc_coords, [vrna_plot_options_puzzler_t](#) *options)
Compute nucleotide coordinates for secondary structure plot using the RNAPuzzler algorithm [30].
- int [vrna_plot_coords_puzzler_pt](#) (short const *const pair_table, float **x, float **y, double **arc_coords, [vrna_plot_options_puzzler_t](#) *puzzler)
Compute nucleotide coordinates for secondary structure plot using the RNAPuzzler algorithm [30].
- [vrna_plot_options_puzzler_t](#) * [vrna_plot_options_puzzler](#) (void)
Create an RNAPuzzler options data structure.
- void [vrna_plot_options_puzzler_free](#) ([vrna_plot_options_puzzler_t](#) *options)
Free memory occupied by an RNAPuzzler options data structure.

18.211.1 Detailed Description

Implementation of the RNAPuzzler RNA secondary structure layout algorithm [30].

18.212 RNAPuzzler.h

[Go to the documentation of this file.](#)

```

00001 #ifndef RNAPUZZLER_H
00002 #define RNAPUZZLER_H
00003
00020 typedef struct {
00021     /*
00022      * variables fixed during operation
00023      * drawing behavior
00024      */
00025     short    drawArcs;
00026     double   paired;
00027     double   unpaired;
00028
00029     /* intersection resolution behavior */
00030     short    checkAncestorIntersections;
00031     short    checkSiblingIntersections;
00032     short    checkExteriorIntersections;
00033     short    allowFlipping;
00034     short    optimize;
00035     int      maximumNumberOfConfigChangesAllowed;
00036
00037
00038     /* import behavior - unused for now */
00039     char     *config; /* file path */
00040

```

```

00041  /* other stuff */
00042  const char *filename;
00043
00044  /* variables changed during operation */
00045  int      numberOfChangesAppliedToConfig;
00046  int      psNumber;
00047 } vrna_plot_options_puzzler_t;
00048
00049
00087 int
00088 vrna_plot_coords_puzzler(const char      *structure,
00089                          float          **x,
00090                          float          **y,
00091                          double         **arc_coords,
00092                          vrna_plot_options_puzzler_t *options);
00093
00094
00115 int
00116 vrna_plot_coords_puzzler_pt(short const *const pair_table,
00117                             float       **x,
00118                             float       **y,
00119                             double      **arc_coords,
00120                             vrna_plot_options_puzzler_t *puzzler);
00121
00122
00131 vrna_plot_options_puzzler_t *
00132 vrna_plot_options_puzzler(void);
00133
00134
00143 void
00144 vrna_plot_options_puzzler_free(vrna_plot_options_puzzler_t *options);
00145
00146
00152 #endif

```

18.213 ViennaRNA/plotting/RNApuzzler/RNAturtle.h File Reference

Implementation of the RNAturtle RNA secondary structure layout algorithm [30].

This graph shows which files directly or indirectly include this file:

Functions

- int [vrna_plot_coords_turtle](#) (const char *structure, float **x, float **y, double **arc_coords)
Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [30].
- int [vrna_plot_coords_turtle_pt](#) (short const *const pair_table, float **x, float **y, double **arc_coords)
Compute nucleotide coordinates for secondary structure plot using the RNAturtle algorithm [30].

18.213.1 Detailed Description

Implementation of the RNAturtle RNA secondary structure layout algorithm [30].

18.214 RNAturtle.h

[Go to the documentation of this file.](#)

```

00001 #ifndef RNATURTLE_H
00002 #define RNATURTLE_H
00003
00052 int
00053 vrna_plot_coords_turtle(const char *structure,
00054                         float       **x,
00055                         float       **y,
00056                         double      **arc_coords);
00057
00058
00078 int
00079 vrna_plot_coords_turtle_pt(short const *const pair_table,
00080                             float       **x,
00081                             float       **y,
00082                             double      **arc_coords);
00083
00084
00089 #endif

```

18.215 ViennaRNA/plotting/structures.h File Reference

Various functions for plotting RNA secondary structures.

Include dependency graph for structures.h: This graph shows which files directly or indirectly include this file:

Functions

- `int vrna_file_PS_rnaplot` (const char *seq, const char *structure, const char *file, `vrna_md_t` *md_p)
Produce a secondary structure graph in PostScript and write it to 'filename'.
- `int vrna_file_PS_rnaplot_a` (const char *seq, const char *structure, const char *file, const char *pre, const char *post, `vrna_md_t` *md_p)
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.
- `int gmlRNA` (char *string, char *structure, char *ssfile, char option)
Produce a secondary structure graph in Graph Meta Language (gml) and write it to a file.
- `int ssv_rna_plot` (char *string, char *structure, char *ssfile)
Produce a secondary structure graph in SStructView format.
- `int svg_rna_plot` (char *string, char *structure, char *ssfile)
Produce a secondary structure plot in SVG format and write it to a file.
- `int xrna_plot` (char *string, char *structure, char *ssfile)
Produce a secondary structure plot for further editing in XRNA.
- `int PS_rna_plot` (char *string, char *structure, char *file)
Produce a secondary structure graph in PostScript and write it to 'filename'.
- `int PS_rna_plot_a` (char *string, char *structure, char *file, char *pre, char *post)
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename'.
- `int PS_rna_plot_a_gquad` (char *string, char *structure, char *ssfile, char *pre, char *post)
Produce a secondary structure graph in PostScript including additional annotation macros and write it to 'filename' (detect and draw g-quadruplexes)

18.215.1 Detailed Description

Various functions for plotting RNA secondary structures.

18.216 structures.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_STRUCTURE_H
00003
00004 #include <ViennaRNA/model.h>
00005 #include <ViennaRNA/plotting/layouts.h>
00006 #include "ViennaRNA/plotting/RNApuzzler/RNApuzzler.h"
00007
00008 #ifdef VRNA_WARN_DEPRECATED
00009 # if defined(__clang__)
00010 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00011 # elif defined(__GNUC__)
00012 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00013 # else
00014 #  define DEPRECATED(func, msg) func
00015 # endif
00016 #else
00017 # define DEPRECATED(func, msg) func
00018 #endif
00019
00044 int
00045 vrna_file_PS_rnaplot(const char *seq,
00046                    const char *structure,
00047                    const char *file,
00048                    vrna_md_t *md_p);
00049
00050
00070 int vrna_file_PS_rnaplot_a(const char *seq,
00071                          const char *structure,
00072                          const char *file,
00073                          const char *pre,
```

```

00074             const char      *post,
00075             vrna_md_t        *md_p);
00076
00077
00078 int
00079 vrna_file_PS_rnaplot_layout(const char      *seq,
00080                             const char      *structure,
00081                             const char      *ssfile,
00082                             const char      *pre,
00083                             const char      *post,
00084                             vrna_md_t        *md_p,
00085                             vrna_plot_layout_t *layout);
00086
00087 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00088
00089 /* write PostScript drawing of structure to file with annotation */
00090 int
00091 PS_rna_plot_snoop_a(const char *string,
00092                    const char *structure,
00093                    const char *ssfile,
00094                    int *relative_access,
00095                    const char *seqs[]);
00096
00097
00110 int
00111 gmlRNA(char *string,
00112        char *structure,
00113        char *ssfile,
00114        char option);
00115
00116
00127 int
00128 ssv_rna_plot(char *string,
00129             char *structure,
00130             char *ssfile);
00131
00132
00141 int
00142 svg_rna_plot(char *string,
00143             char *structure,
00144             char *ssfile);
00145
00146
00155 int
00156 xrna_plot(char *string,
00157          char *structure,
00158          char *ssfile);
00159
00160
00166 DEPRECATED(int PS_rna_plot(char *string,
00167                             char *structure,
00168                             char *file),
00169             "Use vrna_file_PS_rnaplot() instead");
00170
00177 DEPRECATED(int PS_rna_plot_a(char *string,
00178                               char *structure,
00179                               char *file,
00180                               char *pre,
00181                               char *post),
00182             "Use vrna_file_PS_rnaplot_a() instead");
00183
00190 DEPRECATED(int PS_rna_plot_a_gquad(char *string,
00191                                     char *structure,
00192                                     char *ssfile,
00193                                     char *pre,
00194                                     char *post),
00195             "Use vrna_file_PS_rnaplot_a() instead");
00196
00197 #endif
00198
00203 #endif

```

18.217 ViennaRNA/utils/structures.h File Reference

Various utility- and helper-functions for secondary structure parsing, converting, etc.

Include dependency graph for structures.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_elem_prob_s](#)

Data structure representing a single entry of an element probability list (e.g. list of pair probabilities) [More...](#)

- struct [vrna_hx_s](#)

Data structure representing an entry of a helix list. [More...](#)

Macros

- #define [VRNA_BRACKETS_ALPHA](#) 4U
Bitflag to indicate secondary structure notations using uppercase/lowercase letters from the latin alphabet.
- #define [VRNA_BRACKETS_RND](#) 8U
Bitflag to indicate secondary structure notations using round brackets (parenthesis), ()
- #define [VRNA_BRACKETS_CLY](#) 16U
Bitflag to indicate secondary structure notations using curly brackets, { }
- #define [VRNA_BRACKETS_ANG](#) 32U
Bitflag to indicate secondary structure notations using angular brackets, < >
- #define [VRNA_BRACKETS_SQR](#) 64U
Bitflag to indicate secondary structure notations using square brackets, []
- #define [VRNA_BRACKETS_DEFAULT](#)
Default bitmask to indicate secondary structure notation using any pair of brackets.
- #define [VRNA_BRACKETS_ANY](#)
Bitmask to indicate secondary structure notation using any pair of brackets or uppercase/lowercase alphabet letters.
- #define [VRNA_PLIST_TYPE_BASEPAIR](#) 0
A Base Pair element.
- #define [VRNA_PLIST_TYPE_GQUAD](#) 1
A G-Quadruplex element.
- #define [VRNA_PLIST_TYPE_H_MOTIF](#) 2
A Hairpin loop motif element.
- #define [VRNA_PLIST_TYPE_I_MOTIF](#) 3
An Internal loop motif element.
- #define [VRNA_PLIST_TYPE_UD_MOTIF](#) 4
An Unstructured Domain motif element.
- #define [VRNA_PLIST_TYPE_STACK](#) 5
A Base Pair stack element.
- #define [VRNA_PLIST_TYPE_UNPAIRED](#) 6
An unpaired base.
- #define [VRNA_PLIST_TYPE_TRIPLE](#) 7
One pair of a base triplet.
- #define [VRNA_STRUCTURE_TREE_HIT](#) 1U
Homeomorphically Irreducible [Tree](#) (HIT) representation of a secondary structure.
- #define [VRNA_STRUCTURE_TREE_SHAPIRO_SHORT](#) 2U
(short) Coarse Grained representation of a secondary structure
- #define [VRNA_STRUCTURE_TREE_SHAPIRO](#) 3U
(full) Coarse Grained representation of a secondary structure
- #define [VRNA_STRUCTURE_TREE_SHAPIRO_EXT](#) 4U
(extended) Coarse Grained representation of a secondary structure
- #define [VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT](#) 5U
(weighted) Coarse Grained representation of a secondary structure
- #define [VRNA_STRUCTURE_TREE_EXPANDED](#) 6U
Expanded [Tree](#) representation of a secondary structure.

Typedefs

- typedef struct [vrna_hx_s](#) [vrna_hx_t](#)
Convenience typedef for data structure [vrna_hx_s](#).
- typedef struct [vrna_elem_prob_s](#) [vrna_ep_t](#)
Convenience typedef for data structure [vrna_elem_prob_s](#).

Functions

- char * [vrna_db_pack](#) (const char *struc)
Pack secondary structure, 5:1 compression using base 3 encoding.
- char * [vrna_db_unpack](#) (const char *packed)
Unpack secondary structure previously packed with [vrna_db_pack\(\)](#)
- void [vrna_db_flatten](#) (char *structure, unsigned int options)
Substitute pairs of brackets in a string with parenthesis.
- void [vrna_db_flatten_to](#) (char *string, const char target[3], unsigned int options)
Substitute pairs of brackets in a string with another type of pair characters.
- char * [vrna_db_from_ptable](#) (const short *pt)
Convert a pair table into dot-parenthesis notation.
- char * [vrna_db_from_plist](#) ([vrna_ep_t](#) *pairs, unsigned int n)
Convert a list of base pairs into dot-bracket notation.
- char * [vrna_db_to_element_string](#) (const char *structure)
Convert a secondary structure in dot-bracket notation to a nucleotide annotation of loop contexts.
- char * [vrna_db_pk_remove](#) (const char *structure, unsigned int options)
Remove pseudo-knots from an input structure.
- short * [vrna_ptable](#) (const char *structure)
Create a pair table from a dot-bracket notation of a secondary structure.
- short * [vrna_ptable_from_string](#) (const char *structure, unsigned int options)
Create a pair table for a secondary structure string.
- short * [vrna_pt_pk_get](#) (const char *structure)
Create a pair table of a secondary structure (pseudo-knot version)
- short * [vrna_ptable_copy](#) (const short *pt)
Get an exact copy of a pair table.
- short * [vrna_pt_ali_get](#) (const char *structure)
Create a pair table of a secondary structure (snoop align version)
- short * [vrna_pt_snoop_get](#) (const char *structure)
Create a pair table of a secondary structure (snoop version)
- short * [vrna_pt_pk_remove](#) (const short *ptable, unsigned int options)
Remove pseudo-knots from a pair table.
- [vrna_ep_t](#) * [vrna_plist](#) (const char *struc, float pr)
Create a [vrna_ep_t](#) from a dot-bracket string.
- [vrna_ep_t](#) * [vrna_plist_from_probs](#) ([vrna_fold_compound_t](#) *vc, double cut_off)
Create a [vrna_ep_t](#) from base pair probability matrix.
- char * [vrna_db_from_WUSS](#) (const char *wuss)
Convert a WUSS annotation string to dot-bracket format.
- char * [vrna_abstract_shapes](#) (const char *structure, unsigned int level)
Convert a secondary structure in dot-bracket notation to its abstract shapes representation.
- char * [vrna_abstract_shapes_pt](#) (const short *pt, unsigned int level)
Convert a secondary structure to its abstract shapes representation.
- [vrna_hx_t](#) * [vrna_hx_from_ptable](#) (short *pt)
Convert a pair table representation of a secondary structure into a helix list.
- [vrna_hx_t](#) * [vrna_hx_merge](#) (const [vrna_hx_t](#) *list, int maxdist)

- Create a merged helix list from another helix list.*

 - `int * vrna_loopidx_from_ptable` (const short *pt)
- Get a loop index representation of a structure.*

 - `int vrna_bp_distance_pt` (const short *pt1, const short *pt2)

Compute the "base pair" distance between two pair tables pt1 and pt2 of secondary structures.
- `int vrna_bp_distance` (const char *str1, const char *str2)

Compute the "base pair" distance between two secondary structures s1 and s2.
- `unsigned int * vrna_refBPcnt_matrix` (const short *reference_pt, unsigned int turn)

Make a reference base pair count matrix.
- `unsigned int * vrna_refBPdist_matrix` (const short *pt1, const short *pt2, unsigned int turn)

Make a reference base pair distance matrix.
- `char * vrna_db_from_probs` (const FLT_OR_DBL *pr, unsigned int length)

Create a dot-bracket like structure string from base pair probability matrix.
- `char vrna_bpp_symbol` (const float *x)

Get a pseudo dot bracket notation for a given probability information.
- `char * vrna_db_from_bp_stack` (vrna_bp_stack_t *bp, unsigned int length)

Create a dot-bracket/parenthesis structure from backtracking stack.
- `char * vrna_db_to_tree_string` (const char *structure, unsigned int type)

Convert a Dot-Bracket structure string into tree string representation.
- `char * vrna_tree_string_unweight` (const char *structure)

Remove weights from a linear string tree representation of a secondary structure.
- `char * vrna_tree_string_to_db` (const char *tree)

Convert a linear tree string representation of a secondary structure back to Dot-Bracket notation.
- `void assign_plist_from_db` (vrna_ep_t **pl, const char *struc, float pr)

Create a vrna_ep_t from a dot-bracket string.
- `char * pack_structure` (const char *struc)

Pack secondary secondary structure, 5:1 compression using base 3 encoding.
- `char * unpack_structure` (const char *packed)

Unpack secondary structure previously packed with pack_structure()
- `short * make_pair_table` (const char *structure)

Create a pair table of a secondary structure.
- `short * copy_pair_table` (const short *pt)

Get an exact copy of a pair table.
- `short * alimake_pair_table` (const char *structure)
- `short * make_pair_table_snoop` (const char *structure)
- `int bp_distance` (const char *str1, const char *str2)

Compute the "base pair" distance between two secondary structures s1 and s2.
- `unsigned int * make_referenceBP_array` (short *reference_pt, unsigned int turn)

Make a reference base pair count matrix.
- `unsigned int * compute_BPdifferences` (short *pt1, short *pt2, unsigned int turn)

Make a reference base pair distance matrix.
- `void assign_plist_from_pr` (vrna_ep_t **pl, FLT_OR_DBL *probs, int length, double cutoff)

Create a vrna_ep_t from a probability matrix.
- `void parenthesis_structure` (char *structure, vrna_bp_stack_t *bp, int length)

Create a dot-bracket/parenthesis structure from backtracking stack.
- `void parenthesis_zuker` (char *structure, vrna_bp_stack_t *bp, int length)

Create a dot-bracket/parenthesis structure from backtracking stack obtained by Zuker suboptimal calculation in cofold.c.
- `void bppm_to_structure` (char *structure, FLT_OR_DBL *pr, unsigned int length)

Create a dot-bracket like structure string from base pair probability matrix.
- `char bppm_symbol` (const float *x)

Get a pseudo dot bracket notation for a given probability information.

18.217.1 Detailed Description

Various utility- and helper-functions for secondary structure parsing, converting, etc.

18.218 structures.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_STRUCT_UTILS_H
00002 #define VIENNA_RNA_PACKAGE_STRUCT_UTILS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00033 typedef struct vrna_hx_s vrna_hx_t;
00034
00035
00040 typedef struct vrna_elem_prob_s vrna_ep_t;
00041
00042
00073 #define VRNA_BRACKETS_ALPHA      4U
00074
00075
00081 #define VRNA_BRACKETS_RND        8U
00082
00083
00089 #define VRNA_BRACKETS_CLY        16U
00090
00091
00097 #define VRNA_BRACKETS_ANG        32U
00098
00099
00105 #define VRNA_BRACKETS_SQR        64U
00106
00107
00119 #define VRNA_BRACKETS_DEFAULT \
00120 (VRNA_BRACKETS_RND | \
00121  VRNA_BRACKETS_CLY | \
00122  VRNA_BRACKETS_ANG | \
00123  VRNA_BRACKETS_SQR)
00124
00125
00132 #define VRNA_BRACKETS_ANY \
00133 (VRNA_BRACKETS_RND | \
00134  VRNA_BRACKETS_CLY | \
00135  VRNA_BRACKETS_ANG | \
00136  VRNA_BRACKETS_SQR | \
00137  VRNA_BRACKETS_ALPHA)
00138
00139
00140 #include <stdio.h>
00141
00142 #include <ViennaRNA/datastructures/basic.h>
00143
00156 char *
00157 vrna_db_pack(const char *struc);
00158
00159
00170 char *
00171 vrna_db_unpack(const char *packed);
00172
00173
00189 void
00190 vrna_db_flatten(char          *structure,
00191                unsigned int  options);
00192
00193
00213 void
00214 vrna_db_flatten_to(char          *string,
00215                   const char  target[3],
00216                   unsigned int  options);
00217
00218
00236 char *
00237 vrna_db_from_ptable(const short *pt);
00238

```

```
00239
00249 char *
00250 vrna_db_from_plist(vrna_ep_t *pairs,
00251                   unsigned int n);
00252
00253
00260 char *
00261 vrna_db_to_element_string(const char *structure);
00262
00263
00295 char *
00296 vrna_db_pk_remove(const char *structure,
00297                   unsigned int options);
00298
00299 /* End dot-bracket interface */
00318 short *
00319 vrna_ptable(const char *structure);
00320
00321
00341 short *
00342 vrna_ptable_from_string(const char *structure,
00343                         unsigned int options);
00344
00345
00363 short *
00364 vrna_pt_pk_get(const char *structure);
00365
00366
00373 short *
00374 vrna_ptable_copy(const short *pt);
00375
00376
00381 short *
00382 vrna_pt.ali_get(const char *structure);
00383
00384
00392 short *
00393 vrna_pt.snoop_get(const char *structure);
00394
00395
00412 short *
00413 vrna_pt_pk_remove(const short *ptable,
00414                  unsigned int options);
00415
00416
00417 /* End pair table interface */
00429 #define VRNA_PLIST_TYPE_BASEPAIR 0
00430
00431
00435 #define VRNA_PLIST_TYPE_GQUAD 1
00436
00437
00441 #define VRNA_PLIST_TYPE_H_MOTIF 2
00442
00443
00447 #define VRNA_PLIST_TYPE_I_MOTIF 3
00448
00449
00453 #define VRNA_PLIST_TYPE_UD_MOTIF 4
00454
00455
00459 #define VRNA_PLIST_TYPE_STACK 5
00460
00461
00465 #define VRNA_PLIST_TYPE_UNPAIRED 6
00466
00467
00471 #define VRNA_PLIST_TYPE_TRIPLE 7
00472
00473
00482 struct vrna_elem_prob_s {
00483     int i;
00484     int j;
00485     float p;
00486     int type;
00487 };
00488
00504 vrna_ep_t *vrna_plist(const char *struc,
00505                      float pr);
00506
00507
00524 vrna_ep_t *vrna_plist_from_probs(vrna_fold_compound_t *vc,
00525                                  double cut_off);
00526
00527
00528 /* End pair list interface */
00596 char *
```

```
00597 vrna_db_from_WUSS(const char *wuss);
00598
00599
00600 /* End WUSS notation interface */
00648 char *
00649 vrna_abstract_shapes(const char      *structure,
00650                      unsigned int   level);
00651
00652
00668 char *
00669 vrna_abstract_shapes_pt(const short   *pt,
00670                        unsigned int level);
00671
00672
00673 /* End abstract shapes interface */
00685 struct vrna_hx_s {
00686     unsigned int   start;
00687     unsigned int   end;
00688     unsigned int   length;
00689     unsigned int   up5;
00690     unsigned int   up3;
00691 };
00692
00693
00700 vrna_hx_t *
00701 vrna_hx_from_ptable(short *pt);
00702
00703
00707 vrna_hx_t *
00708 vrna_hx_merge(const vrna_hx_t *list,
00709              int               maxdist);
00710
00711
00712 /* End helix list interface */
00719 int *
00720 vrna_loopidx_from_ptable(const short *pt);
00721
00722
00741 int
00742 vrna_bp_distance_pt(const short *pt1,
00743                    const short *pt2);
00744
00759 int
00760 vrna_bp_distance(const char *str1,
00761                  const char *str2);
00762
00763
00764 double
00765 vrna_dist_mountain(const char   *str1,
00766                   const char   *str2,
00767                   unsigned int p);
00768
00769
00770 /* End metrics interface */
00779 unsigned int *
00780 vrna_refBPcnt_matrix(const short *reference_pt,
00781                     unsigned int turn);
00782
00783
00791 unsigned int *
00792 vrna_refBPdist_matrix(const short *pt1,
00793                      const short *pt2,
00794                      unsigned int turn);
00795
00796
00800 char *
00801 vrna_db_from_probs(const FLT_OR_DBL *pr,
00802                   unsigned int   length);
00803
00804
00808 char
00809 vrna_bpp_symbol(const float *x);
00810
00811
00823 char *
00824 vrna_db_from_bp_stack(vrna_bp_stack_t *bp,
00825                      unsigned int   length);
00826
00827
00828 void
00829 vrna_letter_structure(char      *structure,
00830                      vrna_bp_stack_t *bp,
00831                      unsigned int   length);
00832
00833
00919 #define   VRNA_STRUCTURE_TREE_HIT           1U
00920
```

```

00921
00926 #define VRNA_STRUCTURE_TREE_SHAPIRO_SHORT 2U
00927
00928
00933 #define VRNA_STRUCTURE_TREE_SHAPIRO 3U
00934
00935
00940 #define VRNA_STRUCTURE_TREE_SHAPIRO_EXT 4U
00941
00942
00947 #define VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT 5U
00948
00953 #define VRNA_STRUCTURE_TREE_EXPANDED 6U
00954
00955
00985 char *
00986 vrna_db_to_tree_string(const char *structure,
00987 unsigned int type);
00988
00989
01001 char *
01002 vrna_tree_string_unweight(const char *structure);
01003
01004
01016 char *
01017 vrna_tree_string_to_db(const char *tree);
01018
01019
01020 /* End tree representations */
01023 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
01024
01025 /*#####*/
01026 /*# deprecated functions below */
01027 /*#####*/
01028
01048 DEPRECATED(void assign_plist_from_db(vrna_ep_t **pl,
01049 const char *struc,
01050 float pr),
01051 "Use vrna_plist() instead");
01052
01066 DEPRECATED(char *pack_structure(const char *struc),
01067 "Use vrna_db_pack() instead");
01068
01080 DEPRECATED(char *unpack_structure(const char *packed),
01081 "Use vrna_db_unpack() instead");
01082
01095 DEPRECATED(short *make_pair_table(const char *structure),
01096 "Use vrna_ptable() instead");
01097
01098 DEPRECATED(short *make_pair_table_pk(const char *structure),
01099 "Use vrna_ptable_from_string() instead");
01100
01110 DEPRECATED(short *copy_pair_table(const short *pt),
01111 "Use vrna_ptable_copy() instead");
01112
01119 DEPRECATED(short *alimake_pair_table(const char *structure),
01120 "Use vrna_pt_ali_get() instead");
01121
01129 DEPRECATED(short *make_pair_table_snoop(const char *structure),
01130 "Use vrna_pt_snoop_get() instead");
01131
01132 DEPRECATED(int *make_loop_index_pt(short *pt),
01133 "Use vrna_loopidx_from_ptable() instead");
01134
01148 DEPRECATED(int bp_distance(const char *str1,
01149 const char *str2),
01150 "Use vrna_bp_distance() instead");
01151
01161 DEPRECATED(unsigned int *make_referenceBP_array(short *reference_pt,
01162 unsigned int turn),
01163 "Use vrna_refBPcnt_matrix() instead");
01164
01174 DEPRECATED(unsigned int *compute_BPdifferences(short *pt1,
01175 short *pt2,
01176 unsigned int turn),
01177 "Use vrna_refBPdist_matrix() instead");
01178
01199 DEPRECATED(void assign_plist_from_pr(vrna_ep_t **pl,
01200 FLT_OR_DBL *probs,
01201 int length,
01202 double cutoff),
01203 "Use vrna_plist_from_probs() instead");
01204
01213 DEPRECATED(void parenthesis_structure(char *structure,
01214 vrna_bp_stack_t *bp,
01215 int length),
01216 "Use vrna_parenthesis_structure() instead");

```

```

01217
01227 DEPRECATED(void parenthesis_zuker(char          *structure,
01228                                vrna_bp_stack_t *bp,
01229                                int length),
01230                                "Use vrna_parenthesis_zuker() instead");
01231
01232 DEPRECATED(void letter_structure(char          *structure,
01233                                vrna_bp_stack_t *bp,
01234                                int length),
01235                                "Use vrna_letter_structure() instead");
01236
01242 DEPRECATED(void bppm_to_structure(char          *structure,
01243                                FLT_OR_DBL *pr,
01244                                unsigned int length),
01245                                "Use vrna_db_from_probs() instead");
01246
01252 DEPRECATED(char      bppm_symbol(const float *x),
01253                                "Use vrna_bpp_symbol() instead");
01254
01255 #endif
01256
01261 #endif

```

18.219 ProfileAln.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_PROFILEALN_H
00002 #define VIENNA_RNA_PACKAGE_PROFILEALN_H
00003
00004 float profile_aln(const float *T1,
00005                  const char *seq1,
00006                  const float *T2,
00007                  const char *seq2);
00008
00009
00010 int set_paln_params(double gap_open,
00011                    double gap_ext,
00012                    double seqweight,
00013                    int free_ends);
00014
00015
00016 #endif

```

18.220 ViennaRNA/profiledist.h File Reference

Include dependency graph for profiledist.h:

Functions

- float [profile_edit_distance](#) (const float *T1, const float *T2)
Align the 2 probability profiles T1, T2
- float * [Make_bp_profile_bppm](#) (FLT_OR_DBL *bppm, int length)
condense pair probability matrix into a vector containing probabilities for unpaired, upstream paired and downstream paired.
- void [print_bppm](#) (const float *T)
print string representation of probability profile
- void [free_profile](#) (float *T)
free space allocated in Make_bp_profile
- float * [Make_bp_profile](#) (int length)

18.220.1 Detailed Description

18.220.2 Function Documentation

18.220.2.1 profile_edit_distance()

```
float profile_edit_distance (
    const float * T1,
    const float * T2 )
```

Align the 2 probability profiles T1, T2

.

This is like a Needleman-Wunsch alignment, we should really use affine gap-costs ala Gotoh

18.220.2.2 Make_bp_profile_bppm()

```
float * Make_bp_profile_bppm (
    FLT_OR_DBL * bppm,
    int length )
```

condense pair probability matrix into a vector containing probabilities for unpaired, upstream paired and downstream paired.

This resulting probability profile is used as input for profile_edit_distance

Parameters

<i>bppm</i>	A pointer to the base pair probability matrix
<i>length</i>	The length of the sequence

Returns

The bp profile

18.220.2.3 free_profile()

```
void free_profile (
    float * T )
```

free space allocated in Make_bp_profile

Backward compatibility only. You can just use plain free()

18.220.2.4 Make_bp_profile()

```
float * Make_bp_profile (
    int length )
```

Note

This function is NOT threadsafe

See also

[Make_bp_profile_bppm\(\)](#)

Deprecated This function is deprecated and will be removed soon! See [Make_bp_profile_bppm\(\)](#) for a replacement

18.221 profiledist.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PROFILEDIST_H
00002 #define VIENNA_RNA_PACKAGE_PROFILEDIST_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00009 # else
```

```

00010 # define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00016 #include <ViennaRNA/datastructures/basic.h>
00017
00020 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00021
00028 float profile_edit_distance(const float *T1,
00029                             const float *T2);
00030
00031
00042 float *Make_bp_profile_bppm(FLT_OR_DBL *bppm,
00043                             int length);
00044
00045
00049 void print_bppm(const float *T);
00050
00051
00057 void free_profile(float *T);
00058
00059
00068 DEPRECATED(float *Make_bp_profile(int length),
00069 "Use Make_bp_profile_bppm() instead");
00070
00071 #endif
00072
00073 #endif

```

18.222 ViennaRNA/PS_dot.h File Reference

Use [ViennaRNA/plotting/probabilities.h](#) instead.

Include dependency graph for PS_dot.h:

18.222.1 Detailed Description

Use [ViennaRNA/plotting/probabilities.h](#) instead.

Deprecated Use [ViennaRNA/plotting/probabilities.h](#) instead

18.223 PS_dot.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_PROBABILITIES_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/PS_dot.h>! Use
00013         <ViennaRNA/plotting/probabilities.h> instead!"
00013 # endif
00014 #include <ViennaRNA/plotting/probabilities.h>
00015 #include <ViennaRNA/plotting/layouts.h>
00016 #include <ViennaRNA/plotting/structures.h>
00017 #endif
00018
00019 #endif

```

18.224 ViennaRNA/read_epars.h File Reference

Use [ViennaRNA/params/io.h](#) instead.

Include dependency graph for read_epars.h:

18.224.1 Detailed Description

Use [ViennaRNA/params/io.h](#) instead.

Deprecated Use [ViennaRNA/params/io.h](#) instead

18.225 read_epars.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_PARAMS_IO_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_PARAMS_IO_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/read_epars.h>! Use <ViennaRNA/params/io.h>
         instead!"
00013 # endif
00014 #include <ViennaRNA/params/io.h>
00015 #endif
00016
00017 #endif
```

18.226 ViennaRNA/ribo.h File Reference

Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments.

This graph shows which files directly or indirectly include this file:

Functions

- float ** [get_ribosum](#) (const char **Alseq, int n_seq, int length)
Retrieve a RiboSum Scoring Matrix for a given Alignment.
- float ** [readribosum](#) (char *name)
Read a RiboSum or other user-defined Scoring Matrix and Store into global Memory.

18.226.1 Detailed Description

Parse RiboSum Scoring Matrices for Covariance Scoring of Alignments.

18.227 ribo.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_RIBOSUM_H
00002 #define VIENNA_RNA_PACKAGE_RIBOSUM_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011
00022 float **get_ribosum(const char **Alseq,
00023                    int          n_seq,
00024                    int          length);
00025
00026
00031 float **readribosum(char *name);
00032
00033
00037 #endif
00038
00039 #endif
```

18.228 ViennaRNA/RNAstruct.h File Reference

Parsing and Coarse Graining of Structures.

Functions

- char * [b2HIT](#) (const char *structure)
Converts the full structure from bracket notation to the HIT notation including root.
- char * [b2C](#) (const char *structure)
Converts the full structure from bracket notation to the a coarse grained notation using the 'H' 'B' 'I' 'M' and 'R' identifiers.
- char * [b2Shapiro](#) (const char *structure)

Converts the full structure from bracket notation to the weighted coarse grained notation using the 'H' 'B' 'I' 'M' 'S' 'E' and 'R' identifiers.

- char * [add_root](#) (const char *structure)
Adds a root to an un-rooted tree in any except bracket notation.
- char * [expand_Shapiro](#) (const char *coarse)
Inserts missing 'S' identifiers in unweighted coarse grained structures as obtained from [b2C\(\)](#).
- char * [expand_Full](#) (const char *structure)
Convert the full structure from bracket notation to the expanded notation including root.
- char * [unexpand_Full](#) (const char *ffull)
Restores the bracket notation from an expanded full or HIT tree, that is any tree using only identifiers 'U' 'P' and 'R'.
- char * [unweight](#) (const char *wcoarse)
Strip weights from any weighted tree.
- void [unexpand_aligned_F](#) (char *align[2])
Converts two aligned structures in expanded notation.
- void [parse_structure](#) (const char *structure)
Collects a statistic of structure elements of the full structure in bracket notation.

Variables

- int **loop_size** [STRUC]
contains a list of all loop sizes. loop_size[0] contains the number of external bases.
- int **helix_size** [STRUC]
contains a list of all stack sizes.
- int **loop_degree** [STRUC]
contains the corresponding list of loop degrees.
- int **loops**
contains the number of loops (and therefore of stacks).
- int **unpaired**
contains the number of unpaired bases.
- int **pairs**
contains the number of base pairs in the last parsed structure.

18.228.1 Detailed Description

Parsing and Coarse Graining of Structures.

Example:

```
*  .(((.....))..((.....)).  is the bracket or full tree
*  becomes expanded:  - expand_Full() -
*  ((U)((U)(U)((U)(U)(U)P)P)(U)(U)((U)(U)P)P)P)(U)R)
*  HIT:  - b2HIT() -
*  ((U1)((U2)((U3)P3)(U2)((U2)P2)P2)(U1)R)
*  Coarse:  - b2C() -
*  ((H)((H)M)R)
*  becomes expanded:  - expand_Shapiro() -
*  (((((H)S)((H)S)M)S)R)
*  weighted Shapiro:  - b2Shapiro() -
*  (((((H3)S3)((H2)S2)M4)S2)E2)R)
*
```

18.229 RNAstruct.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_RNASTRUCT_H
00002 #define VIENNA_RNA_PACKAGE_RNASTRUCT_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
```

```

00006 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00009 # else
00010 # define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00040 #define STRUC      2000
00041
00052 DEPRECATED(char *b2HIT(const char *structure),
00053             "Use vrna_db_to_tree_string() instead!");           /* Full   -> HIT      [incl. root] */
00054
00055
00066 DEPRECATED(char *b2C(const char *structure),
00067             "Use vrna_db_to_tree_string() instead!");           /* Full   -> Coarse [incl. root] */
00068
00069
00081 DEPRECATED(char *b2Shapiro(const char *structure),
00082             "Use vrna_db_to_tree_string() instead!");           /* Full -> weighted Shapiro [i.r.] */
00083
00084
00091 DEPRECATED(char *add_root(const char *structure),
00092             "");                                                 /* {Tree} -> ({Tree}R) */
00093
00094
00102 DEPRECATED(char *expand_Shapiro(const char *coarse),
00103             "Use vrna_db_to_tree_string() instead!");
00104
00105
00106 /* add S for stacks to coarse struct */
00114 DEPRECATED(char *expand_Full(const char *structure),
00115             "Use vrna_db_to_tree_string() instead!");           /* Full   -> FFull      */
00116
00117
00125 DEPRECATED(char *unexpand_Full(const char *ffull),
00126             "Use vrna_tree_string_to_db() instead!");           /* FFull  -> Full      */
00127
00128
00135 DEPRECATED(char *unweight(const char *wcoarse),
00136             "Use vrna_tree_string_unweight() instead!");        /* remove weights from coarse
struct */
00137
00138
00148 DEPRECATED(void      unexpand_aligned_F(char *align[2]),
00149             "");
00150
00151
00161 DEPRECATED(void      parse_structure(const char *structure),
00162             ""); /* make structure statistics */
00163
00164
00169 DEPRECATED(extern int loop_size[STRUC],
00170             ""); /* loop sizes of a structure */
00171
00175 DEPRECATED(extern int helix_size[STRUC],
00176             ""); /* helix sizes of a structure */
00177
00181 DEPRECATED(extern int loop_degree[STRUC],
00182             ""); /* loop degrees of a structure */
00183
00187 DEPRECATED(extern int loops,
00188             ""); /* n of loops and stacks */
00189
00193 DEPRECATED(extern int unpaired,
00194             "");
00195
00199 DEPRECATED(extern int pairs,
00200             ""); /* n of unpaired digits and pairs */
00201
00206 #endif

```

18.230 ViennaRNA/search/BoyerMoore.h File Reference

Variants of the Boyer-Moore string search algorithm.

Functions

- `const unsigned int * vrna_search_BMH_num` (const unsigned int *needle, size_t needle_size, const unsigned int *haystack, size_t haystack_size, size_t start, size_t *badchars, unsigned char cyclic)

Search for a string of elements in a larger string of elements using the Boyer-Moore-Horspool algorithm.

- `const char * vrna_search_BMH` (`const char *needle`, `size_t needle_size`, `const char *haystack`, `size_t haystack_size`, `size_t start`, `size_t *badchars`, `unsigned char cyclic`)

Search for an ASCII pattern within a larger ASCII string using the Boyer-Moore-Horspool algorithm.

- `size_t * vrna_search_BM_BCT_num` (`const unsigned int *pattern`, `size_t pattern_size`, `unsigned int num_max`)

Retrieve a Boyer-Moore Bad Character Table for a pattern of elements represented by natural numbers.

- `size_t * vrna_search_BM_BCT` (`const char *pattern`)

Retrieve a Boyer-Moore Bad Character Table for a NULL-terminated pattern of ASCII characters.

18.230.1 Detailed Description

Variants of the Boyer-Moore string search algorithm.

,

18.231 BoyerMoore.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_SEARCH_BOYER_MOORE_H
00002 #define VIENNA_RNA_PACKAGE_SEARCH_BOYER_MOORE_H
00003
00036 const unsigned int *
00037 vrna_search_BMH_num(const unsigned int *needle,
00038                    size_t needle_size,
00039                    const unsigned int *haystack,
00040                    size_t haystack_size,
00041                    size_t start,
00042                    size_t *badchars,
00043                    unsigned char cyclic);
00044
00045
00067 const char *
00068 vrna_search_BMH(const char *needle,
00069                size_t needle_size,
00070                const char *haystack,
00071                size_t haystack_size,
00072                size_t start,
00073                size_t *badchars,
00074                unsigned char cyclic);
00075
00076
00092 size_t *
00093 vrna_search_BM_BCT_num(const unsigned int *pattern,
00094                       size_t pattern_size,
00095                       unsigned int num_max);
00096
00097
00110 size_t *
00111 vrna_search_BM_BCT(const char *pattern);
00112
00113
00117 #endif
```

18.232 ViennaRNA/sequence.h File Reference

Functions and data structures related to sequence representations ,.

Include dependency graph for sequence.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct `vrna_sequence_s`
Data structure representing a nucleotide sequence. [More...](#)
- struct `vrna_alignment_s`

Typedefs

- typedef struct [vrna_sequence_s](#) [vrna_seq_t](#)

Typename for nucleotide sequence representation data structure [vrna_sequence_s](#).

Enumerations

- enum [vrna_seq_type_e](#) { [VRNA_SEQ_UNKNOWN](#) , [VRNA_SEQ_RNA](#) , [VRNA_SEQ_DNA](#) }

A enumerator used in [vrna_sequence_s](#) to distinguish different nucleotide sequences.

18.232.1 Detailed Description

Functions and data structures related to sequence representations ,.

18.233 sequence.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_SEQUENCE_H
00002 #define VIENNA_RNA_PACKAGE_SEQUENCE_H
00003
00017 typedef struct vrna\_sequence\_s vrna\_seq\_t;
00018
00019 typedef struct vrna\_alignment\_s vrna\_msa\_t;
00020
00021 #include <ViennaRNA/fold\_compound.h>
00022
00023
00024 #define VRNA_SEQUENCE_RNA      1U
00025
00026 #define VRNA_SEQUENCE_DNA      2U
00027
00031 typedef enum {
00032     VRNA\_SEQ\_UNKNOWN,
00033     VRNA\_SEQ\_RNA,
00034     VRNA\_SEQ\_DNA
00035 } vrna\_seq\_type\_e;
00036
00037
00041 struct vrna\_sequence\_s {
00042     vrna\_seq\_type\_e type;
00043     char            *name;
00044     char            *string;
00045     short           *encoding;
00046     short           *encoding5;
00047     short           *encoding3;
00048     unsigned int    length;
00049 };
00050
00051
00052 struct vrna\_alignment\_s {
00053     unsigned int    n_seq;
00054     vrna\_seq\_t      *sequences;
00055     char            **gapfree_seq;
00056     unsigned int    *gapfree_size; /* for MAF alignment coordinates */
00057     unsigned long long *genome_size; /* for MAF alignment coordinates */
00058     unsigned long long *start; /* for MAF alignment coordinates */
00059     unsigned char      *orientation; /* for MAF alignment coordinates */
00060     unsigned int      **a2s;
00061 };
00062
00063
00064 vrna\_seq\_t *
00065 vrna\_sequence(const char      *string,
00066               unsigned int    options);
00067
00068
00069 int
00070 vrna\_sequence\_add(vrna\_fold\_compound\_t *fc,
00071                  const char      *string,
00072                  unsigned int    options);
00073
00074
00075 int
00076 vrna\_sequence\_remove(vrna\_fold\_compound\_t *fc,
00077                     unsigned int    i);
00078
00079
00080 void

```

```

00081 vrna_sequence_remove_all(vrna_fold_compound_t *fc);
00082
00083
00084 void
00085 vrna_sequence_prepare(vrna_fold_compound_t *fc);
00086
00087
00088 int
00089 vrna_sequence_order_update(vrna_fold_compound_t *fc,
00090                           const unsigned int *order);
00091
00092
00093 int
00094 vrna_msa_add( vrna_fold_compound_t *fc,
00095              const char **alignment,
00096              const char **names,
00097              const unsigned char *orientation,
00098              const unsigned long long *start,
00099              const unsigned long long *genome_size,
00100              unsigned int options);
00101
00102
00107 #endif

```

18.234 snofold.h

```

00001 /* function from fold.c */
00002 #ifndef VIENNA_RNA_PACKAGE_SNOFOLD_H
00003 #define VIENNA_RNA_PACKAGE_SNOFOLD_H
00004
00005 #include <ViennaRNA/datastructures/basic.h>
00006
00007 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00008
00009 /* Normal fold */
00010
00014 int snofold(const char *sequence,
00015            char *structure,
00016            const int max_assym,
00017            const int threshold,
00018            const int min_s2,
00019            const int max_s2,
00020            const int half_stem,
00021            const int max_half_stem);
00022
00023
00028 void snofree_arrays(const int length); /* free arrays for mfe folding */
00029
00030
00031 void snoinitialize_fold(int length); /* allocate arrays for folding */
00032
00033
00034 void snoupdate_fold_params(void); /* recalculate parameters */
00035
00036
00037 int snoloop_energy(short *ptable,
00038                  short *s,
00039                  short *sl,
00040                  int i);
00041
00042
00043 void snoexport_fold_arrays(int **indx_p,
00044                          int **mLoop_p,
00045                          int **cLoop_p,
00046                          folden ***fold_p,
00047                          folden ***fold_p_XS);
00048
00049
00050 char *snobacktrack_fold_from_pair(const char *sequence,
00051                                  int i,
00052                                  int j);
00053
00054
00055 /* alifold */
00056 float alisnofold(const char **strings,
00057                 const int max_assym,
00058                 const int threshloop,
00059                 const int min_s2,
00060                 const int max_s2,
00061                 const int half_stem,
00062                 const int max_half_stem);
00063
00064
00065 void alisnofree_arrays(const int length);
00066

```



```

00067
00068 char *alishnbacktrack_fold_from_pair(const char **sequence,
00069                                     int i,
00070                                     int j,
00071                                     int *cov);
00072
00073
00074 extern double cv_fact /* =1 */;
00075 extern double nc_fact /* =1 */;
00076
00077 /* max number of mismatch >>>..(( ))>>> */
00078 #define MISMATCH 3
00079
00080 #endif
00081
00082 #endif

```

18.235 snoop.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_SNOOP_H
00002 #define VIENNA_RNA_PACKAGE_SNOOP_H
00003
00004 #include <ViennaRNA/datastructures/basic.h>
00005
00006 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00007
00012 snoopT snoopfold(const char *s1,
00013                 const char *s2,
00014                 const int penalty,
00015                 const int threshloop,
00016                 const int threshLE,
00017                 const int threshRE,
00018                 const int threshDE,
00019                 const int threshD,
00020                 const int half_stem,
00021                 const int max_half_stem,
00022                 const int min_s2,
00023                 const int max_s2,
00024                 const int min_s1,
00025                 const int max_s1,
00026                 const int min_d1,
00027                 const int min_d2,
00028                 const int fullStemEnergy);
00029
00030
00036 snoopT *snoop_subopt(const char *s1,
00037                     const char *s2,
00038                     int delta,
00039                     int w,
00040                     const int penalty,
00041                     const int threshloop,
00042                     const int threshLE,
00043                     const int threshRE,
00044                     const int threshDE,
00045                     const int threshTE,
00046                     const int threshSE,
00047                     const int threshD,
00048                     const int distance,
00049                     const int half_stem,
00050                     const int max_half_stem,
00051                     const int min_s2,
00052                     const int max_s2,
00053                     const int min_s1,
00054                     const int max_s1,
00055                     const int min_d1,
00056                     const int min_d2,
00057                     const int fullStemEnergy);
00058
00059
00065 void lsnoop_subopt(const char *s1,
00066                  const char *s2,
00067                  int delta,
00068                  int w,
00069                  const int penalty,
00070                  const int threshloop,
00071                  const int threshLE,
00072                  const int threshRE,
00073                  const int threshDE,
00074                  const int threshTE,
00075                  const int threshSE,
00076                  const int threshD,
00077                  const int distance,
00078                  const int half_stem,
00079                  const int max_half_stem,
00080                  const int min_s2,

```

```
00081         const int max_s2,
00082         const int min_s1,
00083         const int max_s1,
00084         const int min_d1,
00085         const int min_d2,
00086         const int alignment_length,
00087         const char *name,
00088         const int fullStemEnergy);
00089
00090
00096 void Lsnoop_subopt_list(const char *s1,
00097                         const char *s2,
00098                         int delta,
00099                         int w,
00100                         const int penalty,
00101                         const int threshloop,
00102                         const int threshLE,
00103                         const int threshRE,
00104                         const int threshDE,
00105                         const int threshTE,
00106                         const int threshSE,
00107                         const int threshD,
00108                         const int distance,
00109                         const int half_stem,
00110                         const int max_half_stem,
00111                         const int min_s2,
00112                         const int max_s2,
00113                         const int min_s1,
00114                         const int max_s1,
00115                         const int min_d1,
00116                         const int min_d2,
00117                         const int alignment_length,
00118                         const char *name,
00119                         const int fullStemEnergy);
00120
00121
00127 void Lsnoop_subopt_list_XS(const char *s1,
00128                           const char *s2,
00129                           const int **access_s1,
00130                           int delta,
00131                           int w,
00132                           const int penalty,
00133                           const int threshloop,
00134                           const int threshLE,
00135                           const int threshRE,
00136                           const int threshDE,
00137                           const int threshTE,
00138                           const int threshSE,
00139                           const int threshD,
00140                           const int distance,
00141                           const int half_stem,
00142                           const int max_half_stem,
00143                           const int min_s2,
00144                           const int max_s2,
00145                           const int min_s1,
00146                           const int max_s1,
00147                           const int min_d1,
00148                           const int min_d2,
00149                           const int alignment_length,
00150                           const char *name,
00151                           const int fullStemEnergy);
00152
00153
00159 void snoop_subopt_XS(const char *s1,
00160                     const char *s2,
00161                     const int **access_s1,
00162                     int delta,
00163                     int w,
00164                     const int penalty,
00165                     const int threshloop,
00166                     const int threshLE,
00167                     const int threshRE,
00168                     const int threshDE,
00169                     const int threshTE,
00170                     const int threshSE,
00171                     const int threshD,
00172                     const int distance,
00173                     const int half_stem,
00174                     const int max_half_stem,
00175                     const int min_s2,
00176                     const int max_s2,
00177                     const int min_s1,
00178                     const int max_s1,
00179                     const int min_d1,
00180                     const int min_d2,
00181                     const int alignment_length,
00182                     const char *name,
```

```
00183         const int    fullStemEnergy);
00184
00185
00190 snoopT *alisnoop_subopt(const char  **s1,
00191         const char  **s2,
00192         int          delta,
00193         int          w,
00194         const int    penalty,
00195         const int    threshloop,
00196         const int    threshLE,
00197         const int    threshRE,
00198         const int    threshDE,
00199         const int    threshTE,
00200         const int    threshSE,
00201         const int    threshD,
00202         const int    distance,
00203         const int    half_stem,
00204         const int    max_half_stem,
00205         const int    min_s2,
00206         const int    max_s2,
00207         const int    min_s1,
00208         const int    max_s1,
00209         const int    min_d1,
00210         const int    min_d2);
00211
00212
00218 snoopT *aliIsnoop_subopt_list(const char  **s1,
00219         const char  **s2,
00220         int          delta,
00221         int          w,
00222         const int    penalty,
00223         const int    threshloop,
00224         const int    threshLE,
00225         const int    threshRE,
00226         const int    threshDE,
00227         const int    threshTE,
00228         const int    threshSE,
00229         const int    threshD,
00230         const int    distance,
00231         const int    half_stem,
00232         const int    max_half_stem,
00233         const int    min_s2,
00234         const int    max_s2,
00235         const int    min_s1,
00236         const int    max_s1,
00237         const int    min_d1,
00238         const int    min_d2,
00239         const int    alignment_length);
00240
00241
00247 snoopT alisnoopfold(const char  **s1,
00248         const char  **s2,
00249         const int    penalty,
00250         const int    threshloop,
00251         const int    threshLE,
00252         const int    threshRE,
00253         const int    threshDE,
00254         const int    threshD,
00255         const int    half_stem,
00256         const int    max_half_stem,
00257         const int    min_s2,
00258         const int    max_s2,
00259         const int    min_s1,
00260         const int    max_s1,
00261         const int    min_d1,
00262         const int    min_d2);
00263
00264
00269 snoopT snoopfold_XS(const char  *s1,
00270         const char  *s2,
00271         const int    **access_s1,
00272         const int    pos,
00273         const int    max_pos_j,
00274         const int    penalty,
00275         const int    threshloop,
00276         const int    threshLE,
00277         const int    threshRE,
00278         const int    threshDE,
00279         const int    threshD,
00280         const int    half_stem,
00281         const int    max_half_stem,
00282         const int    min_s2,
00283         const int    max_s2,
00284         const int    min_s1,
00285         const int    max_s1,
00286         const int    min_d1,
00287         const int    min_d2,
```

```

00288             const int    fullStemEnergy);
00289
00290
00291 extern int snoop_subopt_sorted;
00292 #endif
00293
00294 #endif

```

18.236 special_const.h

```

00001 extern const char    wall;
00002 extern const char    bg;
00003 extern const char    star;
00004 extern const char    probe;
00005 extern const char    start[];
00006 extern const char    end[];
00007 extern const char    success[];
00008 extern const char    injector[];
00009 extern unsigned int  injector_len;
00010 extern const char    flash[];
00011 extern const char    head1l[];
00012 extern const char    head2l[];
00013 extern const char    lvlstr[];
00014 extern const char    inv[];

```

18.237 ViennaRNA/datastructures/stream_output.h File Reference

An implementation of a buffered, ordered stream output data structure.

This graph shows which files directly or indirectly include this file:

Typedefs

- typedef struct vrna_ordered_stream_s * **vrna_ostream_t**
An ordered output stream structure with unordered insert capabilities.
- typedef void(* **vrna_stream_output_f**) (void *auxdata, unsigned int i, void *data)
Ordered stream processing callback.

Functions

- **vrna_ostream_t** **vrna_ostream_init** (**vrna_stream_output_f** output, void *auxdata)
Get an initialized ordered output stream.
- void **vrna_ostream_free** (**vrna_ostream_t** dat)
Free an initialized ordered output stream.
- void **vrna_ostream_request** (**vrna_ostream_t** dat, unsigned int num)
Request index in ordered output stream.
- void **vrna_ostream_provide** (**vrna_ostream_t** dat, unsigned int i, void *data)
Provide output stream data for a particular index.

18.237.1 Detailed Description

An implementation of a buffered, ordered stream output data structure.

,

18.238 stream_output.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_STREAM_OUTPUT_H
00002 #define VIENNA_RNA_PACKAGE_STREAM_OUTPUT_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)

```

```

00009 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00010 # elif defined(__GNUC__)
00011 # define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00012 # else
00013 # define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
00018
00033 typedef struct vrna_ordered_stream_s *vrna_ostream_t;
00034
00049 typedef void (*vrna_stream_output_f)(void *auxdata,
00050 unsigned int i,
00051 void *data);
00052 DEPRECATED(typedef void (vrna_callback_stream_output)(void *auxdata,
00053 unsigned int i,
00054 void *data),
00055 "Use vrna_stream_output_f instead!");
00056
00057
00067 vrna_ostream_t
00068 vrna_ostream_init(vrna_stream_output_f output,
00069 void *auxdata);
00070
00071
00079 void
00080 vrna_ostream_free(vrna_ostream_t dat);
00081
00082
00083 int
00084 vrna_ostream_threadsafe(void);
00085
00086
00099 void
00100 vrna_ostream_request(vrna_ostream_t dat,
00101 unsigned int num);
00102
00103
00116 void
00117 vrna_ostream_provide(vrna_ostream_t dat,
00118 unsigned int i,
00119 void *data);
00120
00121
00127 #endif

```

18.239 ViennaRNA/stream_output.h File Reference

Use [ViennaRNA/datastructures/stream_output.h](#) instead.

Include dependency graph for stream_output.h:

18.239.1 Detailed Description

Use [ViennaRNA/datastructures/stream_output.h](#) instead.

Deprecated Use [ViennaRNA/datastructures/stream_output.h](#) instead

18.240 stream_output.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_DATA_STRUCTURES_STREAM_OUTPUT_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_DATA_STRUCTURES_STREAM_OUTPUT_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/stream_output.h>! Use
00013 <ViennaRNA/datastructures/stream_output.h> instead!"
00013 # endif
00014 #include <ViennaRNA/datastructures/stream_output.h>
00015 #endif
00016
00017 #endif

```

18.241 ViennaRNA/string_utils.h File Reference

Use [ViennaRNA/utls/strings.h](#) instead.

Include dependency graph for string_utils.h:

18.241.1 Detailed Description

Use [ViennaRNA/utls/strings.h](#) instead.

Deprecated Use [ViennaRNA/utls/strings.h](#) instead

18.242 string_utils.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_STRING_UTILS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_STRING_UTILS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/string_utils.h>! Use <ViennaRNA/utls/strings.h>
         instead!"
00013 #   endif
00014 #include <ViennaRNA/utls/strings.h>
00015 #endif
00016
00017 #endif
```

18.243 ViennaRNA/stringdist.h File Reference

Functions for String Alignment.

Include dependency graph for stringdist.h:

Functions

- [swString](#) * [Make_swString](#) (char *string)
Convert a structure into a format suitable for [string_edit_distance\(\)](#).
- float [string_edit_distance](#) (swString *T1, swString *T2)
Calculate the string edit distance of T1 and T2.

18.243.1 Detailed Description

Functions for String Alignment.

18.243.2 Function Documentation

18.243.2.1 Make_swString()

```
swString * Make_swString (
    char * string )
```

Convert a structure into a format suitable for [string_edit_distance\(\)](#).

Parameters

string	
------------------------	--

Returns

18.243.2.2 string_edit_distance()

```
float string_edit_distance (
    swString * T1,
    swString * T2 )
```

Calculate the string edit distance of T1 and T2.

Parameters

<i>T1</i>	
<i>T2</i>	

Returns

18.244 stringdist.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_STRING_DIST_H
00002 #define VIENNA_RNA_PACKAGE_STRING_DIST_H
00003
00009 #include <ViennaRNA/dist_vars.h>
00010
00011
00018 swString *Make_swString(char *string);
00019
00027 float      string_edit_distance( swString *T1,
00028                                swString *T2);
00029
00030 #endif
```

18.245 ViennaRNA/structure_utils.h File Reference

Use [ViennaRNA/Utils/structures.h](#) instead.

Include dependency graph for structure_utils.h:

18.245.1 Detailed Description

Use [ViennaRNA/Utils/structures.h](#) instead.

Deprecated Use [ViennaRNA/Utils/structures.h](#) instead

18.246 structure_utils.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_STRUCTURE_UTILS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_STRUCTURE_UTILS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/structure_utils.h>! Use
    <ViennaRNA/Utils/structures.h> instead!"
00013 #   endif
00014 #include <ViennaRNA/Utils/structures.h>
00015 #endif
00016
00017 #endif
```

18.247 ViennaRNA/structured_domains.h File Reference

This module provides interfaces that deal with additional structured domains in the folding grammar. This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_structured_domains_s](#)

18.247.1 Detailed Description

This module provides interfaces that deal with additional structured domains in the folding grammar.

18.248 structured_domains.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_STRUCTURAL_DOMAINS_H
00002 #define VIENNA_RNA_PACKAGE_STRUCTURAL_DOMAINS_H
00003
00023 typedef struct vrna_structured_domains_s vrna_sd_t;
00024
00025
00026 struct vrna_structured_domains_s {
00027     char __placeholder; /* dummy placeholder to not leave this struct empty */
00028 };
00029
00030 #endif
```

18.249 ViennaRNA/subopt.h File Reference

RNAsubopt and density of states declarations.

Include dependency graph for subopt.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_subopt_sol_s](#)
Solution element from subopt.c.

Macros

- #define **MAXDOS** 1000
Maximum density of states discretization for subopt.

Typedefs

- typedef struct [vrna_subopt_sol_s](#) [vrna_subopt_solution_t](#)
Typename for the subopt solution list representing data structure [vrna_subopt_sol_s](#).
- typedef void(* [vrna_subopt_result_f](#)) (const char *structure, float energy, void *data)
Callback for [vrna_subopt_cb\(\)](#)
- typedef struct [vrna_subopt_sol_s](#) SOLUTION
Backward compatibility typedef for [vrna_subopt_sol_s](#).

Functions

- [vrna_subopt_solution_t](#) * [vrna_subopt](#) ([vrna_fold_compound_t](#) *fc, int delta, int sorted, FILE *fp)
Returns list of subopt structures or writes to fp.
- void [vrna_subopt_cb](#) ([vrna_fold_compound_t](#) *fc, int delta, [vrna_subopt_result_f](#) cb, void *data)
Generate suboptimal structures within an energy band around the MFE.

- **SOLUTION** * **subopt** (char *seq, char *structure, int delta, FILE *fp)
Returns list of subopt structures or writes to fp.
- **SOLUTION** * **subopt_par** (char *seq, char *structure, **vrna_param_t** *parameters, int delta, int is_↔ constrained, int is_circular, FILE *fp)
Returns list of subopt structures or writes to fp.
- **SOLUTION** * **subopt_circ** (char *seq, char *sequence, int delta, FILE *fp)
Returns list of circular subopt structures or writes to fp.
- **SOLUTION** * **zukersubopt** (const char *string)
Compute Zuker type suboptimal structures.
- **SOLUTION** * **zukersubopt_par** (const char *string, **vrna_param_t** *parameters)
Compute Zuker type suboptimal structures.

Variables

- double **print_energy**
printing threshold for use with logML
- int **subopt_sorted**
Sort output by energy.
- int **density_of_states** [MAXDOS+1]
The Density of States.

18.249.1 Detailed Description

RNAsubopt and density of states declarations.

18.249.2 Typedef Documentation

18.249.2.1 SOLUTION

typedef struct **vrna_subopt_sol_s** **SOLUTION**
Backward compatibility typedef for **vrna_subopt_sol_s**.

Deprecated Use **vrna_subopt_solution_t** instead!

18.250 subopt.h

[Go to the documentation of this file.](#)

```
00001 /* subopt.h */
00002 #ifndef VIENNA_RNA_PACKAGE_SUBOPT_H
00003 #define VIENNA_RNA_PACKAGE_SUBOPT_H
00004
00005 #ifdef VRNA_WARN_DEPRECATED
00006 # if defined(__clang__)
00007 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00008 # elif defined(__GNUC__)
00009 #  define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00010 # else
00011 #  define DEPRECATED(func, msg) func
00012 # endif
00013 #else
00014 # define DEPRECATED(func, msg) func
00015 #endif
00016
00031 typedef struct vrna_subopt_sol_s vrna_subopt_solution_t;
00032
00048 typedef void (*vrna_subopt_result_f)(const char *structure,
00049                                     float energy,
00050                                     void *data);
00051
00052 DEPRECATED(typedef void (vrna_subopt_callback)(const char *structure,
00053                                               float energy,
00054                                               void *data),
```

```

00055         "Use vrna_subopt_result_f instead!");
00056
00057 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00058
00063 typedef struct vrna_subopt_sol_s SOLUTION;
00064
00065 #endif
00066
00067 #include <stdio.h>
00068
00069 #include <ViennaRNA/datastructures/basic.h>
00070 #include <ViennaRNA/fold_compound.h>
00071 #include <ViennaRNA/params/basic.h>
00072
00073
00077 struct vrna_subopt_sol_s {
00078     float energy;
00079     char *structure;
00080 };
00081
00085 #define MAXDOS          1000
00086
00094 #define VRNA_UNSORTED          0
00095 #define VRNA_SORT_BY_ENERGY_LEXICOGRAPHIC_ASC  1
00096 #define VRNA_SORT_BY_ENERGY_ASC                2
00097
00128 vrna_subopt_solution_t *
00129 vrna_subopt(vrna_fold_compound_t *fc,
00130             int delta,
00131             int sorted,
00132             FILE *fp);
00133
00134
00169 void
00170 vrna_subopt_cb(vrna_fold_compound_t *fc,
00171               int delta,
00172               vrna_subopt_result_f cb,
00173               void *data);
00174
00175
00182 extern double print_energy;
00183
00190 extern int subopt_sorted;
00191
00208 extern int density_of_states[MAXDOS + 1];
00209 /* End of group dos */
00211
00212 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00213
00231 DEPRECATED(SOLUTION * subopt(char *seq, char *structure, int delta, FILE * fp),
00232             "Use vrna_subopt() or vrna_subopt_cb() instead");
00233
00239 DEPRECATED(SOLUTION *
00240             subopt_par(char *seq, char *structure, vrna_param_t * parameters, int delta,
00241                       int is_constrained,
00242                       int is_circular, FILE * fp),
00243             "Use vrna_subopt() or vrna_subopt_cb() instead");
00244
00259 DEPRECATED(SOLUTION * subopt_circ(char *seq, char *sequence, int delta, FILE * fp),
00260             "Use vrna_subopt() or vrna_subopt_cb() instead");
00261
00276 DEPRECATED(SOLUTION * zukersubopt(const char *string),
00277             "Use vrna_subopt_zuker() instead");
00278
00287 DEPRECATED(SOLUTION * zukersubopt_par(const char *string, vrna_param_t * parameters),
00288             "Use vrna_subopt_zuker() instead");
00289
00290
00291 #endif
00292
00293 #endif

```

18.251 subopt_zuker.h

```

00001 /* subopt_zuker.h */
00002 #ifndef VIENNA_RNA_PACKAGE_SUBOPT_ZUKER_H
00003 #define VIENNA_RNA_PACKAGE_SUBOPT_ZUKER_H
00004
00005 #include <ViennaRNA/fold_compound.h>
00006 #include <ViennaRNA/subopt.h>
00007
00032 vrna_subopt_solution_t *
00033 vrna_subopt_zuker(vrna_fold_compound_t *fc);
00034
00035

```

```
00036 #endif
```

18.252 ViennaRNA/svm_utils.h File Reference

Use [ViennaRNA/utis/svm.h](#) instead.

Include dependency graph for svm_utils.h:

18.252.1 Detailed Description

Use [ViennaRNA/utis/svm.h](#) instead.

Deprecated Use [ViennaRNA/utis/svm.h](#) instead

18.253 svm_utils.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_UTILS_SVM_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_UTILS_SVM_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/svm_utils.h>! Use <ViennaRNA/utis/svm.h>
         instead!"
00013 #   endif
00014 #include <ViennaRNA/utis/svm.h>
00015 #endif
00016
00017 #endif
```

18.254 ViennaRNA/treedist.h File Reference

Functions for [Tree](#) Edit Distances.

Include dependency graph for treedist.h:

Functions

- [Tree](#) * [make_tree](#) (char *struc)
Constructs a [Tree](#) (essentially the postorder list) of the structure 'struc', for use in [tree_edit_distance\(\)](#).
- float [tree_edit_distance](#) ([Tree](#) *T1, [Tree](#) *T2)
Calculates the edit distance of the two trees.
- void [print_tree](#) ([Tree](#) *t)
Print a tree (mainly for debugging)
- void [free_tree](#) ([Tree](#) *t)
Free the memory allocated for [Tree](#) t.

18.254.1 Detailed Description

Functions for [Tree](#) Edit Distances.

18.254.2 Function Documentation

18.254.2.1 make_tree()

```
Tree * make_tree (
    char * struc )
```

Constructs a [Tree](#) (essentially the postorder list) of the structure 'struc', for use in [tree_edit_distance\(\)](#).

Parameters

<i>struc</i>	may be any rooted structure representation.
--------------	---

Returns

18.254.2.2 tree_edit_distance()

```
float tree_edit_distance (
    Tree * T1,
    Tree * T2 )
```

Calculates the edit distance of the two trees.

Parameters

<i>T1</i>	
<i>T2</i>	

Returns

18.254.2.3 free_tree()

```
void free_tree (
    Tree * t )
```

Free the memory allocated for [Tree](#) t.

Parameters

<i>t</i>	
----------	--

18.255 treedist.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_TREE_DIST_H
00002 #define VIENNA_RNA_PACKAGE_TREE_DIST_H
00003
00009 #include <ViennaRNA/dist_vars.h>
00010
00018 Tree *make_tree(char *struc);
00019
00020
00028 float tree_edit_distance(Tree *T1,
00029                          Tree *T2);
00030
00031
00035 void print_tree(Tree *t);
00036
00037
00043 void free_tree(Tree *t);
00044
00045
00046 #endif
```

18.256 ViennaRNA/units.h File Reference

Use [ViennaRNA/units/units.h](#) instead.

Include dependency graph for units.h:

18.256.1 Detailed Description

Use [ViennaRNA/units/units.h](#) instead.

Deprecated Use [ViennaRNA/units/units.h](#) instead

18.257 units.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_UNITS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_UNITS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 #   ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/units.h>! Use <ViennaRNA/units/units.h>
         instead!"
00013 #   endif
00014 #include <ViennaRNA/units/units.h>
00015 #endif
00016
00017 #endif
```

18.258 ViennaRNA/units/units.h File Reference

Physical Units and Functions to convert them into each other.

This graph shows which files directly or indirectly include this file:

Enumerations

- enum [vrna_unit_energy_e](#) {
[VRNA_UNIT_J](#), [VRNA_UNIT_KJ](#), [VRNA_UNIT_CAL_IT](#), [VRNA_UNIT_DACAL_IT](#),
[VRNA_UNIT_KCAL_IT](#), [VRNA_UNIT_CAL](#), [VRNA_UNIT_DACAL](#), [VRNA_UNIT_KCAL](#),
[VRNA_UNIT_G_TNT](#), [VRNA_UNIT_KG_TNT](#), [VRNA_UNIT_T_TNT](#), [VRNA_UNIT_EV](#),
[VRNA_UNIT_WH](#), [VRNA_UNIT_KWH](#) }
Energy / Work Units.
- enum [vrna_unit_temperature_e](#) {
[VRNA_UNIT_K](#), [VRNA_UNIT_DEG_C](#), [VRNA_UNIT_DEG_F](#), [VRNA_UNIT_DEG_R](#),
[VRNA_UNIT_DEG_N](#), [VRNA_UNIT_DEG_DE](#), [VRNA_UNIT_DEG_RE](#), [VRNA_UNIT_DEG_RO](#) }
Temperature Units.

Functions

- double [vrna_convert_energy](#) (double energy, [vrna_unit_energy_e](#) from, [vrna_unit_energy_e](#) to)
Convert between energy / work units.
- double [vrna_convert_temperature](#) (double temp, [vrna_unit_temperature_e](#) from, [vrna_unit_temperature_e](#) to)
Convert between temperature units.
- int [vrna_convert_kcal_to_dcal](#) (double energy)
Convert floating point energy value into integer representation.
- double [vrna_convert_dcal_to_kcal](#) (int energy)
Convert an integer representation of free energy in deka-cal/mol to kcal/mol.

18.258.1 Detailed Description

Physical Units and Functions to convert them into each other.

,

18.259 units.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_UNITS_H
00002 #define VIENNA_RNA_PACKAGE_UNITS_H
00003
00021 typedef enum {
00022     VRNA_UNIT_J,
00023     VRNA_UNIT_KJ,
00024     VRNA_UNIT_CAL_IT,
00025     VRNA_UNIT_DACAL_IT,
00026     VRNA_UNIT_KCAL_IT,
00027     VRNA_UNIT_CAL,
00028     VRNA_UNIT_DACAL,
00029     VRNA_UNIT_KCAL,
00030     VRNA_UNIT_G_TNT,
00031     VRNA_UNIT_KG_TNT,
00032     VRNA_UNIT_T_TNT,
00033     VRNA_UNIT_EV,
00034     VRNA_UNIT_WH,
00035     VRNA_UNIT_KWH,
00036 } vrna_unit_energy_e;
00037
00038
00044 typedef enum {
00045     VRNA_UNIT_K,
00046     VRNA_UNIT_DEG_C,
00047     VRNA_UNIT_DEG_F,
00048     VRNA_UNIT_DEG_R,
00049     VRNA_UNIT_DEG_N,
00050     VRNA_UNIT_DEG_DE,
00051     VRNA_UNIT_DEG_RE,
00052     VRNA_UNIT_DEG_RO,
00053 } vrna_unit_temperature_e;
00054
00055
00065 double
00066 vrna_convert_energy(double          energy,
00067                    vrna_unit_energy_e from,
00068                    vrna_unit_energy_e to);
00069
00070
00080 double
00081 vrna_convert_temperature(double      temp,
00082                        vrna_unit_temperature_e from,
00083                        vrna_unit_temperature_e to);
00084
00085
00097 int
00098 vrna_convert_kcal_to_dcal(double energy);
00099
00100
00111 double
00112 vrna_convert_dcal_to_kcal(int energy);
00113
00114
00119 #endif
```

18.260 ViennaRNA/unstructured_domains.h File Reference

Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches.

Include dependency graph for unstructured_domains.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [vrna_unstructured_domain_s](#)
Data structure to store all functionality for ligand binding. [More...](#)
- struct [vrna_unstructured_domain_motif_s](#)

Macros

- `#define VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP 1U`
Flag to indicate ligand bound to unpaired stretch in the exterior loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP 2U`
Flag to indicate ligand bound to unpaired stretch in a hairpin loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP 4U`
Flag to indicate ligand bound to unpaired stretch in an interior loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP 8U`
Flag to indicate ligand bound to unpaired stretch in a multibranch loop.
- `#define VRNA_UNSTRUCTURED_DOMAIN_MOTIF 16U`
Flag to indicate ligand binding without additional unbound nucleotides (motif-only)
- `#define VRNA_UNSTRUCTURED_DOMAIN_ALL_LOOPS`
Flag to indicate ligand bound to unpaired stretch in any loop (convenience macro)

Typedefs

- `typedef struct vrna_unstructured_domain_s vrna_ud_t`
Typename for the ligand binding extension data structure `vrna_unstructured_domain_s`.
- `typedef int(* vrna_ud_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`
Callback to retrieve binding free energy of a ligand bound to an unpaired sequence segment.
- `typedef FLT_OR_DBL(* vrna_ud_exp_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, void *data)`
Callback to retrieve Boltzmann factor of the binding free energy of a ligand bound to an unpaired sequence segment.
- `typedef void(* vrna_ud_production_f) (vrna_fold_compound_t *vc, void *data)`
Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature.
- `typedef void(* vrna_ud_exp_production_f) (vrna_fold_compound_t *vc, void *data)`
Callback for pre-processing the production rule of the ligand binding to unpaired stretches feature (partition function variant)
- `typedef void(* vrna_ud_add_probs_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, FLT_OR_DBL exp_energy, void *data)`
Callback to store/add equilibrium probability for a ligand bound to an unpaired sequence segment.
- `typedef FLT_OR_DBL(* vrna_ud_get_probs_f) (vrna_fold_compound_t *vc, int i, int j, unsigned int loop_type, int motif, void *data)`
Callback to retrieve equilibrium probability for a ligand bound to an unpaired sequence segment.

Functions

- `vrna_ud_motif_t * vrna_ud_motifs_centroid (vrna_fold_compound_t *fc, const char *structure)`
Detect unstructured domains in centroid structure.
- `vrna_ud_motif_t * vrna_ud_motifs_MEA (vrna_fold_compound_t *fc, const char *structure, vrna_ep_t *probability_list)`
Detect unstructured domains in MEA structure.
- `vrna_ud_motif_t * vrna_ud_motifs_MFE (vrna_fold_compound_t *fc, const char *structure)`
Detect unstructured domains in MFE structure.
- `void vrna_ud_add_motif (vrna_fold_compound_t *vc, const char *motif, double motif_en, const char *motif↵_name, unsigned int loop_type)`
Add an unstructured domain motif, e.g. for ligand binding.
- `int * vrna_ud_get_motif_size_at (vrna_fold_compound_t *vc, int i, unsigned int loop_type)`
Get a list of unique motif sizes that start at a certain position within the sequence.
- `void vrna_ud_remove (vrna_fold_compound_t *vc)`
Remove ligand binding to unpaired stretches.
- `void vrna_ud_set_data (vrna_fold_compound_t *vc, void *data, vrna_auxdata_free_f free_cb)`

Attach an auxiliary data structure.

- void `vrna_ud_set_prod_rule_cb` (`vrna_fold_compound_t` *vc, `vrna_ud_production_f` pre_cb, `vrna_ud_f_e_cb`)

Attach production rule callbacks for free energies computations.

- void `vrna_ud_set_exp_prod_rule_cb` (`vrna_fold_compound_t` *vc, `vrna_ud_exp_production_f` pre_cb, `vrna_ud_exp_f_exp_e_cb`)

Attach production rule for partition function.

- void `vrna_ud_set_prob_cb` (`vrna_fold_compound_t` *vc, `vrna_ud_add_probs_f` setter, `vrna_ud_get_probs_f` getter)

18.260.1 Detailed Description

Functions to modify unstructured domains, e.g. to incorporate ligands binding to unpaired stretches.

18.260.2 Function Documentation

18.260.2.1 `vrna_ud_get_motif_size_at()`

```
int * vrna_ud_get_motif_size_at (
    vrna_fold_compound_t * vc,
    int i,
    unsigned int loop_type )
```

Get a list of unique motif sizes that start at a certain position within the sequence.

18.260.2.2 `vrna_ud_set_prob_cb()`

```
void vrna_ud_set_prob_cb (
    vrna_fold_compound_t * vc,
    vrna_ud_add_probs_f setter,
    vrna_ud_get_probs_f getter )
```

SWIG Wrapper Notes This function is attached as method `ud_set_prob_cb()` to objects of type `fold_compound`

18.261 `unstructured_domains.h`

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_UNSTRUCTURED_DOMAIN_H
00002 #define VIENNA_RNA_PACKAGE_UNSTRUCTURED_DOMAIN_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(DEPRECATED)
00006 #   undef DEPRECATED
00007 # endif
00008 # if defined(__clang__)
00009 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00010 # elif defined(__GNUC__)
00011 #   define DEPRECATED(func, msg) func __attribute__((deprecated(msg)))
00012 # else
00013 #   define DEPRECATED(func, msg) func
00014 # endif
00015 #else
00016 # define DEPRECATED(func, msg) func
00017 #endif
00018
00084 typedef struct vrna_unstructured_domain_s vrna_ud_t;
00085
00086 typedef struct vrna_unstructured_domain_motif_s vrna_ud_motif_t;
00087
00088 #include <ViennaRNA/datastructures/basic.h>
00089 #include <ViennaRNA/fold_compound.h>
00090 #include <ViennaRNA/utils/structures.h>
00091
00110 typedef int (*vrna_ud_f) (vrna_fold_compound_t *vc,
00111                           int i,
00112                           int j,
```



```

00113             unsigned int      loop_type,
00114             void              *data);
00115
00116 DEPRECATED(typedef int (vrna_callback_ud_energy) (vrna_fold_compound_t *vc,
00117             int              i,
00118             int              j,
00119             unsigned int      loop_type,
00120             void              *data),
00121             "Use vrna_ud_f instead!");
00122
00141 typedef FLT_OR_DBL (*vrna_ud_exp_f) (vrna_fold_compound_t *vc,
00142             int              i,
00143             int              j,
00144             unsigned int      loop_type,
00145             void              *data);
00146
00147 DEPRECATED(typedef FLT_OR_DBL (vrna_callback_ud_exp_energy) (vrna_fold_compound_t *vc,
00148             int              i,
00149             int              j,
00150             unsigned int      loop_type,
00151             void              *data),
00152             "Use vrna_ud_exp_f instead!");
00153
00164 typedef void (*vrna_ud_production_f) (vrna_fold_compound_t *vc,
00165             void              *data);
00166
00167 DEPRECATED(typedef void (vrna_callback_ud_production) (vrna_fold_compound_t *vc,
00168             void              *data),
00169             "Use vrna_ud_production_f instead!");
00170
00181 typedef void (*vrna_ud_exp_production_f) (vrna_fold_compound_t *vc,
00182             void              *data);
00183
00184 DEPRECATED(typedef void (vrna_callback_ud_exp_production) (vrna_fold_compound_t *vc,
00185             void              *data),
00186             "Use vrna_ud_exp_production_f instead!");
00187
00188
00198 typedef void (*vrna_ud_add_probs_f) (vrna_fold_compound_t *vc,
00199             int              i,
00200             int              j,
00201             unsigned int      loop_type,
00202             FLT_OR_DBL        exp_energy,
00203             void              *data);
00204
00205 DEPRECATED(typedef void (vrna_callback_ud_probs_add) (vrna_fold_compound_t *vc,
00206             int              i,
00207             int              j,
00208             unsigned int      loop_type,
00209             FLT_OR_DBL        exp_energy,
00210             void              *data),
00211             "Use vrna_ud_add_probs_f instead!");
00212
00222 typedef FLT_OR_DBL (*vrna_ud_get_probs_f) (vrna_fold_compound_t *vc,
00223             int              i,
00224             int              j,
00225             unsigned int      loop_type,
00226             int              motif,
00227             void              *data);
00228
00229 DEPRECATED(typedef FLT_OR_DBL (vrna_callback_ud_probs_get) (vrna_fold_compound_t *vc,
00230             int              i,
00231             int              j,
00232             unsigned int      loop_type,
00233             int              motif,
00234             void              *data),
00235             "Use vrna_ud_get_probs_f instead!");
00236
00237
00242 #define VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP    1U
00243
00248 #define VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP     2U
00249
00254 #define VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP    4U
00255
00260 #define VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP     8U
00261
00266 #define VRNA_UNSTRUCTURED_DOMAIN_MOTIF      16U
00267
00272 #define VRNA_UNSTRUCTURED_DOMAIN_ALL_LOOPS  (VRNA_UNSTRUCTURED_DOMAIN_EXT_LOOP | \
00273             VRNA_UNSTRUCTURED_DOMAIN_HP_LOOP | \
00274             VRNA_UNSTRUCTURED_DOMAIN_INT_LOOP | \
00275             VRNA_UNSTRUCTURED_DOMAIN_MB_LOOP)
00276
00281 struct vrna_unstructured_domain_s {
00282     /*
00283     *****

```

```
00284  * Keep track of all motifs added
00285  ****
00286  */
00287  int      uniq_motif_count;
00288  unsigned int  *uniq_motif_size;
00290  int      motif_count;
00291  char     **motif;
00292  char     **motif_name;
00293  unsigned int  *motif_size;
00294  double    *motif_en;
00295  unsigned int  *motif_type;
00297  /*
00298  ****
00299  * Grammar extension for ligand
00300  * binding
00301  ****
00302  */
00303  vrna_ud_production_f  prod_cb;
00307  vrna_ud_exp_production_f  exp_prod_cb;
00308  vrna_ud_f  energy_cb;
00309  vrna_ud_exp_f  exp_energy_cb;
00310  void      *data;
00311  vrna_auxdata_free_f  free_data;
00312  vrna_ud_add_probs_f  probs_add;
00313  vrna_ud_get_probs_f  probs_get;
00314  };
00315
00316
00317  struct vrna_unstructured_domain_motif_s {
00318      int start;
00319      int number;
00320  };
00321
00322
00342  vrna_ud_motif_t *
00343  vrna_ud_motifs_centroid(vrna_fold_compound_t *vc,
00344                          const char *structure);
00345
00346
00367  vrna_ud_motif_t *
00368  vrna_ud_motifs_MEA(vrna_fold_compound_t *vc,
00369                     const char *structure,
00370                     vrna_ep_t *probability_list);
00371
00372
00391  vrna_ud_motif_t *
00392  vrna_ud_motifs_MFE(vrna_fold_compound_t *vc,
00393                     const char *structure);
00394
00395
00421  void  vrna_ud_add_motif(vrna_fold_compound_t *vc,
00422                          const char *motif,
00423                          double motif_en,
00424                          const char *motif_name,
00425                          unsigned int loop_type);
00426
00427
00432  int *vrna_ud_get_motif_size_at(vrna_fold_compound_t *vc,
00433                                 int i,
00434                                 unsigned int loop_type);
00435
00436
00437  int *
00438  vrna_ud_get_motifs_at(vrna_fold_compound_t *vc,
00439                        int i,
00440                        unsigned int loop_type);
00441
00442
00443  vrna_ud_motif_t *
00444  vrna_ud_detect_motifs(vrna_fold_compound_t *vc,
00445                        const char *structure);
00446
00447
00458  void  vrna_ud_remove(vrna_fold_compound_t *vc);
00459
00460
00477  void  vrna_ud_set_data(vrna_fold_compound_t *vc,
00478                          void *data,
00479                          vrna_auxdata_free_f free_cb);
00480
00481
00516  void  vrna_ud_set_prod_rule_cb(vrna_fold_compound_t *vc,
00517                                vrna_ud_production_f pre_cb,
00518                                vrna_ud_f e_cb);
00519
00520
00545  void  vrna_ud_set_exp_prod_rule_cb(vrna_fold_compound_t *vc,
```

```

00546             vrna_ud_exp_production_f pre_cb,
00547             vrna_ud_exp_f exp_e_cb);
00548
00549
00550 void vrna_ud_set_prob_cb(vrna_fold_compound_t *vc,
00551             vrna_ud_add_probs_f setter,
00552             vrna_ud_get_probs_f getter);
00553
00554
00555 #endif

```

18.262 ViennaRNA/io/utils.h File Reference

Several utilities for file handling.

Include dependency graph for utils.h: This graph shows which files directly or indirectly include this file:

Functions

- void **vrna_file_copy** (FILE *from, FILE *to)
Inefficient 'cp'.
- char * **vrna_read_line** (FILE *fp)
Read a line of arbitrary length from a stream.
- int **vrna_mkdir_p** (const char *path)
Recursively create a directory tree.
- char * **vrna_basename** (const char *path)
Extract the filename from a file path.
- char * **vrna_dirname** (const char *path)
Extract the directory part of a file path.
- char * **vrna_filename_sanitize** (const char *name, const char *replacement)
Sanitize a file name.
- int **vrna_file_exists** (const char *filename)
Check if a file already exists in the file system.

18.262.1 Detailed Description

Several utilities for file handling.

,

18.263 utils.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_FILE_UTILS_H
00002 #define VIENNA_RNA_PACKAGE_FILE_UTILS_H
00003
00010 #include <stdio.h>
00011
00021 void vrna_file_copy(FILE *from,
00022             FILE *to);
00023
00024
00035 char *vrna_read_line(FILE *fp);
00036
00037
00041 int vrna_mkdir_p(const char *path);
00042
00043
00047 char *vrna_basename(const char *path);
00048
00049
00053 char *vrna_dirname(const char *path);
00054
00055
00097 char *vrna_filename_sanitize(const char *name,
00098             const char *replacement);
00099
00100

```

```

00107 int
00108 vrna_file_exists(const char *filename);
00109
00110
00115 #endif

```

18.264 ViennaRNA/plotting/utls.h File Reference

Various utilities to assist in plotting secondary structures and consensus structures.

Include dependency graph for utls.h: This graph shows which files directly or indirectly include this file:

Functions

- char ** [vrna_annotate_covar_db](#) (const char **alignment, const char *structure, [vrna_md_t](#) *md_p)
Produce covariance annotation for an alignment given a secondary structure.
- [vrna_cpair_t](#) * [vrna_annotate_covar_pairs](#) (const char **alignment, [vrna_ep_t](#) *pl, [vrna_ep_t](#) *mfel, double threshold, [vrna_md_t](#) *md)
Produce covariance annotation for an alignment given a set of base pairs.

18.264.1 Detailed Description

Various utilities to assist in plotting secondary structures and consensus structures.

,

18.265 utls.h

[Go to the documentation of this file.](#)

```

00001 #ifndef VIENNA_RNA_PACKAGE_PLOT_UTILS_H
00002 #define VIENNA_RNA_PACKAGE_PLOT_UTILS_H
00003
00010 #include <ViennaRNA/datastructures/basic.h>
00011 #include <ViennaRNA/model.h>
00012 #include <ViennaRNA/utls/structures.h>
00013
00032 char **
00033 vrna_annotate_covar_db(const char **alignment,
00034                      const char *structure,
00035                      vrna_md_t *md_p);
00036
00037
00038 char **
00039 vrna_annotate_covar_db_extended(const char **alignment,
00040                               const char *structure,
00041                               vrna_md_t *md_p,
00042                               unsigned int options);
00043
00044
00049 vrna_cpair_t *
00050 vrna_annotate_covar_pairs(const char **alignment,
00051                          vrna_ep_t *pl,
00052                          vrna_ep_t *mfel,
00053                          double threshold,
00054                          vrna_md_t *md);
00055
00056
00061 #endif

```

18.266 ViennaRNA/utls.h File Reference

Use [ViennaRNA/utls/basic.h](#) instead.

Include dependency graph for utls.h:

18.266.1 Detailed Description

Use [ViennaRNA/utls/basic.h](#) instead.

Deprecated Use [ViennaRNA/utls/basic.h](#) instead

Deprecated Use [ViennaRNA/utils/basic.h](#) instead

18.267 utils.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_UTILS_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_UTILS_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/utils.h>! Use <ViennaRNA/utils/basic.h>
      instead!"
00013 # endif
00014 #include <ViennaRNA/utils/basic.h>
00015 #include <ViennaRNA/utils/strings.h>
00016 #include <ViennaRNA/utils/structures.h>
00017 #include <ViennaRNA/io/utils.h>
00018 #include <ViennaRNA/alphabet.h>
00019 #endif
00020
00021 #endif
```

18.268 cpu.h

```
00001 #ifndef VIENNA_RNA_PACKAGE_UTILS_CPU_H
00002 #define VIENNA_RNA_PACKAGE_UTILS_CPU_H
00003
00004 #define VRNA_CPU_SIMD_NONE      0U
00005 #define VRNA_CPU_SIMD_SSE2     1U
00006 #define VRNA_CPU_SIMD_SSE3     2U
00007 #define VRNA_CPU_SIMD_SSE41    4U
00008 #define VRNA_CPU_SIMD_SSE42    8U
00009 #define VRNA_CPU_SIMD_AVX      16U
00010 #define VRNA_CPU_SIMD_AVX2     32U
00011 #define VRNA_CPU_SIMD_AVX512F  64U
00012
00013
00014 char *
00015 vrna_cpu_vendor_string(void);
00016
00017
00018 unsigned int
00019 vrna_cpu_simd_capabilities(void);
00020
00021
00022 #endif
```

18.269 higher_order_functions.h

```
00001 #ifndef VIENNA_RNA_PACKAGE_UTILS_FUN_H
00002 #define VIENNA_RNA_PACKAGE_UTILS_FUN_H
00003
00004 void
00005 vrna_fun_dispatch_disable(void);
00006
00007
00008 void
00009 vrna_fun_dispatch_enable(void);
00010
00011
00012 int
00013 vrna_fun_zip_add_min(const int *e1,
00014                     const int *e2,
00015                     int count);
00016
00017
00018 #endif
```

18.270 ViennaRNA/utils/strings.h File Reference

General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.

Include dependency graph for strings.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define XSTR(s) STR(s)`
Stringify a macro after expansion.
- `#define STR(s) #s`
Stringify a macro argument.
- `#define FILENAME_MAX_LENGTH 80`
Maximum length of filenames that are generated by our programs.
- `#define FILENAME_ID_LENGTH 42`
Maximum length of id taken from fasta header for filename generation.
- `#define VRNA_TRIM_LEADING 1U`
Trim only characters leading the string.
- `#define VRNA_TRIM_TRAILING 2U`
Trim only characters trailing the string.
- `#define VRNA_TRIM_IN_BETWEEN 4U`
Trim only characters within the string.
- `#define VRNA_TRIM_SUBST_BY_FIRST 8U`
Replace remaining characters after trimming with the first delimiter in list.
- `#define VRNA_TRIM_DEFAULT (VRNA_TRIM_LEADING | VRNA_TRIM_TRAILING)`
Default settings for trimming, i.e. trim leading and trailing.
- `#define VRNA_TRIM_ALL (VRNA_TRIM_DEFAULT | VRNA_TRIM_IN_BETWEEN)`
Trim characters anywhere in the string.

Functions

- `char * vrna_strdup_printf (const char *format,...)`
Safely create a formatted string.
- `char * vrna_strdup_vprintf (const char *format, va_list argp)`
Safely create a formatted string.
- `int vrna_strcat_printf (char **dest, const char *format,...)`
Safely append a formatted string to another string.
- `int vrna_strcat_vprintf (char **dest, const char *format, va_list args)`
Safely append a formatted string to another string.
- `unsigned int vrna_strtrim (char *string, const char *delimiters, unsigned int keep, unsigned int options)`
Trim a string by removing (multiple) occurrences of a particular character.
- `char ** vrna_strsplit (const char *string, const char *delimiter)`
Split a string into tokens using a delimiting character.
- `char * vrna_random_string (int l, const char symbols[])`
Create a random string using characters from a specified symbol set.
- `int vrna_hamming_distance (const char *s1, const char *s2)`
Calculate hamming distance between two sequences.
- `int vrna_hamming_distance_bound (const char *s1, const char *s2, int n)`
Calculate hamming distance between two sequences up to a specified length.
- `void vrna_seq_toRNA (char *sequence)`
Convert an input sequence (possibly containing DNA alphabet characters) to RNA alphabet.
- `void vrna_seq_toupper (char *sequence)`
Convert an input sequence to uppercase.
- `void vrna_seq_reverse (char *sequence)`
Reverse a string in-place.
- `char * vrna_DNA_complement (const char *sequence)`
Retrieve a DNA sequence which resembles the complement of the input sequence.
- `char * vrna_seq_ungapped (const char *sequence)`

- Remove gap characters from a nucleotide sequence.*
- char * [vrna_cut_point_insert](#) (const char *string, int cp)
Add a separating '&' character into a string according to cut-point position.
- char * [vrna_cut_point_remove](#) (const char *string, int *cp)
Remove a separating '&' character from a string.
- void [str_uppercase](#) (char *sequence)
Convert an input sequence to uppercase.
- void [str_DNA2RNA](#) (char *sequence)
Convert a DNA input sequence to RNA alphabet.
- char * [random_string](#) (int l, const char symbols[])
Create a random string using characters from a specified symbol set.
- int [hamming](#) (const char *s1, const char *s2)
Calculate hamming distance between two sequences.
- int [hamming_bound](#) (const char *s1, const char *s2, int n)
Calculate hamming distance between two sequences up to a specified length.

18.270.1 Detailed Description

General utility- and helper-functions for RNA sequence and structure strings used throughout the ViennaRNA Package.

18.270.2 Function Documentation

18.270.2.1 str_uppercase()

```
void str_uppercase (
    char * sequence )
```

Convert an input sequence to uppercase.

Deprecated Use [vrna_seq_toupper\(\)](#) instead!

18.270.2.2 str_DNA2RNA()

```
void str_DNA2RNA (
    char * sequence )
```

Convert a DNA input sequence to RNA alphabet.

Deprecated Use [vrna_seq_toRNA\(\)](#) instead!

18.270.2.3 random_string()

```
char * random_string (
    int l,
    const char symbols[] )
```

Create a random string using characters from a specified symbol set.

Deprecated Use [vrna_random_string\(\)](#) instead!

18.270.2.4 hamming()

```
int hamming (
    const char * s1,
    const char * s2 )
```

Calculate hamming distance between two sequences.

Deprecated Use [vrna_hamming_distance\(\)](#) instead!

18.270.2.5 hamming_bound()

```
int hamming_bound (
    const char * s1,
    const char * s2,
    int n )
```

Calculate hamming distance between two sequences up to a specified length.

Deprecated Use [vrna_hamming_distance_bound\(\)](#) instead!

18.271 strings.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_STRING_UTILS_H
00002 #define VIENNA_RNA_PACKAGE_STRING_UTILS_H
00003
00004 #ifdef VRNA_WARN_DEPRECATED
00005 # if defined(__clang__)
00006 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00007 # elif defined(__GNUC__)
00008 #  define DEPRECATED(func, msg) func __attribute__ ((deprecated(msg)))
00009 # else
00010 #  define DEPRECATED(func, msg) func
00011 # endif
00012 #else
00013 # define DEPRECATED(func, msg) func
00014 #endif
00015
00028 #include <stdarg.h>
00029 #include <ViennaRNA/datastructures/basic.h>
00030
00034 #define XSTR(s) STR(s)
00035
00039 #define STR(s) #s
00040
00041 #ifndef FILENAME_MAX_LENGTH
00042
00049 #define FILENAME_MAX_LENGTH 80
00050
00057 #define FILENAME_ID_LENGTH 42
00058
00059 #endif
00060
00061 #ifdef HAVE_CONFIG_H
00062 #include <config.h>
00063 #ifndef HAVE_STRDUP
00064 char *
00065 strdup(const char *s);
00066
00067
00068 #endif
00069 #endif
00070
00089 char *
00090 vrna_strdup_printf(const char *format,
00091 ...);
00092
00093
00108 char *
00109 vrna_strdup_vprintf(const char *format,
00110 va_list argp);
00111
00112
00131 int
00132 vrna_strcat_printf(char **dest,
00133 const char *format,
```



```
00134         ...);
00135
00136
00149 int
00150 vrna_strcat_vprintf(char      **dest,
00151                    const char *format,
00152                    va_list    args);
00153
00154
00159 #define VRNA_TRIM_LEADING      1U
00160
00165 #define VRNA_TRIM_TRAILING    2U
00166
00171 #define VRNA_TRIM_IN_BETWEEN  4U
00172
00177 #define VRNA_TRIM_SUBST_BY_FIRST 8U
00178
00183 #define VRNA_TRIM_DEFAULT      ( VRNA_TRIM_LEADING | VRNA_TRIM_TRAILING )
00184
00189 #define VRNA_TRIM_ALL          ( VRNA_TRIM_DEFAULT | VRNA_TRIM_IN_BETWEEN )
00190
00237 unsigned int
00238 vrna_strtrim(char      *string,
00239              const char *delimiters,
00240              unsigned int keep,
00241              unsigned int options);
00242
00243
00290 char **
00291 vrna_strsplit(const char *string,
00292              const char *delimiter);
00293
00294
00295 char *
00296 vrna_strjoin(const char **strings,
00297             const char *delimiter);
00298
00299
00307 char *
00308 vrna_random_string(int      l,
00309                   const char symbols[]);
00310
00311
00319 int
00320 vrna_hamming_distance(const char *s1,
00321                     const char *s2);
00322
00323
00334 int
00335 vrna_hamming_distance_bound(const char *s1,
00336                           const char *s2,
00337                           int      n);
00338
00339
00347 void
00348 vrna_seq_toRNA(char *sequence);
00349
00350
00356 void
00357 vrna_seq_toupper(char *sequence);
00358
00359
00374 void
00375 vrna_seq_reverse(char *sequence);
00376
00377
00396 char *
00397 vrna_DNA_complement(const char *sequence);
00398
00399
00406 char *
00407 vrna_seq_ungapped(const char *sequence);
00408
00409
00421 char *
00422 vrna_cut_point_insert(const char *string,
00423                     int      cp);
00424
00425
00438 char *
00439 vrna_cut_point_remove(const char *string,
00440                     int      *cp);
00441
00442
00447 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00448
00453 DEPRECATED(void
```

```

00454         str_uppercase(char *sequence),
00455         "Use vrna_seq_toupper() instead");
00456
00462 DEPRECATED(void
00463         str_DNA2RNA(char *sequence),
00464         "Use vrna_seq_toRNA() instead");
00465
00471 DEPRECATED(char *random_string(int l,
00472                                 const char symbols[]),
00473         "Use vrna_random_string() instead");
00474
00480 DEPRECATED(int
00481         hamming(const char *s1,
00482                 const char *s2),
00483         "Use vrna_hamming_distance() instead");
00484
00490 DEPRECATED(int
00491         hamming_bound(const char *s1,
00492                       const char *s2,
00493                       int n),
00494         "Use vrna_hamming_distance_bound() instead");
00495
00496 #endif
00497
00498 #endif

```

18.272 svm.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_UTILS_SVM_H
00002 #define VIENNA_RNA_PACKAGE_UTILS_SVM_H
00003
00004 extern char *avg_model_string;
00005 extern char *sd_model_string;
00006
00007 float      get_z(char *sequence,
00008                 double energy);
00009 double     avg_regression (int N,
00010                           int A,
00011                           int C,
00012                           int G,
00013                           int T,
00014                           struct svm_model *avg_model,
00015                           int *info );
00016 double     sd_regression  (int N,
00017                           int A,
00018                           int C,
00019                           int G,
00020                           int T,
00021                           struct svm_model *sd_model);
00022 double     minimal_sd    (int N,
00023                           int A,
00024                           int C,
00025                           int G,
00026                           int T);
00027 struct svm_model *svm_load_model_string(char *modelString);
00028 int         *get_seq_composition( short *S,
00029                                   unsigned int start,
00030                                   unsigned int stop,
00031                                   unsigned int length);
00032
00033 #endif

```

18.273 vrna_config.h

```

00001 #ifndef VIENNA_RNA_PACKAGE_CONFIG_H
00002 #define VIENNA_RNA_PACKAGE_CONFIG_H
00003
00004 /* version number */
00005 #define VRNA_VERSION  "2.6.0b"
00006
00007 #define VRNA_VERSION_MAJOR  2
00008 #define VRNA_VERSION_MINOR  6
00009 #define VRNA_VERSION_PATCH  0b
00010
00011 /*
00012  * The following pre-processor definitions specify whether
00013  * or not certain features were activated upon build-time
00014  */
00015
00016 /*
00017  * Build with deactivated C11 Features
00018  */

```

```

00019  * It this feature is missing, the next line defines
00020  * 'VRNA_DISABLE_C11_FEATURES'
00021  */
00022
00023
00024 /*
00025  * Build with OpenMP support
00026  *
00027  * If this feature is present, the next line defines
00028  * 'VRNA_WITH_OPENMP'
00029  */
00030 #define VRNA_WITH_OPENMP
00031
00032 /*
00033  * Build with single precision partition function
00034  *
00035  * If this feature is present, the next line defines
00036  * 'USE_FLOAT_PF'
00037  */
00038
00039
00040 /*
00041  * Build with JSON input/output support
00042  *
00043  * If this feature is present, the next line defines
00044  * 'VRNA_WITH_JSON_SUPPORT'
00045  */
00046 #define VRNA_WITH_JSON_SUPPORT
00047
00048 /*
00049  * Build with Support Vector Machine (SVM) Z-score feature in RNALfold
00050  *
00051  * If this feature is present, the next line defines
00052  * 'VRNA_WITH_SVM'
00053  */
00054 #define VRNA_WITH_SVM
00055
00056 /*
00057  * Build with GSL minimizers
00058  *
00059  * If this feature is present, the next line defines
00060  * 'VRNA_WITH_GSL'
00061  */
00062 #define VRNA_WITH_GSL
00063
00064 /*
00065  * Build with colored TTY output
00066  *
00067  * If this feature is missing, the next line defines
00068  * 'VRNA_WITHOUT_TTY_COLORS'
00069  */
00070
00071
00072 /*
00073  * Build with Link Time Optimization support
00074  *
00075  * If this feature is enabled, the next line defines
00076  * 'VRNA_WITH_LTO'
00077  */
00078 #define VRNA_WITH_LTO
00079
00080 /*
00081  * Build with Naview Layout algorithm of Brucoleri 1988
00082  *
00083  * If this feature is enabled, the next line defines
00084  * 'VRNA_WITH_NAVIEW_LAYOUT'
00085  */
00086 #define VRNA_WITH_NAVIEW_LAYOUT
00087
00088
00089
00090 #endif

```

18.274 ViennaRNA/landscape/walk.h File Reference

Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence.

Include dependency graph for walk.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define VRNA_PATH_STEEPEST_DESCENT 128`
Option flag to request a steepest descent / gradient path.
- `#define VRNA_PATH_RANDOM 256`
Option flag to request a random walk path.
- `#define VRNA_PATH_NO_TRANSITION_OUTPUT 512`
Option flag to omit returning the transition path.
- `#define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)`
Option flag to request defaults (steepest descent / default move set)

Functions

- `vrna_move_t * vrna_path (vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)`
Compute a path, store the final structure, and return a list of transition moves from the start to the final structure.
- `vrna_move_t * vrna_path_gradient (vrna_fold_compound_t *vc, short *pt, unsigned int options)`
Compute a steepest descent / gradient path, store the final structure, and return a list of transition moves from the start to the final structure.
- `vrna_move_t * vrna_path_random (vrna_fold_compound_t *vc, short *pt, unsigned int steps, unsigned int options)`
Generate a random walk / path of a given length, store the final structure, and return a list of transition moves from the start to the final structure.

18.274.1 Detailed Description

Methods to generate particular paths such as gradient or random walks through the energy landscape of an RNA sequence.

18.275 walk.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_WALK_H
00002 #define VIENNA_RNA_PACKAGE_WALK_H
00003
00011 #include <ViennaRNA/fold_compound.h>
00012 #include <ViennaRNA/landscape/move.h>
00013
00024 #define VRNA_PATH_STEEPEST_DESCENT 128
00025
00030 #define VRNA_PATH_RANDOM 256
00031
00036 #define VRNA_PATH_NO_TRANSITION_OUTPUT 512
00037
00043 #define VRNA_PATH_DEFAULT (VRNA_PATH_STEEPEST_DESCENT | VRNA_MOVESET_DEFAULT)
00044
00073 vrna_move_t *
00074 vrna_path(vrna_fold_compound_t *vc,
00075          short *pt,
00076          unsigned int steps,
00077          unsigned int options);
00078
00079
00101 vrna_move_t *
00102 vrna_path_gradient(vrna_fold_compound_t *vc,
00103                  short *pt,
00104                  unsigned int options);
00105
00106
00129 vrna_move_t *
00130 vrna_path_random(vrna_fold_compound_t *vc,
00131                 short *pt,
00132                 unsigned int steps,
00133                 unsigned int options);
00134
00135
00140 #endif /* VIENNA_RNA_PACKAGE_WALK_H */
```

18.276 ViennaRNA/walk.h File Reference

Use [ViennaRNA/landscape/walk.h](#) instead.

Include dependency graph for walk.h:

18.276.1 Detailed Description

Use [ViennaRNA/landscape/walk.h](#) instead.

Deprecated Use [ViennaRNA/landscape/walk.h](#) instead

18.277 walk.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VIENNA_RNA_PACKAGE_WALK_DEPRECATED_H
00002 #define VIENNA_RNA_PACKAGE_WALK_DEPRECATED_H
00003
00010 #ifndef VRNA_DISABLE_BACKWARD_COMPATIBILITY
00011 # ifdef VRNA_WARN_DEPRECATED
00012 #warning "Including deprecated header file <ViennaRNA/walk.h>! Use <ViennaRNA/landscape/walk.h>
         instead!"
00013 # endif
00014 #include <ViennaRNA/landscape/walk.h>
00015 #include <ViennaRNA/landscape/neighbor.h>
00016 #endif
00017
00018 #endif
```

18.278 wrap_dlib.h

```
00001 #ifndef VIENNARNA_DLIB_WRAPPER_H
00002 #define VIENNARNA_DLIB_WRAPPER_H
00003
00004 #ifdef __cplusplus
00005 extern "C" {
00006 #endif
00007
00008 double *
00009 vrna_equilibrium_conc(const double      *eq_constants,
00010                     double             *concentration_strands,
00011                     const unsigned int **A,
00012                     size_t             num_strands,
00013                     size_t             num_complexes);
00014
00015
00016 #ifdef __cplusplus
00017 }
00018 #endif
00019
00020 #endif
```

18.279 zscore.h

```
00001 #ifndef VIENNA_RNA_PACKAGE_ZSCORE_H
00002 #define VIENNA_RNA_PACKAGE_ZSCORE_H
00003
00004 typedef struct vrna_zsc_dat_s *vrna_zsc_dat_t;
00005
00006 #define VRNA_ZSCORE_OPTIONS_NONE      1U
00007 #define VRNA_ZSCORE_FILTER_ON         2U
00008 #define VRNA_ZSCORE_PRE_FILTER        4U
00009 #define VRNA_ZSCORE_REPORT_SUBSUMED   8U
00010 #define VRNA_ZSCORE_MODEL_DEFAULT     16U
00011 #define VRNA_ZSCORE_SETTINGS_DEFAULT (VRNA_ZSCORE_FILTER_ON | VRNA_ZSCORE_MODEL_DEFAULT)
00012
00013 int
00014 vrna_zsc_filter_init(vrna_fold_compound_t *fc,
00015                    double                 min_z,
00016                    unsigned int           options);
00017
00018
00019 int
00020 vrna_zsc_filter_update(vrna_fold_compound_t *fc,
00021                      double                 min_z,
00022                      unsigned int           options);
```

```
00023
00024
00025 void
00026 vrna_zsc_filter_free(vrna_fold_compound_t *fc);
00027
00028
00029 int
00030 vrna_zsc_filter_on(vrna_fold_compound_t *fc);
00031
00032
00033 double
00034 vrna_zsc_filter_threshold(vrna_fold_compound_t *fc);
00035
00036
00037 double
00038 vrna_zsc_compute(vrna_fold_compound_t *fc,
00039                 unsigned int    i,
00040                 unsigned int    j,
00041                 int             e);
00042
00043
00044 double
00045 vrna_zsc_compute_raw(vrna_fold_compound_t *fc,
00046                     unsigned int    i,
00047                     unsigned int    j,
00048                     int             e,
00049                     double          *avg,
00050                     double          *sd);
00051
00052
00053 #endif
```

Bibliography

- [1] S.H. Bernhart, I.L. Hofacker, S. Will, A.R. Gruber, and P.F. Stadler. RNAalifold: Improved consensus structure prediction for RNA alignments. *BMC bioinformatics*, 9(1):474, 2008. [27](#)
- [2] S.H. Bernhart, H. Tafer, U. Mückstein, C. Flamm, P.F. Stadler, and I.L. Hofacker. Partition function and base pairing probabilities of RNA heterodimers. *Algorithms for Molecular Biology*, 1(1):3, 2006. [398](#)
- [3] Stephan H Bernhart, Ivo L Hofacker, and Peter F Stadler. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–615, 2005. [280](#)
- [4] Stephan H Bernhart, Ullrike Mückstein, and Ivo L Hofacker. RNA accessibility in cubic time. *Algorithms for Molecular Biology*, 6(1):3, 2011. [280](#), [739](#), [1233](#)
- [5] Pietro Boccaletto, Filip Stefaniak, Angana Ray, Andrea Cappannini, Sunandan Mukherjee, Elżbieta Purta, Małgorzata Kurkowska, Niloofar Shirvanizadeh, Eliana Destefanis, Paula Groza, et al. MODOMICS: a database of RNA modification pathways. 2021 update. *Nucleic Acids Research*, 50(D1):D231–D235, 2022. [376](#)
- [6] R.E. Brucoleri and G. Heinrich. An improved algorithm for nucleic acid secondary structure display. *Computer applications in the biosciences: CABIOS*, 4(1):167–173, 1988. [482](#), [598](#)
- [7] Joseph J Dalluge, Takeshi Hashizume, Alan E Sopchik, James A McCloskey, and Darrell R Davis. Conformational flexibility in RNA: the role of dihydrouridine. *Nucleic acids research*, 24(6):1073–1079, 1996. [382](#)
- [8] Katherine E. Deigan, Tian W. Li, David H. Mathews, and Kevin M. Weeks. Accurate SHAPE-directed RNA structure determination. *PNAS*, 106:97–102, 2009. [361](#)
- [9] Christoph Flamm, Ivo L Hofacker, Sebastian Maurer-Stroh, Peter F Stadler, and Martin Zehl. Design of multi-stable RNA molecules. *RNA*, 7(02):254–265, 2001. [352](#), [353](#), [354](#), [355](#)
- [10] W. Fontana, P.F. Stadler, E.G. Bornberg-Bauer, T. Griesmacher, I.L. Hofacker, M. Tacker, P. Tarazona, E.D. Weinberger, and P. Schuster. RNA folding and combinatorial landscapes. *Physical review E*, 47(3):2083, 1993. [33](#), [34](#), [443](#), [444](#), [446](#)
- [11] Eva Freyhult, Vincent Moulton, and Paul Gardner. Predicting RNA structure using mutual information. *Applied bioinformatics*, 4(1):53–59, 2005. [455](#)
- [12] Robert Giegerich, Björn Voß, and Marc Rehmsmeier. Abstract shapes of RNA. *Nucleic Acids Research*, 32(16):4843–4851, 2004. [33](#), [426](#), [440](#), [441](#), [442](#)
- [13] I.L. Hofacker, M. Fekete, and P.F. Stadler. Secondary structure prediction for aligned RNA sequences. *Journal of molecular biology*, 319(5):1059–1066, 2002. [27](#)
- [14] I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125(2):167–188, 1994. [1](#)
- [15] I.L. Hofacker and P.F. Stadler. Memory efficient folding algorithms for circular RNA secondary structures. *Bioinformatics*, 22(10):1172–1176, 2006. [24](#), [263](#), [264](#), [276](#), [277](#)
- [16] Graham A Hudson, Richard J Bloomingdale, and Brent M Znosko. Thermodynamic contribution and nearest-neighbor parameters of pseudouridine-adenosine base pairs in oligoribonucleotides. *RNA*, 19(11):1474–1482, 2013. [381](#)
- [17] Elizabeth A Jolley and Brent M Znosko. The loss of a hydrogen bond: Thermodynamic contributions of a non-standard nucleotide. *Nucleic acids research*, 45(3):1479–1487, 2017. [382](#)

Index

- (Abstract) Data Structures, [505](#)
 - [bondT](#), [509](#)
 - [cpair](#), [509](#)
 - [PAIR](#), [509](#)
 - [plist](#), [509](#)
 - [sect](#), [509](#)
 - [vrna_C11_features](#), [509](#)
- (Nucleic Acid Sequence) String Utilites, [415](#)
 - [FILENAME_ID_LENGTH](#), [417](#)
 - [FILENAME_MAX_LENGTH](#), [416](#)
 - [vrna_cut_point_insert](#), [425](#)
 - [vrna_cut_point_remove](#), [426](#)
 - [vrna_DNA_complement](#), [425](#)
 - [vrna_hamming_distance](#), [422](#)
 - [vrna_hamming_distance_bound](#), [422](#)
 - [vrna_random_string](#), [422](#)
 - [vrna_seq_reverse](#), [424](#)
 - [vrna_seq_toRNA](#), [424](#)
 - [vrna_seq_toupper](#), [424](#)
 - [vrna_seq_ungapped](#), [425](#)
 - [vrna_strcat_printf](#), [419](#)
 - [vrna_strcat_vprintf](#), [419](#)
 - [vrna_strdup_printf](#), [418](#)
 - [vrna_strdup_vprintf](#), [418](#)
 - [vrna_strsplit](#), [421](#)
 - [vrna_strtrim](#), [420](#)
 - [VRNA_TRIM_ALL](#), [418](#)
 - [VRNA_TRIM_DEFAULT](#), [417](#)
 - [VRNA_TRIM_IN_BETWEEN](#), [417](#)
 - [VRNA_TRIM_LEADING](#), [417](#)
 - [VRNA_TRIM_SUBST_BY_FIRST](#), [417](#)
 - [VRNA_TRIM_TRAILING](#), [417](#)
- (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [348](#)
 - [vrna_path_free](#), [350](#)
 - [vrna_path_options_free](#), [350](#)
 - [VRNA_PATH_TYPE_DOT_BRACKET](#), [350](#)
 - [VRNA_PATH_TYPE_MOVES](#), [350](#)
- [_struct_en](#), [601](#)
- [2Dpfold.h](#)
 - [destroy_TwoDpfold_variables](#), [608](#)
 - [get_TwoDpfold_variables](#), [608](#)
 - [TwoDpfold_pbacktrack](#), [609](#)
 - [TwoDpfold_pbacktrack5](#), [610](#)
 - [TwoDpfoldList](#), [609](#)
- Abstract Shapes Representation of Secondary Structures, [440](#)
 - [vrna_abstract_shapes](#), [441](#)
 - [vrna_abstract_shapes_pt](#), [442](#)
- [add_root](#)
 - Deprecated Interface for Secondary Structure Utilities, [590](#)
- [alifold](#)
 - Deprecated Interface for Global MFE Prediction, [556](#)
- [alifold.h](#)
 - [cv_fact](#), [615](#)
 - [energy_of_alistruct](#), [614](#)
 - [nc_fact](#), [615](#)
 - [update_alifold_params](#), [615](#)
- Alignment Plots, [494](#)
 - [vrna_file_PS_aln](#), [494](#)
 - [vrna_file_PS_aln_slice](#), [495](#)
- [alimake_pair_table](#)
 - Deprecated Interface for Secondary Structure Utilities, [593](#)
- [alipbacktrack](#)
 - Deprecated Interface for Global Partition Function Computation, [578](#)
- [alipf_circ_fold](#)
 - Deprecated Interface for Global Partition Function Computation, [577](#)
- [alipf_fold](#)
 - Deprecated Interface for Global Partition Function Computation, [577](#)
- [alipf_fold_par](#)
 - Deprecated Interface for Global Partition Function Computation, [567](#)
- [aliPS_color_aln](#)
 - Deprecated Interface for Plotting Utilities, [596](#)
- [alloc_sequence_arrays](#)
 - Deprecated Interface for Multiple Sequence Alignment Utilities, [586](#)
- [alpha](#)
 - [vrna_exp_param_s](#), [208](#)
- Annotation, [493](#)
 - [vrna_annotate_covar_db](#), [493](#)
 - [vrna_annotate_covar_pairs](#), [494](#)
- Arrays, [548](#)
 - [vrna__array_set_capacity](#), [550](#)
 - [vrna_array_init_size](#), [550](#)
- [assign_plist_from_db](#)
 - Deprecated Interface for Global Partition Function Computation, [576](#)
- [assign_plist_from_pr](#)
 - Deprecated Interface for Global Partition Function Computation, [576](#)
- [auxdata](#)

- vrna_fc_s, [521](#)
- b2C
 - Deprecated Interface for Secondary Structure Utilities, [589](#)
- b2HIT
 - Deprecated Interface for Secondary Structure Utilities, [589](#)
- b2Shapiro
 - Deprecated Interface for Secondary Structure Utilities, [590](#)
- backtrack_GQuad_IntLoop
 - G-Quadruplexes, [375](#)
- backtrack_GQuad_IntLoop_L
 - G-Quadruplexes, [375](#)
- backtrack_type
 - Fine-tuning of the Implemented Models, [205](#)
- Backtracking MFE structures, [269](#)
 - vrna_backtrack5, [270](#)
 - vrna_BT_hp_loop, [270](#)
 - vrna_BT_int_loop, [271](#)
 - vrna_BT_mb_loop, [271](#)
 - vrna_BT_stack, [270](#)
- base_pair
 - fold_vars.h, [691](#)
- basic.h
 - filecopy, [929](#)
 - get_line, [927](#)
 - init_rand, [929](#)
 - int_urn, [929](#)
 - nerror, [928](#)
 - print_tty_input_seq, [928](#)
 - print_tty_input_seq_str, [928](#)
 - space, [928](#)
 - time_stamp, [929](#)
 - urn, [929](#)
 - warn_user, [928](#)
 - xrealloc, [929](#)
- bondT
 - (Abstract) Data Structures, [509](#)
- BONUS
 - constants.h, [933](#)
- bp_distance
 - Deprecated Interface for Secondary Structure Utilities, [594](#)
- bppm_symbol
 - Deprecated Interface for Secondary Structure Utilities, [595](#)
- bppm_to_structure
 - Deprecated Interface for Secondary Structure Utilities, [595](#)
- bt
 - vrna_sc_s, [250](#)
- Buffers, [550](#)
 - vrna_cstr, [551](#)
 - vrna_cstr_close, [552](#)
 - vrna_cstr_discard, [552](#)
 - vrna_cstr_fflush, [553](#)
 - vrna_cstr_free, [552](#)
 - vrna_ostream_free, [553](#)
 - vrna_ostream_init, [553](#)
 - vrna_ostream_provide, [554](#)
 - vrna_ostream_request, [554](#)
 - vrna_stream_output_f, [551](#)
- centroid
 - part_func.h, [1225](#)
- centroid.h
 - get_centroid_struct_pl, [623](#)
 - get_centroid_struct_pr, [623](#)
- circularfold
 - Deprecated Interface for Global MFE Prediction, [563](#)
- circfold
 - Deprecated Interface for Global MFE Prediction, [560](#)
- Classified Dynamic Programming Variants, [316](#)
- co_pf_fold
 - Deprecated Interface for Global Partition Function Computation, [573](#)
- co_pf_fold_par
 - Deprecated Interface for Global Partition Function Computation, [573](#)
- cofold
 - Deprecated Interface for Global MFE Prediction, [556](#)
- cofold_par
 - Deprecated Interface for Global MFE Prediction, [557](#)
- Combinatorics Algorithms, [498](#)
 - vrna_boustrophedon, [504](#)
 - vrna_boustrophedon_pos, [504](#)
 - vrna_enumerate_necklaces, [499](#)
 - vrna_n_multichoose_k, [503](#)
 - vrna_rotational_symmetry, [501](#)
 - vrna_rotational_symmetry_db, [502](#)
 - vrna_rotational_symmetry_db_pos, [502](#)
 - vrna_rotational_symmetry_num, [499](#)
 - vrna_rotational_symmetry_pos, [501](#)
 - vrna_rotational_symmetry_pos_num, [500](#)
- Command Files, [471](#)
 - VRNA_CMD_PARSE_DEFAULTS, [472](#)
 - VRNA_CMD_PARSE_HC, [471](#)
 - VRNA_CMD_PARSE_SC, [472](#)
 - VRNA_CMD_PARSE_SD, [472](#)
 - VRNA_CMD_PARSE_UD, [472](#)
 - vrna_commands_apply, [473](#)
 - vrna_commands_free, [474](#)
 - vrna_file_commands_apply, [473](#)
 - vrna_file_commands_read, [472](#)
- Compute the Centroid Structure, [314](#)
 - vrna_centroid, [314](#)
 - vrna_centroid_from_plist, [314](#)
 - vrna_centroid_from_probs, [315](#)
- Compute the Density of States, [335](#)
 - density_of_states, [335](#)
- Compute the Structure with Maximum Expected Accuracy (MEA), [312](#)

- MEA, [313](#)
- vrna_MEA, [312](#)
- vrna_MEA_from_plist, [313](#)
- compute_BPdifferences
 - Deprecated Interface for Secondary Structure Utilities, [594](#)
- compute_probabilities
 - Deprecated Interface for Global Partition Function Computation, [574](#)
- Computing MFE representatives of a Distance Based Partitioning, [316](#)
- destroy_TwoDfold_variables, [321](#)
- get_TwoDfold_variables, [320](#)
- TwoDfold_backtrack_f5, [322](#)
- TwoDfold_vars, [319](#)
- TwoDfoldList, [321](#)
- vrna_backtrack5_TwoD, [320](#)
- vrna_mfe_TwoD, [319](#)
- vrna_sol_TwoD_t, [318](#)
- Computing Partition Functions of a Distance Based Partitioning, [322](#)
- vrna_pf_TwoD, [323](#)
- vrna_sol_TwoD_pf_t, [323](#)
- concentrations.h
 - get_concentrations, [632](#)
- cons_seq
 - vrna_fc_s, [523](#)
- constants.h
 - BONUS, [933](#)
 - FORBIDDEN, [933](#)
 - GASCONST, [932](#)
 - INF, [933](#)
 - K0, [932](#)
 - MAXLOOP, [933](#)
 - NBPAIRS, [933](#)
 - TURN, [933](#)
- constrain, [508](#)
- constrain_ptypes
 - hard.h, [637](#)
- Constraining the RNA Folding Grammar, [226](#)
- VRNA_CONSTRAINT_FILE, [229](#)
- VRNA_CONSTRAINT_SOFT_MFE, [230](#)
- VRNA_CONSTRAINT_SOFT_PF, [230](#)
- vrna_constraints_add, [237](#)
- VRNA_DECOMP_EXT_EXT, [235](#)
- VRNA_DECOMP_EXT_EXT_EXT, [236](#)
- VRNA_DECOMP_EXT_EXT_STEM, [236](#)
- VRNA_DECOMP_EXT_EXT_STEM1, [237](#)
- VRNA_DECOMP_EXT_STEM, [235](#)
- VRNA_DECOMP_EXT_STEM_EXT, [236](#)
- VRNA_DECOMP_EXT_STEM_OUTSIDE, [236](#)
- VRNA_DECOMP_EXT_UP, [235](#)
- VRNA_DECOMP_ML_COAXIAL, [234](#)
- VRNA_DECOMP_ML_COAXIAL_ENC, [234](#)
- VRNA_DECOMP_ML_ML, [233](#)
- VRNA_DECOMP_ML_ML_ML, [232](#)
- VRNA_DECOMP_ML_ML_STEM, [233](#)
- VRNA_DECOMP_ML_STEM, [232](#)
- VRNA_DECOMP_ML_UP, [233](#)
- VRNA_DECOMP_PAIR_HP, [230](#)
- VRNA_DECOMP_PAIR_IL, [231](#)
- VRNA_DECOMP_PAIR_ML, [231](#)
- vrna_message_constraint_options, [238](#)
- vrna_message_constraint_options_all, [239](#)
- convert_parameter_file
 - Converting Energy Parameter Files, [411](#)
- Converting Energy Parameter Files, [408](#)
- convert_parameter_file, [411](#)
- VRNA_CONVERT_OUTPUT_ALL, [408](#)
- VRNA_CONVERT_OUTPUT_BULGE, [410](#)
- VRNA_CONVERT_OUTPUT_DANGLE3, [410](#)
- VRNA_CONVERT_OUTPUT_DANGLE5, [409](#)
- VRNA_CONVERT_OUTPUT_DUMP, [411](#)
- VRNA_CONVERT_OUTPUT_HP, [409](#)
- VRNA_CONVERT_OUTPUT_INT, [410](#)
- VRNA_CONVERT_OUTPUT_INT_11, [410](#)
- VRNA_CONVERT_OUTPUT_INT_21, [410](#)
- VRNA_CONVERT_OUTPUT_INT_22, [410](#)
- VRNA_CONVERT_OUTPUT_MISC, [410](#)
- VRNA_CONVERT_OUTPUT_ML, [410](#)
- VRNA_CONVERT_OUTPUT_MM_EXT, [409](#)
- VRNA_CONVERT_OUTPUT_MM_HP, [409](#)
- VRNA_CONVERT_OUTPUT_MM_INT, [409](#)
- VRNA_CONVERT_OUTPUT_MM_INT_1N, [409](#)
- VRNA_CONVERT_OUTPUT_MM_INT_23, [409](#)
- VRNA_CONVERT_OUTPUT_MM_MULTI, [409](#)
- VRNA_CONVERT_OUTPUT_NINIO, [411](#)
- VRNA_CONVERT_OUTPUT_SPECIAL_HP, [411](#)
- VRNA_CONVERT_OUTPUT_STACK, [409](#)
- VRNA_CONVERT_OUTPUT_VANILLA, [411](#)
- COORDINATE, [596](#)
- copy_pair_table
 - Deprecated Interface for Secondary Structure Utilities, [593](#)
- cost_matrix
 - dist_vars.h, [661](#)
- cpair
 - (Abstract) Data Structures, [509](#)
- cut_point
 - fold_vars.h, [691](#)
- cv_fact
 - alifold.h, [615](#)
- dangles
 - Fine-tuning of the Implemented Models, [204](#)
 - vrna_md_s, [182](#)
- density_of_states
 - Compute the Density of States, [335](#)
- Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, [598](#)
- find_saddle, [599](#)
- free_path, [599](#)
- get_path, [599](#)
- path_t, [598](#)
- Deprecated Interface for Free Energy Evaluation, [162](#)
- E_IntLoop, [172](#)
- E_Stem, [170](#)

- energy_of_circ_struct, 170
- energy_of_circ_struct_par, 165
- energy_of_circ_structure, 164
- energy_of_move, 167
- energy_of_move_pt, 167
- energy_of_struct, 168
- energy_of_struct_par, 164
- energy_of_struct_pt, 169
- energy_of_struct_pt_par, 166
- energy_of_structure, 163
- energy_of_structure_pt, 166
- exp_E_ExtLoop, 171
- exp_E_IntLoop, 173
- exp_E_Stem, 171
- loop_energy, 168
- Deprecated Interface for Global MFE Prediction, 555
 - alifold, 556
 - circularfold, 563
 - circfold, 560
 - cofold, 556
 - cofold_par, 557
 - export_circfold_arrays, 562
 - export_circfold_arrays_par, 562
 - export_cofold_arrays, 558
 - export_cofold_arrays_gq, 557
 - export_fold_arrays, 561
 - export_fold_arrays_par, 562
 - fold, 560
 - fold_par, 559
 - free_alifold_arrays, 564
 - free_arrays, 561
 - free_co_arrays, 557
 - HairpinE, 563
 - initialize_cofold, 559
 - initialize_fold, 563
 - LoopEnergy, 563
 - update_cofold_params, 557
 - update_cofold_params_par, 557
 - update_fold_params, 561
 - update_fold_params_par, 561
- Deprecated Interface for Global Partition Function Computation, 565
 - alipbacktrack, 578
 - alipf_circ_fold, 577
 - alipf_fold, 577
 - alipf_fold_par, 567
 - assign_plist_from_db, 576
 - assign_plist_from_pr, 576
 - co_pf_fold, 573
 - co_pf_fold_par, 573
 - compute_probabilities, 574
 - export_ali_bppm, 578
 - export_bppm, 570
 - export_co_bppm, 575
 - free_alipf_arrays, 578
 - free_co_pf_arrays, 575
 - free_pf_arrays, 570
 - get_alipf_arrays, 579
 - get_pf_arrays, 571
 - get_subseq_F, 572
 - init_co_pf_fold, 575
 - init_pf_fold, 573
 - mean_bp_distance, 572
 - mean_bp_distance_pr, 572
 - pf_circ_fold, 569
 - pf_fold, 568
 - pf_fold_par, 567
 - stackProb, 573
 - update_co_pf_params, 575
 - update_co_pf_params_par, 576
 - update_pf_params, 570
 - update_pf_params_par, 570
- Deprecated Interface for Local (Sliding Window) MFE Prediction, 564
 - Lfold, 564
 - Lfoldz, 565
- Deprecated Interface for Local (Sliding Window) Partition Function Computation, 580
 - pfl_fold, 580
 - pfl_fold_par, 581
 - putoutpU_prob, 581
 - putoutpU_prob_bin, 583
 - update_pf_paramsLP, 580
- Deprecated Interface for Multiple Sequence Alignment Utilities, 585
 - alloc_sequence_arrays, 586
 - encode_ali_sequence, 586
 - free_sequence_arrays, 587
 - get_mpi, 585
 - pair_info, 585
- Deprecated Interface for Plotting Utilities, 596
 - aliPS_color_aln, 596
 - PS_color_aln, 596
 - rna_plot_type, 598
 - simple_circplot_coordinates, 597
 - simple_xy_coordinates, 597
- Deprecated Interface for Secondary Structure Utilities, 588
 - add_root, 590
 - alimake_pair_table, 593
 - b2C, 589
 - b2HIT, 589
 - b2Shapiro, 590
 - bp_distance, 594
 - bppm_symbol, 595
 - bppm_to_structure, 595
 - compute_BPdifferences, 594
 - copy_pair_table, 593
 - expand_Full, 591
 - expand_Shapiro, 590
 - make_pair_table, 593
 - make_pair_table_snoop, 594
 - make_referenceBP_array, 594
 - pack_structure, 592
 - parenthesis_structure, 595
 - parenthesis_zucker, 595

- parse_structure, [592](#)
 - unexpand_aligned_F, [591](#)
 - unexpand_Full, [591](#)
 - unpack_structure, [592](#)
 - unweight, [591](#)
- Deprecated Interface for Stochastic Backtracking, [583](#)
 - pbacktrack, [583](#)
 - pbacktrack5, [584](#)
 - pbacktrack_circ, [584](#)
 - st_back, [584](#)
- destroy_TwoDfold_variables
 - Computing MFE representatives of a Distance Based Partitioning, [321](#)
- destroy_TwoDpfold_variables
 - 2Dpfold.h, [608](#)
- Direct Refolding Paths between two Secondary Structures, [351](#)
 - vrna_path_direct, [355](#)
 - vrna_path_direct_ub, [355](#)
 - vrna_path_findpath, [353](#)
 - vrna_path_findpath_saddle, [351](#)
 - vrna_path_findpath_saddle_ub, [352](#)
 - vrna_path_findpath_ub, [353](#)
 - vrna_path_options_findpath, [354](#)
- dist_vars.h
 - cost_matrix, [661](#)
 - edit_backtrack, [661](#)
- Distance Based Partitioning of the Secondary Structure Space, [316](#)
- Distance measures between Secondary Structures, [447](#)
 - vrna_bp_distance, [448](#)
 - vrna_bp_distance_pt, [448](#)
- do_backtrack
 - Fine-tuning of the Implemented Models, [205](#)
- Dot-Bracket Notation of Secondary Structures, [428](#)
 - VRNA_BRACKETS_ALPHA, [429](#)
 - VRNA_BRACKETS_ANG, [430](#)
 - VRNA_BRACKETS_ANY, [430](#)
 - VRNA_BRACKETS_CLY, [429](#)
 - VRNA_BRACKETS_DEFAULT, [430](#)
 - VRNA_BRACKETS_RND, [429](#)
 - VRNA_BRACKETS_SQR, [430](#)
 - vrna_db_flatten, [431](#)
 - vrna_db_flatten_to, [432](#)
 - vrna_db_from_plist, [433](#)
 - vrna_db_from_ptable, [432](#)
 - vrna_db_pack, [431](#)
 - vrna_db_pk_remove, [434](#)
 - vrna_db_to_element_string, [433](#)
 - vrna_db_unpack, [431](#)
- duplexT, [508](#)
- dupVar, [508](#)
- E_Hairpin
 - Hairpin Loops, [393](#)
- E_IntLoop
 - Deprecated Interface for Free Energy Evaluation, [172](#)
- E_Stem
 - Deprecated Interface for Free Energy Evaluation, [170](#)
- edit_backtrack
 - dist_vars.h, [661](#)
- encode_ali_sequence
 - Deprecated Interface for Multiple Sequence Alignment Utilities, [586](#)
- energy
 - vrna_ht_entry_db_t, [536](#)
- Energy Evaluation for Atomic Moves, [161](#)
 - vrna_eval_move, [161](#)
 - vrna_eval_move_pt, [162](#)
- Energy Evaluation for Individual Loops, [159](#)
 - vrna_eval_loop_pt, [160](#)
 - vrna_eval_loop_pt_v, [160](#)
- Energy Parameters, [205](#)
 - get_boltzmann_factor_copy, [214](#)
 - get_boltzmann_factors, [213](#)
 - get_boltzmann_factors_ali, [215](#)
 - get_scaled_alipf_parameters, [214](#)
 - get_scaled_parameters, [215](#)
 - get_scaled_pf_parameters, [213](#)
 - paramT, [208](#)
 - pf_paramT, [208](#)
 - scale_parameters, [215](#)
 - vrna_exp_params, [209](#)
 - vrna_exp_params_comparative, [209](#)
 - vrna_exp_params_copy, [210](#)
 - vrna_exp_params_rescale, [211](#)
 - vrna_exp_params_reset, [213](#)
 - vrna_exp_params_subst, [211](#)
 - vrna_params, [208](#)
 - vrna_params_copy, [209](#)
 - vrna_params_reset, [212](#)
 - vrna_params_subst, [210](#)
 - vrna_salt_loop, [216](#)
 - vrna_salt_loop_int, [216](#)
 - vrna_salt_stack, [216](#)
- energy_corrections, [601](#)
- energy_of_alistruct
 - alifold.h, [614](#)
- energy_of_circ_struct
 - Deprecated Interface for Free Energy Evaluation, [170](#)
- energy_of_circ_struct_par
 - Deprecated Interface for Free Energy Evaluation, [165](#)
- energy_of_circ_structure
 - Deprecated Interface for Free Energy Evaluation, [164](#)
- energy_of_move
 - Deprecated Interface for Free Energy Evaluation, [167](#)
- energy_of_move_pt
 - Deprecated Interface for Free Energy Evaluation, [167](#)
- energy_of_struct

- Deprecated Interface for Free Energy Evaluation, [168](#)
- energy_of_struct_par
 - Deprecated Interface for Free Energy Evaluation, [164](#)
- energy_of_struct_pt
 - Deprecated Interface for Free Energy Evaluation, [169](#)
- energy_of_struct_pt_par
 - Deprecated Interface for Free Energy Evaluation, [166](#)
- energy_of_structure
 - Deprecated Interface for Free Energy Evaluation, [163](#)
- energy_of_structure_pt
 - Deprecated Interface for Free Energy Evaluation, [166](#)
- energy_set
 - Fine-tuning of the Implemented Models, [204](#)
- exp_E_ExtLoop
 - Deprecated Interface for Free Energy Evaluation, [171](#)
- exp_E_Hairpin
 - Hairpin Loops, [394](#)
- exp_E_IntLoop
 - Deprecated Interface for Free Energy Evaluation, [173](#)
- exp_E_Stem
 - Deprecated Interface for Free Energy Evaluation, [171](#)
- exp_f
 - vrna_sc_s, [250](#)
- expand_Full
 - Deprecated Interface for Secondary Structure Utilities, [591](#)
- expand_Shapiro
 - Deprecated Interface for Secondary Structure Utilities, [590](#)
- Experimental Structure Probing Data, [360](#)
- expHairpinEnergy
 - part_func.h, [1226](#)
- expLoopEnergy
 - part_func.h, [1226](#)
- expMLbase
 - vrna_mx_pf_s, [532](#)
- export_al_i_bppm
 - Deprecated Interface for Global Partition Function Computation, [578](#)
- export_bppm
 - Deprecated Interface for Global Partition Function Computation, [570](#)
- export_circfold_arrays
 - Deprecated Interface for Global MFE Prediction, [562](#)
- export_circfold_arrays_par
 - Deprecated Interface for Global MFE Prediction, [562](#)
- export_co_bppm
 - Deprecated Interface for Global Partition Function Computation, [575](#)
- export_cofold_arrays
 - Deprecated Interface for Global MFE Prediction, [558](#)
- export_cofold_arrays_gq
 - Deprecated Interface for Global MFE Prediction, [557](#)
- export_fold_arrays
 - Deprecated Interface for Global MFE Prediction, [561](#)
- export_fold_arrays_par
 - Deprecated Interface for Global MFE Prediction, [562](#)
- Extending the Folding Grammar with Additional Domains, [217](#)
- Exterior Loops, [389](#)
 - vrna_E_ext_stem, [390](#)
 - vrna_eval_ext_stem, [391](#)
 - vrna_exp_E_ext_stem, [391](#)
 - vrna_mx_pf_aux_el_t, [390](#)
- f
 - vrna_sc_s, [250](#)
- filecopy
 - basic.h, [929](#)
- FILENAME_ID_LENGTH
 - (Nucleic Acid Sequence) String Utilities, [417](#)
- FILENAME_MAX_LENGTH
 - (Nucleic Acid Sequence) String Utilities, [416](#)
- Files and I/O, [455](#)
 - get_ribosum, [456](#)
 - readribosum, [456](#)
 - vrna_file_exists, [458](#)
 - vrna_filename_sanitize, [457](#)
 - vrna_read_line, [456](#)
- final_cost
 - Inverse Folding (Design), [337](#)
- find_saddle
 - Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, [599](#)
- Fine-tuning of the Implemented Models, [175](#)
 - backtrack_type, [205](#)
 - dangles, [204](#)
 - do_backtrack, [205](#)
 - energy_set, [204](#)
 - max_bp_span, [205](#)
 - noLonelyPairs, [204](#)
 - nonstandards, [205](#)
 - pf_scale, [204](#)
 - set_model_details, [203](#)
 - temperature, [203](#)
 - tetra_loop, [204](#)
 - vrna_md_copy, [187](#)
 - vrna_md_defaults_backtrack, [195](#)
 - vrna_md_defaults_backtrack_get, [195](#)
 - vrna_md_defaults_backtrack_type, [196](#)
 - vrna_md_defaults_backtrack_type_get, [196](#)
 - vrna_md_defaults_betaScale, [189](#)

- [vrna_md_defaults_betaScale_get](#), [189](#)
- [vrna_md_defaults_circ](#), [193](#)
- [vrna_md_defaults_circ_get](#), [193](#)
- [vrna_md_defaults_compute_bpp](#), [196](#)
- [vrna_md_defaults_compute_bpp_get](#), [197](#)
- [vrna_md_defaults_cv_fact](#), [200](#)
- [vrna_md_defaults_cv_fact_get](#), [200](#)
- [vrna_md_defaults_dangles](#), [189](#)
- [vrna_md_defaults_dangles_get](#), [190](#)
- [vrna_md_defaults_energy_set](#), [195](#)
- [vrna_md_defaults_energy_set_get](#), [195](#)
- [vrna_md_defaults_gquad](#), [193](#)
- [vrna_md_defaults_gquad_get](#), [194](#)
- [vrna_md_defaults_logML](#), [192](#)
- [vrna_md_defaults_logML_get](#), [193](#)
- [vrna_md_defaults_max_bp_span](#), [197](#)
- [vrna_md_defaults_max_bp_span_get](#), [197](#)
- [vrna_md_defaults_min_loop_size](#), [197](#)
- [vrna_md_defaults_min_loop_size_get](#), [198](#)
- [vrna_md_defaults_nc_fact](#), [200](#)
- [vrna_md_defaults_nc_fact_get](#), [201](#)
- [vrna_md_defaults_noGU](#), [191](#)
- [vrna_md_defaults_noGU_get](#), [191](#)
- [vrna_md_defaults_noGUclosure](#), [192](#)
- [vrna_md_defaults_noGUclosure_get](#), [192](#)
- [vrna_md_defaults_noLP](#), [191](#)
- [vrna_md_defaults_noLP_get](#), [191](#)
- [vrna_md_defaults_oldAliEn](#), [199](#)
- [vrna_md_defaults_oldAliEn_get](#), [199](#)
- [vrna_md_defaults_reset](#), [188](#)
- [vrna_md_defaults_ribo](#), [199](#)
- [vrna_md_defaults_ribo_get](#), [199](#)
- [vrna_md_defaults_salt](#), [201](#)
- [vrna_md_defaults_salt_get](#), [202](#)
- [vrna_md_defaults_saltDPXInit](#), [203](#)
- [vrna_md_defaults_saltDPXInit_get](#), [203](#)
- [vrna_md_defaults_saltMLLower](#), [202](#)
- [vrna_md_defaults_saltMLLower_get](#), [202](#)
- [vrna_md_defaults_saltMLUpper](#), [202](#)
- [vrna_md_defaults_saltMLUpper_get](#), [202](#)
- [vrna_md_defaults_sfact](#), [201](#)
- [vrna_md_defaults_sfact_get](#), [201](#)
- [vrna_md_defaults_special_hp](#), [190](#)
- [vrna_md_defaults_special_hp_get](#), [190](#)
- [vrna_md_defaults_temperature](#), [188](#)
- [vrna_md_defaults_temperature_get](#), [189](#)
- [vrna_md_defaults_uniq_ML](#), [194](#)
- [vrna_md_defaults_uniq_ML_get](#), [194](#)
- [vrna_md_defaults_window_size](#), [198](#)
- [vrna_md_defaults_window_size_get](#), [198](#)
- [vrna_md_option_string](#), [188](#)
- [vrna_md_set_default](#), [187](#)
- [vrna_md_update](#), [187](#)
- [VRNA_MODEL_DEFAULT_ALI_CV_FACT](#), [186](#)
- [VRNA_MODEL_DEFAULT_ALI_NC_FACT](#), [186](#)
- [VRNA_MODEL_DEFAULT_ALI_OLD_EN](#), [186](#)
- [VRNA_MODEL_DEFAULT_ALI_RIBO](#), [186](#)
- [VRNA_MODEL_DEFAULT_BACKTRACK_TYPE](#), [185](#)
- [VRNA_MODEL_DEFAULT_BETA_SCALE](#), [183](#)
- [VRNA_MODEL_DEFAULT_CIRC](#), [184](#)
- [VRNA_MODEL_DEFAULT_COMPUTE_BPP](#), [185](#)
- [VRNA_MODEL_DEFAULT_DANGLES](#), [183](#)
- [VRNA_MODEL_DEFAULT_ENERGY_SET](#), [185](#)
- [VRNA_MODEL_DEFAULT_GQUAD](#), [184](#)
- [VRNA_MODEL_DEFAULT_LOG_ML](#), [186](#)
- [VRNA_MODEL_DEFAULT_MAX_BP_SPAN](#), [185](#)
- [VRNA_MODEL_DEFAULT_NO_GU](#), [184](#)
- [VRNA_MODEL_DEFAULT_NO_GU_CLOSURE](#), [184](#)
- [VRNA_MODEL_DEFAULT_NO_LP](#), [184](#)
- [VRNA_MODEL_DEFAULT_PF_SCALE](#), [183](#)
- [VRNA_MODEL_DEFAULT_SPECIAL_HP](#), [184](#)
- [VRNA_MODEL_DEFAULT_TEMPERATURE](#), [183](#)
- [VRNA_MODEL_DEFAULT_UNIQ_ML](#), [185](#)
- [VRNA_MODEL_DEFAULT_WINDOW_SIZE](#), [186](#)
- [fold](#)
 - Deprecated Interface for Global MFE Prediction, [560](#)
- [fold_par](#)
 - Deprecated Interface for Global MFE Prediction, [559](#)
- [fold_vars.h](#)
 - [base_pair](#), [691](#)
 - [cut_point](#), [691](#)
 - [iindx](#), [692](#)
 - [james_rule](#), [691](#)
 - [logML](#), [691](#)
 - [pr](#), [692](#)
 - [RibosumFile](#), [691](#)
- [Folding Paths that start at a single Secondary Structure](#), [356](#)
 - [vrna_path](#), [357](#)
 - [VRNA_PATH_DEFAULT](#), [357](#)
 - [vrna_path_gradient](#), [358](#)
 - [VRNA_PATH_NO_TRANSITION_OUTPUT](#), [357](#)
 - [VRNA_PATH_RANDOM](#), [357](#)
 - [vrna_path_random](#), [359](#)
 - [VRNA_PATH_STEEPEST_DESCENT](#), [357](#)
- [FORBIDDEN](#)
 - [constants.h](#), [933](#)
- [Free Energy Evaluation](#), [139](#)
 - [vrna_eval_circ_consensus_structure](#), [151](#)
 - [vrna_eval_circ_consensus_structure_v](#), [154](#)
 - [vrna_eval_circ_gquad_consensus_structure](#), [152](#)
 - [vrna_eval_circ_gquad_consensus_structure_v](#), [156](#)
 - [vrna_eval_circ_gquad_structure](#), [147](#)
 - [vrna_eval_circ_gquad_structure_v](#), [150](#)
 - [vrna_eval_circ_structure](#), [146](#)
 - [vrna_eval_circ_structure_v](#), [149](#)
 - [vrna_eval_consensus_structure_pt_simple](#), [158](#)
 - [vrna_eval_consensus_structure_pt_simple_v](#), [159](#)
 - [vrna_eval_consensus_structure_pt_simple_verbose](#), [159](#)

- vrna_eval_consensus_structure_simple, [151](#)
- vrna_eval_consensus_structure_simple_v, [154](#)
- vrna_eval_consensus_structure_simple_verbose, [153](#)
- vrna_eval_covar_structure, [142](#)
- vrna_eval_gquad_consensus_structure, [152](#)
- vrna_eval_gquad_consensus_structure_v, [155](#)
- vrna_eval_gquad_structure, [146](#)
- vrna_eval_gquad_structure_v, [149](#)
- vrna_eval_structure, [142](#)
- vrna_eval_structure_pt, [144](#)
- vrna_eval_structure_pt_simple, [157](#)
- vrna_eval_structure_pt_simple_v, [157](#)
- vrna_eval_structure_pt_simple_verbose, [157](#)
- vrna_eval_structure_pt_v, [145](#)
- vrna_eval_structure_pt_verbose, [144](#)
- vrna_eval_structure_simple, [145](#)
- vrna_eval_structure_simple_v, [148](#)
- vrna_eval_structure_simple_verbose, [148](#)
- vrna_eval_structure_v, [143](#)
- vrna_eval_structure_verbose, [143](#)
- free_alifold_arrays
 - Deprecated Interface for Global MFE Prediction, [564](#)
- free_alipf_arrays
 - Deprecated Interface for Global Partition Function Computation, [578](#)
- free_arrays
 - Deprecated Interface for Global MFE Prediction, [561](#)
- free_auxdata
 - vrna_fc_s, [521](#)
- free_co_arrays
 - Deprecated Interface for Global MFE Prediction, [557](#)
- free_co_pf_arrays
 - Deprecated Interface for Global Partition Function Computation, [575](#)
- free_data
 - vrna_hc_s, [241](#)
- free_path
 - Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, [599](#)
- free_pf_arrays
 - Deprecated Interface for Global Partition Function Computation, [570](#)
- free_profile
 - profiledist.h, [1263](#)
- free_sequence_arrays
 - Deprecated Interface for Multiple Sequence Alignment Utilities, [587](#)
- free_tree
 - treedist.h, [1282](#)
- G-Quadruplexes, [374](#)
 - backtrack_GQuad_IntLoop, [375](#)
 - backtrack_GQuad_IntLoop_L, [375](#)
 - get_gquad_matrix, [374](#)
 - parse_gquad, [375](#)
- GASCONST
 - constants.h, [932](#)
- Generate Soft Constraints from Data, [363](#)
 - progress_callback, [365](#)
 - VRNA_MINIMIZER_CONJUGATE_FR, [364](#)
 - VRNA_MINIMIZER_CONJUGATE_PR, [365](#)
 - VRNA_MINIMIZER_DEFAULT, [364](#)
 - VRNA_MINIMIZER_STEEPEST_DESCENT, [365](#)
 - VRNA_MINIMIZER_VECTOR_BFGS, [365](#)
 - VRNA_MINIMIZER_VECTOR_BFGS2, [365](#)
 - VRNA_OBJECTIVE_FUNCTION_ABSOLUTE, [364](#)
 - VRNA_OBJECTIVE_FUNCTION_QUADRATIC, [364](#)
 - vrna_sc_minimize_perturbation, [365](#)
- get_alipf_arrays
 - Deprecated Interface for Global Partition Function Computation, [579](#)
- get_boltzmann_factor_copy
 - Energy Parameters, [214](#)
- get_boltzmann_factors
 - Energy Parameters, [213](#)
- get_boltzmann_factors_al
 - Energy Parameters, [215](#)
- get_centroid_struct_gquad_pr
 - part_func.h, [1226](#)
- get_centroid_struct_pl
 - centroid.h, [623](#)
- get_centroid_struct_pr
 - centroid.h, [623](#)
- get_concentrations
 - concentrations.h, [632](#)
- get_gquad_matrix
 - G-Quadruplexes, [374](#)
- get_input_line
 - Utilities, [387](#)
- get_line
 - basic.h, [927](#)
- get_mpi
 - Deprecated Interface for Multiple Sequence Alignment Utilities, [585](#)
- get_path
 - Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, [599](#)
- get_pf_arrays
 - Deprecated Interface for Global Partition Function Computation, [571](#)
- get_plist
 - part_func_co.h, [1230](#)
- get_ribosum
 - Files and I/O, [456](#)
- get_scaled_alipf_parameters
 - Energy Parameters, [214](#)
- get_scaled_parameters
 - Energy Parameters, [215](#)
- get_scaled_pf_parameters
 - Energy Parameters, [213](#)
- get_subseq_F

- Deprecated Interface for Global Partition Function Computation, [572](#)
- get_TwoDfold_variables
 - Computing MFE representatives of a Distance Based Partitioning, [320](#)
- get_TwoDpfold_variables
 - 2Dpfold.h, [608](#)
- give_up
 - Inverse Folding (Design), [337](#)
- Global MFE Prediction, [260](#)
 - vrna_alifold, [263](#)
 - vrna_circalifold, [264](#)
 - vrna_circfold, [263](#)
 - vrna_cofold, [264](#)
 - vrna_fold, [262](#)
 - vrna_mfe, [261](#)
 - vrna_mfe_dimer, [261](#)
- Global Partition Function and Equilibrium Probabilities, [272](#)
 - vrna_pf, [273](#)
 - vrna_pf_alifold, [276](#)
 - vrna_pf_circalifold, [277](#)
 - vrna_pf_circfold, [275](#)
 - vrna_pf_co_fold, [278](#)
 - vrna_pf_dimer, [274](#)
 - vrna_pf_fold, [275](#)
 - vrna_plist_from_probs, [277](#)
- gmIRNA
 - Plotting, [477](#)
- Hairpin Loops, [392](#)
 - E_Hairpin, [393](#)
 - exp_E_Hairpin, [394](#)
 - vrna_E_ext_hp_loop, [393](#)
 - vrna_E_hp_loop, [392](#)
 - vrna_eval_hp_loop, [393](#)
 - vrna_exp_E_hp_loop, [395](#)
- HairpinE
 - Deprecated Interface for Global MFE Prediction, [563](#)
- hamming
 - strings.h, [1293](#)
- hamming_bound
 - strings.h, [1294](#)
- Hard Constraints, [239](#)
 - VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS, [245](#)
 - VRNA_CONSTRAINT_CONTEXT_EXT_LOOP, [244](#)
 - VRNA_CONSTRAINT_CONTEXT_HP_LOOP, [244](#)
 - VRNA_CONSTRAINT_CONTEXT_INT_LOOP, [244](#)
 - VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC, [244](#)
 - VRNA_CONSTRAINT_CONTEXT_MB_LOOP, [244](#)
 - VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC, [245](#)
 - VRNA_CONSTRAINT_DB, [242](#)
 - VRNA_CONSTRAINT_DB_DEFAULT, [244](#)
 - VRNA_CONSTRAINT_DB_DOT, [242](#)
 - VRNA_CONSTRAINT_DB_ENFORCE_BP, [242](#)
 - VRNA_CONSTRAINT_DB_GQUAD, [243](#)
 - VRNA_CONSTRAINT_DB_INTERMOL, [243](#)
 - VRNA_CONSTRAINT_DB_INTRAMOL, [243](#)
 - VRNA_CONSTRAINT_DB_PIPE, [242](#)
 - VRNA_CONSTRAINT_DB_RND_BRACK, [243](#)
 - VRNA_CONSTRAINT_DB_WUSS, [243](#)
 - VRNA_CONSTRAINT_DB_X, [242](#)
 - vrna_hc_add_bp, [247](#)
 - vrna_hc_add_bp_nonspecific, [247](#)
 - vrna_hc_add_from_db, [248](#)
 - vrna_hc_add_up, [246](#)
 - vrna_hc_add_up_batch, [246](#)
 - vrna_hc_eval_f, [245](#)
 - vrna_hc_free, [248](#)
 - vrna_hc_init, [246](#)
- hard.h
 - constrain_ptypes, [637](#)
 - print_tty_constraint, [637](#)
 - print_tty_constraint_full, [637](#)
 - VRNA_CONSTRAINT_DB_ANG_BRACK, [636](#)
 - VRNA_CONSTRAINT_NO_HEADER, [636](#)
 - vrna_hc_add_data, [637](#)
 - VRNA_HC_DEFAULT, [636](#)
 - vrna_hc_type_e, [636](#)
 - VRNA_HC_WINDOW, [636](#)
- Hash Tables, [534](#)
 - vrna_hash_table_t, [536](#)
 - vrna_ht_clear, [539](#)
 - vrna_ht_cmp_f, [536](#)
 - vrna_ht_collisions, [538](#)
 - vrna_ht_db_comp, [540](#)
 - vrna_ht_db_free_entry, [541](#)
 - vrna_ht_db_hash_func, [540](#)
 - vrna_ht_free, [540](#)
 - vrna_ht_free_f, [537](#)
 - vrna_ht_get, [538](#)
 - vrna_ht_hashfunc_f, [536](#)
 - vrna_ht_init, [537](#)
 - vrna_ht_insert, [539](#)
 - vrna_ht_remove, [539](#)
 - vrna_ht_size, [538](#)
- Heaps, [541](#)
 - vrna_heap_cmp_f, [542](#)
 - vrna_heap_free, [545](#)
 - vrna_heap_get_pos_f, [544](#)
 - vrna_heap_init, [544](#)
 - vrna_heap_insert, [546](#)
 - vrna_heap_pop, [546](#)
 - vrna_heap_remove, [547](#)
 - vrna_heap_set_pos_f, [544](#)
 - vrna_heap_size, [545](#)
 - vrna_heap_t, [542](#)
 - vrna_heap_top, [546](#)
 - vrna_heap_update, [547](#)

- Helix List Representation of Secondary Structures, [442](#)
 - vrna_hx_from_ptable, [443](#)
- id
 - vrna_exp_param_s, [208](#)
- iindx
 - fold_vars.h, [692](#)
- Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints, [367](#)
 - vrna_sc_add_hi_motif, [368](#)
- INF
 - constants.h, [933](#)
- init_co_pf_fold
 - Deprecated Interface for Global Partition Function Computation, [575](#)
- init_pf_fold
 - Deprecated Interface for Global Partition Function Computation, [573](#)
- init_pf_foldLP
 - LPfold.h, [740](#)
- init_rand
 - basic.h, [929](#)
- initialize_cofold
 - Deprecated Interface for Global MFE Prediction, [559](#)
- initialize_fold
 - Deprecated Interface for Global MFE Prediction, [563](#)
- int_urn
 - basic.h, [929](#)
- interact, [507](#)
- Internal Loops, [395](#)
 - vrna_eval_int_loop, [396](#)
- inv_verbose
 - Inverse Folding (Design), [337](#)
- Inverse Folding (Design), [336](#)
 - final_cost, [337](#)
 - give_up, [337](#)
 - inv_verbose, [337](#)
 - inverse_fold, [336](#)
 - inverse_pf_fold, [336](#)
- inverse_fold
 - Inverse Folding (Design), [336](#)
- inverse_pf_fold
 - Inverse Folding (Design), [336](#)
- james_rule
 - fold_vars.h, [691](#)
- K0
 - constants.h, [932](#)
- last_parameter_file
 - Reading/Writing Energy Parameter Sets from/to File, [407](#)
- Layouts and Coordinates, [480](#)
 - vrna_plot_coords, [485](#)
 - vrna_plot_coords_circular, [488](#)
 - vrna_plot_coords_circular_pt, [489](#)
 - vrna_plot_coords_pt, [486](#)
 - vrna_plot_coords_puzzler, [490](#)
 - vrna_plot_coords_puzzler_pt, [490](#)
 - vrna_plot_coords_simple, [487](#)
 - vrna_plot_coords_simple_pt, [488](#)
 - vrna_plot_coords_turtle, [492](#)
 - vrna_plot_coords_turtle_pt, [492](#)
 - vrna_plot_layout, [482](#)
 - vrna_plot_layout_circular, [484](#)
 - vrna_plot_layout_free, [485](#)
 - vrna_plot_layout_puzzler, [485](#)
 - vrna_plot_layout_simple, [483](#)
 - vrna_plot_layout_t, [482](#)
 - vrna_plot_layout_turtle, [484](#)
 - vrna_plot_options_puzzler, [491](#)
 - vrna_plot_options_puzzler_free, [491](#)
 - VRNA_PLOT_TYPE_CIRCULAR, [482](#)
 - VRNA_PLOT_TYPE_NAVIEW, [481](#)
 - VRNA_PLOT_TYPE_PUZZLER, [482](#)
 - VRNA_PLOT_TYPE_SIMPLE, [481](#)
 - VRNA_PLOT_TYPE_TURTLE, [482](#)
- length
 - vrna_mx_pf_s, [532](#)
- Lfold
 - Deprecated Interface for Local (Sliding Window) MFE Prediction, [564](#)
- Lfoldz
 - Deprecated Interface for Local (Sliding Window) MFE Prediction, [565](#)
- Ligands Binding to RNA Structures, [367](#)
- Ligands Binding to Unstructured Domains, [367](#)
- LIST, [601](#)
- Local (sliding window) MFE Prediction, [265](#)
 - vrna_Lfold, [268](#)
 - vrna_Lfoldz, [268](#)
 - vrna_mfe_window, [267](#)
 - vrna_mfe_window_f, [266](#)
 - vrna_mfe_window_zscore, [267](#)
- Local (sliding window) Partition Function and Equilibrium Probabilities, [278](#)
 - vrna_pfl_fold, [283](#)
 - vrna_pfl_fold_cb, [283](#)
 - vrna_pfl_fold_up, [284](#)
 - vrna_pfl_fold_up_cb, [284](#)
 - vrna_probs_window, [282](#)
 - VRNA_PROBS_WINDOW_BPP, [279](#)
 - vrna_probs_window_f, [281](#)
 - VRNA_PROBS_WINDOW_PF, [281](#)
 - VRNA_PROBS_WINDOW_STACKP, [280](#)
 - VRNA_PROBS_WINDOW_UP, [280](#)
 - VRNA_PROBS_WINDOW_UP_SPLIT, [280](#)
- logML
 - fold_vars.h, [691](#)
- loop_energy
 - Deprecated Interface for Free Energy Evaluation, [168](#)
- LoopEnergy

- Deprecated Interface for Global MFE Prediction, 563
- LPfold.h
 - init_pf_foldLP, 740
- LST_BUCKET, 601
- Make_bp_profile
 - profiledist.h, 1263
- Make_bp_profile_bppm
 - profiledist.h, 1263
- make_pair_table
 - Deprecated Interface for Secondary Structure Utilities, 593
- make_pair_table_snoop
 - Deprecated Interface for Secondary Structure Utilities, 594
- make_referenceBP_array
 - Deprecated Interface for Secondary Structure Utilities, 594
- Make_swString
 - stringdist.h, 1276
- make_tree
 - treedist.h, 1281
- max_bp_span
 - Fine-tuning of the Implemented Models, 205
- MAXLOOP
 - constants.h, 933
- MEA
 - Compute the Structure with Maximum Expected Accuracy (MEA), 313
- mean_bp_dist
 - part_func.h, 1226
- mean_bp_distance
 - Deprecated Interface for Global Partition Function Computation, 572
- mean_bp_distance_pr
 - Deprecated Interface for Global Partition Function Computation, 572
- Messages, 510
 - vrna_message_error, 511
 - vrna_message_info, 512
 - vrna_message_input_seq, 513
 - vrna_message_input_seq_simple, 513
 - vrna_message_verror, 511
 - vrna_message_vinfo, 512
 - vrna_message_vwarning, 512
 - vrna_message_warning, 511
- min_loop_size
 - vrna_md_s, 183
- Minimum Free Energy (MFE) Algorithms, 258
- mm.h
 - vrna_maximum_matching, 746
 - vrna_maximum_matching_simple, 746
- Multibranch Loops, 396
 - vrna_E_mb_loop_stack, 397
 - vrna_mx_pf_aux_ml_t, 397
- Multiple Sequence Alignment Utilities, 449
 - vrna_aln_consensus_mis, 455
 - vrna_aln_consensus_sequence, 454
 - vrna_aln_conservation_col, 454
 - vrna_aln_conservation_struct, 453
 - vrna_aln_copy, 453
 - vrna_aln_free, 452
 - vrna_aln_mpi, 451
 - vrna_aln_pinfo, 451
 - vrna_aln_slice, 451
 - vrna_aln_toRNA, 453
 - vrna_aln_uppercase, 452
 - VRNA_MEASURE_SHANNON_ENTROPY, 451
- Multiple Sequence Alignments, 464
 - VRNA_FILE_FORMAT_MSA_APPEND, 466
 - VRNA_FILE_FORMAT_MSA_CLUSTAL, 465
 - VRNA_FILE_FORMAT_MSA_DEFAULT, 466
 - VRNA_FILE_FORMAT_MSA_FASTA, 465
 - VRNA_FILE_FORMAT_MSA_MAF, 465
 - VRNA_FILE_FORMAT_MSA_MIS, 465
 - VRNA_FILE_FORMAT_MSA_NOCHECK, 466
 - VRNA_FILE_FORMAT_MSA_QUIET, 466
 - VRNA_FILE_FORMAT_MSA_SILENT, 467
 - VRNA_FILE_FORMAT_MSA_STOCKHOLM, 465
 - VRNA_FILE_FORMAT_MSA_UNKNOWN, 466
 - vrna_file_msa_detect_format, 469
 - vrna_file_msa_read, 467
 - vrna_file_msa_read_record, 468
 - vrna_file_msa_write, 470
- n_seq
 - vrna_fc_s, 522
- NBPAIRS
 - constants.h, 933
- nc_fact
 - alifold.h, 615
- Neighborhood Relation and Move Sets for Secondary Structures, 337
 - vrna_loopidx_update, 345
 - vrna_move_apply, 343
 - vrna_move_compare, 344
 - vrna_move_init, 343
 - vrna_move_is_insertion, 344
 - vrna_move_is_removal, 343
 - vrna_move_is_shift, 344
 - vrna_move_list_free, 343
 - vrna_move_neighbor_diff, 347
 - vrna_move_neighbor_diff_cb, 346
 - vrna_move_update_f, 342
 - VRNA_MOVESET_DEFAULT, 341
 - VRNA_MOVESET_DELETION, 341
 - VRNA_MOVESET_INSERTION, 341
 - VRNA_MOVESET_NO_LP, 341
 - VRNA_MOVESET_SHIFT, 341
 - VRNA_NEIGHBOR_CHANGE, 342
 - VRNA_NEIGHBOR_INVALID, 342
 - VRNA_NEIGHBOR_NEW, 342
 - vrna_neighbors, 345
 - vrna_neighbors_successive, 346
- node, 508
- noLonelyPairs
 - Fine-tuning of the Implemented Models, 204

- nonstandards
 - Fine-tuning of the Implemented Models, 205
- nrrerror
 - basic.h, 928
- Nucleic Acid Sequences and Structures, 458
 - read_record, 463
 - VRNA_CONSTRAINT_MULTILINE, 459
 - vrna_extract_record_rest_constraint, 463
 - vrna_extract_record_rest_structure, 462
 - vrna_file_bpseq, 460
 - vrna_file_connect, 460
 - vrna_file_fasta_read_record, 461
 - vrna_file_helixlist, 459
 - vrna_file_json, 460
 - vrna_file_SHAPE_read, 463
 - VRNA_OPTION_MULTILINE, 459
- pack_structure
 - Deprecated Interface for Secondary Structure Utilities, 592
- PAIR
 - (Abstract) Data Structures, 509
- Pair List Representation of Secondary Structures, 439
 - vrna_plist, 440
- Pair Table Representation of Secondary Structures, 436
 - vrna_pt_aliget, 438
 - vrna_pt_pk_get, 437
 - vrna_pt_pk_remove, 438
 - vrna_pt_snoop_get, 438
 - vrna_ptable, 436
 - vrna_ptable_copy, 438
 - vrna_ptable_from_string, 437
- pair_info
 - Deprecated Interface for Multiple Sequence Alignment Utilities, 585
- paramT
 - Energy Parameters, 208
- parenthesis_structure
 - Deprecated Interface for Secondary Structure Utilities, 595
- parenthesis_zucker
 - Deprecated Interface for Secondary Structure Utilities, 595
- parse_gquad
 - G-Quadruplexes, 375
- parse_structure
 - Deprecated Interface for Secondary Structure Utilities, 592
- part_func.h
 - centroid, 1225
 - expHairpinEnergy, 1226
 - expLoopEnergy, 1226
 - get_centroid_struct_gquad_pr, 1226
 - mean_bp_dist, 1226
- part_func_co.h
 - get_plist, 1230
- Partition Function and Equilibrium Properties, 259
 - vrna_pf_float_precision, 259
- Partition Function for Two Hybridized Sequences, 397
 - vrna_pf_co_fold, 398
 - vrna_pf_dimer_concentrations, 399
- Partition Function for two Hybridized Sequences as a Stepwise Process, 400
 - pf_interact, 401
 - pf_unstru, 400
- path_t
 - Deprecated Interface for (Re-)folding Paths, Saddle Points, and Energy Barriers, 598
- pbacktrack
 - Deprecated Interface for Stochastic Backtracking, 583
- pbacktrack5
 - Deprecated Interface for Stochastic Backtracking, 584
- pbacktrack_circ
 - Deprecated Interface for Stochastic Backtracking, 584
- pf_circ_fold
 - Deprecated Interface for Global Partition Function Computation, 569
- pf_fold
 - Deprecated Interface for Global Partition Function Computation, 568
- pf_fold_par
 - Deprecated Interface for Global Partition Function Computation, 567
- pf_interact
 - Partition Function for two Hybridized Sequences as a Stepwise Process, 401
- pf_paramT
 - Energy Parameters, 208
- pf_scale
 - Fine-tuning of the Implemented Models, 204
- pf_unstru
 - Partition Function for two Hybridized Sequences as a Stepwise Process, 400
- pfl_fold
 - Deprecated Interface for Local (Sliding Window) Partition Function Computation, 580
- pfl_fold_par
 - Deprecated Interface for Local (Sliding Window) Partition Function Computation, 581
- plist
 - (Abstract) Data Structures, 509
- Plotting, 474
 - gmlRNA, 477
 - PS_dot_plot, 476
 - PS_dot_plot_list, 475
 - PS_rna_plot, 479
 - PS_rna_plot_a, 479
 - PS_rna_plot_a_gquad, 479
 - ssv_rna_plot, 478
 - svg_rna_plot, 478
 - vrna_file_PS_rnaplot, 476
 - vrna_file_PS_rnaplot_a, 477
 - xrna_plot, 479
- Post-transcriptional Modifications, 376

- vrna_sc_mod, 379
- vrna_sc_mod_7DA, 381
- vrna_sc_mod_dihydrouridine, 382
- vrna_sc_mod_inosine, 381
- vrna_sc_mod_json, 378
- vrna_sc_mod_jsonfile, 379
- vrna_sc_mod_m6A, 380
- vrna_sc_mod_param_t, 377
- vrna_sc_mod_parameters_free, 378
- vrna_sc_mod_pseudouridine, 380
- vrna_sc_mod_purine, 382
- vrna_sc_mod_read_from_json, 378
- vrna_sc_mod_read_from_jsonfile, 377
- Postorder_list, 601
- pr
 - fold_vars.h, 692
- Predicting various thermodynamic properties, 326
 - vrna_ensemble_defect, 330
 - vrna_ensemble_defect_pt, 329
 - vrna_heat_capacity, 333
 - vrna_heat_capacity_cb, 333
 - vrna_heat_capacity_f, 328
 - vrna_heat_capacity_simple, 334
 - vrna_heat_capacity_t, 328
 - vrna_mean_bp_distance, 329
 - vrna_mean_bp_distance_pr, 328
 - vrna_pf_dimer_probs, 331
 - vrna_positional_entropy, 331
 - vrna_pr_energy, 333
 - vrna_pr_structure, 332
 - vrna_stack_prob, 331
- print_energy
 - Suboptimal Structures within an Energy Band around the MFE, 291
- print_tty_constraint
 - hard.h, 637
- print_tty_constraint_full
 - hard.h, 637
- print_tty_input_seq
 - basic.h, 928
- print_tty_input_seq_str
 - basic.h, 928
- prod_cb
 - vrna_unstructured_domain_s, 220
- profile_edit_distance
 - profiledist.h, 1262
- profiledist.h
 - free_profile, 1263
 - Make_bp_profile, 1263
 - Make_bp_profile_bppm, 1263
 - profile_edit_distance, 1262
- progress_callback
 - Generate Soft Constraints from Data, 365
- PS_color_aln
 - Deprecated Interface for Plotting Utilities, 596
- PS_dot_plot
 - Plotting, 476
- PS_dot_plot_list
 - Plotting, 475
- PS_rna_plot
 - Plotting, 479
- PS_rna_plot_a
 - Plotting, 479
- PS_rna_plot_a_gquad
 - Plotting, 479
- pscore
 - vrna_fc_s, 523
- pscore_local
 - vrna_fc_s, 524
- pscore_pf_compat
 - vrna_fc_s, 524
- Pseudoknots, 369
 - vrna_pk_plex, 372
 - vrna_pk_plex_accessibility, 372
 - vrna_pk_plex_opt, 373
 - vrna_pk_plex_opt_defaults, 373
 - vrna_pk_plex_opt_fun, 373
 - vrna_pk_plex_opt_t, 371
 - vrna_pk_plex_score_f, 371
 - vrna_pk_plex_t, 371
- ptype
 - vrna_fc_s, 521
- ptype_pf_compat
 - vrna_fc_s, 522
- pu_contrib, 507
- pu_out, 508
- putoutpU_prob
 - Deprecated Interface for Local (Sliding Window) Partition Function Computation, 581
- putoutpU_prob_bin
 - Deprecated Interface for Local (Sliding Window) Partition Function Computation, 583
- Random Structure Samples from the Ensemble, 291
 - vrna_bs_result_f, 293
 - vrna_pbacktrack, 300
 - vrna_pbacktrack5, 294
 - vrna_pbacktrack5_cb, 296
 - vrna_pbacktrack5_num, 295
 - vrna_pbacktrack5_resume, 297
 - vrna_pbacktrack5_resume_cb, 298
 - vrna_pbacktrack_cb, 301
 - VRNA_PBACKTRACK_DEFAULT, 293
 - vrna_pbacktrack_mem_free, 311
 - vrna_pbacktrack_mem_t, 294
 - VRNA_PBACKTRACK_NON_REDUNDANT, 293
 - vrna_pbacktrack_num, 300
 - vrna_pbacktrack_resume, 302
 - vrna_pbacktrack_resume_cb, 304
 - vrna_pbacktrack_sub, 305
 - vrna_pbacktrack_sub_cb, 307
 - vrna_pbacktrack_sub_num, 306
 - vrna_pbacktrack_sub_resume, 308
 - vrna_pbacktrack_sub_resume_cb, 310
- random_string
 - strings.h, 1293
- read_parameter_file

- Reading/Writing Energy Parameter Sets from/to File, [407](#)
- read_record
 - Nucleic Acid Sequences and Structures, [463](#)
- Reading/Writing Energy Parameter Sets from/to File, [402](#)
 - last_parameter_file, [407](#)
 - read_parameter_file, [407](#)
 - VRNA_PARAMETER_FORMAT_DEFAULT, [403](#)
 - vrna_params_load, [403](#)
 - vrna_params_load_defaults, [404](#)
 - vrna_params_load_DNA_Mathews1999, [406](#)
 - vrna_params_load_DNA_Mathews2004, [406](#)
 - vrna_params_load_from_string, [404](#)
 - vrna_params_load_RNA_Andronescu2007, [405](#)
 - vrna_params_load_RNA_Langdon2018, [405](#)
 - vrna_params_load_RNA_misc_special_hairpins, [406](#)
 - vrna_params_load_RNA_Turner1999, [405](#)
 - vrna_params_load_RNA_Turner2004, [405](#)
 - vrna_params_save, [403](#)
 - write_parameter_file, [407](#)
- readribosum
 - Files and I/O, [456](#)
- RibosumFile
 - fold_vars.h, [691](#)
- RNA-RNA Interaction, [315](#)
- rna_plot_type
 - Deprecated Interface for Plotting Utilities, [598](#)
- S
 - vrna_fc_s, [523](#)
- S3
 - vrna_fc_s, [523](#)
- S5
 - vrna_fc_s, [523](#)
- S_cons
 - vrna_fc_s, [523](#)
- sc
 - vrna_fc_s, [522](#)
- scale
 - vrna_mx_pf_s, [532](#)
- scale_parameters
 - Energy Parameters, [215](#)
- scs
 - vrna_fc_s, [524](#)
- Search Algorithms, [496](#)
 - vrna_search_BM_BCT, [498](#)
 - vrna_search_BM_BCT_num, [497](#)
 - vrna_search_BMH, [497](#)
 - vrna_search_BMH_num, [496](#)
- Secondary Structure Utilities, [426](#)
 - vrna_db_from_bp_stack, [428](#)
 - vrna_db_from_probs, [427](#)
 - vrna_refBPcnt_matrix, [427](#)
 - vrna_refBPdist_matrix, [427](#)
- sect
 - (Abstract) Data Structures, [509](#)
- sequence
 - vrna_fc_s, [521](#)
- sequence_encoding
 - vrna_fc_s, [521](#)
- sequences
 - vrna_fc_s, [522](#)
- set_model_details
 - Fine-tuning of the Implemented Models, [203](#)
- SHAPE Reactivity Data, [360](#)
 - vrna_sc_add_SHAPE_deigan, [361](#)
 - vrna_sc_add_SHAPE_deigan_ali, [361](#)
 - vrna_sc_add_SHAPE_zarringhalam, [362](#)
 - vrna_sc_SHAPE_to_pr, [363](#)
- SHAPE.h
 - vrna_sc_SHAPE_parse_method, [643](#)
- simple_circplot_coordinates
 - Deprecated Interface for Plotting Utilities, [597](#)
- simple_xy_coordinates
 - Deprecated Interface for Plotting Utilities, [597](#)
- snoopT, [508](#)
- Soft Constraints, [248](#)
 - vrna_sc_add_bp, [254](#)
 - vrna_sc_add_bt, [257](#)
 - vrna_sc_add_data, [256](#)
 - vrna_sc_add_exp_f, [257](#)
 - vrna_sc_add_f, [256](#)
 - vrna_sc_add_up, [255](#)
 - vrna_sc_bt_f, [252](#)
 - vrna_sc_exp_f, [251](#)
 - vrna_sc_f, [251](#)
 - vrna_sc_free, [256](#)
 - vrna_sc_init, [253](#)
 - vrna_sc_remove, [255](#)
 - vrna_sc_set_bp, [253](#)
 - vrna_sc_set_up, [254](#)
- soft.h
 - VRNA_SC_DEFAULT, [646](#)
 - vrna_sc_type_e, [646](#)
 - VRNA_SC_WINDOW, [646](#)
- SOLUTION
 - subopt.h, [1279](#)
- space
 - basic.h, [928](#)
- ssv_rna_plot
 - Plotting, [478](#)
- st_back
 - Deprecated Interface for Stochastic Backtracking, [584](#)
- stackProb
 - Deprecated Interface for Global Partition Function Computation, [573](#)
- stat_cb
 - vrna_fc_s, [521](#)
- Stochastic Backtracking of Structures from Distance Based Partitioning, [324](#)
 - vrna_pbacktrack5_TwoD, [325](#)
 - vrna_pbacktrack_TwoD, [324](#)
- str_DNA2RNA
 - strings.h, [1293](#)

- str_uppercase
 - strings.h, [1293](#)
- strands
 - vrna_mx_mfe_s, [531](#)
- string_edit_distance
 - stringdist.h, [1277](#)
- stringdist.h
 - Make_swString, [1276](#)
 - string_edit_distance, [1277](#)
- strings.h
 - hamming, [1293](#)
 - hamming_bound, [1294](#)
 - random_string, [1293](#)
 - str_DNA2RNA, [1293](#)
 - str_uppercase, [1293](#)
- structure
 - vrna_ht_entry_db_t, [536](#)
- Structure Modules and Pseudoknots, [369](#)
- Structured Domains, [226](#)
- subopt
 - Suboptimal Structures within an Energy Band around the MFE, [290](#)
- subopt.h
 - SOLUTION, [1279](#)
- subopt_circ
 - Suboptimal Structures within an Energy Band around the MFE, [290](#)
- subopt_par
 - Suboptimal Structures within an Energy Band around the MFE, [290](#)
- subopt_sorted
 - Suboptimal Structures within an Energy Band around the MFE, [291](#)
- Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, [286](#)
 - vrna_subopt_zuker, [287](#)
 - zukersubopt, [286](#)
 - zukersubopt_par, [286](#)
- Suboptimal Structures within an Energy Band around the MFE, [287](#)
 - print_energy, [291](#)
 - subopt, [290](#)
 - subopt_circ, [290](#)
 - subopt_par, [290](#)
 - subopt_sorted, [291](#)
 - vrna_subopt, [288](#)
 - vrna_subopt_cb, [289](#)
 - vrna_subopt_result_f, [288](#)
- Suboptimals and Representative Structures, [285](#)
- svg_rna_plot
 - Plotting, [478](#)
- swString, [602](#)
- temperature
 - Fine-tuning of the Implemented Models, [203](#)
- tetra_loop
 - Fine-tuning of the Implemented Models, [204](#)
- The Dynamic Programming Matrices, [530](#)
 - VRNA_MX_2DFOLD, [533](#)
 - vrna_mx_add, [533](#)
 - VRNA_MX_DEFAULT, [533](#)
 - vrna_mx_mfe_free, [534](#)
 - vrna_mx_pf_free, [534](#)
 - vrna_mx_type_e, [532](#)
 - VRNA_MX_WINDOW, [533](#)
- The Fold Compound, [517](#)
 - vrna_auxdata_free_f, [526](#)
 - VRNA_FC_TYPE_COMPARATIVE, [527](#)
 - vrna_fc_type_e, [527](#)
 - VRNA_FC_TYPE_SINGLE, [527](#)
 - vrna_fold_compound, [527](#)
 - vrna_fold_compound_add_auxdata, [529](#)
 - vrna_fold_compound_add_callback, [530](#)
 - vrna_fold_compound_comparative, [528](#)
 - vrna_fold_compound_free, [529](#)
 - VRNA_OPTION_EVAL_ONLY, [525](#)
 - VRNA_OPTION_MFE, [525](#)
 - VRNA_OPTION_PF, [525](#)
 - vrna_recursion_status_f, [526](#)
 - VRNA_STATUS_MFE_POST, [524](#)
 - VRNA_STATUS_MFE_PRE, [524](#)
 - VRNA_STATUS_PF_POST, [525](#)
 - VRNA_STATUS_PF_PRE, [525](#)
- The RNA Folding Grammar, [174](#)
 - vrna_grammar_data_free_f, [175](#)
- The RNA Secondary Structure Landscape, [258](#)
- time_stamp
 - basic.h, [929](#)
- Tree, [602](#)
- Tree Representation of Secondary Structures, [443](#)
 - vrna_db_to_tree_string, [446](#)
 - VRNA_STRUCTURE_TREE_EXPANDED, [446](#)
 - VRNA_STRUCTURE_TREE_HIT, [445](#)
 - VRNA_STRUCTURE_TREE_SHAPIRO, [445](#)
 - VRNA_STRUCTURE_TREE_SHAPIRO_EXT, [445](#)
 - VRNA_STRUCTURE_TREE_SHAPIRO_SHORT, [445](#)
 - VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT, [445](#)
 - vrna_tree_string_to_db, [447](#)
 - vrna_tree_string_unweight, [447](#)
- tree_edit_distance
 - treedist.h, [1282](#)
- treedist.h
 - free_tree, [1282](#)
 - make_tree, [1281](#)
 - tree_edit_distance, [1282](#)
- TURN
 - constants.h, [933](#)
- TwoDfold_backtrack_f5
 - Computing MFE representatives of a Distance Based Partitioning, [322](#)
- TwoDfold_vars, [318](#)
 - Computing MFE representatives of a Distance Based Partitioning, [319](#)
- TwoDfoldList

- Computing MFE representatives of a Distance Based Partitioning, 321
- TwoDpfold_pbacktrack
 - 2Dpfold.h, 609
- TwoDpfold_pbacktrack5
 - 2Dpfold.h, 610
- TwoDpfold_vars, 602
- TwoDpfoldList
 - 2Dpfold.h, 609
- type
 - vrna_fc_s, 520
 - vrna_mx_mfe_s, 531
 - vrna_mx_pf_s, 532
 - vrna_path_s, 350
- unexpand_aligned_F
 - Deprecated Interface for Secondary Structure Utilities, 591
- unexpand_Full
 - Deprecated Interface for Secondary Structure Utilities, 591
- Unit Conversion, 513
 - vrna_convert_dcal_to_kcal, 516
 - vrna_convert_energy, 515
 - vrna_convert_kcal_to_dcal, 516
 - vrna_convert_temperature, 515
 - VRNA_UNIT_CAL, 514
 - VRNA_UNIT_CAL_IT, 514
 - VRNA_UNIT_DACAL, 514
 - VRNA_UNIT_DACAL_IT, 514
 - VRNA_UNIT_DEG_C, 515
 - VRNA_UNIT_DEG_DE, 515
 - VRNA_UNIT_DEG_F, 515
 - VRNA_UNIT_DEG_N, 515
 - VRNA_UNIT_DEG_R, 515
 - VRNA_UNIT_DEG_RE, 515
 - VRNA_UNIT_DEG_RO, 515
 - vrna_unit_energy_e, 514
 - VRNA_UNIT_EV, 514
 - VRNA_UNIT_G_TNT, 514
 - VRNA_UNIT_J, 514
 - VRNA_UNIT_K, 515
 - VRNA_UNIT_KCAL, 514
 - VRNA_UNIT_KCAL_IT, 514
 - VRNA_UNIT_KG_TNT, 514
 - VRNA_UNIT_KJ, 514
 - VRNA_UNIT_KWH, 514
 - VRNA_UNIT_T_TNT, 514
 - vrna_unit_temperature_e, 514
 - VRNA_UNIT_WH, 514
- unpack_structure
 - Deprecated Interface for Secondary Structure Utilities, 592
- Unstructured Domains, 217
 - vrna_ud_add_motif, 223
 - vrna_ud_add_probs_f, 221
 - vrna_ud_exp_f, 220
 - vrna_ud_exp_production_f, 221
 - vrna_ud_f, 220
 - vrna_ud_get_probs_f, 221
 - vrna_ud_motifs_centroid, 221
 - vrna_ud_motifs_MEA, 222
 - vrna_ud_motifs_MFE, 222
 - vrna_ud_production_f, 221
 - vrna_ud_remove, 224
 - vrna_ud_set_data, 224
 - vrna_ud_set_exp_prod_rule_cb, 225
 - vrna_ud_set_prod_rule_cb, 224
- unstructured_domains.h
 - vrna_ud_get_motif_size_at, 1286
 - vrna_ud_set_prob_cb, 1286
- unweight
 - Deprecated Interface for Secondary Structure Utilities, 591
- update_alifold_params
 - alifold.h, 615
- update_co_pf_params
 - Deprecated Interface for Global Partition Function Computation, 575
- update_co_pf_params_par
 - Deprecated Interface for Global Partition Function Computation, 576
- update_cofold_params
 - Deprecated Interface for Global MFE Prediction, 557
- update_cofold_params_par
 - Deprecated Interface for Global MFE Prediction, 557
- update_fold_params
 - Deprecated Interface for Global MFE Prediction, 561
- update_fold_params_par
 - Deprecated Interface for Global MFE Prediction, 561
- update_pf_params
 - Deprecated Interface for Global Partition Function Computation, 570
- update_pf_params_par
 - Deprecated Interface for Global Partition Function Computation, 570
- update_pf_paramsLP
 - Deprecated Interface for Local (Sliding Window) Partition Function Computation, 580
- urn
 - basic.h, 929
- Utilities, 383
 - get_input_line, 387
 - vrna_alloc, 385
 - vrna_idx_col_wise, 388
 - vrna_idx_row_wise, 388
 - vrna_init_rand, 386
 - vrna_init_rand_seed, 386
 - VRNA_INPUT_CONSTRAINT, 385
 - VRNA_INPUT_FASTA_HEADER, 385
 - vrna_int_urn, 387
 - vrna_realloc, 386
 - vrna_time_stamp, 387

- vrna_urn, [387](#)
- xsubi, [389](#)
- Utilities to deal with Nucleotide Alphabets, [412](#)
 - vrna_nucleotide_decode, [415](#)
 - vrna_nucleotide_encode, [414](#)
 - vrna_ptypes, [413](#)
 - VRNA_SEQ_DNA, [413](#)
 - vrna_seq_encode, [413](#)
 - vrna_seq_encode_simple, [414](#)
 - VRNA_SEQ_RNA, [413](#)
 - vrna_seq_type_e, [413](#)
 - VRNA_SEQ_UNKNOWN, [413](#)
- ViennaRNA/2Dfold.h, [605](#)
- ViennaRNA/2Dpfold.h, [607](#), [611](#)
- ViennaRNA/ali_plex.h, [613](#)
- ViennaRNA/alifold.h, [613](#), [616](#)
- ViennaRNA/aln_util.h, [617](#)
- ViennaRNA/alphabet.h, [617](#), [618](#)
- ViennaRNA/boltzmann_sampling.h, [619](#), [620](#)
- ViennaRNA/centroid.h, [622](#), [623](#)
- ViennaRNA/char_stream.h, [624](#)
- ViennaRNA/cofold.h, [627](#), [628](#)
- ViennaRNA/combinatorics.h, [628](#), [629](#)
- ViennaRNA/commands.h, [630](#), [631](#)
- ViennaRNA/concentrations.h, [631](#), [632](#)
- ViennaRNA/constraints.h, [633](#)
- ViennaRNA/constraints/basic.h, [915](#), [916](#)
- ViennaRNA/constraints/hard.h, [634](#), [638](#)
- ViennaRNA/constraints/ligand.h, [641](#)
- ViennaRNA/constraints/sc_cb_intern.h, [642](#)
- ViennaRNA/constraints/SHAPE.h, [643](#), [644](#)
- ViennaRNA/constraints/soft.h, [645](#), [646](#)
- ViennaRNA/constraints/soft_special.h, [649](#), [650](#)
- ViennaRNA/constraints_hard.h, [651](#)
- ViennaRNA/constraints_ligand.h, [651](#)
- ViennaRNA/constraints_SHAPE.h, [651](#), [652](#)
- ViennaRNA/constraints_soft.h, [652](#)
- ViennaRNA/convert_epars.h, [652](#)
- ViennaRNA/data_structures.h, [653](#)
- ViennaRNA/datastructures/array.h, [653](#), [654](#)
- ViennaRNA/datastructures/basic.h, [917](#), [919](#)
- ViennaRNA/datastructures/char_stream.h, [624](#), [625](#)
- ViennaRNA/datastructures/hash_tables.h, [655](#), [656](#)
- ViennaRNA/datastructures/heap.h, [657](#), [658](#)
- ViennaRNA/datastructures/lists.h, [659](#)
- ViennaRNA/datastructures/stream_output.h, [1274](#)
- ViennaRNA/datastructures/string.h, [660](#)
- ViennaRNA/dist_vars.h, [660](#), [661](#)
- ViennaRNA/dp_matrices.h, [662](#)
- ViennaRNA/duplex.h, [666](#)
- ViennaRNA/edit_cost.h, [666](#)
- ViennaRNA/energy_const.h, [667](#), [668](#)
- ViennaRNA/energy_par.h, [668](#)
- ViennaRNA/equilibrium_probs.h, [668](#), [669](#)
- ViennaRNA/eval.h, [670](#), [673](#)
- ViennaRNA/exterior_loops.h, [677](#)
- ViennaRNA/file_formats.h, [677](#)
- ViennaRNA/file_formats_msa.h, [680](#)
- ViennaRNA/file_utils.h, [682](#)
- ViennaRNA/findpath.h, [682](#), [683](#)
- ViennaRNA/fold.h, [684](#), [685](#)
- ViennaRNA/fold_compound.h, [687](#), [688](#)
- ViennaRNA/fold_vars.h, [690](#), [692](#)
- ViennaRNA/gquad.h, [692](#), [693](#)
- ViennaRNA/grammar.h, [712](#)
- ViennaRNA/hairpin_loops.h, [714](#)
- ViennaRNA/heat_capacity.h, [714](#), [715](#)
- ViennaRNA/interior_loops.h, [716](#)
- ViennaRNA/inverse.h, [716](#), [717](#)
- ViennaRNA/io/file_formats.h, [678](#)
- ViennaRNA/io/file_formats_msa.h, [680](#), [681](#)
- ViennaRNA/io/utils.h, [1289](#)
- ViennaRNA/landscape/findpath.h, [683](#)
- ViennaRNA/landscape/move.h, [717](#), [718](#)
- ViennaRNA/landscape/neighbor.h, [759](#)
- ViennaRNA/landscape/paths.h, [719](#), [720](#)
- ViennaRNA/landscape/walk.h, [1297](#), [1298](#)
- ViennaRNA/Lfold.h, [720](#), [721](#)
- ViennaRNA/loop_energies.h, [721](#), [722](#)
- ViennaRNA/loops/all.h, [722](#)
- ViennaRNA/loops/external.h, [722](#), [723](#)
- ViennaRNA/loops/hairpin.h, [725](#)
- ViennaRNA/loops/internal.h, [728](#), [729](#)
- ViennaRNA/loops/multibranch.h, [736](#), [737](#)
- ViennaRNA/LPfold.h, [739](#), [740](#)
- ViennaRNA/MEA.h, [741](#)
- ViennaRNA/mfe.h, [742](#), [743](#)
- ViennaRNA/mfe_window.h, [744](#)
- ViennaRNA/mm.h, [746](#), [747](#)
- ViennaRNA/model.h, [747](#), [752](#)
- ViennaRNA/move_set.h, [757](#)
- ViennaRNA/multibranch_loops.h, [758](#)
- ViennaRNA/naview.h, [758](#)
- ViennaRNA/neighbor.h, [760](#), [761](#)
- ViennaRNA/pair_mat.h, [761](#)
- ViennaRNA/params.h, [763](#)
- ViennaRNA/params/1.8.4_epars.h, [763](#), [764](#)
- ViennaRNA/params/1.8.4_intloops.h, [768](#)
- ViennaRNA/params/basic.h, [921](#), [923](#)
- ViennaRNA/params/constants.h, [932](#), [933](#)
- ViennaRNA/params/convert.h, [933](#), [934](#)
- ViennaRNA/params/default.h, [935](#)
- ViennaRNA/params/intl11.h, [936](#)
- ViennaRNA/params/intl11dH.h, [940](#)
- ViennaRNA/params/intl21.h, [945](#)
- ViennaRNA/params/intl21dH.h, [968](#)
- ViennaRNA/params/intl22.h, [991](#)
- ViennaRNA/params/intl22dH.h, [1106](#)
- ViennaRNA/params/io.h, [1221](#), [1222](#)
- ViennaRNA/params/salt.h, [1223](#)
- ViennaRNA/part_func.h, [1224](#), [1227](#)
- ViennaRNA/part_func_co.h, [1229](#), [1230](#)
- ViennaRNA/part_func_up.h, [1231](#), [1232](#)
- ViennaRNA/part_func_window.h, [1232](#), [1233](#)
- ViennaRNA/perturbation_fold.h, [1235](#), [1236](#)
- ViennaRNA/pf_multifold.h, [1236](#)

- ViennaRNA/pk_plex.h, 1236, 1237
- ViennaRNA/PKplex.h, 1238
- ViennaRNA/plex.h, 1239
- ViennaRNA/plot_aln.h, 1239, 1240
- ViennaRNA/plot_layouts.h, 1240
- ViennaRNA/plot_structure.h, 1240
- ViennaRNA/plot_utils.h, 1241
- ViennaRNA/plotting/alignments.h, 1241, 1242
- ViennaRNA/plotting/layouts.h, 1246, 1247
- ViennaRNA/plotting/probabilities.h, 1249
- ViennaRNA/plotting/RNApuzzler/RNApuzzler.h, 1251
- ViennaRNA/plotting/RNApuzzler/RNAturtle.h, 1252
- ViennaRNA/plotting/structures.h, 1253
- ViennaRNA/plotting/utils.h, 1290
- ViennaRNA/ProfileAln.h, 1262
- ViennaRNA/profiledist.h, 1262, 1263
- ViennaRNA/PS_dot.h, 1264
- ViennaRNA/read_epars.h, 1264, 1265
- ViennaRNA/ribo.h, 1265
- ViennaRNA/RNAstruct.h, 1265, 1266
- ViennaRNA/search/BoyerMoore.h, 1267, 1268
- ViennaRNA/sequence.h, 1268, 1269
- ViennaRNA/snofold.h, 1270
- ViennaRNA/snoop.h, 1271
- ViennaRNA/special_const.h, 1274
- ViennaRNA/stream_output.h, 1275
- ViennaRNA/string_utils.h, 1276
- ViennaRNA/stringdist.h, 1276, 1277
- ViennaRNA/structure_utils.h, 1277
- ViennaRNA/structured_domains.h, 1278
- ViennaRNA/subopt.h, 1278, 1279
- ViennaRNA/subopt_zuker.h, 1280
- ViennaRNA/svm_utils.h, 1281
- ViennaRNA/treedist.h, 1281, 1282
- ViennaRNA/units.h, 1283
- ViennaRNA/unstructured_domains.h, 1284, 1286
- ViennaRNA/utils.h, 1290, 1291
- ViennaRNA/utils/alignments.h, 1242, 1244
- ViennaRNA/utils/basic.h, 925, 930
- ViennaRNA/utils/cpu.h, 1291
- ViennaRNA/utils/higher_order_functions.h, 1291
- ViennaRNA/utils/strings.h, 1291, 1294
- ViennaRNA/utils/structures.h, 1254, 1258
- ViennaRNA/utils/svm.h, 1296
- ViennaRNA/utils/units.h, 1283, 1284
- ViennaRNA/vrna_config.h, 1296
- ViennaRNA/walk.h, 1299
- ViennaRNA/wrap_dlib.h, 1299
- ViennaRNA/zscore.h, 1299
- vrna__array_set_capacity
 - Arrays, 550
- vrna_abstract_shapes
 - Abstract Shapes Representation of Secondary Structures, 441
- vrna_abstract_shapes_pt
 - Abstract Shapes Representation of Secondary Structures, 442
- vrna_alifold
 - Global MFE Prediction, 263
- vrna_alignment_s, 413
- vrna_alloc
 - Utilities, 385
- vrna_aln_consensus_mis
 - Multiple Sequence Alignment Utilities, 455
- vrna_aln_consensus_sequence
 - Multiple Sequence Alignment Utilities, 454
- vrna_aln_conservation_col
 - Multiple Sequence Alignment Utilities, 454
- vrna_aln_conservation_struct
 - Multiple Sequence Alignment Utilities, 453
- vrna_aln_copy
 - Multiple Sequence Alignment Utilities, 453
- vrna_aln_free
 - Multiple Sequence Alignment Utilities, 452
- vrna_aln_mpi
 - Multiple Sequence Alignment Utilities, 451
- vrna_aln_pinfo
 - Multiple Sequence Alignment Utilities, 451
- vrna_aln_slice
 - Multiple Sequence Alignment Utilities, 451
- vrna_aln_toRNA
 - Multiple Sequence Alignment Utilities, 453
- vrna_aln_uppercase
 - Multiple Sequence Alignment Utilities, 452
- vrna_annotate_covar_db
 - Annotation, 493
- vrna_annotate_covar_pairs
 - Annotation, 494
- vrna_array_header_s, 549
- vrna_array_init_size
 - Arrays, 550
- vrna_auxdata_free_f
 - The Fold Compound, 526
- vrna_backtrack5
 - Backtracking MFE structures, 270
- vrna_backtrack5_TwoD
 - Computing MFE representatives of a Distance Based Partitioning, 320
- vrna_basepair_s, 507
- vrna_boustrophedon
 - Combinatorics Algorithms, 504
- vrna_boustrophedon_pos
 - Combinatorics Algorithms, 504
- vrna_bp_distance
 - Distance measures between Secondary Structures, 448
- vrna_bp_distance_pt
 - Distance measures between Secondary Structures, 448
- vrna_bp_stack_s, 507
- VRNA_BRACKETS_ALPHA
 - Dot-Bracket Notation of Secondary Structures, 429
- VRNA_BRACKETS_ANG
 - Dot-Bracket Notation of Secondary Structures, 430
- VRNA_BRACKETS_ANY
 - Dot-Bracket Notation of Secondary Structures, 430

- VRNA_BRACKETS_CLY
 - Dot-Bracket Notation of Secondary Structures, [429](#)
- VRNA_BRACKETS_DEFAULT
 - Dot-Bracket Notation of Secondary Structures, [430](#)
- VRNA_BRACKETS_RND
 - Dot-Bracket Notation of Secondary Structures, [429](#)
- VRNA_BRACKETS_SQR
 - Dot-Bracket Notation of Secondary Structures, [430](#)
- vrna_bs_result_f
 - Random Structure Samples from the Ensemble, [293](#)
- vrna_BT_hp_loop
 - Backtracking MFE structures, [270](#)
- vrna_BT_int_loop
 - Backtracking MFE structures, [271](#)
- vrna_BT_mb_loop
 - Backtracking MFE structures, [271](#)
- vrna_BT_stack
 - Backtracking MFE structures, [270](#)
- vrna_C11_features
 - (Abstract) Data Structures, [509](#)
- vrna_centroid
 - Compute the Centroid Structure, [314](#)
- vrna_centroid_from_plist
 - Compute the Centroid Structure, [314](#)
- vrna_centroid_from_probs
 - Compute the Centroid Structure, [315](#)
- vrna_circalifold
 - Global MFE Prediction, [264](#)
- vrna_circfold
 - Global MFE Prediction, [263](#)
- VRNA_CMD_PARSE_DEFAULTS
 - Command Files, [472](#)
- VRNA_CMD_PARSE_HC
 - Command Files, [471](#)
- VRNA_CMD_PARSE_SC
 - Command Files, [472](#)
- VRNA_CMD_PARSE_SD
 - Command Files, [472](#)
- VRNA_CMD_PARSE_UD
 - Command Files, [472](#)
- vrna_cofold
 - Global MFE Prediction, [264](#)
- vrna_color_s, [507](#)
- vrna_commands_apply
 - Command Files, [473](#)
- vrna_commands_free
 - Command Files, [474](#)
- VRNA_CONSTRAINT_CONTEXT_ALL_LOOPS
 - Hard Constraints, [245](#)
- VRNA_CONSTRAINT_CONTEXT_EXT_LOOP
 - Hard Constraints, [244](#)
- VRNA_CONSTRAINT_CONTEXT_HP_LOOP
 - Hard Constraints, [244](#)
- VRNA_CONSTRAINT_CONTEXT_INT_LOOP
 - Hard Constraints, [244](#)
- VRNA_CONSTRAINT_CONTEXT_INT_LOOP_ENC
 - Hard Constraints, [244](#)
- VRNA_CONSTRAINT_CONTEXT_MB_LOOP
 - Hard Constraints, [244](#)
- VRNA_CONSTRAINT_CONTEXT_MB_LOOP_ENC
 - Hard Constraints, [245](#)
- VRNA_CONSTRAINT_DB
 - Hard Constraints, [242](#)
- VRNA_CONSTRAINT_DB_ANG_BRACK
 - hard.h, [636](#)
- VRNA_CONSTRAINT_DB_DEFAULT
 - Hard Constraints, [244](#)
- VRNA_CONSTRAINT_DB_DOT
 - Hard Constraints, [242](#)
- VRNA_CONSTRAINT_DB_ENFORCE_BP
 - Hard Constraints, [242](#)
- VRNA_CONSTRAINT_DB_GQUAD
 - Hard Constraints, [243](#)
- VRNA_CONSTRAINT_DB_INTERMOL
 - Hard Constraints, [243](#)
- VRNA_CONSTRAINT_DB_INTRAMOL
 - Hard Constraints, [243](#)
- VRNA_CONSTRAINT_DB_PIPE
 - Hard Constraints, [242](#)
- VRNA_CONSTRAINT_DB_RND_BRACK
 - Hard Constraints, [243](#)
- VRNA_CONSTRAINT_DB_WUSS
 - Hard Constraints, [243](#)
- VRNA_CONSTRAINT_DB_X
 - Hard Constraints, [242](#)
- VRNA_CONSTRAINT_FILE
 - Constraining the RNA Folding Grammar, [229](#)
- VRNA_CONSTRAINT_MULTILINE
 - Nucleic Acid Sequences and Structures, [459](#)
- VRNA_CONSTRAINT_NO_HEADER
 - hard.h, [636](#)
- VRNA_CONSTRAINT_SOFT_MFE
 - Constraining the RNA Folding Grammar, [230](#)
- VRNA_CONSTRAINT_SOFT_PF
 - Constraining the RNA Folding Grammar, [230](#)
- vrna_constraints_add
 - Constraining the RNA Folding Grammar, [237](#)
- vrna_convert_dcal_to_kcal
 - Unit Conversion, [516](#)
- vrna_convert_energy
 - Unit Conversion, [515](#)
- vrna_convert_kcal_to_dcal
 - Unit Conversion, [516](#)
- VRNA_CONVERT_OUTPUT_ALL
 - Converting Energy Parameter Files, [408](#)
- VRNA_CONVERT_OUTPUT_BULGE
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_DANGLE3
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_DANGLE5
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_DUMP
 - Converting Energy Parameter Files, [411](#)
- VRNA_CONVERT_OUTPUT_HP
 - Converting Energy Parameter Files, [409](#)

- VRNA_CONVERT_OUTPUT_INT
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_INT_11
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_INT_21
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_INT_22
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_MISC
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_ML
 - Converting Energy Parameter Files, [410](#)
- VRNA_CONVERT_OUTPUT_MM_EXT
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_MM_HP
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_MM_INT
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_MM_INT_1N
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_MM_INT_23
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_MM_MULTI
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_NINIO
 - Converting Energy Parameter Files, [411](#)
- VRNA_CONVERT_OUTPUT_SPECIAL_HP
 - Converting Energy Parameter Files, [411](#)
- VRNA_CONVERT_OUTPUT_STACK
 - Converting Energy Parameter Files, [409](#)
- VRNA_CONVERT_OUTPUT_VANILLA
 - Converting Energy Parameter Files, [411](#)
- vrna_convert_temperature
 - Unit Conversion, [515](#)
- vrna_cpair_s, [507](#)
- vrna_cstr
 - Buffers, [551](#)
- vrna_cstr_close
 - Buffers, [552](#)
- vrna_cstr_discard
 - Buffers, [552](#)
- vrna_cstr_fflush
 - Buffers, [553](#)
- vrna_cstr_free
 - Buffers, [552](#)
- vrna_cut_point_insert
 - (Nucleic Acid Sequence) String Utilities, [425](#)
- vrna_cut_point_remove
 - (Nucleic Acid Sequence) String Utilities, [426](#)
- vrna_data_linear_s, [507](#)
- vrna_db_flatten
 - Dot-Bracket Notation of Secondary Structures, [431](#)
- vrna_db_flatten_to
 - Dot-Bracket Notation of Secondary Structures, [432](#)
- vrna_db_from_bp_stack
 - Secondary Structure Utilities, [428](#)
- vrna_db_from_plist
 - Dot-Bracket Notation of Secondary Structures, [433](#)
- vrna_db_from_probs
 - Secondary Structure Utilities, [427](#)
- vrna_db_from_ptable
 - Dot-Bracket Notation of Secondary Structures, [432](#)
- vrna_db_from_WUSS
 - Washington University Secondary Structure (WUSS) notation, [435](#)
- vrna_db_pack
 - Dot-Bracket Notation of Secondary Structures, [431](#)
- vrna_db_pk_remove
 - Dot-Bracket Notation of Secondary Structures, [434](#)
- vrna_db_to_element_string
 - Dot-Bracket Notation of Secondary Structures, [433](#)
- vrna_db_to_tree_string
 - Tree Representation of Secondary Structures, [446](#)
- vrna_db_unpack
 - Dot-Bracket Notation of Secondary Structures, [431](#)
- VRNA_DECOMP_EXT_EXT
 - Constraining the RNA Folding Grammar, [235](#)
- VRNA_DECOMP_EXT_EXT_EXT
 - Constraining the RNA Folding Grammar, [236](#)
- VRNA_DECOMP_EXT_EXT_STEM
 - Constraining the RNA Folding Grammar, [236](#)
- VRNA_DECOMP_EXT_EXT_STEM1
 - Constraining the RNA Folding Grammar, [237](#)
- VRNA_DECOMP_EXT_STEM
 - Constraining the RNA Folding Grammar, [235](#)
- VRNA_DECOMP_EXT_STEM_EXT
 - Constraining the RNA Folding Grammar, [236](#)
- VRNA_DECOMP_EXT_STEM_OUTSIDE
 - Constraining the RNA Folding Grammar, [236](#)
- VRNA_DECOMP_EXT_UP
 - Constraining the RNA Folding Grammar, [235](#)
- VRNA_DECOMP_ML_COAXIAL
 - Constraining the RNA Folding Grammar, [234](#)
- VRNA_DECOMP_ML_COAXIAL_ENC
 - Constraining the RNA Folding Grammar, [234](#)
- VRNA_DECOMP_ML_ML
 - Constraining the RNA Folding Grammar, [233](#)
- VRNA_DECOMP_ML_ML_ML
 - Constraining the RNA Folding Grammar, [232](#)
- VRNA_DECOMP_ML_ML_STEM
 - Constraining the RNA Folding Grammar, [233](#)
- VRNA_DECOMP_ML_STEM
 - Constraining the RNA Folding Grammar, [232](#)
- VRNA_DECOMP_ML_UP
 - Constraining the RNA Folding Grammar, [233](#)
- VRNA_DECOMP_PAIR_HP
 - Constraining the RNA Folding Grammar, [230](#)
- VRNA_DECOMP_PAIR_IL
 - Constraining the RNA Folding Grammar, [231](#)
- VRNA_DECOMP_PAIR_ML
 - Constraining the RNA Folding Grammar, [231](#)
- vrna_dimer_conc_s, [603](#)
- vrna_dimer_pf_s, [273](#)
- vrna_DNA_complement
 - (Nucleic Acid Sequence) String Utilities, [425](#)
- vrna_dotplot_auxdata_t, [475](#)

- vrna_E_ext_hp_loop
 - Hairpin Loops, [393](#)
- vrna_E_ext_stem
 - Exterior Loops, [390](#)
- vrna_E_hp_loop
 - Hairpin Loops, [392](#)
- vrna_E_mb_loop_stack
 - Multibranch Loops, [397](#)
- vrna_elem_prob_s, [440](#)
- vrna_ensemble_defect
 - Predicting various thermodynamic properties, [330](#)
- vrna_ensemble_defect_pt
 - Predicting various thermodynamic properties, [329](#)
- vrna_enumerate_necklaces
 - Combinatorics Algorithms, [499](#)
- vrna_eval_circ_consensus_structure
 - Free Energy Evaluation, [151](#)
- vrna_eval_circ_consensus_structure_v
 - Free Energy Evaluation, [154](#)
- vrna_eval_circ_gquad_consensus_structure
 - Free Energy Evaluation, [152](#)
- vrna_eval_circ_gquad_consensus_structure_v
 - Free Energy Evaluation, [156](#)
- vrna_eval_circ_gquad_structure
 - Free Energy Evaluation, [147](#)
- vrna_eval_circ_gquad_structure_v
 - Free Energy Evaluation, [150](#)
- vrna_eval_circ_structure
 - Free Energy Evaluation, [146](#)
- vrna_eval_circ_structure_v
 - Free Energy Evaluation, [149](#)
- vrna_eval_consensus_structure_pt_simple
 - Free Energy Evaluation, [158](#)
- vrna_eval_consensus_structure_pt_simple_v
 - Free Energy Evaluation, [159](#)
- vrna_eval_consensus_structure_pt_simple_verbose
 - Free Energy Evaluation, [159](#)
- vrna_eval_consensus_structure_simple
 - Free Energy Evaluation, [151](#)
- vrna_eval_consensus_structure_simple_v
 - Free Energy Evaluation, [154](#)
- vrna_eval_consensus_structure_simple_verbose
 - Free Energy Evaluation, [153](#)
- vrna_eval_covar_structure
 - Free Energy Evaluation, [142](#)
- vrna_eval_ext_stem
 - Exterior Loops, [391](#)
- vrna_eval_gquad_consensus_structure
 - Free Energy Evaluation, [152](#)
- vrna_eval_gquad_consensus_structure_v
 - Free Energy Evaluation, [155](#)
- vrna_eval_gquad_structure
 - Free Energy Evaluation, [146](#)
- vrna_eval_gquad_structure_v
 - Free Energy Evaluation, [149](#)
- vrna_eval_hp_loop
 - Hairpin Loops, [393](#)
- vrna_eval_int_loop
 - Internal Loops, [396](#)
- vrna_eval_loop_pt
 - Energy Evaluation for Individual Loops, [160](#)
- vrna_eval_loop_pt_v
 - Energy Evaluation for Individual Loops, [160](#)
- vrna_eval_move
 - Energy Evaluation for Atomic Moves, [161](#)
- vrna_eval_move_pt
 - Energy Evaluation for Atomic Moves, [162](#)
- vrna_eval_structure
 - Free Energy Evaluation, [142](#)
- vrna_eval_structure_pt
 - Free Energy Evaluation, [144](#)
- vrna_eval_structure_pt_simple
 - Free Energy Evaluation, [157](#)
- vrna_eval_structure_pt_simple_v
 - Free Energy Evaluation, [157](#)
- vrna_eval_structure_pt_simple_verbose
 - Free Energy Evaluation, [157](#)
- vrna_eval_structure_pt_v
 - Free Energy Evaluation, [145](#)
- vrna_eval_structure_pt_verbose
 - Free Energy Evaluation, [144](#)
- vrna_eval_structure_simple
 - Free Energy Evaluation, [145](#)
- vrna_eval_structure_simple_v
 - Free Energy Evaluation, [148](#)
- vrna_eval_structure_simple_verbose
 - Free Energy Evaluation, [148](#)
- vrna_eval_structure_v
 - Free Energy Evaluation, [143](#)
- vrna_eval_structure_verbose
 - Free Energy Evaluation, [143](#)
- vrna_exp_E_ext_stem
 - Exterior Loops, [391](#)
- vrna_exp_E_hp_loop
 - Hairpin Loops, [395](#)
- vrna_exp_param_s, [207](#)
 - alpha, [208](#)
 - id, [208](#)
- vrna_exp_params
 - Energy Parameters, [209](#)
- vrna_exp_params_comparative
 - Energy Parameters, [209](#)
- vrna_exp_params_copy
 - Energy Parameters, [210](#)
- vrna_exp_params_rescale
 - Energy Parameters, [211](#)
- vrna_exp_params_reset
 - Energy Parameters, [213](#)
- vrna_exp_params_subst
 - Energy Parameters, [211](#)
- vrna_extract_record_rest_constraint
 - Nucleic Acid Sequences and Structures, [463](#)
- vrna_extract_record_rest_structure
 - Nucleic Acid Sequences and Structures, [462](#)
- vrna_fc_s, [518](#)
 - auxdata, [521](#)

- cons_seq, [523](#)
- free_auxdata, [521](#)
- n_seq, [522](#)
- pscore, [523](#)
- pscore_local, [524](#)
- pscore_pf_compat, [524](#)
- pptype, [521](#)
- pptype_pf_compat, [522](#)
- S, [523](#)
- S3, [523](#)
- S5, [523](#)
- S_cons, [523](#)
- sc, [522](#)
- scs, [524](#)
- sequence, [521](#)
- sequence_encoding, [521](#)
- sequences, [522](#)
- stat_cb, [521](#)
- type, [520](#)
- VRNA_FC_TYPE_COMPARATIVE
 - The Fold Compound, [527](#)
- vrna_fc_type_e
 - The Fold Compound, [527](#)
- VRNA_FC_TYPE_SINGLE
 - The Fold Compound, [527](#)
- vrna_file_bpseq
 - Nucleic Acid Sequences and Structures, [460](#)
- vrna_file_commands_apply
 - Command Files, [473](#)
- vrna_file_commands_read
 - Command Files, [472](#)
- vrna_file_connect
 - Nucleic Acid Sequences and Structures, [460](#)
- vrna_file_exists
 - Files and I/O, [458](#)
- vrna_file_fasta_read_record
 - Nucleic Acid Sequences and Structures, [461](#)
- VRNA_FILE_FORMAT_MSA_APPEND
 - Multiple Sequence Alignments, [466](#)
- VRNA_FILE_FORMAT_MSA_CLUSTAL
 - Multiple Sequence Alignments, [465](#)
- VRNA_FILE_FORMAT_MSA_DEFAULT
 - Multiple Sequence Alignments, [466](#)
- VRNA_FILE_FORMAT_MSA_FASTA
 - Multiple Sequence Alignments, [465](#)
- VRNA_FILE_FORMAT_MSA_MAF
 - Multiple Sequence Alignments, [465](#)
- VRNA_FILE_FORMAT_MSA_MIS
 - Multiple Sequence Alignments, [465](#)
- VRNA_FILE_FORMAT_MSA_NOCHECK
 - Multiple Sequence Alignments, [466](#)
- VRNA_FILE_FORMAT_MSA_QUIET
 - Multiple Sequence Alignments, [466](#)
- VRNA_FILE_FORMAT_MSA_SILENT
 - Multiple Sequence Alignments, [467](#)
- VRNA_FILE_FORMAT_MSA_STOCKHOLM
 - Multiple Sequence Alignments, [465](#)
- VRNA_FILE_FORMAT_MSA_UNKNOWN
 - Multiple Sequence Alignments, [466](#)
- vrna_file_helixlist
 - Nucleic Acid Sequences and Structures, [459](#)
- vrna_file_json
 - Nucleic Acid Sequences and Structures, [460](#)
- vrna_file_msa_detect_format
 - Multiple Sequence Alignments, [469](#)
- vrna_file_msa_read
 - Multiple Sequence Alignments, [467](#)
- vrna_file_msa_read_record
 - Multiple Sequence Alignments, [468](#)
- vrna_file_msa_write
 - Multiple Sequence Alignments, [470](#)
- vrna_file_PS_aln
 - Alignment Plots, [494](#)
- vrna_file_PS_aln_slice
 - Alignment Plots, [495](#)
- vrna_file_PS_rnaplot
 - Plotting, [476](#)
- vrna_file_PS_rnaplot_a
 - Plotting, [477](#)
- vrna_file_SHAPE_read
 - Nucleic Acid Sequences and Structures, [463](#)
- vrna_filename_sanitize
 - Files and I/O, [457](#)
- vrna_fold
 - Global MFE Prediction, [262](#)
- vrna_fold_compound
 - The Fold Compound, [527](#)
- vrna_fold_compound_add_auxdata
 - The Fold Compound, [529](#)
- vrna_fold_compound_add_callback
 - The Fold Compound, [530](#)
- vrna_fold_compound_comparative
 - The Fold Compound, [528](#)
- vrna_fold_compound_free
 - The Fold Compound, [529](#)
- vrna_gr_aux_s, [175](#)
- vrna_grammar_data_free_f
 - The RNA Folding Grammar, [175](#)
- vrna_hamming_distance
 - (Nucleic Acid Sequence) String Utilities, [422](#)
- vrna_hamming_distance_bound
 - (Nucleic Acid Sequence) String Utilities, [422](#)
- vrna_hash_table_t
 - Hash Tables, [536](#)
- vrna_hc_add_bp
 - Hard Constraints, [247](#)
- vrna_hc_add_bp_nonspecific
 - Hard Constraints, [247](#)
- vrna_hc_add_data
 - hard.h, [637](#)
- vrna_hc_add_from_db
 - Hard Constraints, [248](#)
- vrna_hc_add_up
 - Hard Constraints, [246](#)
- vrna_hc_add_up_batch
 - Hard Constraints, [246](#)

- VRNA_HC_DEFAULT
 - hard.h, [636](#)
- vrna_hc_eval_f
 - Hard Constraints, [245](#)
- vrna_hc_free
 - Hard Constraints, [248](#)
- vrna_hc_init
 - Hard Constraints, [246](#)
- vrna_hc_s, [241](#)
 - free_data, [241](#)
- vrna_hc_type_e
 - hard.h, [636](#)
- vrna_hc_up_s, [241](#)
- VRNA_HC_WINDOW
 - hard.h, [636](#)
- vrna_heap_cmp_f
 - Heaps, [542](#)
- vrna_heap_free
 - Heaps, [545](#)
- vrna_heap_get_pos_f
 - Heaps, [544](#)
- vrna_heap_init
 - Heaps, [544](#)
- vrna_heap_insert
 - Heaps, [546](#)
- vrna_heap_pop
 - Heaps, [546](#)
- vrna_heap_remove
 - Heaps, [547](#)
- vrna_heap_set_pos_f
 - Heaps, [544](#)
- vrna_heap_size
 - Heaps, [545](#)
- vrna_heap_t
 - Heaps, [542](#)
- vrna_heap_top
 - Heaps, [546](#)
- vrna_heap_update
 - Heaps, [547](#)
- vrna_heat_capacity
 - Predicting various thermodynamic properties, [333](#)
- vrna_heat_capacity_cb
 - Predicting various thermodynamic properties, [333](#)
- vrna_heat_capacity_f
 - Predicting various thermodynamic properties, [328](#)
- vrna_heat_capacity_s, [327](#)
- vrna_heat_capacity_simple
 - Predicting various thermodynamic properties, [334](#)
- vrna_heat_capacity_t
 - Predicting various thermodynamic properties, [328](#)
- vrna_ht_clear
 - Hash Tables, [539](#)
- vrna_ht_cmp_f
 - Hash Tables, [536](#)
- vrna_ht_collisions
 - Hash Tables, [538](#)
- vrna_ht_db_comp
 - Hash Tables, [540](#)
- vrna_ht_db_free_entry
 - Hash Tables, [541](#)
- vrna_ht_db_hash_func
 - Hash Tables, [540](#)
- vrna_ht_entry_db_t, [535](#)
 - energy, [536](#)
 - structure, [536](#)
- vrna_ht_free
 - Hash Tables, [540](#)
- vrna_ht_free_f
 - Hash Tables, [537](#)
- vrna_ht_get
 - Hash Tables, [538](#)
- vrna_ht_hashfunc_f
 - Hash Tables, [536](#)
- vrna_ht_init
 - Hash Tables, [537](#)
- vrna_ht_insert
 - Hash Tables, [539](#)
- vrna_ht_remove
 - Hash Tables, [539](#)
- vrna_ht_size
 - Hash Tables, [538](#)
- vrna_hx_from_ptable
 - Helix List Representation of Secondary Structures, [443](#)
- vrna_hx_s, [443](#)
- vrna_idx_col_wise
 - Utilities, [388](#)
- vrna_idx_row_wise
 - Utilities, [388](#)
- vrna_init_rand
 - Utilities, [386](#)
- vrna_init_rand_seed
 - Utilities, [386](#)
- VRNA_INPUT_CONSTRAINT
 - Utilities, [385](#)
- VRNA_INPUT_FASTA_HEADER
 - Utilities, [385](#)
- vrna_int_urn
 - Utilities, [387](#)
- vrna_Lfold
 - Local (sliding window) MFE Prediction, [268](#)
- vrna_Lfoldz
 - Local (sliding window) MFE Prediction, [268](#)
- vrna_loopidx_update
 - Neighborhood Relation and Move Sets for Secondary Structures, [345](#)
- vrna_maximum_matching
 - mm.h, [746](#)
- vrna_maximum_matching_simple
 - mm.h, [746](#)
- vrna_md_copy
 - Fine-tuning of the Implemented Models, [187](#)
- vrna_md_defaults_backtrack
 - Fine-tuning of the Implemented Models, [195](#)
- vrna_md_defaults_backtrack_get
 - Fine-tuning of the Implemented Models, [195](#)

- vrna_md_defaults_backtrack_type
 - Fine-tuning of the Implemented Models, 196
- vrna_md_defaults_backtrack_type_get
 - Fine-tuning of the Implemented Models, 196
- vrna_md_defaults_betaScale
 - Fine-tuning of the Implemented Models, 189
- vrna_md_defaults_betaScale_get
 - Fine-tuning of the Implemented Models, 189
- vrna_md_defaults_circ
 - Fine-tuning of the Implemented Models, 193
- vrna_md_defaults_circ_get
 - Fine-tuning of the Implemented Models, 193
- vrna_md_defaults_compute_bpp
 - Fine-tuning of the Implemented Models, 196
- vrna_md_defaults_compute_bpp_get
 - Fine-tuning of the Implemented Models, 197
- vrna_md_defaults_cv_fact
 - Fine-tuning of the Implemented Models, 200
- vrna_md_defaults_cv_fact_get
 - Fine-tuning of the Implemented Models, 200
- vrna_md_defaults_dangles
 - Fine-tuning of the Implemented Models, 189
- vrna_md_defaults_dangles_get
 - Fine-tuning of the Implemented Models, 190
- vrna_md_defaults_energy_set
 - Fine-tuning of the Implemented Models, 195
- vrna_md_defaults_energy_set_get
 - Fine-tuning of the Implemented Models, 195
- vrna_md_defaults_gquad
 - Fine-tuning of the Implemented Models, 193
- vrna_md_defaults_gquad_get
 - Fine-tuning of the Implemented Models, 194
- vrna_md_defaults_logML
 - Fine-tuning of the Implemented Models, 192
- vrna_md_defaults_logML_get
 - Fine-tuning of the Implemented Models, 193
- vrna_md_defaults_max_bp_span
 - Fine-tuning of the Implemented Models, 197
- vrna_md_defaults_max_bp_span_get
 - Fine-tuning of the Implemented Models, 197
- vrna_md_defaults_min_loop_size
 - Fine-tuning of the Implemented Models, 197
- vrna_md_defaults_min_loop_size_get
 - Fine-tuning of the Implemented Models, 198
- vrna_md_defaults_nc_fact
 - Fine-tuning of the Implemented Models, 200
- vrna_md_defaults_nc_fact_get
 - Fine-tuning of the Implemented Models, 201
- vrna_md_defaults_noGU
 - Fine-tuning of the Implemented Models, 191
- vrna_md_defaults_noGU_get
 - Fine-tuning of the Implemented Models, 191
- vrna_md_defaults_noGUclosure
 - Fine-tuning of the Implemented Models, 192
- vrna_md_defaults_noGUclosure_get
 - Fine-tuning of the Implemented Models, 192
- vrna_md_defaults_noLP
 - Fine-tuning of the Implemented Models, 191
- vrna_md_defaults_noLP_get
 - Fine-tuning of the Implemented Models, 191
- vrna_md_defaults_oldAliEn
 - Fine-tuning of the Implemented Models, 199
- vrna_md_defaults_oldAliEn_get
 - Fine-tuning of the Implemented Models, 199
- vrna_md_defaults_reset
 - Fine-tuning of the Implemented Models, 188
- vrna_md_defaults_ribo
 - Fine-tuning of the Implemented Models, 199
- vrna_md_defaults_ribo_get
 - Fine-tuning of the Implemented Models, 199
- vrna_md_defaults_salt
 - Fine-tuning of the Implemented Models, 201
- vrna_md_defaults_salt_get
 - Fine-tuning of the Implemented Models, 202
- vrna_md_defaults_saltDPXInit
 - Fine-tuning of the Implemented Models, 203
- vrna_md_defaults_saltDPXInit_get
 - Fine-tuning of the Implemented Models, 203
- vrna_md_defaults_saltMLLower
 - Fine-tuning of the Implemented Models, 202
- vrna_md_defaults_saltMLLower_get
 - Fine-tuning of the Implemented Models, 202
- vrna_md_defaults_saltMLUpper
 - Fine-tuning of the Implemented Models, 202
- vrna_md_defaults_saltMLUpper_get
 - Fine-tuning of the Implemented Models, 202
- vrna_md_defaults_sfact
 - Fine-tuning of the Implemented Models, 201
- vrna_md_defaults_sfact_get
 - Fine-tuning of the Implemented Models, 201
- vrna_md_defaults_special_hp
 - Fine-tuning of the Implemented Models, 190
- vrna_md_defaults_special_hp_get
 - Fine-tuning of the Implemented Models, 190
- vrna_md_defaults_temperature
 - Fine-tuning of the Implemented Models, 188
- vrna_md_defaults_temperature_get
 - Fine-tuning of the Implemented Models, 189
- vrna_md_defaults_uniq_ML
 - Fine-tuning of the Implemented Models, 194
- vrna_md_defaults_uniq_ML_get
 - Fine-tuning of the Implemented Models, 194
- vrna_md_defaults_window_size
 - Fine-tuning of the Implemented Models, 198
- vrna_md_defaults_window_size_get
 - Fine-tuning of the Implemented Models, 198
- vrna_md_option_string
 - Fine-tuning of the Implemented Models, 188
- vrna_md_s, 180
 - dangles, 182
 - min_loop_size, 183
- vrna_md_set_default
 - Fine-tuning of the Implemented Models, 187
- vrna_md_update
 - Fine-tuning of the Implemented Models, 187
- vrna_MEA

- Compute the Structure with Maximum Expected Accuracy (MEA), [312](#)
- `vrna_MEA_from_plist`
 - Compute the Structure with Maximum Expected Accuracy (MEA), [313](#)
- `vrna_mean_bp_distance`
 - Predicting various thermodynamic properties, [329](#)
- `vrna_mean_bp_distance_pr`
 - Predicting various thermodynamic properties, [328](#)
- `VRNA_MEASURE_SHANNON_ENTROPY`
 - Multiple Sequence Alignment Utilities, [451](#)
- `vrna_message_constraint_options`
 - Constraining the RNA Folding Grammar, [238](#)
- `vrna_message_constraint_options_all`
 - Constraining the RNA Folding Grammar, [239](#)
- `vrna_message_error`
 - Messages, [511](#)
- `vrna_message_info`
 - Messages, [512](#)
- `vrna_message_input_seq`
 - Messages, [513](#)
- `vrna_message_input_seq_simple`
 - Messages, [513](#)
- `vrna_message_verror`
 - Messages, [511](#)
- `vrna_message_vinfo`
 - Messages, [512](#)
- `vrna_message_vwarning`
 - Messages, [512](#)
- `vrna_message_warning`
 - Messages, [511](#)
- `vrna_mfe`
 - Global MFE Prediction, [261](#)
- `vrna_mfe_dimer`
 - Global MFE Prediction, [261](#)
- `vrna_mfe_TwoD`
 - Computing MFE representatives of a Distance Based Partitioning, [319](#)
- `vrna_mfe_window`
 - Local (sliding window) MFE Prediction, [267](#)
- `vrna_mfe_window_f`
 - Local (sliding window) MFE Prediction, [266](#)
- `vrna_mfe_window_zscore`
 - Local (sliding window) MFE Prediction, [267](#)
- `VRNA_MINIMIZER_CONJUGATE_FR`
 - Generate Soft Constraints from Data, [364](#)
- `VRNA_MINIMIZER_CONJUGATE_PR`
 - Generate Soft Constraints from Data, [365](#)
- `VRNA_MINIMIZER_DEFAULT`
 - Generate Soft Constraints from Data, [364](#)
- `VRNA_MINIMIZER_STEEPEST_DESCENT`
 - Generate Soft Constraints from Data, [365](#)
- `VRNA_MINIMIZER_VECTOR_BFGS`
 - Generate Soft Constraints from Data, [365](#)
- `VRNA_MINIMIZER_VECTOR_BFGS2`
 - Generate Soft Constraints from Data, [365](#)
- `VRNA_MODEL_DEFAULT_ALI_CV_FACT`
 - Fine-tuning of the Implemented Models, [186](#)
- `VRNA_MODEL_DEFAULT_ALI_NC_FACT`
 - Fine-tuning of the Implemented Models, [186](#)
- `VRNA_MODEL_DEFAULT_ALI_OLD_EN`
 - Fine-tuning of the Implemented Models, [186](#)
- `VRNA_MODEL_DEFAULT_ALI_RIBO`
 - Fine-tuning of the Implemented Models, [186](#)
- `VRNA_MODEL_DEFAULT_BACKTRACK`
 - Fine-tuning of the Implemented Models, [185](#)
- `VRNA_MODEL_DEFAULT_BACKTRACK_TYPE`
 - Fine-tuning of the Implemented Models, [185](#)
- `VRNA_MODEL_DEFAULT_BETA_SCALE`
 - Fine-tuning of the Implemented Models, [183](#)
- `VRNA_MODEL_DEFAULT_CIRC`
 - Fine-tuning of the Implemented Models, [184](#)
- `VRNA_MODEL_DEFAULT_COMPUTE_BPP`
 - Fine-tuning of the Implemented Models, [185](#)
- `VRNA_MODEL_DEFAULT_DANGLES`
 - Fine-tuning of the Implemented Models, [183](#)
- `VRNA_MODEL_DEFAULT_ENERGY_SET`
 - Fine-tuning of the Implemented Models, [185](#)
- `VRNA_MODEL_DEFAULT_GQUAD`
 - Fine-tuning of the Implemented Models, [184](#)
- `VRNA_MODEL_DEFAULT_LOG_ML`
 - Fine-tuning of the Implemented Models, [186](#)
- `VRNA_MODEL_DEFAULT_MAX_BP_SPAN`
 - Fine-tuning of the Implemented Models, [185](#)
- `VRNA_MODEL_DEFAULT_NO_GU`
 - Fine-tuning of the Implemented Models, [184](#)
- `VRNA_MODEL_DEFAULT_NO_GU_CLOSURE`
 - Fine-tuning of the Implemented Models, [184](#)
- `VRNA_MODEL_DEFAULT_NO_LP`
 - Fine-tuning of the Implemented Models, [184](#)
- `VRNA_MODEL_DEFAULT_PF_SCALE`
 - Fine-tuning of the Implemented Models, [183](#)
- `VRNA_MODEL_DEFAULT_SPECIAL_HP`
 - Fine-tuning of the Implemented Models, [184](#)
- `VRNA_MODEL_DEFAULT_TEMPERATURE`
 - Fine-tuning of the Implemented Models, [183](#)
- `VRNA_MODEL_DEFAULT_UNIQ_ML`
 - Fine-tuning of the Implemented Models, [185](#)
- `VRNA_MODEL_DEFAULT_WINDOW_SIZE`
 - Fine-tuning of the Implemented Models, [186](#)
- `vrna_move_apply`
 - Neighborhood Relation and Move Sets for Secondary Structures, [343](#)
- `vrna_move_compare`
 - Neighborhood Relation and Move Sets for Secondary Structures, [344](#)
- `vrna_move_init`
 - Neighborhood Relation and Move Sets for Secondary Structures, [343](#)
- `vrna_move_is_insertion`
 - Neighborhood Relation and Move Sets for Secondary Structures, [344](#)
- `vrna_move_is_removal`
 - Neighborhood Relation and Move Sets for Secondary Structures, [343](#)
- `vrna_move_is_shift`

- Neighborhood Relation and Move Sets for Secondary Structures, [344](#)
- `vrna_move_list_free`
 - Neighborhood Relation and Move Sets for Secondary Structures, [343](#)
- `vrna_move_neighbor_diff`
 - Neighborhood Relation and Move Sets for Secondary Structures, [347](#)
- `vrna_move_neighbor_diff_cb`
 - Neighborhood Relation and Move Sets for Secondary Structures, [346](#)
- `vrna_move_s`, [340](#)
- `vrna_move_update_f`
 - Neighborhood Relation and Move Sets for Secondary Structures, [342](#)
- `VRNA_MOVESET_DEFAULT`
 - Neighborhood Relation and Move Sets for Secondary Structures, [341](#)
- `VRNA_MOVESET_DELETION`
 - Neighborhood Relation and Move Sets for Secondary Structures, [341](#)
- `VRNA_MOVESET_INSERTION`
 - Neighborhood Relation and Move Sets for Secondary Structures, [341](#)
- `VRNA_MOVESET_NO_LP`
 - Neighborhood Relation and Move Sets for Secondary Structures, [341](#)
- `VRNA_MOVESET_SHIFT`
 - Neighborhood Relation and Move Sets for Secondary Structures, [341](#)
- `vrna_multimer_pf_s`, [273](#)
- `VRNA_MX_2DFOLD`
 - The Dynamic Programming Matrices, [533](#)
- `vrna_mx_add`
 - The Dynamic Programming Matrices, [533](#)
- `VRNA_MX_DEFAULT`
 - The Dynamic Programming Matrices, [533](#)
- `vrna_mx_mfe_free`
 - The Dynamic Programming Matrices, [534](#)
- `vrna_mx_mfe_s`, [531](#)
 - strands, [531](#)
 - type, [531](#)
- `vrna_mx_pf_aux_el_t`
 - Exterior Loops, [390](#)
- `vrna_mx_pf_aux_ml_t`
 - Multibranch Loops, [397](#)
- `vrna_mx_pf_free`
 - The Dynamic Programming Matrices, [534](#)
- `vrna_mx_pf_s`, [532](#)
 - expMLbase, [532](#)
 - length, [532](#)
 - scale, [532](#)
 - type, [532](#)
- `vrna_mx_type_e`
 - The Dynamic Programming Matrices, [532](#)
- `VRNA_MX_WINDOW`
 - The Dynamic Programming Matrices, [533](#)
- `vrna_n_multichoose_k`
 - Combinatorics Algorithms, [503](#)
- `VRNA_NEIGHBOR_CHANGE`
 - Neighborhood Relation and Move Sets for Secondary Structures, [342](#)
- `VRNA_NEIGHBOR_INVALID`
 - Neighborhood Relation and Move Sets for Secondary Structures, [342](#)
- `VRNA_NEIGHBOR_NEW`
 - Neighborhood Relation and Move Sets for Secondary Structures, [342](#)
- `vrna_neighbors`
 - Neighborhood Relation and Move Sets for Secondary Structures, [345](#)
- `vrna_neighbors_successive`
 - Neighborhood Relation and Move Sets for Secondary Structures, [346](#)
- `vrna_nucleotide_decode`
 - Utilities to deal with Nucleotide Alphabets, [415](#)
- `vrna_nucleotide_encode`
 - Utilities to deal with Nucleotide Alphabets, [414](#)
- `VRNA_OBJECTIVE_FUNCTION_ABSOLUTE`
 - Generate Soft Constraints from Data, [364](#)
- `VRNA_OBJECTIVE_FUNCTION_QUADRATIC`
 - Generate Soft Constraints from Data, [364](#)
- `VRNA_OPTION_EVAL_ONLY`
 - The Fold Compound, [525](#)
- `VRNA_OPTION_MFE`
 - The Fold Compound, [525](#)
- `VRNA_OPTION_MULTILINE`
 - Nucleic Acid Sequences and Structures, [459](#)
- `VRNA_OPTION_PF`
 - The Fold Compound, [525](#)
- `vrna_ostream_free`
 - Buffers, [553](#)
- `vrna_ostream_init`
 - Buffers, [553](#)
- `vrna_ostream_provide`
 - Buffers, [554](#)
- `vrna_ostream_request`
 - Buffers, [554](#)
- `vrna_param_s`, [207](#)
- `VRNA_PARAMETER_FORMAT_DEFAULT`
 - Reading/Writing Energy Parameter Sets from/to File, [403](#)
- `vrna_params`
 - Energy Parameters, [208](#)
- `vrna_params_copy`
 - Energy Parameters, [209](#)
- `vrna_params_load`
 - Reading/Writing Energy Parameter Sets from/to File, [403](#)
- `vrna_params_load_defaults`
 - Reading/Writing Energy Parameter Sets from/to File, [404](#)
- `vrna_params_load_DNA_Mathews1999`
 - Reading/Writing Energy Parameter Sets from/to File, [406](#)
- `vrna_params_load_DNA_Mathews2004`

- Reading/Writing Energy Parameter Sets from/to File, [406](#)
- `vrna_params_load_from_string`
 - Reading/Writing Energy Parameter Sets from/to File, [404](#)
- `vrna_params_load_RNA_Andronescu2007`
 - Reading/Writing Energy Parameter Sets from/to File, [405](#)
- `vrna_params_load_RNA_Langdon2018`
 - Reading/Writing Energy Parameter Sets from/to File, [405](#)
- `vrna_params_load_RNA_misc_special_hairpins`
 - Reading/Writing Energy Parameter Sets from/to File, [406](#)
- `vrna_params_load_RNA_Turner1999`
 - Reading/Writing Energy Parameter Sets from/to File, [405](#)
- `vrna_params_load_RNA_Turner2004`
 - Reading/Writing Energy Parameter Sets from/to File, [405](#)
- `vrna_params_reset`
 - Energy Parameters, [212](#)
- `vrna_params_save`
 - Reading/Writing Energy Parameter Sets from/to File, [403](#)
- `vrna_params_subst`
 - Energy Parameters, [210](#)
- `vrna_path`
 - Folding Paths that start at a single Secondary Structure, [357](#)
- `VRNA_PATH_DEFAULT`
 - Folding Paths that start at a single Secondary Structure, [357](#)
- `vrna_path_direct`
 - Direct Refolding Paths between two Secondary Structures, [355](#)
- `vrna_path_direct_ub`
 - Direct Refolding Paths between two Secondary Structures, [355](#)
- `vrna_path_findpath`
 - Direct Refolding Paths between two Secondary Structures, [353](#)
- `vrna_path_findpath_saddle`
 - Direct Refolding Paths between two Secondary Structures, [351](#)
- `vrna_path_findpath_saddle_ub`
 - Direct Refolding Paths between two Secondary Structures, [352](#)
- `vrna_path_findpath_ub`
 - Direct Refolding Paths between two Secondary Structures, [353](#)
- `vrna_path_free`
 - (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [350](#)
- `vrna_path_gradient`
 - Folding Paths that start at a single Secondary Structure, [358](#)
- `VRNA_PATH_NO_TRANSITION_OUTPUT`
 - Folding Paths that start at a single Secondary Structure, [357](#)
- `vrna_path_options_findpath`
 - Direct Refolding Paths between two Secondary Structures, [354](#)
- `vrna_path_options_free`
 - (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [350](#)
- `VRNA_PATH_RANDOM`
 - Folding Paths that start at a single Secondary Structure, [357](#)
- `vrna_path_random`
 - Folding Paths that start at a single Secondary Structure, [359](#)
- `vrna_path_s`, [349](#)
 - type, [350](#)
- `VRNA_PATH_STEEPEST_DESCENT`
 - Folding Paths that start at a single Secondary Structure, [357](#)
- `VRNA_PATH_TYPE_DOT_BRACKET`
 - (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [350](#)
- `VRNA_PATH_TYPE_MOVES`
 - (Re-)folding Paths, Saddle Points, Energy Barriers, and Local Minima, [350](#)
- `vrna_pbacktrack`
 - Random Structure Samples from the Ensemble, [300](#)
- `vrna_pbacktrack5`
 - Random Structure Samples from the Ensemble, [294](#)
- `vrna_pbacktrack5_cb`
 - Random Structure Samples from the Ensemble, [296](#)
- `vrna_pbacktrack5_num`
 - Random Structure Samples from the Ensemble, [295](#)
- `vrna_pbacktrack5_resume`
 - Random Structure Samples from the Ensemble, [297](#)
- `vrna_pbacktrack5_resume_cb`
 - Random Structure Samples from the Ensemble, [298](#)
- `vrna_pbacktrack5_TwoD`
 - Stochastic Backtracking of Structures from Distance Based Partitioning, [325](#)
- `vrna_pbacktrack_cb`
 - Random Structure Samples from the Ensemble, [301](#)
- `VRNA_PBACKTRACK_DEFAULT`
 - Random Structure Samples from the Ensemble, [293](#)
- `vrna_pbacktrack_mem_free`
 - Random Structure Samples from the Ensemble, [311](#)
- `vrna_pbacktrack_mem_t`
 - Random Structure Samples from the Ensemble, [294](#)

- VRNA_PBACKTRACK_NON_REDUNDANT
 - Random Structure Samples from the Ensemble, [293](#)
- vrna_pbacktrack_num
 - Random Structure Samples from the Ensemble, [300](#)
- vrna_pbacktrack_resume
 - Random Structure Samples from the Ensemble, [302](#)
- vrna_pbacktrack_resume_cb
 - Random Structure Samples from the Ensemble, [304](#)
- vrna_pbacktrack_sub
 - Random Structure Samples from the Ensemble, [305](#)
- vrna_pbacktrack_sub_cb
 - Random Structure Samples from the Ensemble, [307](#)
- vrna_pbacktrack_sub_num
 - Random Structure Samples from the Ensemble, [306](#)
- vrna_pbacktrack_sub_resume
 - Random Structure Samples from the Ensemble, [308](#)
- vrna_pbacktrack_sub_resume_cb
 - Random Structure Samples from the Ensemble, [310](#)
- vrna_pbacktrack_TwoD
 - Stochastic Backtracking of Structures from Distance Based Partitioning, [324](#)
- vrna_pf
 - Global Partition Function and Equilibrium Probabilities, [273](#)
- vrna_pf_alifold
 - Global Partition Function and Equilibrium Probabilities, [276](#)
- vrna_pf_circalifold
 - Global Partition Function and Equilibrium Probabilities, [277](#)
- vrna_pf_circfold
 - Global Partition Function and Equilibrium Probabilities, [275](#)
- vrna_pf_co_fold
 - Global Partition Function and Equilibrium Probabilities, [278](#)
 - Partition Function for Two Hybridized Sequences, [398](#)
- vrna_pf_dimer
 - Global Partition Function and Equilibrium Probabilities, [274](#)
- vrna_pf_dimer_concentrations
 - Partition Function for Two Hybridized Sequences, [399](#)
- vrna_pf_dimer_probs
 - Predicting various thermodynamic properties, [331](#)
- vrna_pf_float_precision
 - Partition Function and Equilibrium Properties, [259](#)
- vrna_pf_fold
 - Global Partition Function and Equilibrium Probabilities, [275](#)
- vrna_pf_TwoD
 - Computing Partition Functions of a Distance Based Partitioning, [323](#)
- vrna_pfl_fold
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [283](#)
- vrna_pfl_fold_cb
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [283](#)
- vrna_pfl_fold_up
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [284](#)
- vrna_pfl_fold_up_cb
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [284](#)
- vrna_pinfo_s, [450](#)
- vrna_pk_plex
 - Pseudoknots, [372](#)
- vrna_pk_plex_accessibility
 - Pseudoknots, [372](#)
- vrna_pk_plex_opt
 - Pseudoknots, [373](#)
- vrna_pk_plex_opt_defaults
 - Pseudoknots, [373](#)
- vrna_pk_plex_opt_fun
 - Pseudoknots, [373](#)
- vrna_pk_plex_opt_t
 - Pseudoknots, [371](#)
- vrna_pk_plex_result_s, [370](#)
- vrna_pk_plex_score_f
 - Pseudoknots, [371](#)
- vrna_pk_plex_t
 - Pseudoknots, [371](#)
- vrna_plist
 - Pair List Representation of Secondary Structures, [440](#)
- vrna_plist_from_probs
 - Global Partition Function and Equilibrium Probabilities, [277](#)
- vrna_plot_coords
 - Layouts and Coordinates, [485](#)
- vrna_plot_coords_circular
 - Layouts and Coordinates, [488](#)
- vrna_plot_coords_circular_pt
 - Layouts and Coordinates, [489](#)
- vrna_plot_coords_pt
 - Layouts and Coordinates, [486](#)
- vrna_plot_coords_puzzler
 - Layouts and Coordinates, [490](#)
- vrna_plot_coords_puzzler_pt
 - Layouts and Coordinates, [490](#)
- vrna_plot_coords_simple
 - Layouts and Coordinates, [487](#)
- vrna_plot_coords_simple_pt
 - Layouts and Coordinates, [488](#)
- vrna_plot_coords_turtle

- Layouts and Coordinates, [492](#)
- `vrna_plot_coords_turtle_pt`
 - Layouts and Coordinates, [492](#)
- `vrna_plot_layout`
 - Layouts and Coordinates, [482](#)
- `vrna_plot_layout_circular`
 - Layouts and Coordinates, [484](#)
- `vrna_plot_layout_free`
 - Layouts and Coordinates, [485](#)
- `vrna_plot_layout_puzzler`
 - Layouts and Coordinates, [485](#)
- `vrna_plot_layout_s`, [481](#)
- `vrna_plot_layout_simple`
 - Layouts and Coordinates, [483](#)
- `vrna_plot_layout_t`
 - Layouts and Coordinates, [482](#)
- `vrna_plot_layout_turtle`
 - Layouts and Coordinates, [484](#)
- `vrna_plot_options_puzzler`
 - Layouts and Coordinates, [491](#)
- `vrna_plot_options_puzzler_free`
 - Layouts and Coordinates, [491](#)
- `vrna_plot_options_puzzler_t`, [481](#)
- `VRNA_PLOT_TYPE_CIRCULAR`
 - Layouts and Coordinates, [482](#)
- `VRNA_PLOT_TYPE_NAVIEW`
 - Layouts and Coordinates, [481](#)
- `VRNA_PLOT_TYPE_PUZZLER`
 - Layouts and Coordinates, [482](#)
- `VRNA_PLOT_TYPE_SIMPLE`
 - Layouts and Coordinates, [481](#)
- `VRNA_PLOT_TYPE_TURTLE`
 - Layouts and Coordinates, [482](#)
- `vrna_positional_entropy`
 - Predicting various thermodynamic properties, [331](#)
- `vrna_pr_energy`
 - Predicting various thermodynamic properties, [333](#)
- `vrna_pr_structure`
 - Predicting various thermodynamic properties, [332](#)
- `vrna_probs_window`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [282](#)
- `VRNA_PROBS_WINDOW_BPP`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [279](#)
- `vrna_probs_window_f`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [281](#)
- `VRNA_PROBS_WINDOW_PF`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [281](#)
- `VRNA_PROBS_WINDOW_STACKP`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [280](#)
- `VRNA_PROBS_WINDOW_UP`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [280](#)
- `VRNA_PROBS_WINDOW_UP_SPLIT`
 - Local (sliding window) Partition Function and Equilibrium Probabilities, [280](#)
- `vrna_pt_all_get`
 - Pair Table Representation of Secondary Structures, [438](#)
- `vrna_pt_pk_get`
 - Pair Table Representation of Secondary Structures, [437](#)
- `vrna_pt_pk_remove`
 - Pair Table Representation of Secondary Structures, [438](#)
- `vrna_pt_snoop_get`
 - Pair Table Representation of Secondary Structures, [438](#)
- `vrna_ptable`
 - Pair Table Representation of Secondary Structures, [436](#)
- `vrna_ptable_copy`
 - Pair Table Representation of Secondary Structures, [438](#)
- `vrna_ptable_from_string`
 - Pair Table Representation of Secondary Structures, [437](#)
- `vrna_ptypes`
 - Utilities to deal with Nucleotide Alphabets, [413](#)
- `vrna_random_string`
 - (Nucleic Acid Sequence) String Utilities, [422](#)
- `vrna_read_line`
 - Files and I/O, [456](#)
- `vrna_realloc`
 - Utilities, [386](#)
- `vrna_recursion_status_f`
 - The Fold Compound, [526](#)
- `vrna_refBPcnt_matrix`
 - Secondary Structure Utilities, [427](#)
- `vrna_refBPdist_matrix`
 - Secondary Structure Utilities, [427](#)
- `vrna_rotational_symmetry`
 - Combinatorics Algorithms, [501](#)
- `vrna_rotational_symmetry_db`
 - Combinatorics Algorithms, [502](#)
- `vrna_rotational_symmetry_db_pos`
 - Combinatorics Algorithms, [502](#)
- `vrna_rotational_symmetry_num`
 - Combinatorics Algorithms, [499](#)
- `vrna_rotational_symmetry_pos`
 - Combinatorics Algorithms, [501](#)
- `vrna_rotational_symmetry_pos_num`
 - Combinatorics Algorithms, [500](#)
- `vrna_salt_loop`
 - Energy Parameters, [216](#)
- `vrna_salt_loop_int`
 - Energy Parameters, [216](#)
- `vrna_salt_stack`
 - Energy Parameters, [216](#)
- `vrna_sc_add_bp`
 - Soft Constraints, [254](#)
- `vrna_sc_add_bt`

- Soft Constraints, [257](#)
- `vrna_sc_add_data`
 - Soft Constraints, [256](#)
- `vrna_sc_add_exp_f`
 - Soft Constraints, [257](#)
- `vrna_sc_add_f`
 - Soft Constraints, [256](#)
- `vrna_sc_add_hi_motif`
 - Incorporating Ligands Binding to Specific Sequence/Structure Motifs using Soft Constraints, [368](#)
- `vrna_sc_add_SHAPE_deigan`
 - SHAPE Reactivity Data, [361](#)
- `vrna_sc_add_SHAPE_deigan_al`
 - SHAPE Reactivity Data, [361](#)
- `vrna_sc_add_SHAPE_zarringhalam`
 - SHAPE Reactivity Data, [362](#)
- `vrna_sc_add_up`
 - Soft Constraints, [255](#)
- `vrna_sc_bp_storage_t`, [603](#)
- `vrna_sc_bt_f`
 - Soft Constraints, [252](#)
- `VRNA_SC_DEFAULT`
 - `soft.h`, [646](#)
- `vrna_sc_exp_f`
 - Soft Constraints, [251](#)
- `vrna_sc_f`
 - Soft Constraints, [251](#)
- `vrna_sc_free`
 - Soft Constraints, [256](#)
- `vrna_sc_init`
 - Soft Constraints, [253](#)
- `vrna_sc_minimize_perturbation`
 - Generate Soft Constraints from Data, [365](#)
- `vrna_sc_mod`
 - Post-transcriptional Modifications, [379](#)
- `vrna_sc_mod_7DA`
 - Post-transcriptional Modifications, [381](#)
- `vrna_sc_mod_dihydrouridine`
 - Post-transcriptional Modifications, [382](#)
- `vrna_sc_mod_inosine`
 - Post-transcriptional Modifications, [381](#)
- `vrna_sc_mod_json`
 - Post-transcriptional Modifications, [378](#)
- `vrna_sc_mod_jsonfile`
 - Post-transcriptional Modifications, [379](#)
- `vrna_sc_mod_m6A`
 - Post-transcriptional Modifications, [380](#)
- `vrna_sc_mod_param_s`, [603](#)
- `vrna_sc_mod_param_t`
 - Post-transcriptional Modifications, [377](#)
- `vrna_sc_mod_parameters_free`
 - Post-transcriptional Modifications, [378](#)
- `vrna_sc_mod_pseudouridine`
 - Post-transcriptional Modifications, [380](#)
- `vrna_sc_mod_purine`
 - Post-transcriptional Modifications, [382](#)
- `vrna_sc_mod_read_from_json`
 - Post-transcriptional Modifications, [378](#)
- `vrna_sc_mod_read_from_jsonfile`
 - Post-transcriptional Modifications, [377](#)
- `vrna_sc_motif_s`, [368](#)
- `vrna_sc_remove`
 - Soft Constraints, [255](#)
- `vrna_sc_s`, [249](#)
 - `bt`, [250](#)
 - `exp_f`, [250](#)
 - `f`, [250](#)
- `vrna_sc_set_bp`
 - Soft Constraints, [253](#)
- `vrna_sc_set_up`
 - Soft Constraints, [254](#)
- `vrna_sc_SHAPE_parse_method`
 - `SHAPE.h`, [643](#)
- `vrna_sc_SHAPE_to_pr`
 - SHAPE Reactivity Data, [363](#)
- `vrna_sc_type_e`
 - `soft.h`, [646](#)
- `VRNA_SC_WINDOW`
 - `soft.h`, [646](#)
- `vrna_search_BM_BCT`
 - Search Algorithms, [498](#)
- `vrna_search_BM_BCT_num`
 - Search Algorithms, [497](#)
- `vrna_search_BMH`
 - Search Algorithms, [497](#)
- `vrna_search_BMH_num`
 - Search Algorithms, [496](#)
- `vrna_sect_s`, [507](#)
- `VRNA_SEQ_DNA`
 - Utilities to deal with Nucleotide Alphabets, [413](#)
- `vrna_seq_encode`
 - Utilities to deal with Nucleotide Alphabets, [413](#)
- `vrna_seq_encode_simple`
 - Utilities to deal with Nucleotide Alphabets, [414](#)
- `vrna_seq_reverse`
 - (Nucleic Acid Sequence) String Utilities, [424](#)
- `VRNA_SEQ_RNA`
 - Utilities to deal with Nucleotide Alphabets, [413](#)
- `vrna_seq_toRNA`
 - (Nucleic Acid Sequence) String Utilities, [424](#)
- `vrna_seq_toupper`
 - (Nucleic Acid Sequence) String Utilities, [424](#)
- `vrna_seq_type_e`
 - Utilities to deal with Nucleotide Alphabets, [413](#)
- `vrna_seq_ungapped`
 - (Nucleic Acid Sequence) String Utilities, [425](#)
- `VRNA_SEQ_UNKNOWN`
 - Utilities to deal with Nucleotide Alphabets, [413](#)
- `vrna_sequence_s`, [413](#)
- `vrna_sol_TwoD_pf_t`, [323](#)
 - Computing Partition Functions of a Distance Based Partitioning, [323](#)
- `vrna_sol_TwoD_t`, [317](#)
 - Computing MFE representatives of a Distance Based Partitioning, [318](#)

- `vrna_stack_prob`
 - Predicting various thermodynamic properties, [331](#)
- `VRNA_STATUS_MFE_POST`
 - The Fold Compound, [524](#)
- `VRNA_STATUS_MFE_PRE`
 - The Fold Compound, [524](#)
- `VRNA_STATUS_PF_POST`
 - The Fold Compound, [525](#)
- `VRNA_STATUS_PF_PRE`
 - The Fold Compound, [525](#)
- `vrna_strcat_printf`
 - (Nucleic Acid Sequence) String Utilites, [419](#)
- `vrna_strcat_vprintf`
 - (Nucleic Acid Sequence) String Utilites, [419](#)
- `vrna_strdup_printf`
 - (Nucleic Acid Sequence) String Utilites, [418](#)
- `vrna_strdup_vprintf`
 - (Nucleic Acid Sequence) String Utilites, [418](#)
- `vrna_stream_output_f`
 - Buffers, [551](#)
- `vrna_string_header_s`, [603](#)
- `vrna_strsplit`
 - (Nucleic Acid Sequence) String Utilites, [421](#)
- `vrna_strtrim`
 - (Nucleic Acid Sequence) String Utilites, [420](#)
- `VRNA_STRUCTURE_TREE_EXPANDED`
 - Tree Representation of Secondary Structures, [446](#)
- `VRNA_STRUCTURE_TREE_HIT`
 - Tree Representation of Secondary Structures, [445](#)
- `VRNA_STRUCTURE_TREE_SHAPIRO`
 - Tree Representation of Secondary Structures, [445](#)
- `VRNA_STRUCTURE_TREE_SHAPIRO_EXT`
 - Tree Representation of Secondary Structures, [445](#)
- `VRNA_STRUCTURE_TREE_SHAPIRO_SHORT`
 - Tree Representation of Secondary Structures, [445](#)
- `VRNA_STRUCTURE_TREE_SHAPIRO_WEIGHT`
 - Tree Representation of Secondary Structures, [445](#)
- `vrna_structured_domains_s`, [604](#)
- `vrna_subopt`
 - Suboptimal Structures within an Energy Band around the MFE, [288](#)
- `vrna_subopt_cb`
 - Suboptimal Structures within an Energy Band around the MFE, [289](#)
- `vrna_subopt_result_f`
 - Suboptimal Structures within an Energy Band around the MFE, [288](#)
- `vrna_subopt_sol_s`, [604](#)
- `vrna_subopt_zuker`
 - Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, [287](#)
- `vrna_time_stamp`
 - Utilities, [387](#)
- `vrna_tree_string_to_db`
 - Tree Representation of Secondary Structures, [447](#)
- `vrna_tree_string_unweight`
 - Tree Representation of Secondary Structures, [447](#)
- `VRNA_TRIM_ALL`
 - (Nucleic Acid Sequence) String Utilites, [418](#)
- `VRNA_TRIM_DEFAULT`
 - (Nucleic Acid Sequence) String Utilites, [417](#)
- `VRNA_TRIM_IN_BETWEEN`
 - (Nucleic Acid Sequence) String Utilites, [417](#)
- `VRNA_TRIM_LEADING`
 - (Nucleic Acid Sequence) String Utilites, [417](#)
- `VRNA_TRIM_SUBST_BY_FIRST`
 - (Nucleic Acid Sequence) String Utilites, [417](#)
- `VRNA_TRIM_TRAILING`
 - (Nucleic Acid Sequence) String Utilites, [417](#)
- `vrna_ud_add_motif`
 - Unstructured Domains, [223](#)
- `vrna_ud_add_probs_f`
 - Unstructured Domains, [221](#)
- `vrna_ud_exp_f`
 - Unstructured Domains, [220](#)
- `vrna_ud_exp_production_f`
 - Unstructured Domains, [221](#)
- `vrna_ud_f`
 - Unstructured Domains, [220](#)
- `vrna_ud_get_motif_size_at`
 - `unstructured_domains.h`, [1286](#)
- `vrna_ud_get_probs_f`
 - Unstructured Domains, [221](#)
- `vrna_ud_motifs_centroid`
 - Unstructured Domains, [221](#)
- `vrna_ud_motifs_MEA`
 - Unstructured Domains, [222](#)
- `vrna_ud_motifs_MFE`
 - Unstructured Domains, [222](#)
- `vrna_ud_production_f`
 - Unstructured Domains, [221](#)
- `vrna_ud_remove`
 - Unstructured Domains, [224](#)
- `vrna_ud_set_data`
 - Unstructured Domains, [224](#)
- `vrna_ud_set_exp_prod_rule_cb`
 - Unstructured Domains, [225](#)
- `vrna_ud_set_prob_cb`
 - `unstructured_domains.h`, [1286](#)
- `vrna_ud_set_prod_rule_cb`
 - Unstructured Domains, [224](#)
- `VRNA_UNIT_CAL`
 - Unit Conversion, [514](#)
- `VRNA_UNIT_CAL_IT`
 - Unit Conversion, [514](#)
- `VRNA_UNIT_DACAL`
 - Unit Conversion, [514](#)
- `VRNA_UNIT_DACAL_IT`
 - Unit Conversion, [514](#)
- `VRNA_UNIT_DEG_C`
 - Unit Conversion, [515](#)
- `VRNA_UNIT_DEG_DE`
 - Unit Conversion, [515](#)
- `VRNA_UNIT_DEG_F`
 - Unit Conversion, [515](#)
- `VRNA_UNIT_DEG_N`

- Unit Conversion, [515](#)
- VRNA_UNIT_DEG_R
 - Unit Conversion, [515](#)
- VRNA_UNIT_DEG_RE
 - Unit Conversion, [515](#)
- VRNA_UNIT_DEG_RO
 - Unit Conversion, [515](#)
- vrna_unit_energy_e
 - Unit Conversion, [514](#)
- VRNA_UNIT_EV
 - Unit Conversion, [514](#)
- VRNA_UNIT_G_TNT
 - Unit Conversion, [514](#)
- VRNA_UNIT_J
 - Unit Conversion, [514](#)
- VRNA_UNIT_K
 - Unit Conversion, [515](#)
- VRNA_UNIT_KCAL
 - Unit Conversion, [514](#)
- VRNA_UNIT_KCAL_IT
 - Unit Conversion, [514](#)
- VRNA_UNIT_KG_TNT
 - Unit Conversion, [514](#)
- VRNA_UNIT_KJ
 - Unit Conversion, [514](#)
- VRNA_UNIT_KWH
 - Unit Conversion, [514](#)
- VRNA_UNIT_T_TNT
 - Unit Conversion, [514](#)
- vrna_unit_temperature_e
 - Unit Conversion, [514](#)
- VRNA_UNIT_WH
 - Unit Conversion, [514](#)
- vrna_unstructured_domain_motif_s, [604](#)
- vrna_unstructured_domain_s, [219](#)
 - prod_cb, [220](#)
- vrna_urn
 - Utilities, [387](#)
- warn_user
 - basic.h, [928](#)
- Washington University Secondary Structure (WUSS) notation, [434](#)
 - vrna_db_from_WUSS, [435](#)
- write_parameter_file
 - Reading/Writing Energy Parameter Sets from/to File, [407](#)
- xrealloc
 - basic.h, [929](#)
- xrna_plot
 - Plotting, [479](#)
- xsubi
 - Utilities, [389](#)
- zukersubopt
 - Suboptimal Structures sensu Stiegler et al. 1984 / Zuker et al. 1989, [286](#)
- zukersubopt_par