Graph Canonicalization and Algorithmic Engineering

Jakob L. Andersen, Thomas G. Hansen, and Daniel Merkle

Department of Mathematics and Computer Science,
University of Southern Denmark

February 2019

Graph Canonicalization

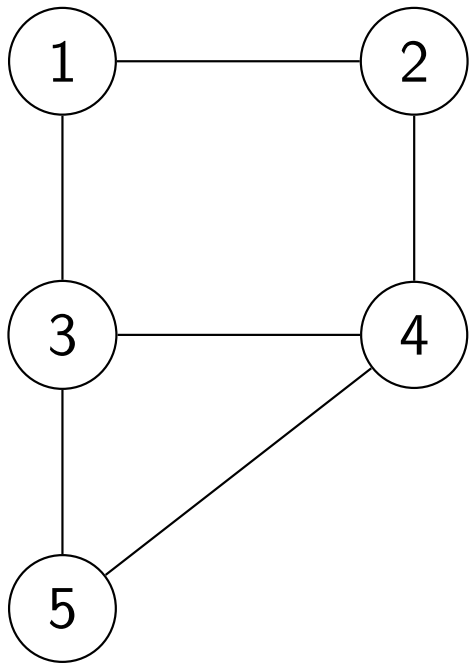
Quick Introduction

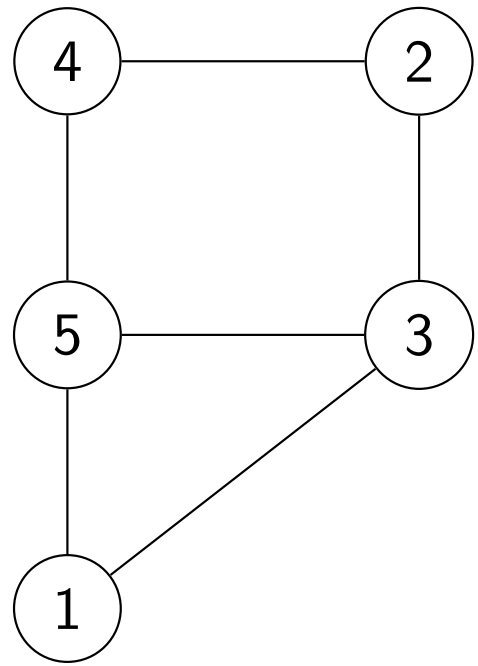
- Same graph can be represented in multiple ways
- Canonical form is a ‘standard’ representation
- Obvious applications
  - Graph isomorphism
  - Unique identifier for databases

Graph Canonicalization

Quick Introduction

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{?\}$



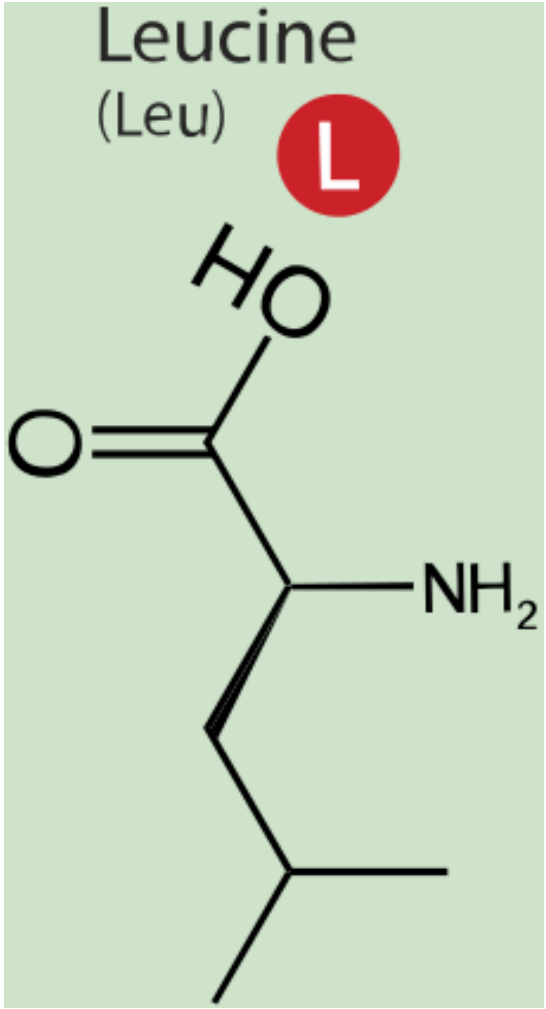


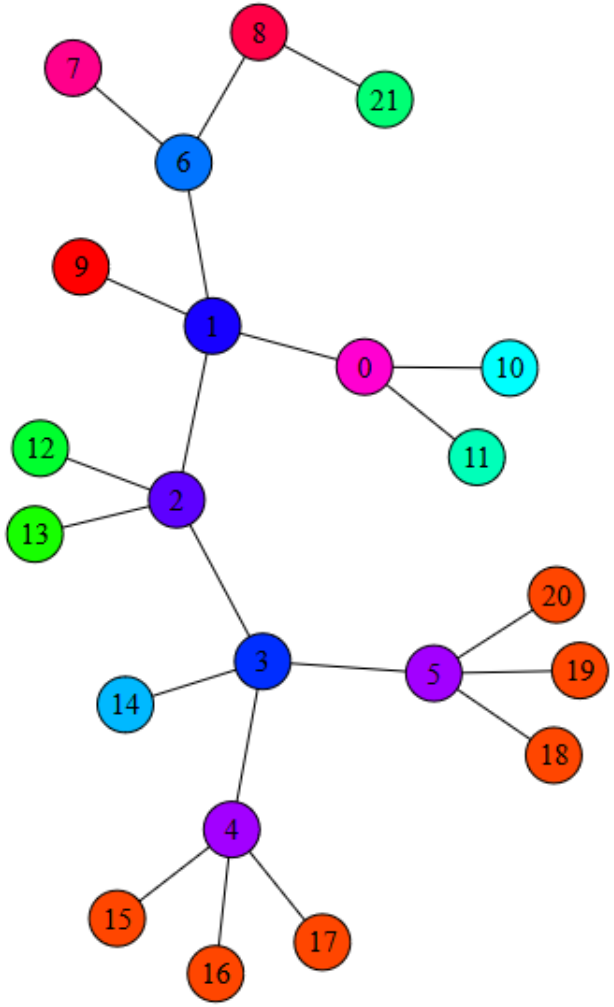
(1,2) (1,3) (2,4) (3,4) (3,5)

(1,3) (1,5) (2,3) (2,4) (3,5)

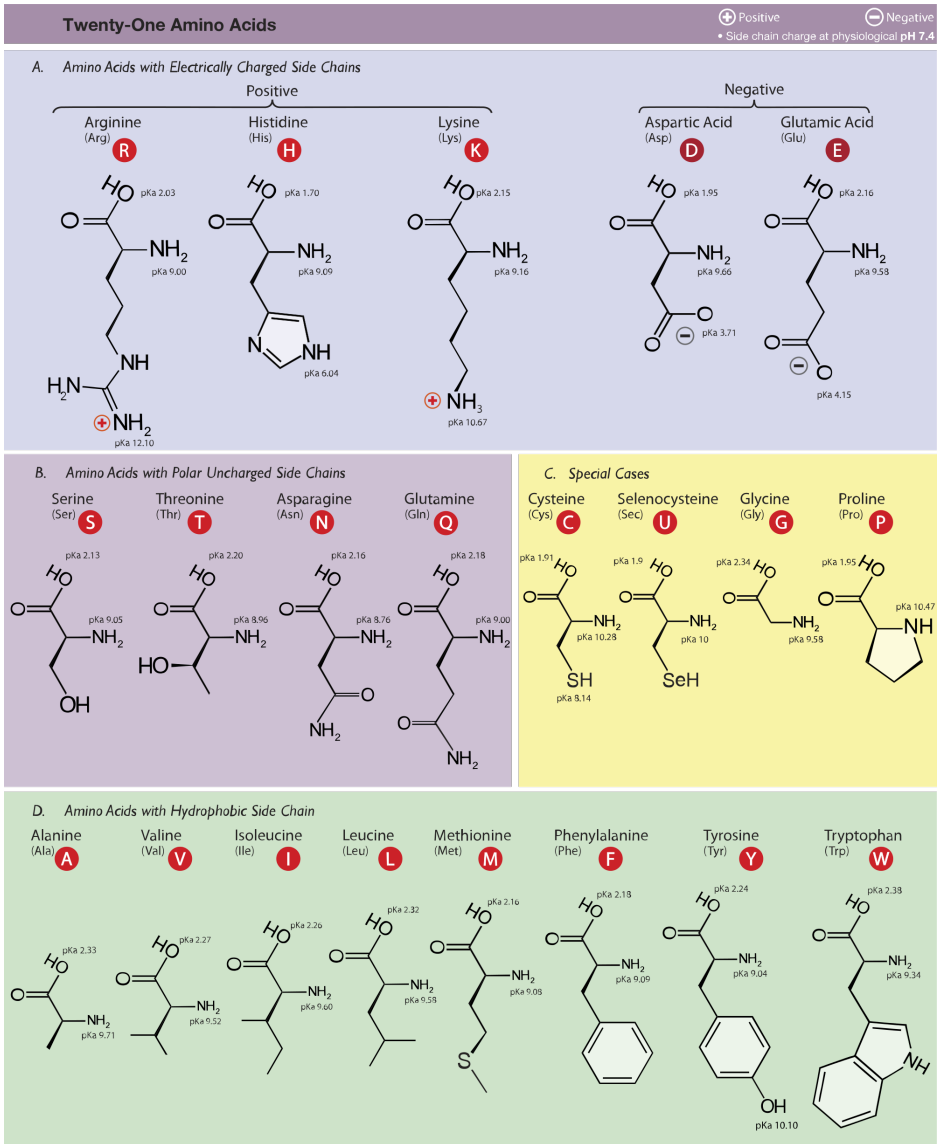
Graph Canonicalization

- Molecules as graphs (Atoms as vertices, bonds as edges)
- Example: Leucine as a graph

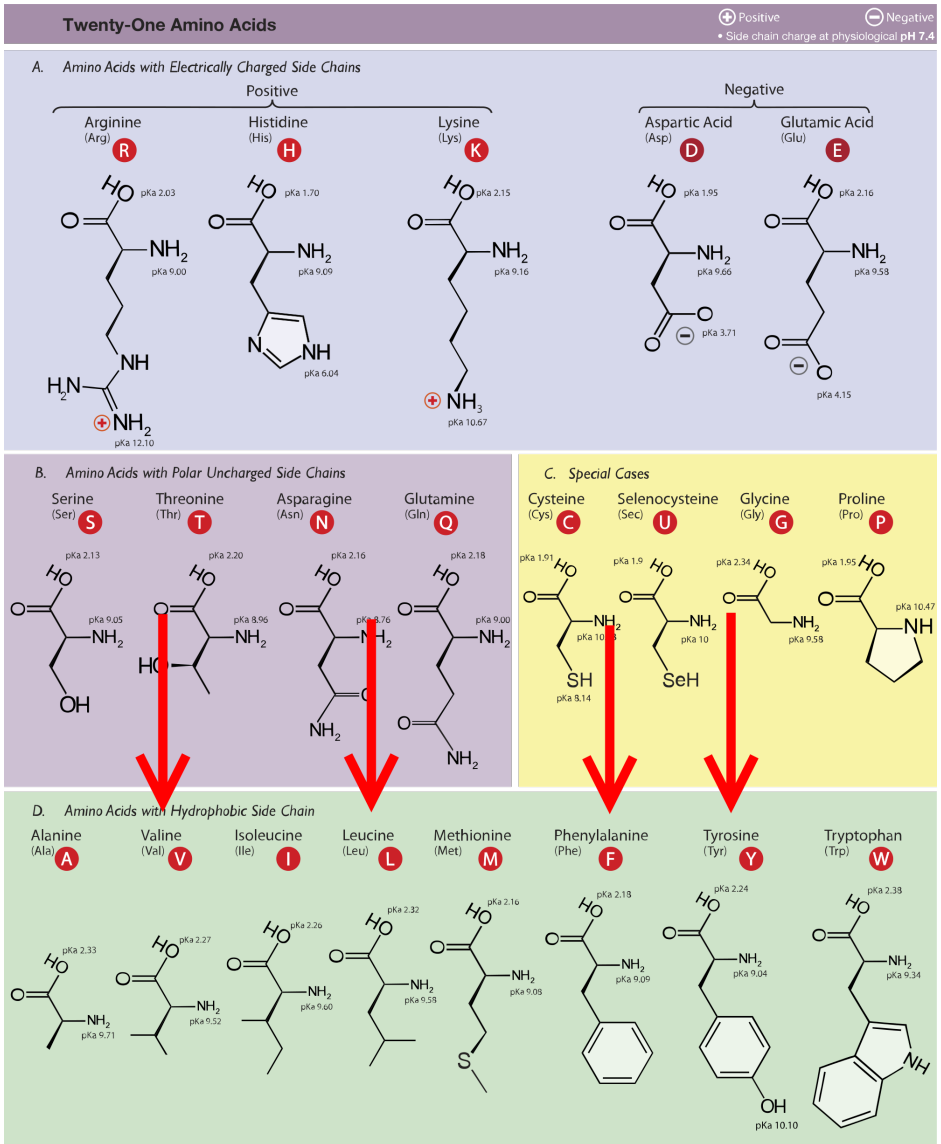




Graph Canonicalization



Graph Canonicalization



Graph Canonicalization

- InChI
  - Unique identifier for each molecule
  - Performs graph canonicalization as a sub-process
- InChI, problems as a computer scientist
  - 'Implementation is the specification'
  - – experimental support of large molecules containing up to 32767 atoms was added; [1]

[1] InChI 1.05, latest release as of now
https://iupac.org/new-inchi-software-release/

Graph Canonicalization

- InChI
  - Unique identifier for each molecule
  - Performs graph canonicalization as a sub-process
- InChI, problems as a computer scientist
  - 'Implementation is the specification'
  - – experimental support of large molecules containing up to 32767 atoms was added; [1]
  - InChI for large molecules

[1] InChI 1.05, latest release as of now
https://iupac.org/new-inchi-software-release/

RECORD BREAKING INCHI-KEY

- Sequence Identifier: UTP10_KLULA
- Sequence Length: 1774 amino acids
- Molecule size: 28509 atoms
- InChI Length: 119699 characters
- InChI key: PHBRSEQMAKHFGD-ZBXWIJJNSA-N
- InChI Canonicalization Time: 73.2s
- Canonical SMILES Length: 35408 chars
- SMILES Canonicalization Time: 0.4s

Graph Canonicalization

- Amino acids are very tree-like in structure
- Canonicalizing trees is fast
- 73 seconds sounds slow, given above

Graph Canonicalization

- Amino acids are very tree-like in structure
- Canonicalizing trees is fast
- 73 seconds sounds slow, given above
- So what's going on in InChI?

Graph Canonicalization

- Amino acids are very tree-like in structure
- Canonicalizing trees is fast
- 73 seconds sounds slow, given above
- So what's going on in InChI?
- Can we canonicalize faster?

Algorithmic Engineering

- I created Euthyphro
- A general-purpose graph canonicalization tool
- Use and extend pre-existing graph canonicalization library
- Optimization focused on canonicalization of molecule graphs
- Faster than InChI — 0.8 seconds for `UTP10_KLULA`
  - Not entirely fair comparison, InChI does more
  - However: We don't believe the other parts should be slow

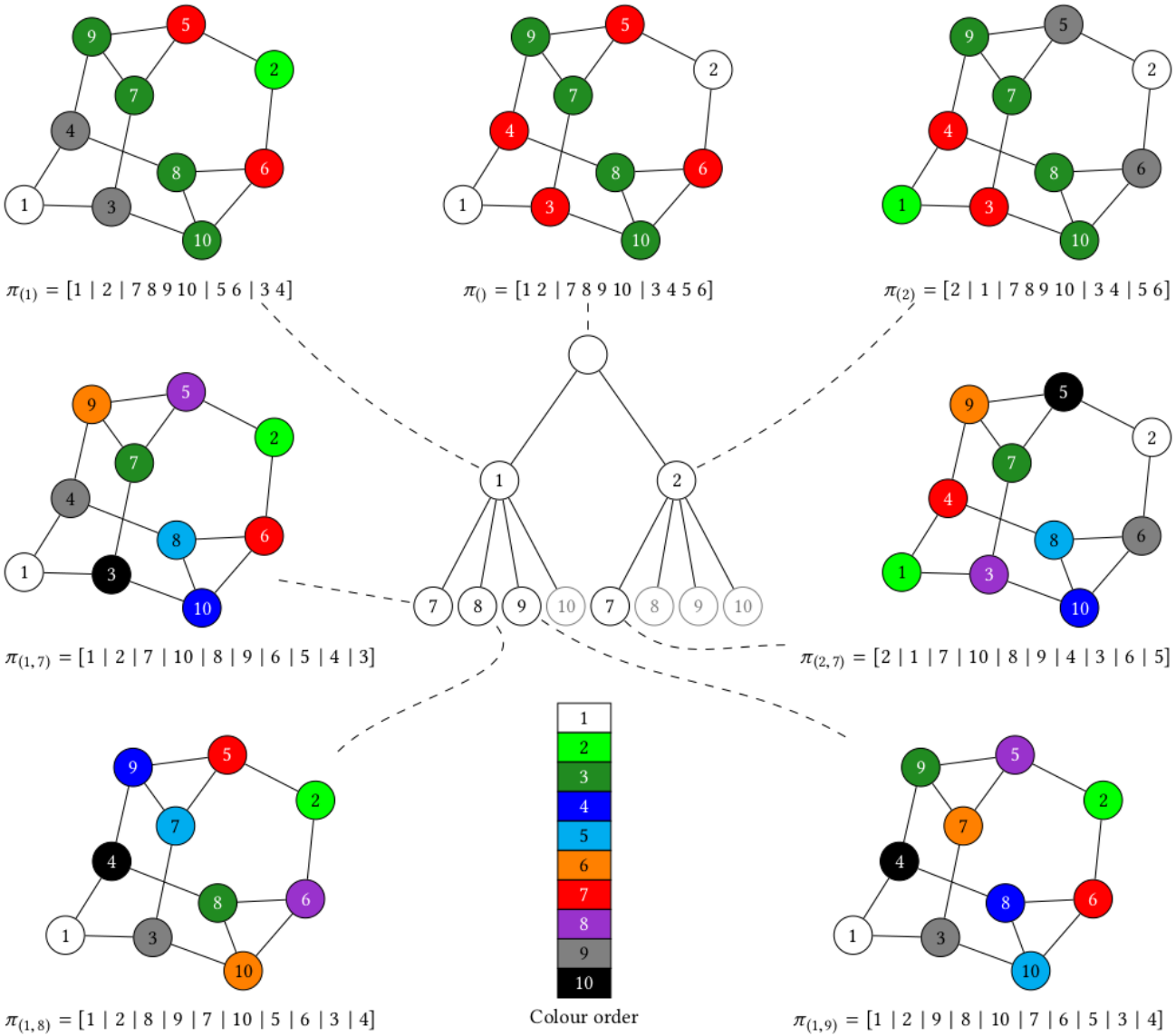
Algorithmic Engineering

Individualization-refinement

- `Graph Canon` library by Jakob L. Andersen, Daniel Merkle
- 'Simple' process
  - Initial refinement: Divide vertices into cells
  - Individualization: Split a cell, making a new cell of size 1
- All cells size 1: Candidate for canonical form
- Explore all permutations of individualization
- Choose lexicographically smallest candidate

Algorithmic Engineering

Individualization-refinement



Algorithmic Engineering

Individualization-refinement

- Initial refinements by invariants
  - Degree, vertex label, edge label, degree of edge labels, etc.
- Use symmetries to prune branches from search tree
  - Find symmetries by comparing candidates
  - Find symmetries by inspecting graph
- 'Manually' split to skip deeper into search tree
- Cell selector (First, First Largest, Smallest Non-Trivial, etc.)
- Search strategy (DFS, BFS[2])

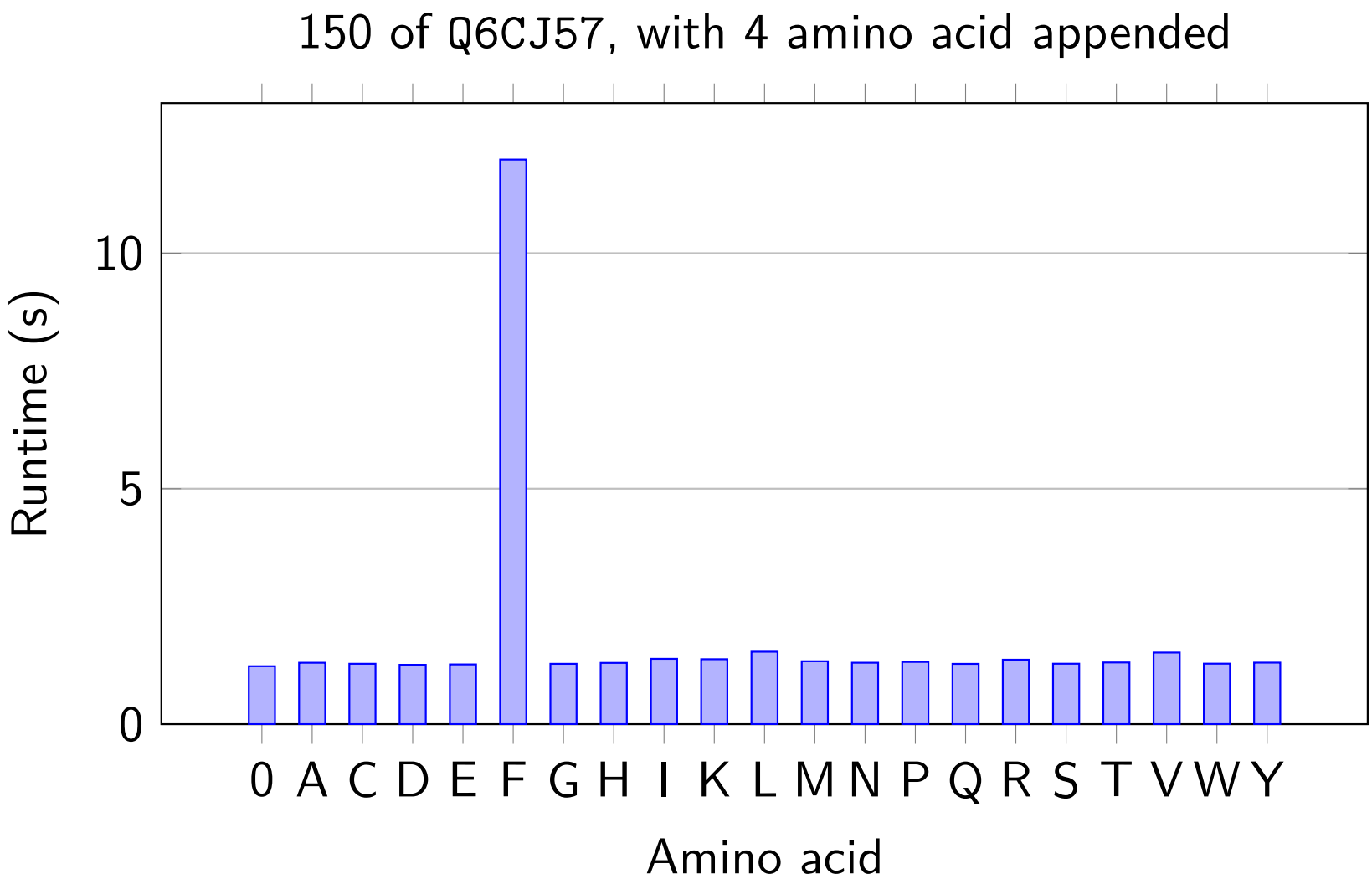
[2] BFS uses a lot of memory. There exists variants that use less.

Algorithmic Engineering

- FASTA $\rightarrow$ Graph tooling made
- Initial Euthyphro implementation finished
- Acquired `UTP10_KLULA` as test dataset
  - Also known as `Q6CJ57`
- Begin testing!

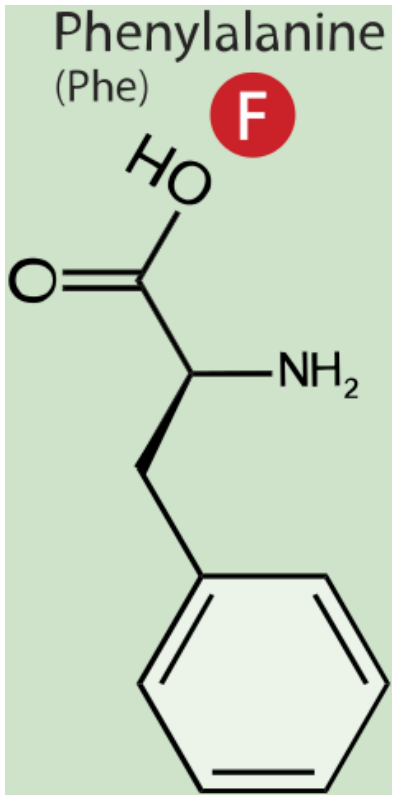
Algorithmic Engineering

Test 1



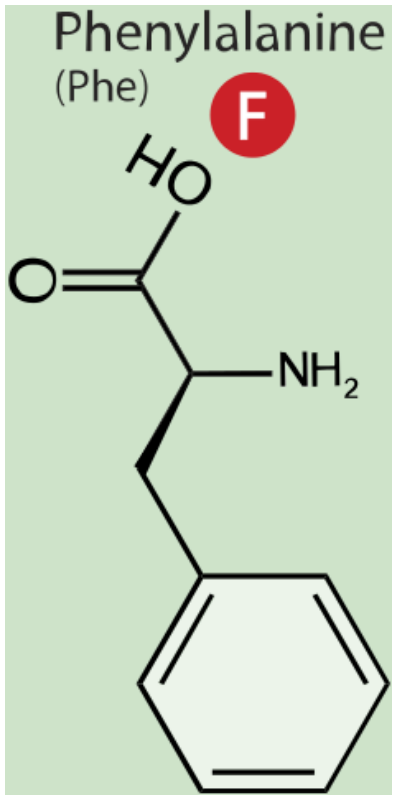
Algorithmic Engineering

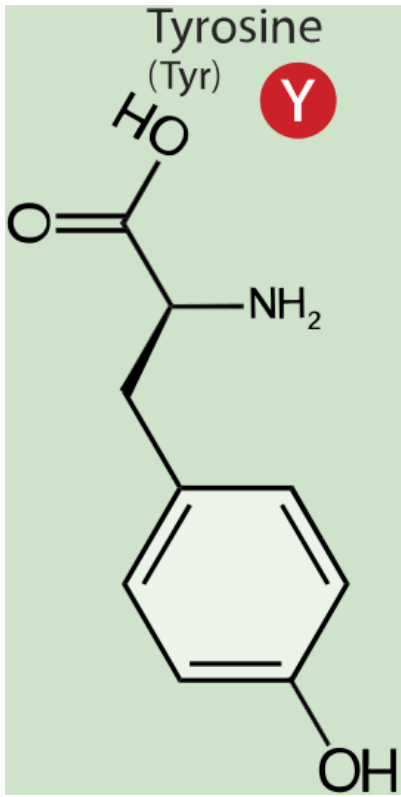
- Phenylalanine has significant impact
- Is it the cycle?



Algorithmic Engineering

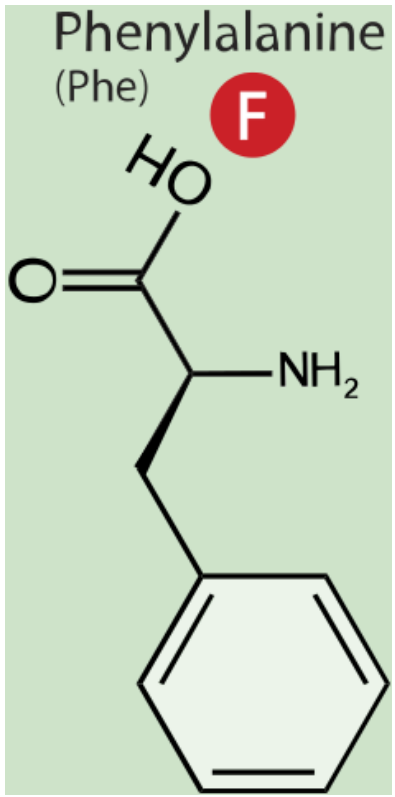
- Phenylalanine has significant impact
- Is it the cycle?
- Tyrosine has a cycle and no problem.

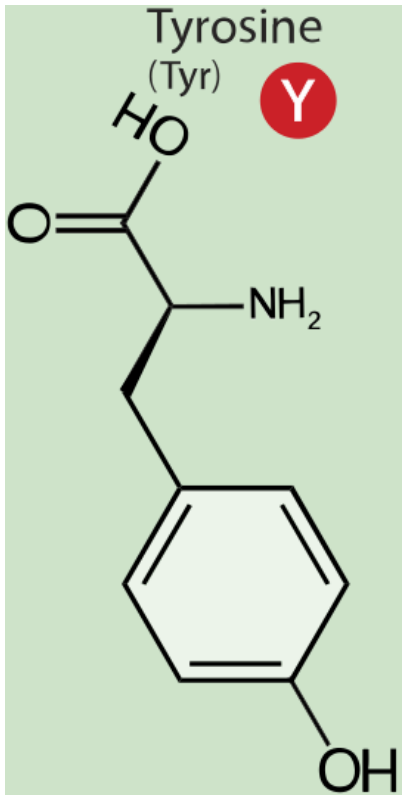




Algorithmic Engineering

- Phenylalanine has significant impact
- Is it the cycle?
- Tyrosine has a cycle and no problem.
- SMILES I found online for Phenylalanine is Kekulé form!



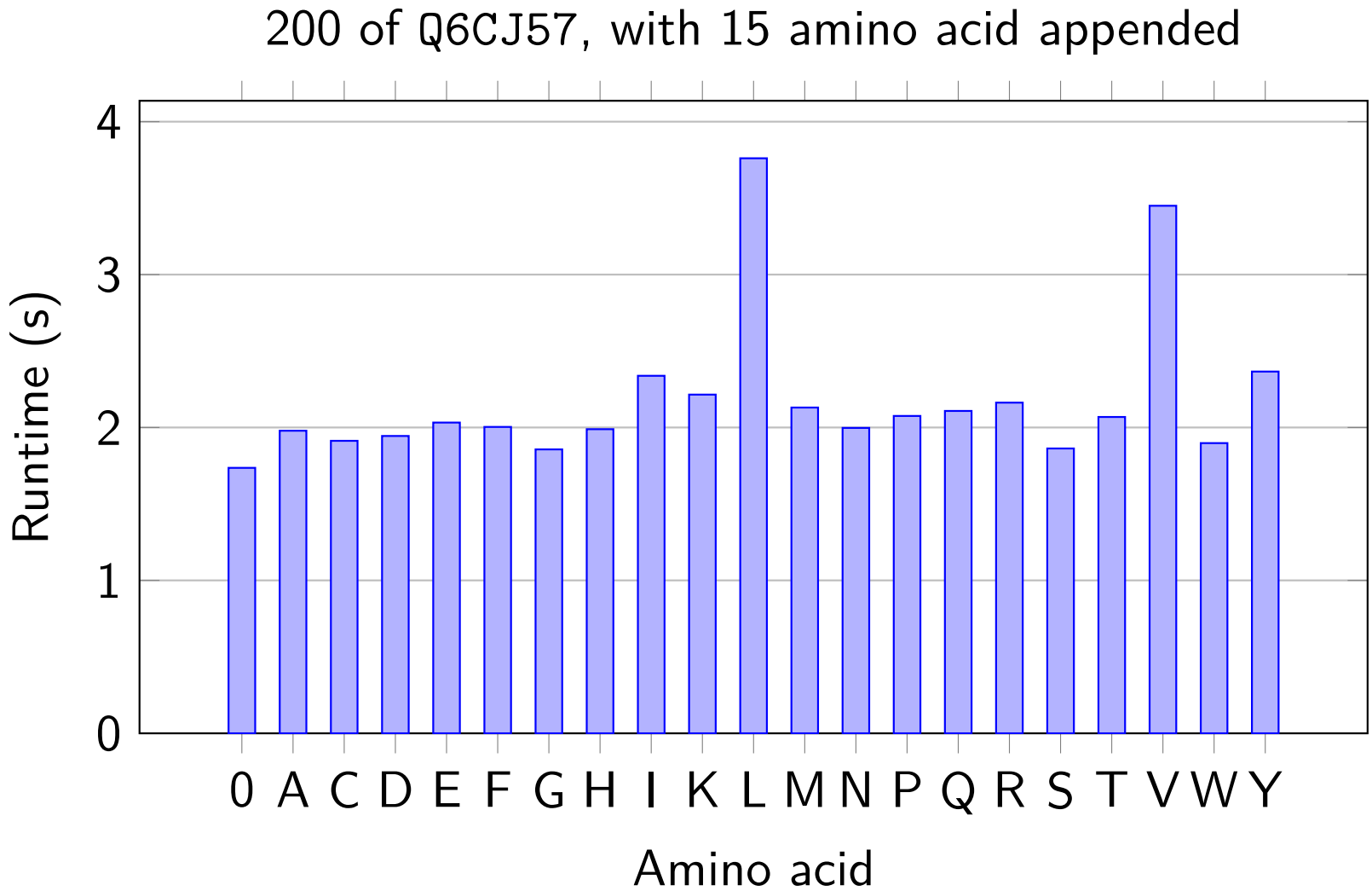


Algorithmic Engineering

- Problem: Implementation assumed all edges were equal
- Makes Kekulé appear symmetric without being so
- Makes canonicalization unsound
- Solution: Don't assume all edges are equal (oops)
- Note: Different edge types only makes canonicalization faster

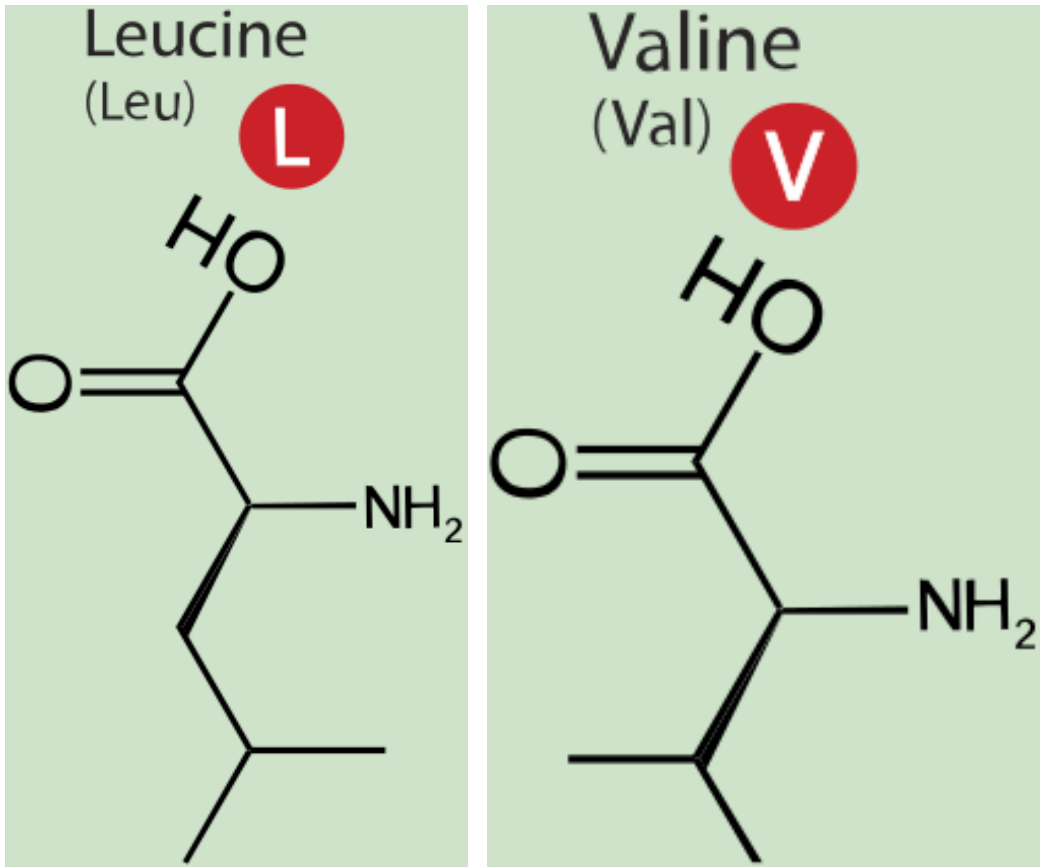
Algorithmic Engineering

Test 2



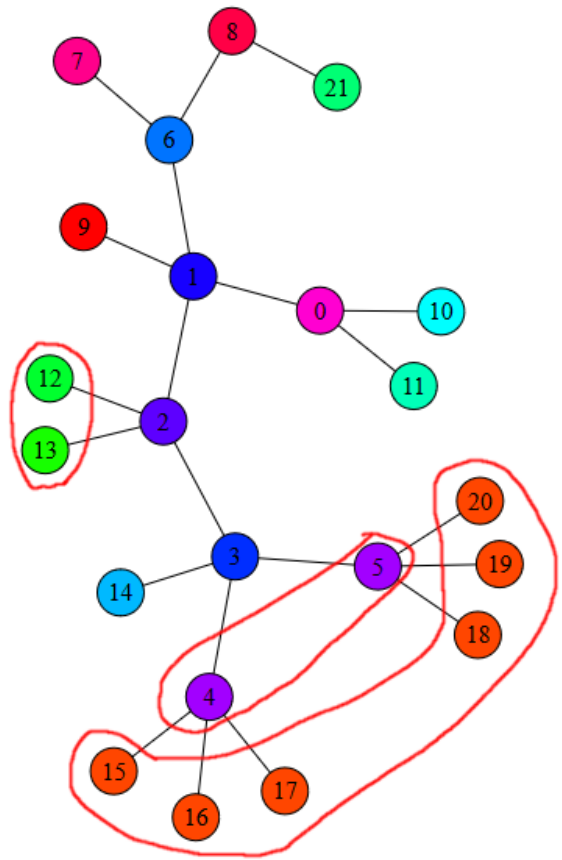
Algorithmic Engineering

- Leucine and Valine have biggest impact
- Visual inspection shows symmetric tree-like structure
- The methyl groups are the symmetric part



Algorithmic Engineering

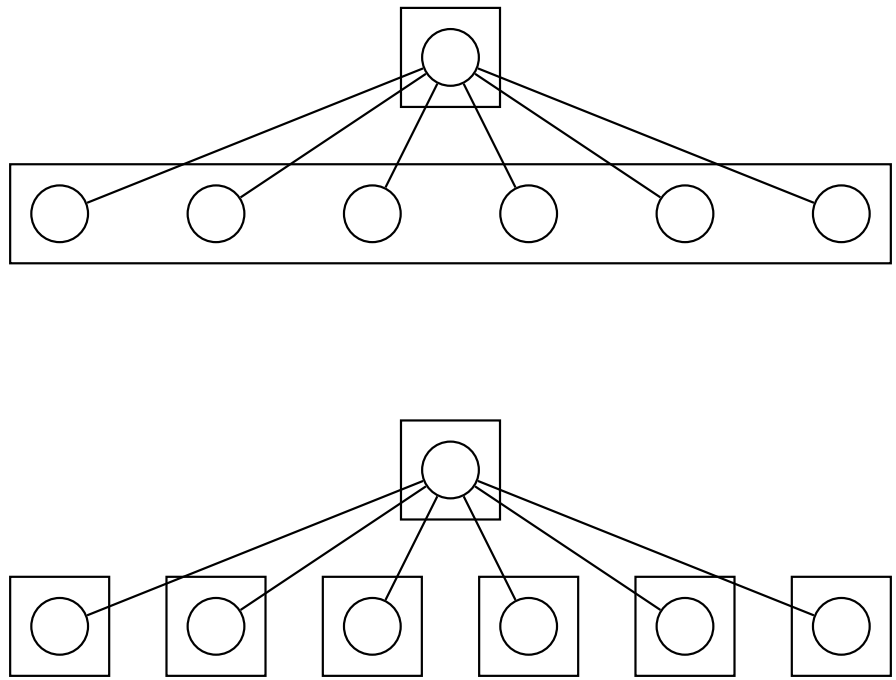
- Consider the cells after initial refinement
- Naively: We must branch on both $12, 13$ to find they are symmetric
- We can be smart — Degree 1 refiner



Algorithmic Engineering

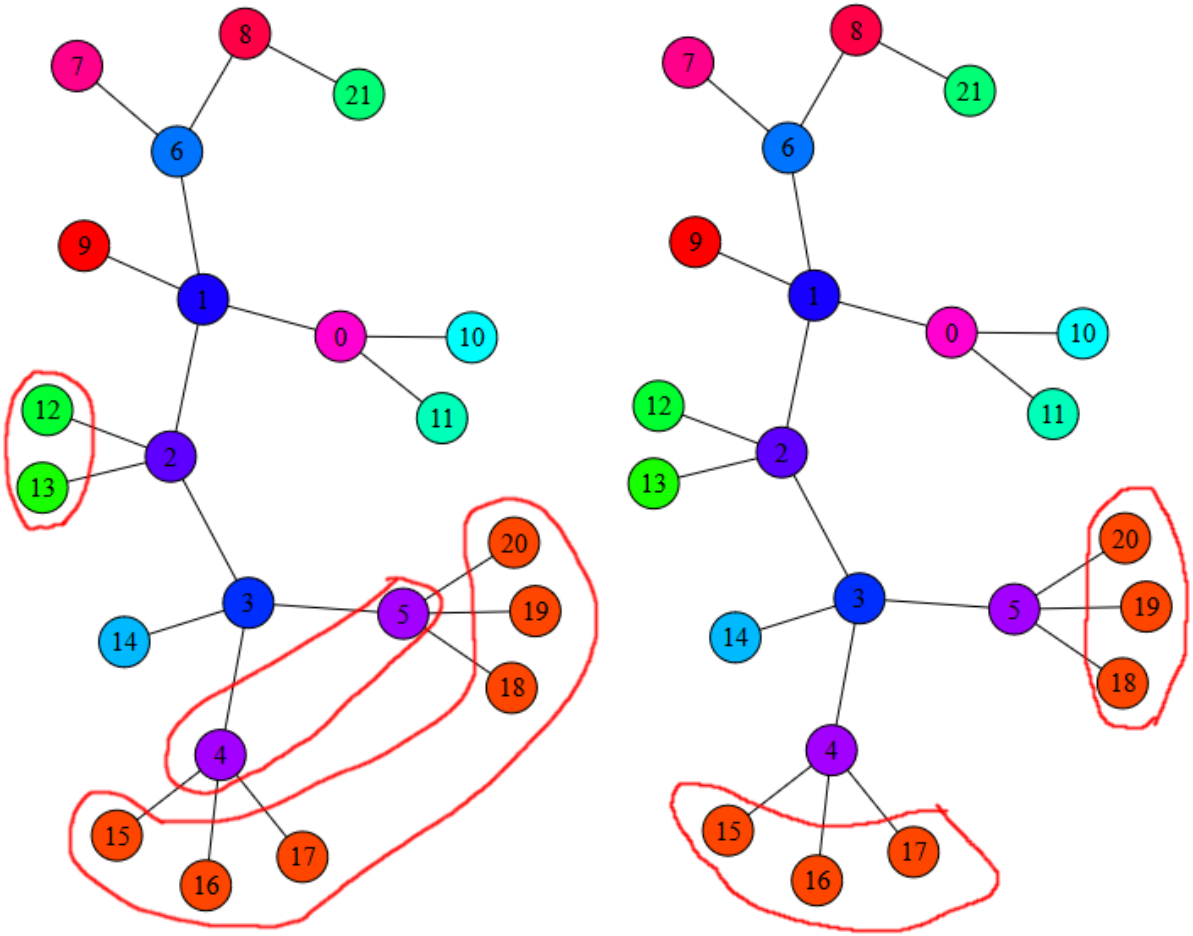
Degree 1 refiner

- Consider: A cell has vertices of degree 1, and all have the same neighbor
- They *must* be symmetric
- We can immediately split them



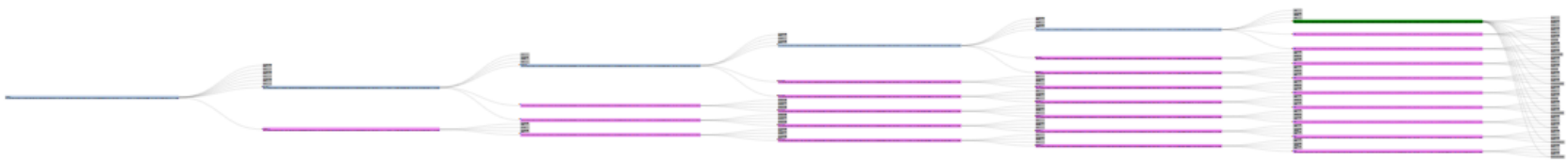
Algorithmic Engineering

- Has to branch on $4, 5$
- Degree-1 splits $15, \ldots, 20$ after branch



Algorithmic Engineering

- Example: Search tree for degree-1 on FASTA LLLLL
- Branching, branching, branching



Algorithmic Engineering

- Majority of our branching comes from subtrees
- Subtrees are shallow — Can we make them depth 1?

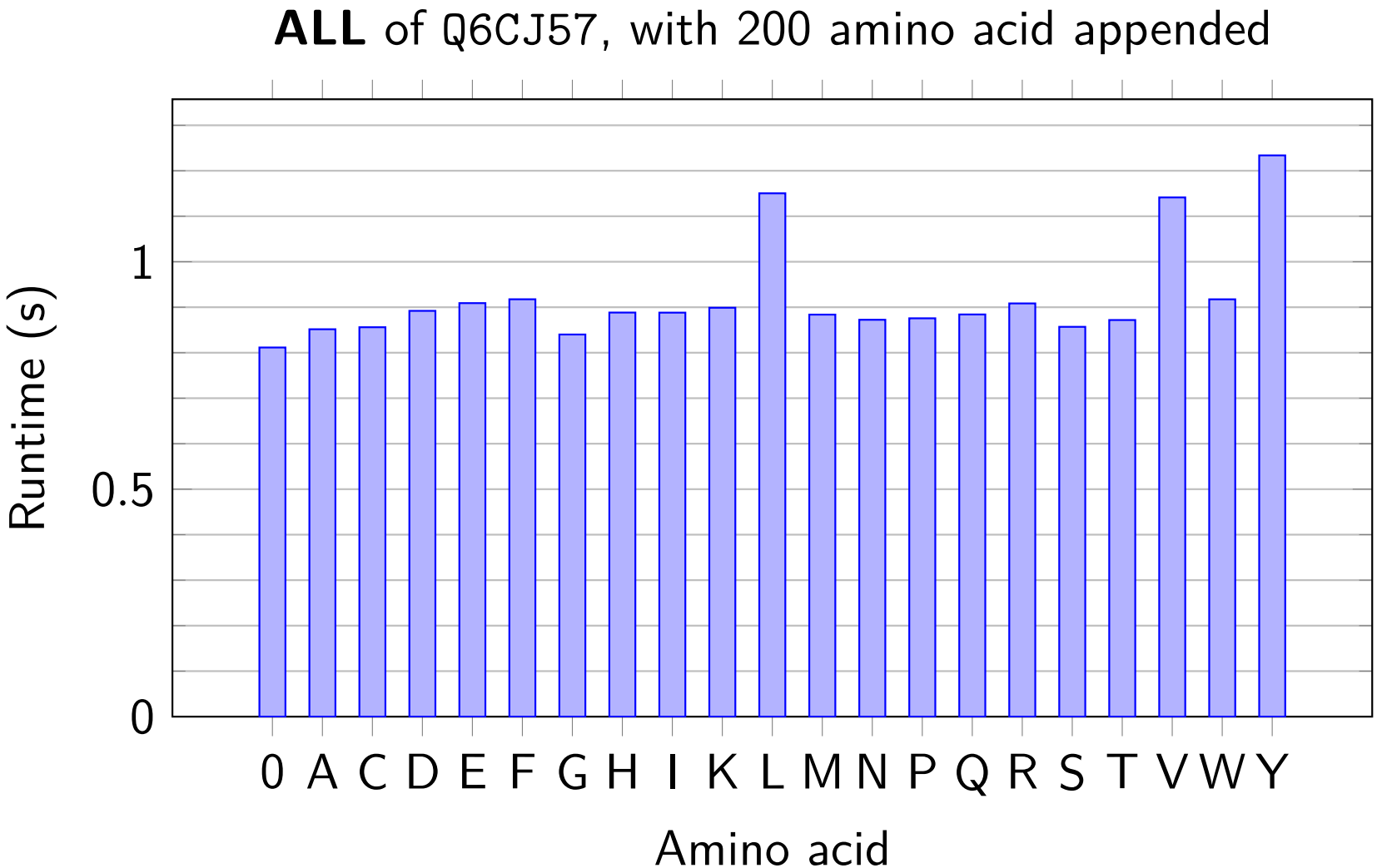
Algorithmic Engineering

- Majority of our branching comes from subtrees
- Subtrees are shallow — Can we make them depth 1?
- We can strip hydrogens and reinsert them after canonicalization

Algorithmic Engineering

Test 3

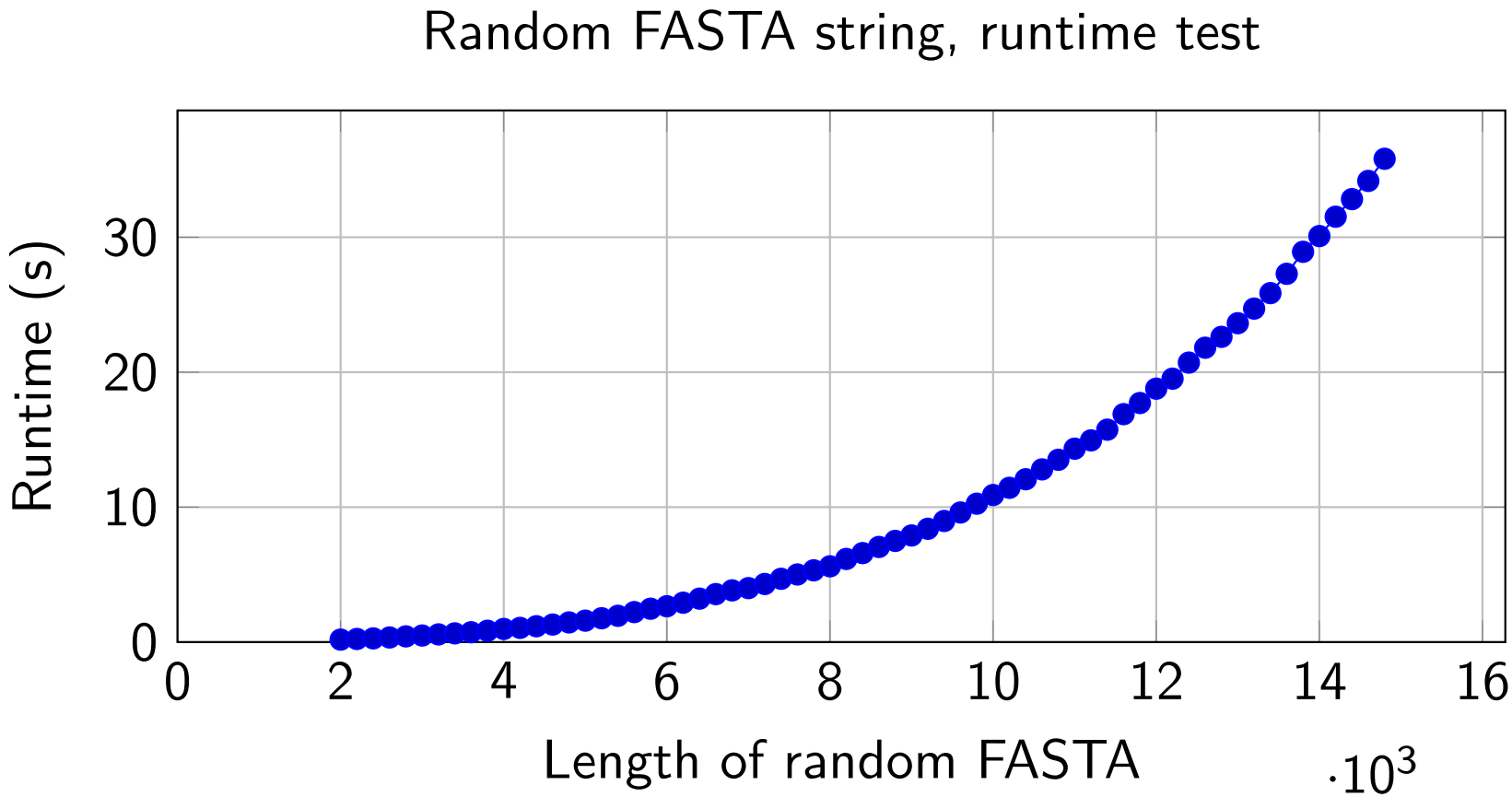
- Reminder: InChI took 73 seconds



Algorithmic Engineering

Test 4

- Generated length 15000 FASTA has 241882 atoms
- Reminder: InChI supports 32767



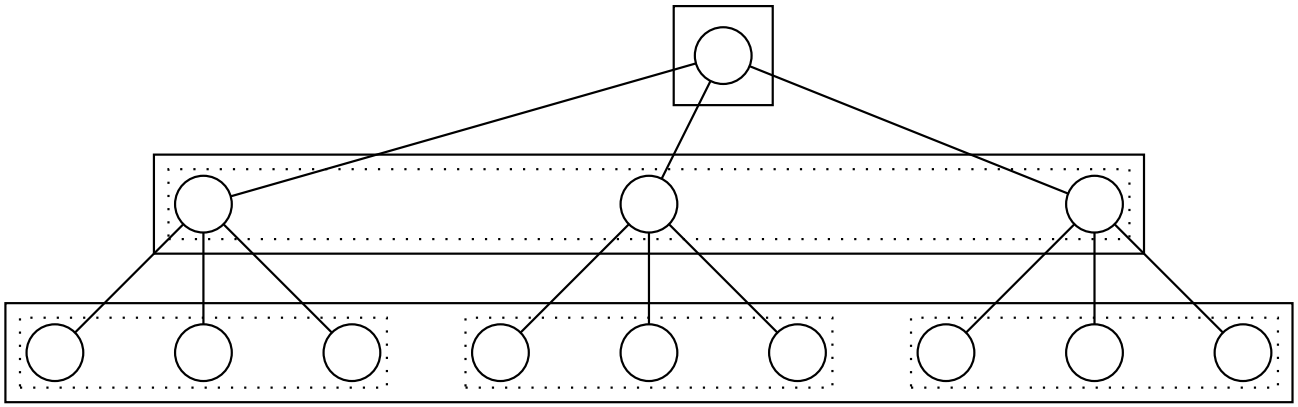
Algorithmic Engineering

Subtree Refiner

- Degree 1 finds subtrees of depth 1
- Stripping hydrogens shows we have depth 2 subtrees
- What if we could find arbitrary depth subtrees?
- Would it be sound?

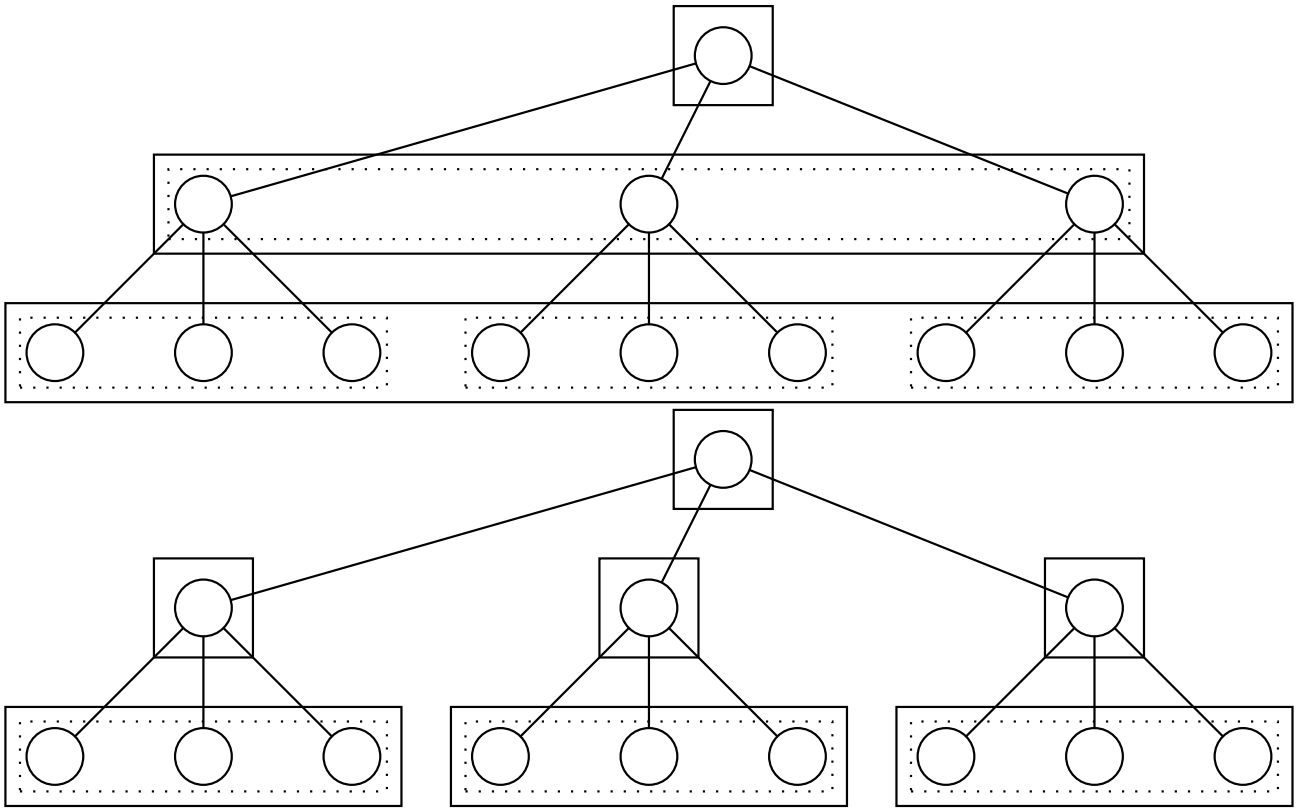
Algorithmic Engineering

Subtree Refiner



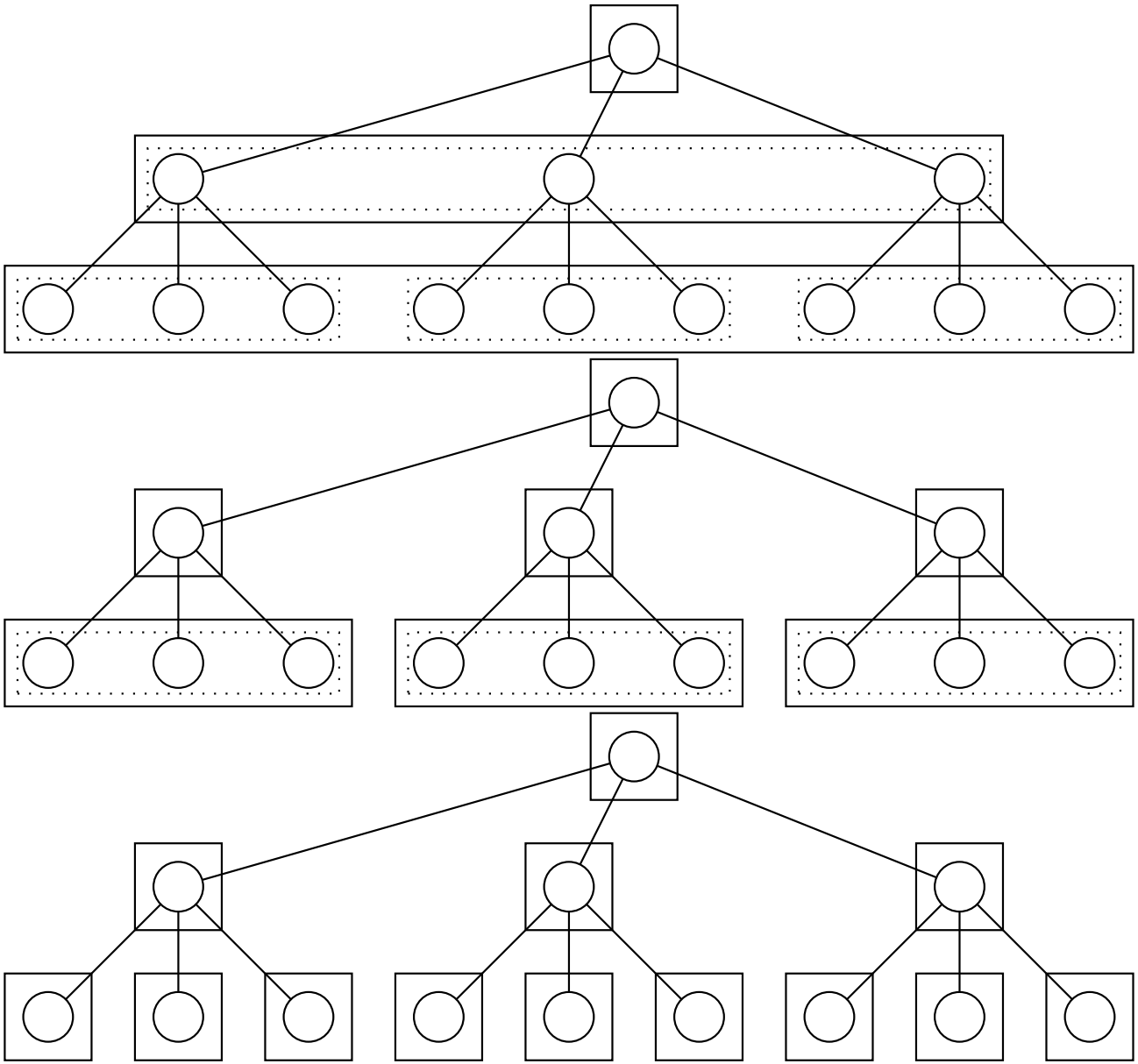
Algorithmic Engineering

Subtree Refiner



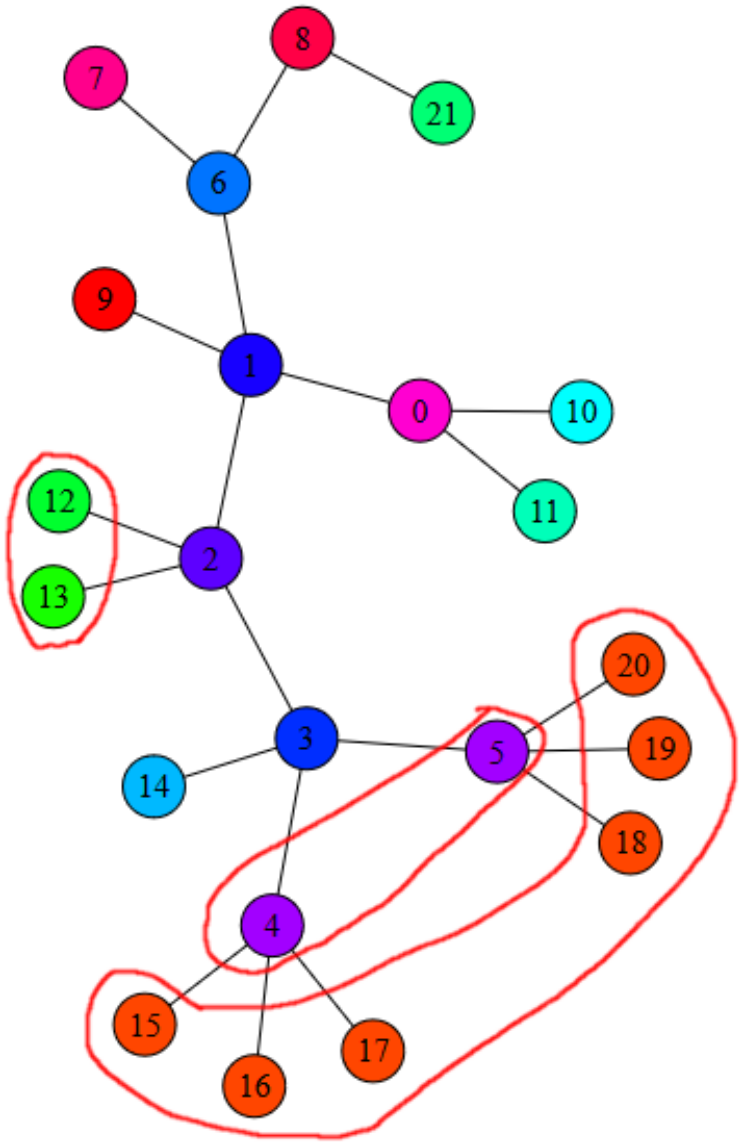
Algorithmic Engineering

Subtree Refiner



Graph Canonicalization

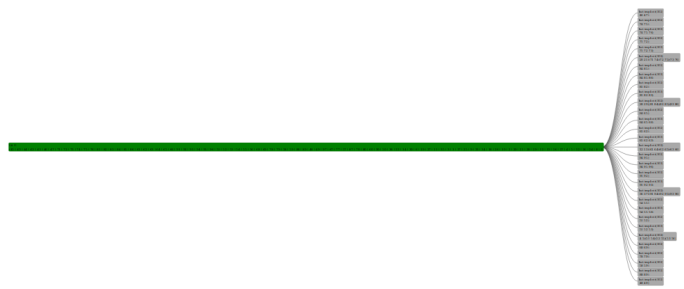
Subtree Refiner



Graph Canonicalization

Subtree Refiner

- I implemented a subtree refiner in `Graph Canon`
- Performs manual split
- Inefficient 'proof of concept' implementation
- Example: Search tree using subtree refiner on FASTA `LLLLL`



- Search tree is just one node
- 30 second runtime for 1200 symbols of `Q6CJ57`
- Number of nodes in search tree dramatically reduced for large proteins
  - Test with 250 symbols of `Q6CJ57`
  - Degree 1: 3571 nodes in search tree, 4.38s runtime.
  - Subtree: 10 nodes in search tree, 0.23s runtime.

Further work

- Other large molecules/structures
  - Linear proteins are boring, want something not tree-like
  - Circular RNA?
- Testing different cell selectors and search strategies
- Extend tooling to allow user to switch benzene form
- Faster, more efficient implementation of subtree refiner
- Stereochemistry — Hard problem, maybe ph.d.