Sorting by TDRL and iTDRL

Bruno Schmidt, Tom Hartmann, Peter Stadler

Genome Rearrangements

Genome Rearrangements

- Not only genes undergo mutation
  → Genome mutations, or *genome rearrangements*

Genome Rearrangements

- Not only genes undergo mutation
  → Genome mutations, or *genome rearrangements*
- Many rearrangement **types** have been considered
  → Inversion, Transposition, inverse Transposition, Cut and Join, ...

Genome Rearrangements

- Not only genes undergo mutation
  → Genome mutations, or *genome rearrangements*
- Many rearrangement **types** have been considered
  → Inversion, Transposition, inverse Transposition, Cut and Join, ...
- **Sorting Problem:** *Given a set of rearrangement types, what is the shortest sequence of those operations to transform one genome into another?*

Genome Rearrangements

- Not only genes undergo mutation
→ Genome mutations, or *genome rearrangements*
- Many rearrangement **types** have been considered
→ Inversion, Transposition, inverse Transposition, Cut and Join, ...
- **Sorting Problem:** *Given a set of rearrangement types, what is the shortest sequence of those operations to transform one genome into another?*
- Many rearrangement **models** have been considered
→ Inversion + Transposition, Transposition + inverse Transposition, ...
→ Depending on rearrangement types considered, sorting problem is NP-hard

Genome Rearrangements

- Not only genes undergo mutation
  → Genome mutations, or *genome rearrangements*
- Many rearrangement **types** have been considered
  → Inversion, Transposition, inverse Transposition, Cut and Join, ...
- **Sorting Problem:** *Given a set of rearrangement types, what is the shortest sequence of those operations to transform one genome into another?*
- Many rearrangement **models** have been considered
  → Inversion + Transposition, Transposition + inverse Transposition, ...
  → Depending on rearrangement types considered, sorting problem is NP-hard
- The **Tandem Duplication Random Loss/inverse Tandem Duplication Random Loss model** can mimic most "popular" rearrangements, and we developed a polynomial time algorithm for the sorting problem with TDRL/iTDRL

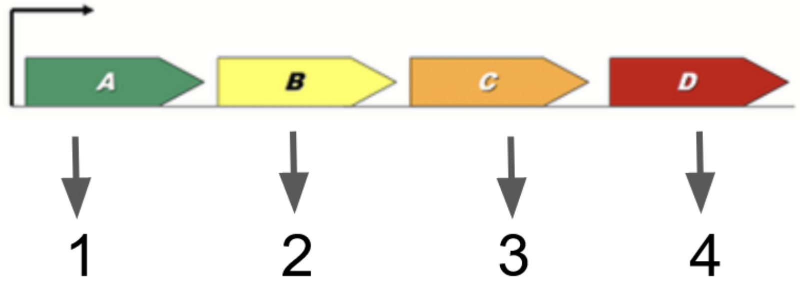
Genome Model

Genome Model

- Unichromosomal Genomes can be modelled by permutations

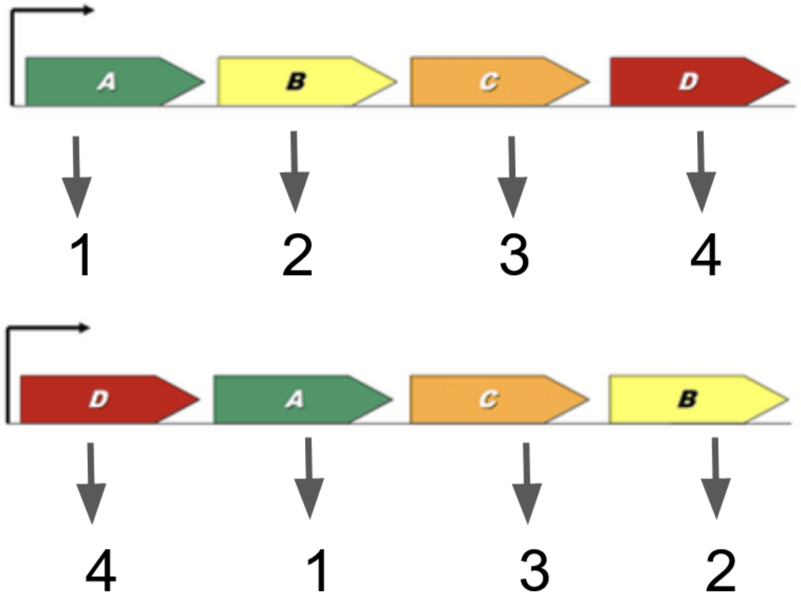
Genome Model

- Unichromosomal Genomes can be modelled by permutations



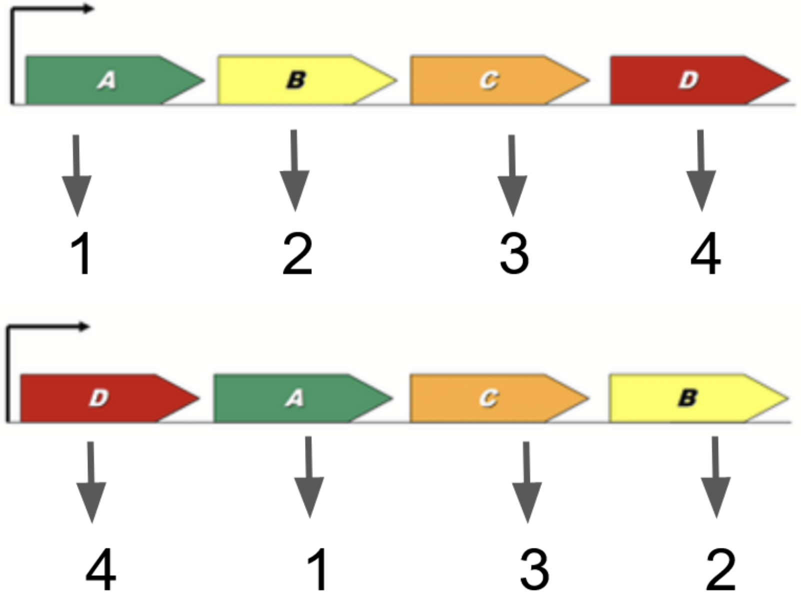
Genome Model

- Unichromosomal Genomes can be modelled by permutations



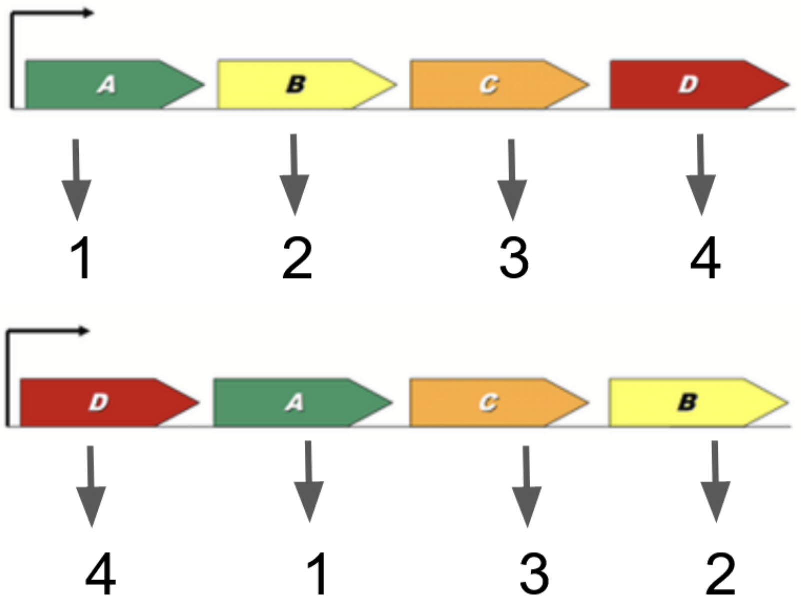
Genome Model

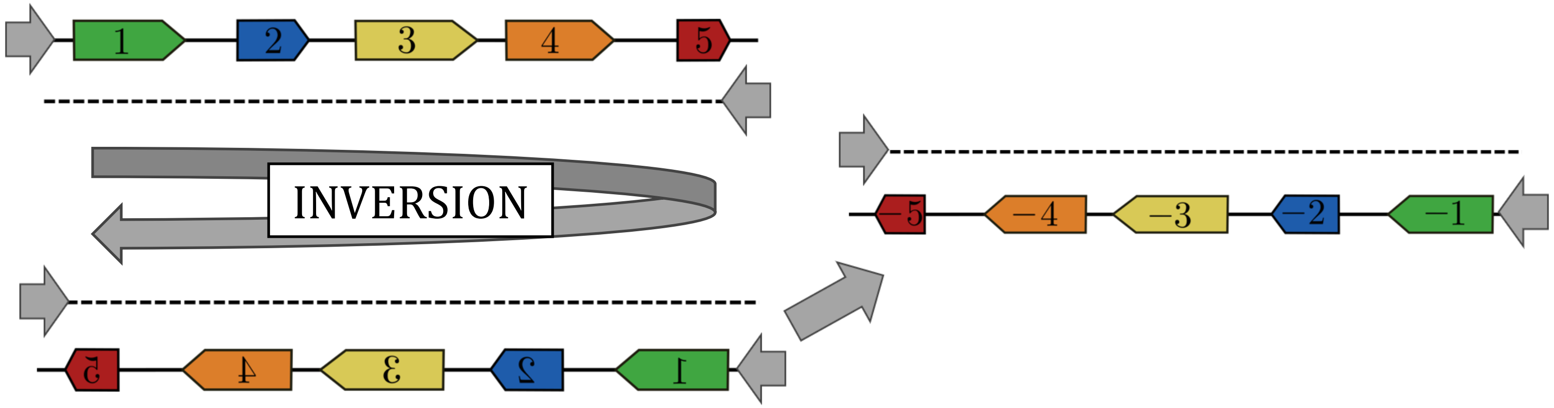
- Unichromosomal Genomes can be modelled by permutations
- "Strandedness" can be modelled by *signed permutations*
  → Important when considering inversion-like mutations



Genome Model

- Unichromosomal Genomes can be modelled by permutations
- "Strandedness" can be modelled by *signed permutations*
  → Important when considering inversion-like mutations



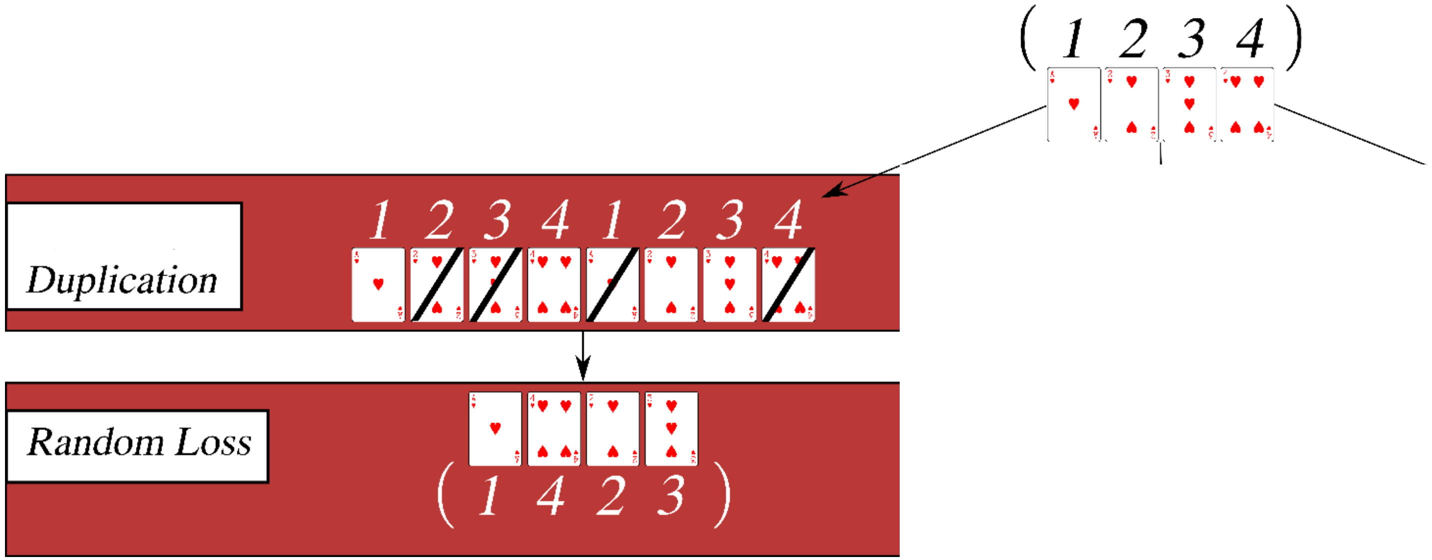


TDRL and iTDRL?

- ***T**andem **D**uplication **R**andom **L**oss* (TDRL) and
  ***i**nverse **T**andem **D**uplication **R**andom **L**oss* (iTDRL)
  → Duplication of genome/permutation (inverted for iTDRL)
  → Followed by random loss of one copy for each gene

TDRL and iTDRL?

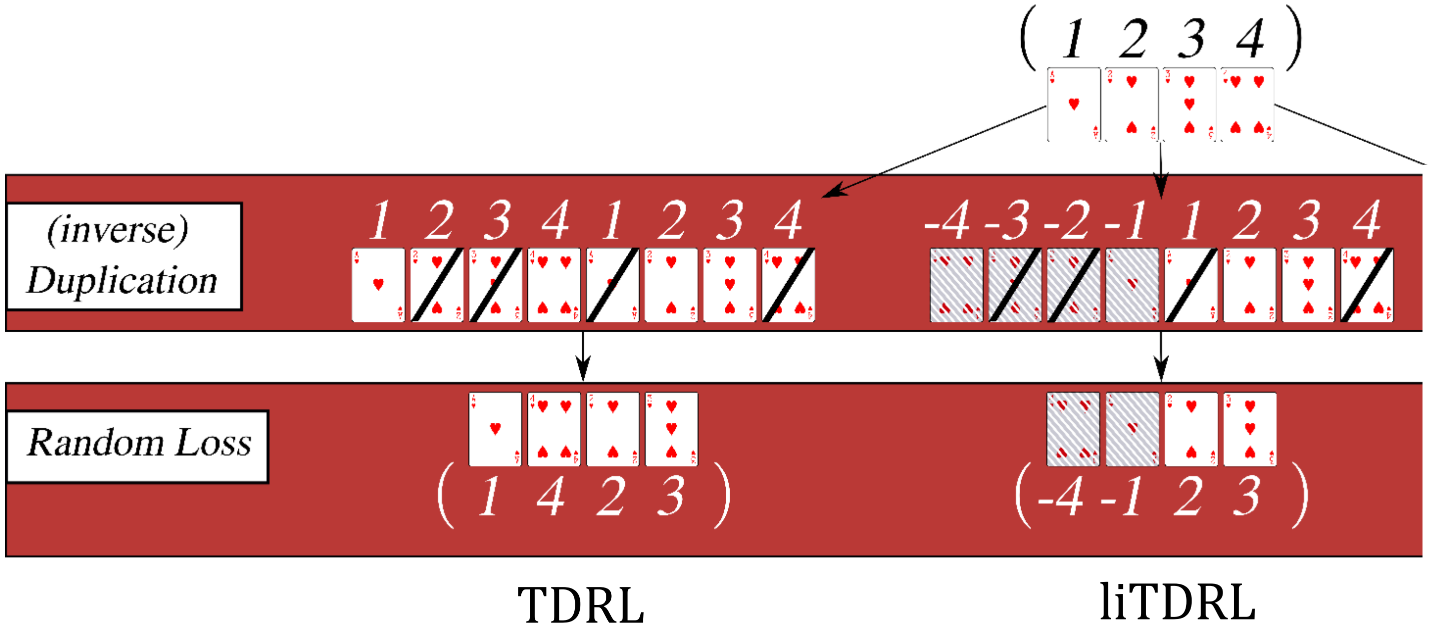
- ***T**andem **D**uplication **R**andom **L**oss* (TDRL) and
  ***i**nverse **T**andem **D**uplication **R**andom **L**oss* (iTDRL)
  → Duplication of genome/permutation (inverted for iTDRL)
  → Followed by random loss of one copy for each gene



TDRL

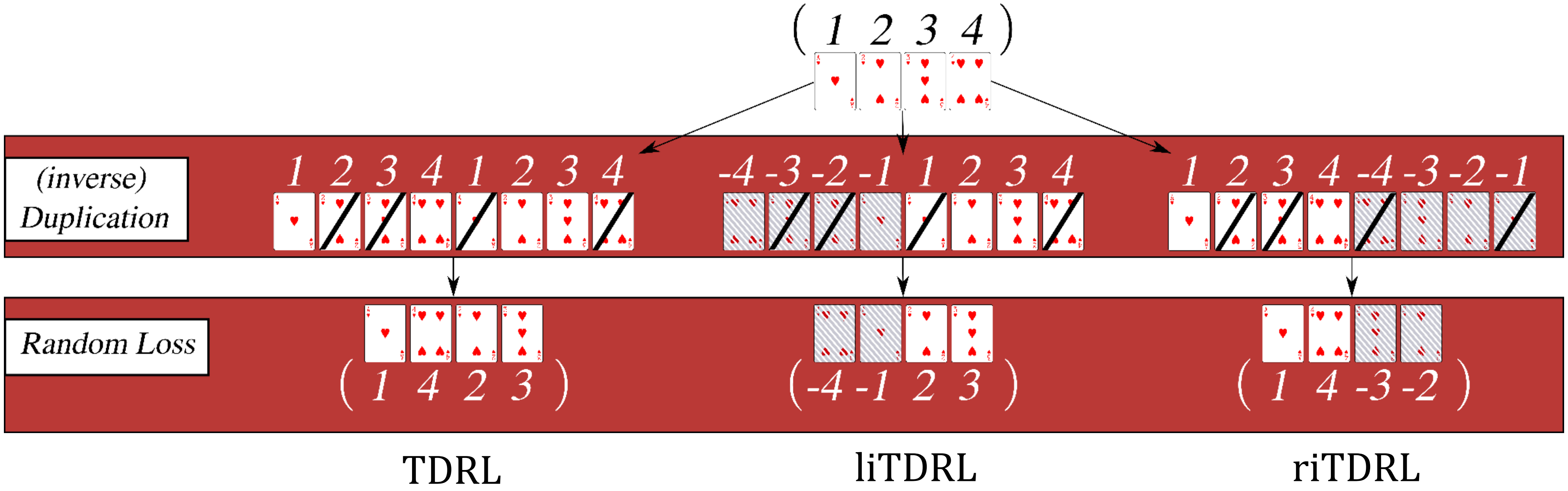
TDRL and iTDRL?

- ***T**andem **D**uplication **R**andom **L**oss* (TDRL) and
  ***i**nverse **T**andem **D**uplication **R**andom **L**oss* (iTDRL)
  → Duplication of genome/permutation (inverted for iTDRL)
  → Followed by random loss of one copy for each gene



TDRL and iTDRL?

- ***T****andem* ***D****uplication* ***R****andom* ***L****oss* (TDRL) and
  i*nverse* ***T****andem* ***D****uplication* ***R****andom* ***L****oss* (iTDRL)
  → Duplication of genome/permutation (inverted for iTDRL)
  → Followed by random loss of one copy for each gene



Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- “*maximal increasing sign-consistent substrings*” of a permutation:
  - → **Substring:** Appear as substring of the permutation

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order
  →***Maximal***: Cannot be extended (to left or right) in permutation

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order
  →***Maximal***: Cannot be extended (to left or right) in permutation
  →**Sign-Consistent:** Contain only positive or negative elements

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  - → **Substring:** Appear as substring of the permutation
  - →**Increasing**: Elements of appear in increasing/ascending order
  - →***Maximal***: Cannot be extended (to left or right) in permutation
  - →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\ 3\ \ 7\ \ -6\ \ -2\ \ -1\ \ 4\ \ 5\ \ -8\ )$$

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order
  →***Maximal***: Cannot be extended (to left or right) in permutation
  →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\ \boxed{3\ \ 7}\ -6\ \ -2\ \ -1\ \ 4\ \ 5\ \ -8\ )$$

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  → **Increasing**: Elements of appear in increasing/ascending order
  → ***Maximal***: Cannot be extended (to left or right) in permutation
  → **Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\ \boxed{3\ \ 7}\ \boxed{-6\ \ -2\ \ -1}\ 4\ \ 5\ \ -8\ )$$

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  - → **Substring:** Appear as substring of the permutation
  - → **Increasing**: Elements of appear in increasing/ascending order
  - → ***Maximal***: Cannot be extended (to left or right) in permutation
  - → **Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\boxed{3\ 7}\ \boxed{-6\ \ -2\ \ -1}\ \boxed{4\ 5}\ -8\ )$$

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order
  →***Maximal***: Cannot be extended (to left or right) in permutation
  →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\ \boxed{3\ \ 7}\ \boxed{-6\ \ -2\ \ -1}\ \boxed{4\ \ 5}\ \boxed{-8}\ )$$

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  - → **Substring:** Appear as substring of the permutation
  - →**Increasing**: Elements of appear in increasing/ascending order
  - →***Maximal***: Cannot be extended (to left or right) in permutation
  - →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\ \boxed{3\ \ 7}\ \boxed{-6\ \ -2\ \ -1}\ \boxed{4\ \ 5}\ \boxed{-8}\ )$$

- ***MISC-Encoding*** einer Permutation:

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order
  →***Maximal***: Cannot be extended (to left or right) in permutation
  →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\boxed{3\ 7}\ \boxed{-6\ \ -2\ \ -1}\ \boxed{4\ 5}\ \boxed{-8})$$

- ***MISC-Encoding*** einer Permutation:
  - Each MISC-substring is assigned a letter (p or n)

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  → **Substring:** Appear as substring of the permutation
  →**Increasing**: Elements of appear in increasing/ascending order
  →***Maximal***: Cannot be extended (to left or right) in permutation
  →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\boxed{3\ 7}\ \boxed{-6\ \ -2\ \ -1}\ \boxed{4\ 5}\ \boxed{-8})$$

- ***MISC-Encoding*** einer Permutation:
  - Each MISC-substring is assigned a letter (p or n)
  - Capture number, and structure of MISC-substrings

Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

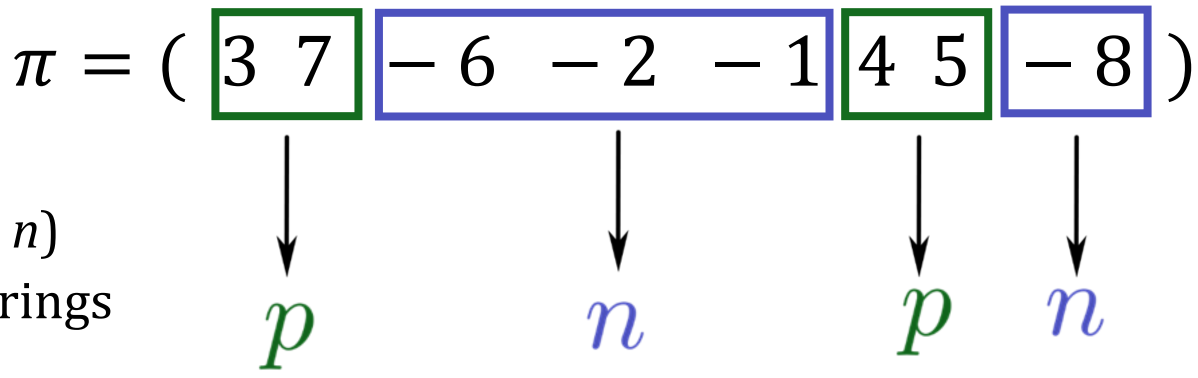
- "*maximal increasing sign-consistent substrings*" of a permutation:
  - → **Substring:** Appear as substring of the permutation
  - →**Increasing**: Elements of appear in increasing/ascending order
  - →***Maximal***: Cannot be extended (to left or right) in permutation
  - →**Sign-Consistent:** Contain only positive or negative elements

$$\pi = (\boxed{3\ 7}\ \boxed{-6\ \ -2\ \ -1}\ \boxed{4\ 5}\ \boxed{-8})$$

- ***MISC-Encoding*** einer Permutation:
  - Each MISC-substring is assigned a letter (*p* or *n*)
  - Capture number, and structure of MISC-substrings
  - **Allow us to ignore pesky integers**

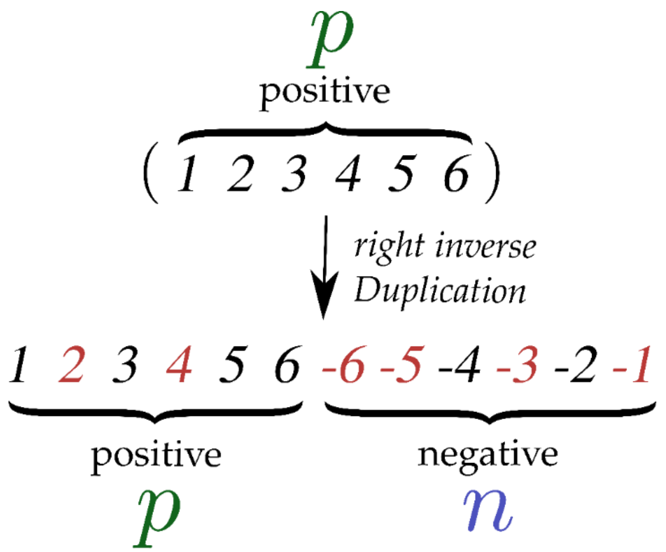
Maximal **I**ncreasing **S**ign-**C**onsistent Substrings (MISC)

- "*maximal increasing sign-consistent substrings*" of a permutation:
  - → **Substring:** Appear as substring of the permutation
  - → **Increasing**: Elements of appear in increasing/ascending order
  - → ***Maximal***: Cannot be extended (to left or right) in permutation
  - → **Sign-Consistent:** Contain only positive or negative elements

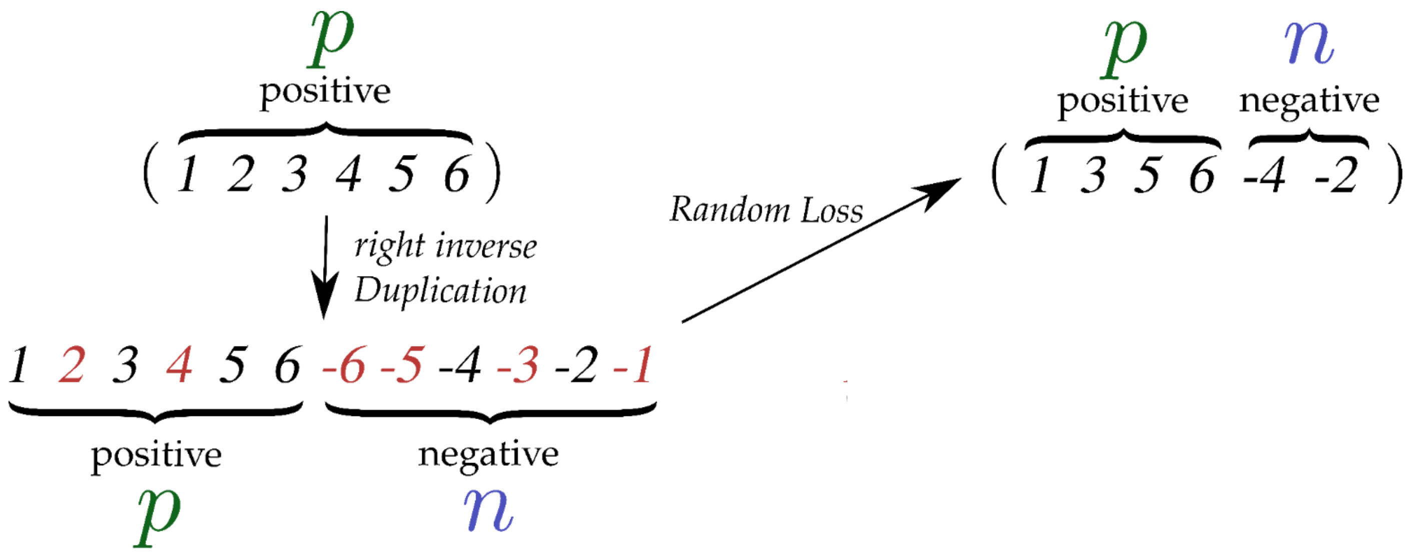


- ***MISC-Encoding*** einer Permutation:
  - Each MISC-substring is assigned a letter (*p* or *n*)
  - Capture number, and structure of MISC-substrings
  - **Allow us to ignore pesky integers**

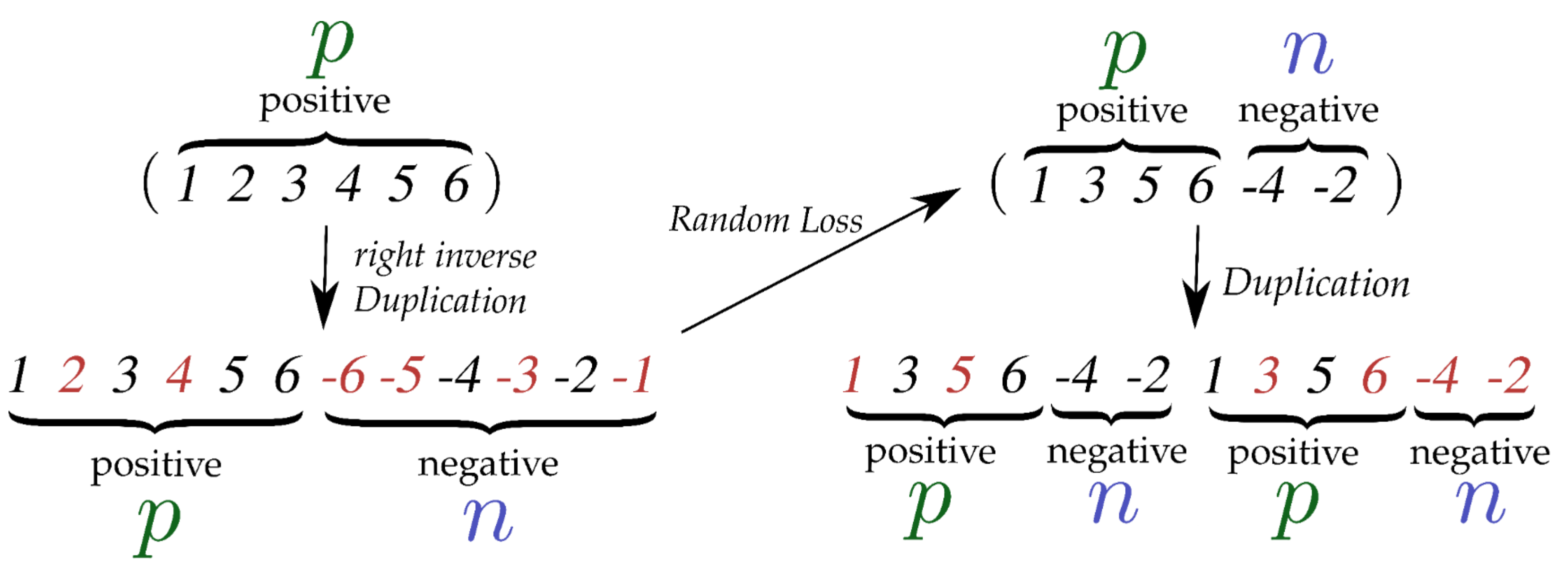
TDRL, iTDRL, and MISC-substrings



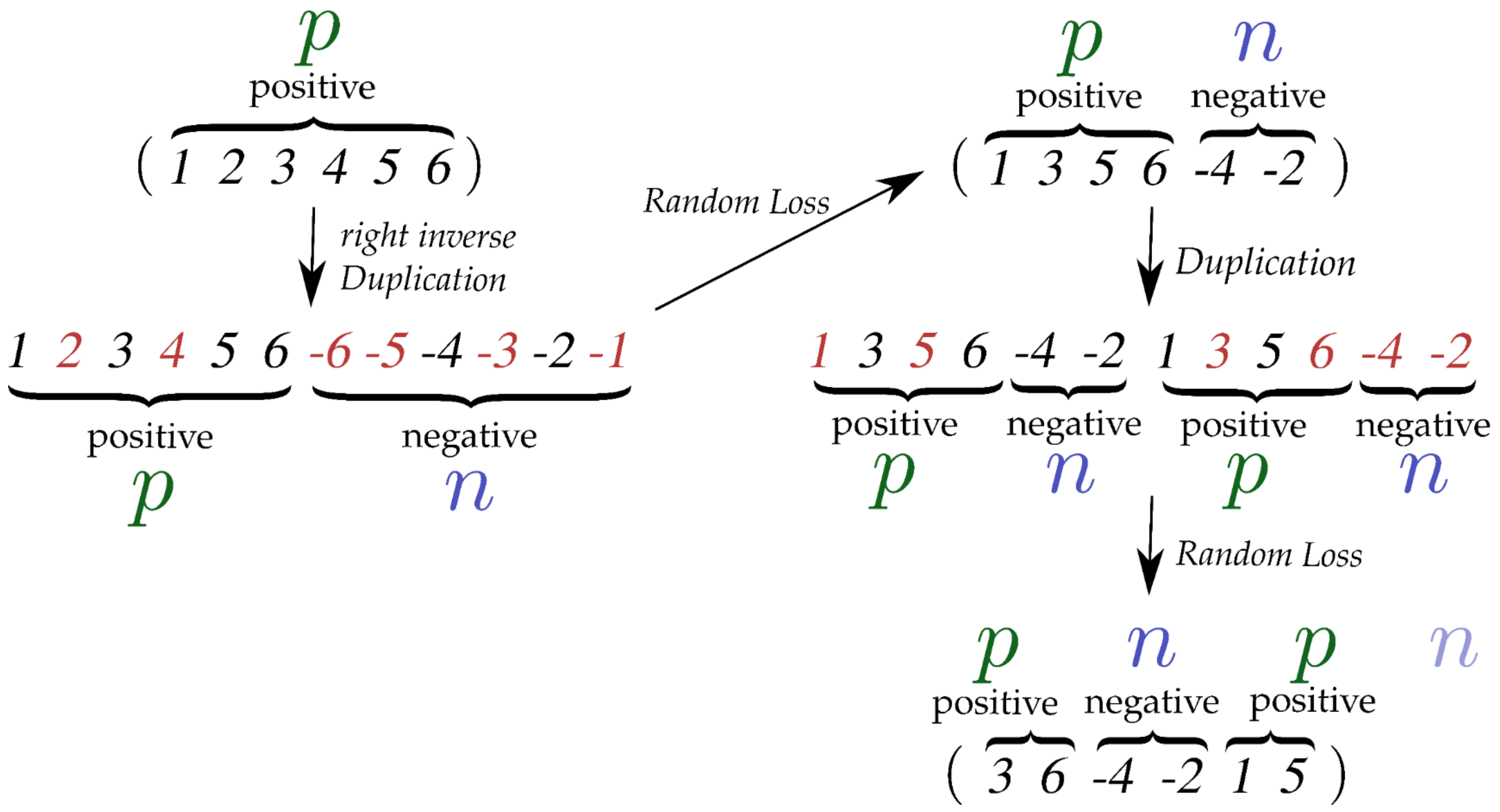
TDRL, iTDRL, and MISC-substrings



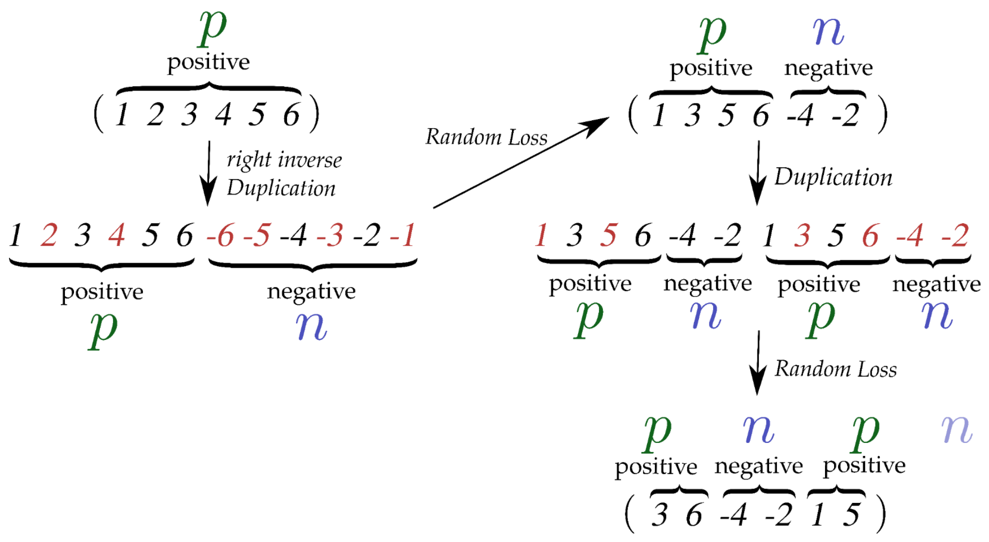
TDRL, iTDRL, and MISC-substrings



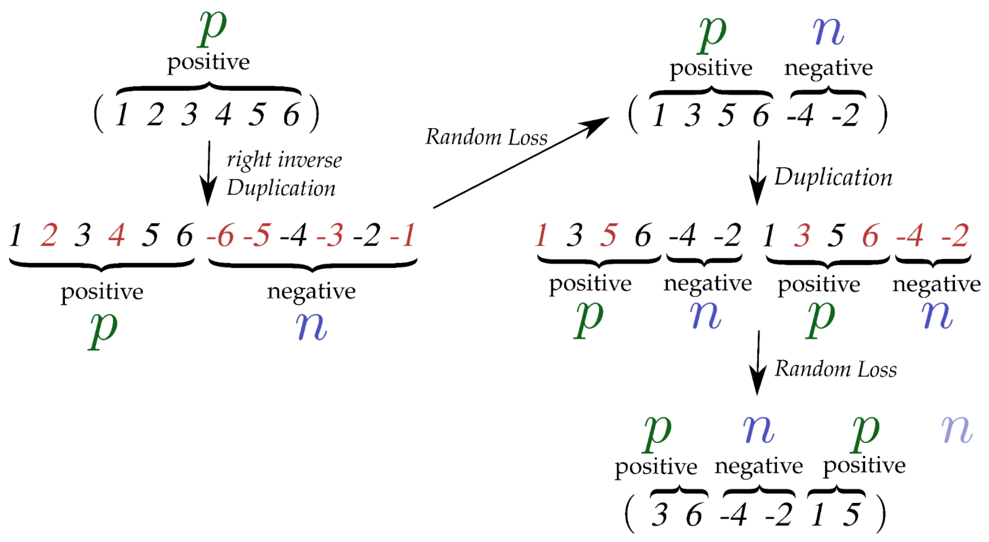
TDRL, iTDRL, and MISC-substrings



MISC-Substring Patterns

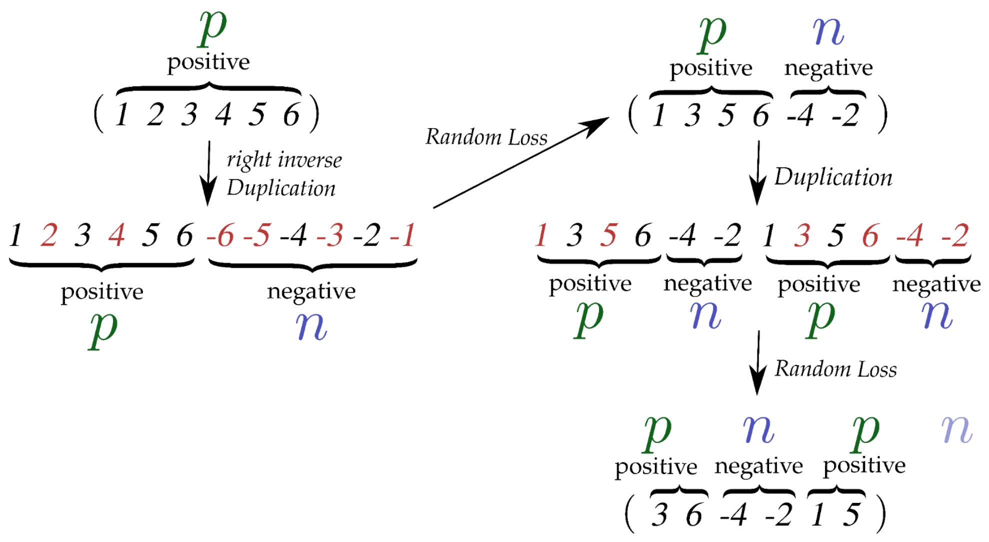


MISC-Substring Patterns

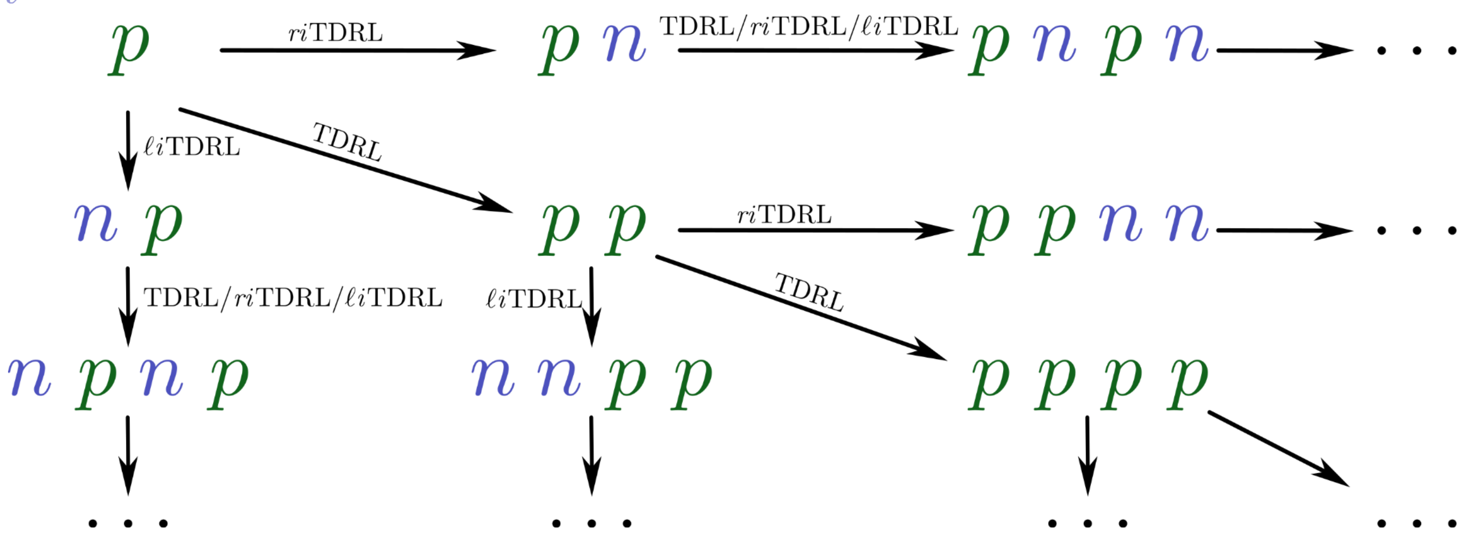


- MISC-encoding of permutation after applying TDRL/iTDRL continuously follows a pattern
  →different orders of applying TDRL/iTDRL yields different ***MISC-Substring Patterns***

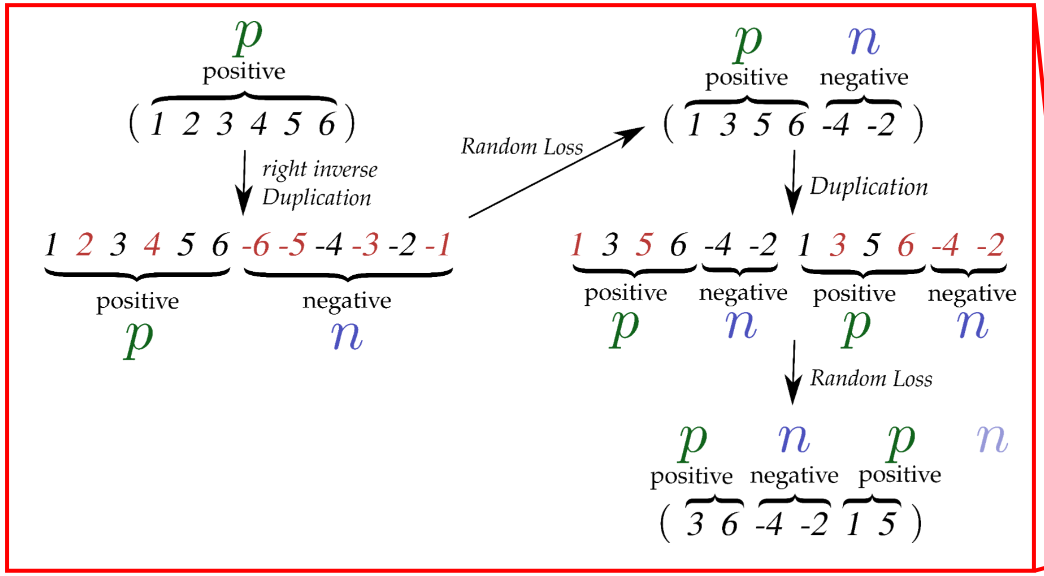
MISC-Substring Patterns



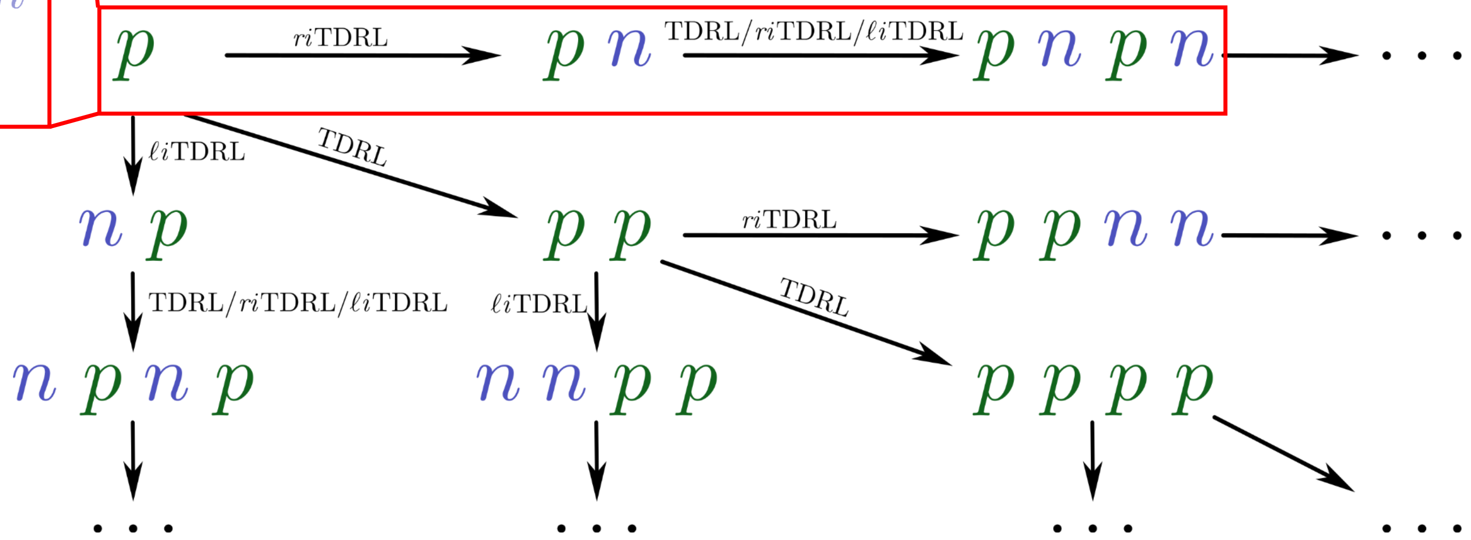
- MISC-encoding of permutation after applying TDRL/iTDRL continuously follows a pattern
  →different orders of applying TDRL/iTDRL yields different ***MISC-Substring Patterns***



MISC-Substring Patterns

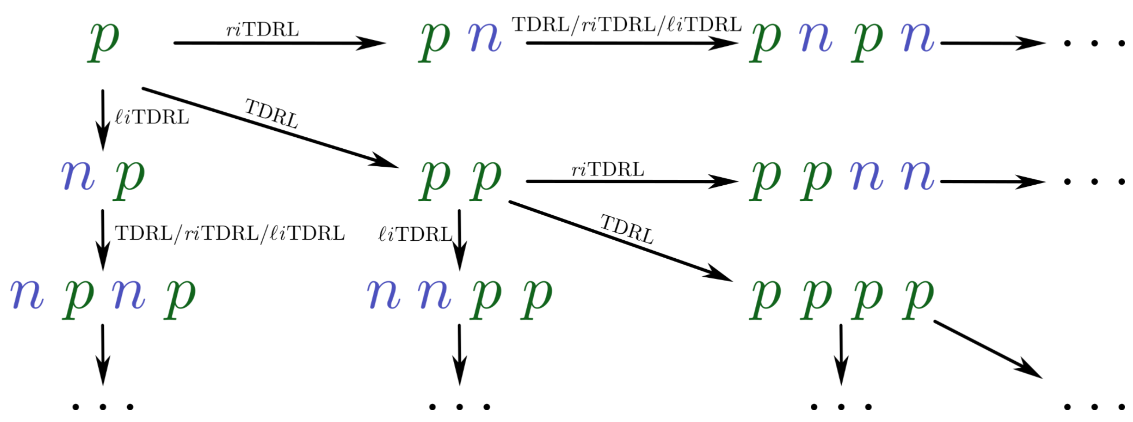


- MISC-encoding of permutation after applying TDRL/iTDRL continuously follows a pattern
  →different orders of applying TDRL/iTDRL yields different ***MISC-Substring Patterns***



An Algorithm

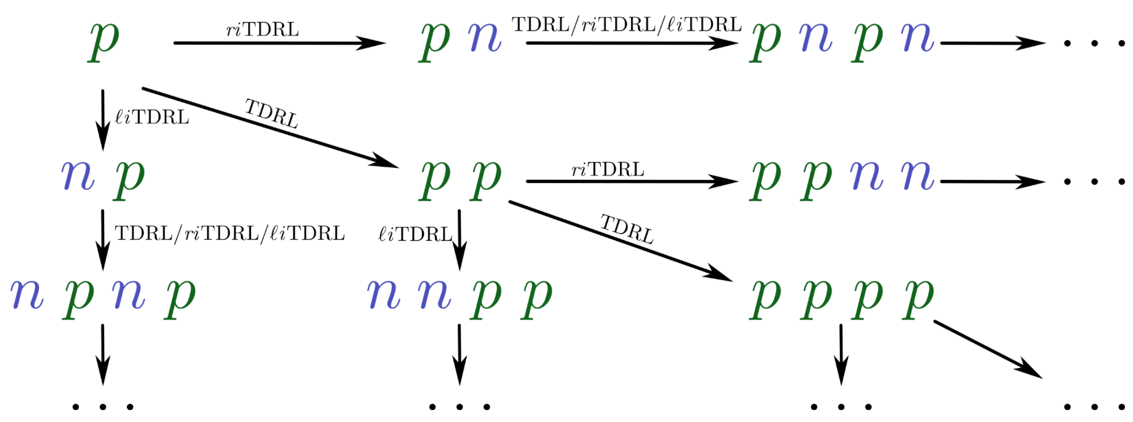
***Algorithm – Sketch*:**



An Algorithm

***Algorithm – Sketch*:**

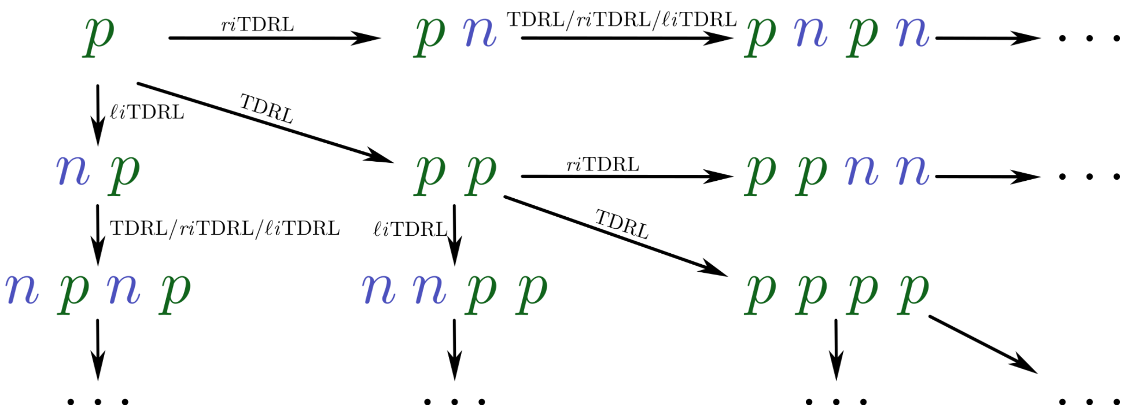
1. Greedy matching to fitting pattern.



An Algorithm

***Algorithm – Sketch*:**

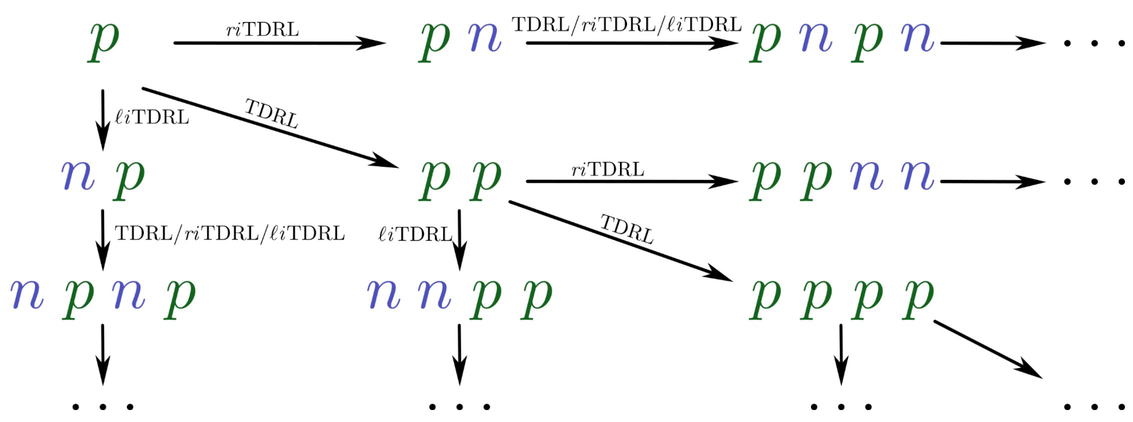
1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.



An Algorithm

***Algorithm – Sketch*:**

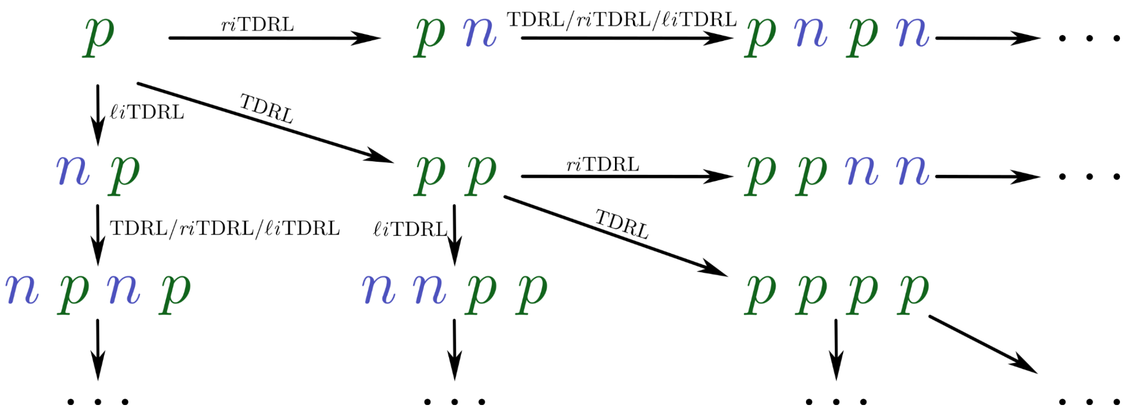
1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.



An Algorithm

***Algorithm – Sketch*:**

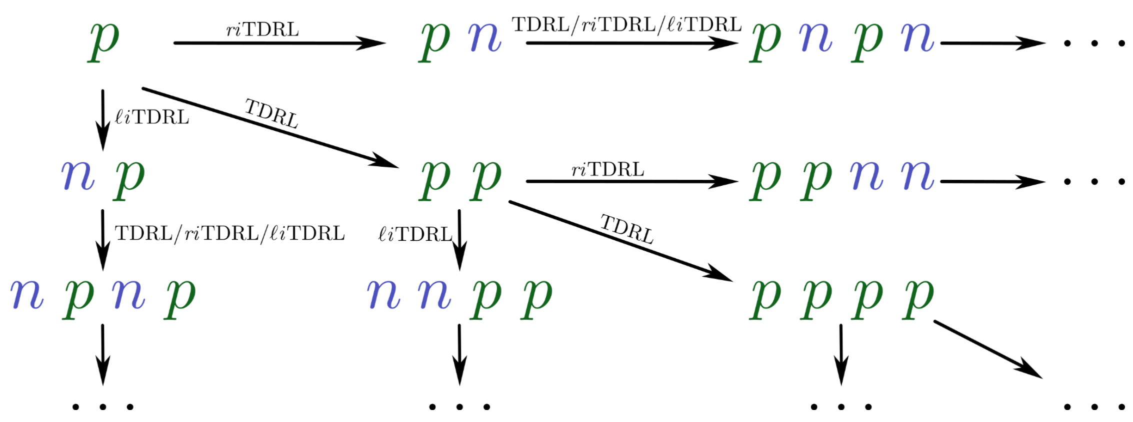
1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.



An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

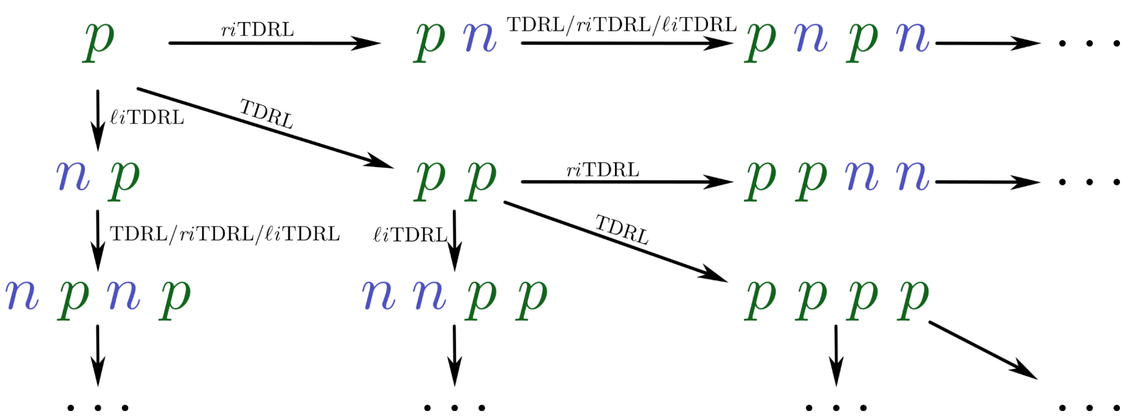


An Algorithm

$n\ n\ p\ p\ n\ p$

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.



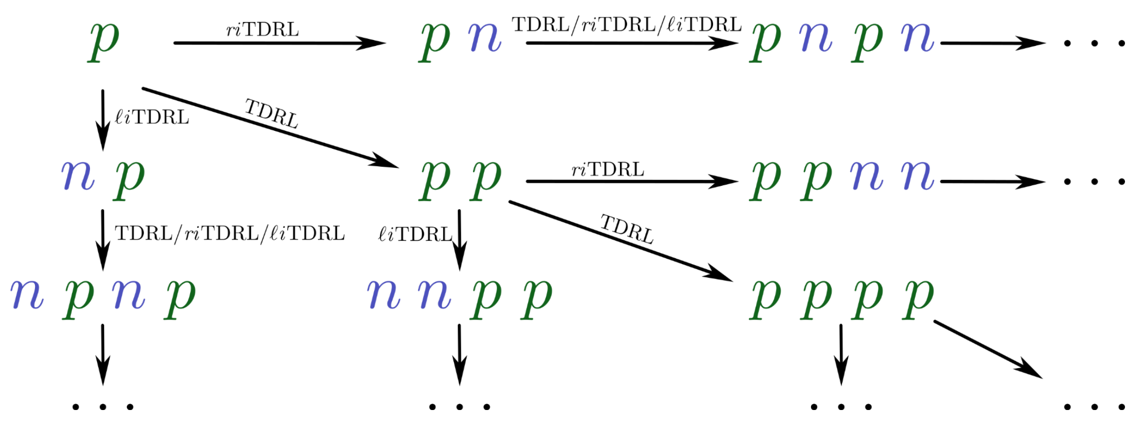
An Algorithm

$n\ n\ p\ p\ n\ p$

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

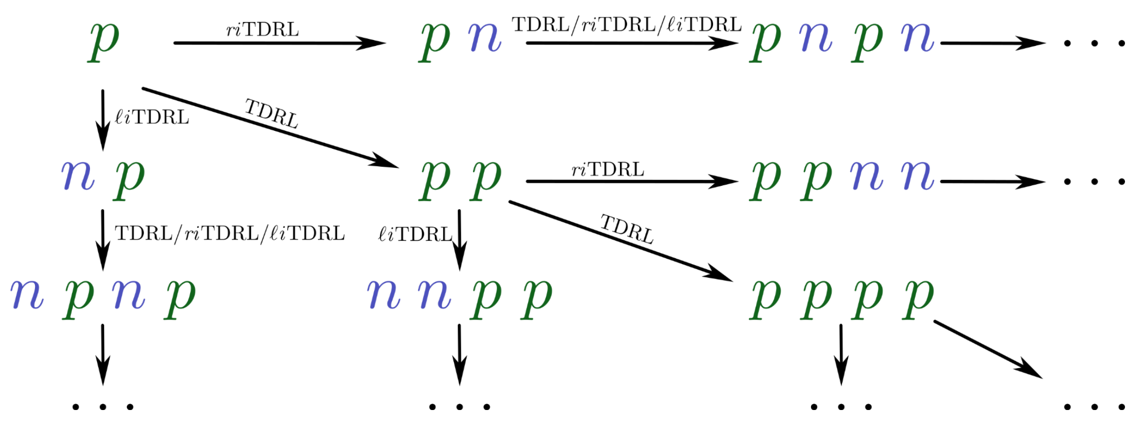
$n\ p\ n\ p \quad n\ p\ n\ p$

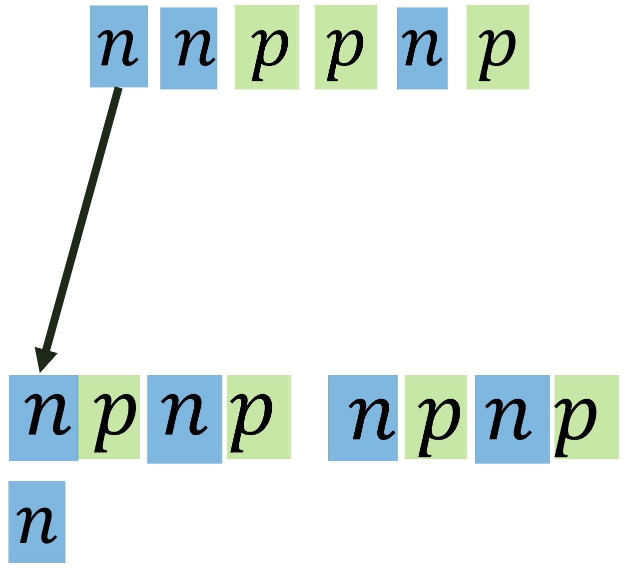


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

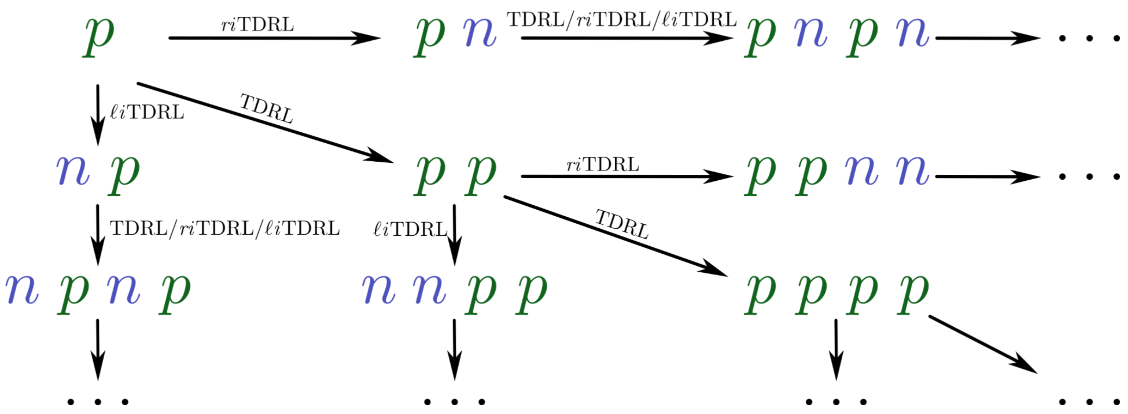


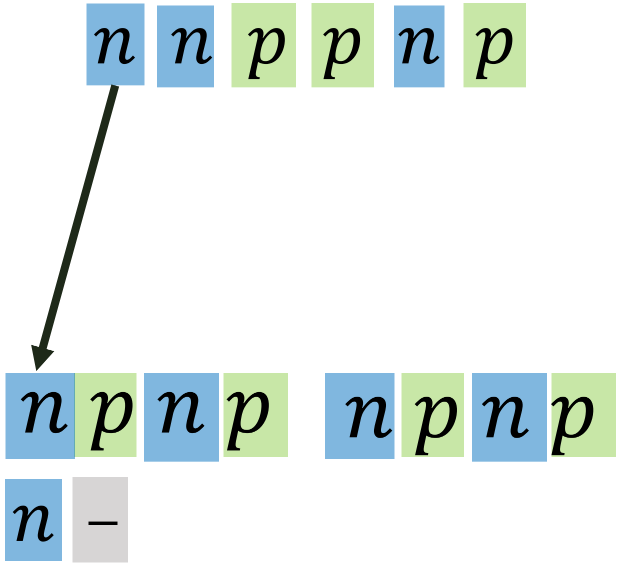


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

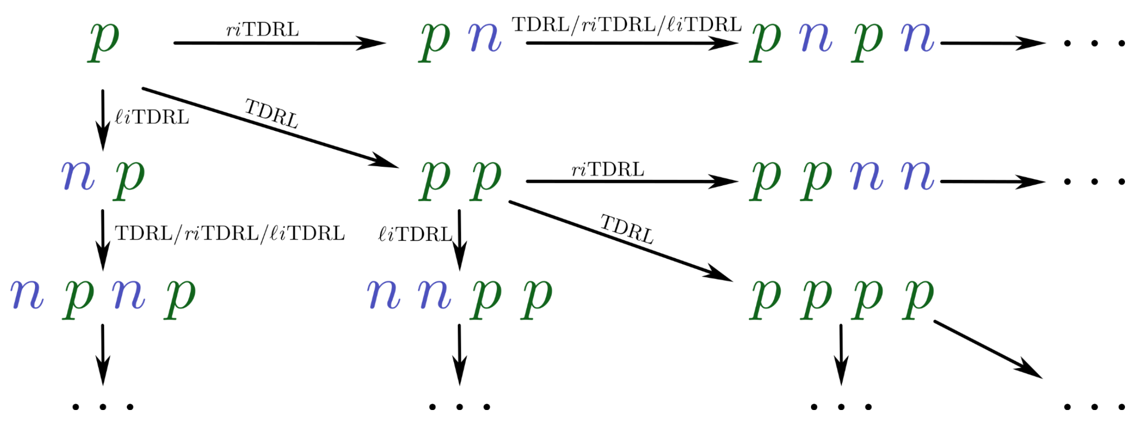


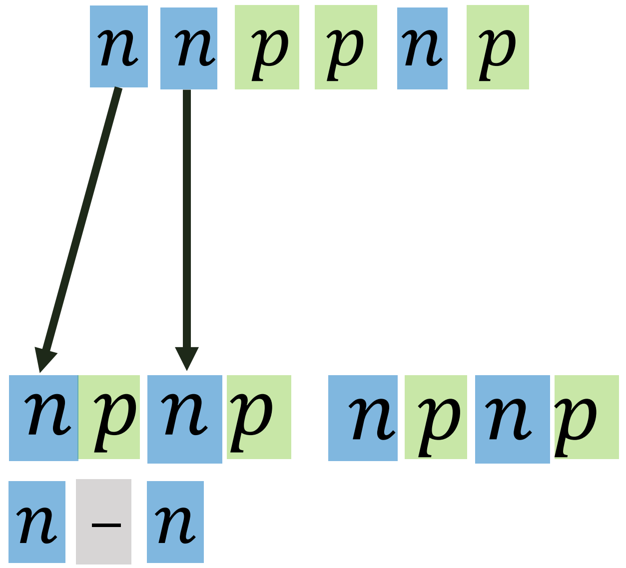


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

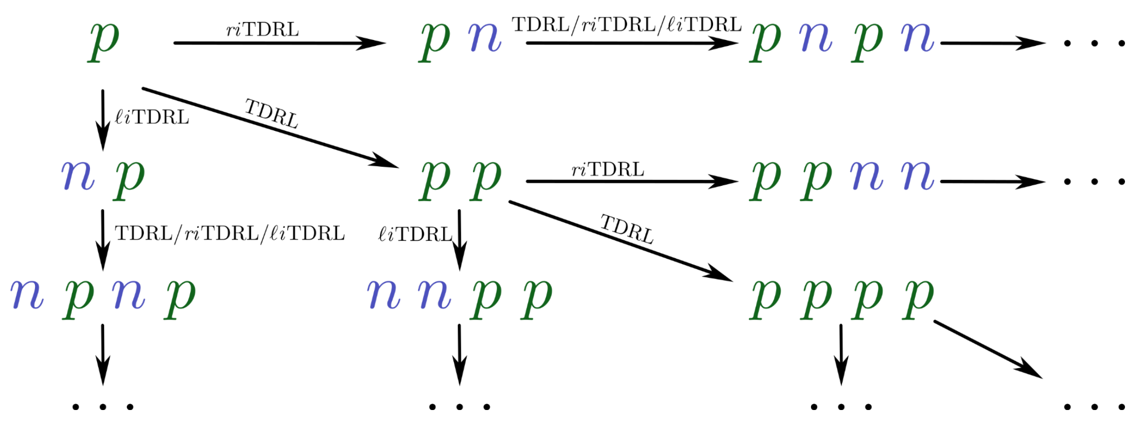


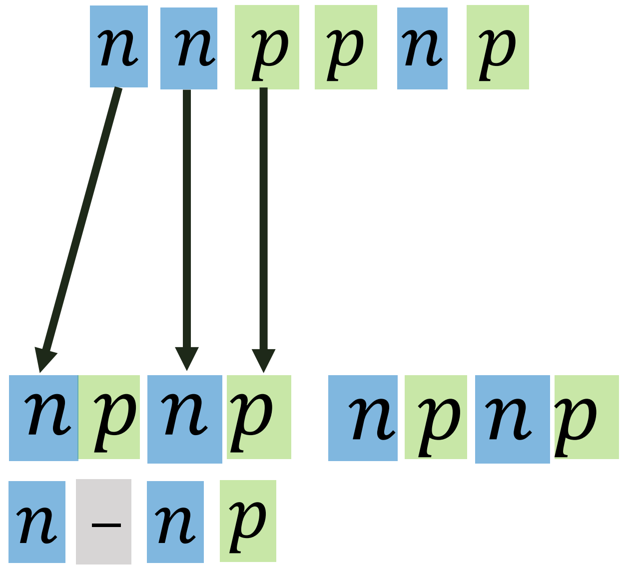


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

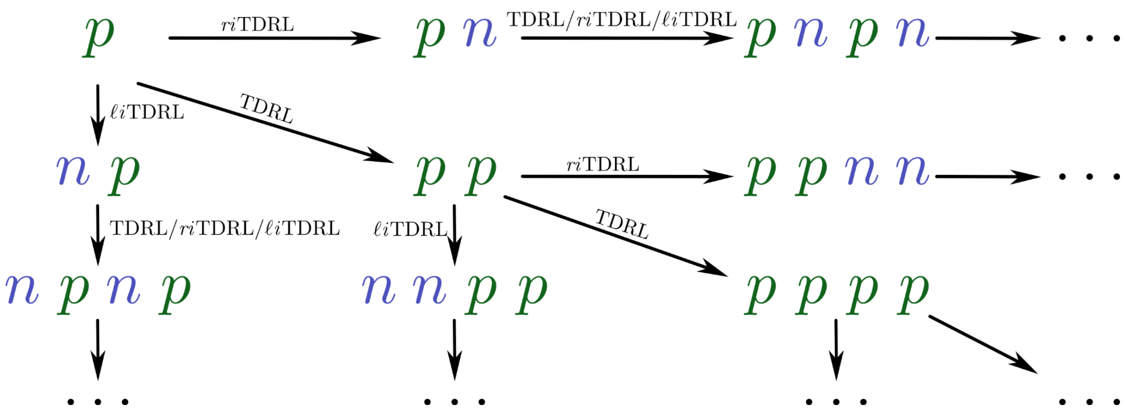


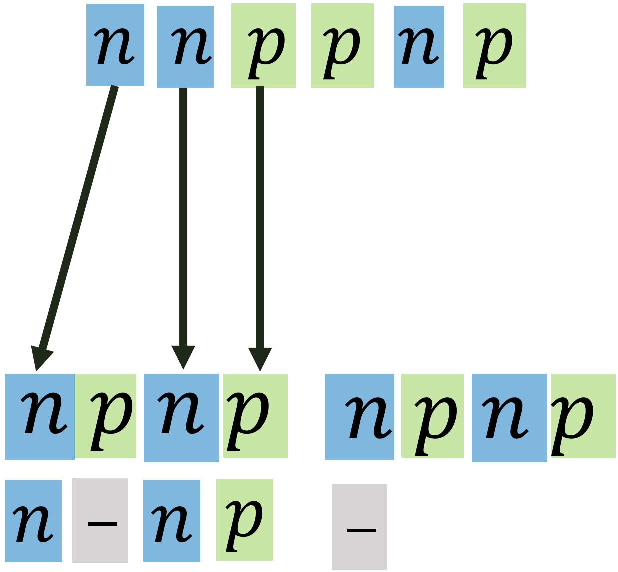


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

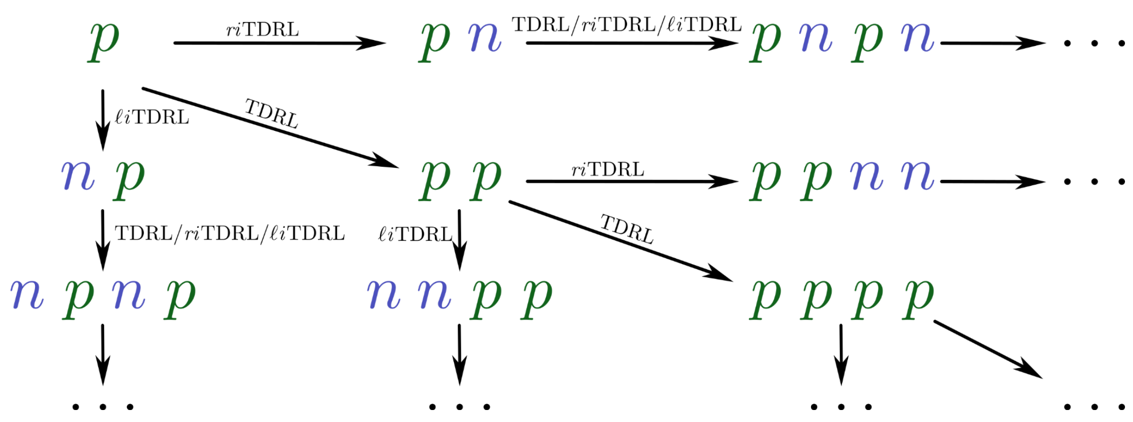


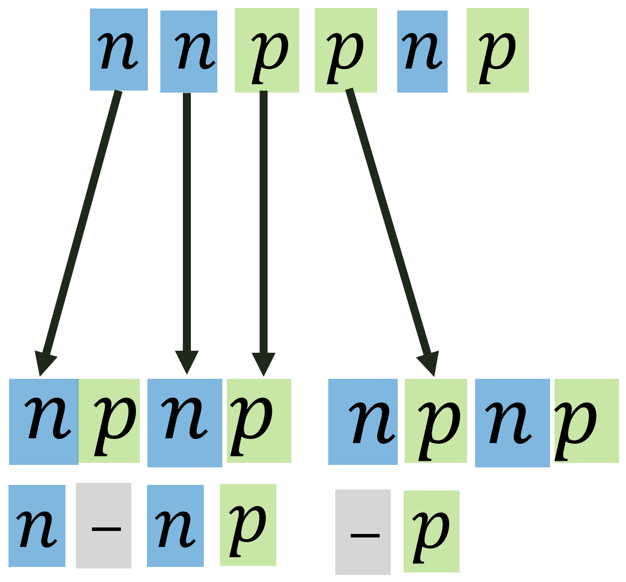


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

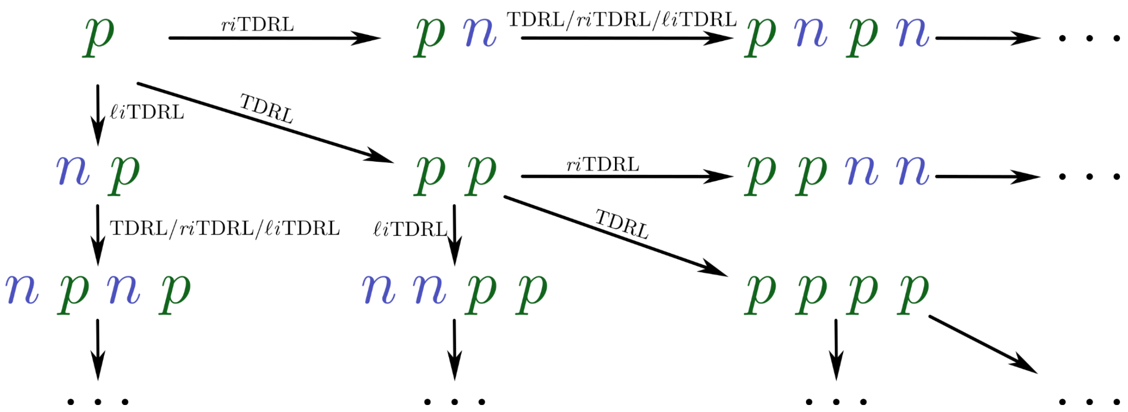


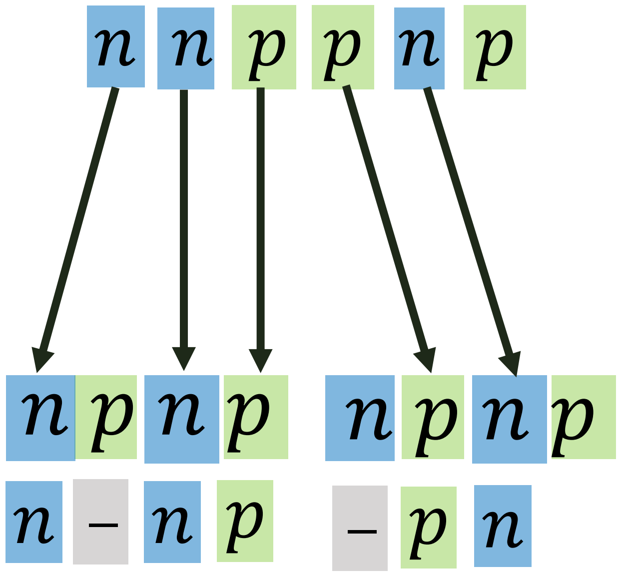


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

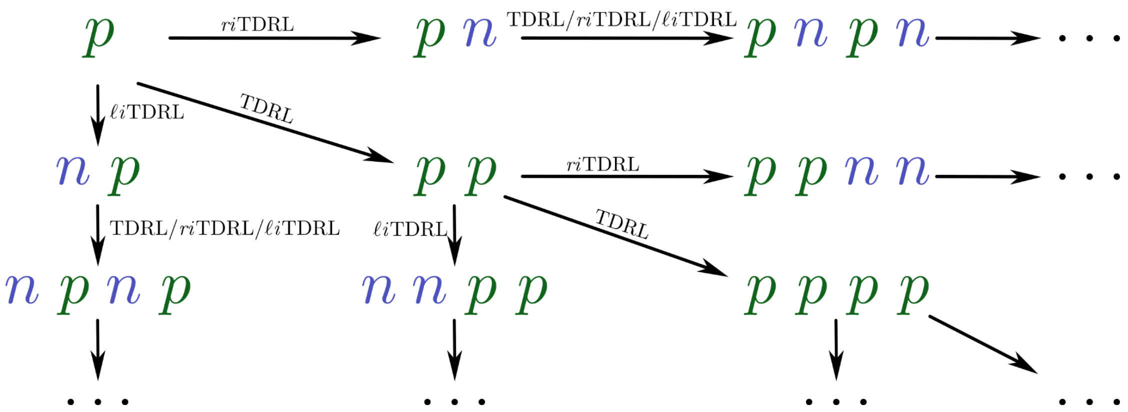


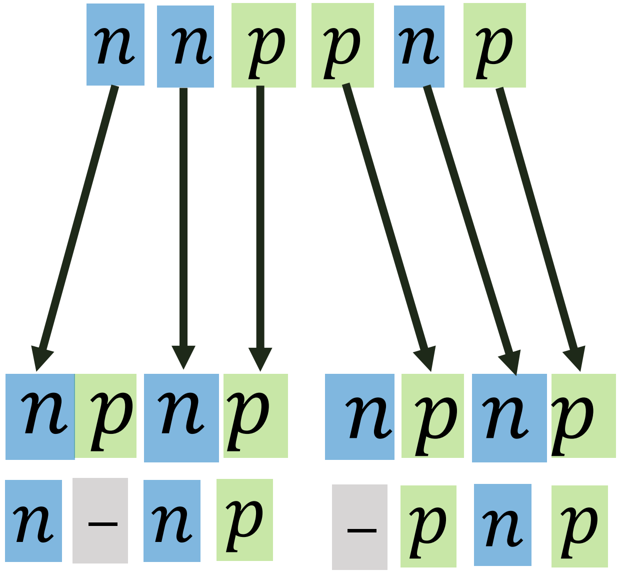


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

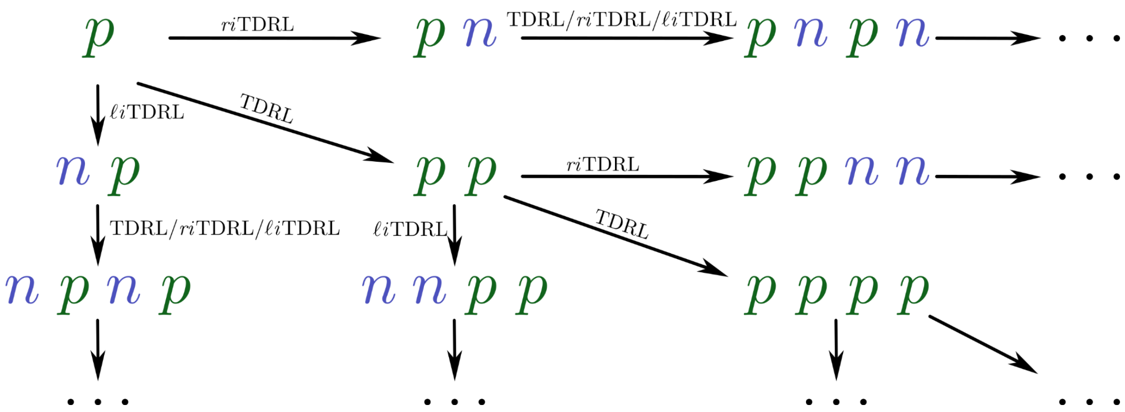


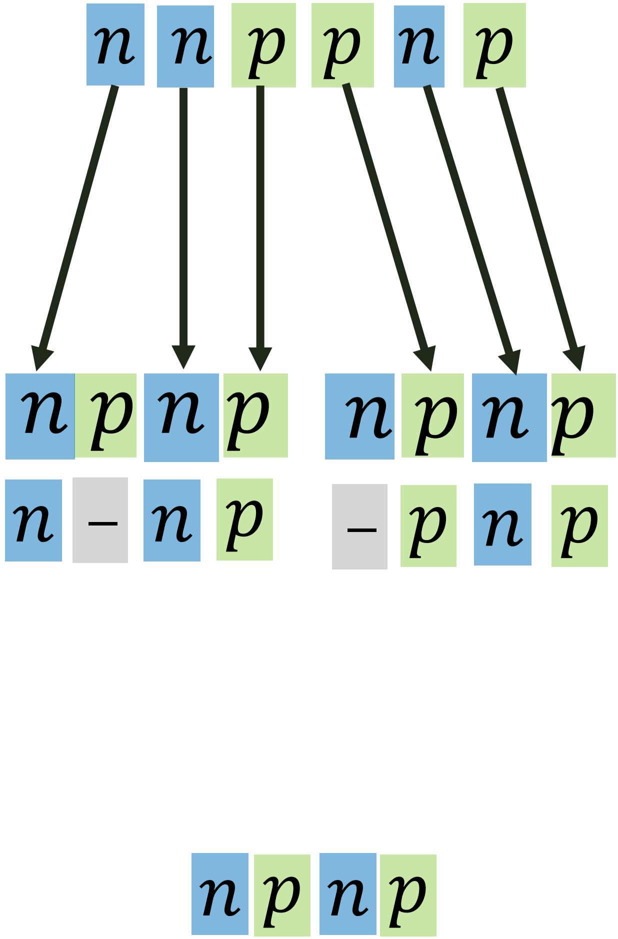


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

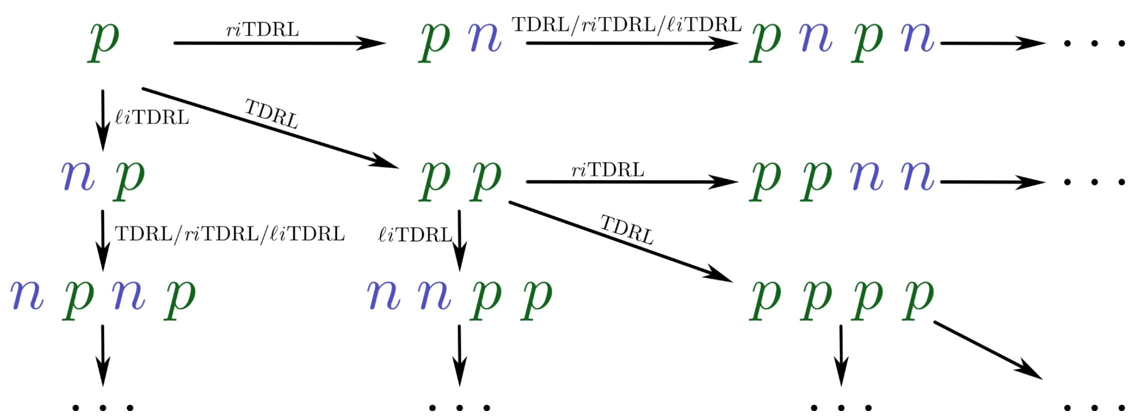


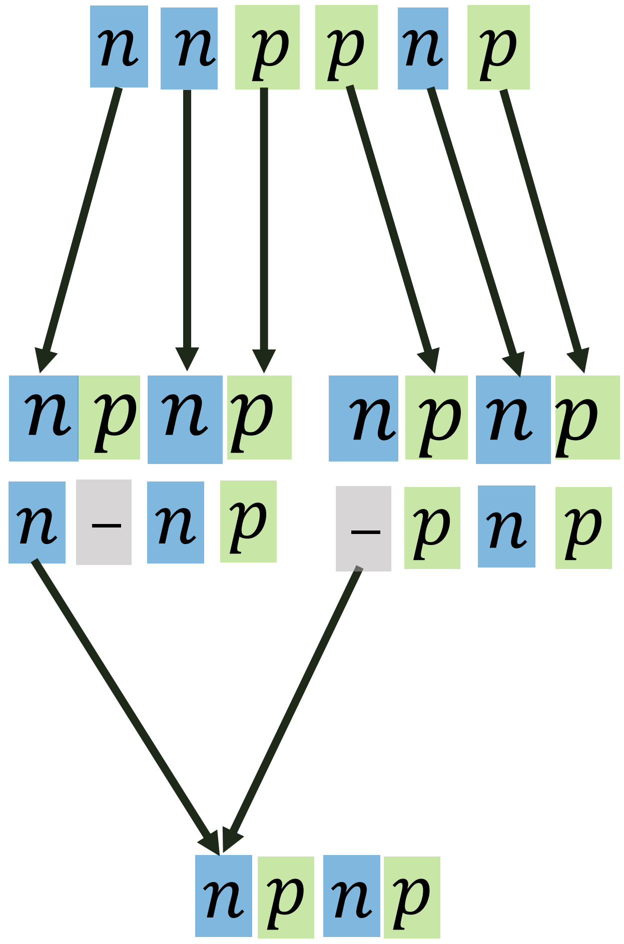


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

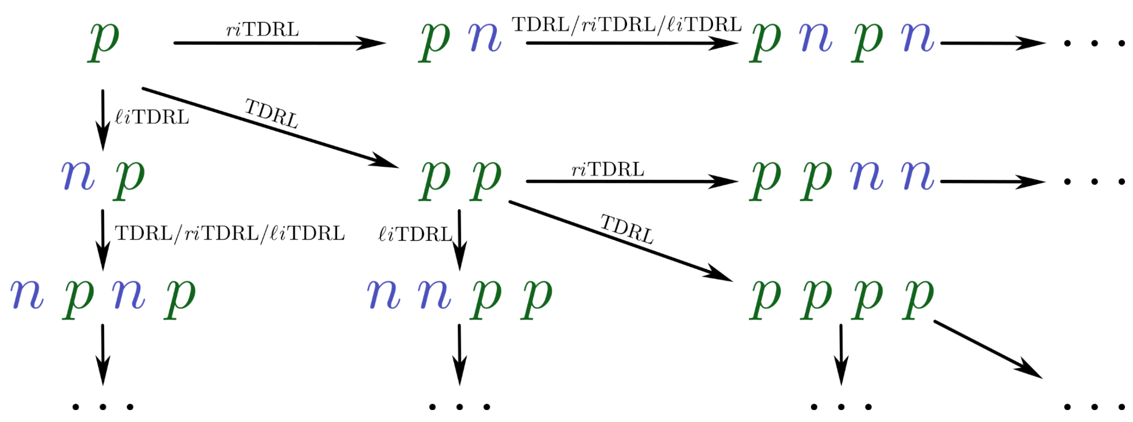


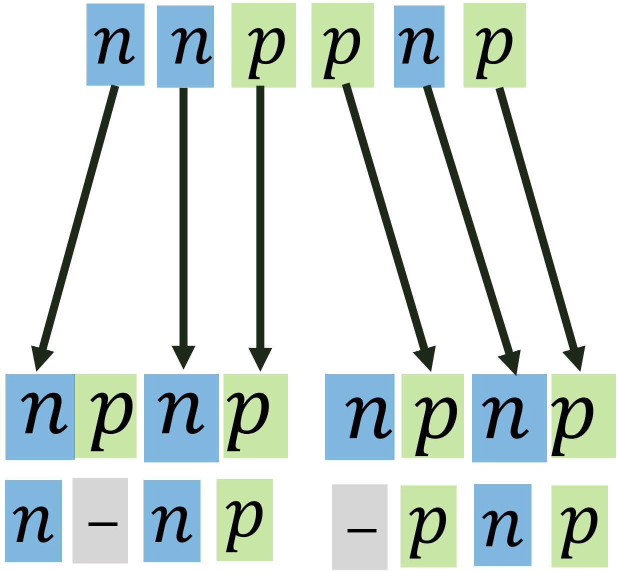


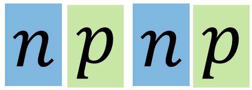
An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.



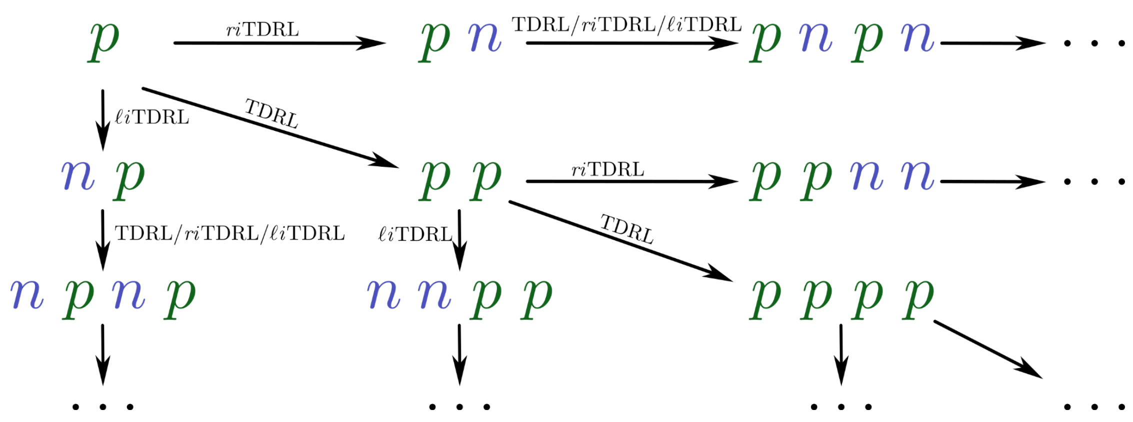


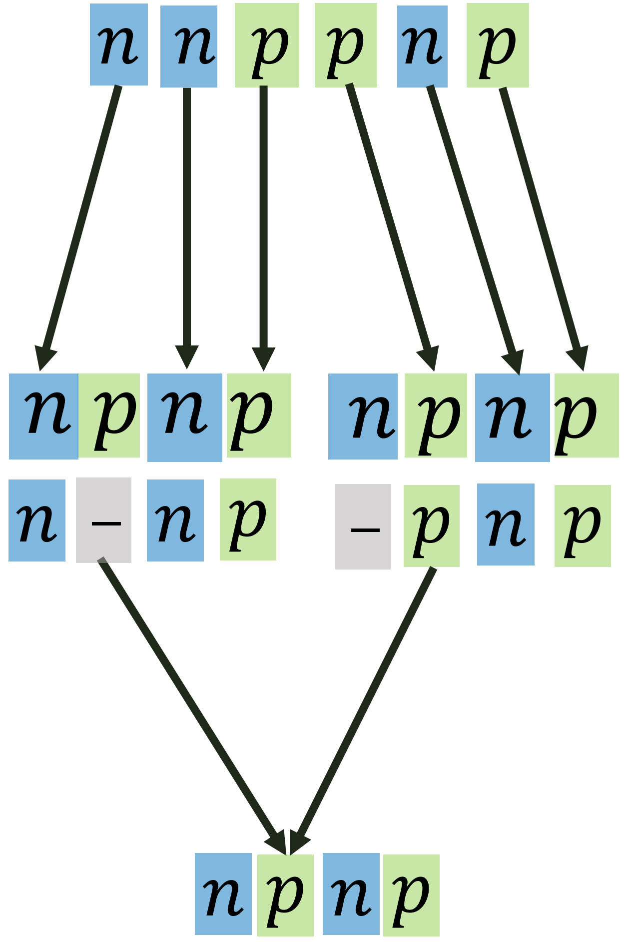


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

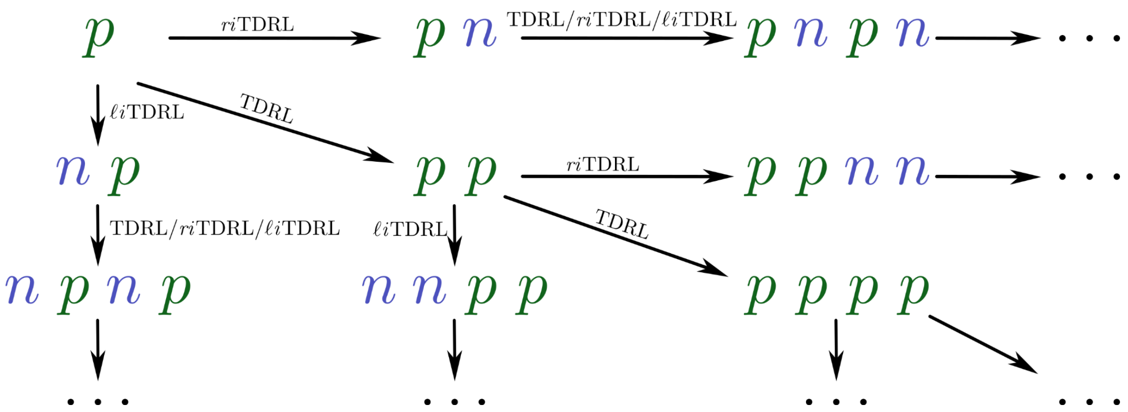


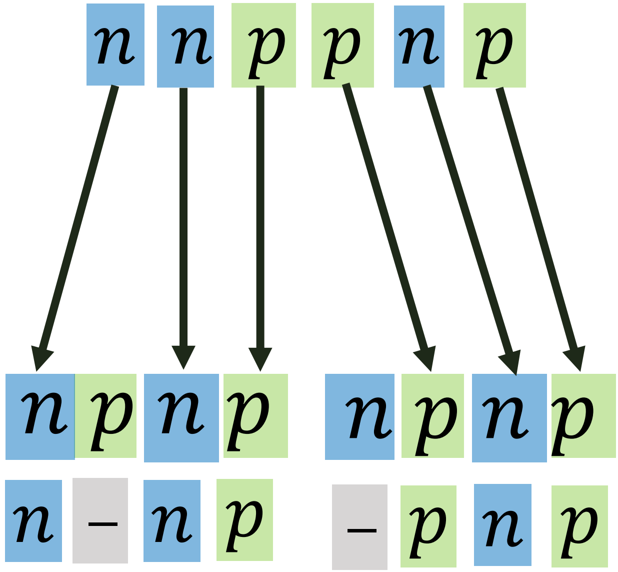


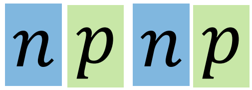
An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.



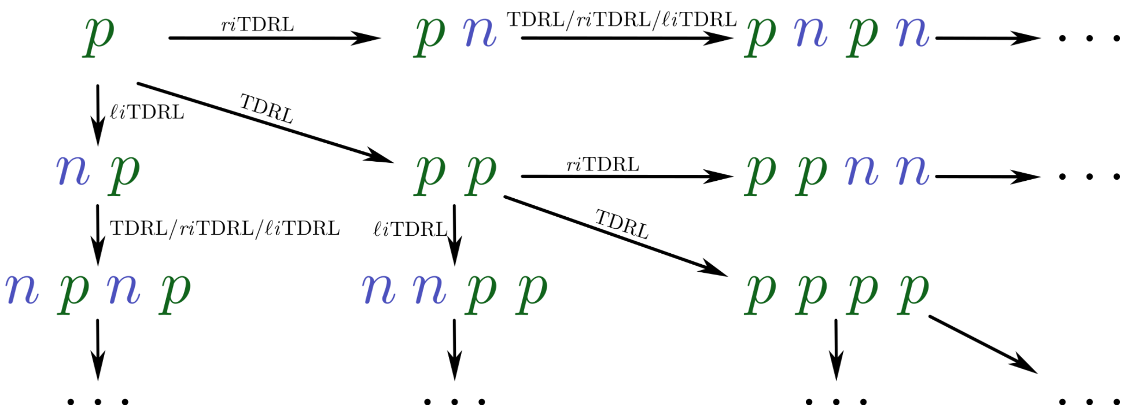


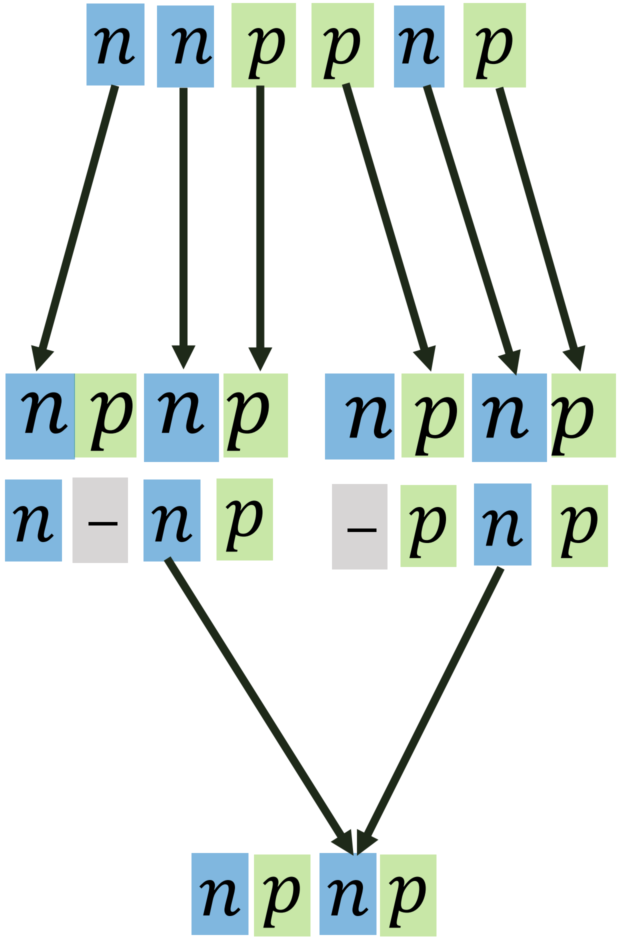


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

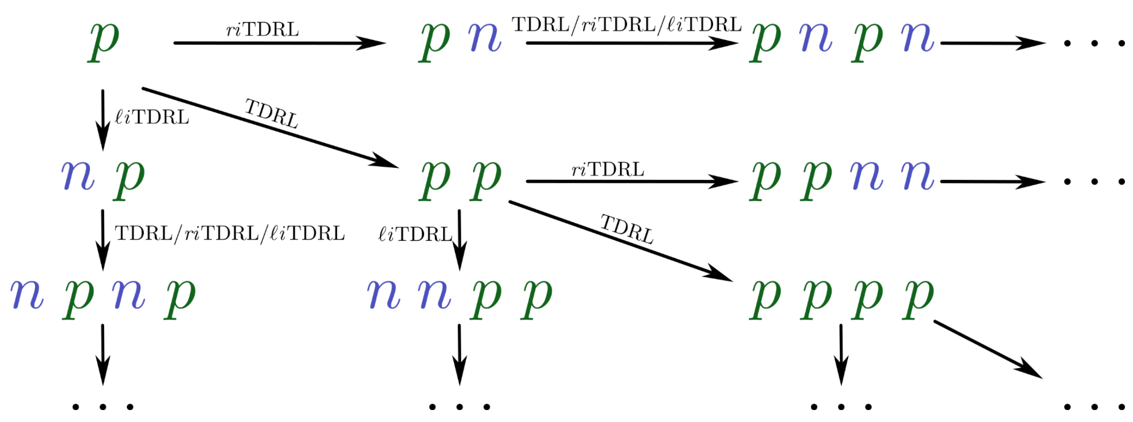


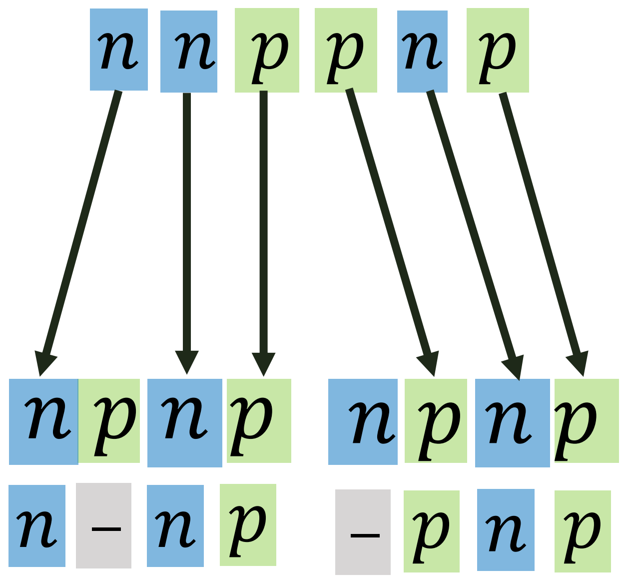


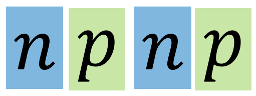
An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.



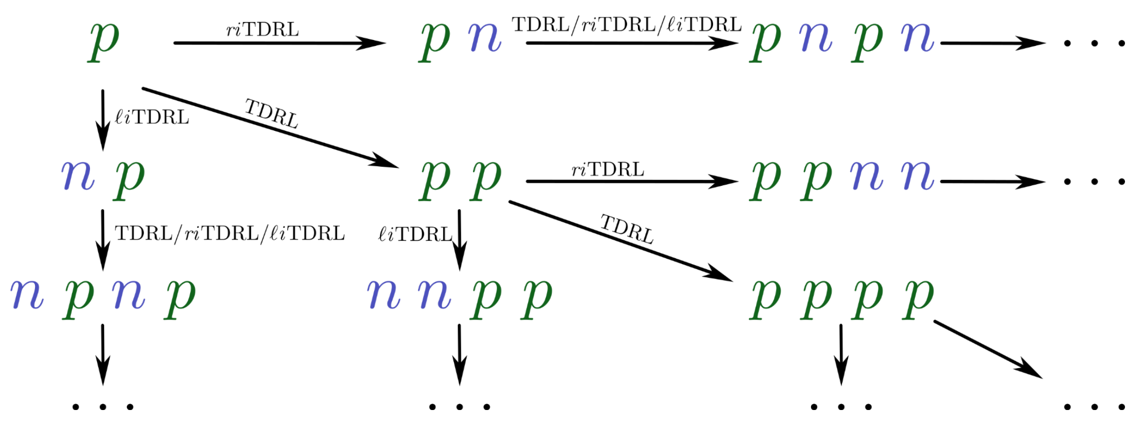


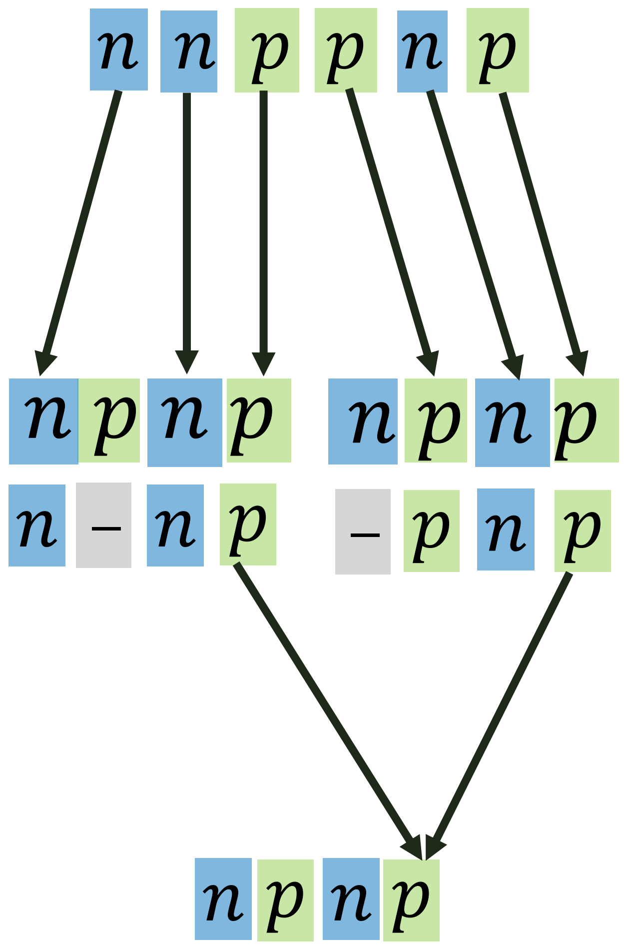


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

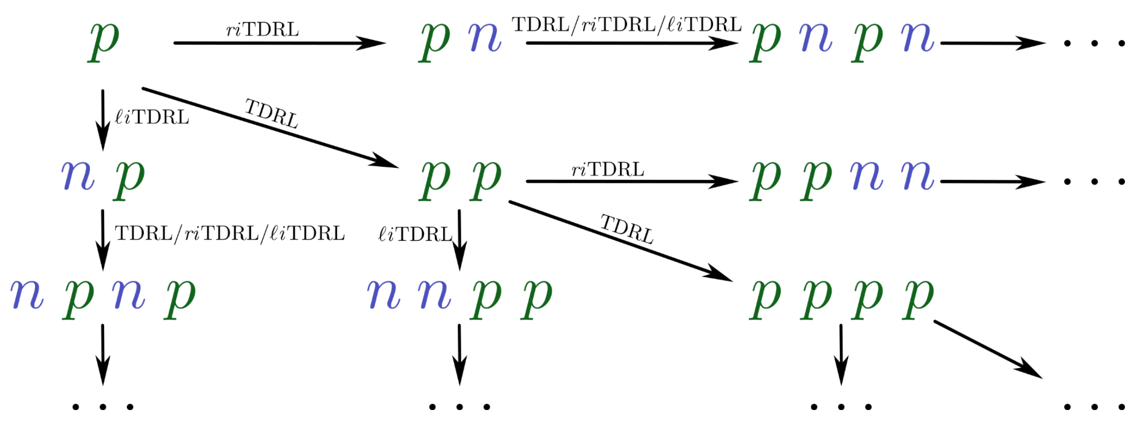


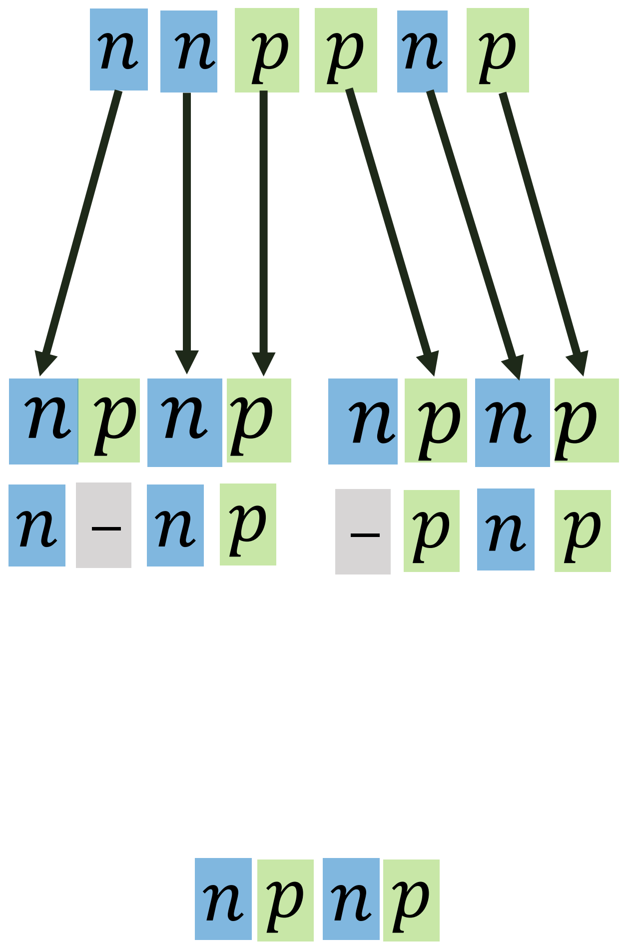


An Algorithm

***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as “sorting guide”.
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

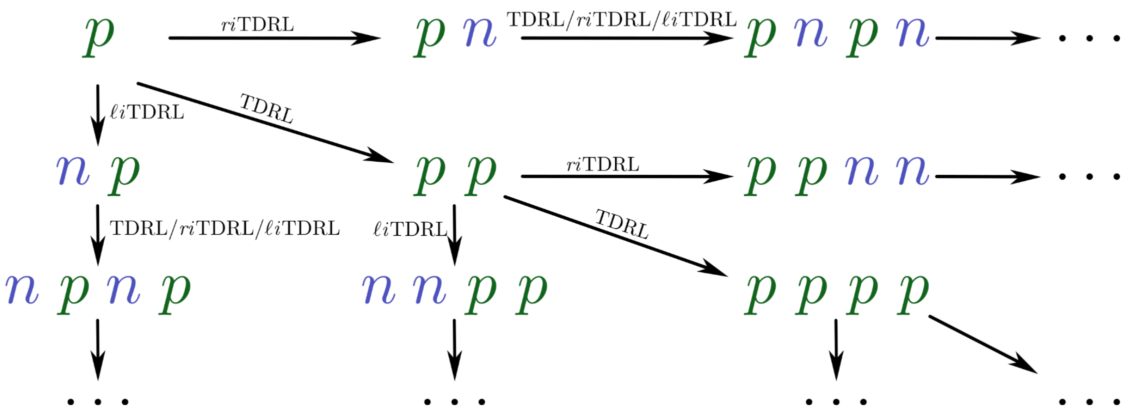


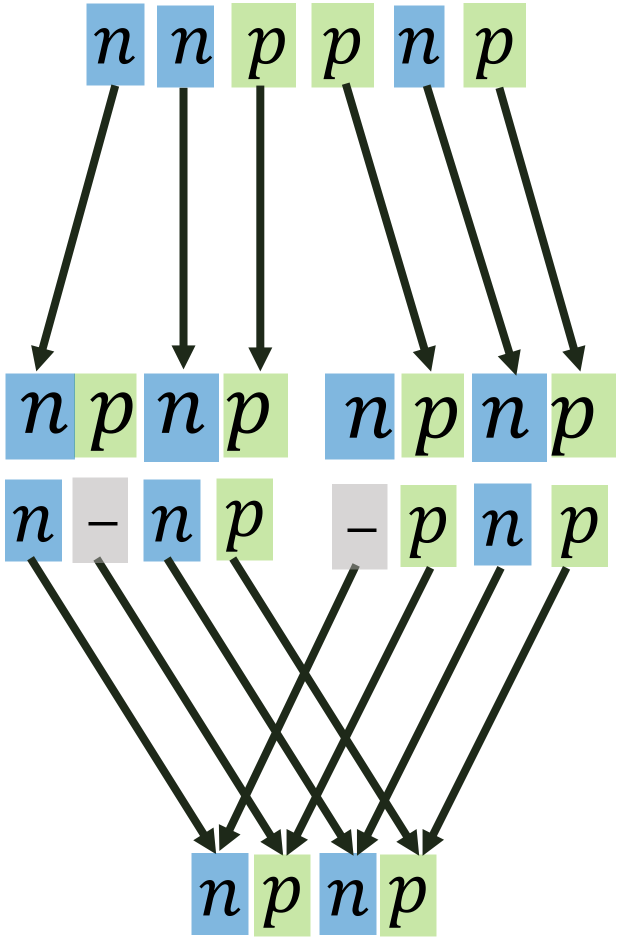


An Algorithm

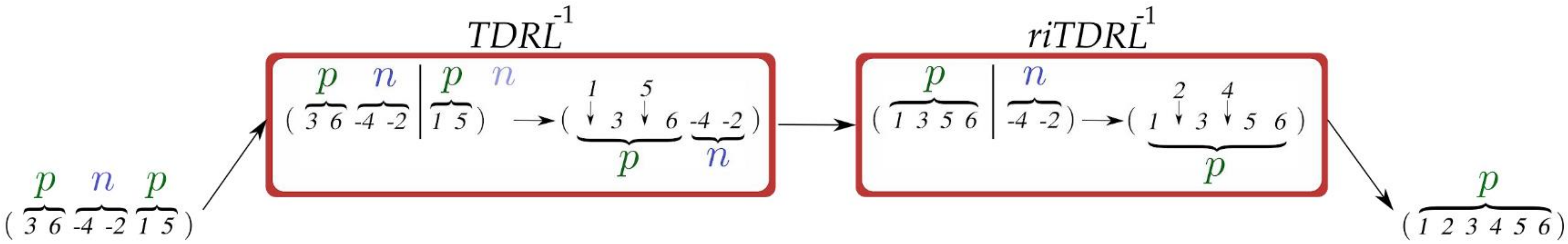
***Algorithm – Sketch*:**

1. Greedy matching to fitting pattern.
2. Use pattern as "sorting guide".
3. Sort matching intervals sequentially.
4. Derive a TDRL/iTDRL corresponding to sorting step.
5. Repeat **1-4,** until sorted.

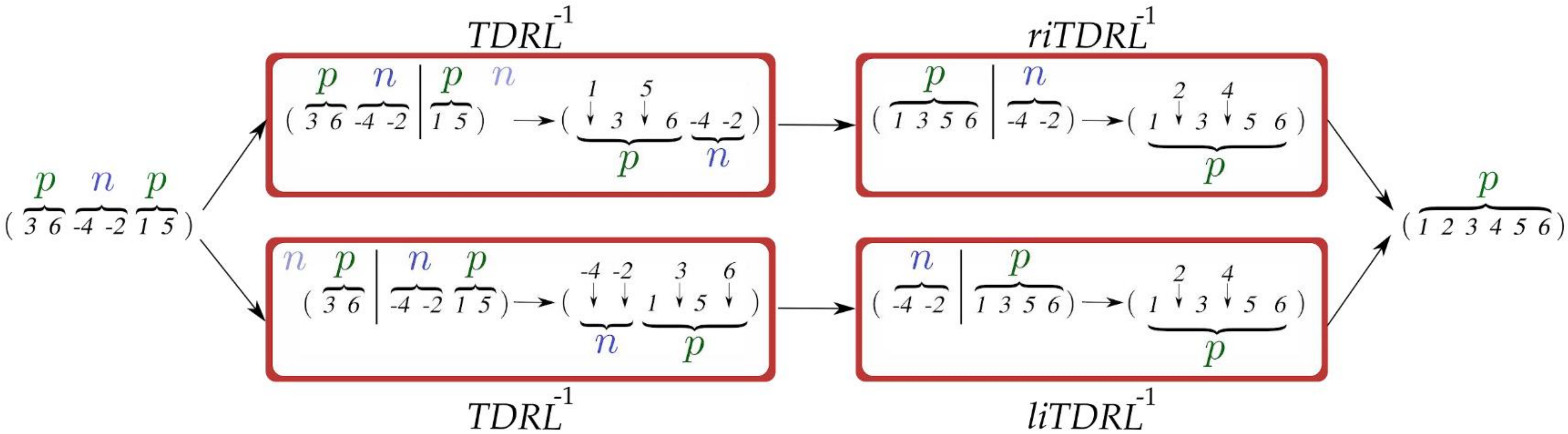




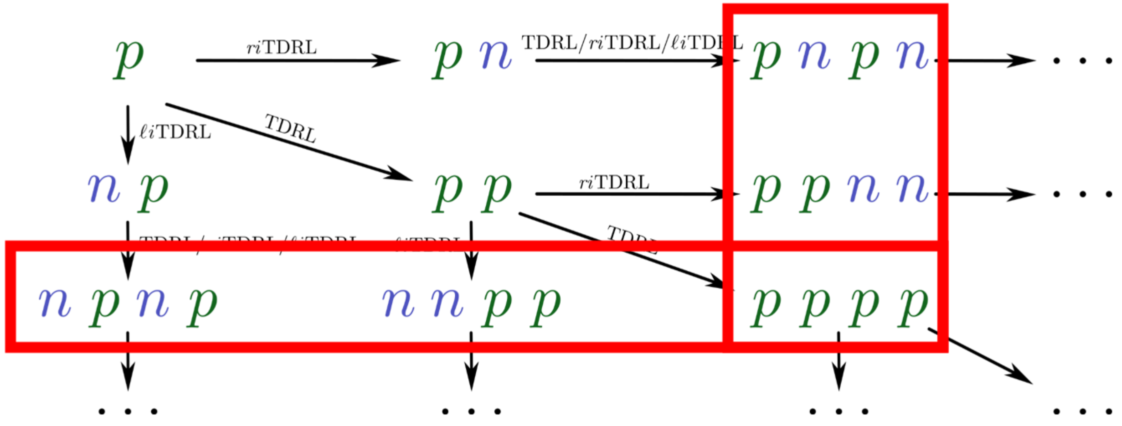
Sorting by TDRL and iTDRL



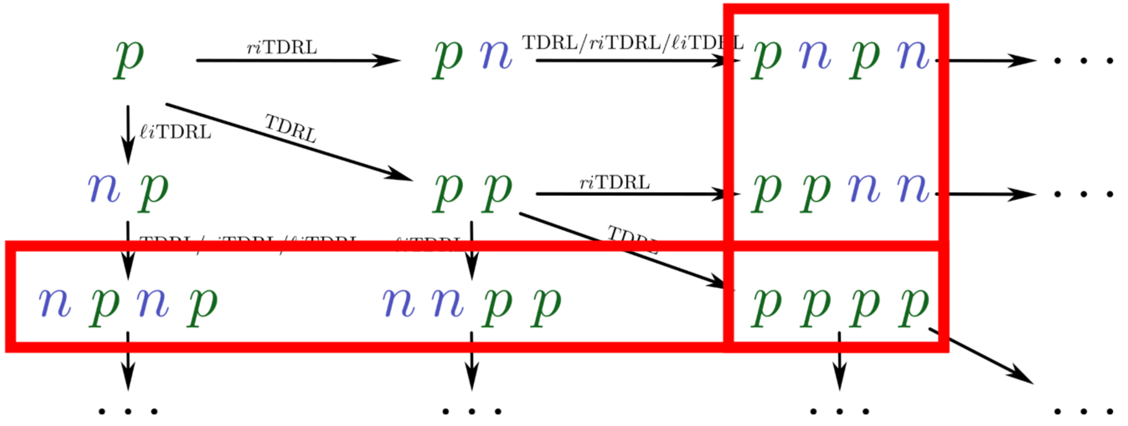
Sorting by TDRL and iTDRL



Complexity

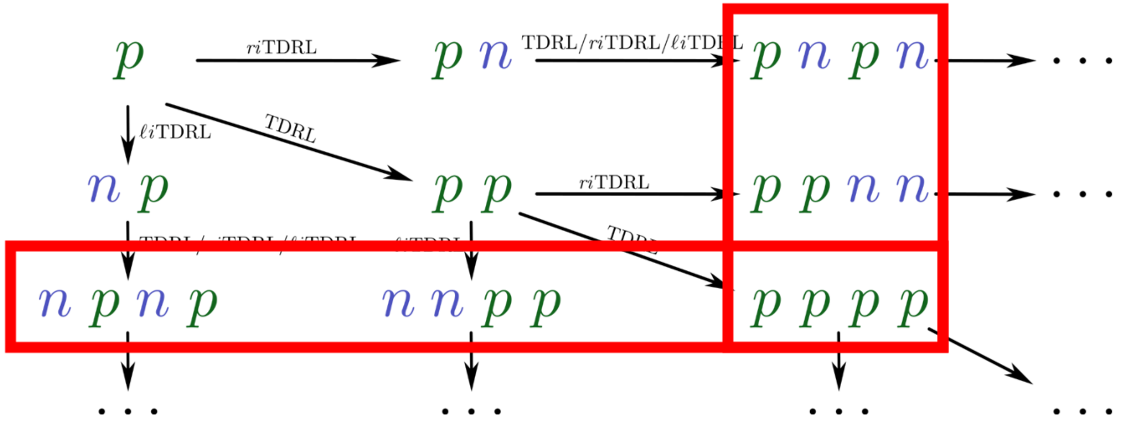

p
riTDRL
p n
TDRL/riTDRL/ℓiTDRL
p n p n
ℓiTDRL
TDRL
n p
p p
riTDRL
p p n n
n p n p
n n p p
p p p p
. . .

Complexity



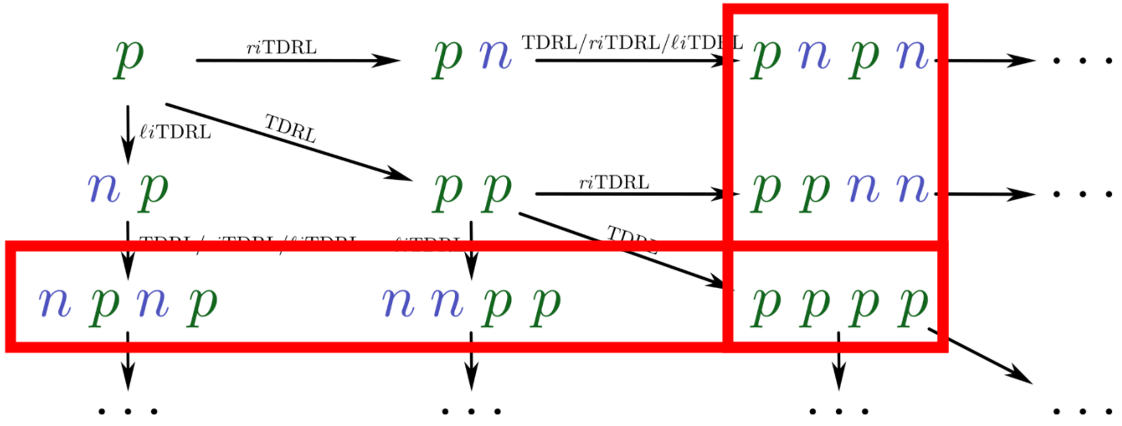
- Number of characters in MISC-encoding: $\mathcal{O}(n)$

Complexity



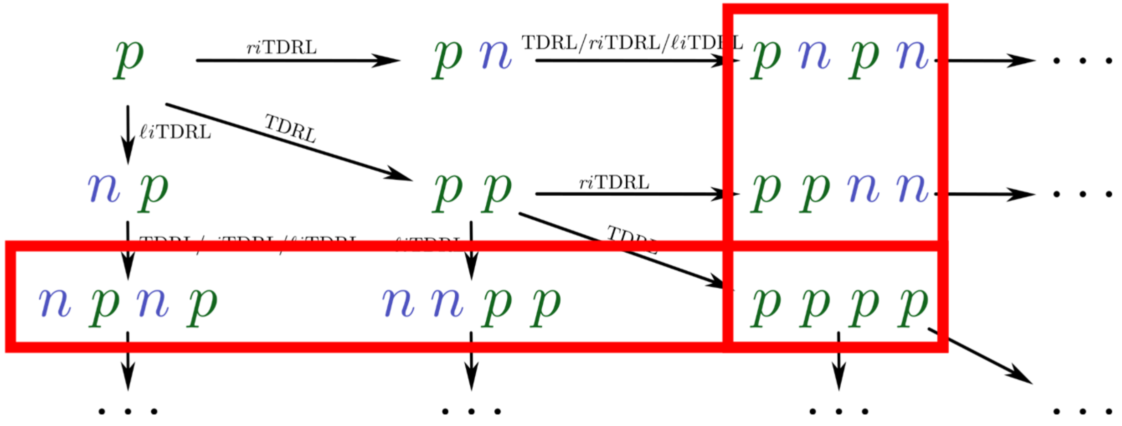
- Number of characters in MISC-encoding: $\mathcal{O}(n)$
- Number of characters in matching MISC-pattern: $\mathcal{O}(n)$

Complexity



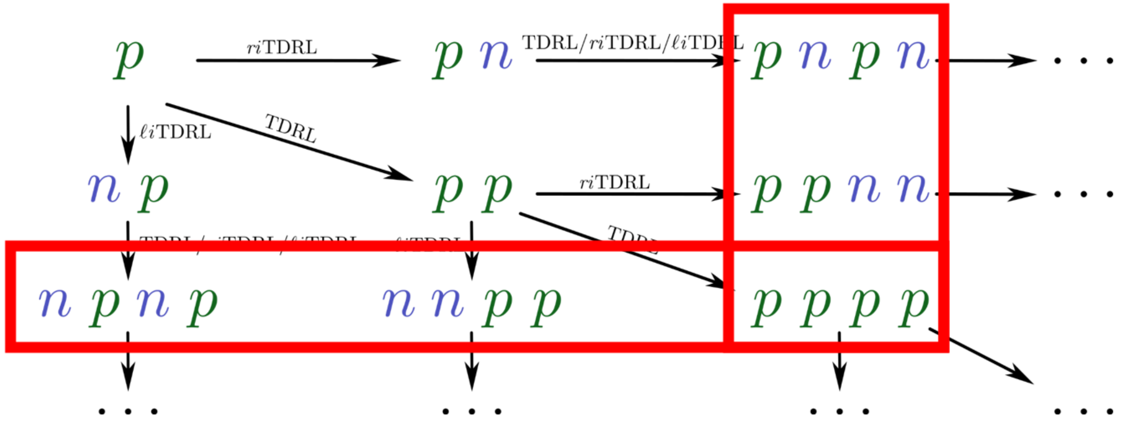
- Number of characters in MISC-encoding: $\mathcal{O}(n)$
- Number of characters in matching MISC-pattern: $\mathcal{O}(n)$
- Number of patterns to check: $\mathcal{O}(\log n)$

Complexity



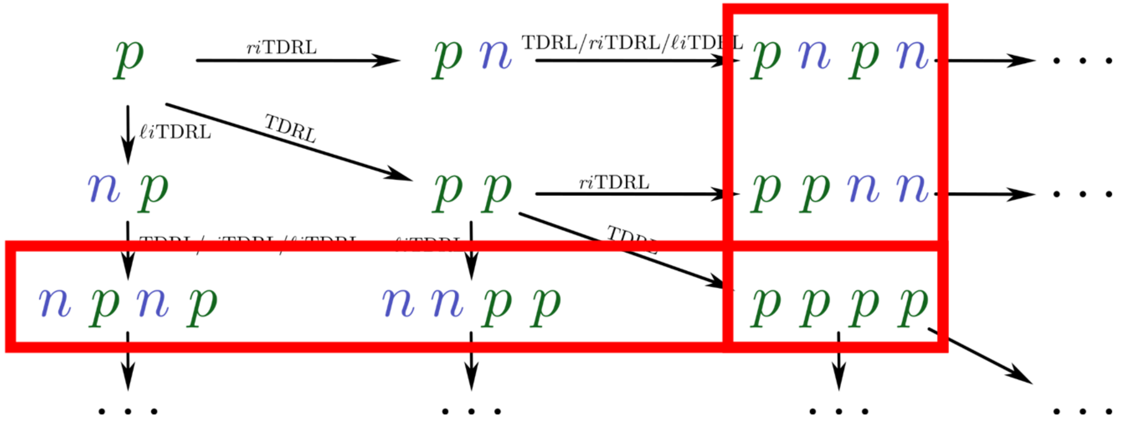
- Number of characters in MISC-encoding: $\mathcal{O}(n)$
- Number of characters in matching MISC-pattern: $\mathcal{O}(n)$
- Number of patterns to check: $\mathcal{O}(\log n)$
- Check which pattern MISC-encoding is subsequence of: $\mathcal{O}(n)$

Complexity



- Number of characters in MISC-encoding: $\mathcal{O}(n)$
- Number of characters in matching MISC-pattern: $\mathcal{O}(n)$
- Number of patterns to check: $\mathcal{O}(\log n)$
- Check which pattern MISC-encoding is subsequence of: $\mathcal{O}(n)$
- Sorting of permutation: $\mathcal{O}(\log n)$ sorting steps with $\mathcal{O}(n)$ cost

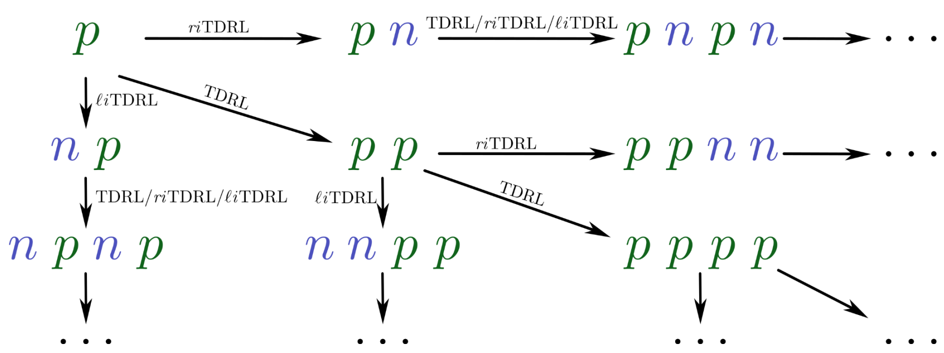
Complexity



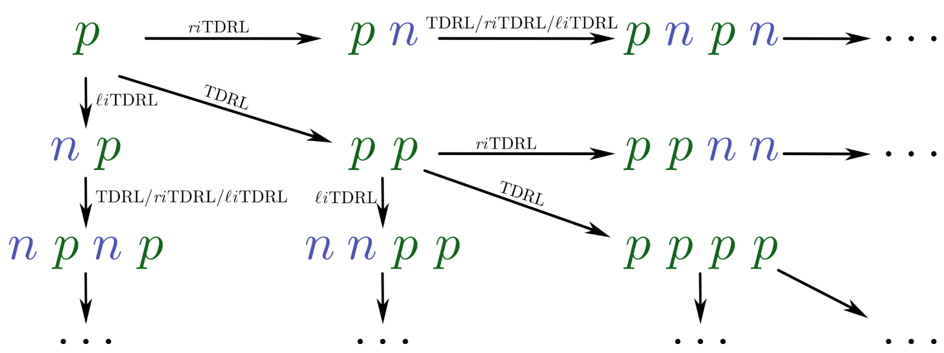
- Number of characters in MISC-encoding: $\mathcal{O}(n)$
- Number of characters in matching MISC-pattern: $\mathcal{O}(n)$
- Number of patterns to check: $\mathcal{O}(\log n)$
- Check which pattern MISC-encoding is subsequence of: $\mathcal{O}(n)$
- Sorting of permutation: $\mathcal{O}(\log n)$ sorting steps with $\mathcal{O}(n)$ cost

$$\mathcal{O}(n \log n).$$

Conclusion

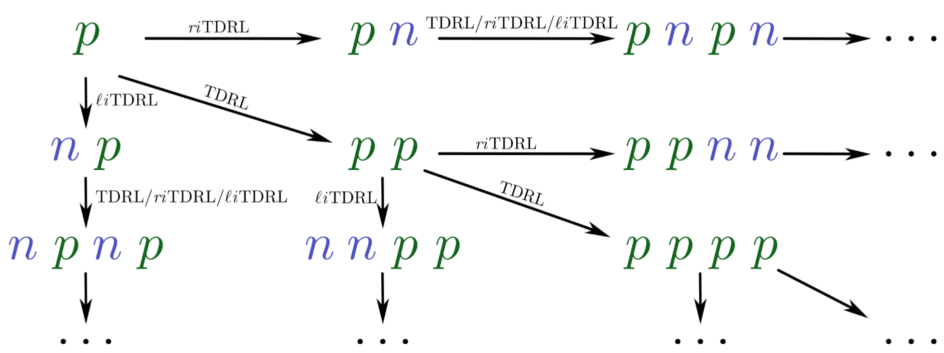

p
riTDRL
p n
TDRL/riTDRL/ℓiTDRL
p n p n
ℓiTDRL
TDRL
n p
p p
riTDRL
p p n n
TDRL/riTDRL/ℓiTDRL
ℓiTDRL
TDRL
n p n p
n n p p
p p p p

Conclusion



- TDRL/iTDRL distance between the identity, and any other permutation π is bounded by the number of characters in π's MISC-encoding (logarithmic).
  → The sorting scenario can be computed in $\mathcal{O}(n \log n)$.

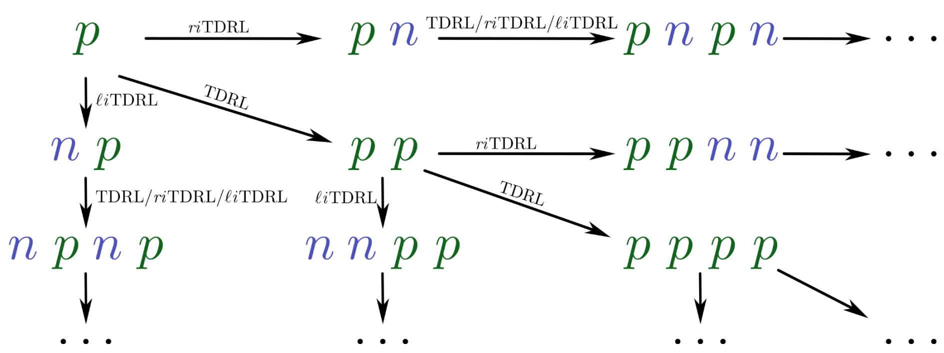
Conclusion



- TDRL/iTDRL distance between the identity, and any other permutation π is bounded by the number of characters in π's MISC-encoding (logarithmic).
  → The sorting scenario can be computed in $\mathcal{O}(n \log n)$.

- Powerful model as, for example for metazoan mitochondria, **at most 7 TDRLs/iTDRLs** are necessary to rearrange any of their gene orders into another.

Conclusion



- TDRL/iTDRL distance between the identity, and any other permutation π is bounded by the number of characters in π's MISC-encoding (logarithmic).
  → The sorting scenario can be computed in $\mathcal{O}(n \log n)$.

- Powerful model as, for example for metazoan mitochondria, **at most 7 TDRLs/iTDRLs** are necessary to rearrange any of their gene orders into another.

- Biological constraints need to be considered
  → Keep common gene clusters at each sorting step
  → restrict number of genes on which a TDRL/iTDRL can act on