EFFICIENT PATTERN MATCHING IN ELASTIC-DEGENERATE TEXTS

Costas S. Iliopoulos **Ritu Kundu** Solon P. Pissis

March 06, 2017

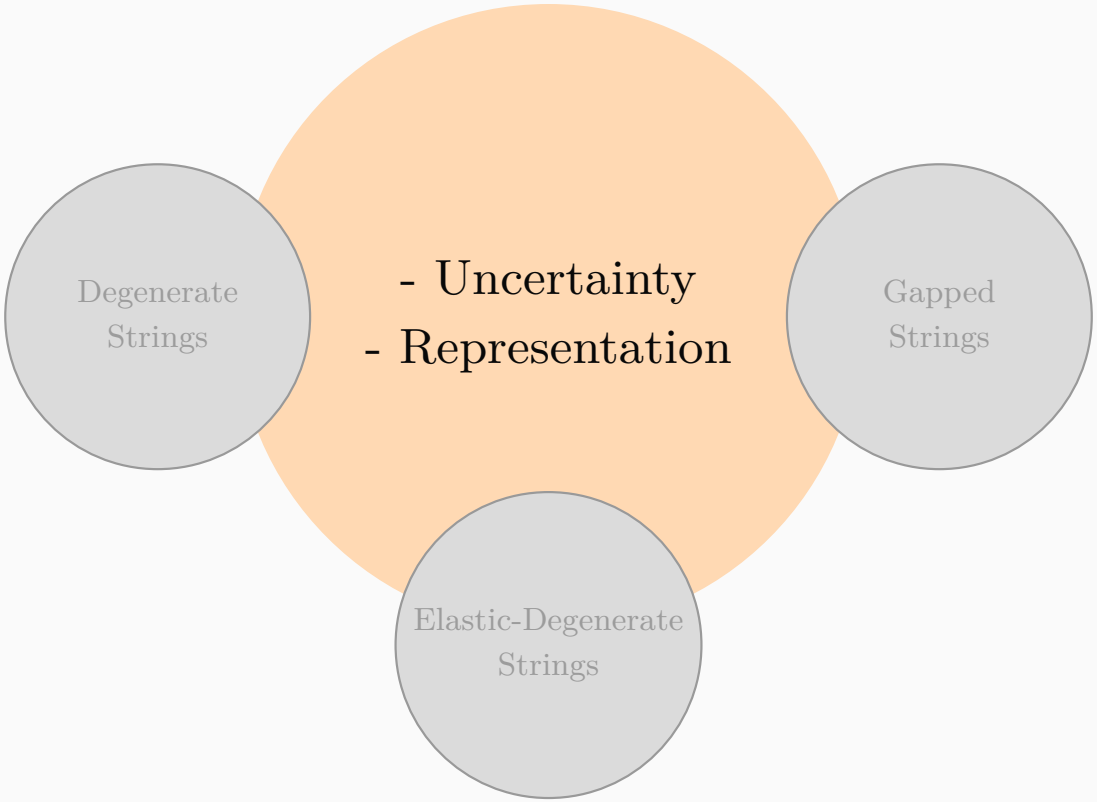
Presenting at:

11th International Conference on Language and Automata Theory and Applications (LATA 2017)

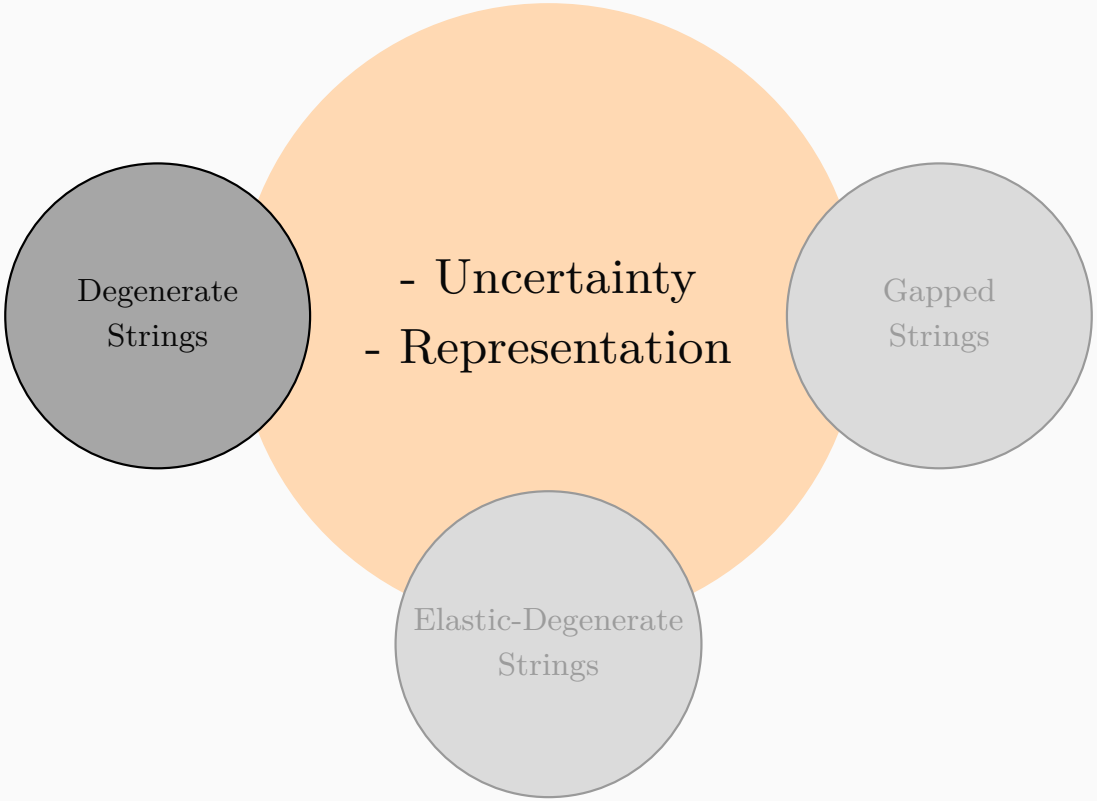
Outlines

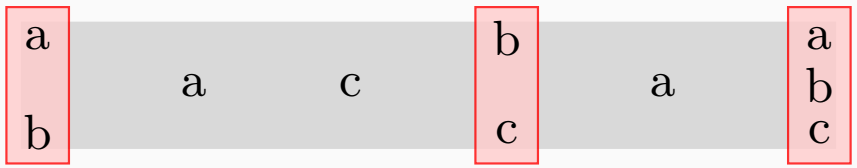
INTRODUCTION

Introduction

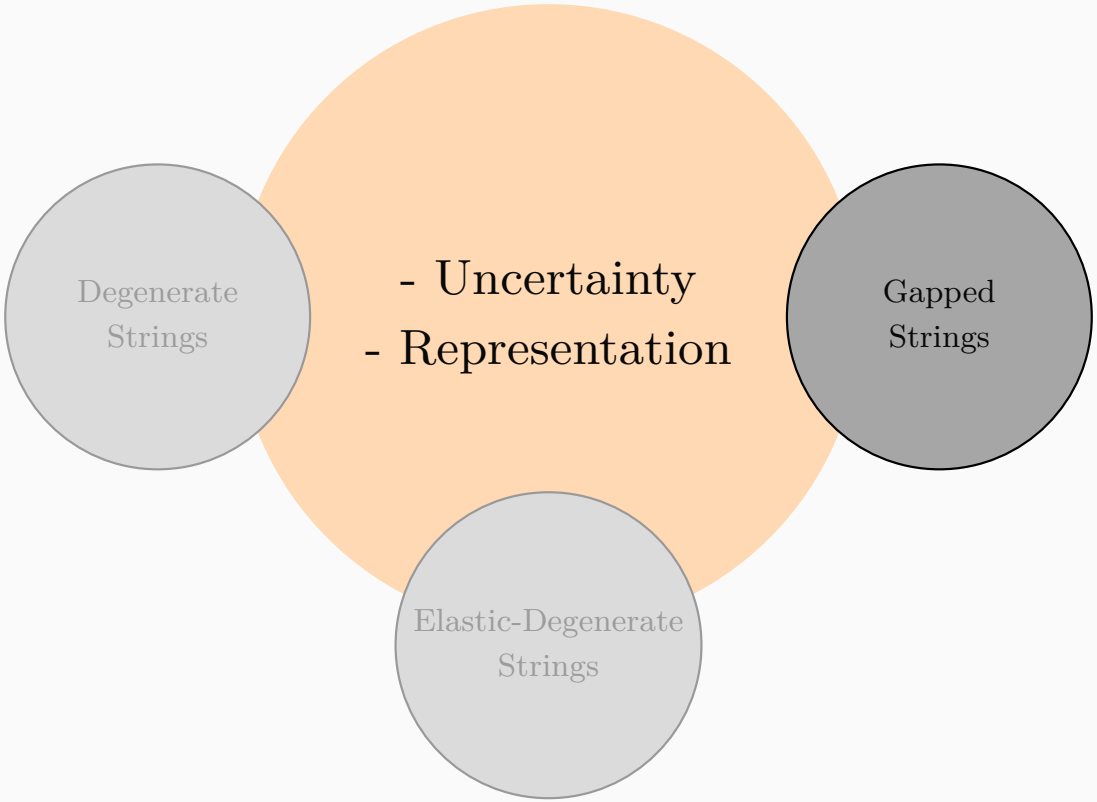


Introduction



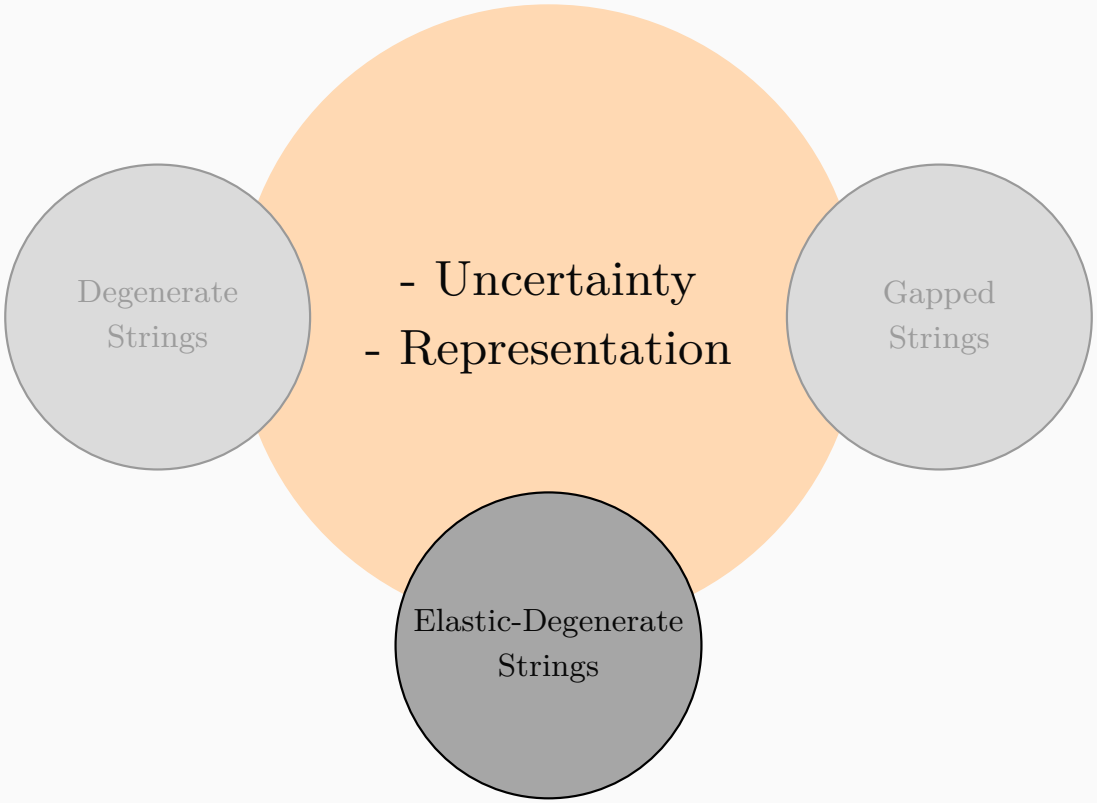


Introduction

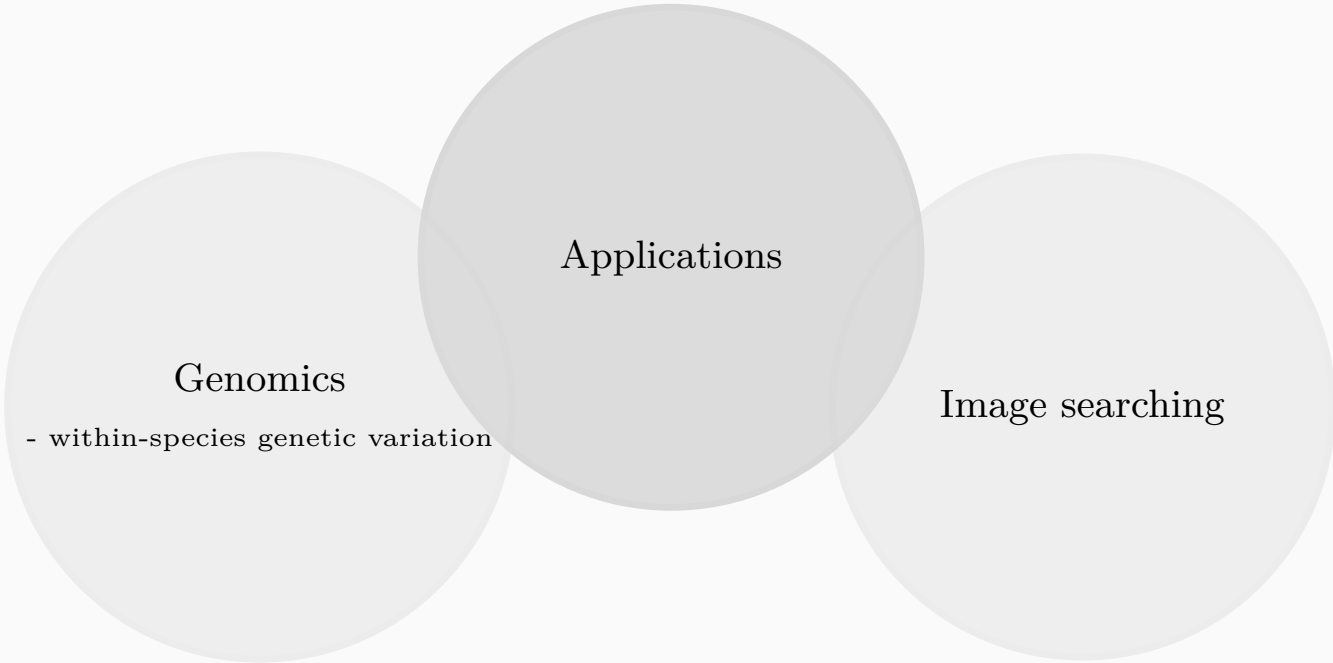


$a\ b\ b\ c$ | $2 - -4$ | $c\ c\ a$ | $3 - -6$ | $b\ b$

Introduction



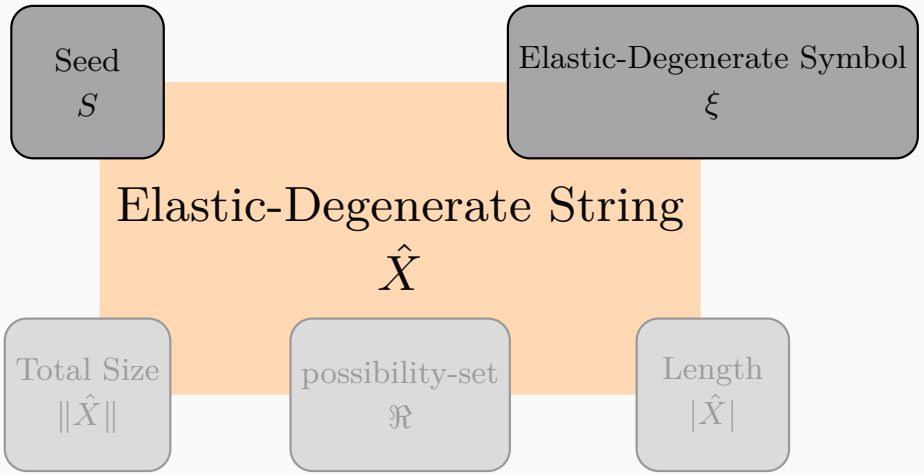
Motivation


Applications
Genomics
- within-species genetic variation
Image searching

TECHNICAL BACKGROUND

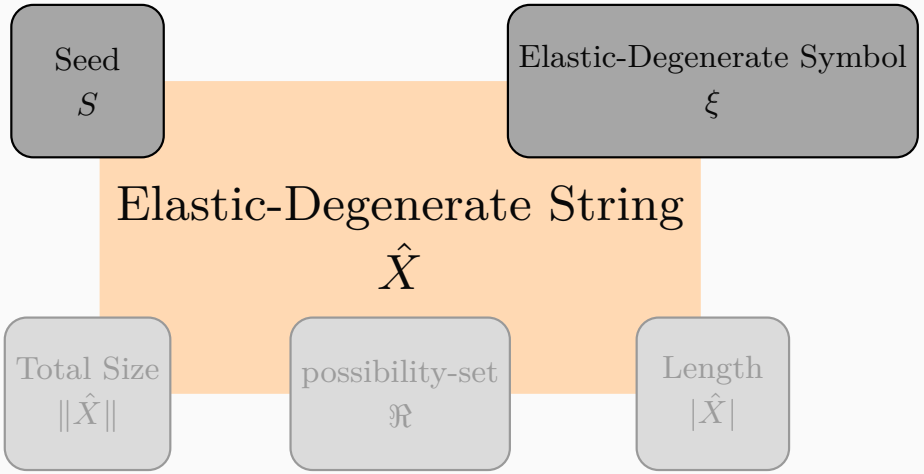
Terminology I

$$\hat{X} = S_1\xi_1 S_2\xi_2 S_3 \ldots S_{k-1}\xi_{k-1}S_k$$

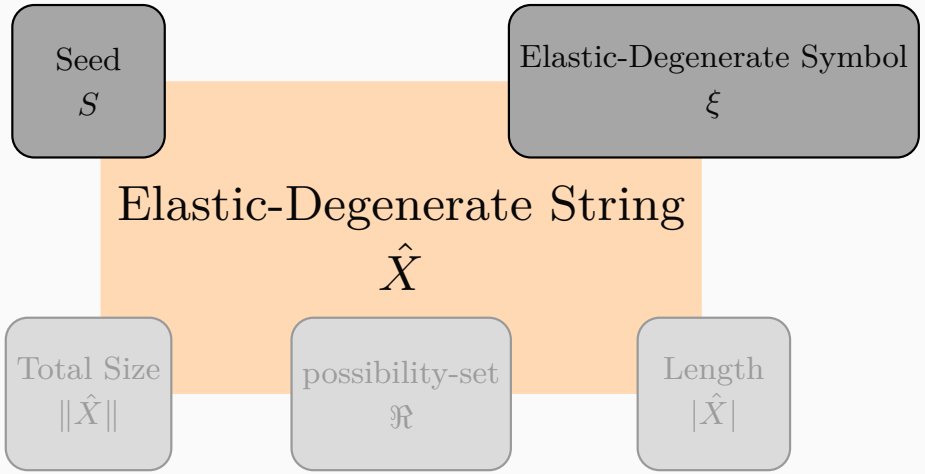


Terminology I

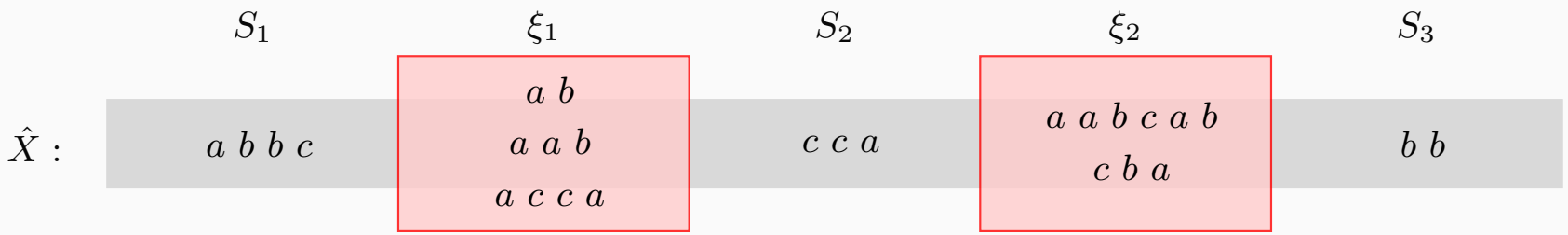
$$\hat{X} = S_1 \begin{bmatrix} E_{1,1} \\ E_{1,2} \\ \vdots \\ E_{1,|\xi_1|} \end{bmatrix} S_2 \begin{bmatrix} E_{2,1} \\ E_{2,2} \\ \vdots \\ E_{2,|\xi_2|} \end{bmatrix} S_3 \ldots S_{k-1} \begin{bmatrix} E_{k-1,1} \\ E_{k-1,2} \\ \vdots \\ E_{k-1,|\xi_{k-1}|} \end{bmatrix} S_k$$



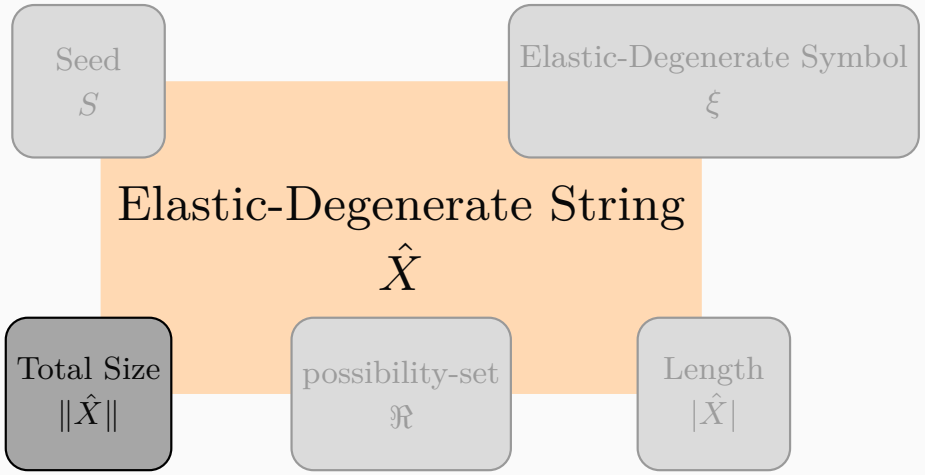
Terminology I



Example

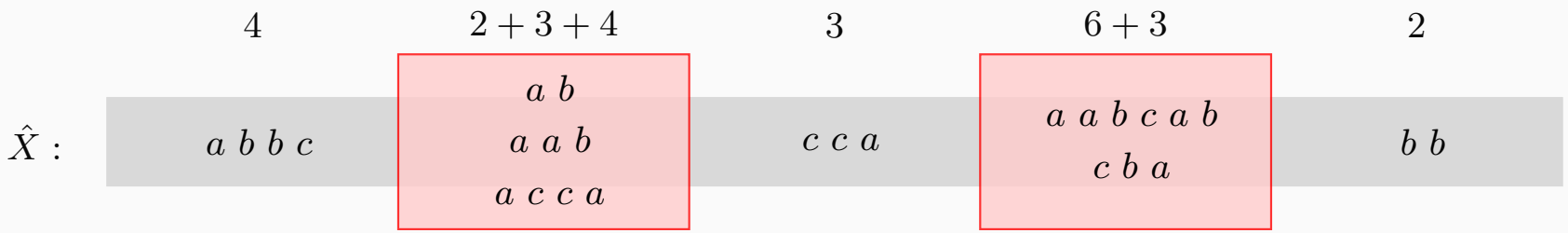


Terminology I



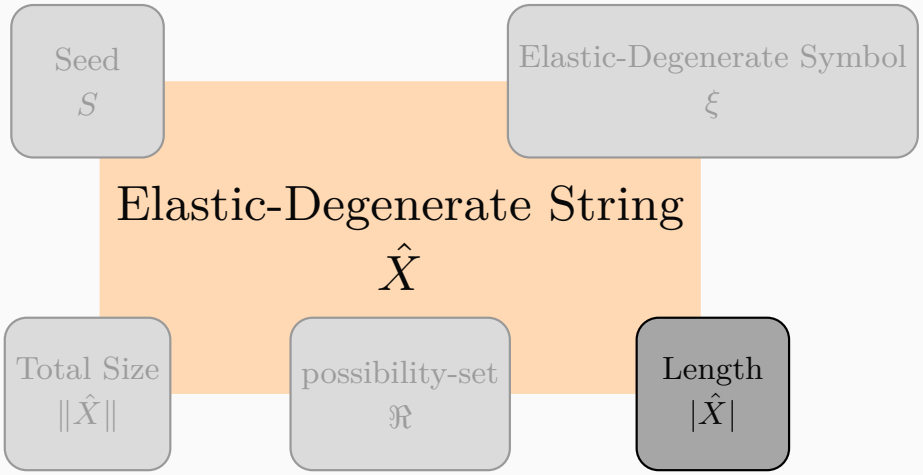
$$\|\hat{X}\| = \sum_{i=1}^{k} |S_i| + \sum_{i=1}^{k-1} \sum_{j=1}^{|\xi_i|} |E_{i,j}|$$

Example



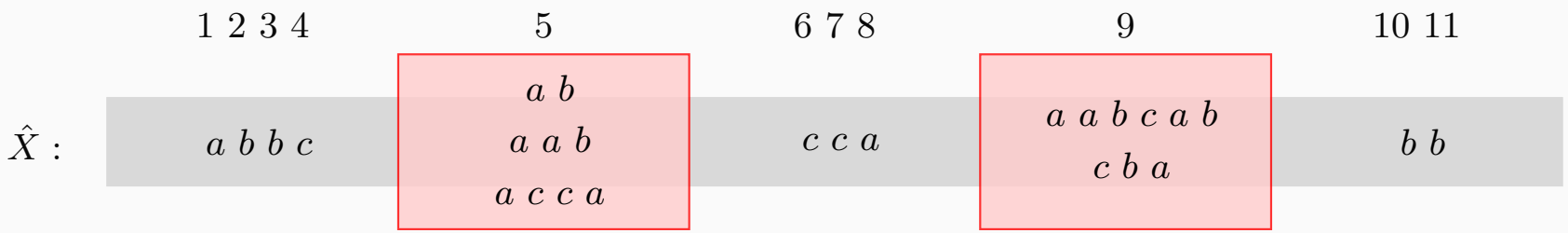
Total size ($\|\hat{X}\|$) = 27

Terminology I



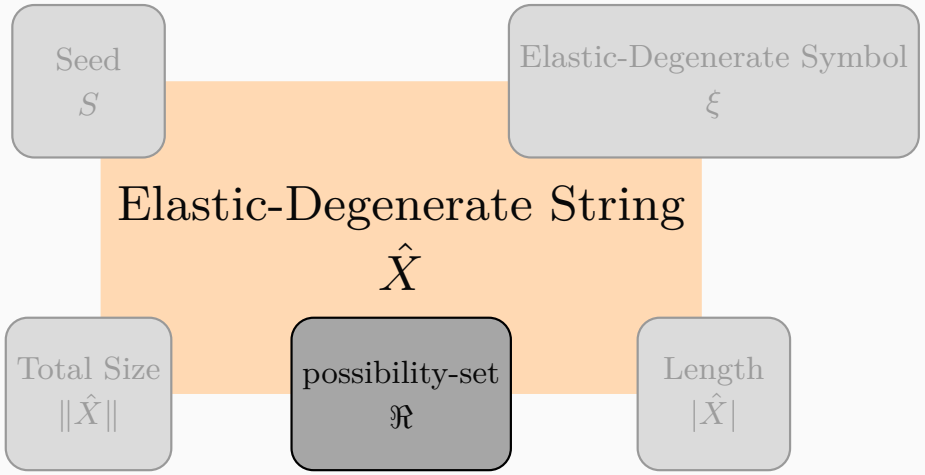
$$|\hat{X}| = \sum_{i=1}^{k} |S_i| + k - 1$$

Example



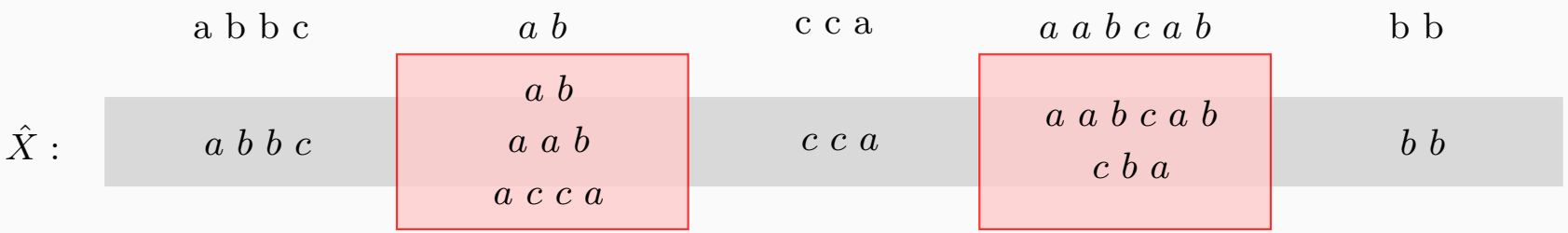
Length $(|\hat{X}|) = 11$

Terminology I



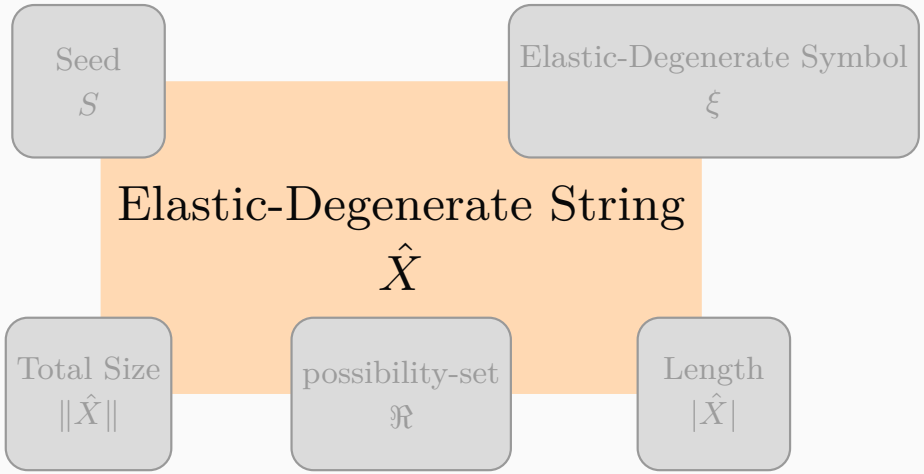
$\{S_1 E_{1,r_1} S_2 E_{2,r_2} \dots E_{k-1,r_{k-1}} S_k\} \quad \forall r_i, 1 \leq i \leq k-1 \text{ such that } 1 \leq r_i \leq |\xi_i|$

Example

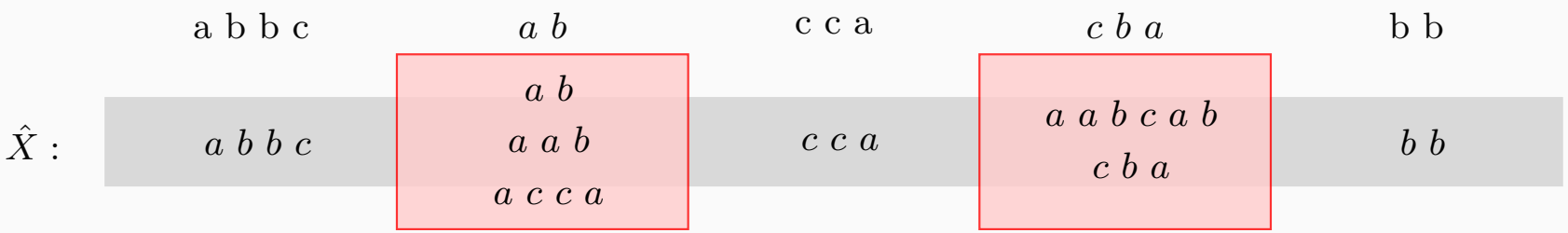


$\Re = \{abbcabccaaabcabbb, abbcabccacbabb, abbcaabccaaabcabbb, abbaabccacbabb, abbcaccaccaaabcabbb, abbcaccaccacbabb\}$

Terminology I

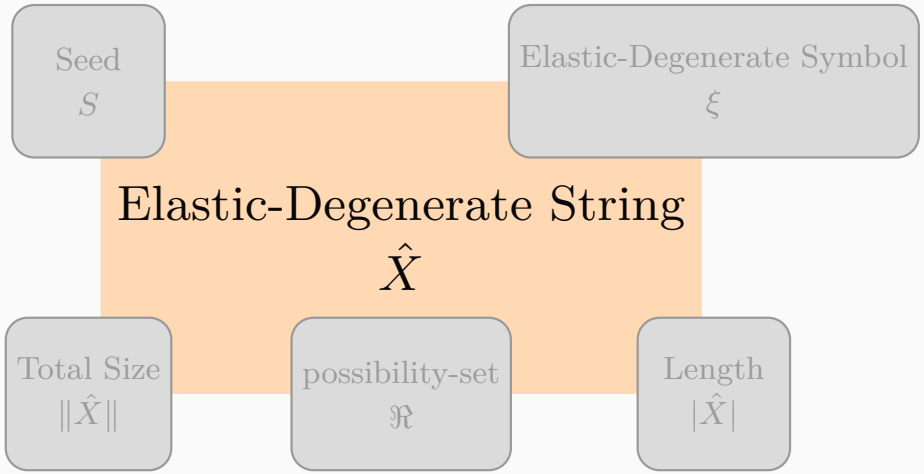


Example

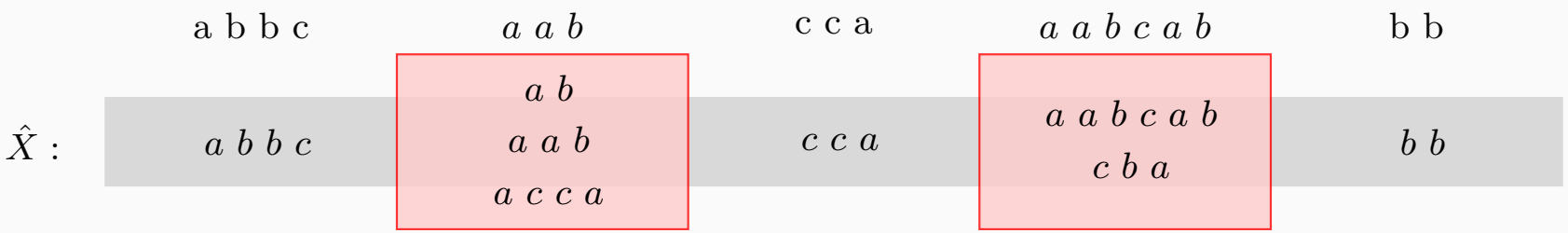


$\Re = \{abbcabccaaabcabbb, abbcabccacbabb, abbcaabccaaabcabbb, abbaabccacbabb, abbcaccaccaaabcabbb, abbcaccaccacbabb\}$

Terminology I

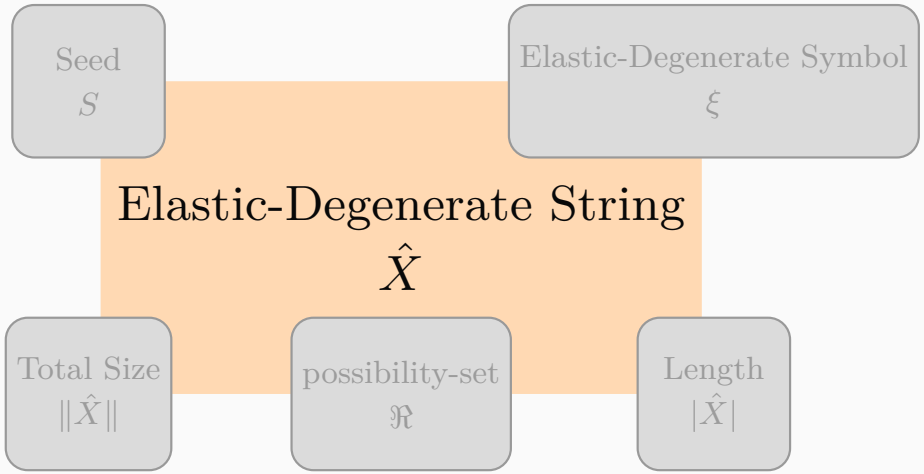


Example

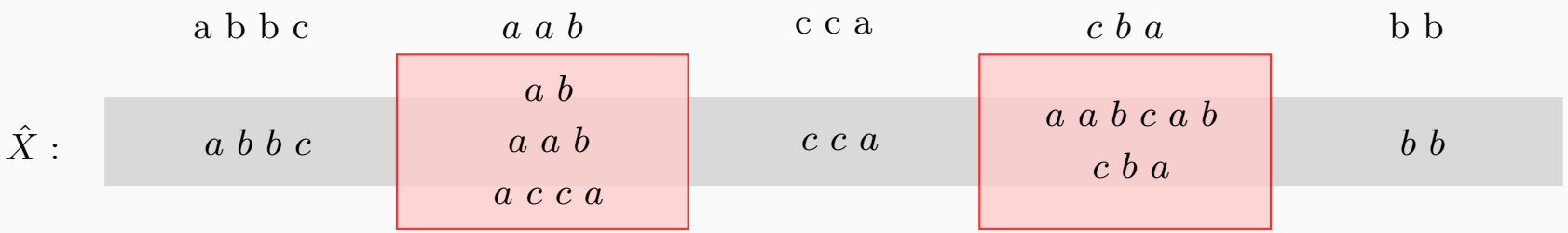


$\Re = \{abbcabccaaabcabbb, abbcabccacbabb, abbcaabccaaabcabbb, abbaabccacbabb, abbcaccaccaaabcabbb, abbcaccaccacbabb\}$

Terminology I

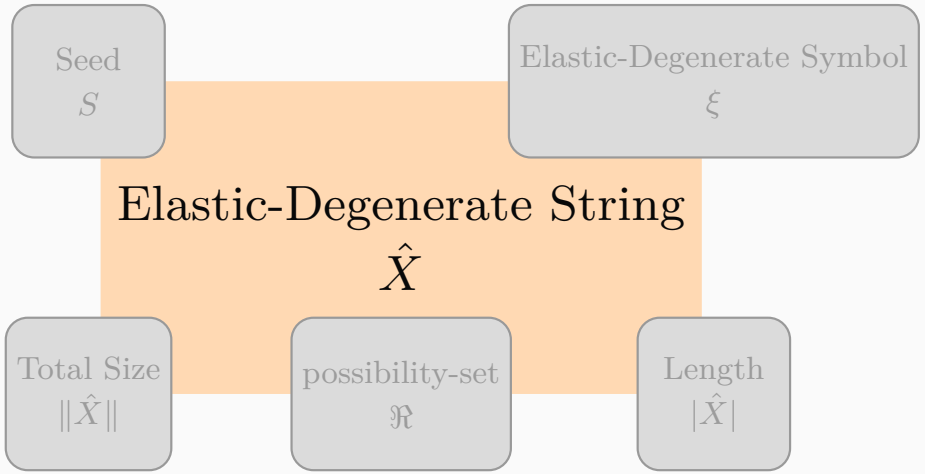


Example

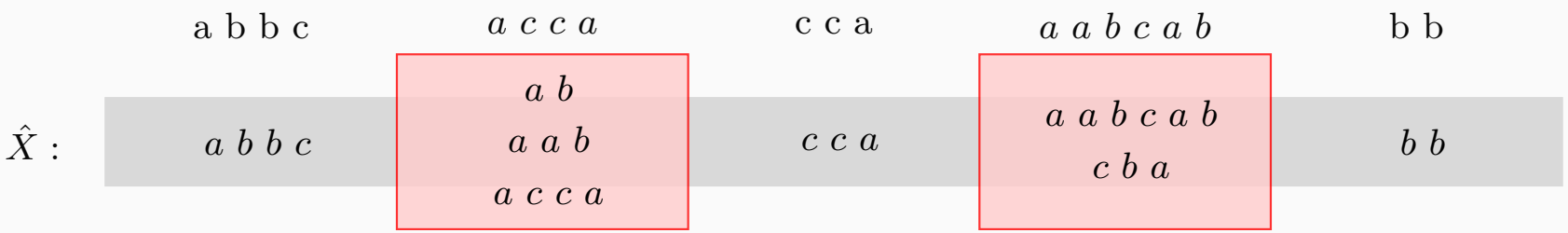


$\Re = \{abbcabccaaabcabbb, abbcabccacbabb, abbcaabccaaabcabbb, abbaabccacbabb, abbcaccaccaaabcabbb, abbcaccaccacbabb\}$

Terminology I

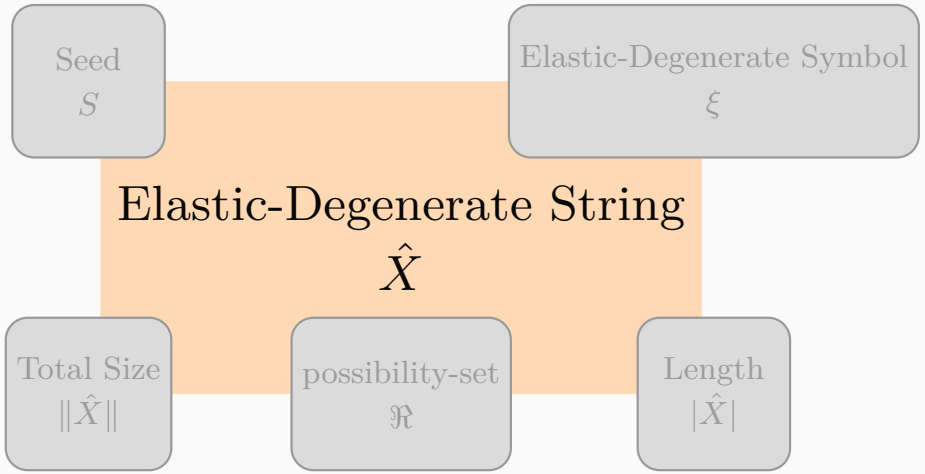


Example

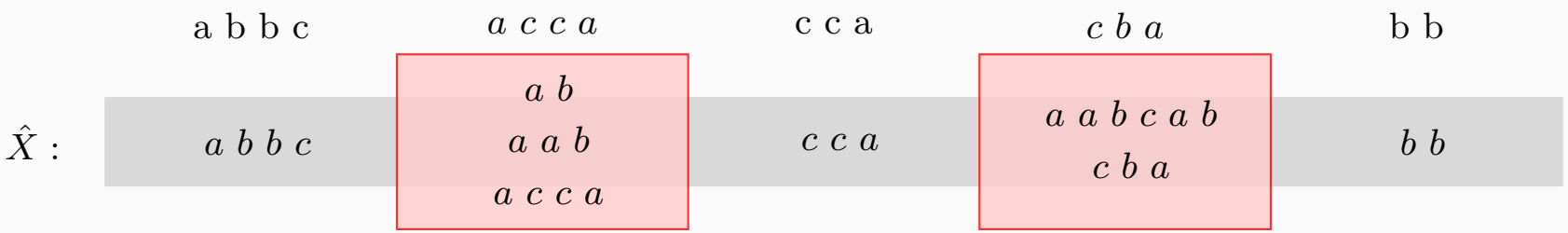


$\Re = \{abbcabccaaabcabbb, abbcabccacbabb, abbcaabccaaabcabbb, abbaabccacbabb, abbcaccaccaaabcabbb, abbcaccaccacbabb\}$

Terminology I



Example



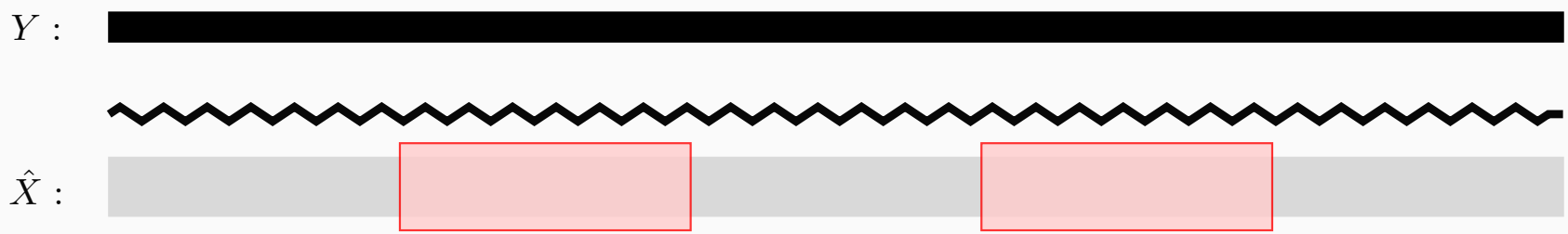
$\Re = \{abbcabccaaabcabbb, abbcabccacbabb, abbcaabccaaabcabbb, abbaabccacbabb, abbcaccaccaaabcabbb, abbcaccaccacbabb\}$

Matching ($\hat{X} \simeq Y$)

An elastic-degenerate string $\hat{X}$ with k seeds and a solid string Y are said to match, denoted by $\hat{X} \simeq Y$, if, and only if, there exists a solid string $S = S_1 E_{1,r_1} S_2 E_{2,r_2} \cdots E_{k-1,r_{k-1}} S_k, 1 \leq r_i \leq |\xi_i|$, obtained from $\hat{X}$ (i.e. $S \in \Re$ of $\hat{X}$), such that $S = UYV$, where $U, V \in \Sigma^*$, satisfying:

$$\begin{cases} U = \varepsilon,\ V = \varepsilon & \text{if } S_1 \neq \varepsilon,\ S_k \neq \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ V = \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1} & \text{if } S_1 = \varepsilon,\ S_k \neq \varepsilon \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ U = \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 \neq \varepsilon,\ S_k = \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1}, \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 = \varepsilon,\ S_k = \varepsilon. \end{cases}$$

Illustration

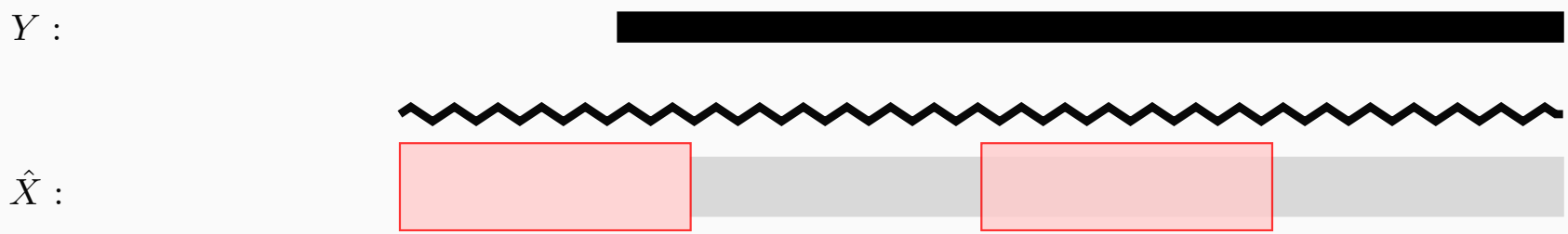


Matching ($\hat{X} \simeq Y$)

An elastic-degenerate string $\hat{X}$ with k seeds and a solid string Y are said to match, denoted by $\hat{X} \simeq Y$, if, and only if, there exists a solid string $S = S_1 E_{1,r_1} S_2 E_{2,r_2} \cdots E_{k-1,r_{k-1}} S_k, 1 \leq r_i \leq |\xi_i|$, obtained from $\hat{X}$ (i.e. $S \in \Re$ of $\hat{X}$), such that $S = UYV$, where $U, V \in \Sigma^*$, satisfying:

$$\begin{cases} U = \varepsilon,\ V = \varepsilon & \text{if } S_1 \neq \varepsilon,\ S_k \neq \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ V = \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1} & \text{if } S_1 = \varepsilon,\ S_k \neq \varepsilon \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ U = \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 \neq \varepsilon,\ S_k = \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1}, \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 = \varepsilon,\ S_k = \varepsilon. \end{cases}$$

Illustration

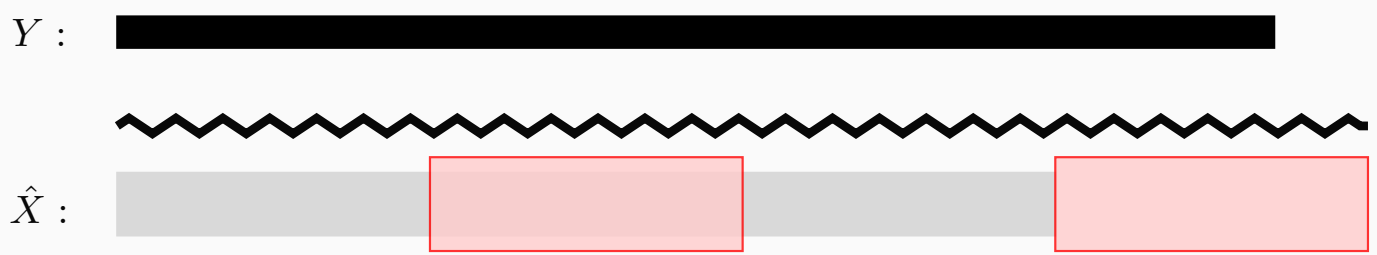


Matching ($\hat{X} \simeq Y$)

An elastic-degenerate string $\hat{X}$ with k seeds and a solid string Y are said to match, denoted by $\hat{X} \simeq Y$, if, and only if, there exists a solid string $S = S_1 E_{1,r_1} S_2 E_{2,r_2} \cdots E_{k-1,r_{k-1}} S_k, 1 \leq r_i \leq |\xi_i|$, obtained from $\hat{X}$ (i.e. $S \in \Re$ of $\hat{X}$), such that $S = UYV$, where $U, V \in \Sigma^*$, satisfying:

$$\begin{cases} U = \varepsilon,\ V = \varepsilon & \text{if } S_1 \neq \varepsilon,\ S_k \neq \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ V = \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1} & \text{if } S_1 = \varepsilon,\ S_k \neq \varepsilon \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ U = \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 \neq \varepsilon,\ S_k = \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1}, \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 = \varepsilon,\ S_k = \varepsilon. \end{cases}$$

Illustration

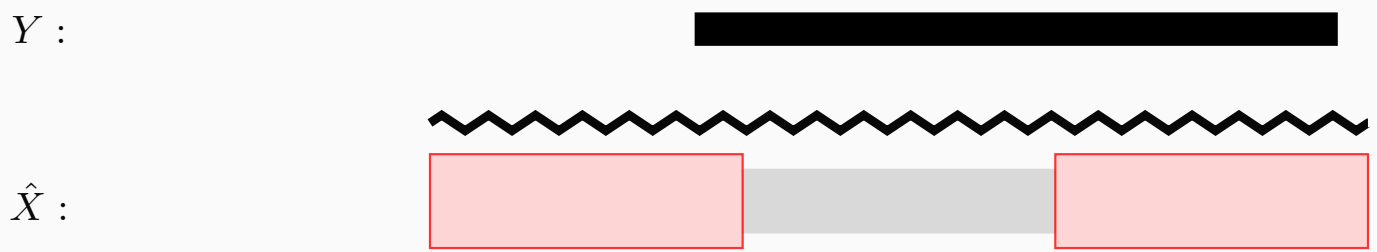


Matching ($\hat{X} \simeq Y$)

An elastic-degenerate string $\hat{X}$ with k seeds and a solid string Y are said to match, denoted by $\hat{X} \simeq Y$, if, and only if, there exists a solid string $S = S_1 E_{1,r_1} S_2 E_{2,r_2} \cdots E_{k-1,r_{k-1}} S_k, 1 \leq r_i \leq |\xi_i|$, obtained from $\hat{X}$ (i.e. $S \in \Re$ of $\hat{X}$), such that $S = UYV$, where $U, V \in \Sigma^*$, satisfying:

$$\begin{cases} U = \varepsilon,\ V = \varepsilon & \text{if } S_1 \neq \varepsilon,\ S_k \neq \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ V = \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1} & \text{if } S_1 = \varepsilon,\ S_k \neq \varepsilon \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ U = \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 \neq \varepsilon,\ S_k = \varepsilon \\ E_{1,r_1} \neq \varepsilon,\ U \text{ is either empty or a prefix of } E_{1,r_1}, \\ E_{k-1,r_{k-1}} \neq \varepsilon,\ V \text{ is either empty or a suffix of } E_{k-1,r_{k-1}} & \text{if } S_1 = \varepsilon,\ S_k = \varepsilon. \end{cases}$$

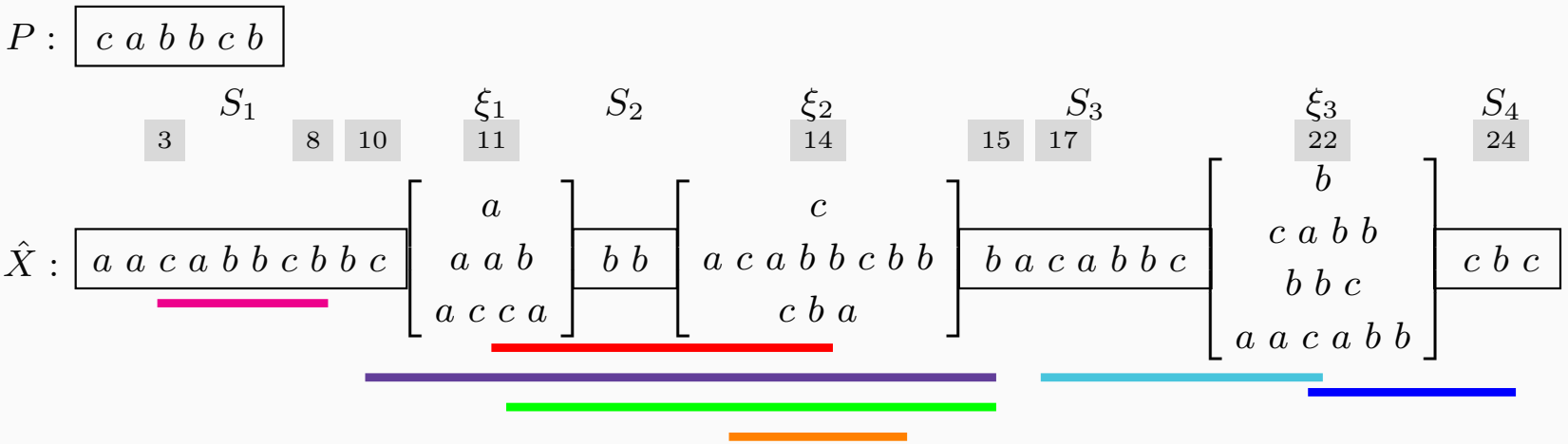
Illustration



Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

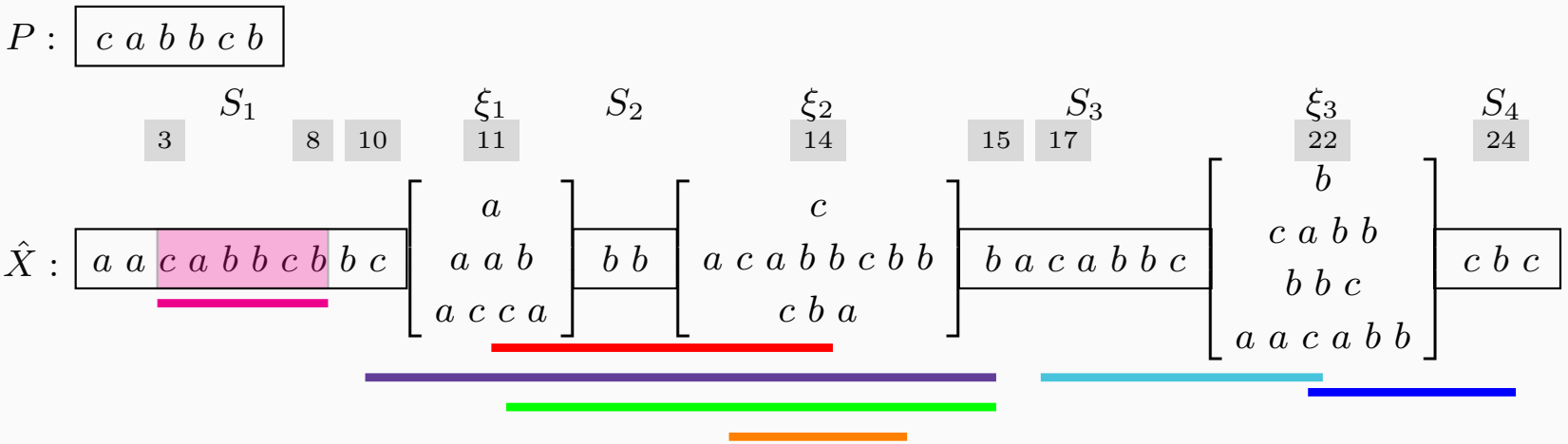


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

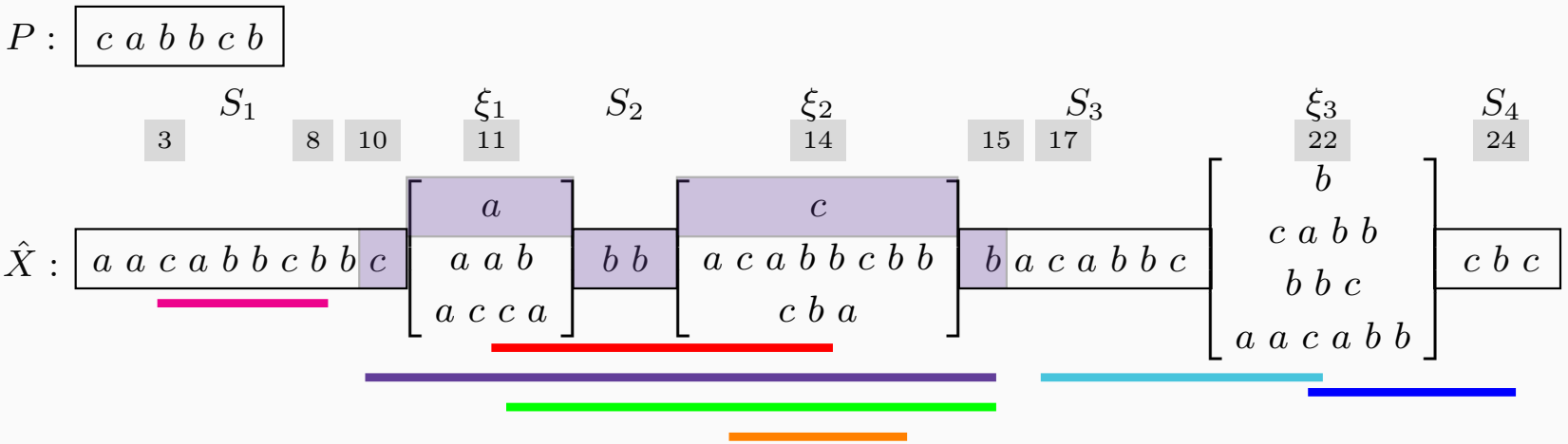


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

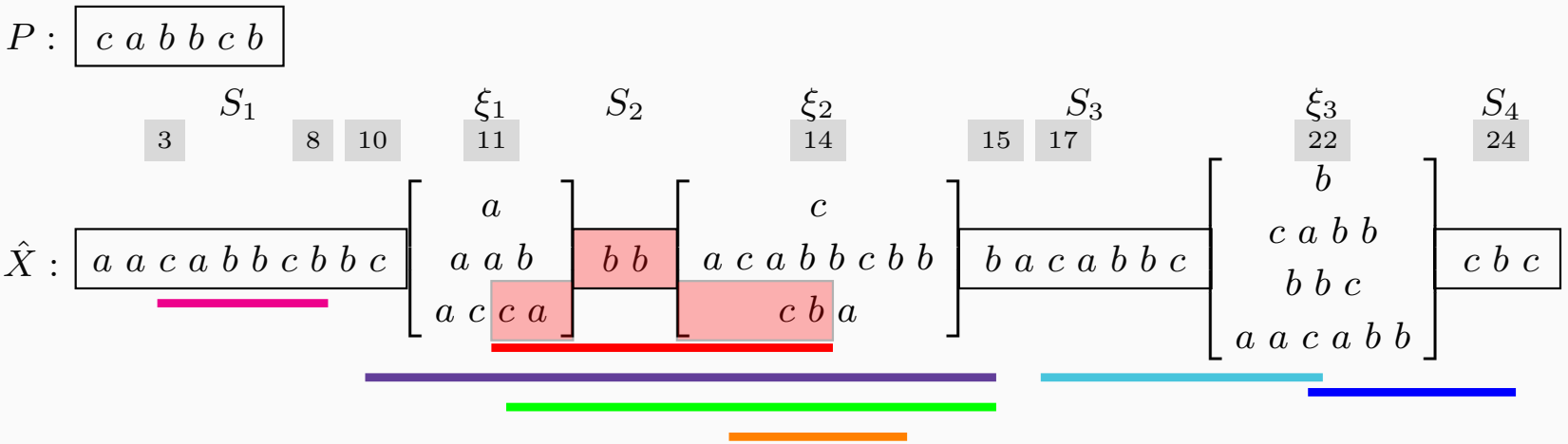


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

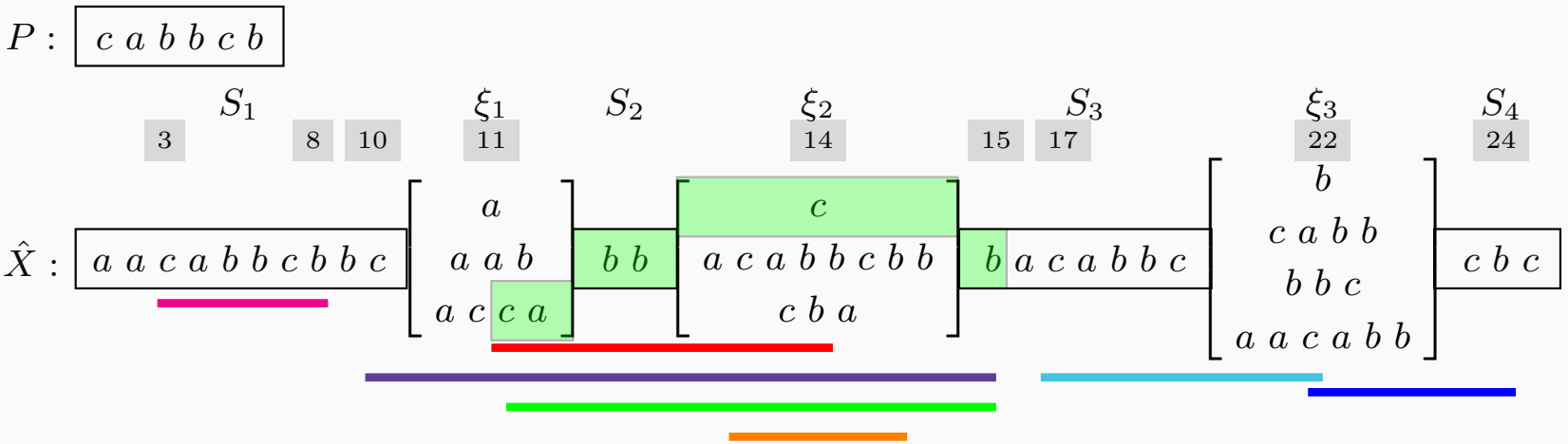


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

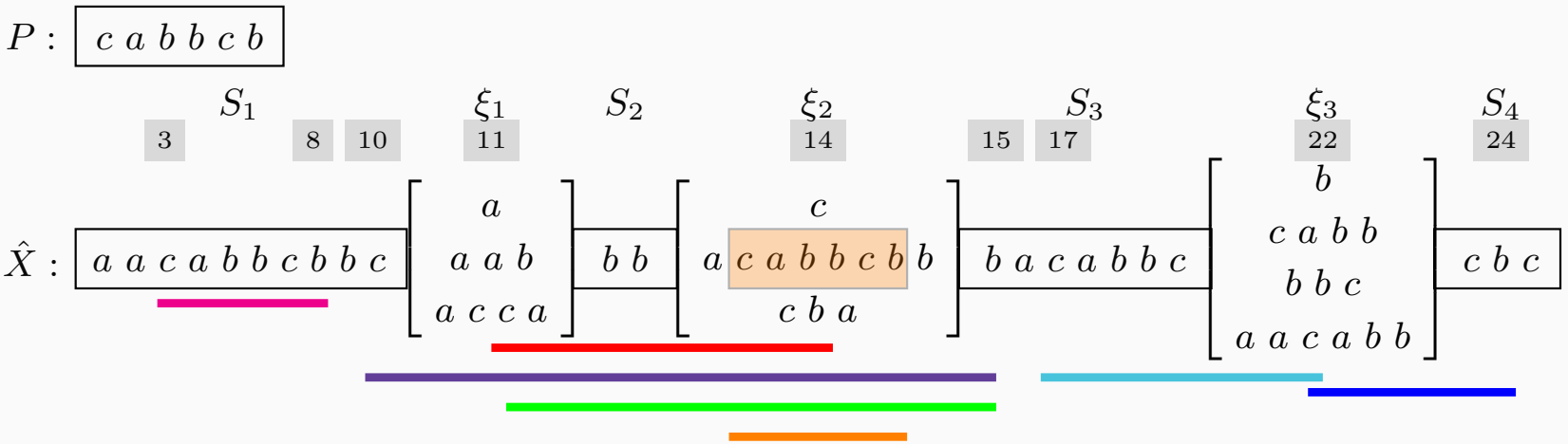


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

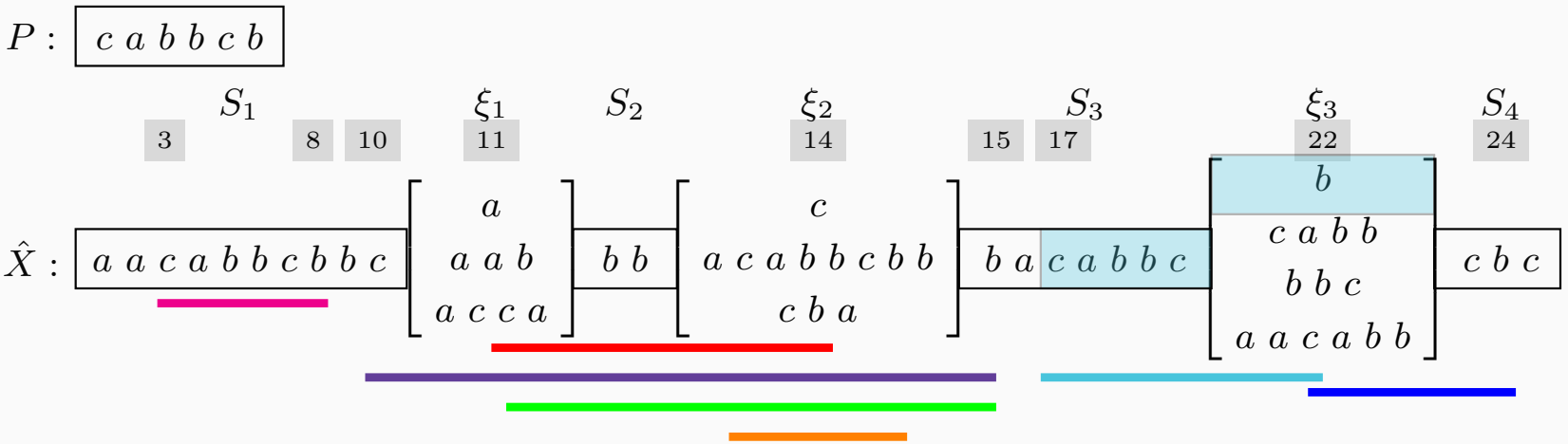


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

Illustration

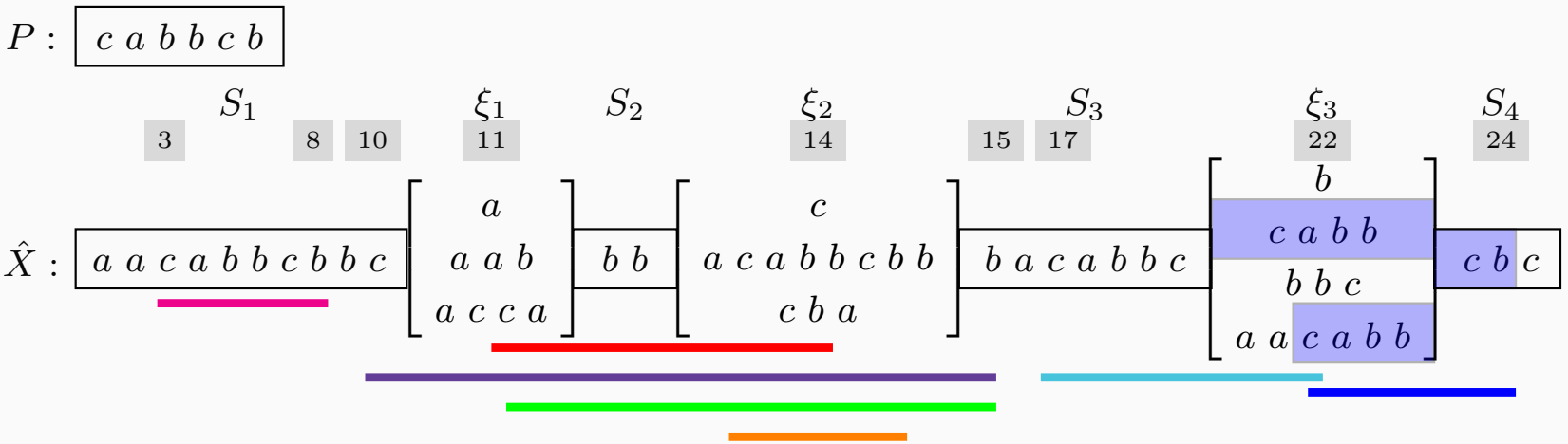


Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Occurrence ($\hat{X} \simeq Y$)

In an elastic-degenerate string (text) $\hat{T}$, a solid string (pattern) P is said to have an occurrence starting and ending at positions i and j respectively, if $P \simeq \hat{T}[i\,..\,j]$. An occurrence is represented as the pair of starting position i (head) and ending position j (tail).

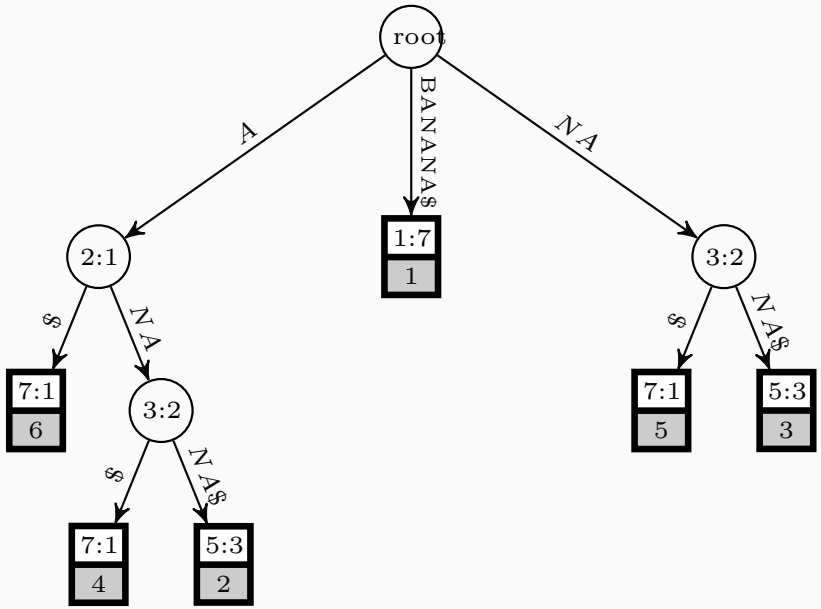
Illustration



Occurrences: $(3, 8), (10, 15), (11, 14), (11, 15), (14, 14), (17, 22), (22, 24)$

Suffix Tree ([Crochemore et al., 2007])

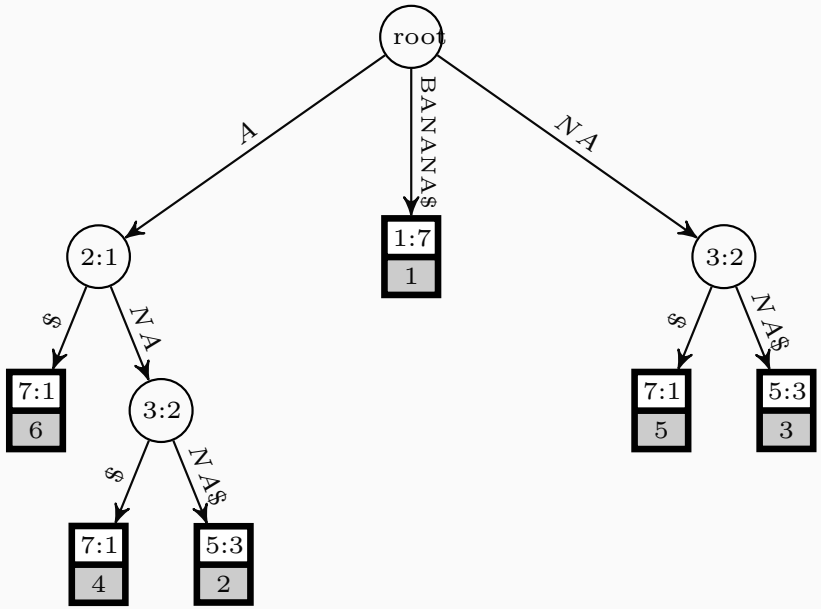
- The suffix tree $\mathcal{S}(X)$ of a non-empty string X of length n is a compact trie representing all the suffixes of X such that $\mathcal{S}(X)$ has n leaves, labelled from 1 to n.
- Example: $X =$ "BANANA\$";



- Linear Time and Space Construction ([Weiner, 1973, McCreight, 1976, Ukkonen, 1995])
- After Linear time pre-processing, constant-time answer to LCP queries.
- Generalised Suffix trees

Suffix Tree ([Crochemore et al., 2007])

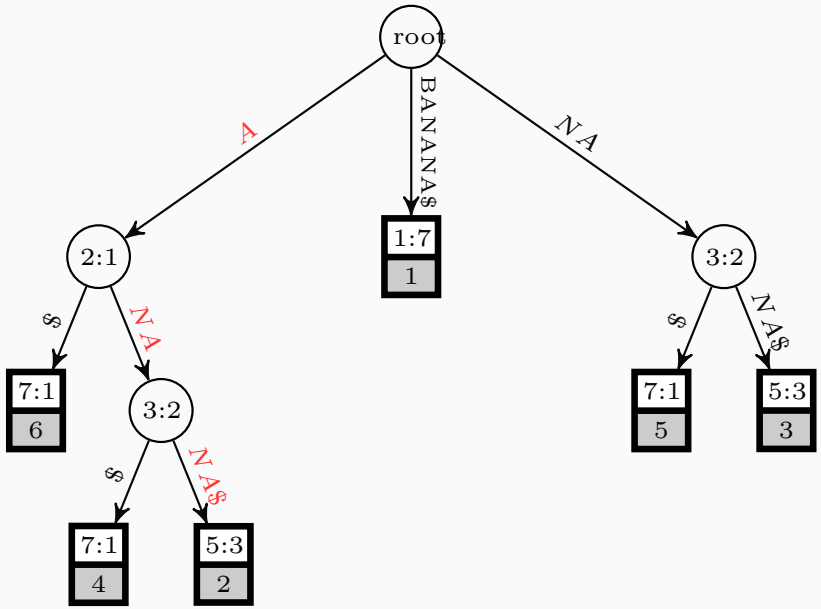
- The suffix tree $\mathcal{S}(X)$ of a non-empty string X of length n is a compact trie representing all the suffixes of X such that $\mathcal{S}(X)$ has n leaves, labelled from 1 to n.
- Example: $X =$ "BANANA\$"; suffices: { BANANA\$, ANANA\$, NANA\$, ANA\$, NA\$, A\$, \$ }



- Linear Time and Space Construction ([Weiner, 1973, McCreight, 1976, Ukkonen, 1995])
- After Linear time pre-processing, constant-time answer to LCP queries.
- Generalised Suffix trees

Suffix Tree ([Crochemore et al., 2007])

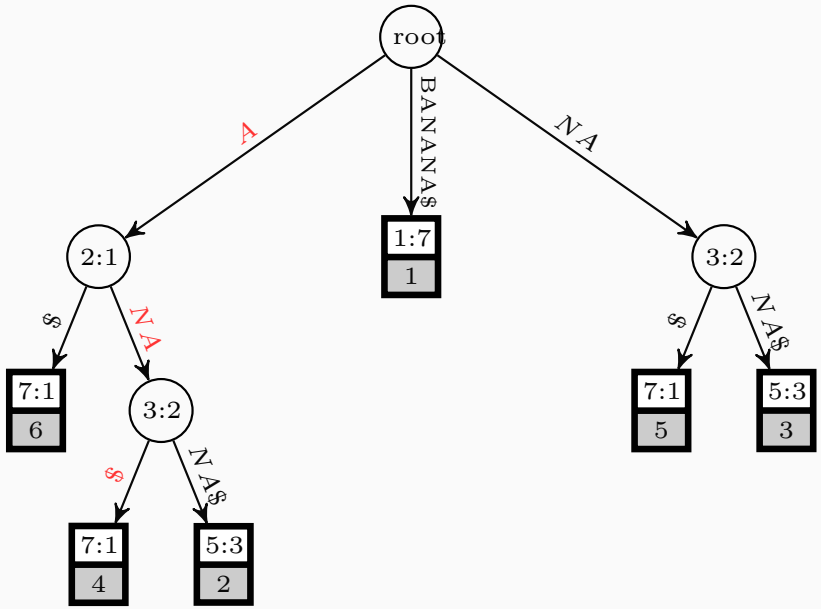
- The suffix tree $\mathcal{S}(X)$ of a non-empty string X of length n is a compact trie representing all the suffixes of X such that $\mathcal{S}(X)$ has n leaves, labelled from 1 to n.
- Example: X = "BANANA$"; suffices: { BANANA$, ANANA$, NANA$, ANA$, NA$, A$, $ }



- Linear Time and Space Construction ([Weiner, 1973, McCreight, 1976, Ukkonen, 1995])
- After Linear time pre-processing, constant-time answer to LCP queries.
- Generalised Suffix trees

Suffix Tree ([Crochemore et al., 2007])

- The suffix tree $\mathcal{S}(X)$ of a non-empty string X of length n is a compact trie representing all the suffixes of X such that $\mathcal{S}(X)$ has n leaves, labelled from 1 to n.
- Example: X = "BANANA$"; suffices: { BANANA$, ANANA$, NANA$, ANA$, NA$, A$, $ }



- Linear Time and Space Construction ([Weiner, 1973, McCreight, 1976, Ukkonen, 1995])
- After Linear time pre-processing, constant-time answer to LCP queries.
- Generalised Suffix trees

KMP Algorithm ([Knuth et al., 1977])

- A linear time algorithm for finding all occurrences of a pattern P in a text T.
- It pre-processes P by computing a failure function f that indicates the maximum possible shift using previously performed symbol comparisons.
- By using the failure function, it achieves an optimal search time of $\mathcal{O}(n)$ after $\mathcal{O}(m)$-time pre-processing, where n is the length of T and $m < n$ is the length of P.

ALGORITHM

Problem Definition

Input

An elastic-degenerate text $\hat{T} = S_1\xi_1 S_2 \dots \xi_{k-1} S_k$ of length n and total size N, a pattern P of length $m < N$.

Output

All the occurrences of P in $\hat{T}$.

Types of occurrences

- Type 1: Solid starting position
- Type 2: Elastic-Degenerate starting position

Algorithm

- Preprocessing
- Search
  - Type 1:
  - Type 2:

Algorithm

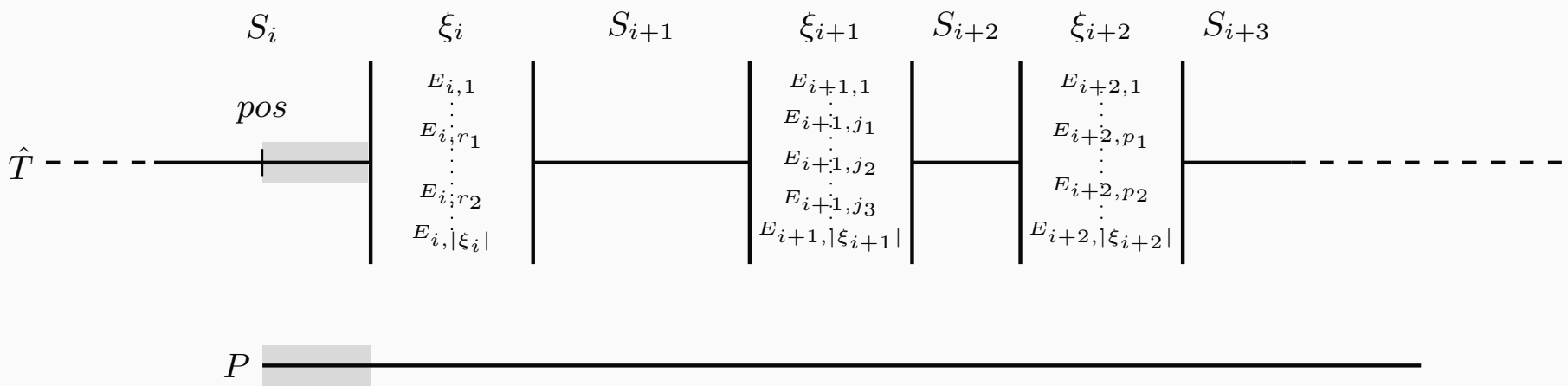
- Preprocessing
  - Compute failure-function of P.
  - Create the generalised suffix tree $\mathbb{S}_S$ for the set $\{P, S_1, S_2, \ldots, S_k\}$ of strings.
  - Create the generalised suffix tree $\mathbb{S}_\xi$ for the set $\{P\} \cup \xi_1 \cup \xi_2 \cup \ldots \cup \xi_{k-1}$.
  - Pre-process these two suffix trees so as to answer LCA queries in constant time.
- Search
  - Type 1:
  - Type 2:

Algorithm

- Preprocessing
- Search
  - Type 1:
    - Use KMP to search (shifting using failure function) in some seed S_i.
    - When an elastic-degenerate symbol is hit, mark the corresponding position: Incremental extension of marked positions using LCP queries.
  - Type 2:

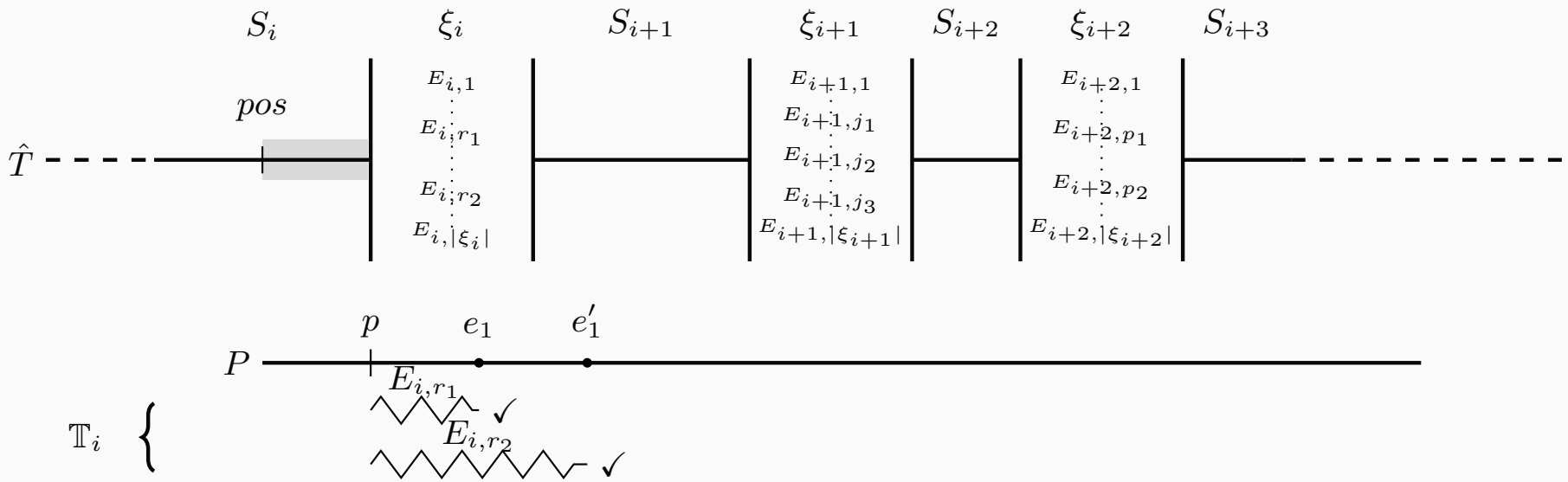
- Preprocessing
- Search
  - Type 1:
    - Use KMP to search (shifting using failure function) in some seed S_i.
    - When an elastic-degenerate symbol is hit, mark the corresponding position: Incremental extension of marked positions using LCP queries.
  - Type 2:

Illustration



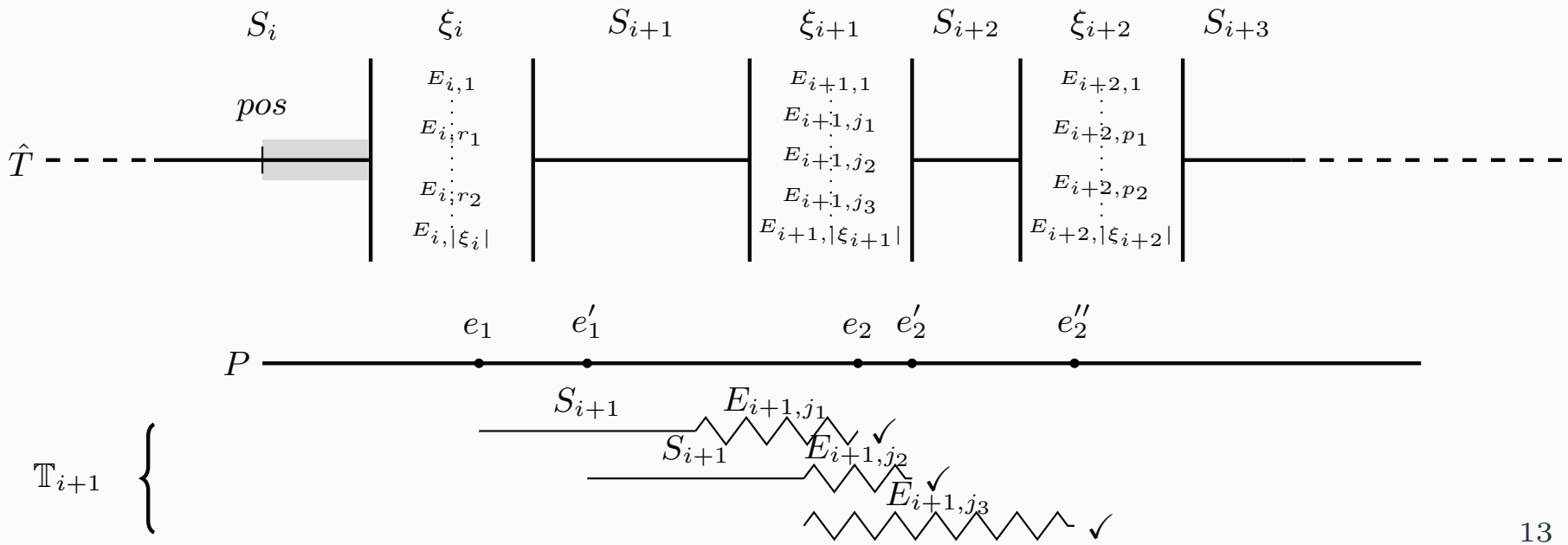
- Preprocessing
- Search
  - Type 1:
    - Use KMP to search (shifting using failure function) in some seed S_i.
    - When an elastic-degenerate symbol is hit, mark the corresponding position: Incremental extension of marked positions using LCP queries.
  - Type 2:

Illustration



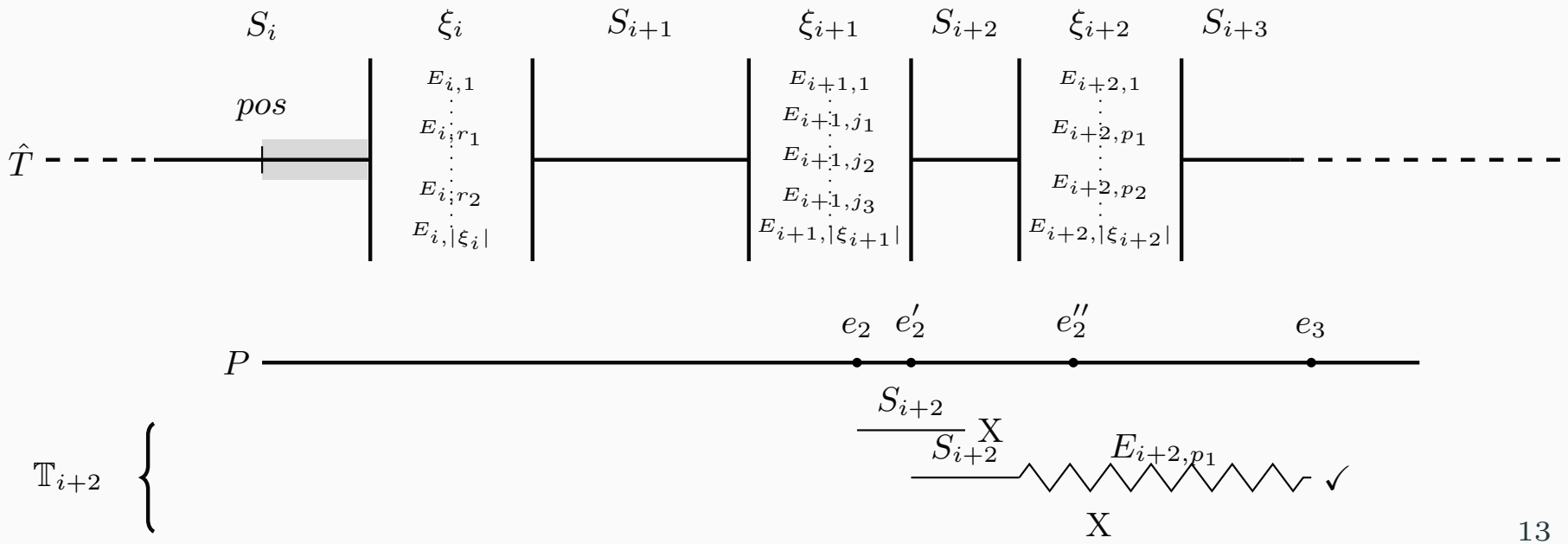
- Preprocessing
- Search
  - Type 1:
    - Use KMP to search (shifting using failure function) in some seed S_i.
    - When an elastic-degenerate symbol is hit, mark the corresponding position: Incremental extension of marked positions using LCP queries.
  - Type 2:

Illustration



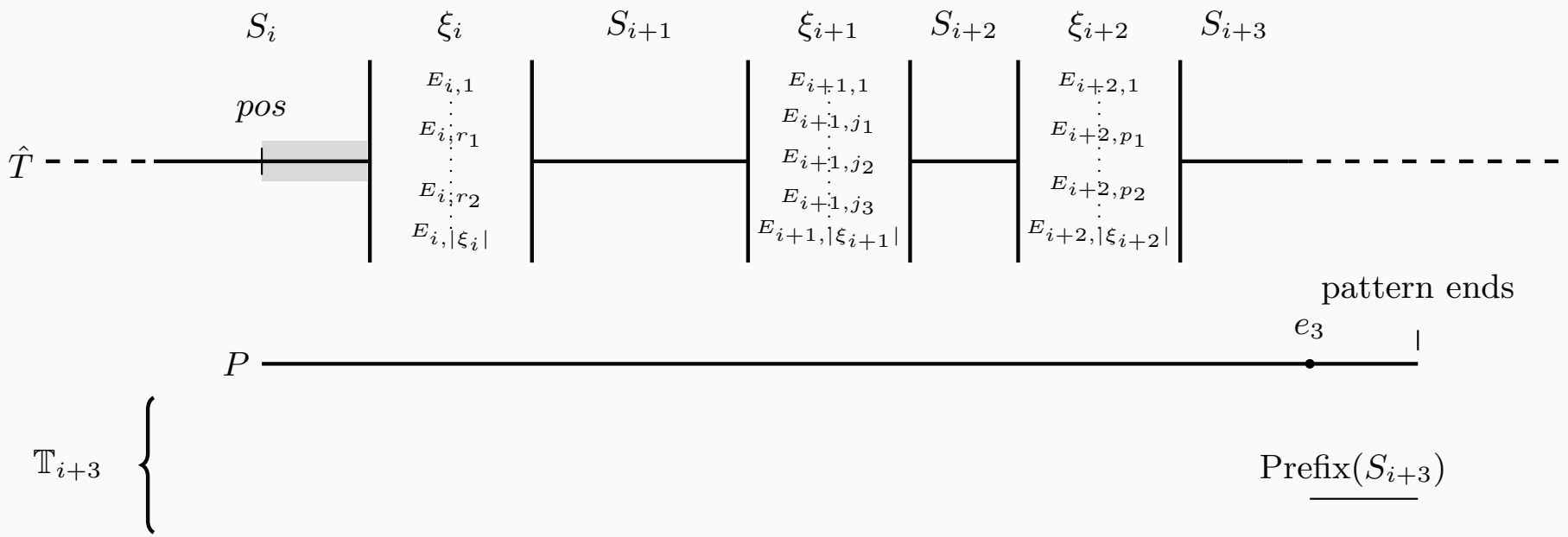
- Preprocessing
- Search
  - Type 1:
    - Use KMP to search (shifting using failure function) in some seed S_i.
    - When an elastic-degenerate symbol is hit, mark the corresponding position: Incremental extension of marked positions using LCP queries.
  - Type 2:

Illustration



- Preprocessing
- Search
  - Type 1:
  - Type 2:

Illustration



Algorithm

- Preprocessing
- Search
  - Type 1:
  - Type 2:
    - Use KMP to search in as well as marking suffixes of each $E_{i,j}$ appearing as prefixes of P.
    - Incremental extension of marked positions using LCP queries.

ANALYSIS

Analysis

- Preprocessing.
- Search.
  - KMP.
  - Extension:
    - Checking each string in ξ_i: Let parameter α represent the maximum number of strings in any elastic-degenerate symbol of the text.
    - Number of ticks in an iteration.
    - Number of iterations: the number of the ξ_i spanned by the occurrence of P being tested. Let γ is the parameter representing the maximum number of elastic-degenerate symbols spanned by any occurrence of the pattern in the text.

Analysis

- Preprocessing. → $O(N + m)$
- Search.
  - KMP.
  - Extension:
    - Checking each string in ξ_i: Let parameter α represent the maximum number of strings in any elastic-degenerate symbol of the text.
    - Number of ticks in an iteration.
    - Number of iterations: the number of the ξ_i spanned by the occurrence of P being tested. Let γ is the parameter representing the maximum number of elastic-degenerate symbols spanned by any occurrence of the pattern in the text.

Analysis

- Preprocessing. → $O(N + m)$
- Search.
  - KMP. → $O(N)$
  - Extension:
    - Checking each string in ξ_i: Let parameter α represent the maximum number of strings in any elastic-degenerate symbol of the text.
    - Number of ticks in an iteration.
    - Number of iterations: the number of the ξ_i spanned by the occurrence of P being tested. Let γ is the parameter representing the maximum number of elastic-degenerate symbols spanned by any occurrence of the pattern in the text.

Analysis

- Preprocessing. → $O(N + m)$
- Search.
  - KMP. → $O(N)$
  - Extension:
    - Checking each string in ξ_i: Let parameter α represent the maximum number of strings in any elastic-degenerate symbol of the text. → $O(\alpha)$
    - Number of ticks in an iteration. → $O(m)$
    - Number of iterations: the number of the ξ_i spanned by the occurrence of P being tested. Let γ is the parameter representing the maximum number of elastic-degenerate symbols spanned by any occurrence of the pattern in the text. → $O(\gamma)$

Analysis

- Preprocessing. → $O(N + m)$
- Search.
  - KMP. → $O(N)$
  - Extension: → $O(\alpha\gamma mn)$
    - Checking each string in ξ_i: Let parameter α represent the maximum number of strings in any elastic-degenerate symbol of the text.
    - Number of ticks in an iteration.
    - Number of iterations: the number of the ξ_i spanned by the occurrence of P being tested. Let γ is the parameter representing the maximum number of elastic-degenerate symbols spanned by any occurrence of the pattern in the text.

Analysis

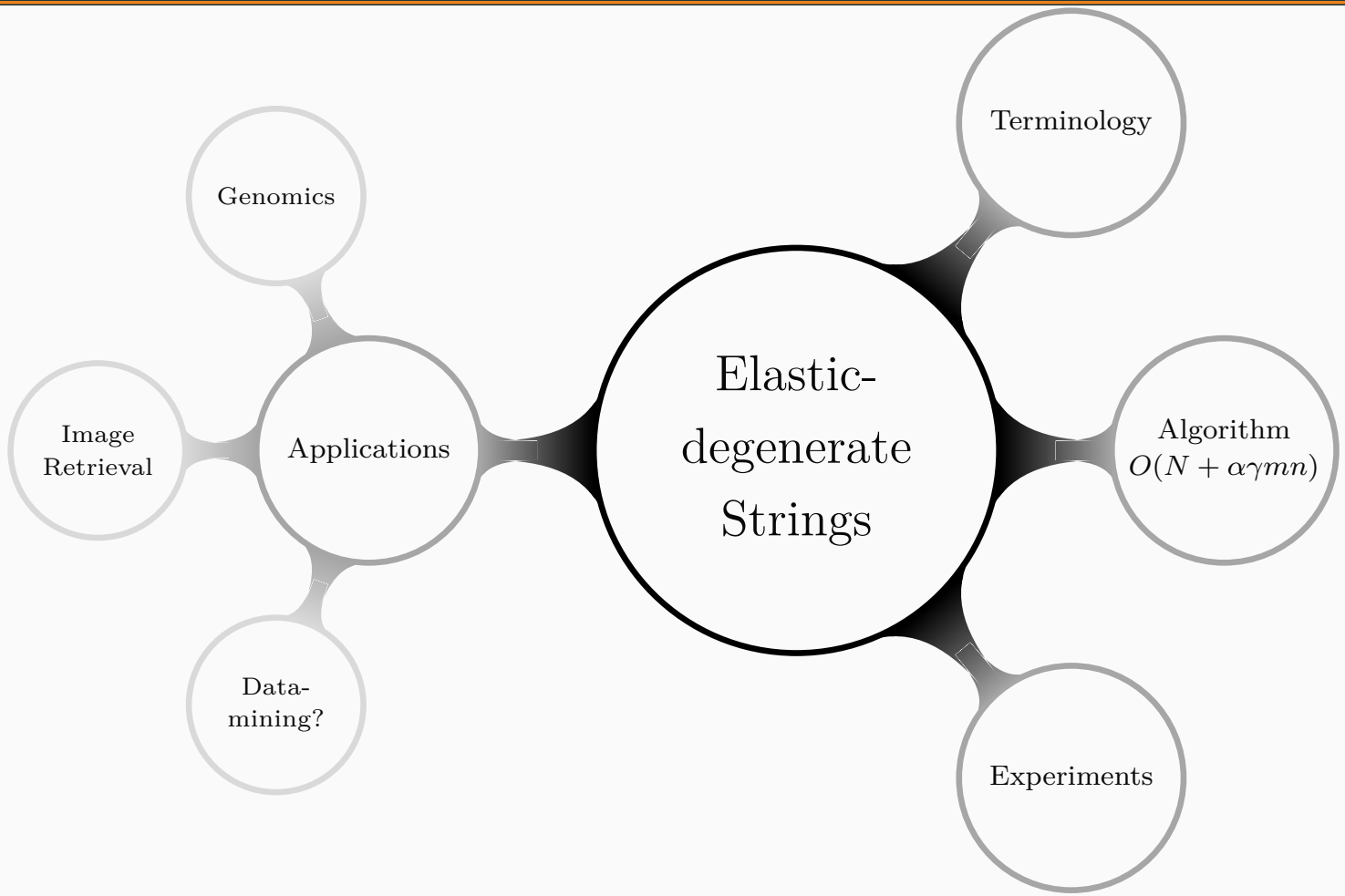
- Preprocessing. → $O(N + m)$
- Search.
  - KMP. → $O(N)$
  - Extension: → $O(\alpha\gamma mn)$
    - Checking each string in ξ_i: Let parameter α represent the maximum number of strings in any elastic-degenerate symbol of the text.
    - Number of ticks in an iteration.
    - Number of iterations: the number of the ξ_i spanned by the occurrence of P being tested. Let γ is the parameter representing the maximum number of elastic-degenerate symbols spanned by any occurrence of the pattern in the text.

Running time

$O(N + \alpha\gamma mn)$

SUMMARY

Summary



A proof-of-concept implementation of this algorithm can be accessed at https://github.com/Ritu-Kundu/ElDeS.

References

Crochemore, M., Hancart, C., and Lecroq, T. (2007).
Algorithms on Strings.
Cambridge University Press.
392 pages.

Knuth, D. E., James H. Morris, J., and Pratt, V. R. (1977).
Fast pattern matching in strings.
SIAM Journal on Computing, 6(2):323–350.

McCreight, E. M. (1976).
A space-economical suffix tree construction algorithm.
Journal of the ACM (JACM), 23(2):262–272.

Ukkonen, E. (1995).
On-line construction of suffix trees.
Algorithmica, 14(3):249–260.

Weiner, P. (1973).
Linear pattern matching algorithms.
In Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory, pages 1–11. Institute of Electrical Electronics Engineer.

Thank You!