Dynamic index and LZ factorization in compressed space

Takaaki Nishimoto[1], Tomohiro I[2], Shunsuke Inenaga[1], Hideo Bannai[1], and Masayuki Takeda[1]
1. Kyushu University, Japan
2. Kyushu Institute of Technology, Japan

Our contributions

1. We proposed a new dynamic index working in compressed space.
2. We proposed a new Lempel-Ziv 77(LZ77) factorization algorithm working in compressed space.

In this presentation, we focus on the first result.

Dynamic text indexing problem

Consider a dynamic text T	
Find(P)	Return all occurrences of a given pattern P in T
Insert(Y, i)	Insert a given string Y into T at a given position i
Delete(i, k)	Delete a given substring $T[i..i+k-1]$ from T

Find(TGT) = 1, 3, 14

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	G	T	G	T	T	A	T	T	G	G	T	T	T	G	T	C	G

Text T = TGTGTTATTGGTTTGTCG

Dynamic text indexing problem

Consider a dynamic text T	
Find(P)	Return all occurrences of a given pattern P in T
Insert(Y, i)	Insert a given string Y into T at a given position i
Delete(i, k)	Delete a given substring $T[i..i+k\text{-}1]$ from T

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

Text T = TAGAGT$\underbrace{\text{GTTA}}_{\text{Insert(GTTA, 7)}}$TTGGTTTGTCG

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Text T = TA$\underbrace{\text{GAGTTT}}_{\text{Delete(3, 6)}}$GGTTTGTCG

Dynamic text indexing problem

Consider a dynamic text T	
Find(P)	Return all occurrences of a given pattern P in T
Insert(Y, i)	Insert a given string Y into T at a given position i
Delete(i, k)	Delete a given substring $T[i..i+k-1]$ from T

1 2 3 4 5 6 7 8 9

Text T = TAGAGTTTG

Edit

Find(TGAT) = 9, 14

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Text T = AGAGTTATTGATTTGATG

Background

- Pattern matching on strings is a basic operation and is used by various applications.
- Non-compressed dynamic indexes require at least $N \log \sigma$ bits of space, where N is the length of a given text and σ is the alphabet size. This is inefficient when the text is large.
- Many non-compressed dynamic indexes have been proposed but only a few compressed dynamic indexes exist.
- Hence we propose a new compressed dynamic index.

Previous results

Hon et al, '04	
Space	$O((NH_0+N)/\varepsilon)$ bits
Update	$O((\|Y\| + \sqrt{N}) \log^{2+\varepsilon} N)$ time
Find(P)	$O(\|P\| \log^2 N (\log^\varepsilon N + \log \sigma) + occ \log^{1+\varepsilon} N)$ time

Salson et al, '10 (Dynamic FM-Index)
Experimental result. Although their approach works well in practice, updates require $O(N \log N)$ time in the worst case.

- N : length of a text T
- $|Y|$: length of an inserted string or deleted substring
- σ : alphabet size
- $0 < \varepsilon \leq 1$: parameter
- occ : the number of occurrences of a given pattern P in T
- H_0 : the zeroth order empirical entropy of T, $H_0 \leq \log \sigma$

Our result

This work	
Space	$O(\min\{z \log N \log^* N, N\} \log N)$ bits
Update	amortized $O((\lvert Y\rvert + \log N \log^* N) \log w \log N \log^* N)$ time
Find(P)	$O(c\lvert P\rvert + \log N \log w \log \lvert P\rvert (\log^* N)^2 + occ \log N)$ time

- N : length of a text T
- c : time for predecessor queries
- z : size of LZ77 factorization of T, $z = O(N/\log_\sigma N)$ [e.g. Kärkkäinen]
- σ : alphabet size
- $\lvert Y\rvert$: length of an inserted string or deleted substring
- occ : the number of occurrences of a given pattern P in T
- $w = O(\min\{z \log N \log^* N, N\})$

➢ Our amortized update time is better than Hon et al.'s.

➢ Our find queries are faster than Hon et al.'s when the $\lvert P\rvert$ term is dominating in find query time.

Table of contents

1. Our contributions
2. Preliminaries
   - Locally Consistent Parsing
   - Signature Encoding
   - Properties of Signature Encoding
3. Our dynamic index

Locally consistent parsing is an important function in our dynamic index.

Locally Consistent Parsing [Mehlhorn+, '97, Alstrup+, '98]

For any m, there exists a function $f : [1..m]^k \rightarrow \{0,1\}^k$ that satisfies the following properties :

1. the output binary string $d = d_1..d_k$ can be computed in $O(|d|)$ time;
2. no 1's appear consecutively in d;
3. at most three 0's appear consecutively in d;
4. d_i is locally determined by $s_{i-\Delta_\mathrm{L}}, \ldots, s_{i+\Delta_\mathrm{R}}$ ($\Delta_\mathrm{L} = \log^* m + 6$, $\Delta_\mathrm{R} = 4$);

provided that the input sequence $s = s_1..s_k$ does not contain a run.

Input	s	1	2	3	5	2	3	4	2	5	1	2	3	5	2	3	4	2	5

$\Downarrow f$

Output	d	1	0	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0

Locally Consistent Parsing [Mehlhorn+, '97, Alstrup+, '98]

For any m, there exists a function $f : [1..m]^k \rightarrow \{0,1\}^k$ that satisfies the following properties :

1. the output binary string $d = d_1..d_k$ can be computed in $O(|d|)$ time;
2. no 1's appear consecutively in d;
3. at most three 0's appear consecutively in d;
4. d_i is locally determined by $s_{i-\Delta_L}, .., s_{i+\Delta_R}$ ($\Delta_L = \log^* m + 6$, $\Delta_R = 4$);

provided that the input sequence $s = s_1..s_k$ does not contain a run.

Input	s	1	2	3	5	2	3	4	2	5	1	2	3	5	2	3	4	2	5
										$\Downarrow f$									
Output	d	1	0	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0

Locally Consistent Parsing [Mehlhorn+, '97, Alstrup+, '98]

For any m, there exists a function $f : [1..m]^k \to \{0,1\}^k$ that satisfies the following properties :

1. the output binary string $d = d_1..d_k$ can be computed in $O(|d|)$ time;
2. no 1's appear consecutively in d;
3. at most three 0's appear consecutively in d;
4. d_i is locally determined by $s_{i-\Delta_L}, .., s_{i+\Delta_R}$ ($\Delta_L = \log^* m + 6$, $\Delta_R = 4$);

provided that the input sequence $s = s_1..s_k$ does not contain a run.

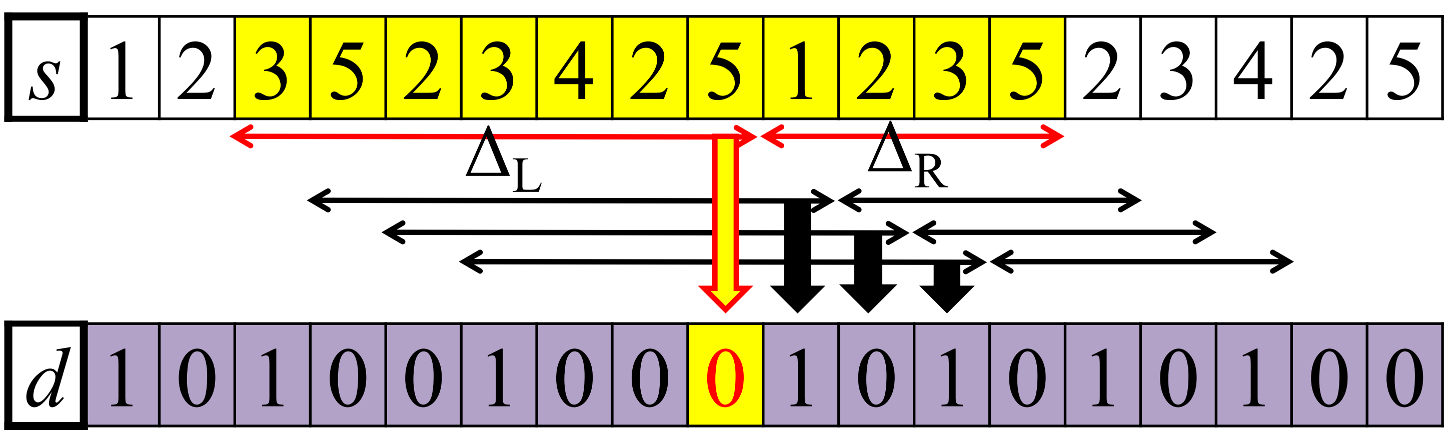
Input	s	1	2	3	5	2	3	4	2	5	1	2	3	5	2	3	4	2	5
										$\Downarrow f$									
Output	d	1	0	1	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0

Locally Consistent Parsing [Mehlhorn+, '97, Alstrup+, '98]

For any m, there exists a function $f : [1..m]^k \rightarrow \{0,1\}^k$ that satisfies the following properties :

1. the output binary string $d = d_1..d_k$ can be computed in $O(|d|)$ time;
2. no 1's appear consecutively in d;
3. at most three 0's appear consecutively in d;
4. d_i is locally determined by $s_{i-\Delta_L}, .. , s_{i+\Delta_R}$ ($\Delta_L = \log^* m + 6$, $\Delta_R = 4$);

provided that the input sequence $s = s_1..s_k$ does not contain a run.



Locally Consistent Parsing [Mehlhorn+, '97, Alstrup+, '98]

For any m, there exists a function $f : [1..m]^k \rightarrow \{0,1\}^k$ that satisfies the following properties :

1. the output binary string $d = d_1..d_k$ can be computed in $O(|d|)$ time;
2. no 1's appear consecutively in d;
3. at most three 0's appear consecutively in d;
4. d_i is locally determined by $s_{i-\Delta_L}, .. , s_{i+\Delta_R}$ ($\Delta_L = \log^* m + 6$, $\Delta_R = 4$);

provided that the input sequence $s = s_1..s_k$ does not contain a run.

A *run* is a string of length at least 2 consisting of the same character

s	1	2	3	5	5	5	4	2	5	1	2	3	5	2	3	4	2	5

s'	1	2	3	5	2	3	4	2	5	1	2	3	5	2	3	4	2	5

Table of contents

1. Our contributions
2. Preliminary
   - Locally Consistent Parsing
   - Signature Encoding
   - Properties of Signature Encoding
3. Our dynamic index

- A signature encoding is a compressed representation of a given text.
- Our dynamic index has the input text using signature encodings.

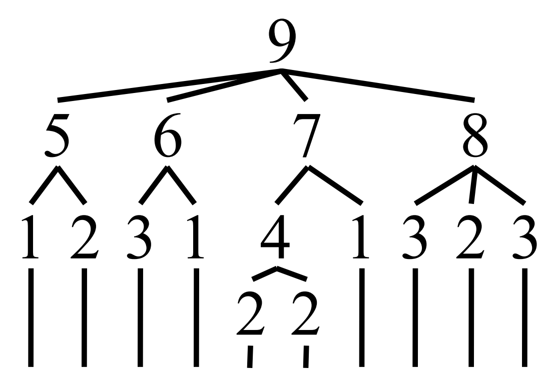
Signature Encoding (SE) [Mehlhorn et al, '97]

- SE is essentially a context free grammar, that generates a single text T.
- In SE, a *signature* represents any of the following; (1) a character, (2) 2-4 signatures, or (3) a run of a signature.
- The SE of T is determined by **locally consistent parsing**.

$1 \rightarrow C$
$2 \rightarrow A$
$3 \rightarrow B$
$4 \rightarrow 2^2$
$5 \rightarrow 1, 2$
$6 \rightarrow 3, 1$
$7 \rightarrow 4, 1$
$8 \rightarrow 3, 2, 3$
$9 \rightarrow 5, 6, 7, 8$

Signature list

The derivation tree of T w.r.t. Signature Encoding



Text $T =$ C A B C A A C B A B

Signature Encoding construction(1/5)

Assign a new signature to each distinct character of T.

While $|T_i| > 1$

1. Assign a new signature to each distinct run.
2. Compute blocks by **locally consistent parsing.**
3. Assign a new signature to each distinct block.

$1 \rightarrow C$
$2 \rightarrow A$
$3 \rightarrow B$

T_0=	1	2	3	1	2	3	2	3	2	3	2	3	2	3	2	3	1	1	1	1	2	3	2	3	2	3	2	3
Text T=	C	A	B	C	A	B	A	B	A	B	A	B	A	B	A	B	C	C	C	C	A	B	A	B	A	B	A	B

Signature Encoding construction(2/5)

Assign a new signature to each distinct character of T.
While $|T_i| > 1$
1. Assign a new signature to each distinct run.
2. Compute blocks by **locally consistent parsing.**
3. Assign a new signature to each distinct block.

$1 \rightarrow C$
$2 \rightarrow A$
$3 \rightarrow B$
$4 \rightarrow 1^4$

																	4										
T_0=	1	2	3	1	2	3	2	3	2	3	2	3	2	3	2	3	1	1	1	1	2	3	2	3	2	3	2	3
Text T=	C	A	B	C	A	B	A	B	A	B	A	B	A	B	A	B	C	C	C	C	A	B	A	B	A	B	A	B

Signature Encoding construction(3/5)

Assign a new signature to each distinct character of T.

While $|T_i| > 1$

1. Assign a new signature to each distinct run.
2. Compute blocks by **locally consistent parsing.**
3. Assign a new signature to each distinct block.

$1 \rightarrow C$
$2 \rightarrow A$
$3 \rightarrow B$
$4 \rightarrow 1^4$

locally consistent parsing

<table>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="4">1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>T_0=</td><td>1</td><td>2</td><td>3</td><td>1</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td colspan="4">4</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td></tr>
<tr><td>Text T=</td><td>C</td><td>A</td><td>B</td><td>C</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>C</td><td>C</td><td>C</td><td>C</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td></tr>
</table>

Signature Encoding construction(4/5)

Assign a new signature to each distinct character of T.

While $|T_i| > 1$

1. Assign a new signatures to each distinct run.
2. Compute blocks by **locally consistent parsing.**
3. Assign a new signatures to each distinct block.

$1 \rightarrow C$
$2 \rightarrow A$
$3 \rightarrow B$
$4 \rightarrow 1^4$
$5 \rightarrow 1, 2$
$6 \rightarrow 3, 1$
$7 \rightarrow 2, 3$
$8 \rightarrow 2, 3, 2$
$9 \rightarrow 3, 2$
$10 \rightarrow 3, 2, 3$
$11 \rightarrow 4, 2$

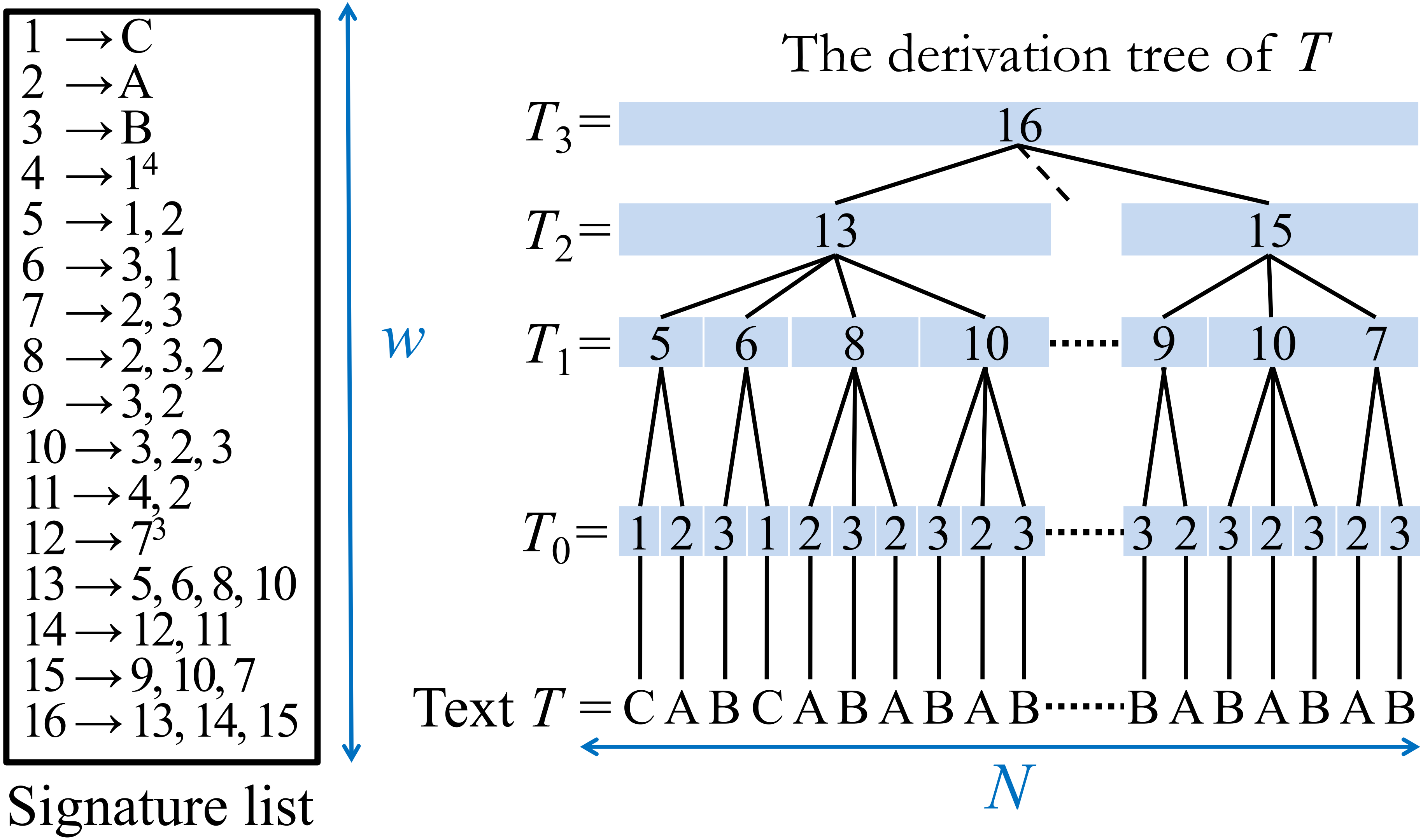
<table>
<tr><td>T_1=</td><td colspan="2">5</td><td colspan="2">6</td><td colspan="3">8</td><td colspan="3">10</td><td colspan="2">7</td><td colspan="2">7</td><td colspan="2">7</td><td colspan="5">11</td><td colspan="2">9</td><td colspan="3">10</td><td colspan="2">7</td></tr>
<tr><td>T_0=</td><td>1</td><td>2</td><td>3</td><td>1</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td colspan="4">4</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td></tr>
<tr><td>Text T=</td><td>C</td><td>A</td><td>B</td><td>C</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>C</td><td>C</td><td>C</td><td>C</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td></tr>
</table>

Signature Encoding construction(5/5)

Assign a new signature to each distinct character of T.
While $|T_i| > 1$
1. Assign a new signatures to each distinct run.
2. Compute blocks by **locally consistent parsing.**
3. Assign a new signatures to each distinct block.

$1 \rightarrow C$	$14 \rightarrow 12, 11$
...	$15 \rightarrow 9, 10, 7$
$12 \rightarrow 7^3$	$16 \rightarrow 13, 14, 15$
$13 \rightarrow 5, 6, 8, 10$	

<table>
<tr><td>T_3=</td><td colspan="28">16</td></tr>
<tr><td>T_2=</td><td colspan="10">13</td><td colspan="11">14</td><td colspan="7">15</td></tr>
<tr><td>T_1=</td><td colspan="2">5</td><td colspan="2">6</td><td colspan="3">8</td><td colspan="3">10</td><td colspan="6">12</td><td colspan="5">11</td><td colspan="2">9</td><td colspan="3">10</td><td colspan="2">7</td></tr>
<tr><td>T_0=</td><td>1</td><td>2</td><td>3</td><td>1</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td colspan="4">4</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td><td>2</td><td>3</td></tr>
<tr><td>Text T=</td><td>C</td><td>A</td><td>B</td><td>C</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>C</td><td>C</td><td>C</td><td>C</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td><td>A</td><td>B</td></tr>
</table>

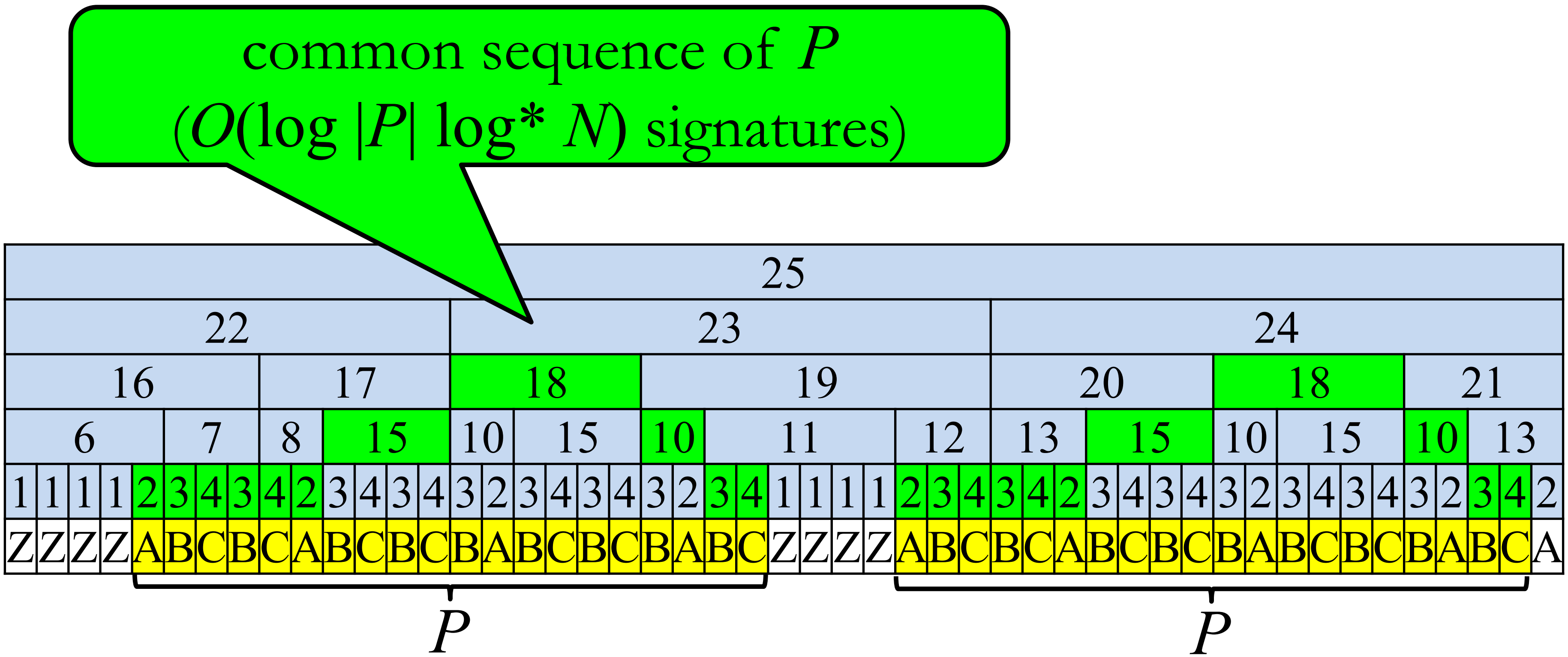


The size of derivation tree of T is $O(N)$, however, we can represent the derivation tree by the signature list of size w.

Table of contents

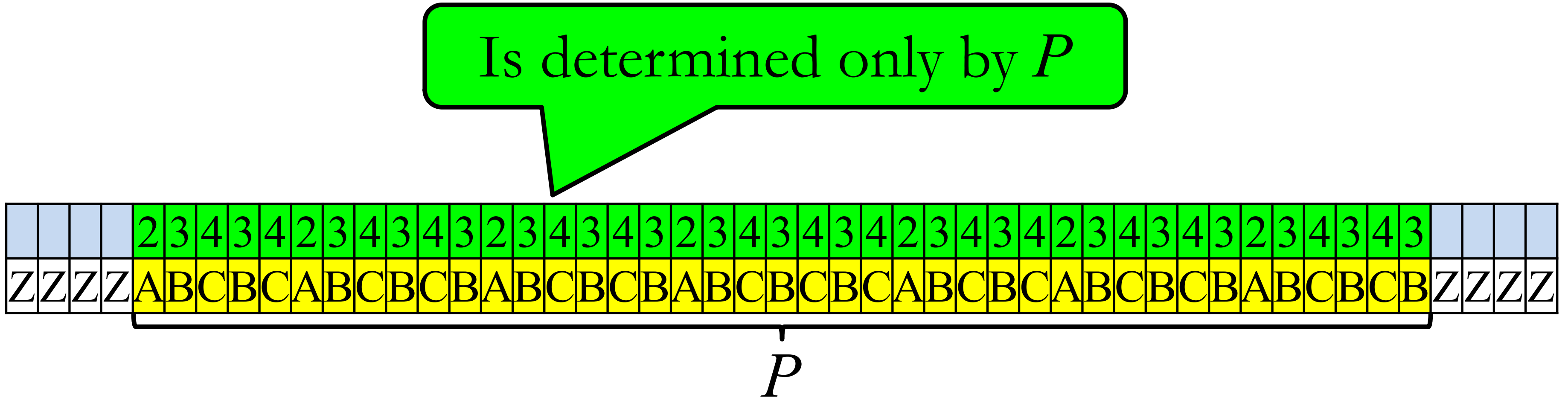
Common sequence [Sahinalp and Vishkin, '95]

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



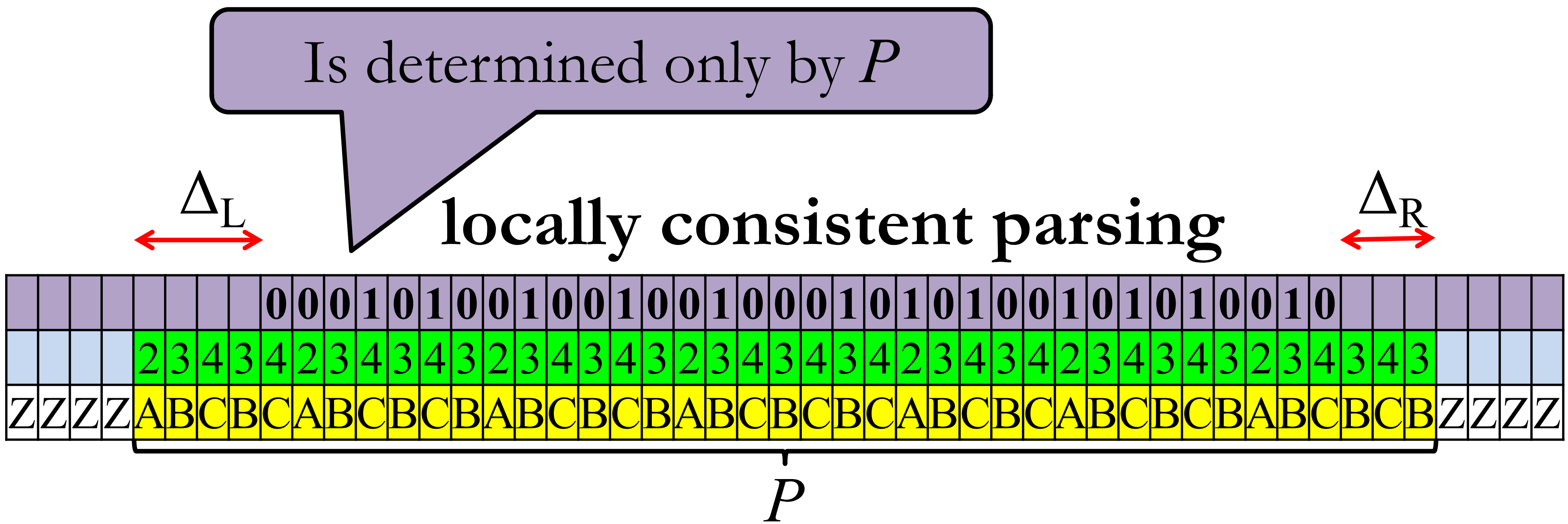
Proof of existence of common sequence(1/6)

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



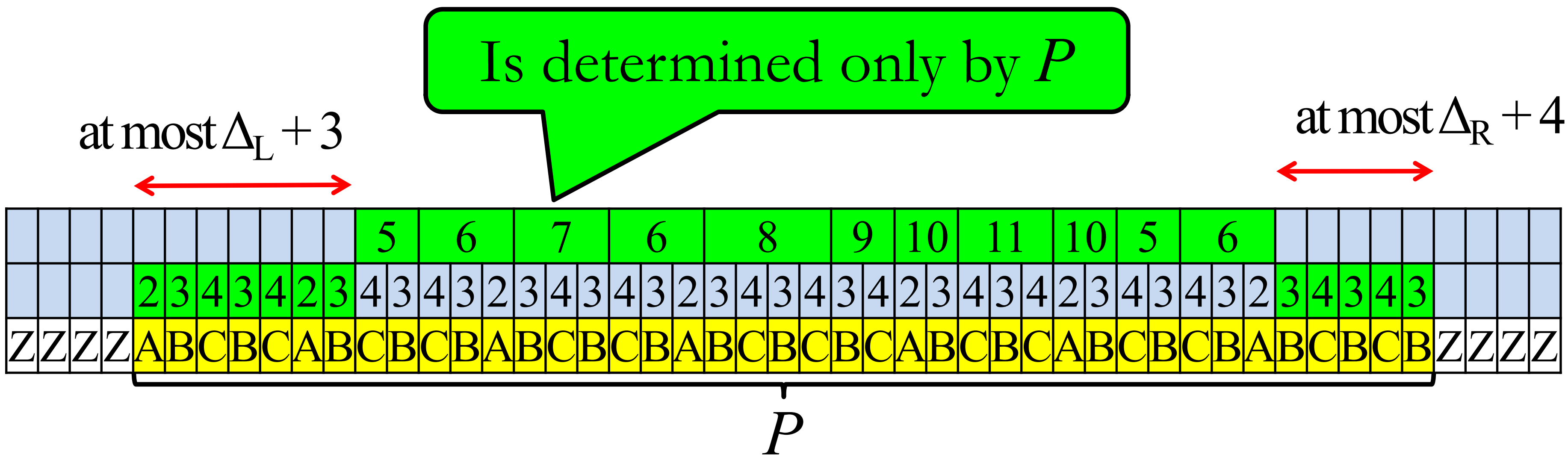
Proof of existence of common sequence(2/6)

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



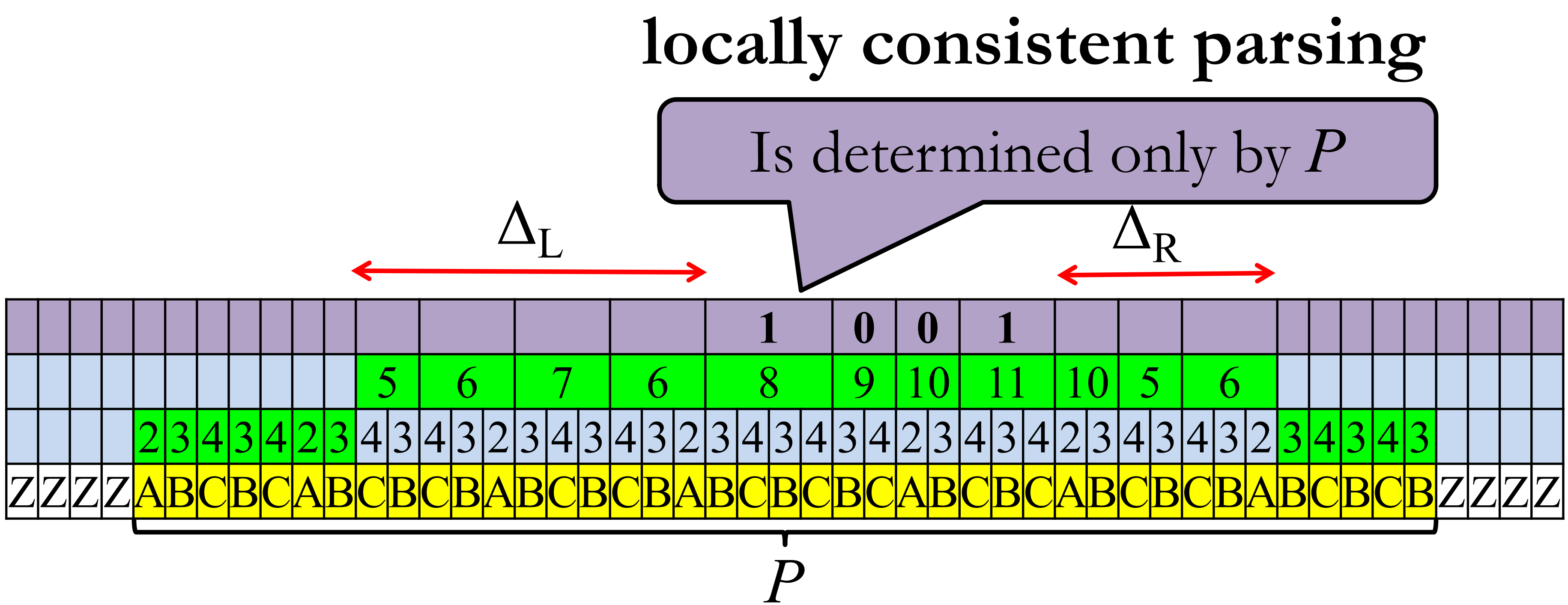
Proof of existence of common sequence(3/6)

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



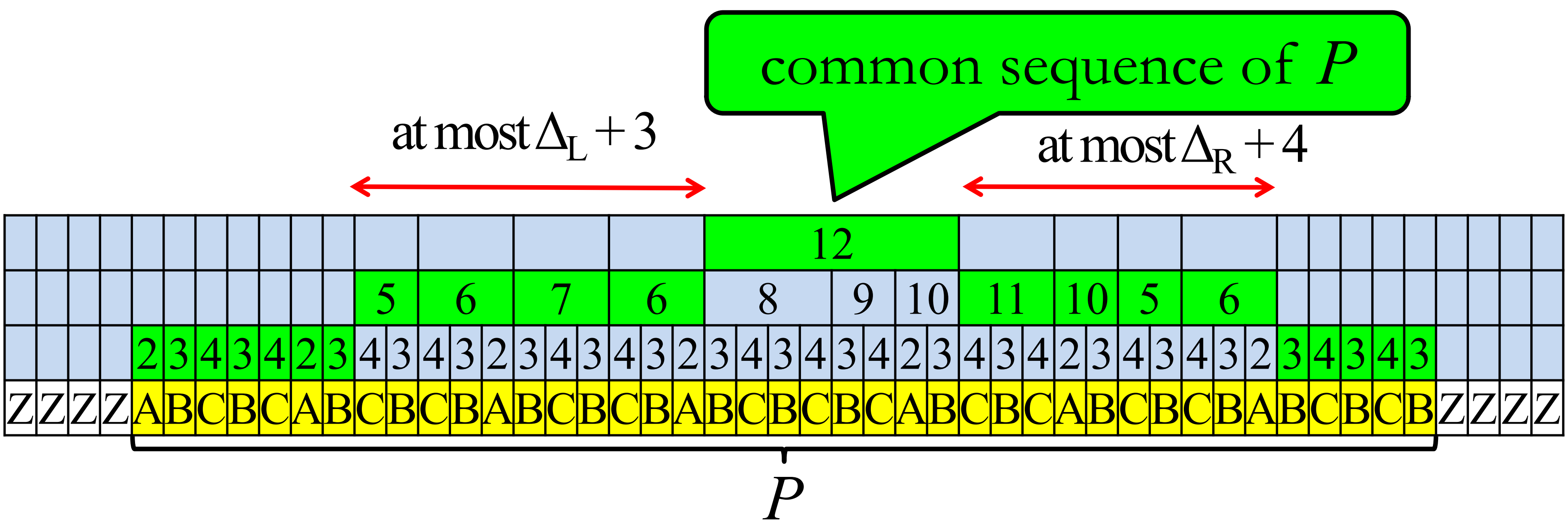
Proof of existence of common sequence(4/6)

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



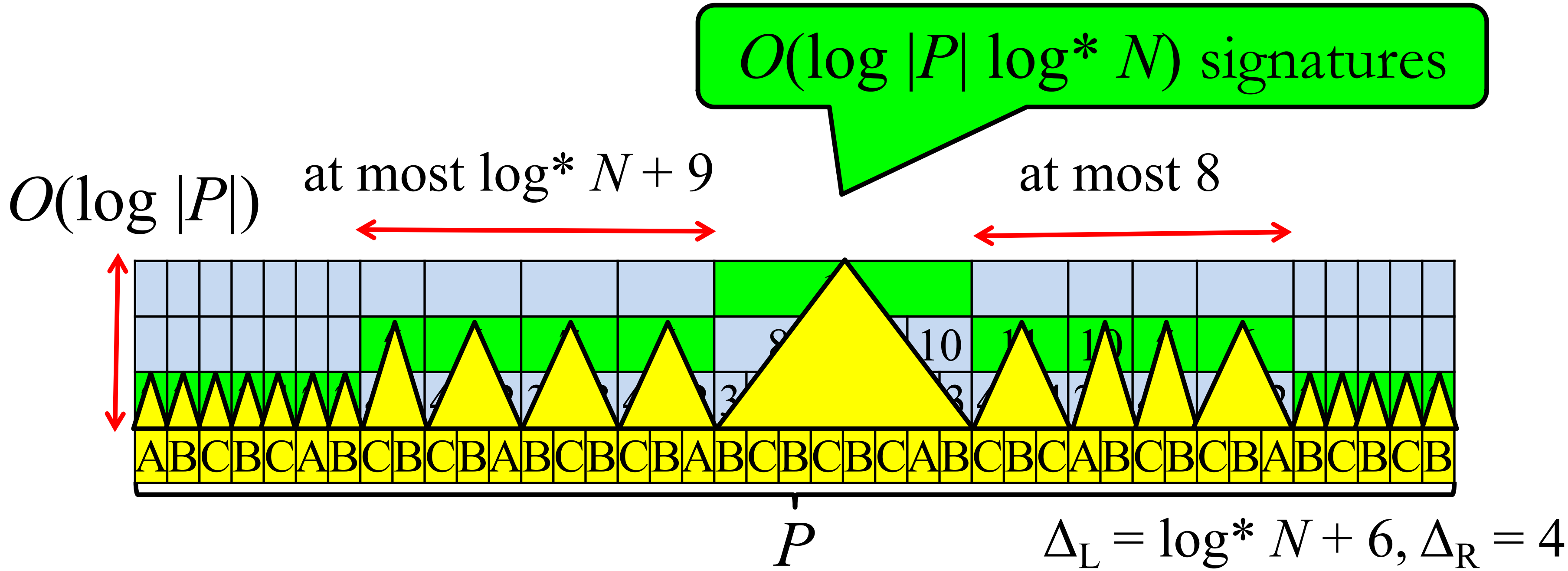
Proof of existence of common sequence(5/6)

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



Proof of existence of common sequence(6/6)

Every occurrence of substring P in a text T of length N is represented by a unique sequence of $O(\log |P| \log^* N)$ signatures



Properties of Signature Encoding

1. [LCE Query]The signature encoding of a text T supports lexicographical comparison of two suffixes of T in $O(\log N \log^* N)$ time. [Nishimoto et al, 16]
2. The size w of the signature encoding of T is $O(\min\{z \log N \log^* N, N\})$ space. [Sahinalp and Vishkin, '95]
3. The signature encoding of T supports update operations in $O(c(|Y| + \log N \log^* N))$ time. [Alstrup et al, '98]

- N : length of a given text
- c : time for predecessor queries
- z : size of LZ77 factorization of T
- ℓ : LCE length, $\ell \leq N$
- $|Y|$: length of an inserted string or deleted substring

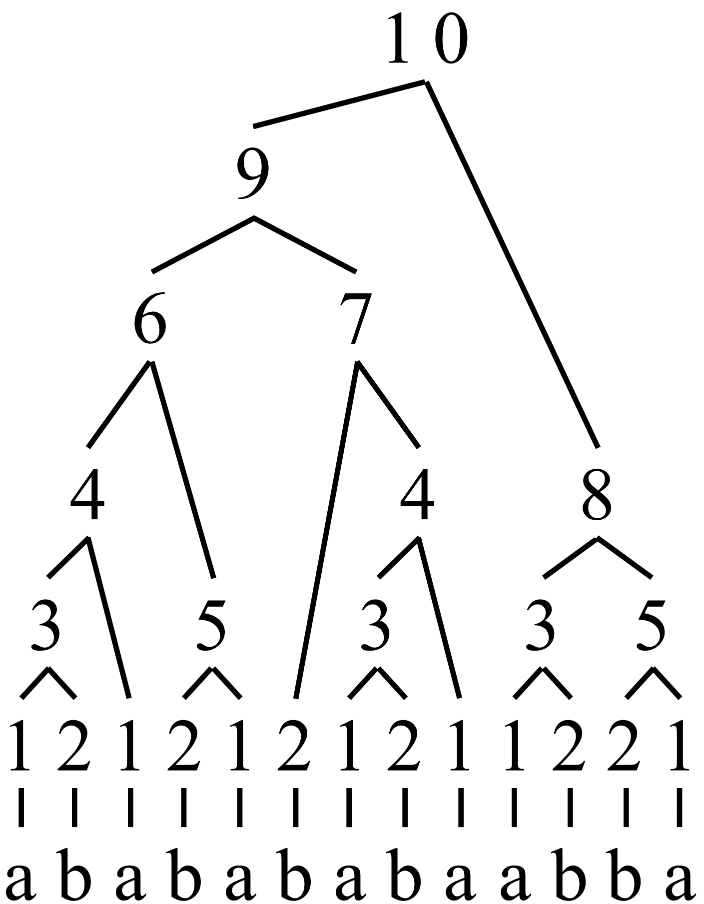
We archive a dynamic index working in compressed space using these properties.

Table of contents

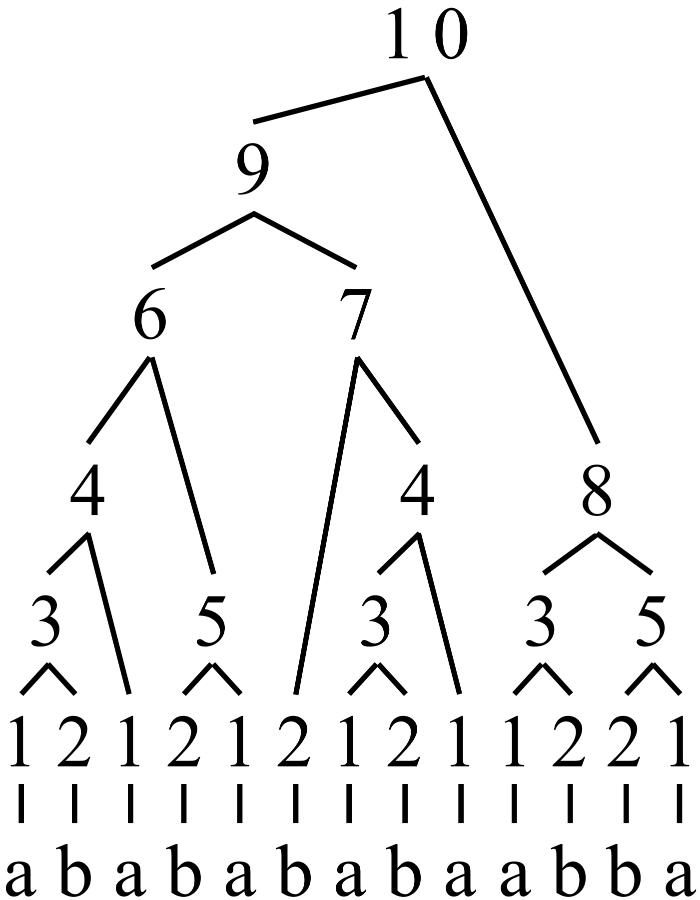
1. Our contributions
2. Preliminaries
3. Our dynamic index
   - Basic idea of pattern search[e.g., Claude+ '08]
   - Faster pattern search
   - Data structures & Update

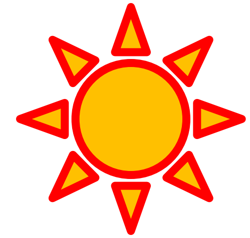
- Our dynamic index finds patterns using signature encoding.
- We follow an approach from literature, e.g., Claude et al.'s.

Preparation of our explanation

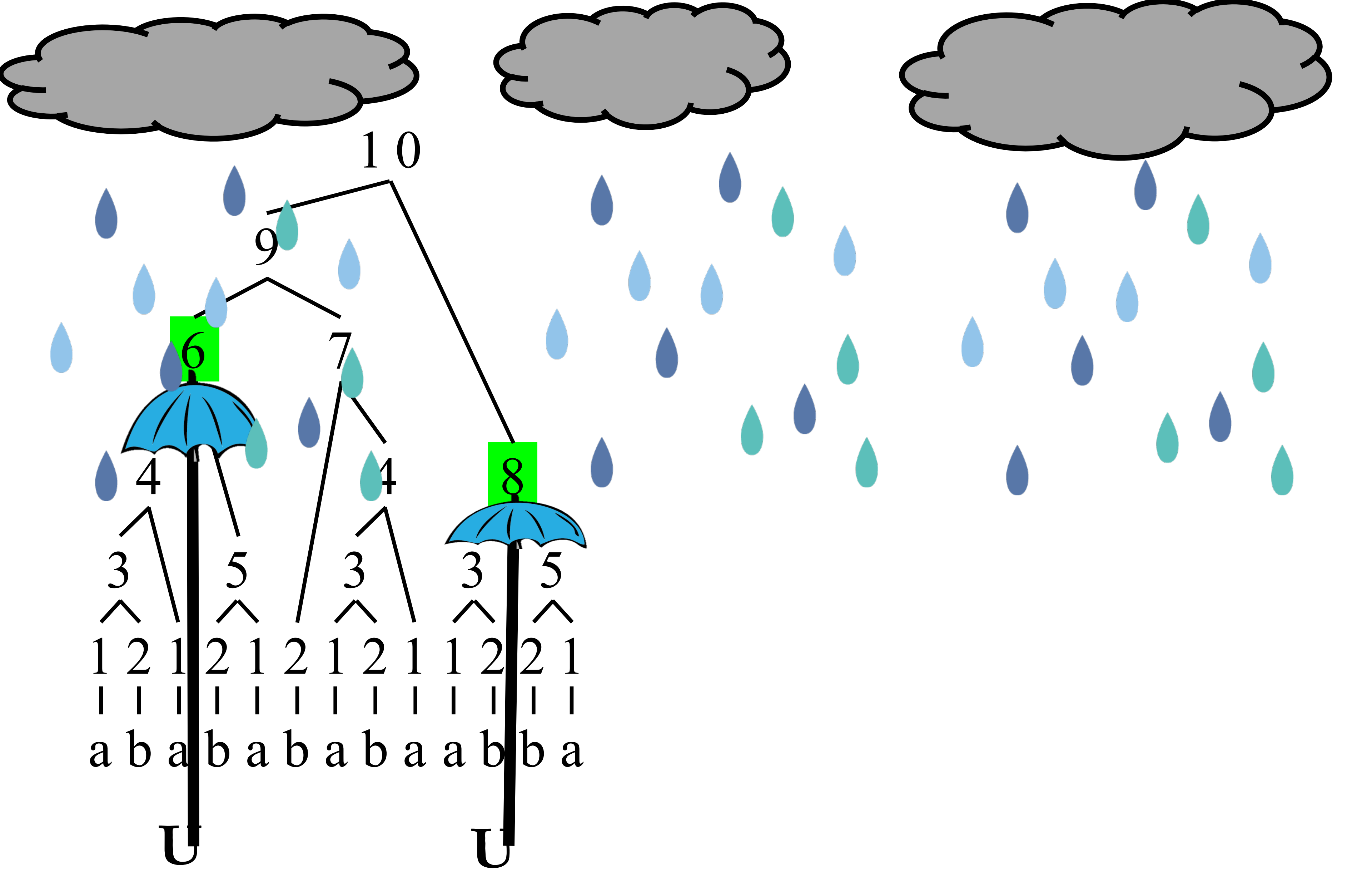


To simplify the explanation of our approach, we assume that every internal node has two children in the derivation tree of signature encoding of T.



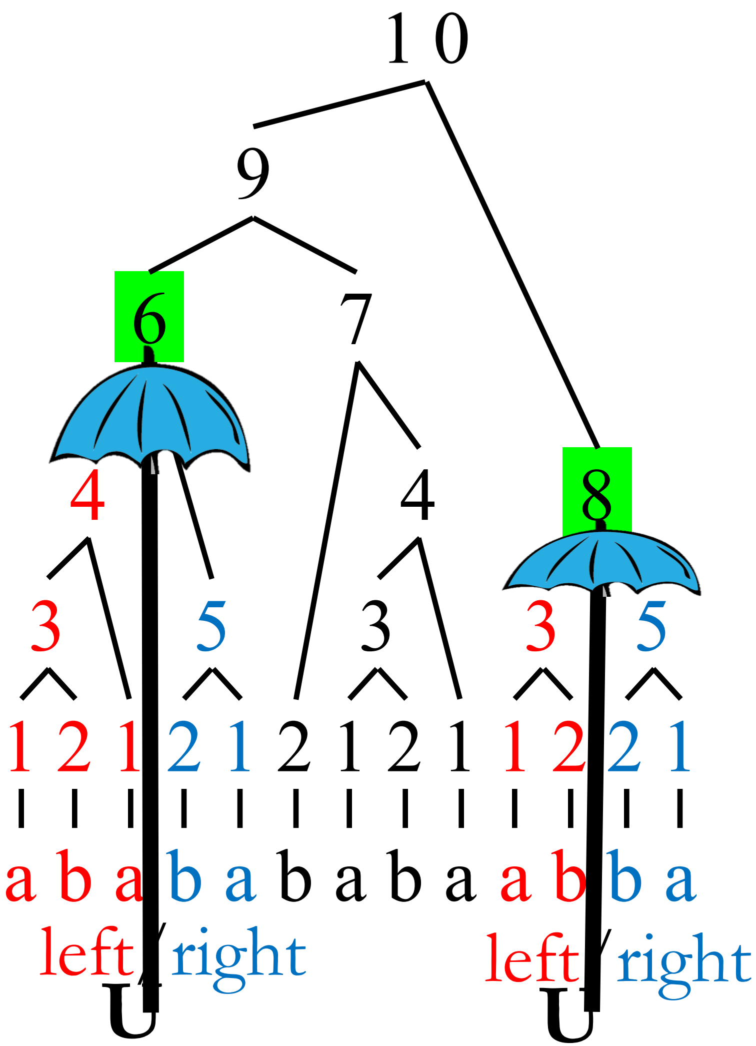


Oh, it’s raining.



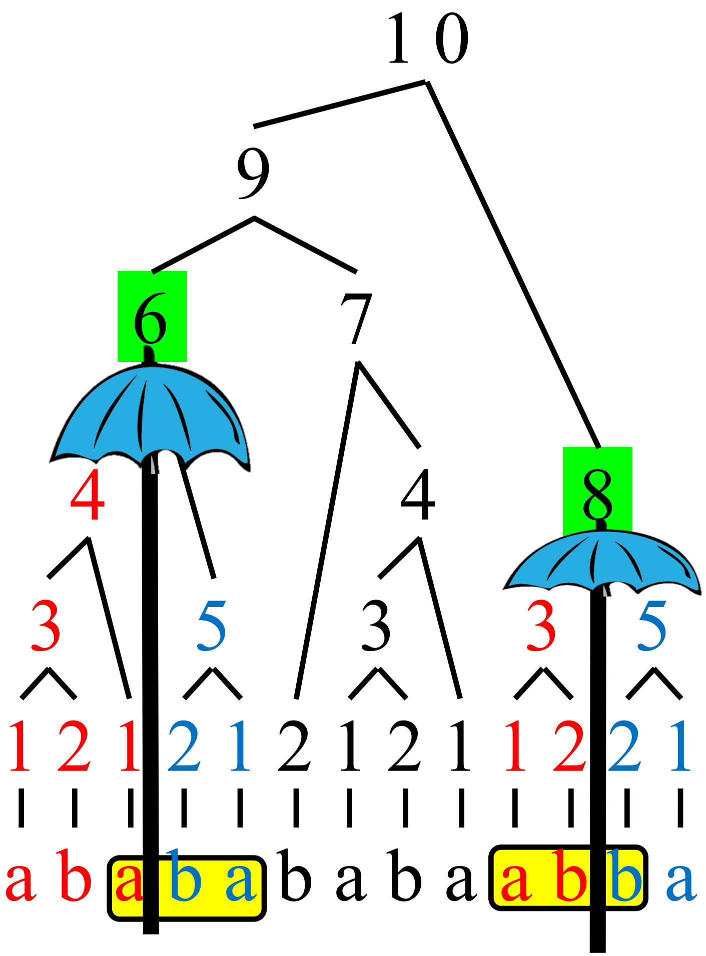
We need umbrellas. A signature has its umbrella.

Basic idea of pattern search[e.g., Claude+ '08]



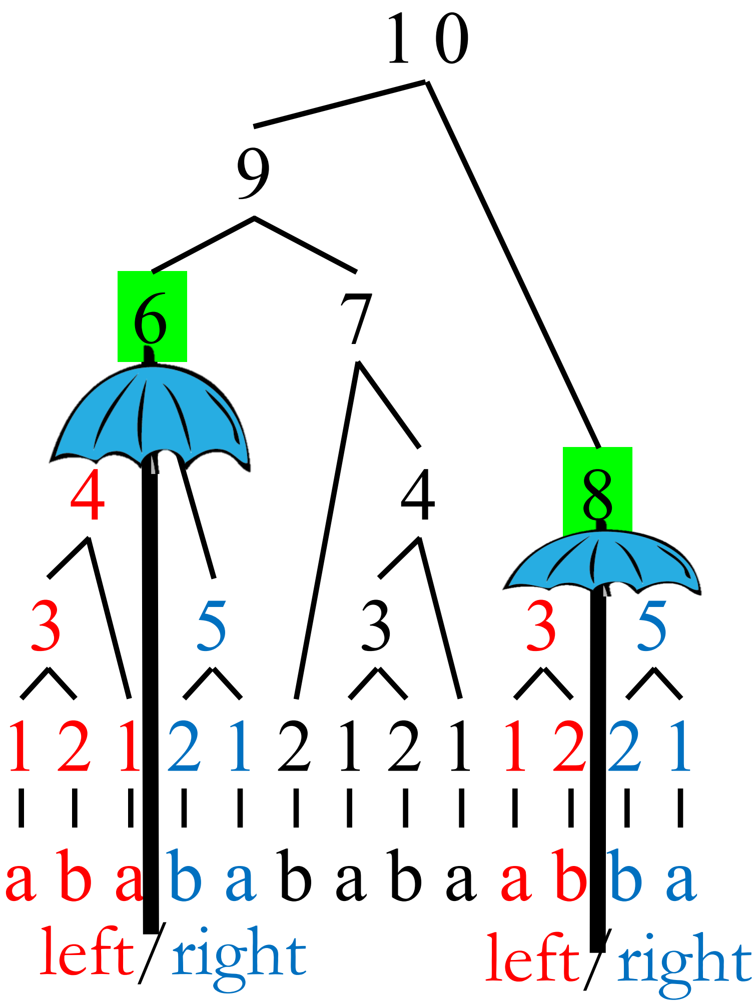
A signature derives left and right strings.

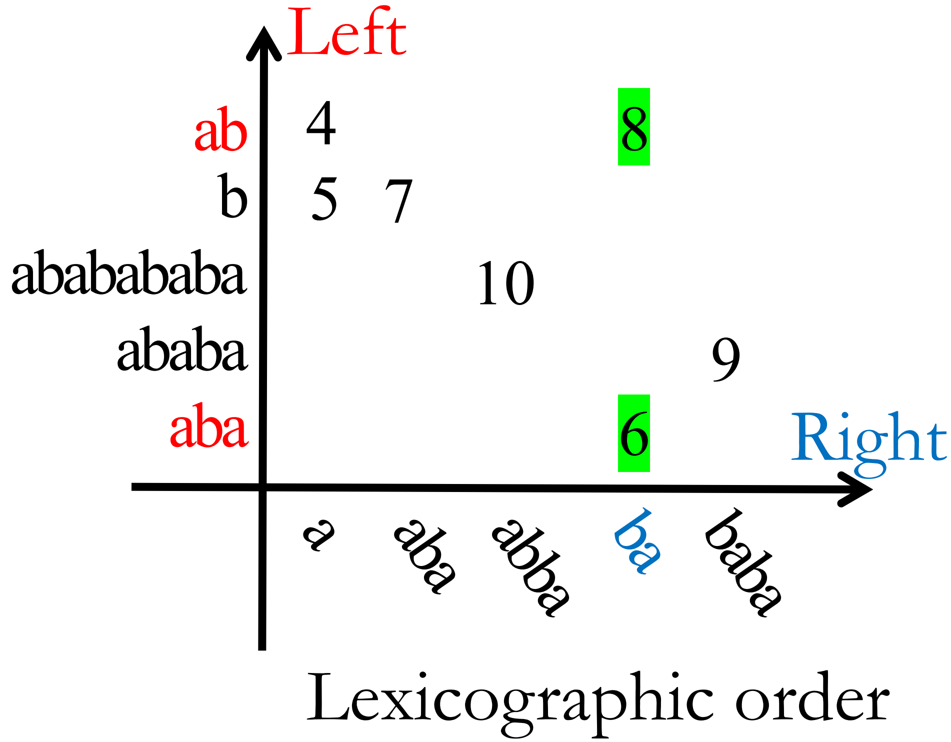
Basic idea of pattern search[e.g., Claude+ '08]



A substring of length at least 2 is divided by left and right strings of a signature.

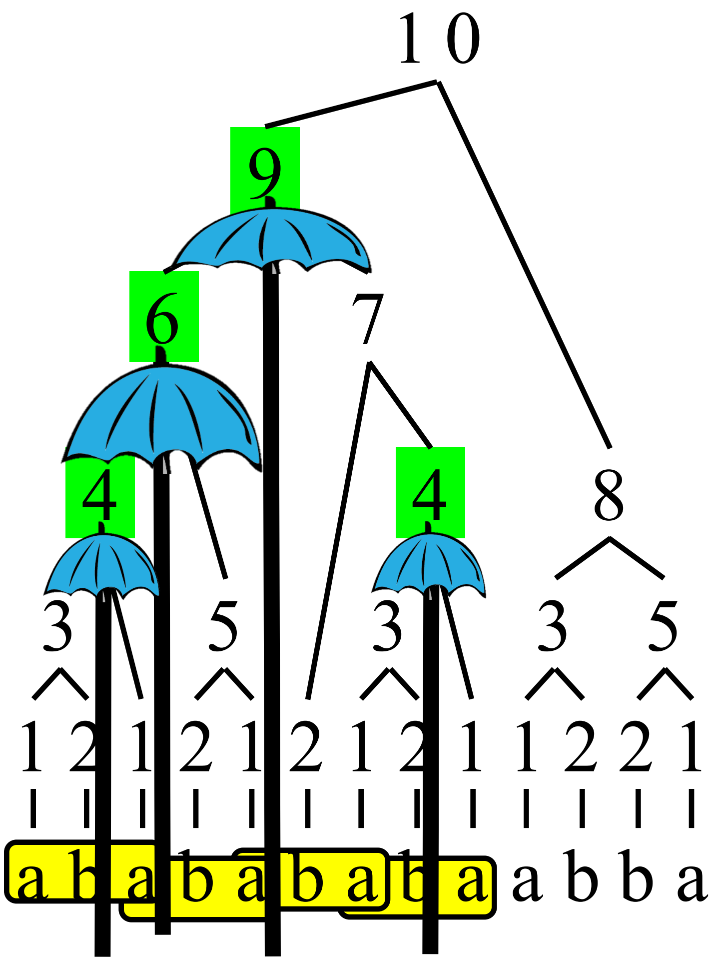
Basic idea of pattern search[e.g., Claude+ '08]

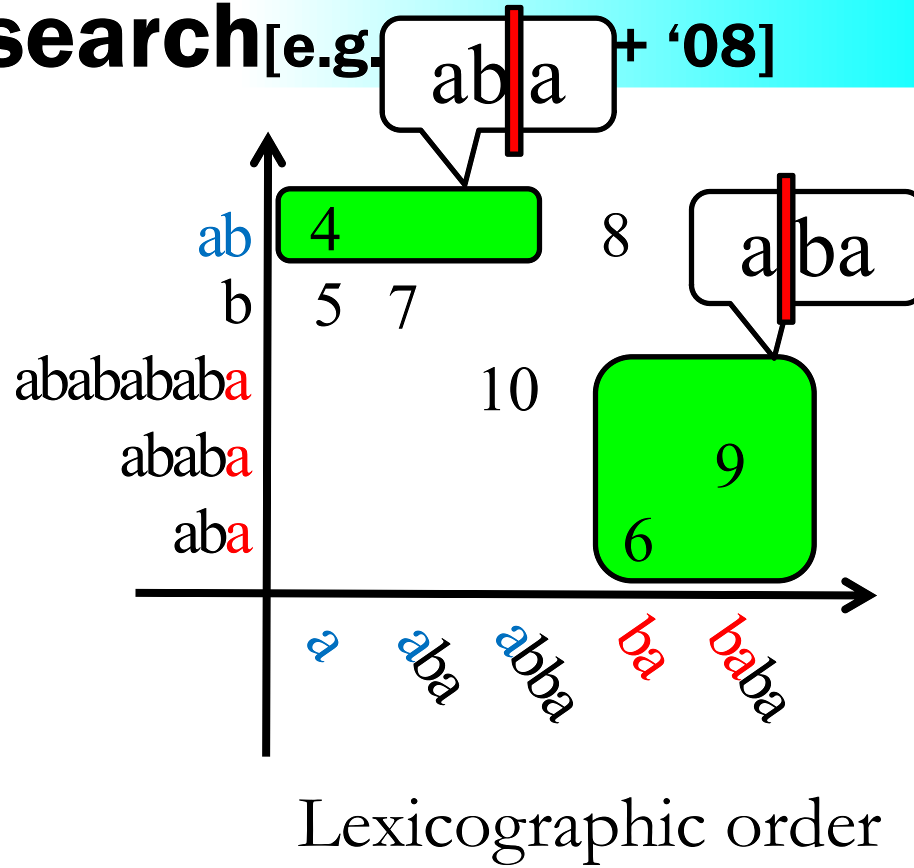




We can arrange signatures on 2-Dimensional plane by left and strings.

Basic idea of pattern search[e.g. + '08]





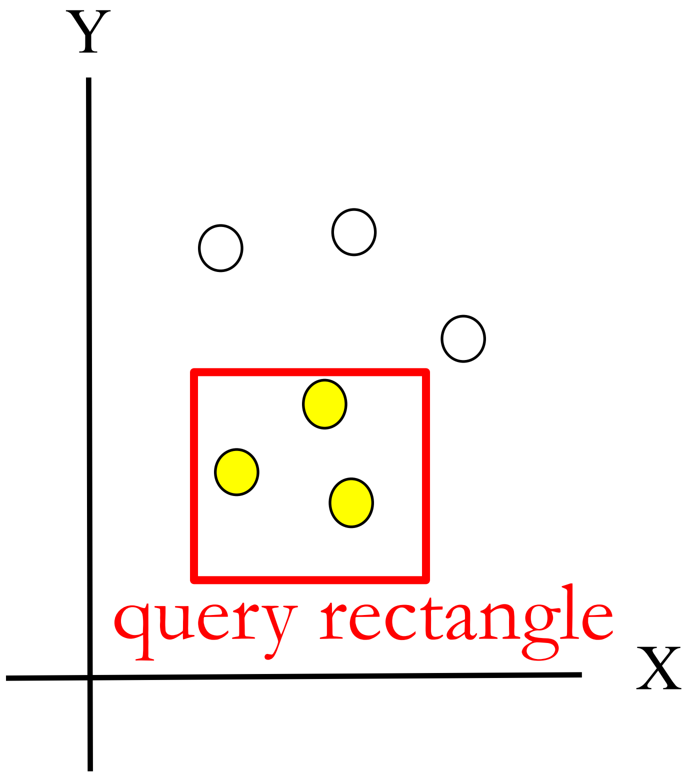
We can compute all signatures dividing a substring P by $|P|$ - 1 range queries on the 2D plane

Dynamic 2D Range Reporting(2DRR)[Blelloch, '11]

Space	$O(n \log n)$ bits
Insert/Delete	amortized $O(\log n)$ time
Range report	$O(\log n + k \frac{\log n}{\log \log n})$ time

n : the number of points

k : the number of output points



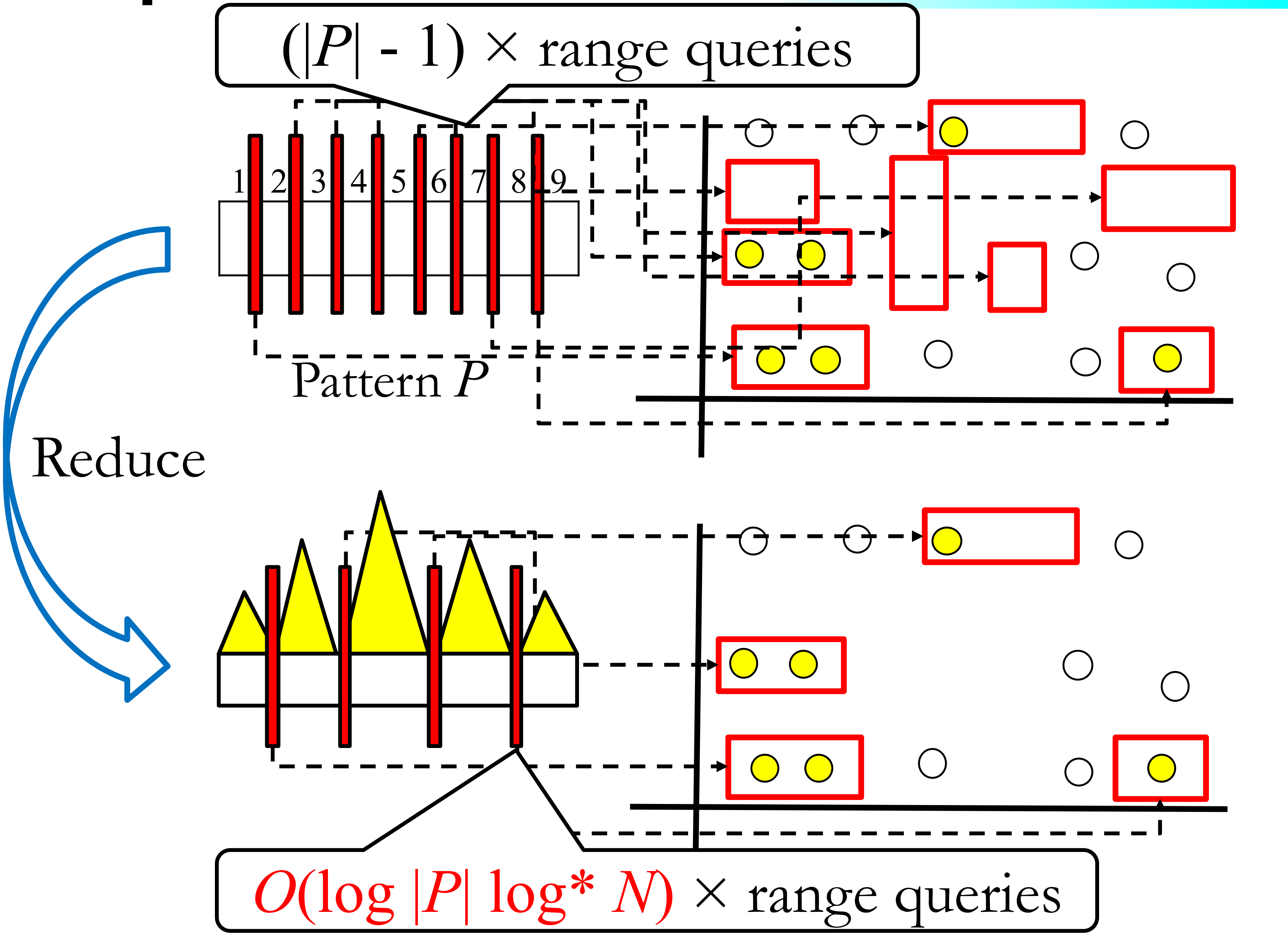
Our dynamic index uses this data structure.

Table of contents

1. Our contributions
2. Preliminaries
3. Our dynamic index
   - Basic idea of pattern search
   - Faster pattern search[main contribution]
   - Data structures & Update

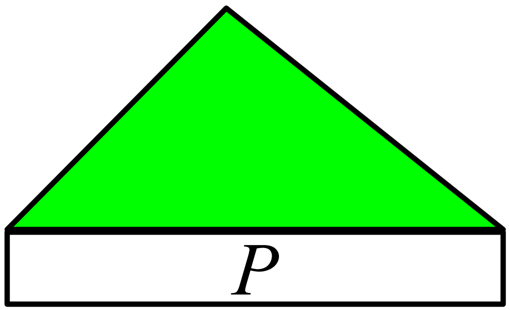
We can quickly find patterns in a given text using signature encoding.

Faster pattern search

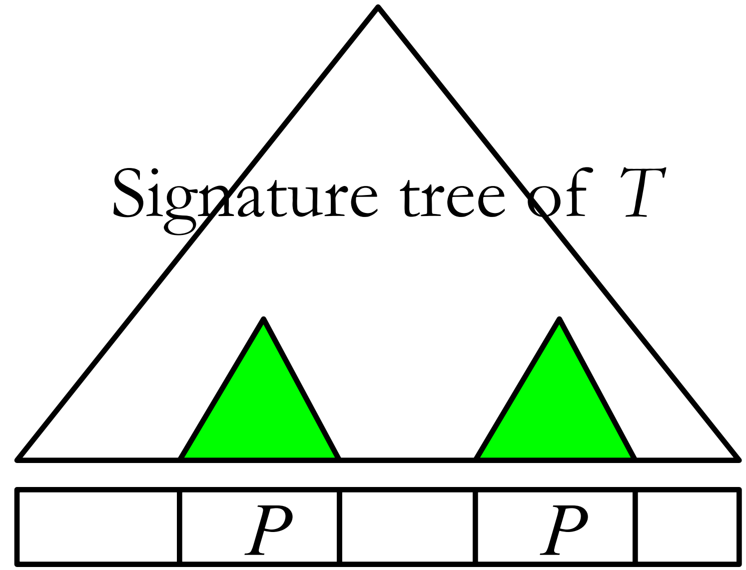


Faster pattern searching idea(1/6)

Common sequence of P



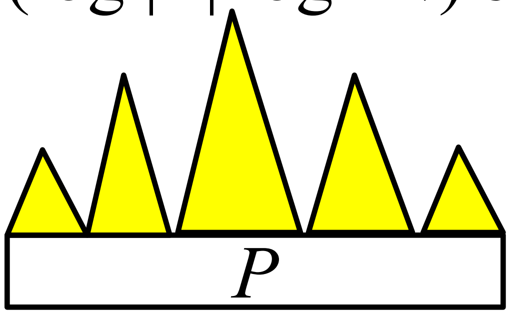
Pattern P



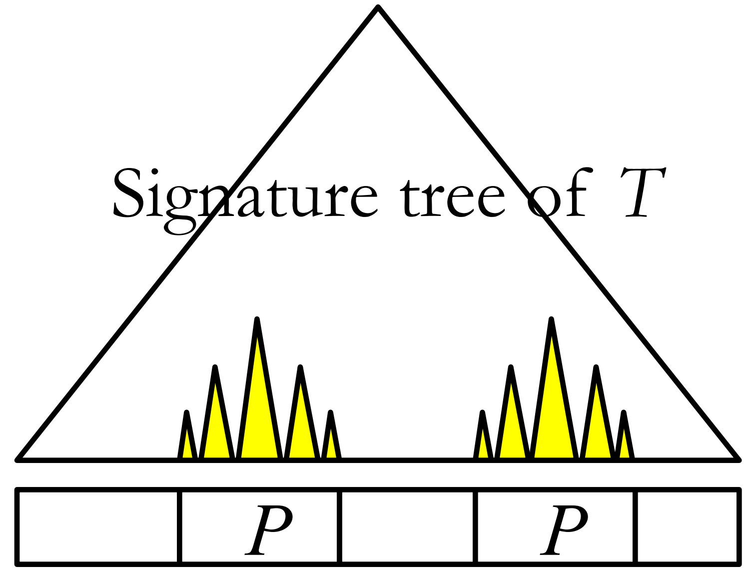
The common sequence of P occurs on every occurrence of P in T.

Faster pattern searching idea(2/6)

Common sequence of P
($O(\log |P| \log^* N)$ signatures)

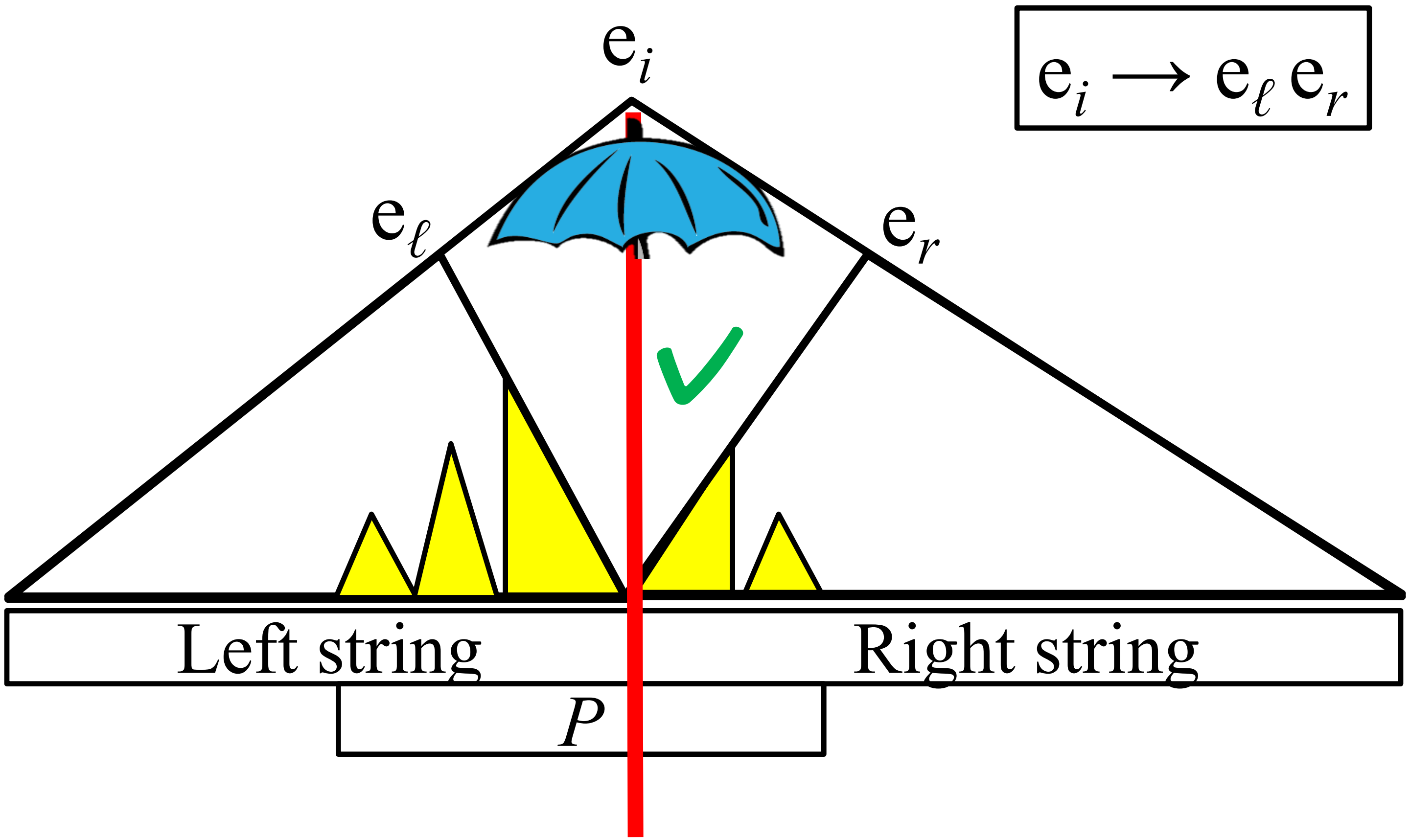


Pattern P



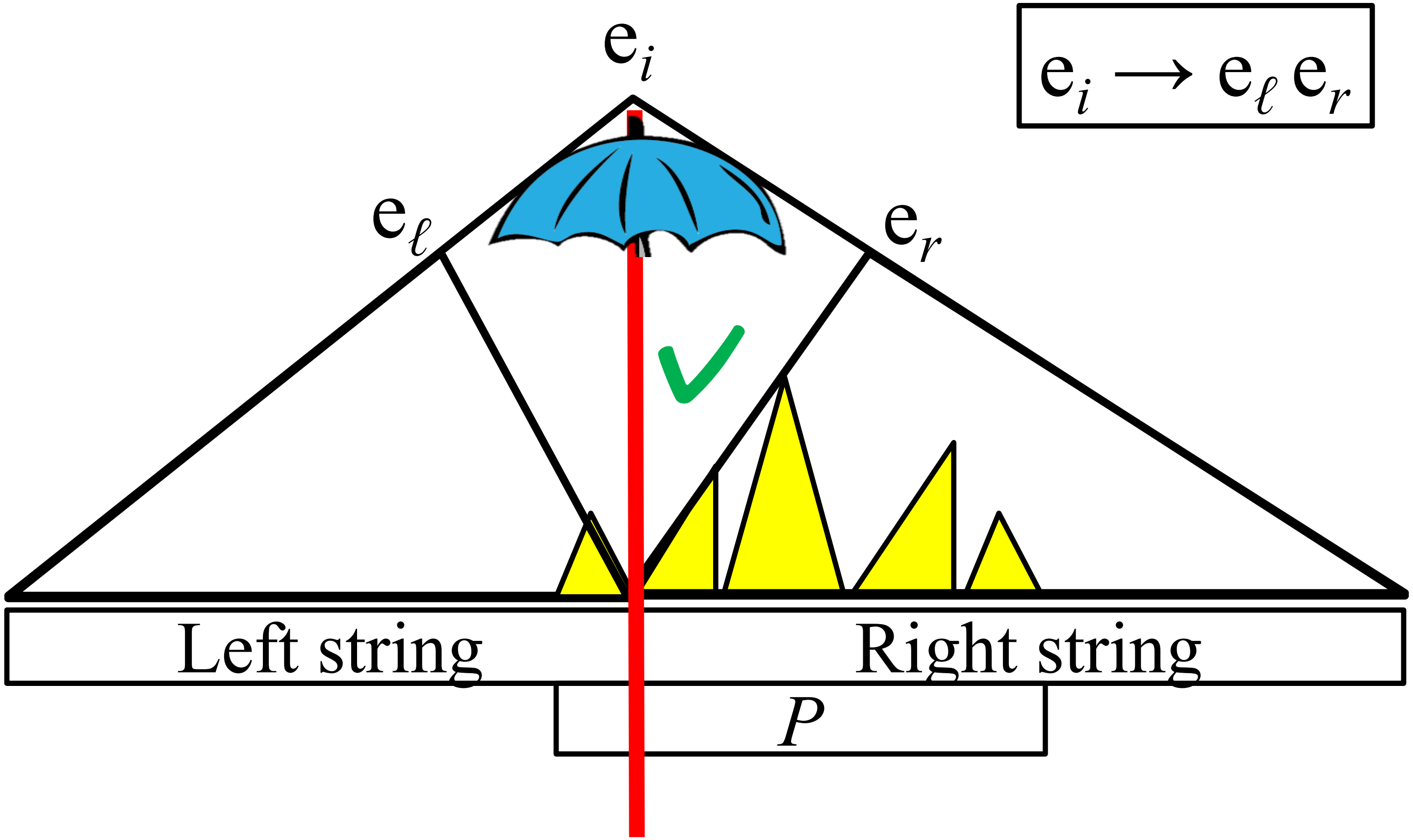
The common sequence of P consists of
$O(\log |P| \log^* N)$ signatures.

Faster pattern searching idea(3/6)



Each of $O(\log |P| \log^* N)$ signatures occurs in the left or right string of a signature.

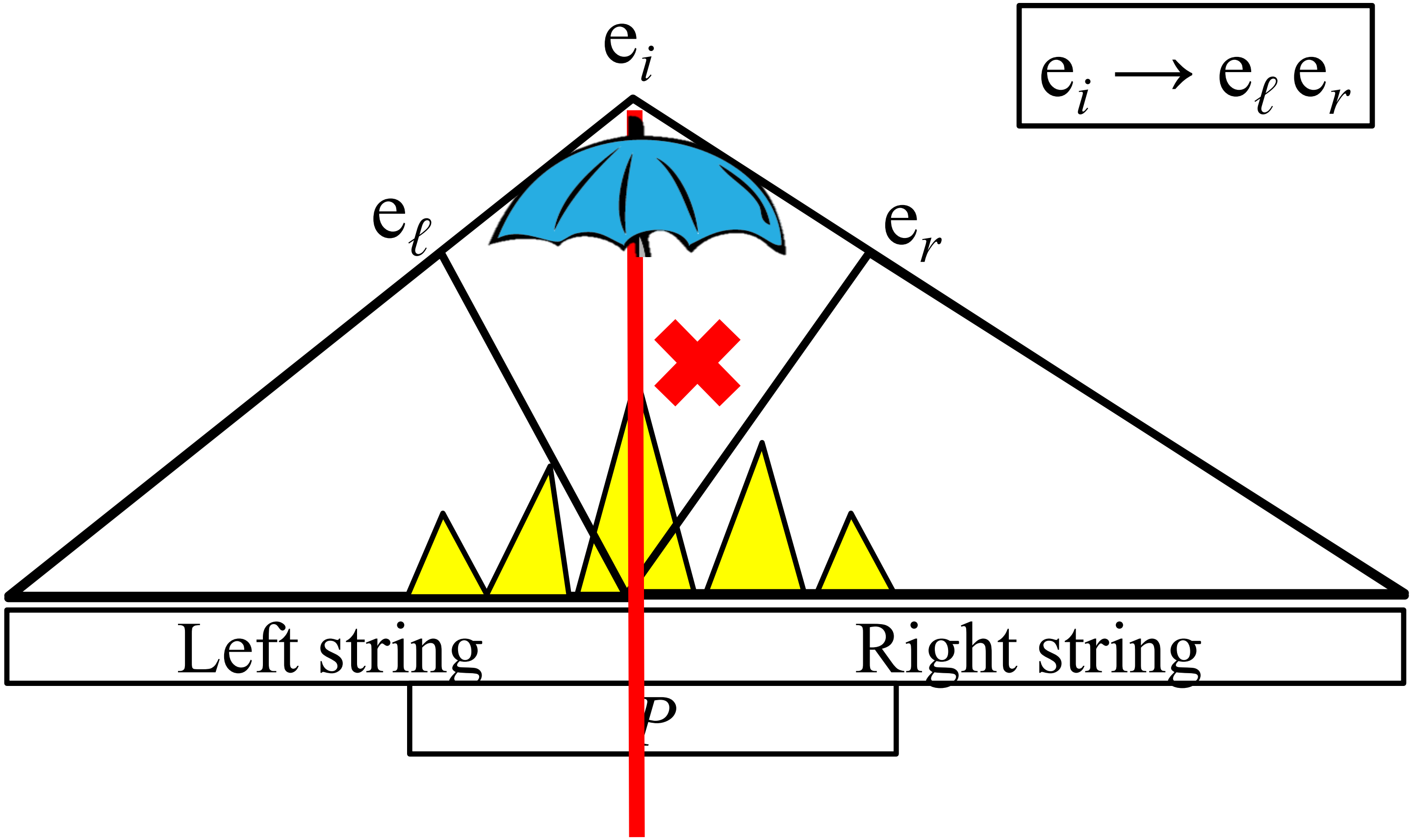
Faster pattern searching idea(4/6)



Each of $O(\log |P| \log^* N)$ signatures occurs in the left or right string of a signature.

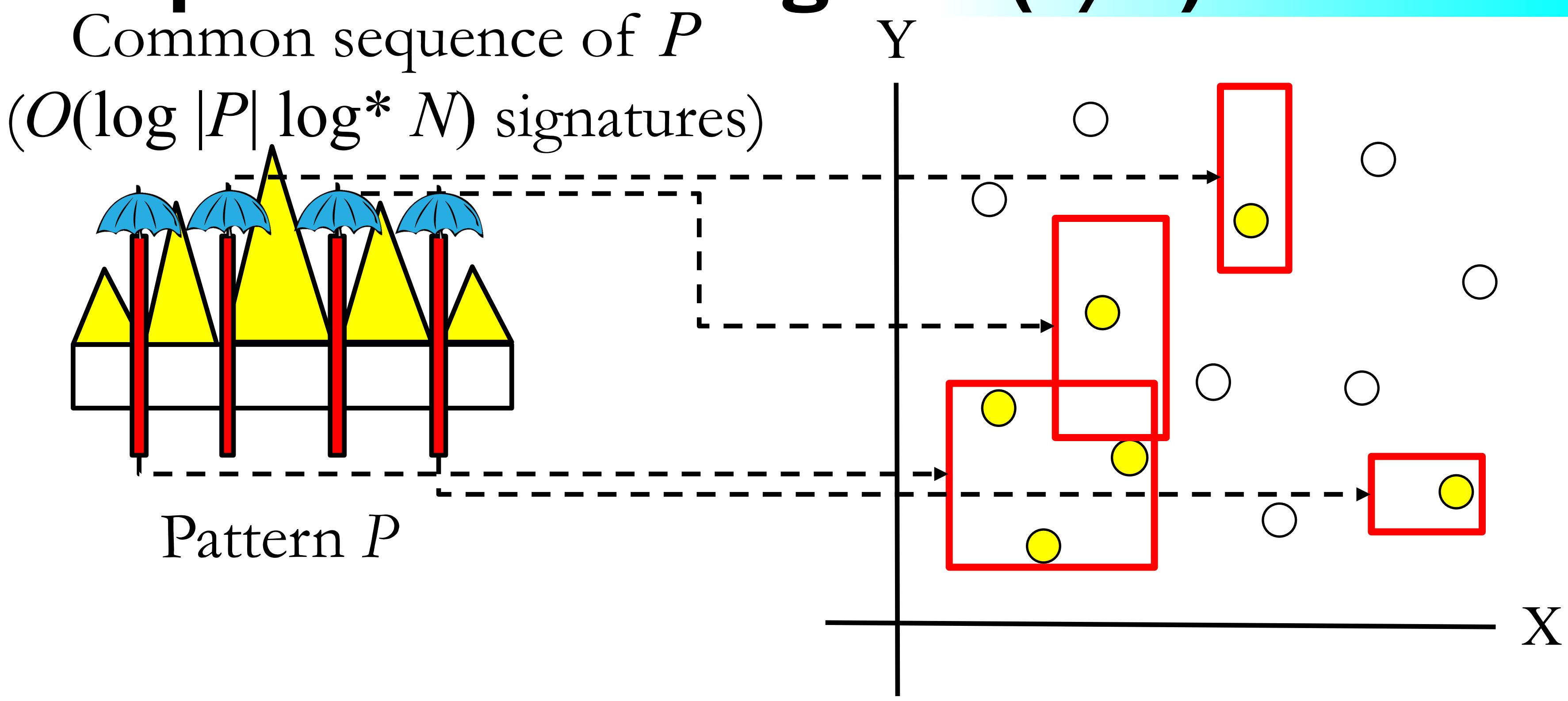
Faster pattern searching idea(5/6)

$e_i \rightarrow e_\ell \, e_r$



Each of $O(\log |P| \log^* N)$ signatures occurs in the left or right string of a signature.

Faster pattern searching idea(6/6)



Hence $O(\log |P| \log^* N) \times$ range queries

Table of contents

1. Our contributions
2. Preliminaries
3. Our dynamic index
   - Basic idea of pattern search
   - Faster pattern search
   - Data structures & Update

Data structures of our dynamic index

1 → a
2 → b
3 → 1, 2
4 → 3, 1
5 → 2, 1
6 → 4, 5
7 → 2, 4
8 → 3, 5
9 → 6, 7
10 → 9, 8

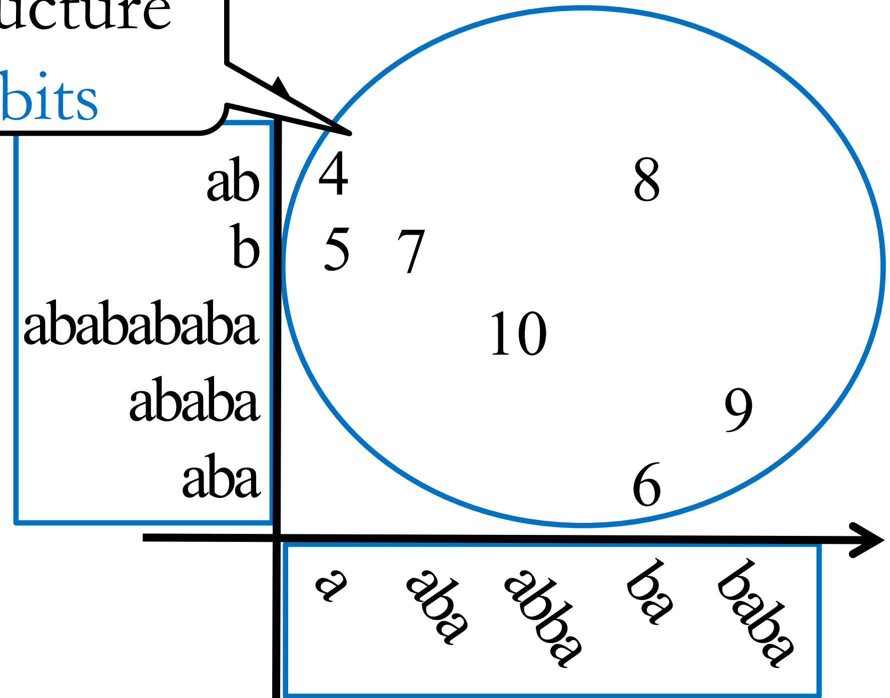
w

Signature list

$O(w \log N)$ bits

2DRR data structure

$O(w \log w)$ bits



Sorted

left/right strings

$O(wN \log \sigma)$ bits

Total $O(wN \log \sigma)$ bits?

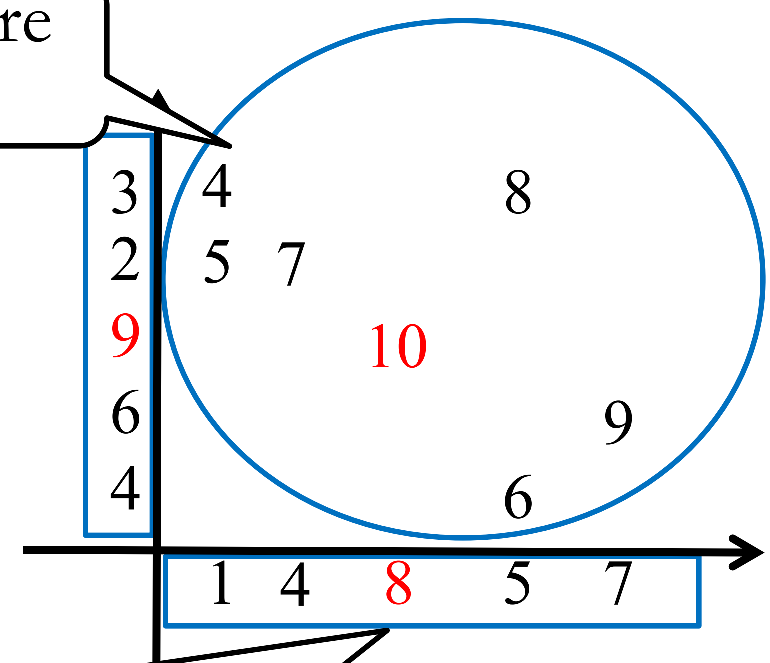
Data structures of our dynamic index

$1 \rightarrow a$
$2 \rightarrow b$
$3 \rightarrow 1, 2$
$4 \rightarrow 3, 1$
$5 \rightarrow 2, 1$
$6 \rightarrow 4, 5$
$7 \rightarrow 2, 4$
$8 \rightarrow 3, 5$
$9 \rightarrow 6, 7$
$10 \rightarrow 9, 8$

w

Signature list
$O(w \log N)$ bits

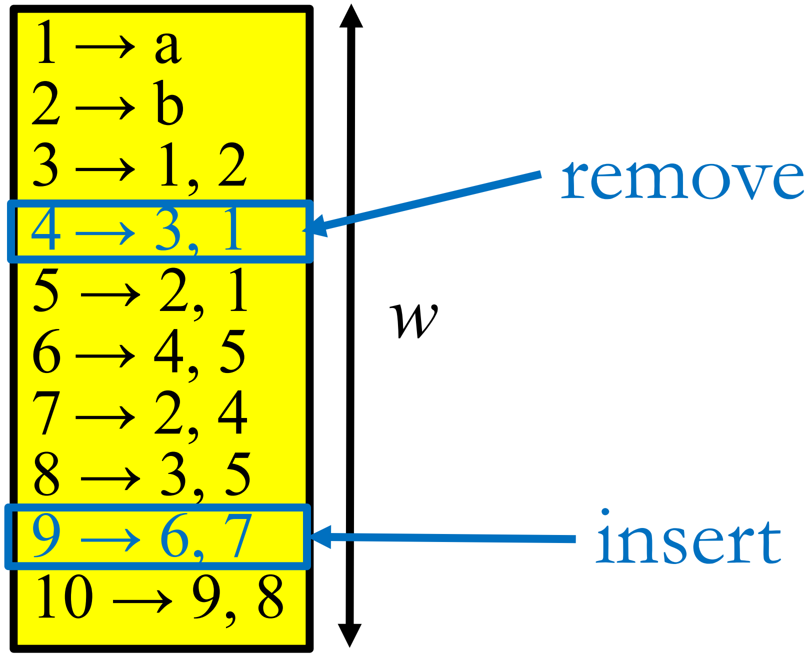
2DRR data structure
$O(w \log w)$ bits



Sorted
left/right strings
$O(wN \log \sigma) \rightarrow O(w \log w)$ bits

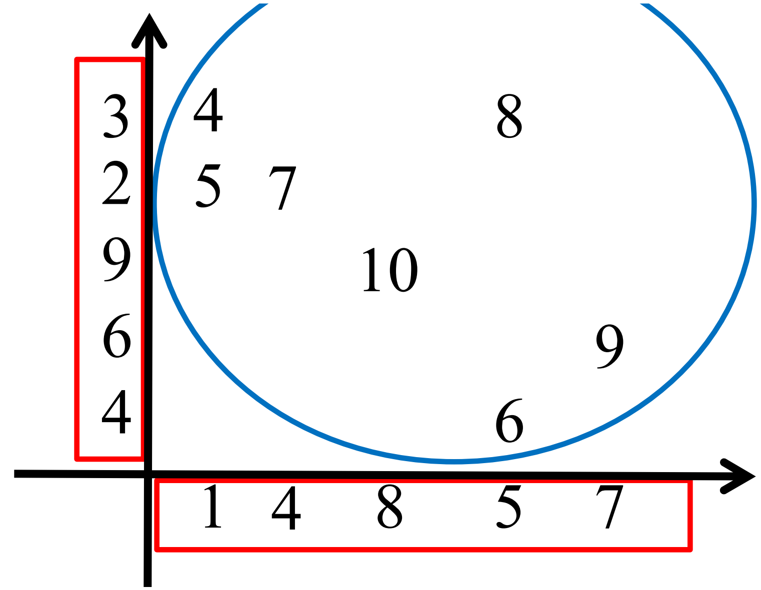
Total $O(w \log N)$ bits

Update(1/3)



Signature list

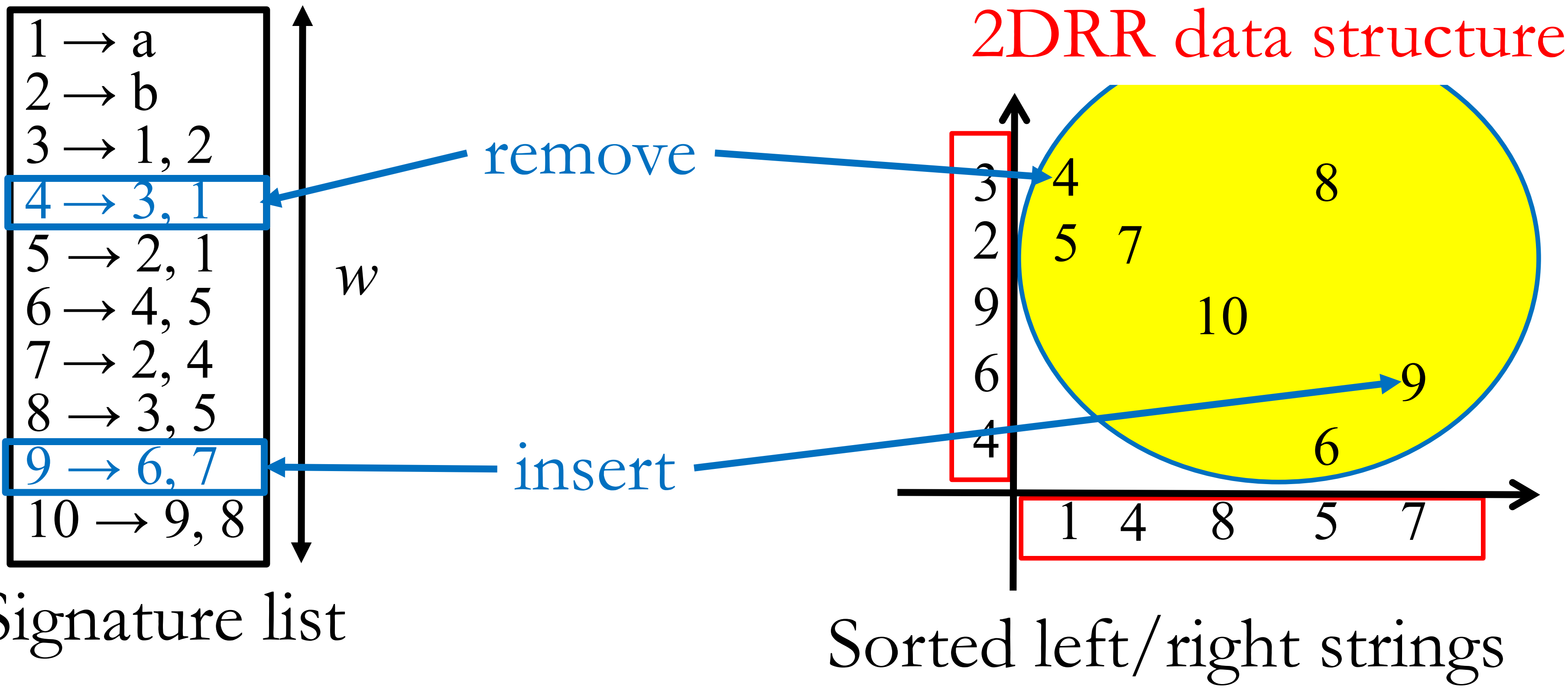
2DRR data structure



Sorted left/right strings

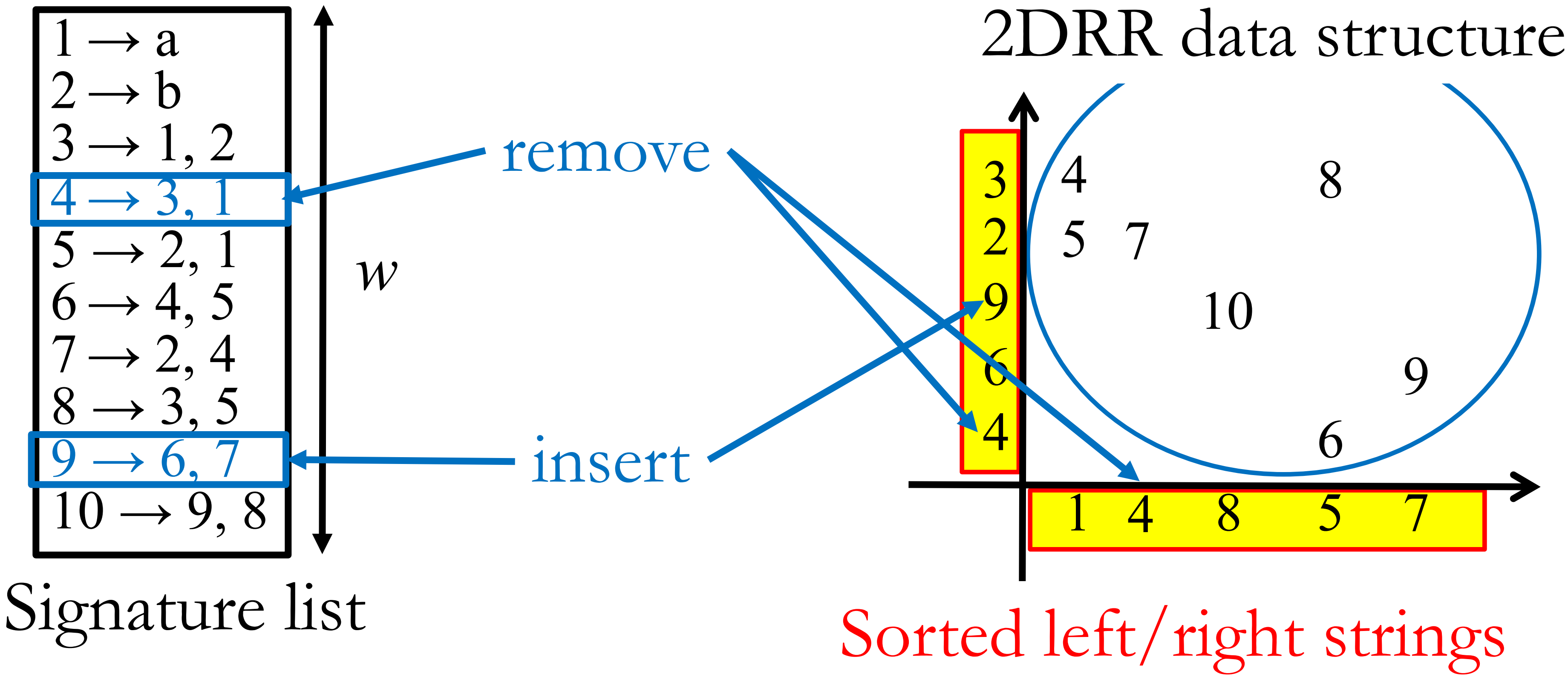
We can efficiently update the signature encoding of T by using Alstrup et al.'s technique.

Update(2/3)



We can efficiently update the 2DRR data structure proposed by Blleloch.

Update(3/3)



We can efficiently update the sorted left/right strings by using binary search and LCE queries.

Conclusion

This work	
Space	$O(w \log N) = O(\min\{z \log N \log^* N, N\} \log N)$ bits
Update	amortized $O((\lvert Y\rvert + \log N \log^* N) \log w \log N \log^* N)$ time
Find(P)	$O(c\lvert P\rvert + \log N \log w \log \lvert P\rvert (\log^* N)^2 + occ \log N)$ time

- N : length of a text T
- c : time for predecessor queries
- z : size of LZ77 factorization of T, $z = O(N/\log_\sigma N)$ [e.g. Kärkkäinen]
- σ : alphabet size
- $\lvert Y\rvert$: length of an inserted string or deleted substring
- occ : the number of occurrences of a given pattern P in T
- $w = O(\min\{z \log N \log^* N, N\})$

Thank you!

Conclusion

Hon et al, '04	
Space	$O((N\mathrm{H}_0+N)/\varepsilon)$ bits
Update	$O((\lvert Y\rvert + \sqrt{N}) \log^{2+\varepsilon} N)$ time
Find(P)	$O(\lvert P\rvert \log^2 N (\log^{\varepsilon} N + \log \sigma) + occ \log^{1+\varepsilon} N)$ time

This work	
Space	$O(w \log N) = O(\min\{z \log N \log^* N, N\} \log N)$ bits
Update	amortized $O((\lvert Y\rvert + \log N \log^* N) \log w \log N \log^* N)$ time
Find(P)	$O(c\lvert \mathrm{P}\rvert + \log N \log w \log \lvert \mathrm{P}\rvert (\log^* N)^2 + occ \log N)$ time

- N : length of a text T
- c : time for predecessor queries
- z : size of LZ77 factorization of T, $z = O(N/\log_\sigma N)$ [e.g. Kärkkäinen]
- σ : alphabet size
- $\lvert Y\rvert$: length of an inserted string or deleted substring
- occ : the number of occurrences of a given pattern P in T
- $w = O(\min\{z \log N \log^* N, N\})$
- H_0 : the zeroth order empirical entropy of T, $H_0 \leq \log \sigma$