An improvement of the Franek-Jennings-Smyth pattern matching algorithm

Satoshi Kobayashi, Diptarama Hendrian, Ryo Yoshinaka, Ayumi Shinohara

Graduate School of Information Sciences, Tohoku University, Japan

Exact pattern matching problem

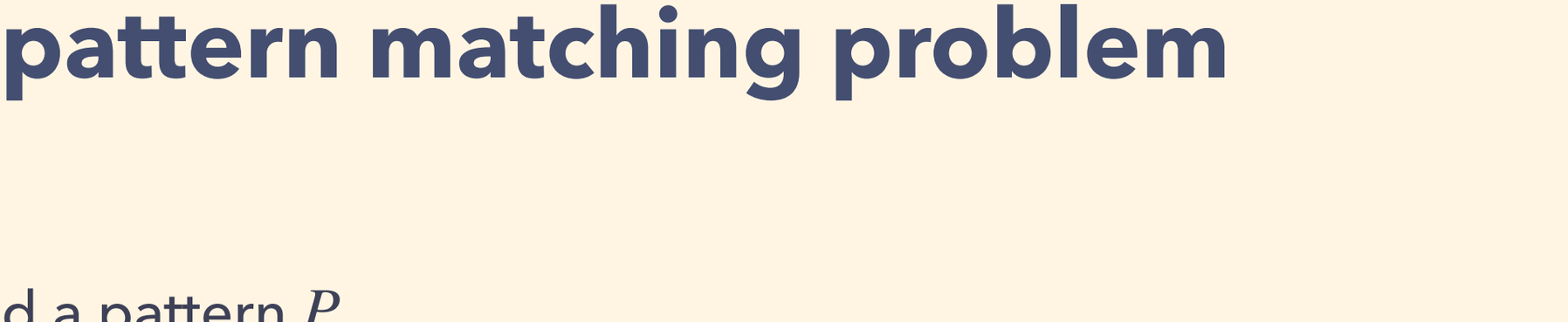
Input

A text T and a pattern P

Output

All positions i in T such that $T[i : i + |P| - 1] = P$

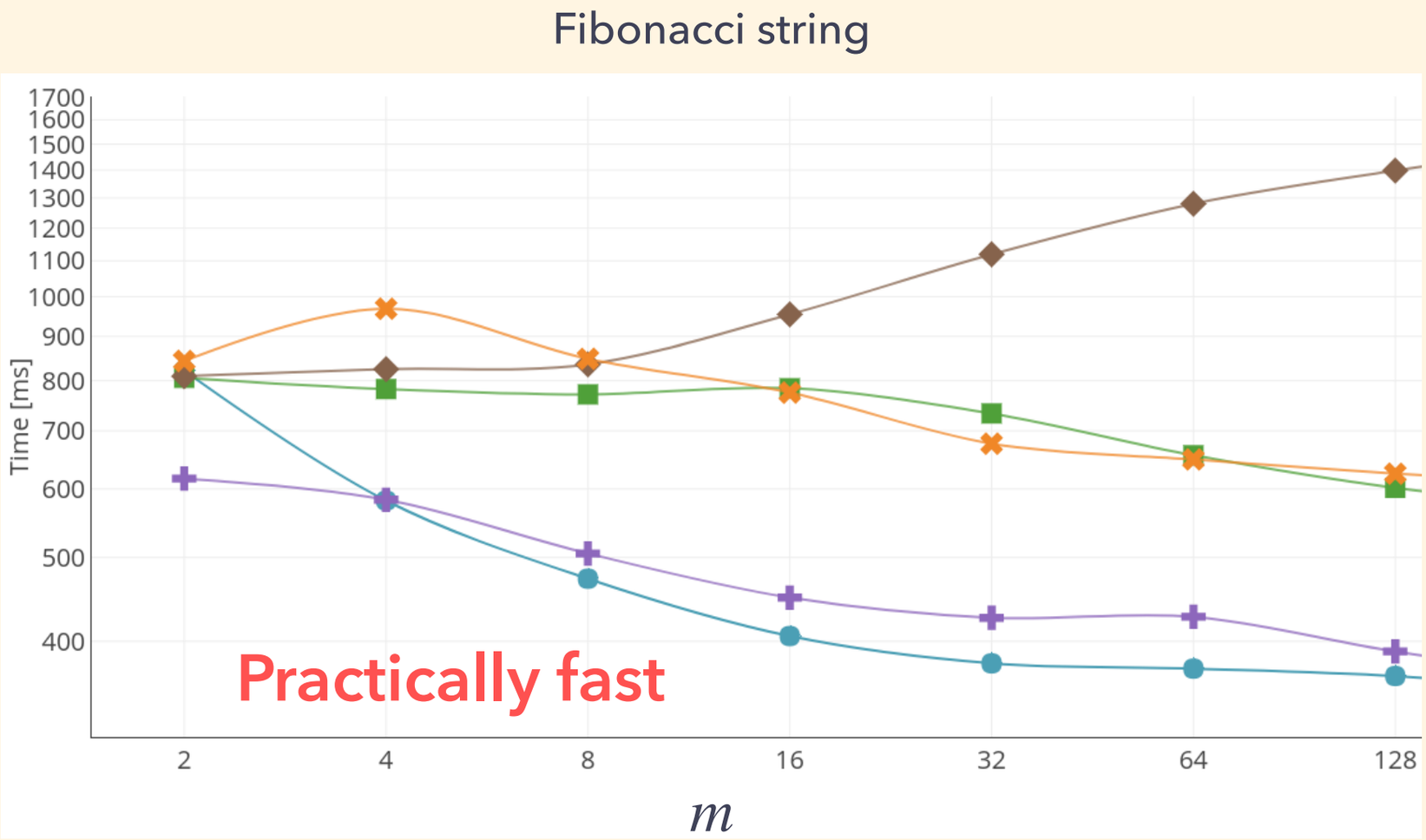
Example



Output : 6, 9

Our contributions

- A new exact pattern matching algorithm
  based on the Franek-Jennings-Smyth (FJS) algorithm [Franek+, 2005]
- Combine the idea of the Quite-Naive algorithm [Cantone & Faro, 2004]

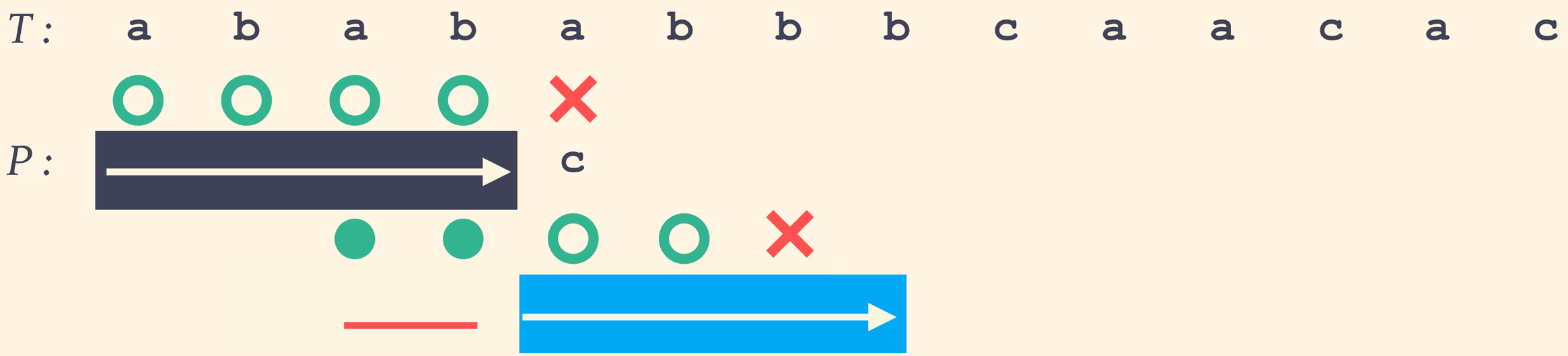


Algorithm	Preprocess	Search
Quick-Search [Sunday, 1990]	$O(m+\sigma)$	$O(nm)$
LWFR [Cantone+, 2019]	$O(m^2)$	$O(n)$
KMP [Knuth+, 1977]	$O(m)$	$O(n)$
FJS [Franek+, 2005]	$O(m+\sigma)$	$O(n)$
Ours	$O(m+\sigma)$	$O(n)$

Linear time

$n = |T|$: text length $m = |P|$: pattern length $\sigma = |\Sigma|$: alphabet size

KMP algorithm [Knuth+, 1977]



If a mismatch occurs at j, we shift P by $KMP\_Shift[j]$

$$Strong\_Board_P[j] = \begin{cases} -1 & \text{if } j = 1 \\ Strong\_Board_P[k] & \text{if } j \leq m \text{ and } P[k+1] = P[j] \\ k & \text{otherwise} \end{cases}$$

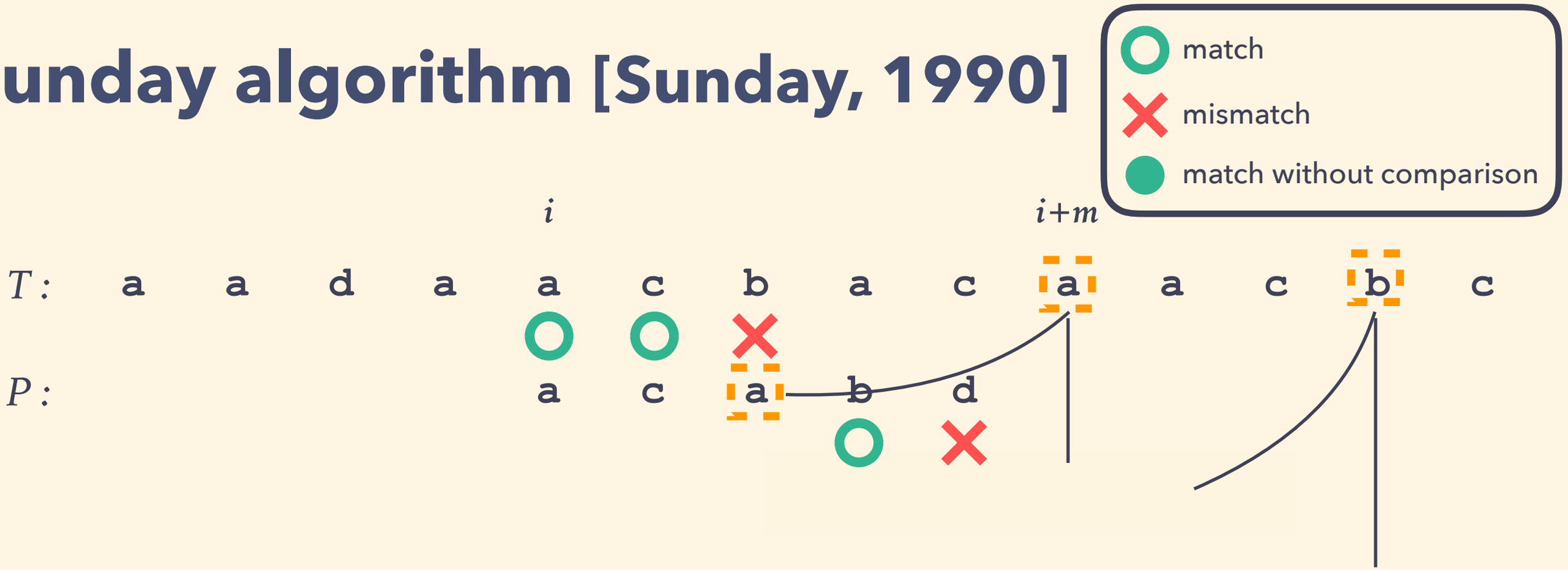
k : length of the longest proper border of $P[1 : j-1]$

$$KMP\_Shift[j] = j - Strong\_Board_P[j] - 1$$

j	1	2	3	4	5	6
P	a	b	a	b	c	
KMP_Shift	1	1	3	3	2	5

Preprocessing time : $O(m)$ Searching time : $O(n)$

Sunday algorithm [Sunday, 1990]



If a mismatch occurs, we shift P by $Sunday\_Shift[T[i+m]]$

$$Sunday\_Shift[c] = m + 1 - \max(\{\, j \mid P[j] = c \,\} \cup \{0\}) \text{ for } c \in \Sigma$$

c	a	b	c	d
$Sunday\_Shift$	3	2	4	1

	5	4	3	2	1
P :	a	c	a	b	d

Preprocessing time : $O(m + \sigma)$ **Searching time :** $O(nm)$

FJS algorithm [Franek+, 2005]

Practically fast

$O(n)$ time

Sunday + KMP algorithm

match

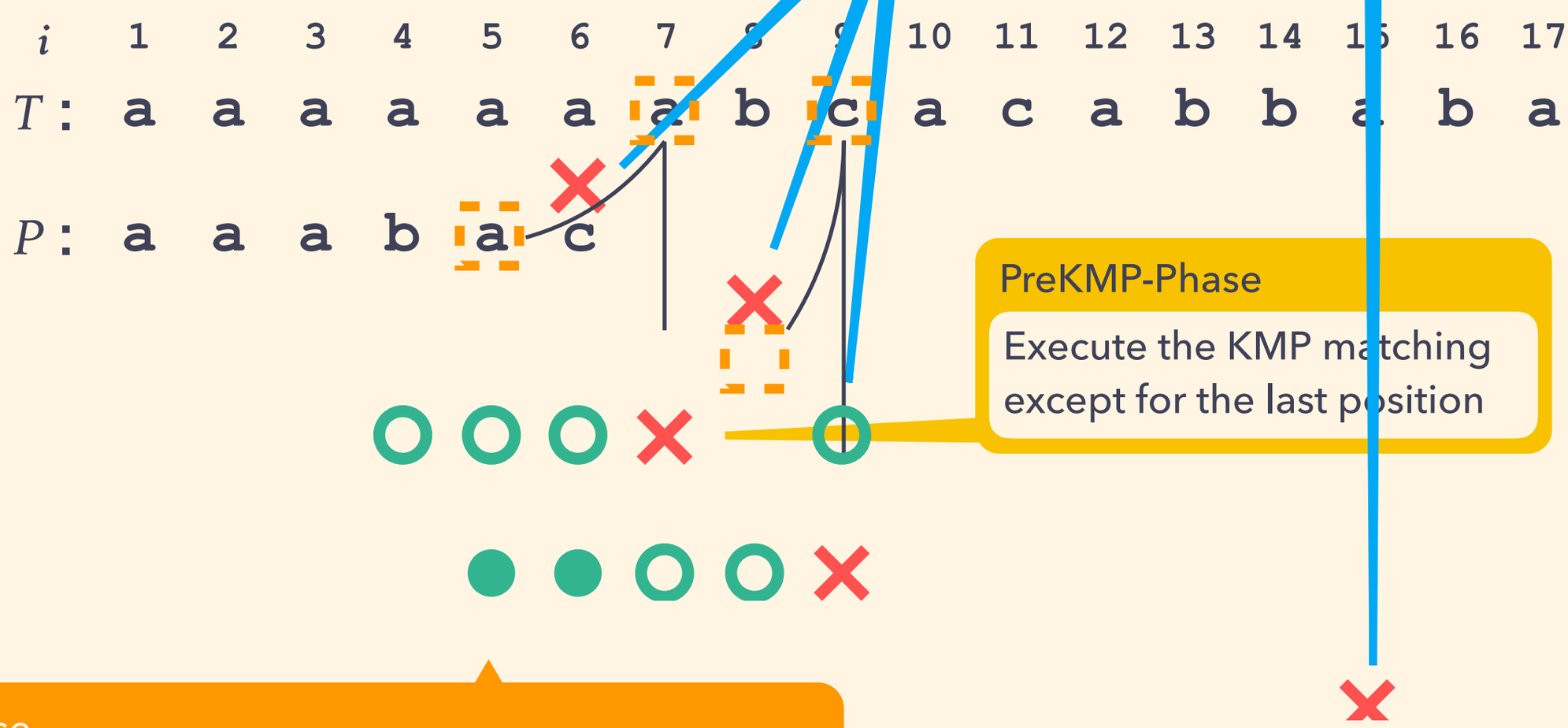
mismatch

match without comparison

Sunday-Phase

Shift P by $Sunday\_Shift[T[i+m]]$

until $P[m] = T[i+m-1]$



PreKMP-Phase

Execute the KMP matching except for the last position

KMP-Phase

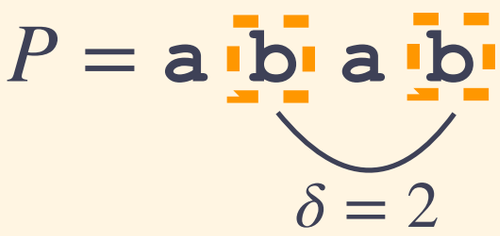
If some non-empty prefixes of the pattern and the text substring match, execute the KMP matching

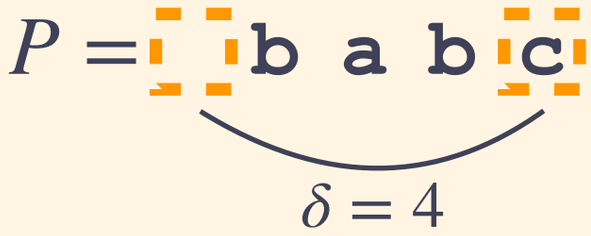
Preprocessing time : $O(m+\sigma)$ Searching time : $O(n)$

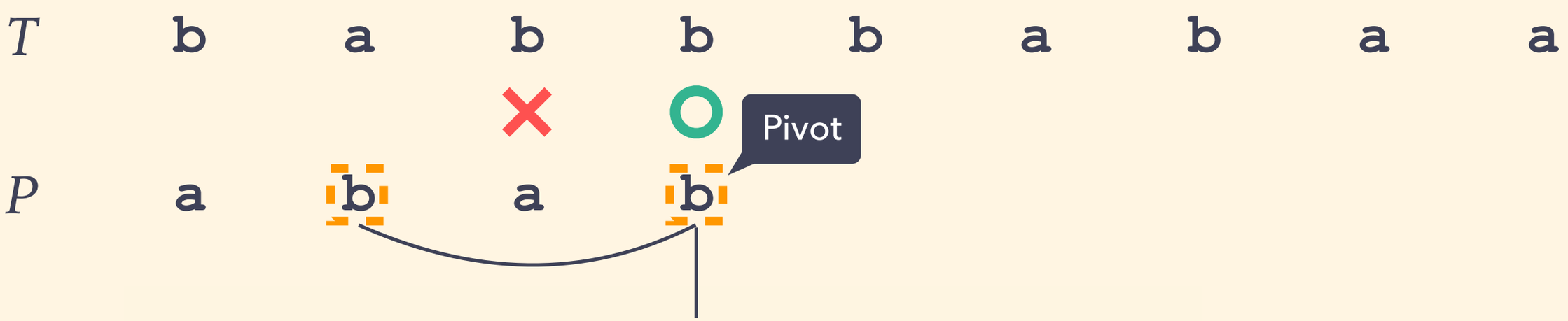
Quite-Naive algorithm [Cantone & Faro, 2004] (in part)

δ is the distance between $P[m]$ and the same rightmost character in $P[1 : m-1]$

$$\delta = \min(\{\, j \mid P[m-j] = P[m],\ 1 \le j < i \,\} \cup \{m\})$$







Proposed algorithm

Proposed algorithm - Basic idea (1 / 2)

Practically fast | $O(n)$ time | Use δ

Sunday + **KMP** + **Quite-Naive** algorithm

i	1	2	3	4	5	6	7	8	9	10	11	12	13
T	a	a	a	b	c	a	b	a	b	b	b	c	a
P	a	b	a	b	c	a							

$\delta = 3$

j	1	2	3	4	5	6	7
P	a	b	a	b	c	a	
KMP_Shift	1	1	3	3	2	6	5

Shift by $\max\{KMP\_Shift[j], \delta\}$

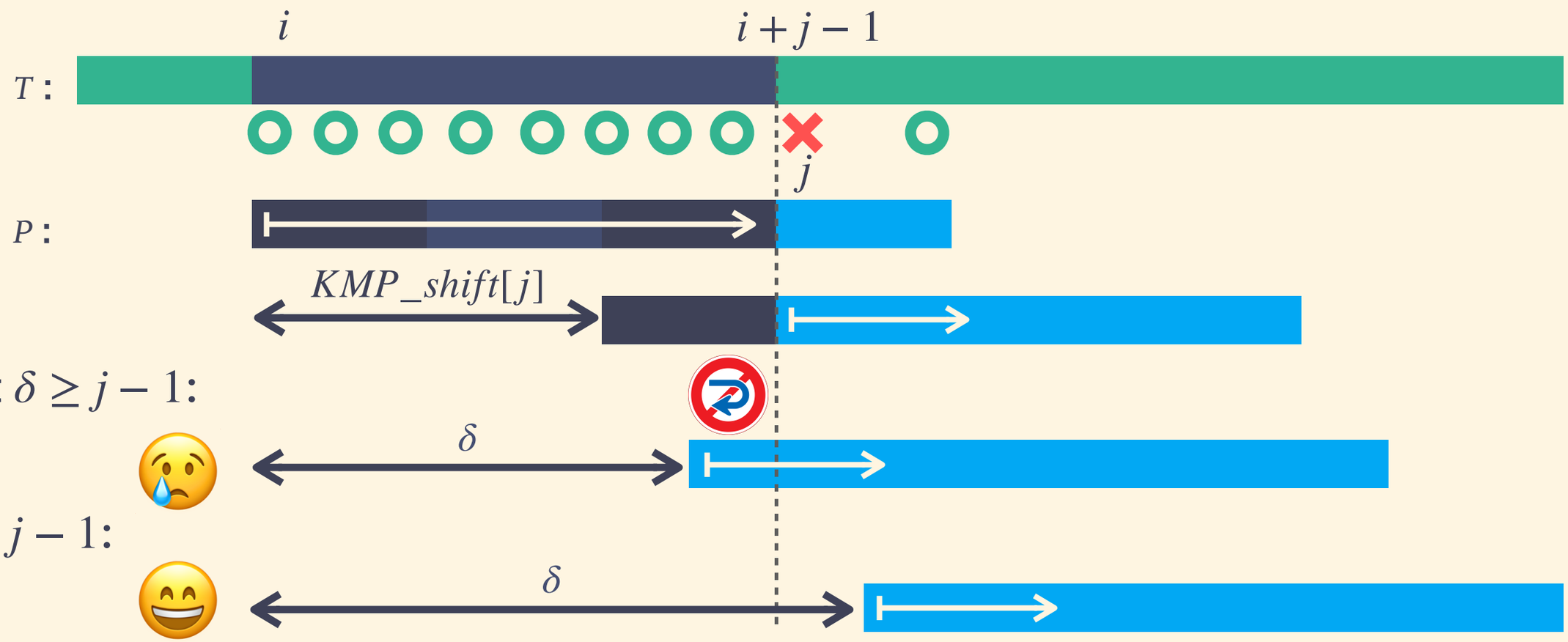
Proposed algorithm - Basic idea (2 / 2)

We use MAX_Shift instead of KMP_Shift in the PreKMP-Phase

To guarantee $O(n)$ time

$$MAX\_Shift[j] = \begin{cases} \delta & \text{if } \delta \geq KMP\_Shift[j] \text{ and } \delta \geq j-1 \\ KMP\_Shift[j] & \text{othewise} \end{cases}$$

In the PreKMP-Phase



Proposed algorithm (Get a bigger shift)

We generalize δ to calculate the distance of characters **at all positions** in the pattern and **obtain the maximum value and its position**

$$\delta = \min(\{\, j \mid P[m-j] = P[m],\ 1 \le j < i \,\} \cup \{m\})$$

generalize

$$d[i] = \min(\{\, j \mid P[i-j] = P[i],\ 1 \le j < i \,\} \cup \{i\})$$

$$d[m] = \delta$$

We define the following two values

- **Maximal Distance**

$$md = \max_{1 \le j \le m} d[j]$$

- **Maximal Distance Position (pivot)**

$$mdp = \arg\max_{1 \le j \le m} d[j]$$

Example: $P =$ `acabbacb`

$mdp = 7$

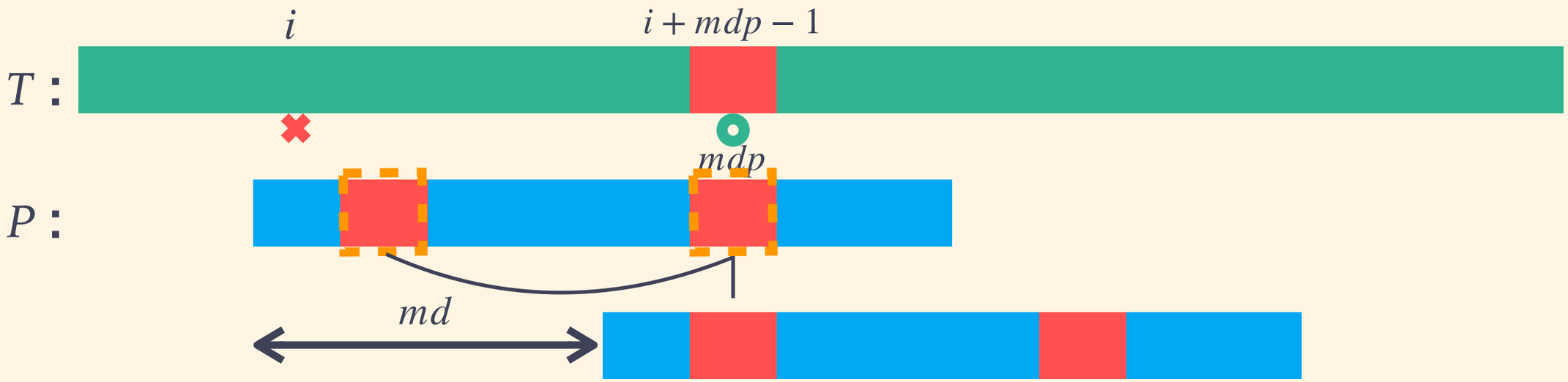
j	1	2	3	4	5	6	7	8
P	a	c	a	b	b	a	c	b
d	1	2	2	4	1	3	5	3

$md = 5$

表アニメ

Proposed algorithm (Modify MAX_Shift)

If a mismatch occurs when $P[mdp] = T[i + mdp - 1]$,
we can shift the pattern by md



We can use MAX_Shift instead of KMP_Shift only when $P[mdp] = T[i + mdp - 1]$

$$MAX\_Shift[j] = \begin{cases} \underline{md} & \text{if } \underline{md} \geq \max\{KMP\_Shift[j],\ j-1\} \\ KMP\_Shift[j] & \text{othewise} \end{cases}$$

j	1	2	3	4	5	6	7	8	9
$P[j]$	a	c	a	b	b	a	c	b	
$KMP\_Shift[j]$	1	1	3	2	4	6	6	5	8
$MAX\_Shift[j]$	5	5	5	5	5	6	6	5	8

$md = 5, mdp = 7$

Proposed algorithm

Practically fast | $O(n)$ time | Generalized δ

Sunday + KMP + Generalized Quite-Naive

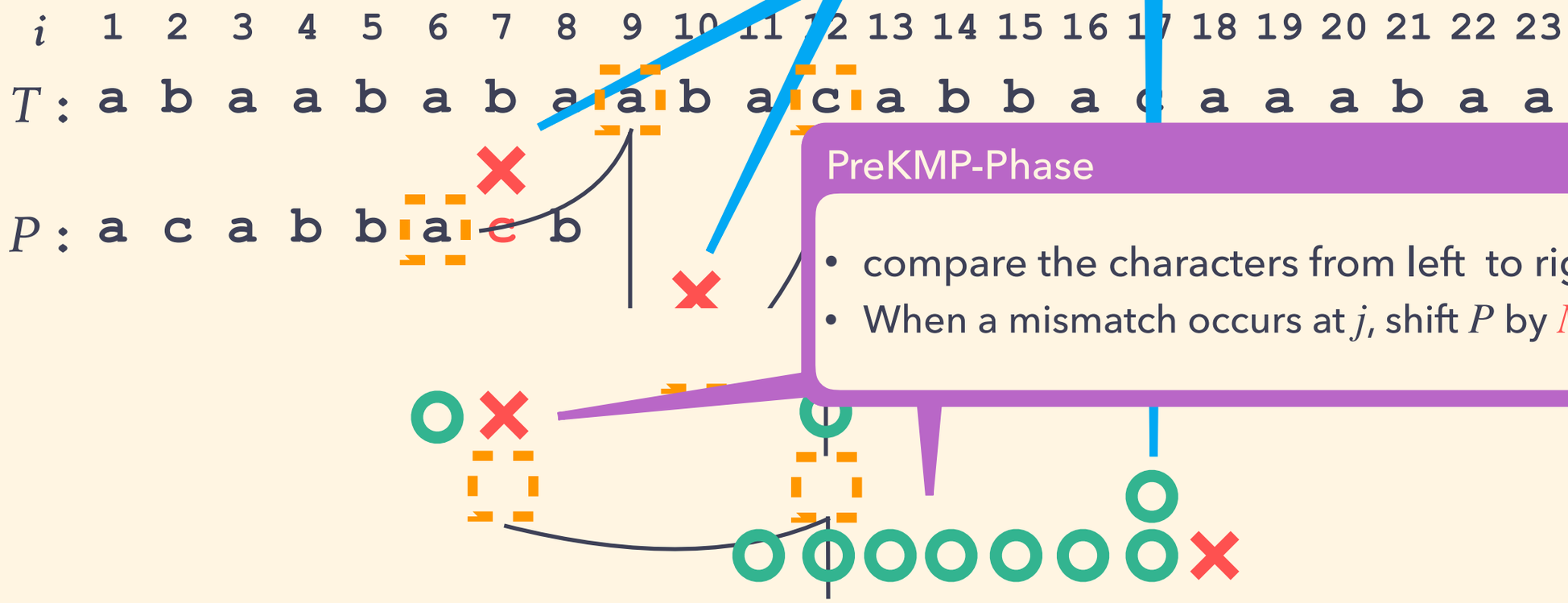
$md = 5, mdp = 7$

j	1	2	3	4	5	6	7	8	9
$P[j]$	a	c	a	b	b	a	c	b	
$KMP\_Shift[j]$	1	1	3	2	4	6	6	5	8
$MAX\_Shift[j]$	5	5	5	5	5	6	6	5	8

c	a	b	c
$Sunday\_Shift$	3	1	2

Sunday-Phase

Shift P by $Sunday\_Shift[T[i+m]]$ until $P[mdp] = T[i+mdp-1]$



PreKMP-Phase

- compare the characters from left to right
- When a mismatch occurs at j, shift P by $MAX\_Shift[j]$

KMP-Phase

If some non-empty prefixes of the pattern and the text substring match, execute the KMP matching

Time complexity of the proposed algorithm

[Preprocessing]

- Compute KMP_Shift ⟶ $O(m)$
- Compute $Sunday\_Shift$ ⟶ $O(m+\sigma)$
- Compute md, mdp ⟶ $O(m+\sigma)$
- Compute MAX_Shift ⟶ $O(m)$

$O(m+\sigma)$

[Searching]

- Just like the FJS algorithm ⟶ $O(n)$

Preprocessing time : $O(m+\sigma)$ Searching time : $O(n)$

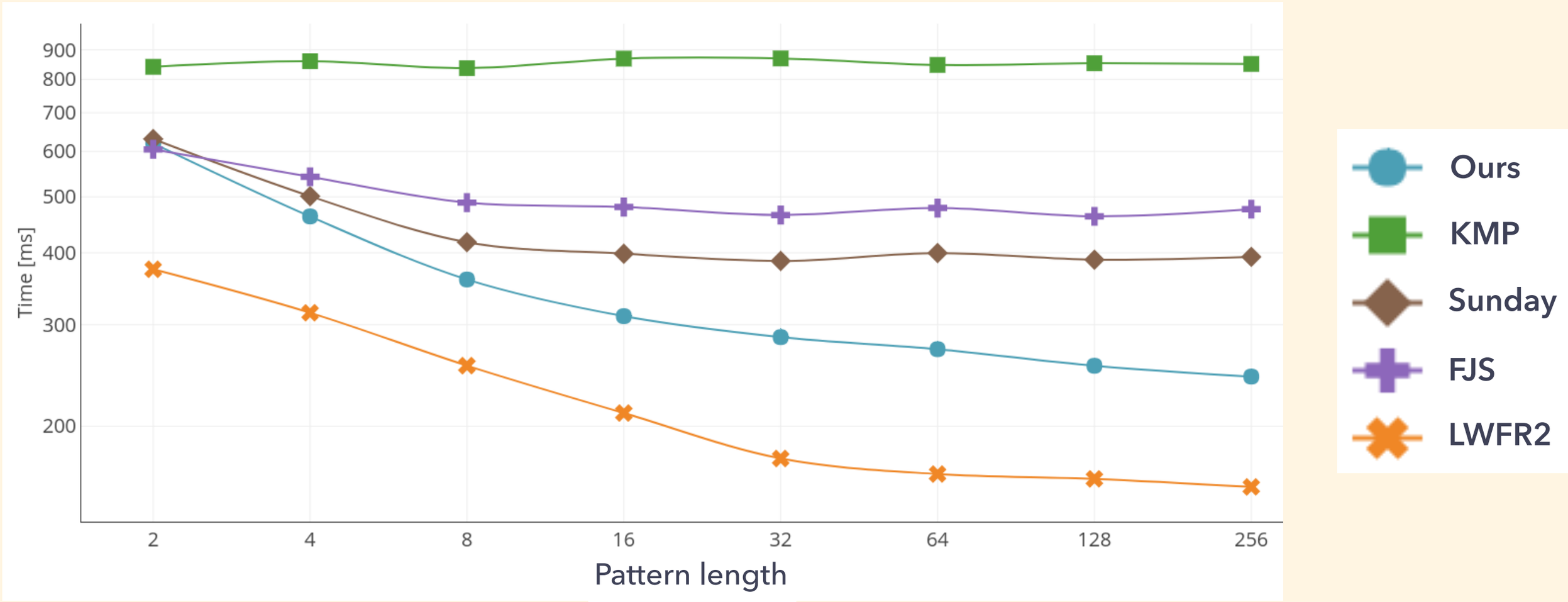
$n = |T|$: text length $m = |P|$: pattern length $\sigma = |\Sigma|$: alphabet size

Experiments (Genome sequence)

ATCGGTAGAGTAGATAG

- The genome sequence of E.coli of length $n = 4641652$ with $\sigma = 4$

Algorithm	Description
KMP [Knuth+, 1977]	Knuth-Morris-Pratt
Sunday [Sunday, 1990]	Sunday's Quick-Search
FJS [Franek+, 2005]	Franek-Jennings-Smyth
LWFR2 [Cantone+, 2019]	Linear Weak Factor Recognization with 2-chained loop
Ours	Proposed algorithm



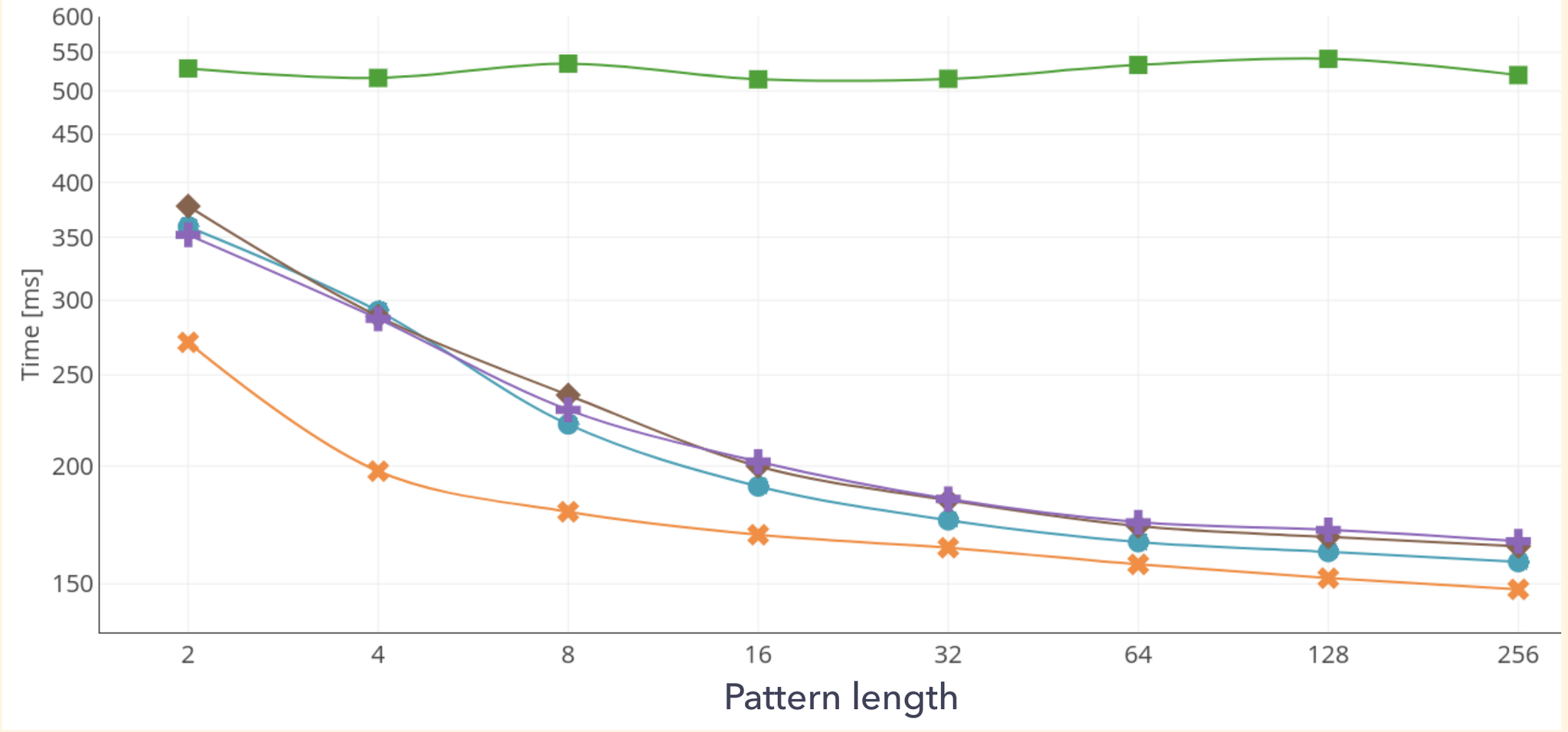
implemented in C language, compiled using GCC 5.4.0 with -O3 option

Experiments (English text)

In the beginning,

- The King James version of the Bible of length $n = 4017009$ with $\sigma = 62$

Algorithm	Description
KMP [Knuth+, 1977]	Knuth-Morris-Pratt
Sunday [Sunday, 1990]	Sunday's Quick-Search
FJS [Franek+, 2005]	Franek-Jennings-Smyth
LWFR2 [Cantone+, 2019]	Linear Weak Factor Recognization with 2-chained loop
Ours	Proposed algorithm



Experiments (Fibonacci string)

abaababaabaababaa

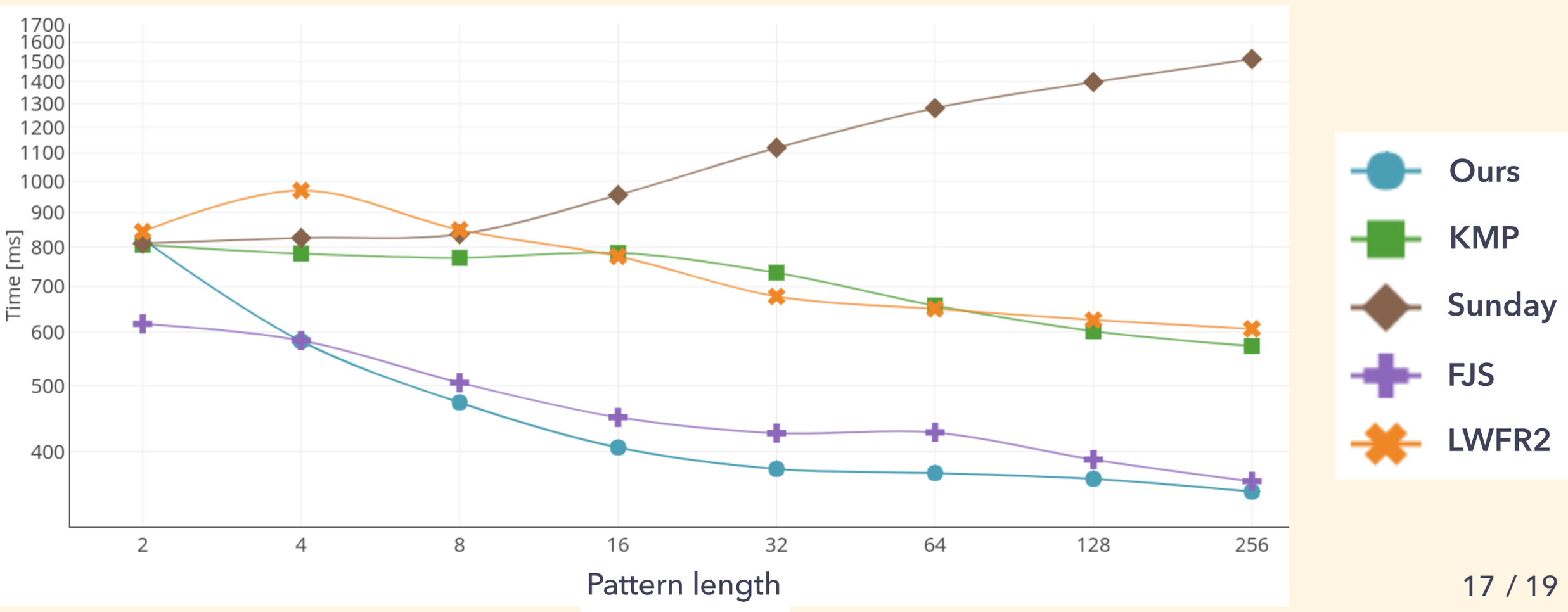
- Fib_{32} of length $n = 2178309$ with $\sigma = 2$ generated by

  $Fib_1 = \mathtt{b}$

  $Fib_2 = \mathtt{a}$

  $Fib_n = Fib_{n-1} \cdot Fib_{n-2}$ for $n > 2$

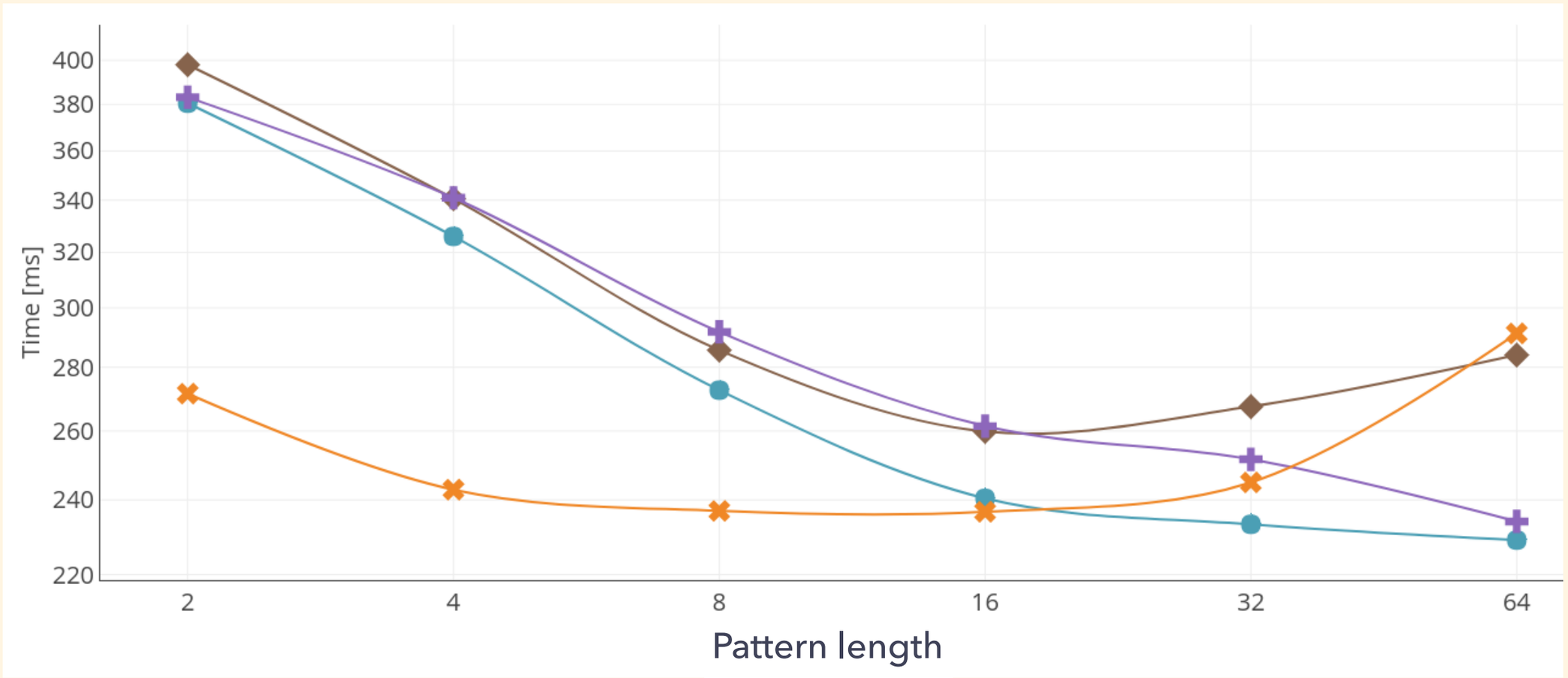
Algorithm	Description
KMP [Knuth+, 1977]	Knuth-Morris-Pratt
Sunday [Sunday, 1990]	Sunday's Quick-Search
FJS [Franek+, 2005]	Franek-Jennings-Smyth
LWFR2 [Cantone+, 2019]	Linear Weak Factor Recognization with 2-chained loop
Ours	Proposed algorithm



Experiments (many pattern occurrences)

- A random pattern is embedded 32768 times into a random text of length $n = 4000000$ with $\sigma = 8$

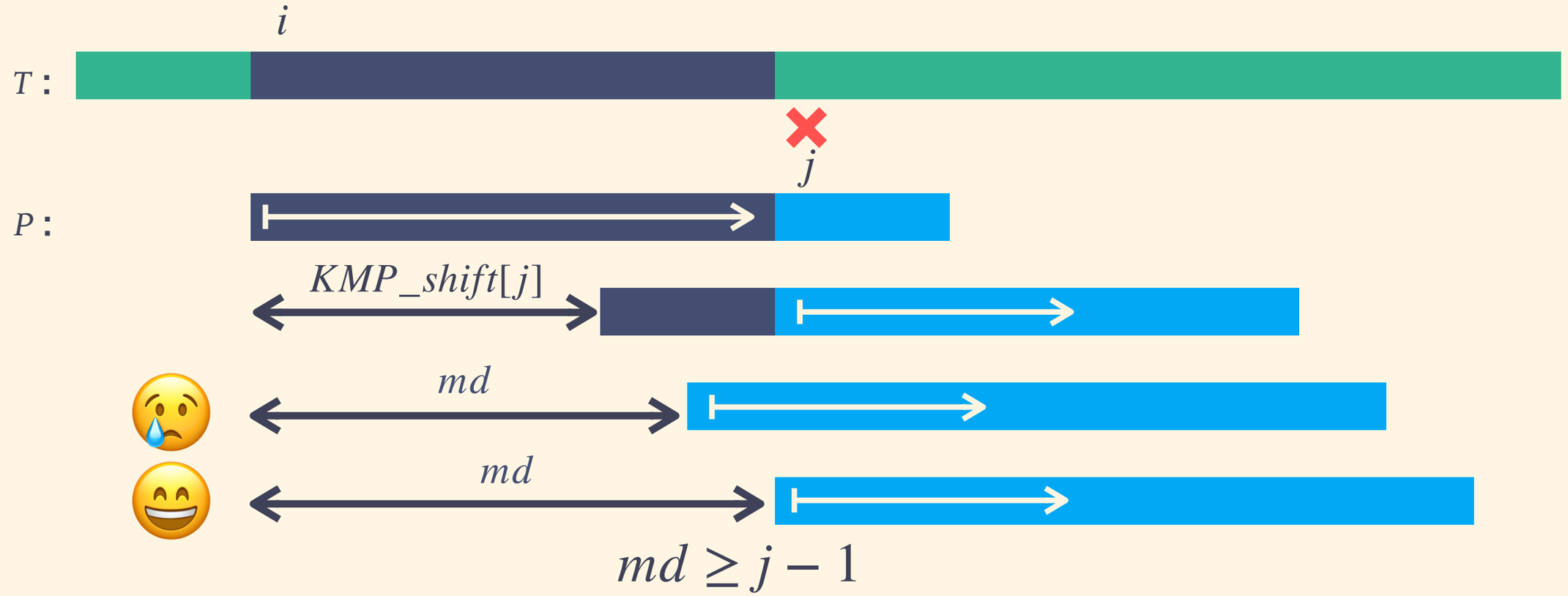
Algorithm	Description
KMP [Knuth+, 1977]	Knuth-Morris-Pratt
Sunday [Sunday, 1990]	Sunday's Quick-Search
FJS [Franek+, 2005]	Franek-Jennings-Smyth
LWFR2 [Cantone+, 2019]	Linear Weak Factor Recognization with 2-chained loop
Ours	Proposed algorithm



Conclusion

- We proposed a new pattern matching algorithm
  - Based on the FJS algorithm
  - Generalize the idea of the Quite-Naive algorithm
  - Preprocessing and searching time are $O(m + \sigma)$ and $O(n)$, respectively
  - Runs faster than the FJS algorithm in general except when a pattern is extremely short
  - Effective when a pattern frequently appears in a text

Case 1: $KMP\_Shift[j] < j$



Case 2: $KMP\_Shift[j] = j$