Fast and Linear-Time String Matching Algorithms Based on the Distances of q-Gram Occurrences

Satoshi Kobayashi, Diptarama Hendrian, Ryo Yoshinaka, Ayumi Shinohara

Graduate School of Information Sciences, Tohoku University, Japan

String matching problem

Input

Text T , Pattern P

Output

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

Example

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

Output : 6, 9

Naive solution：$O(nm)$ $n = |T|$ $m = |P|$

String matching problem

Input

Text T , Pattern P

Output

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

Example

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

Output : 6, 9

Naive solution：$O(nm)$ $n = |T|$ $m = |P|$

String matching problem

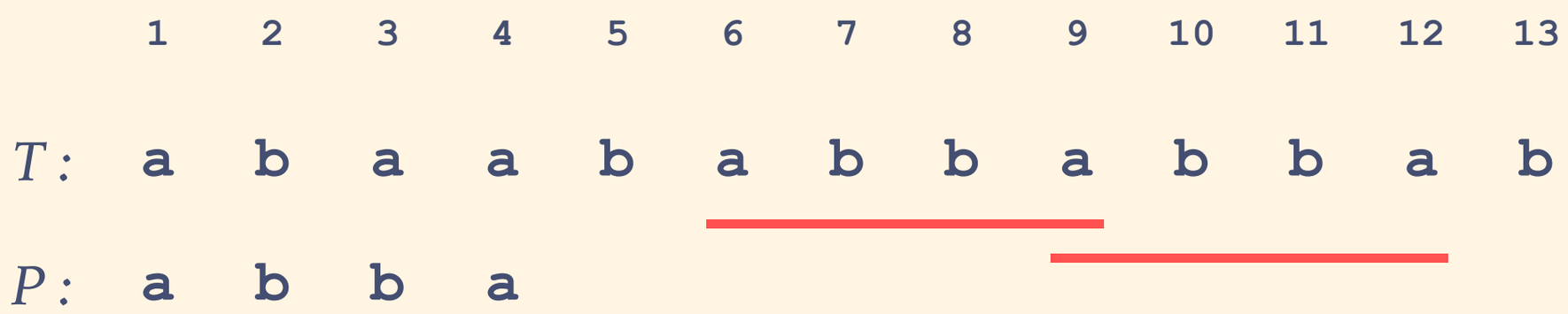
Input

Text T , Pattern P

Output

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

Example



Output : 6, 9

Naive solution：$O(nm)$ $n = |T|$ $m = |P|$

String matching algorithms

- Knuth-Morris-Pratt (KMP) algorithm [Knuth+, 1977]
  - Preprocessing time : $O(m)$
  - Searching time : $O(n)$

- Boyer-Moore algorithm [Boyer & Moore, 1977]
  - Preprocessing time : $O(m + \sigma)$
  - Searching time : $O(nm)$
  - **Runs fast in practice**

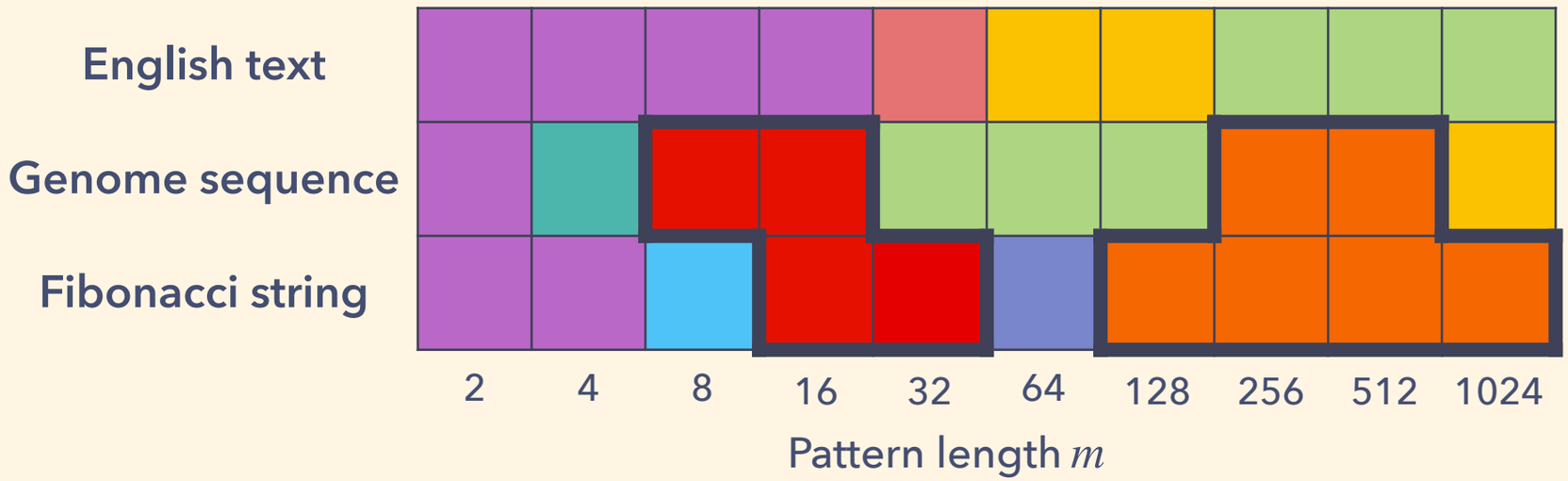
n : Text length
m : Pattern length
σ : Alphabet size

Our contributions

$n = |T|$　$m = |P|$　ω : Word length
$\sigma = |\Sigma|$ ：Alphabet size　q：q-gram

- Propose two string matching algorithms based on the distances of the q-gram occurrences
- Both algorithms work in linear time in the input string size

Fastest algorithm map for each dataset



Comparing 15 powerful algorithms announced from 1977 to 2019 with the proposed algorithms

Algorithm	Preprosess	Search
BNDMq [Navarro & Raffinot, 1998]	$O(m+\sigma)$	$O(nm\lceil m/\omega \rceil)$
SBNDMq [Holub & Durian, 2005]	$O(m+\sigma)$	$O(nm\lceil m/\omega \rceil)$
FJS [Franek+, 2005]	$O(m+\sigma)$	$O(n)$
HASHq [Leqroq, 2007]	$O(mq)$	$O(n(m+q))$
BSDMq [Faro & Leqroq, 2012]	$O(m)$	$O(nm)$

Algorithm	Preprocess	Search
WFRq [Cantone+, 2017]	$O(m)$	$O(nm)$
LWFRq [Cantone+, 2019]	$O(m)$	$O(n)$
DISTq New	$O(mq)$	$O(nq)$
LDISTq New	$O(m)$	$O(n)$

Naive solution：$O(nm)$

Existing algorithms

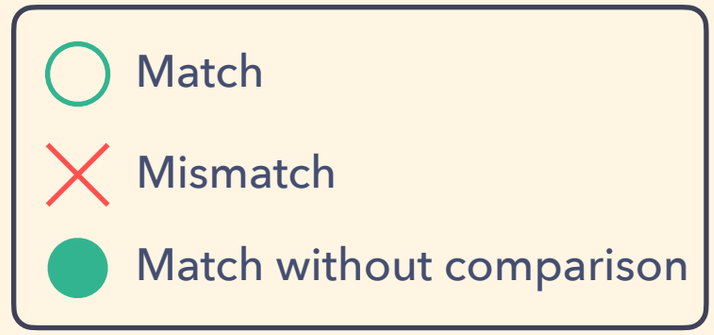
KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length

T: a b a b a b b b c a a c a c

P: a b a b c



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

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length

T: a b a b a b b b c a a c a c

P: a b a b c

Match

Mismatch

Match without comparison

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

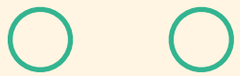
KMP algorithm [Knuth+, 1977]

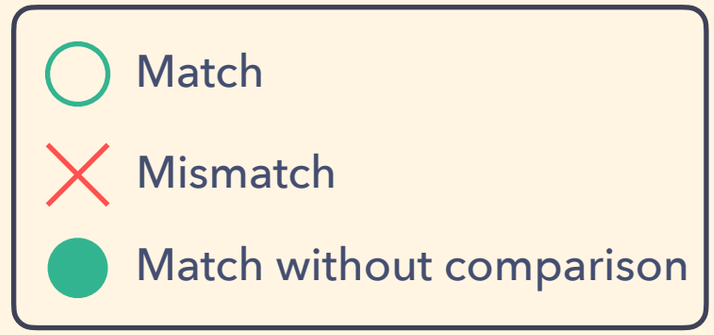
n：Text length

m：Pattern length

T: a b a b a b b b c a a c a c

P: a b a b c





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

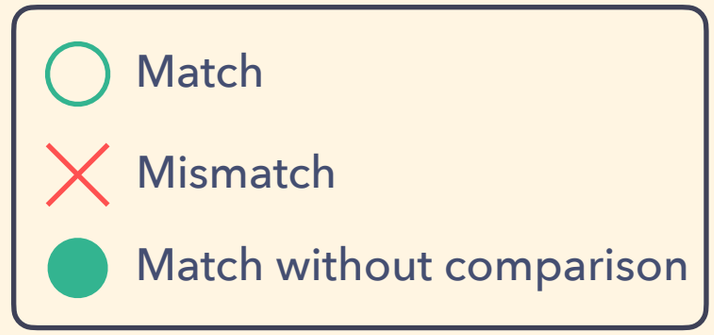
KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length

T: a b a b a b b b c a a c a c

P: a b a b c



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

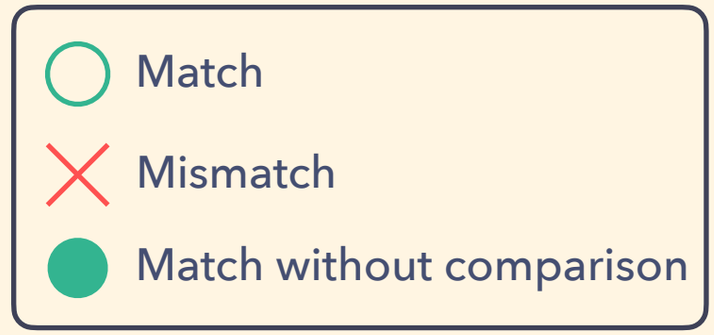
KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length

T: a b a b a b b b c a a c a c

P: a b a b c

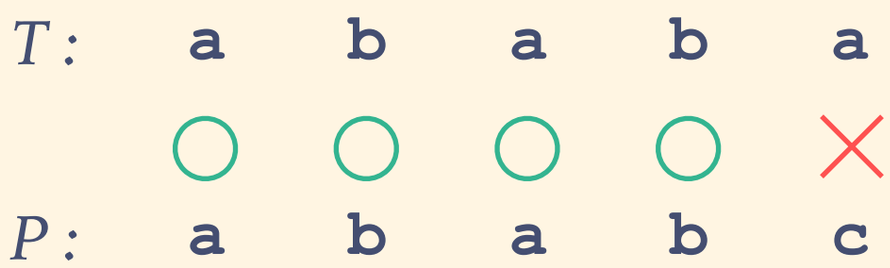


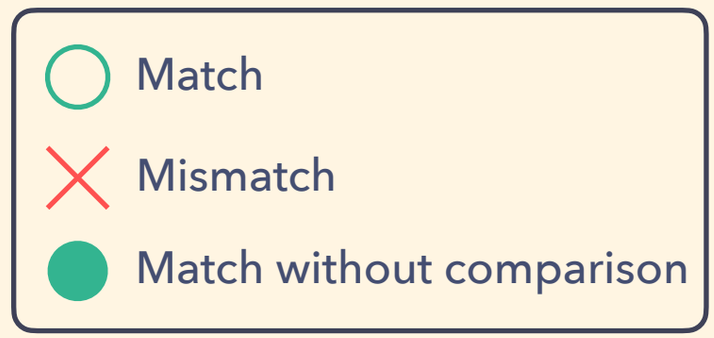
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



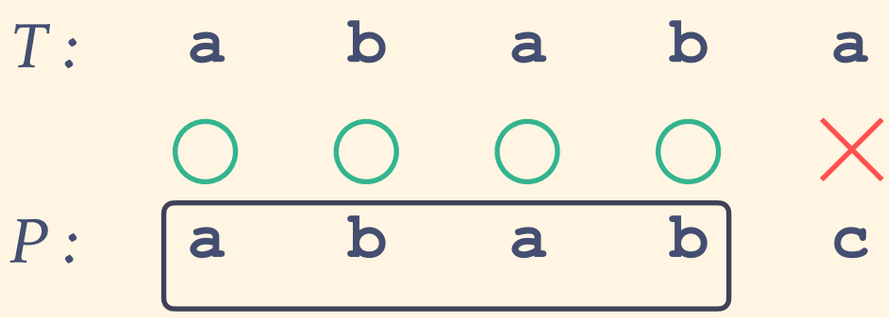


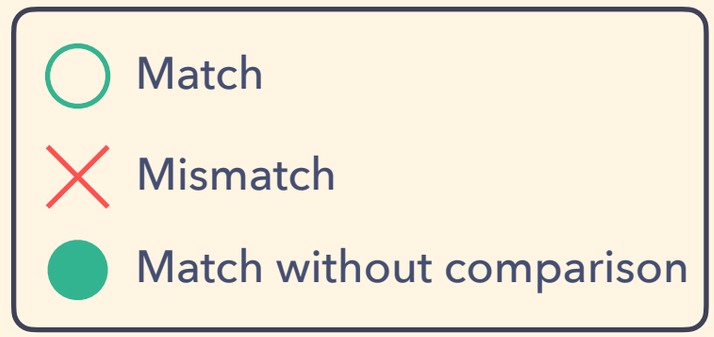
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



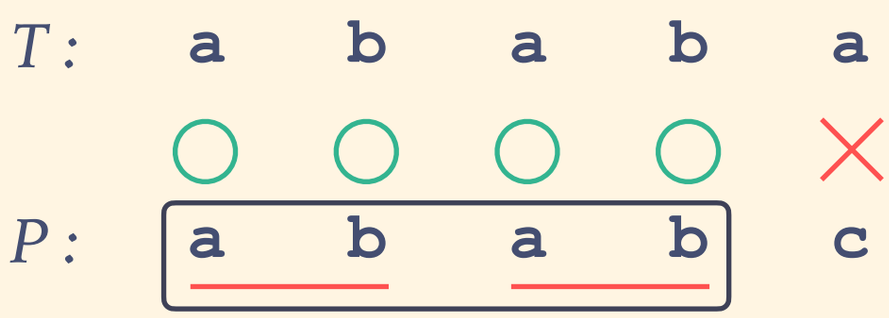


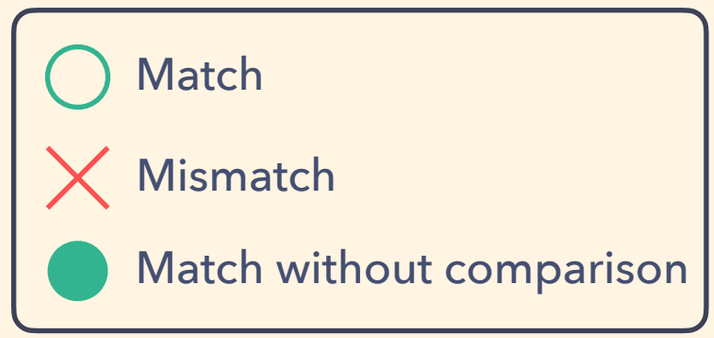
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



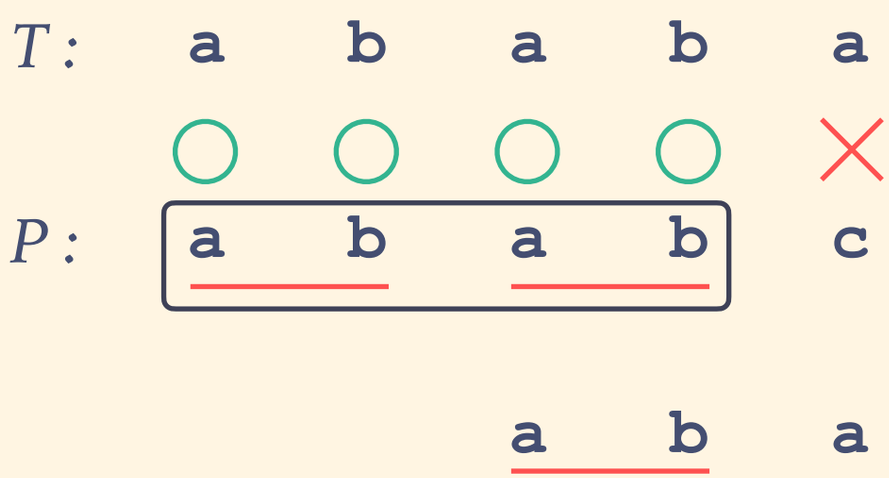


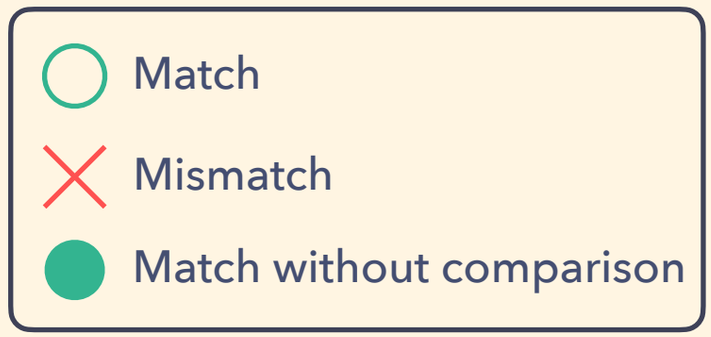
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



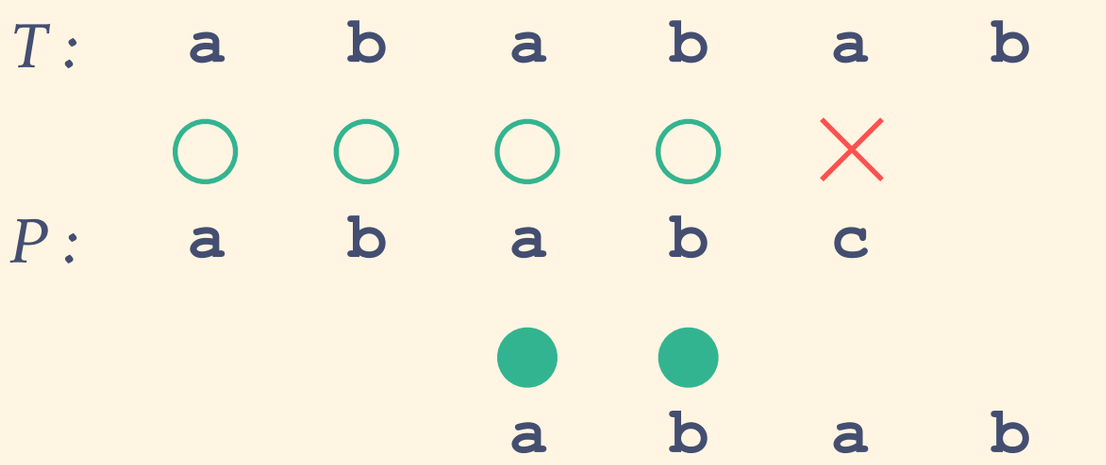


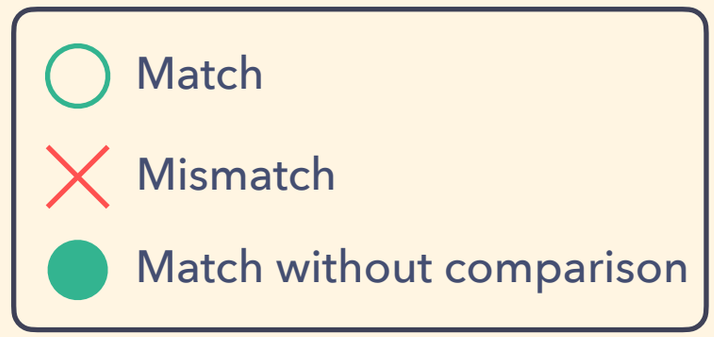
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



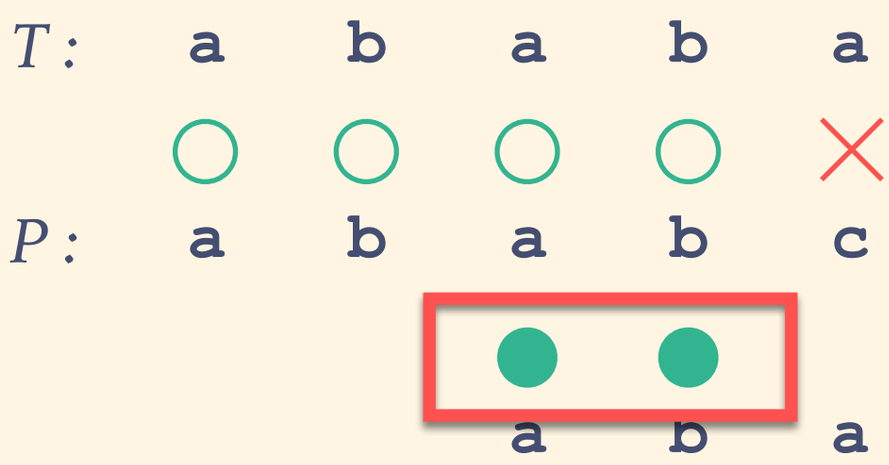


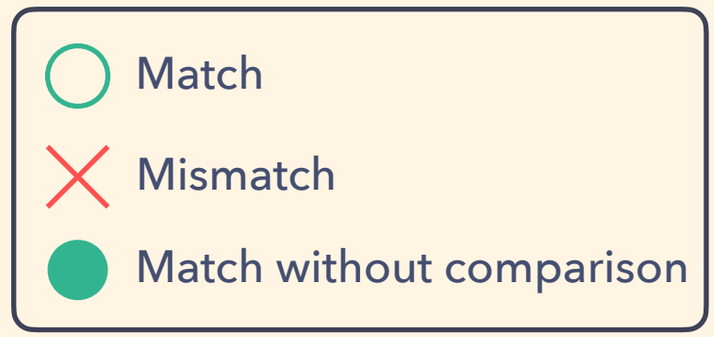
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



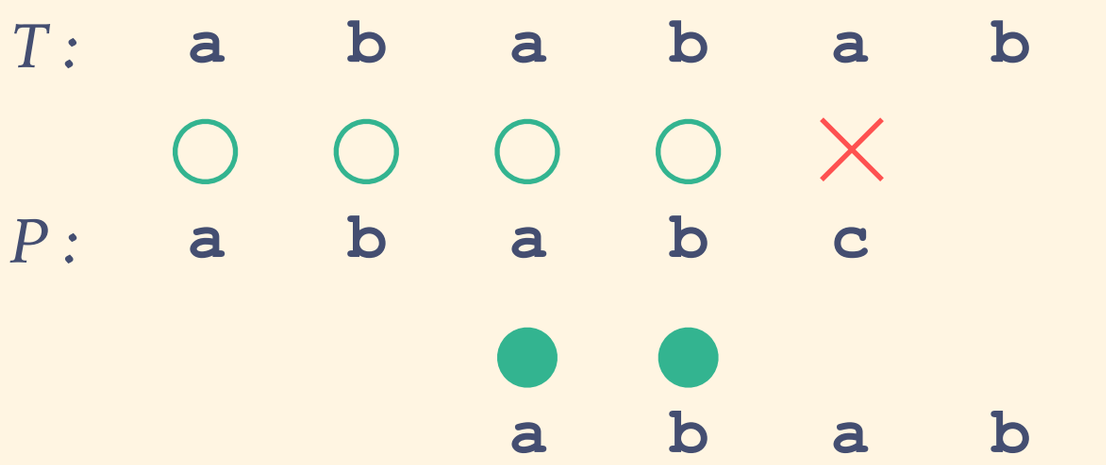


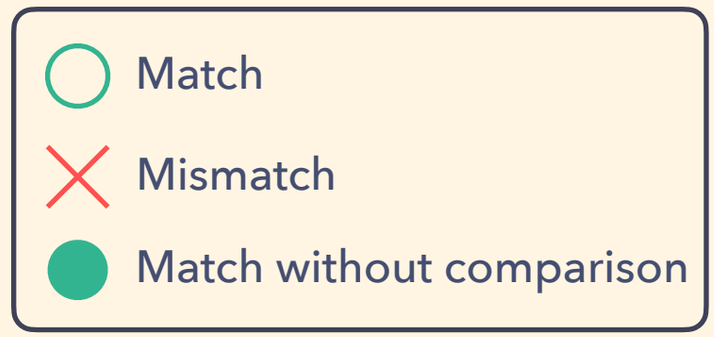
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



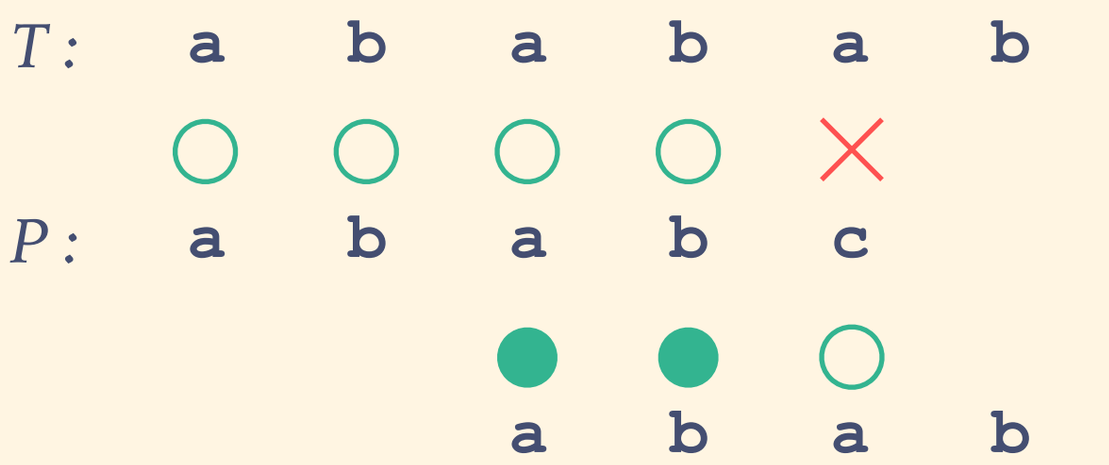


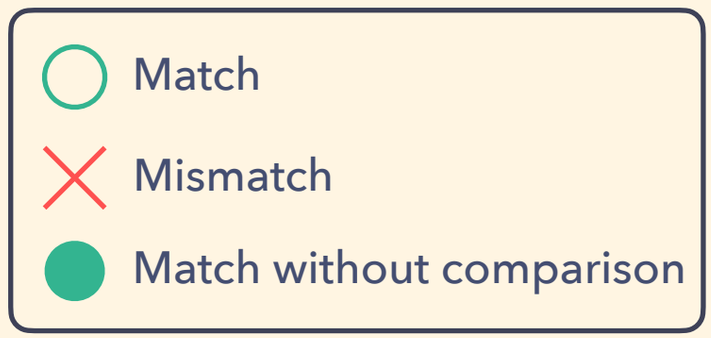
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



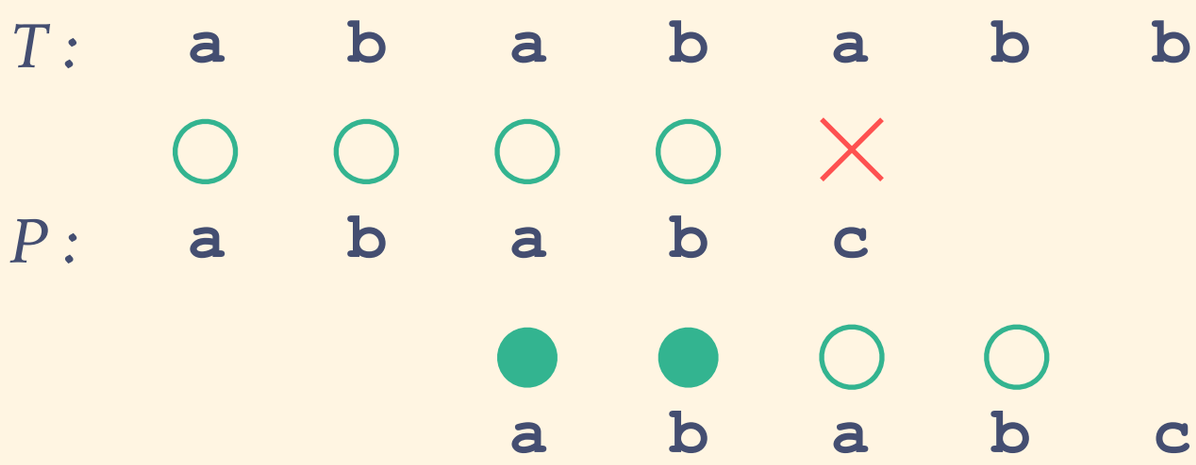


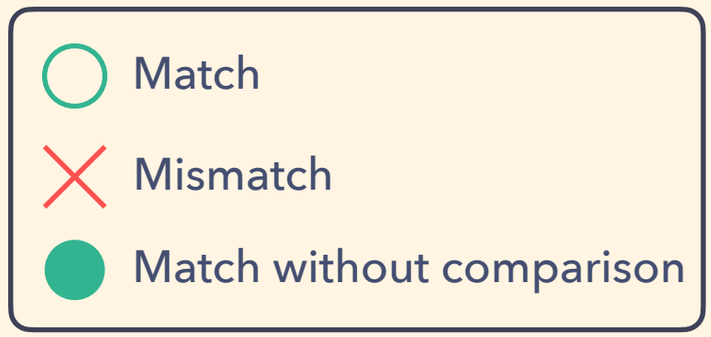
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



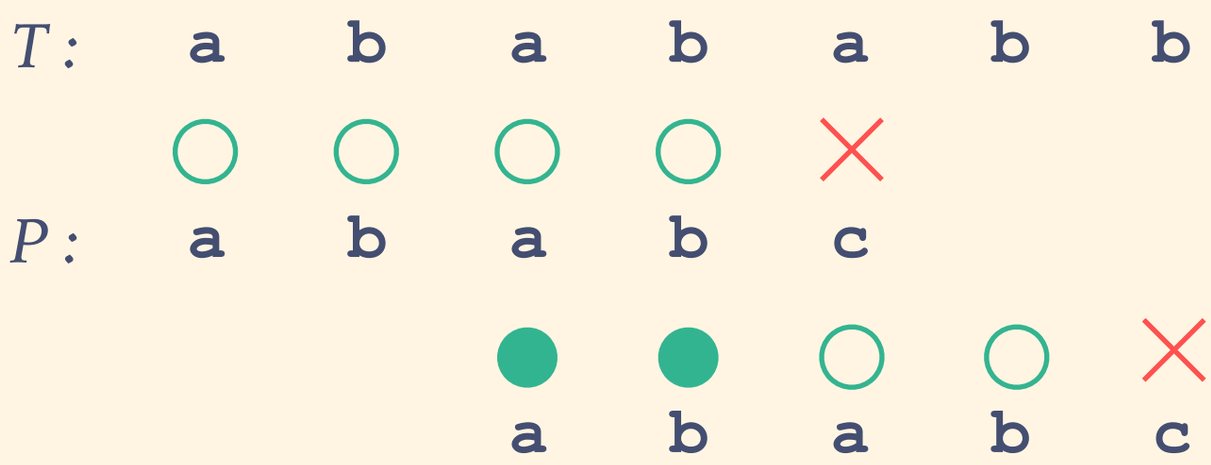


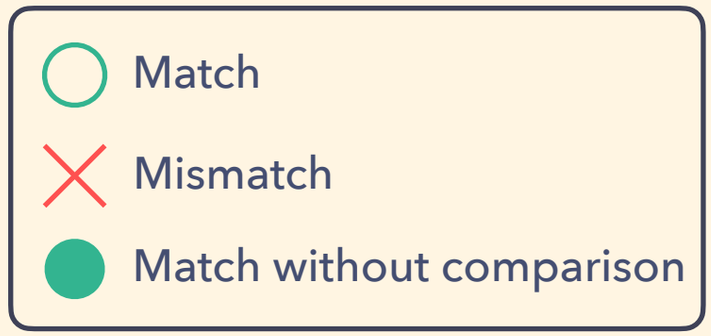
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



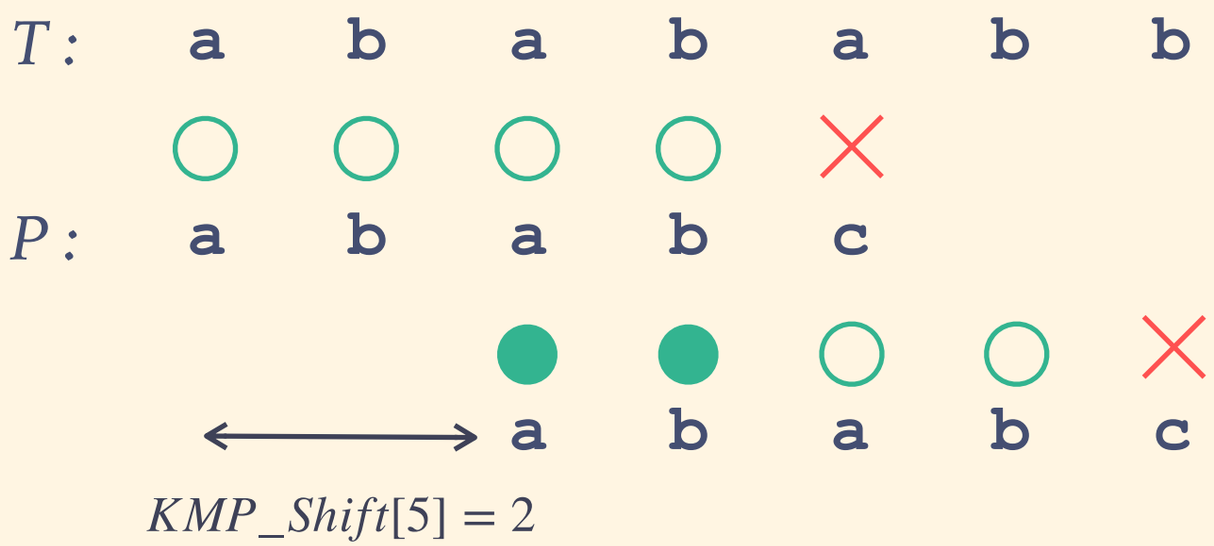


Preprocessing time：$O(m)$　Searching time：$O(n)$

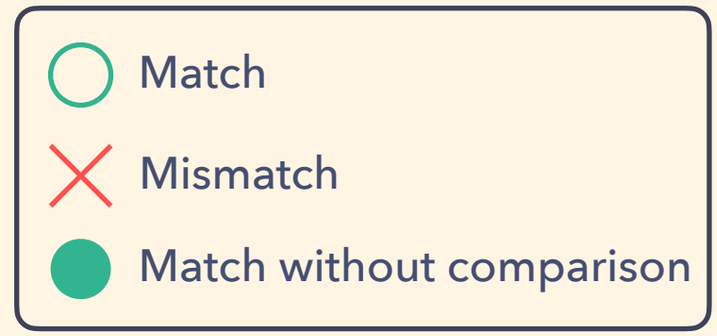
KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length



$KMP\_Shift[5] = 2$



$Strong\_Bord(j)$

Input　: A mismatch position j in the pattern

Output : A maximum value $k(0 \leq k < j)$ that satisfies $P[1:k] = P[j-k:j-1]$ and $P[k+1] \neq P[j]$

(-1 if no such k exists)

A shift amount when there is a mismatch in the j-th pattern

$KMP\_Shift[j] = j - Strong\_Bord(j) - 1$

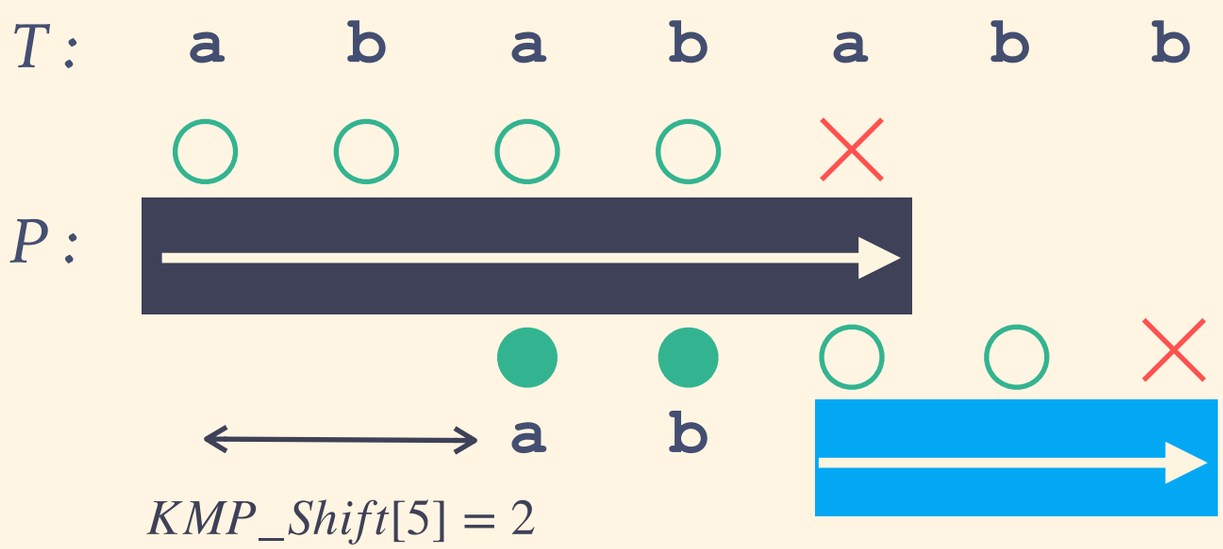
j	1	2	3	4	5	6
P	a	b	a	b	c	
KMP_Shift	1	1	3	3	2	5

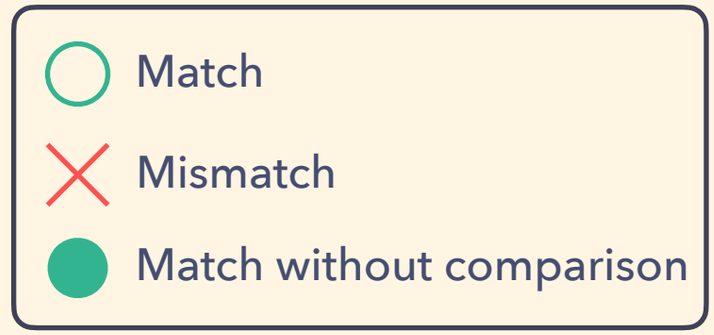
Preprocessing time：$O(m)$　Searching time：$O(n)$

KMP algorithm [Knuth+, 1977]

n：Text length

m：Pattern length





$Strong\_Bord(j)$

Input　: A mismatch position j in the pattern

Output : A maximum value $k(0 \leq k < j)$ that satisfies $P[1:k] = P[j-k:j-1]$ and $P[k+1] \neq P[j]$

(-1 if no such k exists)

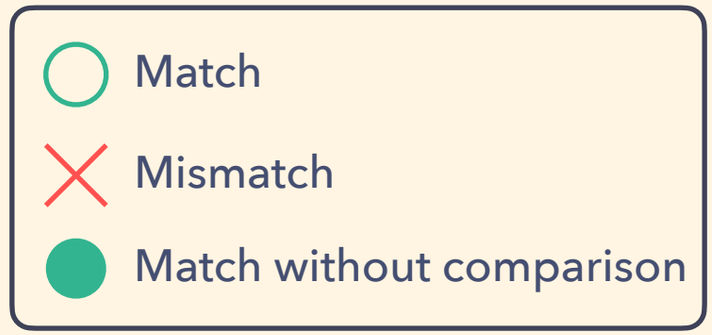
A shift amount when there is a mismatch in the j-th pattern

$KMP\_Shift[j] = j - Strong\_Bord(j) - 1$

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：$O(n)$

HASHq algorithm [Leqroq, 2007]



T: **a a a b a b a a b b a b a b b**

P: **a b a a b b a b**

P = **a b a a b b a b**

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

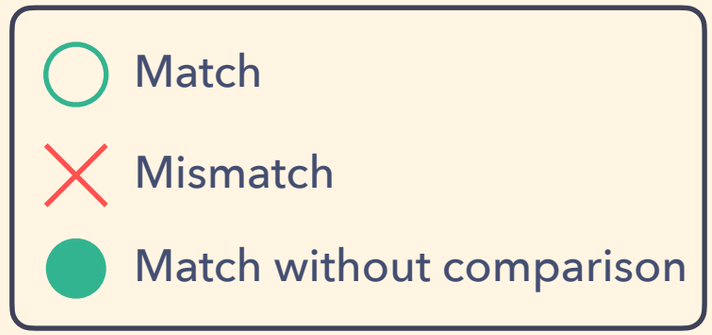
x : String

(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]



T: a a a b a b a a b b a b a b b

P: a b a a b b a b

P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \leq j \leq m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

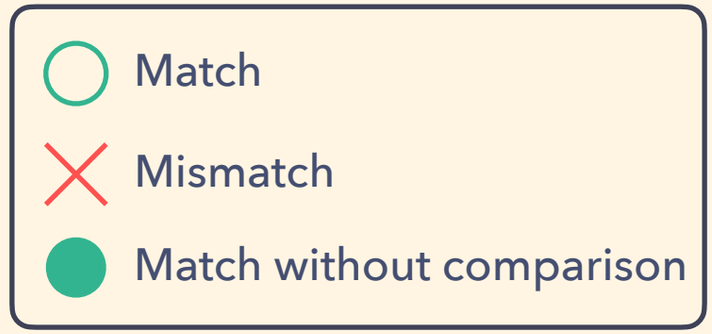
x : String

(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]



T: a a a b a b a a b b a b a b b

P: a b a a b b a b

P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \leq j \leq m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

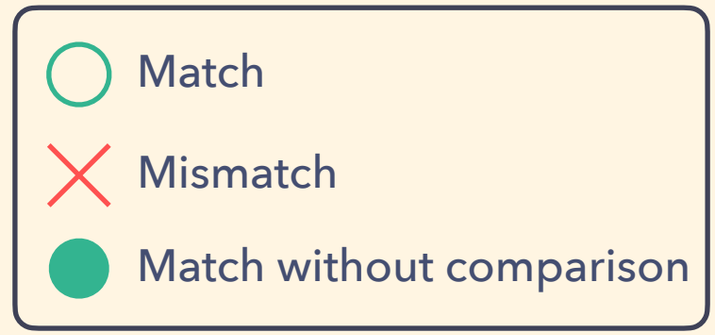
x : String

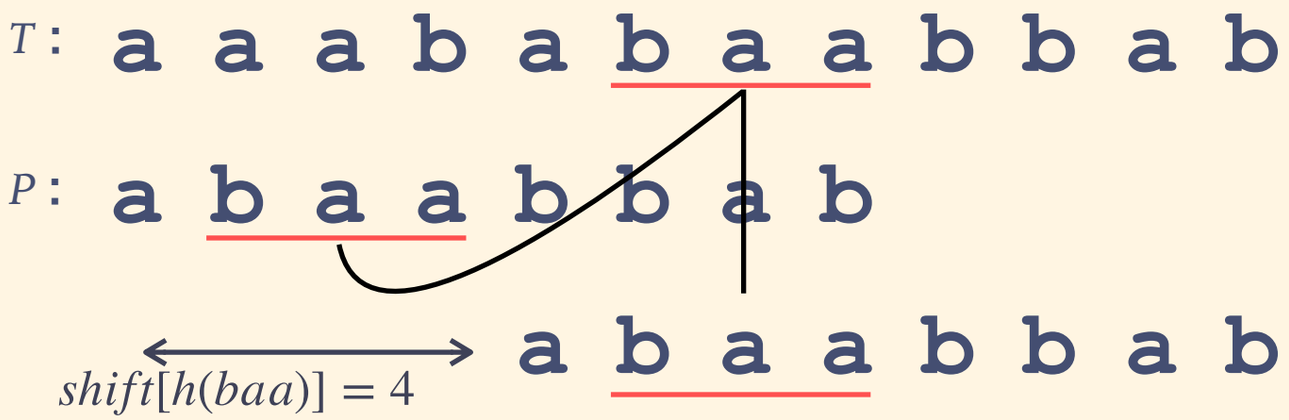
(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]





P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

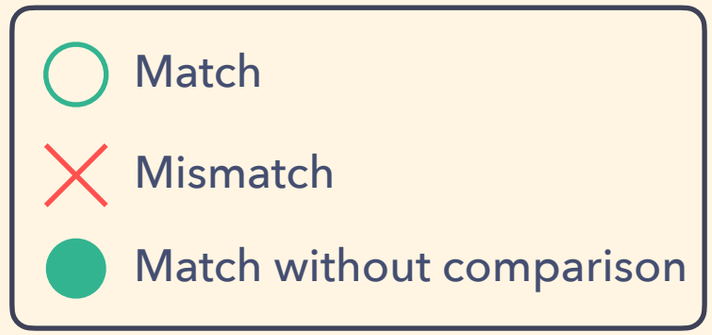
x : String

(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]



T: a a a b a b a a b b a b a b b

P: a b a a b b a b

a b a a b b a b

P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

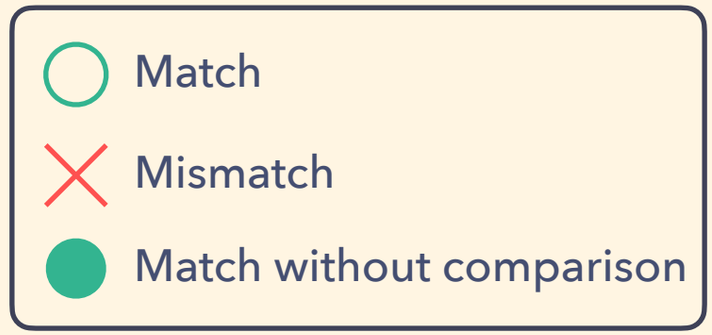
x : String

(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]



T: a a a b a b a a b b a b a b b

P: a b a a b b a b

a b a a b b a b

P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

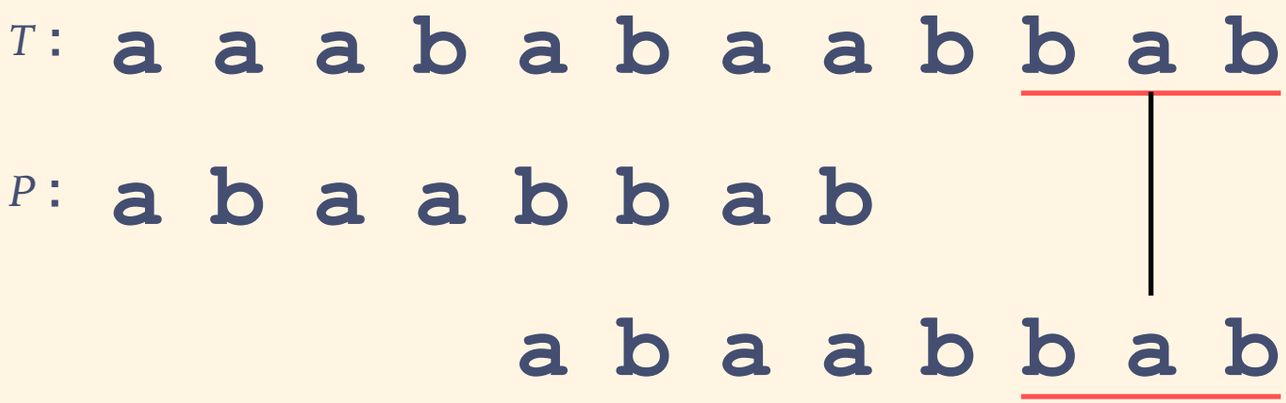
x : String

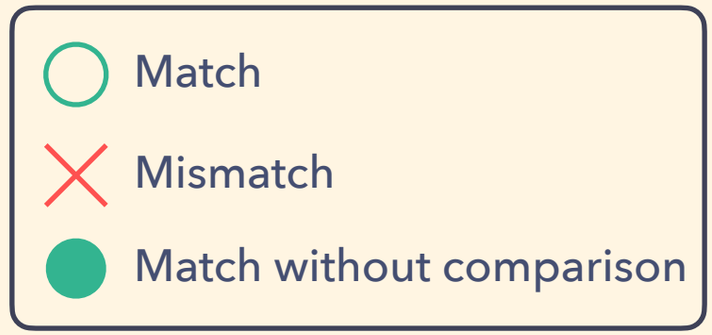
(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]





P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

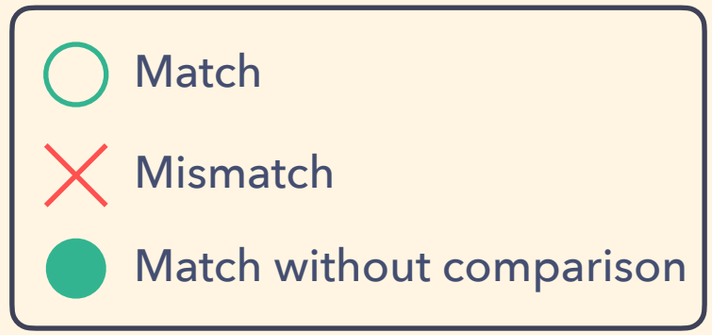
x : String

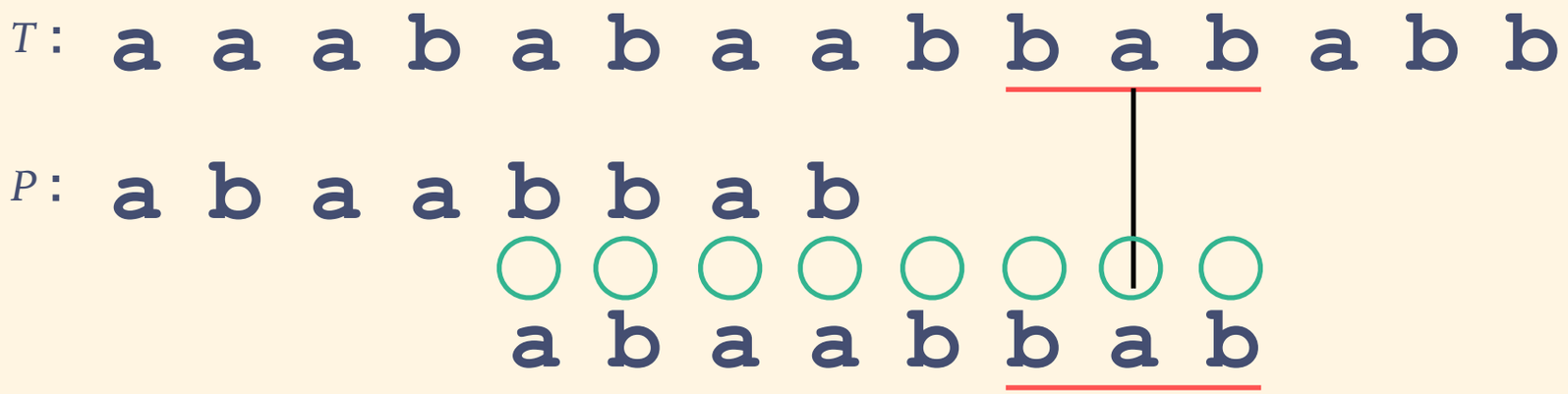
(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

HASHq algorithm [Leqroq, 2007]





P = a b a a b b a b

x	$h(x)$	$Shift[h(x)]$
aba	681	5
baa	683	4
aab	680	3
abb	682	2
bba	685	1
bab	684	0
Others	-	6

$m - q + 1$

$$shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (2^{q-1} \cdot x[1] + 2^{q-2} \cdot x[2] + \cdots + 2 \cdot x[q-1] + x[q]) \bmod 2^8$$

x : String

(Treat characters as the ASCII code)

n : Text length
m : Pattern length
σ : Alphabet size

Preprocessing time：$O(mq)$　Searching time：$O(n(m+q))$

Proposed 1

DISTq algorithm

Idea of DISTq algorithm

Practically fast

Proposed

Linear time

shift (HASHq) + q-gram distance array + KMP algorithm

- q-gram distance array

$$dist[i] = \min(\{\, j \mid h(P[i-j-q+1 : i-j]) = h(P[i-q+1 : i]),\ q-1 \le j < i \,\} \cup \{i-q+1\})$$

- A hash value is used to determine the equivalence of q-grams

$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

x : String

When $q = 3$

i	1	2	3	4	5	6	7	8	9
P	a	b	a	a	b	b	a	a	a
dist	-	-	1	2	3	4	5	4	7

4

T: b b a b a b b a a b b a b a a

P: a b a a b b a a a

Idea of DISTq algorithm

Practically fast

Proposed

Linear time

shift (HASHq) + q-gram distance array + KMP algorithm

- q-gram distance array

$$dist[i] = \min(\{\, j \mid h(P[i-j-q+1 : i-j]) = h(P[i-q+1 : i]),\ q-1 \le j < i \,\} \cup \{i-q+1\})$$

- A hash value is used to determine the equivalence of q-grams

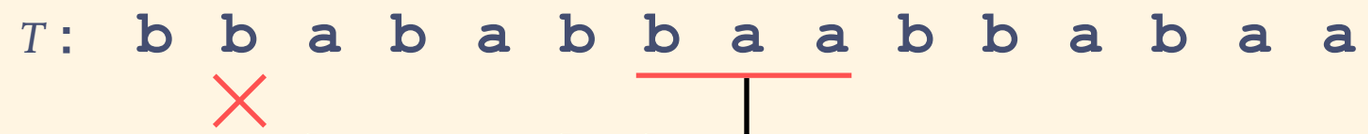
$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

x : String

When $q = 3$

i	1	2	3	4	5	6	7	8	9
P	a	b	a	a	b	b	a	a	a
dist	-	-	1	2	3	4	5	4	7

4



Idea of DISTq algorithm

Practically fast

Proposed

Linear time

shift (HASHq) + q-gram distance array + KMP algorithm

- q-gram distance array

$$dist[i] = \min(\{\, j \mid h(P[i-j-q+1 : i-j]) = h(P[i-q+1 : i]),\ q-1 \le j < i \,\} \cup \{i-q+1\})$$

- A hash value is used to determine the equivalence of q-grams

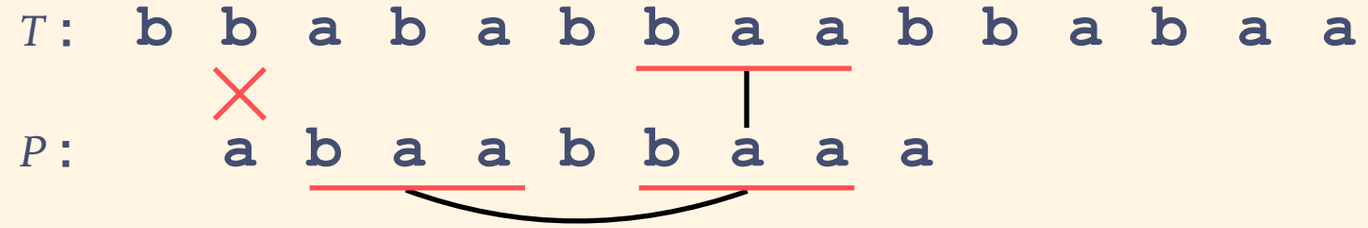
$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

x : String

When $q = 3$

i	1	2	3	4	5	6	7	8	9
P	a	b	a	a	b	b	a	a	a
dist	-	-	1	2	3	4	5	4	7

4



Idea of DISTq algorithm

Practically fast

Proposed

Linear time

shift (HASHq) + q-gram distance array + KMP algorithm

- q-gram distance array

$$dist[i] = \min(\{\, j \mid h(P[i-j-q+1 : i-j]) = h(P[i-q+1 : i]),\ q-1 \le j < i \,\} \cup \{i-q+1\})$$

- A hash value is used to determine the equivalence of q-grams

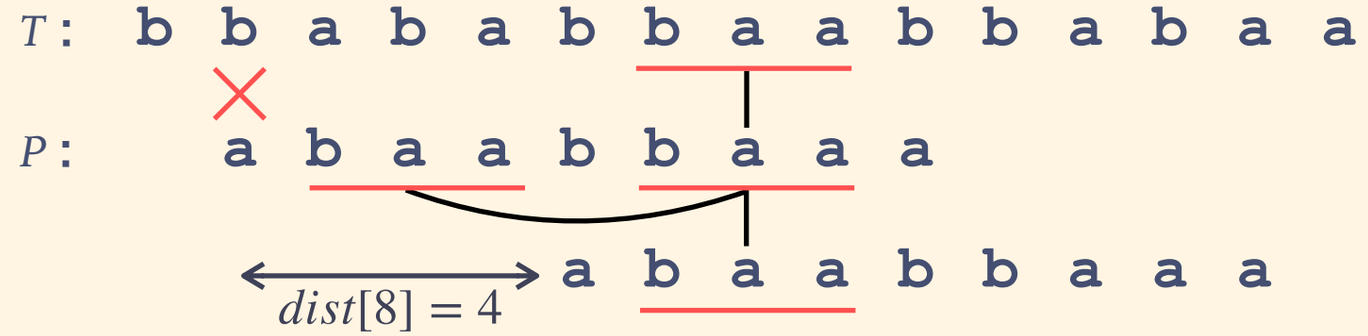
$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

x : String

When $q = 3$

i	1	2	3	4	5	6	7	8	9
P	a	b	a	a	b	b	a	a	a
dist	-	-	1	2	3	4	5	4	7

4



Array HQ_Shift (Almost same as $shift$ of HASHq)

$$HQ\_shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x),\ q \le j \le m\} \cup \{q-1\})$$

- Also determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

P = **a b a a b b a b**

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	5
baa	2053	4
aab	2038	3
abb	2042	2
bba	2057	1
bab	2054	0
Others	-	6

$m - q + 1$

T: **b b a a a b a b a a b a b b a**

P: **a b a a b b a b**

Use this shift to align the q-gram in the pattern and the q-gram in the text which has the same hash value

Array HQ_Shift (Almost same as $shift$ of HASHq)

$$HQ\_shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

- Also determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

P = **a b a a b b a b**

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	5
baa	2053	4
aab	2038	3
abb	2042	2
bba	2057	1
bab	2054	0
Others	-	6

$m - q + 1$

T: **b b a a a b a b a a b a b b a**

P: **a b a a b b a b**

Use this shift to align the q-gram in the pattern and the q-gram in the text which has the same hash value

Array HQ_Shift (Almost same as $shift$ of HASHq)

$$HQ\_shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

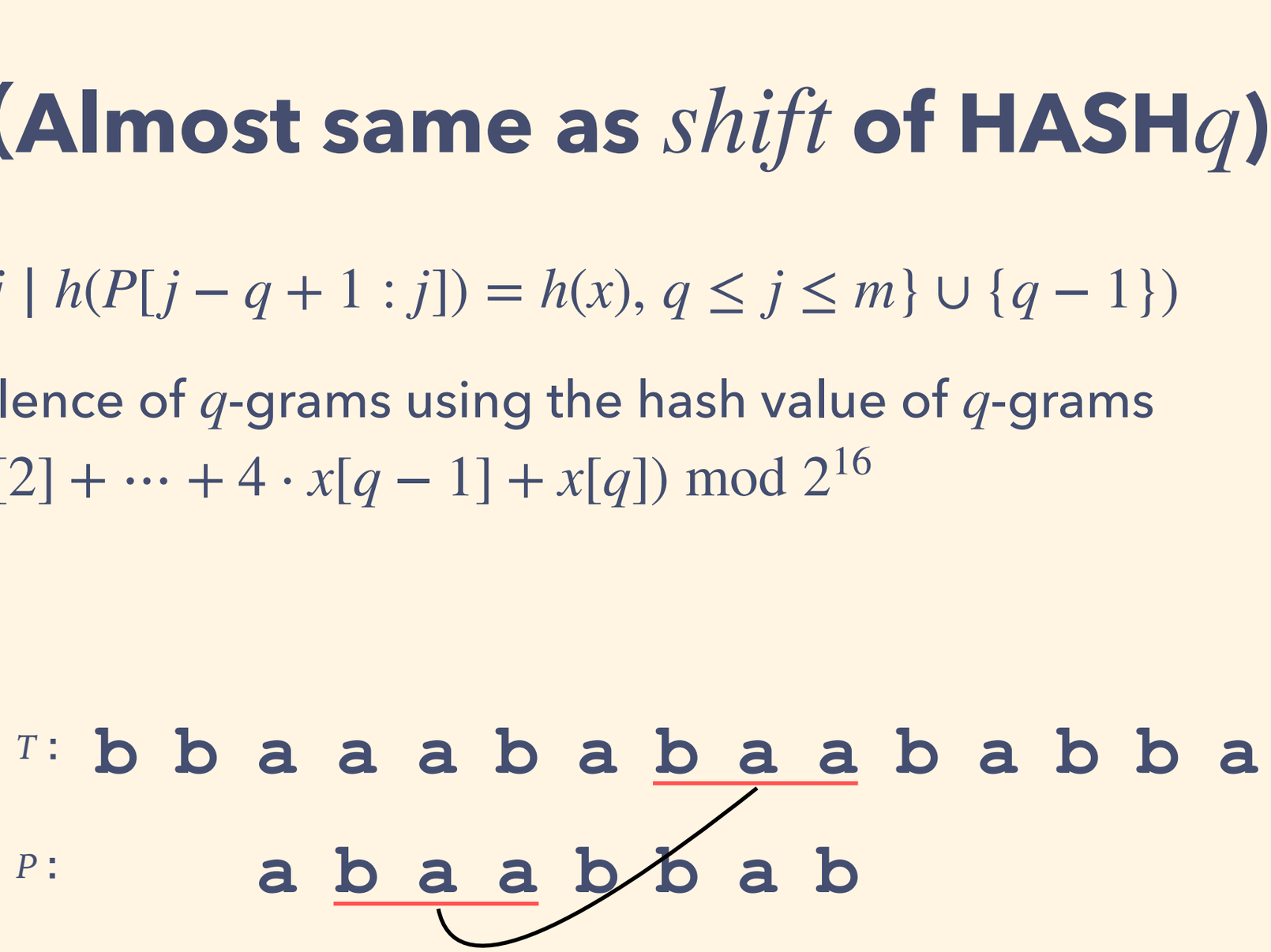
- Also determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

P = **a b a a b b a b**

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	5
baa	2053	4
aab	2038	3
abb	2042	2
bba	2057	1
bab	2054	0
Others	-	6

$m - q + 1$



Use this shift to align the q-gram in the pattern and the q-gram in the text which has the same hash value

Array HQ_Shift (Almost same as $shift$ of HASHq)

$$HQ\_shift[h(x)] = m - \max(\{j \mid h(P[j-q+1:j]) = h(x), q \le j \le m\} \cup \{q-1\})$$

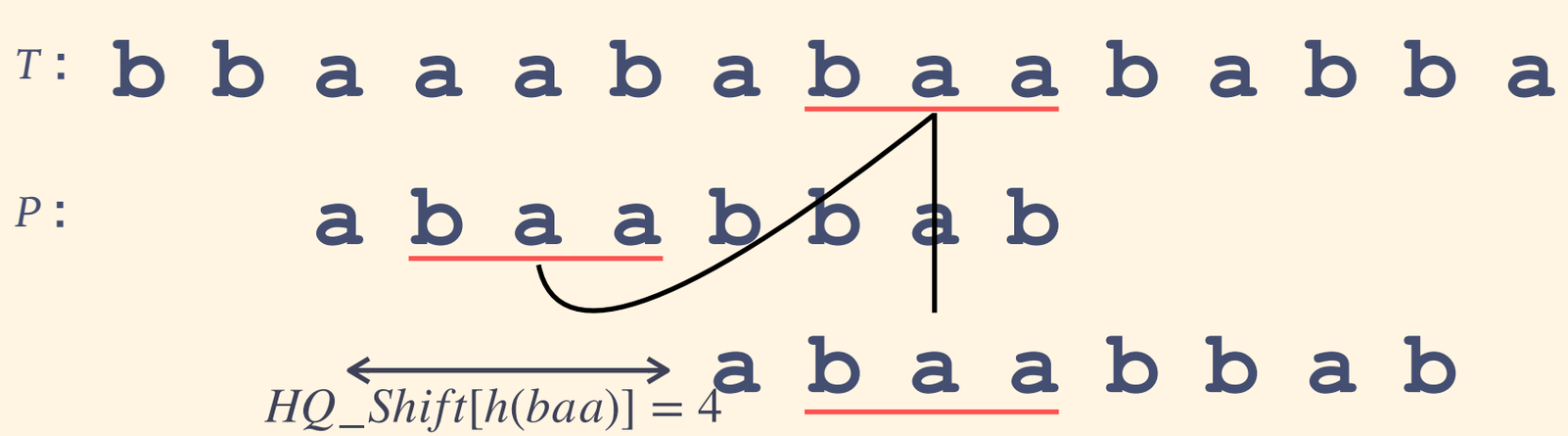
- Also determines the equivalence of q-grams using the hash value of q-grams

$$h(x) = (4^{q-1} \cdot x[1] + 4^{q-2} \cdot x[2] + \cdots + 4 \cdot x[q-1] + x[q]) \bmod 2^{16}$$

P = **a b a a b b a b**

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	5
baa	2053	4
aab	2038	3
abb	2042	2
bba	2057	1
bab	2054	0
Others	-	6

$m - q + 1$



Use this shift to align the q-gram in the pattern and the q-gram in the text which has the same hash value

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “baa” by shifting with array HQ_Shift until P[1] matches the corresponding text

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

Match
Mismatch
Match without comparison

Searching

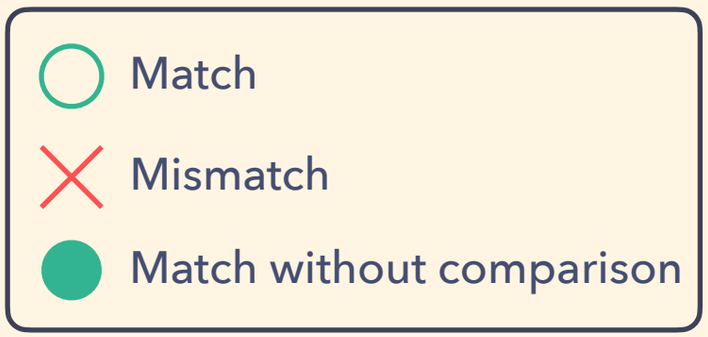
x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “baa” by shifting with array HQ_Shift until P[1] matches the corresponding text

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “baa” by shifting with array HQ_Shift until P[1] matches the corresponding text

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

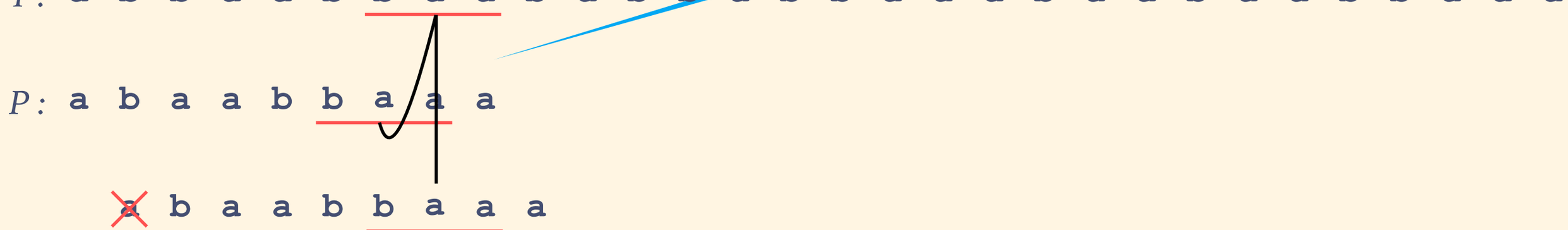
Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “baa” by shifting with array HQ_Shift until P[1] matches the corresponding text



Match
Mismatch
Match without comparison

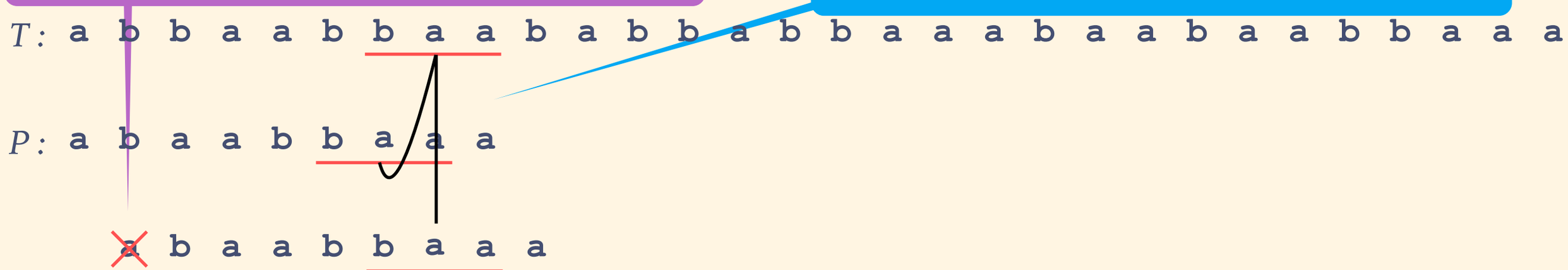
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Shift the pattern using the distance array $dist$ if the first letter do not match

Alignment-Phase
Align "baa" by shifting with array HQ_Shift until P[1] matches the corresponding text



- Match
- Mismatch
- Match without comparison

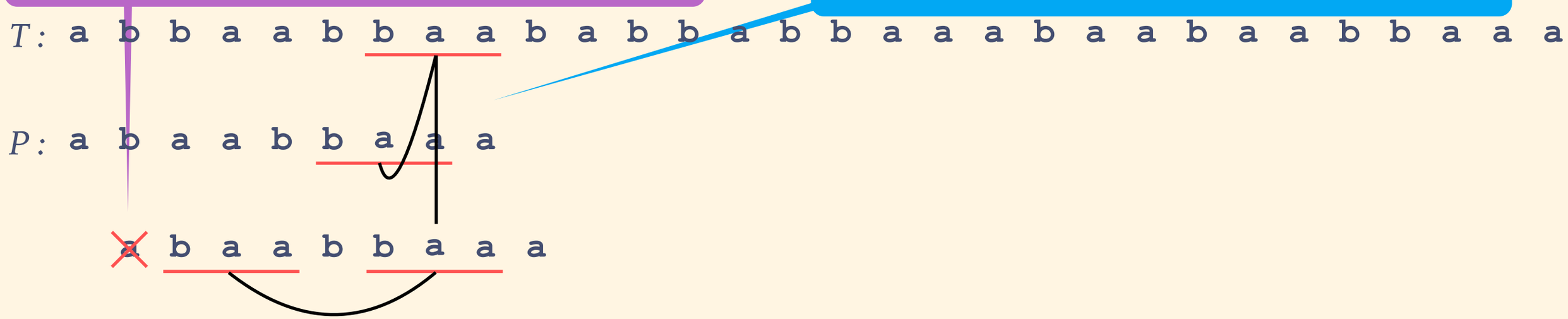
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Shift the pattern using the distance array $dist$ if the first letter do not match

Alignment-Phase
Align "baa" by shifting with array HQ_Shift until P[1] matches the corresponding text



- Match
- Mismatch
- Match without comparison

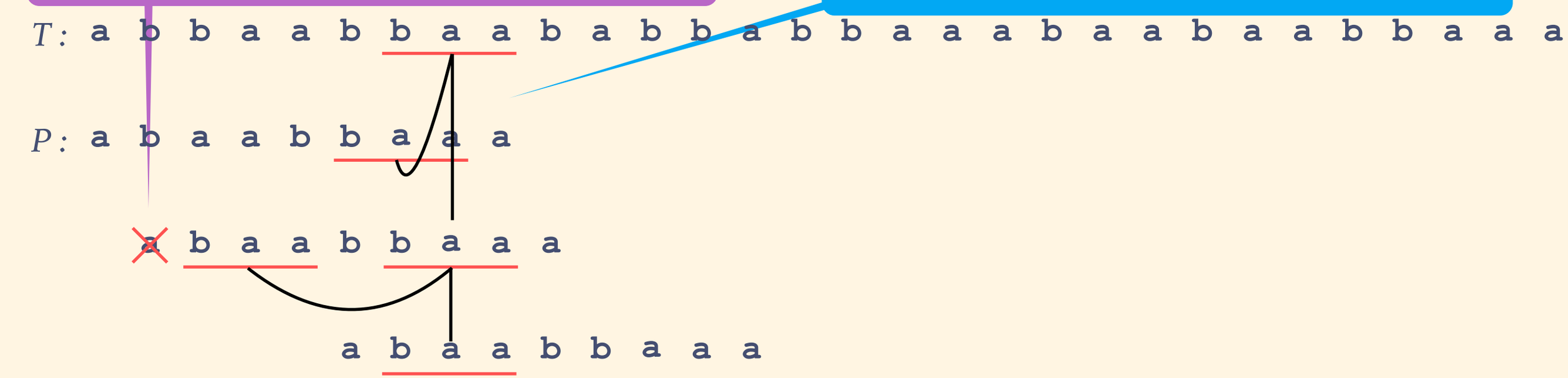
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Shift the pattern using the distance array $dist$ if the first letter do not match

Alignment-Phase
Align "baa" by shifting with array HQ_Shift until P[1] matches the corresponding text



Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

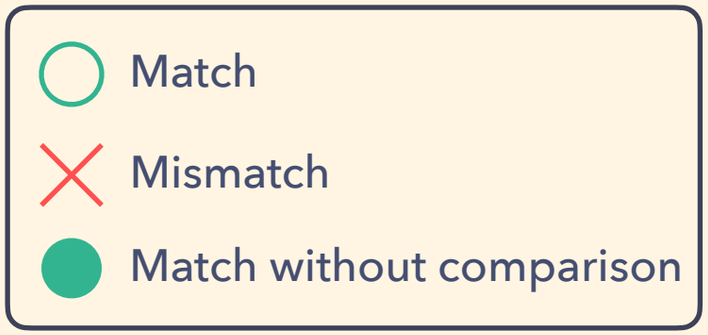
Alignment-Phase
Align “bba” by shifting with array HQ_Shift until P[1] matches the corresponding text

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

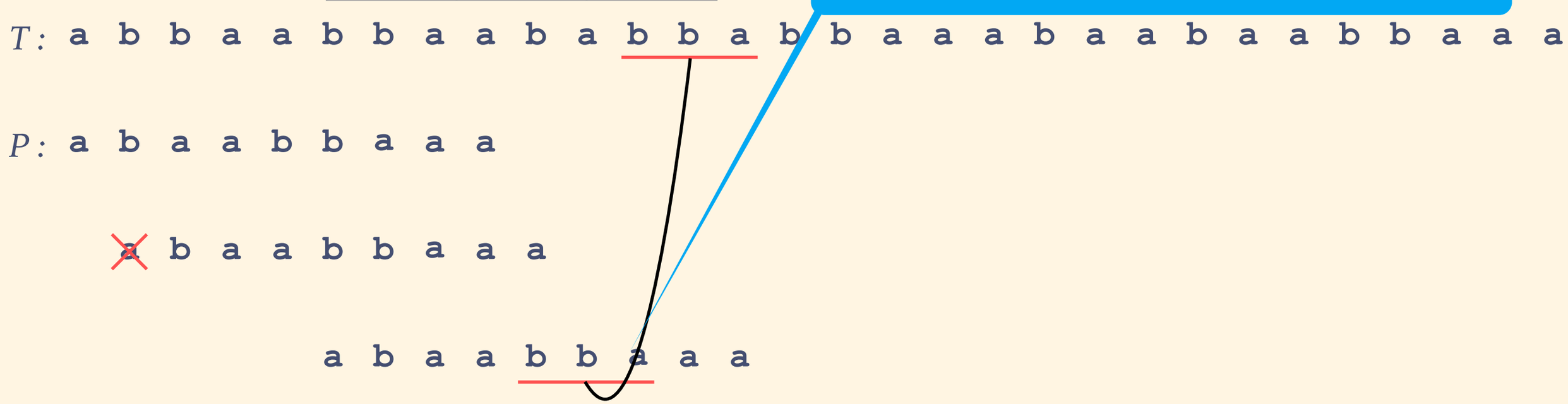


Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “bba” by shifting with array HQ_Shift until P[1] matches the corresponding text



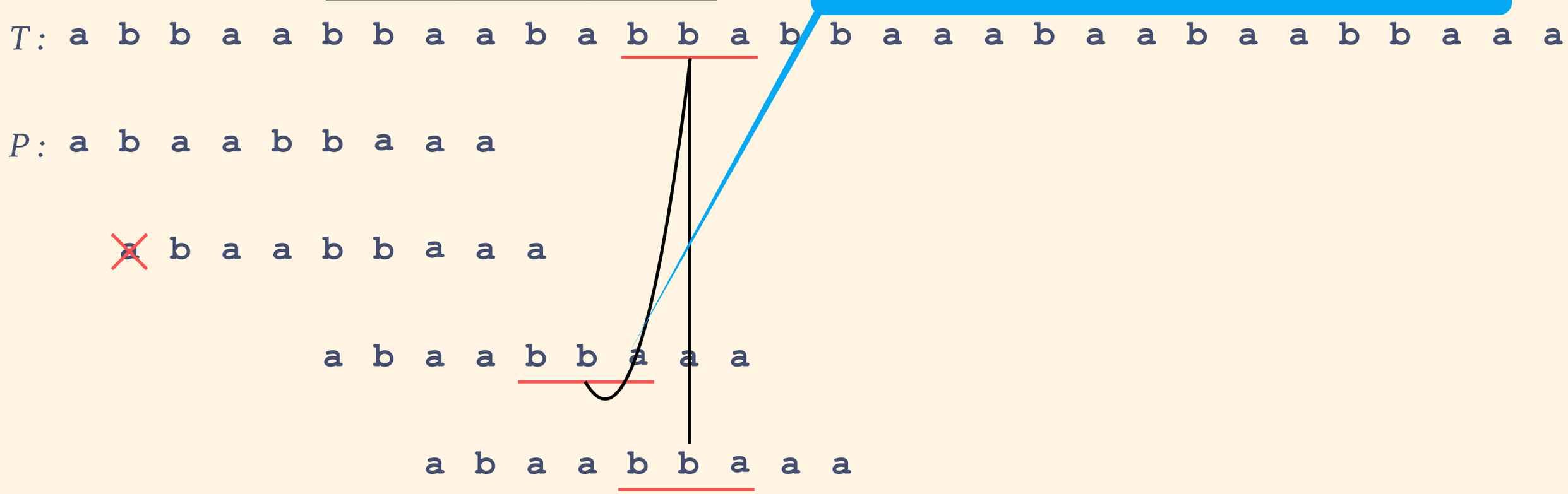
Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “bba” by shifting with array HQ_Shift until P[1] matches the corresponding text



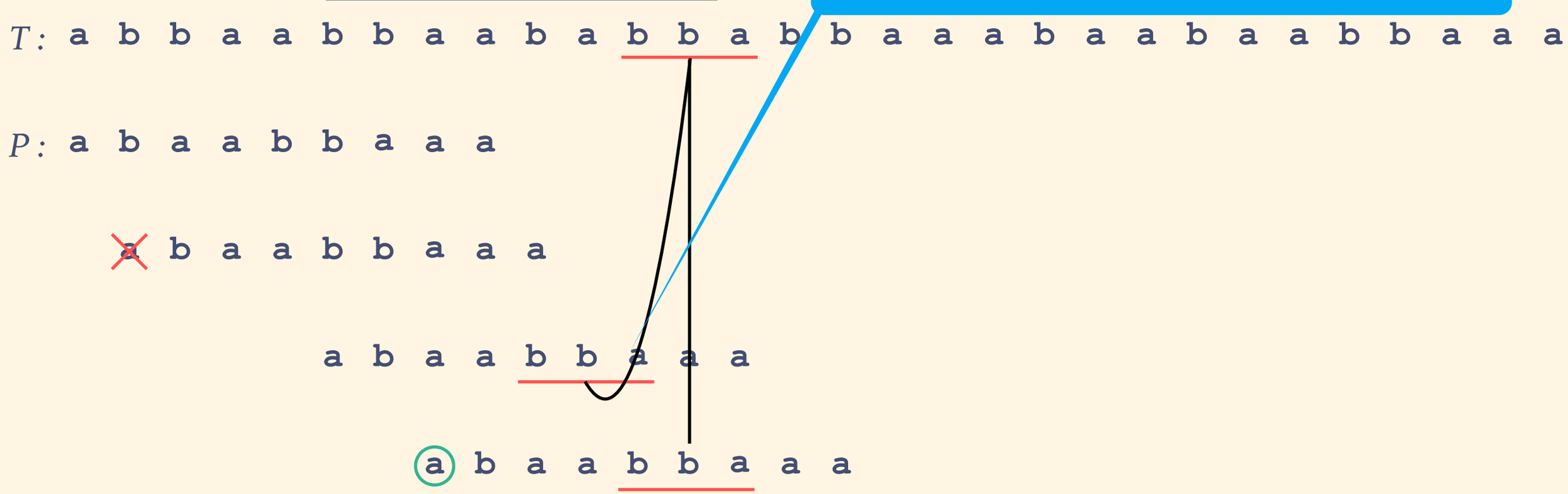
Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase
Align “bba” by shifting with array HQ_Shift until P[1] matches the corresponding text



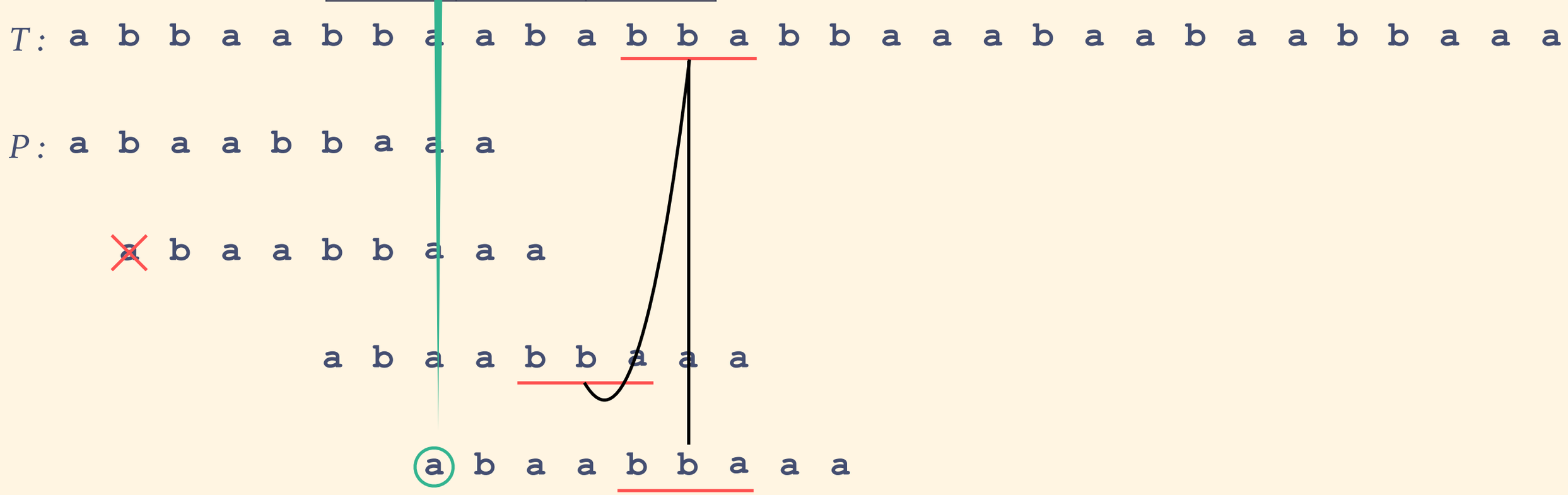
- Match
- Mismatch
- Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Comparison-Phase
compare P[2:m] from left to right if the first letter match



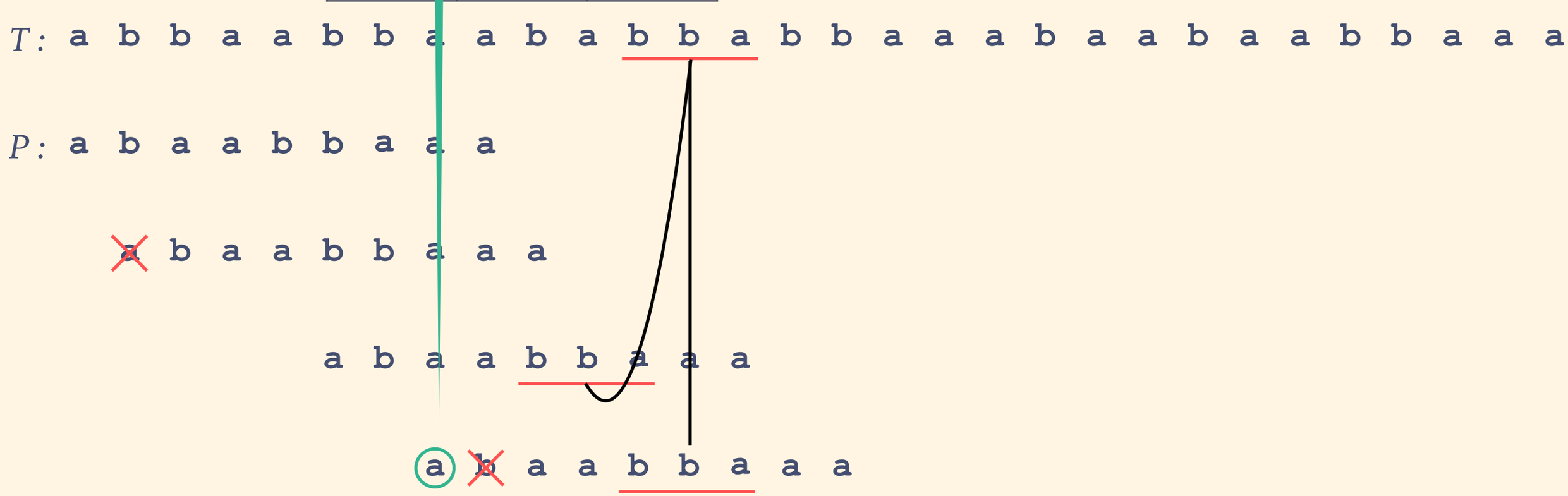
Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Comparison-Phase
compare P[2:m] from left to right if the first letter match

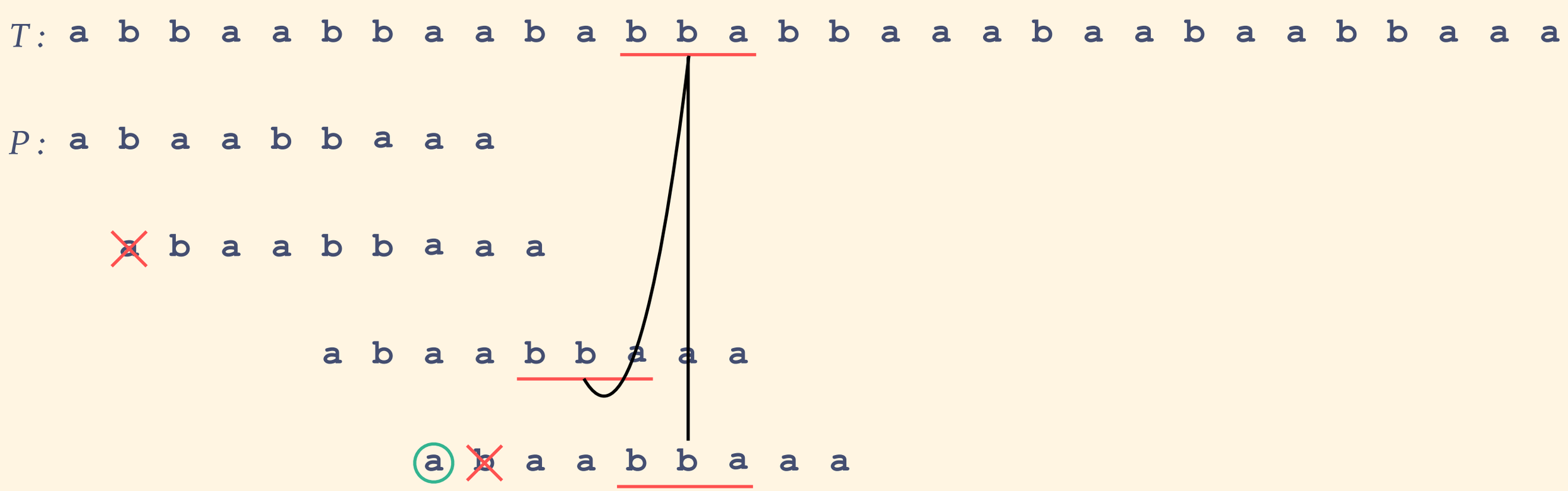


Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

~~a~~ b a a b b a a a

a b a a b b a a

~~a~~ ~~b~~ a a b b a a a

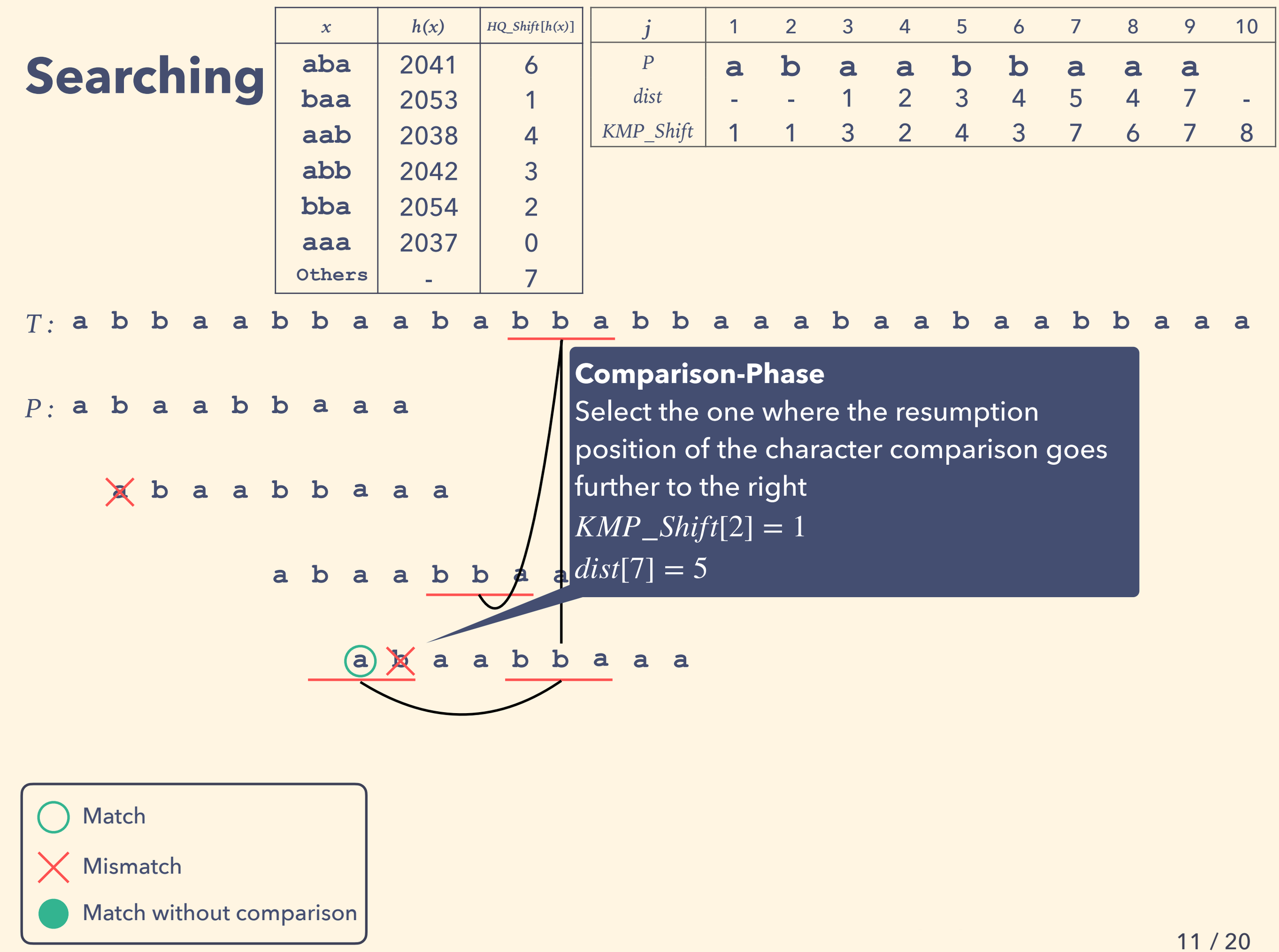
Comparison-Phase
Select the one where the resumption position of the character comparison goes further to the right
$KMP\_Shift[2] = 1$
$dist[7] = 5$

- Match
- Mismatch
- Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

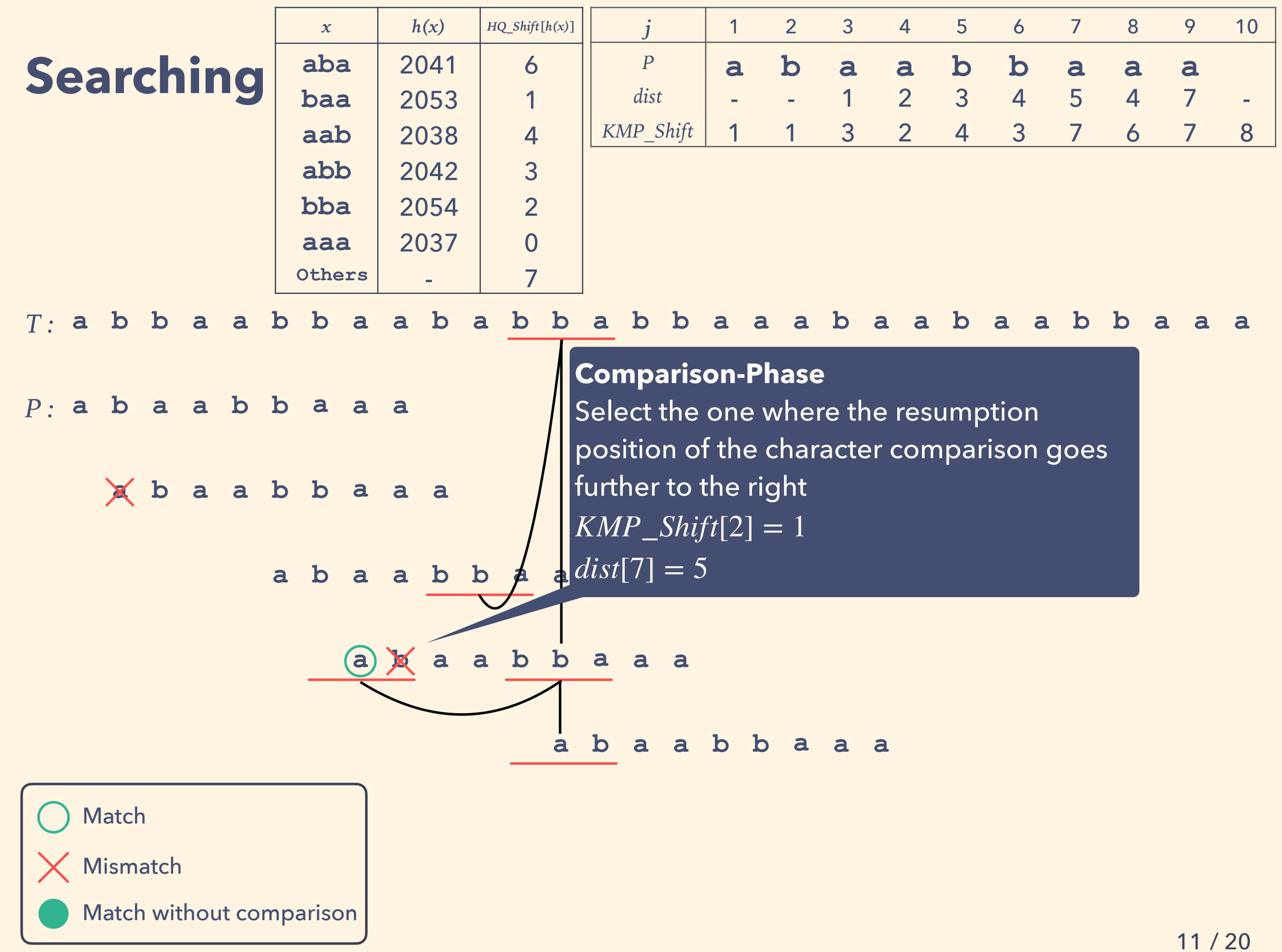
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

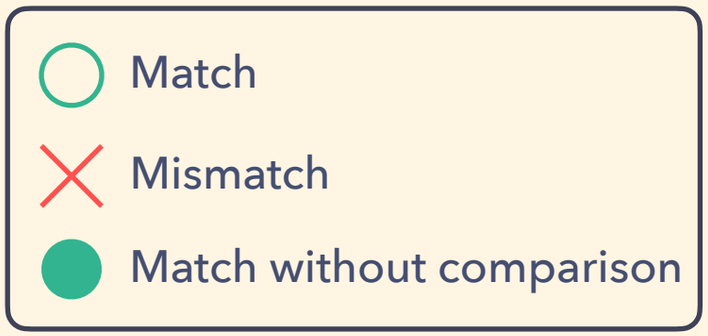
T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a



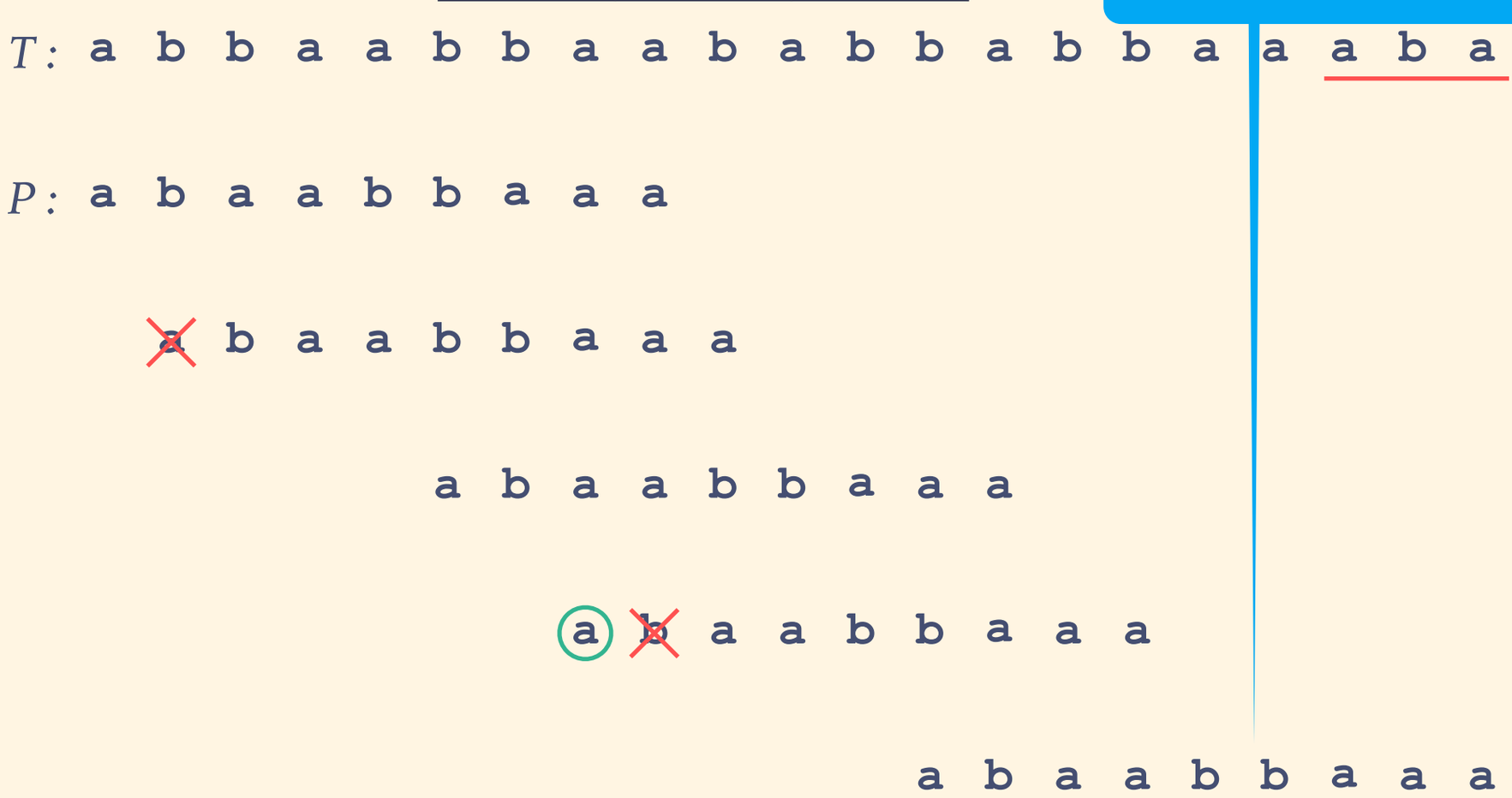
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase

Align “aba” by shifting with array HQ_Shift until P[1] matches the corresponding text character



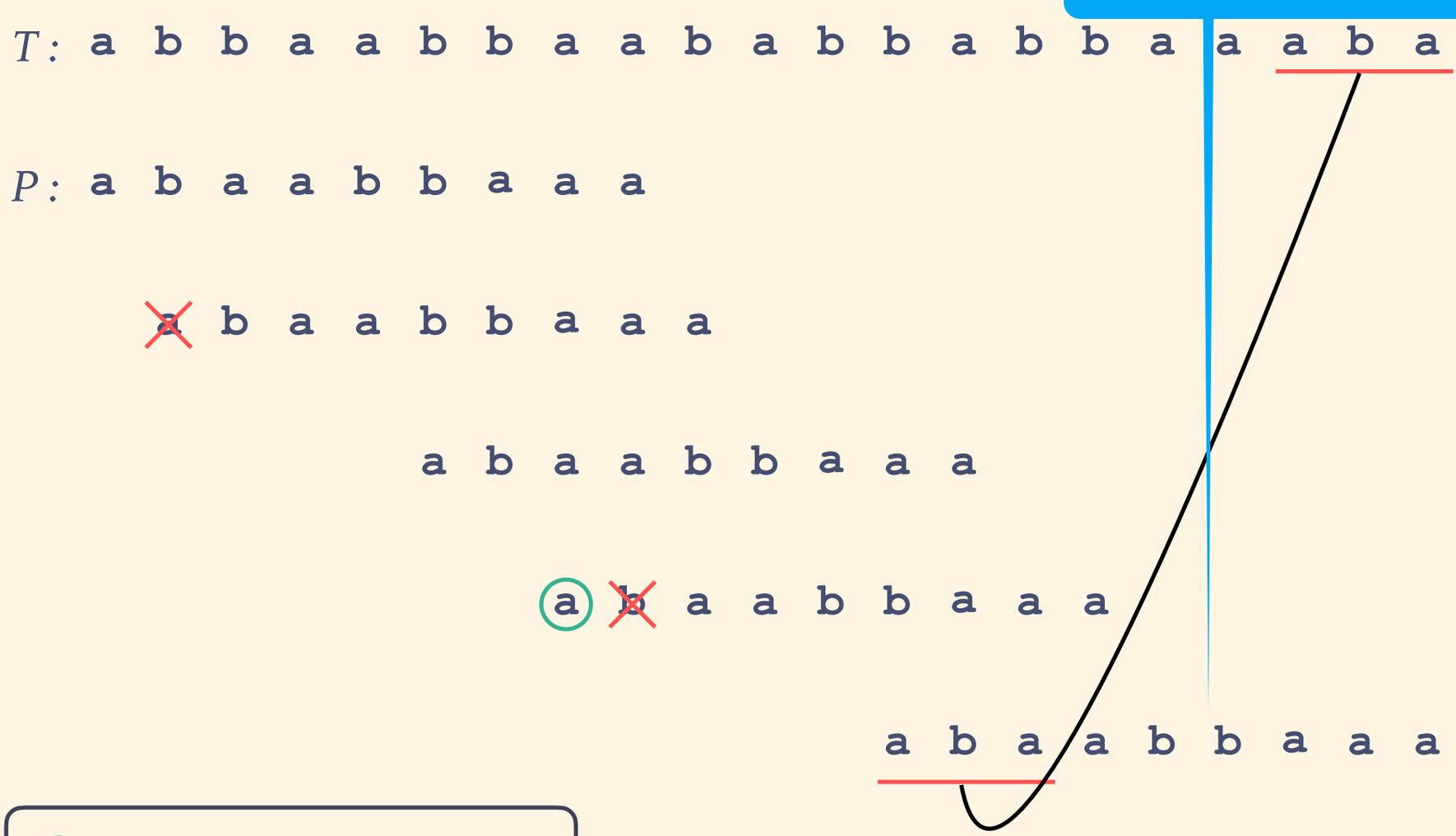
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase

Align “aba” by shifting with array HQ_Shift until P[1] matches the corresponding text character



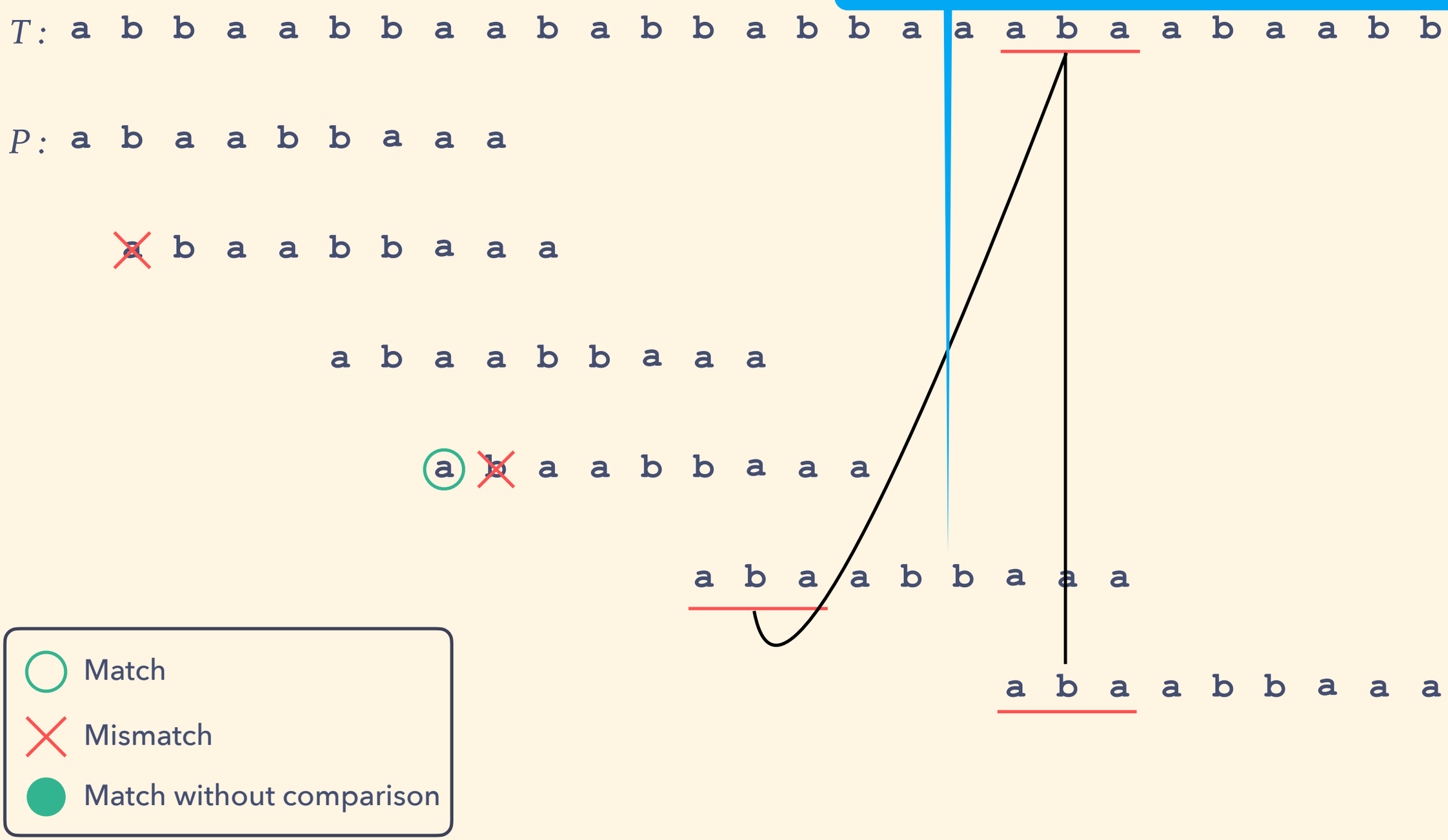
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase

Align “aba” by shifting with array HQ_Shift until P[1] matches the corresponding text character



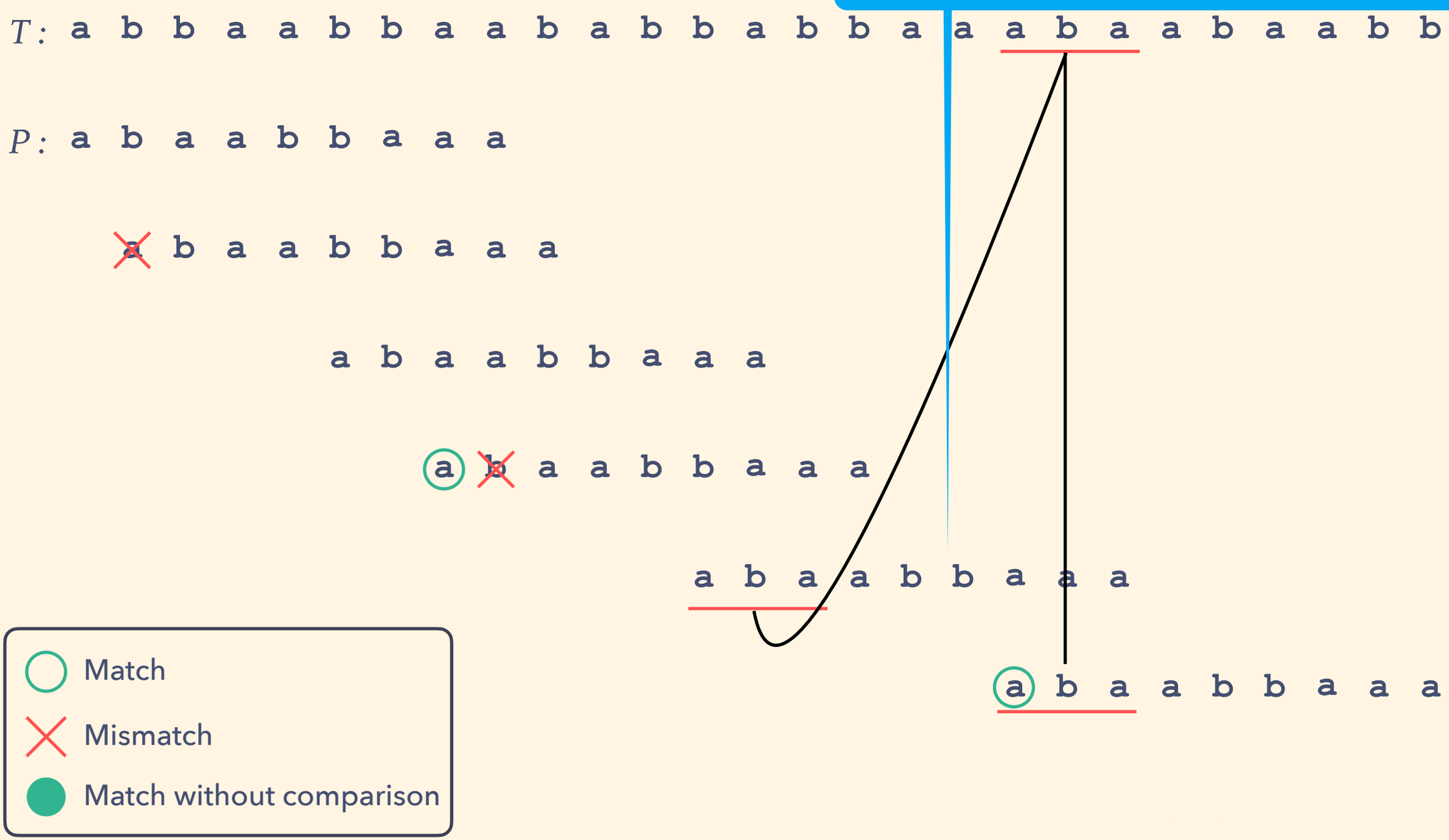
Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

Alignment-Phase

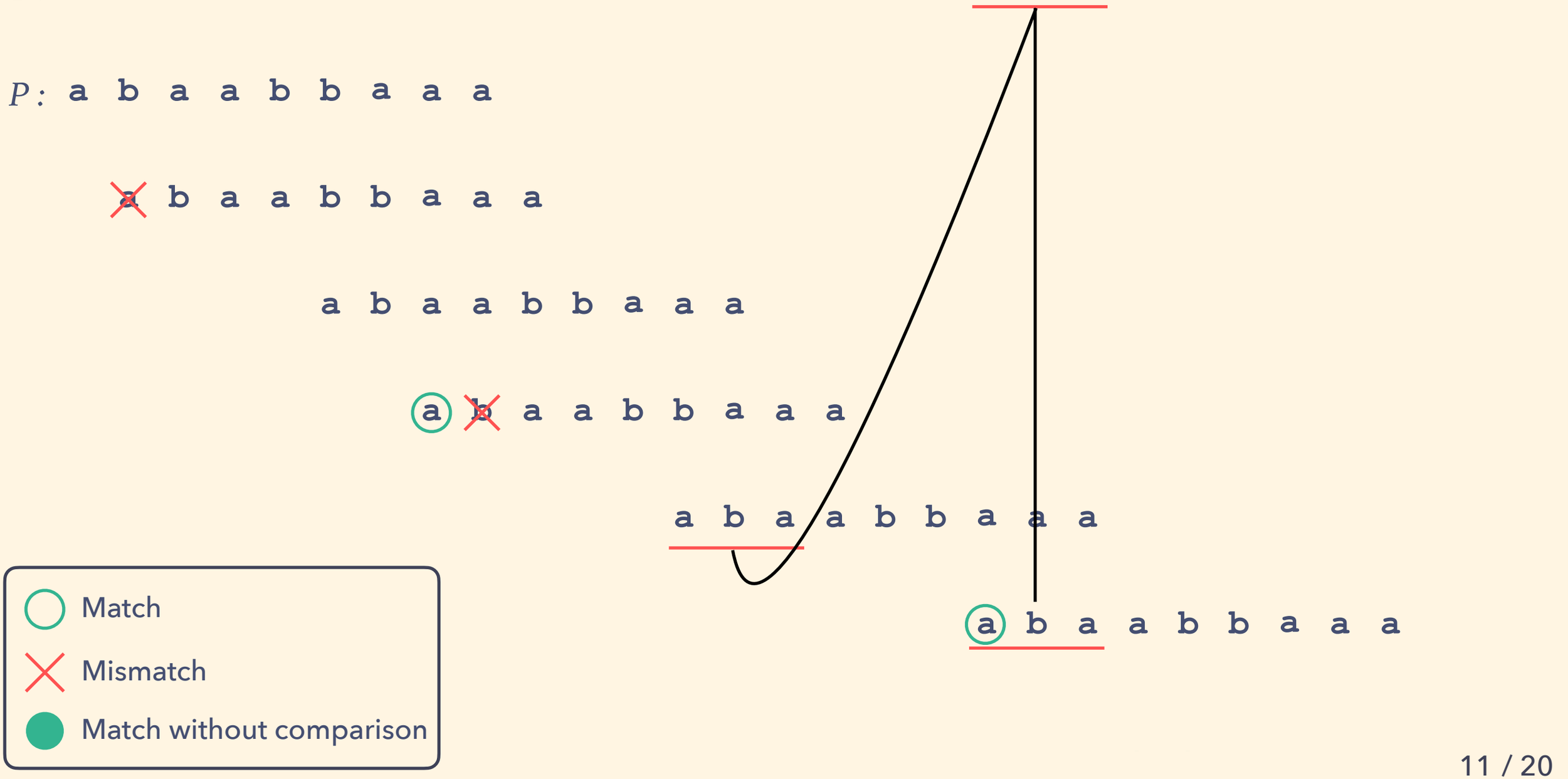
Align “aba” by shifting with array HQ_Shift until P[1] matches the corresponding text character



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

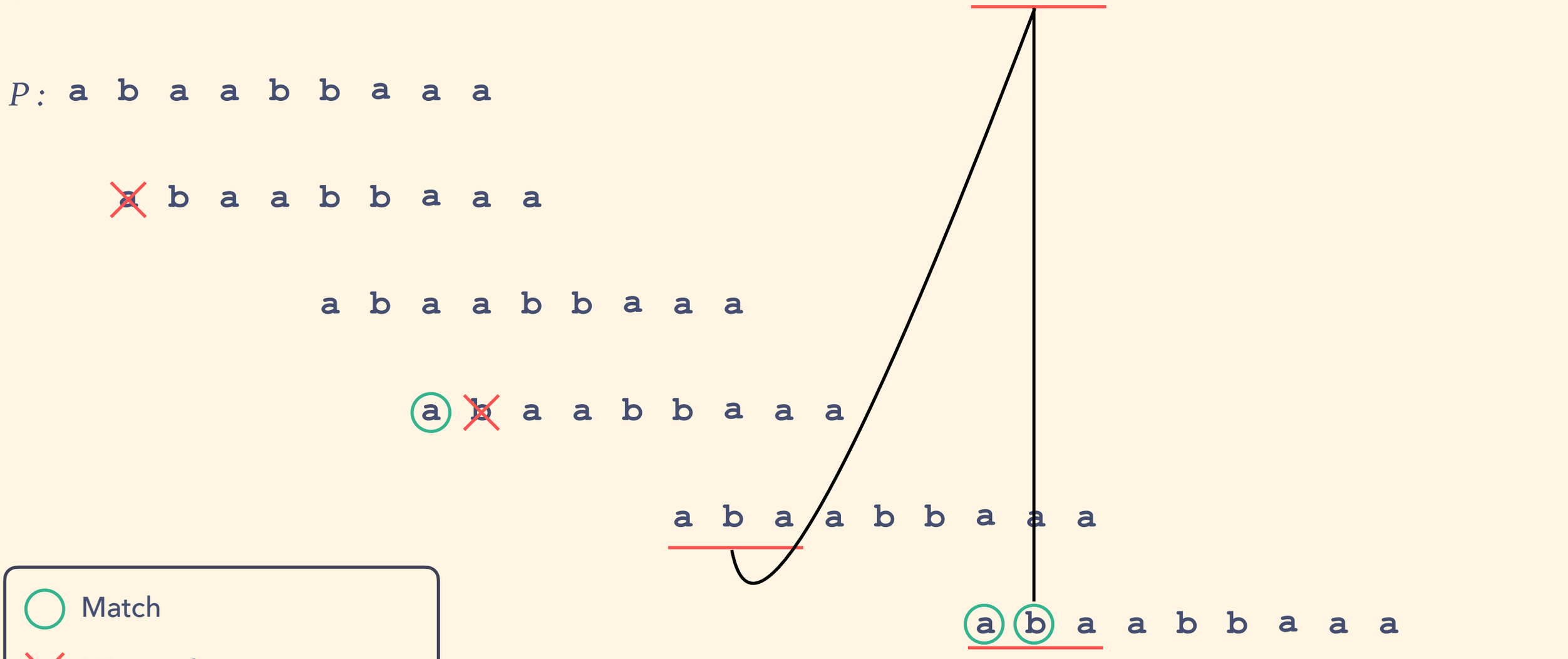
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

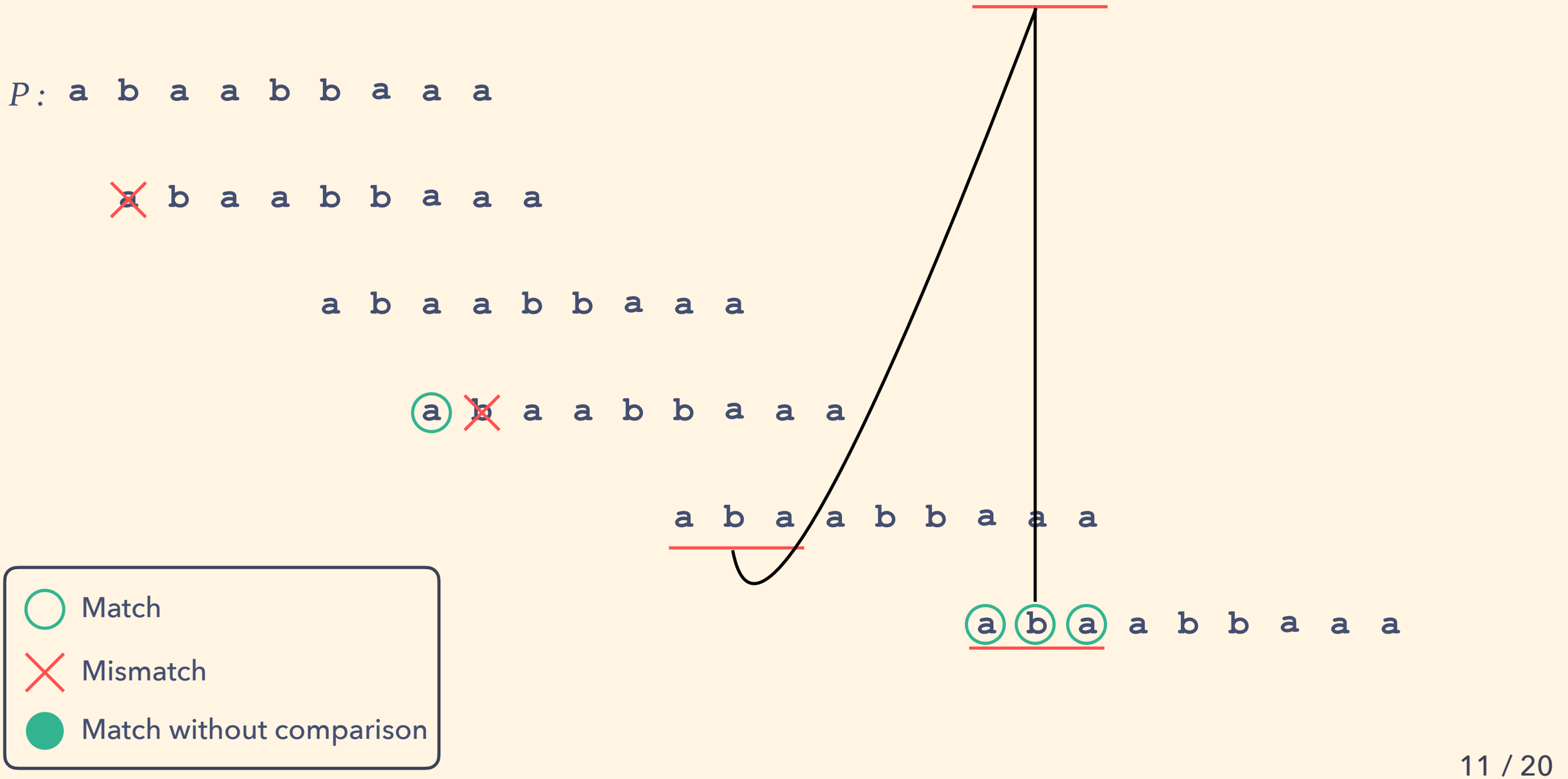
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

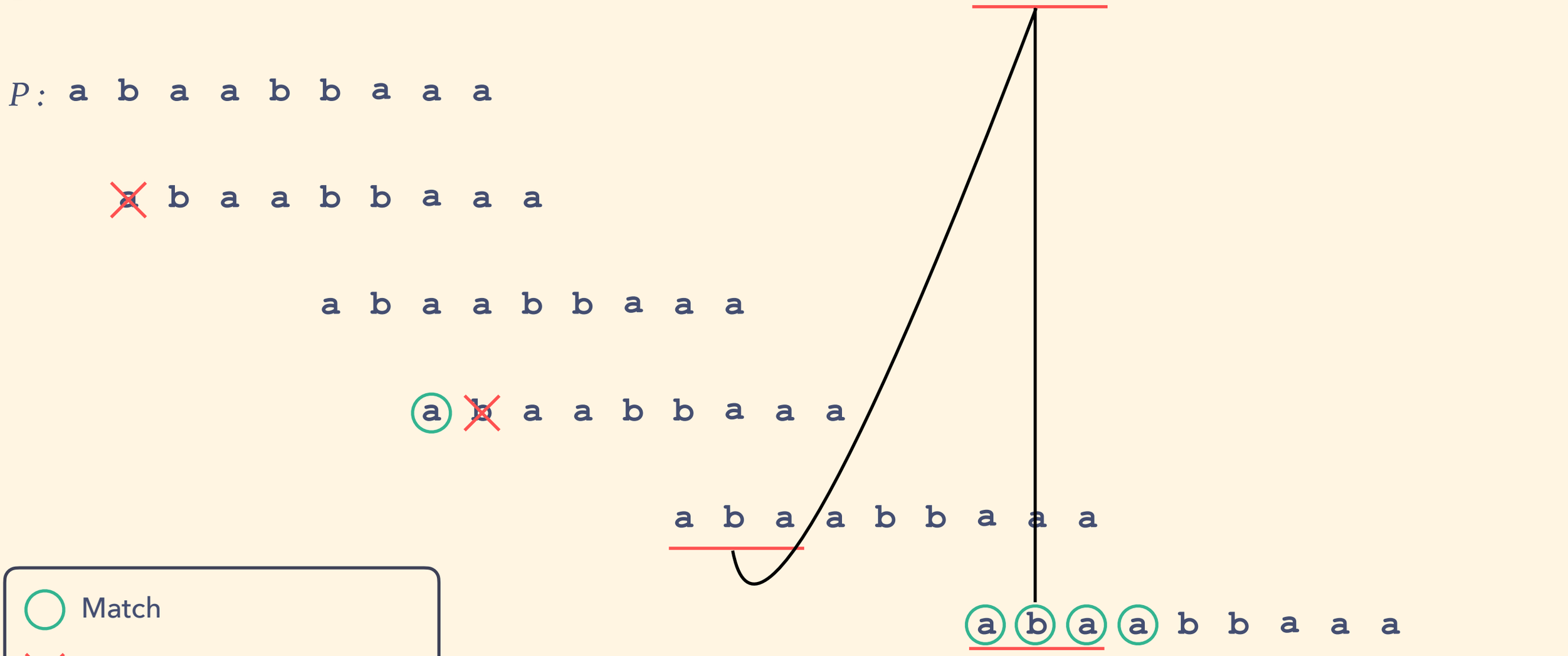
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

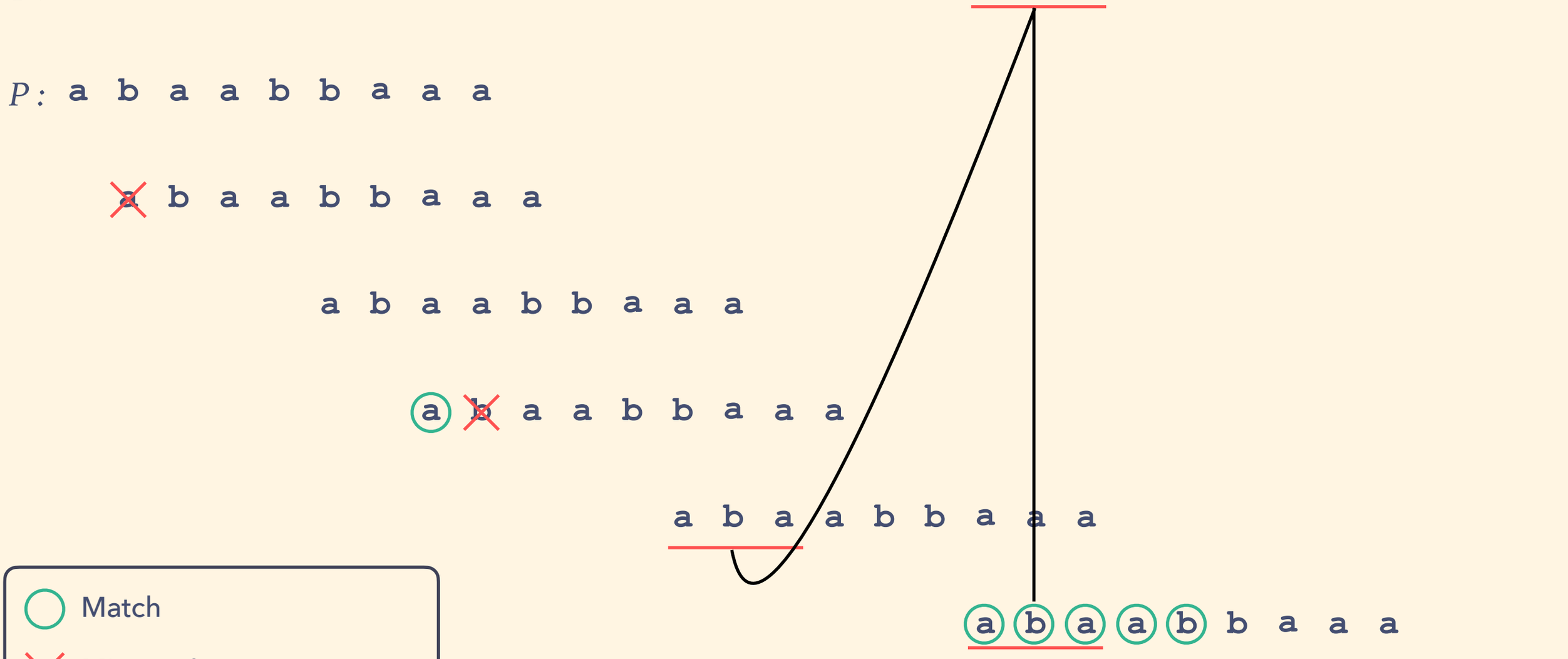
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

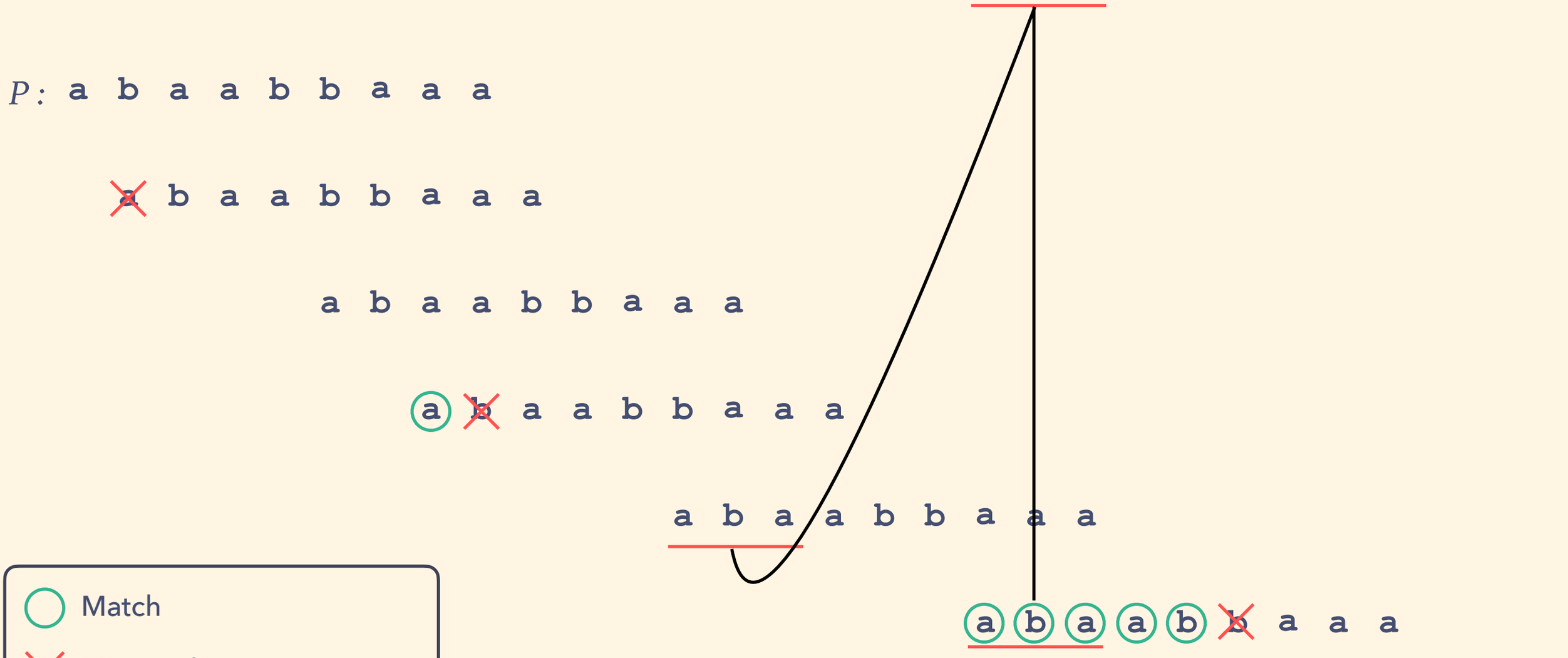
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

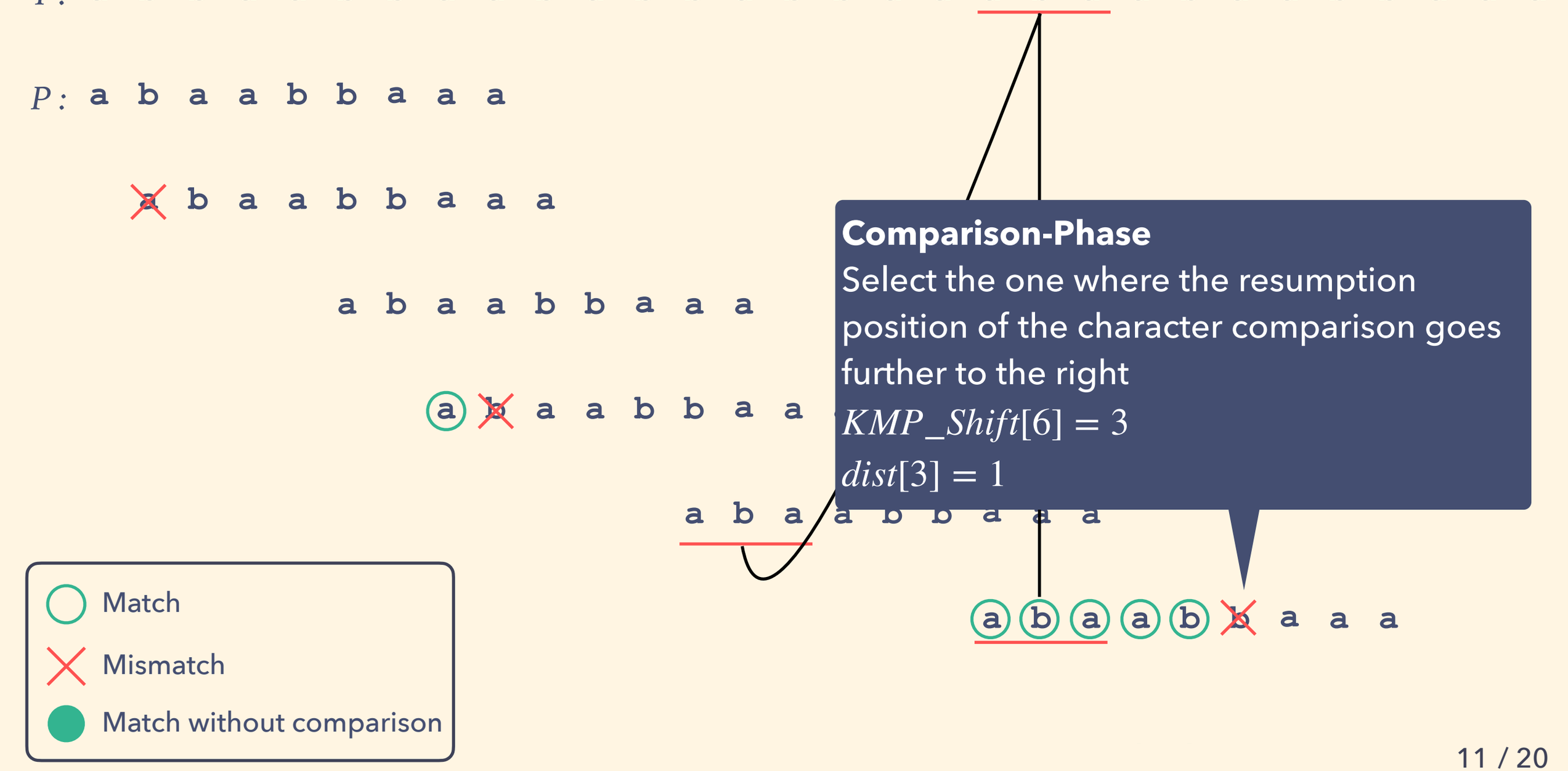
j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Comparison-Phase
Select the one where the resumption position of the character comparison goes further to the right
$KMP\_Shift[6] = 3$
$dist[3] = 1$

a b a a b b a a a

a a b b a a a

Match
Mismatch
Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

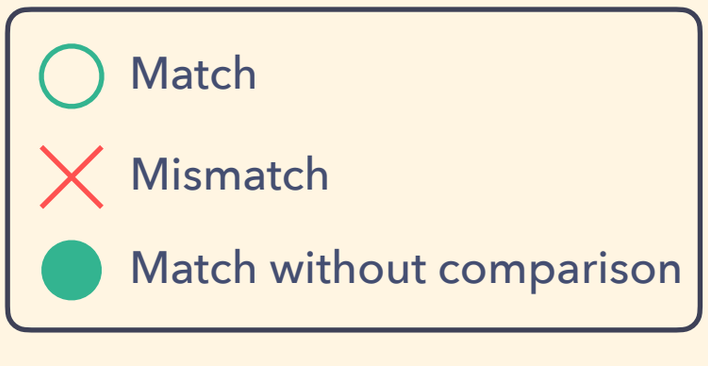
a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

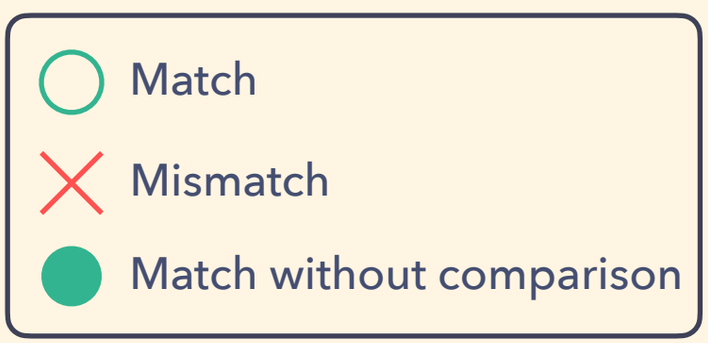
a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a



Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

$T:$ a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

$P:$ a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Match

Mismatch

Match without comparison

Searching

x	$h(x)$	$HQ\_Shift[h(x)]$
aba	2041	6
baa	2053	1
aab	2038	4
abb	2042	3
bba	2054	2
aaa	2037	0
Others	-	7

j	1	2	3	4	5	6	7	8	9	10
P	a	b	a	a	b	b	a	a	a	
$dist$	-	-	1	2	3	4	5	4	7	-
KMP_Shift	1	1	3	2	4	3	7	6	7	8

T: a b b a a b b a a b a b b a b b a a a b a a b a a b b a a a

P: a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

a b a a b b a a a

Match

Mismatch

Match without comparison

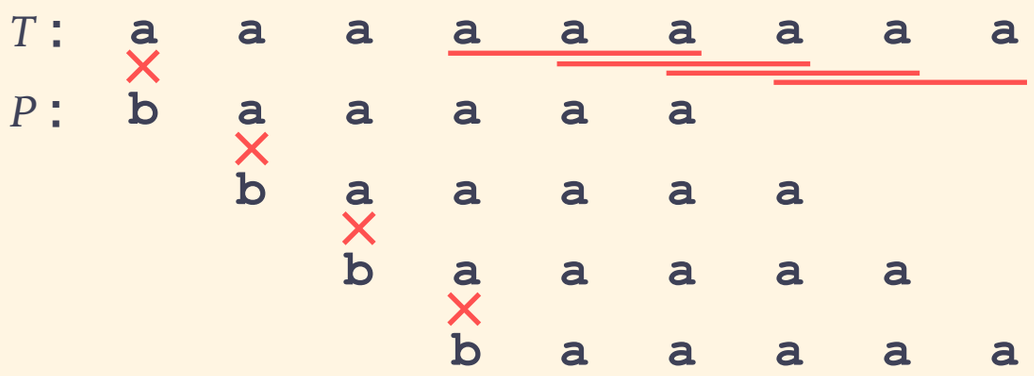
Time complexity of DISTq algorithm

- **Preprocessing**
  - Array KMP_Shift : $O(m)$
  - Array HQ_Shift : $O(mq)$
    - Calculate hash value that takes $O(q)$ time at $m - q + 1$ positions
  - Array $dist$: $O(mq)$

  ⎫ $O(mq)$

- **Searching**
  - Number of character comparisons : $O(n)$
  - Calculate hash value at maximum $n - m + 1$ positions : $O(nq)$

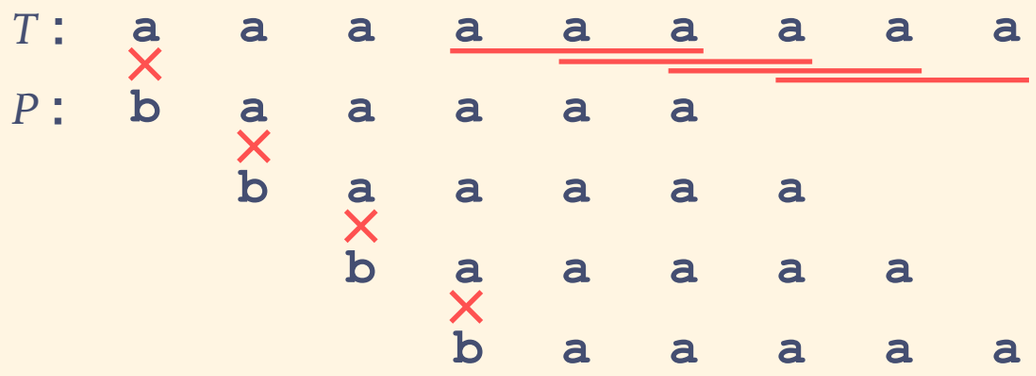
  ⎫ $O(nq)$



Proposed 2
Linear-DIST$_q$ (LDIST$_q$)
algorithm

LDISTq algorithm

- Worst-case time complexity of the search phase of DISTq algorithm is $\Theta(nq)$



- Since the hash function h used in DISTq algorithm is a rolling hash, when $h(T[i:j])$ has already been obtained, the hash value of $h(T[i+1:j+1])$ can be computed in $O(1)$ time $(1 \leq i \leq j < |T|)$

  - $h(T[i+1:j+1]) = (4 \cdot (h(T[i:j]) - 16 \cdot T[i]) + T[j+1]) \bmod 2^{16}$

    $h(x[1:3]) = (16 \cdot x[1] + 4 \cdot x[2] + x[3]) \bmod 2^{16}$

- Searching time can be reduced to $O(n)$

  - by incrementally calculating the hash value of the q-gram using the previously calculated value of the other q-gram

  - Preprocessing time is also reduced to $O(m)$

Experiments

Datasets

- English text
- Genome sequence
- Fibonacci string
- Texts with frequent pattern occurrences

- Implemented with C language
- Compiled with GCC9.2.0
- MacBook Pro (13-inch, 2018), macOS Catalina, Intel Core i7 2.7GHz quad core, 16GB memory

English text

In the beginning,

- Use The King James version of the Bible as text
  - $n = 4017009$
  - $\sigma = 62$
- Patterns are randomly extracted from text

$n = |T|$　$m = |P|$　$\sigma = |\Sigma|$: Alphabet size

	m	2	4	8	16	32	64	128	256	512	1024
	BNDMq	$140.48^{(2)}$	$93.39^{(2)}$	$70.84^{(4)}$	$54.10^{(4)}$	$\mathbf{45.88}^{(4)}$	$47.61^{(4)}$	$47.64^{(4)}$	$46.88^{(4)}$	$47.40^{(6)}$	$45.58^{(4)}$
	SBNDMq	$\mathbf{108.32}^{(2)}$	$\mathbf{73.50}^{(2)}$	$\mathbf{64.87}^{(2)}$	$\mathbf{50.73}^{(4)}$	$47.85^{(4)}$	$48.45^{(4)}$	$48.55^{(4)}$	$47.56^{(4)}$	$47.08^{(4)}$	$45.98^{(4)}$
	KBNDM	192.76	126.56	92.03	71.04	64.17	56.79	49.56	49.09	50.24	47.68
	BSDMq	$117.22^{(2)}$	$79.96^{(2)}$	$70.36^{(3)}$	$57.72^{(4)}$	$49.91^{(8)}$	$48.00^{(8)}$	$44.99^{(8)}$	$43.62^{(8)}$	$43.06^{(8)}$	$42.70^{(8)}$
★	FJS	192.52	158.61	106.40	89.04	78.43	68.14	64.80	61.15	60.48	55.38
★	FJS+	196.88	155.86	101.77	77.74	67.40	60.31	54.77	52.44	51.62	47.74
	HASHq	$312.91^{(2)}$	$218.95^{(2)}$	$107.38^{(2)}$	$70.85^{(2)}$	$56.90^{(3)}$	$49.33^{(6)}$	$46.63^{(5)}$	$45.27^{(8)}$	$44.77^{(3)}$	$42.98^{(7)}$
	FS-w	$129.50^{(6)}$	$104.45^{(6)}$	$71.57^{(6)}$	$61.08^{(4)}$	$54.19^{(6)}$	$49.53^{(4)}$	$48.95^{(6)}$	$47.02^{(4)}$	$46.96^{(6)}$	$43.00^{(4)}$
	IOM	193.37	148.51	104.36	86.22	75.09	69.08	63.63	60.62	58.83	51.76
	WOM	199.23	153.81	112.45	86.61	75.26	62.80	60.54	62.74	58.91	55.65
	SKIPq	$161.38^{(2)}$	$100.52^{(2)}$	$72.33^{(3)}$	$55.71^{(4)}$	$49.06^{(4)}$	$48.73^{(4)}$	$49.87^{(4)}$	$48.81^{(8)}$	$45.75^{(2)}$	$45.32^{(2)}$
	WFRq	$137.40^{(2)}$	$85.22^{(2)}$	$68.88^{(2)}$	$52.85^{(5)}$	$46.67^{(5)}$	$\mathbf{44.39}^{(8)}$	$\mathbf{42.09}^{(8)}$	$41.66^{(5)}$	$41.65^{(6)}$	$40.53^{(2)}$
★	LWFRq	$121.08^{(2)}$	$85.47^{(2)}$	$70.38^{(2)}$	$53.83^{(3)}$	$47.89^{(5)}$	$44.49^{(5)}$	$42.38^{(6)}$	$\mathbf{41.09}^{(8)}$	$\mathbf{41.23}^{(8)}$	$\mathbf{40.30}^{(8)}$
★	DISTq	$115.61^{(2)}$	$80.14^{(2)}$	$65.84^{(3)}$	$52.25^{(4)}$	$48.13^{(4)}$	$46.26^{(4)}$	$43.32^{(4)}$	$42.84^{(8)}$	$42.98^{(4)}$	$41.47^{(7)}$
★	LDISTq	$229.62^{(2)}$	$102.15^{(2)}$	$69.70^{(3)}$	$55.34^{(4)}$	$49.46^{(5)}$	$45.93^{(5)}$	$44.37^{(3)}$	$43.15^{(4)}$	$42.21^{(5)}$	$41.11^{(7)}$

algorithms that run in linear time in the input string size are marked with ★

Unit : millisecond　The value of q or w giving the best performance is shown in round brackets

Genome sequence

- Genome sequence of E. coli
  - $n = 4641652$
  - $\sigma = 4$
- Patterns are randomly extracted from text

$n = |T|$　$m = |P|$　$\sigma = |\Sigma|$: Alphabet size

	m	2	4	8	16	32	64	128	256	512	1024
	BNDMq	$218.34^{(2)}$	$174.55^{(2)}$	$84.14^{(4)}$	$64.89^{(6)}$	$60.39^{(6)}$	$61.07^{(6)}$	$60.85^{(6)}$	$62.17^{(6)}$	$61.20^{(6)}$	$60.16^{(6)}$
	SBNDMq	$\mathbf{179.90^{(2)}}$	$154.20^{(2)}$	$80.94^{(4)}$	$73.29^{(6)}$	$57.10^{(8)}$	$61.01^{(6)}$	$61.74^{(6)}$	$61.20^{(6)}$	$61.67^{(6)}$	$60.80^{(6)}$
	KBNDM	311.78	201.99	150.15	113.84	83.23	67.83	75.65	75.39	76.58	74.72
	BSDMq	$195.38^{(2)}$	$\mathbf{118.86^{(3)}}$	$84.23^{(5)}$	$63.03^{(6)}$	$61.26^{(6)}$	$58.75^{(6)}$	$57.50^{(7)}$	$56.99^{(6)}$	$57.01^{(6)}$	$56.58^{(6)}$
★	FJS	407.02	353.60	311.96	279.13	308.42	297.00	266.12	317.79	317.34	296.19
★	FJS+	388.44	296.70	203.03	171.17	149.52	136.59	128.55	130.39	122.51	112.99
	HASHq	$571.44^{(2)}$	$272.86^{(3)}$	$126.00^{(3)}$	$88.12^{(3)}$	$68.22^{(3)}$	$58.84^{(6)}$	$55.09^{(6)}$	$59.48^{(7)}$	$57.34^{(7)}$	$57.96^{(7)}$
	FS-w	$332.32^{(4)}$	$245.99^{(4)}$	$184.72^{(4)}$	$158.72^{(4)}$	$143.79^{(6)}$	$125.05^{(4)}$	$123.52^{(6)}$	$117.90^{(6)}$	$108.03^{(6)}$	$100.72^{(6)}$
	IOM	377.25	275.36	215.72	220.97	219.86	218.12	210.61	221.31	230.15	211.69
	WOM	381.54	301.46	220.34	182.30	166.27	143.24	136.20	133.75	127.40	114.74
	SKIPq	$250.89^{(2)}$	$136.18^{(3)}$	$91.51^{(4)}$	$63.96^{(6)}$	$56.79^{(7)}$	$53.09^{(7)}$	$52.10^{(7)}$	$57.12^{(7)}$	$56.97^{(8)}$	$58.00^{(6)}$
	WFRq	$219.50^{(2)}$	$168.39^{(2)}$	$88.86^{(4)}$	$65.82^{(4)}$	$57.19^{(8)}$	$55.12^{(2)}$	$51.77^{(3)}$	$50.04^{(3)}$	$55.02^{(6)}$	$\mathbf{54.64^{(8)}}$
★	LWFRq	$216.10^{(2)}$	$173.48^{(3)}$	$88.71^{(4)}$	$60.75^{(5)}$	$\mathbf{53.84^{(5)}}$	$\mathbf{50.48^{(6)}}$	$\mathbf{49.65^{(8)}}$	$48.71^{(6)}$	$54.90^{(6)}$	$54.97^{(7)}$
★	DISTq	$186.10^{(2)}$	$125.44^{(3)}$	$\mathbf{78.56^{(4)}}$	$\mathbf{60.48^{(5)}}$	$55.21^{(6)}$	$52.05^{(7)}$	$51.26^{(8)}$	$50.44^{(8)}$	$54.81^{(7)}$	$55.58^{(8)}$
★	LDISTq	$295.55^{(2)}$	$181.99^{(3)}$	$86.58^{(4)}$	$65.29^{(6)}$	$56.74^{(6)}$	$52.31^{(6)}$	$50.39^{(6)}$	$\mathbf{48.70^{(7)}}$	$\mathbf{54.33^{(4)}}$	$55.22^{(7)}$

algorithms that run in linear time in the input string size are marked with ★

Unit : millisecond　The value of q or w giving the best performance is shown in round brackets

Fibonacci string

abaababaabaababaa

- Definition

  $Fib_1 = \mathtt{b}, \quad Fib_2 = \mathtt{a}, \quad Fib_n = Fib_{n-1} \cdot Fib_{n-2}$ for $n > 2$

- Use Fib_{32} as text
  - $n = 2178309, \quad \sigma = 2$
- Patterns are randomly extracted from text

$n = |T|$ $m = |P|$ $\sigma = |\Sigma|$: Alphabet size

	m	2	4	8	16	32	64	128	256	512	1024
	BNDMq	$343.25^{(2)}$	$308.44^{(2)}$	$283.26^{(4)}$	$257.64^{(6)}$	$233.94^{(6)}$	$285.63^{(4)}$	$284.37^{(4)}$	$293.00^{(4)}$	$307.82^{(4)}$	$315.47^{(4)}$
	SBNDMq	**$286.02^{(2)}$**	**$292.15^{(2)}$**	$272.98^{(4)}$	$276.35^{(6)}$	$306.42^{(6)}$	$372.03^{(6)}$	$432.53^{(6)}$	$493.20^{(6)}$	$546.94^{(6)}$	$602.09^{(6)}$
	KBNDM	541.70	405.78	411.08	422.85	382.25	402.45	425.60	437.67	461.07	451.12
	BSDMq	$482.47^{(2)}$	$500.43^{(3)}$	$397.29^{(5)}$	$362.52^{(8)}$	$330.76^{(6)}$	$736.89^{(1)}$	$766.57^{(1)}$	$782.98^{(1)}$	$790.26^{(1)}$	$508.80^{(3)}$
★	FJS	402.44	362.23	276.97	237.87	218.38	**206.07**	203.86	202.94	196.49	194.01
★	FJS+	456.20	396.04	335.70	319.93	295.64	300.36	296.48	295.37	288.56	289.00
	HASHq	$645.48^{(2)}$	$406.70^{(2)}$	**$257.69^{(4)}$**	$251.91^{(7)}$	$279.81^{(7)}$	$344.99^{(7)}$	$415.16^{(7)}$	$470.71^{(7)}$	$514.05^{(7)}$	$579.57^{(7)}$
	FS-w	$383.23^{(1)}$	$396.23^{(1)}$	$347.72^{(1)}$	$289.15^{(1)}$	$253.36^{(1)}$	$246.82^{(1)}$	$248.73^{(1)}$	$235.66^{(1)}$	$235.35^{(1)}$	$230.61^{(1)}$
	IOM	381.92	414.42	453.54	497.84	543.93	641.13	751.42	839.92	899.19	1019.59
	WOM	552.38	555.43	564.67	617.93	664.47	732.05	852.06	926.35	1036.31	1126.17
	SKIPq	$470.93^{(2)}$	$394.09^{(2)}$	$332.66^{(5)}$	$336.16^{(8)}$	$374.91^{(8)}$	$464.23^{(3)}$	$460.54^{(3)}$	$450.18^{(3)}$	$451.05^{(3)}$	$464.13^{(3)}$
	WFRq	$442.32^{(2)}$	$497.95^{(3)}$	$528.45^{(3)}$	$652.48^{(6)}$	$2132.38^{(7)}$	$3762.19^{(8)}$	$6762.67^{(8)}$	$12624.63^{(8)}$	$24416.65^{(8)}$	$48596.02^{(8)}$
★	LWFRq	$552.64^{(2)}$	$504.77^{(3)}$	$428.34^{(5)}$	$342.80^{(7)}$	$297.07^{(7)}$	$304.57^{(2)}$	$274.47^{(6)}$	$265.28^{(6)}$	$258.06^{(6)}$	$254.92^{(6)}$
★	DISTq	$438.96^{(1)}$	$359.56^{(2)}$	$293.80^{(2)}$	**$235.61^{(2)}$**	**$213.07^{(7)}$**	$206.92^{(2)}$	$201.18^{(8)}$	$196.56^{(5)}$	$194.86^{(4)}$	$193.62^{(5)}$
★	LDISTq	$565.13^{(2)}$	$412.15^{(3)}$	$300.98^{(4)}$	$245.38^{(7)}$	$215.89^{(8)}$	$208.40^{(7)}$	**$200.45^{(4)}$**	**$193.74^{(4)}$**	**$190.85^{(3)}$**	**$193.48^{(8)}$**

algorithms that run in linear time in the input string size are marked with ★

Unit : millisecond The value of q or w giving the best performance is shown in round brackets

In experiment of Fibonacci string

$Fib_{10} =$
abaababaabaababaababaabaababaabaababaababa

$P =$ **baaba**

$Fib_{10} =$
abaababaabaababaababaabaababaabaababaababa

$P =$ **aababaab**

- There are many repeating structures
- The pattern is extracted from the text, so the number of pattern occurrences is very large

Hypothesize

Efficiency of proposed algorithms do not decrease when number of pattern occurrences is large

Texts with frequent pattern occurrences

- Generated by intentionally embedding a lot of randomly generated patterns without overlapping
  - $n = 4000000$
- Fixed pattern length

$m = 8, \sigma = 4$　occ : Pattern occurrences　　$n = |T|$　$m = |P|$　$\sigma = |\Sigma|$: Alphabet size

occ	0	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072
BNDMq	$73.53^{(4)}$	$72.28^{(4)}$	$72.87^{(4)}$	$73.89^{(4)}$	$74.39^{(4)}$	$75.75^{(4)}$	$77.60^{(4)}$	$81.63^{(4)}$	$82.67^{(4)}$	$88.98^{(4)}$	$108.56^{(4)}$	$146.13^{(6)}$
SBNDMq	**$68.58^{(4)}$**	**$68.92^{(4)}$**	$71.17^{(4)}$	$77.26^{(4)}$	$81.04^{(4)}$	$80.76^{(4)}$	$78.33^{(4)}$	$82.82^{(4)}$	$91.01^{(4)}$	$96.83^{(4)}$	$111.75^{(4)}$	$140.21^{(4)}$
KBNDM	127.73	128.41	128.41	126.92	128.74	131.17	129.74	135.70	140.99	152.46	164.59	186.32
BSDMq	$69.85^{(4)}$	$69.04^{(4)}$	**$68.85^{(4)}$**	**$69.19^{(4)}$**	**$70.63^{(4)}$**	**$72.00^{(4)}$**	**$72.48^{(4)}$**	**$76.25^{(4)}$**	**$79.95^{(4)}$**	$89.91^{(4)}$	$110.22^{(4)}$	$143.91^{(4)}$
★ FJS	246.26	253.35	263.94	247.73	255.94	276.44	262.83	256.36	251.33	269.39	259.93	251.06
★ FJS+	174.38	173.86	180.46	173.05	178.65	182.98	177.56	181.17	182.48	190.71	197.69	199.60
HASHq	$121.09^{(3)}$	$121.36^{(3)}$	$122.64^{(3)}$	$122.50^{(3)}$	$119.49^{(3)}$	$123.98^{(3)}$	$124.31^{(3)}$	$124.93^{(3)}$	$126.81^{(3)}$	$128.94^{(3)}$	$143.75^{(3)}$	$153.08^{(4)}$
FS-w	$154.89^{(4)}$	$155.56^{(4)}$	$165.18^{(4)}$	$156.67^{(4)}$	$159.03^{(4)}$	$166.44^{(4)}$	$165.62^{(4)}$	$160.46^{(4)}$	$167.46^{(4)}$	$178.53^{(2)}$	$185.86^{(2)}$	$191.38^{(2)}$
IOM	185.31	181.07	195.53	187.98	194.18	200.99	195.98	195.89	197.67	202.61	214.90	209.52
WOM	192.59	192.28	207.57	189.21	196.02	203.86	196.98	199.79	203.59	215.79	219.94	230.59
SKIPq	$79.32^{(4)}$	$77.14^{(4)}$	$79.19^{(4)}$	$82.08^{(4)}$	$81.82^{(4)}$	$82.55^{(4)}$	$83.83^{(4)}$	$87.08^{(4)}$	$89.82^{(4)}$	$93.09^{(4)}$	$106.41^{(4)}$	$126.22^{(3)}$
WFRq	$76.68^{(4)}$	$76.38^{(4)}$	$77.63^{(4)}$	$83.21^{(4)}$	$80.93^{(4)}$	$81.42^{(4)}$	$83.86^{(4)}$	$88.55^{(4)}$	$93.16^{(4)}$	$106.07^{(4)}$	$123.90^{(4)}$	$164.77^{(4)}$
★ LWFRq	$76.99^{(4)}$	$76.33^{(4)}$	$78.83^{(4)}$	$77.57^{(4)}$	$78.74^{(4)}$	$76.46^{(4)}$	$84.65^{(4)}$	$89.87^{(4)}$	$96.17^{(4)}$	$108.67^{(4)}$	$129.35^{(3)}$	$168.70^{(3)}$
★ DISTq	$69.67^{(4)}$	$73.38^{(4)}$	$74.35^{(4)}$	$74.31^{(4)}$	$75.12^{(4)}$	$74.96^{(4)}$	$77.21^{(4)}$	$77.56^{(4)}$	$80.09^{(4)}$	**$86.25^{(4)}$**	**$101.12^{(4)}$**	**$120.98^{(3)}$**
★ LDISTq	$75.82^{(4)}$	$74.45^{(4)}$	$74.89^{(4)}$	$76.77^{(4)}$	$74.62^{(4)}$	$75.47^{(4)}$	$77.10^{(4)}$	$80.18^{(4)}$	$83.40^{(4)}$	$88.86^{(4)}$	$103.73^{(4)}$	$122.27^{(4)}$

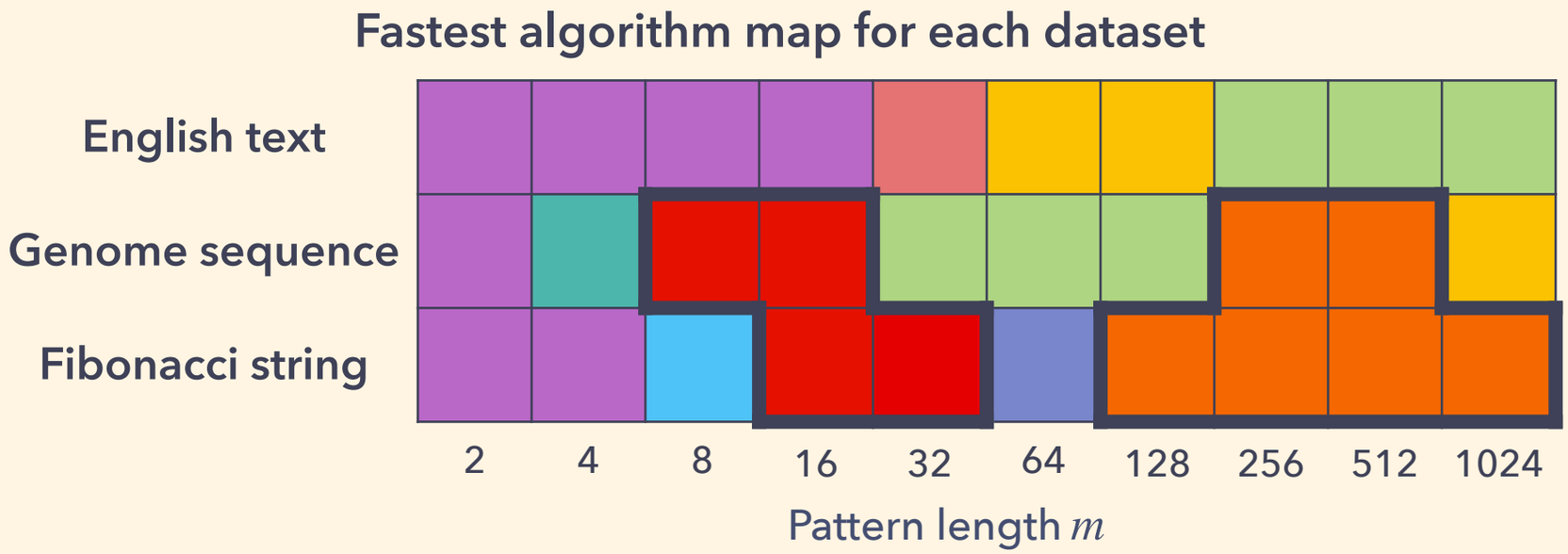
algorithms that run in linear time in the input string size are marked with ★

Unit : millisecond　　The value of q or w giving the best performance is shown in round brackets

Conclusion

$n = |T|$　$m = |P|$　ω : word length
$\sigma = |\Sigma|$ ：alphabet size　q ：q-gram

- Proposed two string matching algorithms based on the distances of the q-gram occurrences
- Both algorithms run in linear time in the input string size



Comparing 15 powerful algorithms announced from 1977 to 2019 with the proposed algorithms

Algorithm	Preprosess	Search
BNDMq [Navarro & Raffinot, 1998]	$O(m+\sigma)$	$O(nm\lceil m/\omega \rceil)$
SBNDMq [Holub & Durian, 2005]	$O(m+\sigma)$	$O(nm\lceil m/\omega \rceil)$
FJS [Franek+, 2005]	$O(m+\sigma)$	$O(n)$
HASHq [Leqroq, 2007]	$O(mq)$	$O(n(m+q))$
BSDMq [Faro & Leqroq, 2012]	$O(m)$	$O(nm)$

Algorithm	Preprocess	Search
WFRq [Cantone+, 2017]	$O(m)$	$O(nm)$
LWFRq [Cantone+, 2019]	$O(m)$	$O(n)$
DISTq New	$O(mq)$	$O(nq)$
LDISTq New	$O(m)$	$O(n)$

Naive solution：$O(nm)$