Pattern Matching in Multiple Streams

CPM, 3–5 July 2012

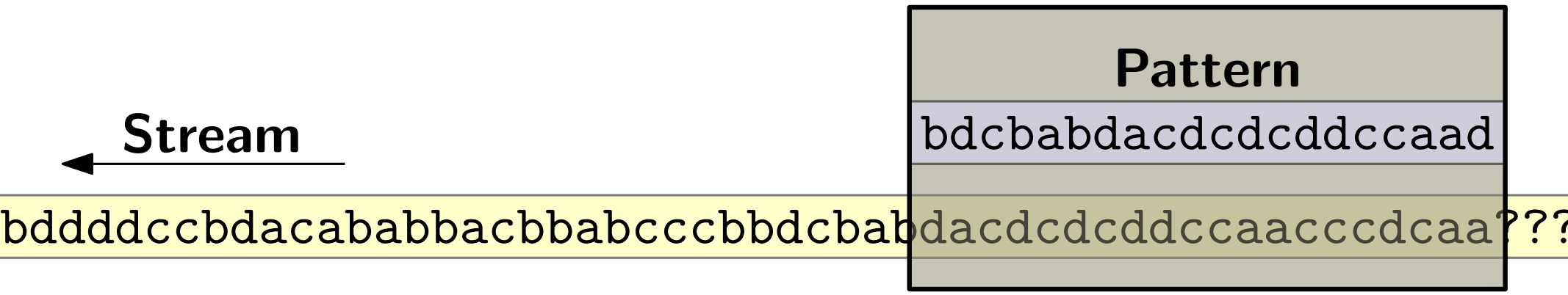
Markus Jalsenius

Joint work with

Raphaël Clifford, Benjamin Sach and Ely Porat

University of BRISTOL

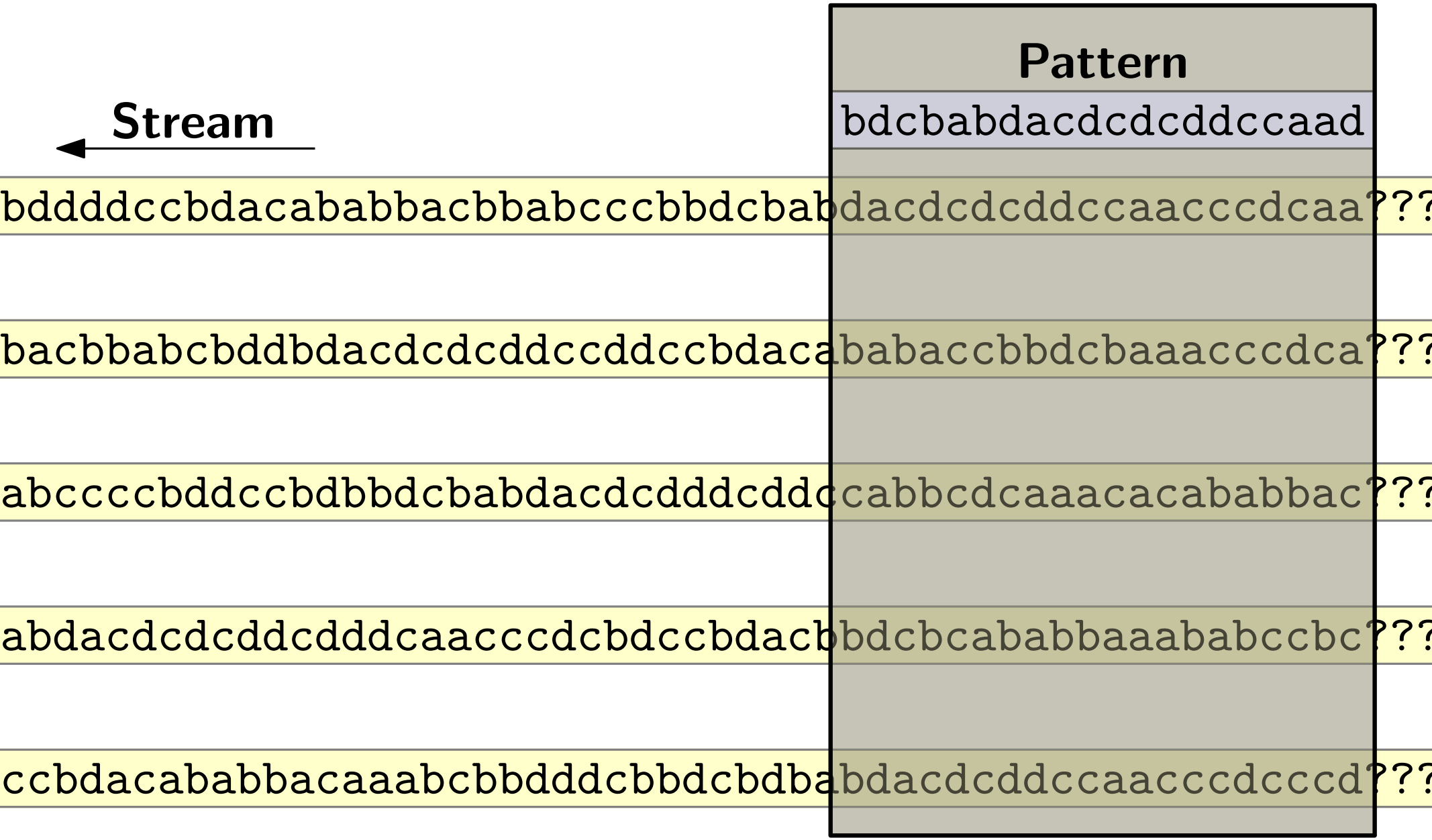
Problem



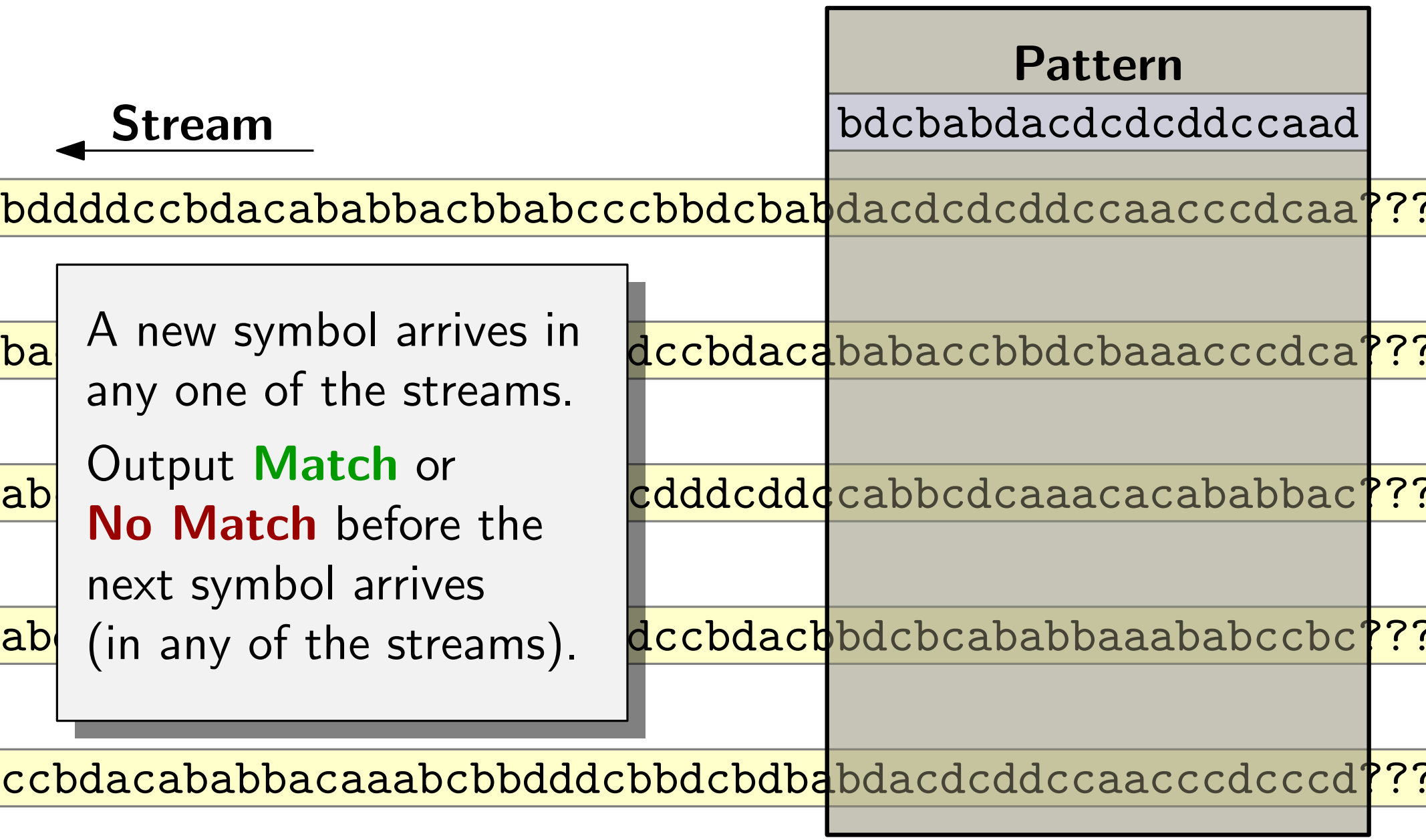
Output **Match** or **No Match** before next symbol arrives.

We consider different notions of a match.

Problem

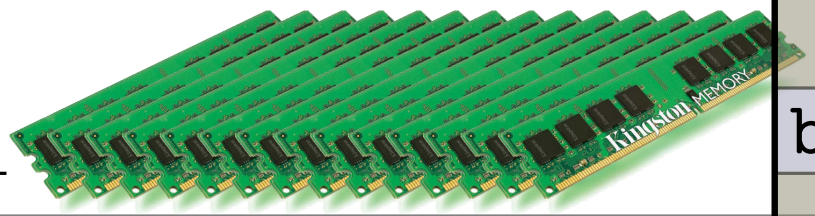

Pattern
bdcbabdacdcdcddccaad
Stream
bddddccbdacababbacbbabcccbbdcbab
bacbbabcbddbdacdcdcddccddccbdaca
abccccbddccbdbbdcbabdacdcdddcddc
abdacdcdcddcdddcaacccdcbdccbdacb
ccbdacababbacaaabcbbdddcbbdcbdba
dacdcdcddccaacccdcaa???
babaccbbdcbaaacccdca???
cabbcdcaaacacababbac???
bdcbcababbaaababccbc???
bdacdcddccaacccdcccd???

Problem


Pattern
bdcbabdacdcdcddccaad
Stream
bddddccbdacababbacbbabcccbbdcbab
dacdcdcddccaacccdcaa???
ba
dccbdaca
babaccbbdcbaaacccdca???
ab
cdddcddc
cabbcdcaaacacababbac???
ab
dccbdacb
bdcbcababbaaababccbc???
ccbdacababbacaaabcbbdddcbbdcbdba
bdacdcddccaacccdcccd???
A new symbol arrives in any one of the streams.
Output Match or No Match before the next symbol arrives (in any of the streams).

Problem

Stream



Pattern

bdcbabdacdcdcddccaad

bddddccbdacababbacbbabcccbbdcbabdacdcdcddccaacccdcaa???

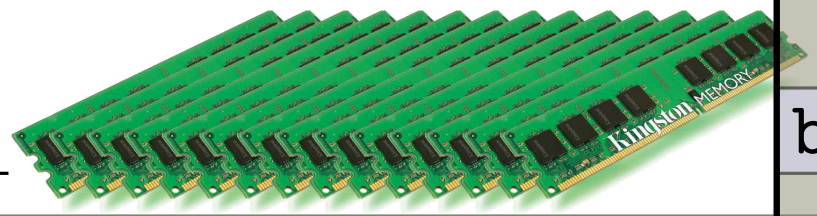
bacbbabcbddbdacdcdcddccddccbdacababaccbbdcbaaacccdca???

abcccbddccbdbbdcbabdacdcdddcddccabbcdcaaacacababbac???

abdacdcdcddcdddcaacccdcbdccbdacbbdcbcababbaaababccbc???

ccbdacababbacaaabcbbdddcbbdcbdbabdacdcddccaacccdcccd???

Problem



Stream

Pattern

bdcbabdacdcdcddccaad

bddddccbdacababbacbbabcccbbdcbabdacdcdcddccaacccdcaa???

bacbbabcbddbdacdcdcddccddccbdacababaccbbdcbaaacccdca???

abccccbddccbdbbdcbabdacdcdddcddccabbcdcaaacacababbac???

abdacdcdcddcdddcaacccdcbdccbdacbbdcbcababbaaababccbc???

ccbdacababbacaaabcbbdddcbbdcbdbabdacdcddccaacccdcccd???

Problem

Read-only memory

Pattern

bdcbabdacdcdcddccaad

Stream

bddddccbdacababbacbbabcccbbdcbabdacdcdcddccaacccdcaa???

bacbbabcbddbdacdcdcddccddccbdacababaccbbdcbaaacccdca???

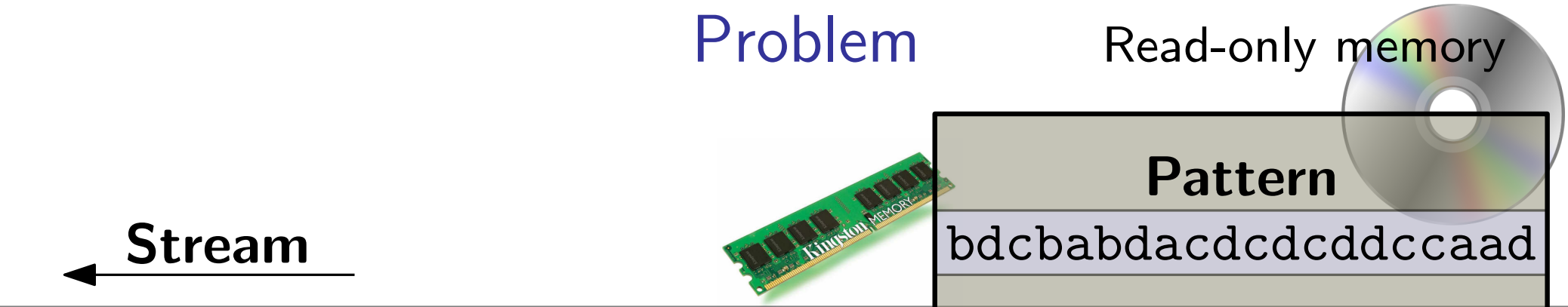
abcccbddccbdbbdcbabdacdcdddcddccabbcdcaaacacababbac???

abdacdcdcddcdddcaacccdcbdccbdacbbdcbcababbaaababccbc???

ccbdacababbacaaabcbbdddcbbdcbdbabdacdcddccaacccdcccd???

Problem

Read-only memory



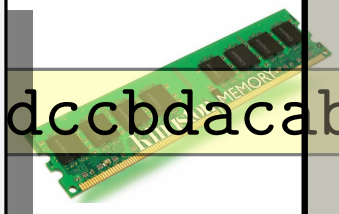
Pattern

bdcbabdacdcdcddccaad

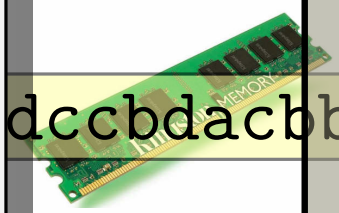
Stream

bddddccbdacababbacbbabcccbbdcbab dacdcdcddccaacccdcaa???

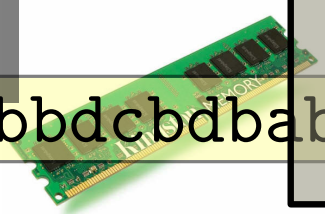
ba dccbdaca babaccbbdcbaaacccdca???



ab cdddcddc cabbcdcaaacacababbac???



ab dccbdacb bdcbcababbaaababccbc???



ccbdacababbacaaabcbbdddcbbdcbdba bdacdcddccaacccdcccd???

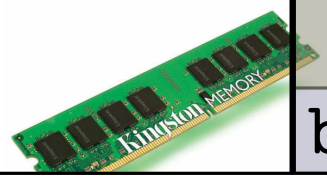
Approach

Preprocess pattern, store the output in **read-only memory** that is shared across the streams.

Eqip each stream with **small working memory**.

Problem

Read-only memory



Pattern

bdcbabdacdcdcddccaad

Stream

bddddccbdacababbac dcddccaacccdcaa???

ba cbbdcbaaacccdca???

ab dcaaacacababbac???

ab ababbaaababccbc???

ccbdacababbacaaabcbbdddcbbdcbdbabdacdcddccaacccdcccd???

Approach

Preprocess patter
the output in **rea**
memory that is s
across the stream

Eqip each stream
small working m

Want fast outputs!

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space 		**Time** 	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space 		Time 	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Naive solution: $O(ms)$ space

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Naive solution: $O(ms)$ space

Read-only space

Read/write space

Preprocessing time is roughly $O(m \log m)$

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Naive solution: $O(ms)$ space

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space 		Time 	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

A. Amir, M. Lewenstein, and E. Porat. Faster algorithms for string matching with k mismatches. *Journal of Algorithms*, 50(2):257–275, 2004.

Also, for one stream: $O(\sqrt{k} \log k + \log m)$

R. Clifford and B. Sach. Pseudo-realtime pattern matching: Closing the gap. *CPM'10*, pages 101–111, 2010.

Notation:

m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$

G. M. Landau and U. Vishkin. Fast string matching with k differences. *Journal of Computer System Sciences*, 37(1):63–78, 1988.

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$
L_1, L_2, Hamming distances, convolution/ cross-correlation		$\Omega(ms)$ bits		

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Results

	Space		Time	
	1 stream	s streams	1 stream	s streams
Exact matching	$O(m)$	$O(m+s)$ words $\Omega(m \log \lvert\Sigma\rvert + s)$ bits	$O(1)$	$O(1)$
k-mismatch	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(n\sqrt{k \log k})$ offline	$O(k)$
k-difference (edit distance)	$O(m)$	$O(m+ks)$ words $\Omega(m \log \lvert\Sigma\rvert + ks)$ bits	$O(nk)$ offline	$O(k)$
L_1, L_2, Hamming distances, convolution/ cross-correlation		$\Omega(ms)$ bits		

Randmosied bounds are open!

Notation:
m = pattern length, n = text length (when offline), Σ = alphabet. We operate in the RAM model.

Exact matching

 $O(m+s)$  $O(1)$

$O(1)$ amortised (e.g. KMP)
$O(1)$ unamortised (e.g. Galil 1981)

Exact matching

 $O(m+s)$  $O(1)$

$O(1)$ amortised (e.g. KMP)
$O(1)$ unamortised (e.g. Galil 1981)

 Buffering the text $\Longrightarrow$ $O(ms)$ space

Exact matching

 $O(m+s)$  $O(1)$

Simple modification of KMP

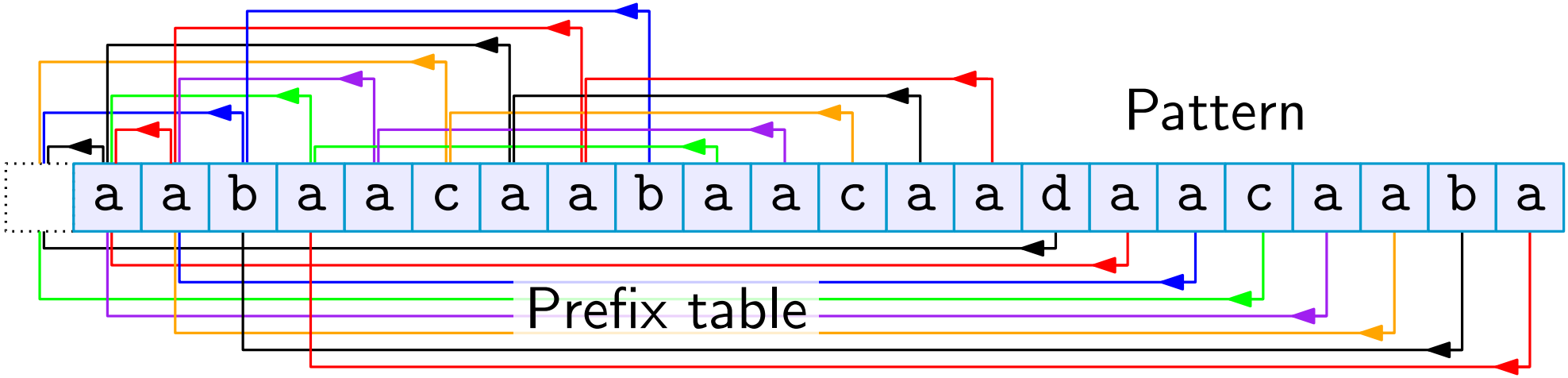
Pattern

a	a	b	a	a	c	a	a	b	a	a	c	a	a	d	a	a	c	a	a	b	a

Exact matching

 $O(m+s)$  $O(1)$

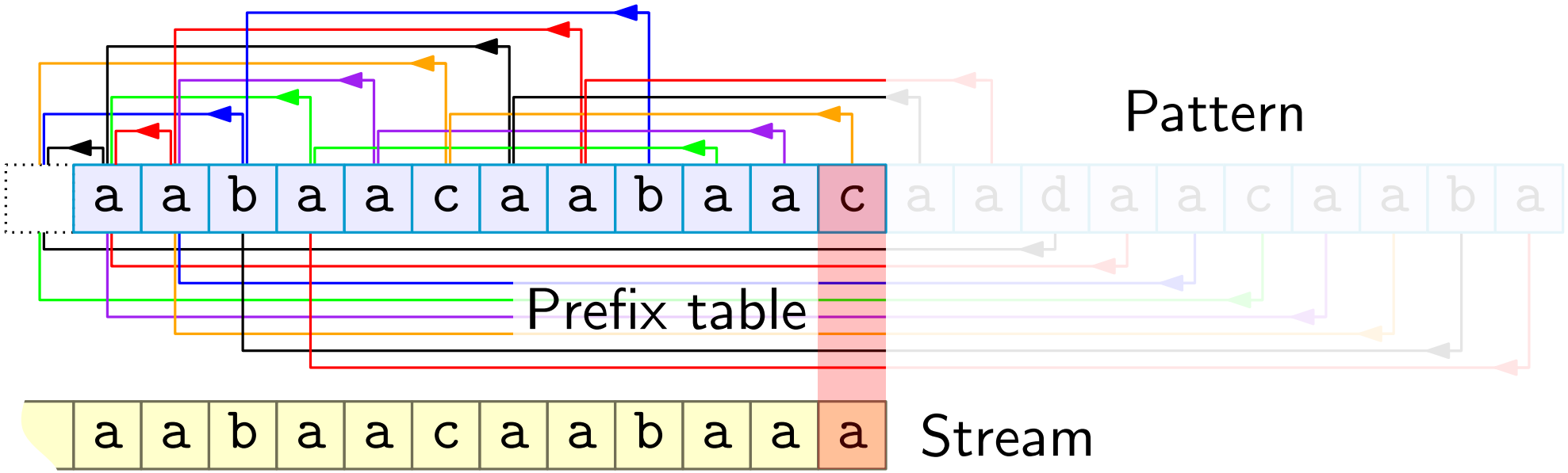
Simple modification of KMP



Exact matching

 $O(m+s)$  $O(1)$

Simple modification of KMP

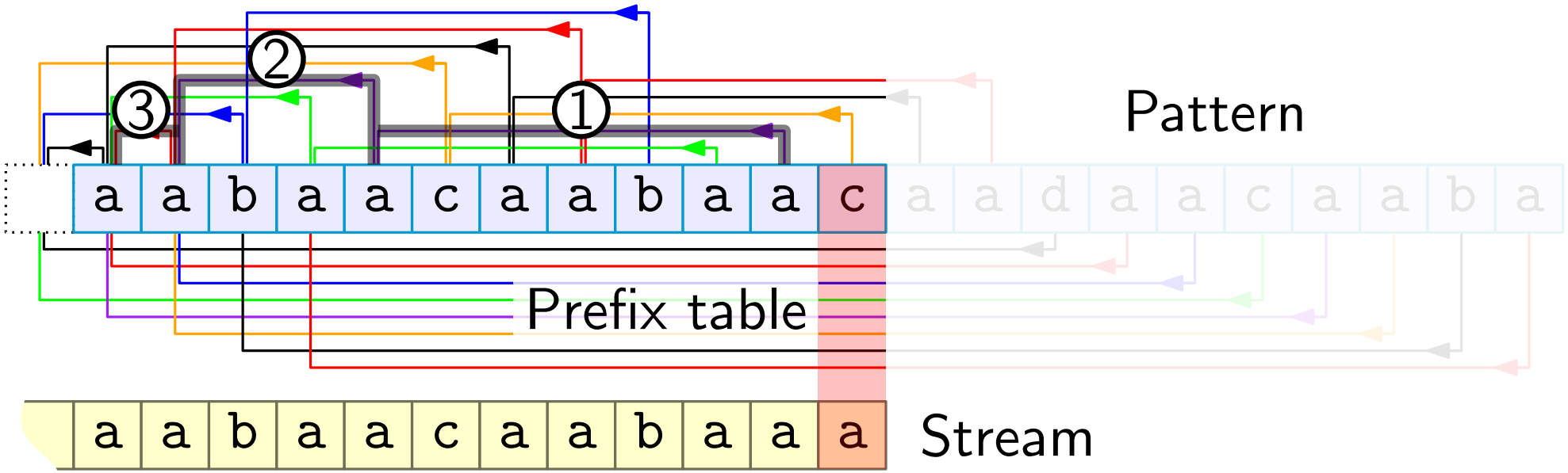


Exact matching


$O(m+s)$


$O(1)$

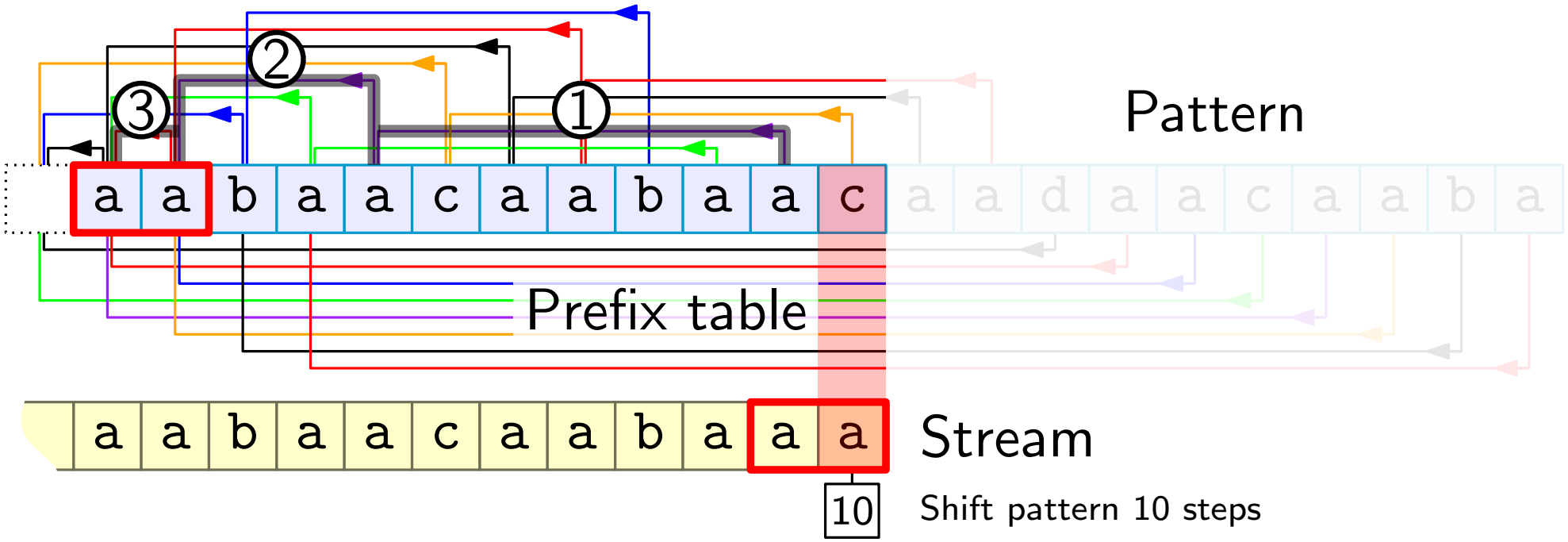
Simple modification of KMP



Exact matching

 $O(m+s)$  $O(1)$

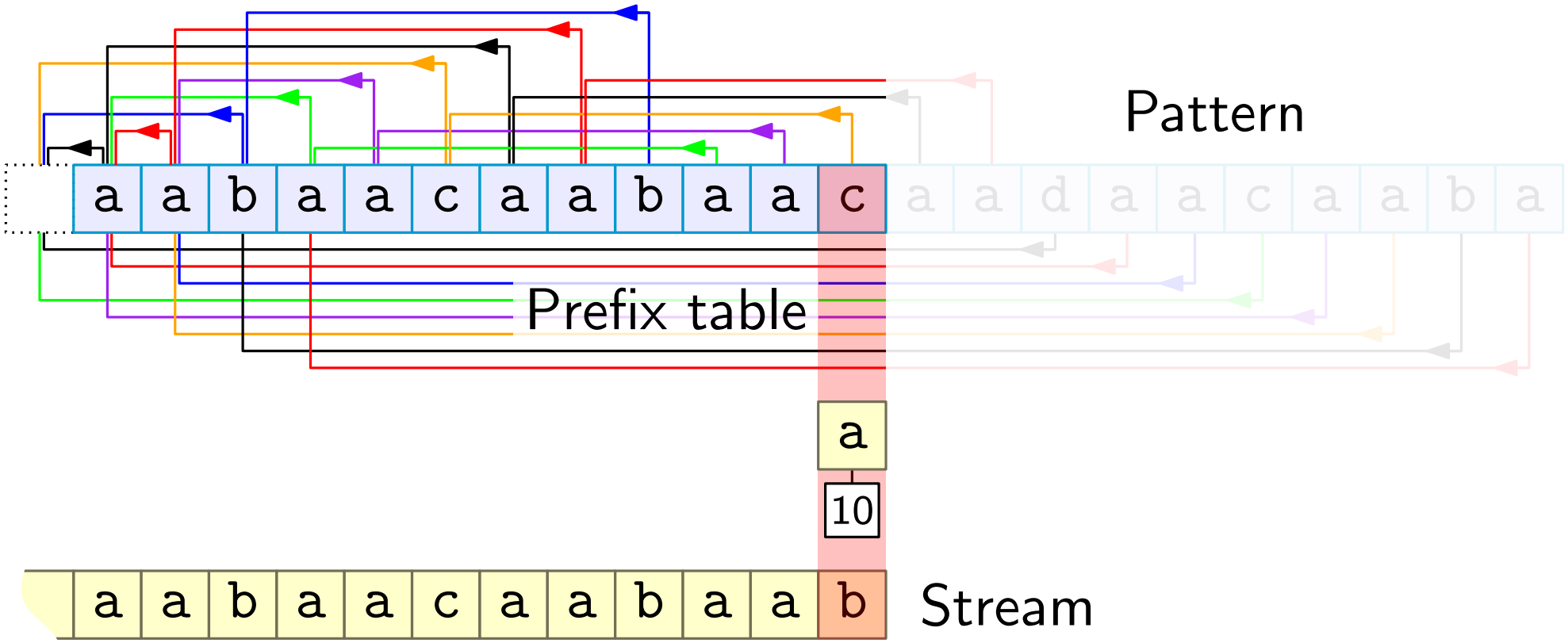
Simple modification of KMP



Exact matching

 $O(m+s)$  $O(1)$

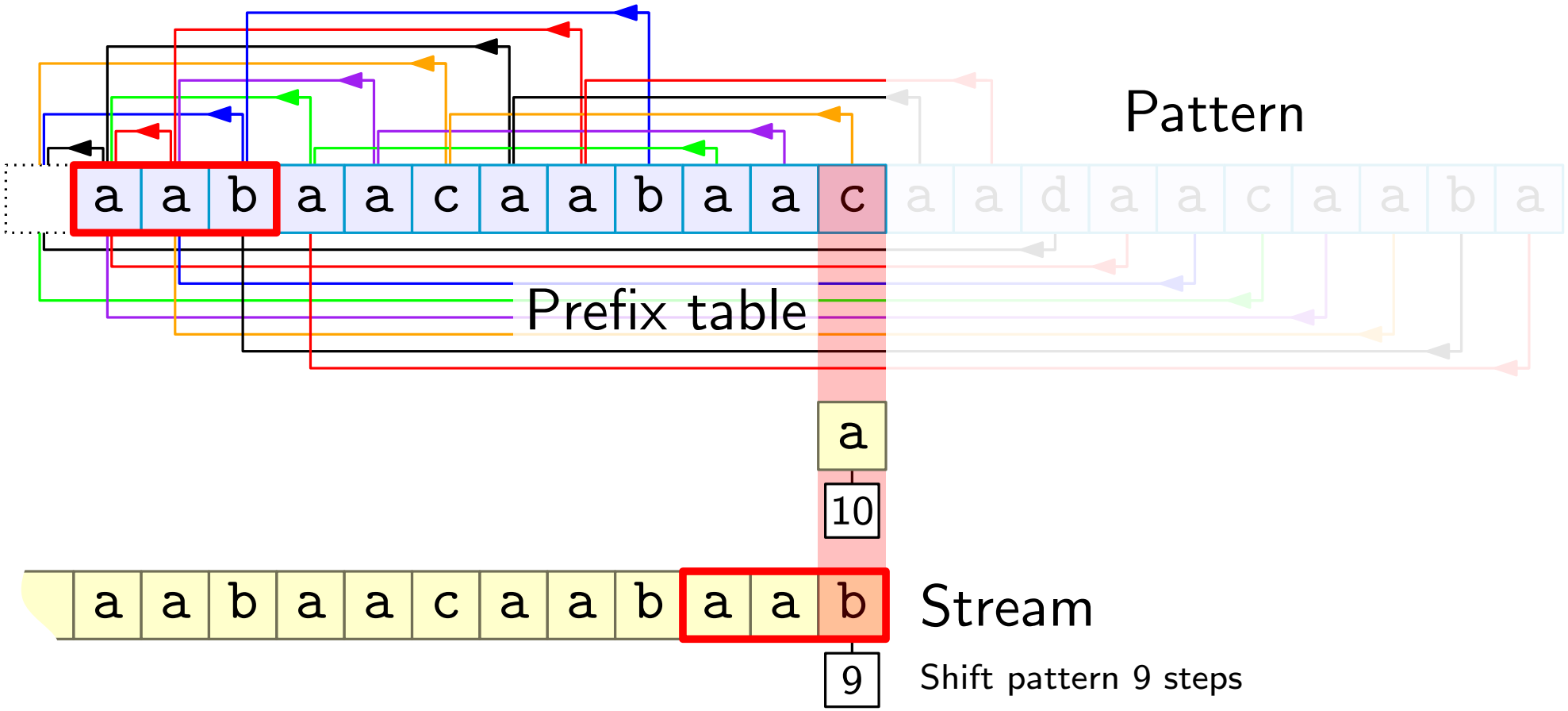
Simple modification of KMP



Exact matching

 $O(m+s)$  $O(1)$

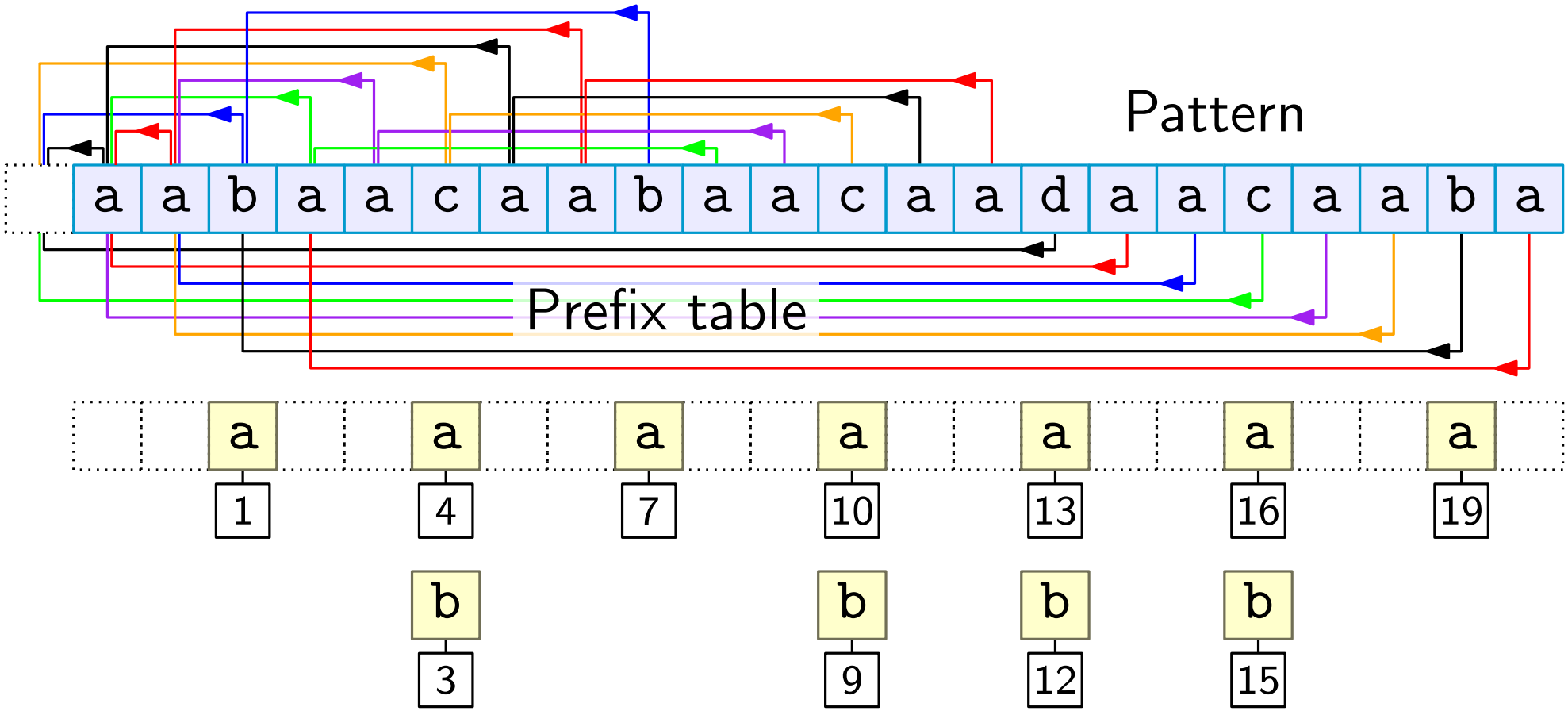
Simple modification of KMP



Exact matching

 $O(m+s)$  $O(1)$

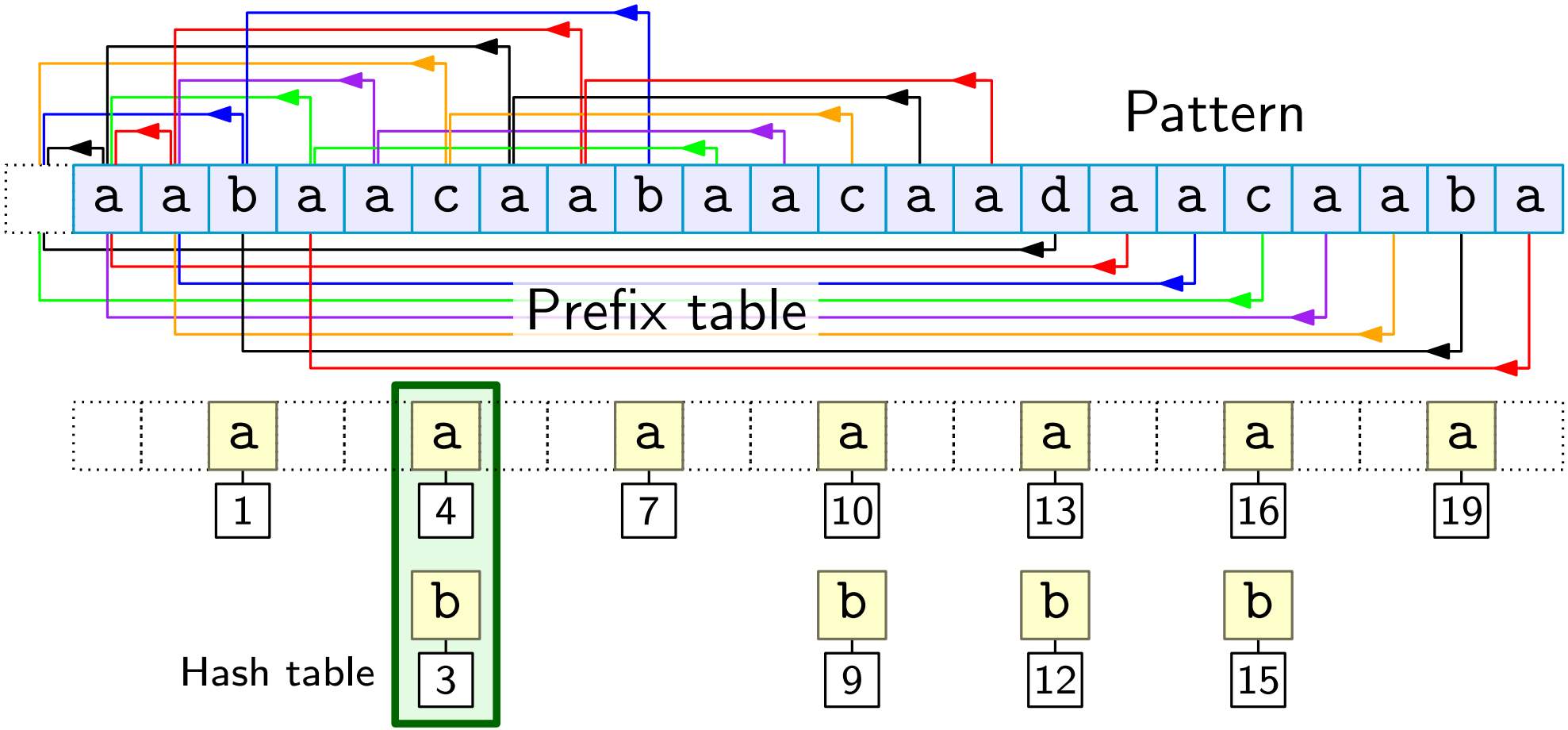
Simple modification of KMP



Exact matching

 $O(m+s)$  $O(1)$

Simple modification of KMP

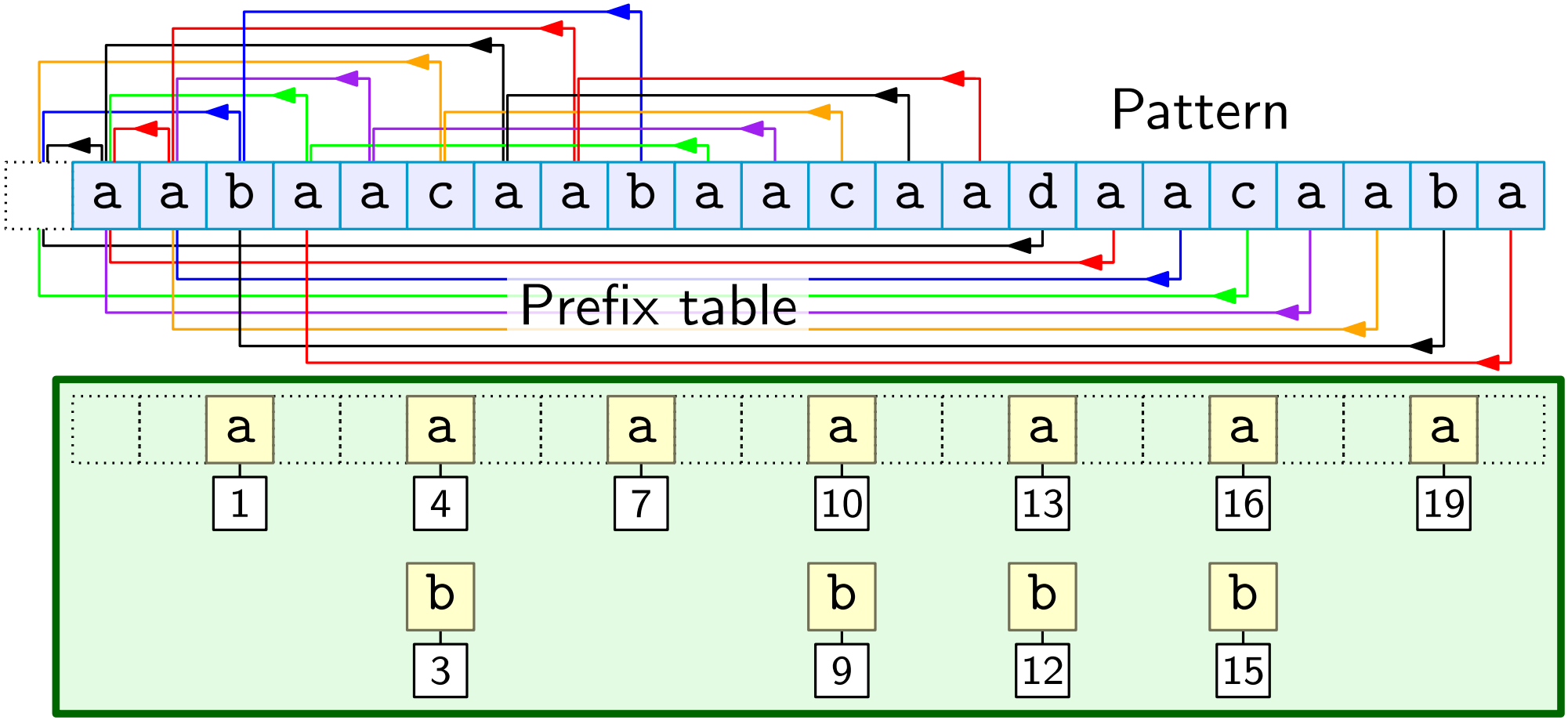


For each position, the shift is found in $O(1)$ time through static perfect hashing.

Exact matching

 $O(m+s)$  $O(1)$

Simple modification of KMP



Total number of elements to store is at most m.

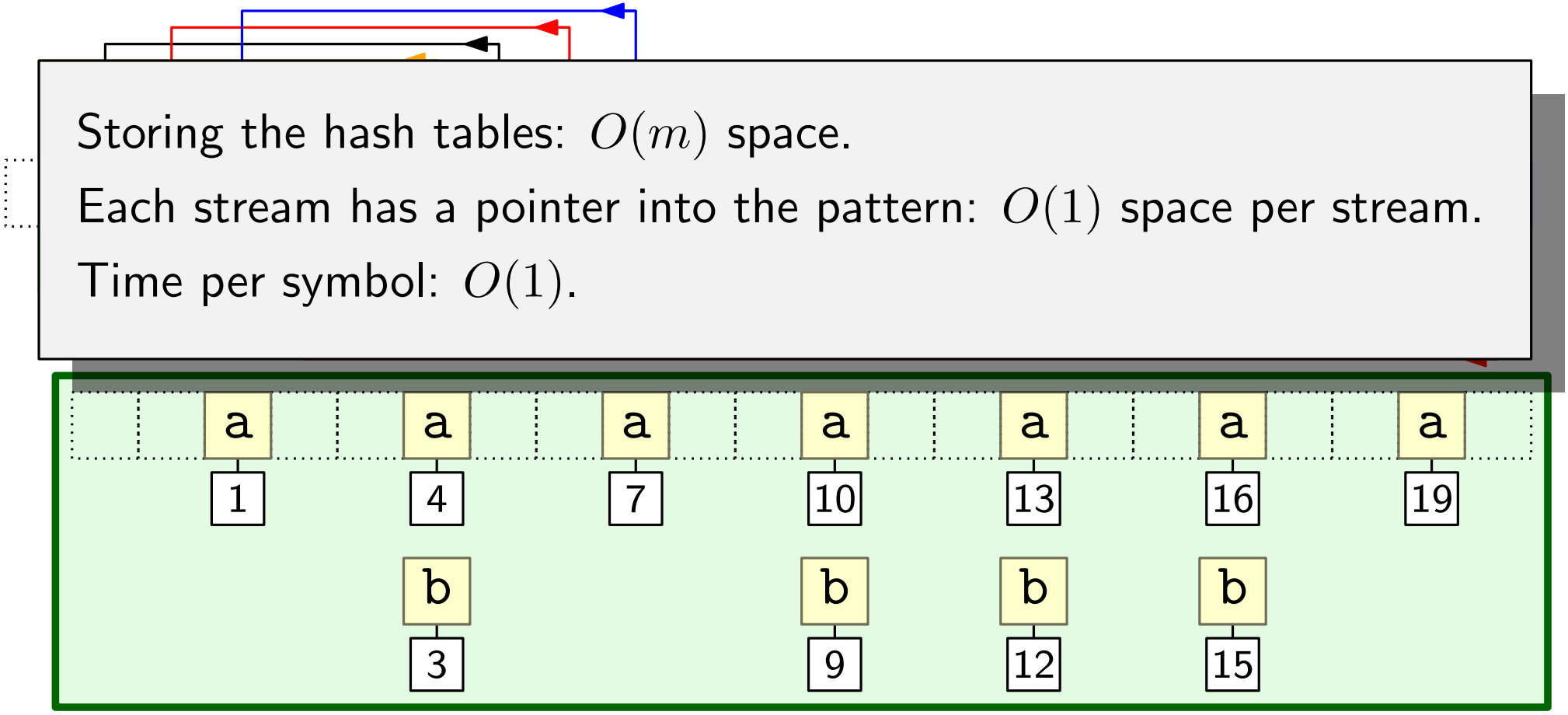
Follows from: I. Simon. String matching algorithms and automata. In *First American Workshop on String Processing*, pages 151–157, 1993.

Exact matching


$O(m+s)$


$O(1)$

Simple modification of KMP

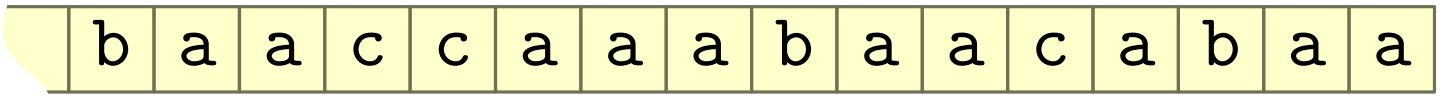


Total number of elements to store is at most m.

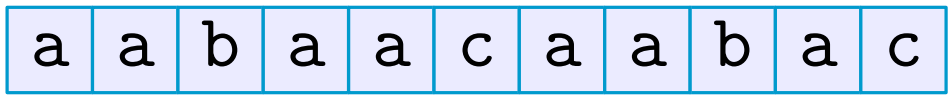
Follows from: I. Simon. String matching algorithms and automata. In *First American Workshop on String Processing*, pages 151–157, 1993.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



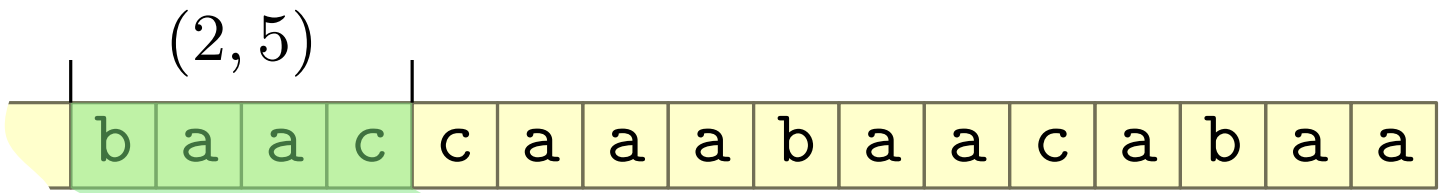
Pattern



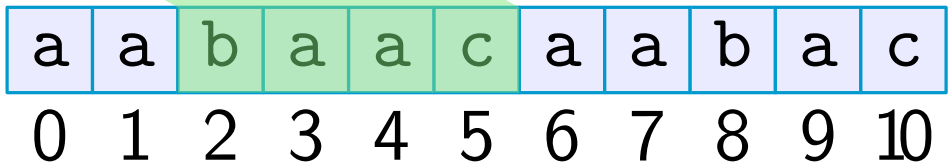
0 1 2 3 4 5 6 7 8 9 10

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.

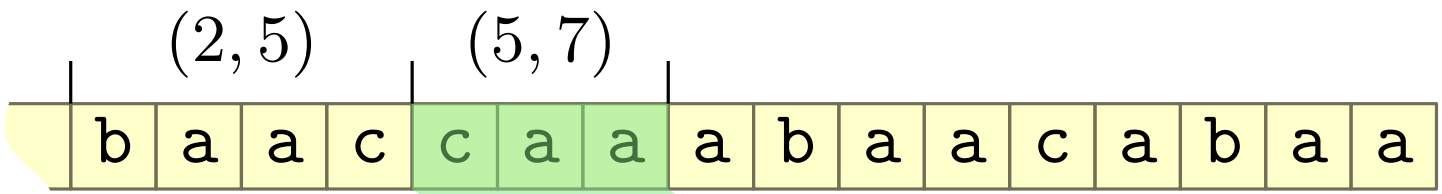


Pattern



Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.

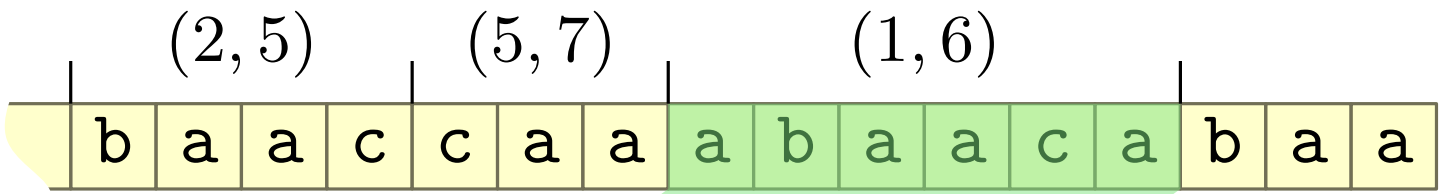


Pattern

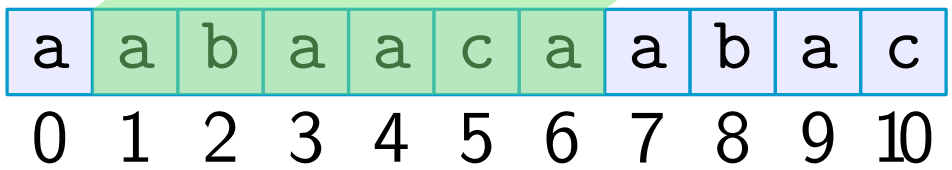
a	a	b	a	a	c	a	a	b	a	c
0	1	2	3	4	5	6	7	8	9	10

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.

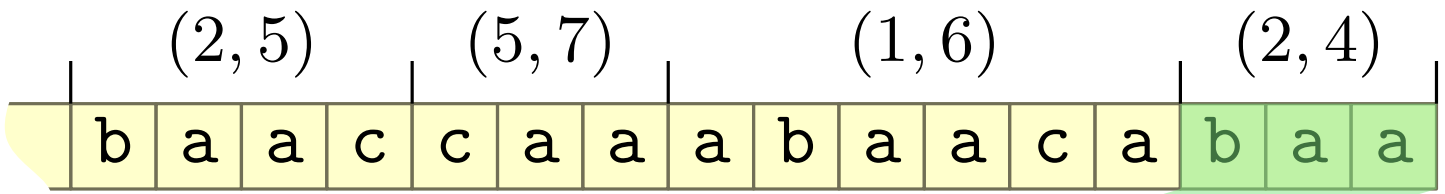


Pattern

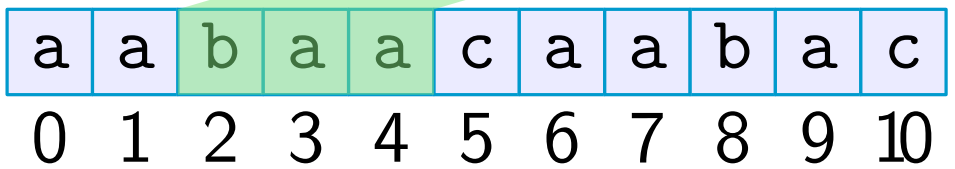


Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



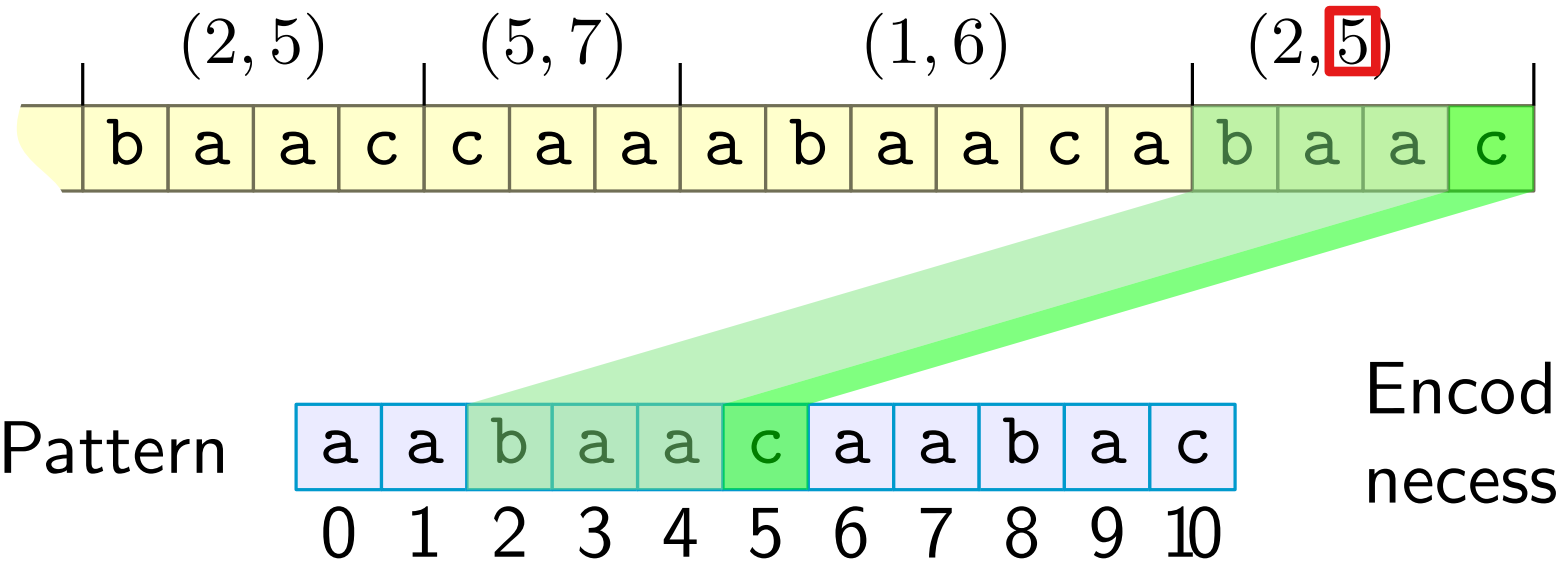
Pattern



Encoding is not necessarily unique.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



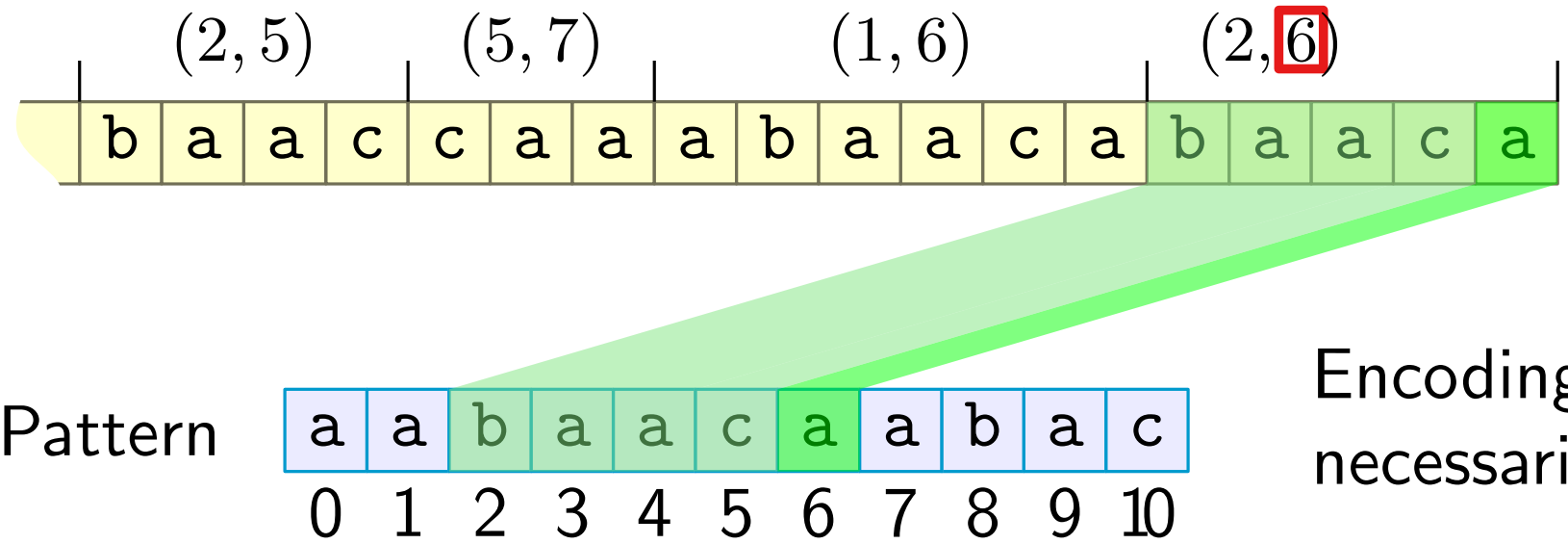
Encoding is not necessarily unique.

Greedy construction

Extend pair if possible...

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



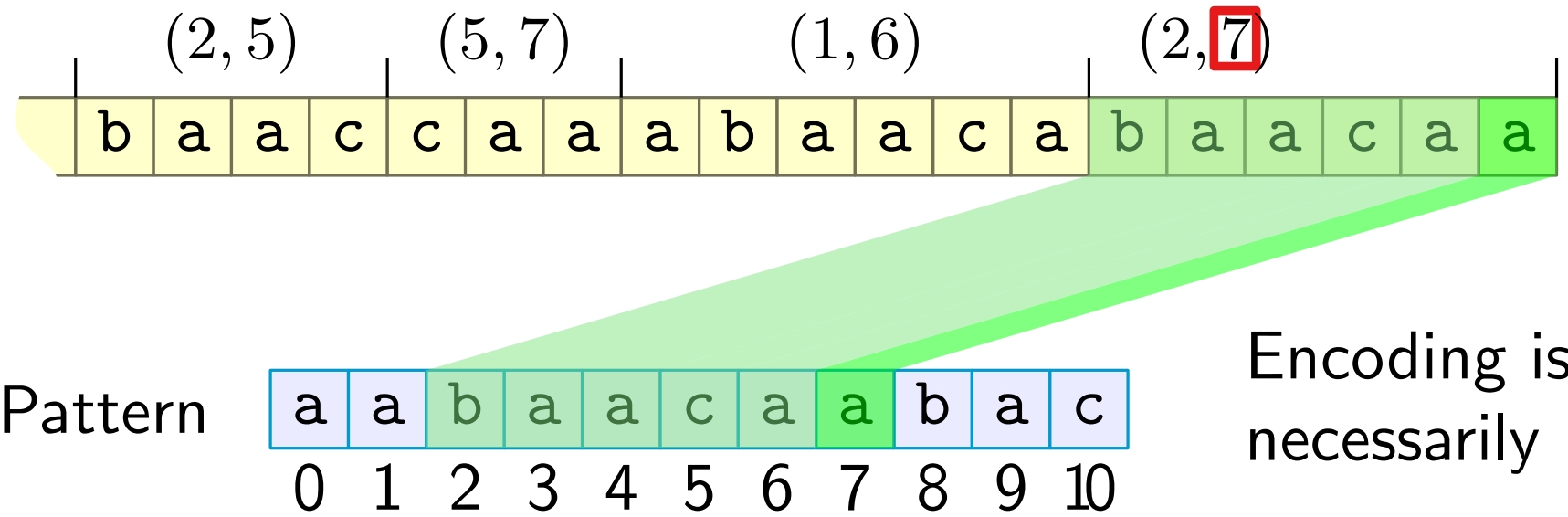
Encoding is not necessarily unique.

Greedy construction

Extend pair if possible...

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



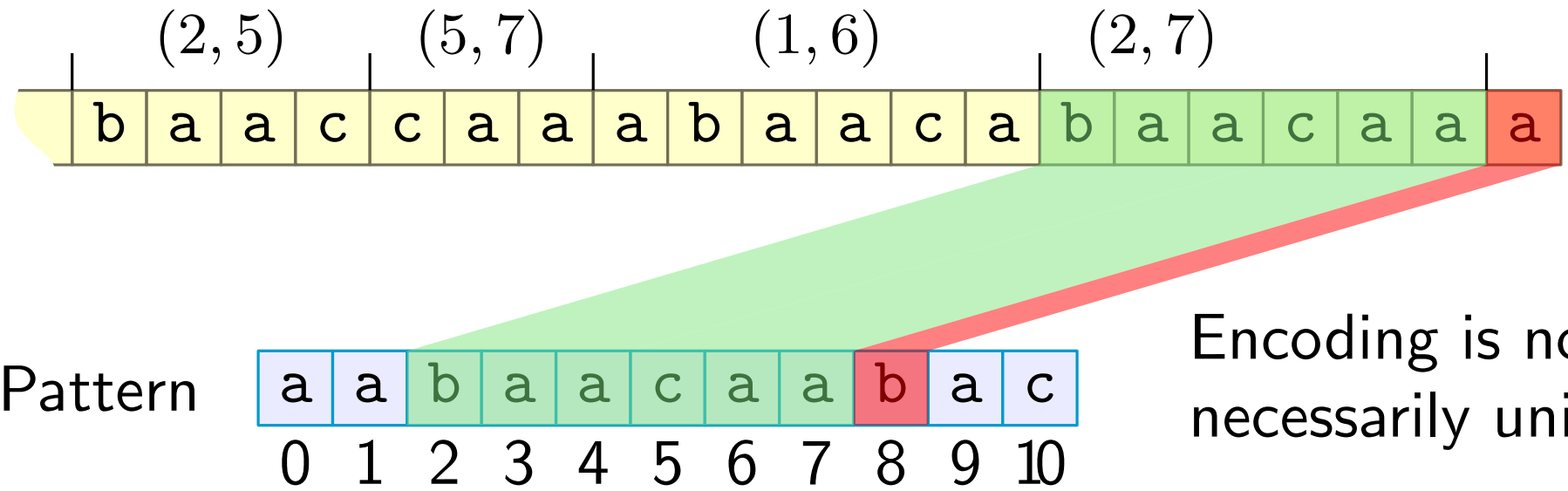
Encoding is not necessarily unique.

Greedy construction

Extend pair if possible...

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



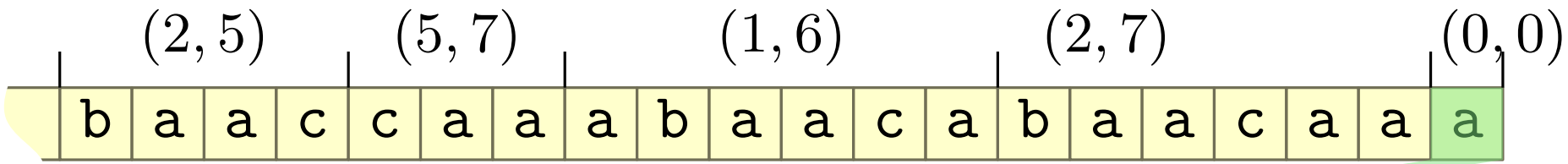
Encoding is not necessarily unique.

Greedy construction

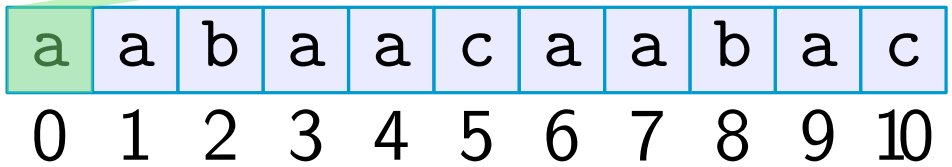
Extend pair if possible... ...if not, start a new pair.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



Pattern



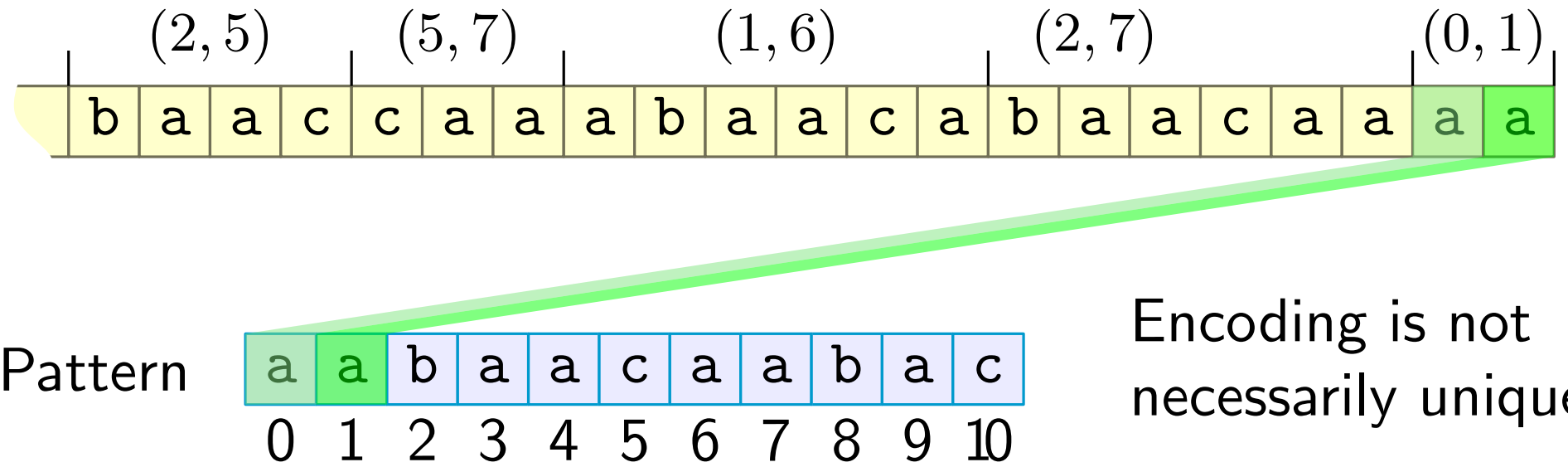
Encoding is not necessarily unique.

Greedy construction

Extend pair if possible... ...if not, start a new pair.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



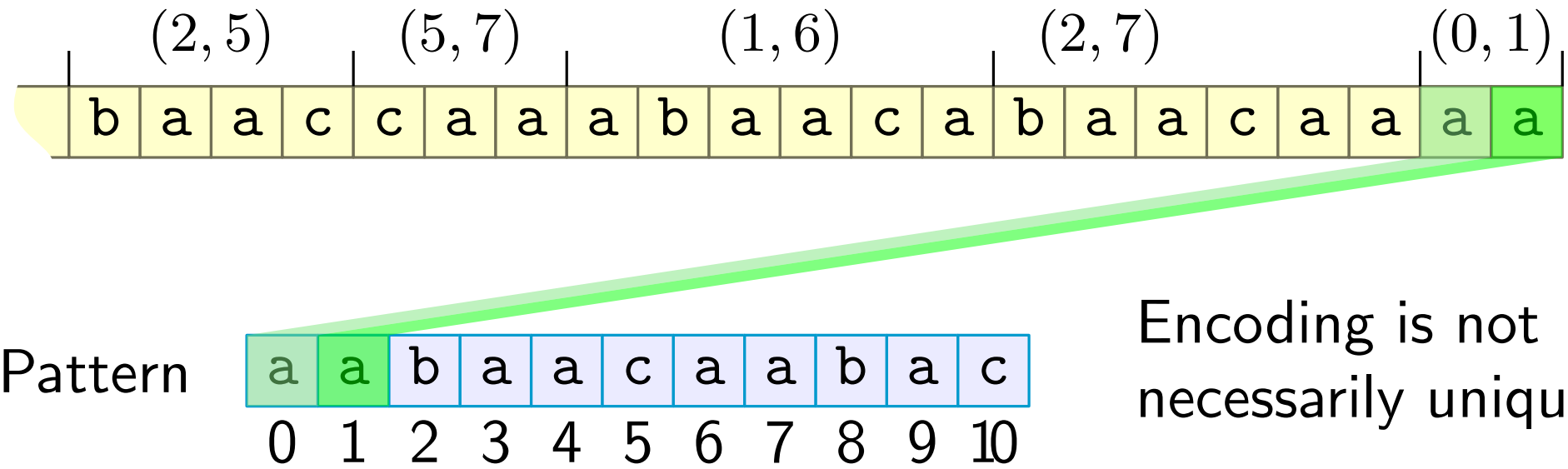
Encoding is not necessarily unique.

Greedy construction

Extend pair if possible... ...if not, start a new pair.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



Encoding is not necessarily unique.

Greedy construction

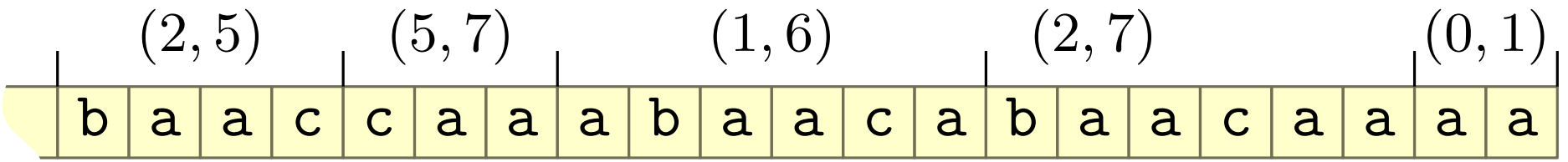
Extend pair if possible... ...if not, start a new pair.

Results in a minimal length encoding.

Takes $O(1)$ time per symbol (using suffix tree of pattern).

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



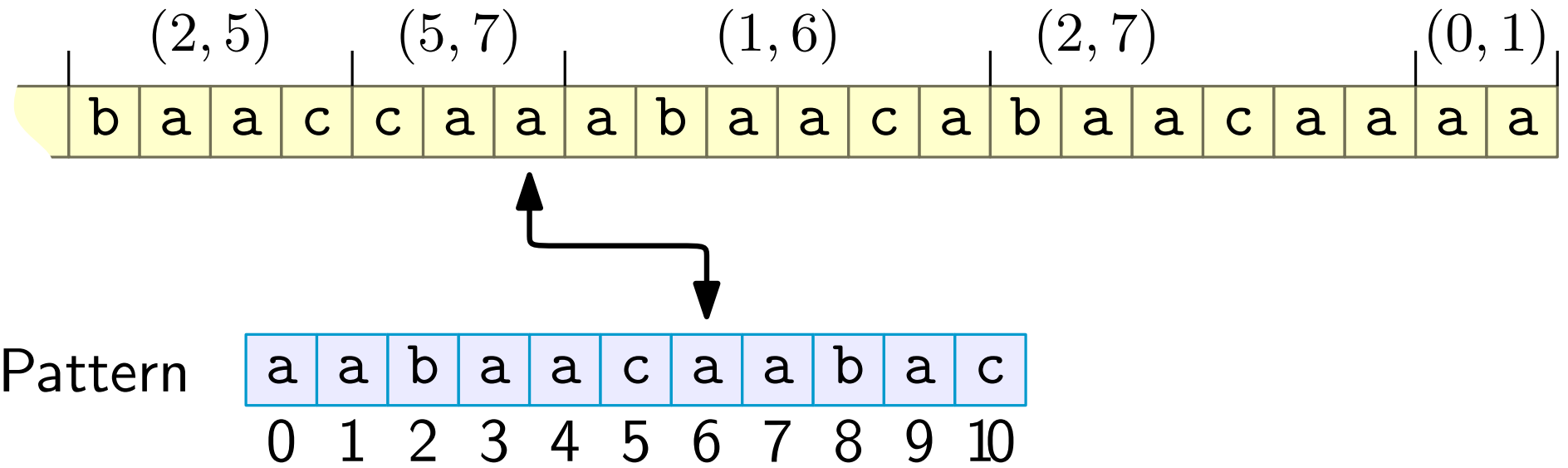
Pattern

a	a	b	a	a	c	a	a	b	a	c
0	1	2	3	4	5	6	7	8	9	10

Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

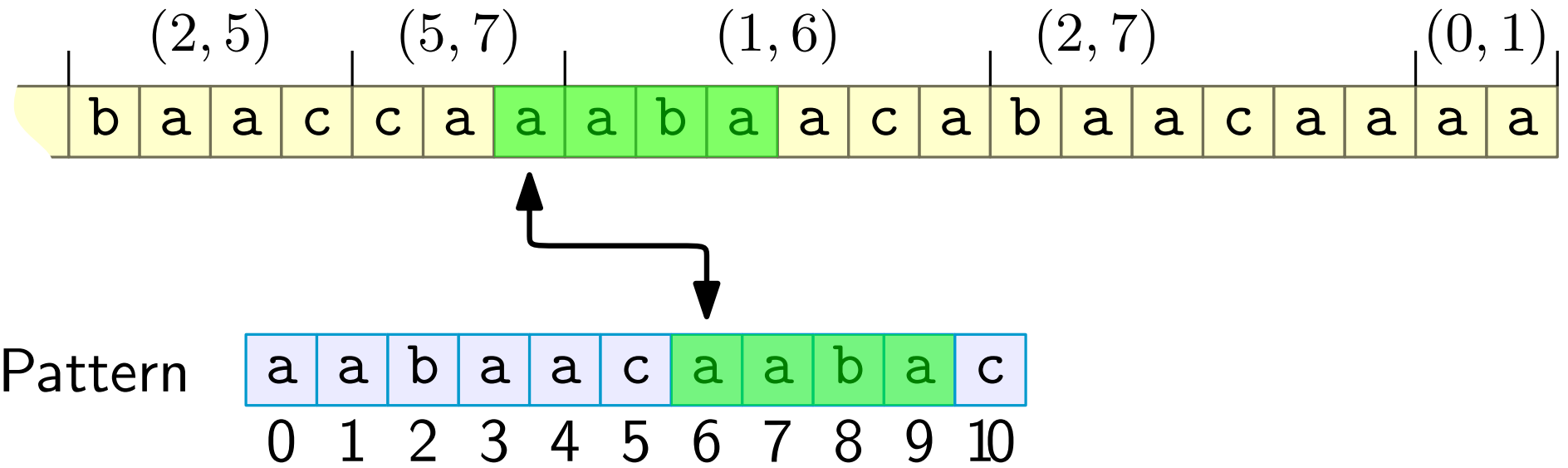
Encoding the stream in terms of the pattern.



Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

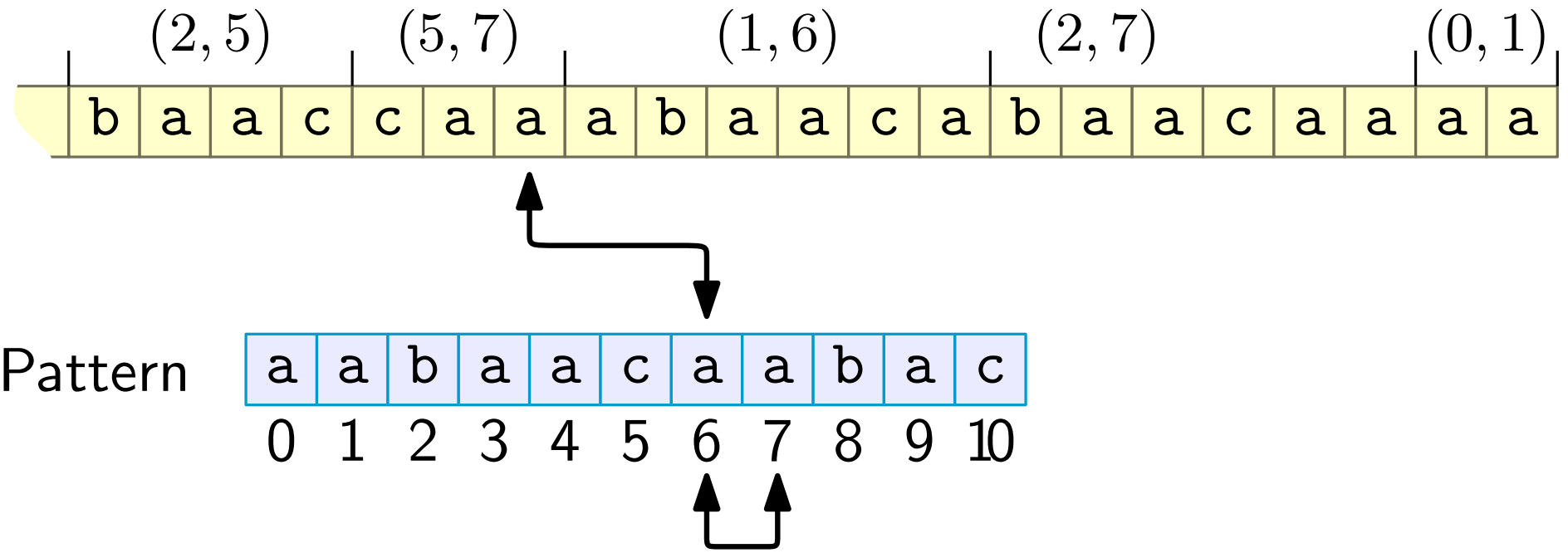
Encoding the stream in terms of the pattern.



Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

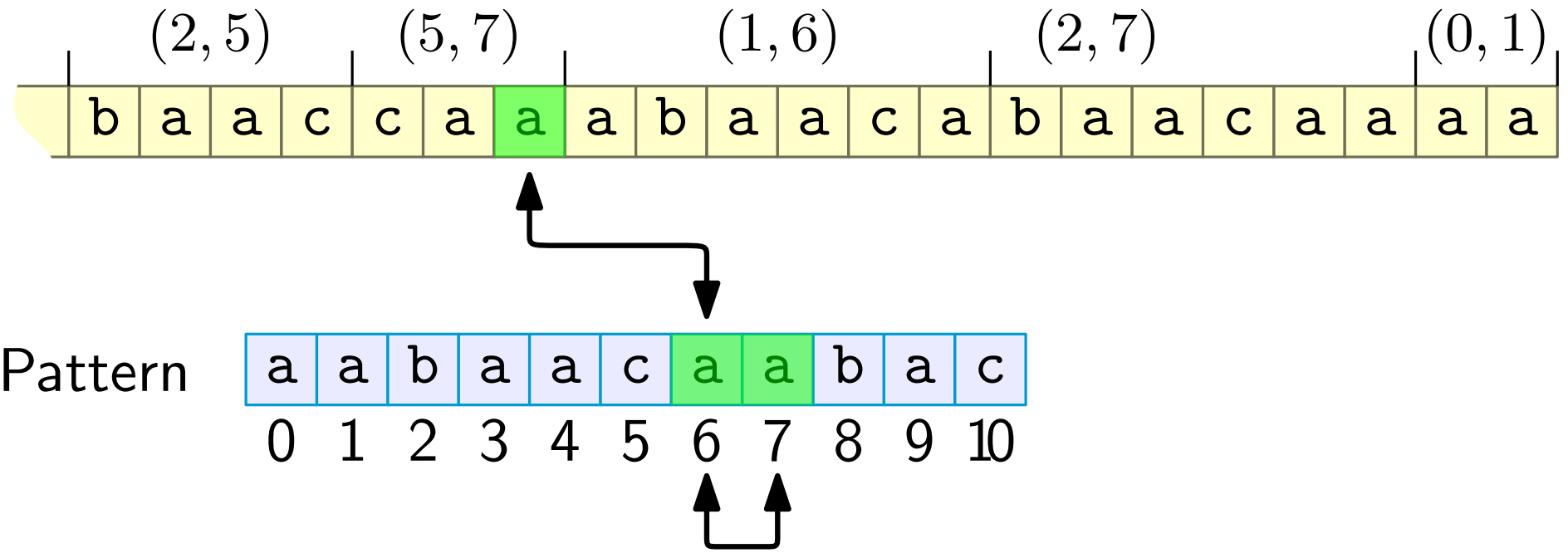
Encoding the stream in terms of the pattern.



Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

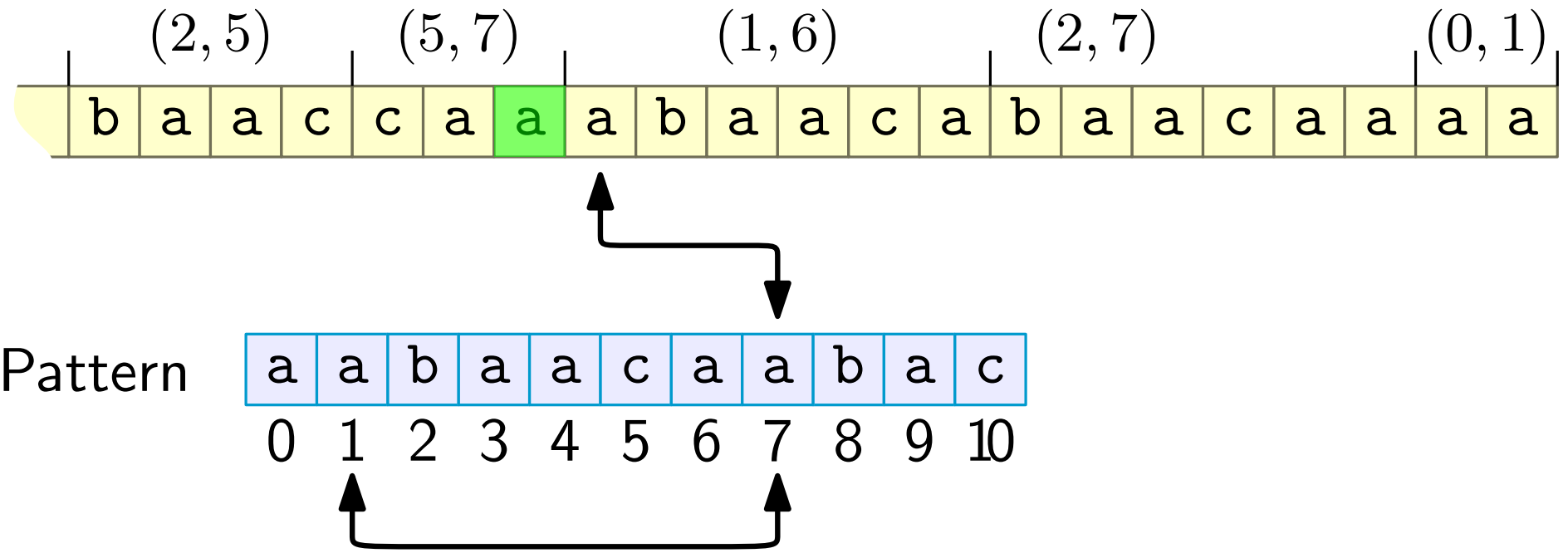
Encoding the stream in terms of the pattern.



Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

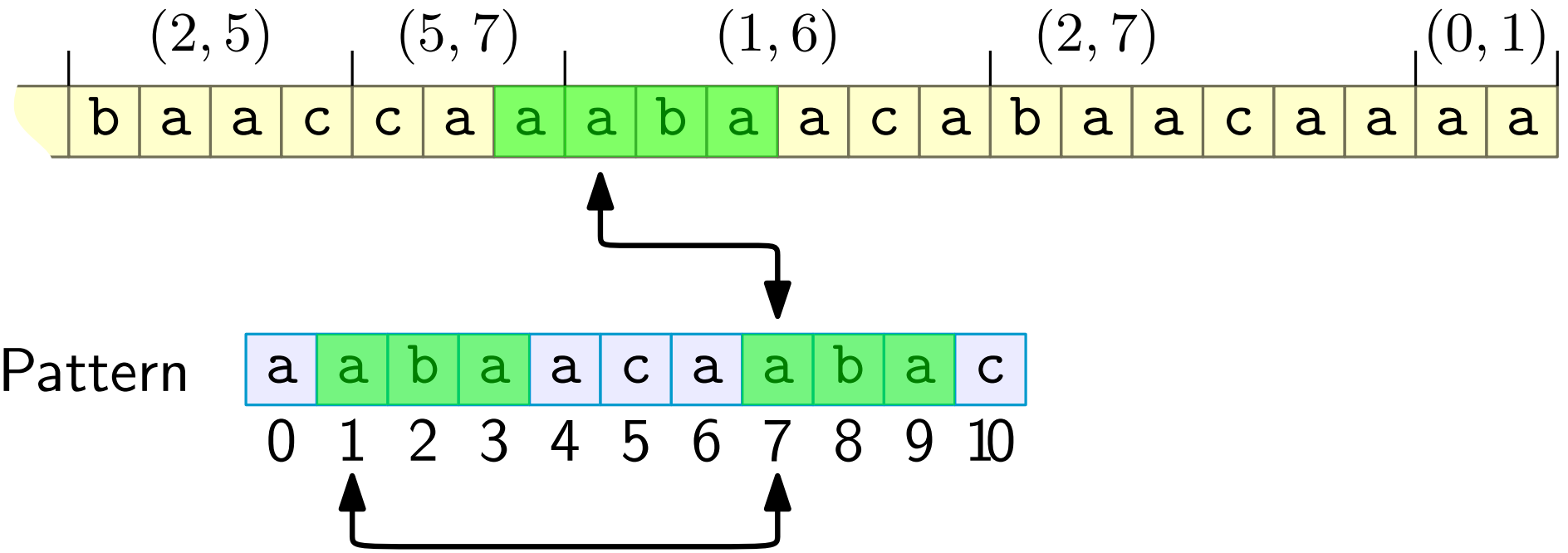
Encoding the stream in terms of the pattern.



Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

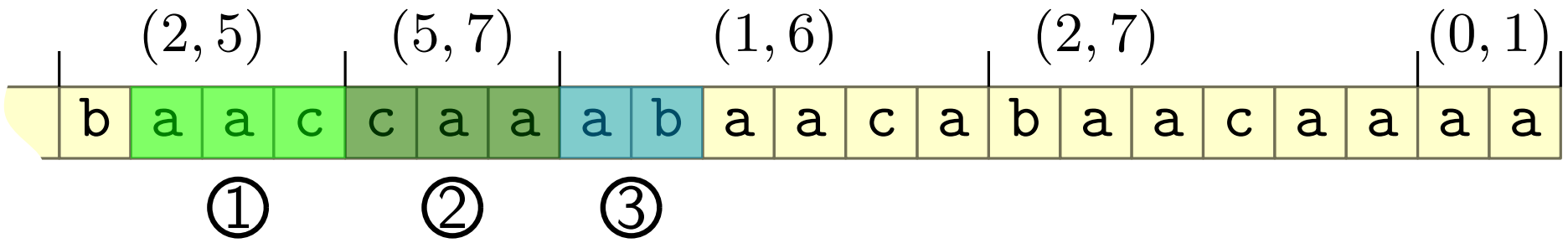
Encoding the stream in terms of the pattern.



Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



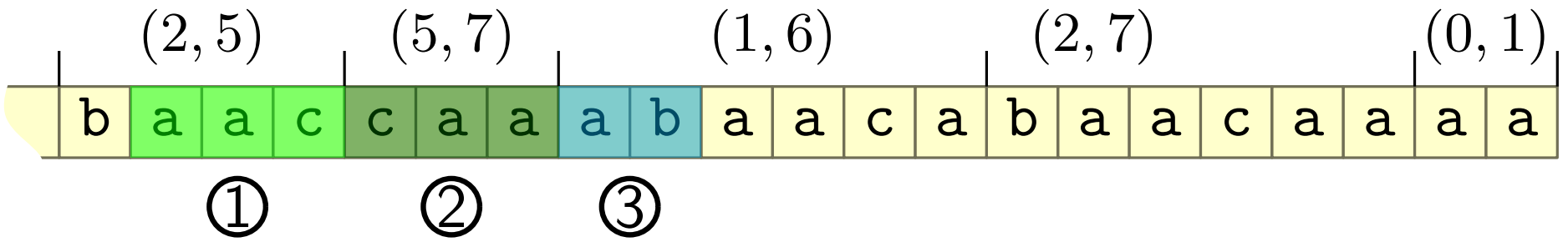
Pattern

a	a	b	a	a	c	a	a	b	a	c
0	1	2	3	4	5	6	7	8	9	10

Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preparation for k-mismatch/difference

Encoding the stream in terms of the pattern.



Pattern

a	a	b	a	a	c	a	a	b	a	c
0	1	2	3	4	5	6	7	8	9	10

Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preprocess pattern to support LCE queries in constant time.

k-mismatch

 $O(m+ks)$  $O(k)$

	b	a	a	c	c	a	a	a	b	a	a	c	a	b	a	a	c	a	a	a	a

Pattern

c	c	c	a	a	b	a	a	a	a	b	a	a	c	b	a	a	a

k-mismatch


$O(m+ks)$

$O(k)$

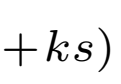
LCE query
('kangaroo jumping')

	b	a	a	c	c	a	a	a	b	a	a	c	a	b	a	a	c	a	a	a	a

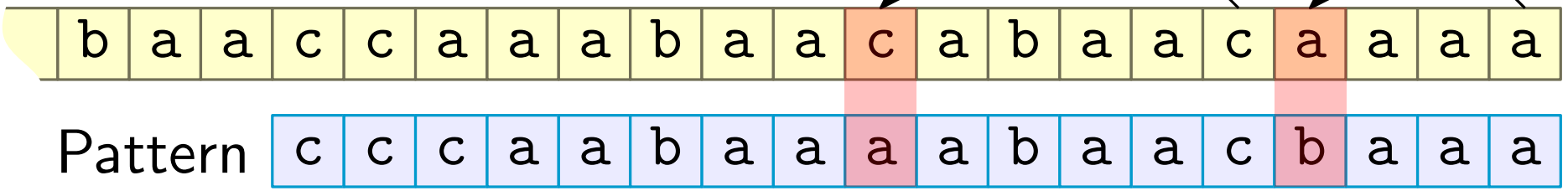
Pattern	c	c	c	a	a	b	a	a	a	a	b	a	a	c	b	a	a	a

k-mismatch


$O(m+ks)$


$O(k)$

LCE query
('kangaroo jumping')

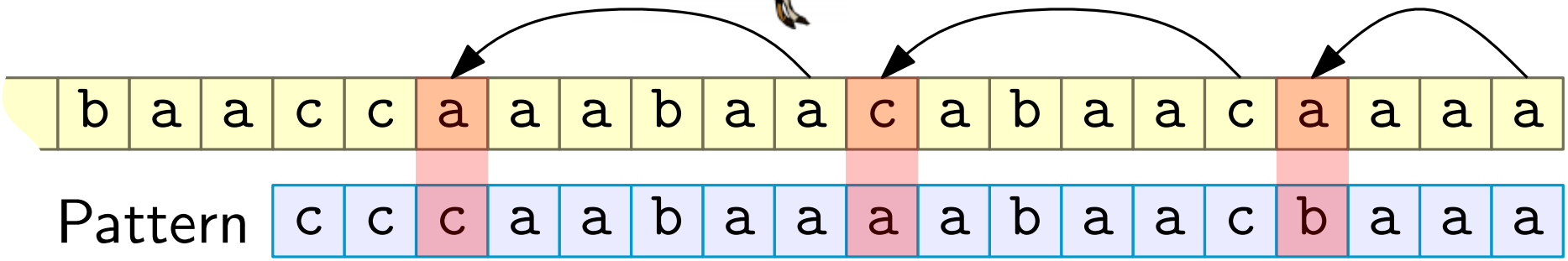


k-mismatch


$O(m+ks)$


$O(k)$

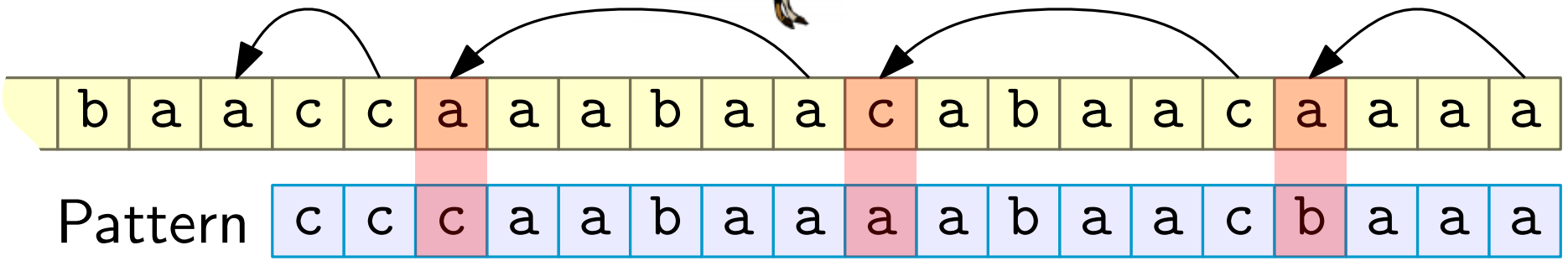
LCE query
('kangaroo jumping')



k-mismatch

 $O(m+ks)$  $O(k)$

LCE query
('kangaroo jumping')



- Each jump spans at most three pairs in stream encoding.
- A mismatch could be in its own pair.

$\Longrightarrow$ Only store the last $4(k+1)$ pairs of each stream.

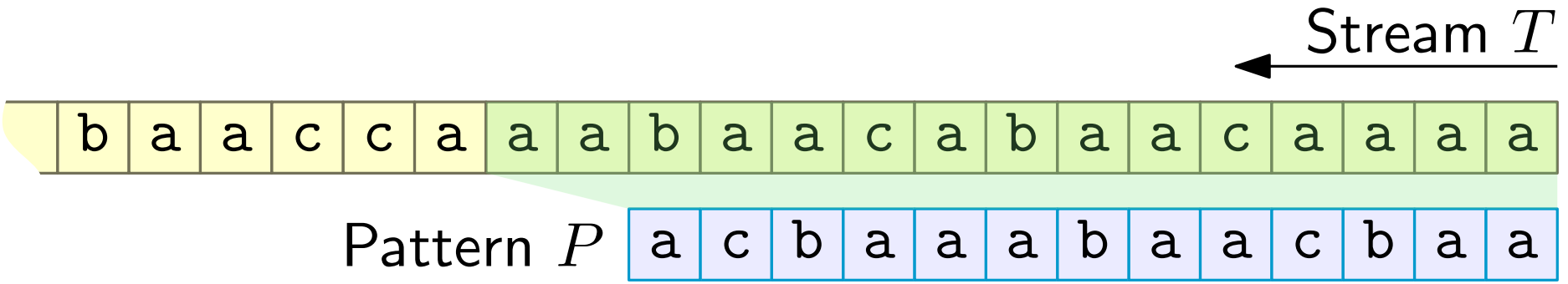
$O(m+ks)$ space (m for pattern/pattern LCE queries),
$O(k)$ time per symbol.

k-difference

 $O(m+ks)$  $O(k)$

Edit operations: **insert**, **delete**, **mismatch**.

Report smallest edit distance between pattern and suffixes of stream if k or less.

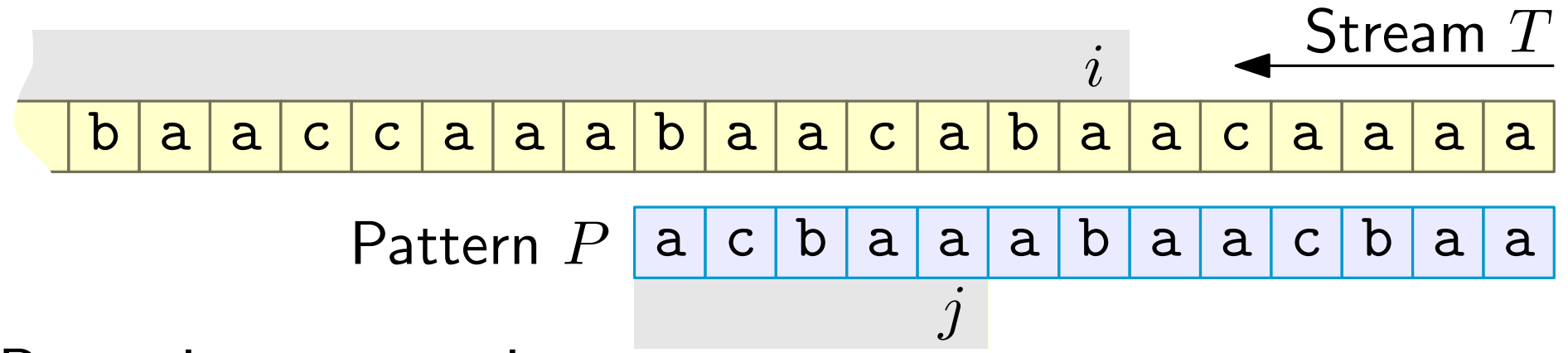


k-difference

 $O(m+ks)$  $O(k)$

Edit operations: **insert**, **delete**, **mismatch**.

Report smallest edit distance between pattern and suffixes of stream if k or less.



Dynamic programming

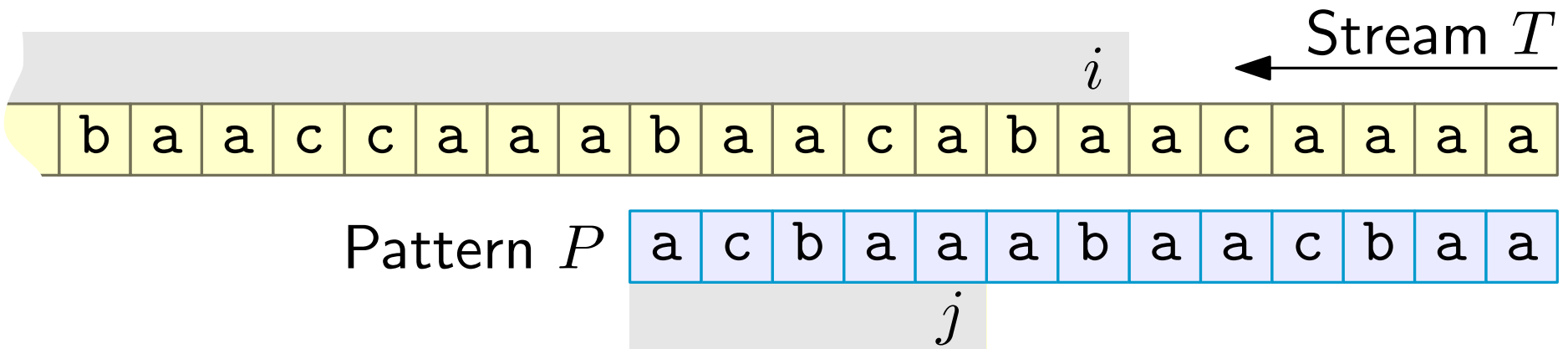
$D[j, i] =$ the minimum of all k-bounded edit distances between pattern prefix $P[0 \ldots j]$ and all suffixes of $T[0 \ldots i]$.

Thus, we want $D[m-1, i]$ as symbol i arrives.

k-difference

 $O(m+ks)$  $O(k)$

Edit operations: **insert**, **delete**, **mismatch**.
Report smallest edit distance between pattern and suffixes of stream if k or less.



Dynamic programming

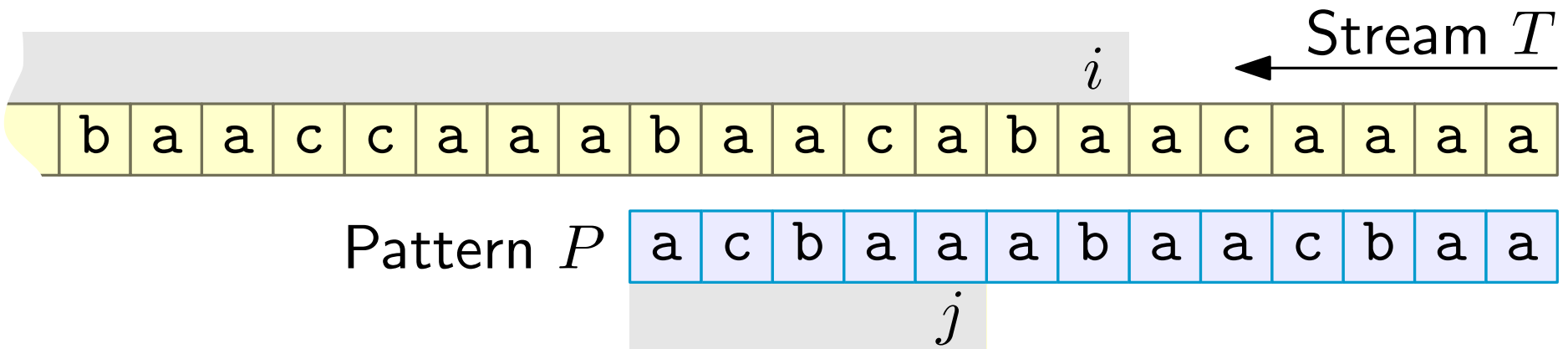
$D[j, i] =$ the minimum of all k-bounded edit distances between pattern prefix $P[0 \ldots j]$ and all suffixes of $T[0 \ldots i]$.
Thus, we want $D[m-1, i]$ as symbol i arrives.

$$D[j,i] = \min \begin{cases} D[j,\ i-1] + 1 & \text{(insert)} \\ D[j-1,\ i] + 1 & \text{(delete)} \\ D[j-1,\ i-1] + 1 - \mathsf{eq}(i,j) & \text{(mismatch)} \\ k+1 & (k\text{-bounded}) \end{cases}$$

k-difference

 $O(m+ks)$  $O(k)$

Edit operations: **insert**, **delete**, **mismatch**.
Report smallest edit distance between pattern and suffixes of stream if k or less.



Question
How do we compute this fast for a stream with small working memory?

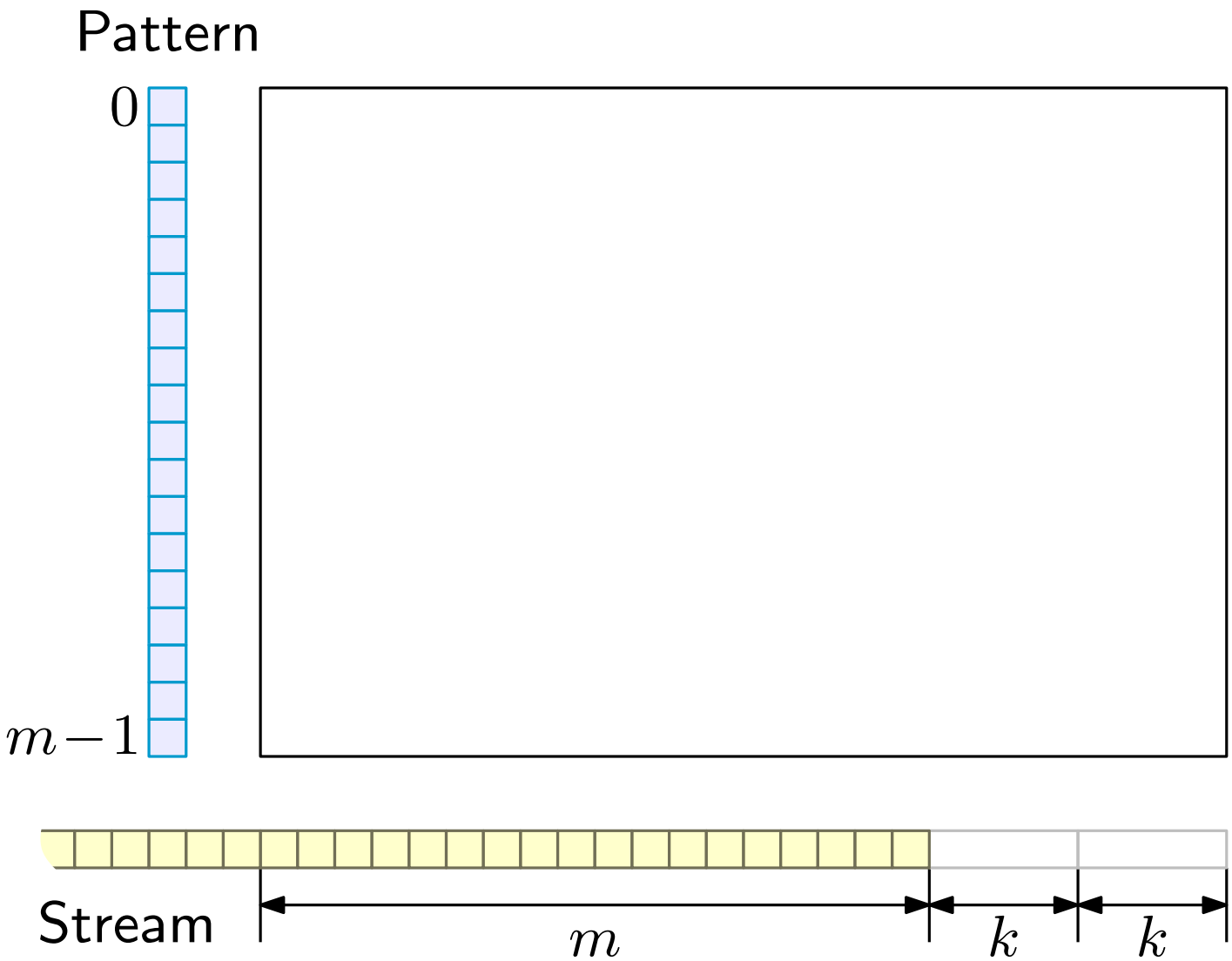
Thus, we want $D[n, i]$ as symbol i arrives.

$$D[j,i] = \min \begin{cases} D[j,\ i-1] + 1 & \text{(insert)} \\ D[j-1,\ i] + 1 & \text{(delete)} \\ D[j-1,\ i-1] + 1 - \mathsf{eq}(i,j) & \text{(mismatch)} \\ k+1 & (k\text{-bounded}) \end{cases}$$

k-difference

 $O(m+ks)$  $O(k)$

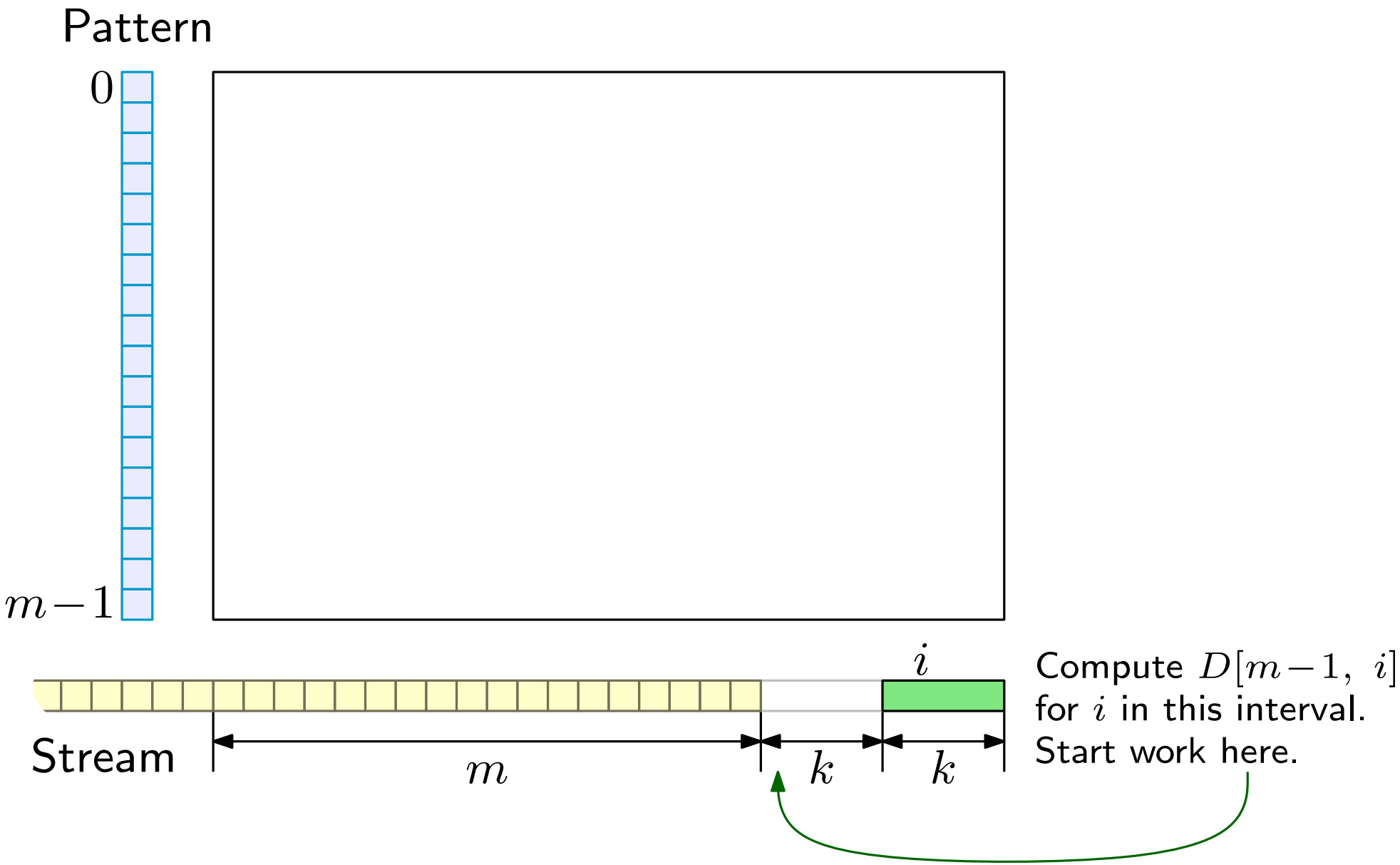
Dynamic programming table



k-difference

 $O(m+ks)$  $O(k)$

Dynamic programming table

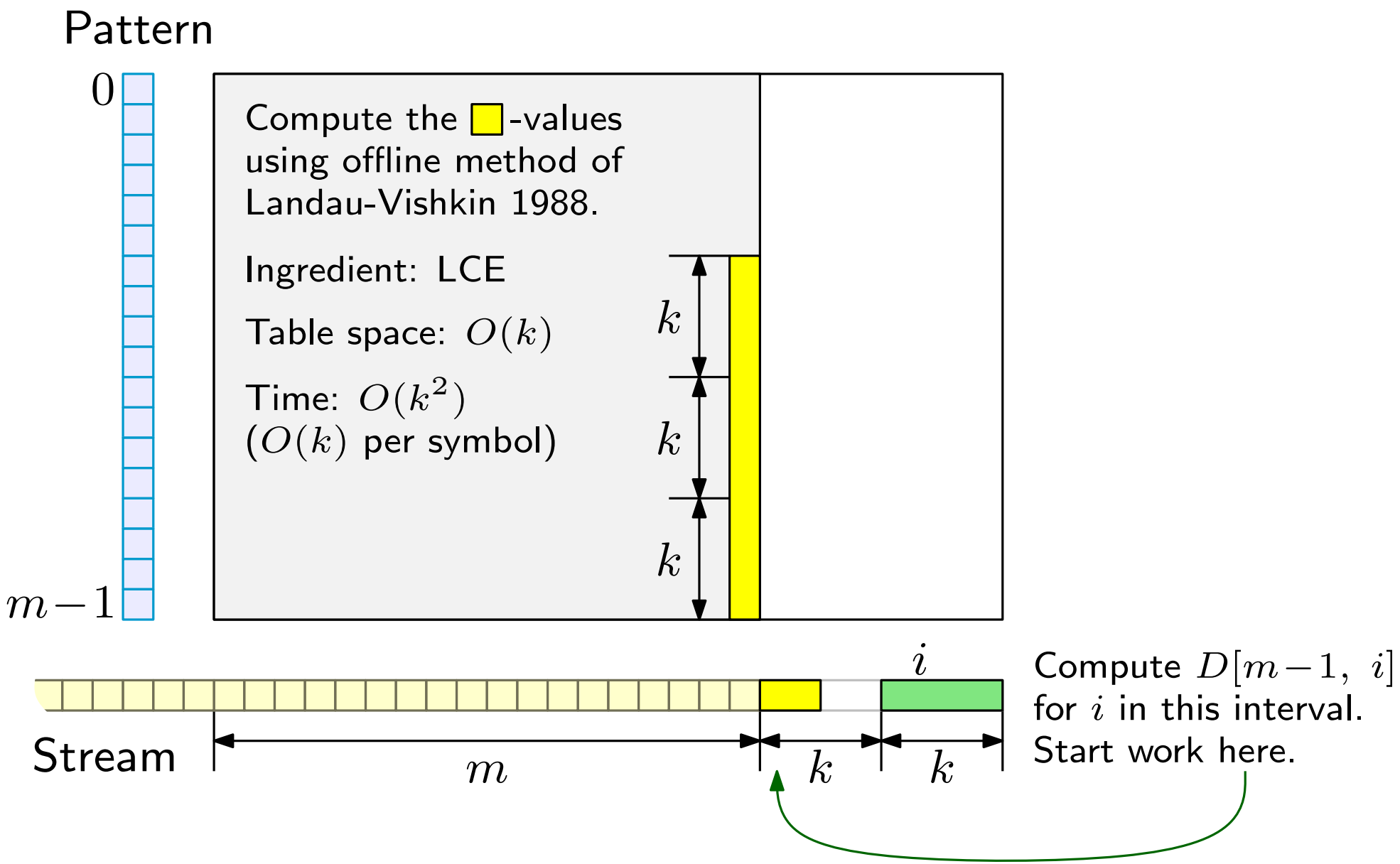


k-difference


$O(m+ks)$


$O(k)$

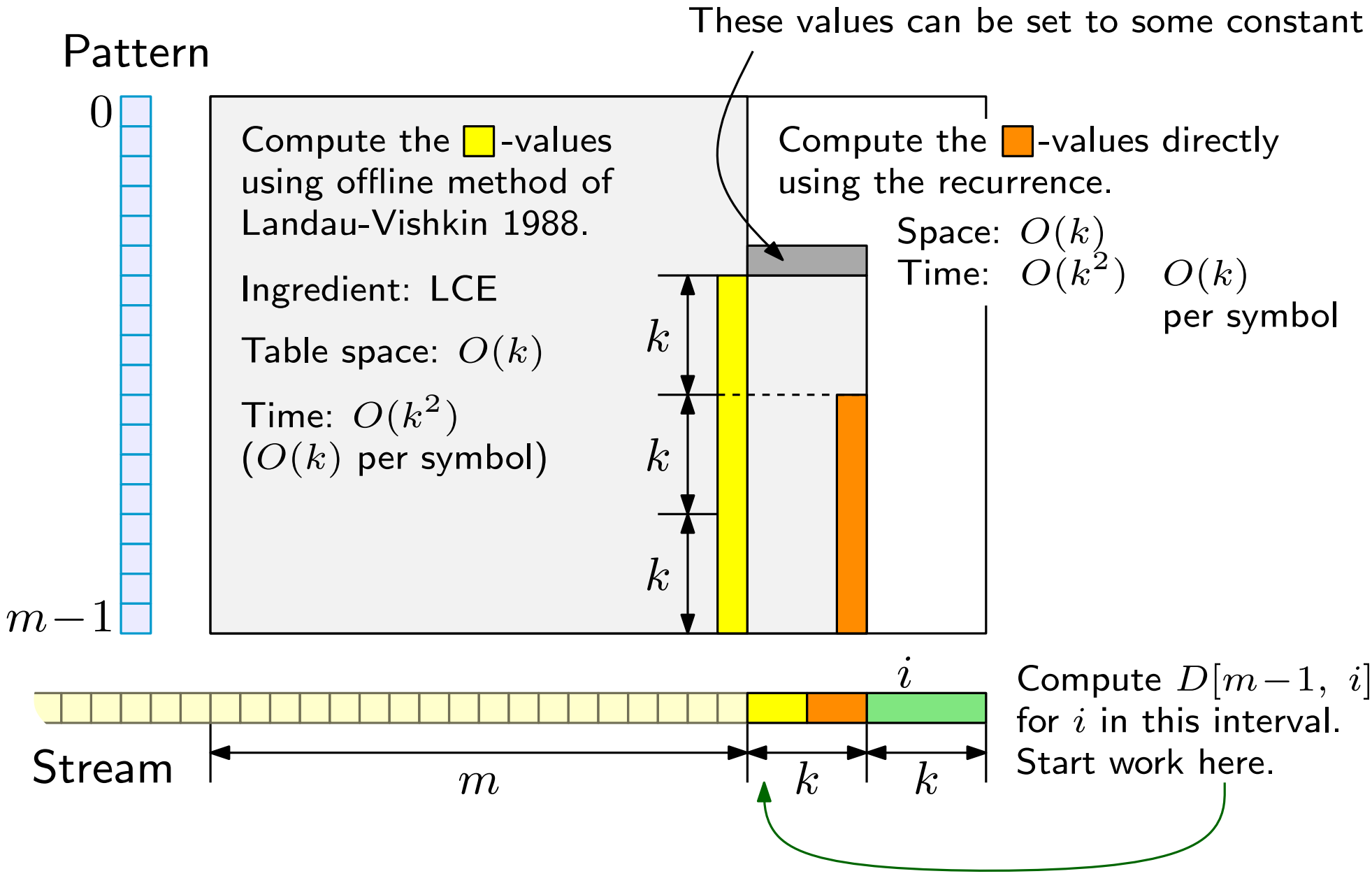
Dynamic programming table



k-difference

 $O(m+ks)$  $O(k)$

Dynamic programming table

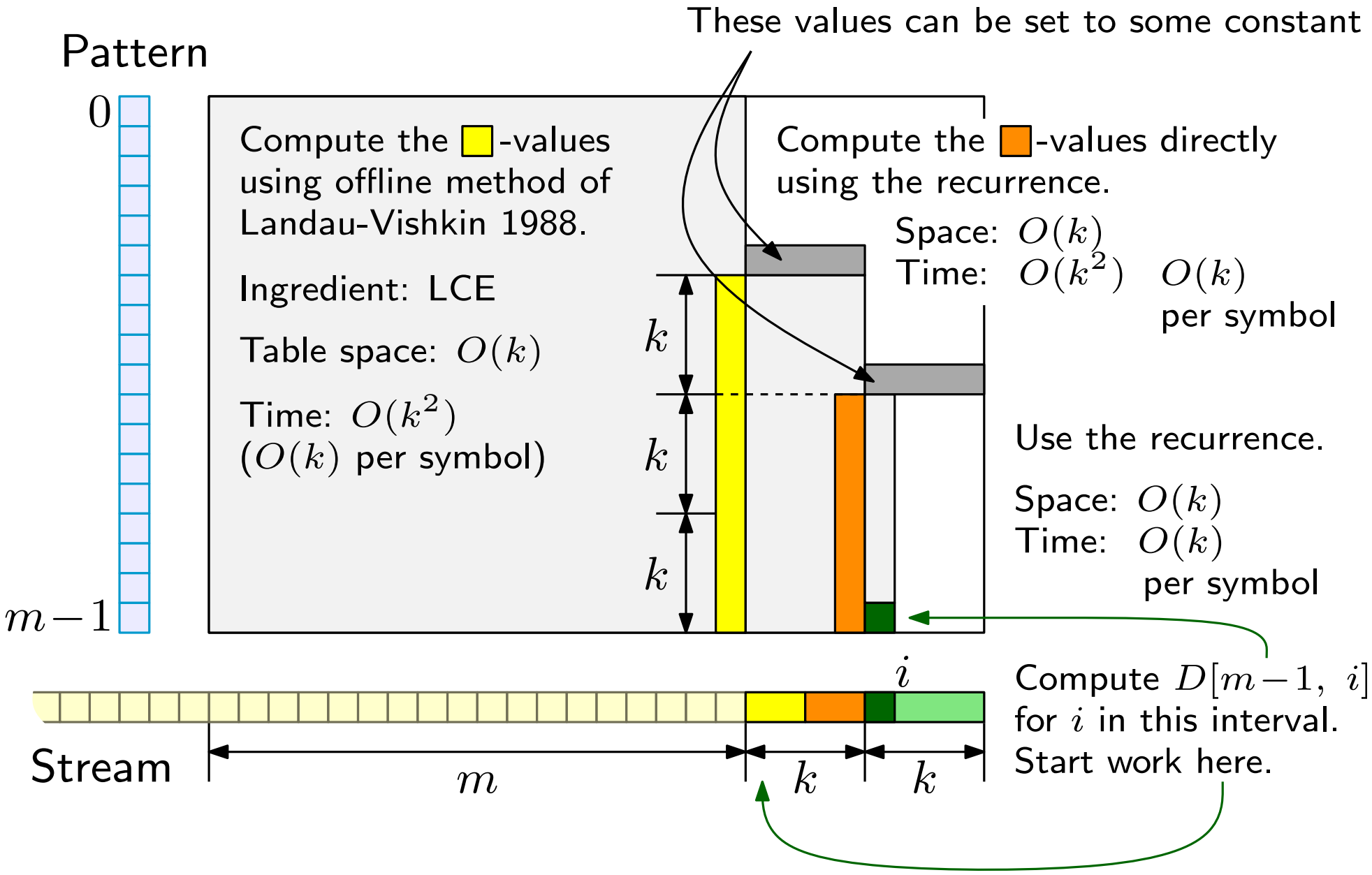


k-difference


$O(m+ks)$


$O(k)$

Dynamic programming table

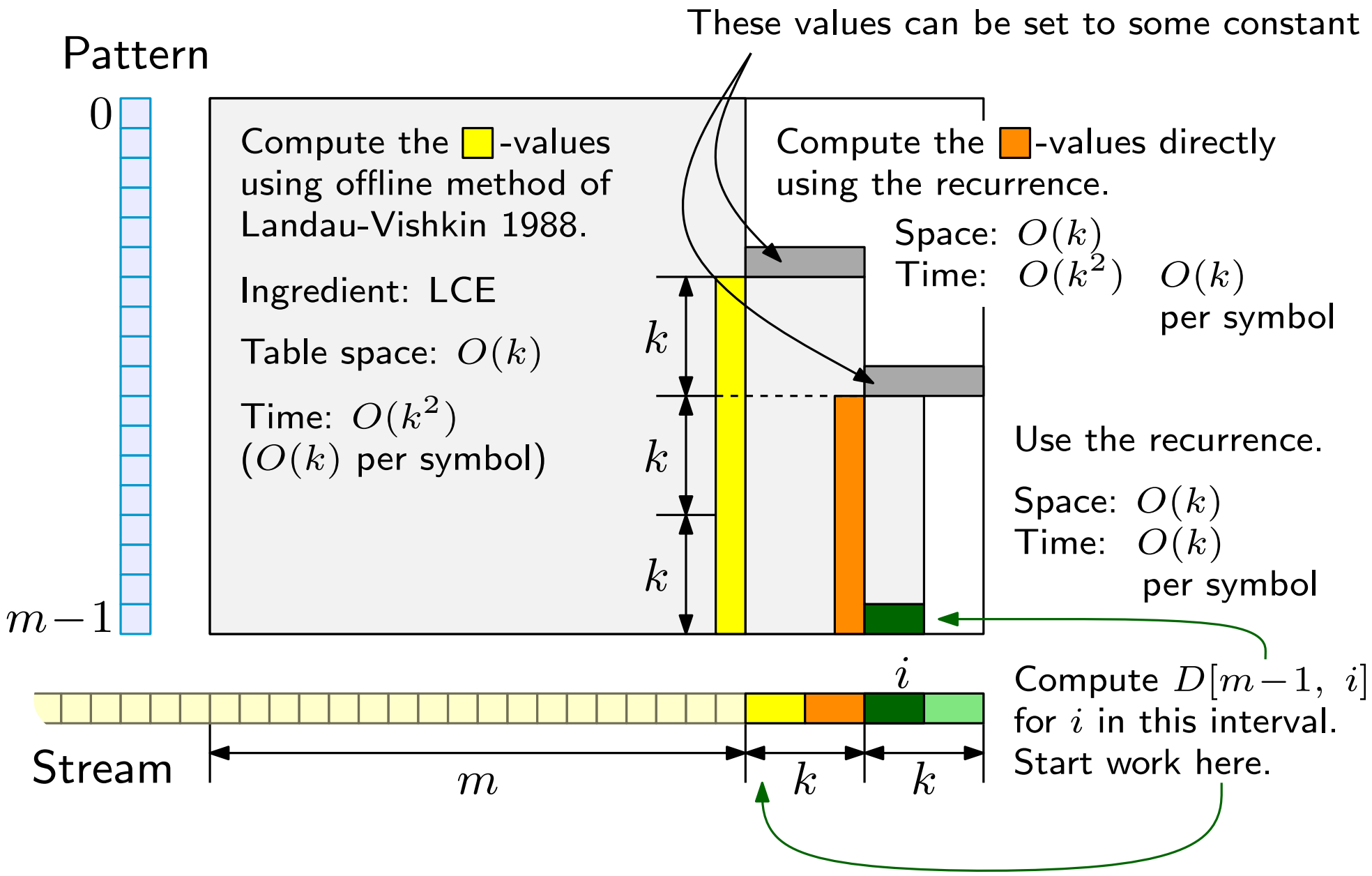


k-difference


$O(m+ks)$


$O(k)$

Dynamic programming table

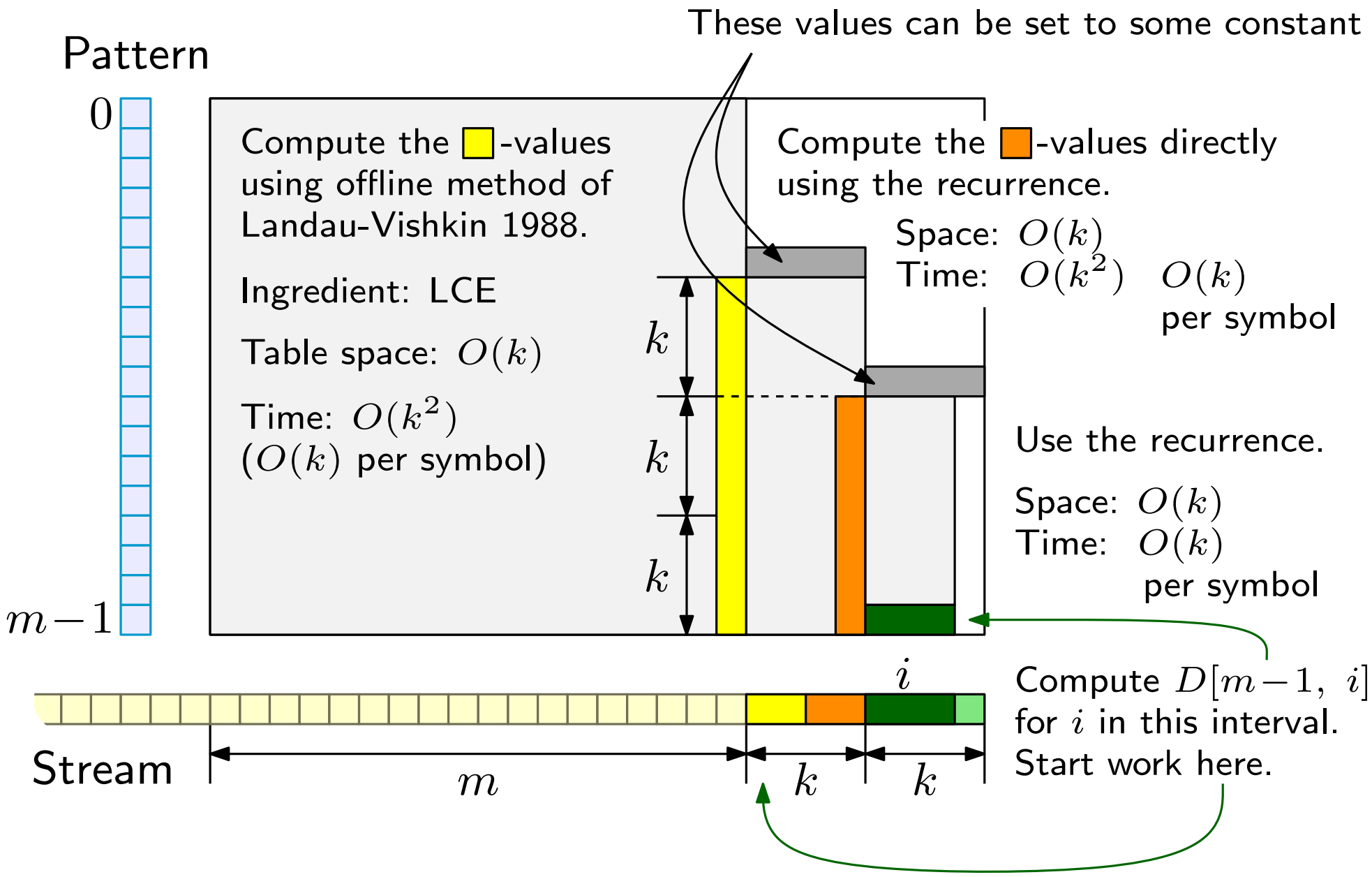


k-difference


$O(m+ks)$


$O(k)$

Dynamic programming table

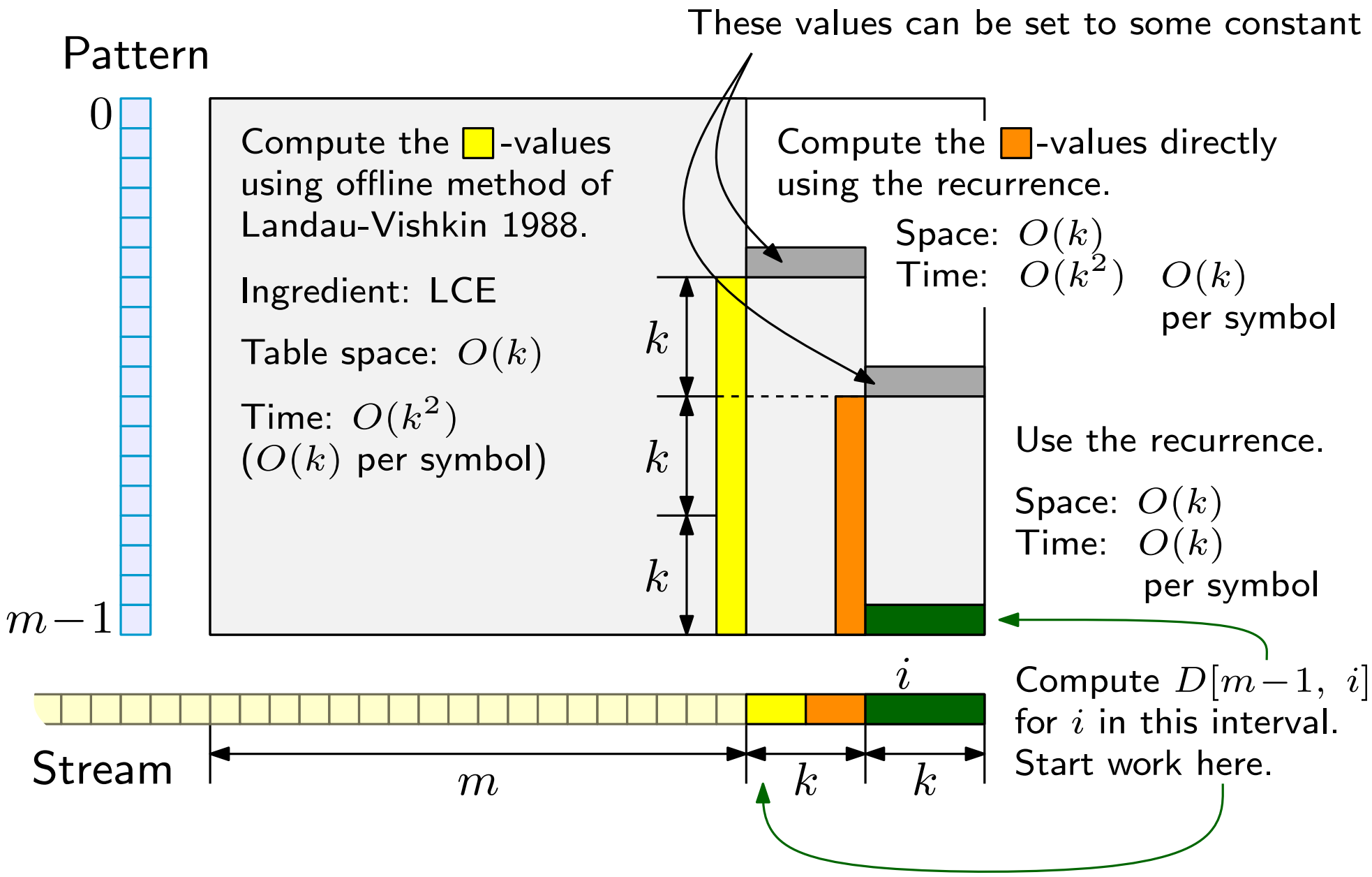


k-difference


$O(m+ks)$


$O(k)$

Dynamic programming table



k-difference

Dynamic programming table

Pattern

0

These values can be set to some constant

Compute the -values using offline method of Landau-Vishkin 1988.

Ingredient: LCE

Table space: $O(k)$

Time: $O(k^2)$
($O(k)$ per symbol)

Compute the -values directly using the recurrence.

Space: $O(k)$
Time: $O(k^2)$ $O(k)$ per symbol

k

k

Use the recurrence.

Space: $O(k)$
Time: $O(k)$ per symbol

Run **two** dynamic programming processes in parallel to cover every symbol in the stream.

i

Compute $D[m-1,\ i]$ for i in this interval. Start work here.

k

k

k-difference

Dynamic programming table

Pattern

These values can be set to some constant

Shared $O(m)$ space for LCE queries.

Each process: $O(k)$ space and $O(k)$ time per arriving symbol.

Multiple streams: $O(m + ks)$ space and $O(k)$ time.

$(O(k)$ per symbol$)$

k

k

Space: $O(k)$
Time: $O(k)$
per symbol

Run **two** dynamic programming processes in parallel to cover every symbol in the stream.

i

Compute $D[m-1,\ i]$ for i in this interval. Start work here.

k

k

One-way communication complexity

The equality problem

Is my string equal to Alice's?

0010101010001011101

n bits

0010101010101001

n bits

One-way communication complexity

The equality problem

Is my string equal to Alice's?

Alice must send n bits.

0010101010001011 01

n bits

0010101010101010 01

n bits

One-way communication complexity

The indexing problem

What's the bit at position i of Alice's string?

0010101010001011101

n bits

Index i

One-way communication complexity

The indexing problem

What's the bit at position i of Alice's string?

Alice must send n bits.

0010101010001011101

n bits

Index i

Space lower bound (k-mismatch/difference)

Part 1 – the equality problem

Has pattern P over alphabet Σ
$= m \log |\Sigma|$ bits.

Bit string T of length $m \log |\Sigma|$.

Space lower bound (k-mismatch/difference)

Part 1 – the equality problem

Step 1
Sends internal state of pattern matching machine on P.

Has pattern P over alphabet Σ
$= m \log |\Sigma|$ bits.

Bit string T of length $m \log |\Sigma|$.

Space lower bound (k-mismatch/difference)

Part 1 – the equality problem

Step 1
Sends internal state of pattern matching machine on P.

Has pattern P over alphabet Σ
$= m \log |\Sigma|$ bits.

Bit string T of length $m \log |\Sigma|$.

Step 2
Bob feeds T into one stream to determine if $P = T$.

Space lower bound (k-mismatch/difference)

Part 1 – the equality problem

Step 1
Sends internal state of pattern matching machine on P.

Has pattern P over alphabet Σ
$= m \log |\Sigma|$ bits.

Bit string T of length $m \log |\Sigma|$.

Step 2
Bob feeds T into one stream to determine if $P = T$.

Conclusion: Space must be $\Omega(m \log |\Sigma|)$ bits.

Space lower bound (k-mismatch/difference)

Part 2 – the indexing problem

Has ks-length bit string

Index i

Space lower bound (k-mismatch/difference)

Part 2 – the indexing problem

Has ks-length bit string

Index i

Step 1

Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ (m zeros).
Feeds in k bits into each stream.

Space lower bound (k-mismatch/difference)

Part 2 – the indexing problem

Step 2

Sends internal state

Has ks-length bit string

Index i

Step 1

Pattern matching machine:

Alphabet $\Sigma = \{0, 1\}$.

$P = 00 \cdots 0$ (m zeros).

Feeds in k bits into each stream.

Space lower bound (k-mismatch/difference)

Part 2 – the indexing problem

Step 2

Sends internal state

Has ks-length bit string

Index i

Step 1

Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ (m zeros).
Feeds in k bits into each stream.

Step 3

Bob feeds 0s into the appropriate stream, takes the outputted distance, feeds in another 0 and compare the two distances. This reveals the bit at position i.

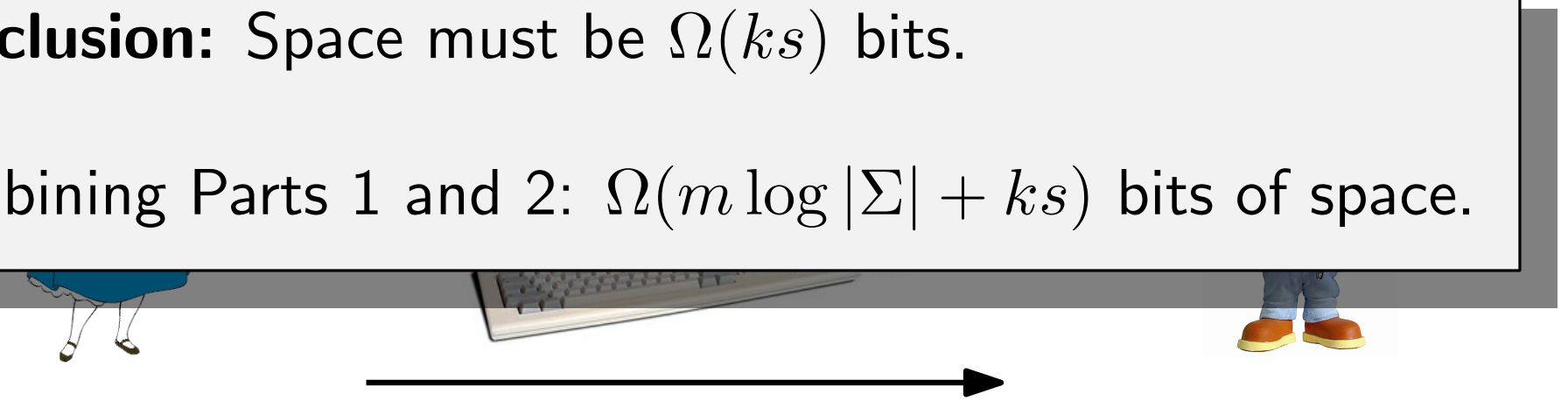
Space lower bound (k-mismatch/difference)

Part 2 – the indexing problem

Step 2

Conclusion: Space must be $\Omega(ks)$ bits.

Combining Parts 1 and 2: $\Omega(m \log |\Sigma| + ks)$ bits of space.



Has ks-length bit string

Index i

Step 1

Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ (m zeros).
Feeds in k bits into each stream.

Step 3

Bob feeds 0s into the appropriate stream, takes the outputted distance, feeds in another 0 and compare the two distances. This reveals the bit at position i.

Space lower bound (k-mismatch/difference)

Part 2 – the indexing problem

Step 2

Conclusion: Space must be $\Omega(ks)$ bits.

Combining Parts 1 and 2: $\Omega(m \log |\Sigma| + ks)$ bits of space.

The bounds $\Omega(m \log |\Sigma| + s)$ for **exact matching** and $\Omega(ms)$ for L_1, L_2, **Hamming distance** and **convolution** are obtained similarly.

Pattern matching machine. Bob feeds us into the appropriate

Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ (m zeros).
Feeds in k bits into each stream.

stream, takes the outputted distance, feeds in another 0 and compare the two distances. This reveals the bit at position i.

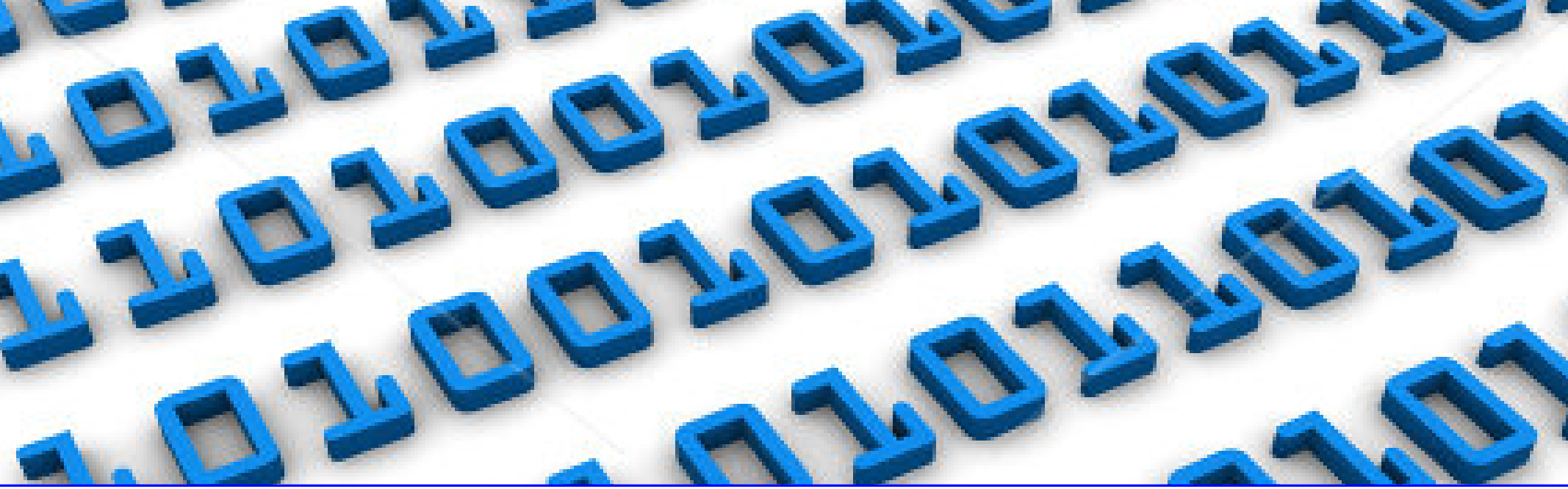
Open problems

- Close the gap for k-mismatch:
  Our $O(k)$ time versus $O(\sqrt{k \log k})$ offline.

  Potentially exponential gap for constant size alphabets:
  Our $O(k)$ time versus $O(\log^2 m)$ in a single stream.

- Randomised space lower bound for k-mismatch/difference is $O(\log m + ks)$ and $O(\log m + s)$ for exact matching.
  Can we get (near) matching upper bounds?

- Conjecture: for every multiple-streams algorithm, there is an equivalent (time and space) one with read-only space that is independent of s (like our bounds).



Thank you!