Three strategies for the dead-zone string matching algorithm

J. Daykin, R. Groult, Y. Guesnet, T. Lecroq, A. Lefebvre, M. Léonard, L. Mouchard, É. Prieur-Gaston and B. Watson

SeqBio 2018

19 – 20 November 2018 – Rouen, France

Outline

Outline

Notations

- finite alphabet Σ
- string $x[0\,..\,m-1]$ on Σ^*
- length $|x| = m$
- $\tilde{x}$ is the reverse of x ($x[m-1]x[m-2]\cdots x[1]x[0]$)
- $x[i\,..\,j]$ is a factor (substring) of x from position i to position j (both inclusive)
- $x[0\,..\,i]$ is a prefix
- $x[i\,..\,m-1]$ is a suffix
- u is a border of x if u is both a prefix and a suffix of x
- *Border*(x) is the longest border of x

Exact String Matching

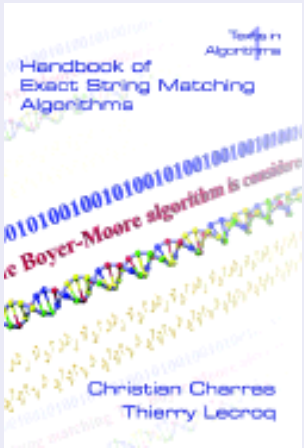
Problem

Searching for all exact occurrences of a pattern x ($|x| = m$) in a text y ($|y| = n$)

2 variants

- on-line (preprocessing of the pattern)
- off-line (preprocessing of the text)

Exact On-Line String Matching



http://www-igm.univ-mlv.fr/~lecroq/string/

Christian Charras and Thierry Lecroq
Handbook of exact string matching algorithms
King's College Publications, 2004

SMART
String Matching Algorithms Research Tool

https://smart-tool.github.io/smart/

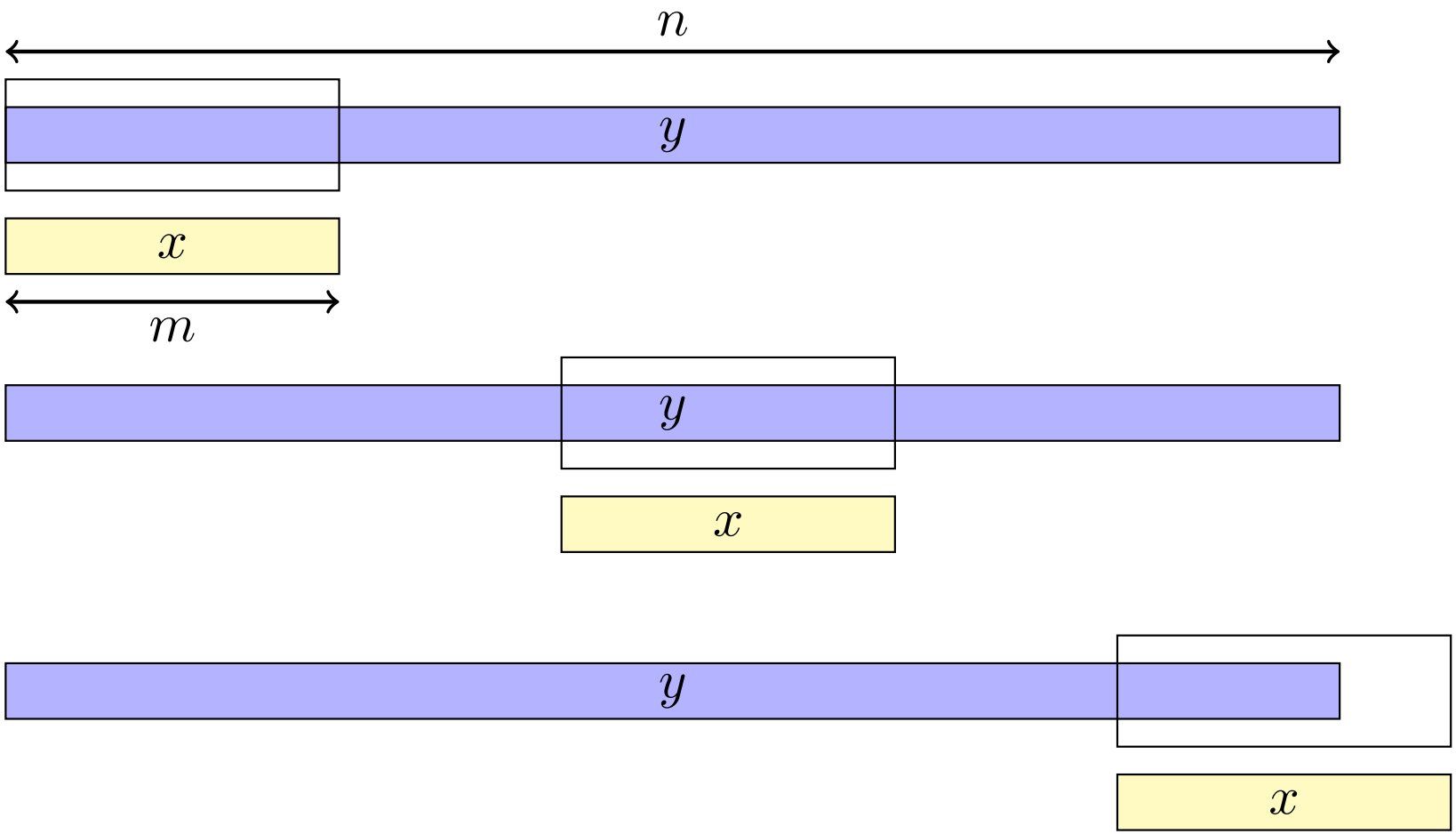
Simone Faro and Thierry Lecroq
The Exact Online String Matching Problem: a Review of the Most Recent Results
ACM Computing Surveys **45**(2) (2013) 13

Sliding window

Classical solutions (KMP, BM, ...)

Preprocessing of the pattern and use of a sliding window

Sliding window

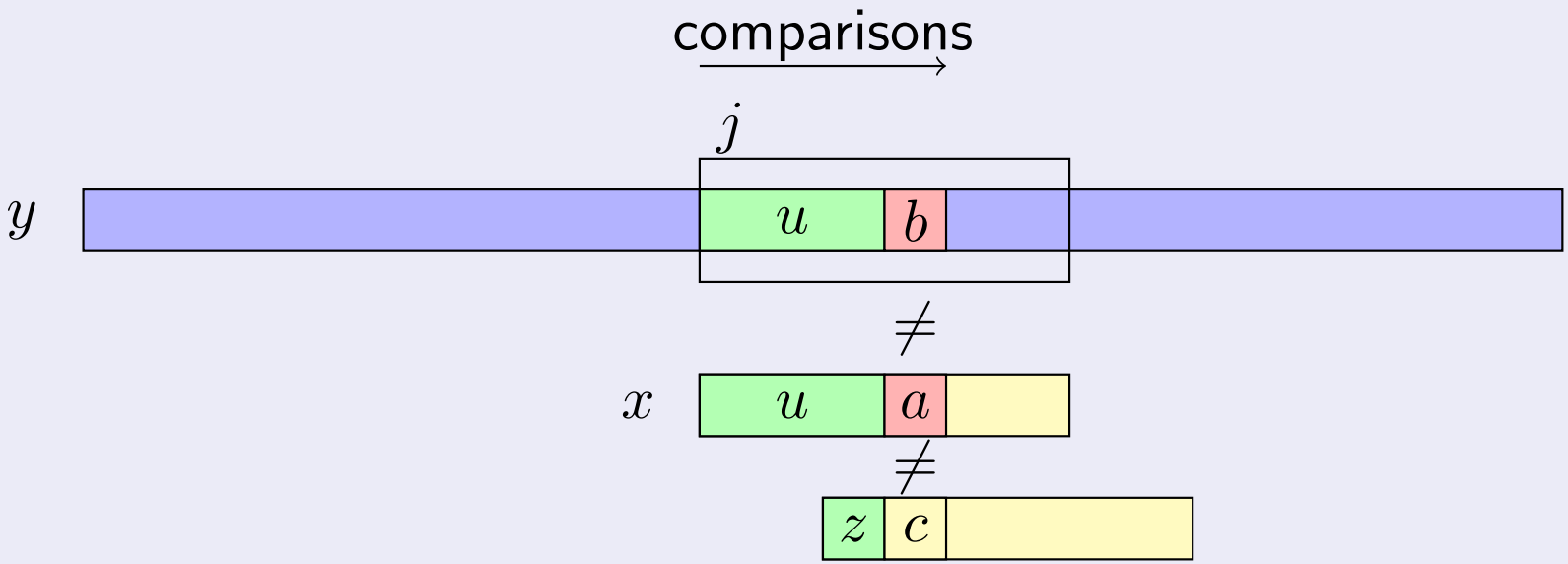


Sliding window

An on-line exact string matching algorithm can then be viwed as a succession of:

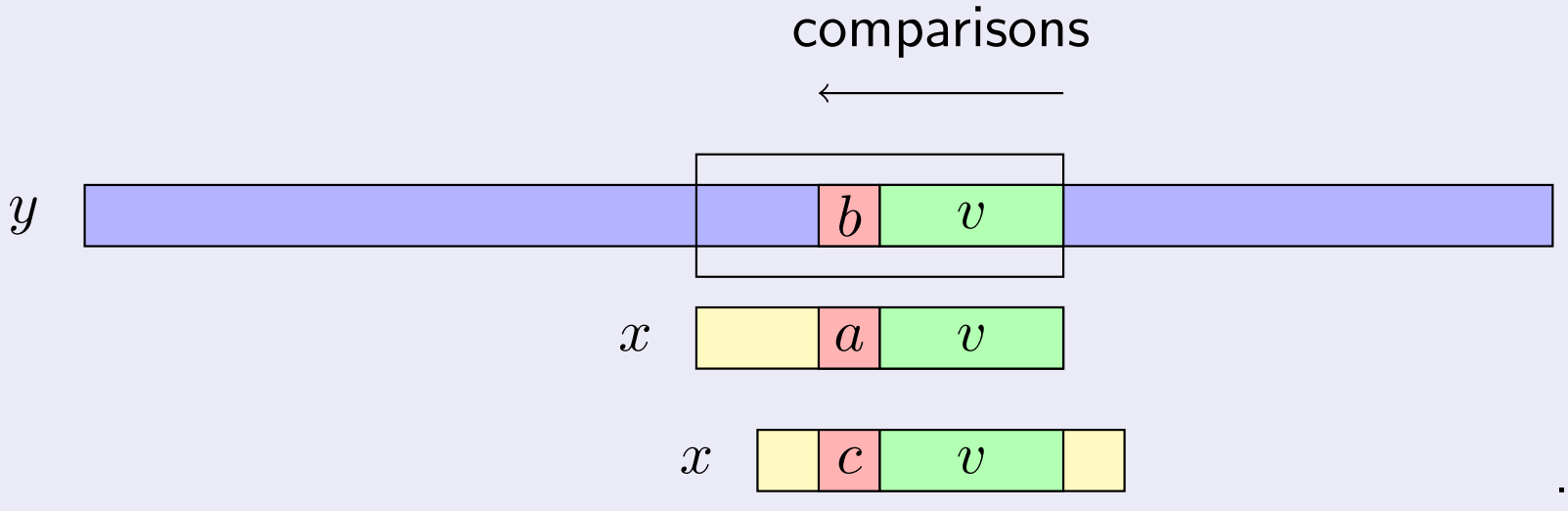
- attempts (comparison of the window content and the pattern);
- shift (of the window to the right).

Knuth-Morris-Pratt algorithm (1977)



$k = \min\{\ell \mid x[|Border^{\ell}(u)|] \neq a\}$ and $z = Border^{k}(u)$

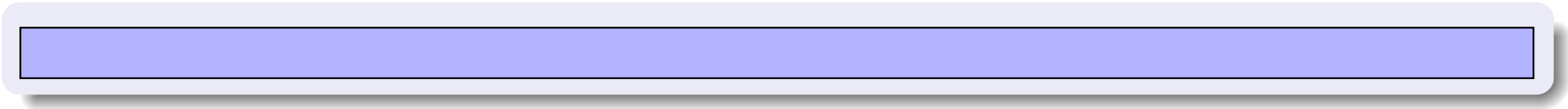
Boyer-Moore algorithm (1977)



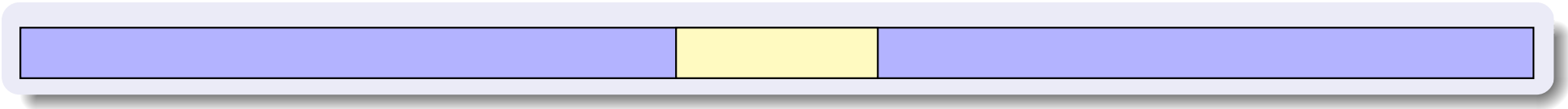
Dead Zone strategy

Bruce W. Watson and Richard E. Watson
A New Family of String Pattern Matching Algorithms
In: Jan Holub editor, *Proceedings of the Prague Stringology Club Workshop 1997, Prague, Czech Republic, July 7, 1997*, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University, 12–23.

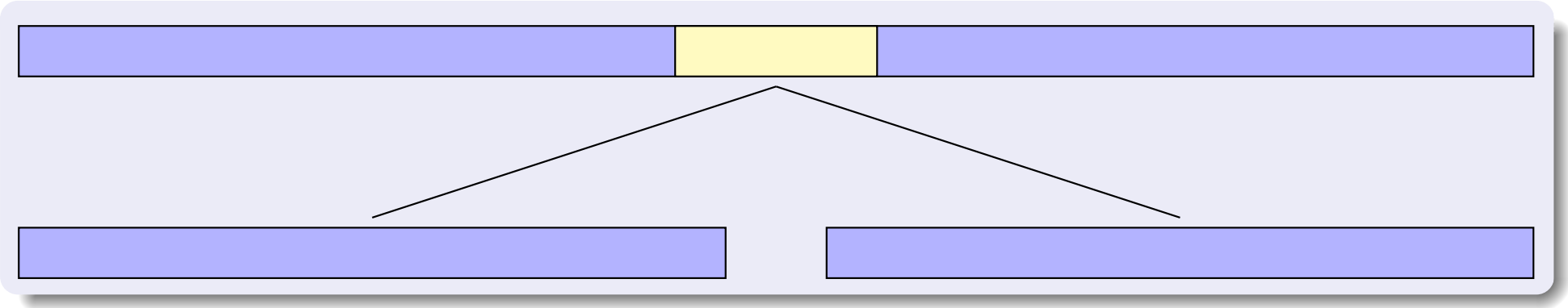
Dead Zone strategy



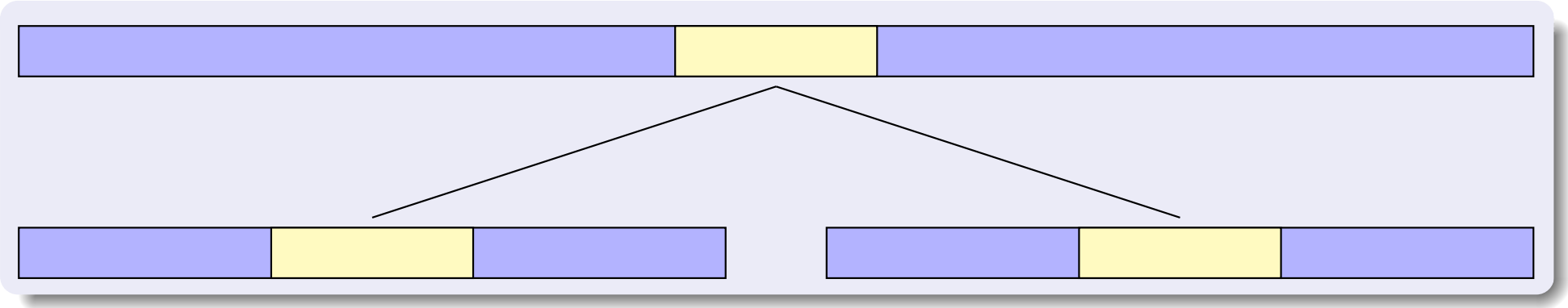
Dead Zone strategy



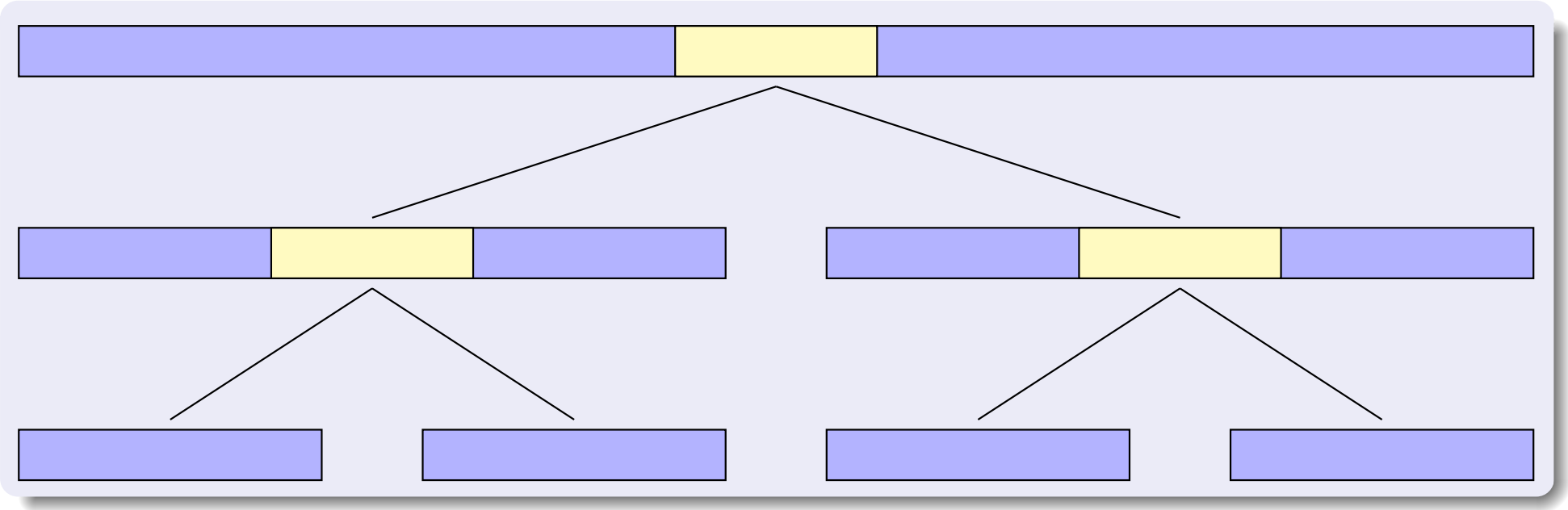
Dead Zone strategy



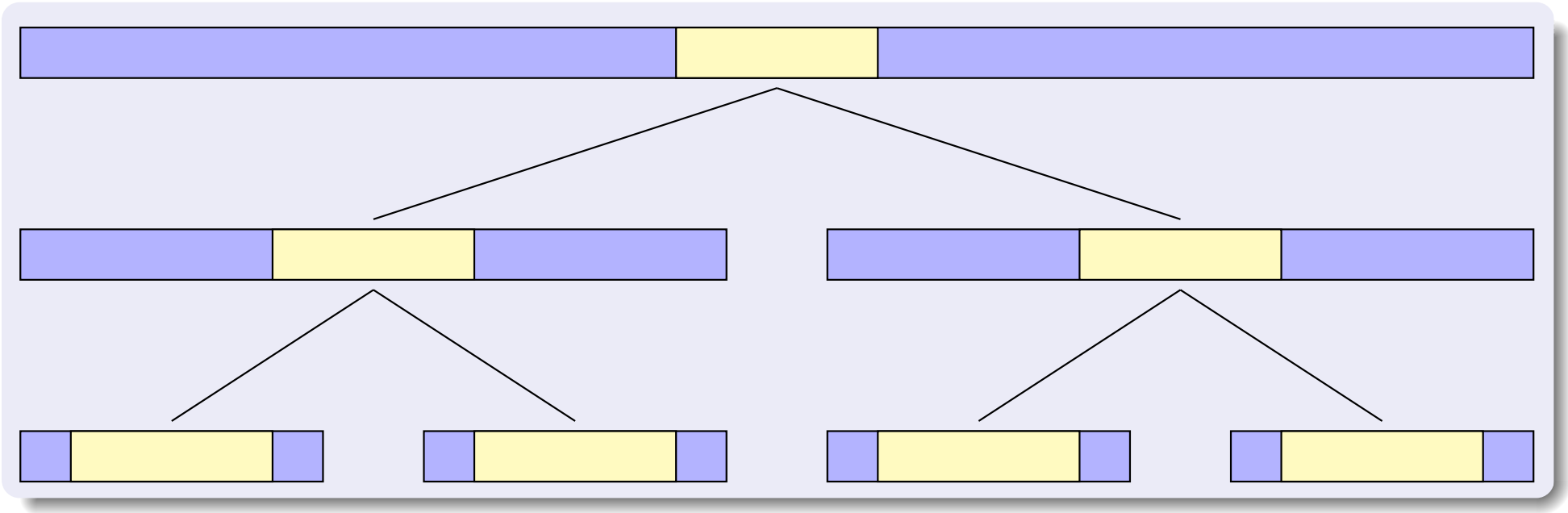
Dead Zone strategy



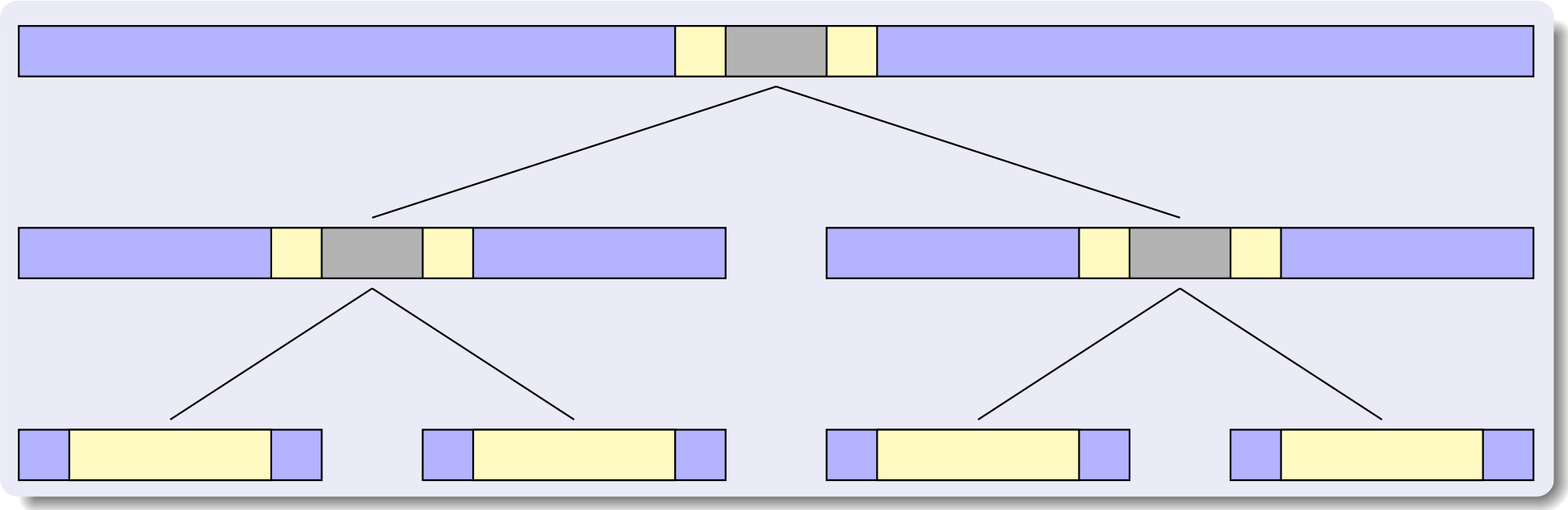
Dead Zone strategy



Dead Zone strategy



Dead Zone strategy



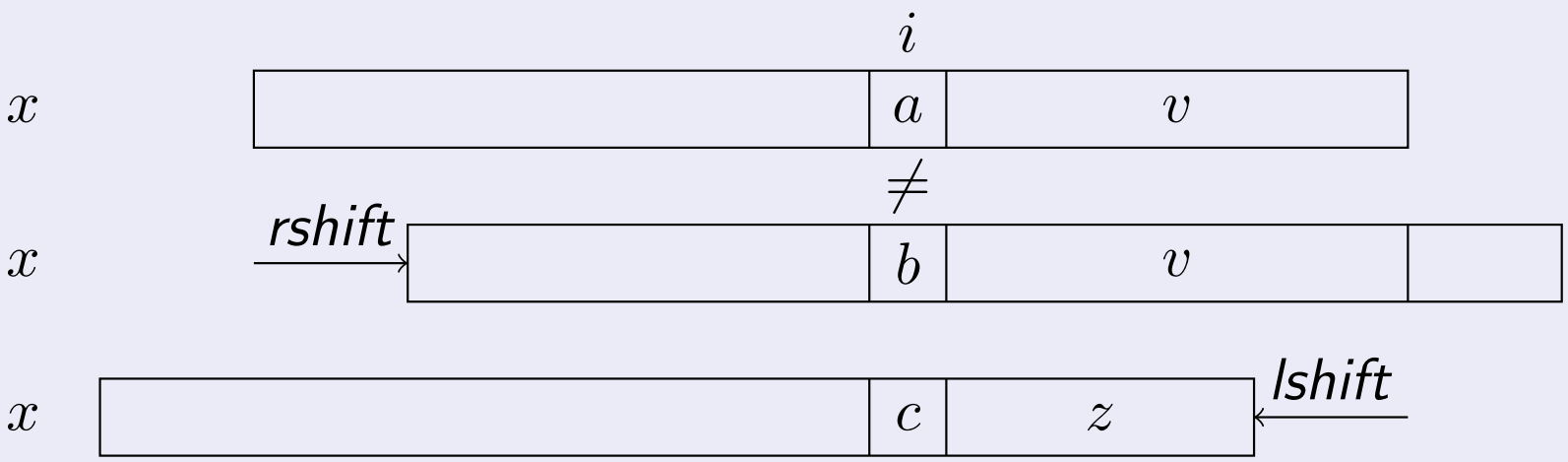
Our contributions

Three strategies

- Right-to-left
- Right-to-left with memory
- Alternating searching: right – left

Outline

Right-to-left



A suffix v of the pattern matches in the text and a mismatch occurs with a at position i in the pattern.

The right shift (*rshift*) consists in finding a re-occurrence of v in the pattern preceded by a symbol b different from a.

The left shift (*lshift*) consists in finding the longest suffix z of the pattern preceded by a symbol c different from a.

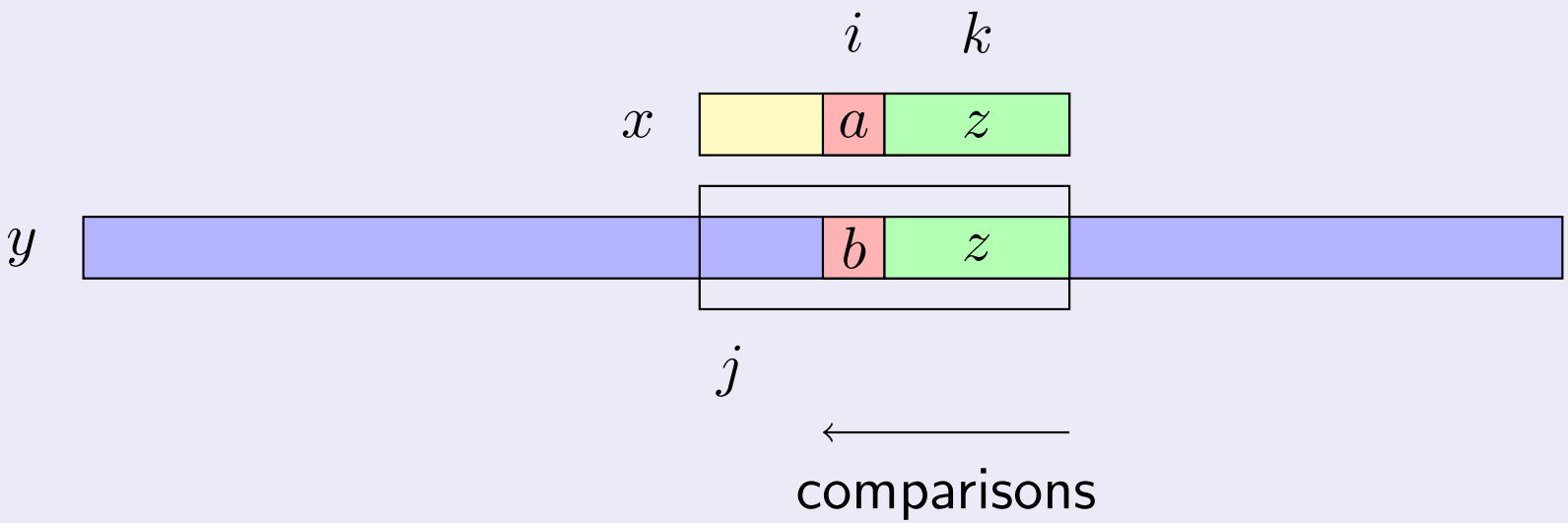
Right-to-left

- right shift: similar as in the Boyer-Moore algorithm
- left shift: similar as in the Knuth-Morris-Pratt algorithm (but from right to left)

Preprocessing phase linear in time and space

Outline

Right-to-left with memory

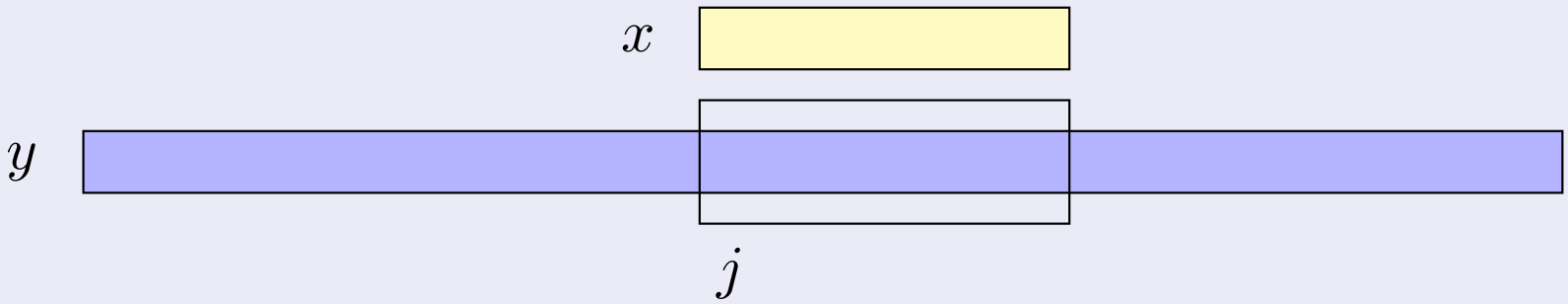


When $x[i] \neq y[j+i]$ and $x[i+1\,..\,m-1] = y[j+i+1\,..\,j+m-1]$ then

- $skip_1[j+k] = k$ and
- $skip_2[j+k] = k-i$

for $i+1 \leq k \leq m-1$

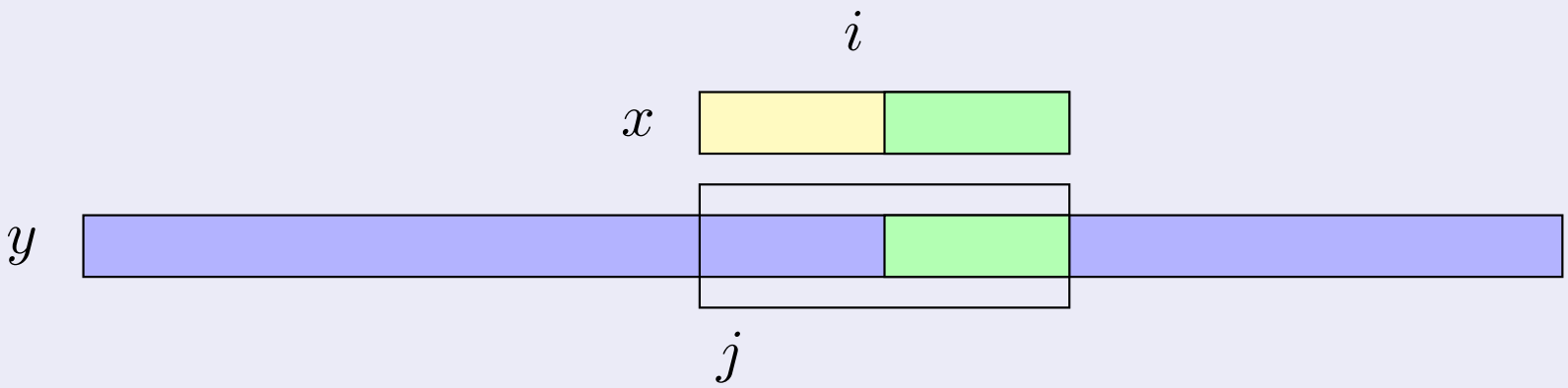
Right-to-left with memory



If $k = \mathit{skip}_2[i+j] > 0$, it means that $x[k-\ell+1\,..\,k] = y[i+j-\ell+1\,..\,i+j]$ with $\ell = \mathit{skip}_1[i+j]$, and furthermore $x[k-\ell] \neq y[i+j-\ell]$ if $k \geq \ell$.

We need to know whether $y[i+j-\ell+1\,..\,i+j]{=}x[i-\ell+1\,..\,i]$ and thus we need to know whether $x[k-\ell+1\,..\,k]{=}x[i-\ell+1\,..\,i]$.

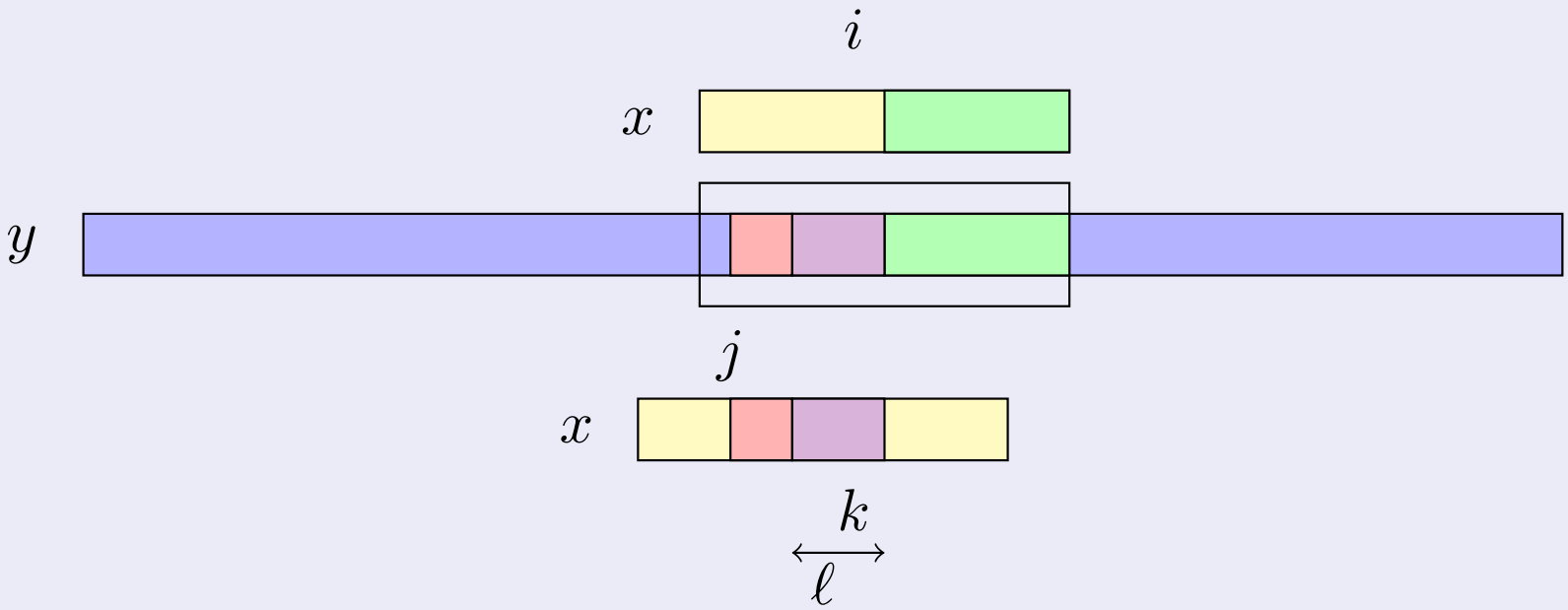
Right-to-left with memory



If $k = \mathit{skip}_2[i+j] > 0$, it means that $x[k-\ell+1\,..\,k] = y[i+j-\ell+1\,..\,i+j]$ with $\ell = \mathit{skip}_1[i+j]$, and furthermore $x[k-\ell] \neq y[i+j-\ell]$ if $k \geq \ell$.

We need to know whether $y[i+j-\ell+1\,..\,i+j]{=}x[i-\ell+1\,..\,i]$ and thus we need to know whether $x[k-\ell+1\,..\,k]{=}x[i-\ell+1\,..\,i]$.

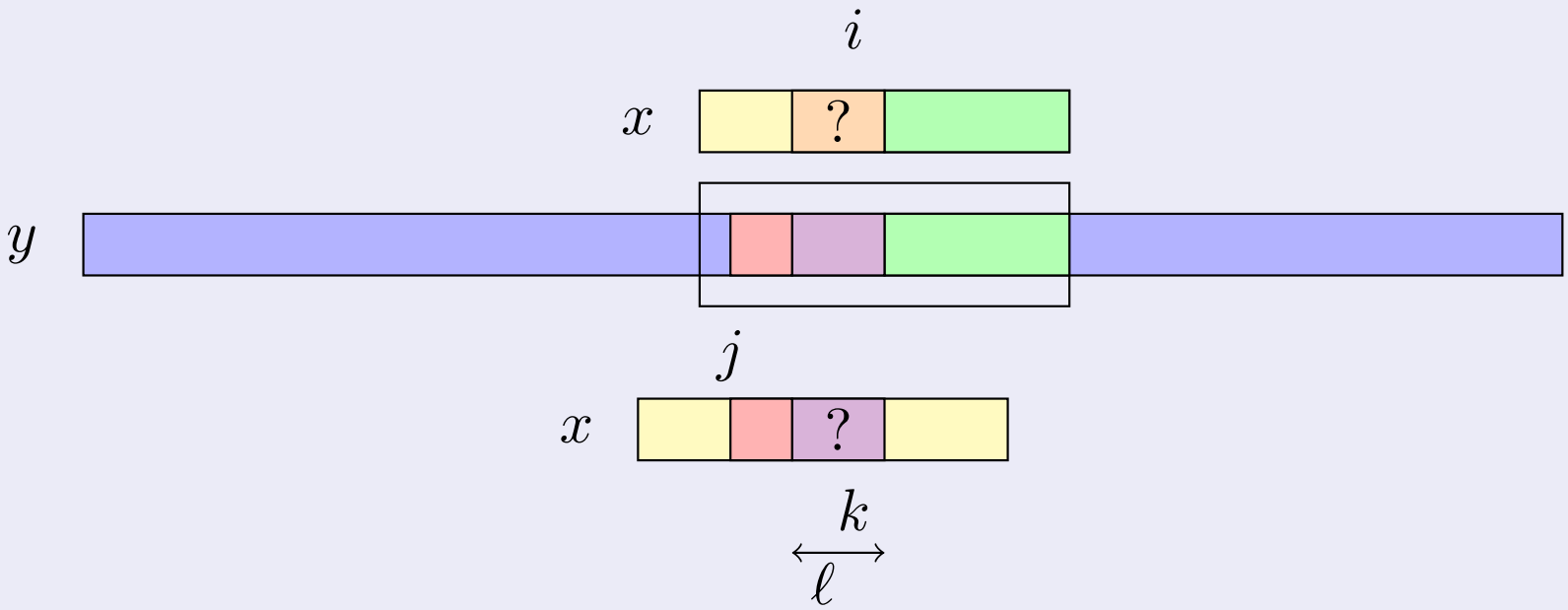
Right-to-left with memory



If $k = \mathit{skip}_2[i+j] > 0$, it means that $x[k-\ell+1\,..\,k] = y[i+j-\ell+1\,..\,i+j]$ with $\ell = \mathit{skip}_1[i+j]$, and furthermore $x[k-\ell] \neq y[i+j-\ell]$ if $k \geq \ell$.

We need to know whether $y[i+j-\ell+1\,..\,i+j]{=}x[i-\ell+1\,..\,i]$ and thus we need to know whether $x[k-\ell+1\,..\,k]{=}x[i-\ell+1\,..\,i]$.

Right-to-left with memory



If $k = \mathit{skip}_2[i+j] > 0$, it means that $x[k-\ell+1\,..\,k] = y[i+j-\ell+1\,..\,i+j]$ with $\ell = \mathit{skip}_1[i+j]$, and furthermore $x[k-\ell] \neq y[i+j-\ell]$ if $k \geq \ell$.

We need to know whether $y[i+j-\ell+1\,..\,i+j]{=}x[i-\ell+1\,..\,i]$ and thus we need to know whether $x[k-\ell+1\,..\,k]{=}x[i-\ell+1\,..\,i]$.

Right-to-left with memory

$x[k-\ell+1\,..\,k] \stackrel{?}{=} x[i-\ell+1\,..\,i]$

Longest common prefix of the suffixes of $\tilde{x}$ starting at positions $m-1-k$ and $m-1-i$

Can be answer in constant time after linear preprocessing: RMQ on LCP of $\tilde{x}$

Right-to-left with memory

skip_1 and skip_2

- needs a stack: the mismatch position is not known for all the matching positions
- $O(n)$ space

Outline

Alternating searching: right – left

Order of comparisons

$x[m-1], x[0], x[m-2], x[1], x[m-3], \ldots$

4 shift arrays

- right shift after a right mismatch stored in array *rsrm*
- left shift after a right mismatch stored in array *lsrm*
- right shift after a left mismatch stored in array *rslm*
- left shift after a left mismatch stored in array *lslm*

Alternating searching: right – left

2 conditions

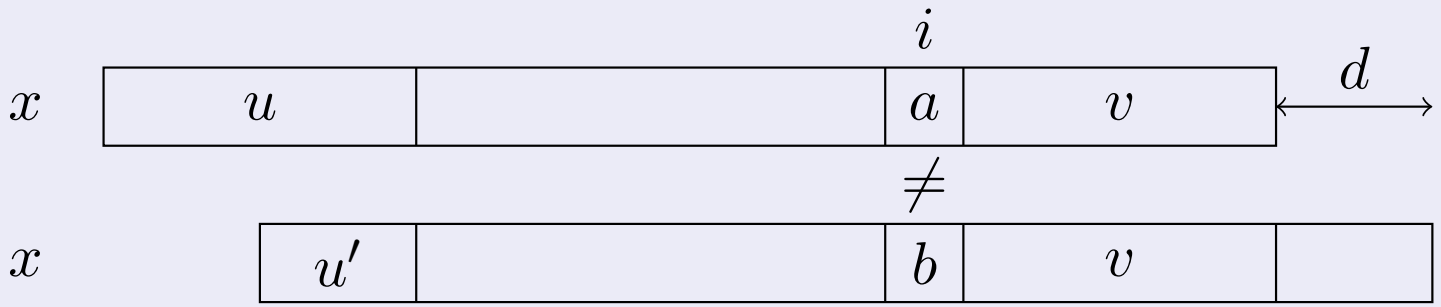
$occCond'_x(i,d) = (0 < d \leq i \text{ and } x[i-d] \neq x[i]) \text{ or } (i < d)$

$$
\begin{aligned}
suffCond'_x(i,d) = \quad & (\quad 0 < d \leq m-2-i \\
& \quad \text{and } x[d\,..\,m-2-i] \text{ is a prefix of } x \\
& \quad \text{and } x[i-d+1\,..\,m-d-1] \text{ is a suffix of } x \quad) \\
& \text{or} \\
& (\quad m-2-i < d \leq i+1 \\
& \quad \text{and } x[i-d+1\,..\,m-d-1] \text{ is a suffix of } x \quad) \\
& \text{or} \\
& (\quad i+1 < d \text{ and } x[0\,..\,m-d-1] \text{ is a suffix of } x \quad)
\end{aligned}
$$

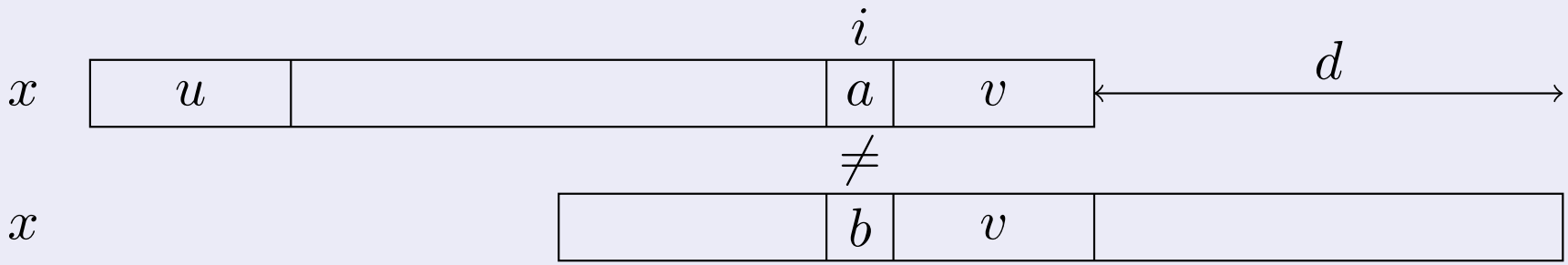
$rsrm[i] = \min\{d \mid occCond'_x(i,d) \text{ and } suffCond'_x(i,d) \text{ are satisfied}\}.$

Right shift after a right mismatch

prefix u and suffix v (of the same length) of x match the text and a mismatch occurs with symbol a at position i of x:

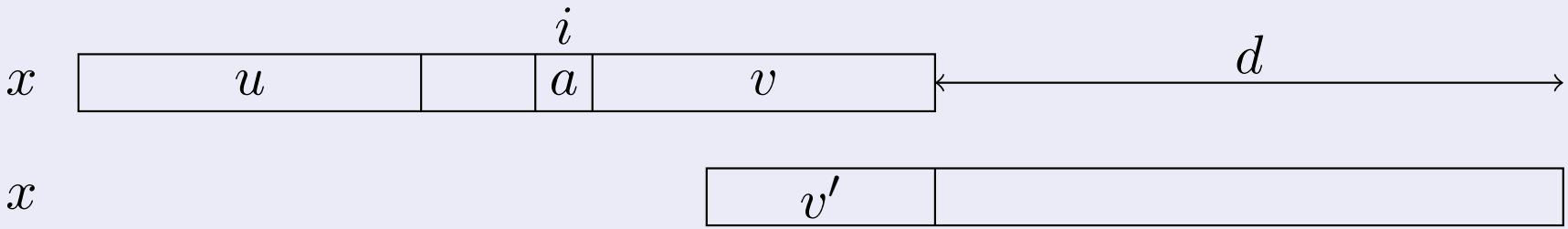


the suffix v of x reoccurs preceded by a symbol b different from a and a prefix u' of x matches a suffix of u;



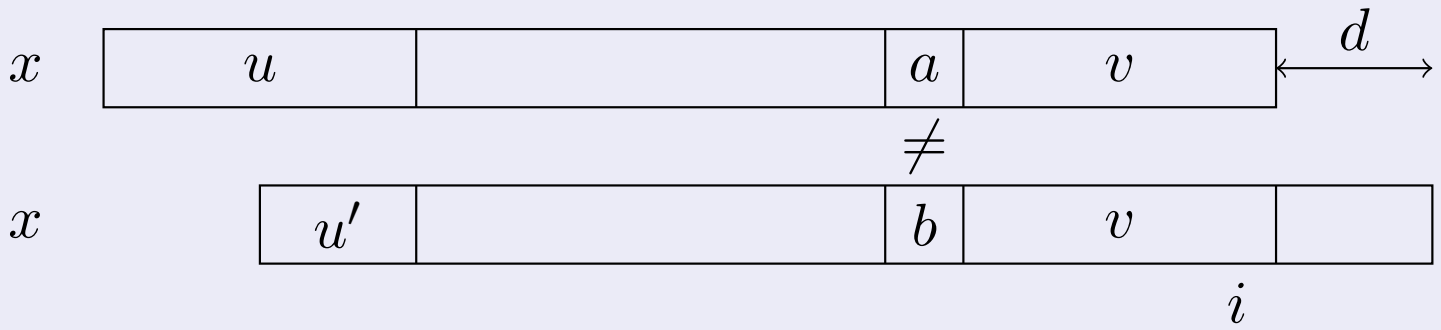
only a suffix v of x reoccurs preceded by a symbol b different from a;

Right shift after a right mismatch



only a prefix v' of x matches a suffix of v.

Right shift after a right mismatch



the suffix v of x reoccurs preceded by a symbol b different from a and a prefix u' of x matches a suffix of u;

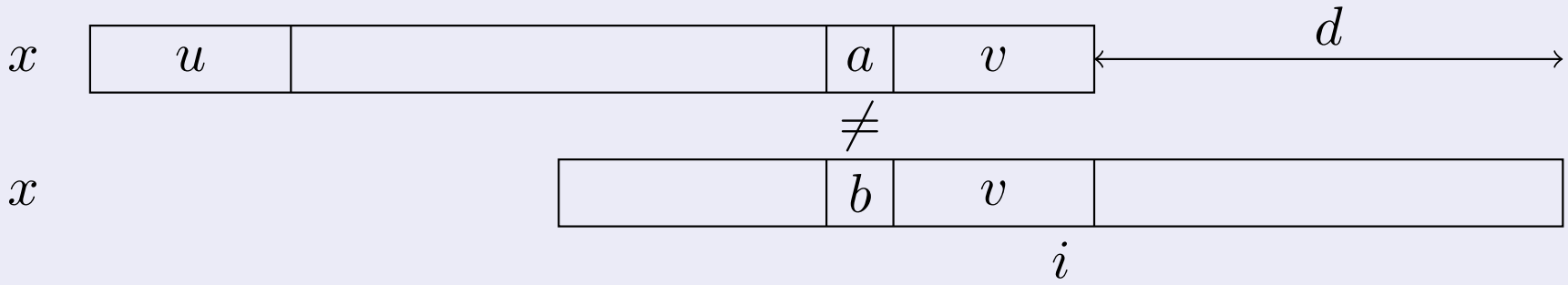
For $0 \leq i \leq m-1$

Lemma

rsrm$[m-1-suff[i]] \leq m-1-i$ *if* $m-1-i < suff[i]$ *and* $pref[m-1-i] \geq suff[i]-m+1+i$

where $suff[i]$ is the length of the longest common suffix between $x[0\,..\,i]$ and x and $pref[i]$ is the length of the longest common prefix between $x[..\,m-1]$ and x (classical computation in linear time and space).

Right shift after a right mismatch



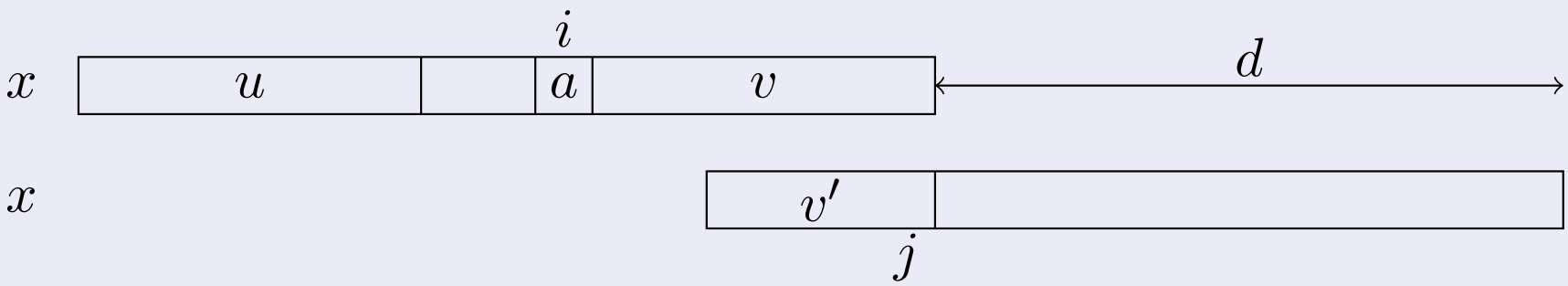
only a suffix v of x reoccurs preceded by a symbol b different from a;

For $0 \leq i \leq m-1$

Lemma

$rsrm[m-1-suff[i]] \leq m-1-i$ *if* $m-1-i \geq suff[i]$

Right shift after a right mismatch



only a prefix v' of x matches a suffix of v.

For $0 \leq i \leq m-1$

Lemma

rsrm$[i] \leq m - \mathit{suff}[j]$ *where*
$j = \max\{k \mid 0 \leq k < m-1-i$ *and* $\mathit{suff}[k] = k+1\}$

Right shift after a right mismatch

Lemma

rsrm can be computed in linear time

The 3 other arrays can be computed similarly $\Longrightarrow$ preprocessing in linear time

Reference

J. Daykin, R. Groult, Y. Guesnet, T. Lecroq, A. Lefebvre, M. Léonard, L. Mouchard, É. Prieur-Gaston and B. Watson
Three Strategies for the Dead-Zone String Matching Algorithm
In: (J. Holub and J. Žďárek editors, *Proceedings of the Prague Stringology Conference 2018 (PSC 2018)*, Prague, Czech Republic, 2018) 117–128.

Perspectives

It remains to do

- exact complexity analysis
- experimentations
- parallel implementation
- explore deterministic sampling (Vishkin, 1990)
- compute a w-matching machine (Didier, 2018)

algorithms

an Open Access Journal by MDPI

String Matching and Its Applications

Guest Editors:

Prof. Dr. Thierry Lecroq
LITIS EA 4108, Normastic FR3638, IRIB, University of Rouen Normandie, Normandie University, Rouen, France

Thierry.Lecroq@univ-rouen.fr

Prof. Dr. Simone Faro
Department of Mathematics and Computer Science, University of Catania, I-95125 Catania, Italy

faro@dmi.unict.it

Deadline for manuscript submissions:
28 February 2019

Message from the Guest Editors

Dear Colleagues,

With the rapid growth of available data in almost all fields, there is large demand for efficient pattern-matching algorithms. We invite you to submit your latest research in the area of string matching (single or multiple, on-line or off-line, exact or approximate, in uncompressed or compressed form) describing new data structures and/or new algorithms. High-quality papers are solicited to address both theoretical and practical issues of string matching including, but not restricted to, natural language processing, text mining, bioinformatics (DNA, RNA, protein sequences), chemoinformatics, intrusion detection, security, plagiarism detection, digital forensics, video retrieval, and music analysis.

Prof. Dr. Thierry Lecroq
Prof. Dr. Simone Faro
Guest Editors

mdpi.com/si/17457

Special Issue

Thank you for your attention!