# Optimal Packed String Matching

O. Ben-Kiki, P. Bille, D. Breslauer, L. Gasieniec, R. Grossi, O. Weimann

# String Matching Problem

- Knuth-Morris-Pratt
- Boyer-Moore
- Karp-Rabin

O(m+n) time solution
[over 85 algs in Faro-Lecroq's survey]

# String Matching Problem

INPUT:

pattern X of m symbols in $\Sigma$ [pattern preprocessing]

text T of n symbols in $\Sigma$ [text processing]

OUTPUT:

positions i s.t. $X = T[i...i+m-1]$

Can we say anything new?

# Model of computation vs commodity processors

Theory: word-RAM $w$ bits per word

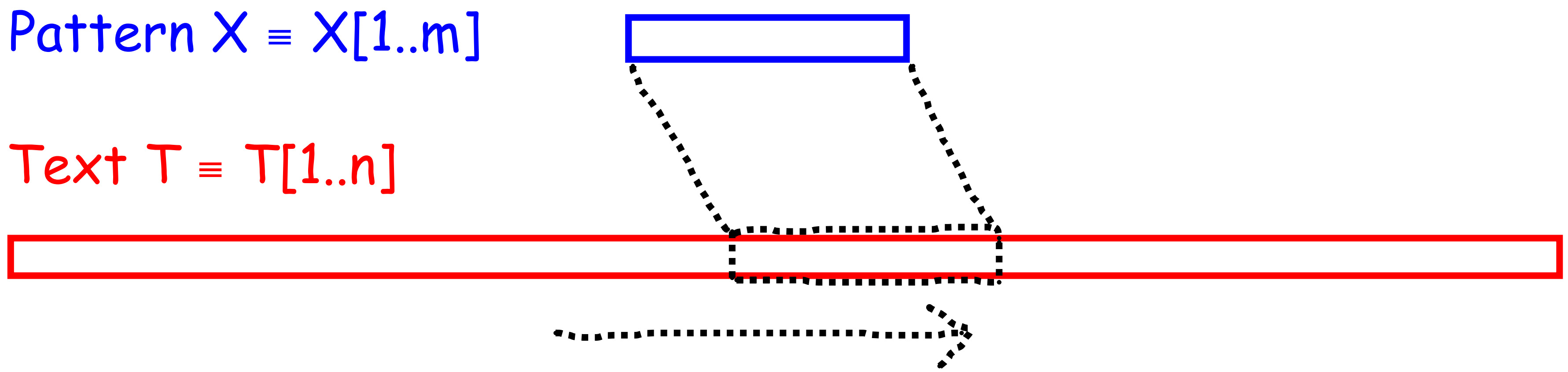
Your laptop:

- $\alpha$ characters per word [ $\alpha = w / \log_2 \Sigma$ ]
- richer instruction set

Example: reading T is $\Omega(n/\alpha)$ not $\Omega(n)$ ...

Filling the gap...

# Packed string matching



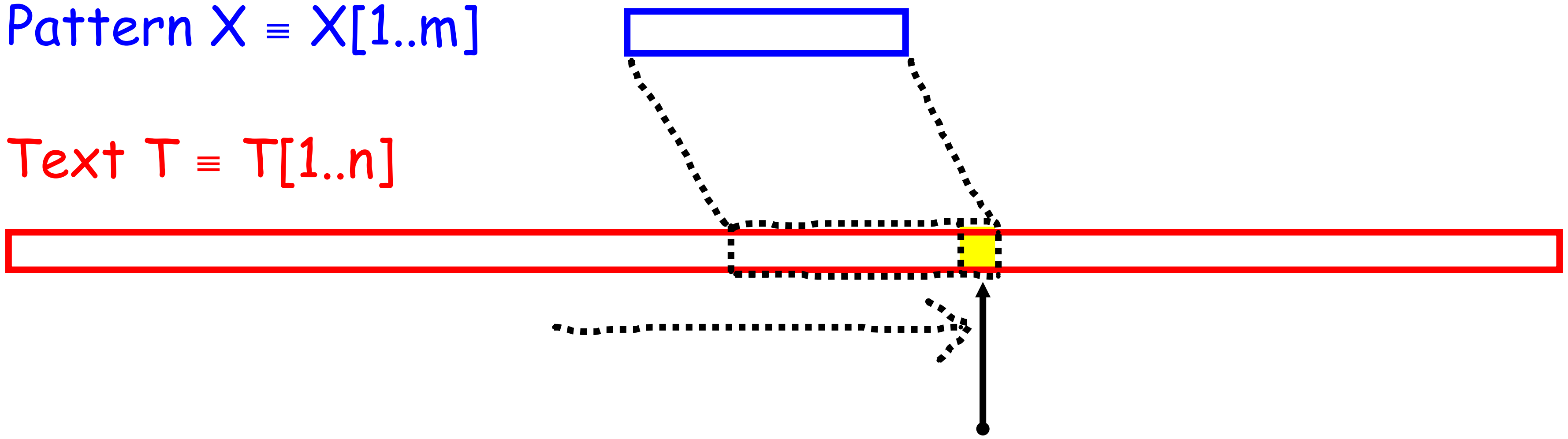

$\alpha$ symbols packed in a word:

bulk comparison in O(1) time

# Real-time string matching

Pattern X ≡ X[1..m]

Text T ≡ T[1..n]

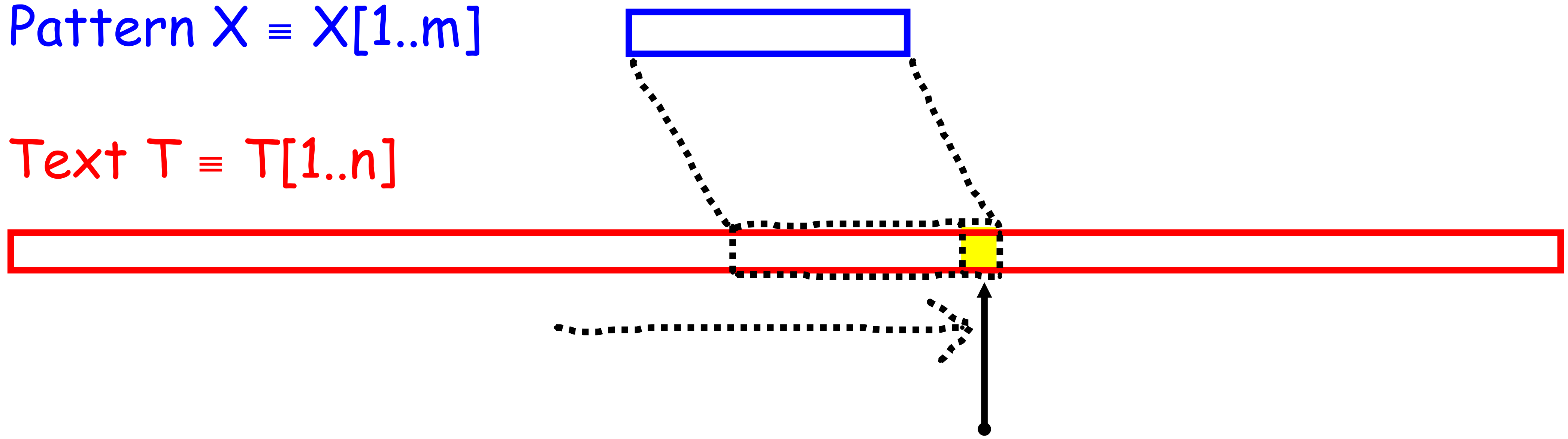


O(1) worst-case time to answer after reading the text symbol

# Real-time string matching

Pattern X ≡ X[1..m]

Text T ≡ T[1..n]



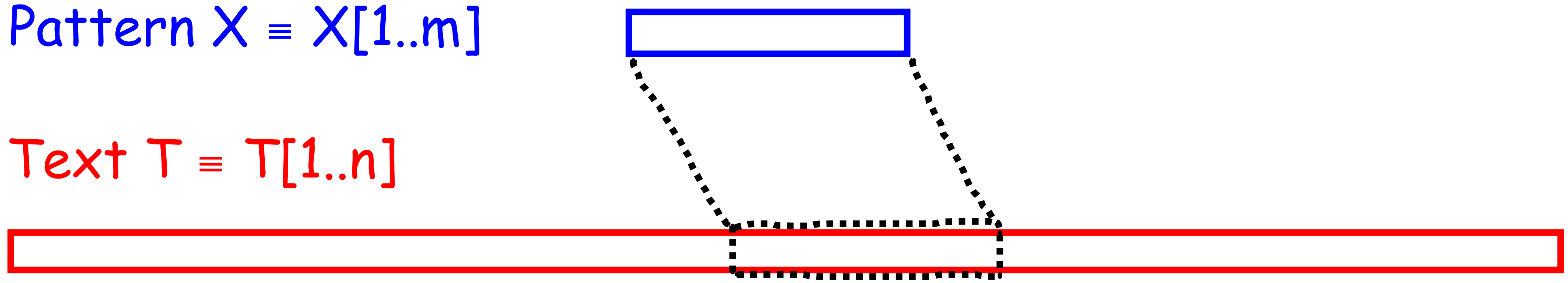
O(1) worst-case time to answer after reading the text symbol

Different from real-time streaming s.m., where X and T cannot be entirely stored!

# Constant-space string matching

Pattern X ≡ X[1..m]

Text T ≡ T[1..n]



(a.k.a. in-place)

# More related work

- Galil '81: real-time string matching
- Galil, Seiferas '83: constant space
- Karp, Rabin '87: randomized constant space real-time
- Crochemore, Perrin '91: constant space
- Gasieniec, Plandowski, Rytter '95: constant space
- Gasienec, Kolpakov '04: real-time + sublinear space (extends GPR'95)
- • • • more papers [Crochemore, Rytter '91,'95] [Crochemore '92] [...]
- Porat, Porat '09: randomized streaming, O(log m) space, no real-time
- Breslauer, Galil '10: randomized real-time streaming, O(log m) space

# History of packed string matching

- mentioned in KMP & BM
- several practical approaches (not discussed here)

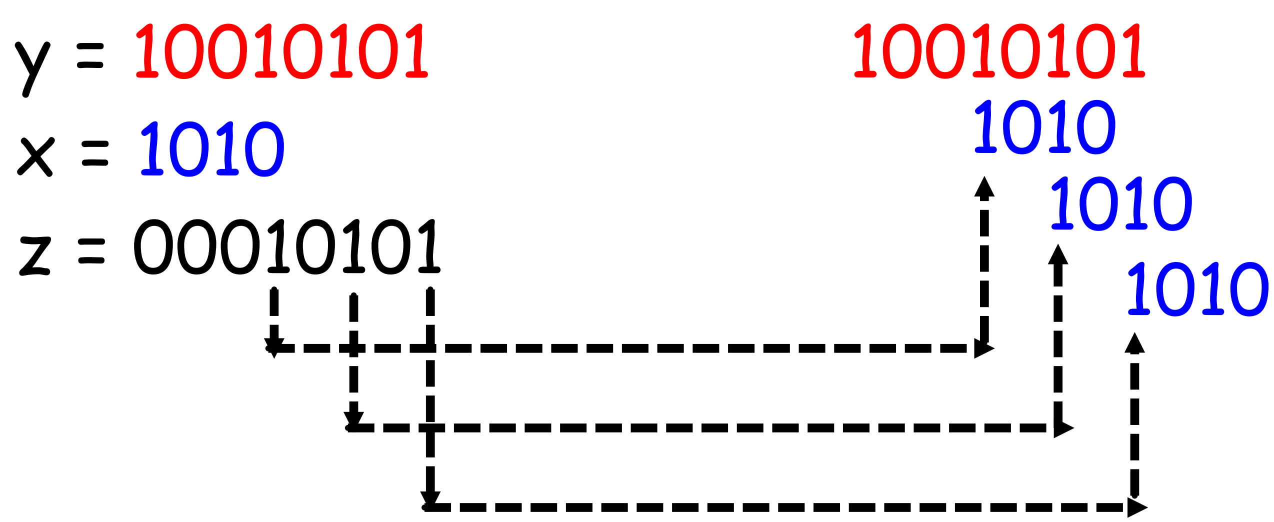
| Time | Space | Reference |
|---|---|---|
| $O(\frac{n}{\log_{\vert\Sigma\vert} n} + n^{\varepsilon}m + occ)$ | $O(n^{\varepsilon}m)$ | Fredriksson [24, 25] |
| $O(\frac{n}{\log_{\vert\Sigma\vert} n} + m + occ)$ | $O(n^{\varepsilon} + m)$ | Bille [10] |
| $O(\frac{n}{\alpha} + \frac{n}{m} + m + occ)$ | $O(m)$ | Belazzougui [8] |
| $O(\frac{n}{\alpha} + \frac{m}{\alpha} + occ)$ | $O(1)$ | using WSSM and WSLM |

real-time

Use two special $AC^0$ instructions

WSM: word-size string matching [text processing]

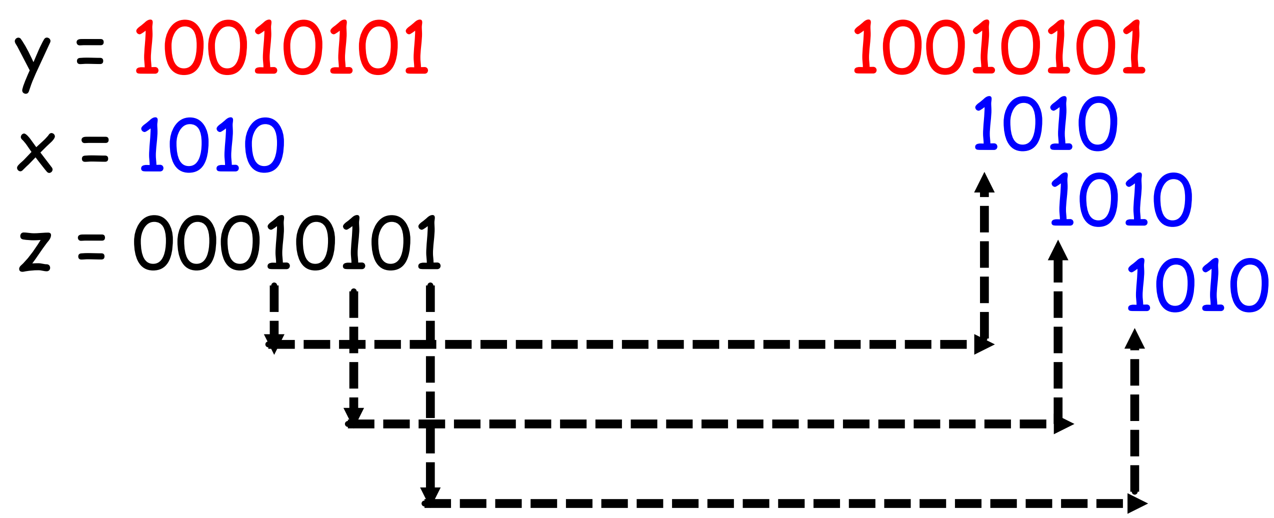
find x in y $|x| \le w$, $|y| \le 2w$




WSL: word-size lex-max suffix [pattern preprocessing]

O(1)-time WSM black box:


find x in y    $|x| \le w$,    $|y| \le 2w$



# Emulation in the word-RAM

| Time | Space | Emulation |
|---|---|---|
| $O(\frac{n \log \alpha}{\alpha} + occ)$ | $O(1)$ | bit-parallel WSSM no pre-processing |
| $O(\frac{n}{\alpha} + \alpha + occ)$ | $O(\alpha)$ | bit-parallel WSSM pre-processing |
| $O(\frac{m}{\log_{\|\Sigma\|} n})$ | $O(n^{\epsilon})$ | four Russian WSLM table lookup |

slowdown factor $\log \alpha$

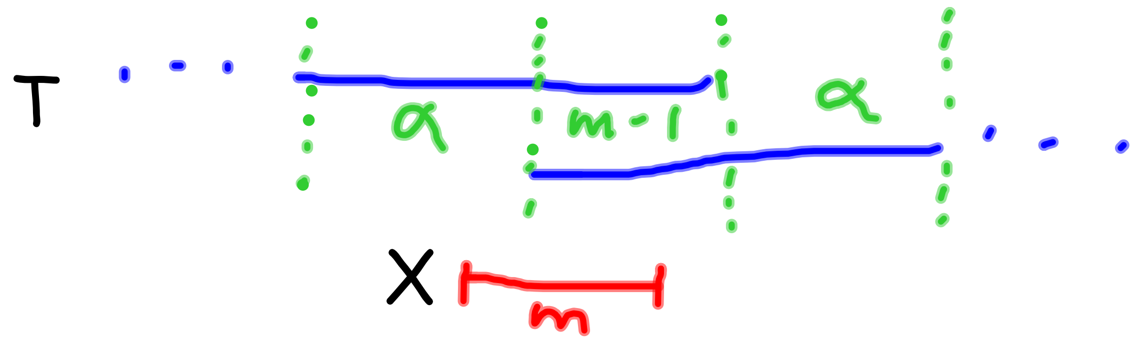
# Focus on: TEXT SCANNING + WSM

RECALL: $\alpha$ characters per word

SHORT pattern X iff its length $m \leq \alpha$ (LONG o.w.)

IDEA:

- split text T into overlapping blocks of $\alpha + m - 1 < 2\alpha$ chars




- each occurrence of X fits one block
- run the WSM black box on each block
- TOTAL COST: $O(n/\alpha)$ time and $O(1)$ space (real-time)

LONG pattern X iff its length $m > \alpha$ (SHORT o.w.)

IDEA for CASE 1: $m > \alpha > \pi(x)$:

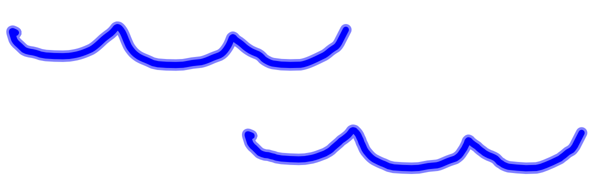
- write $X = p^r p'$, where $p$ = period(X) and $|p| = \pi(x)$
- split text T into words of $\alpha$ chars each
- GOAL: find maximal runs of consecutive ps

↳ easy to get X from them!

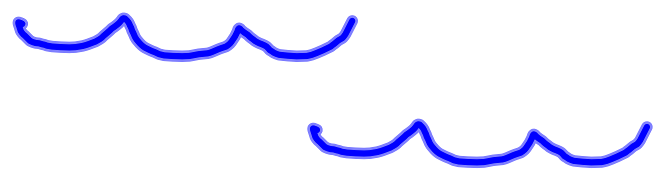
LONG pattern X iff its length $m > \alpha$ (SHORT o.w.)

IDEA for CASE 1: $m > \alpha > \pi(x)$:

- GOAL: find maximal runs of consecutive ps
- let max k s.t. string $p^k$ fits into a word ($k \leq r$)
- run the WSM black box for $p^k$ on each word
- combine the occs of $p^k$ to find maximal runs



extend run



start a new run

TOTAL COST: $O(n/\alpha)$ time and $O(1)$ space (real-time)

LONG pattern X iff its length $m > \alpha$ (SHORT o.w.)

IDEA for CASE 2: $m \geq \pi(x) \geq \alpha$:

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm
- Make CP also real-time by running **two** instances simultaneously
- Use WSM black box

LONG pattern X iff its length $m > \alpha$ (SHORT o.w.)

IDEA for CASE 2: $m \geq \pi(X) \geq \alpha$:

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm

DETOUR...

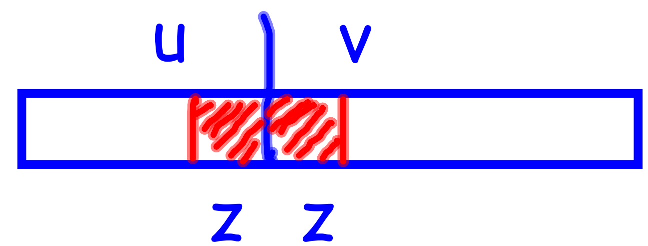
- Make CP also real-time by running **two** instances simultaneously

- Use WSM black box

Simple version of the Crochemore-Perrin (CP) algorithm

Consider a non-empty prefix-suffix factorization $X = u\, v$

The local period is the shortest $z$ such that




$z$ is suffix of $u$ or vice versa

and

$z$ is a prefix of $v$ or vice versa

$\mu(u,v) \equiv$ length $|z|$ of the local period

Example:

X = u v

a baaaba    ab aaaba    aba aaba

ba ba    aaab aaab    a a

z = ba local period

Critical factorization if $\mu$(u,v) = $\pi$(X) [len. of the period of X]

Example:

X = u v

a baaaba

ba ba

ab aaaba

aaab aaab

z = aaab local period

aba aaba

a a

Critical factorization if $\mu(u,v) = \pi(X)$ [len. of the period of X]

Example:

X = u  v = (abaa) aba = p p'

| | | |
|---|---|---|
| a baaaba | ab aaaba | aba aaba |
| ba ba | aaab aaab | a a |
| | z | |

critical!

local period z is as long as period p = abaa

Critical factorization if $\mu(u,v) = \pi(X)$ [len. of the period of X]

Example:

a baaaba    ab aaaba    aba aaba

ba ba    aaab aaab    a a

Critical Factorization Theorem (Cesari and Vincent):

Among $\pi(X) - 1$ consecutive factorizations:

at least one is a critical factorization

Example:

| | | |
|---|---|---|
| a baaaba | ab aaaba | aba aaba |
| ba ba | aaab aaab | a a |

Critical Factorization Theorem (Cesari and Vincent):

Among $\pi(X) - 1$ consecutive factorizations:

at least one is a critical factorization

There always exists a critical factorization $X = u\,v$ such that $|u| < \pi(X)$

## Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\ v$

## Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\,v$

Forward scan: match $v$ left-to-right with the current aligned portion of the text

# Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern X = u v

**Forward scan:** match v left-to-right with the current aligned portion of the text

**Back fill:** match u left-to-right with the current aligned portion of the text [originally right-to-left]

## Crochemore-Perrin (CP) Algorithm:

Take such a critical factorization of the pattern $X = u\ v$

**Forward scan:** match $v$ left-to-right with the current aligned portion of the text

**Back fill:** match $u$ left-to-right with the current aligned portion of the text [originally right-to-left]

How to handle mismatches?

## Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab  aaaba  critical factorization

abaaaba

abaabaaabaa

## Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab  aaaba  critical factorization

abaaaba

abaabaaabaa

## Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

abaaaba

abaabaaabaa

## Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

z

abaaaba

≠

abaabaaabaa

↑ mismatch

## Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

X = ab aaaba critical factorization

z

abaaaba

abaabaaabaa

→

|z|+1

↔ abaaaba

abaabaaabaa

shift by |z|+1 positions

(and charge the O(|z|+1) cost to the symbols in z in real time)

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

Output an occurrence when the forward scan terminates (and interrupt the back fill if needed)

Basic Real-Time Algorithm

Interleave $O(1)$ comparisons from the forward scan with $O(1)$ comparisons from the back fill

Output an occurrence when the forward scan terminates (and interrupt the back fill if needed)

Let $z$ be the matched prefix of $v$, where $X = u\,v$ is c.f.:

- if $z \neq v \Rightarrow$ shift by $|z|+1$ positions and reset $z$ = empty
- if $z = v \Rightarrow$ shift by $\pi(X)$ positions and update $z$

Basic Real-Time Algorithm

Interleave O(1) comparisons from the forward scan with O(1) comparisons from the back fill

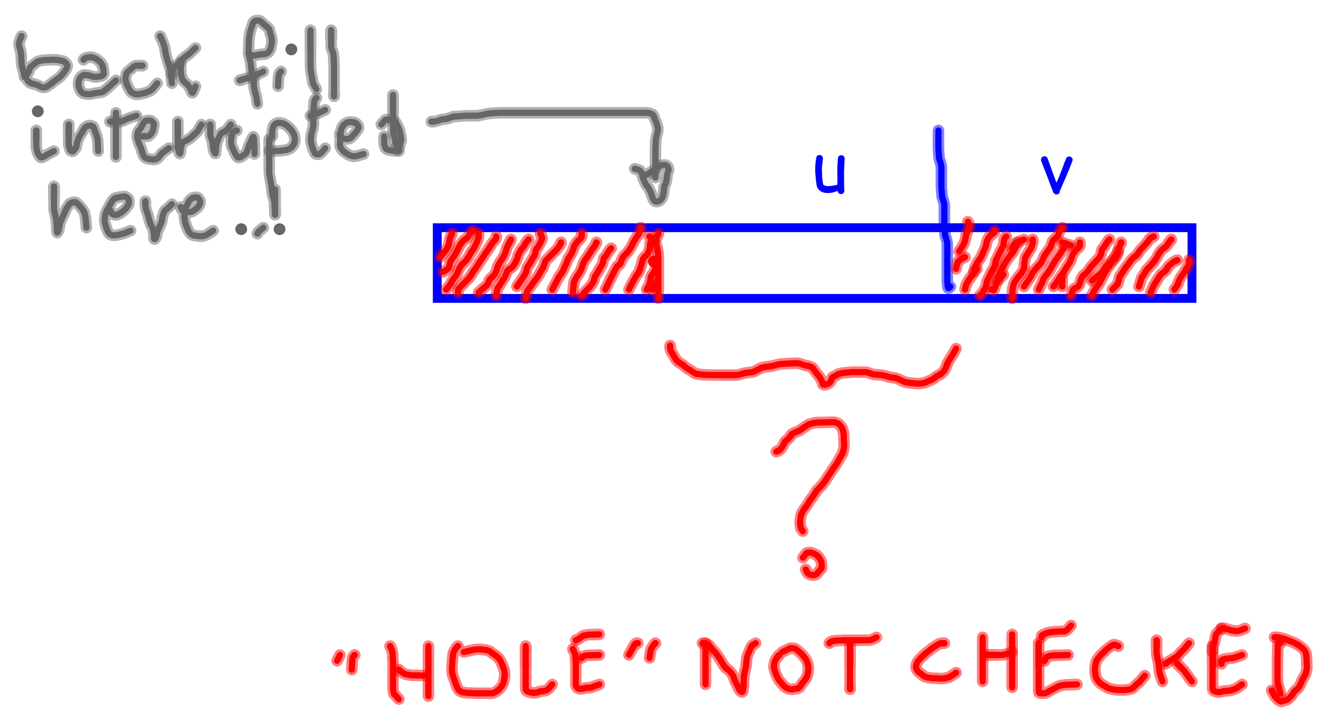
Output an occurrence when the forward scan terminates (and interrupt the back fill if needed)

Let $z$ be the matched prefix of $v$, where $X = u\, v$ is c.f.:

- if $z \neq v \Rightarrow$ shift by $|z|+1$ positions and reset $z$ = empty
- if $z = v \Rightarrow$ shift by $\pi(X)$ positions and update $z$

Total cost is O(1) worst-case per symbol: the algorithm is real-time

Q: What if |u| > |v|?



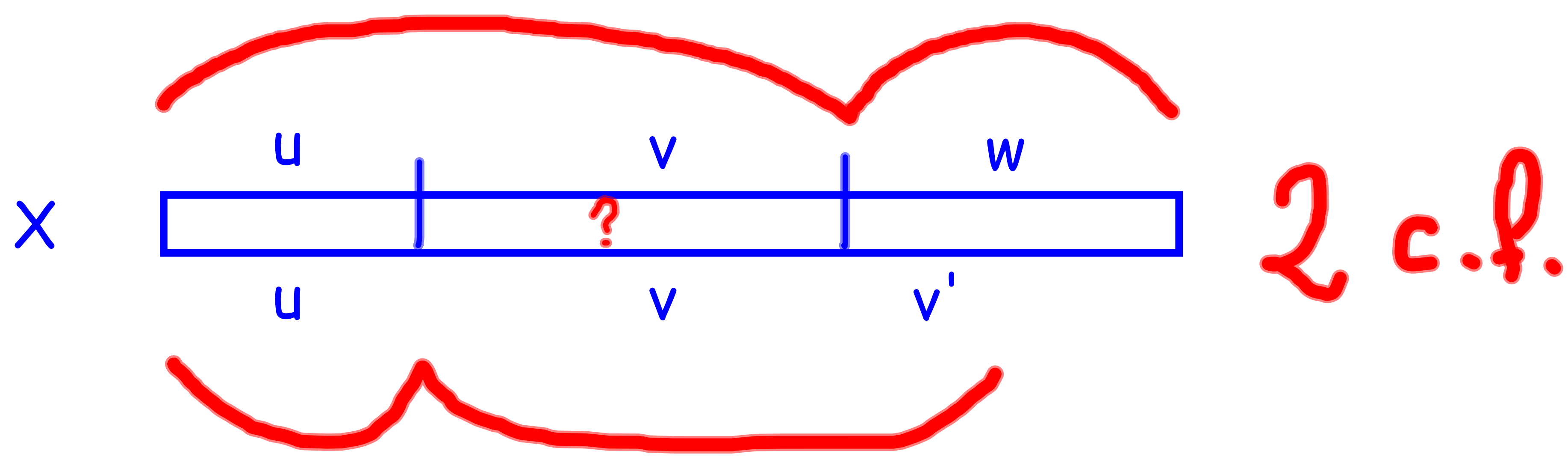
## Real-Time Variation of CP

Consider a 3-way non-empty factorizaton X = u v w such that

X = (uv) w is a critical factorization with |uv| ≤ |w|

OR

X = (uv) w is a critical factorization, and
X' = u (vv') is a critical factorization for a prefix X' of X
with |u| ≤ |vv'|



## Real-Time Variation of the CP Algorithm

Interleave O(1) steps of two instances of the Basic Real-Time Algorithms, one looking for X and the other for X', aligned with |X|-|X'| positions apart.

Total cost is O(1) worst-case per symbol: the algorithm is real-time and reports correctly all the occurrences

Simple pseudocode

WLOG: X = uv match v ...

LONG pattern X iff its length $m > \alpha$ (SHORT o.w.)

IDEA for CASE 2: $m \geq \pi(X) \geq \alpha$:

- Take a simple version of the constant-space Crochemore-Perrin (CP) algorithm
- Make CP also real-time by running **two** instances simultaneously
- Use WSM black box

## Recap the goal for CASE 2: $m \geq \pi(x) \geq \alpha$:

HP: $\alpha$ characters packed per word

GIVEN: pattern $X = u\,v$   (critical factorization)

- HAVE-TO: perform forward scan of v
  (the rest of the cost is covered using CP)

COST: $O(n/\alpha)$ time and $O(1)$ space  (real-time)

IDEA for forward scan in X = u v:

- let v' be the α–long prefix of v (if v is shorter, easy)
- for each text word, use WSM black box on v'
- take the leftmost occurrence of v' (derives from c.f.)
- extend v' to v by bulk comparisons (and check u):

mismatch ⇒ shift by at least α positions

match ⇒ shift by $|\pi(x)| \geq \alpha$ positions

TOTAL COST: $O(n/\alpha)$ time and $O(1)$ space (real-time)

# Simulating the WSM black box on the word-RAM

- reduce the problem to binary convolution ($AC^0$)
- simulate the convolution using int. multiplication (not $AC^0$) and padding each bit by log $\alpha$ bits
- use deterministic sampling to reduce each padding to log log $\alpha$

COST:

- $O(\alpha)$ preprocessing time
- $O(1)$ time on w/log log $\alpha$ bits

# Example

Padding the pattern 101 and the text 01101010 (padding bits are in gray)

$$p = 010001, t = 0001010001000100$$

$$\overline{p} = 000100, \overline{t} = 0100000100010001$$

Doing standard integer multiplication on these vectors we get that:

$$p \times \overline{t} = 1000101001000100001$$

$$\overline{p} \times t = 0000101000100010000$$

Adding these we get the mismatch vector:

$$(p \times \overline{t}) + (\overline{p} \times t) = 1\ 00\ 10\ 10\ 00\ 11\ 00\ 11\ 00\ 01$$

Replacing each field (two bits) by the number it holds gives:

$$(p \times \overline{t}) + (\overline{p} \times t) = 1022030301$$

Taking the $n = 8$ least significant bits gives the mismatch vector 22030301.

## Preliminaries experiments

Intel Sandy Bridge, SSE (Streaming SIMD Extension), AVX (Advanced Vector Extension)

SMART (String MAtching Research Tool) by Faro and Lecroq [library of > 85 algorithms]

| 2 | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|
| **SSECP** 4.44 | **SSECP** 4.57 | UFNDMQ4 4.99 | BNDMQ4 4.23 | BNDMQ4 3.83 | LBNDM 3.91 | BNDMQ4 3.71 |
| SKIP 4.80 | RF 5.07 | **SSECP** 5.00 | SBNDMQ4 4.31 | BNDMQ6 3.86 | BNDMQ4 3.94 | HASH5 3.83 |
| SO 4.84 | BM 5.33 | FSBNDM 5.05 | UFNDMQ4 4.31 | SBNDMQ4 3.95 | SBNDMQ4 3.96 | HASH8 3.93 |
| FNDM 4.94 | BNDMQ2 5.46 | SBNDMQ2 5.08 | UFNDMQ6 4.47 | SBNDMQ6 3.97 | BNDMQ6 3.97 | HASH3 3.94 |
| FSBNDM 5.03 | BF 5.58 | BNDMQ2 5.13 | SBNDMQ6 4.57 | UFNDMQ4 4.00 | HASH5 3.98 | BNDMQ6 3.97 |
| | | | 23 **SSECP** 5.00 | 27 **SSECP** 5.29 | 39 **SSECP** 4.88 | 42 **SSECP** 4.73 |
| **SSECP** 4.28 | **SSECP** 4.49 | BNDMQ2 4.42 | SBNDMQ2 4.08 | UFNDMQ2 3.75 | SBNDMQ4 3.67 | BNDMQ4 3.70 |
| FFS 4.88 | SVM1 4.84 | SBNDMQ2 4.48 | UFNDMQ2 4.08 | BNDMQ4 3.79 | BNDMQ4 3.72 | SBNDMQ4 3.71 |
| GRASPM 4.93 | SBNDMQ4 4.85 | SBNDM 4.59 | SBNDMQ4 4.10 | SBNDMQ4 3.80 | UFNDMQ4 3.80 | HASH5 3.75 |
| BR 5.14 | BOM2 4.95 | SBNDM2 4.59 | SBNDM2 4.13 | UFNDMQ4 3.80 | BNDMQ2 3.89 | UFNDMQ4 3.77 |
| BWW 5.14 | EBOM 5.25 | UFNDMQ2 4.69 | BNDMQ2 4.14 | BNDMQ2 3.89 | SBNDM2 3.96 | HASH8 3.80 |
| | | 13 **SSECP** 5.00 | 22 **SSECP** 5.08 | 35 **SSECP** 4.77 | 39 **SSECP** 4.77 | 45 **SSECP** 4.76 |

- SSECP our implementation, performs well for a wide range of parameters
- algorithms that skips characters are faster than SSECP for long patterns

# Conclusions and further work

*Theoretical models have a restricted set of operations compared to commodity processors in modern computers: design algorithms that exploit the latter and are theoretical*

- Improve WSM blackbox simulation
- Have WSM-based algorithm that can skip words
- Extend our WSM-based approach to other SM algorithms

Questions?