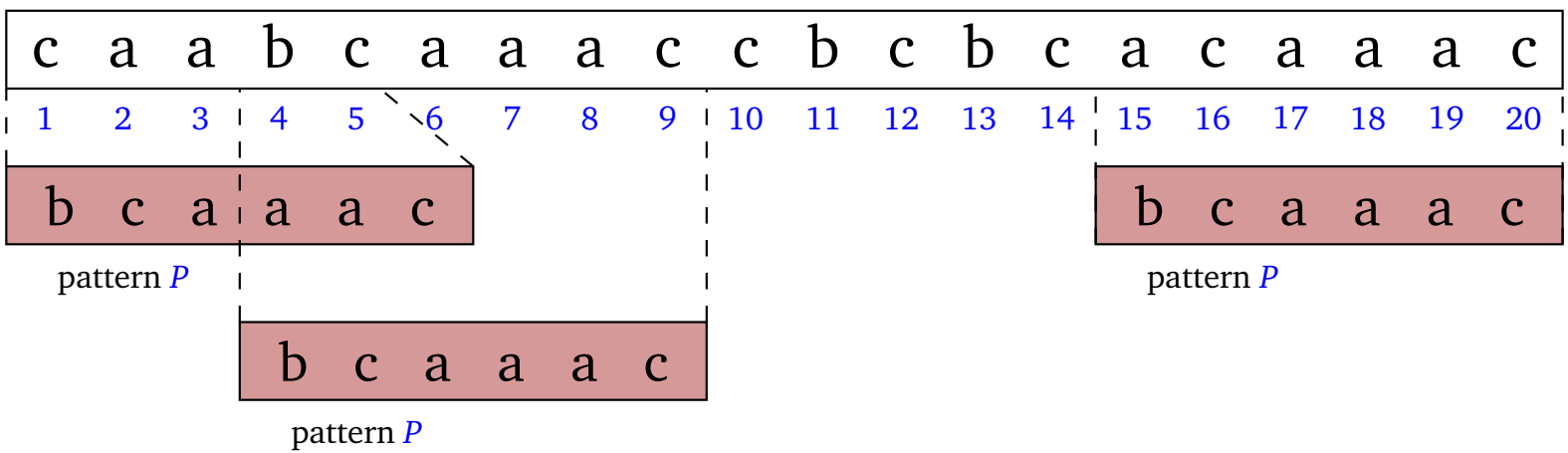
How Probability Helped Achieve Ultra-Efficient Algorithms for Pattern Matching

Tatiana Starikovskaya

The pattern matching problem

Input: A short string P (pattern) of length m, a long string T (text) of length n

Output: All positions i such that for some $1 \leq j \leq i$, a substring $T[j,i]$ is "close" to P



What is “close”?

Hamming distance
(# of mismatching letters)

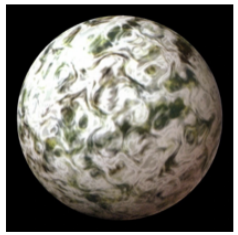
SERRES
| |
SELLES

HD = 2

Edit distance
(# of insertions, deletions, substitutions required to transform string1 into string2)

BORAB~~ORA~~
\
BARAB

ED = 4



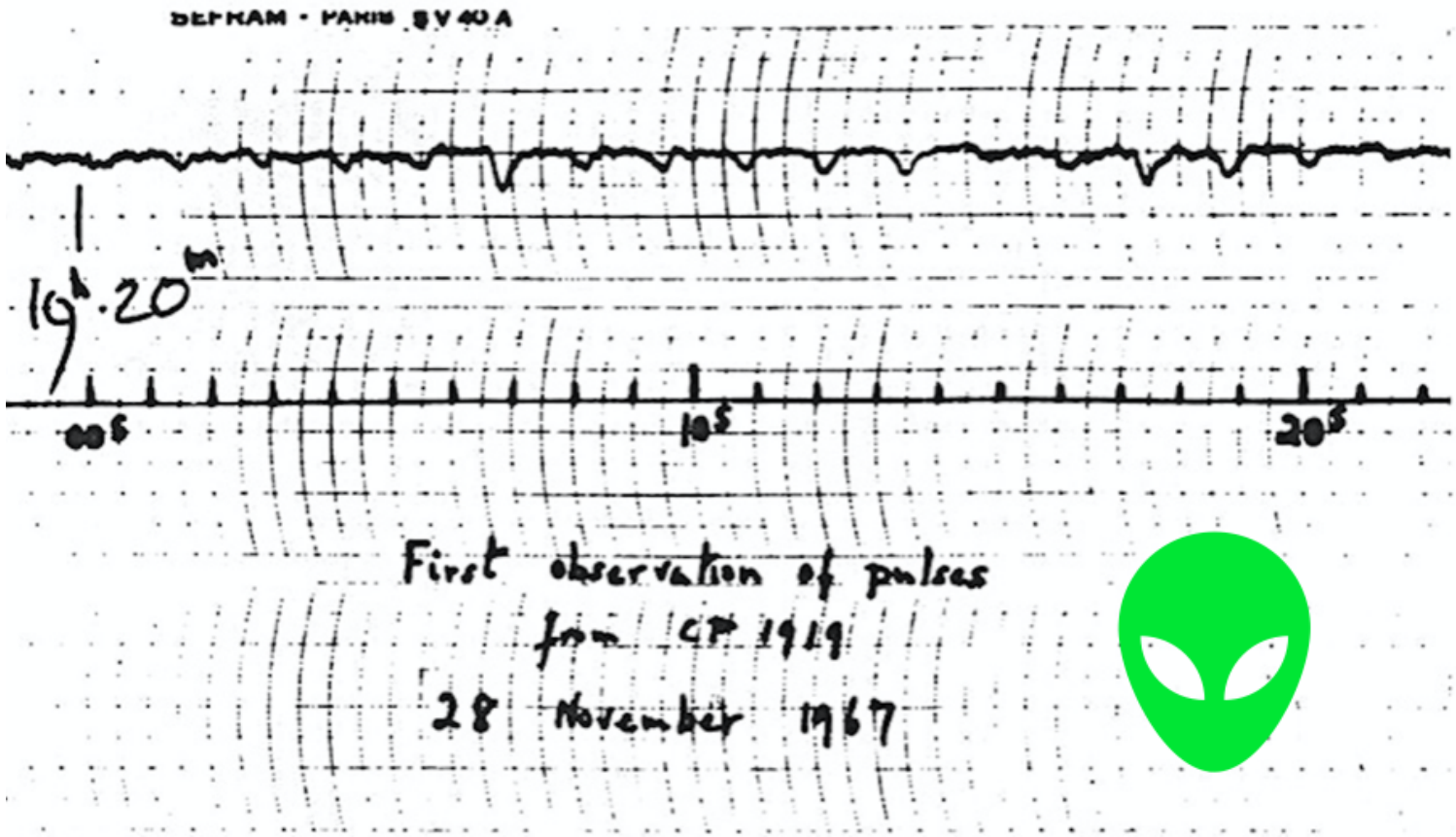
Three variants of pattern matching

Input: A short string P (pattern) of length m, a long string T (text) of length n

Output: All i such that for some $1 \leq j \leq i$, a substring $T[j, i]$

- Exact pattern matching: equals P
- k-mismatch pattern matching: is at Hamming distance $\leq k$ from P
- k-edit pattern matching: is at edit distance $\leq k$ from P

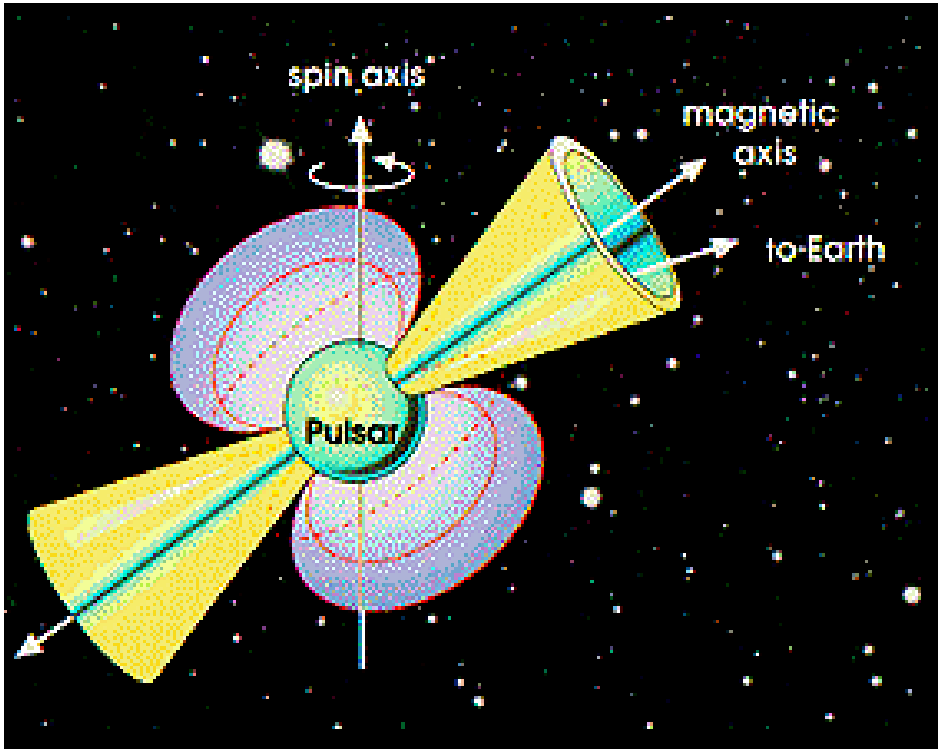
Little Green Men or pulsars



In 1968, Jocelyn Bell Burnell and her supervisor, Anthony Hewish, detected a strange periodic signal. They nicknamed their discovery LGM-1 for Little Green Men.

Little Green Men or pulsars

In fact, it was the first experimental observation of a **pulsar** — a rotating star that emits beams of electromagnetic radiation out of its magnetic poles.

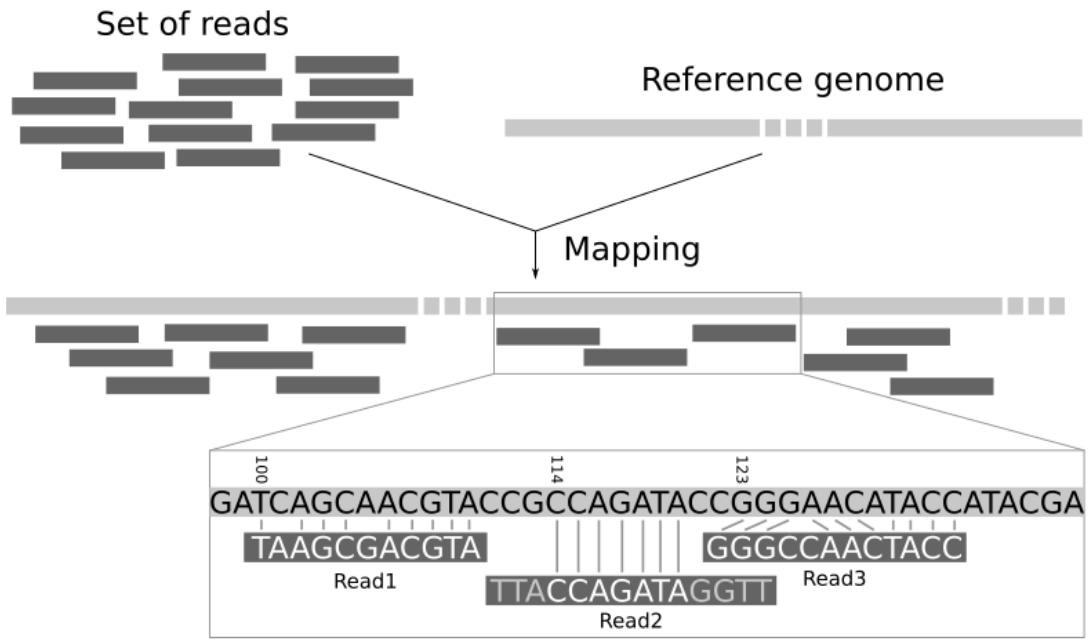


Pulsars play an important role in astronomy, and detecting them is of fundamental importance.

Question: Given a stream of integers, find its period.

Read mapping

Given a reference genome and a set of reads, find the best alignment between each read and the reference genome



©galaxyproject.org

Question: Streaming dictionary matching under edit distance.

Small-space algorithms for pattern matching

We consider a particularly restrictive **streaming setting**:

- Formalised in “The space complexity of approximating the frequency moments” by Alon, Matias, Szegedy (1999)
- The input is presented as a sequence of items and can be examined in only a few passes (typically just one)
- Limited memory (generally logarithmic) and ideally small processing time per item

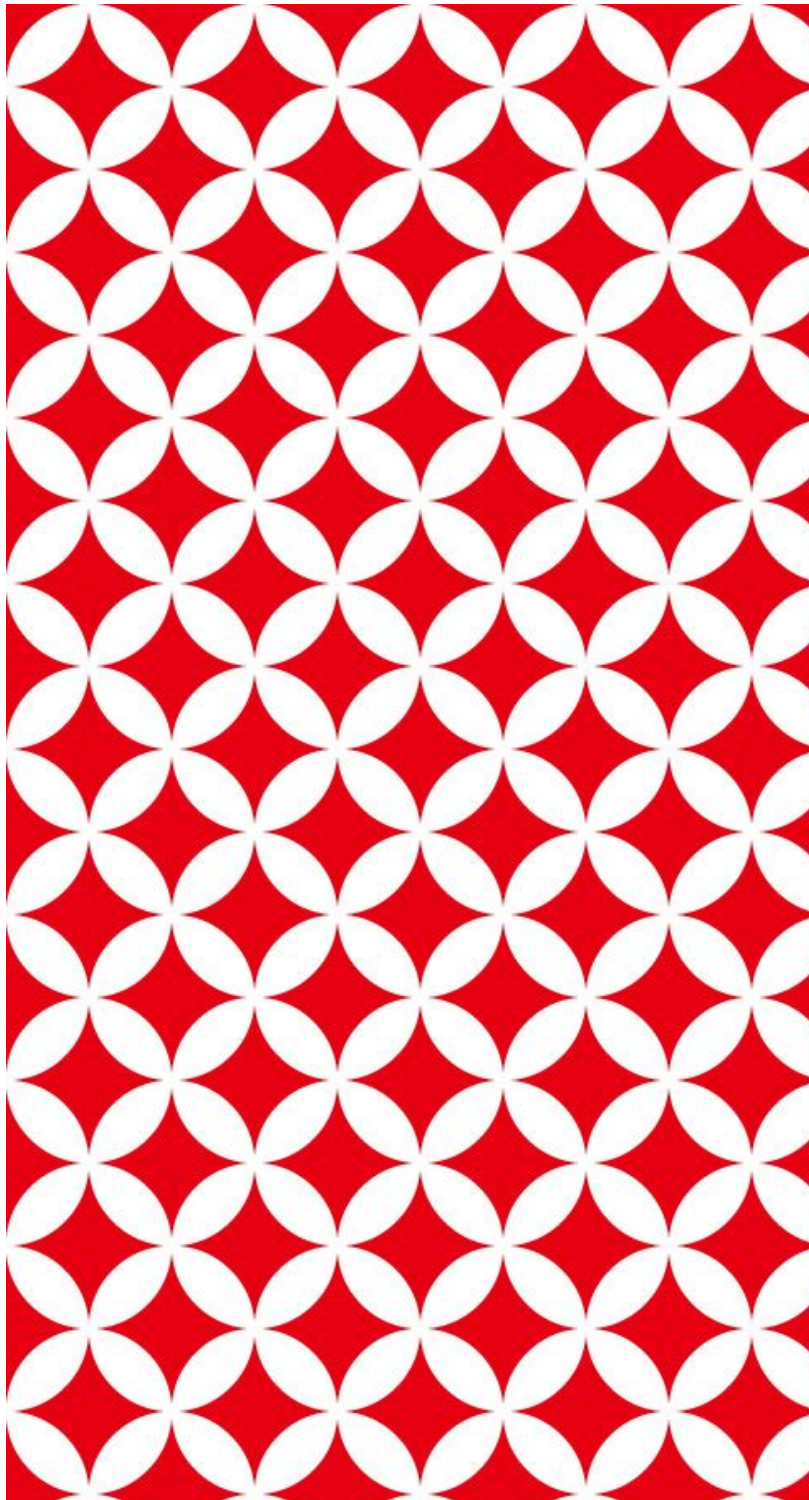
Small-space algorithms for pattern matching

- Unfortunately, **deterministic streaming algorithms** for pattern matching must use $\Omega(m)$ space
- Can we do better if we allow for **probability**?
- YES!

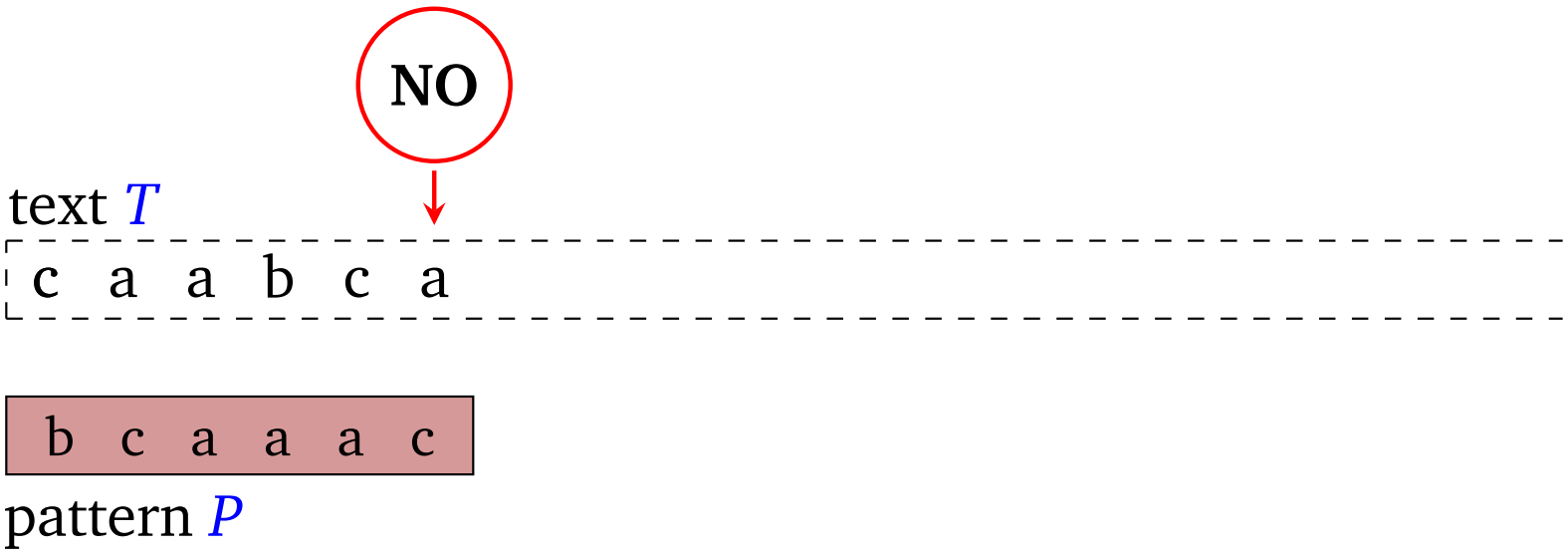
Technical part of today's talk

- Part I: exact pattern matching
- Part II: k-mismatch pattern matching
- Part III: k-edit pattern matching

Part I: Exact pattern matching

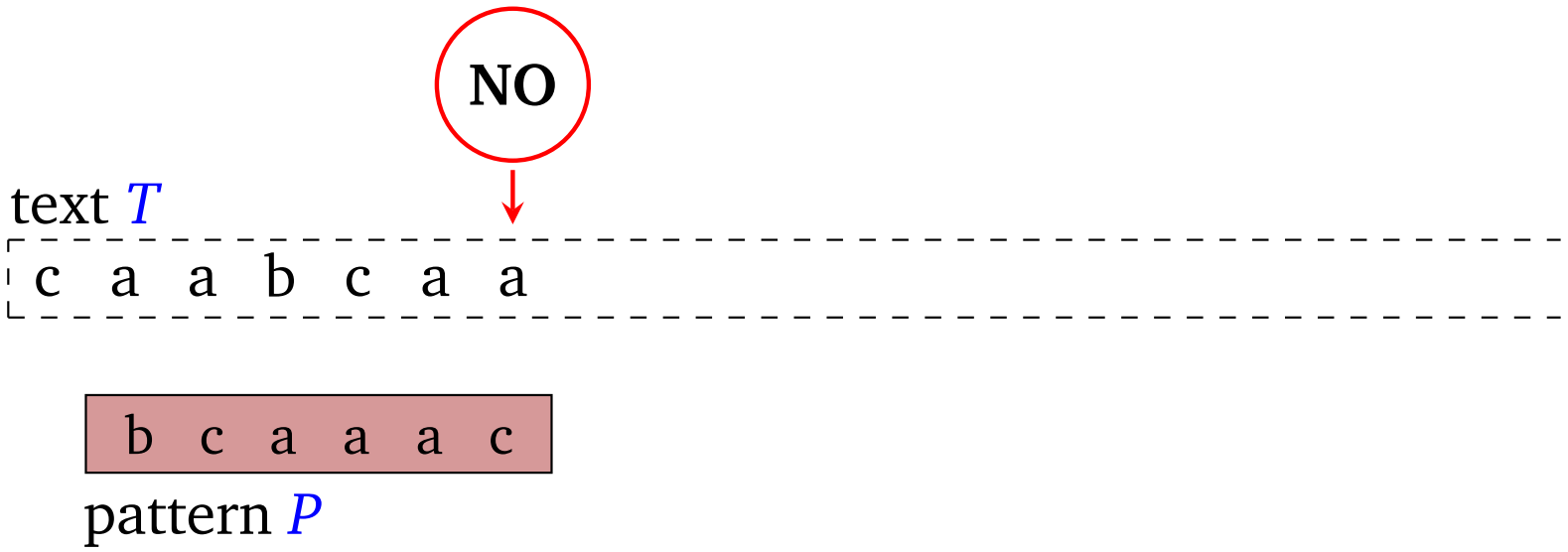


Exact pattern matching



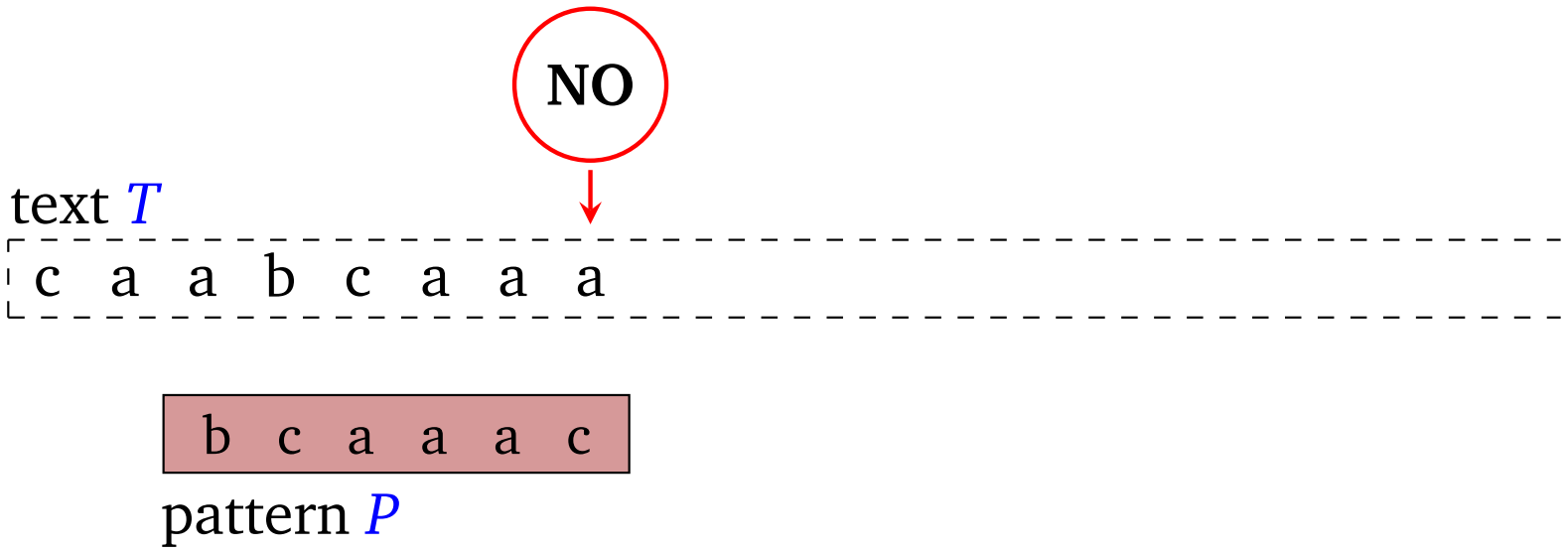
- **Query** = "Is the latest m-length substring equal to P?"
- **Space** = total space
- **Time** = time per position of T

Exact pattern matching



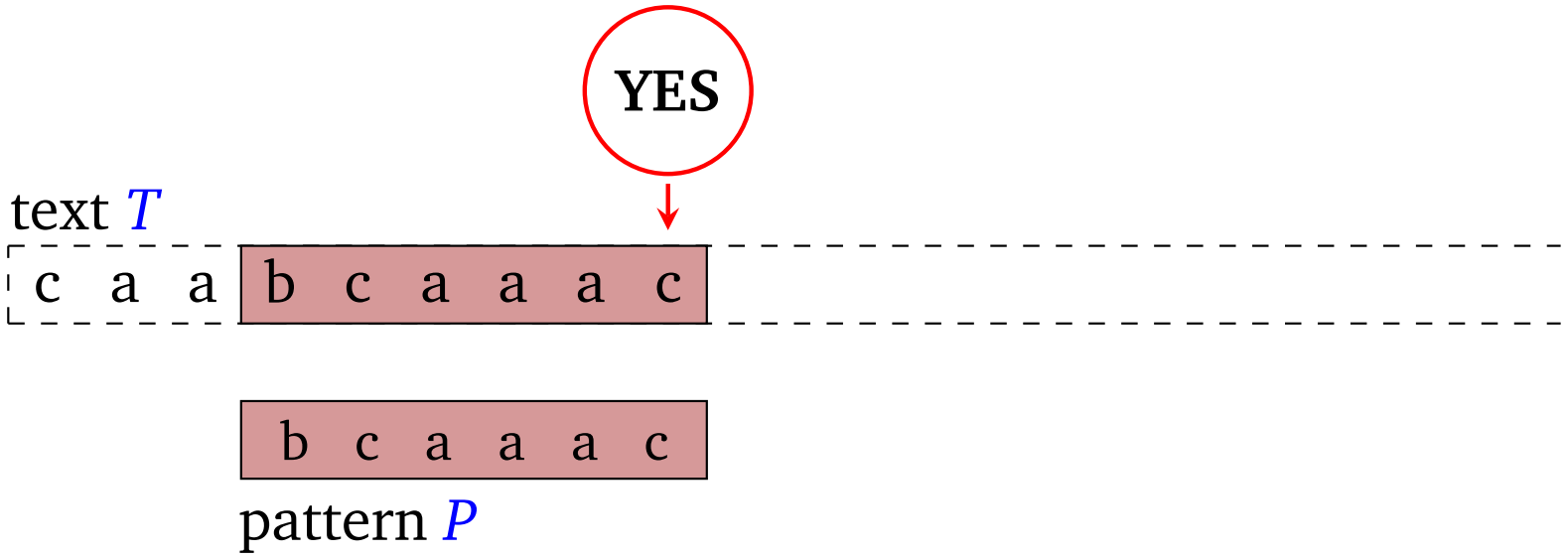
- **Query** = "Is the latest m-length substring equal to P?"
- **Space** = total space
- **Time** = time per position of T

Exact pattern matching



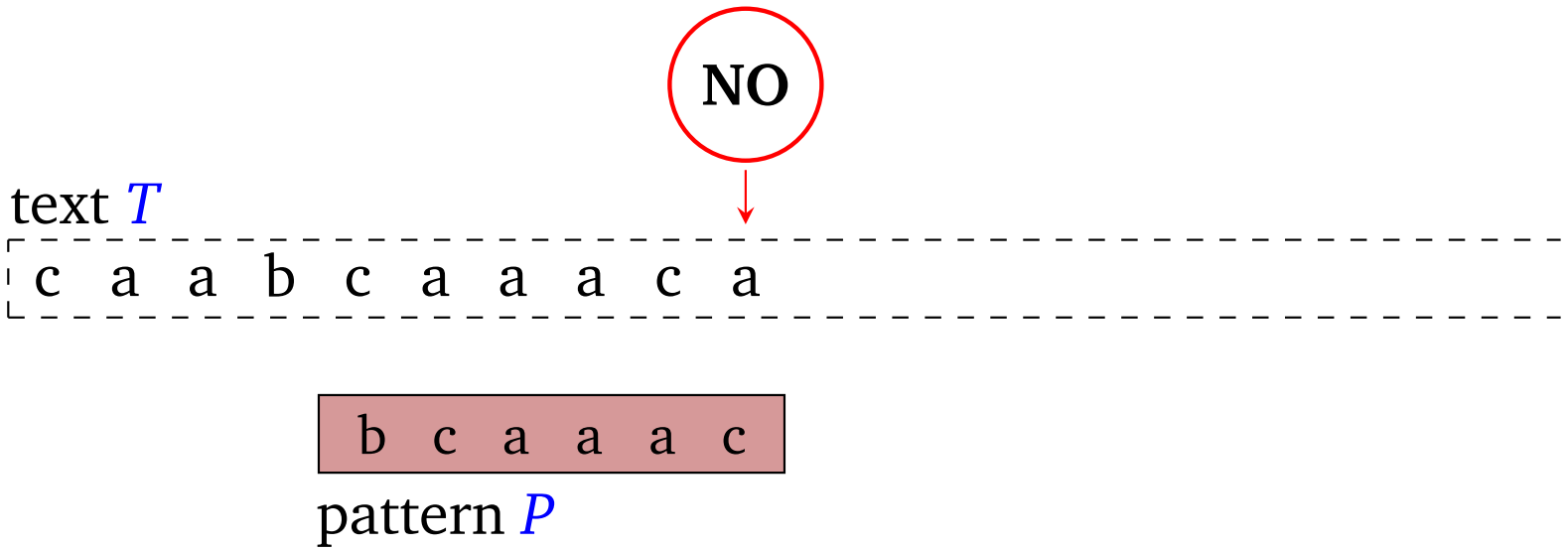
- **Query** = "Is the latest m-length substring equal to P?"
- **Space** = total space
- **Time** = time per position of T

Exact pattern matching



- **Query** = "Is the latest m-length substring equal to P?"
- **Space** = total space
- **Time** = time per position of T

Exact pattern matching



- **Query** = "Is the latest m-length substring equal to P?"
- **Space** = total space
- **Time** = time per position of T

Deterministic lower bound

- Assume binary alphabet $\{0, 1\}$ and $n = m$
- There are 2^m different patterns
- If we use $< m$ bits of memory, there are two patterns $P_1 \neq P_2$ for which the states of the memory of the algorithm are equal after reading them
- Run two instances of the algorithm: for P_1 and $T = P_1$ and for P_2 and $T = P_1$
- The answers must be **different**, but this is impossible!

$\Rightarrow$ We need probability.

Karp–Rabin algorithm

Karp–Rabin fingerprint

$$\varphi(s_1 s_2 \ldots s_m) = \sum_{i=1}^{m} s_i \cdot r^{m-i} \bmod p$$

where p is a prime and r is a random integer $\in [0, p-1]$

Karp–Rabin algorithm

Karp–Rabin fingerprint

$$\varphi(s_1 s_2 \dots s_m) = \sum_{i=1}^{m} s_i \cdot r^{m-i} \bmod p$$

where p is a prime and r is a random integer $\in [0, p-1]$

It's a good hash function:

S_1, S_2 are two strings of length m, the prime p is large

- If $S_1 = S_2$, then $\varphi(S_1) = \varphi(S_2)$
- If $S_1 \neq S_2$, then $\varphi(S_1) \neq \varphi(S_2)$ w.h.p.

Karp–Rabin algorithm

Karp–Rabin fingerprint

$$\varphi(s_1 s_2 \dots s_m) = \sum_{i=1}^{m} s_i \cdot r^{m-i} \bmod p$$

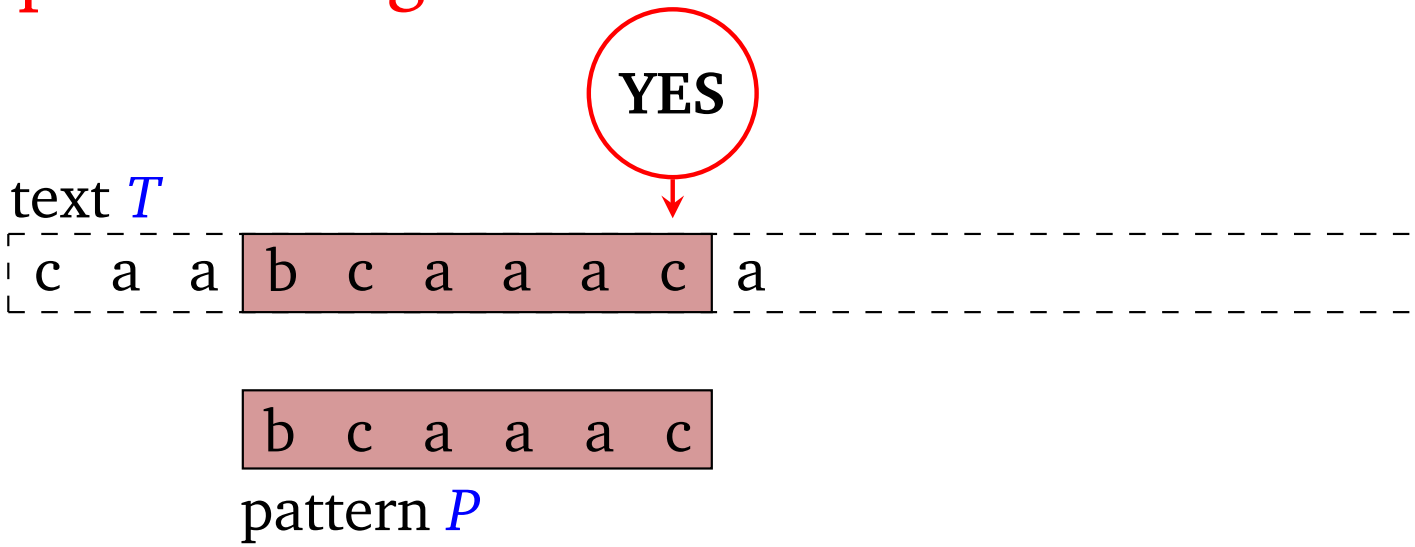
where p is a prime and r is a random integer $\in [0, p-1]$

It's a good hash function:
S_1, S_2 are two strings of length m, the prime p is large

- If $S_1 = S_2$, then $\varphi(S_1) = \varphi(S_2)$
- If $S_1 \neq S_2$, then $\varphi(S_1) \neq \varphi(S_2)$ w.h.p.

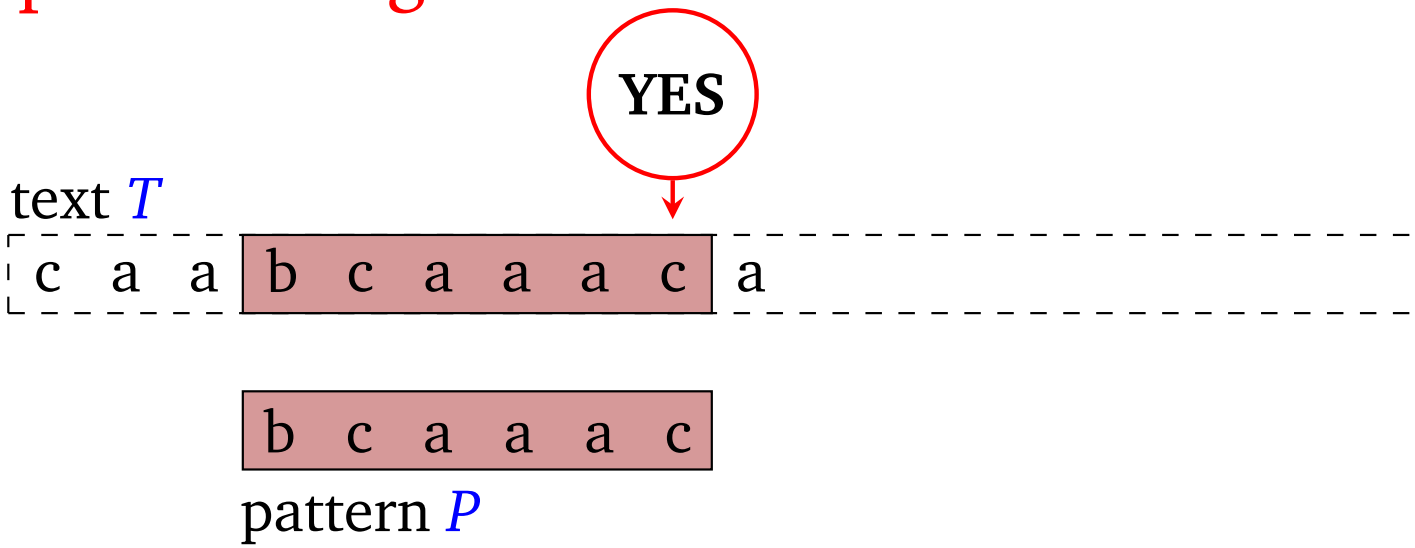
Also, concatenatable: Given two of $\varphi(X)$, $\varphi(Y)$, $\varphi(XY)$, the third can be computed in $\tilde{O}(1)$ time

Karp–Rabin algorithm



1. Compute the Karp–Rabin fingerprint of the pattern, $\varphi(P)$
2. For every $t = m, m+1, \ldots, n$:
   - Compute $\varphi(T[t-m+1, t])$
   - If $\varphi(T[t-m+1, t]) = \varphi(P)$, output t

Karp–Rabin algorithm



To compute the fingerprints of the substrings efficiently, we use concatenability:

1. $\varphi(T[1,m])$ can be computed in $O(m)$ time
2. To compute $\varphi(T[t-m+1,t]) = \sum_{j=1}^{m} T[t-m+j] \cdot r^{m-j} \bmod p$ we need $\tilde{O}(1)$ time:

$$\varphi(T[t-m+1,t]) = (\varphi(T[t-m,t-1]) - T[t-m] \cdot r^{m-1}) \cdot r + T[t] \bmod p$$

Remark: We need $T[t-m]$ for step 2. $\Rightarrow$ we need $\Omega(m)$ memory

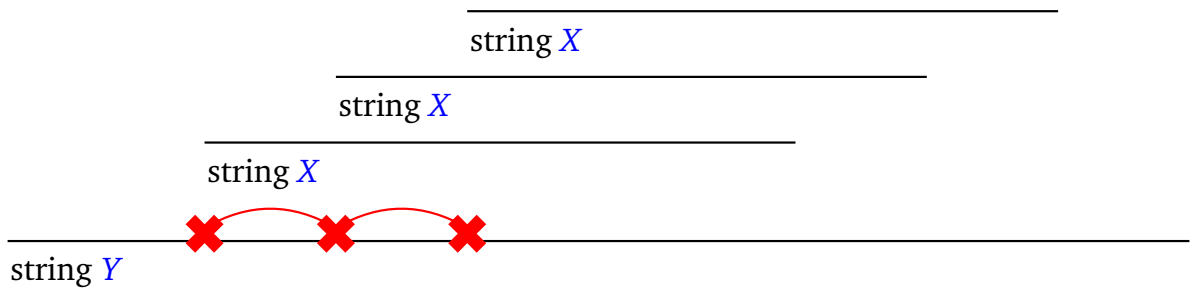
Exact pattern matching

Authors	Space (words)	Time
Single pattern		
Karp & Rabin, 1987	$\Theta(m)$	$O(1)$
Porat & Porat, FOCS'09	$O(\log m)$	$O(\log m)$
Breslauer & Galil, CPM'11	$O(\log m)$	$O(1)$
Dictionary of d patterns		
Clifford, Fontaine, Porat Sach, S., ESA'15	$O(d \log m)$	$O(\log\log(m+d))$
Golan & Porat, ESA'17	$O(d \log m)$ $O(\lvert\Sigma\rvert^{\varepsilon} d \log(m/\varepsilon))$	$O(\log\log \lvert\Sigma\rvert)$ $O(1/\varepsilon)$

OQ: All known lower bounds say we must use $\Omega(\log m)$ **bits** of space per pattern. What is the true space complexity of exact pattern matching?

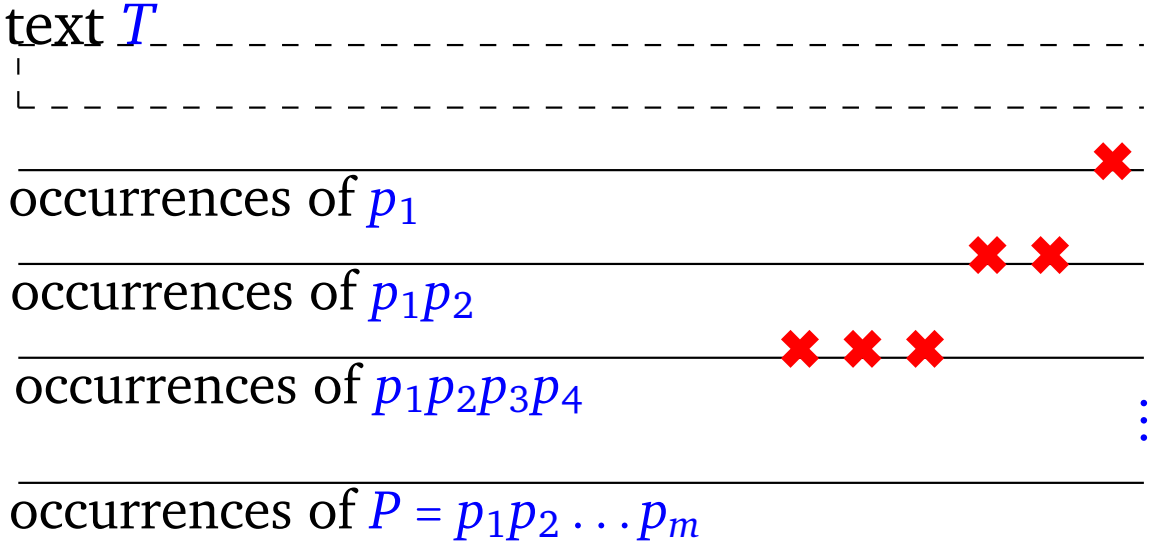
Porat & Porat, FOCS'09

If $|Y| = 2|X|$, and Y contains ≥ 3 occurrences of X, they form an arithmetic progression with difference equal to the smallest period of X (Corollary of Fine and Wilf's periodicity lemma)



Occurrences of X in Y can be stored in $\mathcal{O}(1)$ space

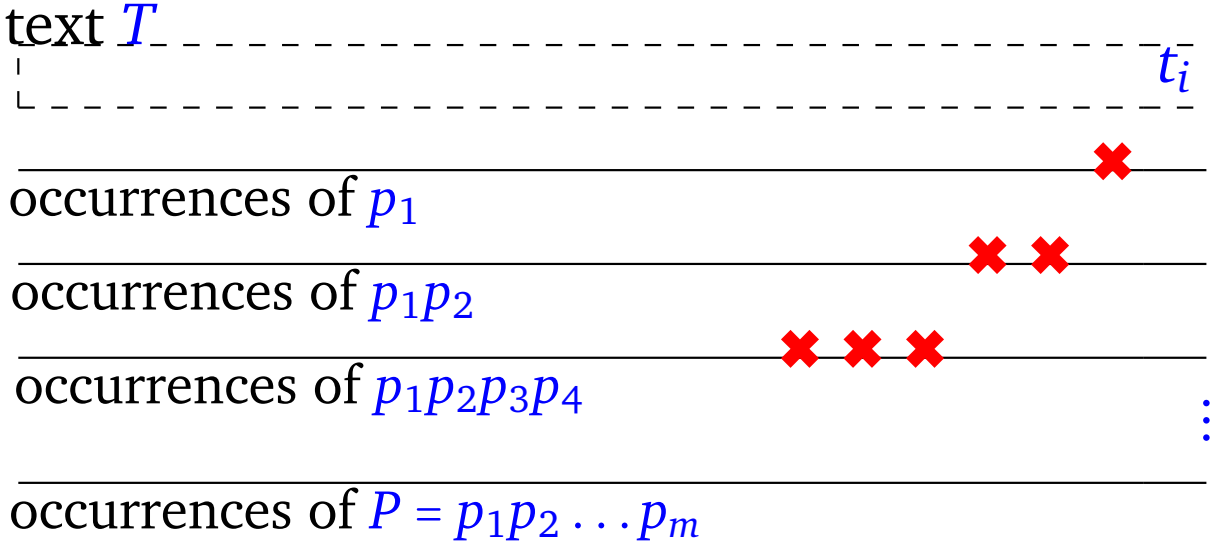
Porat & Porat, 2009



for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if lp is occurrence of $p_1 \ldots p_{2^{j+1}}$ **then** push lp to level $j + 1$

Correctness: if p is an occurrence of P, it is also an occurrence of all $\log m$ prefixes and will pass all the tests.

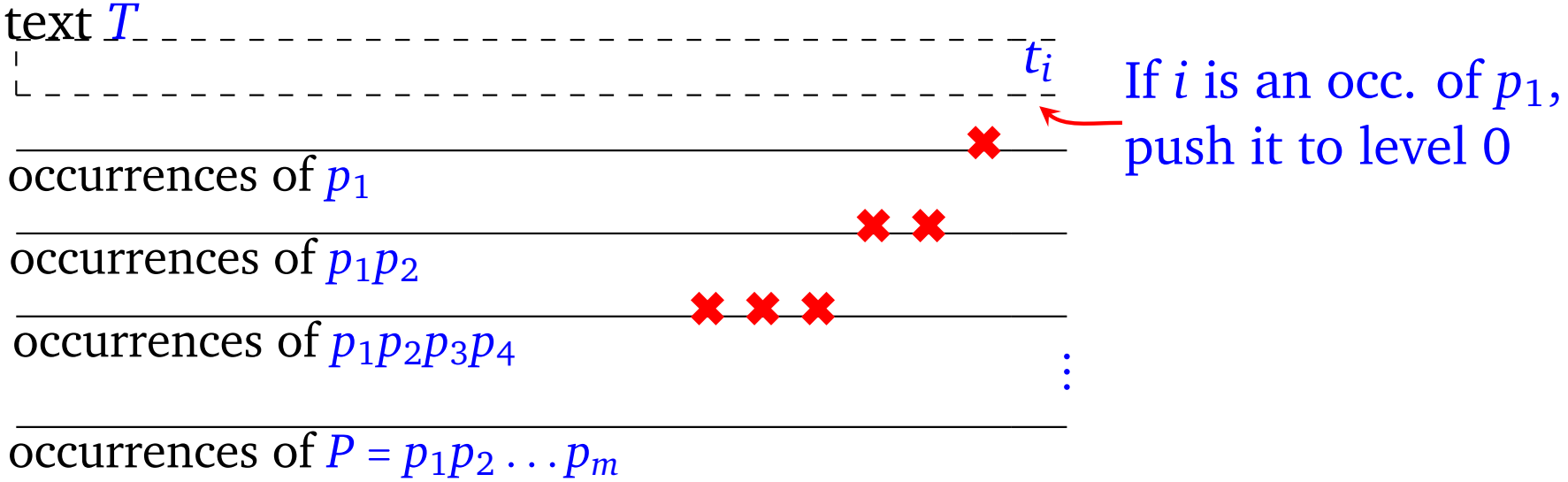
Porat & Porat, 2009



for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if lp is occurrence of $p_1 \ldots p_{2^{j+1}}$ **then** push lp to level $j + 1$

Correctness: if p is an occurrence of P, it is also an occurrence of all $\log m$ prefixes and will pass all the tests.

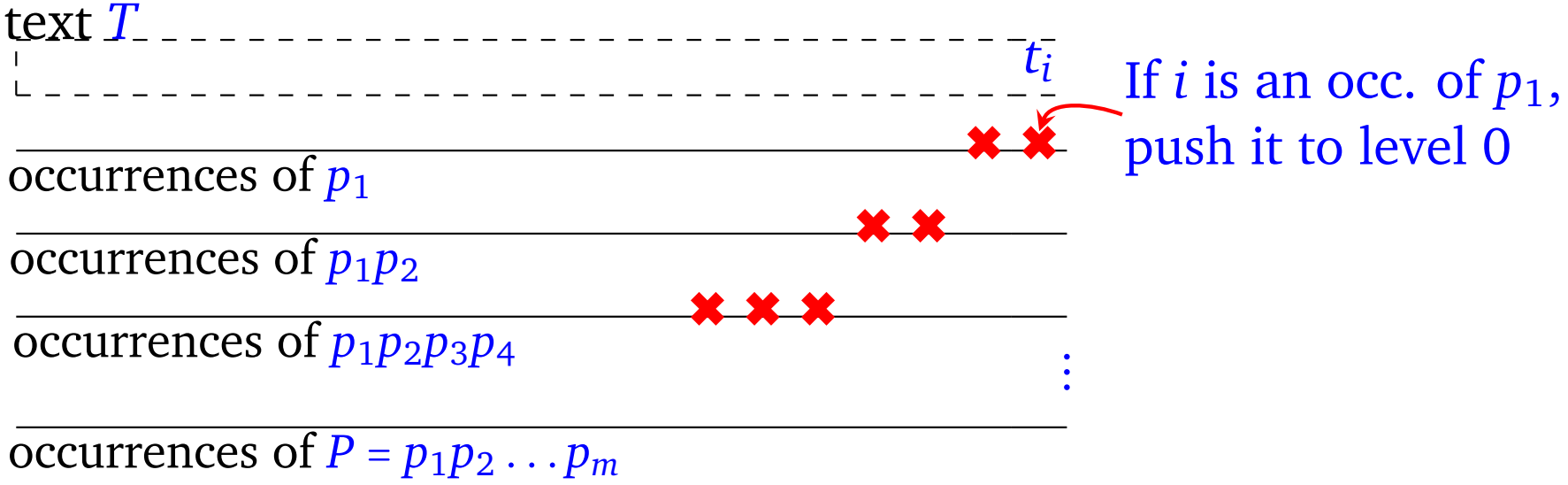
Porat & Porat, 2009



for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if lp is occurrence of $p_1 \ldots p_{2^{j+1}}$ **then** push lp to level $j + 1$

Correctness: if p is an occurrence of P, it is also an occurrence of all $\log m$ prefixes and will pass all the tests.

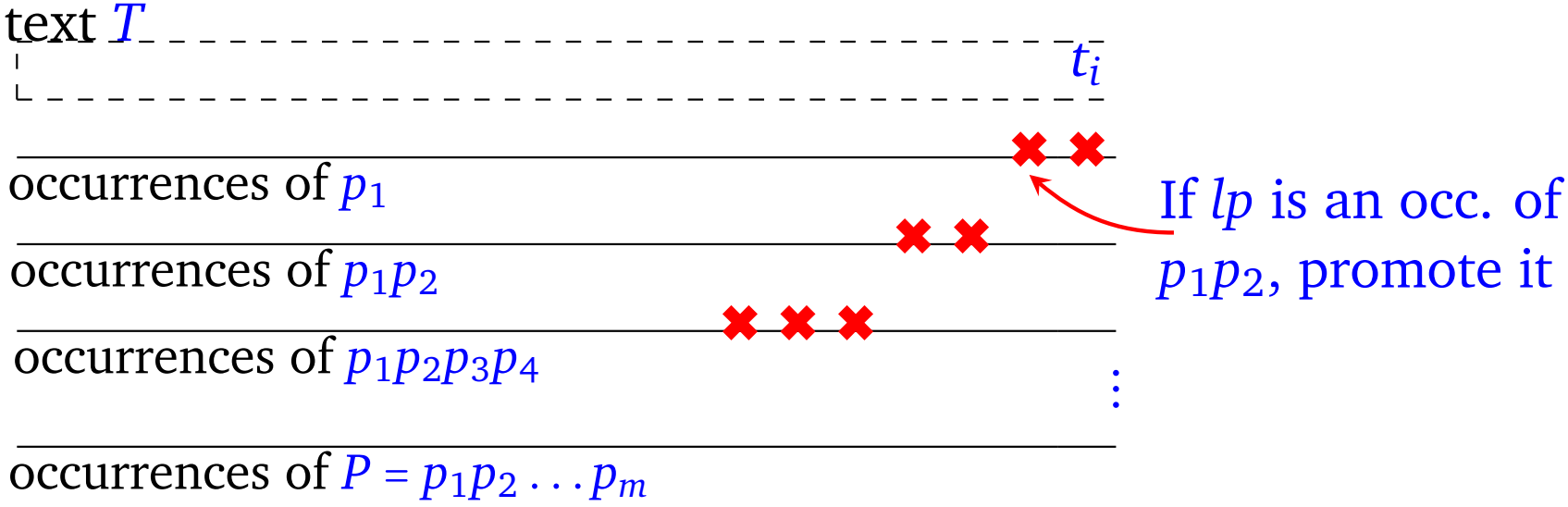
Porat & Porat, 2009



for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if lp is occurrence of $p_1 \ldots p_{2^{j+1}}$ **then** push lp to level $j + 1$

Correctness: if p is an occurrence of P, it is also an occurrence of all $\log m$ prefixes and will pass all the tests.

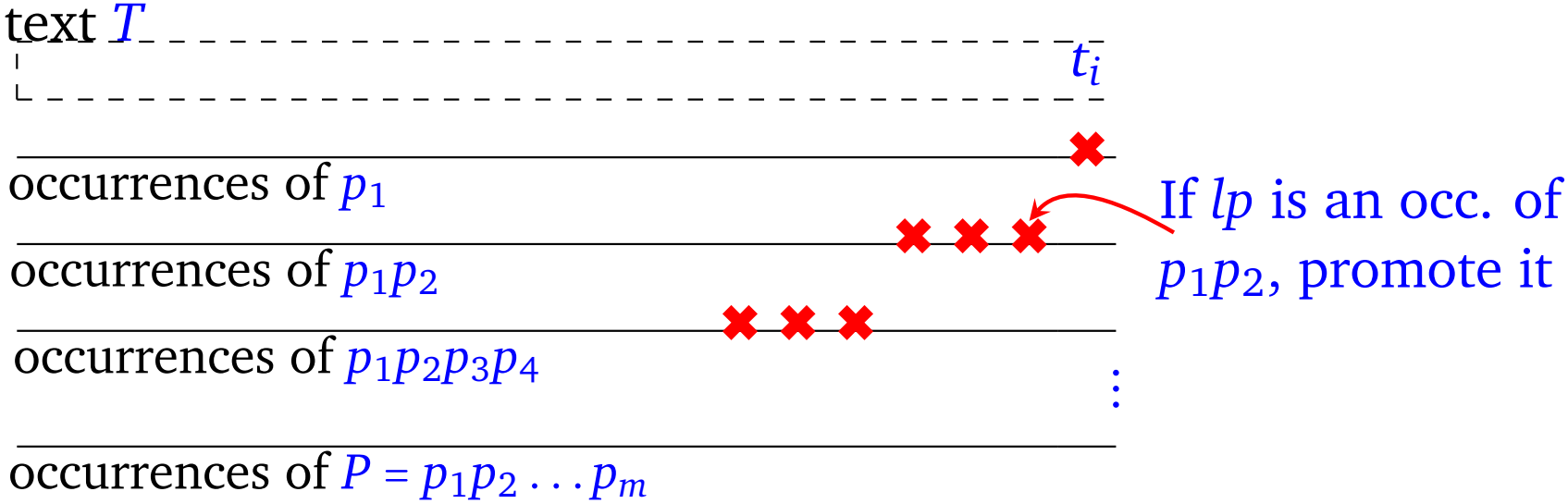
Porat & Porat, 2009



for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if lp is occurrence of $p_1 \ldots p_{2^{j+1}}$ **then** push lp to level $j + 1$

Correctness: if p is an occurrence of P, it is also an occurrence of all $\log m$ prefixes and will pass all the tests.

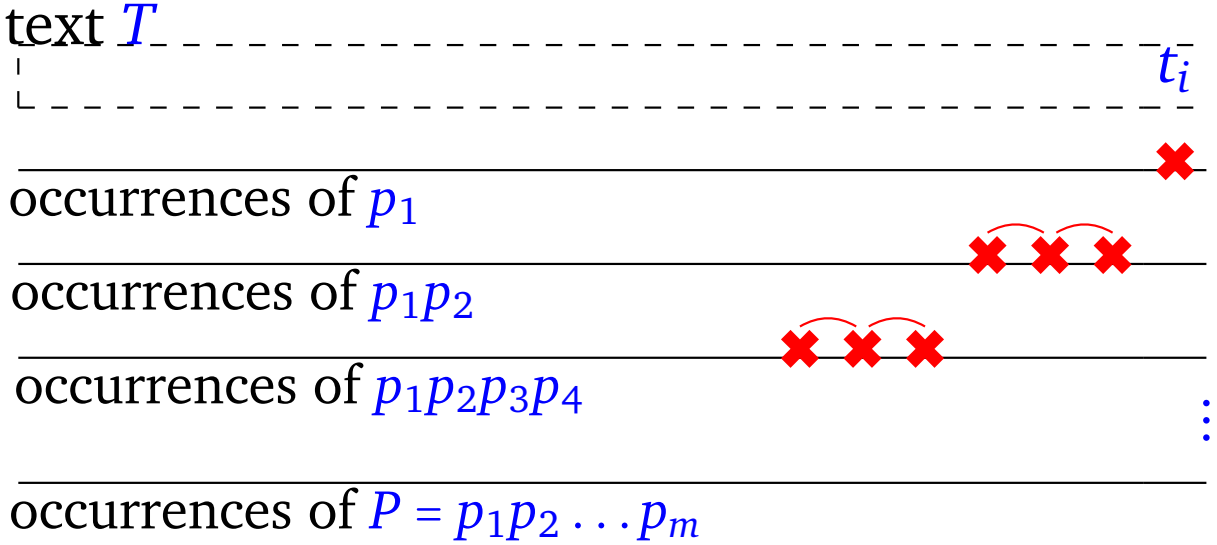
Porat & Porat, 2009



for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if lp is occurrence of $p_1 \ldots p_{2^{j+1}}$ **then** push lp to level $j + 1$

Correctness: if p is an occurrence of P, it is also an occurrence of all $\log m$ prefixes and will pass all the tests.

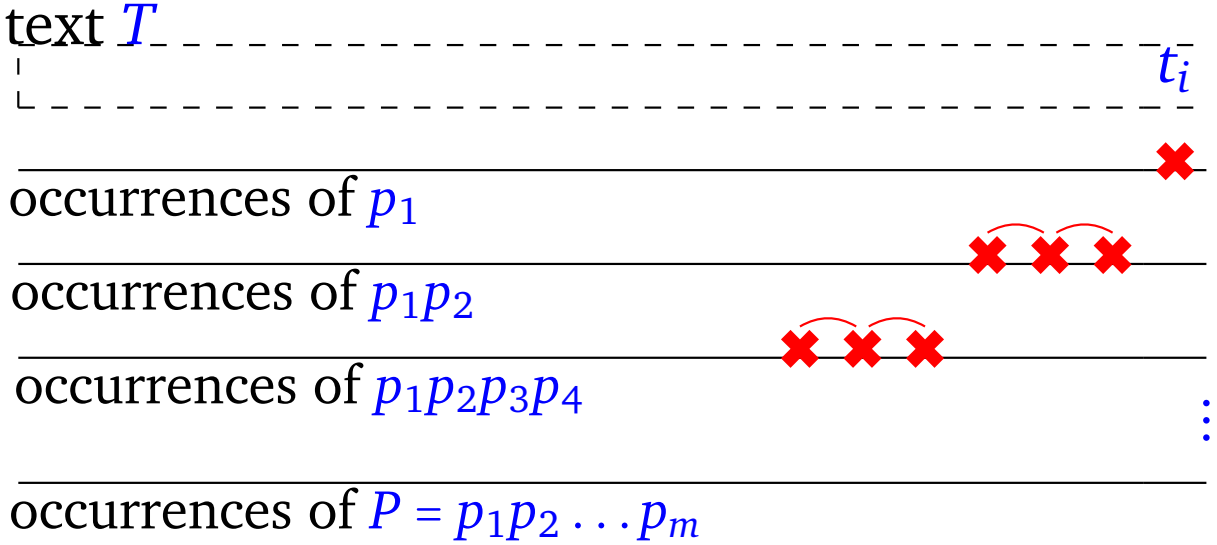
Porat & Porat, 2009



In level j, we store occurrences of $p_1p_2 \ldots p_{2^j}$ in $T[i - 2^{j+1} + 1, i]$. They form an arithmetic progression. We store:

- The progression
- Two fingerprints
- Periodicity and concatenability of fingerprints allow to test a position in constant time.

Porat & Porat, 2009

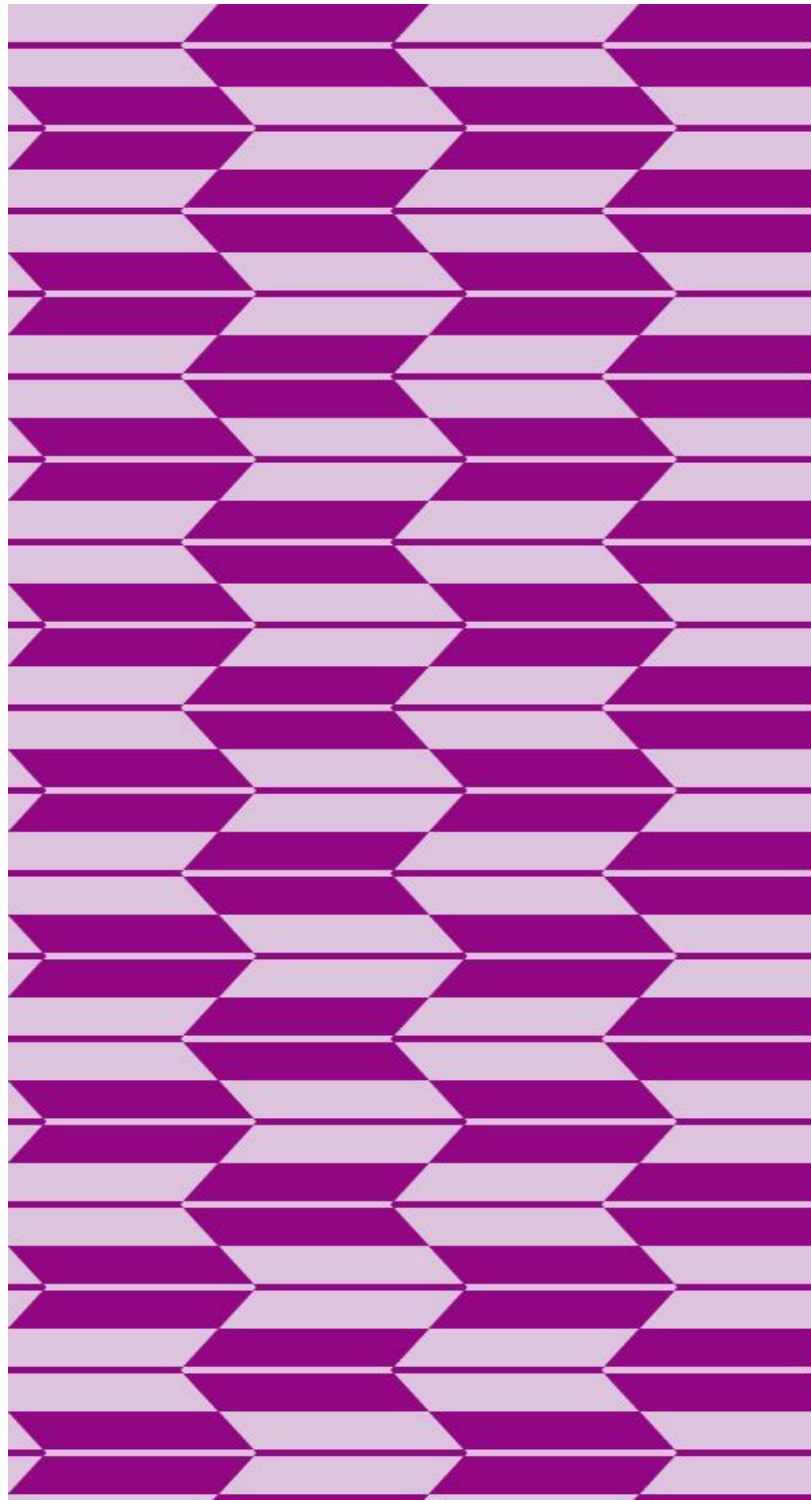


For each level we need:

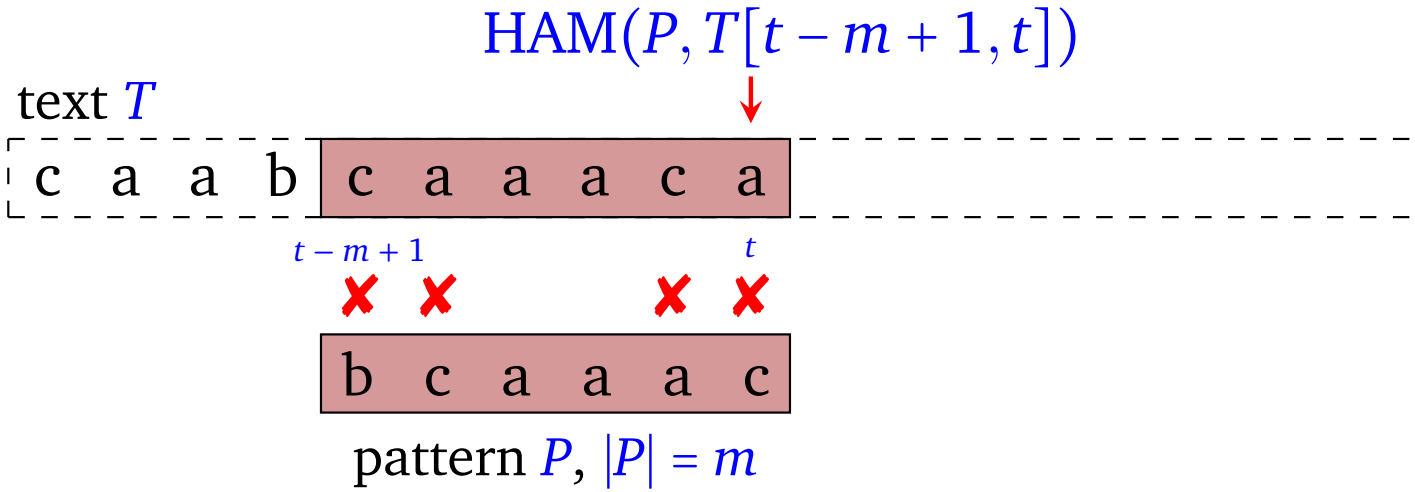
- $O(1)$ space
- $O(1)$ time for updating and extracting $\varphi(t_{lp} \ldots t_i)$

In total, the algorithm uses $O(\log m)$ space and $O(\log m)$ time per character. It is correct w.h.p.

Part II: k-mismatch pattern matching



k-mismatch pattern matching



- If $\mathrm{HAM}(P, T[t-m+1, t]) > k$, output "NO".
- Otherwise, output $\mathrm{HAM}(P, T)$.

Deterministic lower bound

Any deterministic streaming algorithm for 1-mismatch pattern matching must use $\Omega(m)$ space

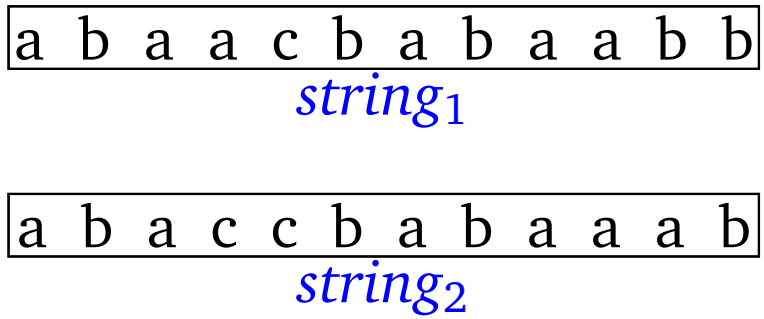
- Assume binary alphabet $\{0,1\}$ and $n = m$
- There are 2^m different patterns
- If we use $\leq m - \log(m+2)$ bits of memory, there are $m+2$ patterns $P_1, P_2, \ldots, P_{m+2}$ for which the states of the memory of the algorithm are equal after reading them
- Run the algorithm for each pattern P_i and $T = P_1$
- There are $m+1$ strings at the Hamming distance at most 1 from T
- At least one answer must be **different**, but this is impossible!

$\Rightarrow$ We need probability.

Upper bounds for k-mismatch pattern matching

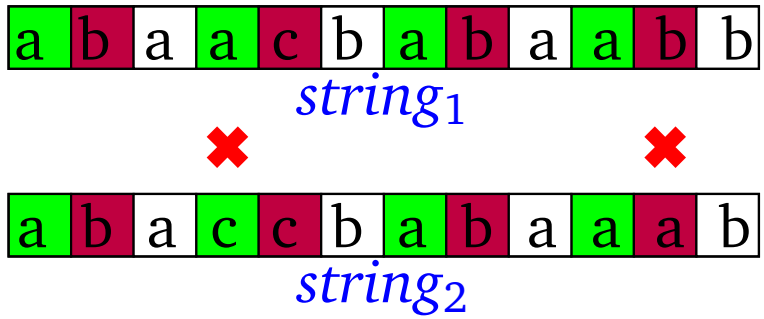
Authors	Space (words)	Time
Single pattern		
Porat & Porat, FOCS'09	$\tilde{O}(k^3)$	$\tilde{O}(k^2)$
Clifford, Fontaine, Porat, Sach, S., SODA'16	$\tilde{O}(k^2)$	$\tilde{O}(\sqrt{k})$
Golan, Kopelowitz, Porat, ICALP'18	$\tilde{O}(k)$	$\tilde{O}(k)$
Clifford, Kociumaka, Porat, SODA'19	$\tilde{O}(k)$	$\tilde{O}(\sqrt{k})$
Golan, Kociumaka, Kopelowitz, Porat, CPM'20	$\tilde{O}(s)$	$\tilde{O}(k/\sqrt{s})$ (amort.)
Dictionary of d patterns		
Gawrychowski, S., CPM'19	$\tilde{O}(kd\log^k d)$	$\tilde{O}(k\log^k d + occ)$

From one mismatch to exact pattern matching



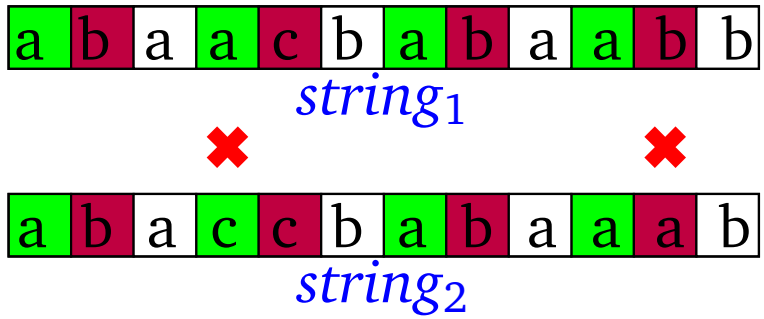
- Is HAM ($string_1$, $string_2$) = 1?

From one mismatch to exact pattern matching



- Is HAM($string_1$, $string_2$) = 1?
- Partition the strings into substrings of q colors
- One mismatch $\Rightarrow$ one pair of substrings does not match
- **Hope:** If there are ≥ 2 mismatches, they will end up in substrings of different colors $\Rightarrow$ at least 2 pairs of substrings do not match

From one mismatch to exact pattern matching



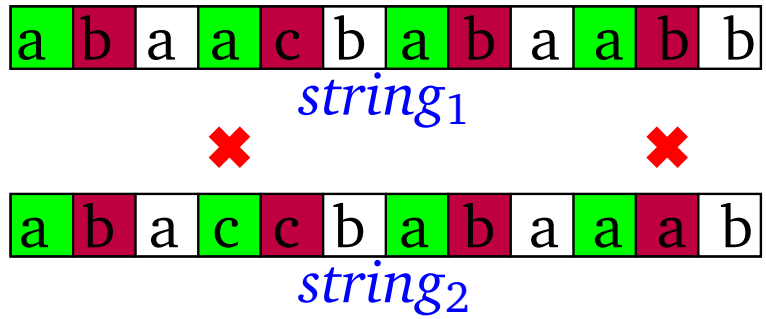
For each prime $q \in [\log m, \log^2 m]$:

Partition $string_1$ into q equi-spaced substrings

Partition $string_2$ into q equi-spaced substrings

In total: $\mathcal{O}(\log m)$ primes, and for each prime there are $O(\log^2 m)$ pairs of substrings

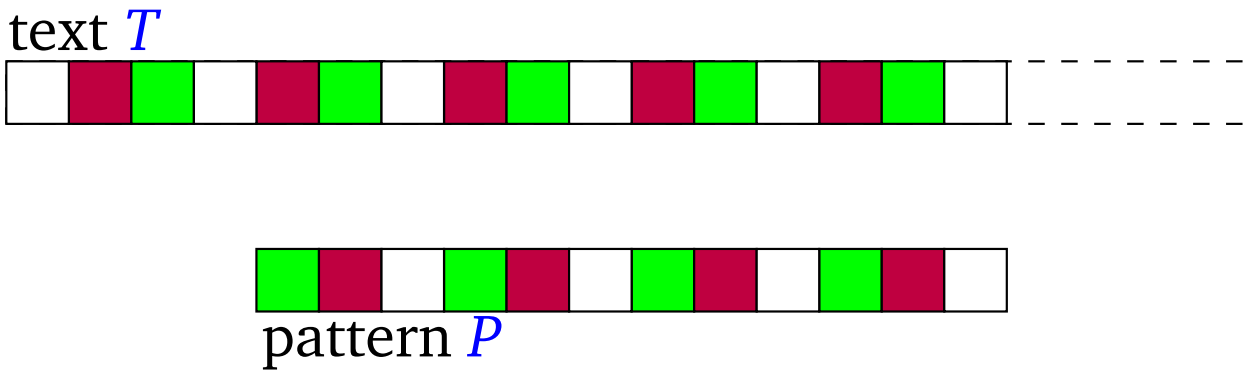
From one mismatch to exact pattern matching



Lemma There are ≥ 2 mismatches $\maltese_1, \maltese_2 \Rightarrow$ there exists a prime q such that at least two pairs of substrings do not match

- $\maltese_1, \maltese_2$ in the same pair $\Leftrightarrow \maltese_1 - \maltese_2 = 0 \pmod{q}$
- $m \geq \maltese_1 - \maltese_2$ cannot be a multiple of $\log m$ distinct primes

From one mismatch to exact pattern matching



Is HAM(P, T) = 1?

for each position of the text T **do**

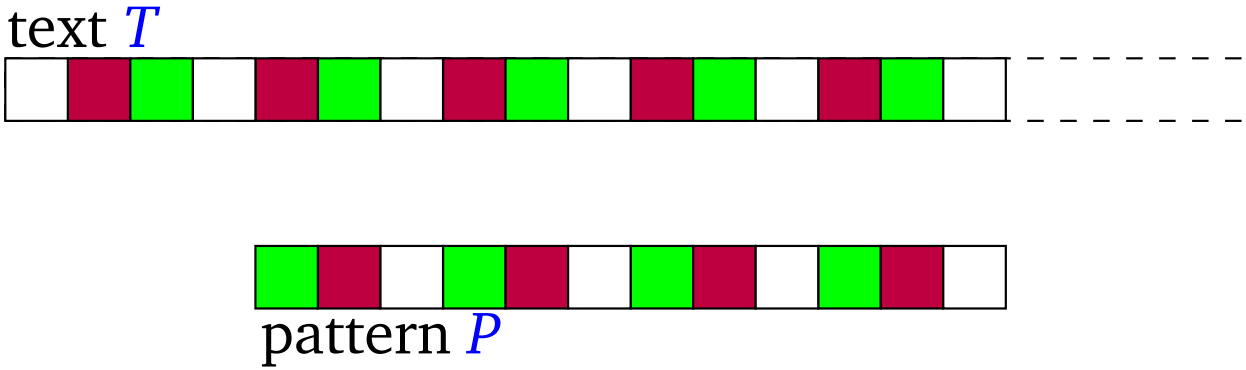
  for each prime q in $[\log m, \log^2 m]$ **do**

    $h \leftarrow$ number of (substream, subpattern) that mismatch

    if $h = 0$ **or** $h > 1$ **return** "NO"

  return "YES"

From one mismatch to exact pattern matching



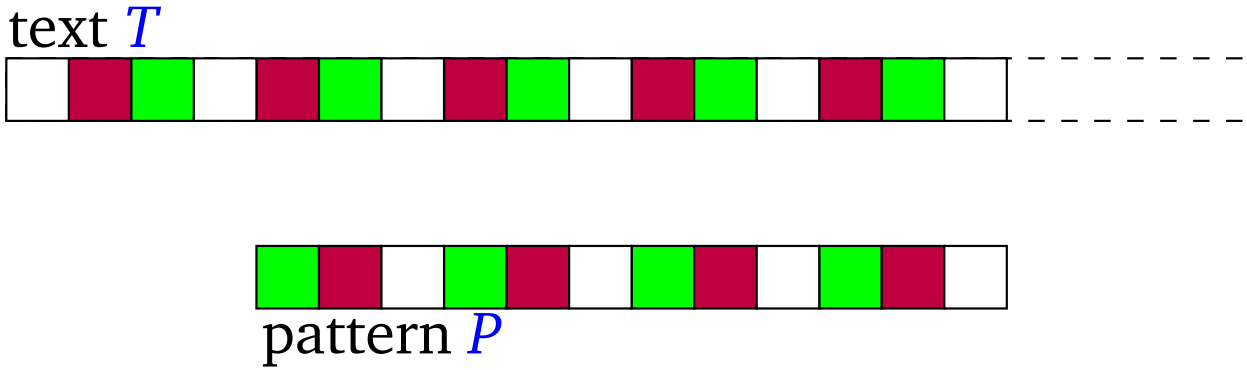
Compute number of mismatching pairs

for each prime q in $[\log m, \log^2 m]$ **do**

  for each (substream, subpattern) **do**

    run streaming exact pattern matching

From one mismatch to exact pattern matching



$$\text{Space} = O(\underbrace{\log m}_{\#\text{ of primes}} \cdot \underbrace{\log^2 m}_{\#\text{ of substr.}} \cdot \underbrace{\log^2 m}_{\#\text{ of subpatterns}} \cdot \log m)$$

$$\text{Time} = O(\underbrace{\log m}_{\#\text{ of primes}} \cdot \underbrace{\log^2 m}_{\#\text{ of substr.}} \cdot \underbrace{\log^2 m}_{\#\text{ of subpatterns}})$$

For arbitrary k: $\tilde{O}(k^3)$ space, $\tilde{O}(k^2)$ time (same as for $k = 1$ but take more subpatterns). This idea was also used in Clifford et al., SODA'16, and Golan, Kopelowitz, Porat, ICALP'18.

Clifford, Kociumaka, Porat, SODA'19

The k-mismatch sketch of a string $S = s_1 s_2 \dots s_m$ consists of:

$$\text{The Karp–Rabin fingerprint } \varphi(\mathbf{S}) = \sum_{i=1}^{i=m} s_i \cdot r^i \bmod p$$

and

$$\phi_{\mathbf{0}}(\mathbf{S}) = \sum_{i=1}^{i=m} s_i \bmod p, \dots, \phi_{\mathbf{2k}}(\mathbf{S}) = \sum_{i=1}^{i=m} s_i \cdot i^{2k} \bmod p$$

where p is a prime and r is a random integer $\in [0, p-1]$

Clifford, Kociumaka, Porat, SODA'19

Given the k-mismatch sketches of two strings S, T, in $\mathcal{O}(k)$ time one can decide whether $HD(S,T) \leq k$ w.h.p.

Suppose that $x_1, x_2, \ldots, x_{k'}$, $k' \leq k$, are the mismatches

$$\begin{cases} (\mathbf{S}[\mathbf{x_1}] - \mathbf{T}[\mathbf{x_1}]) + \ldots + (\mathbf{S}[\mathbf{x_{k'}}] - \mathbf{T}[\mathbf{x_{k'}}]) = \phi_{\mathbf{0}}(\mathbf{S}) - \phi_{\mathbf{0}}(\mathbf{T}) \\ (\mathbf{S}[\mathbf{x_1}] - \mathbf{T}[\mathbf{x_1}]) \cdot x_1 + \ldots + (\mathbf{S}[\mathbf{x_{k'}}] - \mathbf{T}[\mathbf{x_{k'}}]) \cdot x_{k'} = \phi_{\mathbf{1}}(\mathbf{S}) - \phi_{\mathbf{1}}(\mathbf{T}) \\ \ldots \\ (\mathbf{S}[\mathbf{x_1}] - \mathbf{T}[\mathbf{x_1}]) \cdot (x_1)^{2k} + \ldots + (\mathbf{S}[\mathbf{x_{k'}}] - \mathbf{T}[\mathbf{x_{k'}}]) \cdot (x_{k'})^{2k} = \phi_{\mathbf{2k}}(\mathbf{S}) - \phi_{\mathbf{2k}}(\mathbf{T}) \end{cases}$$

This set of equations is similar to those appearing in the decoding procedures for Reed—Solomon codes, and one can use the techniques from 1980's to find $x_1, x_2, \ldots, x_{k'}$

The Karp–Rabin fingerprints are used to verify the answer

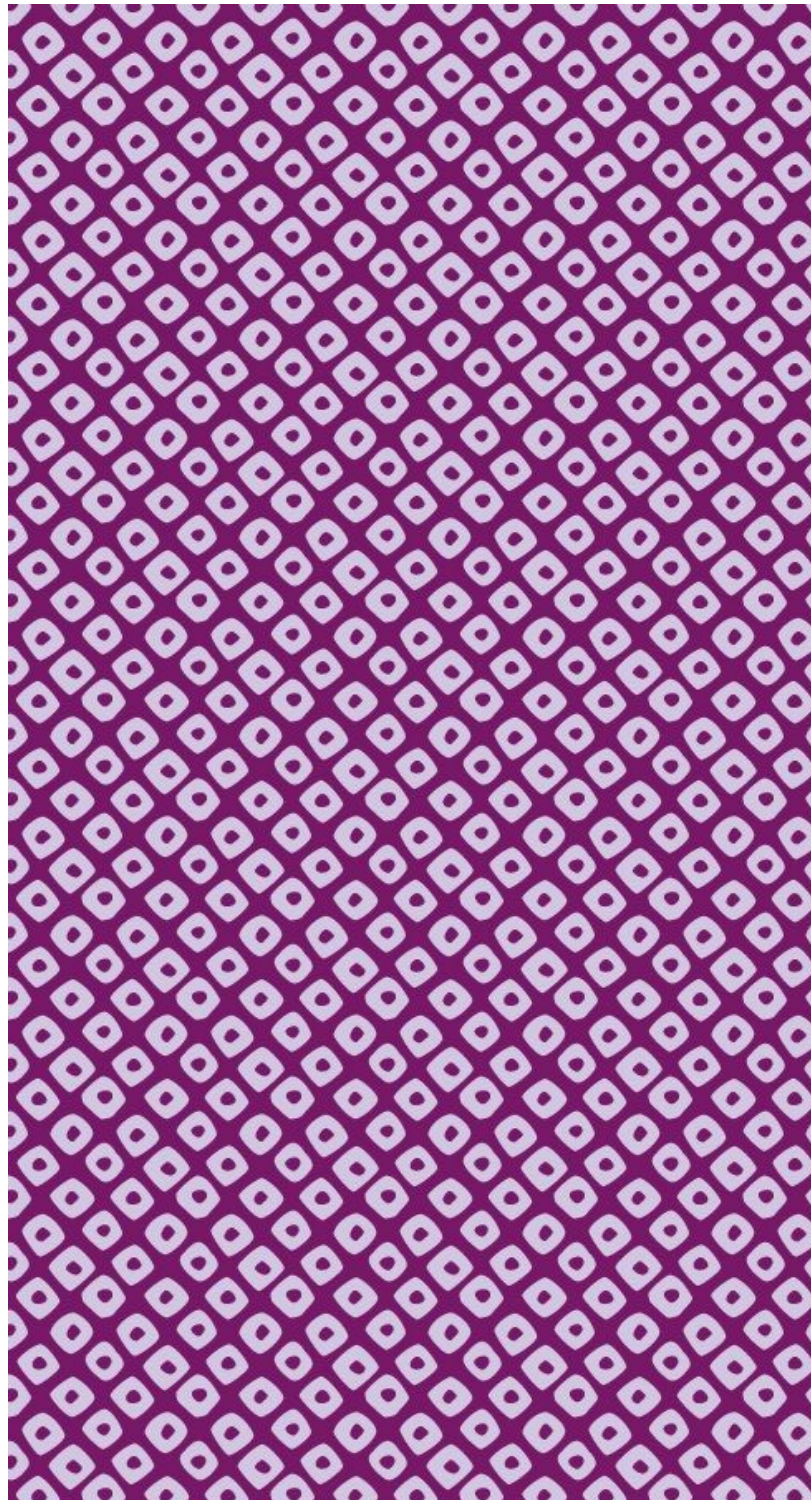
Clifford, Kociumaka, Porat, SODA'19

Given the k-mismatch sketches of two strings S, T, in $\mathcal{O}(k)$ time one can decide whether $HD(S, T) \leq k$ w.h.p.

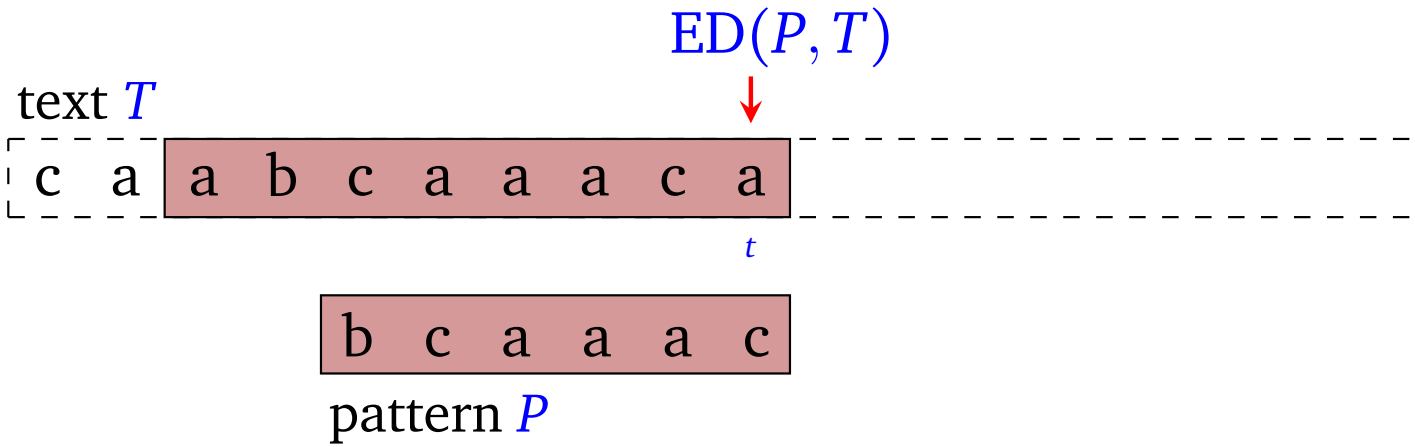
The sketch of Clifford, Kociumaka, Porat is **concatenatable**

Because of that, one can use the exponential prefixes approach, similar to the exact pattern matching algorithm of Porat & Porat, to achieve $\tilde{O}(k)$ space, $\tilde{O}(k)$ time algorithm

Part III: k-edit pattern matching



k-edit pattern matching



$\mathrm{ED}(P,S)$ = minimum number of insertions, deletions, and substitutions that transform P into S

Example: P = aaac, S = abacb, edit distance = 2

- For each position t, $d_t := \min_{S\text{ – suffix of }T[1,t]} \mathrm{ED}(S,P)$
- If $d_t > k$, output "NO", and otherwise output d_t

We can show $\Omega(m)$ lower bound for deterministic algorithms

k-edit pattern matching

Authors	Space (words)	Time
S., CPM'17	$\tilde{O}(\sqrt{m} \cdot poly(k))$	$\tilde{O}(\sqrt{m} \cdot poly(k))$
Kociumaka, Porat, S., FOCS'21	$\tilde{O}(k^5)$	$\tilde{O}(\text{poly}(k))$
Bhattacharya, Koucký, ICALP'23	$\tilde{O}(k^2)$	$\tilde{O}(k^2)$

Main challenge: no concatenatable sketch for edit distance

Edit distance sketches

Ideally: $\tilde{O}(\text{poly}(k))$-space sketches $\text{sk}_{\text{ED}}(X), \text{sk}_{\text{ED}}(Y)$ that can be computed in streaming and allow computing $\min\{\text{ED}(X,Y), k+1\}$ in $\tilde{O}(\text{poly}(k))$ time.

Authors	Space (words)	Time	
Belazzougui, Zhang, FOCS'16	$\tilde{O}(k^8)$	$\tilde{O}(\text{poly}(k))$	random walk embedding from ED to HD (CGK, STOC'16)
Jin, Nelson, Wu, STACS'21	$\tilde{O}(k^3)$	$\tilde{O}(\text{poly}(k))$	
Kociumaka, Porat, S., FOCS'21	$\tilde{O}(k^2)$	$\tilde{O}(k^3)$	
Bhattacharya, Koucký, STOC'23	$\tilde{O}(k^2)$	$\tilde{O}(k^2)$	Locally consistent string parsing

OQ: What are the space-time lower bounds?

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	**0**	1	0	...	0
$\mu(X)$:					

text position = 1, j = 1

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	**0**	1	0	...	0
$\mu(X)$:	0				

text position = 1, j = 1

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	**0**	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	**0**	1	0	...	0
$\mu(X)$:	0				

text position = 1, j = 1

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	**0**	1	0	...	0
$\mu(X)$:	0				

text position = 1, j = 2

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	**0**	1	0	...	0
$\mu(X)$:	0	0			

text position = 1, j = 2

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	**1**	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	**0**	1	0	...	0
$\mu(X)$:	0	0			

text position = 1, j = 2

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	0	**1**	0	...	0
$\mu(X)$:	0	0			

text position = 2, j = 3

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	0	**1**	0	...	0
$\mu(X)$:	0	0	1		

text position = 2, j = 3

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			3n
0	0	1	1	0	1	1	0	0			0
1	1	**1**	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	0	**1**	0	...	0
$\mu(X)$:	0	0	1		

text position = 2, j = 3

1. Copy $X[i]$. If $h_j(X[i]) = 1$, move to the right;
2. $j = j + 1$.

Random walk embedding

Pick $3n$ random functions $h_j : \{0,1\} \to \{0,1\}$

	1	2	3	4	5	6	7	8			$3n$
0	0	1	1	0	1	1	0	0			0
1	1	1	1	1	0	1	0	1			1

Copy letters of X to $\mu(X)$:

	1	2	3		n
X :	0	1	0		0
$\mu(X)$:	0	0	1		

text position = 2, j = 3

When the length of $\mu(X)$ reaches $3n$, stop. If the length of $\mu(X) < 3n$, append with zeros.

The embedding can be considered as a random walk on X

Theorem. [Chakraborty et al., STOC'16] If $\mathrm{ED}(X,Y) = k$, then $k/2 \le \mathrm{HAM}(\mu(X), \mu(Y)) \le \mathcal{O}(k^2)$ with probability 0.99.

Edit distance sketch

Belazzougui and Zhang FOCS'16

- A random walk on X, Y induces an alignment between X, Y of cost $\mathrm{HAM}(\mu(X), \mu(Y))$
- The alignment and all deleted or substituted characters of X, Y can be retrieved from the Hamming sketches of $\mu(X), \mu(Y)$
- It is not necessarily an optimal alignment, but $\mathrm{poly}(k, \log n)$ such alignments contain enough information to recover an optimal one with constant probability

Random walk based ED sketches

Belazzougui and Zhang, FOCS'16: $\tilde{O}(k^2)$ independent embeddings, for each embedding two Hamming distance sketches encoding the *hierarchical* and the *structural* properties — $\tilde{O}(k^8)$ **space,** $\tilde{O}(\text{poly}(k))$ **time**

Jin, Nelson, Wu, STACS'21: more careful analysis of Belazzougui and Zhang — $\tilde{O}(k^3)$ **space,** $\tilde{O}(\text{poly}(k))$ **time**

Kociumaka, Porat, S., FOCS'21: one embedding $X \to \mu(X)$ + sampling compressible regions of $\mu(X)$ + Hamming distance sketch on the resulting sting — $\tilde{O}(k^2)$ **space,** $\tilde{O}(k^3)$ **time**

k-edit pattern matching

Input: Pattern P, text T
Output: all positions r s.t. for some $1 \leq \ell \leq r$, $\mathrm{ED}(P, T[\ell, r]) \leq k$

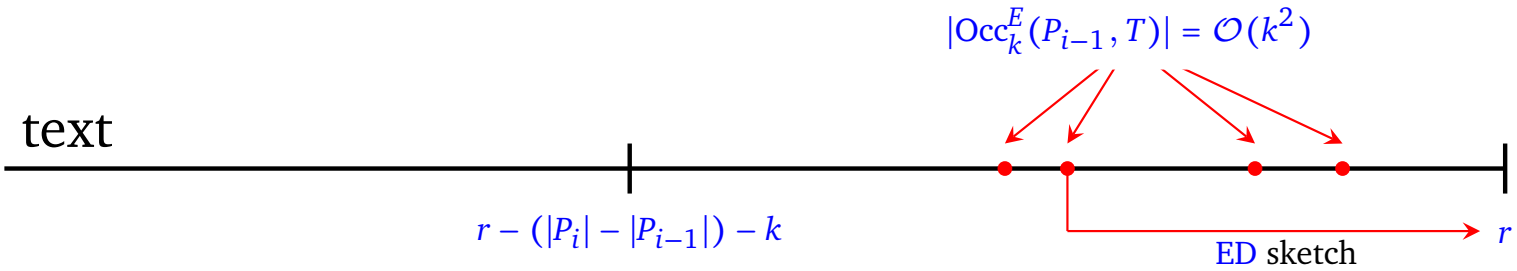
- High-level structure: logarithmic number of prefixes P_i of P of exponentially increasing lengths
- Goal: retrieve k-edit occurrences of P_i from k-edit occurrences of P_{i-1}
- Challenge: still no concatenatable sketches for edit distance

k-edit pattern matching

A string X is *k-periodic* if there exists a (short) string Q such that the edit distance between X and a prefix of Q^∞ is at most $2k$

Charalampopoulos, Kociumaka, Wellnitz, FOCS'20

X is not k-periodic and $|Y| \le 2|X| \Rightarrow |\mathrm{Occ}_k^E(X,Y)| = \mathcal{O}(k^2)$



- For each occurrence in $\mathrm{Occ}_k^E(P_{i-1},T)$, launch the algorithm for computing the edit distance sketch
- When arriving to r, use this sketch and the sketch of $P[|P_{i-1}|+1,|P_i|]$ to decide whether $r \in \mathrm{Occ}_k^E(P_i,T)$

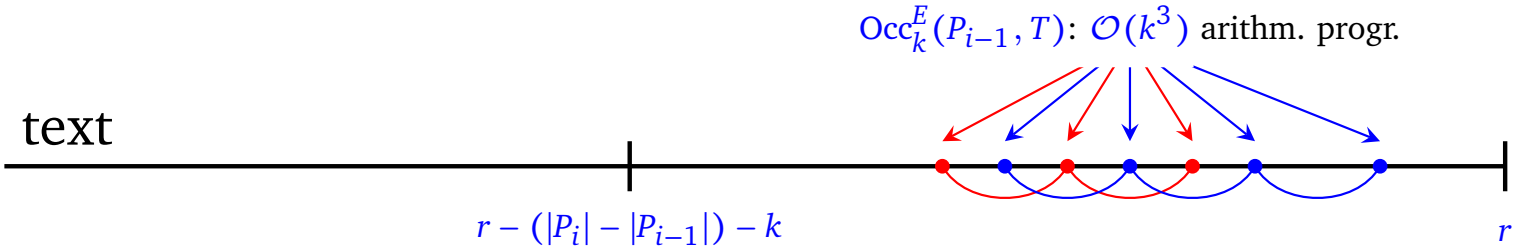
k-edit pattern matching

Kociumaka, Porat, S., FOCS'21

A string X is *k-periodic* if there exist a (short) string Q such that the edit distance between X and a prefix of Q^∞ is at most $2k$

Charalampopoulos, Kociumaka, Wellnitz, FOCS'20

If X is k-periodic and $|Y| \le 2|X|$, then $\mathrm{Occ}_k^E(X,Y)$ can be decomposed in $\mathcal{O}(k^3)$ arithmetic progressions whose differences are one of $\mathcal{O}(k)$ rotations of Q



Unfortunately, we cannot make use of this structure directly

k-edit pattern matching

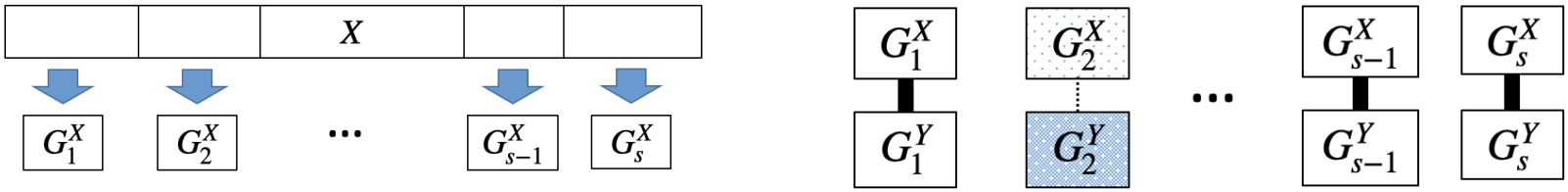
Kociumaka, Porat, S., FOCS'21

- Kociumaka, Porat, S., FOCS'21 introduced a novel *greedy encoding* $\mathrm{Gr}_k(X, Y)$ that represents a large class of low-distance alignments between X and Y
- Can be computed from ED sketches of X and Y and occupies $\tilde{\mathcal{O}}(k^2)$ space
- Supports concatenations and products (transitive compositions)

These properties allow to use the encoding in the way similar we used sketches for exact pattern matching and pattern matching under Hamming distance

k-edit pattern matching

Bhatacharrya, Koucký, ICALP'21



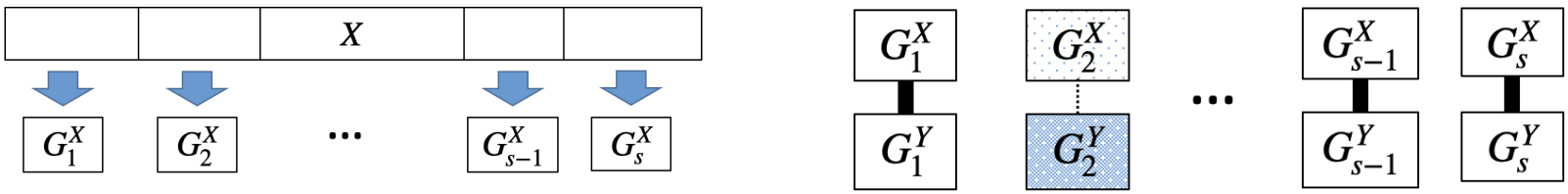
Partition X, Y into blocks so that:

- Blocks of X, Y match each other except for $\leq k$ blocks that contain edits;
- Blocks can be represented with grammars of size $\tilde{O}(k)$;
- $ED(X, Y) = \sum_i ED(eval(G_i^X, G_i^Y))$

Main idea: locally-consistent parsing

k-edit pattern matching

Bhatacharrya, Koucký, ICALP'21



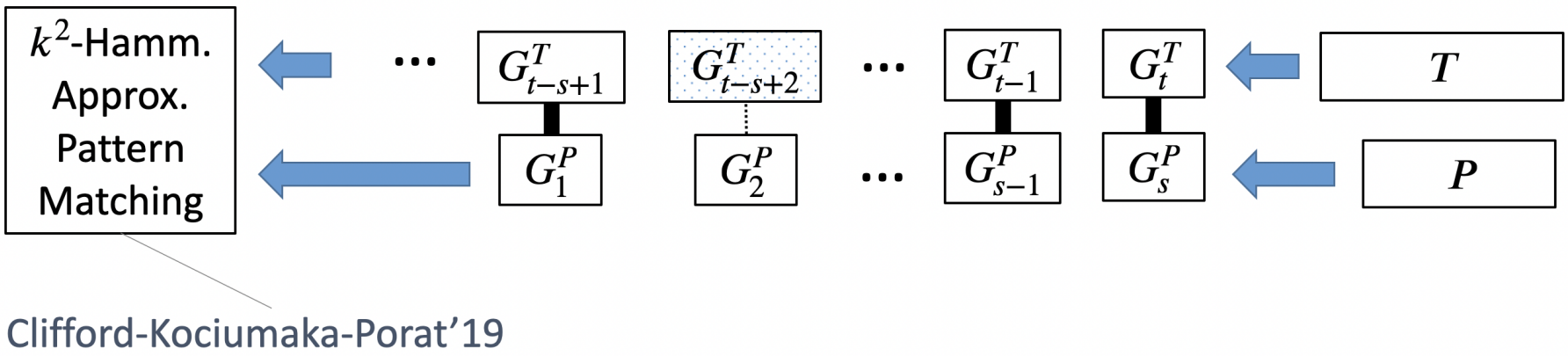
k-edit sketch:

- Concatenate the grammars into a string;
- Build the $\tilde{O}(k^2)$-mismatch sketch on top.

Evaluating ED: Extract the mismatching grammars and compute the edit distances between them.

k-edit pattern matching

Bhatacharrya, Koucký, ICALP'23



Key idea: $\tilde{O}(k^2)$-mismatch pattern matching on grammars

Streaming pattern matching

- Deterministic algorithms: $\Omega(m)$ space
- Probability + periodicity:
  - $\mathcal{O}(\log m)$ space & $\mathcal{O}(1)$ time for exact;
  - $\tilde{\mathcal{O}}(k)$ space & $\tilde{\mathcal{O}}(\sqrt{k})$ time for k-mismatch;
  - $\tilde{\mathcal{O}}(k^2)$ space & $\tilde{\mathcal{O}}(k^2)$ time for k-edit.

What are the limits?