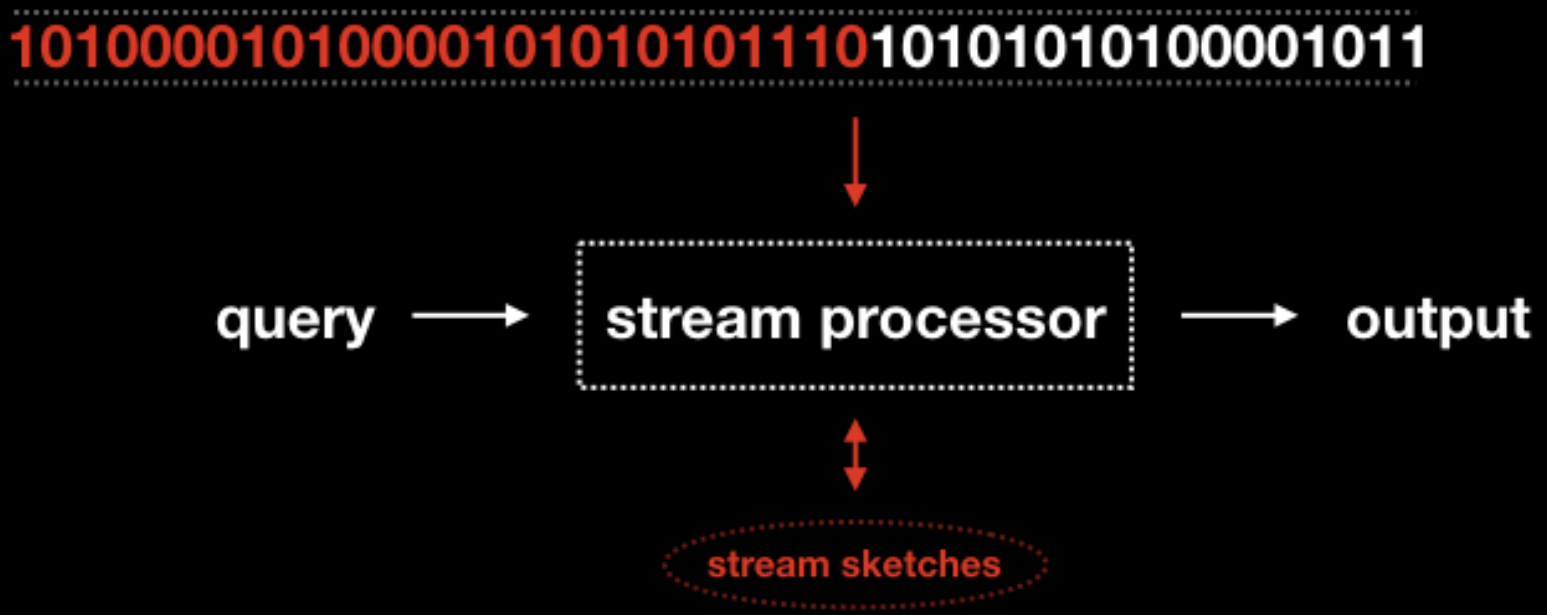
Sketches and streaming algorithms for string processing

Tatiana Starikovskaya

Streaming model



Objectives: real time & small space
When to use: stream data, big data

Classical approaches won't work: to answer a question deterministically and exactly, we need to store the input in full

Relaxations: randomisation + approximation

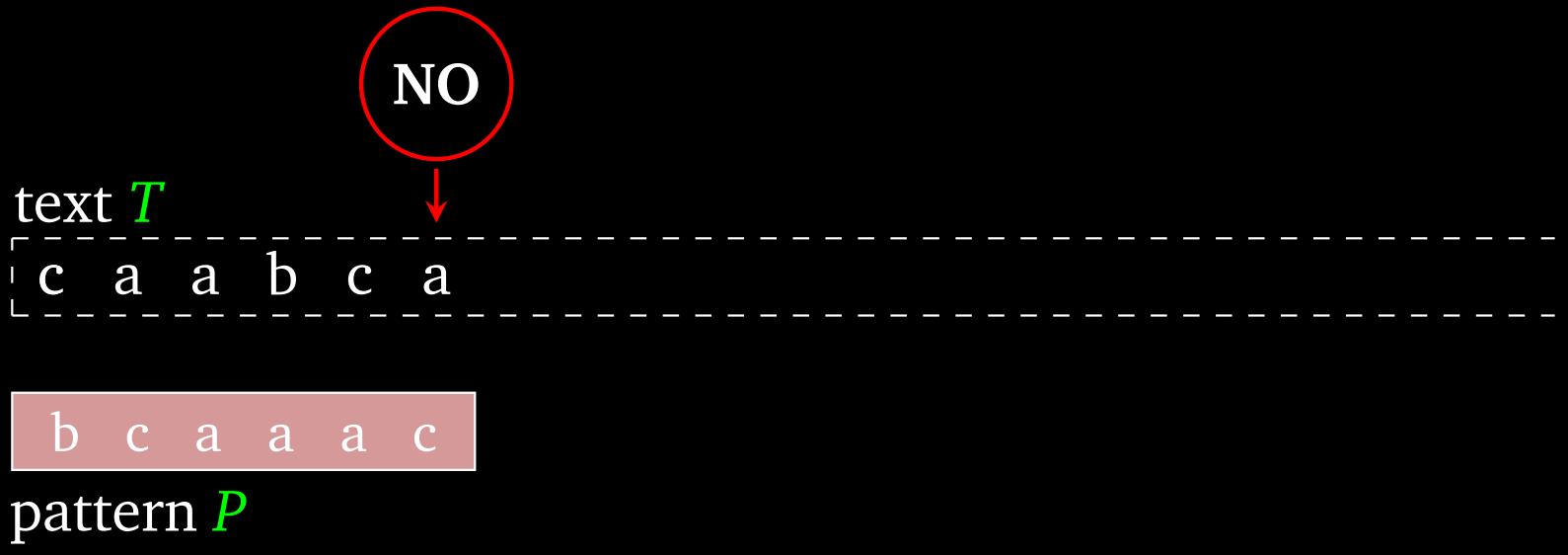
Tools: sketches (= lossy compression of the data) — capture essential properties of the data

Outline of today's talk

- Part I: Exact pattern matching
- Part II: String similarity and approximate pattern matching
- Part III: Periodicity

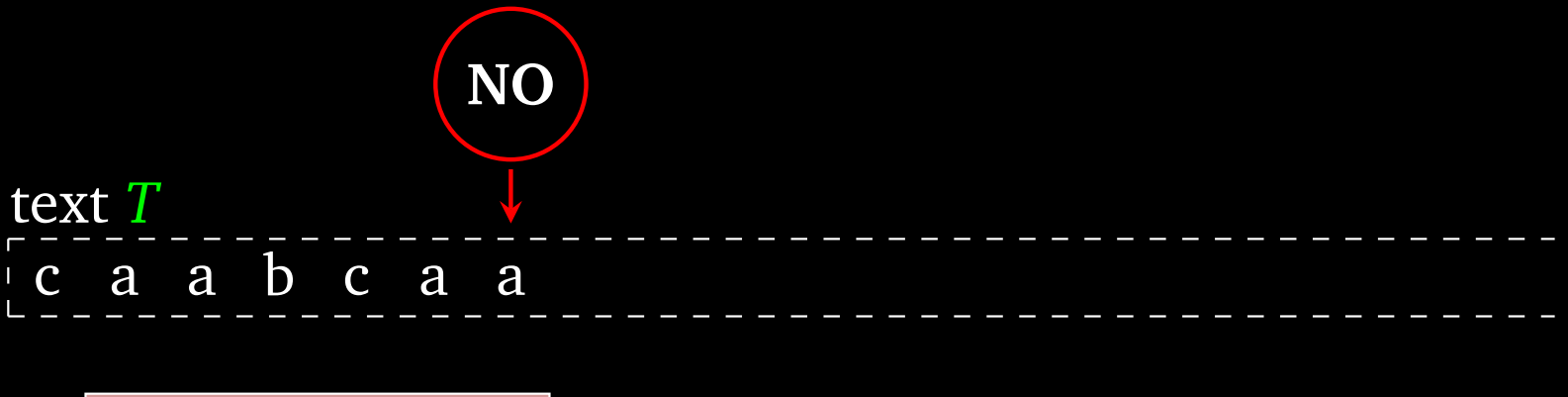
Part I: Exact pattern matching

Exact pattern matching



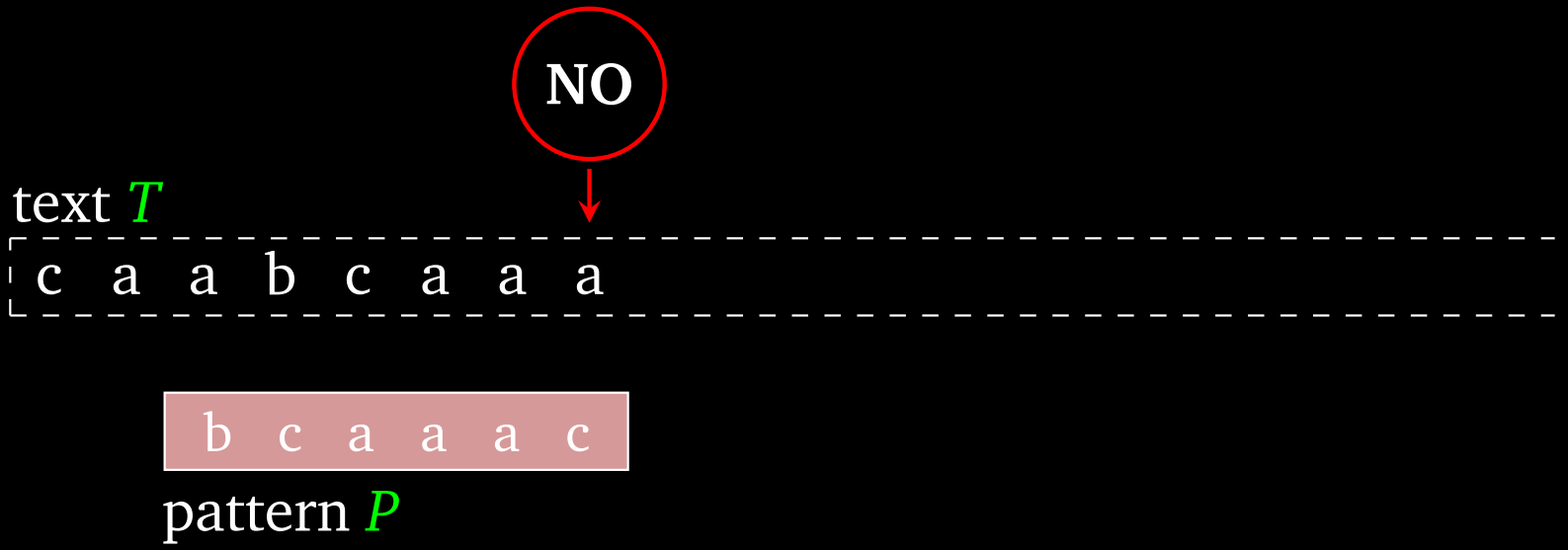
- **Query** = "Is there an occurrence of P?"
- **Space** = total space used by the stream processor
- **Time** = time per position of T

Exact pattern matching



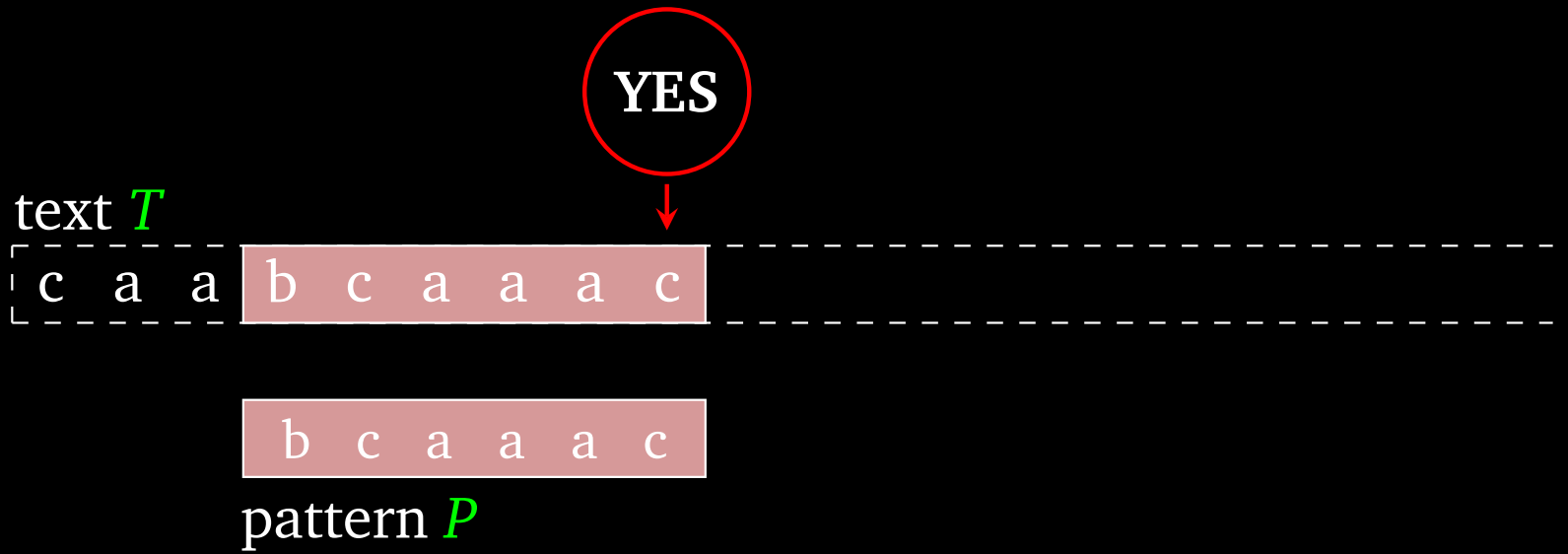
- **Query** = "Is there an occurrence of P?"
- **Space** = total space used by the stream processor
- **Time** = time per position of T

Exact pattern matching



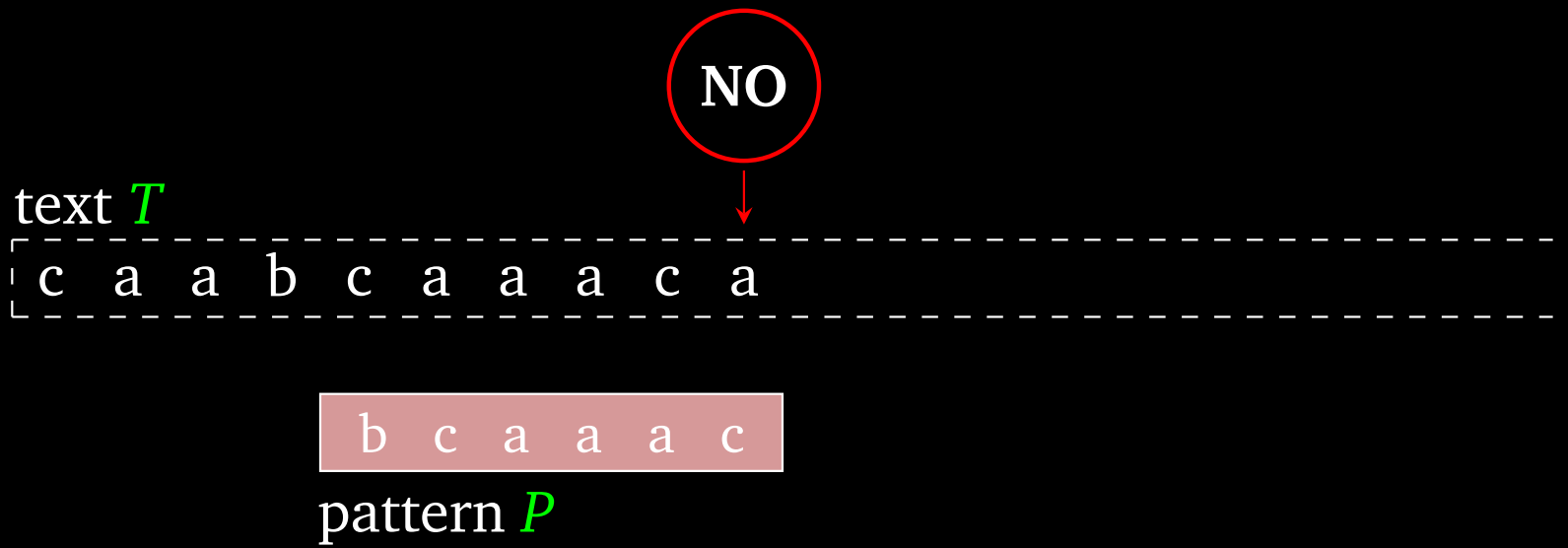
- **Query** = "Is there an occurrence of P?"
- **Space** = total space used by the stream processor
- **Time** = time per position of T

Exact pattern matching



- **Query** = "Is there an occurrence of P?"
- **Space** = total space used by the stream processor
- **Time** = time per position of T

Exact pattern matching



- **Query** = "Is there an occurrence of P?"
- **Space** = total space used by the stream processor
- **Time** = time per position of T

Karp–Rabin algorithm

Karp–Rabin fingerprint

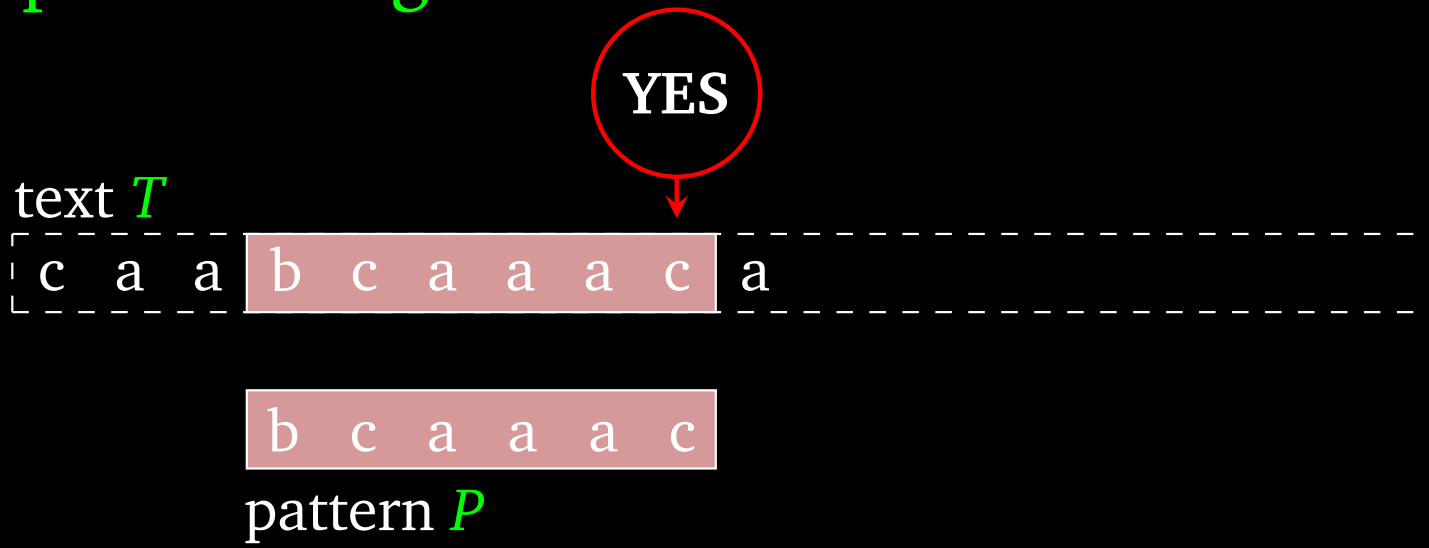
$$\varphi(s_1 s_2 \ldots s_m) = \sum_{i=1}^{m} s_i \cdot r^{m-i} \bmod p$$

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

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



When a new character $t_i = a$ arrives:

1. Update $\varphi(t_{i-m+1} \dots t_{i-1} t_i) = \sum_{i=1}^{m} t_{i-m+j} \cdot r^{m-j} \bmod p$:

$$\varphi(t_{i-m+1} \dots t_{i-1} t_i) = (\varphi(t_{i-m} \dots t_{i-1} t_{i-1}) - t_{i-m} \cdot r^{m-1}) \cdot r + t_i \bmod p$$

2. If $\varphi(t_{i-m+1} \dots t_{i-1} t_i) = \varphi(P)$, output "YES"

We need t_{i-m} to update the fingerprint $\Rightarrow$ **we must store** $t_{i-m}, \dots, t_{i-1}$

Exact pattern matching

Authors	Space [1]	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	$O(d \log m)$ $O(\lvert\Sigma\rvert^{\varepsilon} d \log(m/\varepsilon))$	$O(\log\log \lvert\Sigma\rvert)$ $O(1/\varepsilon)$

[1] In words

Exact pattern matching

Authors	Space [1]	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	$O(d \log m)$ $O(\lvert\Sigma\rvert^{\varepsilon} d \log(m/\varepsilon))$	$O(\log\log \lvert\Sigma\rvert)$ $O(1/\varepsilon)$

[1] In words

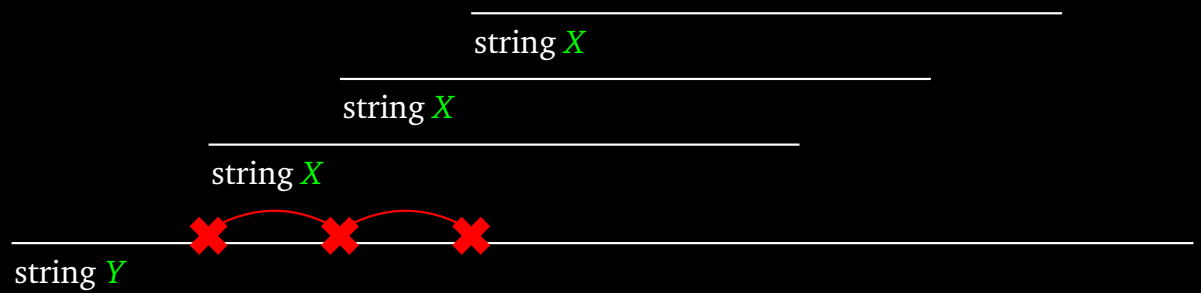
Porat & Porat, FOCS'09 ★

Fine and Wilf's periodicity lemma

If a string Q has two periods of length p and q and $p + q \leq |Q|$, then Q also has a period of length $gcd(p, q)$.

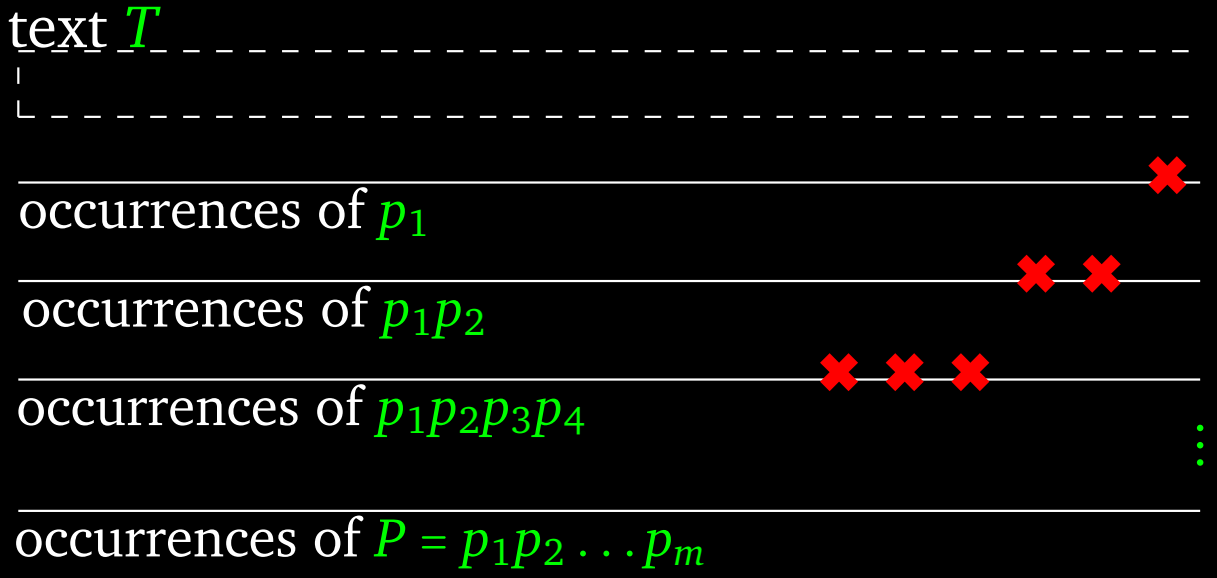
Corollary

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.



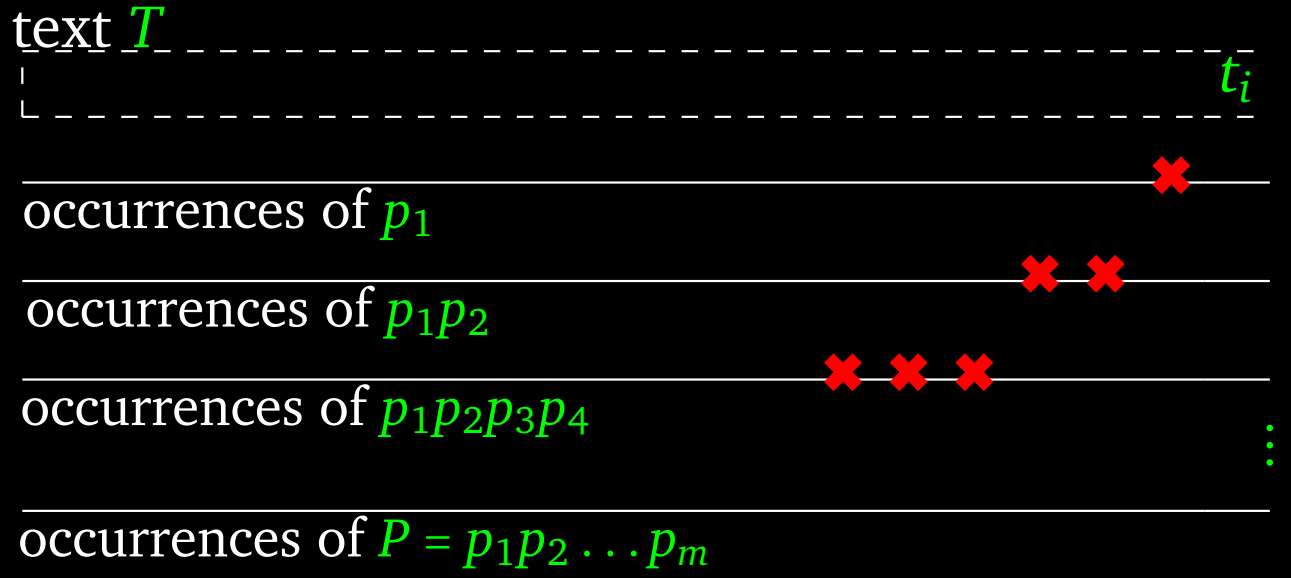
Occurrences of X in Y can be stored in $\mathcal{O}(1)$ space.

Porat & Porat, 2009 ★



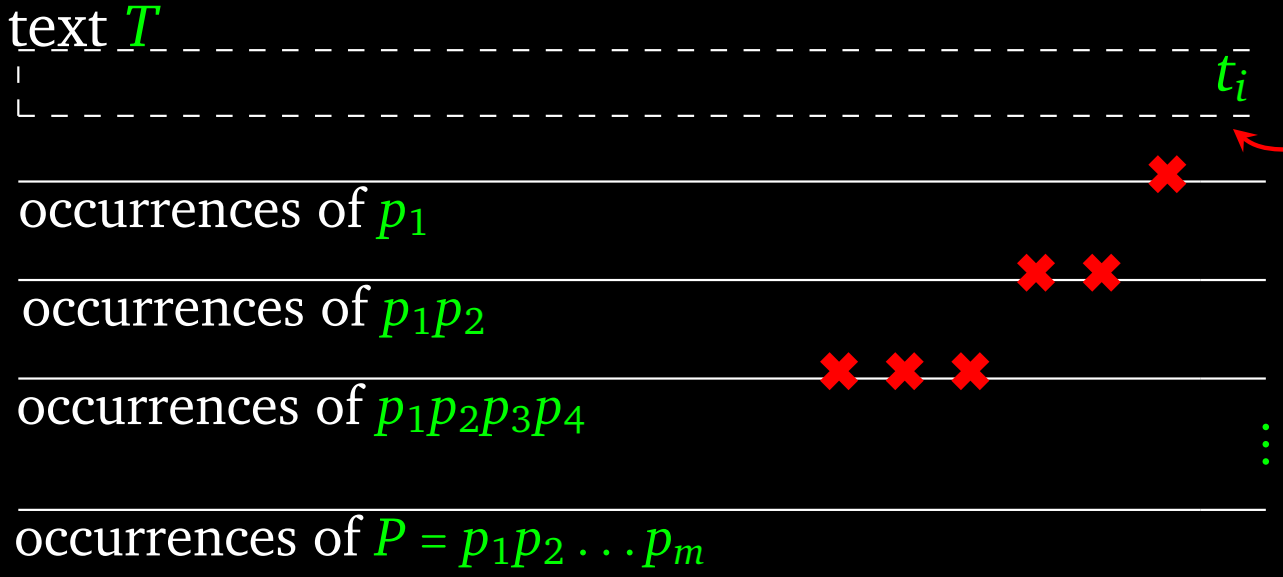
for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ **then**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

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 $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ **then** push lp to level $j + 1$

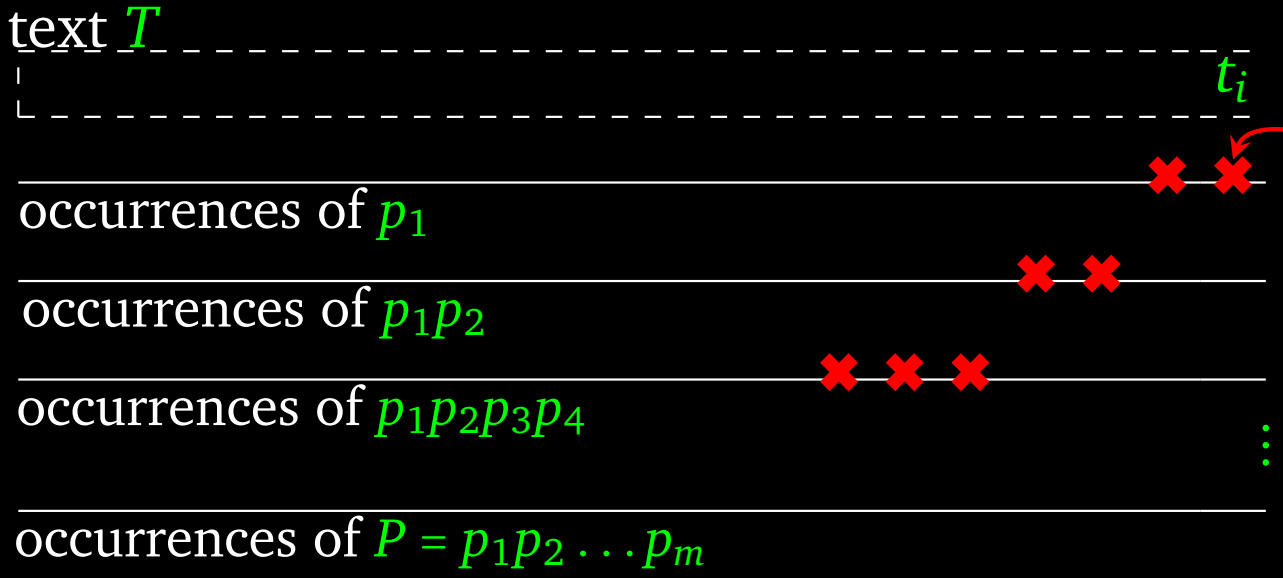
Porat & Porat, 2009 ★



If i is an occ. of p_1, push it to level 0

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 $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ **then** push lp to level $j + 1$

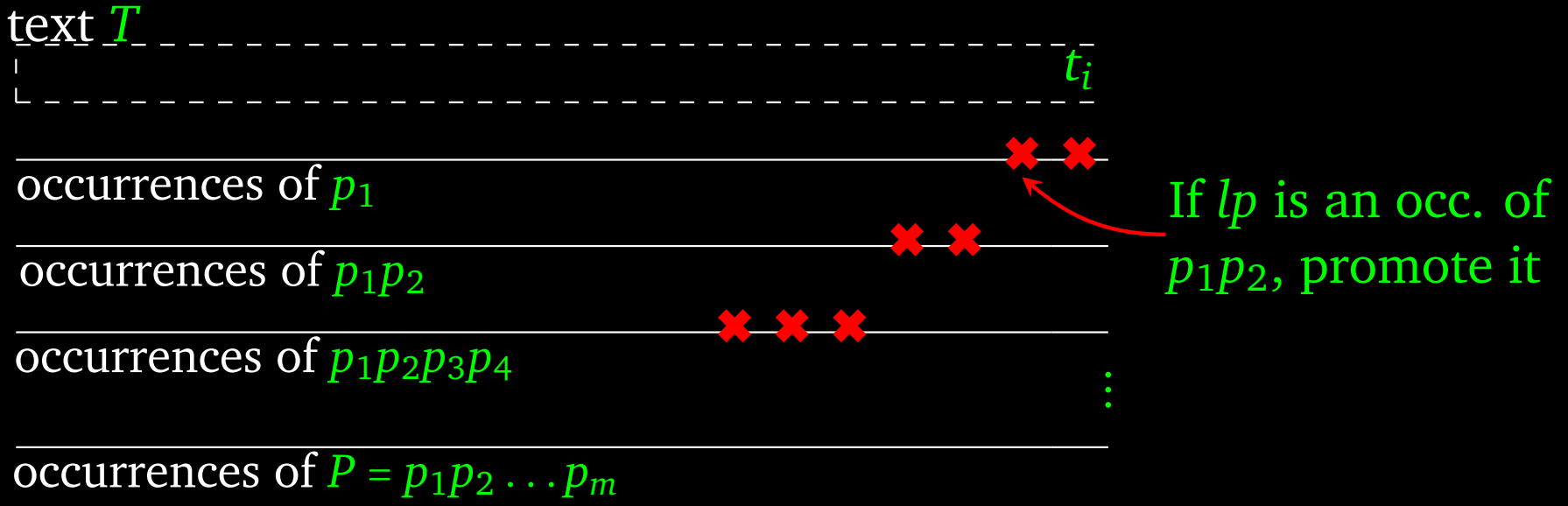
Porat & Porat, 2009 ★



If i is an occ. of p_1, push it to level 0

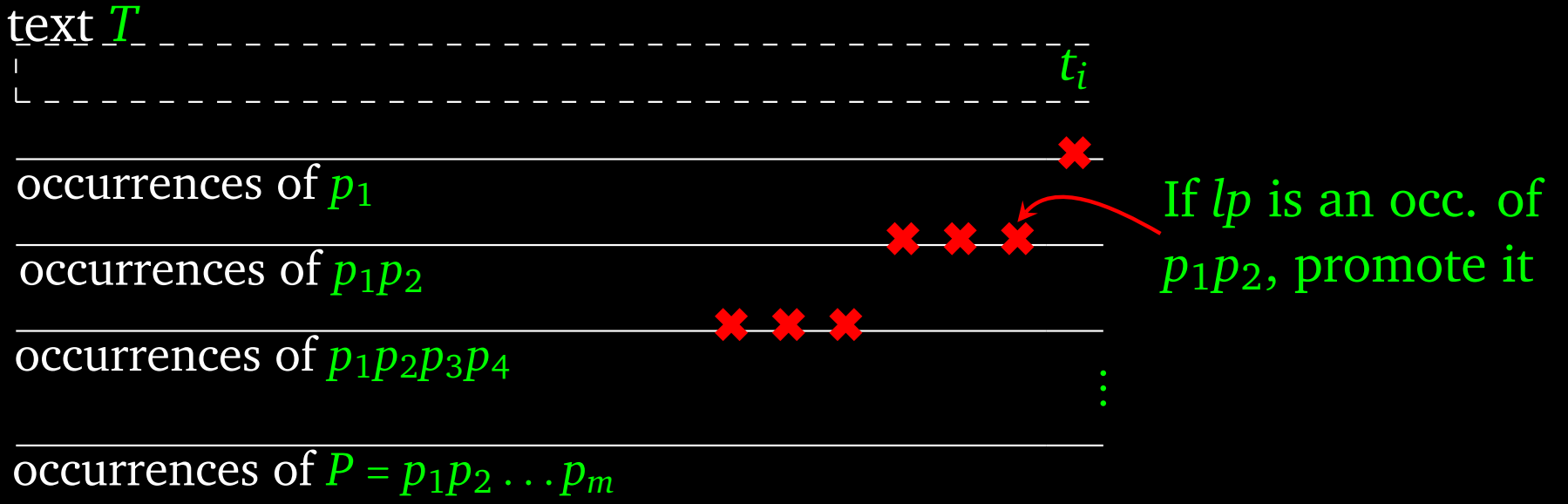
for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ **then**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

Porat & Porat, 2009 ★



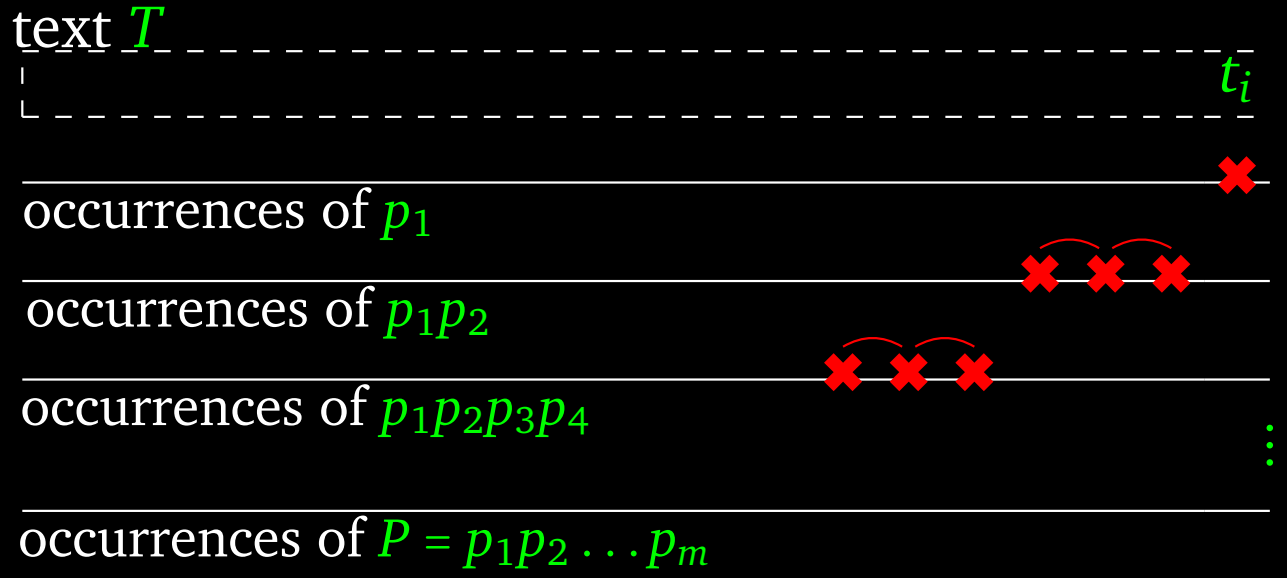
for each character t_i **do**
  if $t_i = p_1$ **then** push i to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level j
    if $i - lp + 1 = 2^{j+1}$ **then**
      Pop lp from level j
      if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ **then**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

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 $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ **then** push lp to level $j + 1$

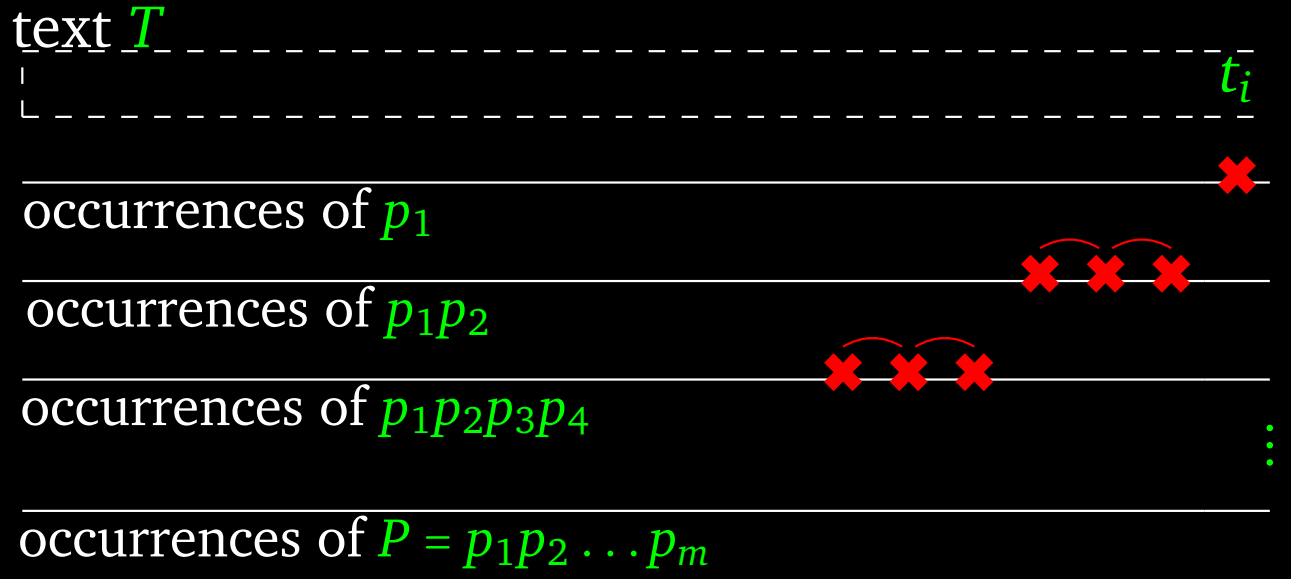
Porat & Porat, 2009 ★



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

- Number of occurrences
- The leftmost and the second leftmost positions lp, lp'
- The fingerprints $\varphi(t_1t_2 \dots t_{lp})$, $\varphi(t_{lp+1} \dots t_{lp'})$, $\varphi(t_1 \dots t_i)$

Porat & Porat, 2009 ★



For each level we need:

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

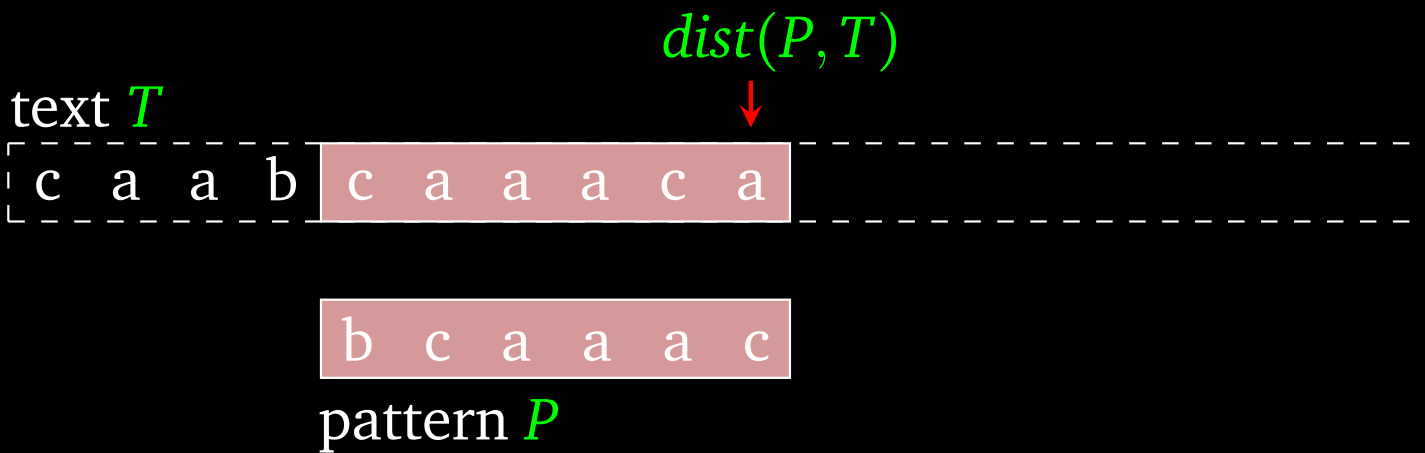
In total, the algorithm uses $O(\log m)$ space and $O(\log m)$ time per character

Part II: String similarity and approximate pattern matching

String similarity

Given two streams S_1, S_2 (S_1 arrives before S_2), compute the distance between them.

Approximate pattern matching



String similarity (Hamming distance)

Johnson & Lindenstrauss, 1984: one can compute $(1+\varepsilon)$-approximation of the Hamming distance between two streams using $\mathcal{O}(\varepsilon^{-2}\log n)$ space and $\mathcal{O}(\varepsilon^{-2}\log n)$ time per character.

Porat & Lipsky, 2007: one can decide if the Hamming distance between two streams is $\leq k$ using $\mathcal{O}(k\log n)$ space and $\mathcal{O}(\log n)$ time per character.

Approximate pattern matching (Hamming distance)

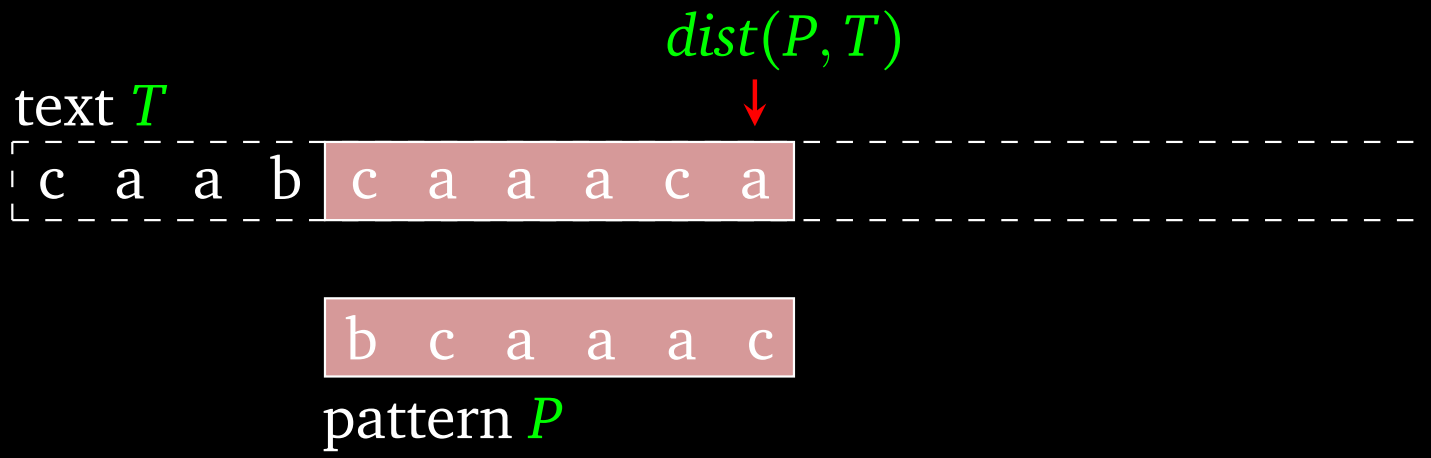
Authors	Space [2]	Time
Single pattern, only distances $\leq k$		
Porat & Porat, FOCS'09	$\tilde{O}(k^3)$	$\tilde{O}(k^2)$
Clifford, Fontaine, Porat, Sach, S., SODA'16	$\tilde{O}(k^2)$	$\tilde{O}(\sqrt{k})$
Golan, Kopelowitz, Porat, ICALP'18	$\tilde{O}(k)$	$\tilde{O}(k)$
Clifford, Kociumaka, Porat, SODA'19	$\tilde{O}(k)$	$\tilde{O}(\sqrt{k})$

Single pattern, $(1+\varepsilon)$-approx.		
Clifford, S., ICALP'16	$O(\varepsilon^{-5}\sqrt{m}\log^4 m)$	$O(\varepsilon^{-4}\log^3 m)$

Dictionary of d patterns, only distances $\leq k$		
Gawrychowski, S. (submitted)	$\tilde{O}(kd\log^k d)$	$\tilde{O}(k\log^k d + occ)$

[2] In words

Porat & Porat, FOCS'09



- If HAM(P,T) $> k$, output "NO"
- Otherwise, output HAM(P,T)
- There is a streaming algorithm that uses $\tilde{\mathcal{O}}(k^3)$ space and $\tilde{\mathcal{O}}(k^2)$ time per character of the text

From 1 mismatch to exact pattern matching

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

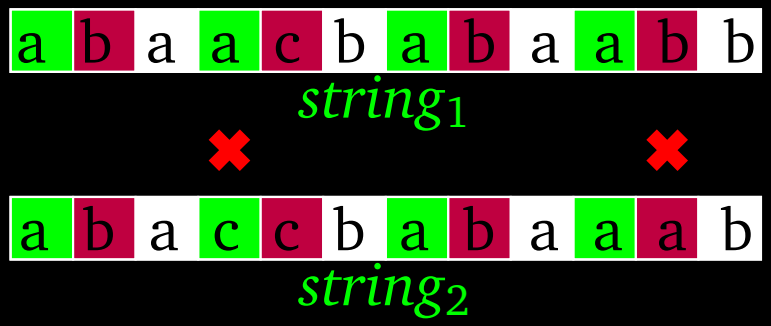
$string_1$

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

$string_2$

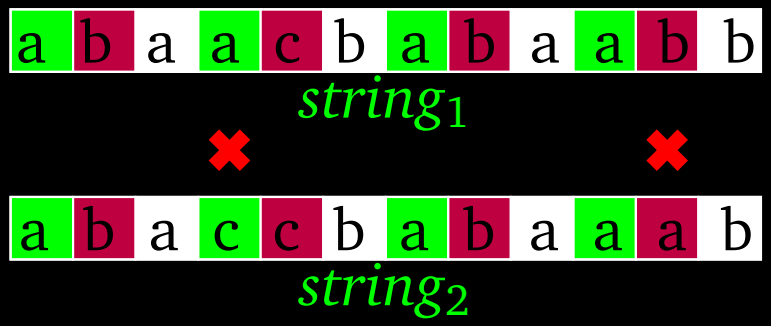
- Is HAM ($string_1$, $string_2$) = 1?

From 1 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 1 mismatch to exact pattern matching



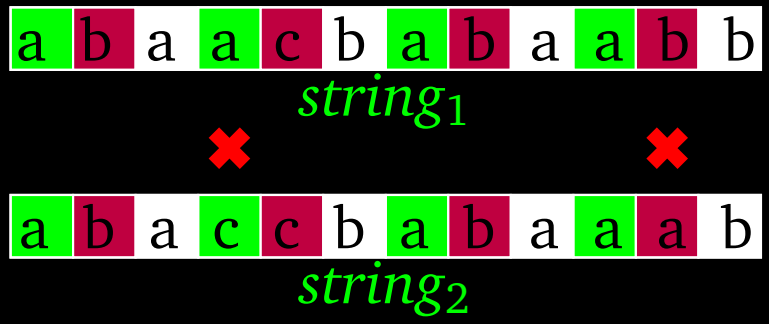
For each prime $q \in [\log m, \log^2 m]$:

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

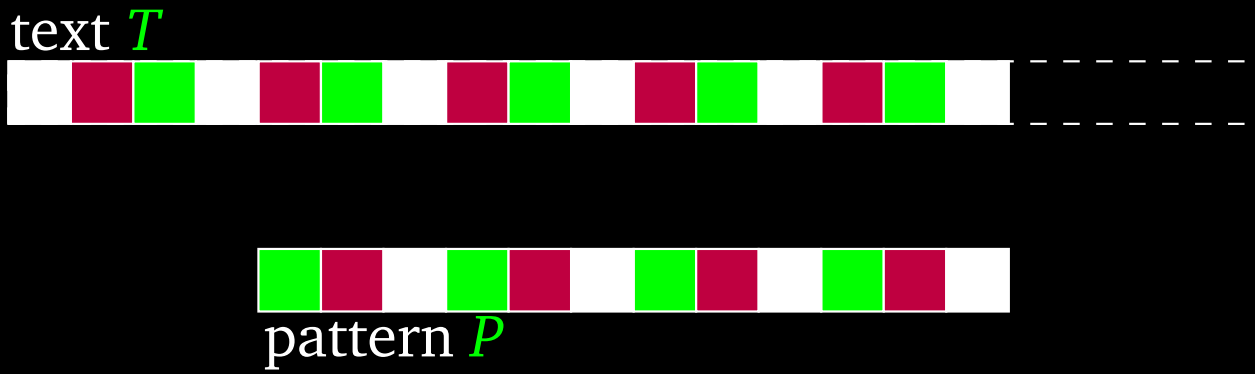
From 1 mismatch to exact pattern matching



Lemma There are ≥ 2 mismatches $\times_1, \times_2 \Rightarrow$ there exists a prime q such that at least two pairs of substrings do not match

- $\times_1, \times_2$ in the same pair $\Leftrightarrow \times_1 - \times_2 = 0 \pmod{q}$
- $m \geq \times_1 - \times_2$ cannot be a multiple of $\log m$ distinct primes

From 1 mismatch to exact pattern matching



Is HAM(P, T) = 1?

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

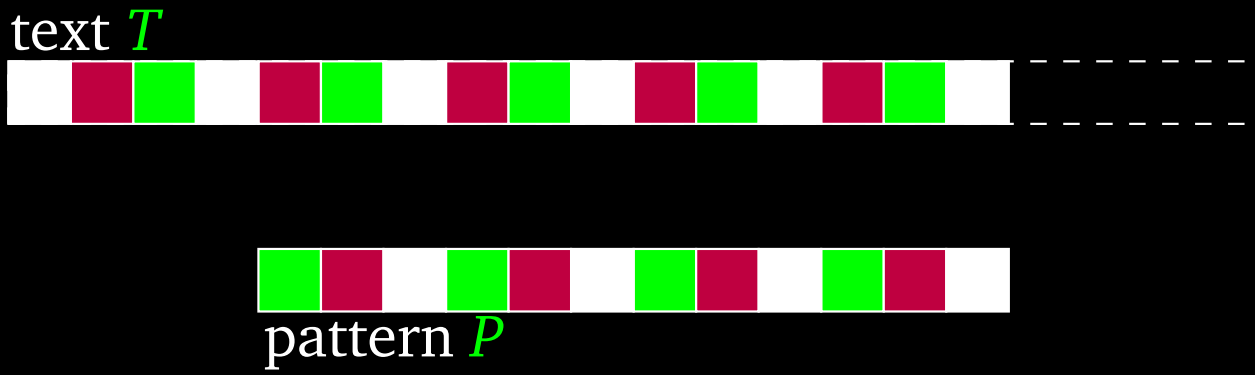
  for each prime q in $[\log m, \log^2 m]$ **do**

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

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

  return "YES"

From 1 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 1 mismatch to exact pattern matching

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

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

In general: $\tilde{O}(k^3)$ space, $\tilde{O}(k^2)$ time
(same as for $k = 1$ but take more subpatterns)

String similarity (edit distance)

Chakraborty, Goldenberg, Koucky, STOC'16: small distortion embedding from edit to Hamming distance

Belazzougui, Zhang, FOCS'16: embedding-based sketches for computing the edit distance exactly given that it is $\leq k$

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

	1	2	3		n
S :	**0**	1	0	…	0
$\mu(S)$:					

text position = 1, j = 1

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

	1	2	3		n
S :	**0**	1	0	…	0
$\mu(S)$:	0				

text position = 1, j = 1

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

	1	2	3		n
S :	**0**	1	0	…	0
$\mu(S)$:	0				

text position = 1, j = 1

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

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

text position = 1, j = 2

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

	1	2	3		n
S :	**0**	1	0	…	0
$\mu(S)$:	0	0			

text position = 1, j = 2

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

	1	2	3		n
S :	**0**	1	0	…	0
$\mu(S)$:	0	0			

text position = 1, j = 2

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

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

text position = 2, j = 3

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

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

text position = 2, j = 3

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

	1	2	3		n
S :	0	**1**	0	…	0
$\mu(S)$:	0	0	1		

text position = 2, j = 3

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

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

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

text position = 2, j = 3

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

Theorem. If $ED(S,T) = k$, then $k/2 \le HD(\mu(S), \mu(T)) \le \mathcal{O}(k^2)$ with probability 0.99.

Embedding from edit to Hamming distance

Pick $3m$ 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 S to $\mu(S)$:

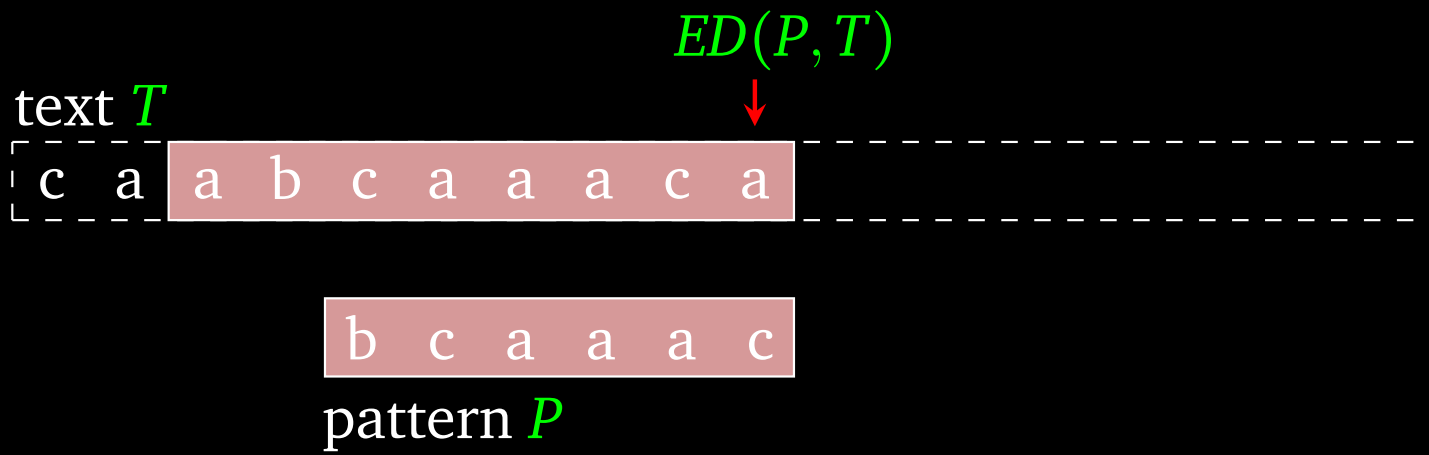
	1	2	3		n
S :	0	1	0	…	0
$\mu(S)$:	0	0	1	…	

text position = 2, j = 3

Belazzougui, Zhang, FOCS'16

- Embedding + streaming alg'm for k^2-mismatch $\Rightarrow$ a good estimate for edit distance
- If $ED(S,T) \le k$, $\tilde{O}(k^2)$ embeddings + streaming alg'm for k^2-mismatch $\Rightarrow$ **exact value!**

Approximate pattern matching (edit distance)

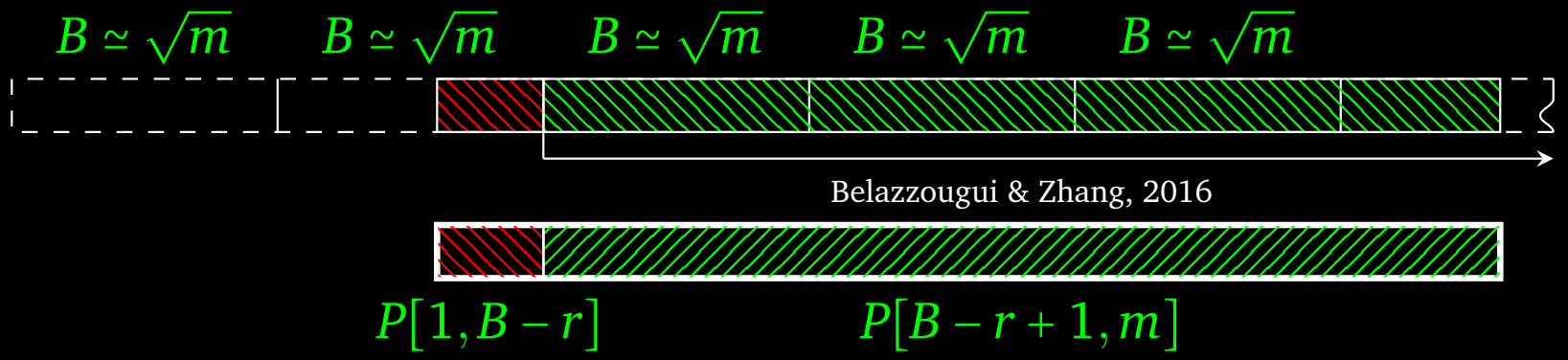


- If $ED(P,S) > k$, output "NO"
- Otherwise, output $ED(P,S)$

Algorithms:

- Hybrid dynamic programming: $\mathcal{O}(m)$ space, $\mathcal{O}(k)$ time
- S., 2017: $\mathcal{O}(\sqrt{m} \cdot poly(k, \log m))$ space, $\mathcal{O}(\sqrt{m} \cdot poly(k, \log m))$ time

Approximate pattern matching (edit distance)



Starting from each block i, run Belazzougui & Zhang, 2016

$$ED[j] = \min_{i \in [r-k, r+k]} ED\big(P[1, B-i], \mathbf{T_1}\big) + ED\big(P[B-i+1, m], \mathbf{T_2}\big)$$

We compute $ED\big(P[1, B-i], \mathbf{T_1}\big)$ while reading $\mathbf{T_1}$ using dynamic programming, then encode the distances to restore later

Part III: Periodicity

Periodicity

For each prefix of the input stream, compute its period

Motivation:

- Detecting anomalies in streams
- Preprocessing for pattern matching

Periodicity

Exact periods

Ergün et al., APPROX-RANDOM'10

- Periodic streams: $O(\log n)$ space, $O(\log n)$ time
- Non-periodic streams: $\Omega(m)$ space

Approximate periods (Hamming distance $\leq k$)

Ergün et al., APPROX-RANDOM'17

- Periods of length $< n/2$: $\tilde{O}(k^4)$ space
- All periods: $\Omega(n)$ space

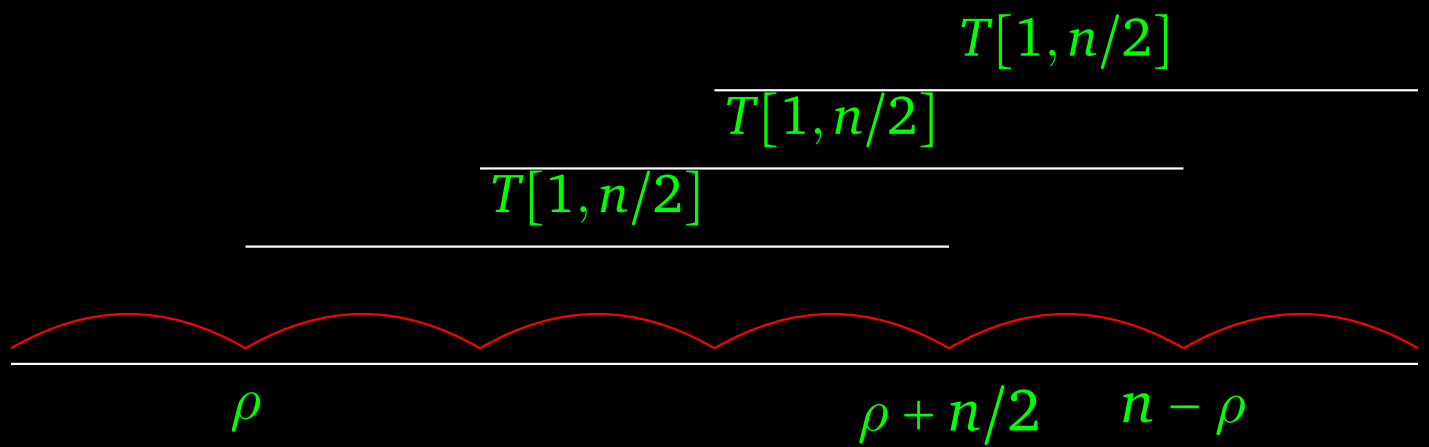
Approximate periods ($\leq k$ wildcards)

Ergün et al., CSR'18

- Periods of length $< n/2$: $\tilde{O}(k^3)$ space
- All periods: $\Omega(n)$ space

Exact periods

We will show how to compute the period if it is $\leq n/4$

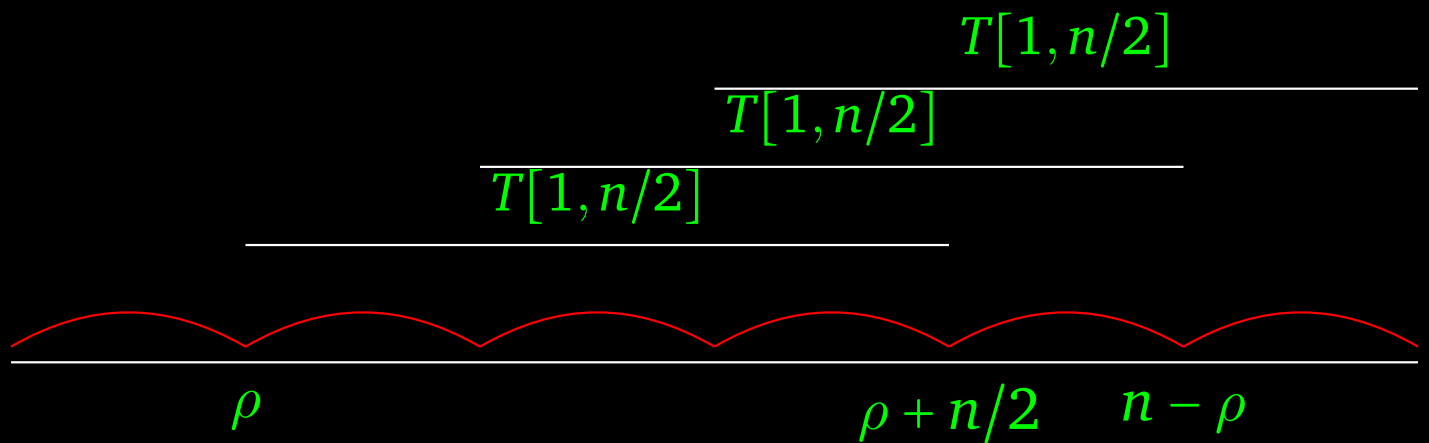


Lemma The period is equal to ρ iff $T[1, n - \rho] = T[\rho, n]$

Lemma The only candidate for the period, ρ is the the first occurrence of $T[1, n/2]$ in T

Exact periods

We will show how to compute the period if it is $\leq n/4$



Algorithm:

- Use the exact pattern matching algorithm to find the first occurrence ρ of $T[1, n/2]$ (happens when $T[\rho + n/2 - 1]$ arrives)
- Memorize $\varphi(T[1, n - \rho])$ (we have $n - \rho \geq \rho + n/2$)
- If $\varphi(T[1, n - \rho]) = \varphi(T[\rho, n])$, then ρ is the period w.h.p

Summary of today's talk

Streaming algorithms:

- String similarity for Hamming and edit distances
- Exact pattern matching — $\mathcal{O}(\log m)$ space, $\mathcal{O}(1)$ time
- k-mismatch (Hamming distance) — $\tilde{\mathcal{O}}(k)$ space, $\tilde{\mathcal{O}}(\sqrt{k})$ time
- k-mismatch (edit distance) — $\tilde{\mathcal{O}}(\sqrt{m}\operatorname{polylog} k)$ space, $\tilde{\mathcal{O}}(\sqrt{m}\operatorname{polylog} k)$ time
- Periodicity

Thank you!