Small-Space and Streaming Pattern Matching with k Edits

Tomasz Kociumaka Ely Porat Tatiana Starikovskaya

max planck institut informatik

Bar-Ilan University אוניברסיטת בר אילן

ENS | PSL

CPM 2022

June 27th, 2022

Pattern matching

Exact pattern matching

Given two strings, a **pattern** P of length m and a **text** T of length n, find all fragments of T **matching** P.

P `bbaabbb` T `abbaabbbaabbbbbbaabbbbaa`

Pattern matching

Exact pattern matching

Given two strings, a **pattern** P of length m and a **text** T of length n, find all fragments of T **matching** P.

P `b b a a b b b` T `a b b a a b b b a a b b b b b a a b b b b a a`

Algorithms: Knuth, Morris, Pratt
1978, SIAM J. Comput. $\mathcal{O}(n+m)$ time

Pattern matching with mismatches

Pattern matching with mismatches

Given a pattern P of length m, a text T of length n, and a **threshold** k, for each position $r \in [m \,..\, n]$, compute the **Hamming distance** $\mathrm{HD}(P, T(r-m \,..\, r])$ **if it does not exceed** k.

$\mathbf{k = 2}$

P `b b a a b b b`

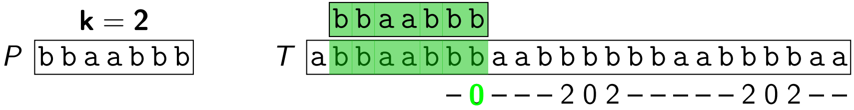
T `a b b a a b b b a a b b b b b b a a b b b b a a`

`– 0 – – – 2 0 2 – – – – – – 2 0 2 – –`

Pattern matching with mismatches

Pattern matching with mismatches

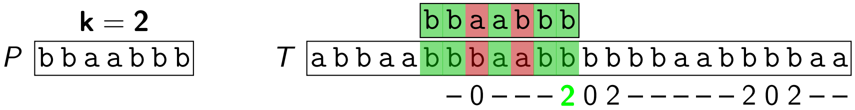
Given a pattern P of length m, a text T of length n, and a **threshold** k, for each position $r \in [m\,..\,n]$, compute the **Hamming distance** $\mathrm{HD}(P, T(r-m\,..\,r])$ **if it does not exceed** k.



Pattern matching with mismatches

Pattern matching with mismatches

Given a pattern P of length m, a text T of length n, and a **threshold** k, for each position $r \in [m\,..\,n]$, compute the **Hamming distance** $\mathrm{HD}(P, T(r-m\,..\,r])$ **if it does not exceed** k.



Pattern matching with mismatches

Given a pattern P of length m, a text T of length n, and a **threshold** k, for each position $r \in [m\,..\,n]$, compute the **Hamming distance** $\mathrm{HD}(P, T(r-m\,..\,r])$ **if it does not exceed** k.

$\mathbf{k} = \mathbf{2}$

P `b b a a b b b`

`b b a a b b b`

T `a b b a a b b b a a b b b b b a a b b b b a a`

– 0 – – – 2 0 2 – – – – – 2 0 2 – –

Pattern matching with mismatches

Pattern matching with mismatches

Given a pattern P of length m, a text T of length n, and a **threshold** k, for each position $r \in [m \,..\, n]$, compute the **Hamming distance** $\mathrm{HD}(P, T(r-m \,..\, r])$ **if it does not exceed** k.

$$\mathbf{k = 2}$$

P `bbaabbb` T `abbaabbbaabbbbbbaabbbbaa`

`      – 0 – – – 2 0 2 – – – – – – 2 0 2 – –`

⋮   ⋮

Algorithms:

Gawrychowski, Uznański
ICALP 2018

$\widetilde{\mathcal{O}}(n + nk/\sqrt{m})$ time

Pattern matching with edits

Pattern matching with edits

Given a pattern P of length m, a text T of length n, and a threshold k, for each position $r \in [1\,..\,n]$, compute the **edit distance** $\min_{\ell \in [1..r]} \mathrm{ED}(P, T(\ell\,..\,r])$ if it does not exceed k.

$\mathbf{k = 2}$

P `b b a a b b b`

T `a b b a a b b b a a b b b b b a a b b b b a a`

– – – – – 2 1 0 1 2 2 1 0 1 2 2 – – 2 1 0 1 2 –

Pattern matching with edits

Pattern matching with edits

Given a pattern P of length m, a text T of length n, and a threshold k, for each position $r \in [1\,..\,n]$, compute the **edit distance** $\min_{\ell \in [1..r]} \mathrm{ED}(P, T(\ell\,..\,r])$ if it does not exceed k.

$\mathbf{k = 2}$

P: b b a a b b b

b b a a b b b

T: a b b a a b b b a a b b b b b a a b b b b a a

– – – – – 2 1 0 1 2 2 1 0 1 2 2 – – 2 1 0 1 2 –

Pattern matching with edits

Given a pattern P of length m, a text T of length n, and a threshold k, for each position $r \in [1\,..\,n]$, compute the **edit distance** $\min_{\ell \in [1\,..\,r]} \mathrm{ED}(P, T(\ell\,..\,r])$ if it does not exceed k.

k = 2

P = b b a a b b b

b b a a b b b

T = a b b a a b b b a a b b b b b b a a b b b b a a

– – – – – 2 1 0 1 2 2 1 0 1 2 2 – – 2 1 0 1 2 –

Pattern matching with edits

Pattern matching with edits

Given a pattern P of length m, a text T of length n, and a threshold k, for each position $r \in [1\,..\,n]$, compute the **edit distance** $\min_{\ell \in [1\,..\,r]} \mathrm{ED}(P, T(\ell\,..\,r])$ if it does not exceed k.

$\mathbf{k = 2}$

P = b b a a b b b

b b a a b b b

T = a b b a a b **b b a a b b b** b b b a a b b b b a a

– – – – – 2 1 0 1 2 2 1 **0** 1 2 2 – – 2 1 0 1 2 –

Pattern matching with edits

Pattern matching with edits

Given a pattern P of length m, a text T of length n, and a threshold k, for each position $r \in [1\,..\,n]$, compute the **edit distance** $\min_{\ell \in [1\,..\,r]} \mathrm{ED}(P, T(\ell\,..\,r])$ if it does not exceed k.

$\mathbf{k = 2}$

P `b b a a b b b`

`b b a a b b b`

T `a b b a a b b b a a b b b b b b a a b b b b a a`

`– – – – – 2 1 0 1 2 2 1 0 1 2 2 – – 2 1 0 1 2 –`

Pattern matching with edits

Pattern matching with edits

Given a pattern P of length m, a text T of length n, and a threshold k, for each position $r \in [1\,..\,n]$, compute the **edit distance** $\min_{\ell \in [1\,..\,r]} \mathrm{ED}(P, T(\ell\,..\,r])$ if it does not exceed k.

k = 2

P `b b a a b b b`

T `a b b a a b b b a a b b b b b b a a b b b b a a`

`– – – – – – 2 1 0 1 2 2 1 0 1 2 2 – – 2 1 0 1 2 –`

Algorithms:

Landau, Vishkin
1989, J. Algorithms — $\mathcal{O}(nk)$ time

Cole, Hariharan
2002, SIAM J. Comput. — $\mathcal{O}(n + \frac{nk^4}{m})$ time

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.

P `bbaabbb` T

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.

P `bbaabbb` T `abbaabb`

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.

P `bbaabbb` T `abbaabb`

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.

P `bbaabbb` T `abbaabb`

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.

P `bbaabbb`

T `abbaabbb`

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.

P `bbaabbb` T `abbaabbba`

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.

P `bbaabbb` T `abbaabbbaa`

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.
- Main efficiency measure: size of the working space.

P | b b a a b b b | T | a b b a a b b b a a b b b b b a a b b b b a a |

Streaming pattern matching

Streaming model:

- Single sequential scan of the text T.
- The answer regarding r to be reported while processing $T[r]$.
- Main efficiency measure: size of the working space.
- Deterministic and Las-Vegas algorithms require $\Omega(m \log \sigma)$ bits for exact matching.

P `bbaabbb` T `abbaabbbaabbbbbaabbbbaa`

Streaming pattern matching algorithms

Exact Pattern Matching: $\Omega(\log m)$ bits

Porat, Porat $\mathcal{O}(\log^2 m)$ bits

FOCS 2009

Streaming pattern matching algorithms

Exact Pattern Matching:	$\Omega(\log m)$ bits
Porat, Porat FOCS 2009	$\mathcal{O}(\log^2 m)$ bits
Pattern Matching with Mismatches:	$\Omega(k \log m)$ bits
Porat, Porat FOCS 2009	$\widetilde{\mathcal{O}}(k^3)$ bits
Clifford et al. SODA 2016	$\widetilde{\mathcal{O}}(k^2)$ bits
Golan, Kopelowitz, Porat ICALP 2018	$\widetilde{\mathcal{O}}(k)$ bits
Clifford, K., Porat SODA 2019	$\mathcal{O}(k \log^2 m)$ bits

Streaming pattern matching algorithms

Exact Pattern Matching:	$\Omega(\log m)$ bits
Porat, Porat FOCS 2009	$\mathcal{O}(\log^2 m)$ bits
Pattern Matching with Mismatches:	$\Omega(k \log m)$ bits
Porat, Porat FOCS 2009	$\widetilde{\mathcal{O}}(k^3)$ bits
Clifford et al. SODA 2016	$\widetilde{\mathcal{O}}(k^2)$ bits
Golan, Kopelowitz, Porat ICALP 2018	$\widetilde{\mathcal{O}}(k)$ bits
Clifford, K., Porat SODA 2019	$\mathcal{O}(k \log^2 m)$ bits
Pattern Matching with Edits:	$\Omega(k \log m)$ bits
Starikovskaya CPM 2017	$\widetilde{\mathcal{O}}(k^8 \sqrt{m})$ bits
K., Porat, Starikovskaya (this work)	$\widetilde{\mathcal{O}}(k^5)$ bits

Sketches

$X \in \Sigma^{\leq n}$

$\mathrm{ED}(X, Y)$

$Y \in \Sigma^{\leq n}$

Sketches

$X \in \Sigma^{\leq n}$

encoding

$\mathrm{sk}(X)$

$\mathrm{ED}(X, Y)$

$Y \in \Sigma^{\leq n}$

Sketches

$X \in \Sigma^{\leq n}$

↓ encoding

$\mathrm{sk}(X)$

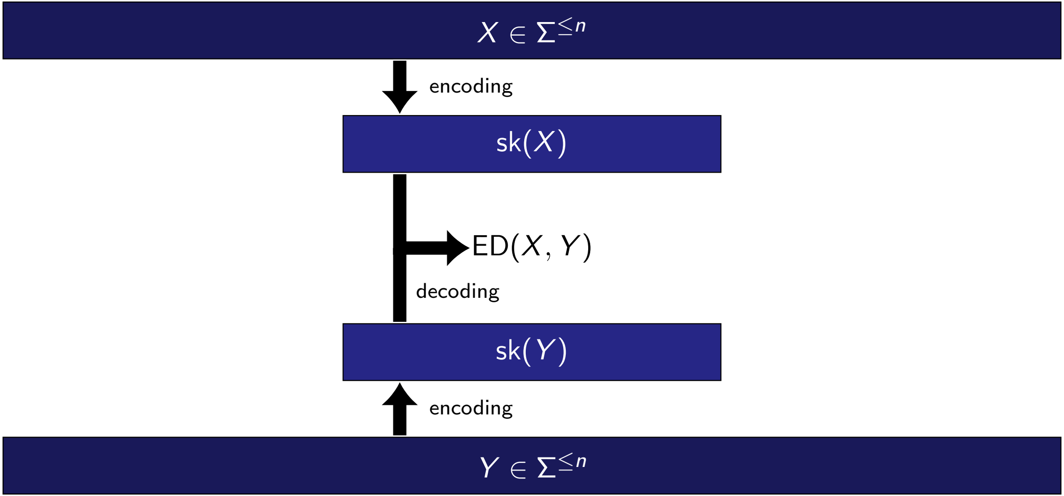
$\mathrm{ED}(X, Y)$

$\mathrm{sk}(Y)$

↑ encoding

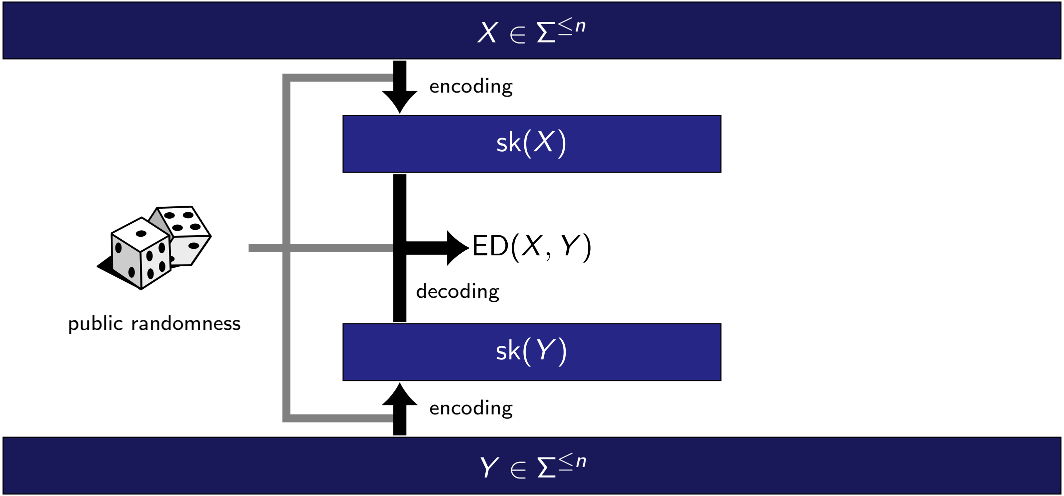
$Y \in \Sigma^{\leq n}$

Sketches



$X \in \Sigma^{\leq n}$
encoding
sk(X)
ED(X, Y)
decoding
sk(Y)
encoding
$Y \in \Sigma^{\leq n}$

Sketches



$X \in \Sigma^{\leq n}$
encoding
$\mathsf{sk}(X)$
$\mathsf{ED}(X, Y)$
decoding
public randomness
$\mathsf{sk}(Y)$
encoding
$Y \in \Sigma^{\leq n}$

Known sketches

Efficiently constructible sketches with error probability $n^{-\Theta(1)}$:

Testing equality $X = Y$ (fingerprints):	$\Omega(\log n)$ bits
folklore	$\mathcal{O}(\log n)$ bits

Known sketches

Efficiently constructible sketches with error probability $n^{-\Theta(1)}$:

Testing equality $X = Y$ (fingerprints):	$\Omega(\log n)$ bits
folklore	$\mathcal{O}(\log n)$ bits
Computing $\mathrm{HD}(X, Y)$ if at most k:	$\Omega(k \log n)$ bits
Lipsky, Porat CPM 2007	$\mathcal{O}(k \log n)$ bits

Known sketches

Efficiently constructible sketches with error probability $n^{-\Theta(1)}$:

Testing equality $X = Y$ (fingerprints):	$\Omega(\log n)$ bits
folklore	$\mathcal{O}(\log n)$ bits
Computing HD(X, Y) if at most k:	$\Omega(k \log n)$ bits
Lipsky, Porat CPM 2007	$\mathcal{O}(k \log n)$ bits
Computing ED(X, Y) if at most k:	$\Omega(k \log n)$ bits
Belazzougui, Zhang FOCS 2016	$\widetilde{\mathcal{O}}(k^8)$ bits
Jin, Nelson, Wu STACS 2021	$\widetilde{\mathcal{O}}(k^3)$ bits
K., Porat, Starikovskaya (this work)	$\widetilde{\mathcal{O}}(k^2)$ bits

Outline of the talk

Introduction

Streaming exact pattern matching

Streaming pattern matching with edits

Conclusions and open problems

Outline of the talk

Introduction

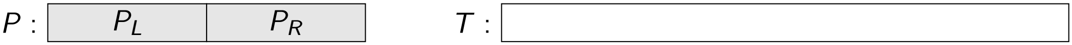
Streaming exact pattern matching

Streaming pattern matching with edits

Conclusions and open problems

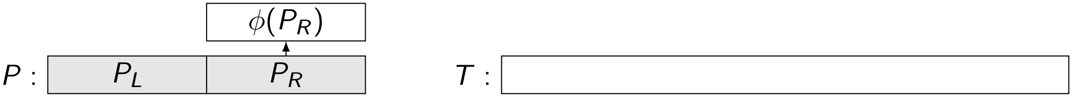
Exact streaming pattern matching: High-level idea

1. Decompose P into two halves P_L and P_R.



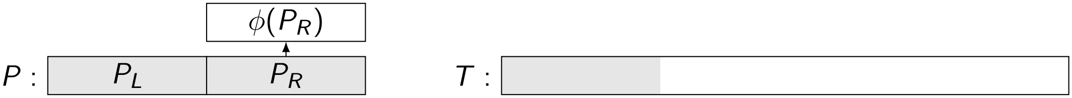
Exact streaming pattern matching: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **fingerprint** $\phi(P_R)$.



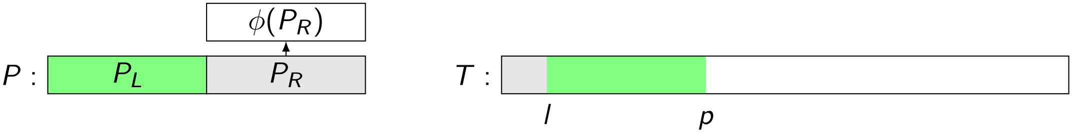
Exact streaming pattern matching: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **fingerprint** $\phi(P_R)$.
3. Recursively look for the occurrences of P_L.



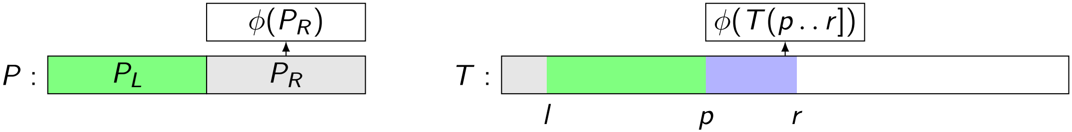
Exact streaming pattern matching: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **fingerprint** $\phi(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] = P_L$ to an occurrence of P:



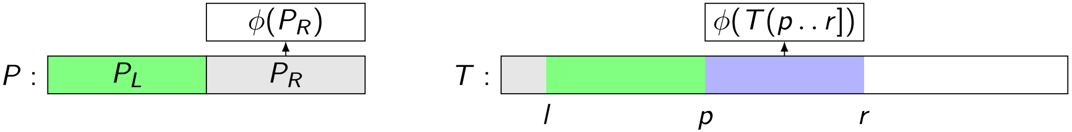
Exact streaming pattern matching: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **fingerprint** $\phi(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] = P_L$ to an occurrence of P:
   - Maintain a **fingerprint** $\phi(T(p\,..\,r])$.



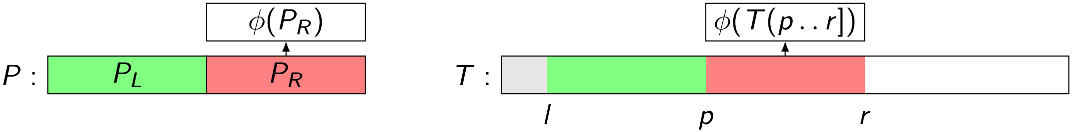
Exact streaming pattern matching: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **fingerprint** $\phi(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] = P_L$ to an occurrence of P:
   - Maintain a **fingerprint** $\phi(T(p\,..\,r])$.
   - Once at $r = p + |P_R|$, compare $\phi(T(p\,..\,r])$ with $\phi(P_R)$.



Exact streaming pattern matching: High-level idea

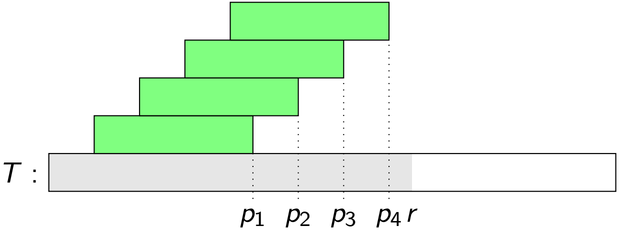
1. Decompose P into two halves P_L and P_R.
2. Precompute a **fingerprint** $\phi(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] = P_L$ to an occurrence of P:
   - Maintain a **fingerprint** $\phi(T(p\,..\,r])$.
   - Once at $r = p + |P_R|$, compare $\phi(T(p\,..\,r])$ with $\phi(P_R)$.



Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

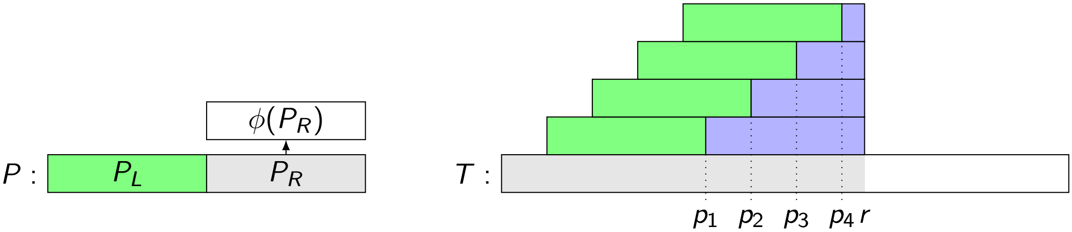


Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.

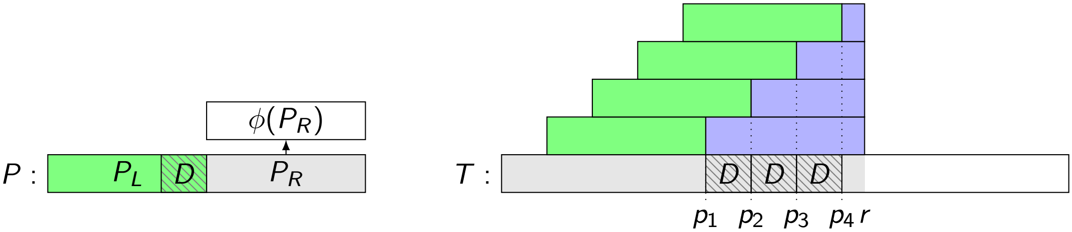


Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.

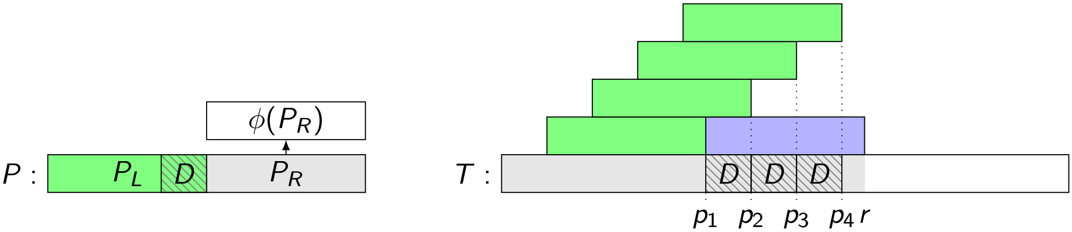


Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only

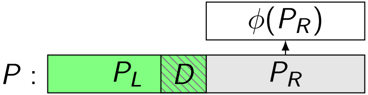


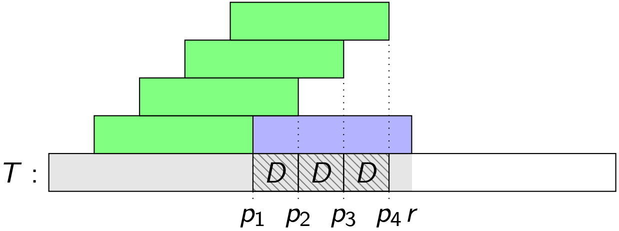
Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only and, after processing $T(l_1\,..\,p_1]$



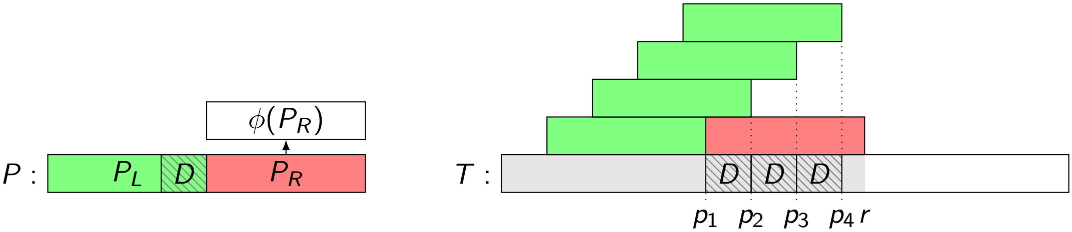


Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only and, after processing $T(l_1\,..\,p_1]$

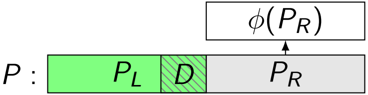


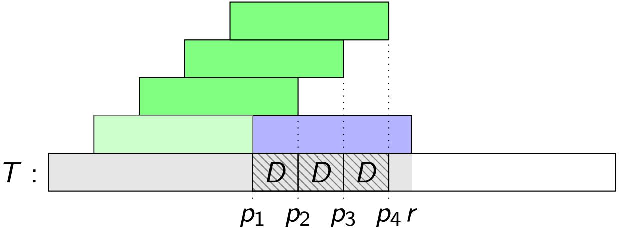
Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only and, after processing $T(l_1\,..\,p_1]$



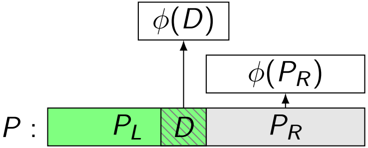


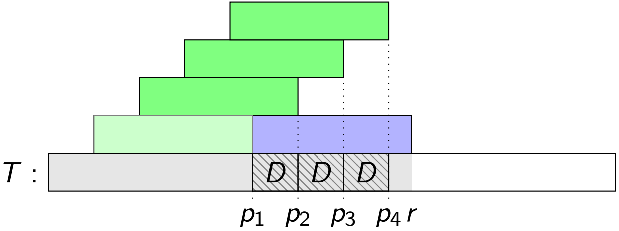
Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only and, after processing $T(l_1\,..\,p_1]$, use $\phi(D) = \phi(T(p_1\,..\,p_2])$



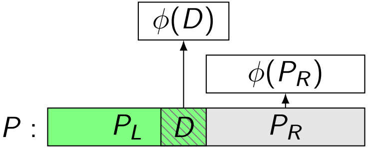


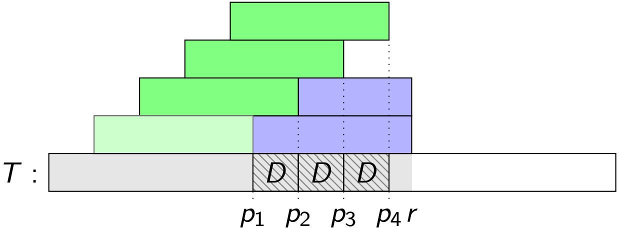
Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only and, after processing $T(l_1\,..\,p_1]$, use $\phi(D) = \phi(T(p_1\,..\,p_2])$ to derive $\phi(T(p_2\,..\,r])$.



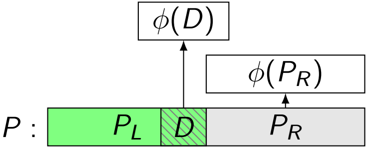


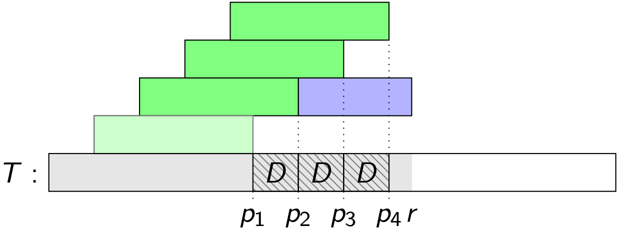
Exact streaming pattern matching: Active occurrences

Active occurrences of P_L

An occurrence $P_L = T(l\,..\,p]$ is **active** if $p \in [r - |P_R|\,..\,r]$.

- For each active occurrence $T(l_i\,..\,p_i]$, we need to maintain $\phi(T(p_i\,..\,r])$.
- The active occurrences form a **chain** with a fixed **difference** $D = T(p_{i-1}\,..\,p_i]$.
- We maintain $\phi(T(p_1\,..\,r])$ only and, after processing $T(l_1\,..\,p_1]$, use $\phi(D) = \phi(T(p_1\,..\,p_2])$ to derive $\phi(T(p_2\,..\,r])$.





Outline of the talk

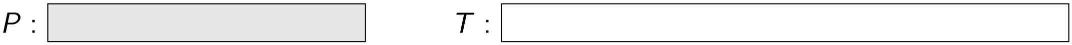
Introduction

Streaming exact pattern matching

Streaming pattern matching with edits

Conclusions and open problems

Streaming pattern matching with edits: High-level idea

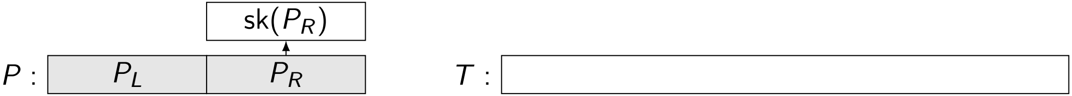


Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.

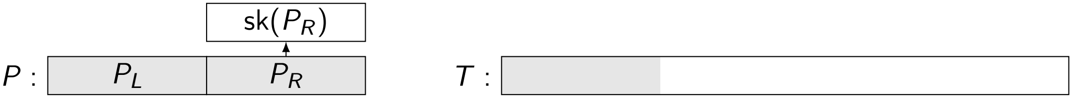
Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **sketch** $\mathsf{sk}(P_R)$.



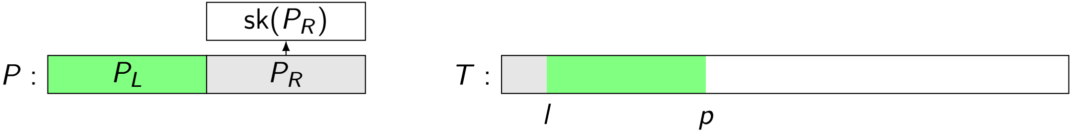
Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **sketch** $\mathrm{sk}(P_R)$.
3. Recursively look for the occurrences of P_L.



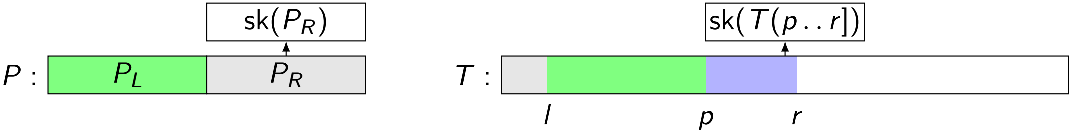
Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **sketch** $\mathrm{sk}(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] \approx P_L$ to an occurrence of P:



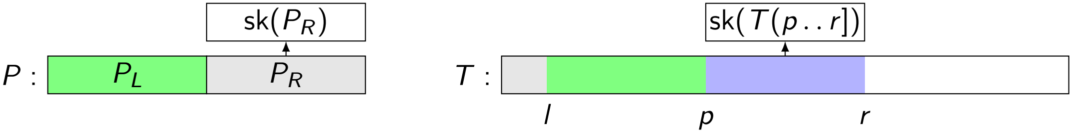
Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **sketch** $\text{sk}(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] \approx P_L$ to an occurrence of P:
   - Maintain a **sketch** $\text{sk}(T(p\,..\,r])$.



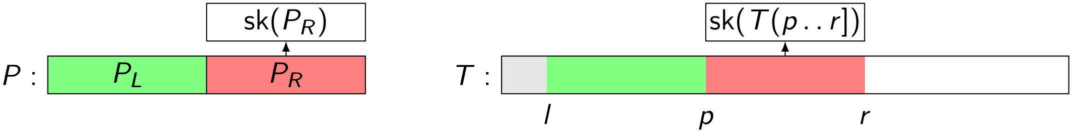
Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **sketch** $\mathrm{sk}(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] \approx P_L$ to an occurrence of P:
   - Maintain a **sketch** $\mathrm{sk}(T(p\,..\,r])$.
   - Once at $r \in [p + |P_R| - k\,..\,p + |P_R| + k]$, compare $\mathrm{sk}(T(p\,..\,r])$ with $\mathrm{sk}(P_R)$.



Streaming pattern matching with edits: High-level idea

1. Decompose P into two halves P_L and P_R.
2. Precompute a **sketch** $\mathrm{sk}(P_R)$.
3. Recursively look for the occurrences of P_L.
4. Try extending each occurrence $T(l\,..\,p] \approx P_L$ to an occurrence of P:
   - Maintain a **sketch** $\mathrm{sk}(T(p\,..\,r])$.
   - Once at $r \in [p+|P_R|-k\,..\,p+|P_R|+k]$, compare $\mathrm{sk}(T(p\,..\,r])$ with $\mathrm{sk}(P_R)$.

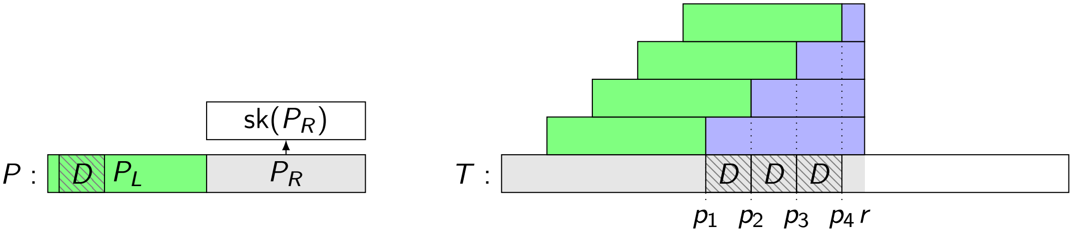


Streaming pattern matching with edits: Active occurrences

Corollary (of Charalampopoulos, K., Wellnitz; FOCS 2020)

Active k-edit occurrences of P_L form $\mathcal{O}(k^3)$ chains whose difference D is among $\mathcal{O}(k)$ prescribed substrings of P_L.

- For each active occurrence $T(l_i \mathinner{.\,.} p_i]$, we need to maintain $\mathrm{sk}(T(p_i \mathinner{.\,.} r])$.

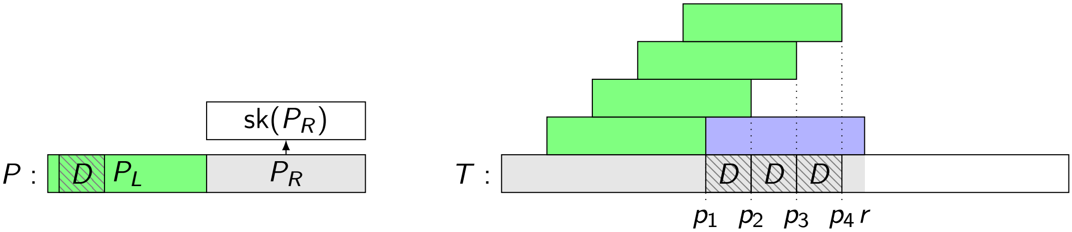


Streaming pattern matching with edits: Active occurrences

Corollary (of Charalampopoulos, K., Wellnitz; FOCS 2020)

Active k-edit occurrences of P_L form $\mathcal{O}(k^3)$ chains whose difference D is among $\mathcal{O}(k)$ prescribed substrings of P_L.

- For each active occurrence $T(l_i \mathinner{.\,.} p_i]$, we need to maintain $\mathrm{sk}(T(p_i \mathinner{.\,.} r])$.
- We maintain $\mathrm{sk}(T(p_1 \mathinner{.\,.} r])$ only

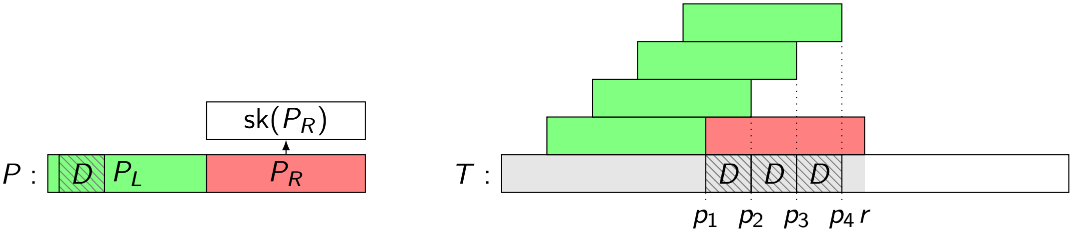


Streaming pattern matching with edits: Active occurrences

Corollary (of Charalampopoulos, K., Wellnitz; FOCS 2020)

Active k-edit occurrences of P_L form $\mathcal{O}(k^3)$ chains whose difference D is among $\mathcal{O}(k)$ prescribed substrings of P_L.

- For each active occurrence $T(l_i \mathinner{.\,.} p_i]$, we need to maintain $\mathrm{sk}(T(p_i \mathinner{.\,.} r])$.
- We maintain $\mathrm{sk}(T(p_1 \mathinner{.\,.} r])$ only and, after processing $T(l_1 \mathinner{.\,.} p_1]$

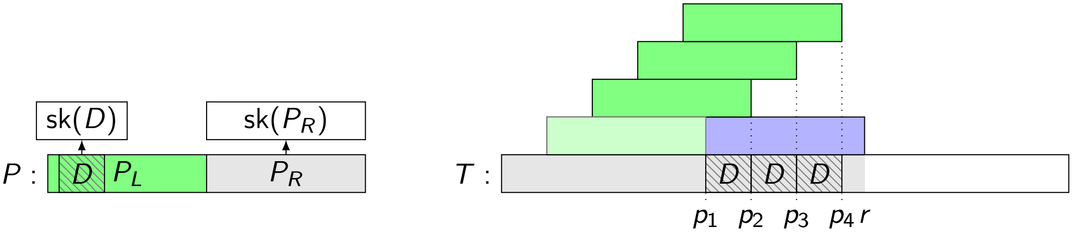


Streaming pattern matching with edits: Active occurrences

Corollary (of Charalampopoulos, K., Wellnitz; FOCS 2020)

Active k-edit occurrences of P_L form $\mathcal{O}(k^3)$ chains whose difference D is among $\mathcal{O}(k)$ prescribed substrings of P_L.

- For each active occurrence $T(l_i \mathinner{..} p_i]$, we need to maintain $\mathsf{sk}(T(p_i \mathinner{..} r])$.
- We maintain $\mathsf{sk}(T(p_1 \mathinner{..} r])$ only and, after processing $T(l_1 \mathinner{..} p_1]$, use $\mathsf{sk}(D) = \mathsf{sk}(T(p_1 \mathinner{..} p_2])$

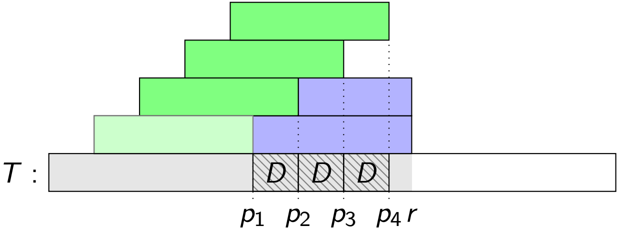


Streaming pattern matching with edits: Active occurrences

Corollary (of Charalampopoulos, K., Wellnitz; FOCS 2020)

Active k-edit occurrences of P_L form $\mathcal{O}(k^3)$ chains whose difference D is among $\mathcal{O}(k)$ prescribed substrings of P_L.

- For each active occurrence $T(l_i \mathinner{.\,.} p_i]$, we need to maintain $\mathrm{sk}(T(p_i \mathinner{.\,.} r])$.
- We maintain $\mathrm{sk}(T(p_1 \mathinner{.\,.} r])$ only and, after processing $T(l_1 \mathinner{.\,.} p_1]$, use $\mathrm{sk}(D) = \mathrm{sk}(T(p_1 \mathinner{.\,.} p_2])$ to derive $\mathrm{sk}(T(p_2 \mathinner{.\,.} r])$.

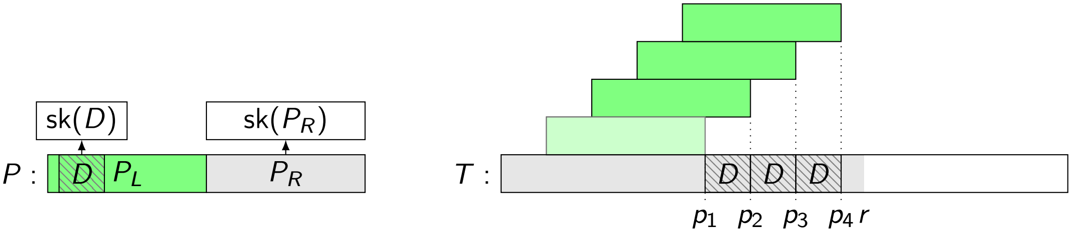


Streaming pattern matching with edits: Active occurrences

Corollary (of Charalampopoulos, K., Wellnitz; FOCS 2020)

Active k-edit occurrences of P_L form $\mathcal{O}(k^3)$ chains whose difference D is among $\mathcal{O}(k)$ prescribed substrings of P_L.

- For each active occurrence $T(l_i \,..\, p_i]$, we need to maintain $\mathrm{sk}(T(p_i \,..\, r])$.
- We maintain $\mathrm{sk}(T(p_1 \,..\, r])$ only and, after processing $T(l_1 \,..\, p_1]$, ~~use $\mathrm{sk}(D) = \mathrm{sk}(T(p_1 \,..\, p_2])$ to derive $\mathrm{sk}(T(p_2 \,..\, r])$.~~ Infeasible for edit distance sketches!

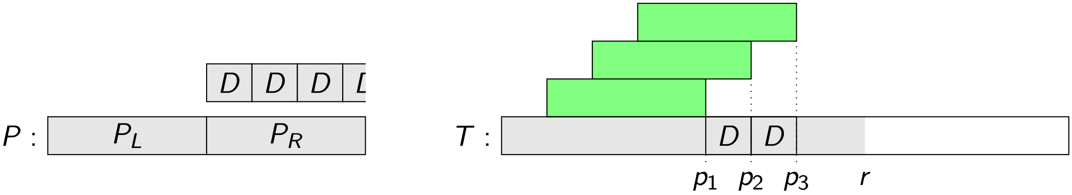


How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\mathinner{.\,.} t]) = \mathcal{O}(k)$ for all feasible chain differences D.

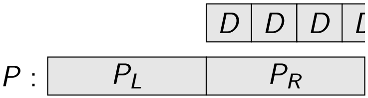
How to circumvent the lack of composable sketches?

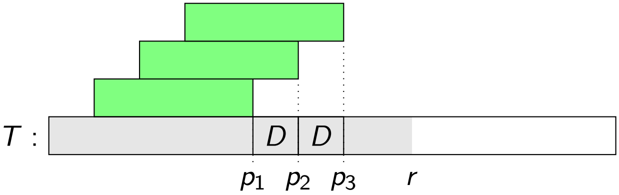
1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,..\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.



How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,.\,.\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:

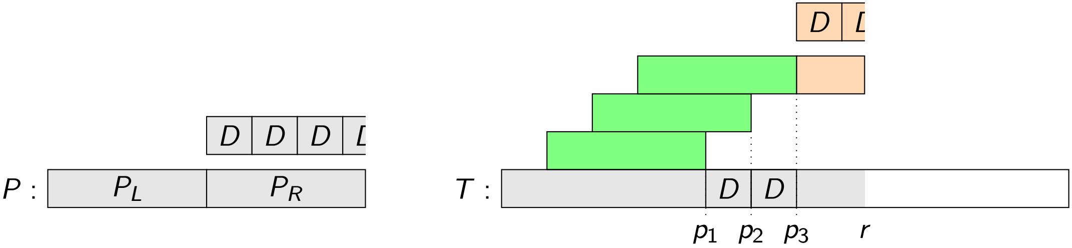




How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,..\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:
   - $\mathrm{sk}(X), \mathrm{sk}(Y) \rightsquigarrow \mathcal{E}(X, Y)$

$\mathcal{E}(T(p_c\,..\,r], D^\infty(0\,..\,t - c|D|])$

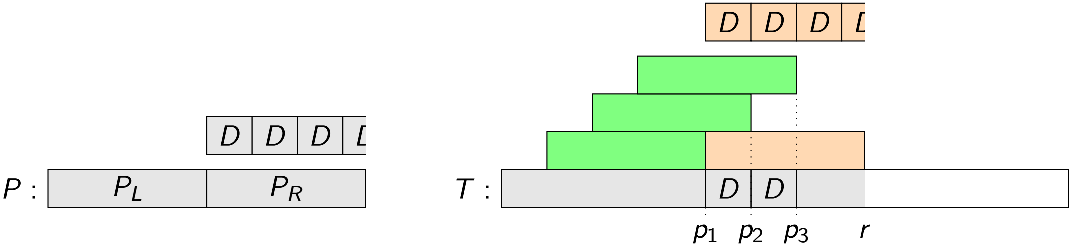


How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,..\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:
   - $\mathrm{sk}(X), \mathrm{sk}(Y) \rightsquigarrow \mathcal{E}(X, Y)$
   - $\mathcal{E}(X, Y), \mathcal{E}(\hat{X}, \hat{Y}) \rightsquigarrow \mathcal{E}(X\hat{X}, Y\hat{Y})$

$\mathcal{E}(T(p_c\,..\,r], D^\infty(0\,..\,t - c|D|])$

$\mathcal{E}(T(p_1\,..\,r], D^\infty(0\,..\,t])$



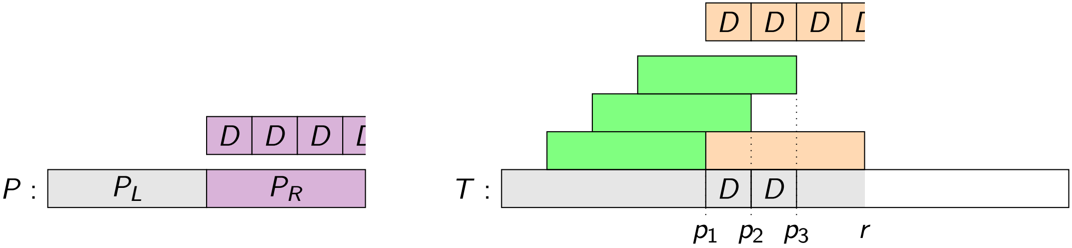
How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,..\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:
   - $\mathrm{sk}(X), \mathrm{sk}(Y) \rightsquigarrow \mathcal{E}(X, Y)$
   - $\mathcal{E}(X, Y), \mathcal{E}(\hat{X}, \hat{Y}) \rightsquigarrow \mathcal{E}(X\hat{X}, Y\hat{Y})$
   - $\mathcal{E}(X, Y), \mathcal{E}(Y, Z) \rightsquigarrow \mathcal{E}(X, Z)$

$\mathcal{E}(T(p_c\,..\,r], D^\infty(0\,..\,t - c|D|])$

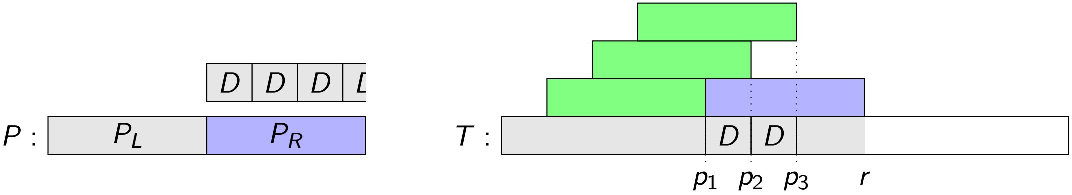
$\mathcal{E}(T(p_1\,..\,r], D^\infty(0\,..\,t])$

$\mathcal{E}(P_R, T(p_1\,..\,r])$



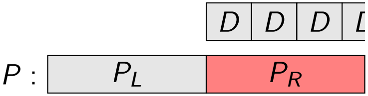
How to circumvent the lack of composable sketches?

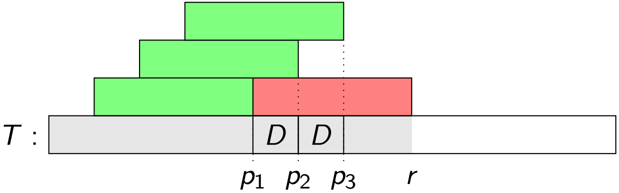
1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,..\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:
   - $\mathrm{sk}(X), \mathrm{sk}(Y) \rightsquigarrow \mathcal{E}(X, Y)$ — $\mathcal{E}(T(p_c\,..\,r], D^\infty(0\,..\,t-c|D|])$
   - $\mathcal{E}(X, Y), \mathcal{E}(\hat{X}, \hat{Y}) \rightsquigarrow \mathcal{E}(X\hat{X}, Y\hat{Y})$ — $\mathcal{E}(T(p_1\,..\,r], D^\infty(0\,..\,t])$
   - $\mathcal{E}(X, Y), \mathcal{E}(Y, Z) \rightsquigarrow \mathcal{E}(X, Z)$ — $\mathcal{E}(P_R, T(p_1\,..\,r])$



How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\mathinner{.\,.} t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:
   - $\mathrm{sk}(X), \mathrm{sk}(Y) \leadsto \mathcal{E}(X, Y)$ $\quad$ $\mathcal{E}(T(p_c\mathinner{.\,.} r], D^\infty(0\mathinner{.\,.} t-c|D|])$
   - $\mathcal{E}(X, Y), \mathcal{E}(\hat{X}, \hat{Y}) \leadsto \mathcal{E}(X\hat{X}, Y\hat{Y})$ $\quad$ $\mathcal{E}(T(p_1\mathinner{.\,.} r], D^\infty(0\mathinner{.\,.} t])$
   - $\mathcal{E}(X, Y), \mathcal{E}(Y, Z) \leadsto \mathcal{E}(X, Z)$ $\quad$ $\mathcal{E}(P_R, T(p_1\mathinner{.\,.} r])$
   - $\mathcal{E}(X, Y) \leadsto \mathrm{ED}(X, Y)$ $\quad$ $\mathrm{ED}(T(p_1\mathinner{.\,.} r], P_R)$





How to circumvent the lack of composable sketches?

1. A careful decomposition $P = P_L P_R$ allows assuming that:
   - P_L has $\mathcal{O}(k^2)$ active occurrences, or
   - $\mathrm{ED}(P_R, D^\infty(0\,..\,t]) = \mathcal{O}(k)$ for all feasible chain differences D.
2. We design an **encoding** $\mathcal{E}(X, Y)$ for strings X, Y at $\mathrm{ED}(X, Y) = \mathcal{O}(k)$ so that:
   - $\mathrm{sk}(X), \mathrm{sk}(Y) \rightsquigarrow \mathcal{E}(X, Y)$ — $\mathcal{E}(T(p_c\,..\,r], D^\infty(0\,..\,t - c|D|])$
   - $\mathcal{E}(X, Y), \mathcal{E}(\hat{X}, \hat{Y}) \rightsquigarrow \mathcal{E}(X\hat{X}, Y\hat{Y})$ — $\mathcal{E}(T(p_1\,..\,r], D^\infty(0\,..\,t])$
   - $\mathcal{E}(X, Y), \mathcal{E}(Y, Z) \rightsquigarrow \mathcal{E}(X, Z)$ — $\mathcal{E}(P_R, T(p_1\,..\,r])$
   - $\mathcal{E}(X, Y) \rightsquigarrow \mathrm{ED}(X, Y)$ — $\mathrm{ED}(T(p_1\,..\,r], P_R)$
3. Techniques behind the encoding $\mathcal{E}(X, Y)$:
   - Distinguish **greedy** edit-distance alignments between X, Y;
   - Observe that any two greedy alignments diverge within few **compressible** regions of X, Y.

Outline of the talk

Introduction

Streaming exact pattern matching

Streaming pattern matching with edits

Conclusions and open problems

Conclusions and open problems

Our main result

The pattern matching with edits problem can be solved using an $\widetilde{\mathcal{O}}(k^5)$-space streaming algorithm that costs $\widetilde{\mathcal{O}}(k^8)$ amortized time per character and outputs answers correct w.h.p.

Conclusions and open problems

Our main result

The pattern matching with edits problem can be solved using an $\widetilde{\mathcal{O}}(k^5)$-space streaming algorithm that costs $\widetilde{\mathcal{O}}(k^8)$ amortized time per character and outputs answers correct w.h.p.

Directions for future work:

- Real-time processing of the text.
- Efficient (streaming) preprocessing of the pattern.
- Improved polynomial dependency on k.
- Lower bounds (so far, we know that $\widetilde{\Omega}(k)$ space is necessary).

Conclusions and open problems

Our main result

The pattern matching with edits problem can be solved using an $\widetilde{\mathcal{O}}(k^5)$-space streaming algorithm that costs $\widetilde{\mathcal{O}}(k^8)$ amortized time per character and outputs answers correct w.h.p.

Directions for future work:

- Real-time processing of the text.
- Efficient (streaming) preprocessing of the pattern.
- Improved polynomial dependency on k.
- Lower bounds (so far, we know that $\widetilde{\Omega}(k)$ space is necessary).

Thank you for your attention!