Minimal Suffix and Rotation of a Substring in Optimal Time

Tomasz Kociumaka

University of Warsaw, Poland

CPM 2016
Tel-Aviv, Israel
June 29, 2016

Minimal & Maximal Suffix and Rotation Queries

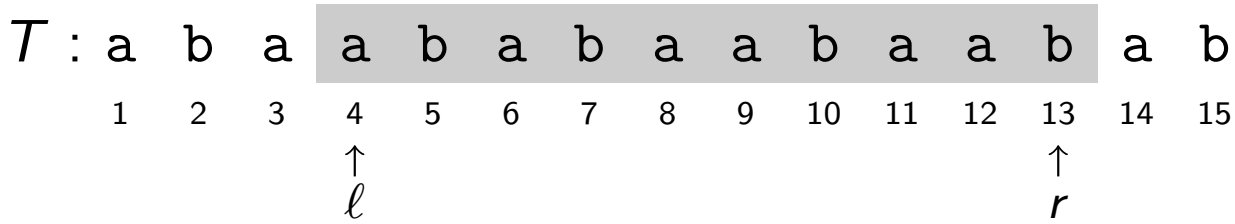
Preprocess a text T of length n.

T :	a	b	a	a	b	a	b	a	a	b	a	a	b	a	b
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Minimal & Maximal Suffix and Rotation Queries

Preprocess a text T of length n.

Given indices ℓ, r compute:

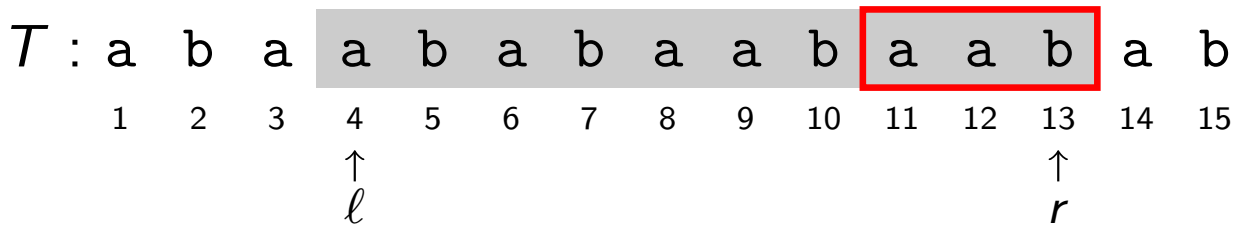


Minimal & Maximal Suffix and Rotation Queries

Preprocess a text T of length n.

Given indices ℓ, r compute:

- the lexicographically smallest non-empty suffix of $T[\ell..r]$,



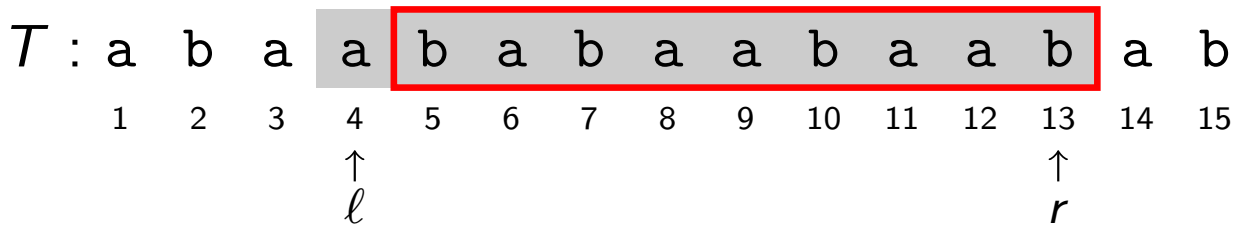
$\mathrm{MinSuf}(T[4..13]) = \mathtt{aab} = T[11..13]$

Minimal & Maximal Suffix and Rotation Queries

Preprocess a text T of length n.

Given indices ℓ, r compute:

- the lexicographically smallest non-empty suffix of $T[\ell..r]$,
- the lexicographically largest suffix of $T[\ell..r]$,



$\mathrm{MinSuf}(T[4..13]) = \texttt{aab} = T[11..13]$

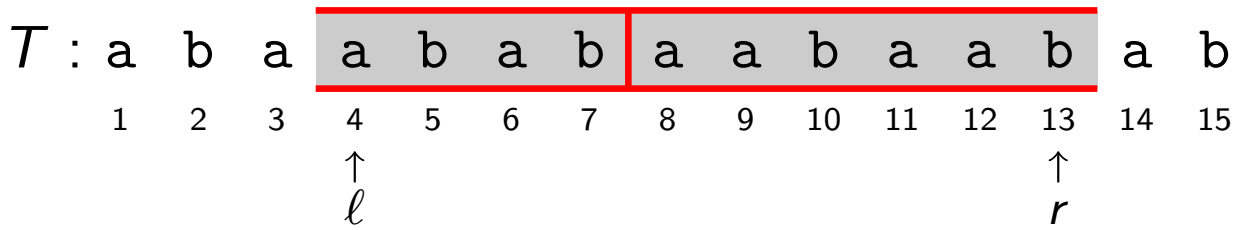
$\mathrm{MaxSuf}(T[4..13]) = \texttt{babaabaab} = T[5..13]$

Minimal & Maximal Suffix and Rotation Queries

Preprocess a text T of length n.

Given indices ℓ, r compute:

- the lexicographically smallest non-empty suffix of $T[\ell..r]$,
- the lexicographically largest suffix of $T[\ell..r]$,
- the lexicographically smallest cyclic rotation of $T[\ell..r]$.



$\mathrm{MinSuf}(T[4..13]) = \mathtt{aab} = T[11..13]$

$\mathrm{MaxSuf}(T[4..13]) = \mathtt{babaabaab} = T[5..13]$

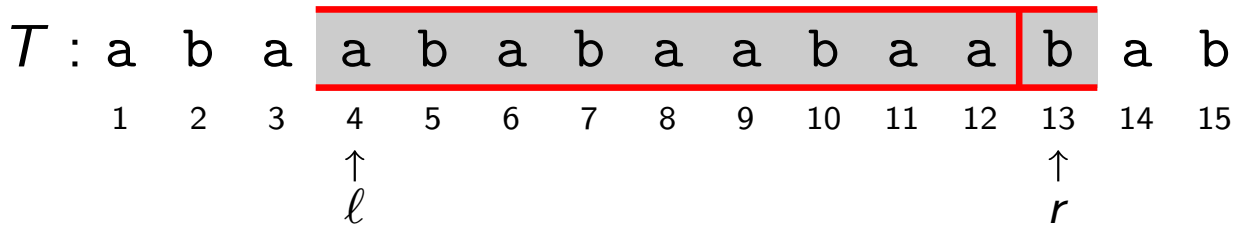
$\mathrm{MinRot}(T[4..13]) = \mathtt{aabaababab} = T[8..13]\,T[4..7]$

Minimal & Maximal Suffix and Rotation Queries

Preprocess a text T of length n.

Given indices ℓ, r compute:

- the lexicographically smallest non-empty suffix of $T[\ell..r]$,
- the lexicographically largest suffix of $T[\ell..r]$,
- the lexicographically smallest cyclic rotation of $T[\ell..r]$.
- the lexicographically largest cyclic rotation of $T[\ell..r]$.



$\mathrm{MinSuf}(T[4..13]) = \texttt{aab} = T[11..13]$

$\mathrm{MaxSuf}(T[4..13]) = \texttt{babaabaab} = T[5..13]$

$\mathrm{MinRot}(T[4..13]) = \texttt{aabaababab} = T[8..13]\,T[4..7]$

$\mathrm{MaxRot}(T[4..13]) = \texttt{bababaabaa} = T[13..13]\,T[4..12]$

Previous & our results

Problem	Preprocessing	Query	Reference

Previous & our results

Problem	Preprocessing	Query	Reference
MinSuf	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log^{1+\varepsilon} n)$	BKS, CPM'13
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	

Previous & our results

Problem	Preprocessing	Query	Reference
MinSuf	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log^{1+\varepsilon} n)$	BKS, CPM'13
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	
MinSuf	$\mathcal{O}(n \log n/\tau)$	$\mathcal{O}(\tau)$	BGKS, CPM'14
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(1)$	

Previous & our results

Problem	Preprocessing	Query	Reference
MinSuf	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log^{1+\varepsilon} n)$	BKS, CPM'13
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	
MinSuf	$\mathcal{O}(n \log n/\tau)$	$\mathcal{O}(\tau)$	BGKS, CPM'14
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(1)$	
Cyclic eq.	$\mathcal{O}(n)$ exp.	$\mathcal{O}(1)$	KRRW, SODA'15

Previous & our results

Problem	Preprocessing	Query	Reference
MinSuf	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log^{1+\varepsilon} n)$	BKS, CPM'13
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	
MinSuf	$\mathcal{O}(n \log n/\tau)$	$\mathcal{O}(\tau)$	BGKS, CPM'14
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(1)$	
Cyclic eq.	$\mathcal{O}(n)$ exp.	$\mathcal{O}(1)$	KRRW, SODA'15
MinSuf	$\mathcal{O}(n)$	$\mathcal{O}(1)$	this work
MinRot	$\mathcal{O}(n)$	$\mathcal{O}(1)$	
MaxRot	$\mathcal{O}(n)$	$\mathcal{O}(1)$	

Previous & our results

Problem	Preprocessing	Query	Reference
MinSuf	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log^{1+\varepsilon} n)$	BKS, CPM'13
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	
MinSuf	$\mathcal{O}(n \log n/\tau)$	$\mathcal{O}(\tau)$	BGKS, CPM'14
MaxSuf	$\mathcal{O}(n)$	$\mathcal{O}(1)$	
Cyclic eq.	$\mathcal{O}(n)$ exp.	$\mathcal{O}(1)$	KRRW, SODA'15
MinSuf	$\mathcal{O}(n)$	$\mathcal{O}(1)$	this work
MinRot	$\mathcal{O}(n)$	$\mathcal{O}(1)$	
MaxRot	$\mathcal{O}(n)$	$\mathcal{O}(1)$	

Common features:

- $\mathcal{O}(n)$ space,
- word RAM model with words of $\Omega(\log n)$ bits,
- integer alphabet $\{1, \ldots, n^{\mathcal{O}(1)}\}$.

Generalized minimal suffix queries

Preprocess a text T of length n.

T :	a	b	a	a	b	a	b	a	a	b	a	a	b	a	b
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Generalized minimal suffix queries

Preprocess a text T of length n.

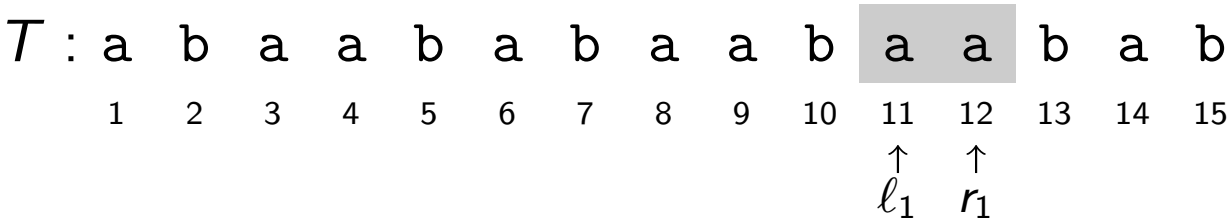
Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$

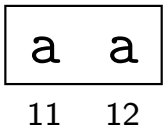
T :	a	b	a	a	b	a	b	a	a	b	a	a	b	a	b
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Generalized minimal suffix queries

Preprocess a text T of length n.

Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$

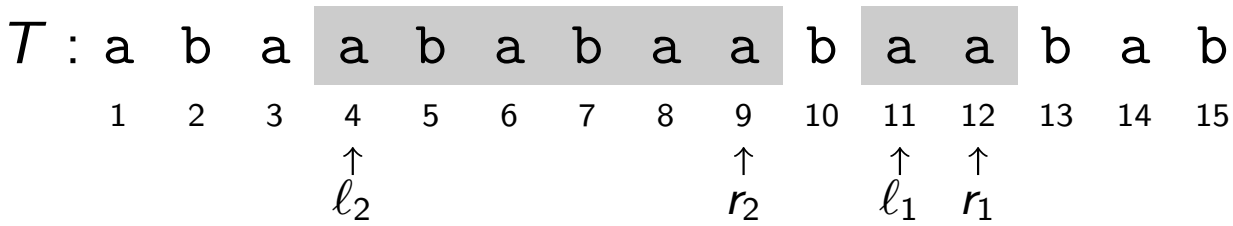


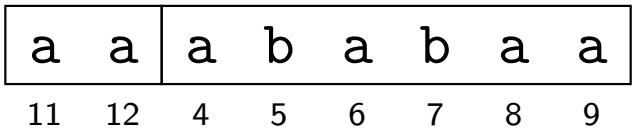


Generalized minimal suffix queries

Preprocess a text T of length n.

Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$

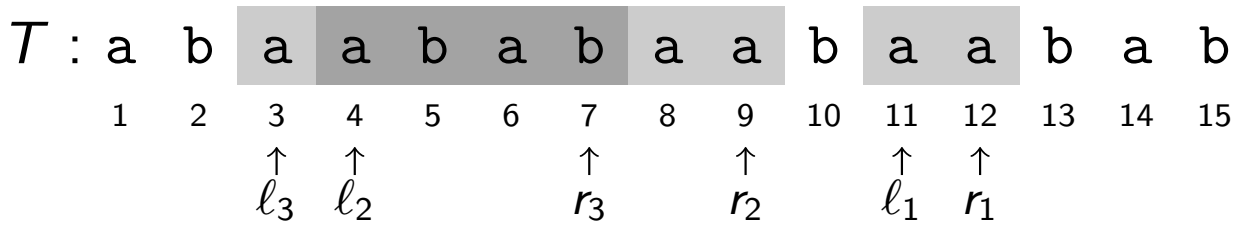




Generalized minimal suffix queries

Preprocess a text T of length n.

Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$

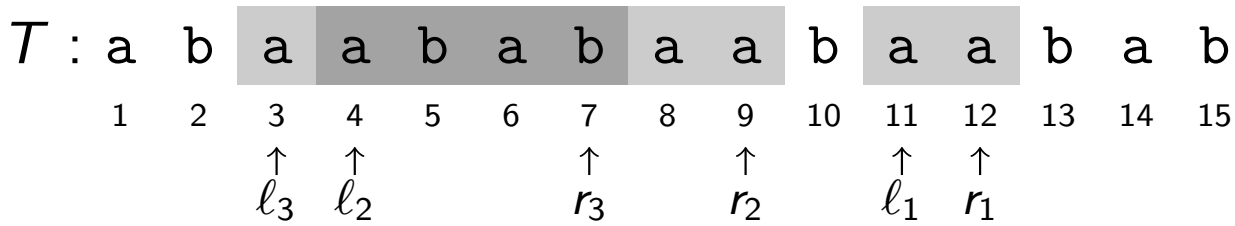


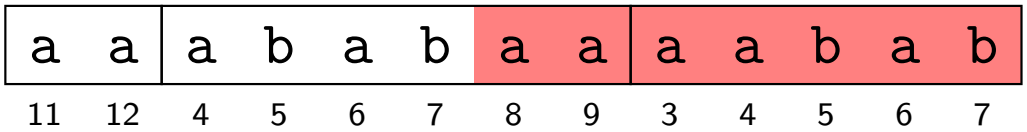
a	a	a	b	a	b	a	a	a	a	b	a	b
11	12	4	5	6	7	8	9	3	4	5	6	7

Generalized minimal suffix queries

Preprocess a text T of length n.

Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$



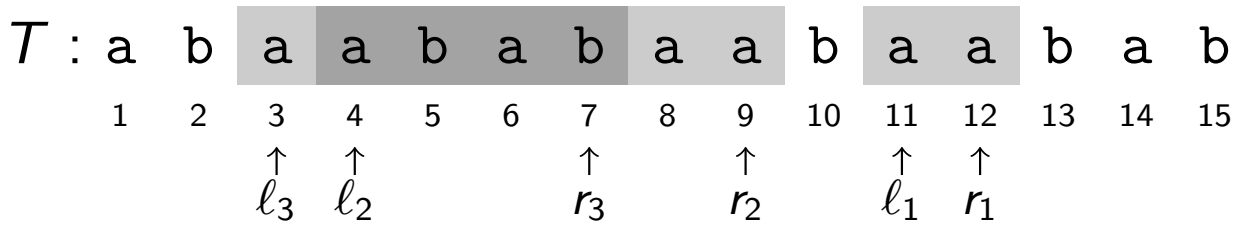


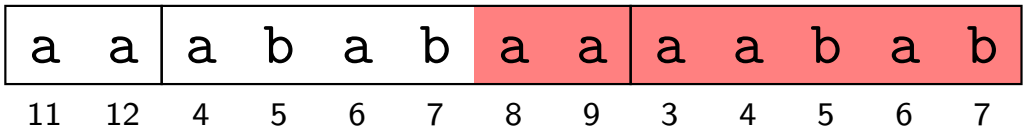
$\mathrm{MinSuf}(T[11..12]\,T[4..9]\,T[3..7]) = \texttt{aaaabab} = T[8..9]\,T[3..7]$

Generalized minimal suffix queries

Preprocess a text T of length n.

Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$





$\mathrm{MinSuf}(T[11..12]\,T[4..9]\,T[3..7]) = \texttt{aaaabab} = T[8..9]\,T[3..7]$

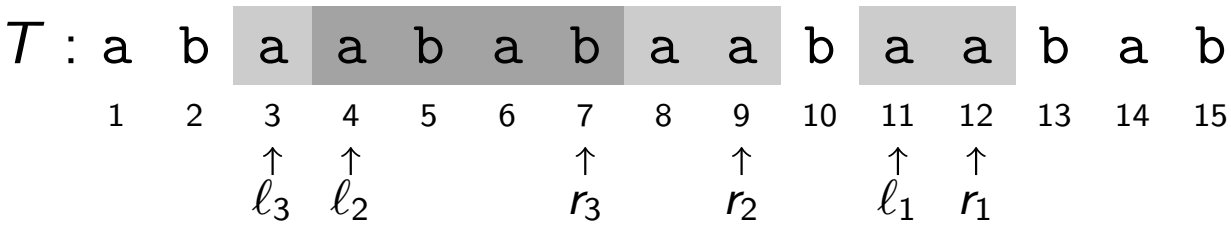
Our result:

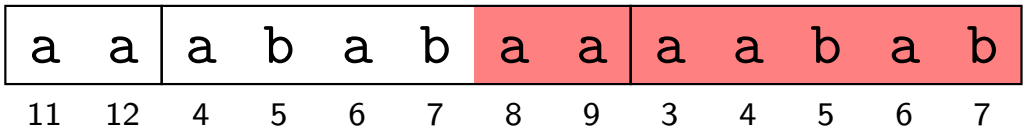
- $\mathcal{O}(k^2)$ query time,
- $\mathcal{O}(n)$ preprocessing time.

Generalized minimal suffix queries

Preprocess a text T of length n.

Given indices $\ell_1, r_1, \ldots, \ell_k, r_k$, compute the lexicographically smallest non-empty suffix of $T[\ell_1..r_1] \cdots T[\ell_k..r_k]$





$\mathrm{MinSuf}(T[11..12]\,T[4..9]\,T[3..7]) = \texttt{aaaabab} = T[8..9]\,T[3..7]$

Our result:

- $\mathcal{O}(k^2)$ query time,
- $\mathcal{O}(n)$ preprocessing time.

- $\mathrm{MaxSuf}^R(v)\$ = \mathrm{MinSuf}(v\$)$ where $\$ > c$ for every $c \in \Sigma$.
- $\mathrm{MinRot}(v) = \mathrm{MinSuf}(vv\$)[1..|v|]$.

Applications

Lyndon word: $w = \mathrm{MinSuf}(w)$

Lyndon factorization: $w = w_1^{\alpha_1} \cdots w_m^{\alpha_m}$ such that:

- $w_1 > \cdots > w_m$,
- $w_1, \ldots, w_m$ are Lyndon words.

Applications

Lyndon word: $w = \mathrm{MinSuf}(w)$

Lyndon factorization: $w = w_1^{\alpha_1} \cdots w_m^{\alpha_m}$ such that:

- $w_1 > \cdots > w_m$,
- $w_1, \ldots, w_m$ are Lyndon words.

Lyndon factorization is unique; $w_m = \mathrm{MinSuf}(w)$.

Applications

Lyndon word: $w = \mathrm{MinSuf}(w)$

Lyndon factorization: $w = w_1^{\alpha_1} \cdots w_m^{\alpha_m}$ such that:

- $w_1 > \cdots > w_m$,
- $w_1, \ldots, w_m$ are Lyndon words.

Lyndon factorization is unique; $w_m = \mathrm{MinSuf}(w)$.

Our result:
After $\mathcal{O}(n)$-time preprocessing, Lyndon factorization of the concatenation of k fragments can be computed in $\mathcal{O}(k^2 m)$ time.

Combinatorial tools

Significant suffixes

$\mathrm{MinSuf}(v, w) = \min\{sw : s \text{ is a suffix of } v\}$

Observation: $\mathrm{MinSuf}(vw) \in \{\mathrm{MinSuf}(v, w), \mathrm{MinSuf}(w)\}$.

Significant suffixes

$\mathrm{MinSuf}(v, w) = \min\{sw : s \text{ is a suffix of } v\}$

Observation: $\mathrm{MinSuf}(vw) \in \{\mathrm{MinSuf}(v, w), \mathrm{MinSuf}(w)\}$.

Definition (I et al., CPM'13, SPIRE'13)

A suffix s of v is **significant** if $sw = \mathrm{MinSuf}(v, w)$ for some $w \neq \varepsilon$.

Notation: $\Lambda(v) = \{s_0, \ldots, s_m\}$ (the set of significant suffixes of v).

- $s_0 = \varepsilon$, $|s_i| > 2|s_{i-1}|$
  - $|\Lambda(v)| = \mathcal{O}(\log |v|)$
- $s_{i+1} = s_i x_i$ (s_i is a border of s_{i-1}),

Significant suffixes

$\mathrm{MinSuf}(v, w) = \min\{sw : s \text{ is a suffix of } v\}$

Observation: $\mathrm{MinSuf}(vw) \in \{\mathrm{MinSuf}(v, w), \mathrm{MinSuf}(w)\}$.

Definition (I et al., CPM'13, SPIRE'13)

A suffix s of v is **significant** if $sw = \mathrm{MinSuf}(v, w)$ for some $w \neq \varepsilon$.

Notation: $\Lambda(v) = \{s_0, \ldots, s_m\}$ (the set of significant suffixes of v).

- $s_0 = \varepsilon$, $|s_i| > 2|s_{i-1}|$
  - $|\Lambda(v)| = \mathcal{O}(\log |v|)$
- $s_{i+1} = s_i x_i$ (s_i is a border of s_{i-1}),

Notation: $X(v) = \{x_0^\infty, \ldots, x_{m-1}^\infty\}$.

- $x_0^\infty < \cdots < x_{m-1}^\infty$.

Characterization of $\mathrm{MinSuf}(v, w)$

Theorem (I et al., SPIRE'13)

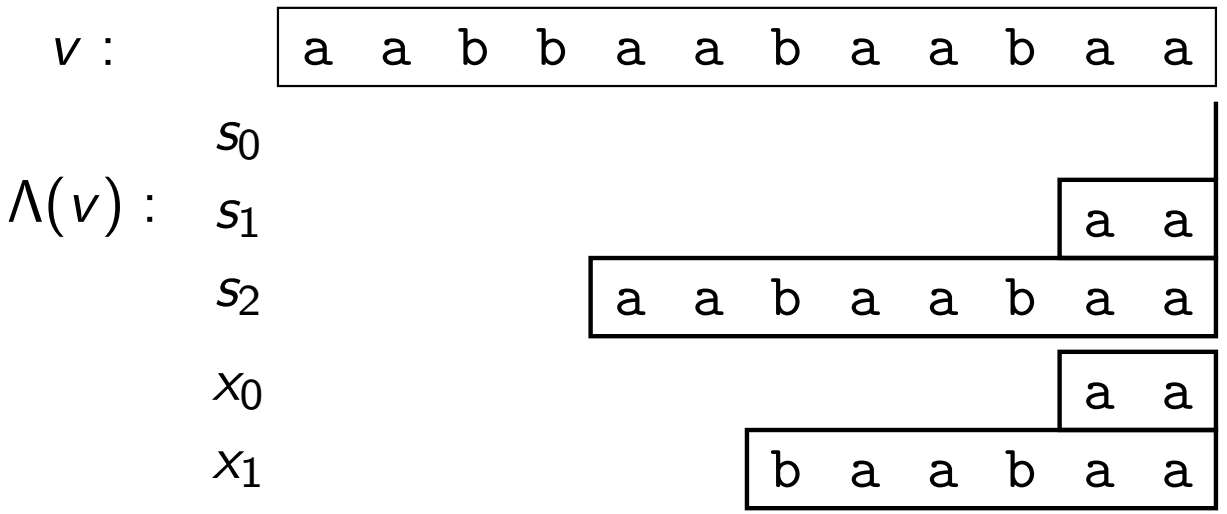
$\mathrm{MinSuf}(v, w) = s_r w$ *where* $r = \mathrm{rank}_{X(v)}(w)$

Characterization of $\mathrm{MinSuf}(v, w)$

Theorem (I et al., SPIRE'13)

$\mathrm{MinSuf}(v, w) = s_r w$ *where* $r = \mathrm{rank}_{X(v)}(w)$

Example:

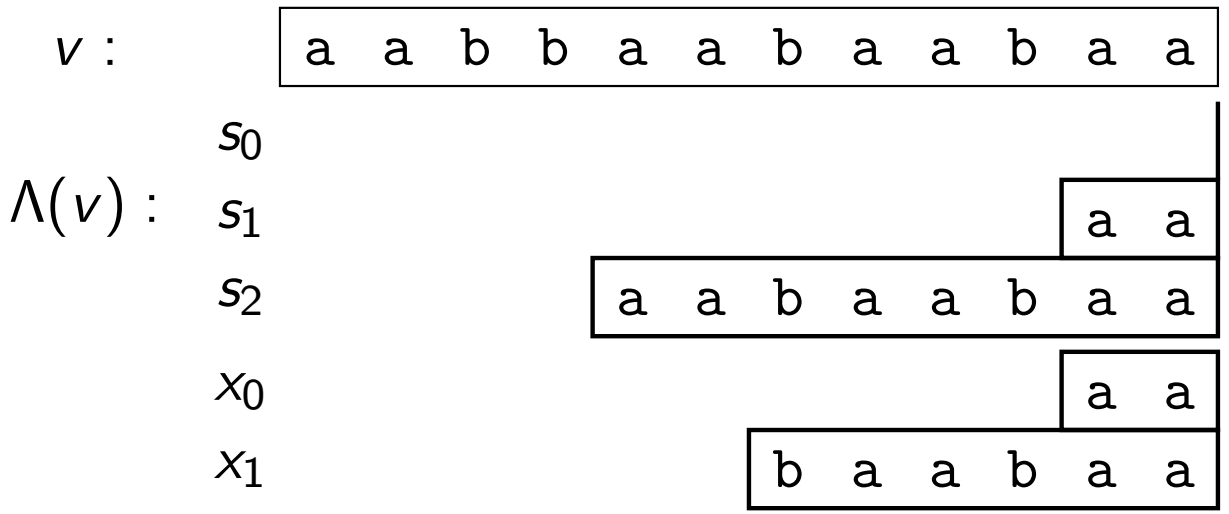


Characterization of $\mathrm{MinSuf}(v, w)$

Theorem (I et al., SPIRE'13)

$\mathrm{MinSuf}(v, w) = s_r w$ *where* $r = \mathrm{rank}_{X(v)}(w)$

Example:



$$\mathrm{MinSuf}(v, w) = \begin{cases} w & \text{if } w < (aa)^\infty \\ aaw & \text{if } (aa)^\infty < w < (baabaa)^\infty \\ aabaabaaw & \text{if } (baabaa)^\infty < w \end{cases}$$

Computing significant suffixes

Problem: compute $\Lambda(v)$ for $v = T[\ell..r]$.

Computing significant suffixes

Problem: compute $\Lambda(v)$ for $v = T[\ell..r]$.

Fact

If $v = uv'$ *for* $|u| \leq |v'|$, *then* $\Lambda(v) \subseteq \{\mathrm{MaxSuf}^R(u, v'), \Lambda(v')\}$.

Computing significant suffixes

Problem: compute $\Lambda(v)$ for $v = T[\ell..r]$.

Fact

If $v = uv'$ for $|u| \leq |v'|$, then $\Lambda(v) \subseteq \{\mathrm{MaxSuf}^R(u, v'), \Lambda(v')\}$.

Lemma

$\mathrm{MaxSuf}^R(u, v')$ *can be computed in $\mathcal{O}(1)$ time using* ***augmented suffix array*** *(SA, ISA+RMQ, LCP+RMQ built for T and T^R).*

Implicit in CPM'13 & CPM'14.

Computing significant suffixes

Problem: compute $\Lambda(v)$ for $v = T[\ell..r]$.

Fact

If $v = uv'$ for $|u| \leq |v'|$, then $\Lambda(v) \subseteq \{\mathrm{MaxSuf}^R(u, v'), \Lambda(v')\}$.

Lemma

$\mathrm{MaxSuf}^R(u, v')$ *can be computed in $\mathcal{O}(1)$ time using* ***augmented suffix array*** *(SA, ISA+RMQ, LCP+RMQ built for T and T^R).*

Implicit in CPM'13 & CPM'14.

Corollary

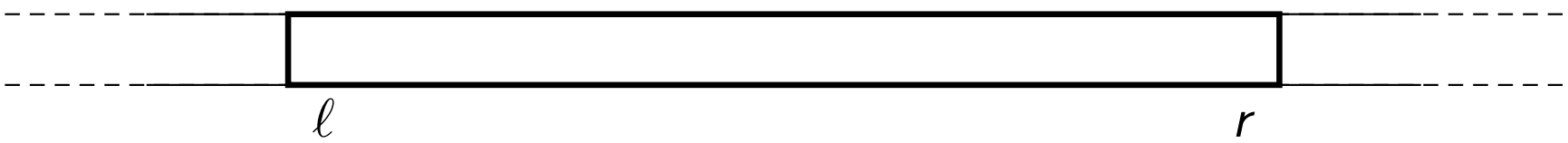
$\Lambda(v)$ *can be computed in $\mathcal{O}(\log |v|)$ time using augmented SA.*

Recurse on v' and filter $\{\mathrm{MaxSuf}^R(u, v'), \Lambda(v')\}$.

Data structure

Inefficient solutions

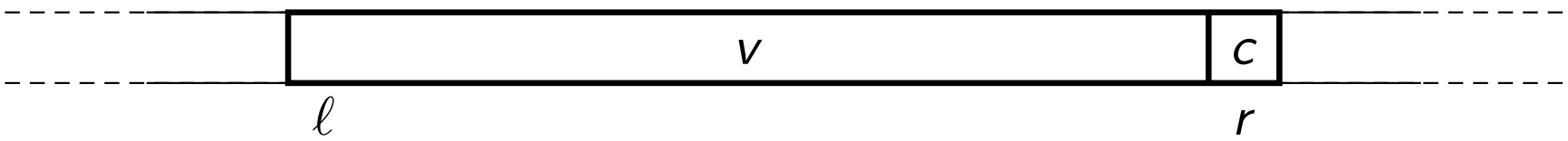
$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:



Inefficient solutions

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

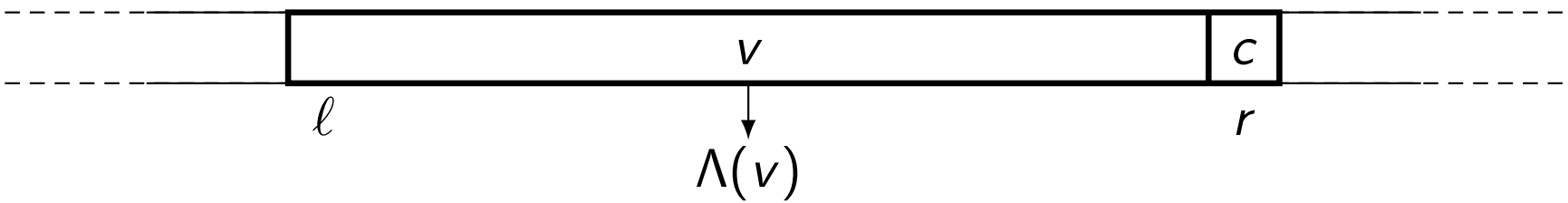
1. Let $T[\ell..r] = vc$ where $|c| = 1$.



Inefficient solutions

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

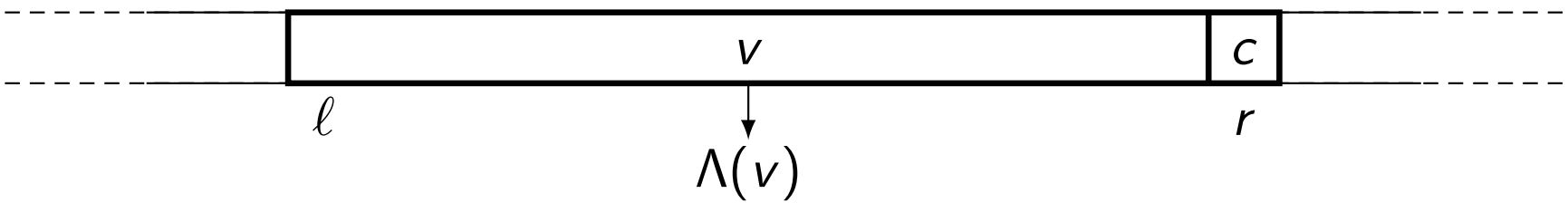
1. Let $T[\ell..r] = vc$ where $|c| = 1$.
2. Compute $\Lambda(v)$.



Inefficient solutions

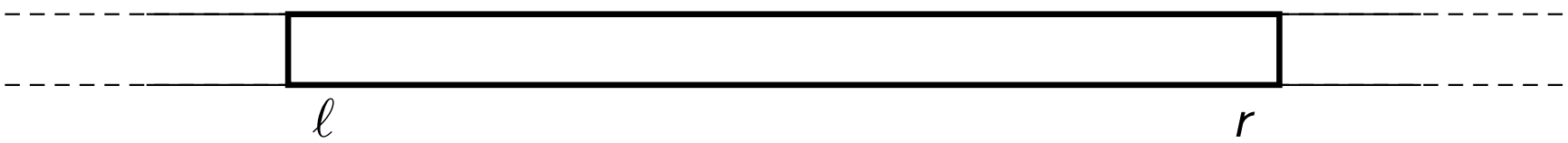
$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

1. Let $T[\ell..r] = vc$ where $|c| = 1$.
2. Compute $\Lambda(v)$.
3. Return $\min\{sc : s \in \Lambda(v)\}$.



Inefficient solutions

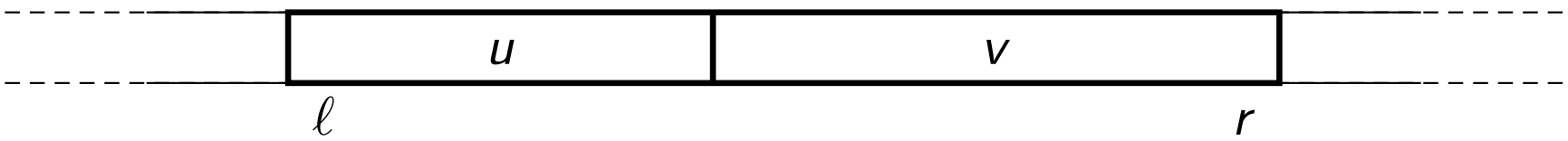
$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:



Inefficient solutions

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

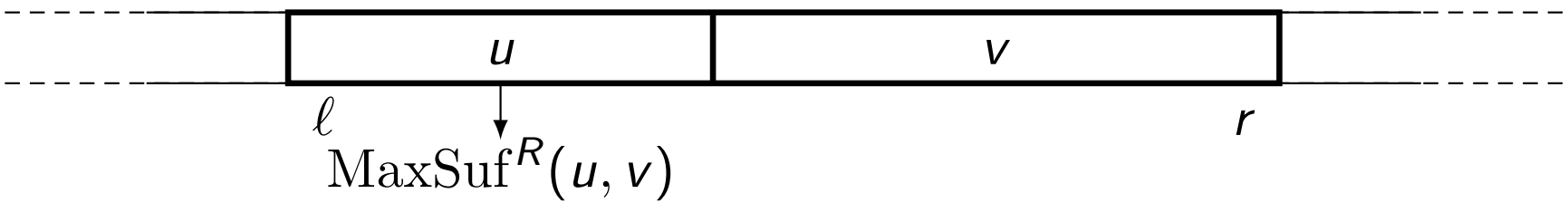
1. Let $T[\ell..r] = uv$ where $|u| + 1 \leq |v| \leq |u| + 2$.



Inefficient solutions

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

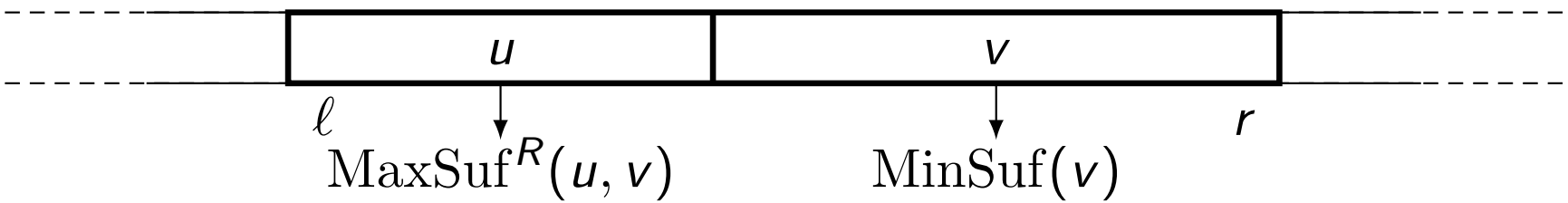
1. Let $T[\ell..r] = uv$ where $|u| + 1 \leq |v| \leq |u| + 2$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)$ using the augmented SA.



Inefficient solutions

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

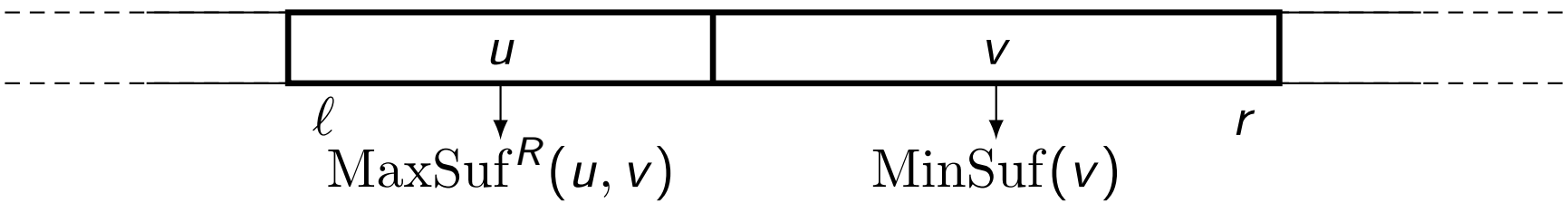
1. Let $T[\ell..r] = uv$ where $|u| + 1 \leq |v| \leq |u| + 2$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)$ using the augmented SA.
3. Recursively compute $S_2 = \mathrm{MinSuf}(v)$.



Inefficient solutions

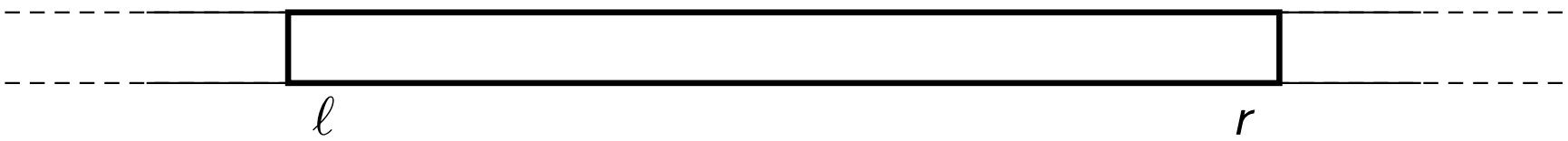
$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log n)$-time queries:

1. Let $T[\ell..r] = uv$ where $|u| + 1 \leq |v| \leq |u| + 2$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)$ using the augmented SA.
3. Recursively compute $S_2 = \mathrm{MinSuf}(v)$.
4. Return $\min(S_1, S_2)$.



Inefficient solutions

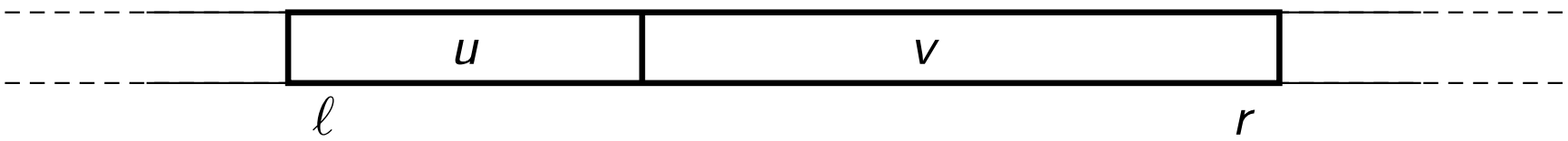
$\mathcal{O}(n \log n)$-time preprocessing, $\mathcal{O}(1)$-time queries:



Inefficient solutions

$\mathcal{O}(n \log n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

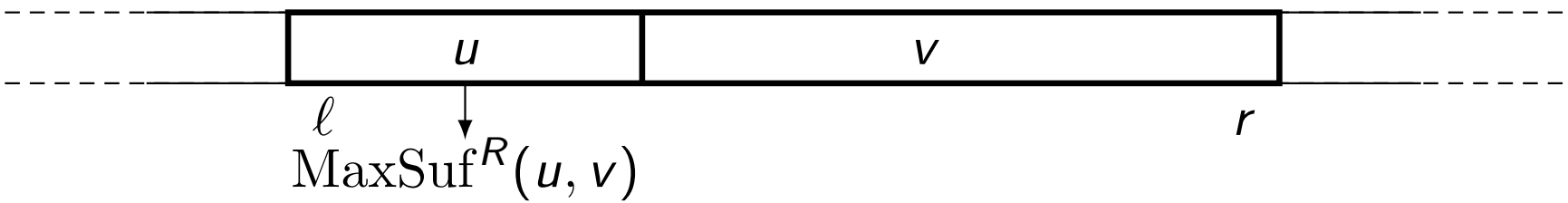
1. Let $T[\ell..r] = uv$ where $|u| < |v|$ and $|v|$ is power of two.



Inefficient solutions

$\mathcal{O}(n \log n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

1. Let $T[\ell..r] = uv$ where $|u| < |v|$ and $|v|$ is power of two.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)$ using the augmented SA.



Inefficient solutions

$\mathcal{O}(n \log n)$-**time preprocessing,** $\mathcal{O}(1)$-**time queries:**

1. Let $T[\ell..r] = uv$ where $|u| < |v|$ and $|v|$ is power of two.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)$ using the augmented SA.
3. Read precomputed $S_2 = \mathrm{MinSuf}(v)$

Inefficient solutions

$\mathcal{O}(n \log n)$-**time preprocessing,** $\mathcal{O}(1)$-**time queries:**

1. Let $T[\ell..r] = uv$ where $|u| < |v|$ and $|v|$ is power of two.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)$ using the augmented SA.
3. Read precomputed $S_2 = \mathrm{MinSuf}(v)$
4. Return $\min(S_1, S_2)$.

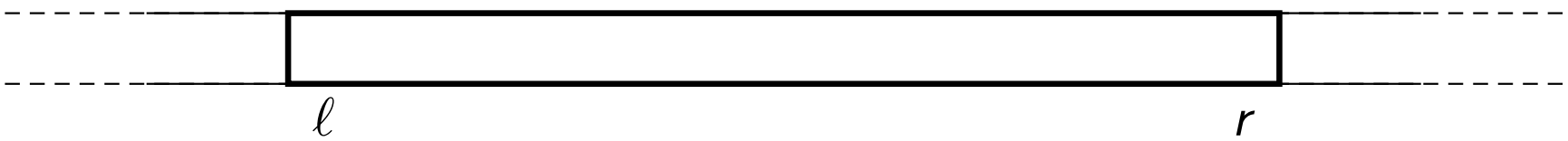
Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

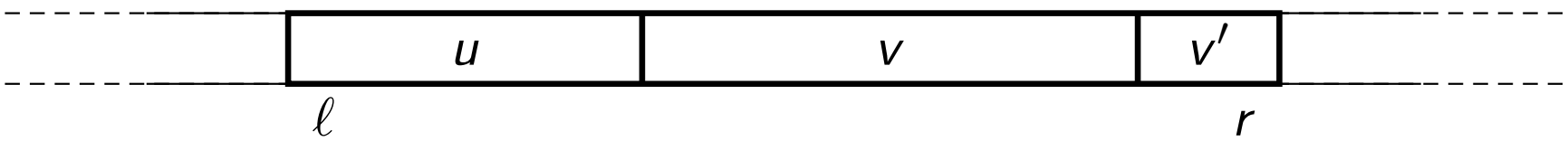


Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

1. Partition $T[\ell..r] = uvv'$ so that $|u| < |v|$, v is distinguished, and $|v'| \leq f(|T[\ell..r]|)$.

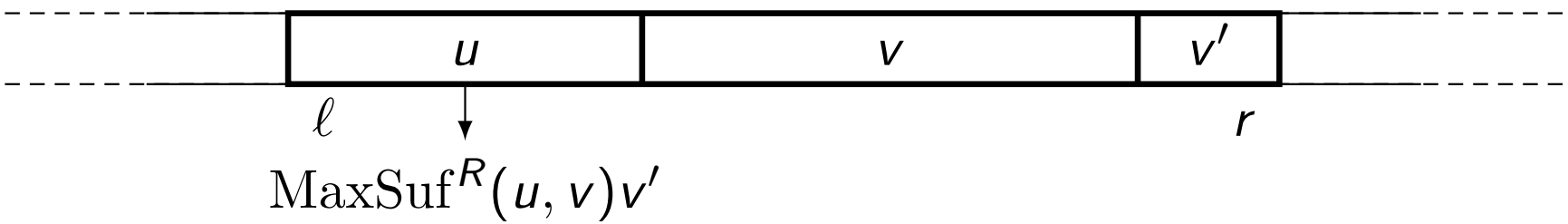


Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

1. Partition $T[\ell..r] = uvv'$ so that $|u| < |v|$, v is distinguished, and $|v'| \leq f(|T[\ell..r]|)$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)v'$.

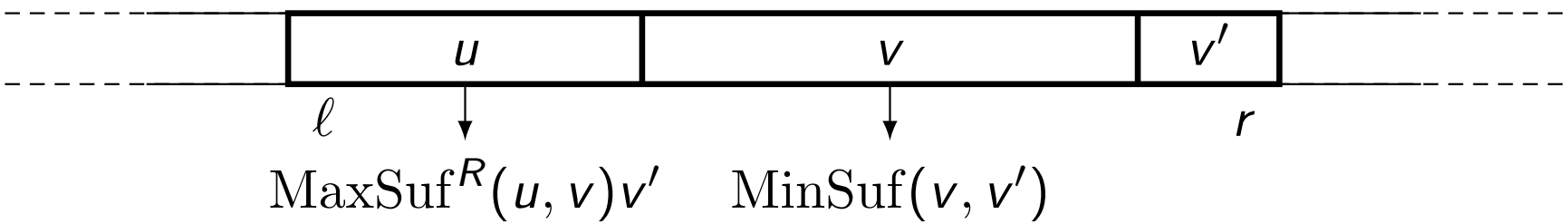


Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

1. Partition $T[\ell..r] = uvv'$ so that $|u| < |v|$, v is distinguished, and $|v'| \leq f(|T[\ell..r]|)$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)v'$.
3. Compute $S_2 = \mathrm{MinSuf}(v, v')$ using precomputed data for v.

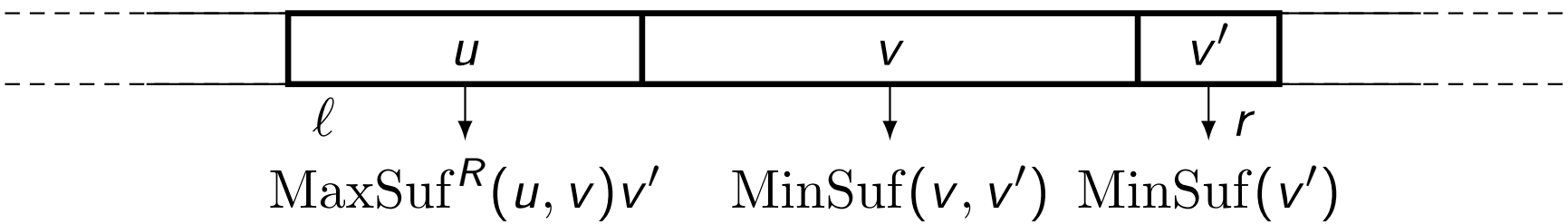


Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

1. Partition $T[\ell..r] = uvv'$ so that $|u| < |v|$, v is distinguished, and $|v'| \leq f(|T[\ell..r]|)$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)v'$.
3. Compute $S_2 = \mathrm{MinSuf}(v, v')$ using precomputed data for v.
4. Recursively compute $S_3 = \mathrm{MinSuf}(v')$.

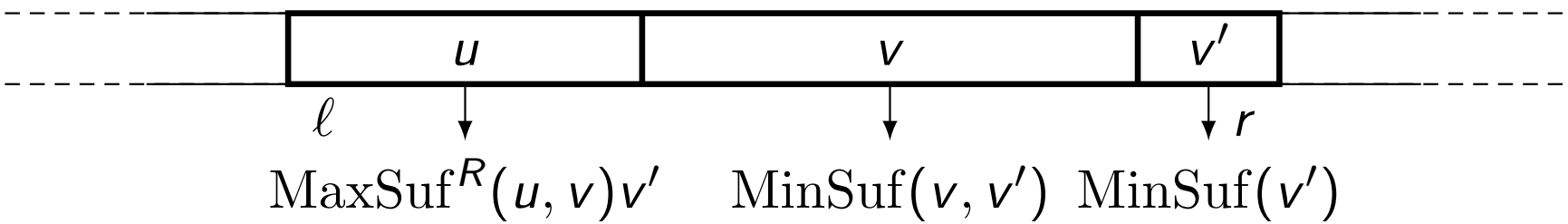


Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

1. Partition $T[\ell..r] = uvv'$ so that $|u| < |v|$, v is distinguished, and $|v'| \leq f(|T[\ell..r]|)$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)v'$.
3. Compute $S_2 = \mathrm{MinSuf}(v, v')$ using precomputed data for v.
4. Recursively compute $S_3 = \mathrm{MinSuf}(v')$.
5. Return $\min(S_1, S_2, S_3)$.

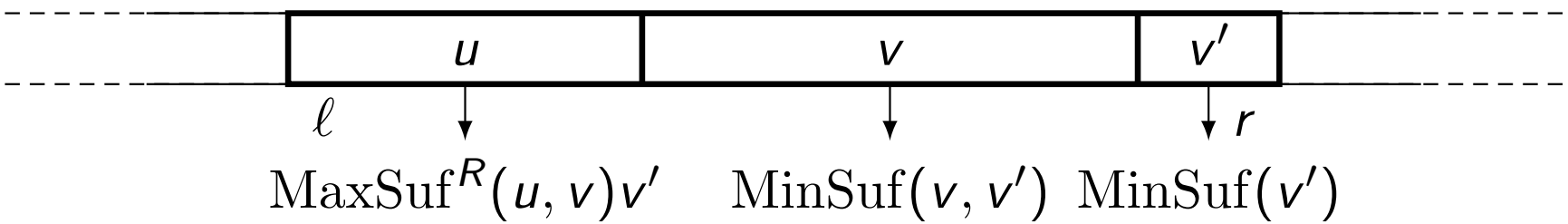


Almost optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(\log^* n)$-time queries

Preprocess only **distinguished fragments**:
every $f(2^q)$-th fragment of length 2^q for $f(x) = 2^{\mathcal{O}(\log^2 \log x)}$.

1. Partition $T[\ell..r] = uvv'$ so that $|u| < |v|$, v is distinguished, and $|v'| \leq f(|T[\ell..r]|)$.
2. Compute $S_1 = \mathrm{MaxSuf}^R(u, v)v'$.
3. Compute $S_2 = \mathrm{MinSuf}(v, v')$ using precomputed data for v.
4. Recursively compute $S_3 = \mathrm{MinSuf}(v')$.
5. Return $\min(S_1, S_2, S_3)$.



Depth of recursion: $\mathcal{O}(\log^* n)$ since $f(f(x)) = \log^{o(1)} x = o(\log x)$.

Main component

Problem:

Preprocess $u = T[\ell..r]$ to determine $\mathrm{MinSuf}(u, T[r+1..r'])$.

Combinatorial tool: Significant suffixes $\Lambda(u) = \{s_0, \ldots, s_m\}$

$\mathrm{MinSuf}(u, v) = s_{\mathrm{rank}_{X(u)}(v)}$ where $X(u) = \{x_0^\infty, \ldots, x_{m-1}^\infty\}$.

Main component

Problem:

Preprocess $u = T[\ell..r]$ to determine $\mathrm{MinSuf}(u, T[r+1..r'])$.

Combinatorial tool: Significant suffixes $\Lambda(u) = \{s_0, \ldots, s_m\}$

$\mathrm{MinSuf}(u, v) = s_{\mathrm{rank}_{X(u)}(v)}$ where $X(u) = \{x_0^\infty, \ldots, x_{m-1}^\infty\}$.

$$x_0^\infty < \cdots < x_{m-1}^\infty$$

$$\varepsilon = T[r+1..r] < T[r+1..r+1] < \cdots < T[r+1..n]$$

Main component

Problem:
Preprocess $u = T[\ell..r]$ to determine $\mathrm{MinSuf}(u, T[r+1..r'])$.

Combinatorial tool: Significant suffixes $\Lambda(u) = \{s_0, \ldots, s_m\}$
$\mathrm{MinSuf}(u, v) = s_{\mathrm{rank}_{X(u)}(v)}$ where $X(u) = \{x_0^\infty, \ldots, x_{m-1}^\infty\}$.

$$x_0^\infty < \cdots < x_{m-1}^\infty$$

$$\varepsilon = T[r+1..r] < T[r+1..r+1] < \cdots < T[r+1..n]$$

Construct an integer set $R(u)$ such that

$$\mathrm{rank}_{R(u)}(r') = \mathrm{rank}_{X(u)}(T[r+1..r']).$$

Main component

Problem:
Preprocess $u = T[\ell..r]$ to determine $\mathrm{MinSuf}(u, T[r+1..r'])$.

Combinatorial tool: Significant suffixes $\Lambda(u) = \{s_0, \ldots, s_m\}$
$\mathrm{MinSuf}(u, v) = s_{\mathrm{rank}_{X(u)}(v)}$ where $X(u) = \{x_0^\infty, \ldots, x_{m-1}^\infty\}$.

$$x_0^\infty < \cdots < x_{m-1}^\infty$$

$$\varepsilon = T[r+1..r] < T[r+1..r+1] < \cdots < T[r+1..n]$$

Construct an integer set $R(u)$ such that

$$\mathrm{rank}_{R(u)}(r') = \mathrm{rank}_{X(u)}(T[r+1..r']).$$

Algorithmic tool: fusion trees [Pătraşcu & Thorup, FOCS 2015]
$\mathcal{O}(1)$-time queries and $\mathcal{O}(R(u))$-time construction
(since $|R(u)| = W^{\mathcal{O}(1)}$ where $W = \Omega(\log n)$ is machine word size).

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Observation: In the third step of recursion, $|v| = \log^{o(1)} n$.

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Observation: In the third step of recursion, $|v| = \log^{o(1)} n$.

How to memorize answers for very short fragments?

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Observation: In the third step of recursion, $|v| = \log^{o(1)} n$.

How to memorize answers for very short fragments?

- Easy for small alphabets ($\sigma^{\log^{o(1)} n} = 2^{o(\log n)}$ possibilities).

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Observation: In the third step of recursion, $|v| = \log^{o(1)} n$.

How to memorize answers for very short fragments?

- Easy for small alphabets ($\sigma^{\log^{o(1)} n} = 2^{o(\log n)}$ possibilities).
- Alphabet size reduction:
  - answers depend on the relative order of characters

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Observation: In the third step of recursion, $|v| = \log^{o(1)} n$.

How to memorize answers for very short fragments?

- Easy for small alphabets ($\sigma^{\log^{o(1)} n} = 2^{o(\log n)}$ possibilities).
- Alphabet size reduction:
  - answers depend on the relative order of characters
  - E.g., `abac`, `bcbe`, `acad` are indistinguishable (**order isomorphic**).

Optimal solution

$\mathcal{O}(n)$-time preprocessing, $\mathcal{O}(1)$-time queries:

Main idea: Indirection

Observation: In the third step of recursion, $|v| = \log^{o(1)} n$.

How to memorize answers for very short fragments?

- Easy for small alphabets ($\sigma^{\log^{o(1)} n} = 2^{o(\log n)}$ possibilities).
- Alphabet size reduction:
  - answers depend on the relative order of characters
  - E.g., `abac`, `bcbe`, `acad` are indistinguishable (**order isomorphic**).

Solution:

1. Consider $\mathcal{O}(n/f(f(n)))$ fragments of length $2f(f(n))$.
2. Group them by order isomorphism (e.g. using fusion trees).
3. Memorize all the answers for a representative of each class.

Summary & open problems

Our results:

Minimal Suffix Queries

$\mathcal{O}(1)$-time queries, $\mathcal{O}(n)$-time preprocessing,

Generalized Minimal Suffix Queries

minimal suffix of the concatenation of k substrings

$\mathcal{O}(k^2)$-time queries, $\mathcal{O}(n)$-time preprocessing,

Minimal Rotation Queries

$\mathcal{O}(1)$-time queries, $\mathcal{O}(n)$-time preprocessing.

Summary & open problems

Our results:

Minimal Suffix Queries

$\mathcal{O}(1)$-time queries, $\mathcal{O}(n)$-time preprocessing,

Generalized Minimal Suffix Queries

minimal suffix of the concatenation of k substrings

$\mathcal{O}(k^2)$-time queries, $\mathcal{O}(n)$-time preprocessing,

Minimal Rotation Queries

$\mathcal{O}(1)$-time queries, $\mathcal{O}(n)$-time preprocessing.

Open problems:

- Encoding with $o(n \log n)$ bits for min/max suffix queries?
- Efficient $o(n \log n)$-bit data structure for constant alphabets?
- Lyndon factorization of RLZ-compressed text in $o(k^2 m)$ time?

Questions?

Thank you for your attention!