Computing minimal and maximal suffixes of a substring revisited

Maxim Babenko[1] Paweł Gawrychowski[2]
Tomasz Kociumaka[3] Tatiana Starikovskaya[1]

National Research University Higher School of Economics (HSE)

Max-Planck-Institut für Informatik

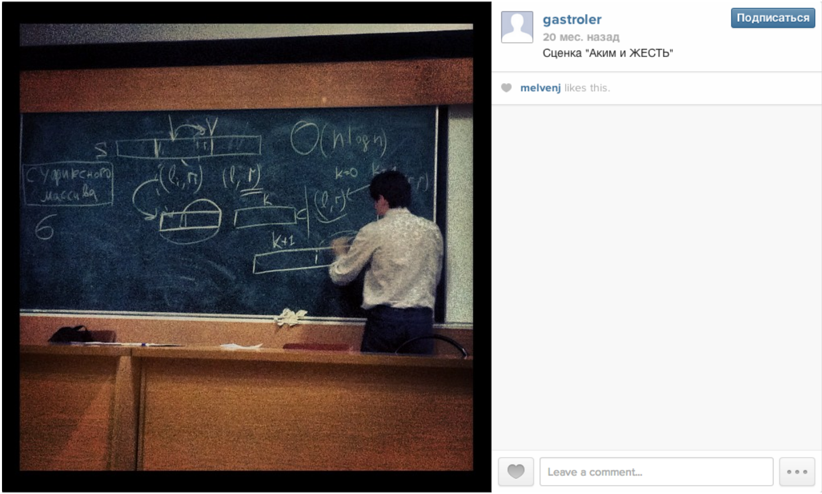
Institute of Informatics, University of Warsaw

Given a string T of length n.

- Find the lex. maximal suffix (**maxsuffix**) of $T[i..j]$
- Find the lex. minimal suffix (**minsuffix**) of $T[i..j]$

Example:

$T = ababc$, for a substring bab:
maxsuffix $= bab$
minsuffix $= ab$



ACM International Collegiate Programming Contest (ICPC) 2012
Darkwing Duck problem by V. Astakhov $\Rightarrow$ **Maxsuffix** problem
q queries $\Rightarrow$ $O(n \log n + q)$ time, $O(n + q)$ space

Substring queries

- Minsuffix and maxsuffix for **all prefixes** of a string in linear time
[1983, Duval]
- Periodicity of a substring
[2010, Crochemore et al.; 2012, Kociumaka et al.]
- All occurrences of a substring in another substring
[2013, Kociumaka et al.]
- Cyclic equivalence queries for substrings
[2013, Kociumaka et al.]
- Compressibility of substrings
[2005, Cormode and Muthukrishnan; 2013, Keller et al.]

CPM'13: Babenko, Kolesnichenko, S.

Minsuffix:

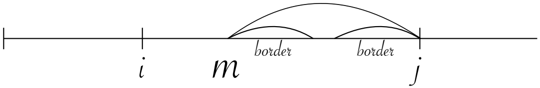
- $O(n)$ space
- $O(\log^{1+\epsilon} n)$ query time

Maxsuffix:

- $O(n)$ space
- $O(\log n)$ query time

Minsuffix: $O(n)$ space, $O(\log^{1+\epsilon} n)$ query time
[CPM'13: Babenko, Kolesnichenko, S.]

- $T[m..]$ — the minsuffix among $T[i..], \ldots, T[j..]$
- $T[m..j]$ **IS NOT YET** the answer:
  $T = babac$, $T[i..j] = aba$
  $T[m..] = \min\{abac,\ bac,\ ac\} = abac$
  $T[m..j] = aba$
- ..but the minimum of $T[m..j]$ and the shortest (proper) border β of $T[m..j]$ **IS** the answer!

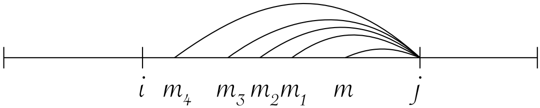


m — $O(1)$ time [suffix arrays + RMQ]

β — $O(\log^{1+\epsilon} n)$ time [2012, Kociumaka et al.]

Maxsuffix: $O(n)$ space, $O(\log n)$ query time
[CPM'13: Babenko, Kolesnichenko, S.]

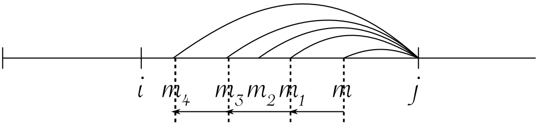
- If $T[m..]$ is the maxsuffix among $T[i..], \ldots, T[j..]$, then $T[m..j]$ is a prefix of the answer
- $T[m_1..]$ — the maxsuffix among $T[i..], \ldots, T[m-1..]$
- $T[m_1..j]$ starts with $T[m..j] \Rightarrow T[m_1..j]$ is a longer prefix, otherwise, $T[m..j]$ is the answer



- Up to $j - i$ next prefix iterations

Maxsuffix: $O(n)$ space, $O(\log n)$ query time
[CPM'13: Babenko, Kolesnichenko, S.]

- If $T[m..]$ is the maxsuffix among $T[i..], \ldots, T[j..]$, then $T[m..j]$ is a prefix of the answer
- $T[m_1..]$ — the maxsuffix among $T[i..], \ldots, T[m-1..]$
- $T[m_1..j]$ starts with $T[m..j] \Rightarrow T[m_1..j]$ is a longer prefix, otherwise, $T[m..j]$ is the answer



- Next prefix iterations + run skips — $O(\log n)$ time

CPM'14: Babenko, Gawrychowski, Kociumaka, S.

Minsuffix:

- $O(n)$ space
- $O(\tau)$ query time, $1 \leq \tau \leq \log n$ (CPM'13: $O(\log^{1+\epsilon} n)$)
- $O(n \log n / \tau)$ construction time

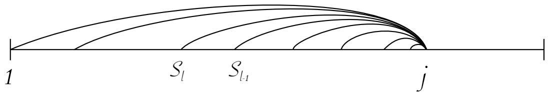
Maxsuffix:

- $O(n)$ space
- $O(1)$ query time (CPM'13: $O(\log n)$)
- $O(n)$ construction time

Minsuffix

Canonical substrings

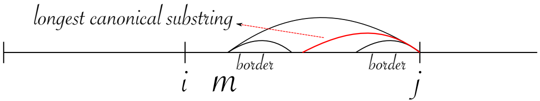
- $O(\log n)$ substrings for a fixed j
- The first substring is $T[j]$, the last substring is $T[1..j]$
- $|S_\ell| \leq 2|S_{\ell-1}|$



- If $|\text{Minsuffix of } S_\ell| > |S_{\ell-1}|$, we can find it in $O(1)$ time

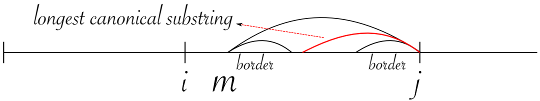
$O(1)$ query / $O(n \log n)$ construction time

- If $T[m..]$ is the minsuffix among $T[i..], \ldots, T[j..]$, then the answer is $T[m..j]$ or the shortest border β of $T[m..j]$
- $|\beta| < |T[m..j]|/2 <$ the longest canonical substring in $T[i..j]$



$O(1)$ query / $O(n \log n)$ construction time

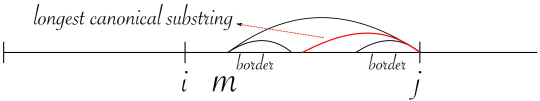
- If $T[m..]$ is the minsuffix among $T[i..], \ldots, T[j..]$, then the answer is $T[m..j]$ or the shortest border β of $T[m..j]$
- $|\beta| < |T[m..j]|/2 <$ the longest canonical substring in $T[i..j]$



- If β is the minsuffix and S_k, $|S_k| \to$ min, contains β

$O(1)$ query / $O(n \log n)$ construction time

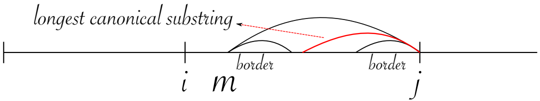
- If $T[m..]$ is the minsuffix among $T[i..], \ldots, T[j..]$, then the answer is $T[m..j]$ or the shortest border β of $T[m..j]$
- $|\beta| < |T[m..j]|/2 <$ the longest canonical substring in $T[i..j]$



- If β is the minsuffix and S_k, $|S_k| \to \min$, contains β
- then β is the minsuffix of S_k, and $|\beta| > |S_{k-1}|$

$O(1)$ query / $O(n \log n)$ construction time

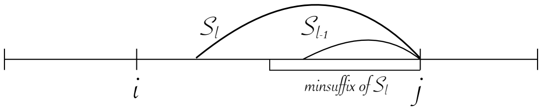
- If $T[m..]$ is the minsuffix among $T[i..], \ldots, T[j..]$, then the answer is $T[m..j]$ or the shortest border β of $T[m..j]$
- $|\beta| < |T[m..j]|/2 <$ the longest canonical substring in $T[i..j]$



- If β is the minsuffix and S_k, $|S_k| \to \min$, contains β
- then β is the minsuffix of S_k, and $|\beta| > |S_{k-1}|$
- β: $O(1)$ time

How to find k?

- Use bit vectors!



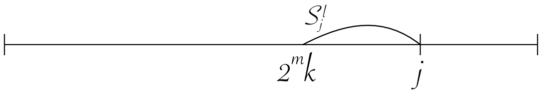
- $B_j[\ell] = 1$ if the minsuffix of S_ℓ is longer than $S_{\ell-1}$
- $B_j[k] = 1$, S_k is a canonical substring of $T[i..j]$, $k \to$ max
- k: $O(1)$ time [standard bit vector operations]

$O(1)$ query / $O(n \log n)$ construction time

- To compute B_j, it suffices to compute minsuffixes of all S_ℓ

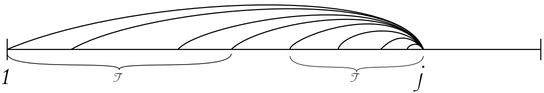
$m = \lfloor \ell/2 \rfloor - 1$

$$|S_\ell| = \begin{cases} 2 \cdot 2^m + (j \bmod 2^m) & \text{if } \ell \text{ is even,} \\ 3 \cdot 2^m + (j \bmod 2^m) & \text{otherwise.} \end{cases}$$



- The substrings are **prefixes** of blocks of length at most 2^{m+2} starting at multiples of 2^m
- Use Duval's algorithm with time $O(2^{m+2})$ to compute minsuffixes of all S_ℓ, $j \in [1, n]$, starting at a fixed multiple of 2^m
- $O(n)$ time for all multiples of 2^m, $O(n \log n)$ time overall

Trade-off: $O(\tau)$ query / $O(n \log n/\tau)$ construction time



- $B_j[\ell] = 1$ if the ℓ-th group of τ consecutive substrings ending at j is interesting
- **Query:** apply the main lemma to the first interesting group of canonical substrings in $[i, j]$ — $O(\tau)$ time
- **Construction:** $\log n/\tau$ rounds of Duval's algorithm — $O(n \log n/\tau)$ time overall

Lyndon decomposition

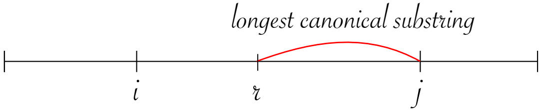
- **Lyndon string** — strictly smaller than any of its rotations
  Example: $aabc < abca, bcaa, cabb$

- **Lyndon decomposition:** $s = s_1^{\alpha_1} \dots s_k^{\alpha_k}$
- $s_1 > \dots > s_k$ — Lyndon strings
- $\exists!$ L.d., s_k is the minsuffix of s
  [1958, Chen, Lyndon, Fox; 1983, Duval]

- **Corollary:** L.d. of $s = T[i..j]$ can be found in $O(\tau k)$ time
- s_k — $O(\tau)$ time, α_k — $O(1)$ time

Maxsuffix

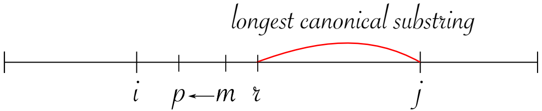
Maxsuffix — main lemma



The answer is the maximum of

- the maxsuffix of the longest canonical substring S in $T[i..j]$
- $T[p..j]$, where $p \in [i, r-1]$ can be found in $O(1)$ time

The maxsuffix of S: $O(1)$ time [bit vector operations]



- $T[m..]$ — the maxsuffix starting within $T[i..r]$
- $|T[m..j]| \geq |S| \geq |T[i..j]|/2$
- p: $O(1)$ time [next prefix iterations + run skips]

Main results

1. Minsuffix of a substring: $O(n)$ space, $O(\tau)$ query time, $O(n \log n / \tau)$ construction time
2. Lyndon decomposition of a substring: $O(n)$ space, $O(size \cdot \tau)$ query time, $O(n \log n / \tau)$ construction time
3. Maxsuffix of a substring: $O(n)$ space, $O(1)$ query time, $O(n)$ construction time

Open problems

1. Better solution for Minsuffix?
2. What makes Minsuffix harder than Maxsuffix?
3. CPM'13: the k-th lex. smallest suffix of a substring?
   $O(n)$ space, $O(\log^{2+\epsilon} n)$ time
   [Babenko, Gawrychowski, Kociumaka, S., arxiv.org]