On Minimal and Maximal Suffixes of a Substring

Maxim Babenko
Ignat Kolesnichenko
Tatiana Starikovskaya

Moscow State University,
Moscow, Russia

Problem

Let T be a string. Consider two problems:

- Find the lexicographical minimal non-empty suffix of $T[i\,..\,j]$
- Find the lexicographical maximal suffix of $T[i\,..\,j]$

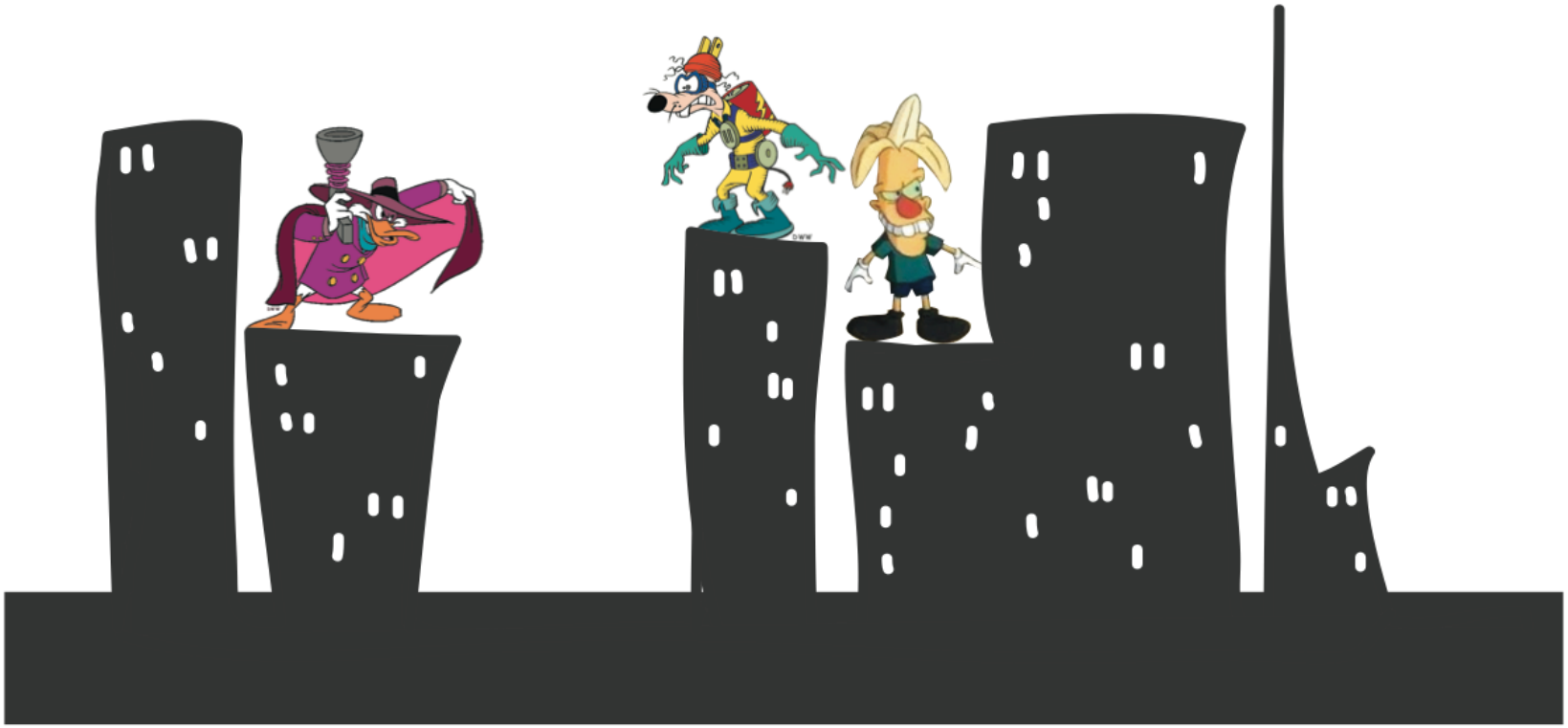
Example

$$T = baabac$$

The minimal suffix of $T[1\,..\,5] = baaba$ is $T[5\,..\,5] = a$

The maximal suffix of $T[1\,..\,5] = baaba$ is $T[1\,..\,5] = baaba$

Inspiration



- "Darkwing Duck" problem by Vasily Astakhov from ACM International Collegiate Programming Contest (ICPC) 2012
- It can be reduced to the problem of computing maximal suffixes of given subtrings

Related Results

The maximal/minimal suffixes of

- a string can be found in linear time (suffix array or suffix tree)
- all prefixes of a string — in linear time (J.-P. Duval, 1983)

Our Contribution

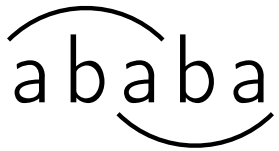
There exists a linear space data structure for T that allows to find the minimal suffix of any substring of T in time $O(\log^{1+\varepsilon} |T|)$ and to find the maximal suffix of any substring of T in time $O(\log |T|)$.

Minimal Suffix Problem

Minimal Suffix Problem

Definition

A *border* of a string T is a string that is both a prefix and a suffix of T and differs from T.



Minimal Suffix Problem

Definition

A *border* of a string T is a string that is both a prefix and a suffix of T and differs from T.

Let $T[m\,..]$ be the min suffix among $T[i\,..], T[i+1\,..], \ldots, T[j\,..]$

Minimal Suffix Problem

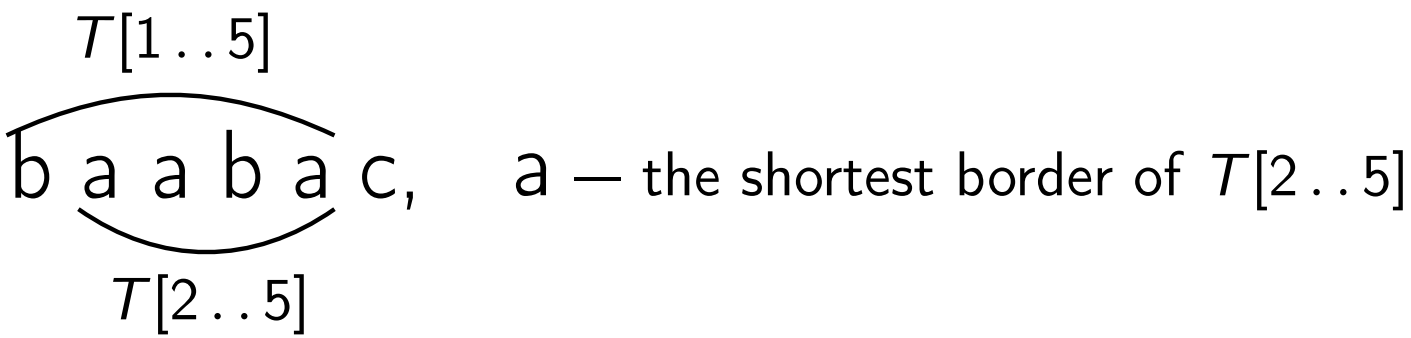
Definition

A *border* of a string T is a string that is both a prefix and a suffix of T and differs from T.

Let $T[m\,..\,]$ be the min suffix among $T[i\,..\,], T[i{+}1\,..\,], \ldots, T[j\,..\,]$

Lemma

If $T[m\,..\,j]$ is *border-free* then it is the minimal suffix of $T[i\,..\,j]$, otherwise the minimal suffix equals to the *shortest* non-empty border of $T[m\,..\,j]$.



Minimal Suffix Problem

To find the min suffix among $T[i\,..], \ldots, T[j\,..]$ in $O(1)$ time:

- Build *inverse suffix array*:
  $ISA[i] = k \Leftrightarrow T[i\,..]$ is the k-th largest suffix of T
- $ISA[m]$ is the minimum among $ISA[i], \ldots, ISA[j] \Rightarrow$
  $T[m\,..]$ is the minimal suffix among $T[i\,..], \ldots, T[j\,..]$
- $ISA[m]$ can be found in $O(1)$ time using RMQ

Example

T	b	a	a	b	a	c
ISA	4	1	2	5	3	6

$T[2\,..]$ is the minimal suffix among $T[1\,..], \ldots, T[5\,..]$

Minimal Suffix Problem

$T[m\,..\,]$ is the minimal suffix among $T[i\,..\,], T[i+1\,..\,], \ldots, T[j\,..\,]$

To find the shortest border of the $T[m\,..\,j]$ we use:

Theorem of T. Kociumaka et al., 2012

For any $\varepsilon > 0$ there is a linear space data structure that can be build in linear time and allows to find the shortest border of any substring of T in time $O(\log^{1+\varepsilon} |T|)$.

This gives a solution with $O(\log^{1+\varepsilon} |T|)$ time per query.

All needed data structures take linear space and can be built in linear time.

Maximal Suffix Problem

Maximal Suffix Problem

Let $T[m\,..\,]$ be the max suffix among $T[i\,..\,], T[i{+}1\,..\,], \ldots, T[j\,..\,]$

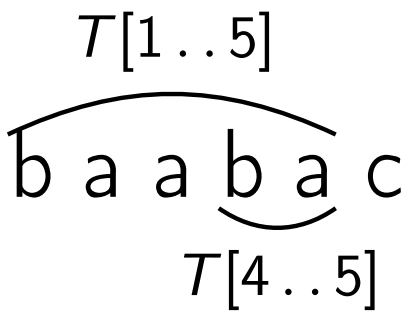
Lemma

The maximal suffix of $T[i\,..\,j]$ starts with $T[m\,..\,j]$.

Let $T[m_1\,..\,]$ be the max suffix among $T[i\,..\,], \ldots, T[m{-}1\,..\,]$

Lemma

$T[m_1\,..\,j] < T[m\,..\,j] \Rightarrow$ the maximal suffix of $T[i\,..\,j]$ is $T[m\,..\,j]$; otherwise the maximal suffix starts with $T[m_1\,..\,j]$.



Maximal Suffix Problem

$T[m\,..]$ is the maximal suffix among $T[i\,..], \ldots, T[j\,..]$

$T[m_1\,..]$ is the maximal suffix among $T[i\,..], \ldots, T[m-1\,..]$

The algorithm

1. Start with $T[m\,..]$
2. While $T[m_1\,..\,j] > T[m\,..\,j]$ reset $m := m_1$

Maximal Suffix Problem

$T[m\,..\,]$ is the maximal suffix among $T[i\,..\,], \ldots, T[j\,..\,]$

$T[m_1\,..\,]$ is the maximal suffix among $T[i\,..\,], \ldots, T[m-1\,..\,]$

The algorithm

1. Start with $T[m\,..\,]$
2. While $T[m_1\,..\,j] > T[m\,..\,j]$ reset $m := m_1$

Too many iterations might be possible.

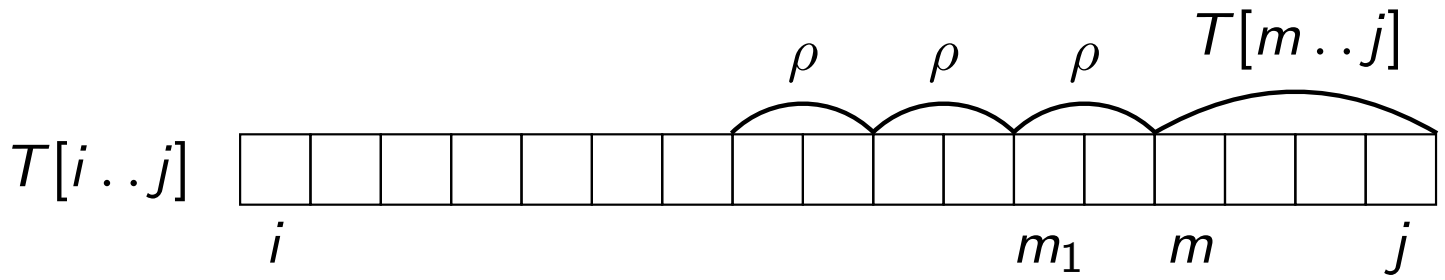
$T = a^n b$, the maximal suffix of $T[1\,..\,n]$ is a^n. Then

$$\begin{aligned} m &= n & T[m\,..\,] &= ab \\ m_1 &= n-1 & T[m_1\,..\,] &= aab \\ &\ldots & &\ldots \\ m_{n-1} &= 1 & T[m_{n-1}\,..\,] &= a^n b \end{aligned}$$

Maximal Suffix Problem

Types of steps:

- *Big:* the length of $T[m_1 \,..\, j]$ is greater than $\frac{3}{2}|T[m \,..\, j]| \Rightarrow$ set $m := m_1$ and proceed
- *Small:* the length of $T[m_1 \,..\, j]$ is less than $\frac{3}{2}|T[m \,..\, j]|$

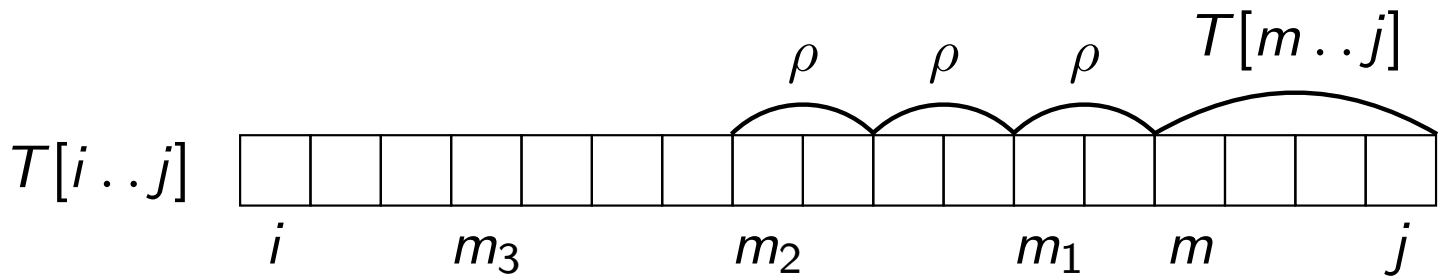


- $T[m \,..\, j]$ and $T[m_1 \,..\, j]$ overlap considerably and $T[m \,..\, j]$ is a prefix of $T[m_1 \,..\, j]$
- $T[m \,..\, j] = \rho^h \rho'$, where $\rho = T[m_1 \,..\, m-1]$
- If ρ^k occurs to the left of $T[m \,..\, j] \Rightarrow \rho^k T[m \,..\, j]$ is a prefix of the max suffix of $T[i \,..\, j]$

Maximal Suffix Problem

Types of steps:

- *Big:* the length of $T[m_1 \,..\, j]$ is greater than $\frac{3}{2}|T[m \,..\, j]| \Rightarrow$ set $m := m_1$ and proceed
- *Small:* the length of $T[m_1 \,..\, j]$ is less than $\frac{3}{2}|T[m \,..\, j]|$



Speeding up small steps:

1. Fast-forward to the longest suffix $T[m_2 \,..\, j] = \rho^k T[m \,..\, j]$
2. Let $T[m_3 \,..]$ be the max suffix among $T[i \,..], \ldots, T[m_2{-}1 \,..]$
3. If $T[m_3 \,..\, j] < T[m_2 \,..\, j]$, return $T[m_2 \,..\, j]$; otherwise $|T[m_3..j]| > \frac{3}{2}|T[m \,..\, j]|$, reset $m := m_3$ and proceed

Maximal Suffix Problem

To find the maximal suffix among $T[i\,..], T[i+1\,..], \ldots, T[j\,..]$ in $O(1)$ employ the range *maximum* queries over the inverse suffix array.

T	b	a	a	b	a	c
ISA	4	1	2	5	3	6

$$RMQ(1,5) = 4$$

Maximal Suffix Problem

To compare $T[m\,..\,j]$ and $T[m_1\,..\,j]$:

- Compute the length of longest common prefix of suffixes $T[m\,..\,]$ and $T[m_1\,..\,]$
- If it less than $|T[m\,..\,j]|$ and $|T[m_1\,..\,j]|$ — compare the first distinguished symbol
- Otherwise a shorter substring is less than a longer one

Maximal Suffix Problem

Let $\rho = T[m_1 \,..\, m-1]$.

To find the longest suffix of the form $\rho^k T[m\,..]$ in $O(1)$:

- Compute the longest common suffix of $T[\,..\,m_1-1]$ and $T[\,..\,m-1]$
- The longest common suffix of prefixes equals to the longest common prefix (LCP) of suffixes for the reversed T
- There exists a data structure to query LCP in $O(1)$

Maximal Suffix Problem

Query time

The step can be done in $O(1)$. Since the length of candidate increases by at least a factor of $3/2$ every step the total time to find the maximal suffix of $T[i\,..\,j]$ equals $O(\log(j-i))$.

Preprocessing time

We use suffix array, inverse suffix array, range maximum queries and longest common prefix structure. All these structures take linear space and can be constructed in linear time.

Results

Problem

Given a string T, find a minimal/maximal suffix of $T[i\,..\,j]$.

Results

	Space	Preprocessing time	Query Time
Minimal suffix	$O(n)$	$O(n)$	$O(\log^{1+\varepsilon} \lvert T \rvert)$
Maximal suffix	$O(n)$	$O(n)$	$O(\log(j-i))$

Further improvements

The minimal suffix problem can also be solved in $O(\log(j-i))$ query time (the preprocessing time and space are the same).

Thank you!

Questions?