Longest Unbordered Factor in Quasilinear Time

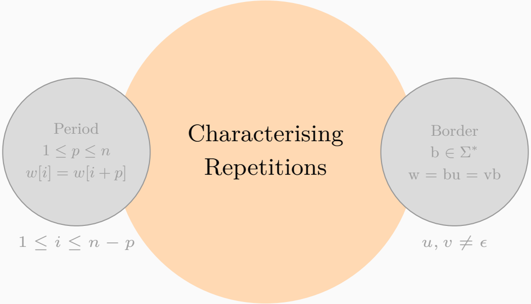
Tomasz Kociumaka **Ritu Kundu** Manal Mohamed Solon P. Pissis

December 19, 2018

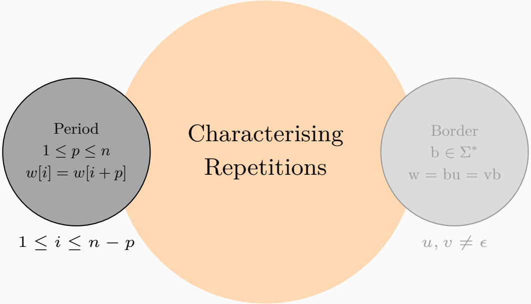
Presenting at:
29th International Symposium on Algorithms and Computation (ISAAC 2018)
Jiaoxi, Yilan, Taiwan

Outlines

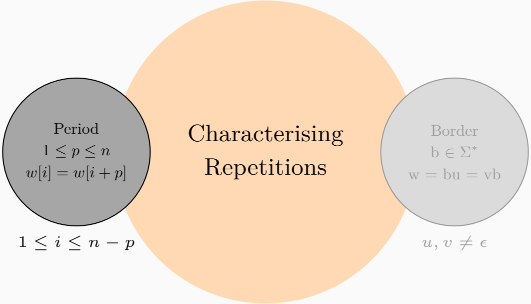
INTRODUCTION


Characterising Repetitions
Period
$1 \leq p \leq n$
$w[i] = w[i+p]$
$1 \leq i \leq n - p$
Border
$\mathrm{b} \in \Sigma^*$
$\mathrm{w} = \mathrm{bu} = \mathrm{vb}$
$u, v \neq \epsilon$

a	a	b	a	a	b	a	a


Period
$1 \leq p \leq n$
$w[i] = w[i+p]$
$1 \leq i \leq n - p$
Characterising
Repetitions
Border
$b \in \Sigma^*$
$w = bu = vb$
$u, v \neq \epsilon$

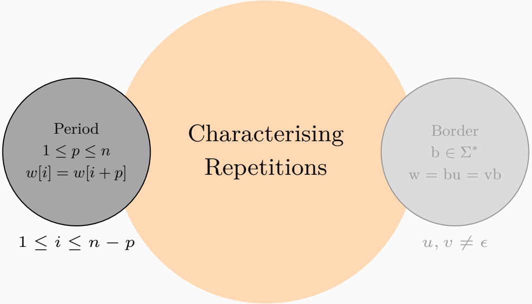
a	a	b	a	a	b	a	a

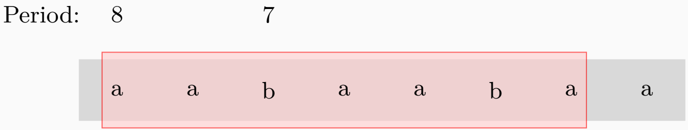


Period: 8

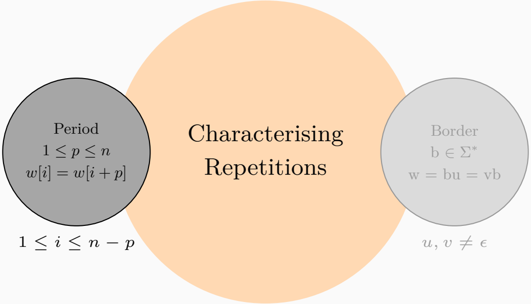
a a b a a b a a

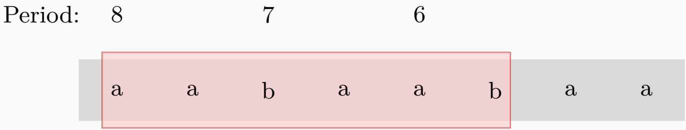
Introduction



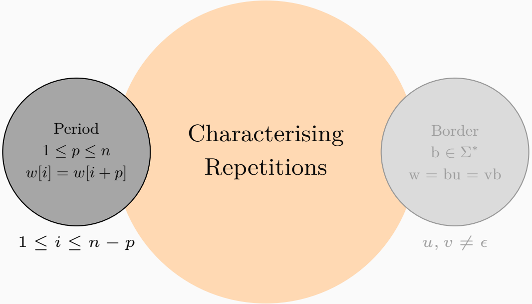


Introduction

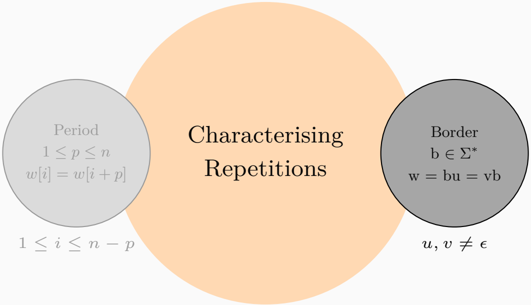




Introduction



Period:	8		7		6		3	
	a	a	b	a	a	b	a	a

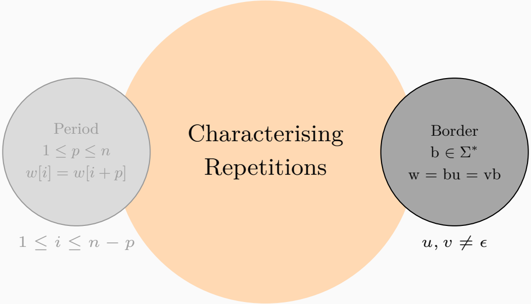


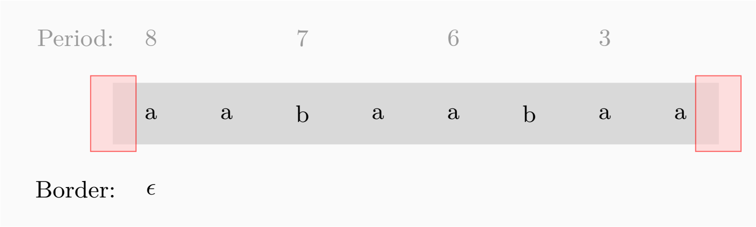
Period: 8 7 6 3

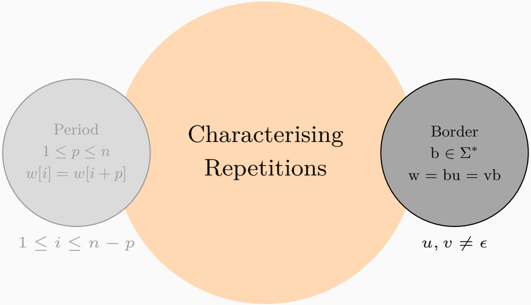
a	a	b	a	a	b	a	a

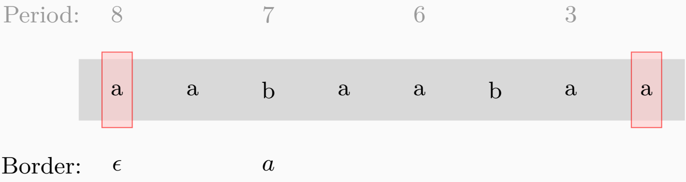
Border:

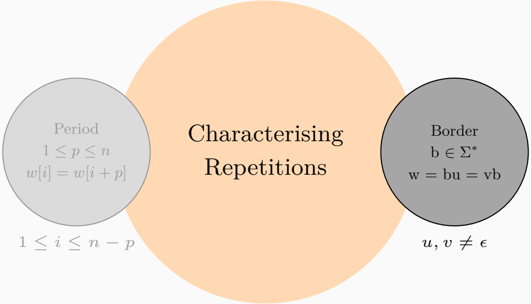
Introduction

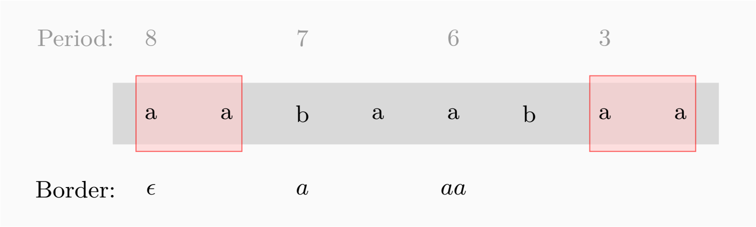




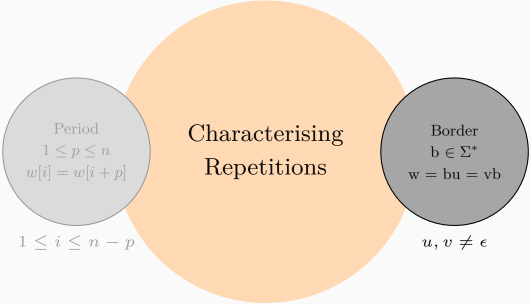

Period
$1 \leq p \leq n$
$w[i] = w[i+p]$
$1 \leq i \leq n-p$
Characterising Repetitions
Border
$b \in \Sigma^*$
$w = bu = vb$
$u, v \neq \epsilon$


Period: 8 7 6 3
a a b a a b a a
Border: ϵ a


Period
$1 \leq p \leq n$
$w[i] = w[i+p]$
$1 \leq i \leq n - p$
Characterising Repetitions
Border
$\mathrm{b} \in \Sigma^*$
$\mathrm{w} = \mathrm{bu} = \mathrm{vb}$
$u, v \neq \epsilon$

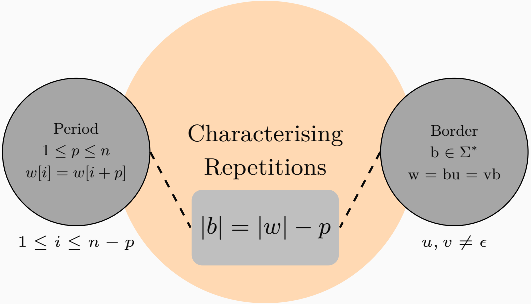

Period: 8 7 6 3
a a b a a b a a
Border: ϵ a aa

Introduction



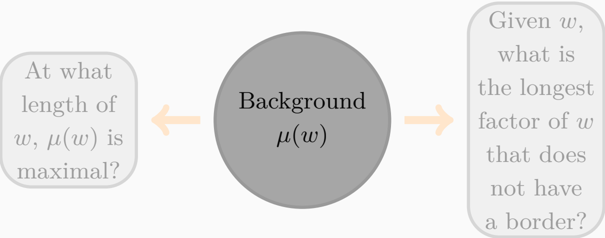
Period:	8		7		6		3	
	a	a	b	a	a	b	a	a
Border:	ϵ		a		aa		$aaabaa$	

Introduction



Period:	8		7		6		3	
	a	a	b	a	a	b	a	a
Border:	ϵ		a		aa		$aaabaa$	

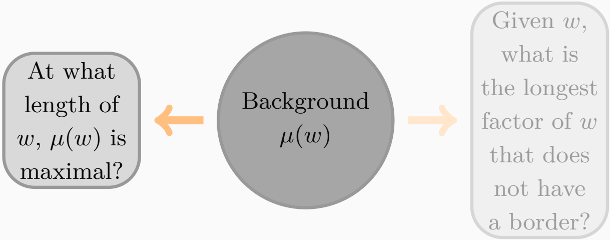
Motivation



Maximal(Longest) Unbordered Factor

- It is the longest factor of w which does not have a (non-empty) border; its length is usually represented by $\mu(w)$
- For the word $w =$ `baababa`, $\mu(w) = 5$.
- $\mu(w) \leq$ the minimal period of w.

Motivation



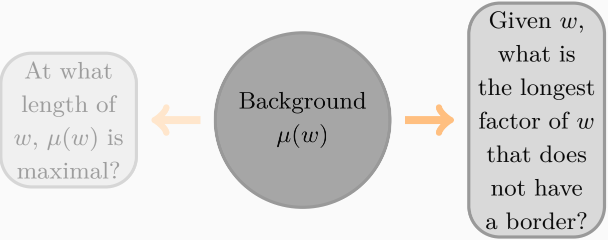
Ehrenfeucht and Silberger (1979)

⋮

Holub and Nowotka (2012)

- Asymptotically optimal upper bound ($\mu(w) \leq \frac{3}{7}n$)

Motivation



Loptev et al. (2015)

- first sub-quadratic-time (average case): $\mathcal{O}(n^2/\sigma^4)$)

Gawrychowski et al. (2015)

- Worst case $\mathcal{O}(n^{1.5})$
- $\mathcal{O}(n \log n)$ time on average
- Cording and Knudsen (2016) $\longrightarrow$ $\mathcal{O}(n)$-time [a] average-case using a refined bound on the expected length of the maximal unbordered factor

[a] improved in journal version (under review)

Our Contribution

Computing the Longest Unbordered Factor Array of a word over a general alphabet in $\mathcal{O}(n \log n)$ time with high probability.

The algorithm can also be implemented deterministically in $\mathcal{O}(n \log n \log^2 \log n)$ time.

Longest Unbordered Factor Array

Input: A word w of length n

Output: An array $\mathsf{LUF}[1..n]$ such that $\mathsf{LUF}[i]$ is the length of the maximal unbordered factor starting at position i in w, for all $1 \leq i \leq n$.

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
LUF[i]	20	3	12	9	12	3	14	3	11	3	10	5	2	3	5	2	2	2	2	1

a a b b a b a a b b a a b a b b a b a b

1 5 15 20

Longest Unbordered Factor Array

Input: A word w of length n

Output: An array $\mathsf{LUF}[1\,..\,n]$ such that $\mathsf{LUF}[i]$ is the length of the maximal unbordered factor starting at position i in w, for all $1 \leq i \leq n$.

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
LUF[i]	20	3	12	9	12	3	14	3	11	3	10	5	2	3	5	2	2	2	2	1

a a b b a b a a b b a a b a b b a b a b

1 5 15 20

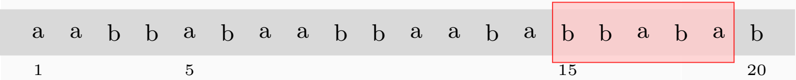
Longest Unbordered Factor Array

Input: A word w of length n

Output: An array $\mathsf{LUF}[1..n]$ such that $\mathsf{LUF}[i]$ is the length of the maximal unbordered factor starting at position i in w, for all $1 \leq i \leq n$.

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
LUF[i]	20	3	12	9	12	3	14	3	11	3	10	5	2	3	5	2	2	2	2	1



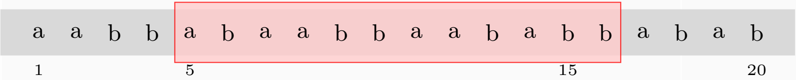
Longest Unbordered Factor Array

Input: A word w of length n

Output: An array $\mathsf{LUF}[1..n]$ such that $\mathsf{LUF}[i]$ is the length of the maximal unbordered factor starting at position i in w, for all $1 \leq i \leq n$.

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
LUF[i]	20	3	12	9	12	3	14	3	11	3	10	5	2	3	5	2	2	2	2	1

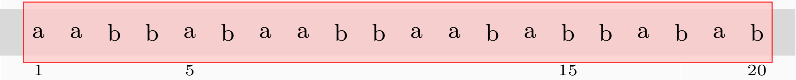


Longest Unbordered Factor Array

Input: A word w of length n
Output: An array $\mathsf{LUF}[1..n]$ such that $\mathsf{LUF}[i]$ is the length of the maximal unbordered factor starting at position i in w, for all $1 \leq i \leq n$.

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
LUF[i]	20	3	12	9	12	3	14	3	11	3	10	5	2	3	5	2	2	2	2	1



Longest Unbordered Factor Array

Input: A word w of length n
Output: An array $\mathsf{LUF}[1\,..\,n]$ such that $\mathsf{LUF}[i]$ is the length of the maximal unbordered factor starting at position i in w, for all $1 \leq i \leq n$.

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
LUF[i]	20	3	12	9	12	3	14	3	11	3	10	5	2	3	5	2	2	2	2	1

PRELIMINARIES

Duval (1982)

The shortest (non-empty) border of w is unique and unbordered.

Proposition: Duval (1982)

For any word w, there exists a unique sequence $(u_1, \cdots, u_k)$ of unbordered prefixes of w such that $w = u_k \cdots u_1$. Furthermore, the following properties hold:

(1) u_1 is the shortest border of w;

(2) u_k is the longest unbordered prefix of w;

(3) for all i, $1 \leq i \leq k$, u_i is an unbordered prefix of u_k.

unbordered-decomposition

The unique sequence described in the above proposition provides a unique unbordered-decomposition of a word.
$w = \mathsf{baababbabab} = \mathsf{baa} \cdot \mathsf{ba} \cdot \mathsf{b} \cdot \mathsf{ba} \cdot \mathsf{ba} \cdot \mathsf{b}$.

Longest Successor Factor (Length and Reference) Arrays

$$\mathsf{LSF}_\ell[i] = \begin{cases} 0 & \text{if} \quad i = n, \\ \max\{k \mid w[i\,..\,i+k-1] = w[j\,..\,j+k-1]\}, & \text{for} \quad i < j \leq n. \end{cases}$$

$$\mathsf{LSF}_r[i] = \begin{cases} nil & \text{if} \quad \mathsf{LSF}_\ell[i] = 0, \\ \max\{j \mid w[j\,..\,j+\mathsf{LSF}_\ell[i]-1] = w[i\,..\,i+\mathsf{LSF}_\ell[i]-1]\} & \text{for} \quad i < j \leq n. \end{cases}$$

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
$\mathsf{LSF}_\ell[i]$	5	6	5	4	3	4	3	4	3	2	1	4	3	2	1	3	2	1	0	0
$\mathsf{LSF}_r[i]$	7	14	15	16	17	10	11	14	15	18	19	17	18	19	20	18	19	20	nil	nil

Longest Successor Factor (Length and Reference) Arrays

7 11 16 18 19 20

a a b b a b a a b b a a b a b b a b a b

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
$\mathsf{LSF}_\ell[i]$	5	6	5	4	3	4	3	4	3	2	1	4	3	2	1	3	2	1	0	0
$\mathsf{LSF}_r[i]$	7	14	15	16	17	10	11	14	15	18	19	17	18	19	20	18	19	20	nil	nil

Computational Tools - 2

Longest Successor Factor (Length and Reference) Arrays

7 11 16 18 19 20

a a b b a b a a b b a a b a b b a b a b

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
$\mathsf{LSF}_\ell[i]$	5	6	5	4	3	4	3	4	3	2	1	4	3	2	1	3	2	1	0	0
$\mathsf{LSF}_r[i]$	7	14	15	16	17	10	11	14	15	18	19	17	18	19	20	18	19	20	nil	nil

Longest Successor Factor (Length and Reference) Arrays

7 11 16 18 19 20

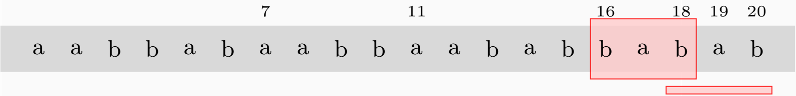
a a b b a b a a b b a a b a b b a b a b

Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
$\mathsf{LSF}_\ell[i]$	5	6	5	4	3	4	3	4	3	2	1	4	3	2	1	3	2	1	0	0
$\mathsf{LSF}_r[i]$	7	14	15	16	17	10	11	14	15	18	19	17	18	19	20	18	19	20	nil	nil

Computational Tools - 2

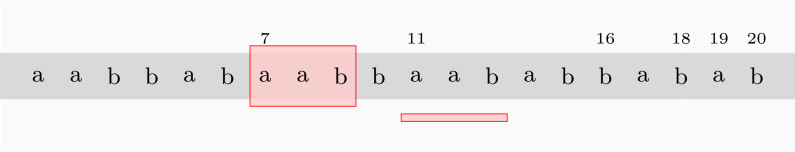
Longest Successor Factor (Length and Reference) Arrays



Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
$\mathsf{LSF}_\ell[i]$	5	6	5	4	3	4	3	4	3	2	1	4	3	2	1	3	2	1	0	0
$\mathsf{LSF}_r[i]$	7	14	15	16	17	10	11	14	15	18	19	17	18	19	20	18	19	20	nil	nil

Longest Successor Factor (Length and Reference) Arrays



Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
$\mathsf{LSF}_\ell[i]$	5	6	5	4	3	4	3	4	3	2	1	4	3	2	1	3	2	1	0	0
$\mathsf{LSF}_r[i]$	7	14	15	16	17	10	11	14	15	18	19	17	18	19	20	18	19	20	nil	nil

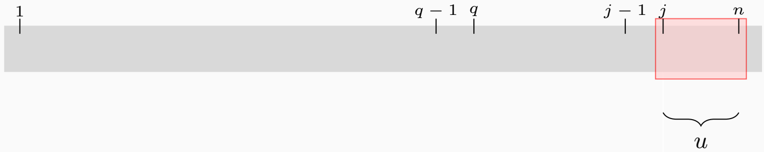
Hook Array ($\mathsf{HOOK}[1\,..\,n]$)

At each position j, $\mathsf{HOOK}[j]$ stores the smallest position q such that the factor $w[q\,..\,j-1]$ can be decomposed into unbordered prefixes of $w[j\,..\,n]$.

Hook Array ($\mathsf{HOOK}[1\,..\,n]$)

At each position j, $\mathsf{HOOK}[j]$ stores the smallest position q such that the factor $w[q\,..\,j-1]$ can be decomposed into unbordered prefixes of $w[j\,..\,n]$.

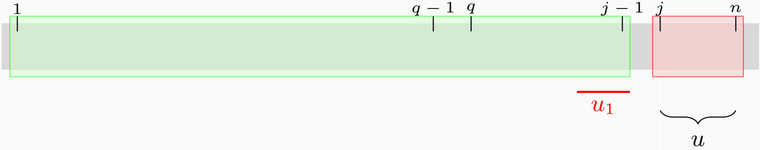
Greedy Construction



Hook Array (HOOK$[1 \mathinner{..} n]$)

At each position j, HOOK$[j]$ stores the smallest position q such that the factor $w[q \mathinner{..} j-1]$ can be decomposed into unbordered prefixes of $w[j \mathinner{..} n]$.

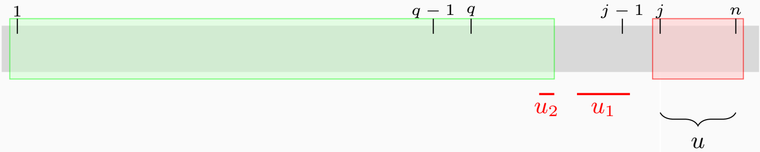
Greedy Construction



Hook Array (HOOK$[1 \,.\,.\, n]$)

At each position j, HOOK$[j]$ stores the smallest position q such that the factor $w[q \,.\,.\, j-1]$ can be decomposed into unbordered prefixes of $w[j \,.\,.\, n]$.

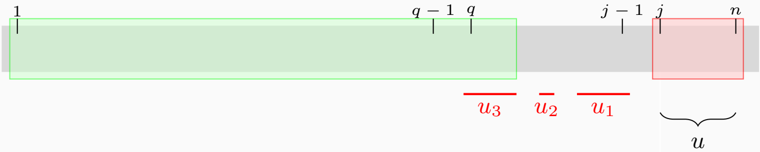
Greedy Construction



Hook Array (HOOK$[1\,..\,n]$)

At each position j, HOOK$[j]$ stores the smallest position q such that the factor $w[q\,..\,j-1]$ can be decomposed into unbordered prefixes of $w[j\,..\,n]$.

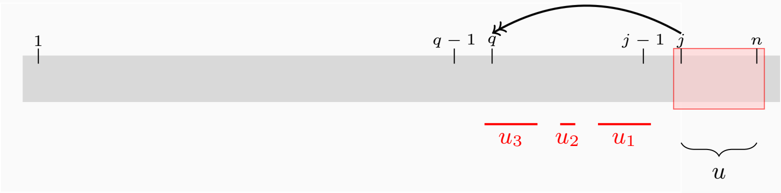
Greedy Construction



Hook Array (HOOK$[1\,..\,n]$)

At each position j, HOOK$[j]$ stores the smallest position q such that the factor $w[q\,..\,j-1]$ can be decomposed into unbordered prefixes of $w[j\,..\,n]$.

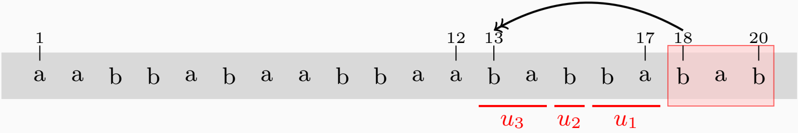
Greedy Construction



Combinatorial Tool

Hook Array (HOOK$[1\,..\,n]$)

At each position j, HOOK$[j]$ stores the smallest position q such that the factor $w[q\,..\,j-1]$ can be decomposed into unbordered prefixes of $w[j\,..\,n]$.



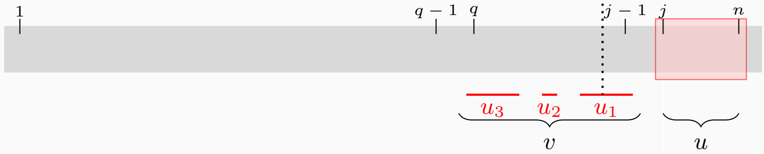
Example

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$w[i]$	a	a	b	b	a	b	a	a	b	b	a	a	b	a	b	b	a	b	a	b
HOOK$[i]$	1	1	3	3	5	3	7	1	9	3	11	11	13	1	15	13	17	13	17	20

Hook Array (HOOK$[1\,..\,n]$)

At each position j, HOOK$[j]$ stores the smallest position q such that the factor $w[q\,..\,j-1]$ can be decomposed into unbordered prefixes of $w[j\,..\,n]$.

Greedy Construction



Observation 1

The decomposition of v into unbordered prefixes of u is unique.

Observation 2

If v can be decomposed into unbordered prefixes of u, then every prefix of v also admits such a decomposition.

ALGORITHM

Case 1

If $\mathsf{LSF}_\ell[i] = 0$ then
$\mathsf{LUF}[i] = n - i + 1$, for $1 \leq i \leq n$.

Case 2

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] < \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = j + \mathsf{LUF}[j] - i$, for $1 \leq i \leq n$.

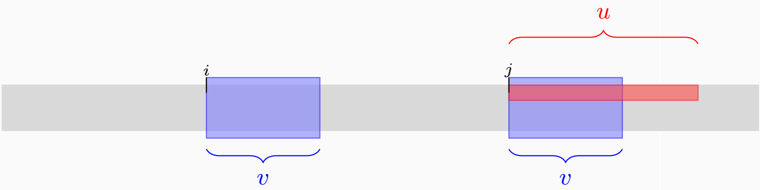
High Level View

Case 1

If $\mathsf{LSF}_\ell[i] = 0$ then
$\mathsf{LUF}[i] = n - i + 1$, for $1 \leq i \leq n$.

Case 2

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] < \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = j + \mathsf{LUF}[j] - i$, for $1 \leq i \leq n$.



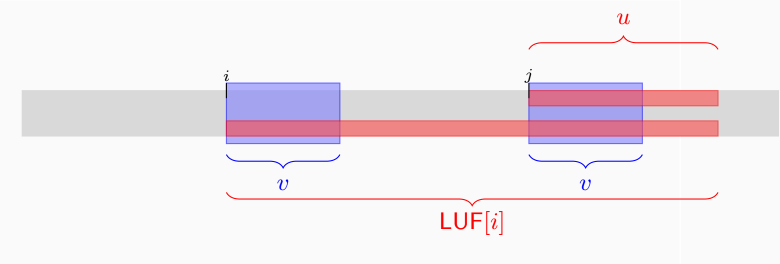
High Level View

Case 1

If $\mathsf{LSF}_\ell[i] = 0$ then
$\mathsf{LUF}[i] = n - i + 1$, for $1 \leq i \leq n$.

Case 2

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] < \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = j + \mathsf{LUF}[j] - i$, for $1 \leq i \leq n$.

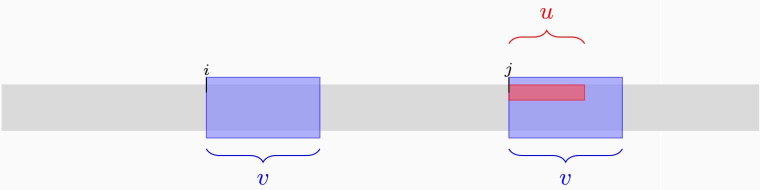


Case 3 (a)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{HOOK}[j] - i$ if $i < \mathsf{HOOK}[j]$, for $1 \leq i \leq n$.

Case 3 (b)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{LUF}[j]$, for $1 \leq i \leq n$.

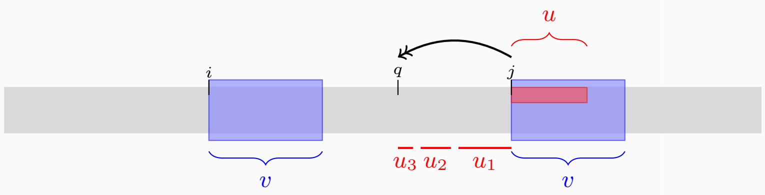


Case 3 (a)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{HOOK}[j] - i$ if $i < \mathsf{HOOK}[j]$, for $1 \leq i \leq n$.

Case 3 (b)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{LUF}[j]$, for $1 \leq i \leq n$.



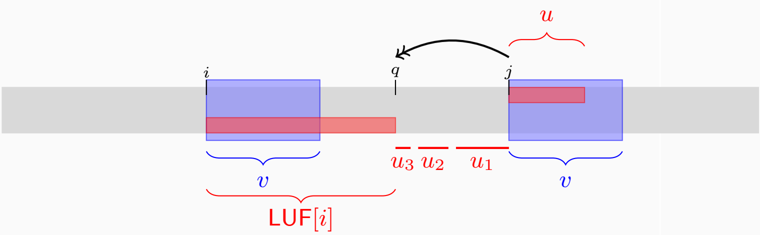
High Level View

Case 3 (a)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{HOOK}[j] - i$ if $i < \mathsf{HOOK}[j]$, for $1 \leq i \leq n$.

Case 3 (b)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{LUF}[j]$, for $1 \leq i \leq n$.



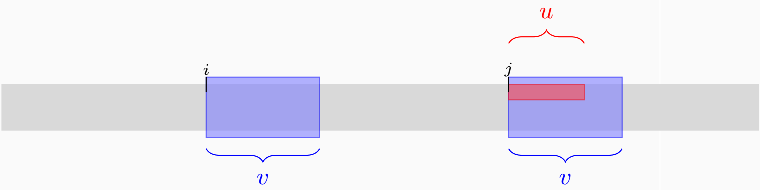
High Level View

Case 3 (a)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{HOOK}[j] - i$ if $i < \mathsf{HOOK}[j]$, for $1 \leq i \leq n$.

Case 3 (b)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathbf{LUF}[j]$, for $1 \leq i \leq n$.



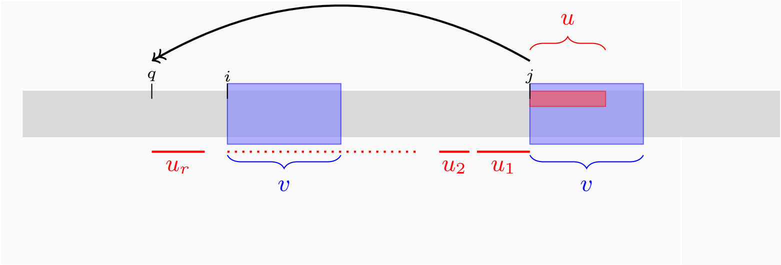
High Level View

Case 3 (a)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{HOOK}[j] - i$ if $i < \mathsf{HOOK}[j]$, for $1 \leq i \leq n$.

Case 3 (b)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathbf{LUF}[j]$, for $1 \leq i \leq n$.

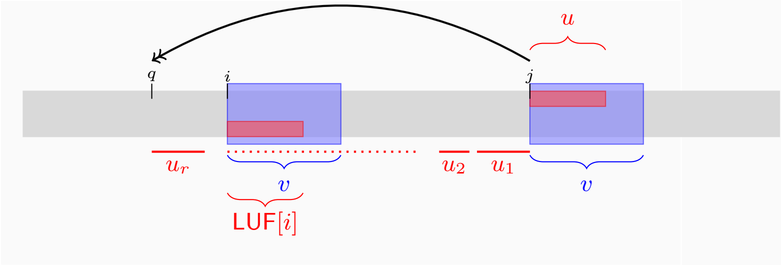


Case 3 (a)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{HOOK}[j] - i$ if $i < \mathsf{HOOK}[j]$, for $1 \leq i \leq n$.

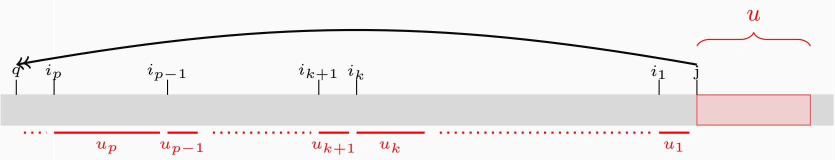
Case 3 (b)

If $\mathsf{LSF}_r[i] = j$ and $\mathsf{LSF}_\ell[i] \geq \mathsf{LUF}[j]$ then
$\mathsf{LUF}[i] = \mathsf{LUF}[j]$, for $1 \leq i \leq n$.



Finding Hook

Naive Construction



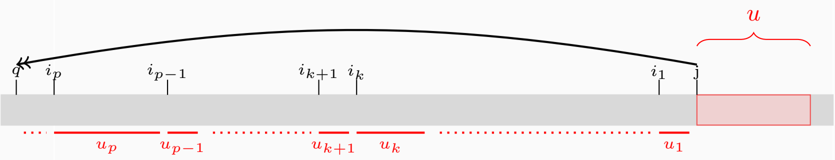
FindBeta Function

- Returns the length β of the shortest prefix of $w[j\,..\,n]$ that is a suffix of $w[1\,..\,q-1]$, or $\beta = 0$.
- Based on 'prefix-suffix queries' of Kociumaka et al. (2015, 2012): Given $d \in \mathbb{N}$; factors x & y of w, reports all prefixes of x of length between d and $2d$ that occur as suffixes of y.
- A single prefix-suffix query can be implemented in $\mathcal{O}(1)$ time after preprocessing of w which takes quasilinear[1] time.

[1] bottleneck; now solved; more later.

Finding Hook

Efficient Construction

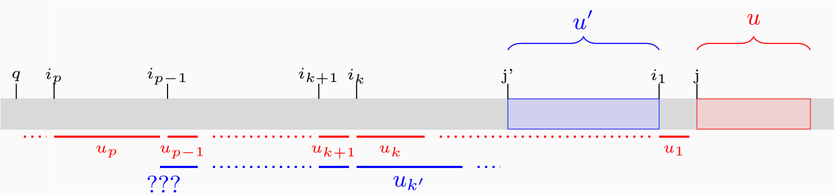


Observations

- In a chain, each u_k is unbordered. $\mathsf{LUF}[i_k] \geq |u_k| \Rightarrow \mathsf{HOOK}[i_k] \leq i_{p-1}$.
- Overlapping Chains.

Finding Hook

Efficient Construction



Observations

- In a chain, each u_k is unbordered. $\mathsf{LUF}[i_k] \geq |u_k| \Rightarrow \mathsf{HOOK}[i_k] \leq i_{p-1}$.
- Overlapping Chains.

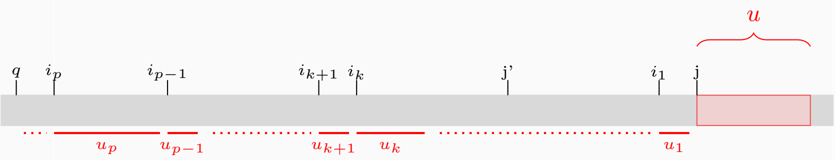
RECYCLE

Shift hook leftwards: Avoid computations between i_k and i_{p-1} w.r.t longer factors at i_k.

Genralised Hook: $\mathcal{H}_j^\ell$

$\mathcal{H}_j^0 = j$ and $\mathcal{H}_j^\ell = \mathcal{H}_j$ if $\ell \geq \mathsf{LUF}[j]$.

Efficient Construction



Implementation

- Right to left.
- Use a stack to keep track of the pairs (ℓ, i) for which the hooks $\mathcal{H}_i^\ell$ need to be determined.
- Update values in HOOK.

ANALYSIS

Twin Set

Purpose

- Correctness
- Running time analysis
- Efficient FindBeta

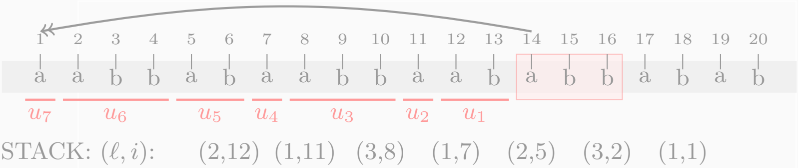
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j\mathinner{.\,.}\mathsf{LUF}[j]-1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set



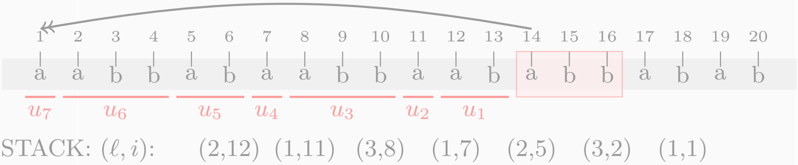
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j \mathinner{.\,.} \mathsf{LUF}[j]-1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set



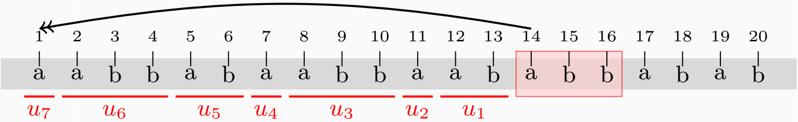
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j \mathinner{..} \mathsf{LUF}[j] - 1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set



STACK: (ℓ, i): (2,12) (1,11) (3,8) (1,7) (2,5) (3,2) (1,1)

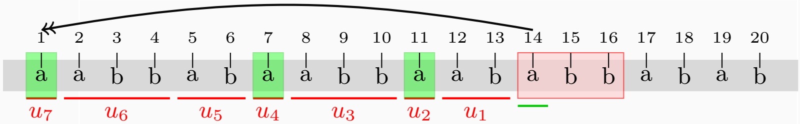
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j \mathinner{.\,.} \mathsf{LUF}[j]-1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set



STACK: (ℓ, i): (2,12) (1,11) (3,8) (1,7) (2,5) (3,2) (1,1)

$\mathcal{T}_{14}^{1} = \{11, 7, 1\}$

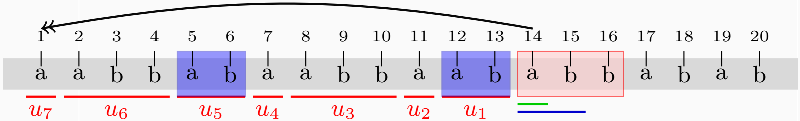
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell.$
- A unique shortest unbordered prefix of $w[j \mathinner{.\,.} \mathsf{LUF}[j] - 1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set



STACK: (ℓ, i): (2,12) (1,11) (3,8) (1,7) (2,5) (3,2) (1,1)

$\mathcal{T}_{14}^{1} = \{11, 7, 1\}$ $\mathcal{T}_{14}^{2} = \{12, 5\}$

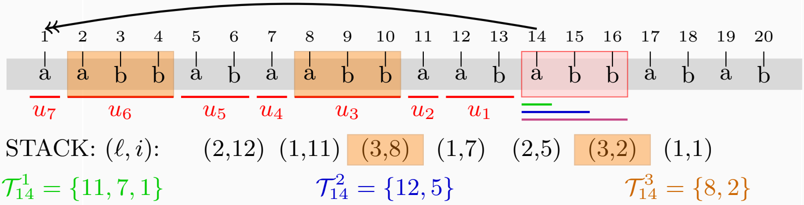
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j\mathinner{.\,.}\mathsf{LUF}[j]-1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set

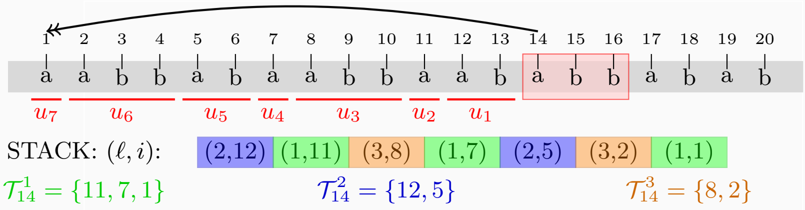


Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j \mathinner{.\,.} \mathsf{LUF}[j] - 1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base



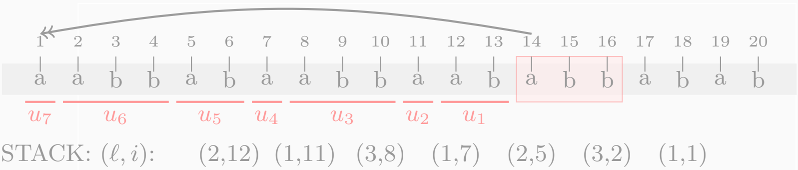
Definition: $\mathcal{T}^\ell_j$

$$\mathcal{T}^\ell_j = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}^\ell_j$.
- A unique shortest unbordered prefix of $w[j\,..\,\mathsf{LUF}[j]-1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Twin Set



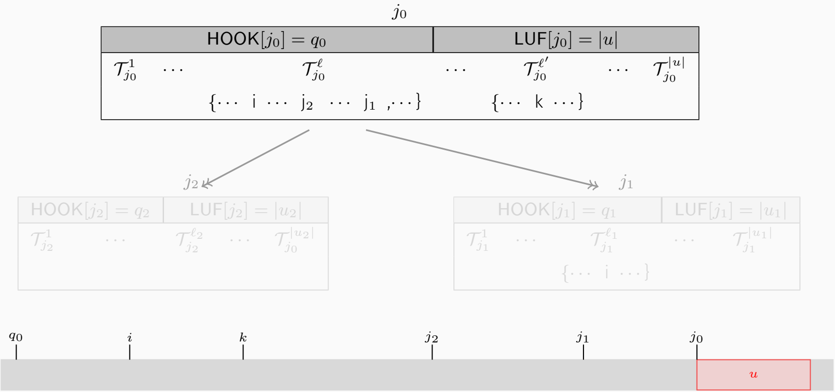
Definition: $\mathcal{T}_j^\ell$

$$\mathcal{T}_j^\ell = \{i \mid (\ell, i) \text{ was pushed onto the stack of } j\}.$$

Characteristics

- $\mathcal{S}_j = \bigcup_{\ell=1}^{\mathsf{LUF}[j]} \mathcal{T}_j^\ell$.
- A unique shortest unbordered prefix of $w[j \mathinner{..} \mathsf{LUF}[j]-1]$ occurs at each i belonging to the same twin set.
- Dynamic: Parent, Base

Behaviour

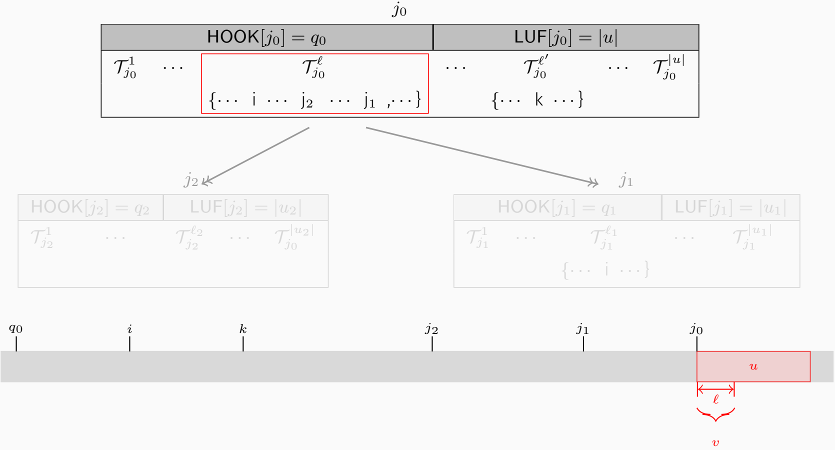


If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}^\ell_{j_0}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}^\ell_{j_0}$;

(2) there exists $k \in \mathcal{T}^{\ell'}_{j_0}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

Behaviour

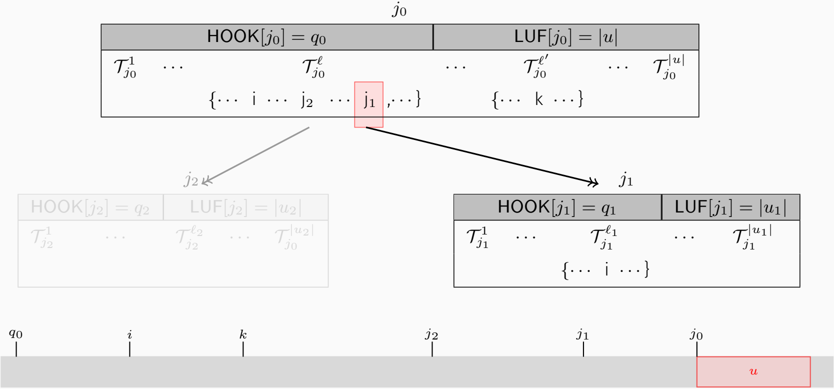


If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}^\ell_{j_0}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}^\ell_{j_0}$;

(2) there exists $k \in \mathcal{T}^{\ell'}_{j_0}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

Behaviour

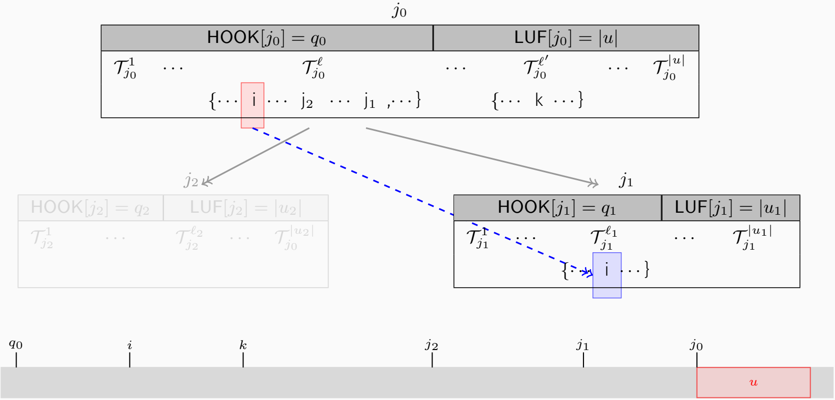


If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}^{\ell}_{j_0}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}^{\ell}_{j_0}$;

(2) there exists $k \in \mathcal{T}^{\ell'}_{j_0}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

Behaviour

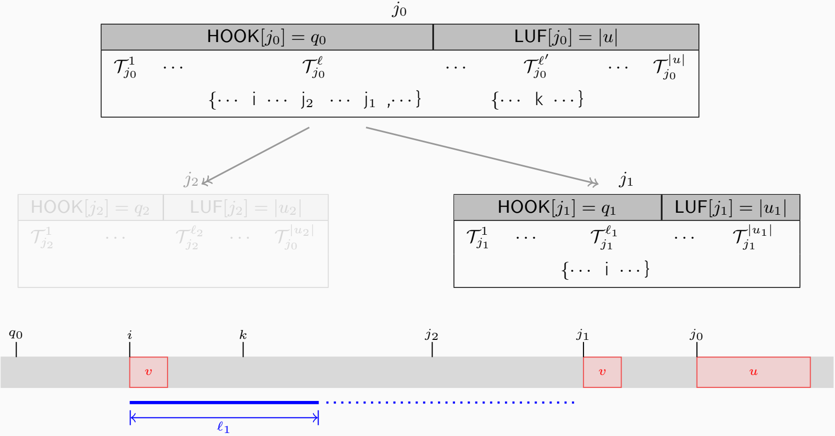


If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}^\ell_{j_0}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}^\ell_{j_0}$;

(2) there exists $k \in \mathcal{T}^{\ell'}_{j_0}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

Behaviour

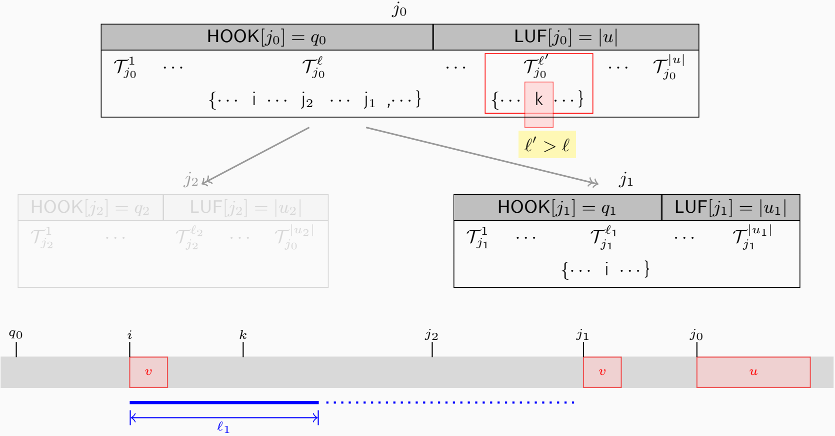


If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}_{j_0}^{\ell}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}_{j_0}^{\ell}$;

(2) there exists $k \in \mathcal{T}_{j_0}^{\ell'}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

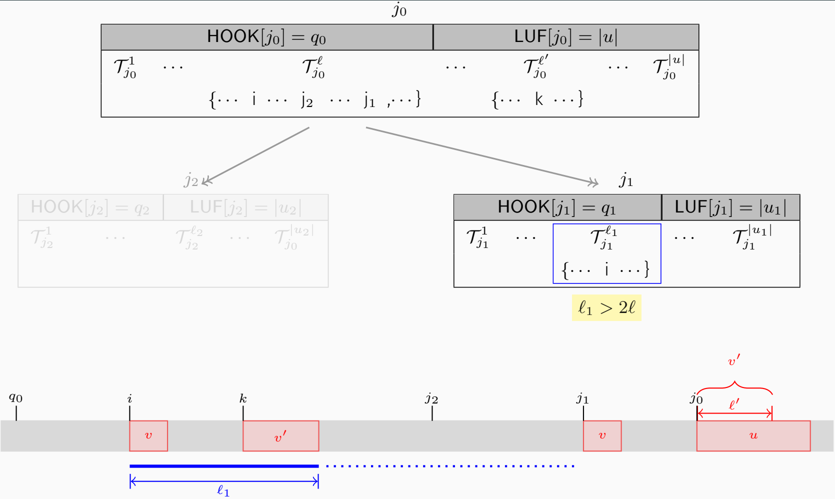
Behaviour



If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}^{\ell}_{j_0}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}^{\ell}_{j_0}$;

(2) there exists $k \in \mathcal{T}^{\ell'}_{j_0}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

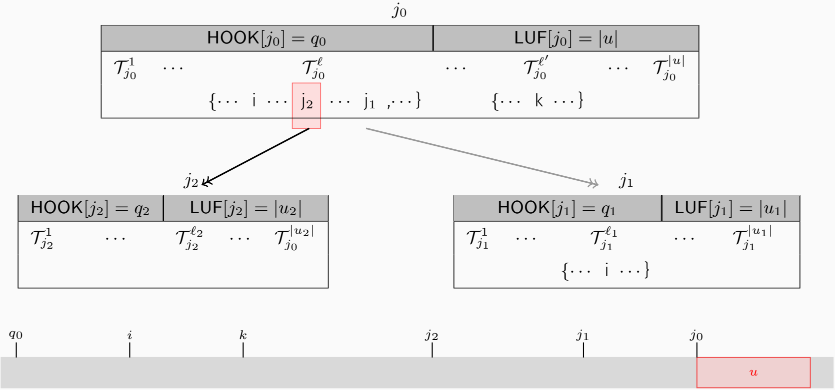


If j_0 and j_1 are two references such that j_0 is the parent of j_1 and $j_1 \in \mathcal{T}^\ell_{j_0}$, then each position $i \in \mathcal{S}_{j_1}$ satisfies the following properties:

(1) $i \in \mathcal{T}^\ell_{j_0}$;

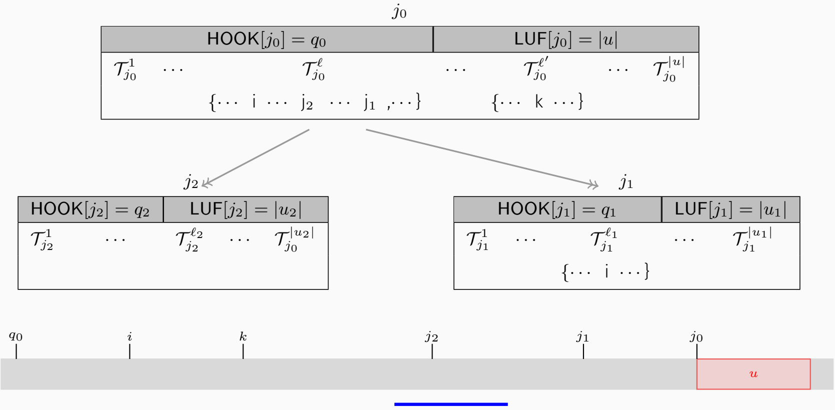
(2) there exists $k \in \mathcal{T}^{\ell'}_{j_0}$, with $\ell' > \ell$, such that $(k + \ell' - i, i)$ is pushed onto the stack of j_1.

Behaviour



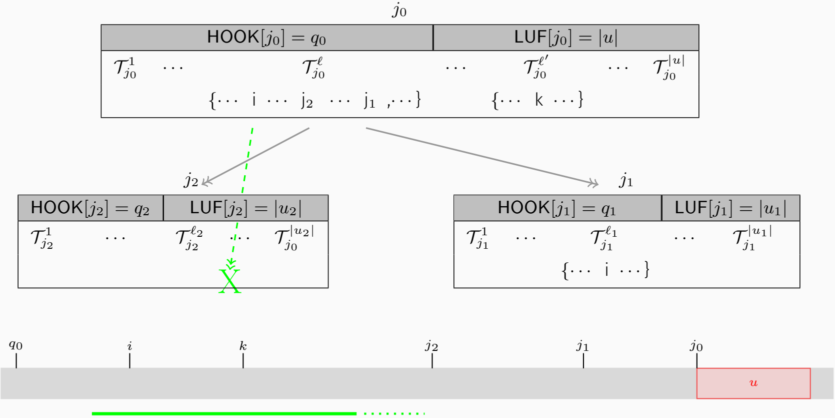
If j_0 is the parent of two references $j_2 < j_1$, both of which belong to $\mathcal{T}^\ell_{j_0}$, then $\mathcal{S}_{j_1} \cap \mathcal{S}_{j_2} = \emptyset$.

Behaviour



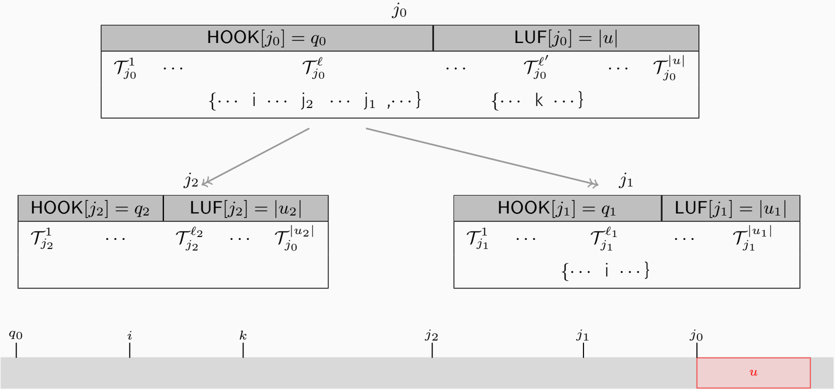
If j_0 is the parent of two references $j_2 < j_1$, both of which belong to $\mathcal{T}_{j_0}^{\ell}$, then $\mathcal{S}_{j_1} \cap \mathcal{S}_{j_2} = \emptyset$.

Behaviour



If j_0 is the parent of two references $j_2 < j_1$, both of which belong to $\mathcal{T}_{j_0}^{\ell}$, then $\mathcal{S}_{j_1} \cap \mathcal{S}_{j_2} = \emptyset$.

Behaviour



If $j_2 < j_1$ are two base references then $\mathcal{S}_{j_1} \cap \mathcal{S}_{j_2} = \emptyset$.

Analysis

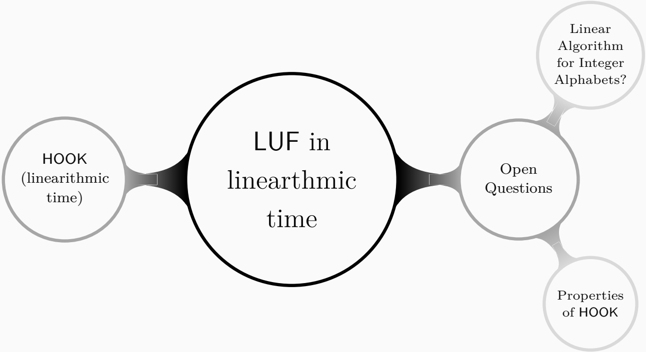
- The total size of all the stacks used throughout the algorithm is $\mathcal{O}(n \log n)$.
- The total running time of the FindBeta function is $\mathcal{O}(n \log n)$
  - Start from $d = 2\ell$ for prefix-suffix queries if the reference's parent twin-set is of length $= \ell$.

Given a word w of length n, our algorithm solves the Longest Unbordered Factor Array problem in $\mathcal{O}(n \log n)$ time with high probability. It can also be implemented deterministically in $\mathcal{O}(n \log n \log^2 \log n)$ time. [1]

[1] Update: Deterministically in $\mathcal{O}(n \log n)$ after the proposed linear time construction of the data structure to answer constant-time prefix-suffix query in Kociumaka (2018).

SUMMARY

Summary


HOOK (linearithmic time)
LUF in linearthmic time
Open Questions
Linear Algorithm for Integer Alphabets?
Properties of HOOK

Patrick Hagge Cording and Mathias Bæk Tejs Knudsen. Maximal unbordered factors of random strings. In Shunsuke Inenaga, Kunihiko Sadakane, and Tetsuya Sakai, editors, String Processing and Information Retrieval - 23rd International Symposium, SPIRE 2016, Beppu, Japan, October 18-20, 2016, Proceedings, volume 9954 of Lecture Notes in Computer Science, pages 93–96, 2016. doi: 10.1007/978-3-319-46049-9_9. URL https://doi.org/10.1007/978-3-319-46049-9_9.

Jean-Pierre Duval. Relationship between the period of a finite word and the length of its unbordered segments. Discrete Mathematics, 40(1):31–44, 1982. doi: 10.1016/0012-365X(82)90186-8. URL https://doi.org/10.1016/0012-365X(82)90186-8.

Andrzej Ehrenfeucht and D. M. Silberger. Periodicity and unbordered segments of words. Discrete Mathematics, 26(2):101–109, 1979. doi: 10.1016/0012-365X(79)90116-X. URL https://doi.org/10.1016/0012-365X(79)90116-X.

Pawel Gawrychowski, Gregory Kucherov, Benjamin Sach, and Tatiana A. Starikovskaya. Computing the longest unbordered substring. In Costas S. Iliopoulos, Simon J. Puglisi, and Emine Yilmaz, editors, String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings, volume 9309 of Lecture Notes in Computer Science, pages 246–257. Springer, 2015. ISBN 978-3-319-23825-8. doi: 10.1007/978-3-319-23826-5_24. URL https://doi.org/10.1007/978-3-319-23826-5_24.

Stepan Holub and Dirk Nowotka. The Ehrenfeucht-Silberger problem. J. Comb. Theory, Ser. A, 119(3):668–682, 2012. doi: 10.1016/j.jcta.2011.11.004. URL https://doi.org/10.1016/j.jcta.2011.11.004.

Tomasz Kociumaka. Efficient Data Structures for Internal Queries in Texts. PhD thesis, University of Warsaw, 2018. URL https://mimuw.edu.pl/~kociumaka/files/phd.pdf.

Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter, and Tomasz Walen. Efficient data structures for the factor periodicity problem. In Liliana Calderón-Benavides, Cristina N. González-Caro, Edgar Chávez, and Nivio Ziviani, editors, String Processing and Information Retrieval - 19th International Symposium, SPIRE 2012, Cartagena de Indias, Colombia, October 21-25, 2012. Proceedings, volume 7608 of Lecture Notes in Computer Science, pages 284–294. Springer, 2012. ISBN 978-3-642-34108-3. doi: 10.1007/978-3-642-34109-0_30. URL https://doi.org/10.1007/978-3-642-34109-0_30.

Thank You!

[Contact: ritu.kundu@kcl.ac.uk]