Left Lyndon tree Construction

Golnaz Badkobeh[1] and Maxime Crochemore[2,3]

1 Goldsmiths University of London, UK
2 King's College London, UK
3 Université Gustave Eiffel, France

Combinatorics on words seminar

June 2021

Definitions

A word (string) is a sequence of symbols, e.g. `abbaaba`

Definition

Lexicographic order:

Definitions

A word (string) is a sequence of symbols, e.g. `abbaaba`

Definition

Lexicographic order:

- $u \ll v$ if $u = ras$, $v = rbt$
  for words r, s and t and letters a and b with and $a < b$.

Definitions

A word (string) is a sequence of symbols, e.g. `abbaaba`

Definition

Lexicographic order:

- $u \ll v$ if $u = ras$, $v = rbt$
  for words r, s and t and letters a and b with and $a < b$.
- $u < v$, word u is smaller than word v,
  if either $u \ll v$ or u is a proper prefix of v.

Definitions

A word (string) is a sequence of symbols, e.g. `abbaaba`

Definition

Lexicographic order:

- $u \ll v$ if $u = ras$, $v = rbt$
  for words r, s and t and letters a and b with and $a < b$.
- $u < v$, word u is smaller than word v,
  if either $u \ll v$ or u is a proper prefix of v.

Definition

Infinite order:

- $u \prec v$ if $u^\infty < v^\infty$ or both $u^\infty = v^\infty$ and $|u| > |v|$.
  [*Dolce*, *Restivo*, *Reutenauer*, 2019].

Definitions

A word (string) is a sequence of symbols, e.g. `abbaaba`

Definition

Lexicographic order:

- $u \ll v$ if $u = ras$, $v = rbt$
  for words r, s and t and letters a and b with and $a < b$.
- $u < v$, word u is smaller than word v,
  if either $u \ll v$ or u is a proper prefix of v.

Definition

Infinite order:

- $u \prec v$ if $u^\infty < v^\infty$ or both $u^\infty = v^\infty$ and $|u| > |v|$.
  [*Dolce*, *Restivo*, *Reutenauer*, 2019].

$ab < aba$ but $ab^\infty = abab \cdots > (aba)^\infty = abaaba \cdots$

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$
2. $w < v$

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$
2. $w < v$
3. $u^\infty < w^\infty$

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$
2. $w < v$
3. $u^\infty < w^\infty$

Binary Lyndon words:

- `a`, `b`, `ab`, `aab`, `abb`, ...

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$
2. $w < v$
3. $u^\infty < w^\infty$

Binary Lyndon words:

- a, b, ab, aab, abb, . . .

Not Lyndon words:

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$
2. $w < v$
3. $u^\infty < w^\infty$

Binary Lyndon words:

- `a`, `b`, `ab`, `aab`, `abb`, ...

Not Lyndon words:

- $w =$ `abbab`: for $u =$ `abb` and $v =$ `ab` we get $w > vu =$ `ababb`.

.

Lyndon Words

Theorem (Lyndon 1954, Ufnarovskij 2011)

A Lyndon word w is either a singleton or defined by any of the following equivalent conditions, in which uv is any non-trivial factorisation of w:

1. $w < vu$
2. $w < v$
3. $u^\infty < w^\infty$

Binary Lyndon words:

- a, b, ab, aab, abb, ...

Not Lyndon words:

- $w =$ abbab: for $u =$ abb and $v =$ ab we get $w > vu =$ ababb.
- $w =$ abab: for $u =$ ab we get $(\mathtt{ab})^\infty = (\mathtt{abab})^\infty$.

Motivation

- Words are structured by their Lyndon factors.
- Lyndon factorisation: a word factorises uniquely into a decreasing sequence of Lyndon words.
- Can be viewed as a preprocessing step to word algorithms.

Motivation

- Words are structured by their Lyndon factors.
- Lyndon factorisation: a word factorises uniquely into a decreasing sequence of Lyndon words.
- Can be viewed as a preprocessing step to word algorithms.
- Help discovering maximal periodicities in words.
  e.g. [*Banai et al.*, 2017], [*C. et al.*, 2012].

Motivation

- Words are structured by their Lyndon factors.
- Lyndon factorisation: a word factorises uniquely into a decreasing sequence of Lyndon words.
- Can be viewed as a preprocessing step to word algorithms.
- Help discovering maximal periodicities in words.
  e.g. [*Banai et al.*, 2017], [*C. et al.*, 2012].
- Help sorting word suffixes to create a Suffix Array.
  e.g. [*Mantaci et al.*, 2013], [*Louza et al.*, 2019].

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \texttt{ababbababbabac}$ on the alphabet of constant letters $\{\texttt{a}, \texttt{b}, \dots\}$ ordered as usual $\texttt{a} < \texttt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$														

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \texttt{ababbababbabac}$ on the alphabet of constant letters $\{\texttt{a}, \texttt{b}, \dots\}$ ordered as usual $\texttt{a} < \texttt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$	1													

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \mathtt{ababbababbabac}$ on the alphabet of constant letters $\{\mathtt{a}, \mathtt{b}, \dots\}$ ordered as usual $\mathtt{a} < \mathtt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$	1	2												

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \mathtt{ababbababbabac}$ on the alphabet of constant letters $\{\mathtt{a}, \mathtt{b}, \ldots\}$ ordered as usual $\mathtt{a} < \mathtt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$	1	2	1											

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \texttt{ababbababbabac}$ on the alphabet of constant letters $\{\texttt{a}, \texttt{b}, \dots\}$ ordered as usual $\texttt{a} < \texttt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$	1	2	1	2										

Lyndon Suffix Table

Table *LynS* of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \texttt{ababbababbabac}$ on the alphabet of constant letters $\{\texttt{a}, \texttt{b}, \dots\}$ ordered as usual $\texttt{a} < \texttt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$	1	2	1	2	5									

Lyndon Suffix Table

Table $LynS$ of a word y is defined, for each position j on y, by

$$LynS[j] = \max\{|w| \mid w \text{ Lyndon suffix of } y[0\,..\,j]\}.$$

Example

Let $y = \texttt{ababbababbabac}$ on the alphabet of constant letters $\{\texttt{a}, \texttt{b}, \dots\}$ ordered as usual $\texttt{a} < \texttt{b} < \cdots$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
$LynS_y[j]$	1	2	1	2	5	1	2	1	2	5	1	2	1	14

Properties of Lyndon Words

- Let z be a word and a a letter for which za is a prefix of a Lyndon word. Let b be a letter with $a < b$. Then zb is a Lyndon word.

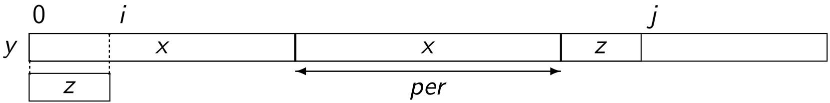
Properties of Lyndon Words

- Let z be a word and a a letter for which za is a prefix of a Lyndon word. Let b be a letter with $a < b$. Then zb is a Lyndon word.
- Let u and v be two Lyndon words with $u < v$, then uv is Lyndon word.

Properties of Lyndon Words

- Let z be a word and a a letter for which za is a prefix of a Lyndon word. Let b be a letter with $a < b$. Then zb is a Lyndon word.
- Let u and v be two Lyndon words with $u < v$, then uv is Lyndon word.

Invariant of *LynS* computation: $w = x^e z$
where x is a Lyndon word and z a proper prefix of x.



If $y[j] > y[i]$ then $y[0 \,.\,.\, j]$ is a Lyndon word **with period** $j + 1$.
[*Duval*, 1983].

Lyndon Suffix Table - Algorithm

LyndonSuffix(y Lyndon word of length n)

```
LynS[0] ← 1
(per, i) ← (1, 0)
for j ← 1 to n − 1 do
        if y[j] ≠ y[i] then                ▷ y[j] > y[i] = y[j − per]
                LynS[j] ← j + 1
                (per, i) ← (j + 1, 0)
        else    LynS[j] ← LynS[i]
                i ← i + 1 mod per
return LynS
```

Lyndon Suffix Table - Algorithm

LyndonSuffix(y Lyndon word of length n)

$LynS[0] \leftarrow 1$
$(per, i) \leftarrow (1, 0)$
for $j \leftarrow 1$ **to** $n - 1$ **do**
  if $y[j] \neq y[i]$ **then** $\triangleright\ y[j] > y[i] = y[j - per]$
    $LynS[j] \leftarrow j + 1$
    $(per, i) \leftarrow (j + 1, 0)$
  else $LynS[j] \leftarrow LynS[i]$
    $i \leftarrow i + 1 \bmod per$
return $LynS$

Proposition

Algorithm LyndonSuffix *computes the Lyndon suffix table of a Lyndon word of length n in time $O(n)$.*

Left Lyndon tree of a Lyndon word

Let y be a Lyndon word.

- Let u be the longest proper Lyndon prefix of y and $y = uv$. Then v is a Lyndon word.
  uv is the **left Lyndon factorisation** of y.
  aaaababbaabaab = aaaababbaab · aab

Left Lyndon tree of a Lyndon word

Let y be a Lyndon word.

- Let u be the longest proper Lyndon prefix of y and $y = uv$. Then v is a Lyndon word.
  uv is the **left Lyndon factorisation** of y.
  $\mathtt{aaaababbaabaab} = \mathtt{aaaababbaab} \cdot \mathtt{aab}$
- Let v be the longest proper Lyndon suffix of y and $y = uv$. Then u is a Lyndon word. uv is the **right Lyndon factorisation** of y.
  $\mathtt{aaaababbaabaab} = \mathtt{a} \cdot \mathtt{aaababbaabaab}$

Left Lyndon tree of a Lyndon word

Let y be a Lyndon word.

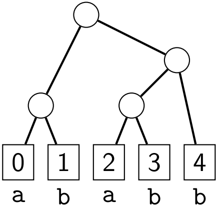
- Let u be the longest proper Lyndon prefix of y and $y = uv$. Then v is a Lyndon word.
  uv is the **left Lyndon factorisation** of y.
  $\mathtt{aaaababbaabaab} = \mathtt{aaaababbaab} \cdot \mathtt{aab}$
- Let v be the longest proper Lyndon suffix of y and $y = uv$. Then u is a Lyndon word. uv is the **right Lyndon factorisation** of y.
  $\mathtt{aaaababbaabaab} = \mathtt{a} \cdot \mathtt{aaababbaabaab}$

Left Lyndon tree of y:

- Obtained by recursive application of left Lyndon factorisation:
  $\mathtt{ababb} = \mathtt{ab} \cdot \mathtt{abb} = (\mathtt{a} \cdot \mathtt{b}) \cdot (\mathtt{ab} \cdot \mathtt{b}) = (\mathtt{a} \cdot \mathtt{b}) \cdot ((\mathtt{a} \cdot \mathtt{b}) \cdot \mathtt{b})$

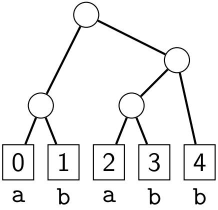
Left, right Lyndon trees

Left Lyndon tree of ababb:

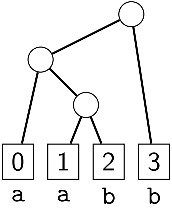


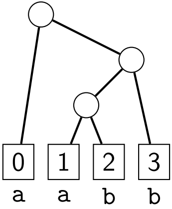
Left, right Lyndon trees

Left Lyndon tree of ababb:



Left and right Lyndon trees of aabb:





Left Lyndon Tree - Algorithm

```
LeftLyndonTree(y Lyndon word of length n)
(LynS[0], root[0]) ← (1, 0)
(per, i) ← (1, 0)
for j ← 1 to n − 1 do
    root[j] ← j
    if y[j] ≠ y[i] then                ▷ y[j] > y[i] = y[j − per]
        LynS[j] ← j + 1
        (per, i) ← (j + 1, 0)
    else  LynS[j] ← LynS[i]
        i ← i + 1 mod per
    (ℓ, k) ← (1, j − 1)
    while ℓ < LynS[j] do
        q ← new node ≥ n
        (left[q], right[q]) ← (root[k], root[j])
        root[j] ← q
        (ℓ, k) ← (ℓ + LynS[k], k − LynS[k])
return root[n − 1]
```

Left Lyndon Tree - Algorithm

```
LeftLyndonTree(y Lyndon word of length n)
(LynS[0], root[0]) ← (1, 0)
(per, i) ← (1, 0)
for j ← 1 to n − 1 do
        root[j] ← j
        if y[j] ≠ y[i] then                ▷ y[j] > y[i] = y[j − per]
                LynS[j] ← j + 1
                (per, i) ← (j + 1, 0)
        else    LynS[j] ← LynS[i]
                i ← i + 1 mod per
        (ℓ, k) ← (1, j − 1)
        while ℓ < LynS[j] do
                q ← new node ≥ n
                (left[q], right[q]) ← (root[k], root[j])
                root[j] ← q
                (ℓ, k) ← (ℓ + LynS[k], k − LynS[k])
return root[n − 1]
```

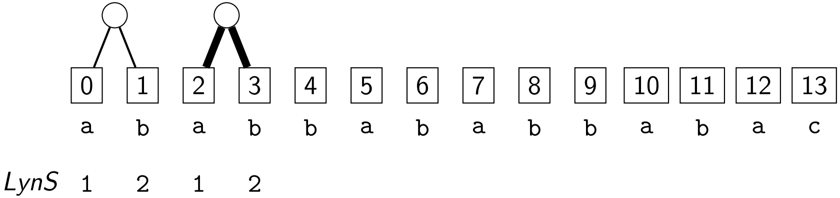
Theorem

Algorithm LeftLyndonTree *builds the left Lyndon tree of a Lyndon word of length n in time $O(n)$.*

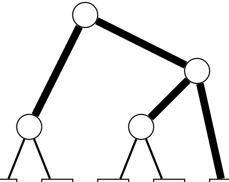
Building the tree

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	a	b	a	b	b	a	b	a	b	b	a	b	a	c
LynS	1	2												

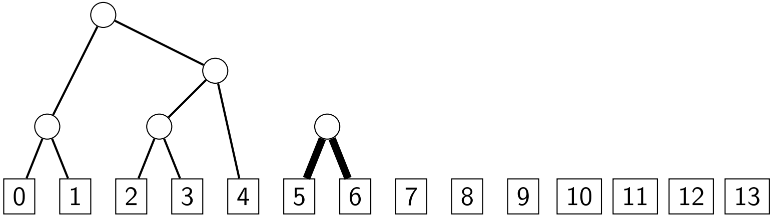
Building the tree



Building the tree

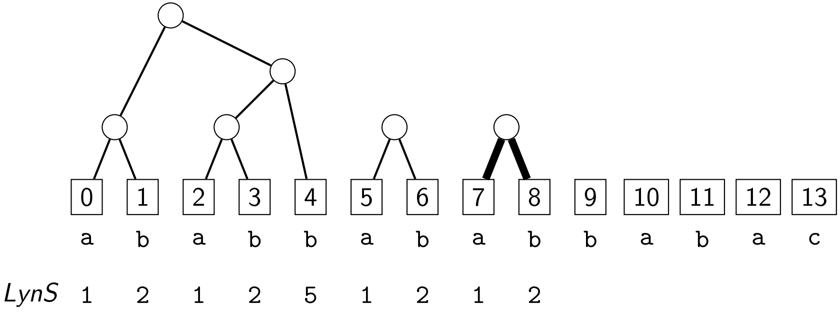


	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	a	b	a	b	b	a	b	a	b	b	a	b	a	c
LynS	1	2	1	2	5									

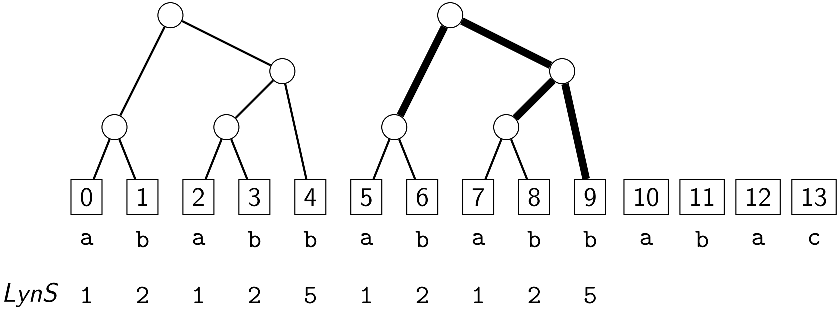


	a	b	a	b	b	a	b	a	b	b	a	b	a	c
LynS	1	2	1	2	5	1	2							

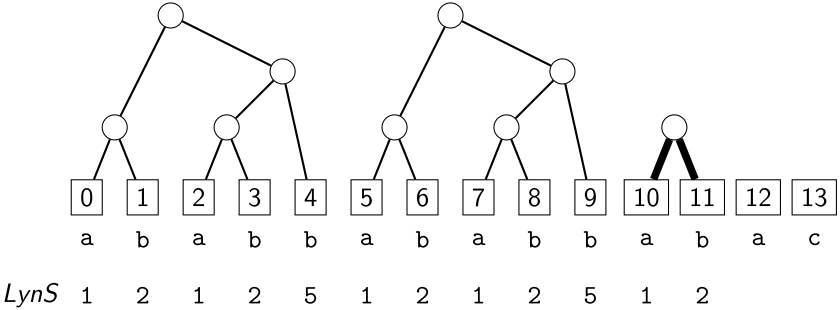
Building the tree



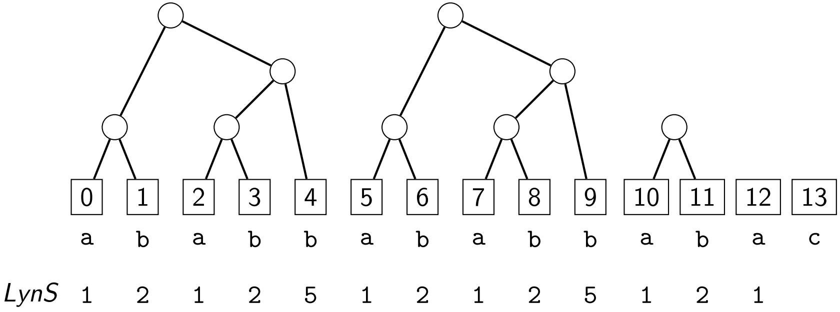
Building the tree



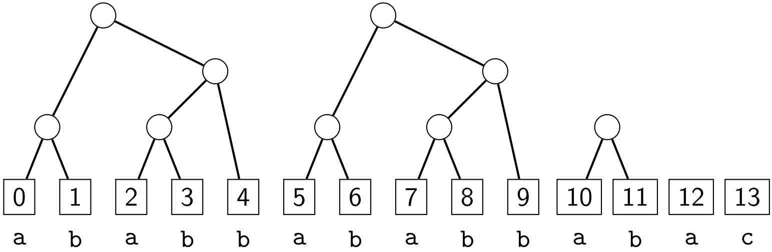
Building the tree



Building the tree

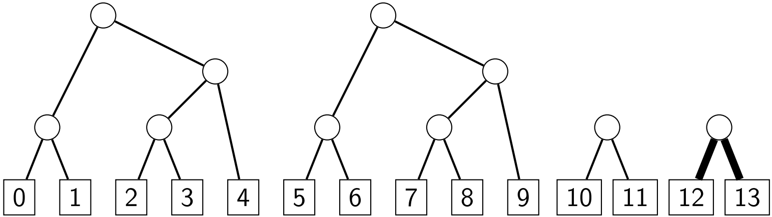


Building the tree



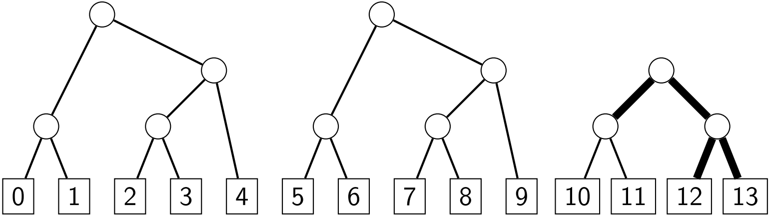
LynS	1	2	1	2	5	1	2	1	2	5	1	2	1	14

Building the tree



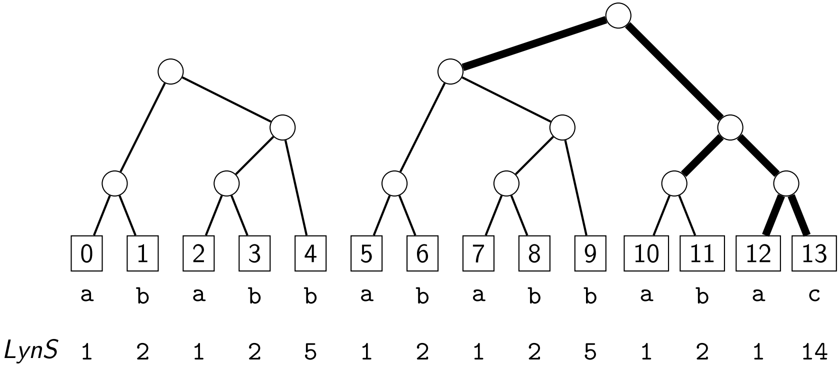
	a	b	a	b	b	a	b	a	b	b	a	b	a	c
LynS	1	2	1	2	5	1	2	1	2	5	1	2	1	14

Building the tree

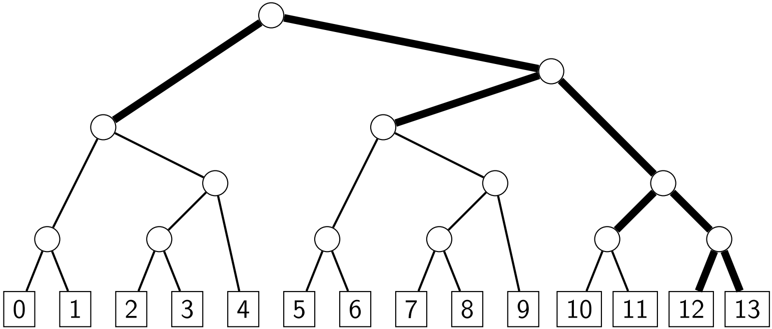


	a	b	a	b	b	a	b	a	b	b	a	b	a	c
LynS	1	2	1	2	5	1	2	1	2	5	1	2	1	14

Building the tree



Building the tree



	a	b	a	b	b	a	b	a	b	b	a	b	a	c
LynS	1	2	1	2	5	1	2	1	2	5	1	2	1	14

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	`a. a`
1		`a b. a b. a b. a b. a b. a b`
2		`a b a. a b a`
3		`a b a b. a b a b. a b a b`
4		`a b a b b. a b a b b`
5		`a b a b b a. a b a b b`
6		`a b a b b a b. a b a b b a b`
7		`a b a b b a b a. a b a b b a b a`
8		`a b a b b a b a b. a b a b b a b a b`
9		`a b a b b a b a b b. a b a b b a b a b b`
10		`a b a b b a b a b b a. a b a b b a b a b b`
11		`a b a b b a b a b b a b. a b a b b a b a b`
12		`a b a b b a b a b b a b a. a b a b b a b a`

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	a a
1		a b. a b. a b. a b. a b. a b
2	2	a b a. a b a
3		a b a b. a b a b. a b a b
4		a b a b b. a b a b b
5		a b a b b a. a b a b b
6		a b a b b a b. a b a b b a b
7		a b a b b a b a. a b a b b a b a
8		a b a b b a b a b. a b a b b a b a b
9		a b a b b a b a b b. a b a b b a b a b b
10		a b a b b a b a b b a. a b a b b a b a b b a
11		a b a b b a b a b b a b. a b a b b a b a b b a
12		a b a b b a b a b b a b a. a b a b b a b a b b

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	a a
1	4	a b. a b. a b. a b. a b. a b
2	2	a b a. a b a
3	3	a b a b. a b a b. a b a b
4		a b a b b. a b a b b
5		a b a b b a. a b a b b
6		a b a b b a b. a b a b b a b
7		a b a b b a b a. a b a b b a b a
8		a b a b b a b a b. a b a b b a b a b
9		a b a b b a b a b b. a b a b b a b a b b
10		a b a b b a b a b b a. a b a b b a b a b b
11		a b a b b a b a b b a b. a b a b b a b a b
12		a b a b b a b a b b a b a. a b a b b a b a

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	a a
1	4	a b. a b. a b. a b. a b. a b
2	2	a b a. a b a
3	3	a b a b. a b a b. a b a b
4		a b a b b. a b a b b
5	5	a b a b b a. a b a b b
6		a b a b b a b. a b a b b a b
7		a b a b b a b a. a b a b b a b a
8		a b a b b a b a b. a b a b b a b a b
9		a b a b b a b a b b. a b a b b a b a b b
10		a b a b b a b a b b a. a b a b b a b a b b
11		a b a b b a b a b b a b. a b a b b a b a b
12		a b a b b a b a b b a b a. a b a b b a b a

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	a a
1	4	a b. a b. a b. a b. a b. a b
2	2	a b a. a b a
3	3	a b a b. a b a b. a b a b
4		a b a b b. a b a b b
5	5	a b a b b a. a b a b b
6	8	a b a b b a b. a b a b b a b
7	6	a b a b b a b a. a b a b b a b a
8	7	a b a b b a b a b. a b a b b a b a b
9		a b a b b a b a b b. a b a b b a b a b b
10		a b a b b a b a b b a. a b a b b a b a b b
11		a b a b b a b a b b a b. a b a b b a b a b
12		a b a b b a b a b b a b a. a b a b b a b a

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	a a
1	4	a b. a b. a b. a b. a b. a b
2	2	a b a. a b a
3	3	a b a b. a b a b. a b a b
4		a b a b b. a b a b b
5	5	a b a b b a. a b a b b
6	8	a b a b b a b. a b a b b a b
7	6	a b a b b a b a. a b a b b a b a
8	7	a b a b b a b a b. a b a b b a b a b
9		a b a b b a b a b b. a b a b b a b a b b
10	9	a b a b b a b a b b a. a b a b b a b a b b
11	11	a b a b b a b a b b a b. a b a b b a b a b
12	10	a b a b b a b a b b a b a. a b a b b a b a

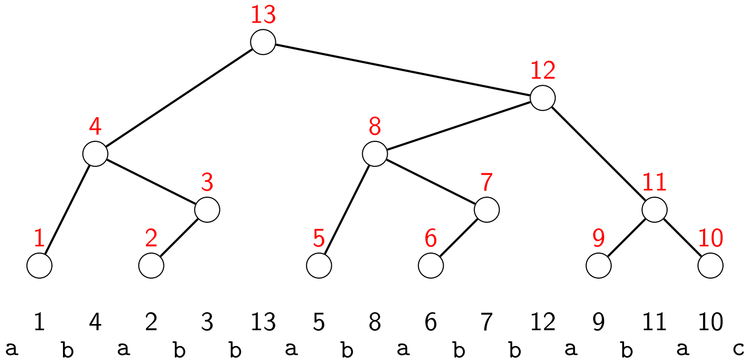
Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	$rank[j]$	
0	1	a a
1	4	a b. a b. a b. a b. a b. a b
2	2	a b a. a b a
3	3	a b a b. a b a b. a b a b
4	13	a b a b b. a b a b b
5	5	a b a b b a. a b a b b
6	8	a b a b b a b. a b a b b a b
7	6	a b a b b a b a. a b a b b a b a
8	7	a b a b b a b a b. a b a b b a b a b
9	12	a b a b b a b a b b. a b a b b a b a b b
10	9	a b a b b a b a b b a. a b a b b a b a b b
11	11	a b a b b a b a b b a b. a b a b b a b a b
12	10	a b a b b a b a b b a b a. a b a b b a b a

Cartesian tree of prefix ranks

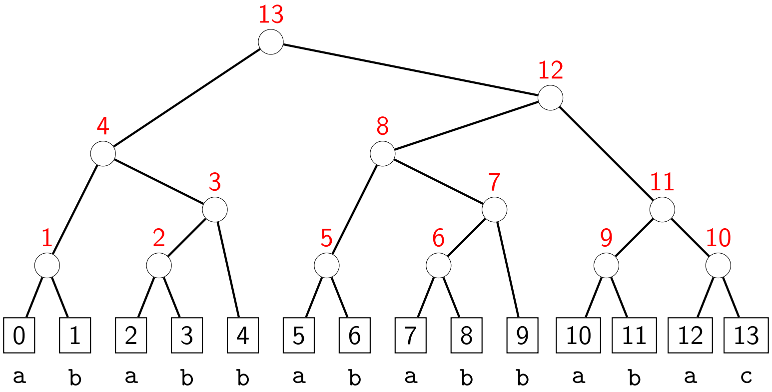
j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
rank[j]	1	4	2	3	13	5	8	6	7	12	9	11	10	



Prefix ranks and Left Lyndon tree

Theorem (Dolce, Restivo, Reutenauer, 2019)

The tree of internal nodes of the left Lyndon tree of a Lyndon word y in which nodes are labelled by the ranks of proper prefixes of y sorted according to the infinite order is the Cartesian tree of the ranks.



Prefix sorting

Theorem

Sorting the proper prefixes of a Lyndon word of length n according to the infinite order $\prec$ can be done in time $O(n)$ in the letter-comparison model.

Proof: In Algorithm $\textsc{LeftLyndonTree}$ list prefixes associated to internal nodes instead of building the tree.

Properties of Prefix standard permutation

A Lyndon word is either a singleton or $x^k zb$ where $k > 0$, z is a proper prefix of x and b is a letter bigger than what follows the occurrence of z in x.
Grouping the prefixes: for $e = 0, 1, \cdots, k$, let
$P_e = \{x^e u | e|x| < |u| < min\{(e+1)|x|, |y|\}\}$

Properties of Prefix standard permutation

A Lyndon word is either a singleton or $x^k zb$ where $k > 0$, z is a proper prefix of x and b is a letter bigger than what follows the occurrence of z in x.
Grouping the prefixes: for $e = 0, 1, \cdots, k$, let
$P_e = \{x^e u | e|x| < |u| < min\{(e+1)|x|, |y|\}\}$

Properties of Prefix standard permutation

A Lyndon word is either a singleton or $x^k zb$ where $k > 0$, z is a proper prefix of x and b is a letter bigger than what follows the occurrence of z in x.

Grouping the prefixes: for $e = 0, 1, \cdots, k$, let
$P_e = \{x^e u | e|x| < |u| < min\{(e+1)|x|, |y|\}\}$

- Prefixes $x^e u \in P_e, 0 < e \leq k$ are in the same relative $\prec$-order as prefixes $u \in P_0$

Properties of Prefix standard permutation

A Lyndon word is either a singleton or $x^k zb$ where $k > 0$, z is a proper prefix of x and b is a letter bigger than what follows the occurrence of z in x.

Grouping the prefixes: for $e = 0, 1, \cdots, k$, let
$P_e = \{x^e u | e|x| < |u| < min\{(e+1)|x|, |y|\}\}$

- Prefixes $x^e u \in P_e, 0 < e \leq k$ are in the same relative $\prec$-order as prefixes $u \in P_0$
- Prefixes $\in P_e$ are $\prec$-smaller than prefixes $\in P_f$ when $0 \leq e < f \leq k$

Properties of Prefix standard permutation

A Lyndon word is either a singleton or $x^k zb$ where $k > 0$, z is a proper prefix of x and b is a letter bigger than what follows the occurrence of z in x.

Grouping the prefixes: for $e = 0, 1, \cdots, k$, let
$P_e = \{x^e u | e|x| < |u| < min\{(e+1)|x|, |y|\}\}$

- Prefixes $x^e u \in P_e, 0 < e \leq k$ are in the same relative $\prec$-order as prefixes $u \in P_0$
- Prefixes $\in P_e$ are $\prec$-smaller than prefixes $\in P_f$ when $0 \leq e < f \leq k$
- Prefixes $\in P_e, 0 \leq e \leq k$, are $\prec$- smaller than prefixes $x^f, 0 < f \leq k$

Our example

$y = ababbababbabac = (ababb)^2abac$
$P_0 = \{a, ab, aba, abab\}$
$P_1 = \{ababba, ababbab, ababbaba, ababbabab\}$
$P_2 = \{ababbababba, ababbababbab, ababbababbaba\}$
$x = ababb, x^2 = ababbababb$

- sort P_0 (recursively)
- same relative order for P_1 and P_2
- $x_2 \prec x$

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
rank[j]	1	4	2	3		5	8	6	7					

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
rank[j]	1	4	2	3		5	8	6	7		9	11	10	

Prefix standard permutation

Ranks according to the infinite ordering $\prec$.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$y[j]$	a	b	a	b	b	a	b	a	b	b	a	b	a	c
rank$[j]$	1	4	2	3	13	5	8	6	7	12	9	11	10	

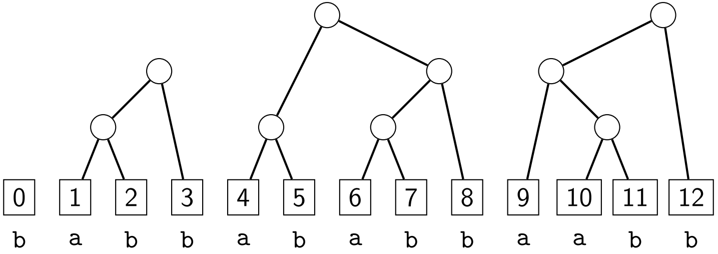
Lyndon Forest: input is any non-empty word

Algorithms extend to factors of the Lyndon factorisation of a non-empty word (algorithm by [*Duval*, 1983]).

Example

The Lyndon suffix table of $y =$ babbababbaabb is as follows.

j	0	1	2	3	4	5	6	7	8	9	10	11	12
$y[j]$	b	a	b	b	a	b	a	b	b	a	a	b	b
$LynS[j]$	1	1	2	3	1	2	1	2	5	1	1	3	4



Works on hand

Reverse engineering

- On a binary alphabet, function psp is one-to-one.
- Given a permutation p of $\{0, 1, \ldots, n-2\}$, the word y of length n for which $\mathrm{psp}(y) = p$ can be found in linear time.
- How much can the right Lyndon tree of y help sort its suffixes?

Works on hand

Reverse engineering

- On a binary alphabet, function psp is one-to-one.
- Given a permutation p of $\{0, 1, \ldots, n-2\}$, the word y of length n for which $\mathrm{psp}(y) = p$ can be found in linear time.
- How much can the right Lyndon tree of y help sort its suffixes?

Relation between left and right Lyndon trees