Lyndon arrays and Lyndon trees

Thierry Lecroq

1st Lyndex meeting 2023, Rouen, France, December 14th, 2023

Outline

1. Lyndon Arrays
2. Runs
3. Lyndon Trees
4. Cartesian Trees

Outline

1. Lyndon Arrays
2. Runs
3. Lyndon Trees
4. Cartesian Trees

Lyndon words

Lyndon word

A non-empty word x is a Lyndon word if

x is smaller than all its proper non-empty suffixes

Lyndon arrays (1)

Lyndon array of a (non-empty) word x

For a position i on x, $i = 0, \ldots, |x| - 1$,
$Lyn[i]$ is the length of the longest Lyndon factor of x starting at i:

$$Lyn[i] = \max\{\ell \mid x[i \mathinner{..} i + \ell - 1] \text{ is a Lyndon word}\}$$

Lyndon arrays (2)

Example

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x[i]$	a	b	b	a	b	a	b	a	a	b	a	b	b	a	b	a
$Lyn[i]$	3	1	1	2	1	2	1	8	5	1	3	1	1	2	1	1

Lyndon arrays (3)

LongestLyndon(x non-empty word of length n)

for $i \leftarrow n-1$ **downto** 0 **do**
  $(Lyn[i], j) \leftarrow (1, i+1)$
  while $j < n$ **and** $x[i \mathinner{..} j-1] < x[j \mathinner{..} j + Lyn[j] - 1]$ **do**
    $(Lyn[i], j) \leftarrow (Lyn[i] + Lyn[j], j + Lyn[j])$
return Lyn

Lyndon arrays (4)

Well known properties

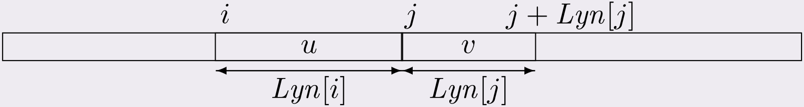
1. If u and v are Lyndon words and $u < v$ then
   - uv is also a Lyndon word
   - $u < uv < v$.
2. Each non-empty word x factorises uniquely as $u_0 u_1 u_2 \cdots$, where
   - each u_i is a Lyndon word
   - $u_0 \geq u_1 \geq u_2 \geq \cdots$.
   - u_0 is the longest Lyndon prefix of x.

Lyndon arrays (5)

Invariant

When computing $Lyn[i]$:

- $Lyn[k]$ has already been computed for $i < k < n$
- $u[i+1\,..\,j-1] \cdot u[j\,..\,j+Lyn[j]-1] \cdots$ is the Lyndon factorisation of $x[i+1\,..\,n-1]$ where $j = i+1+Lyn[i+1]$,



If $u < v$ then $u \leftarrow uv$ and $v \leftarrow$ successor of v

Outline

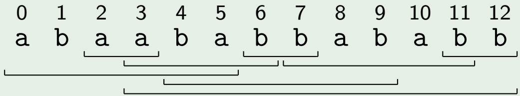
Runs (1)

Definition

A run is a maximal periodicity occurring in a word x:
interval $[i\,..\,j]$ such that

- $x[i\,..\,j]$ is periodic (i.e., its smallest period p satisfies $2p \leq |x[i\,..\,j]| = (j-i+1)$)
- the periodicity does not extend to the right nor to the left (i.e., $x[i-1\,..\,j]$ and $x[i\,..\,j+1]$ have larger periods when defined).

Runs (2)



Runs (3)

Maximal number

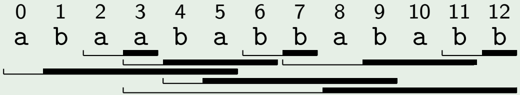
- $O(n)$ Kolpakov and Kucherov, 1999
- $5n$ Rytter, 2006
- $3.48n$ Puglisi, Simpson and Smyth, 2008
- $1.6n$ Crochemore and Ilie, 2008
- $1.49n$ Giraud, 2008
- $1.029n$ Crochemore, Ilie and Tinta, 2008
- $n - 3$ Bannai, I, Inenaga, Nakashima, Takeda, and Tsuruta, 2014
- $0.957n$ Fischer, Holub, I and Lewenstein, 2015

Runs (4)

Special position

Let us consider the orderings $<$ and its reverse $<^{-1}$.
Each run $[i \mathinner{.\,.} j]$ is associated with its greatest suffix according to one of the 2 orderings as follows:
Let $p = per(x[i \mathinner{.\,.} j])$.
If $j+1 < n$ and $x[j+1] > x[j-p+1]$ we assign to the run the position k for which $x[k \mathinner{.\,.} j]$ is the greatest proper suffix of $x[i \mathinner{.\,.} j]$ according to $<$.
Otherwise, k is the starting position of the greatest proper suffix of $x[i \mathinner{.\,.} j]$ according to $<^{-1}$.
The position k assigned this way to a run is called its special position.

Runs (5)



Runs (6)

Lemma

If the special position k of a run of period p is defined according to $<^{-1}$ (resp. $<$) then
$x[k \mathinner{.\,.} k+p-1]$ is the longest Lyndon factor of x starting at position k according to $<$ (resp. $<^{-1}$).

Runs (7)

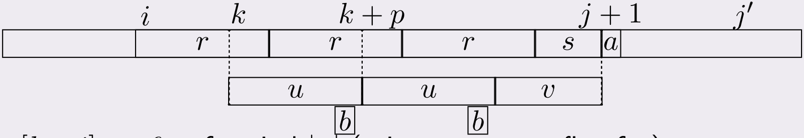
Proof

Let $[i\,..\,j]$ be a run of period p with special position k.
$x[k\,..\,k+p-1]$ is a Lyndon word because it is smaller than all its proper suffixes according to $<$.
Consider a longer factor $x[k\,..\,j']$ for $k+p \leq j' \leq j$.
It has period p which is smaller than its length; equivalently it is not border free, which shows it is not a Lyndon word for any of the 2 orderings.

Runs (8)

Proof

There is nothing else to prove if $j+1 = |x|$.
Assume then that $j' > j$ and $a = x[j+1]$.



$x[k\,..\,j] = u^e v$ of period $|u|$ (v is a proper prefix of u): greatest suffix of $x[i\,..\,j]$ according to $<^{-1}$
Since $a = x[j+1] < x[j-p+1] = b$, we get $x[k+p\,..\,j+1] < x[k\,..\,j-p+1]$, which leads to $x[k+p\,..\,j'] < x[k\,..\,j']$ and shows that $x[k\,..\,j']$ is not a Lyndon word according to $<$.

Runs (9)

RUNS(x non-empty word of length n)

for $i \leftarrow n-1$ **downto** 0 **do**
    $(Lyn[i], j) \leftarrow (1, i+1)$
    while $j < n$ and $x[i \mathinner{..} j-1] < x[j \mathinner{..} j + Lyn[j] - 1]$ **do**
        $(Lyn[i], j) \leftarrow (Lyn[i] + Lyn[j], j + Lyn[j])$
    $\ell \leftarrow |lcs(x[0 \mathinner{..} i-1], x[0 \mathinner{..} i + Lyn[i] - 1])|$
    $r \leftarrow |lcp(x[i \mathinner{..} |x|-1], x[i + Lyn[i] \mathinner{..} |x|-1])|$
    if $\ell + r \geq Lyn[i]$ **then**
        output run $[i - \ell \mathinner{..} i + Lyn[i] + r - 1]$

Runs (10)

Runs

- lcs: longest common suffix
- lcp: longest common prefix
- LCE: longest common extensions, can be computed in constant time after linear preprocessing

Outline

1. Lyndon Arrays
2. Runs
3. Lyndon Trees
4. Cartesian Trees

Lyndon trees (1)

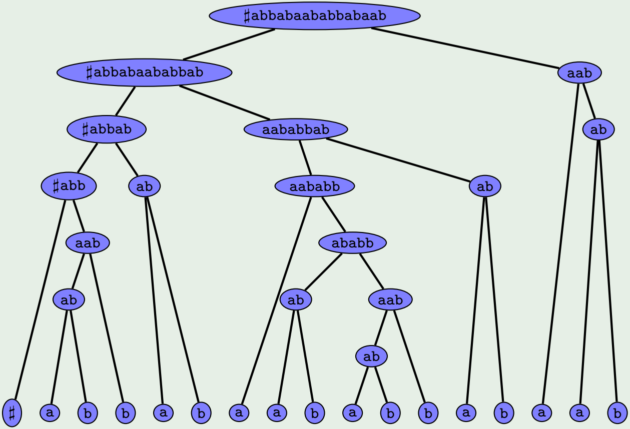
Definition

Let w be a Lyndon word s.t. $|w| > 1$, and let $w = uv$ such that v is the smallest proper non-empty suffix of w (standard factorisation) then u is also a Lyndon word.
Then the Lyndon tree $T(w)$ of w is recursively defined as follows:

- the root is w
- the left subtree of the root is $T(u)$
- the right subtree of the root is $T(v)$

Lyndon trees (2)

Example



Lyndon trees (3)

LyndonTree(x non-empty word of length n)

$(v, T(v)) \leftarrow (x[n-1], (x[n-1], (), ()))$
for $i \leftarrow n-2$ **downto** 0 **do**
    $(u, T(u)) \leftarrow (x[i], (x[i], (), ()))$
    while $u < v$ **do**
        $T(uv) \leftarrow (uv, T(u), T(v))$
        $u \leftarrow uv$
        $v \leftarrow$ next phrase or ε
return $T(x)$

Outline

1. Lyndon Arrays
2. Runs
3. Lyndon Trees
4. Cartesian Trees

Cartesian Trees (1)

Definition

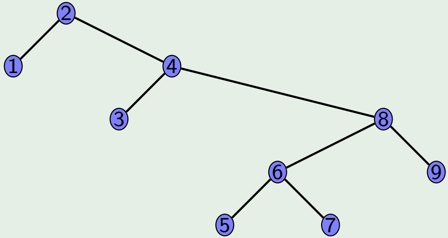
Let x be a string of numbers of length m.
The Cartesian Tree $CT(x)$ [Vuillemin 1980] of x is the binary tree where:

- the root corresponds to the index i of the minimal element of x (if there are several occurrences of the minimal element, the leftmost one is chosen)
- the left subtree is $CT(x[1 \mathinner{.\,.} i-1])$
- the right subtree is $CT(x[i+1 \mathinner{.\,.} m])$

Cartesian Trees (2)

Example

i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

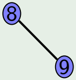
Example

i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9

Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

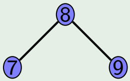
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

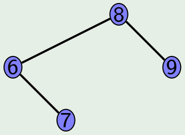
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

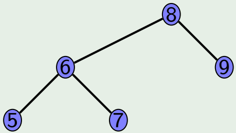
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

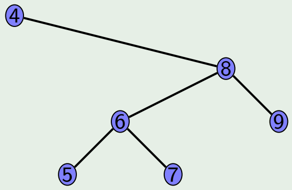
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

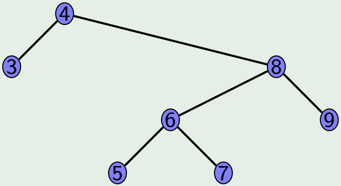
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

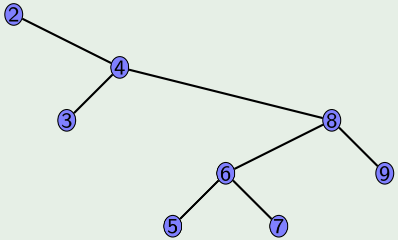
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

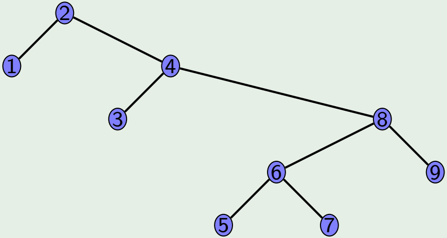
i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



Cartesian Trees: Incremental Construction From Right To Left, Bottom-Up Scan of the Left Path

Example

i	1	2	3	4	5	6	7	8	9
$x[i]$	3	1	6	4	8	6	7	5	9



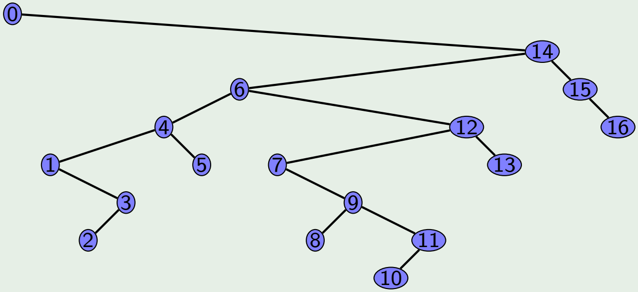
Cartesian Trees (4)

CARTESIANTREE(x non-empty word of length n)

$Root(T) \leftarrow n-1$
$S \leftarrow (n-1)$
for $i \leftarrow n-2$ **downto** 0 **do**
  $r \leftarrow Nil$
  while $S \neq \emptyset$ and $x[i] < x[Top(S)]$ **do**
    $r \leftarrow Pop(S)$
  $Right(i) \leftarrow r$
  if $S \neq \emptyset$ **then**
    $Left(Top(S)) \leftarrow i$
  else $Root(T) \leftarrow i$
return T

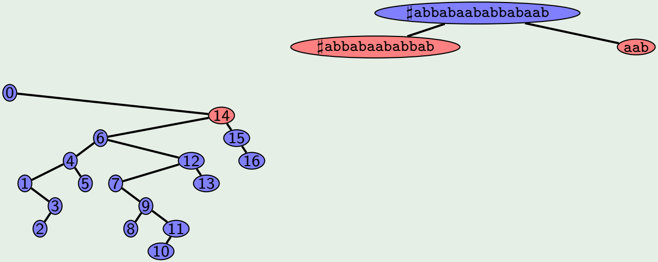
Cartesian Trees (5)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b
SA	0	14	6	15	12	4	7	9	1	16	13	5	11	3	8	10	2
ISA	0	8	16	13	5	11	2	6	14	7	15	12	4	10	1	3	9



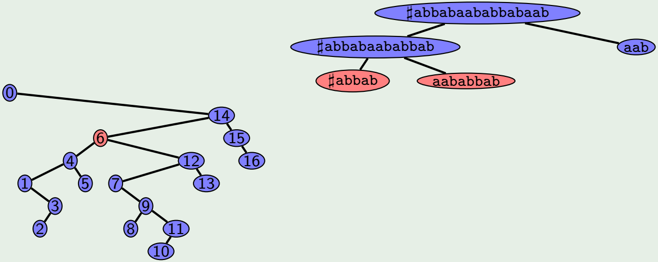
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



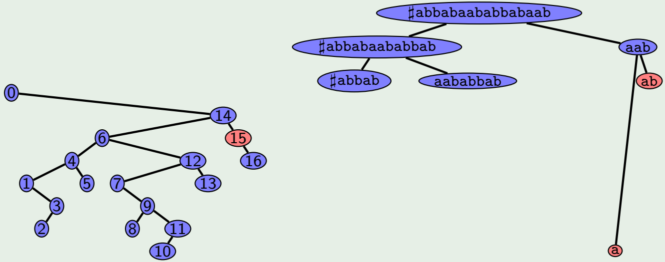
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



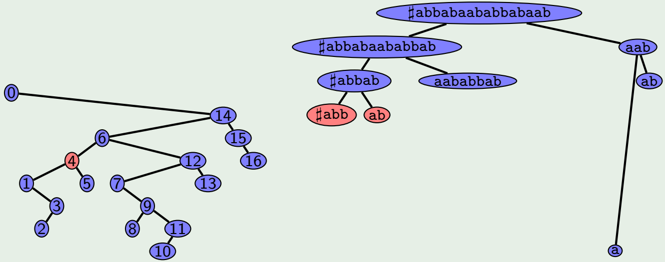
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



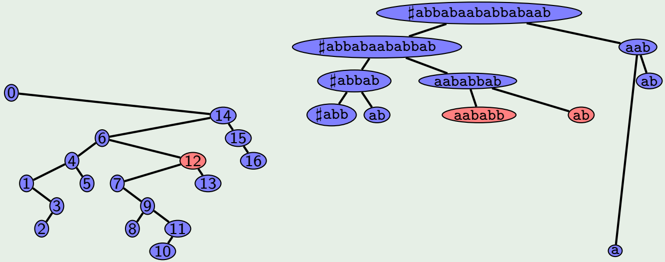
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



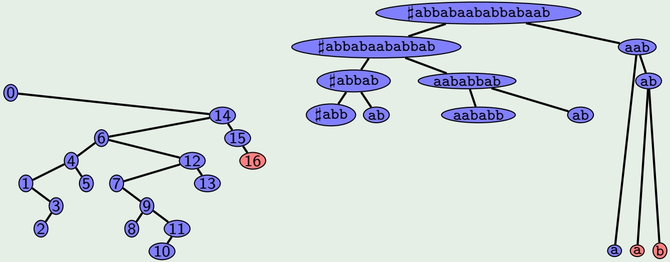
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



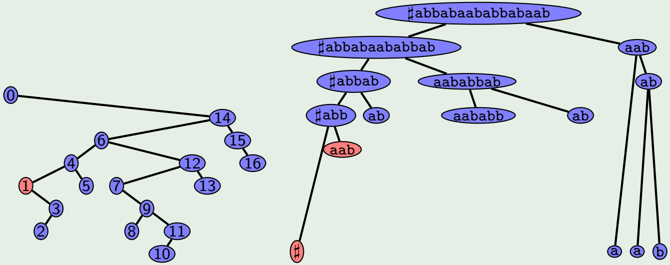
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



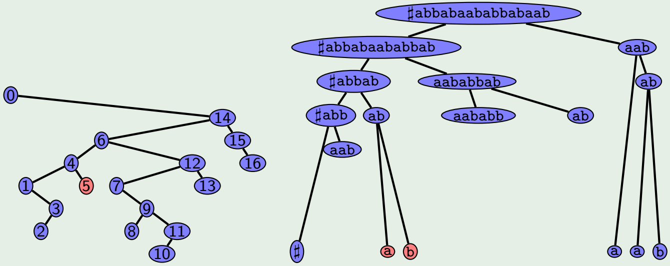
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



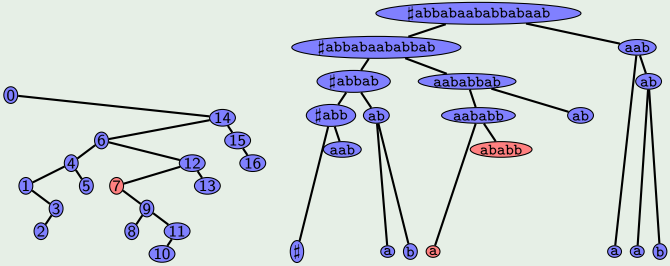
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



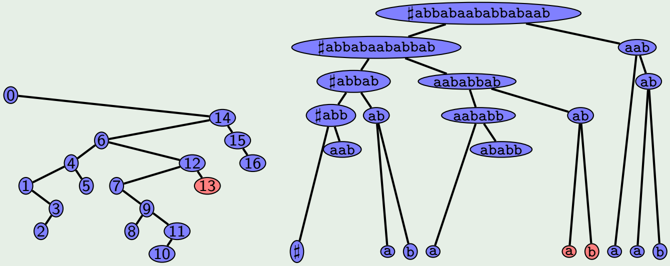
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



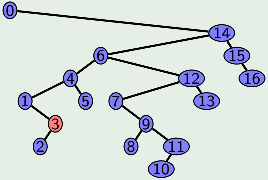
Cartesian Trees (6)

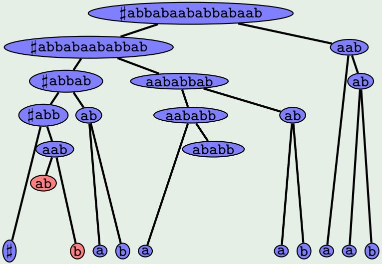
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



Cartesian Trees (6)

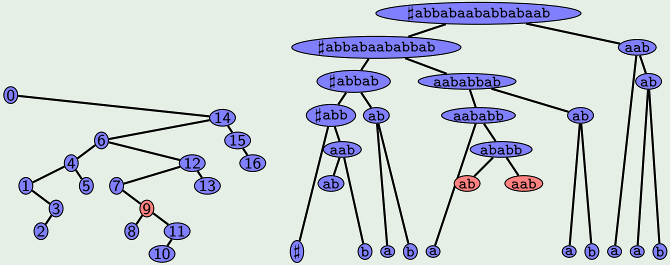
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b





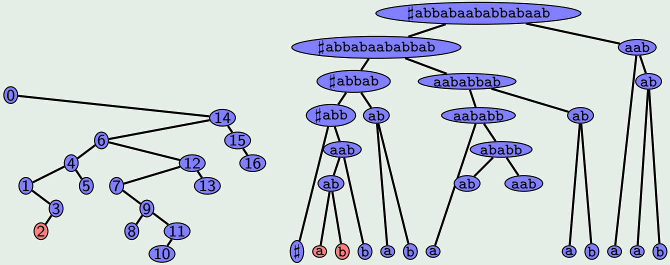
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



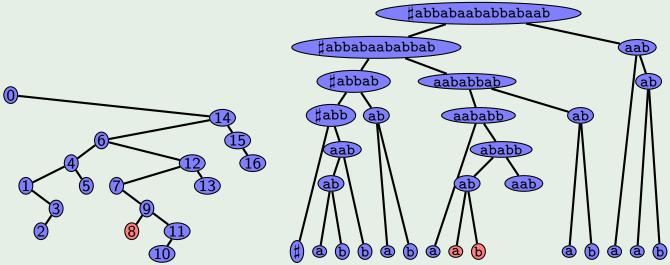
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



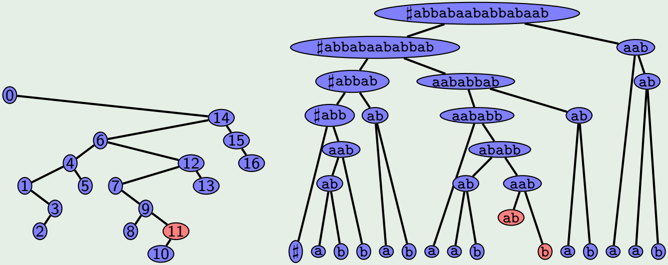
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



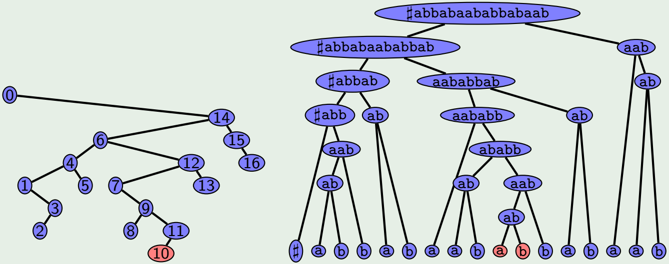
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	♯	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



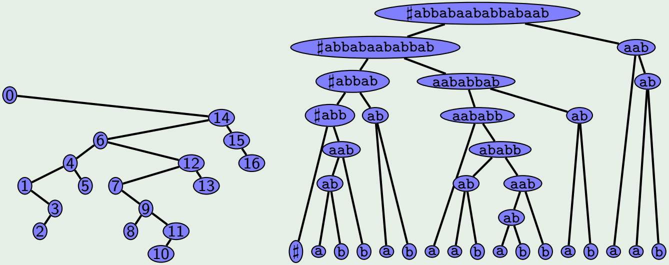
Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



Cartesian Trees (6)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x	$\sharp$	a	b	b	a	b	a	a	b	a	b	b	a	b	a	a	b



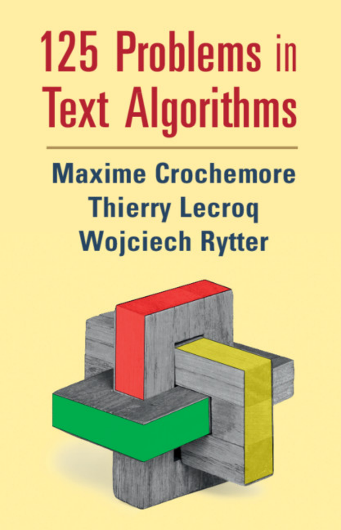
LongestLyndon(x, n)

for $i \leftarrow n-1$ **downto** 0 **do**
  $(Lyn[i], j) \leftarrow (1, i+1)$
  while $j < n$ and $x[i \,..\, j-1] < x[j \,..\, j+Lyn[j]-1]$ **do**
    $(Lyn[i], j) \leftarrow (Lyn[i]+Lyn[j], j+Lyn[j])$
return Lyn

LongestLyndon(x, n)

for $i \leftarrow n-1$ **downto** 0 **do**
  $(Lyn[i], j) \leftarrow (1, i+1)$
  while $j < n$ and $ISA[i] < ISA[j]$ **do**
    $(Lyn[i], j) \leftarrow (Lyn[i]+Lyn[j], j+Lyn[j])$
return Lyn

Main Reference



Other References

M. Crochemore, L.M.S. Russo
Cartesian and Lyndon trees
Theoretical Computer Science **806** (2020) 1–9

C. Hohlweg, C. Reutenauer
Lyndon words, permutations and trees
Theoretical Computer Science **307**(1) (2003) 173–178

Thank you for your attention!