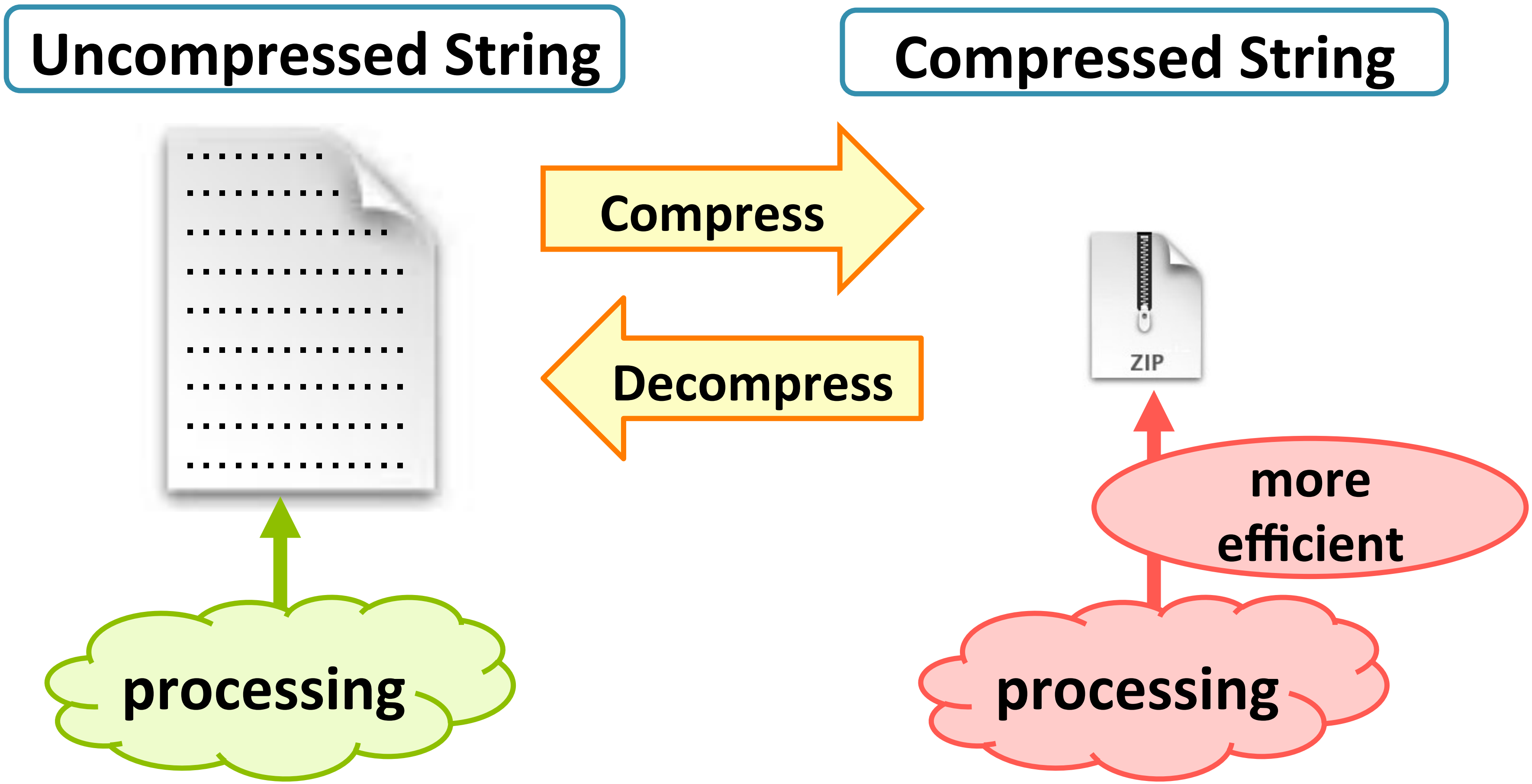
Efficient Lyndon factorization of grammar compressed text

Tomohiro I, Yuto Nakashima,
Shunsuke Inenaga, Hideo Bannai,
Masayuki Takeda

Kyushu University, Japan

Background



We want to process compressed strings without decompressing explicitly.

Problem and our contribution

- We solve the following problem.

Problem

Given an SLP S of size n representing a string w of length N, we compute the Lyndon factorization of w, denoted by $LF(w)$.

Theorem

Given an SLP S of size n representing a string w of length N, we can compute $LF(w)$ in $O(mn^4)$ time and $O(n^2)$ space, where m is the number of factors in $LF(w)$.

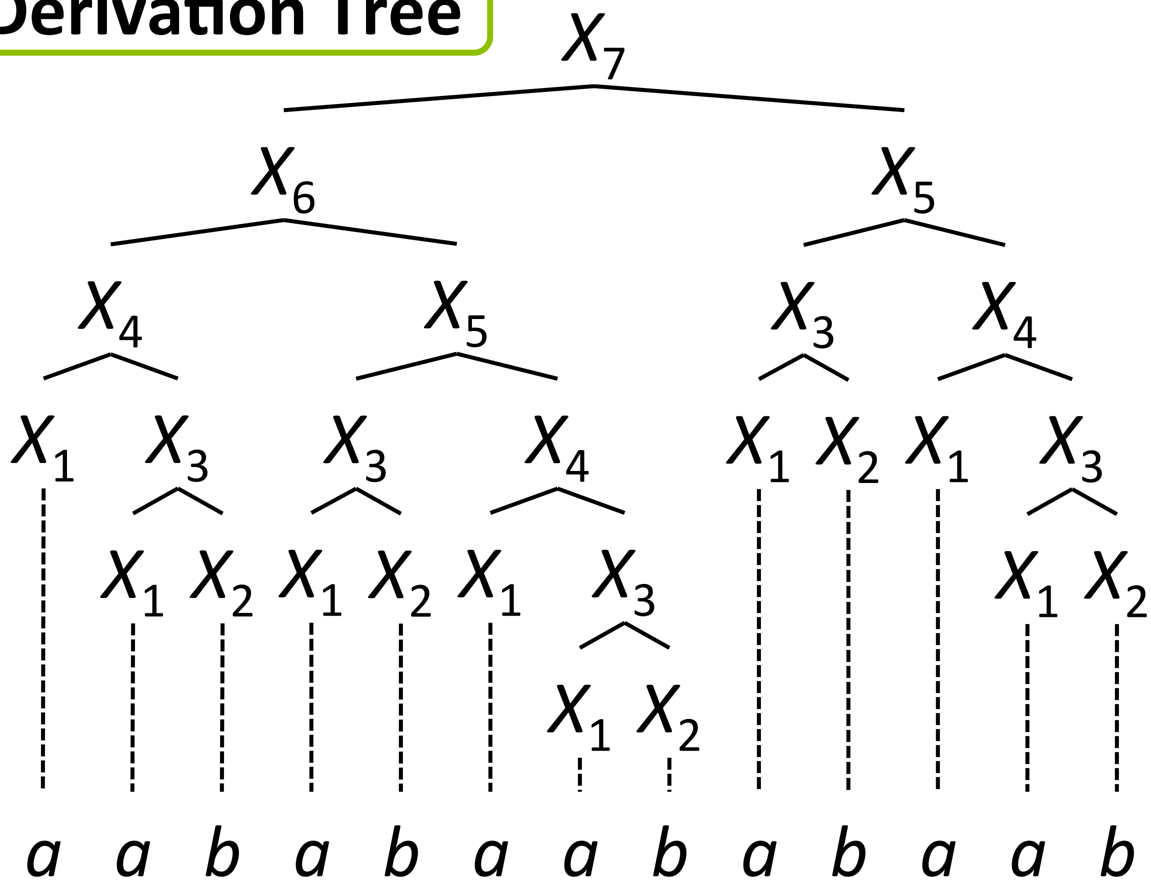
Straight line Program (SLP)

- An SLP is a context free grammar in the Chomsky normal form, that derives a single string.

SLP

$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow X_1 X_2$
$X_4 \rightarrow X_1 X_3$
$X_5 \rightarrow X_3 X_4$
$X_6 \rightarrow X_4 X_5$
$X_7 \rightarrow X_6 X_5$

Derivation Tree



Lyndon word

Definition [R. C. Lyndon, 1954]

A string *w* is a Lyndon word, if *w* is lexicographically strictly smaller than its proper cyclic shifts.

Lyndon word

w = *aababb*

proper cyclic shifts of *w*:

baabab
bbaaba
abbaab
babbaa
ababba

lexicographic order

aababb = *w*
ababba
abbaab
baabab
babbaa
bbaaba

Lyndon factorization

Definition [K. T. Chen et al., 1958]

The Lyndon factorization of a string w, denoted by $LF(w)$, is the factorization $l_1^{p_1} \ldots l_m^{p_m}$ of w, such that each $l_i \in \Sigma^+$ is a Lyndon word, $p_i \geq 1$, and $l_i \succ l_{i+1}$ for all $1 \leq i < m$.

$$w = a\,b\,c \mid a\,b\,b \,\vdots\, a\,b\,b \mid a\,a\,b\,c \mid a \,\vdots\, a \,\vdots\, a$$

$$LF(w) = (abc)(abb)^2(aabc)(a)^3$$

Lyndon factors

lexicographic order $abc \succ abb \succ aabc \succ a$

The Lyndon factorization of any string w is unique.

Representation of factorization

- Each Lyndon factor is represented by pair of length and its power.
- The representation takes $O(m)$ space where m is the number of Lyndon factors.
- We call m is the size of the Lyndon factorization.

Example

$$w = a\,b\,c \mid a\,b\,b \,\vdots\, a\,b\,b \mid a\,a\,b\,c \mid a \,\vdots\, a \,\vdots\, a$$

$$LF(w) = (abc)(abb)^2(aabc)(a)^3$$

$$\boxed{LF(w) = (3,\ 1)(3,\ 2)(4,\ 1)(1,\ 3)}$$

Linear time algorithm

- There exists a linear time algorithm to compute the Lyndon factorization.

Theorem [J. P. Duval, 1983]

Given a string w of length N,
$LF(w)$ can be computed in $O(N)$ time.

- If an SLP is very compressible, we want to compute the $LF(w)$ in polynomial time for n where n is size of an SLP that derives w.
(Since $N = O(2^n)$.)

basic idea of our algorithm

Lemma [J. P. Duval, 1983]

For any string w, let $LF(w) = l_1^{p_1} \ldots l_m^{p_m}$.
Then, l_m is the lexicographically smallest suffix of w.

$w = a\,b\,c\,a\,b\,b\,a\,b\,b\,\boxed{a\,a\,b\,b}$

smallest suffix of w ↑

$w_1 = a\,b\,c\,a\,b\,b\,\boxed{a\,b\,b}\,a\,a\,b\,b$

smallest suffix of w_1 ↑

$w_2 = a\,b\,c\,\boxed{a\,b\,b}\,a\,b\,b\,a\,a\,b\,b$

↑ smallest suffix of w_2

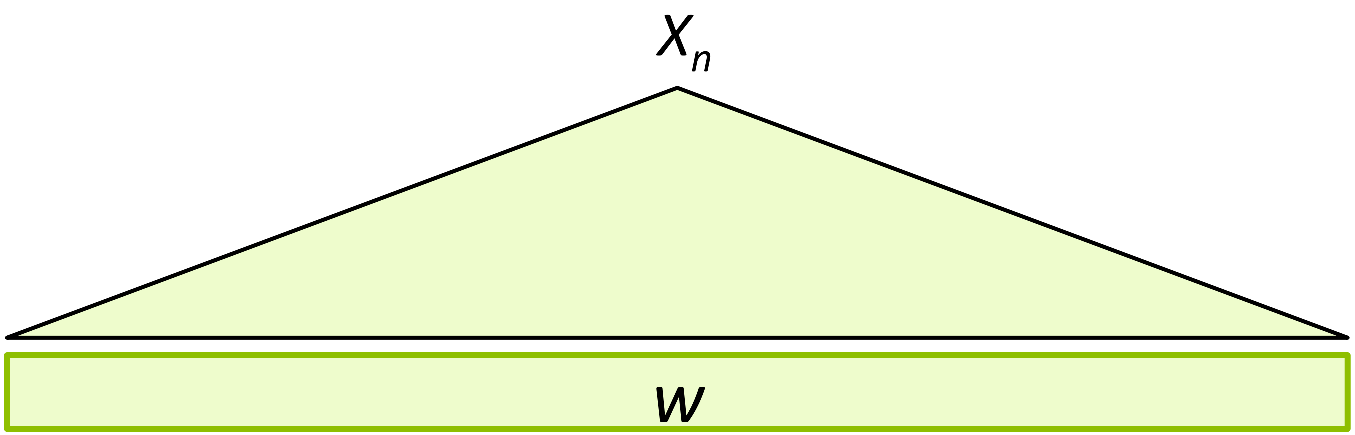
$w_3 = \boxed{a\,b\,c}\,a\,b\,b\,a\,b\,b\,a\,a\,b\,b$

↑ smallest suffix of w_3

$$LF(w) = (abc)(abb)^2(aabb)$$

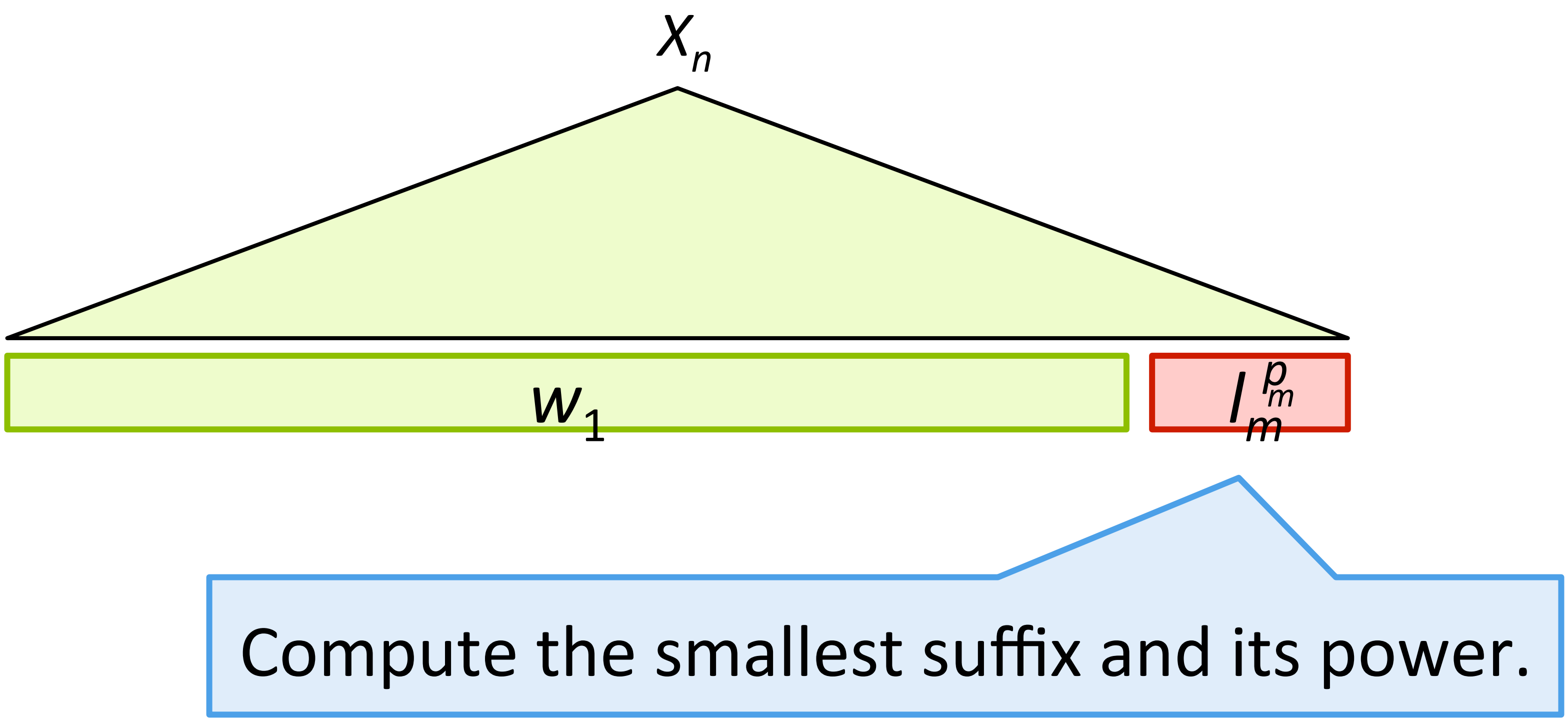
Sub problem

- We want to compute the lexicographically smallest suffix and its power when *w* is given an SLP.



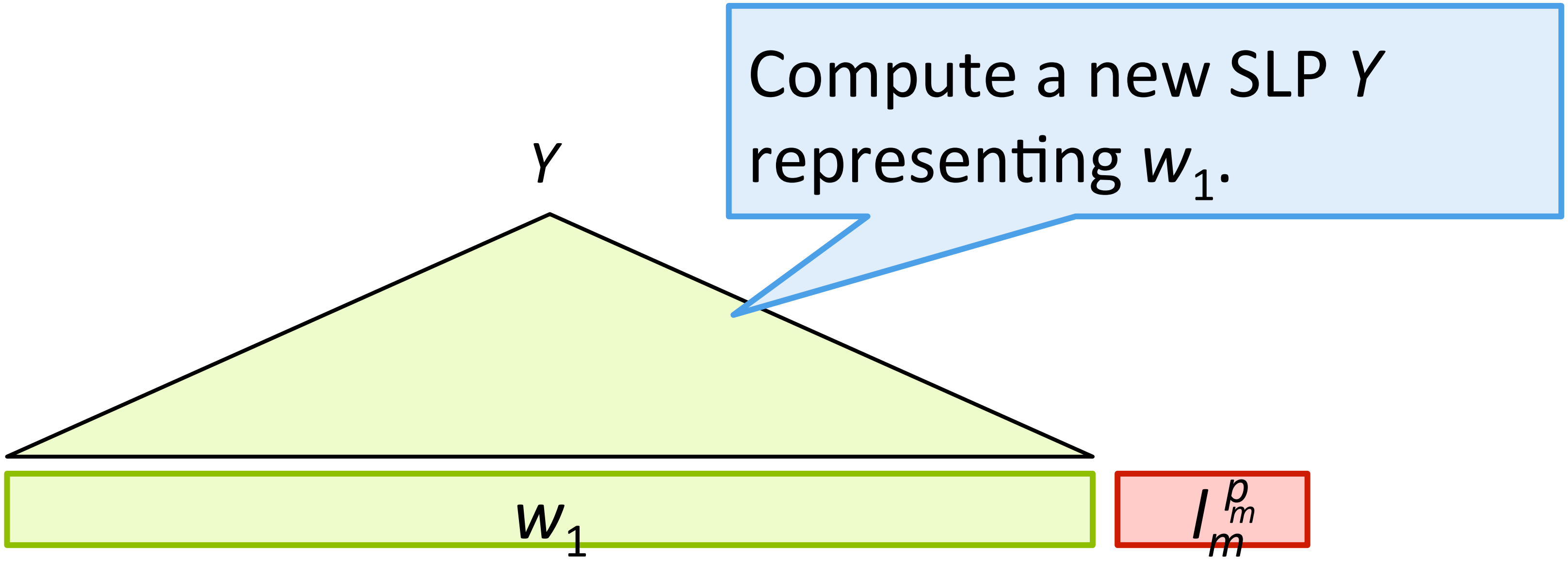
Sub problem

- We want to compute the lexicographically smallest suffix and its power when w is given an SLP.



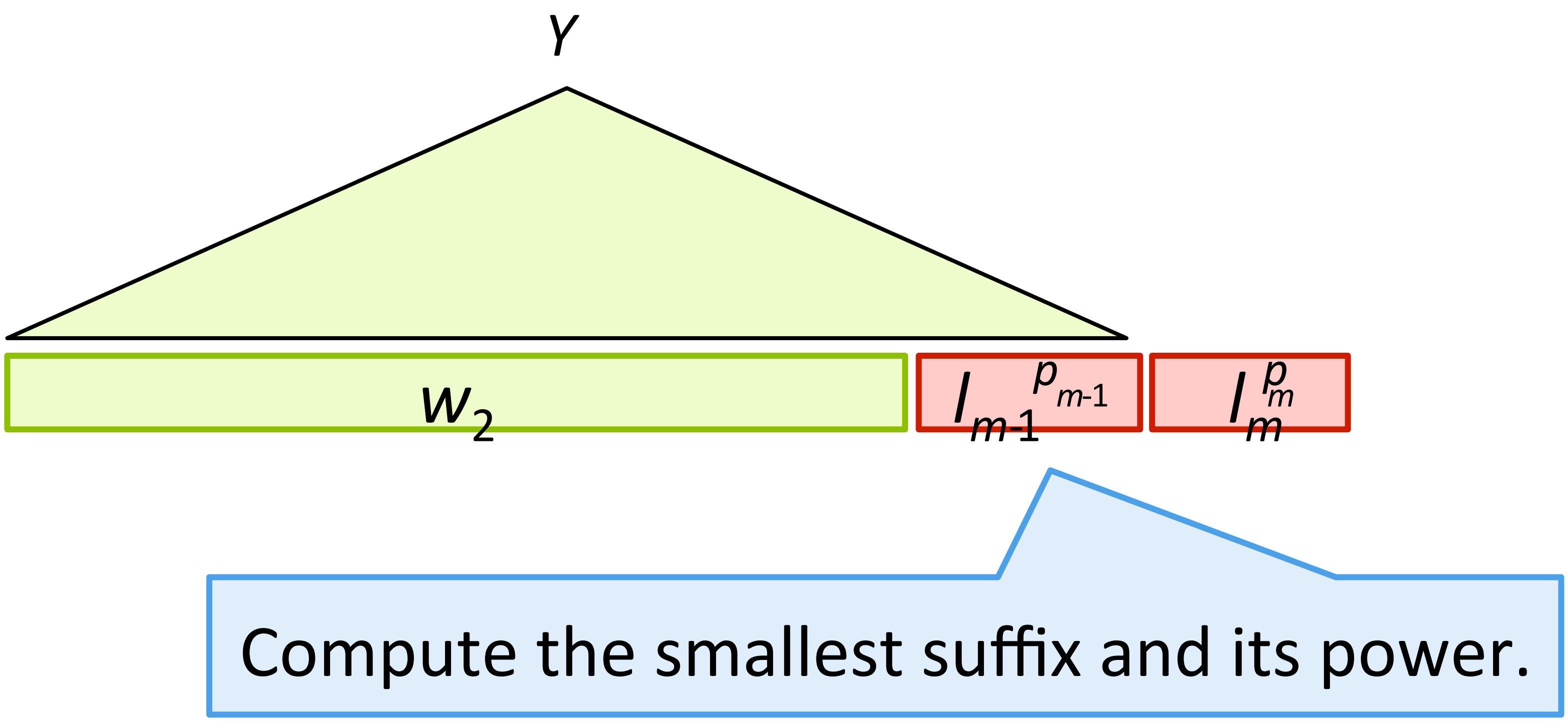
Sub problem

- We want to compute the lexicographically smallest suffix and its power when w is given an SLP.



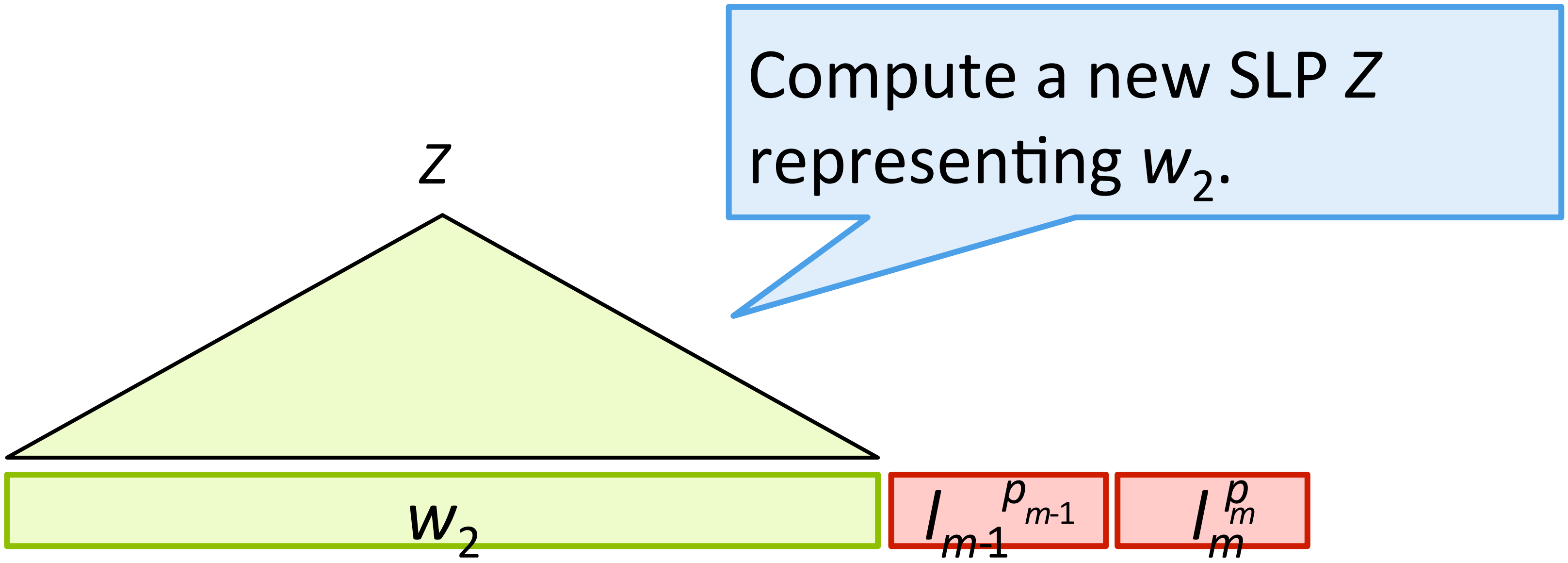
Sub problem

- We want to compute the lexicographically smallest suffix and its power when w is given an SLP.



Sub problem

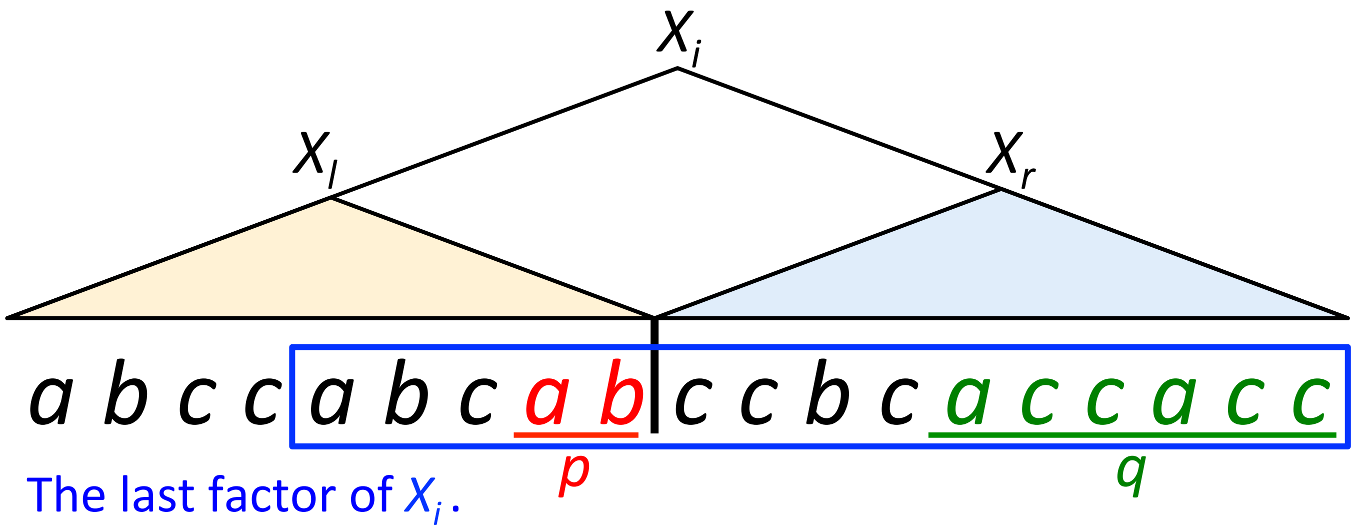
- We want to compute the lexicographically smallest suffix and its power when w is given an SLP.



- We iterate these operations *m* times.

The smallest suffix of an SLP

- Let $X_i = X_l X_r$. Given the last factor p, q of X_l and X_r respectively, can we compute the last factor of X_i ?



- The starting position of the last factor of X_i may not be the same as the starting position of p or q.

- We consider the set of candidates which can be a prefix of the last factor of X_i .

LFCand

Definition

For any non-empty string $w \in \Sigma^+$,
let $LFCand(w) = \{ x \mid x \in Suffix(w), \exists y \in \Sigma^+$
s.t. xy is the lexicographically smallest suffix of wy $\}$.
($Suffix(w)$ is the set of all suffixes of w.)

- $LFCand(w)$ is the set of suffixes of w which can be a prefix of the lexicographically smallest suffix of wy for some non-empty string y.

LFCand

Definition

For any non-empty string $w \in \Sigma^+$,
let $LFCand(w) = \{ x \mid x \in Suffix(w), \exists y \in \Sigma^+$ s.t. xy is the lexicographically smallest suffix of $wy \}$.
($Suffix(w)$ is the set of all suffixes of w.)

$w = a\,b\,a\,b\,c\,a\,b\,a\,b\,d\,a\,b\,a\,b\,c\,\underline{a\,b\,a\,b\;\,a\,b\,b} = y$

LFCand

Definition

For any non-empty string $w \in \Sigma^+$,
let $LFCand(w) = \{ x \mid x \in Suffix(w),\ \exists y \in \Sigma^+$
s.t. xy is the lexicographically smallest suffix of wy $\}$.
($Suffix(w)$ is the set of all suffixes of w.)

w = $a\ b\ a\ b\ c\ a\ b\ a\ b\ d\ \underline{a\ b\ a\ b\ c\ a\ b\ a\ b\ \ c\ c}$ $= y$

LFCand

Definition

For any non-empty string $w \in \Sigma^+$,
let $LFCand(w) = \{ x \mid x \in Suffix(w), \exists y \in \Sigma^+$
s.t. xy is the lexicographically smallest suffix of wy }.
($Suffix(w)$ is the set of all suffixes of w.)

$w = a\,b\,a\,b\,c\,a\,b\,a\,b\,d\,a\,b\,a\,b\,c\,a\,b\,a\,b\;\; d\,c = y$

LFCand

Definition

For any non-empty string $w \in \Sigma^+$,
let $LFCand(w) = \{ x \mid x \in Suffix(w), \exists y \in \Sigma^+$
s.t. xy is the lexicographically smallest suffix of $wy \}$.
($Suffix(w)$ is the set of all suffixes of w.)

w = $a\ b\ a\ b\ c\ a\ b\ a\ b\ d\ a\ b\ a\ b\ c\ a\ b\ a\ b$

Properties of LFCand

- *LFCand* has important properties.
- For any two elements of *LFCand*(w), the shorter one is the prefix of the longer one.

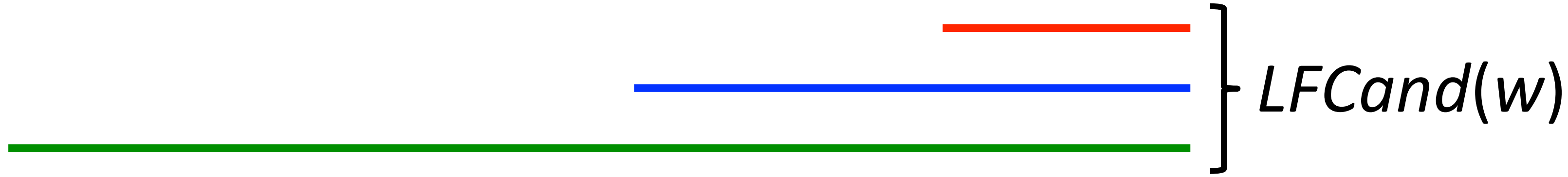
$$w = a\,b\,a\,b\,c\,a\,b\,a\,b\,d\,a\,b\,a\,b\,c\,a\,b\,a\,b$$

Properties of LFCand

Lemma

For any string w, the shortest element of $LFCand(w)$ is the last Lyndon factor of w.

$w = a\,b\,a\,b\,c\,a\,b\,a\,b\,d\,a\,b\,a\,b\,c\,a\,b\,a\,b$



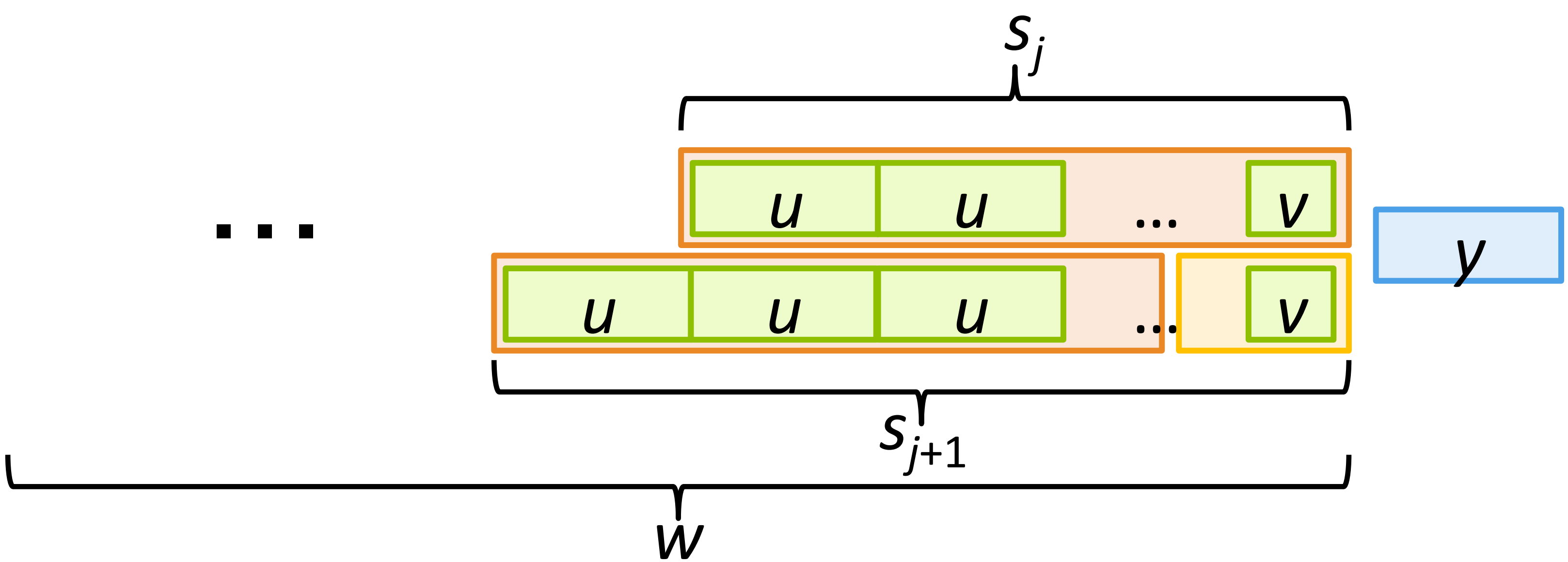
- If we can compute *LFCand*, we can compute the Lyndon factorization.

Properties of LFCand

- Let s_j be the jth shortest string of *LFCand*(w).

Lemma

$|s_{j+1}| > 2|s_j|$ holds.



- If $|s_{j+1}| \le 2|s_j|$, then s_j and s_{j+1} have a period q.
- s_j can not be in *LFCand*(w).

Properties of LFCand

- Thus the following lemma holds.

Lemma

$|s_{j+1}| > 2|s_j|$ holds.

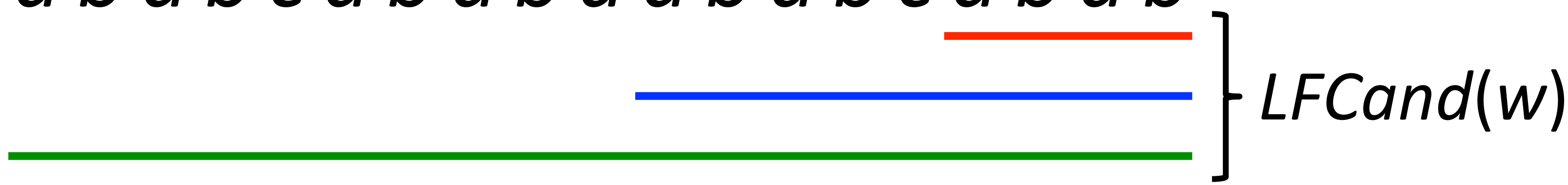
- By the above lemma, the following lemma holds.

Lemma

For any string w of length N, $|LFCand(w)| = O(\log N)$.

<u>Example</u>

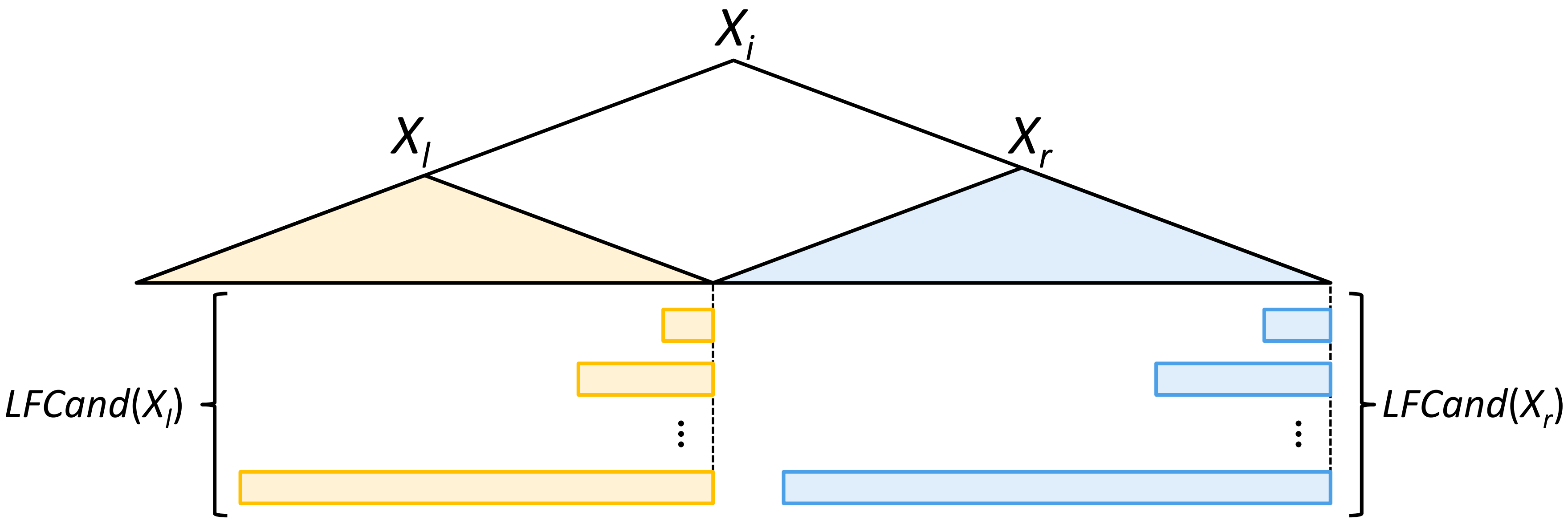
$w = a\,b\,a\,b\,c\,a\,b\,a\,b\,d\,a\,b\,a\,b\,c\,a\,b\,a\,b$



Computing LFCand

Lemma

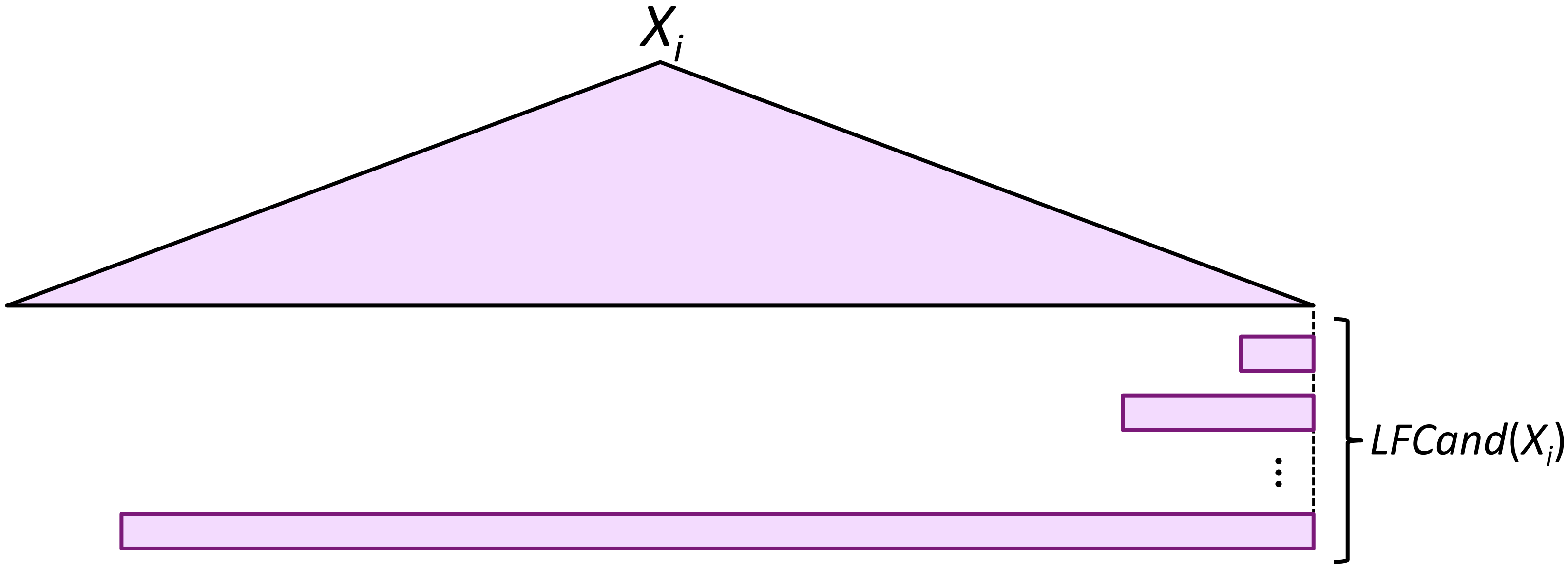
Let $X_i = X_l X_r$ be any production of a given SLP S of size n. Provided that sorted lists for $LFCand(X_l)$ and $LFCand(X_r)$ are already computed, a sorted list for $LFCand(X_i)$ can be computed in $O(n^3)$ time and $O(n^2)$ space.



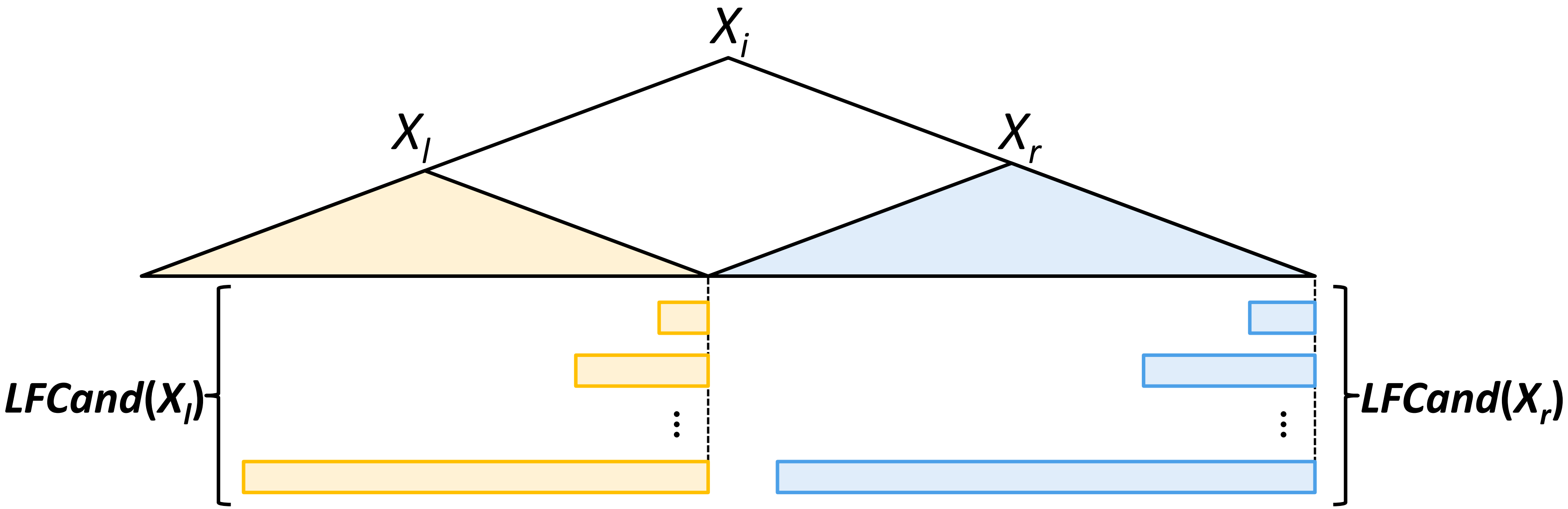
Computing LFCand

Lemma

Let $X_i = X_l X_r$ be any production of a given SLP S of size n. Provided that sorted lists for $LFCand(X_l)$ and $LFCand(X_r)$ are already computed, a sorted list for $LFCand(X_i)$ can be computed in $O(n^3)$ time and $O(n^2)$ space.

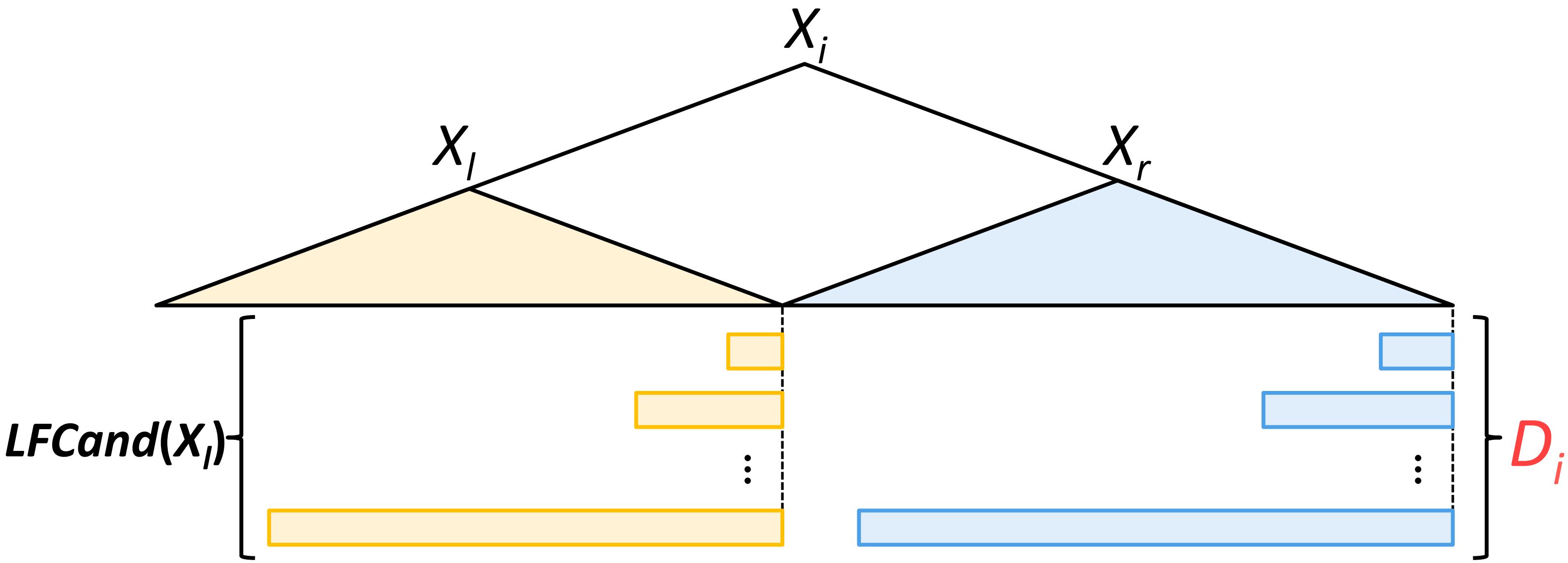


How to compute LFCand



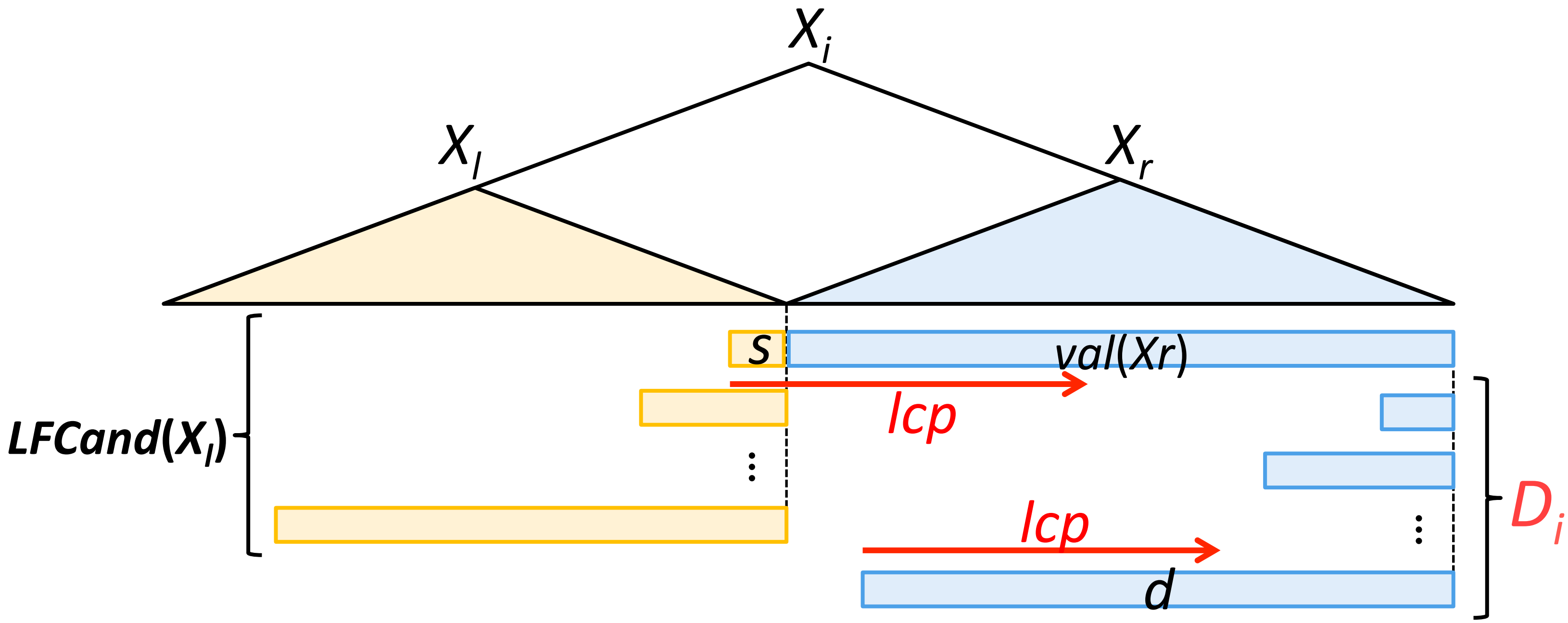
- Let an initially set $D_i \leftarrow LFCand(X_r)$.

How to compute LFCand



- Let an initially set $D_i \leftarrow LFCand(X_r)$.
- We update D_i , and the last D_i is a sorted list of the suffixes of X_i that are candidates of elements of $LFCand(X_i)$.

How to update D_i

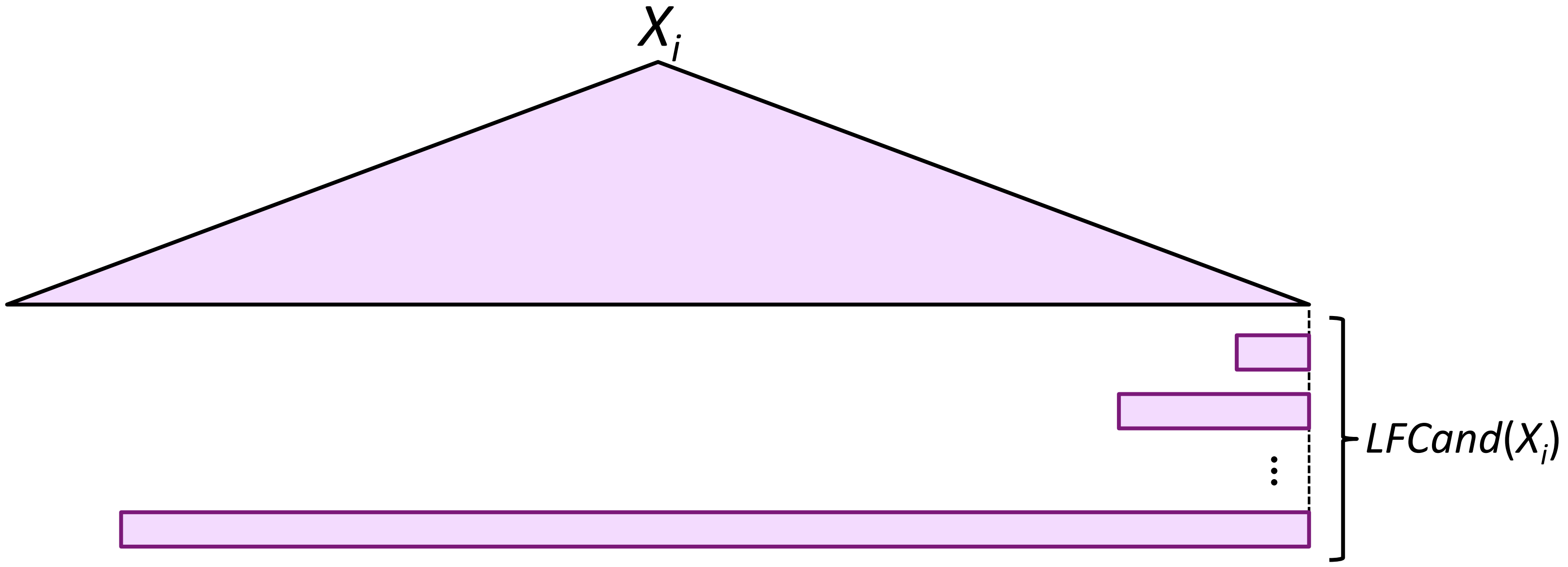


- Let s be the any string of $LFCand(X_l)$.
- We consider whether $s \cdot val(X_r)$ can be in D_i or not by computing the lcp of the longest element d in D_i and $s \cdot val(X_r)$.

Computing LFCands

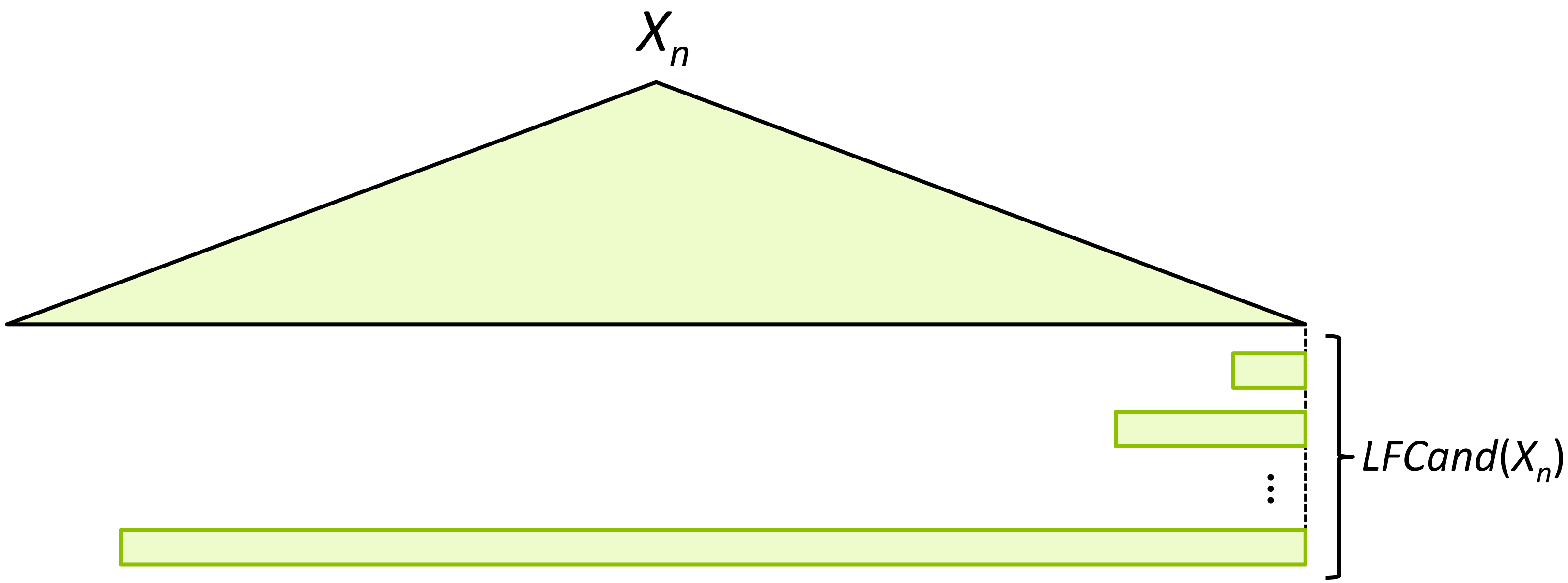
Lemma

Let $X_i = X_l X_r$ be any production of a given SLP S of size n. Provided that sorted lists for $LFCand(X_l)$ and $LFCand(X_r)$ are already computed, a sorted list for $LFCand(X_i)$ can be computed in $O(n^3)$ time and $O(n^2)$ space.



Computing the smallest suffix

We can compute $LFCand(X_n)$ in total of $O(n^4)$ time.



- For all i ($1 \leq i \leq n$), we compute $LFCand(X_i)$ in $O(n^3)$ time, thus we can compute $LFCand(X_n)$ in $O(n^4)$ time.

Complexity of the algorithm

- We repeat the following procedures m times.
  - Compute the $LFCand(X)$ in $O(n^4)$ time, $O(n^2)$ space.
  - Compute a new SLP Y that represents the remaining string in $O(n)$ time.
- In addition to the above procedure, $LFCand(X_i)$ for each variable X_i requires $O(\log N)$ space.

We can compute $LF(w)$ in a total of $O(mn^4)$ time.

We can compute $LF(w)$ in a total of $O(n^2 + n \log N) = O(n^2)$ space.

Coming Soon. (Hopefully)

Theorem [submitted]

Given an SLP S of size n and height h representing a string w of length N, we can compute $LF(w)$ in $O(nh\,(n + \log N \log n))$ time and $O(n^2)$ space.

- In this result, the size m of $LF(w)$ is not written explicitly.
- We show the following interesting lemma.

Lemma

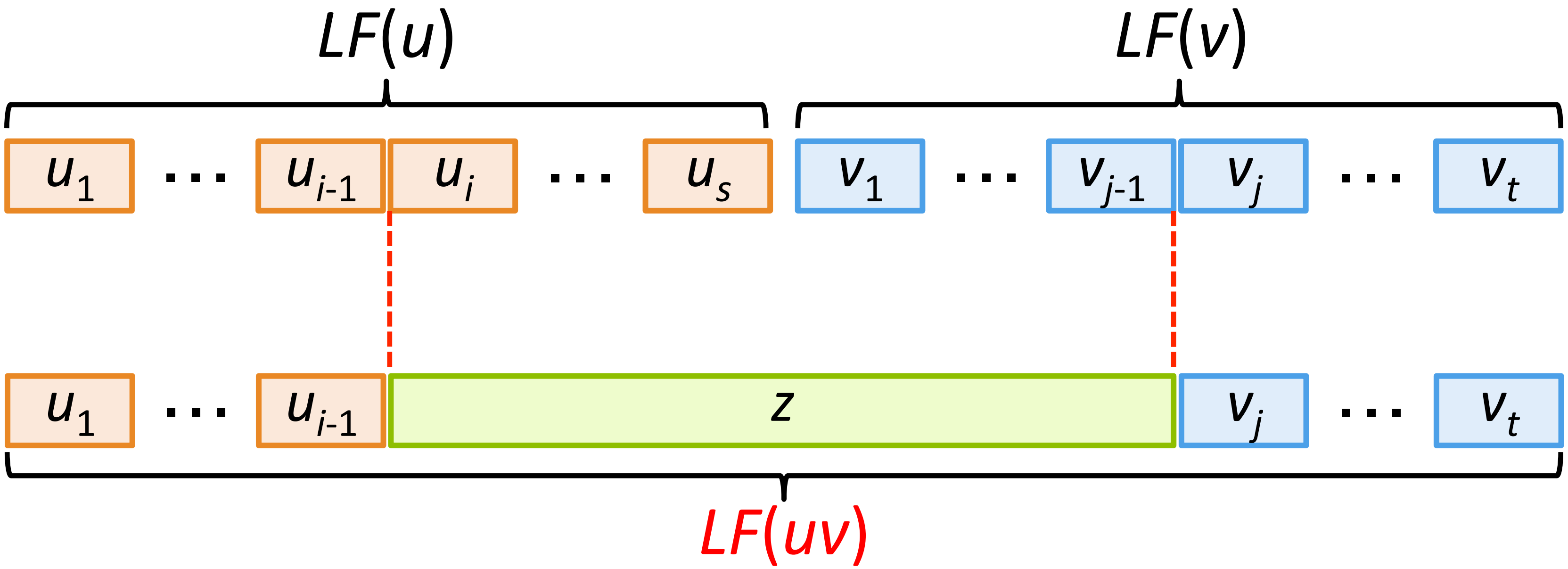
Let n be the size of any SLP representing a string w. The size m of the Lyndon factorization of w is at most n.

Lyndon factorization of concatenated string

Lemma [J. W. Daykin, et al., 1994][A. Apostolico, et al., 1995]

Let $LF(u) = u_1, \ldots, u_s$ and $LF(v) = v_1, \ldots, v_t$ with u, v, u_i, $v_j \in \Sigma^*$, $1 \leq s$, $1 \leq j \leq t$ .Then either $LF(uv) = u_1, \ldots, u_s$, $v_1, \ldots, v_t$ or $LF(uv) = u_1, \ldots, u_{i-1}, z, v_{j+1}, \ldots, v_t$ with $z = u_i \ldots u_s v_1 \ldots v_j$ for some $1 \leq i \leq s$, $1 \leq j \leq t$.

Lyndon factorization of concatenated string



- This lemma implies that we can obtain $LF(uv)$ from $LF(u)$ and $LF(v)$ by computing z since the other Lyndon factors remain unchanged in uv.

Conclusion

Theorem [CPM 2013]

Given an SLP S of size n representing a string w of length N, we can compute $LF(w)$ in $O(mn^4)$ time and $O(n^2)$ space, where m is the number of factors in $LF(w)$.

Theorem [submitted]

Given a SLP S of size n and height h representing a string w of length N, we can compute $LF(w)$ in $O(nh\,(n + \log N \log n))$ time and $O(n^2)$ space.

Thank you !