WAAC2024

LZ78 Substring Compression in CDAWG-compressed space

Hiroki Shibata[1], Dominik Köppl[2]

1. Kyushu University

2. University of Yamanashi

LZ78 Factorization [Ziv & Lempel, '78]

The LZ78 factorization of a string T is a factorization

$$T = F_0 F_1 \dots F_f$$

T' is the unfactorized part of T.

where F_i $(i \geq 1)$ is the longest prefix of $T' = F_i \dots F_f$ that can be represented by $F_i = F_j c$ $(j < i, c \in \Sigma)$.

$$T = \text{abaabaac} = F_0 F_1 \dots F_f$$

$$F = (\varepsilon, \text{a}, \text{b}, \text{aa}, \text{ba}, \text{ac})$$

F' is a compressed representation of T.

$$F' = ((0, \text{a}), (0, \text{b}), (1, \text{a}), (2, \text{a}), (1, \text{c}))$$

($F_0 = \varepsilon$ is the empty string)

Substring Compression Problem

- **The text T of length n is given in advance.**
  - We can construct an index of T before the queries.
- **Queries:**
  - Input: two integers l, r $(1 \leq l \leq r \leq n)$
  - Output: The compressed representation of $T[l..r]$.

We only consider the case $(l, r) = (1, n)$ in this presentation, but the algorithms can be extended for arbitrary (l, r).

T

→ The LZ78 factorization of $T[l..r]$ is $(\mathrm{b}, \mathrm{a}, \mathrm{ab}, \mathrm{aa})$.

Previous Work

We focus on LZ78 substring compression problem.

Recently, [Köppl, '21] proposed a method for LZ78 substring compression.

- **This method is time optimal but consumes $O(n)$ words of space.**

Time/Space Complexity

	Space for Index	Working Space	Preprocessing Time	Query Time
[Köppl, '21]	$O(n)$	$O(z_{l,r})$	$O(n)$	$O(z_{l,r})$

($z_{l,r}$ is the number of LZ78 factors of $T[l..r]$.)

Our Work

We propose a method for LZ78 substring compression in compressed space.

Time/Space Complexity

	Space for Index	Working Space	Preprocessing Time	Query Time
[Köppl, '21]	$O(n)$	$O(z_{l,r})$	$O(n)$	$O(z_{l,r})$
Ours	$O(e)$	$O(z_{l,r})$	$O(n)$	$O(z_{l,r} \log n)$

Our method is the space-efficient variant of [Köppl, '21], which will be recalled in the next slides.

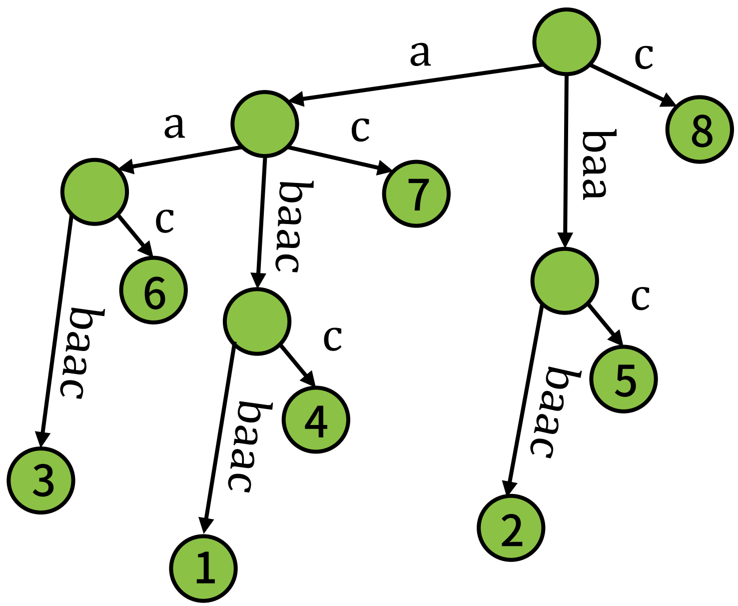
($z_{l,r}$ is the number of LZ78 factors of $T[l..r]$.)

(e is the number of the edges of the CDAWG of T.)

Computing LZ78 by suffix trees (previous method)

Suffix Trees (ST)

Suffix Tree: The compact trie representing all suffixes of T.



$T = \text{abaabaac}$

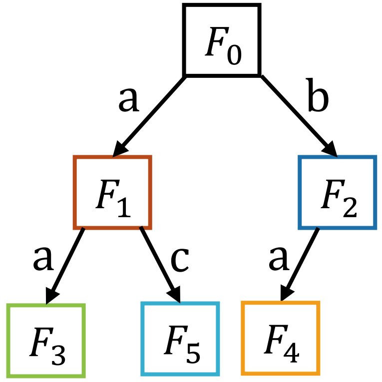
LZ78 Tries

LZ78 Trie: The trie consisting of all LZ78 factors of T.

- **The nodes have a one-to-one correspondence with each LZ78 factor.**

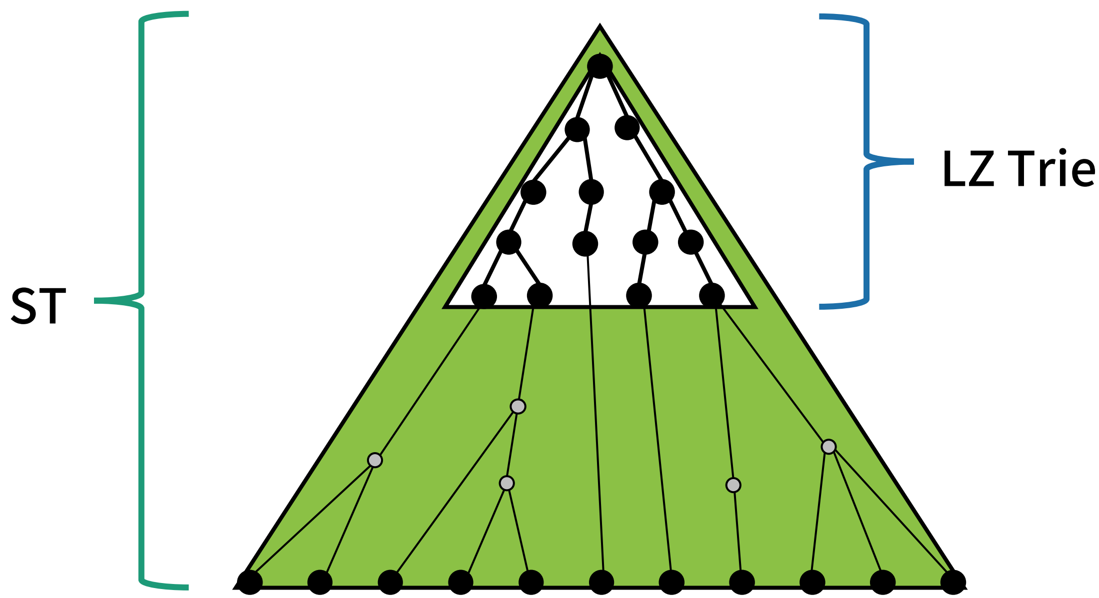
$$T = \text{abaabaac} = F_0 F_1 \dots F_f$$

$$F = (\varepsilon, \text{a}, \text{b}, \text{aa}, \text{ba}, \text{ac})$$



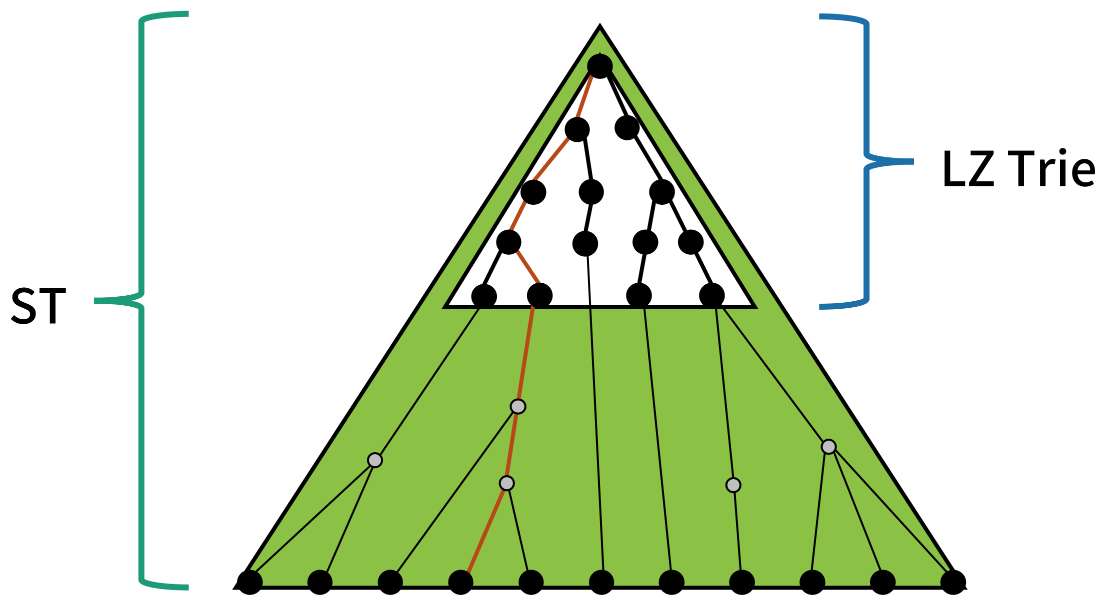
Superimposing LZ78 trie onto ST

- **We can superimpose the LZ78 trie onto the ST.**
  - The LZ trie is an induced subgraph over the ST consisting only of the LZ78 factors.



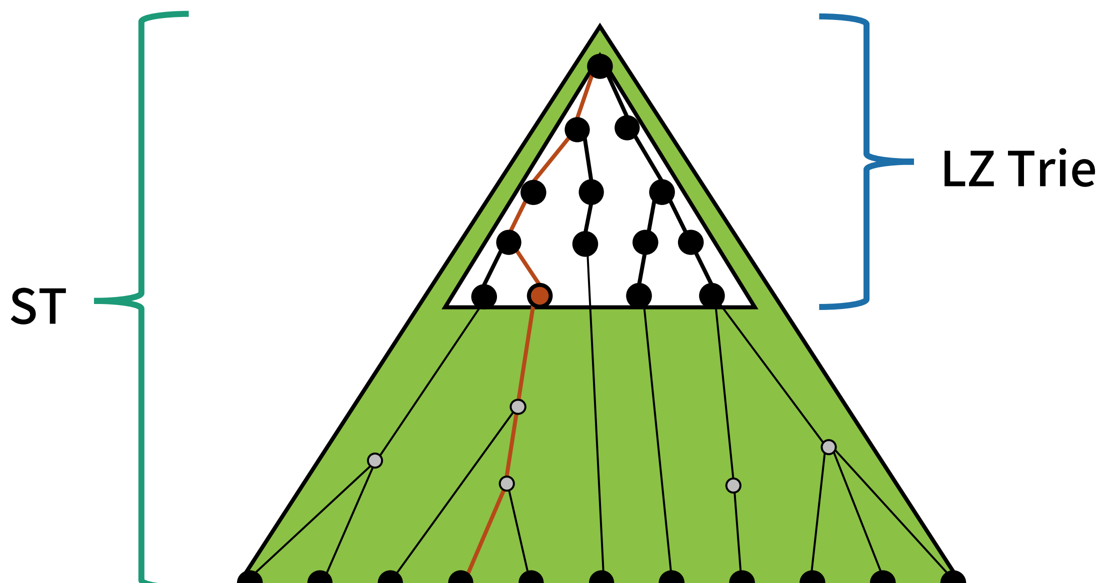
Computing LZ78 by ST

- The new factor can be computed as follows:
  1. Find the path representing the unfactorized part of T.
  2. Find the lowest LZ78 node on the path.
  3. Add an LZ78 node immediately below the lowest LZ78 node.



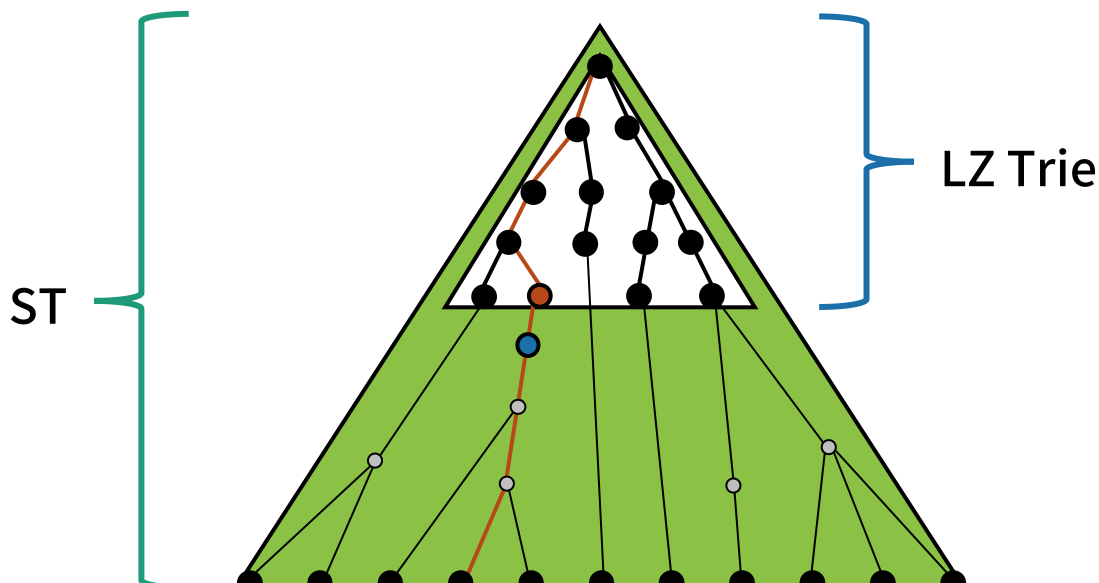
Computing LZ78 by ST

- The new factor can be computed as follows:
  1. Find the path representing unfactorized suffix of T.
  2. Find the lowest LZ78 node on the path.
  3. Add an LZ78 node immediately below the lowest LZ78 node.



Computing LZ78 by ST

- **The new factor can be computed as follows:**
  1. Find the path representing unfactorized suffix of T.
  2. Find the lowest LZ78 node on the path.
  3. Add an LZ78 node immediately below the lowest LZ78 node.



Implementation and Complexity

Required data structures:

- Lowest Marked Ancestor (LMA) [Westbrook, 1992]
  - amortized O(1) time / query
- Weighted Level Ancestor [Gawrychowski et al., 2014]
  - O(1) time / query

Time / Space complexity

- Each data structure can be stored in $O(n)$ words.
- We can find the lowest factor and add a factor in constant time per factor.

■ **Overall Complexity ⇒ $O(n)$ words • $O(z_{l,r})$ time / query**

($z_{l,r}$ is the number of LZ78 factors of $T[l..r]$)

Computing LZ78 in compressed space (proposed method)

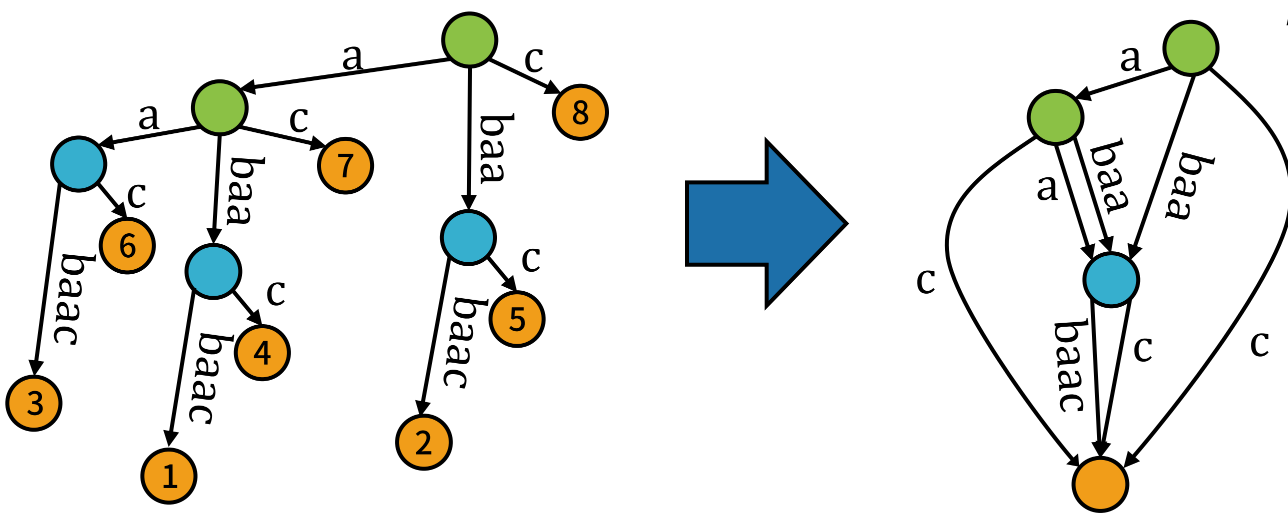
CDAWG (Compact Directed Acyclic Word Graph)

CDAWG: The edge-labelled DAG which obtained by merging isomorphic subtrees of the corresponding ST.

- **Property: The number e of edges of the CDAWG is small for some highly repetitive strings.**

  → CDAWG can be regarded as a compressed representation of STs.

$T = \text{abaabaac}$



Why CDAWG cannot simply replace ST?

Replacing ST with CDAWG is difficult because

1. several edges can end at the same node, and
2. a node of CDAWG can represent several strings.

So we use an additional data structure.

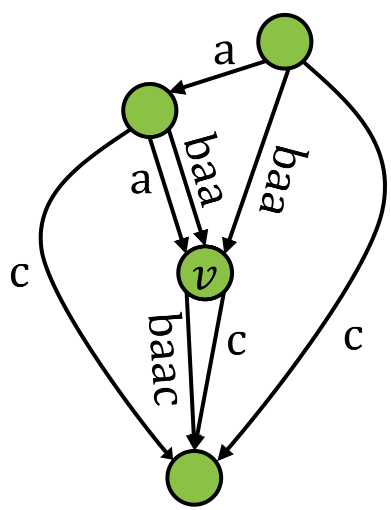


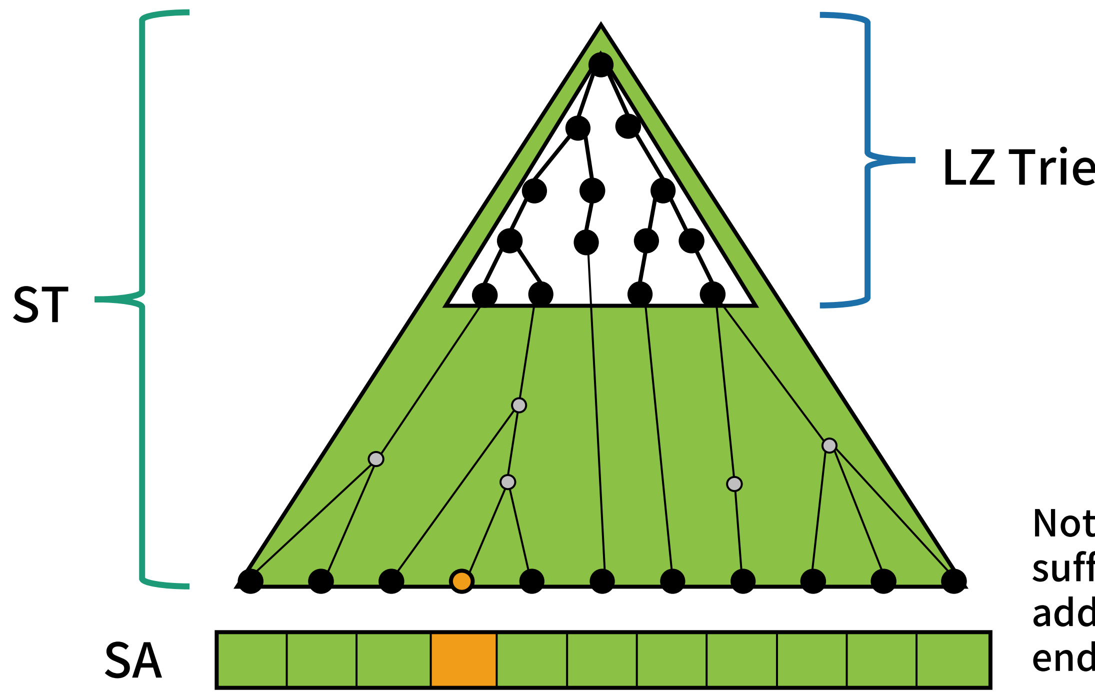
Fig: The CDAWG of $T = \mathrm{abaabaac}$.

- v represents abaa, baa, and ba.
- The indegree of v is 3.

Suffix Arrays (SA)

The suffix array stores the lexicographic order of all suffixes.

- Since the leaves have a one-to-one correspondence to all suffixes, the SA represents the order of the leaves.
- One leaf of the ST corresponds to one position on the SA.

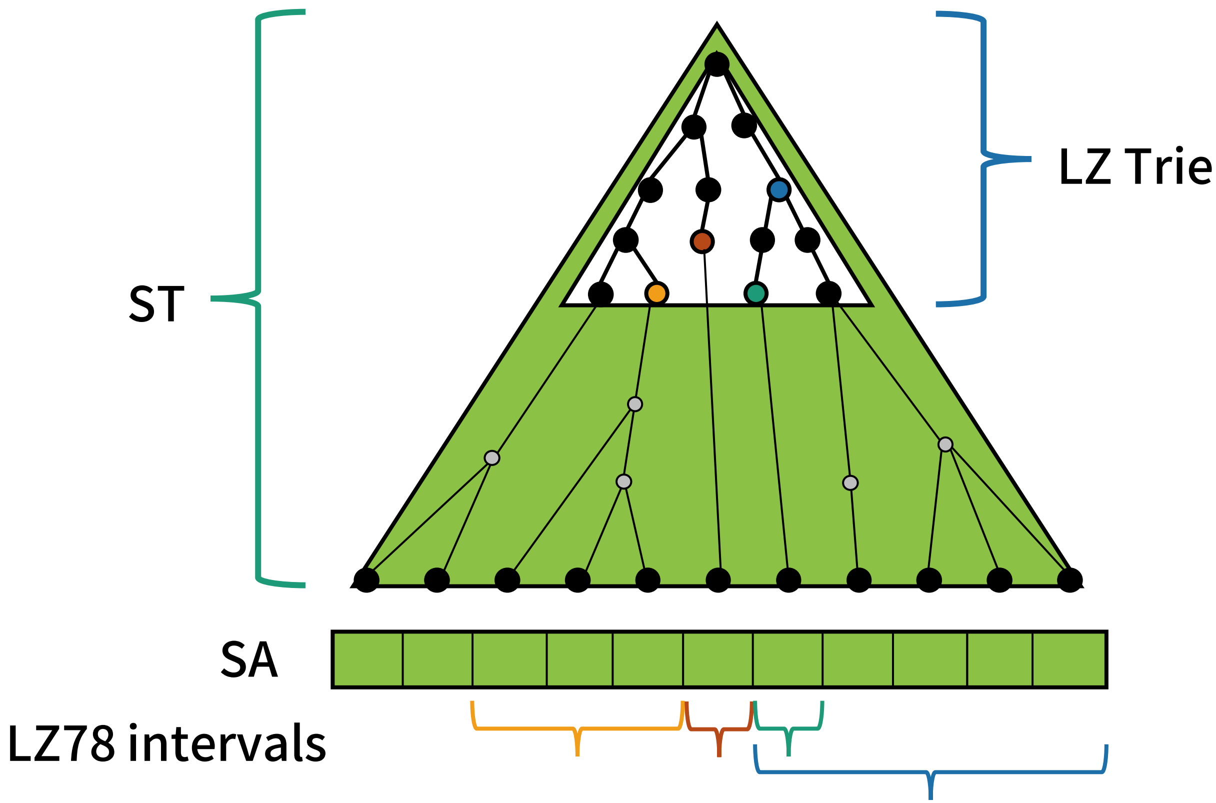


Note: We can guarantee that each suffix corresponds to a leaf node by adding a unique character at the end of T.

Converting LZ78 Nodes to Intervals

Each LZ78 node corresponds to an interval on SA.

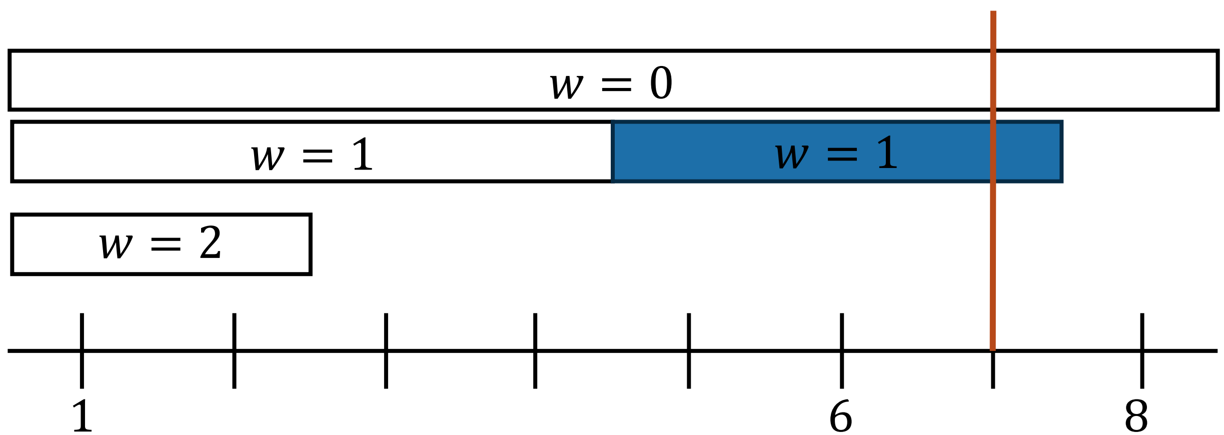
→ The set of all LZ78 nodes can be represented by the set of intervals.



Stabbing-Max Problem

Finding lowest LZ78 node can be reduced to the following queries and operations:

1. **Find the interval $[a, b] \in S$ containing k whose weight is maximum.**
   - Corresponding to find the lowest LZ node.
2. **Add an interval $[a, b]$ with weight w to a set S.**
   - Corresponding to add an LZ node.



Stabbing-Max Problem

Finding lowest LZ78 node can be reduced to the following queries and operations:

1. Find the interval $[a, b] \in S$ containing k whose weight is maximum.
2. Add an interval $[a, b]$ with weight w to a set S.

This problem is known as the stabbing-max problem, and there is a data structure that performs any query with the following complexity [Tarjan, 1979]:

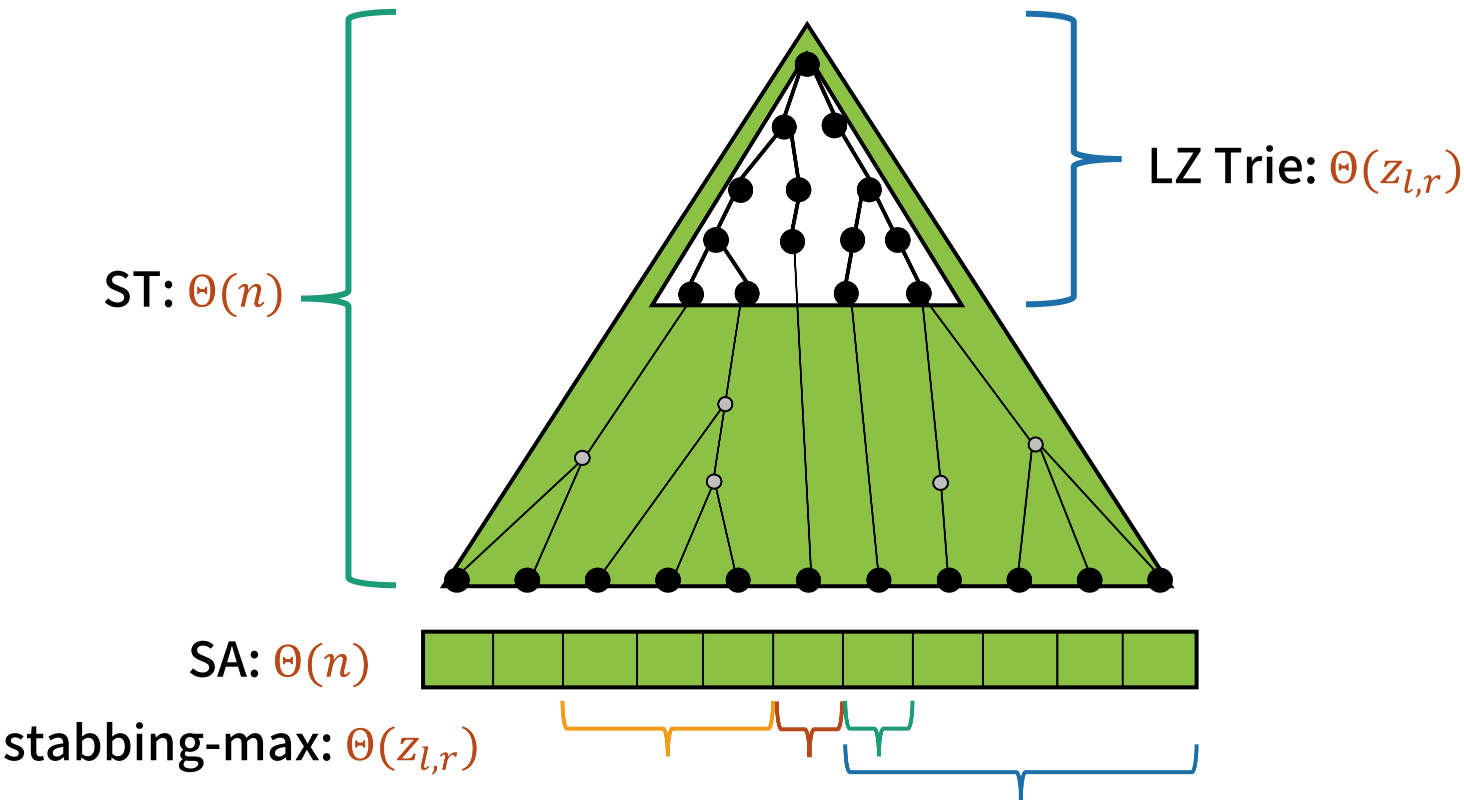
- Add/Find an interval: $O(\log m)$ time / query
- Space complexity: $O(m)$ words

(m is the number of intervals)

Bottoleneck of Space Complexity

Now, our data structure uses $\Theta(n)$ space because of the ST and the SA.

To reduce the space, we need to compute LZ78 factors without them.

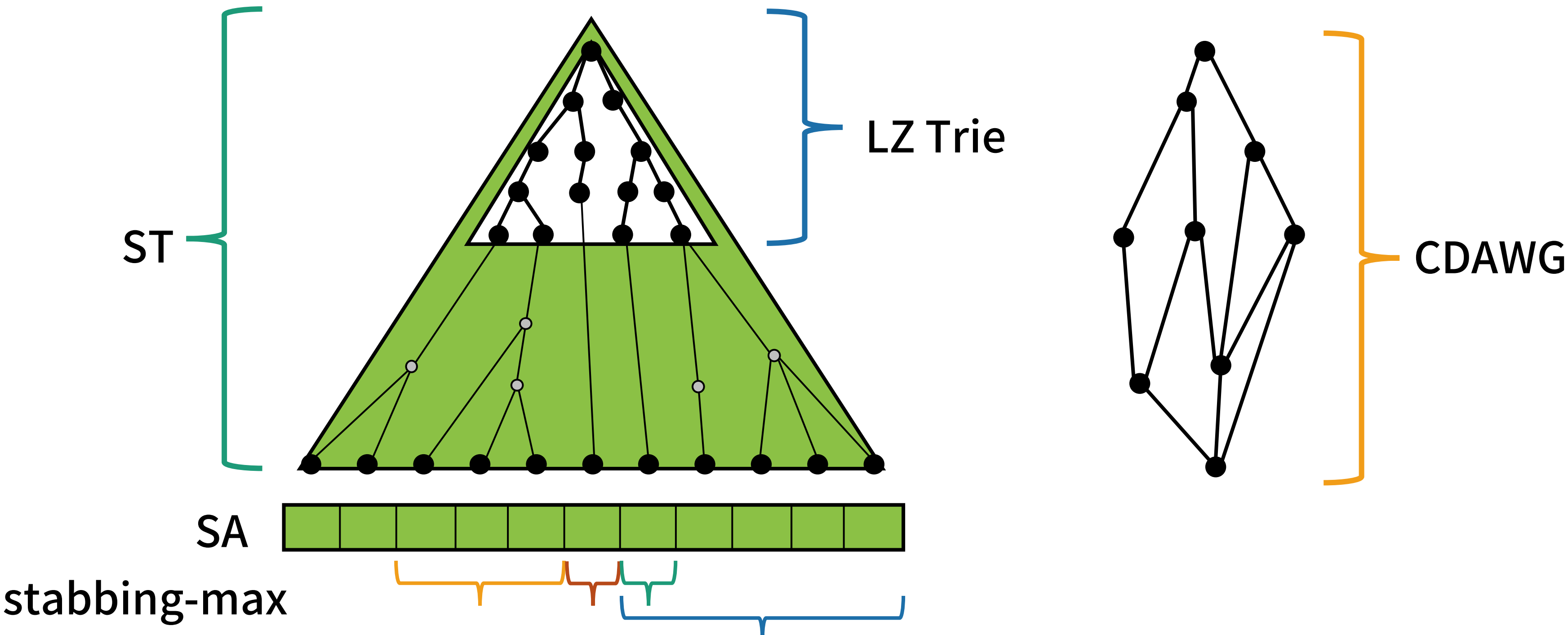


Note: $z_{l,r} \in O(n)$

Replacing ST/SA with CDAWG

Since CDAWG is a compacted variant of ST, some CDAWG-based index efficiently performs ST/SA operation.

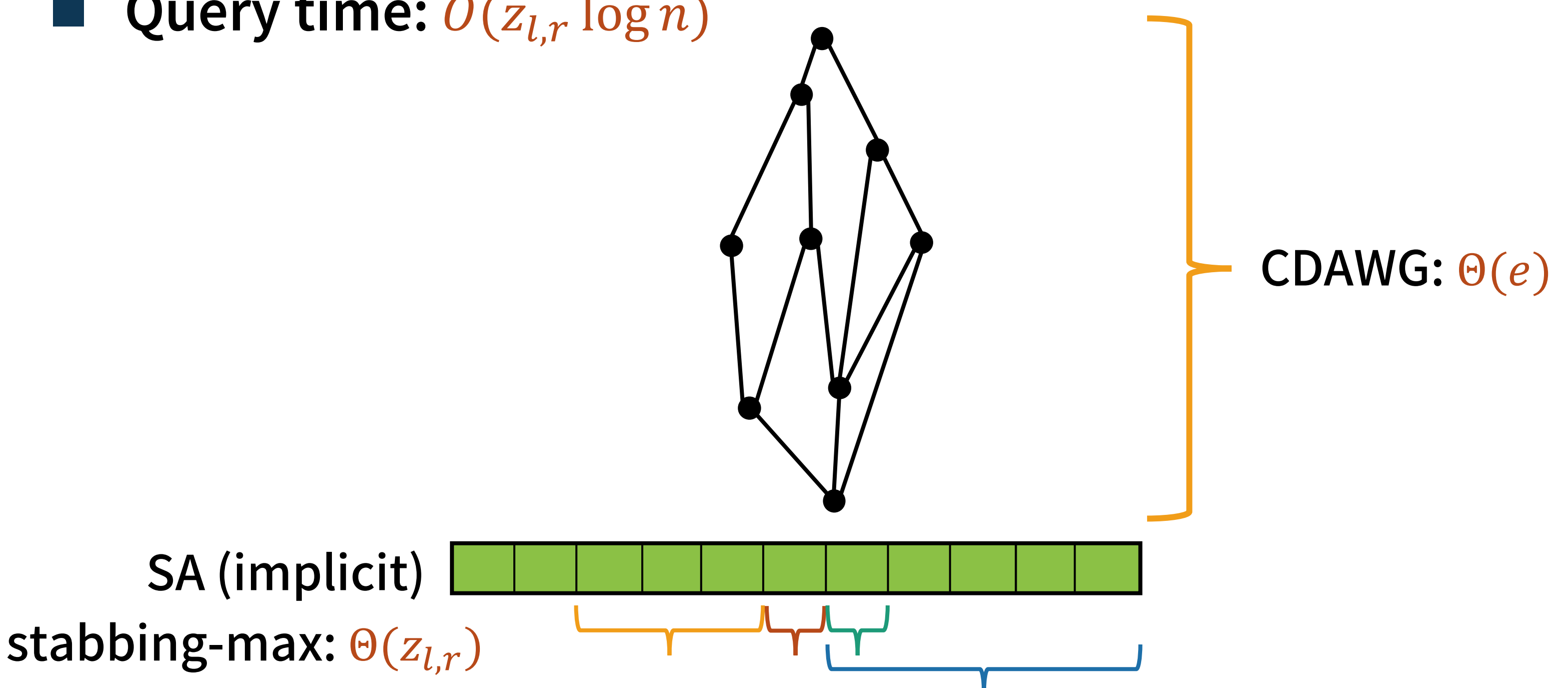
There is a CDAWG-based index that performs ST/SA operation in $O(\log n)$ **time** and is stored in $O(e)$ **space**. [Bealazzougui and Cunial, CPM 2017]



Overall Structure

Our method only uses the CDAWG and the stabbing-max structure.

- Space for index: $\Theta(e)$ words
- Working space for queries: $\Theta(z_{l,r})$ words
- Query time: $O(z_{l,r} \log n)$



Conclusion

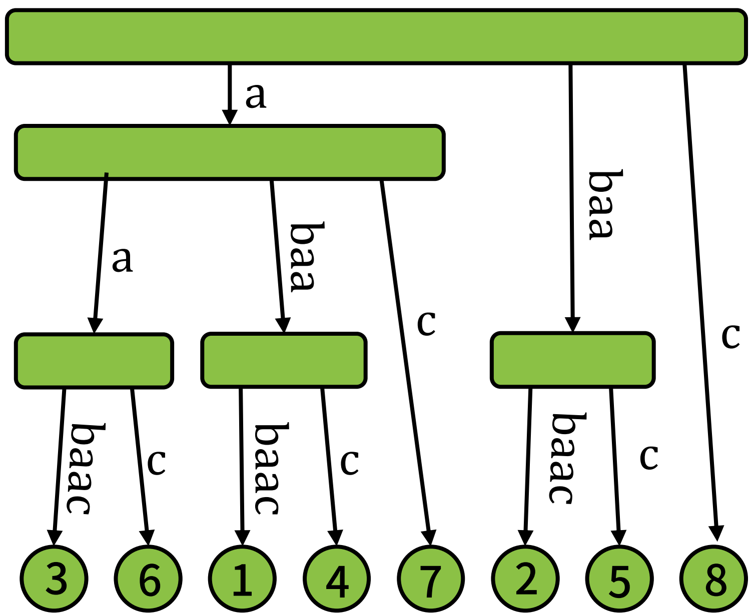
- We proposed a method for LZ78 substring compression in compressed space.
- Some applications:
  - We can replace the CDAWG by some other compressed index.
  - Applying this method for other LZ78-like compressions. (LZD, LZMW)

Time/Space Complexity

	Space for Index	Working Space	Preprocessing Time	Query Time
[Köppl, '21]	$O(n)$	$O(z_{l,r})$	$O(n)$	$O(z_{l,r})$
Ours	$O(e)$	$O(z_{l,r})$	$O(n)$	$O(z_{l,r} \log n)$

Suffix Trees (ST)

Suffix Tree: The compact trie representing all suffixes of T.



$T = \text{abaabaac}$

LZ78 Factorization [Ziv & Lempel, '78]

- One of a text compression method.
- Factorize a string T to $T = F_0F_1 \dots F_f$ by a specific algorithm.

$$T = \text{abbabaaab} = F_0F_1 \dots F_f$$

$$F = (\varepsilon, \text{a}, \text{b}, \text{ba}, \text{baa}, \text{ab})$$

($F_0 = \varepsilon$ is the empty string)

LZ78 Factorization [Ziv & Lempel, '78]

- F_i $(i \geq 1)$ **is computed as follows:**
  1. Compute the longest prefix of $T' = T[1 + |F_0| + |F_1| + \cdots + |F_{i-1}|..n]$ that appears in F.
  2. Set $F_i = F_j T'[\ell + 1]$, where F_j is the longest factor that precedes T'.

$$T = \mathrm{abbabaaab} = F_0 F_1 \ldots F_f \; (F_0 = \varepsilon)$$

$$T' = \mathrm{abbabaaab}$$

$$F = (\varepsilon)$$

LZ78 Factorization [Ziv & Lempel, '78]

- F_i $(i \geq 1)$ **is computed as follows:**
  1. Compute the longest prefix of $T' = T[1 + |F_0| + |F_1| + \cdots + |F_{i-1}|..n]$ that appears in F.
  2. Set $F_i = F_j T'[\ell + 1]$, where F_j is the longest factor that precedes T'.

$$T' = \text{abbabaaab}$$
$$F = (\varepsilon, \text{a})$$

LZ78 Factorization [Ziv & Lempel, '78]

- F_i $(i \geq 1)$ **is computed as follows:**
  1. Compute the longest prefix of $T' = T[1 + |F_0| + |F_1| + \cdots + |F_{i-1}|..n]$ that appears in F.
  2. Set $F_i = F_j T'[\ell + 1]$, where F_j is the longest factor that precedes T'.

$$T' = \text{abbabaaab}$$
$$F = (\varepsilon, \text{a}, \text{b})$$

LZ78 Factorization [Ziv & Lempel, '78]

- F_i $(i \geq 1)$ **is computed as follows:**
  1. Compute the longest prefix of $T' = T[1 + |F_0| + |F_1| + \cdots + |F_{i-1}|..n]$ that appears in F.
  2. Set $F_i = F_j T'[\ell + 1]$, where F_j is the longest factor that precedes T'.

$$T' = \text{abbabaaab}$$
$$F = (\varepsilon, \text{a}, \text{b}, \text{ba})$$

LZ78 Factorization [Ziv & Lempel, '78]

- F_i $(i \geq 1)$ **is computed as follows:**
  1. Compute the longest prefix of $T' = T[1 + |F_0| + |F_1| + \cdots + |F_{i-1}|..n]$ that appears in F.
  2. Set $F_i = F_j T'[\ell + 1]$, where F_j is the longest factor that precedes T'.

$$T' = \text{abbabaaab}$$
$$F = (\varepsilon, \text{a}, \text{b}, \text{ba}, \text{baa})$$

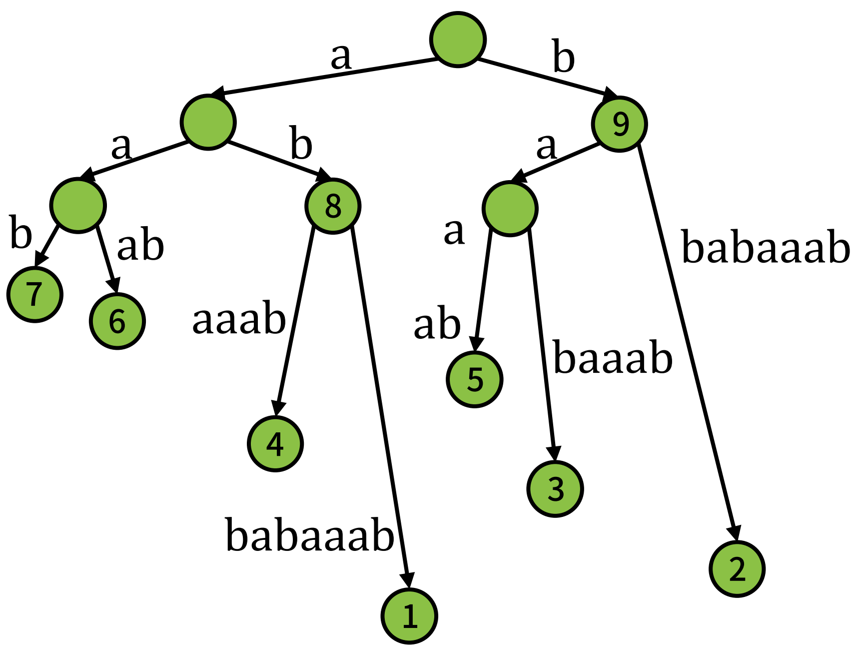
LZ78 Factorization [Ziv & Lempel, '78]

- F_i $(i \geq 1)$ **is computed as follows:**
  1. Compute the longest prefix of $T' = T[1 + |F_0| + |F_1| + \cdots + |F_{i-1}|..n]$ that appears in F.
  2. Set $F_i = F_j T'[\ell + 1]$, where F_j is the longest factor that precedes T'.

$$T' = \text{abbabaaab}$$
$$F = (\varepsilon, \text{a}, \text{b}, \text{ba}, \text{baa}, \text{ab})$$

Suffix Trees (ST)

Suffix Tree: The compact trie representing all suffixes of T.



$T = \text{abbabaaab}$

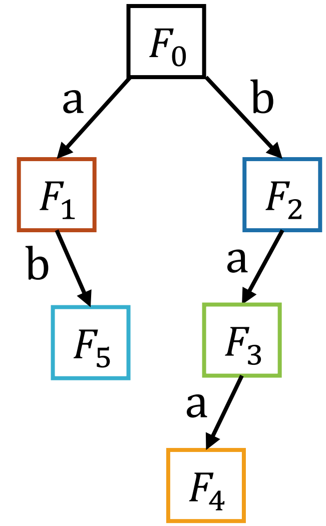
LZ78 Tries

LZ78 Trie: The trie consisting of all LZ78 factors of T.

- **The nodes have a one-to-one correspondence with each LZ78 factor.**

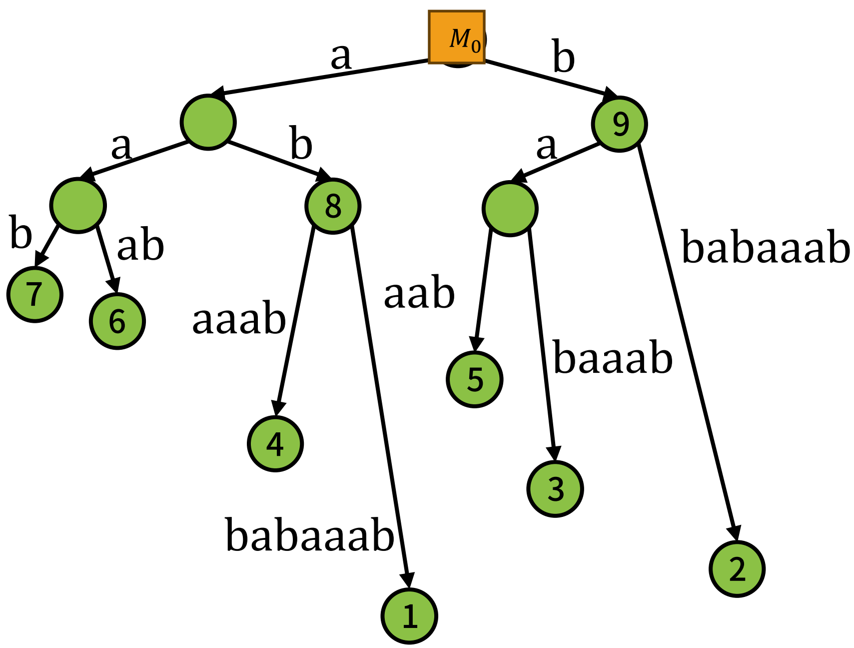
$$T = \text{abbabaaab} = F_0 F_1 \ldots F_f$$

$$F = (\varepsilon, \text{a}, \text{b}, \text{ba}, \text{baa}, \text{ab})$$



Computing LZ78 by ST

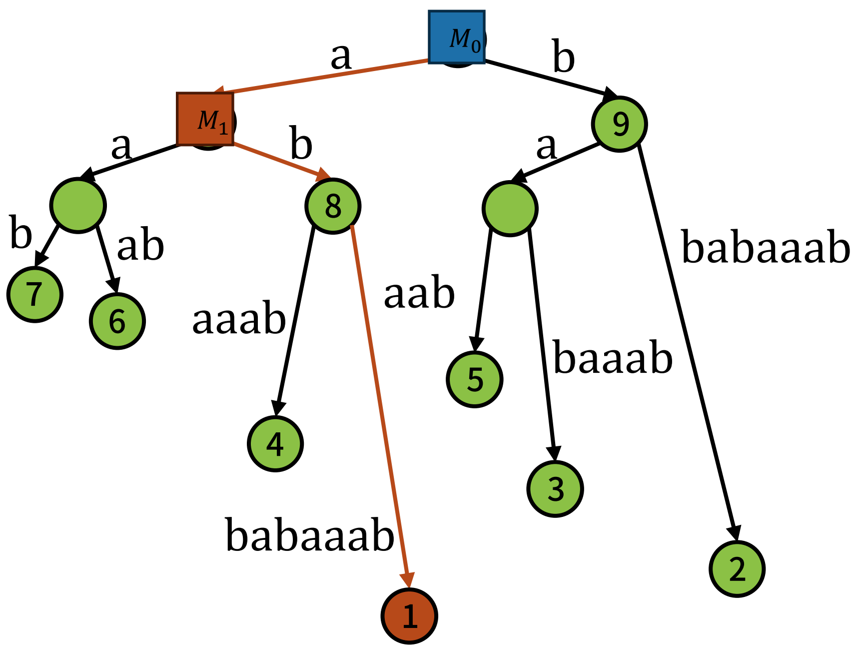
- F_i $(i \geq 1)$ **can be computed as follows:**
  1. Find the path on the ST representing $T' = T[1 + |F_0| + |F_1| + \cdots, |F_{i-1}|..n]$.
  2. Find the lowest mark M_j on the path.
  3. Set $F_i = F_j T'[|F_j| + 1]$ and add a mark M_i to the position on the ST corresponding to F_i.



$T' = \text{abbabaaab}$
$F \ = (\varepsilon)$

Computing LZ78 by ST

- F_i $(i \geq 1)$ **can be computed as follows:**
  1. Find the path on the ST representing $T' = T[1 + |F_0| + |F_1| + \cdots, |F_{i-1}|..n]$.
  2. Find the lowest mark M_j on the path.
  3. Set $F_i = F_j T'\left[\left|F_j\right| + 1\right]$ and add a mark M_i to the position on the ST corresponding to F_i.

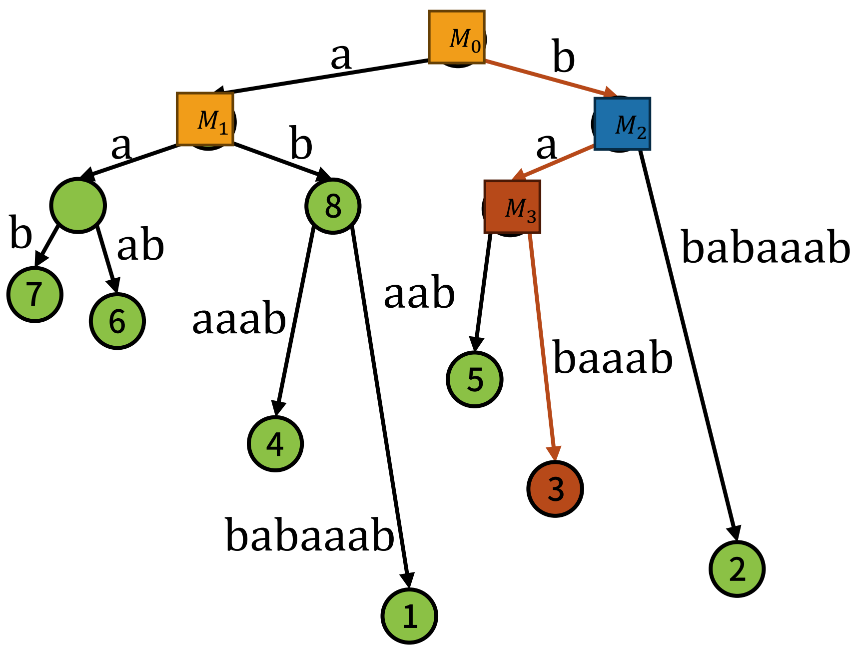


$T' = \text{abbabaaab}$

$F = (\varepsilon, \text{a})$

Computing LZ78 by ST

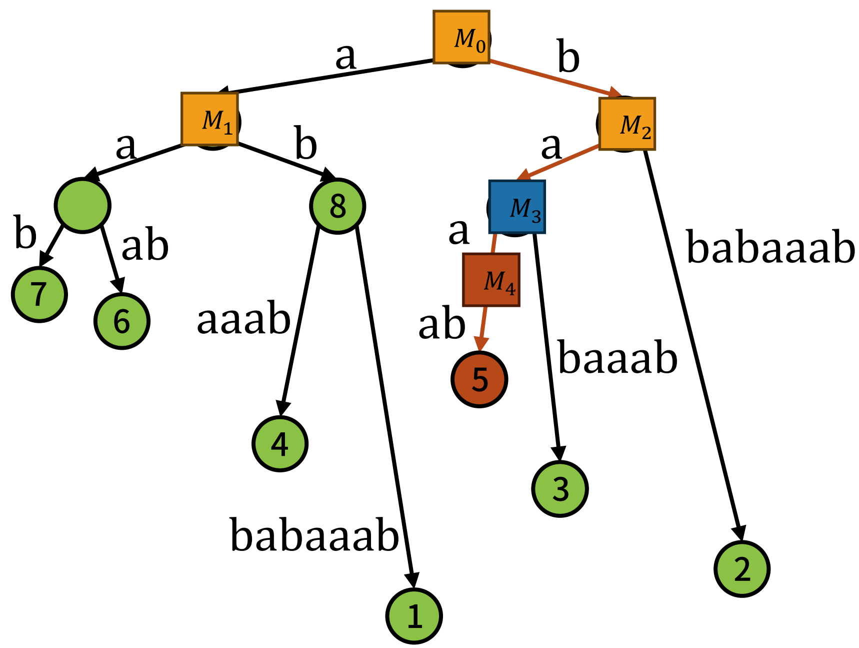
- F_i $(i \geq 1)$ **can be computed as follows:**
  1. Find the path on the ST representing $T' = T[1 + |F_0| + |F_1| + \cdots, |F_{i-1}|..n]$.
  2. Find the lowest mark M_j on the path.
  3. Set $F_i = F_j T'[|F_j| + 1]$ and add a mark M_i to the position on the ST corresponding to F_i.



$$T' = \text{abbabaaab}$$
$$F = (\varepsilon, \text{a}, \text{b}, \text{ba})$$

Computing LZ78 by ST

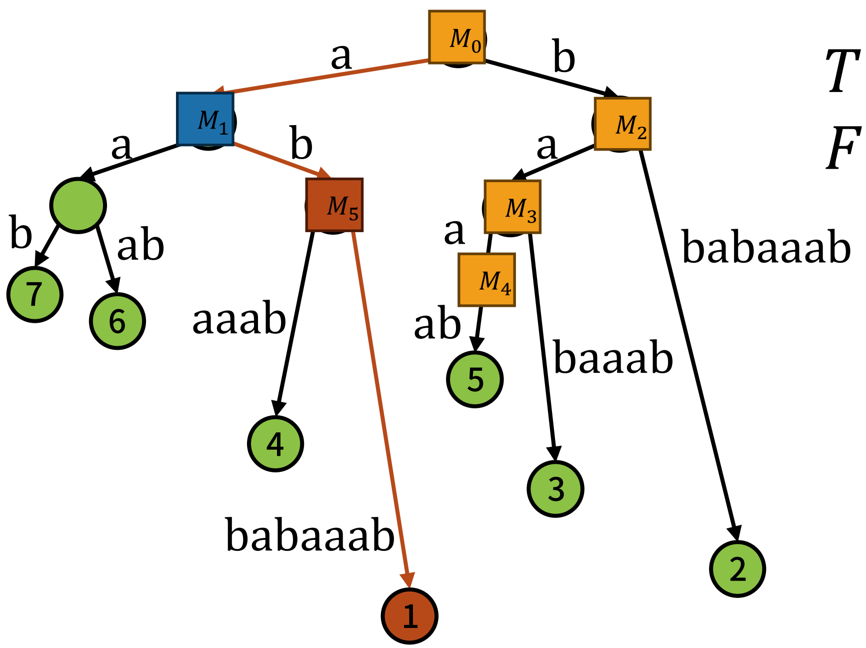
- F_i $(i \geq 1)$ **can be computed as follows:**
  1. Find the path on the ST representing $T' = T[1 + |F_0| + |F_1| + \cdots, |F_{i-1}|..n]$.
  2. Find the lowest mark M_j on the path.
  3. Set $F_i = F_j T'\big[|F_j| + 1\big]$ and add a mark M_i to the position on the ST corresponding to F_i.



$$T' = \text{abbababaaab}$$
$$F = (\varepsilon, \text{a}, \text{b}, \text{ba}, \text{baa})$$

Computing LZ78 by ST

- F_i $(i \geq 1)$ **can be computed as follows:**
  1. Find the path on the ST representing $T' = T[1 + |F_0| + |F_1| + \cdots, |F_{i-1}|..n]$.
  2. Find the lowest mark M_j on the path.
  3. Set $F_i = F_j T'\big[|F_j| + 1\big]$ and add a mark M_i to the position on the ST corresponding to F_i.

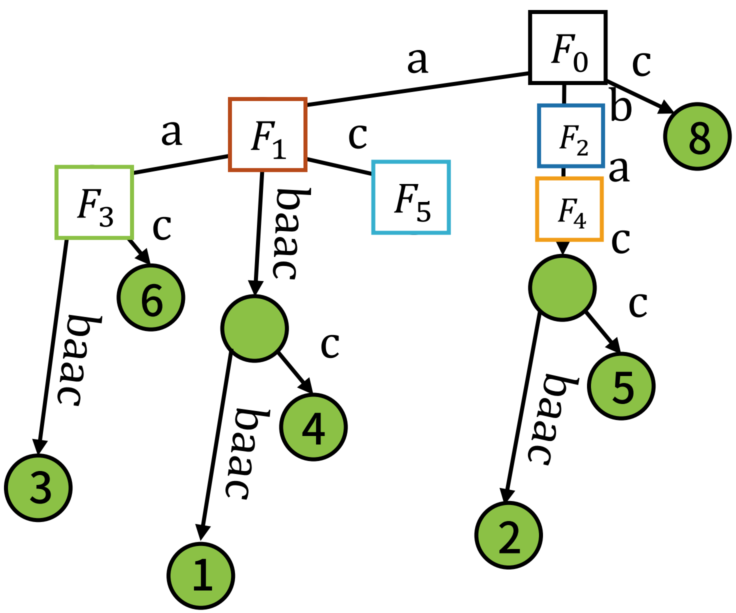


$T' = \text{abbabaaaab}$

$F = (\varepsilon, \text{a}, \text{b}, \text{ba}, \text{baa}, \text{ab})$

Superimposing LZ78 trie and the ST

- We superimpose the LZ78 trie on the ST.



T = abaabaac

F = (ε, a, b, aa, ba, ac)