2026/2/6 Sequences in London 2026

Online Computation of Maximal Closed Substrings

Hiroki Shibata[*1], Haruki Umezaki[*1],

Takuya Mieno[*2], Yuto Nakashima[*1], Shunsuke Inenaga[*1]

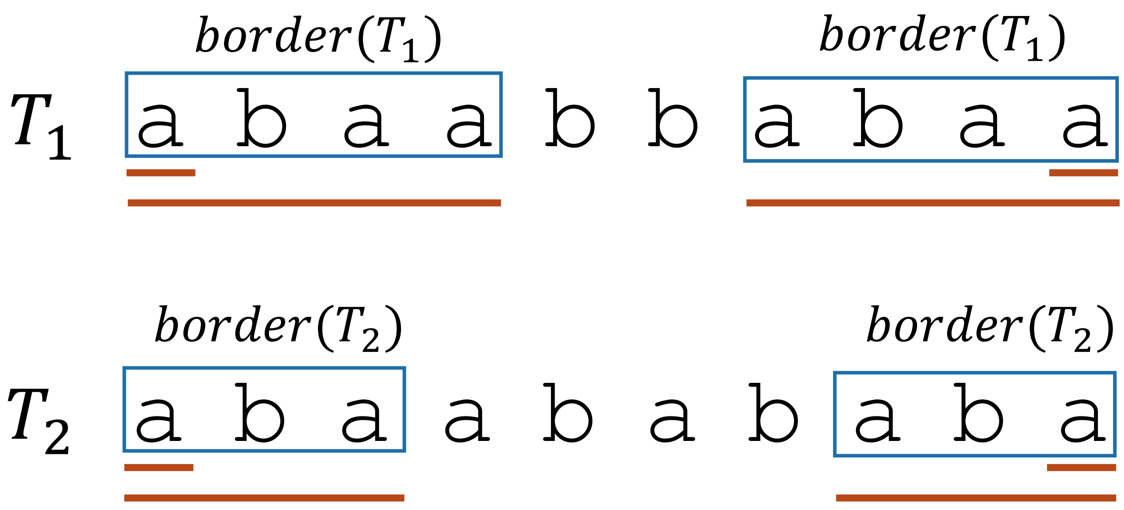
[*1] Kyushu University

[*2] The University of Electro-Communications

Border

Border

A substring that is both a proper prefix and a proper suffix of a string T is called a border of T.



Closed String

Closed String

A string T is called a closed string if the number of occurrences of the longest border of T is exactly two.

A string that is not closed is called an open string.

$border(T_1)$ $border(T_1)$

T_1 [a b a a] b b [a b a a] **Closed**

$border(T_2)$ $border(T_2)$

T_2 [a b a] [a b a] b [a b a] **Open**

Note: strings of length 1 are defined as open strings.

Closed Repeat

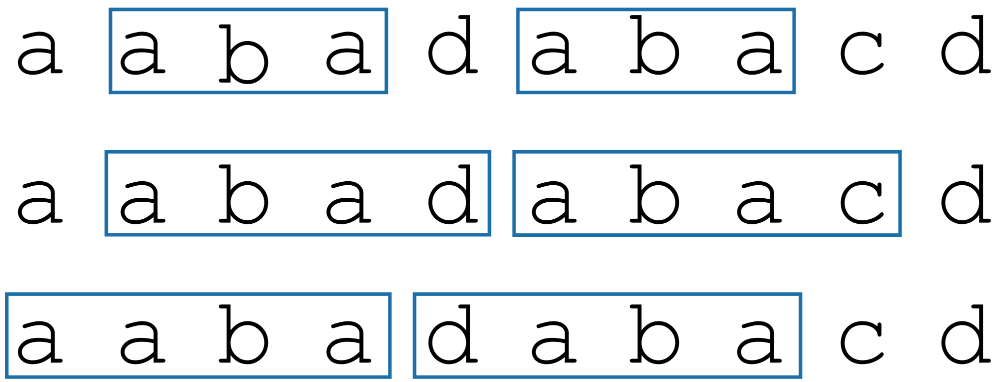
Closed Repeat

A pair of consecutive occurrences of a substring is called a closed repeat if it satisfies both right-maximality and left-maximality:

- Right-maximal: Extending repeats by one character to the right results in a mismatch.
- Left-maximal: Extending repeats by one character to the left results in a mismatch.

The blue-boxed part of the upper right figure is a closed repeat.

- Extending one character to the left or right causes a mismatch (middle / bottom figures)



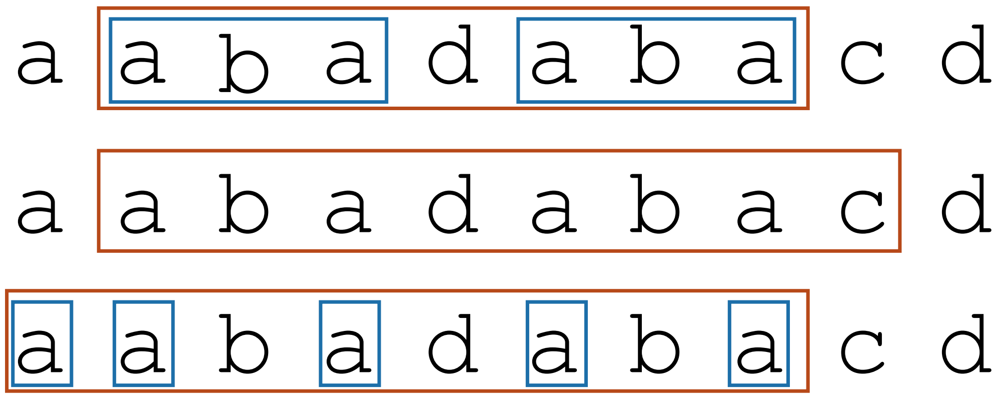
Maximal Closed Substring

Maximal Closed Substring

When a closed substring $T[i..j]$ in T is such that both $T[i-1..j], T[i..j+1]$ are open strings, then $T[i..j]$ is called a maximal closed substring.

The red-boxed part of the upper right figure is an MCS.

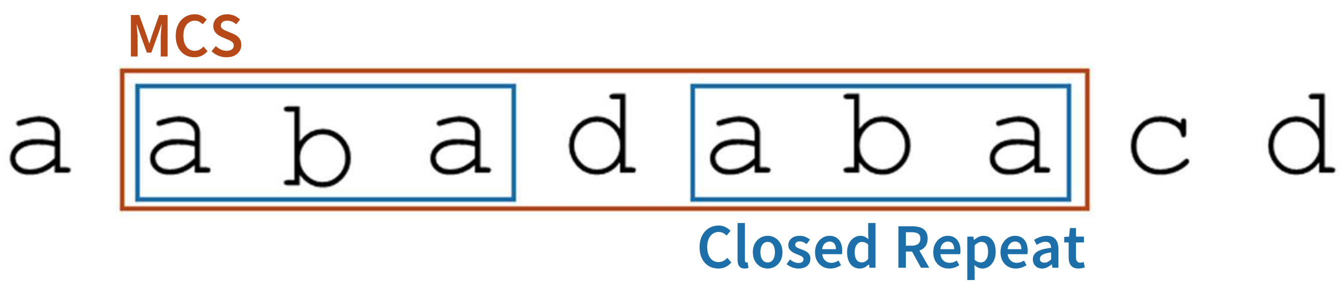
- **The string resulting from extending one character to the left or right is open.**



Relationship Between Closed Repeats and MCSs

Closed repeats and MCSs have a one-to-one correspondence:

- **If $T[i..j']$ is an MCS, then its longest border forms a closed repeat.**
- **If $T[i..i'], T[j..j']$ is a closed repeat, then $T[i..j']$ is an MCS with $T[i..i'] = T[j..j']$ as its longest border.**



Existing Works and Our Result

- $O(n \log n)$ time offline algorithm for computing MCS [Badkobeh et al., 2024]
  - It is time-optimal because the expected number of MCSs in a random binary string is $O(n \log n)$. [Kosolobov, 2024]
- There is no online algorithm yet.

Our result

We can compute MCSs online in amortized $O(\log n)$ time per character.

Note: n denotes the length of the input string.

Main Ideas of Our Method

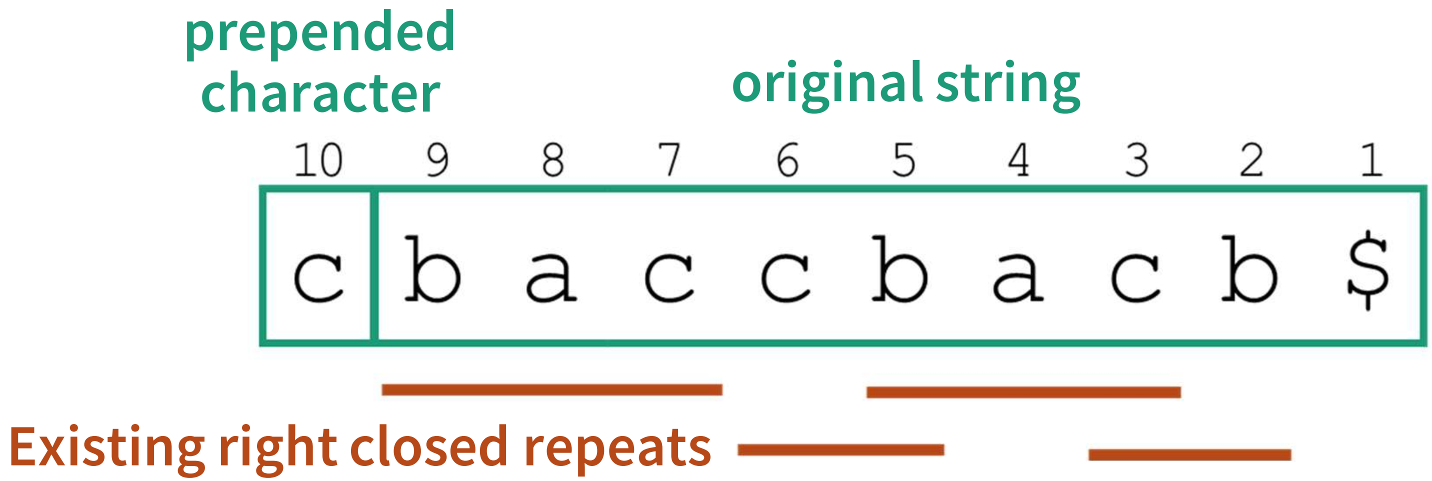
1. Computing right closed repeats online instead of MCSs.
   - Right closed repeat: a variant of closed repeat that does not require left-maximality.
   - We can obtain all closed repeats by checking left-maximality for each right closed repeat, and each closed repeat corresponds to an MCS.
2. Computing right closed repeats by a combination of a suffix tree and a link-cut tree.

Advantage of Considering Right Closed Repeats

Observation

Right closed repeats in the original string remain right closed repeats after adding a character to the beginning.

→ We only need to consider **newly generated right closed repeats**!

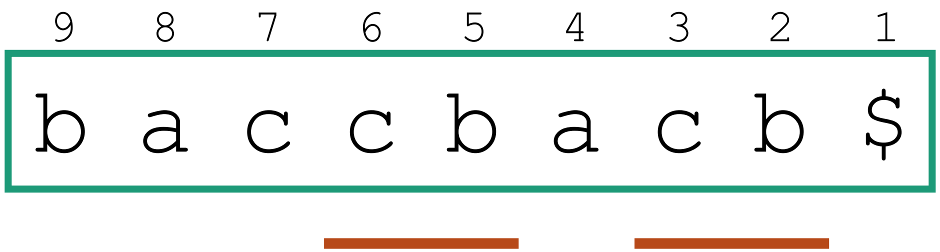


Note: our online algorithm is processed in a right-to-left manner.

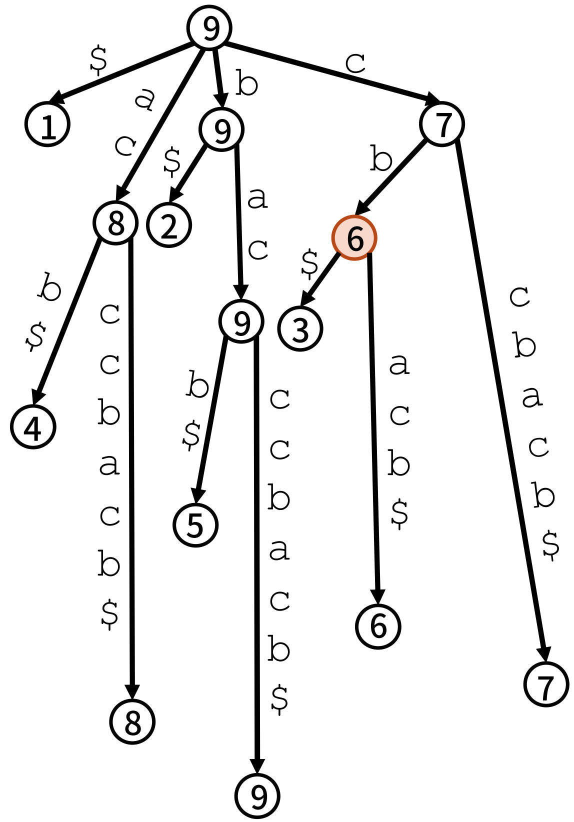
Suffix Tree and Node Labeling

Consider the suffix tree of the string, and label each node with the position of the leftmost occurrence of the string corresponding to that node.

- **Figure example: The red node representing string “cb”:**
  - **“cb” appears at position 6 and 3.**
  - **The leftmost occurrence is 6, so the node label is 6.**



Positions are indexed from right to left because characters are added to the left.

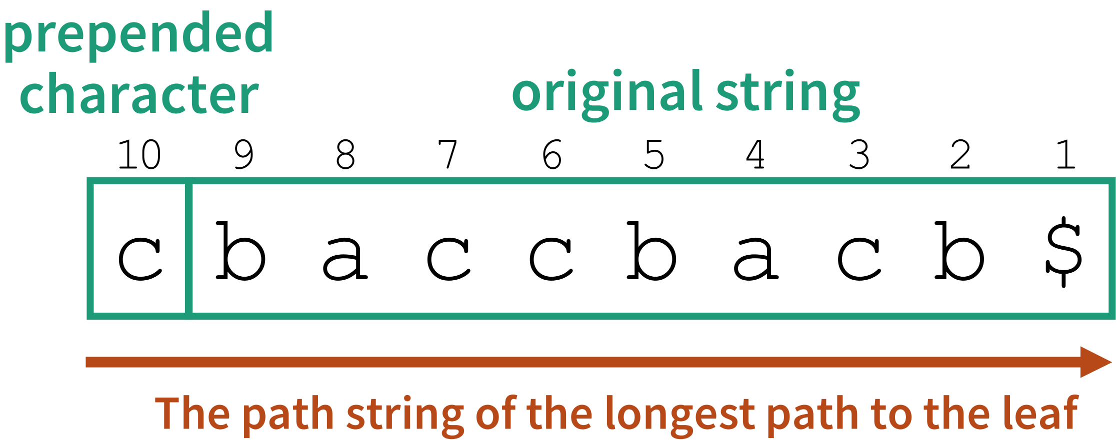


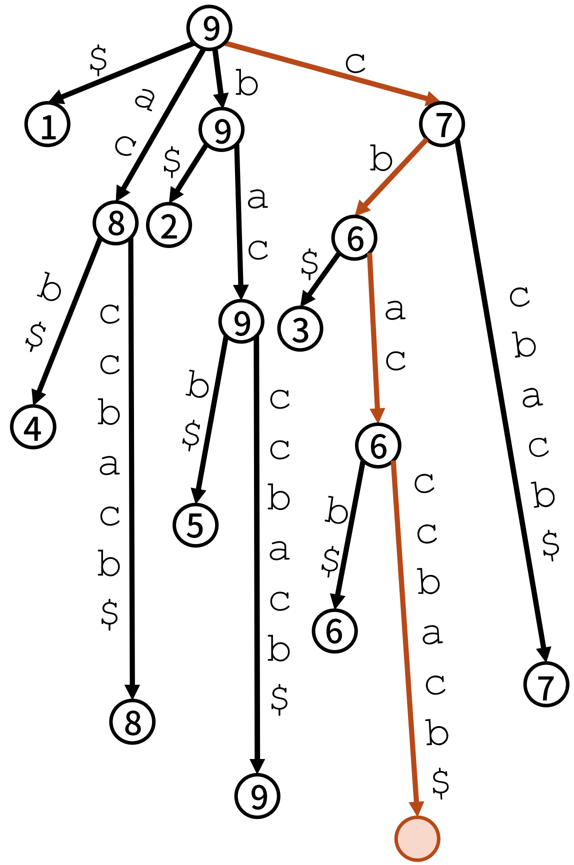
The ST of baccbacb$

Updating the Suffix Tree

The suffix tree can be updated by adding one new leaf representing the entire string.

During this process, we want to compute the new right closed repeats.





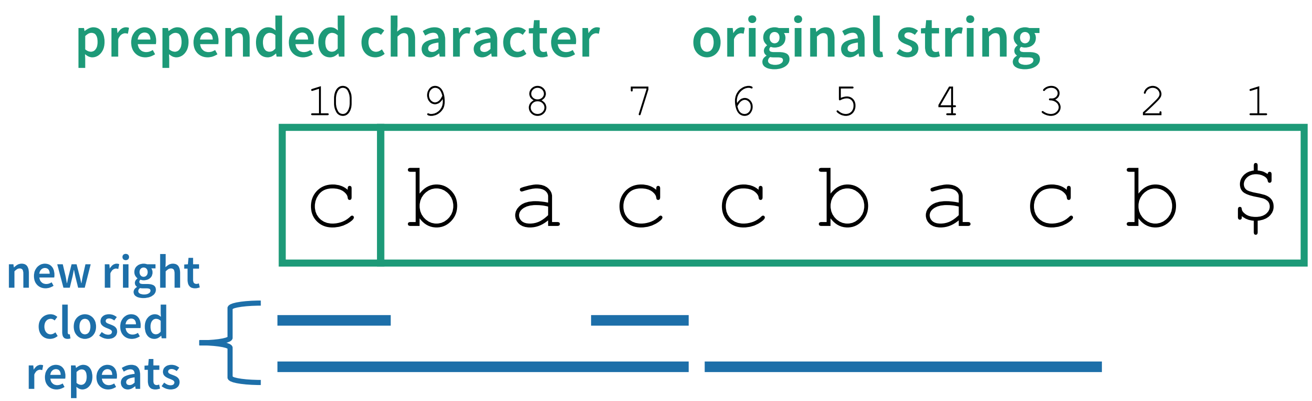
The ST of cbaccbacb$

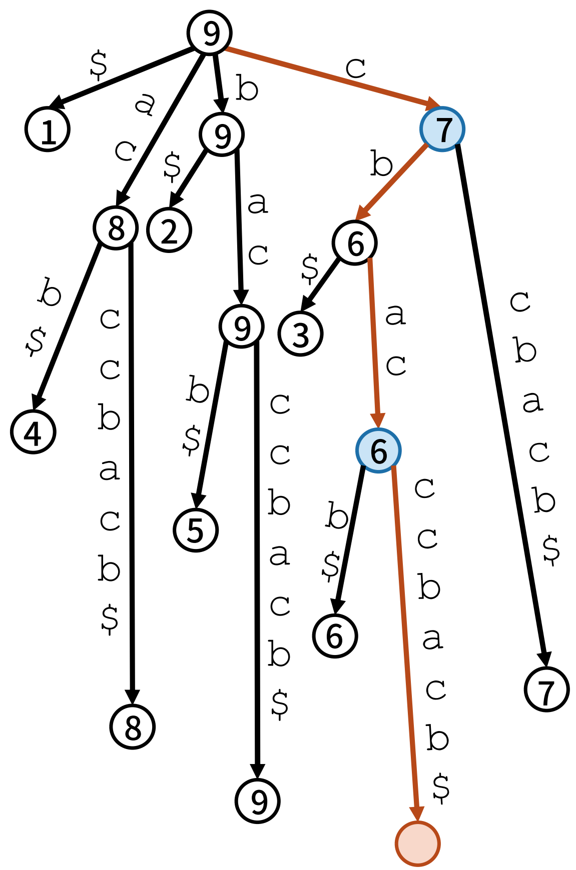
Observation about the New Right Closed Repeats

Observation

New right closed repeats are as follows:

- The Repeat string corresponds to a ST node on the longest path where the node label differs from the label of the node immediately below it on the longest path.
- Left occurrence is a prefix of the updated string.
- Right occurrence ends with the position indicated by the node label.





The ST of cbaccbacb$

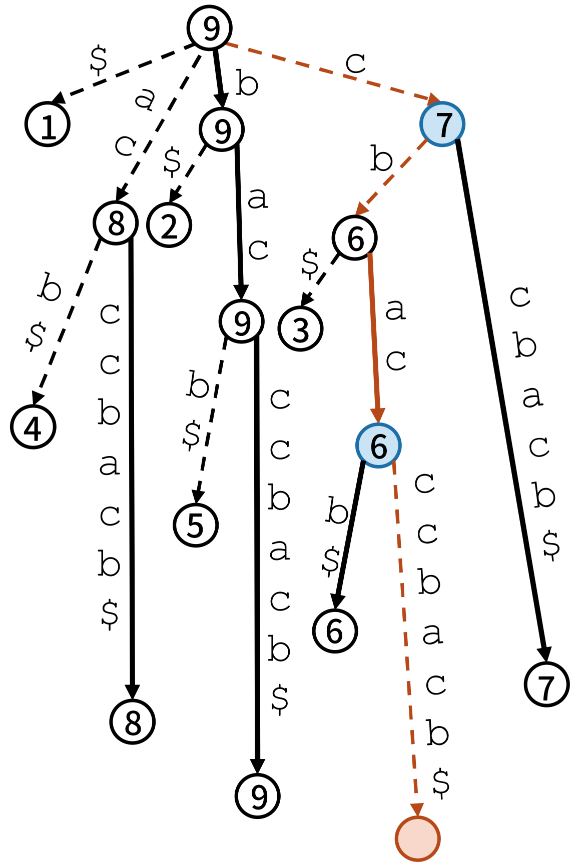
Heavy Edge / Light Edge

We define heavy edges as edges connecting nodes with the same label in the suffix tree, and others as light edges.

Observation

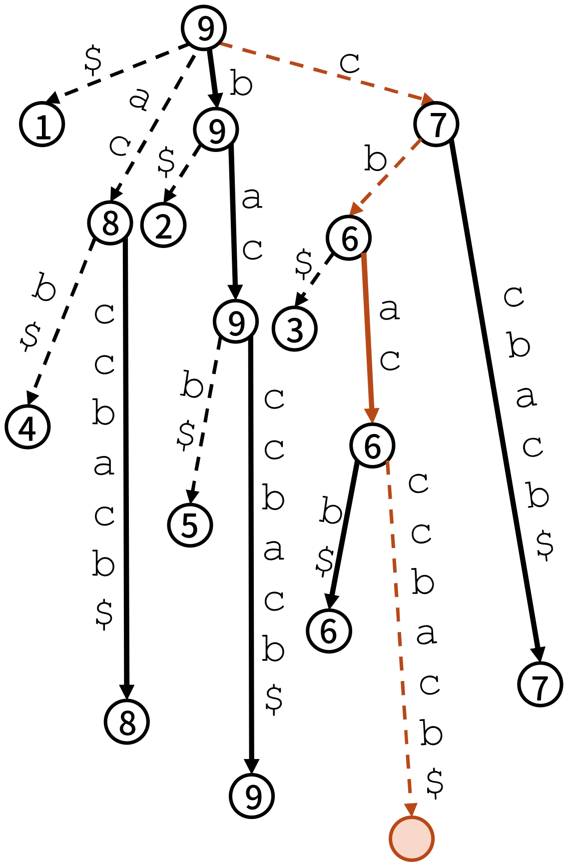
Each new right closed repeat corresponds to the upper endpoint of a light edge.

If we can efficiently enumerate Light Edges on the longest path, we can also enumerate new right closed repeats.

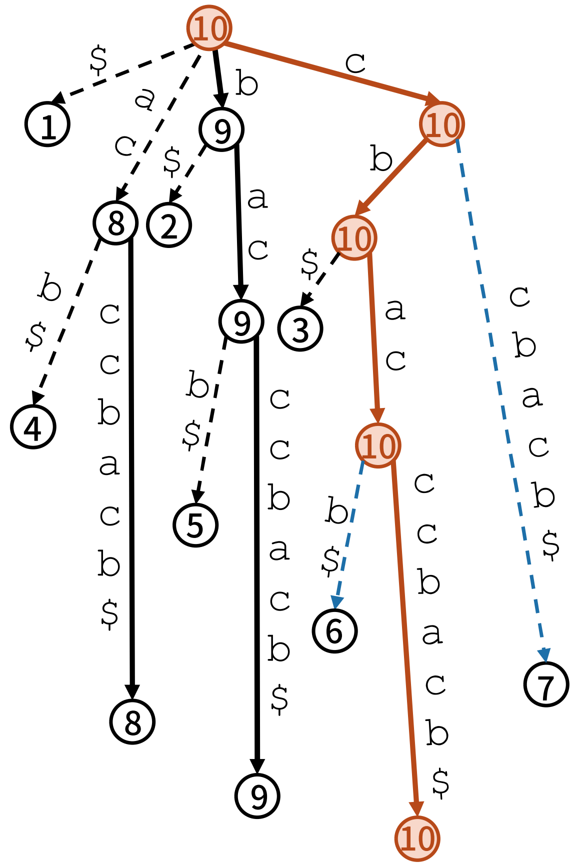


The ST of cbaccbacb$

Updating the Node Labels and Edge Attributes



Before updating



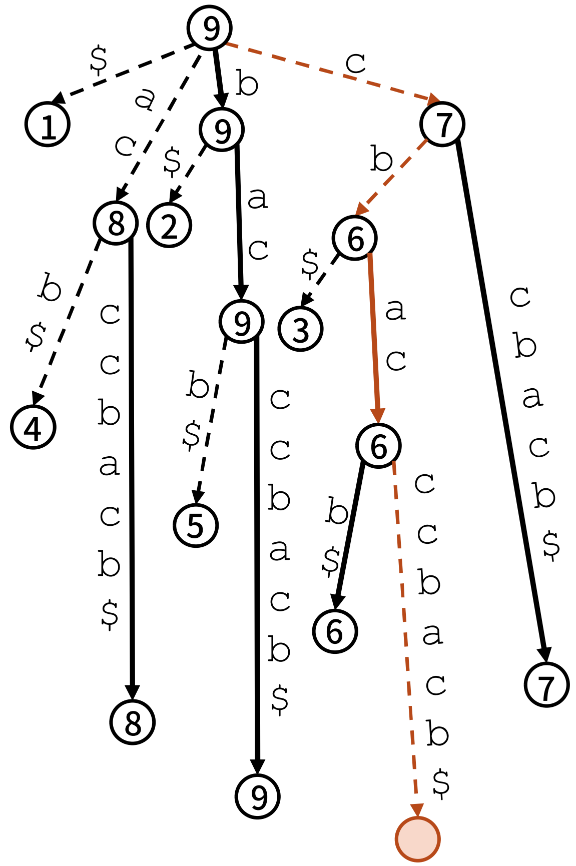
After updating

Summary of Operations on the Suffix Tree

The algorithm performs the following operations:

1. **Adding a leaf,**
2. **Enumerating light edges on the root-to-leaf path, and**
3. **Updating edge attributes of the root-to-leaf path and existing heavy edges.**

> **These operations are <u>essentially the same as the internal processes of the Link-Cut Tree</u> [Sleator & Tarjan, 1983], and can be performed in amortized $O(\log n)$ time!**

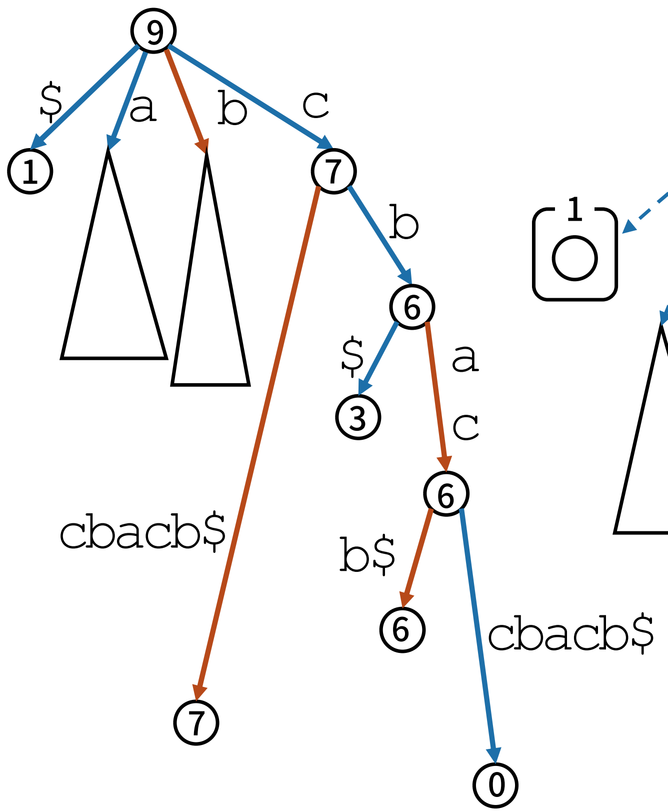


The ST of cbaccbacb$

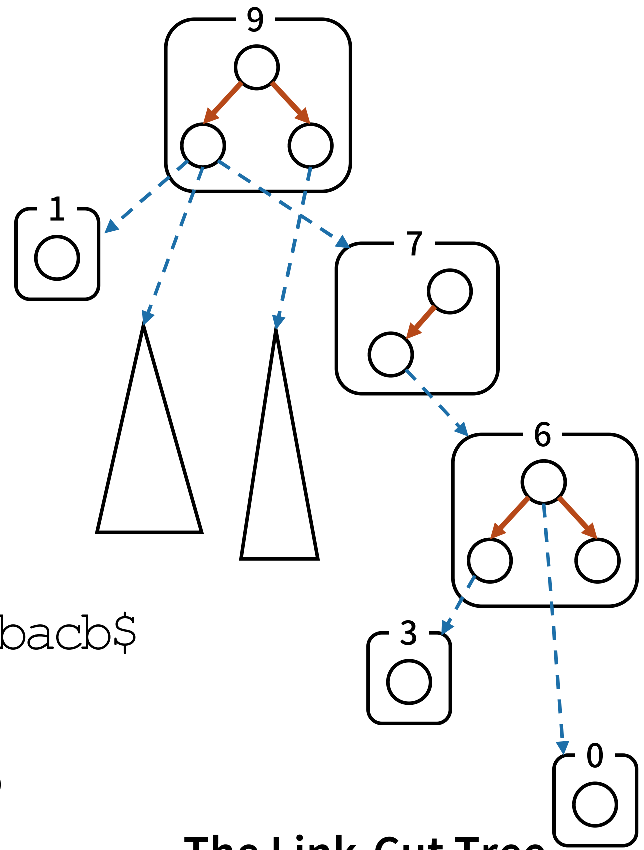
Combining Link-Cut Tree and Suffix Tree

A link-cut tree representation of the suffix tree:

- Each maximal path of heavy edges is represented by a splay tree.
- Each light edge connects between splay trees.



The ST of cbaccbacb$ (partially omitted)



The Link-Cut Tree corresponding to the ST. (partially omitted)

Time Complexity of the Algorithm

Time complexity:

- **Adding a leaf: amortized** $O(\log \sigma) \subseteq O(\log n)$
- **Enumerating light edges: amortized** $O(\log n)$
- **Updating edge attributes: amortized** $O(\log n)$

} **Weiner's algorithm** (adding a leaf; enumerating light edges)

} **Link-Cut Tree** (enumerating light edges; updating edge attributes)

The total time complexity: amortized $O(\log n)$ **per character.**

- **We can also enumerate the closed repeats by checking for left maximality.**
- **MCSs can also be computed because of the one-to-one correspondence between closed repeats and MCSs.**

Note: σ **is the alphabet size.**

Other Applications of Link-Cut Suffix Trees

- **Most Recent Match Index**: amortized $O(\log n)$ time per update, $O(|P| \log \sigma)$ time for each MRM query
  - Existing work: amortized $O(\log n)$ time per update and $O(|P|)$ query time, assuming constant-size alphabet [Larsson, CPM 2014]
- **Online Rightmost LZ**: amortized $O(\log n)$ time per update
  - Existing work: $O\left(n \frac{\log n}{\log \mathrm{lo}}\right)$ using Dynamic RMQ [Sumiyoshi et al., SPIRE 2024]
  - Our method is practically faster.
- **Sliding-window** model (by batch update technique)

Conclusion

- We introduced an online algorithm for computing MCS.
  - Based on the combination of link-cut trees and suffix trees.
  - Can also be used for closed repeats, right-maximal MCS, etc…
  - It achieves the same time complexity as existing offline methods.
- Other Applications: most recent match, rightmost LZ, etc.