CSE 566 Spring 2023

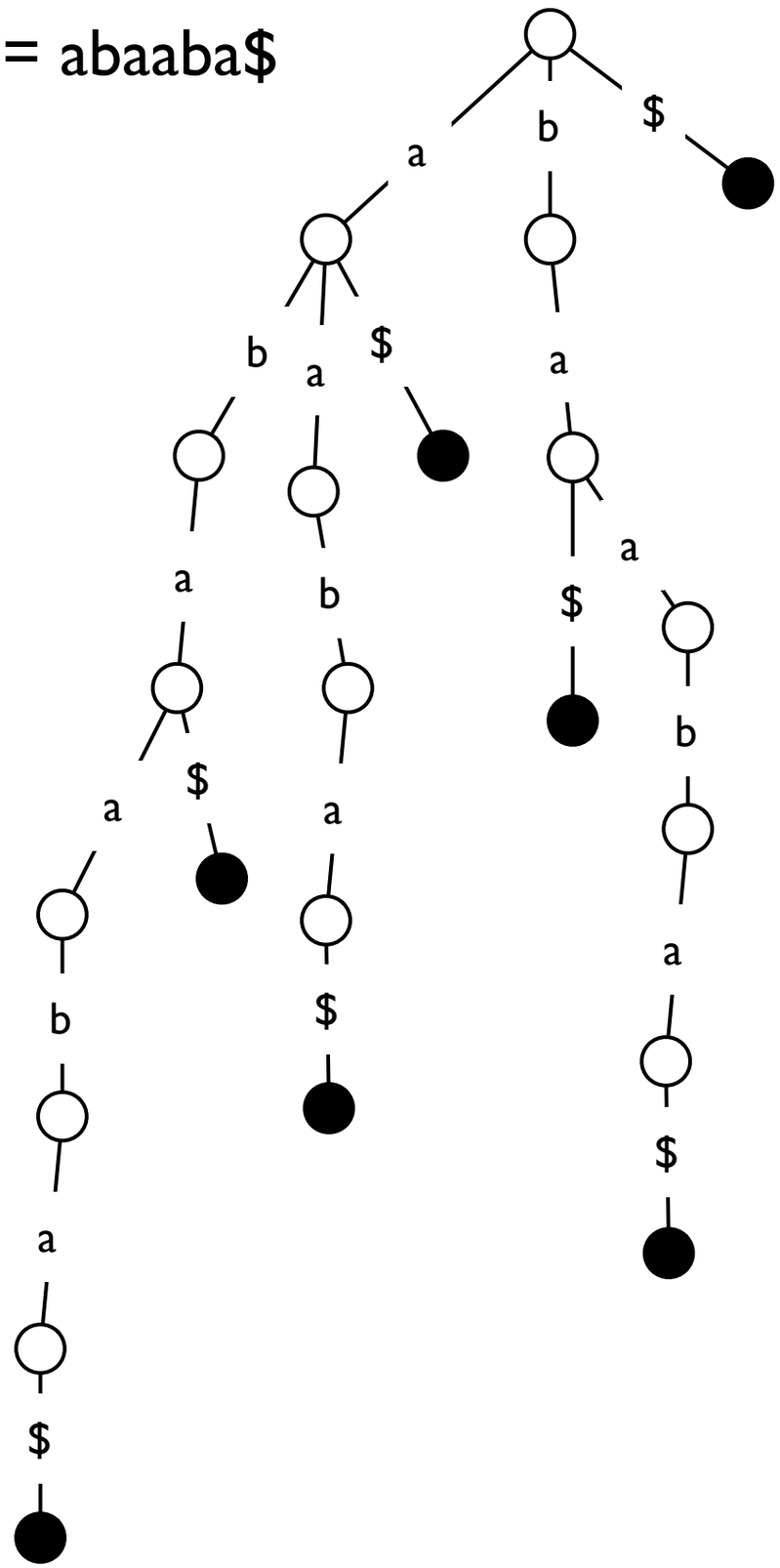
Instructor: Mingfu Shao

Suffix Tree

(Slides copied/edited from these by Dr. Carl Kingsford)

Suffix Trie: Definition

T = abaaba$

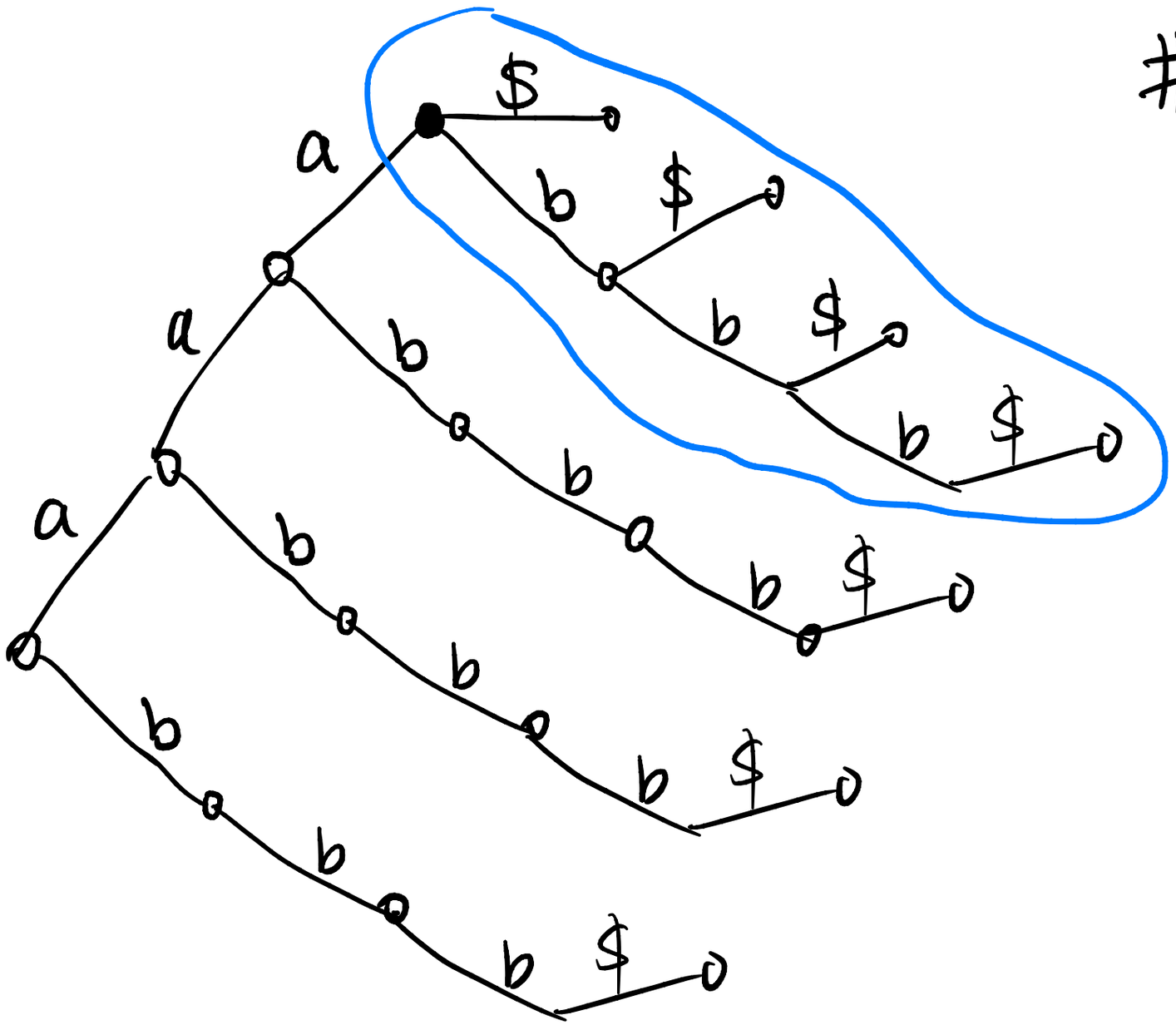


1. A tree structure representing string s.
2. Edges are labeled with letters from the alphabet ∑, say {a,b,$}.
3. Out-edges of the same node are labeled differently.
4. Every path from the root to a leaf represents a suffix of s.
5. Every suffix of s is represented by some path from the root to a leaf.

How many leaves will there be?
Why adding $ to the end?

How many nodes can a suffix trie have?

T = aaabbb$



$T = a^n b^n \$$

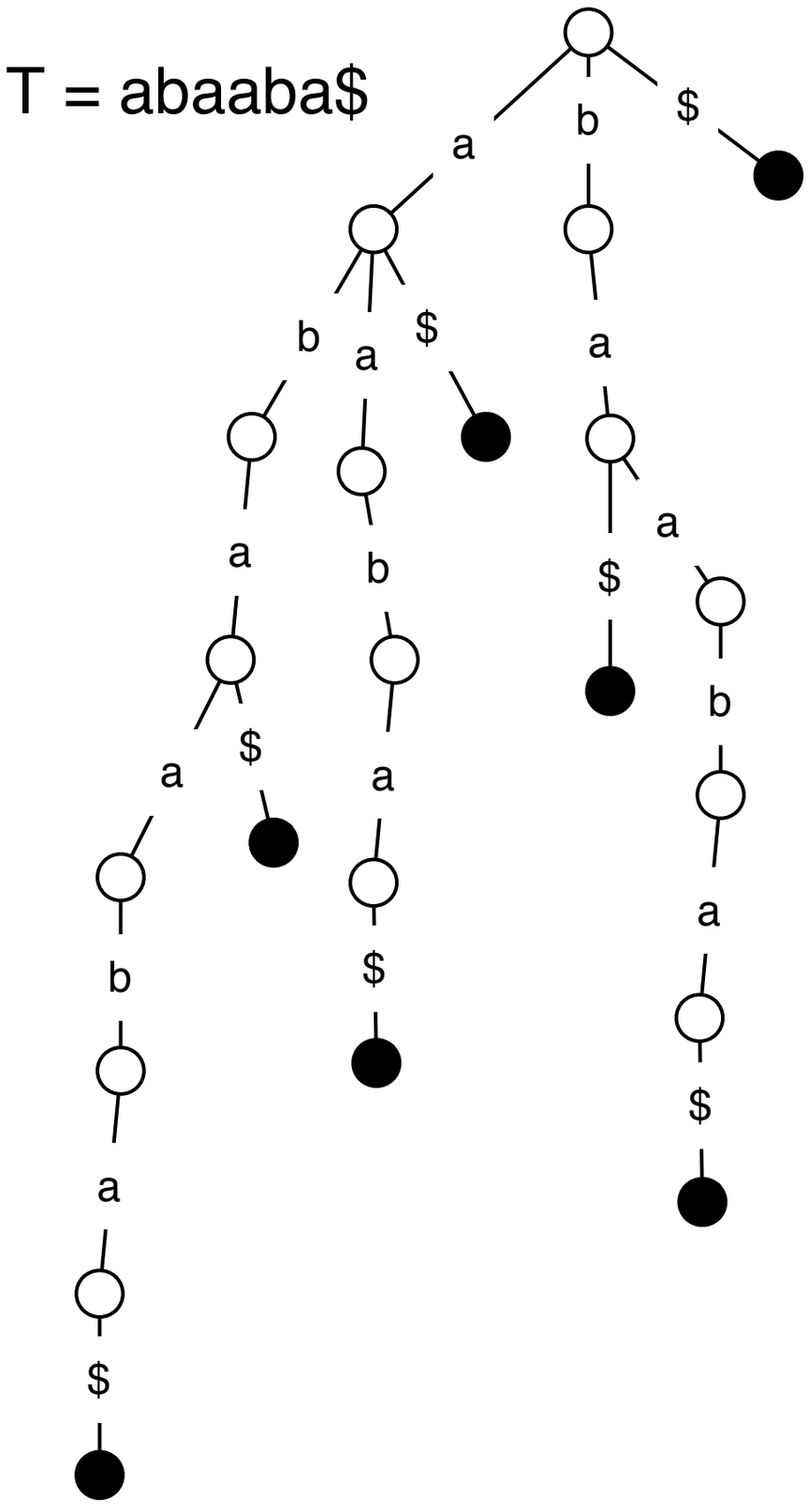
$a \cdots a \; b \cdots b \; \$$

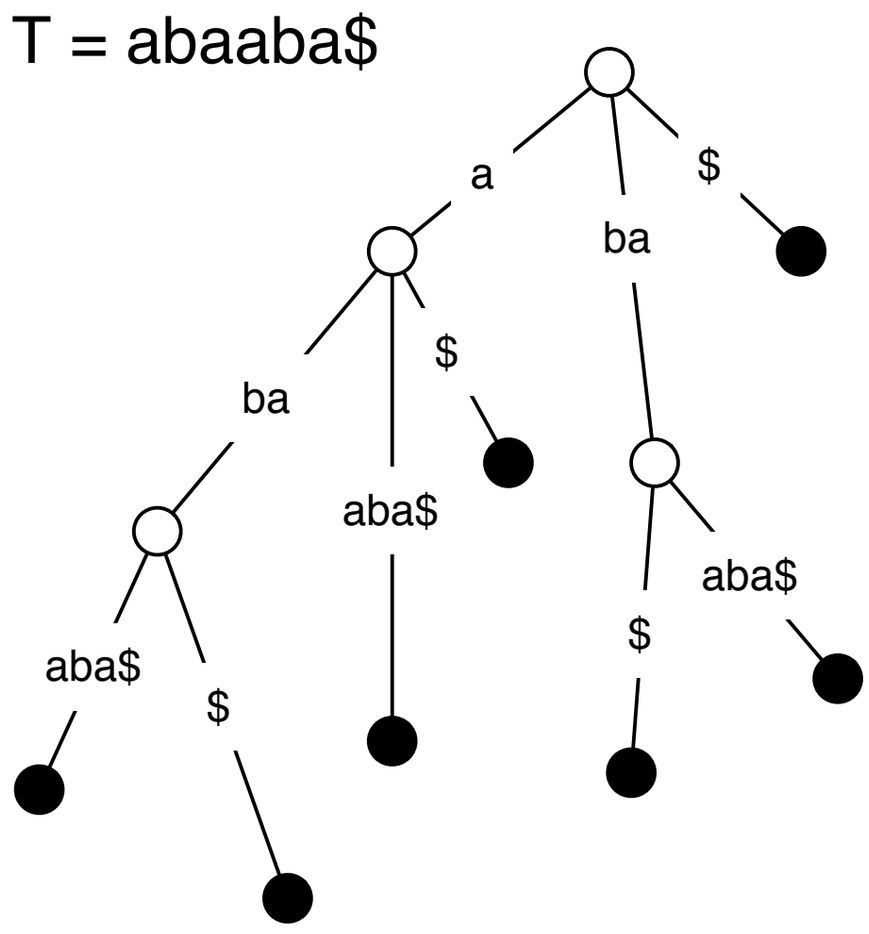
$\#\text{nodes} = \boxed{2(n+1)}$

$+ \; n(n+1)$

$= O(n^2)$

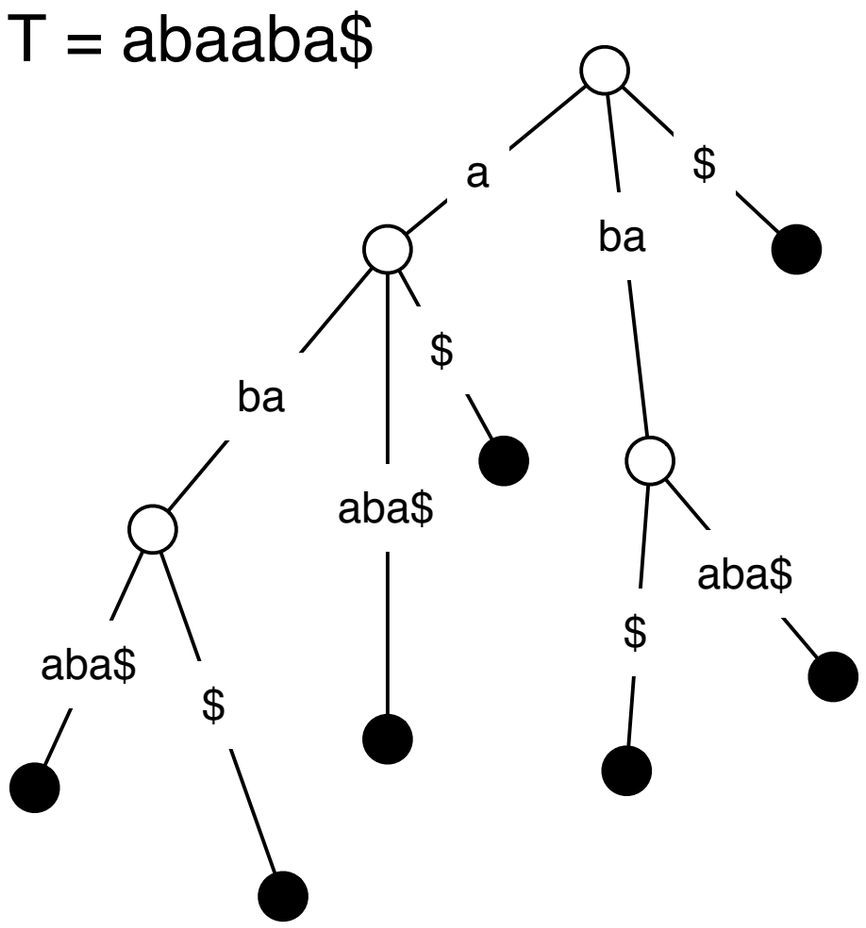
Suffix Tree: A Compact Representation

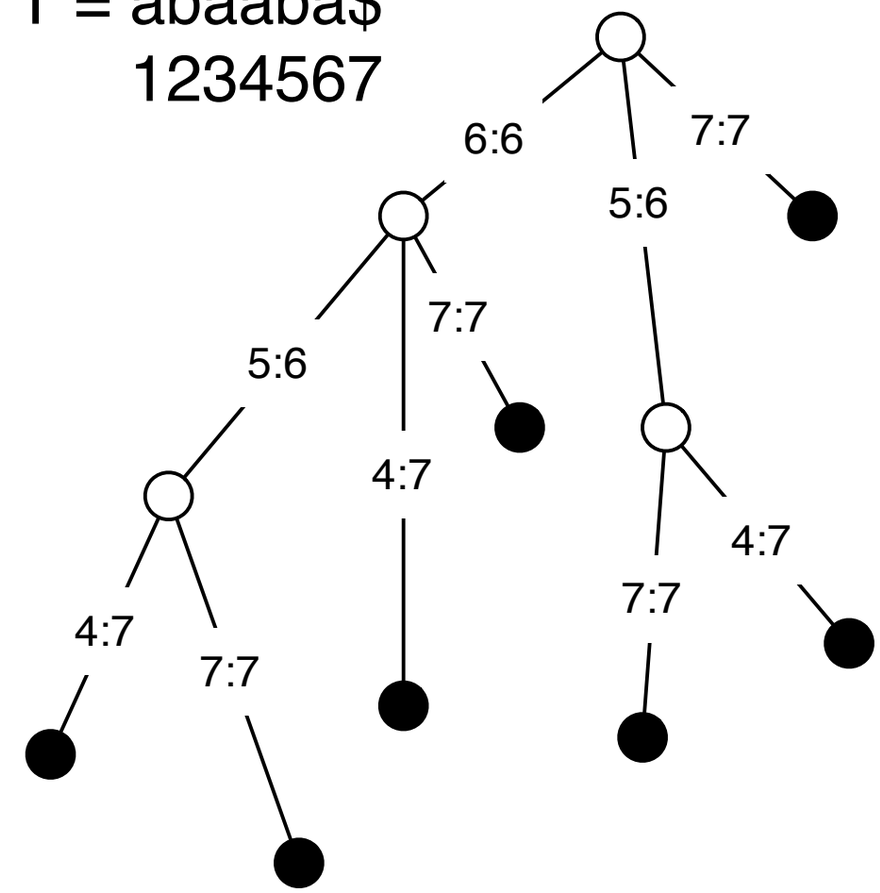




- Compress paths where there are no choices.
- Every internal node has at least two children.
- O(n) nodes/edges (proof)

More Compact Representation



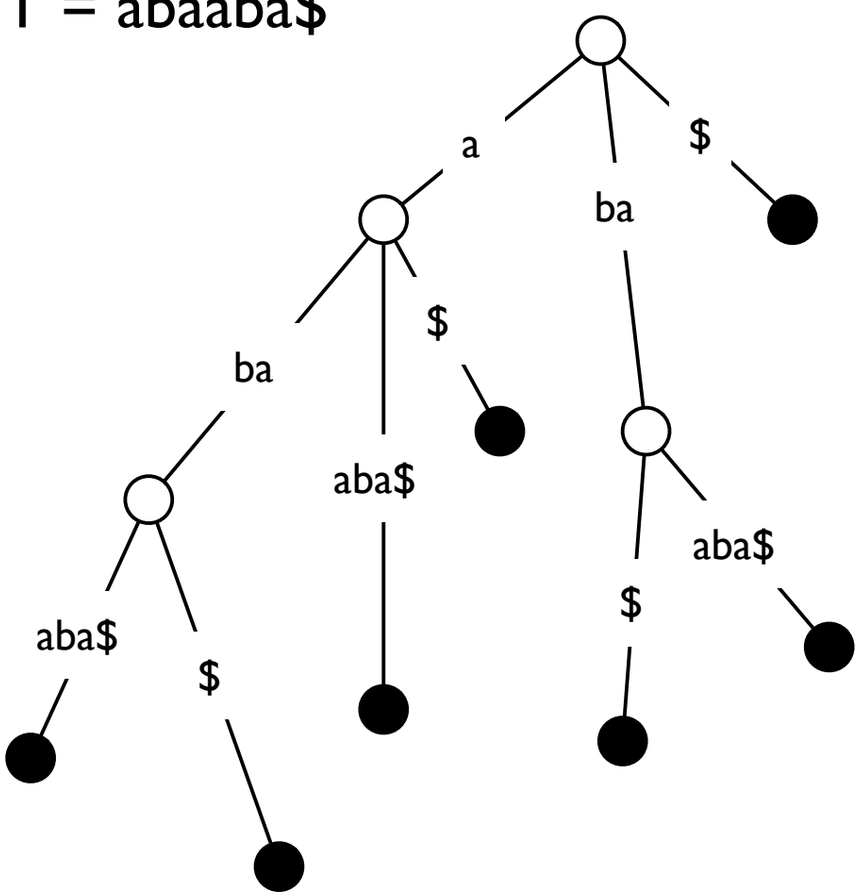


- Represent sequence along the path using a range [i,j] that refers to the input string s.

Search a Suffix Tree for a Query

q = baa

T = abaaba$

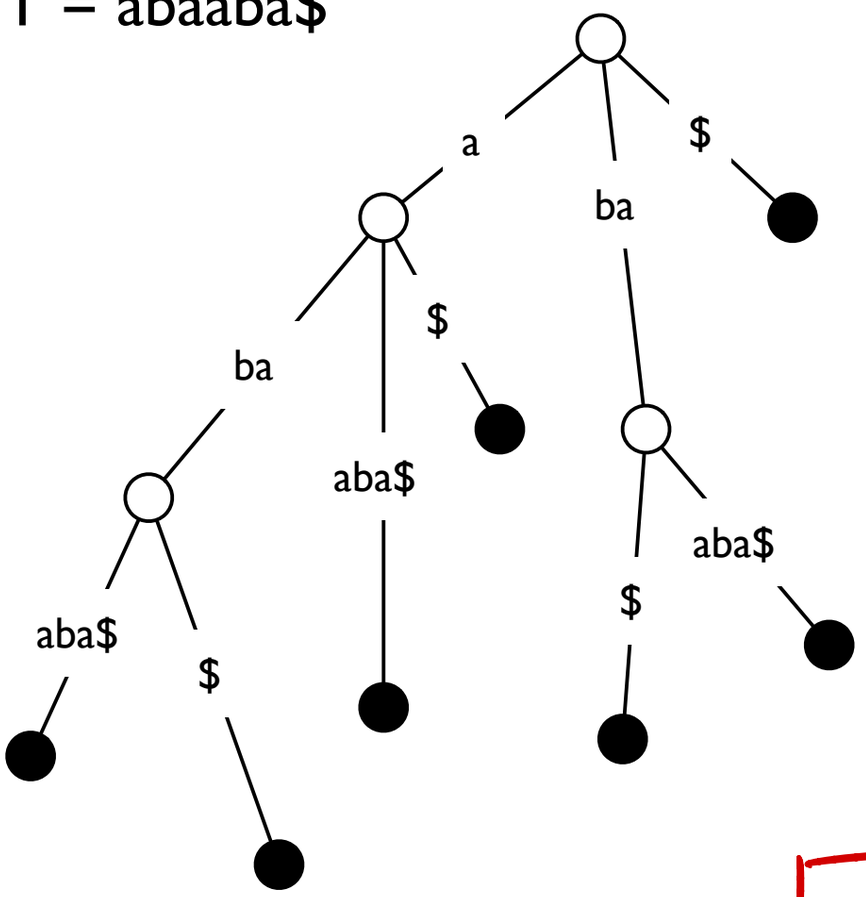


Follow the path guided by query q:

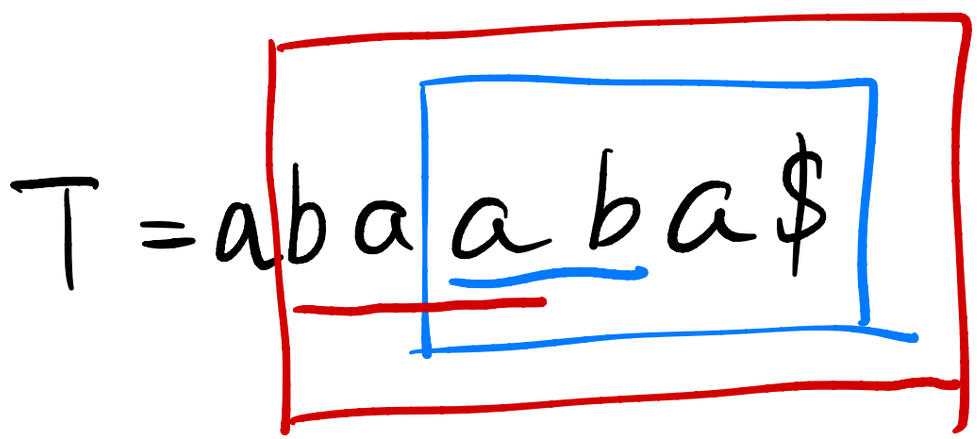
- q is exhausted:
  - ending at a leaf node ba$
  - ending at an internal node aba
  - ending in the middle of an edge; baa
- dead-end:
  - in the middle of an edge aaa
  - at an internal node bab

Substring = Prefix of Suffix

T = abaaba$

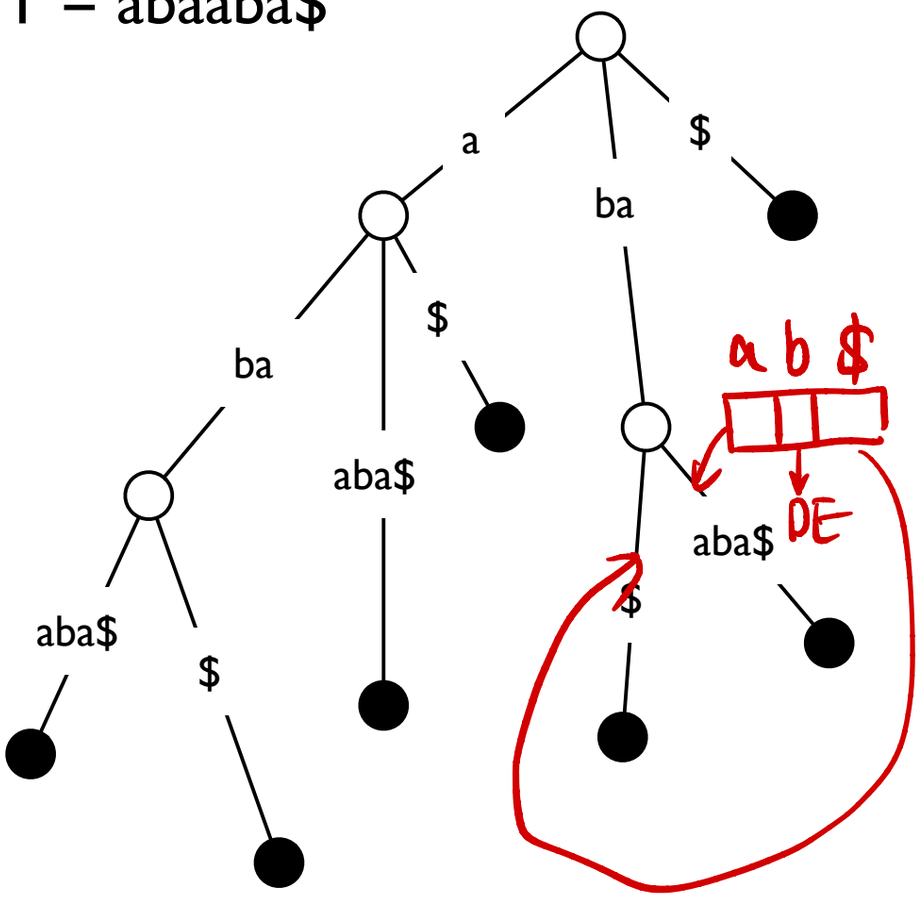


- Suffix tree contains all suffixes.
- All substrings can be found starting from the root.
- Any search that is not dead-end gives a substring.



Application of Suffix Tree

T = abaaba$



Question: check if q is a substring of T.

q = aabb

Algorithm:

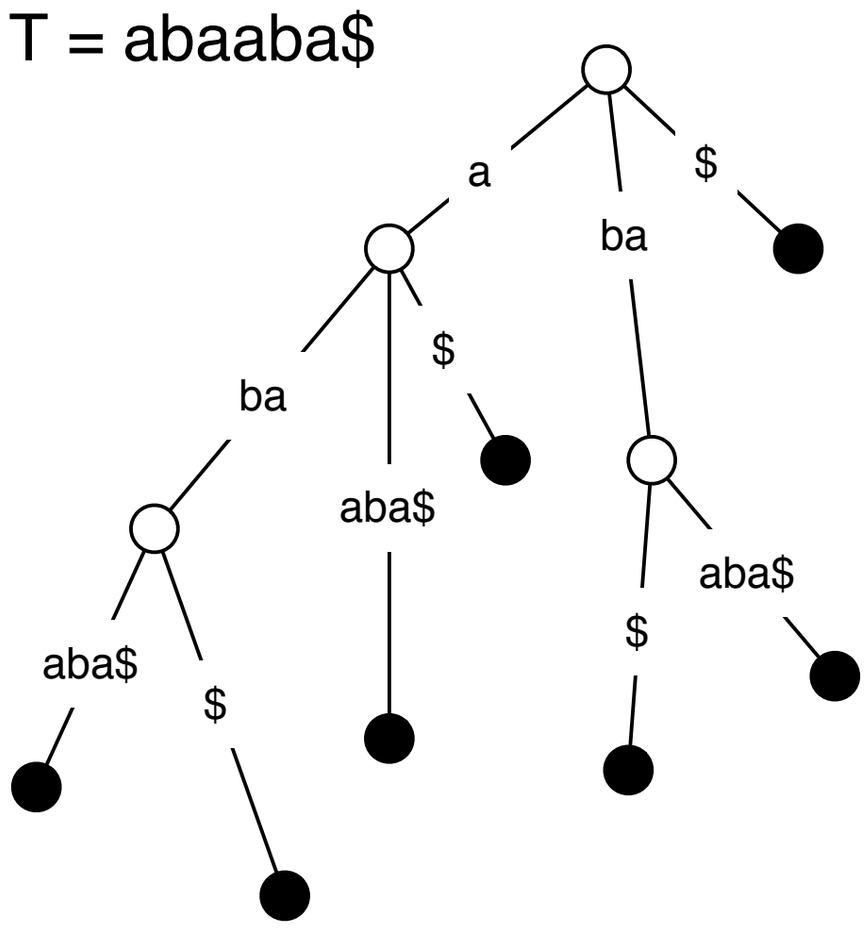
- Follow the path given by q.
- If q is exhausted: yes;
- Otherwise: no.

Running time: O(|q|),
regardless of text size.!

#(out-edges) ≤ |Σ|
for a single-node.

Applications of Suffix Trees

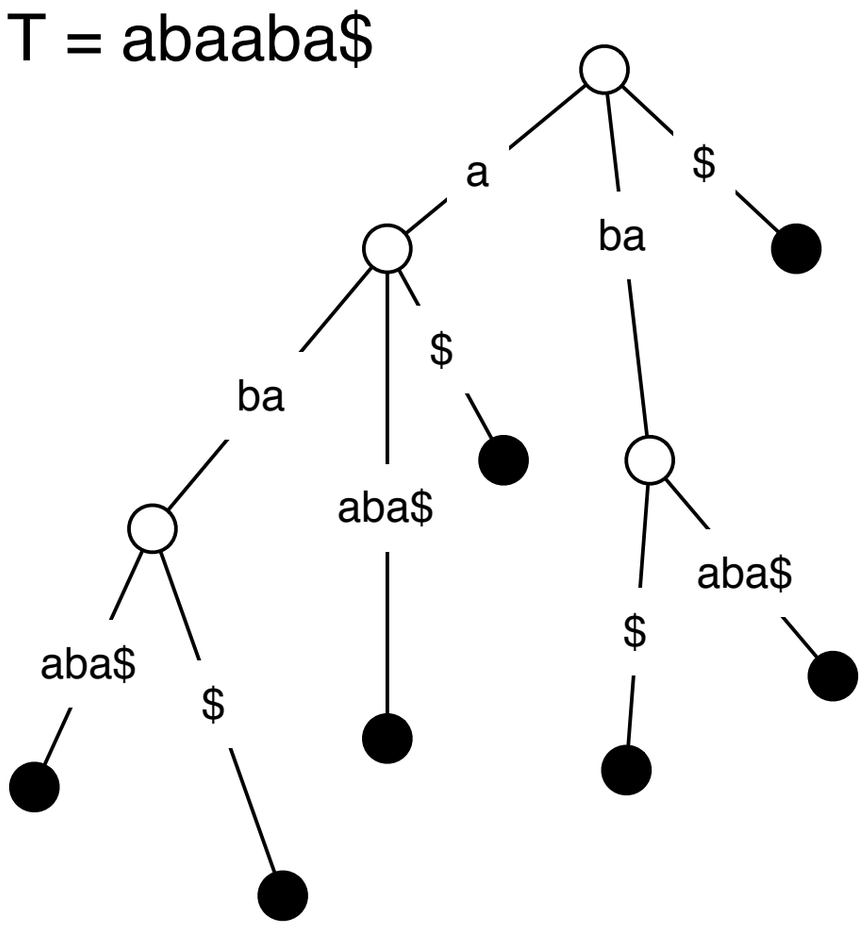
Check whether q is a **suffix** of T:



Applications of Suffix Trees

Check whether q is a **suffix** of T:

Follow the path for q starting from the root. If you end at a leaf at the end of q, then q is a suffix of T



Applications of Suffix Trees

Count # of occurrences of q in T:

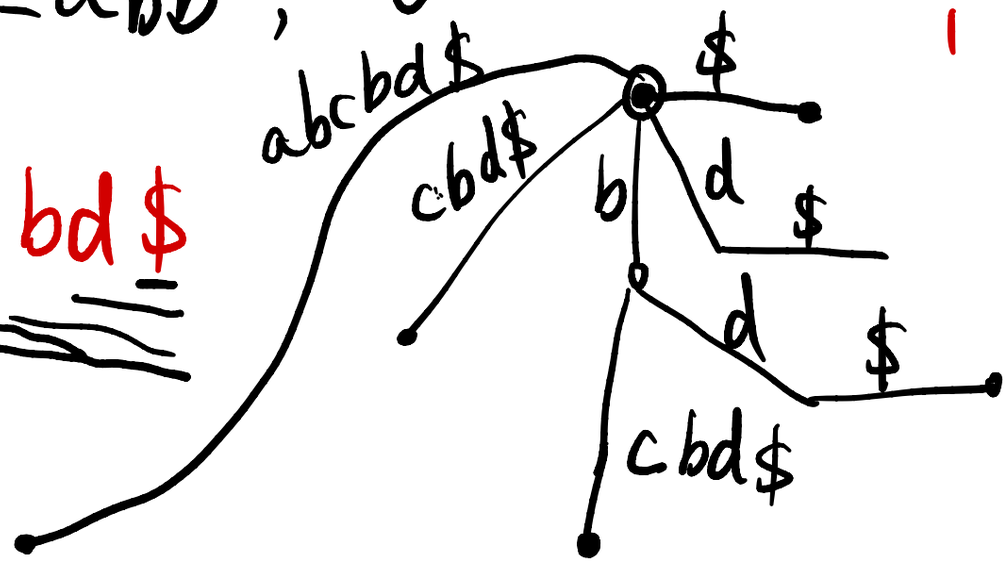
q = ba , 2

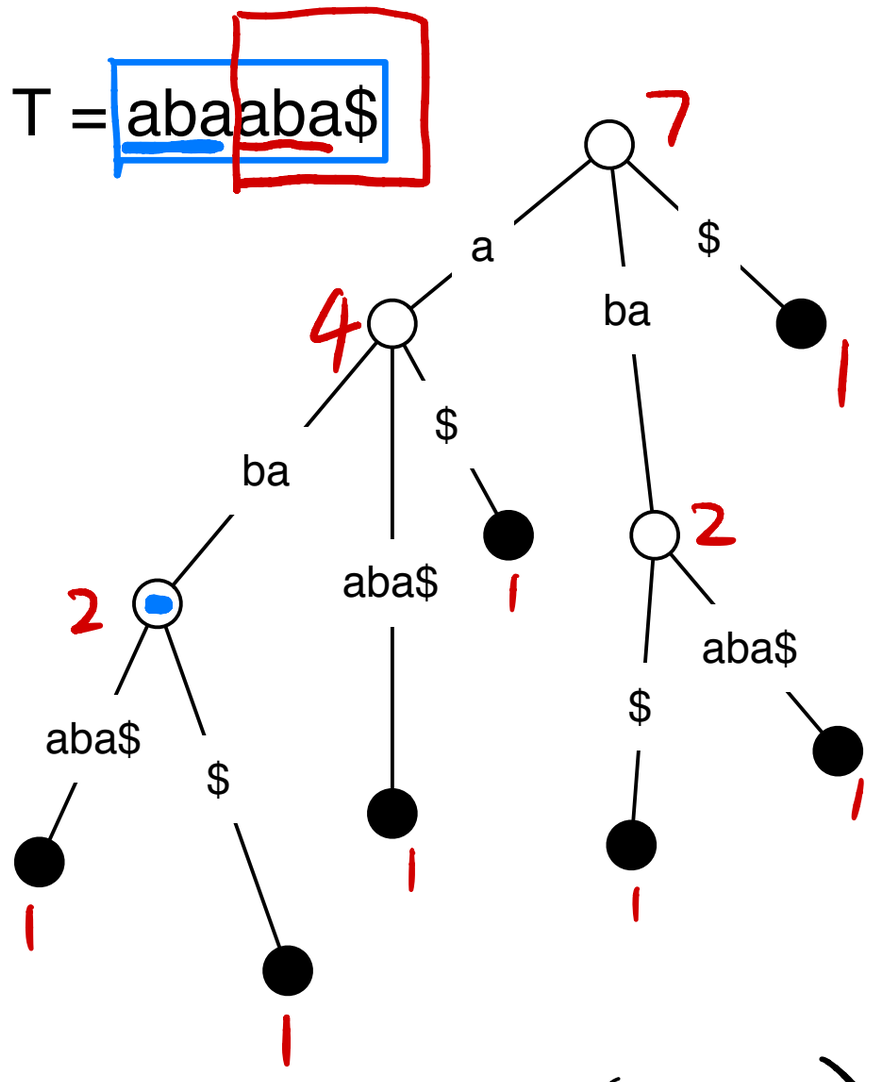
q = aba , 2

q = a , 4

q = abb , 0

T = abcbd$





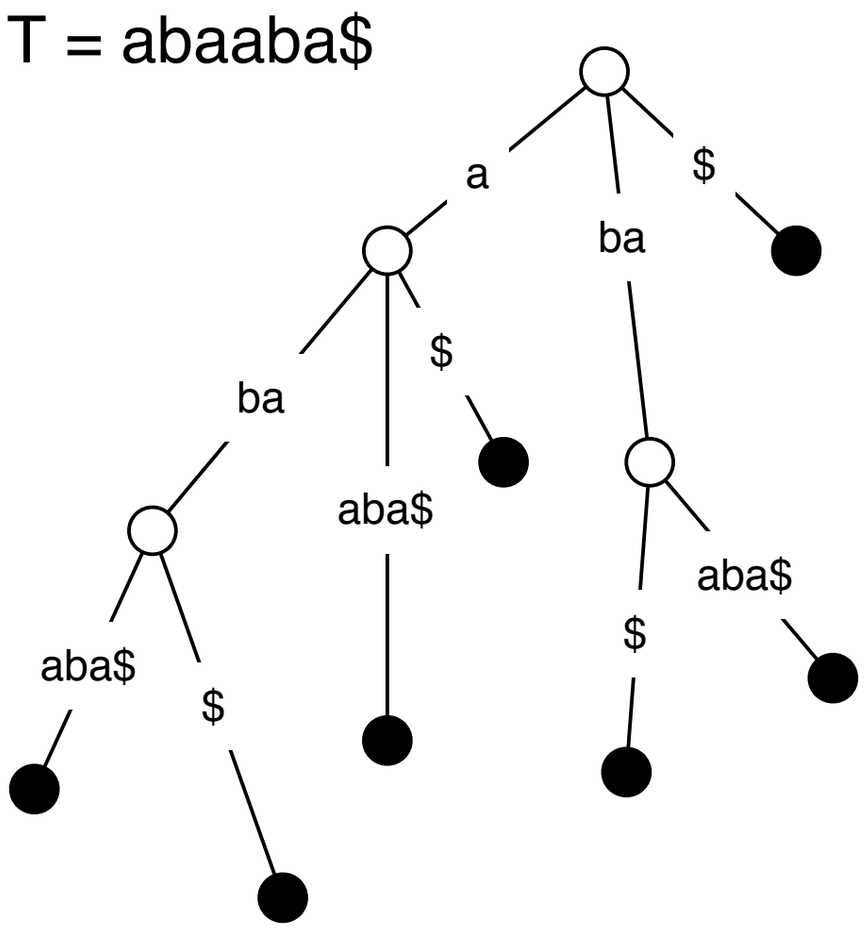
O(|T|)

Applications of Suffix Trees

Count # of occurrences of q in T:

Follow the path for q starting from the root.
The number of leaves under the node/edge you end up in is the number of occurrences of q.

$O(|q|)$



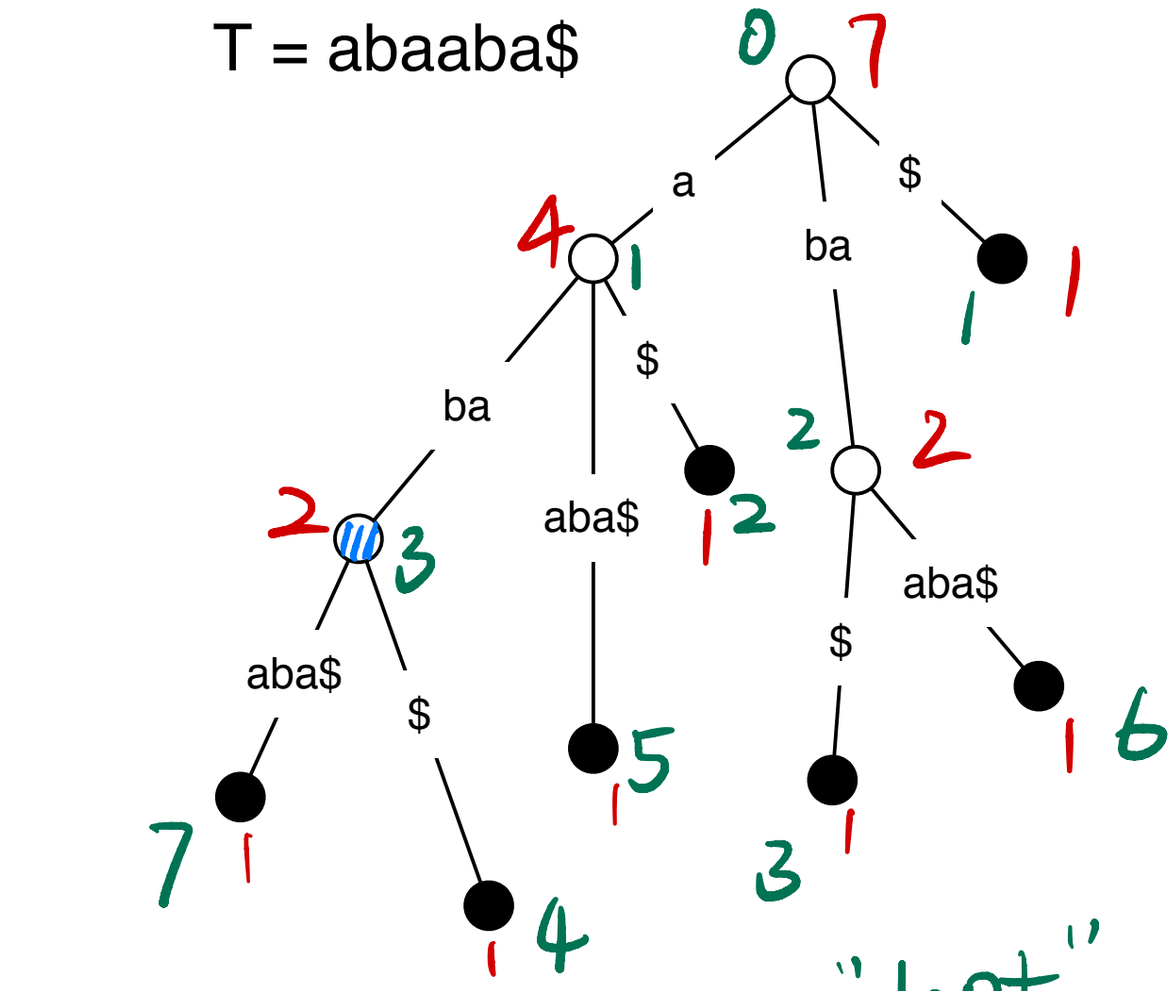
Applications of Suffix Trees

Find the longest repeat in T:

aba

O(|T|)

T = abaaba$



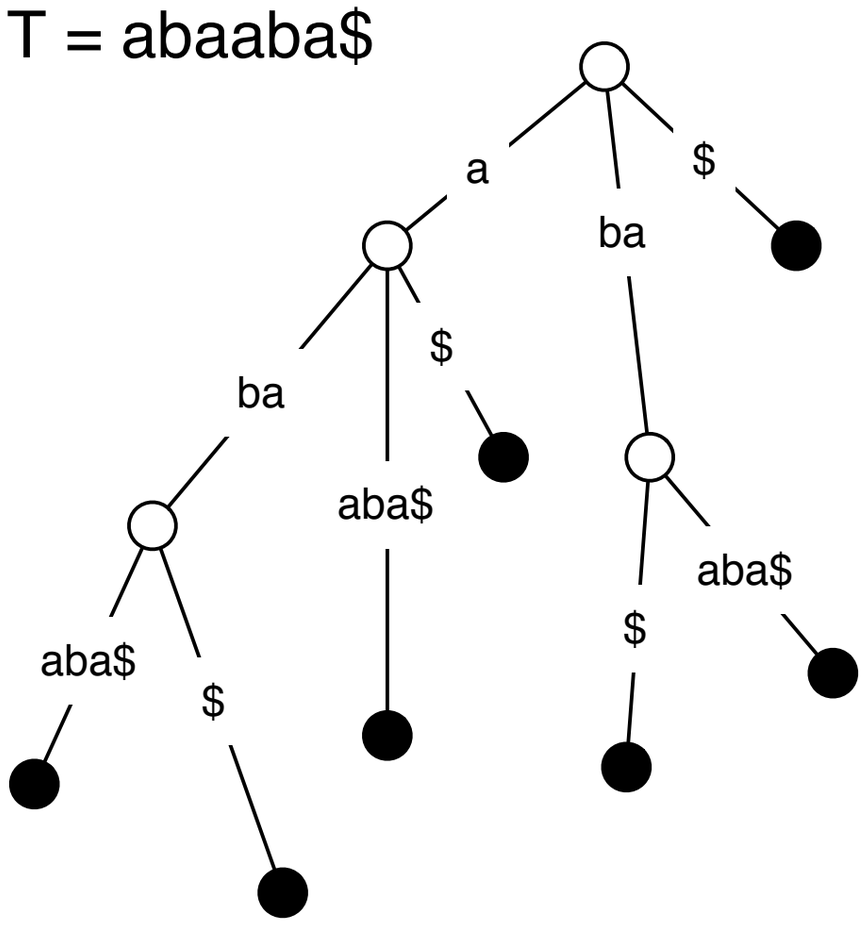
"occurrence" (red numbers)

"dept" (green numbers)

Applications of Suffix Trees

Find the longest repeat in T:

Find the **deepest** node that has at least 2 leaves under it.



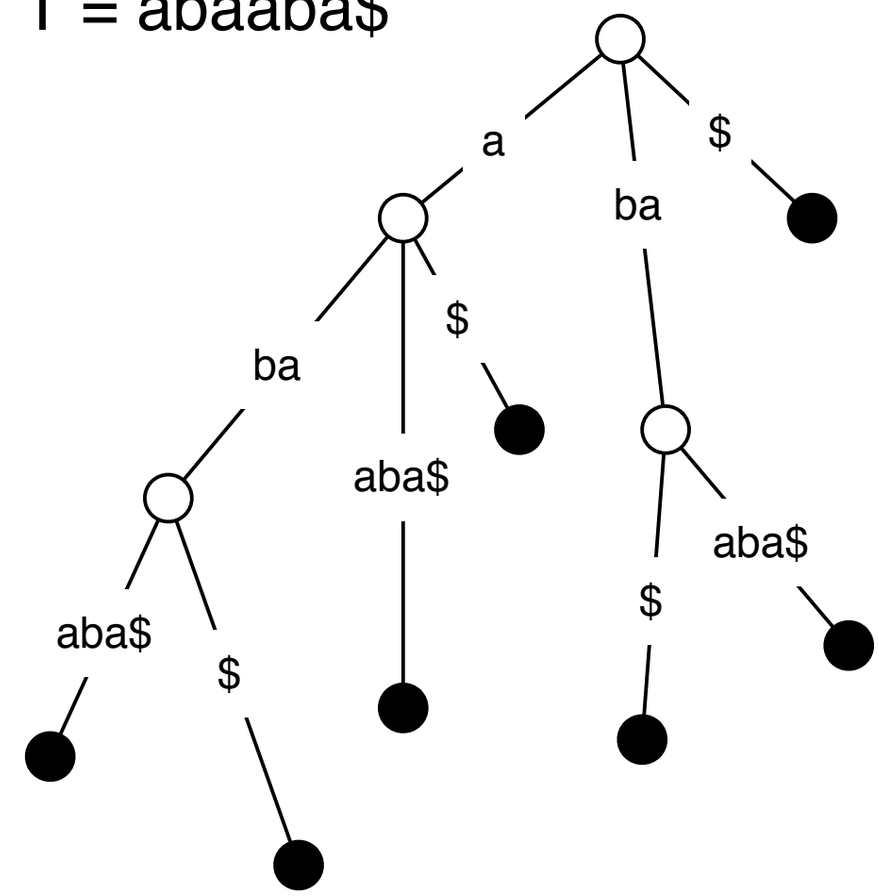
Applications of Suffix Trees

Find the lexicographically (alphabetically) first suffix:

aba

aabb

T = abaaba$

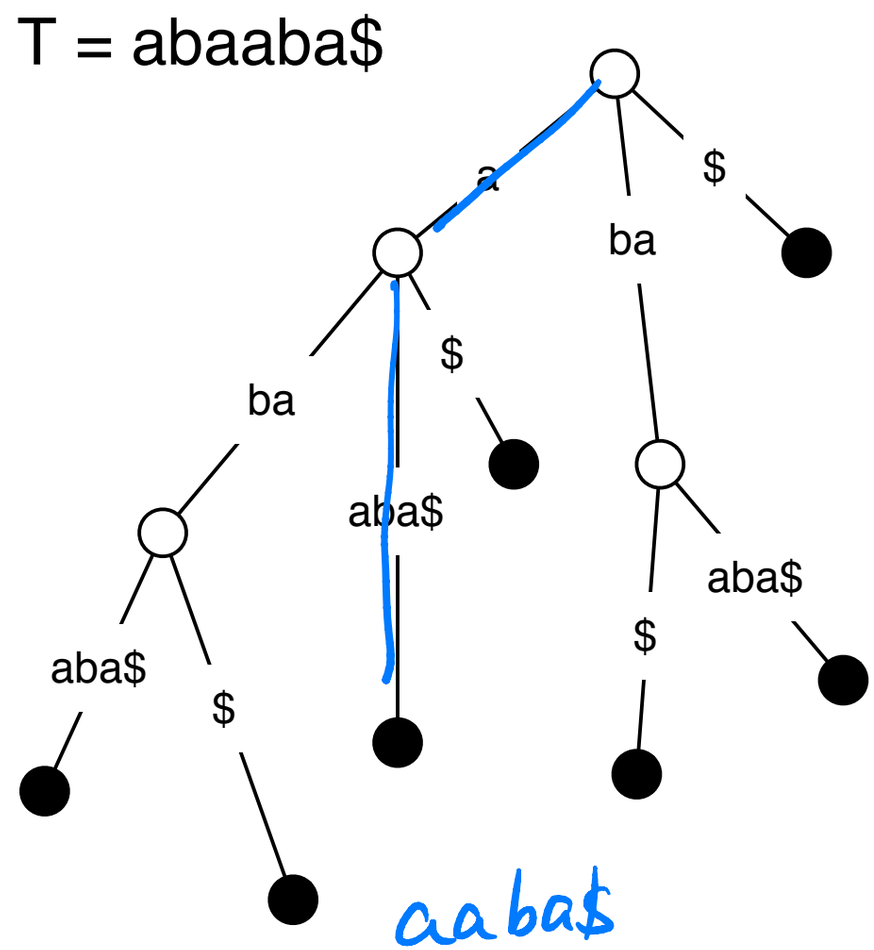


Applications of Suffix Trees

Find the lexicographically (alphabetically) first suffix:

$a < b < \$$

Start at the root, and follow the edge labeled with the lexicographically (alphabetically) smallest letter.

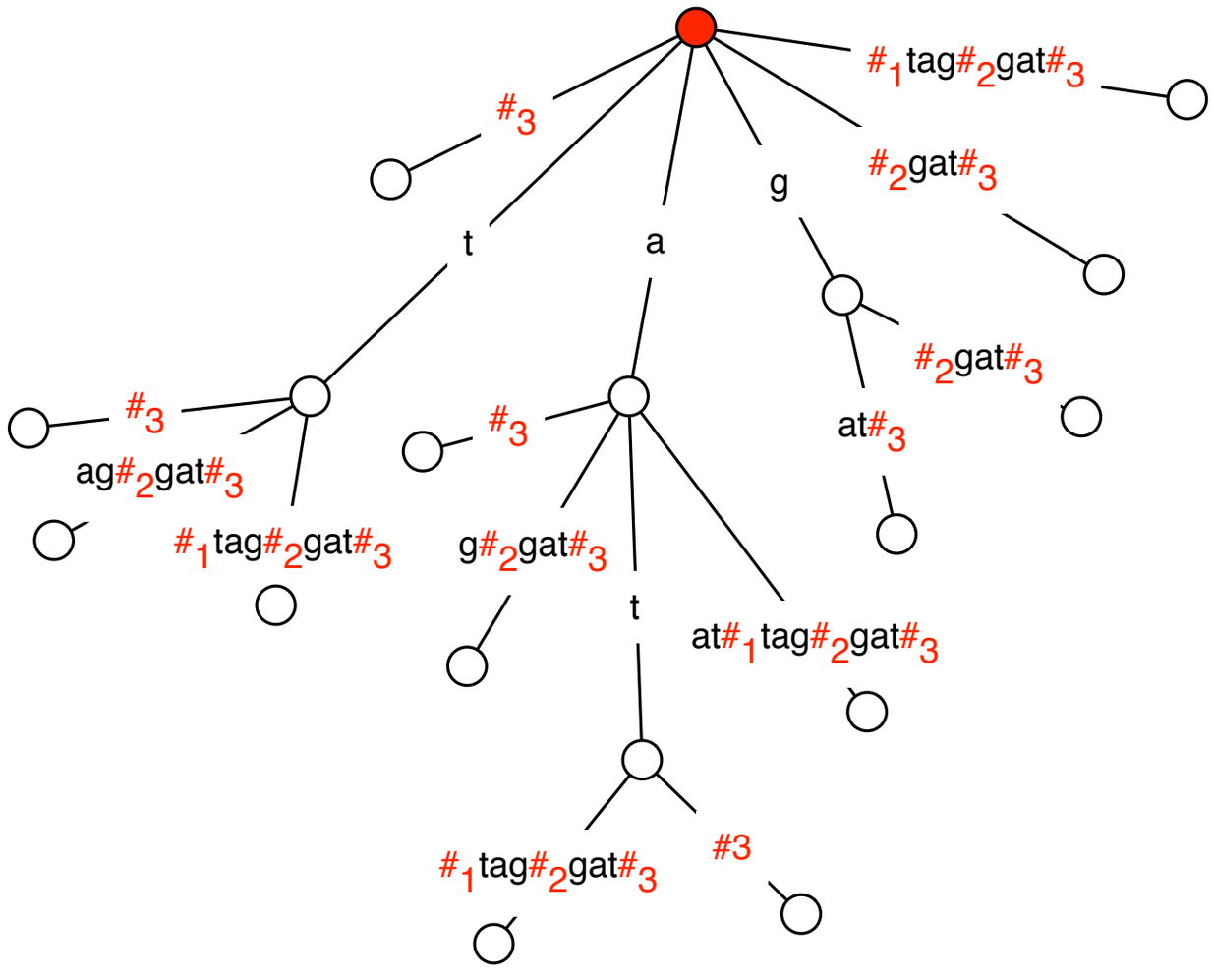


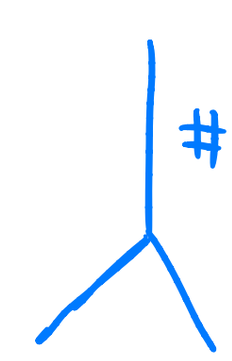
Generalized Suffix Trees

Goal. Represent a set of strings $P = \{s_1, s_2, s_3, ..., s_m\}$.

Example. ~~aat~~ aat, tag, gat

(1) build suffix tree for string $aat\#_1tag\#_2gat\#_3$





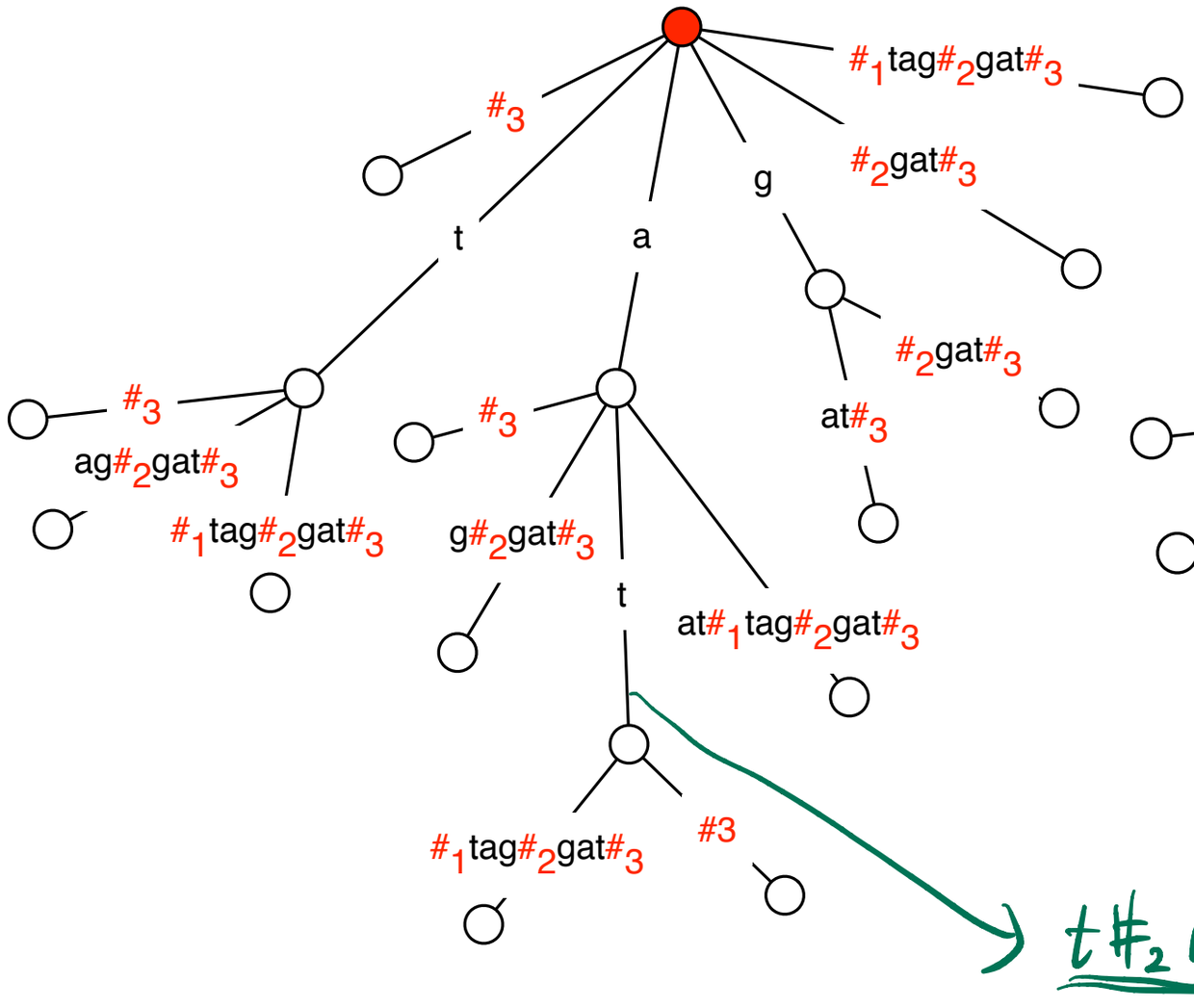
(not possible)

Generalized Suffix Trees

Goal. Represent a set of strings P = {s_1, s_2, s_3, ..., s_m}.

Example. att, tag, gat

(1) build suffix tree for string aat#$_1$tag#$_2$gat#$_3$



(2) For every leaf node, remove any text after the first # symbol.