CSE 566 Spring 2023

Instructor: Mingfu Shao

Generalized Suffix Tree

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

Generalized Suffix Trees

$M = \sum_{k=1}^{m} (|S_k| + 1)$

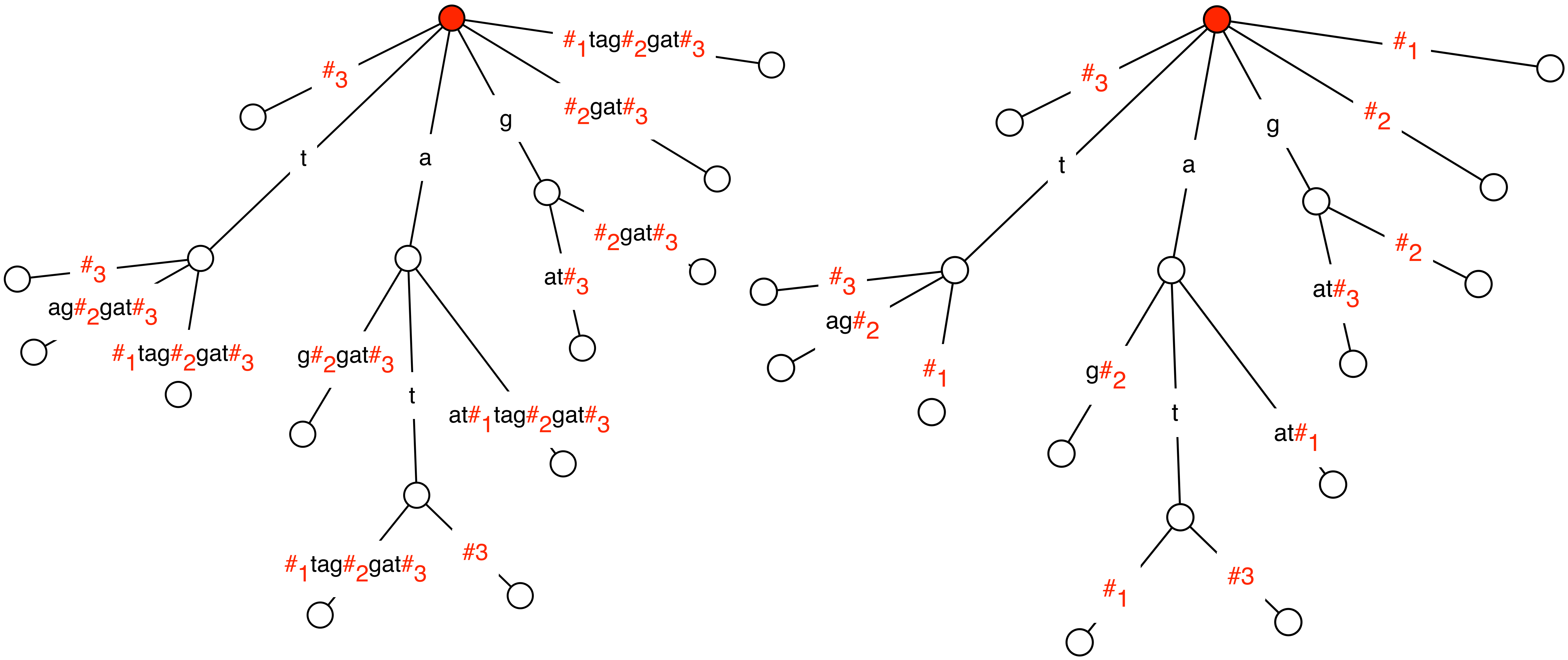
Goal. Represent a set of strings P = {s1, s2, s3, ..., sm}.

Example. aat, tag, gat

$O(M)$

(1) build suffix tree for string $\underline{aat\#1tag\#2gat\#3}$

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

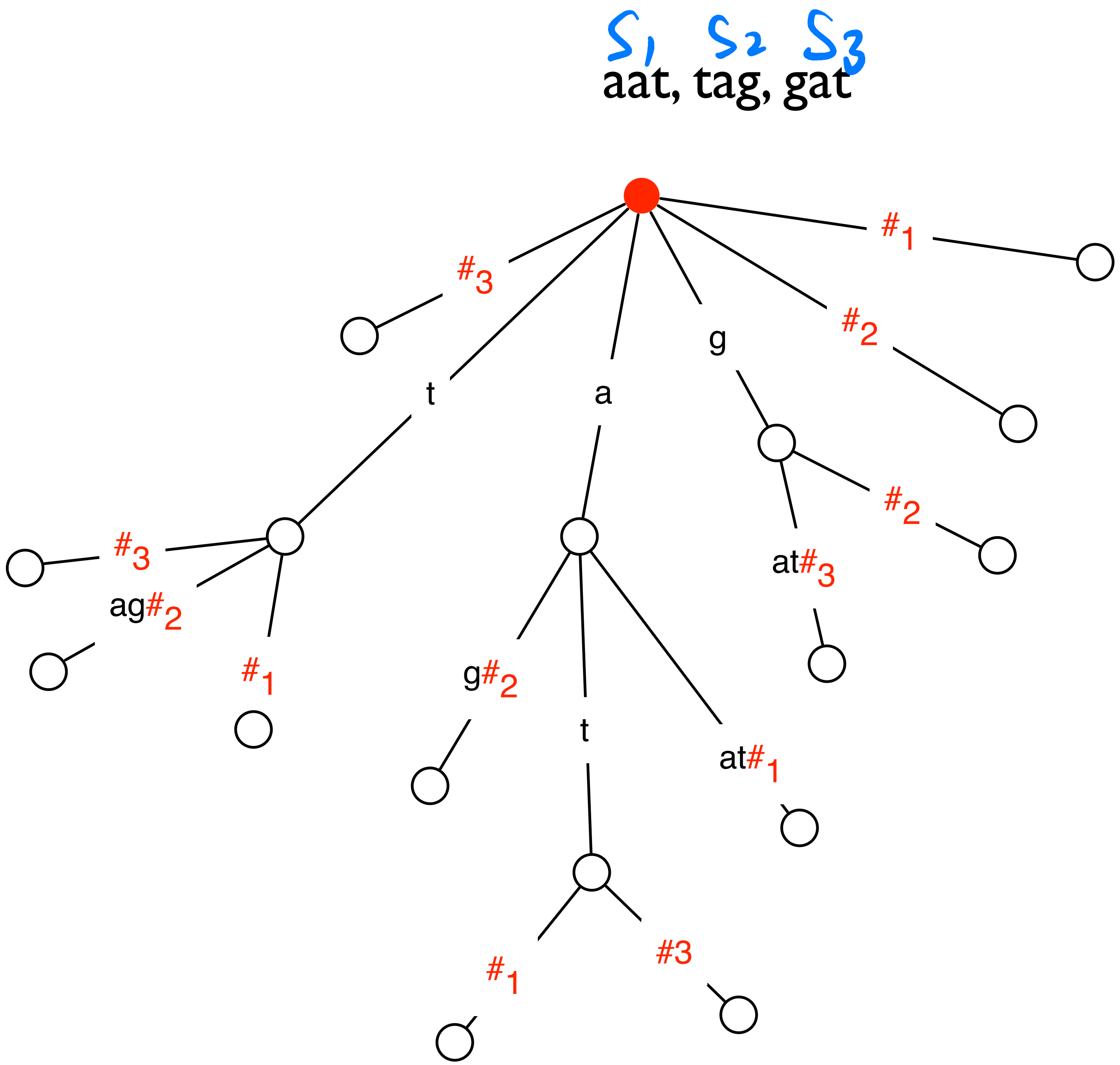


Applications of Generalized Suffix Trees

Determine the strings in a database {S1, S2, S3, ..., Sm} that contain query string q:

$\underline{q = a}$ ⇒ S_1, S_2, S_3

$\underline{q = at}$ ⇒ S_1, S_3

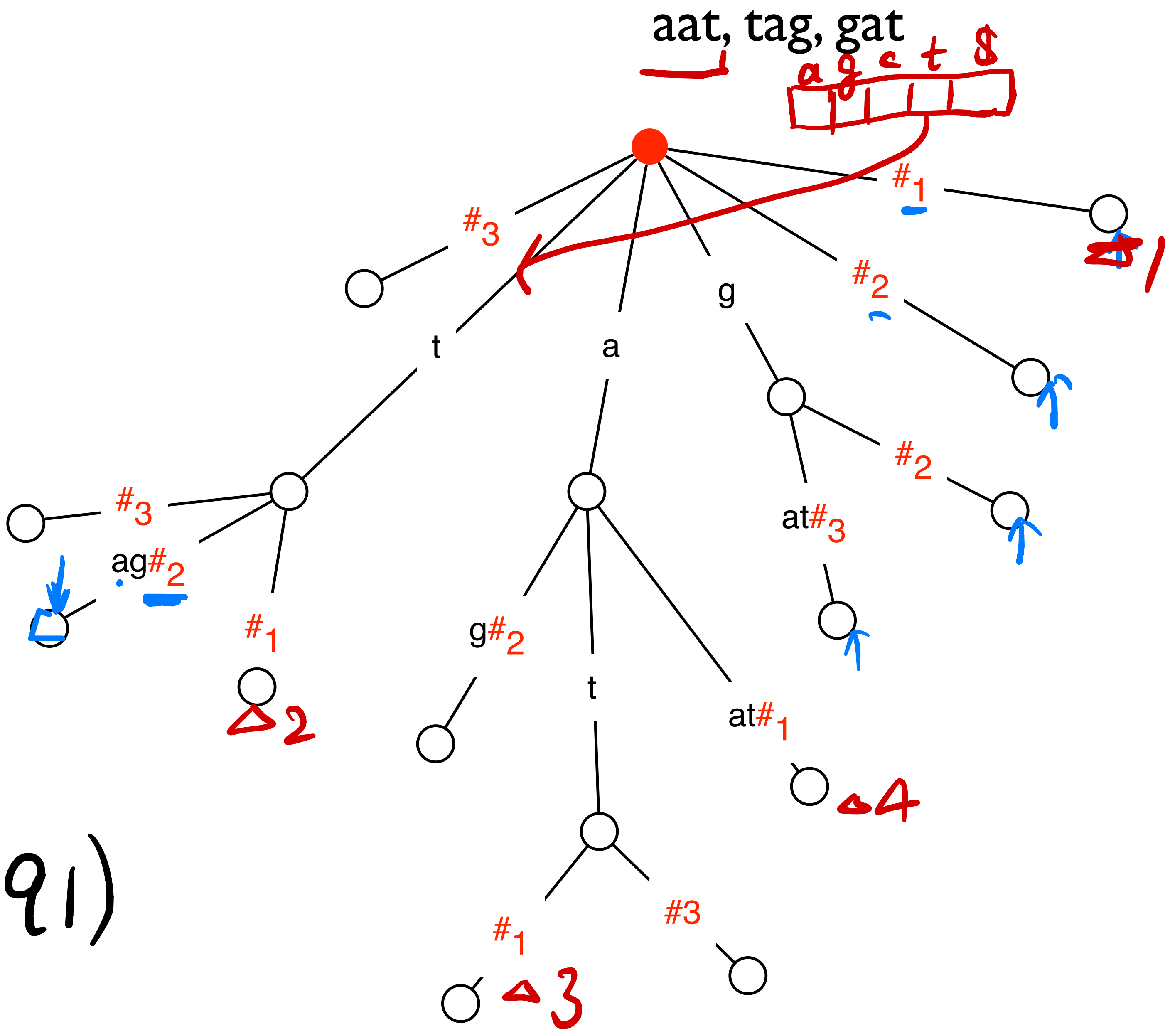


Applications of Generalized Suffix Trees

Determine the strings in a database {S1, S2, S3, ..., Sm} that contain query string q:

$O(M)$

- Build generalized suffix tree for {S1, S2, S3, ..., Sm}
- Follow the path for q in the suffix tree. $O(|q|)$
- Suppose you end at node u: traverse the tree below u, and output i if you find a string containing #i. $O(M)$



$q = ta$

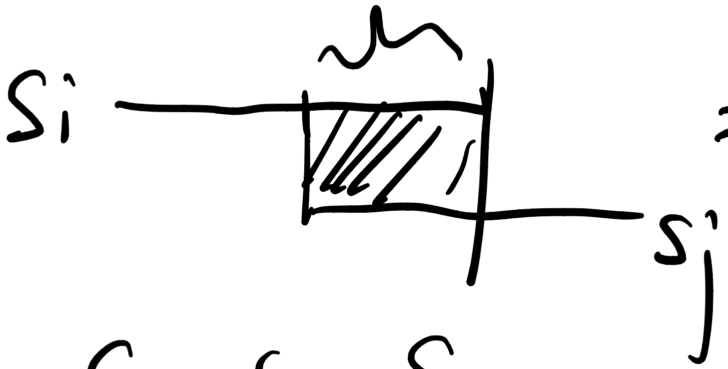
$\Rightarrow O(M+|q|)$

$\#\text{nodes in GST} = O(M)$

Applications of Generalized Suffix Trees

Given {S1, S2, S3, ..., Sm}, find the longest suffix-prefix exact matches for every pair.

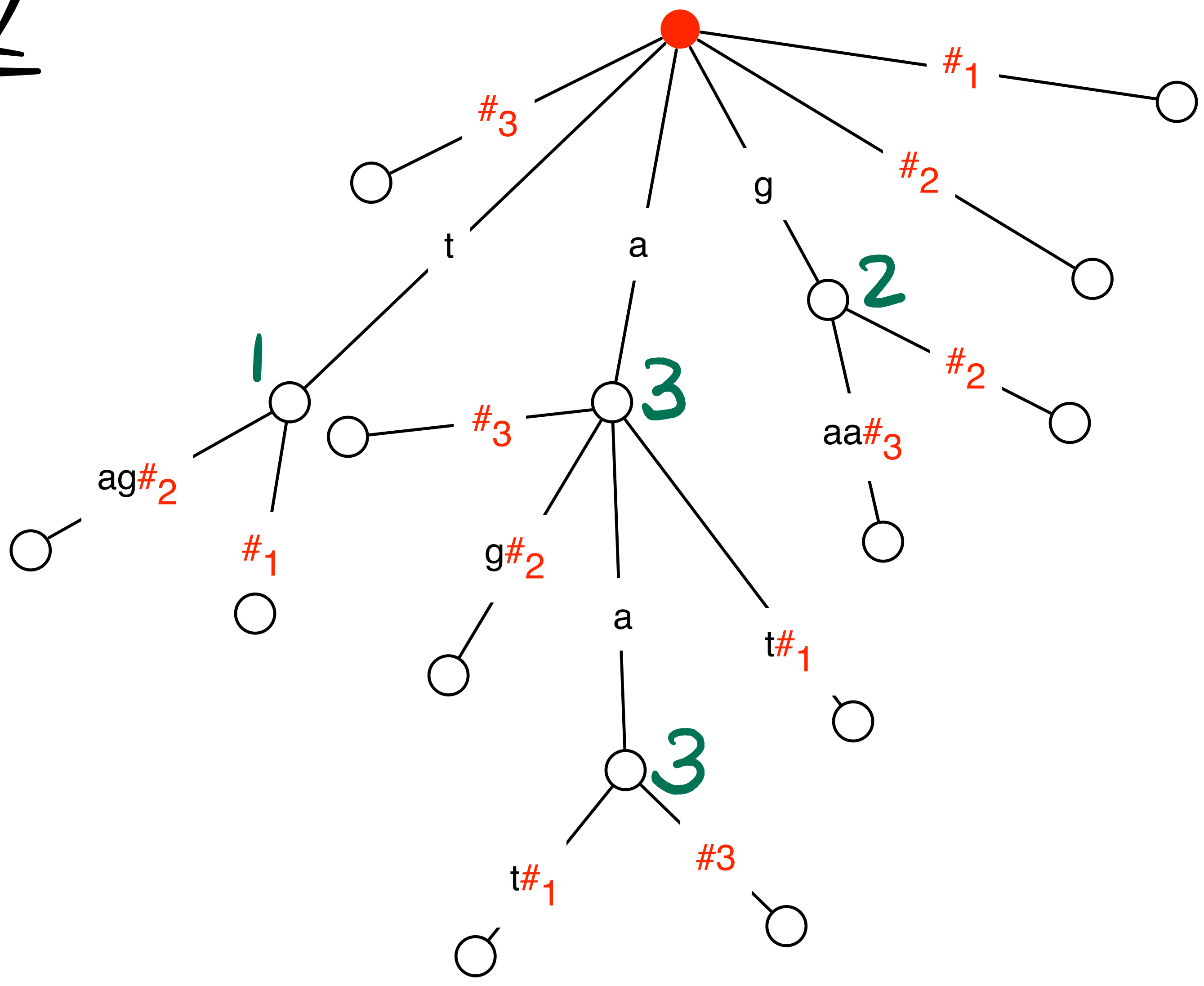
$O(m^2 + M)$



	S_1	S_2	S_3
S_1	–	1	0
S_2	0	–	1
S_3	2	0	–

	S_1	S_2	S_3
S_1	–	~~0~~ 1	0
S_2	0	–	~~0~~ 1
S_3	~~1~~ ~~0~~ 2	0	–

S_1 S_2 S_3
aat, tag, gaa



Applications of Generalized Suffix Trees

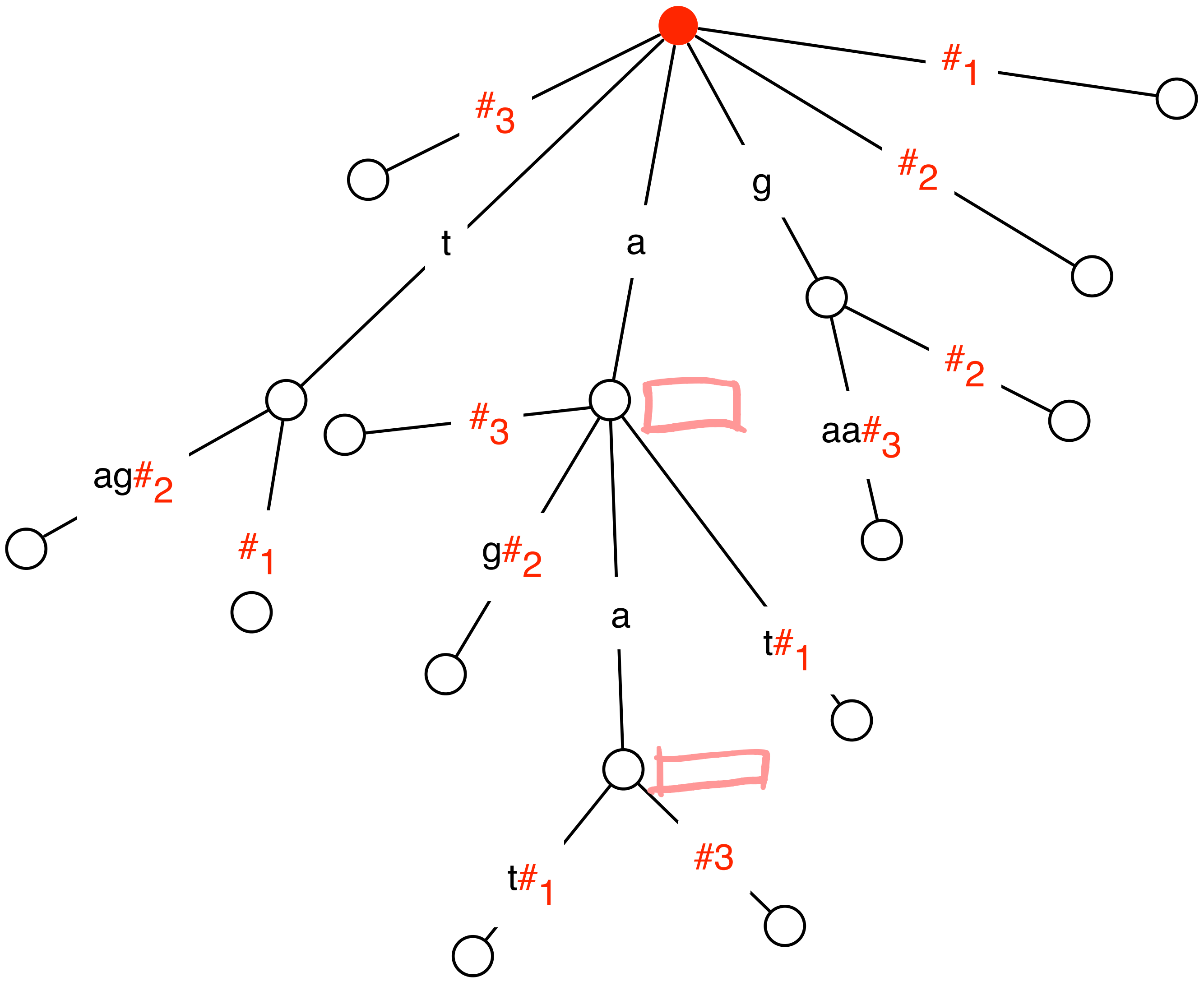
Given {S1, S2, S3, ..., Sm}, find the longest suffix-prefix exact matches for every pair.

- Build generalized suffix tree for {S1, S2, S3, ..., Sm} O(M)
- Init the output table $O(m^2)$
- Create the suffix-list for each node O(M)
- Traverse each string and update the table

total length of lists = O(M)

O(M) + O(M)

aat, tag, gaa

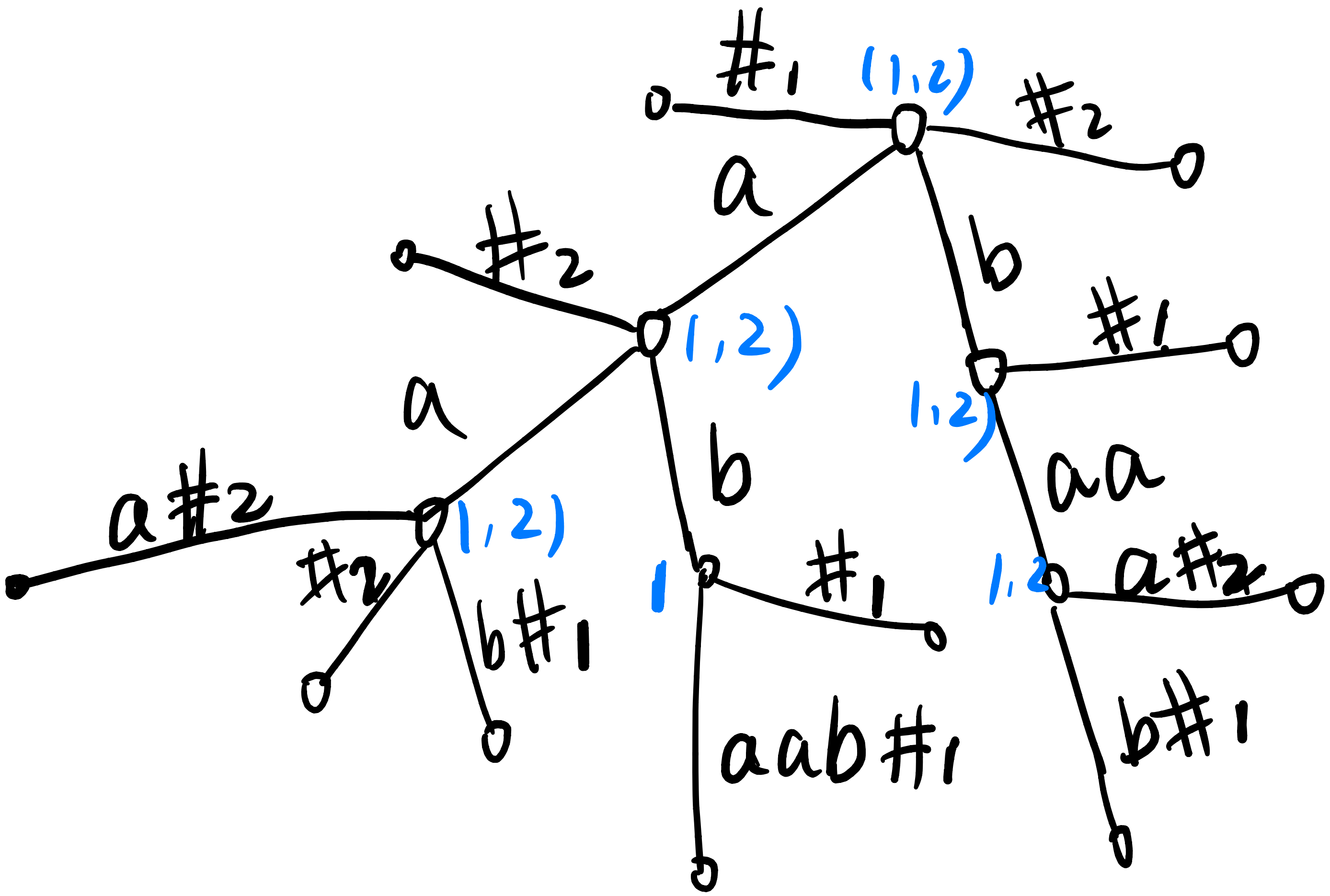


Applications of Generalized Suffix Trees

Longest common substring of S and T:

$S = a\underline{baa}b\#_1$

$T = \underline{baa}a\#_2$



Applications of Generalized Suffix Trees

Longest common substring of S and T:

Build generalized suffix tree for {S, T} $O(|S|+|T|)$

Find the "deepest" node that has has descendants from both strings (containing both #1 and #2)

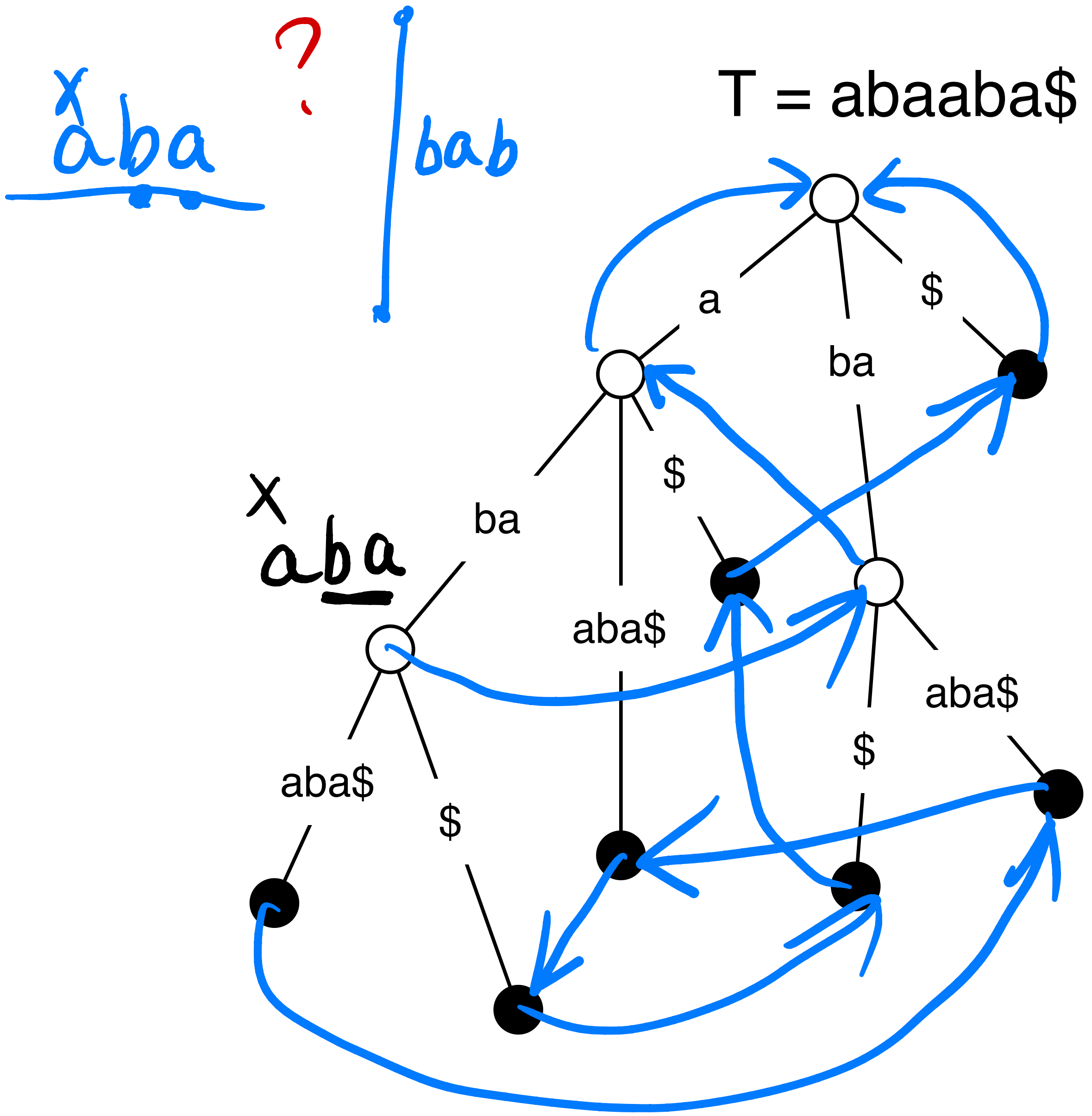
$O(|S|+|T|)$.

Suffix Link

S : substring

x : letter

- Suffix link: pointer connects node represents "xS" to "S"
- Defined for both suffix trie and suffix tree.
- Every node has a suffix link!



T = abaaba$