Suffix tree

- Build a tree from the text
- Used if the text is expected to be the same during several pattern queries
- Tree building is O(m) where m is the size of the text. This is preprocessing.
- Given any pattern of length n, we can answer if it occurs in text in O(n) time
- Suffix tree = “modified” keyword tree of all suffixes of text

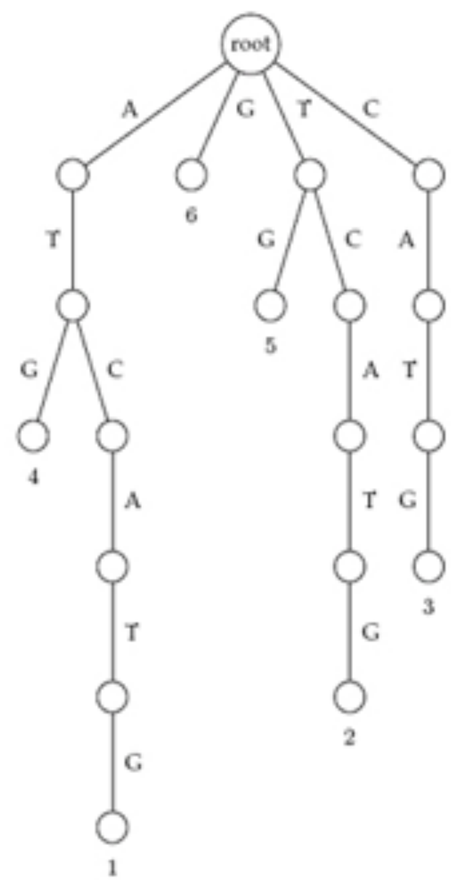
Construct a suffix tree

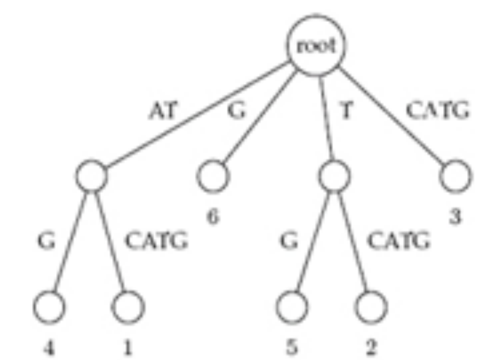
Text: ATCATG

suffixes

ATCATG
TCATG
CATG
ATG
TG
G

Keyword Tree → Suffix Tree



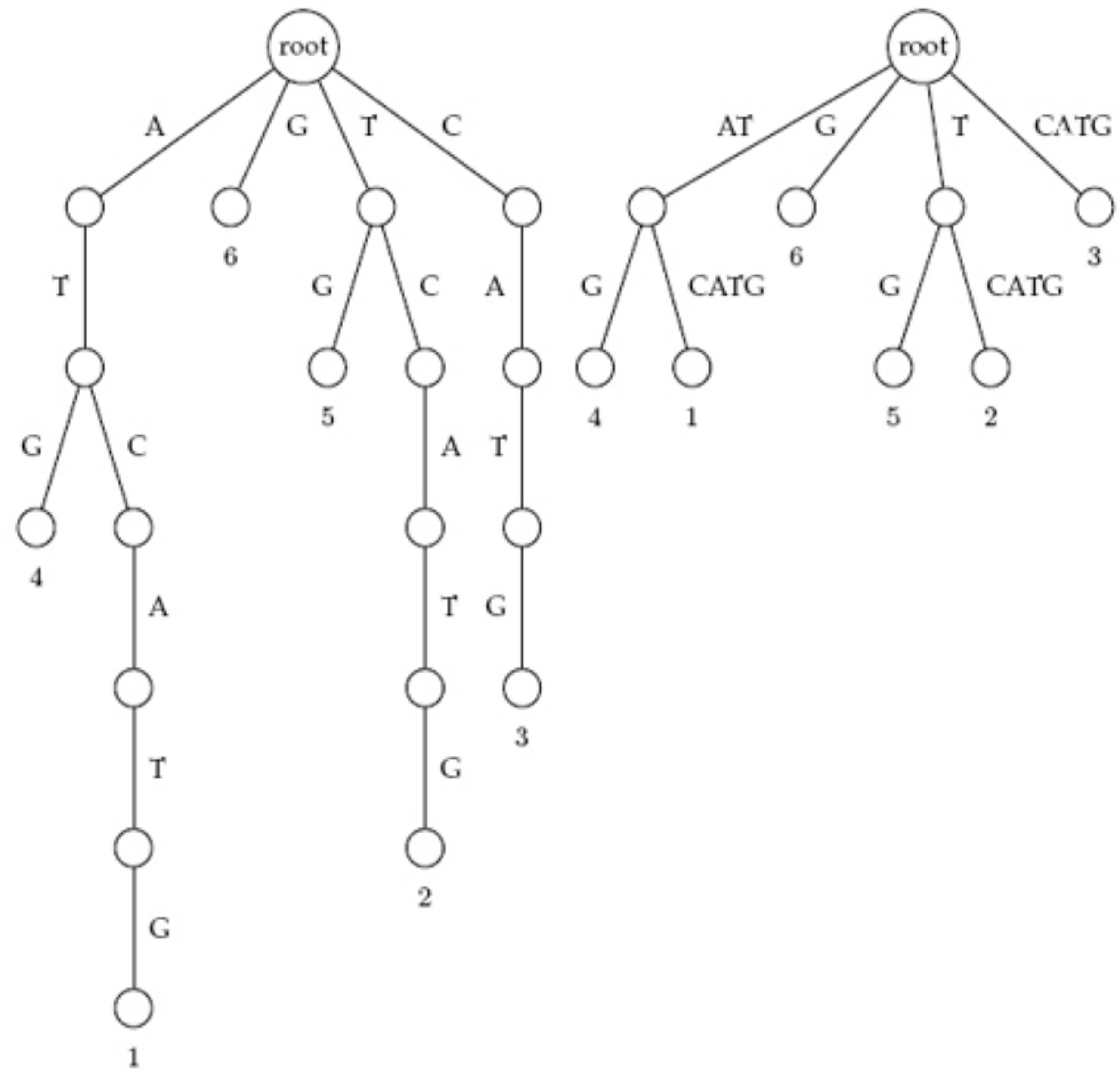


Suffix tree = Collapsed Keyword Tree on Suffixes

Similar to keyword trees, except edges that form paths are collapsed

- Each edge is labeled with a *substring* of a text for less space
- All internal edges have at least two outgoing edges
- Leaves labeled by the location of the suffix on the text.

Text: ATCATG



(a) Keyword tree

(b) Suffix tree

All suffixes of text T

```
T: GTTATAGCTGATCGCGGCGTAGCGG
   GTTATAGCTGATCGCGGCGTAGCGG
    TTATAGCTGATCGCGGCGTAGCGG
     TATAGCTGATCGCGGCGTAGCGG
      ATAGCTGATCGCGGCGTAGCGG
       TAGCTGATCGCGGCGTAGCGG
        AGCTGATCGCGGCGTAGCGG
         GCTGATCGCGGCGTAGCGG
          CTGATCGCGGCGTAGCGG
           TGATCGCGGCGTAGCGG
            GATCGCGGCGTAGCGG
             ATCGCGGCGTAGCGG
              TCGCGGCGTAGCGG
               CGCGGCGTAGCGG
                GCGGCGTAGCGG
                 CGGCGTAGCGG
                  GGCGTAGCGG
                   GCGTAGCGG
                    CGTAGCGG
                     GTAGCGG
                      TAGCGG
                       AGCGG
                        GCGG
                         CGG
                          GG
                           G
```

$m(m+1)/2$ chars

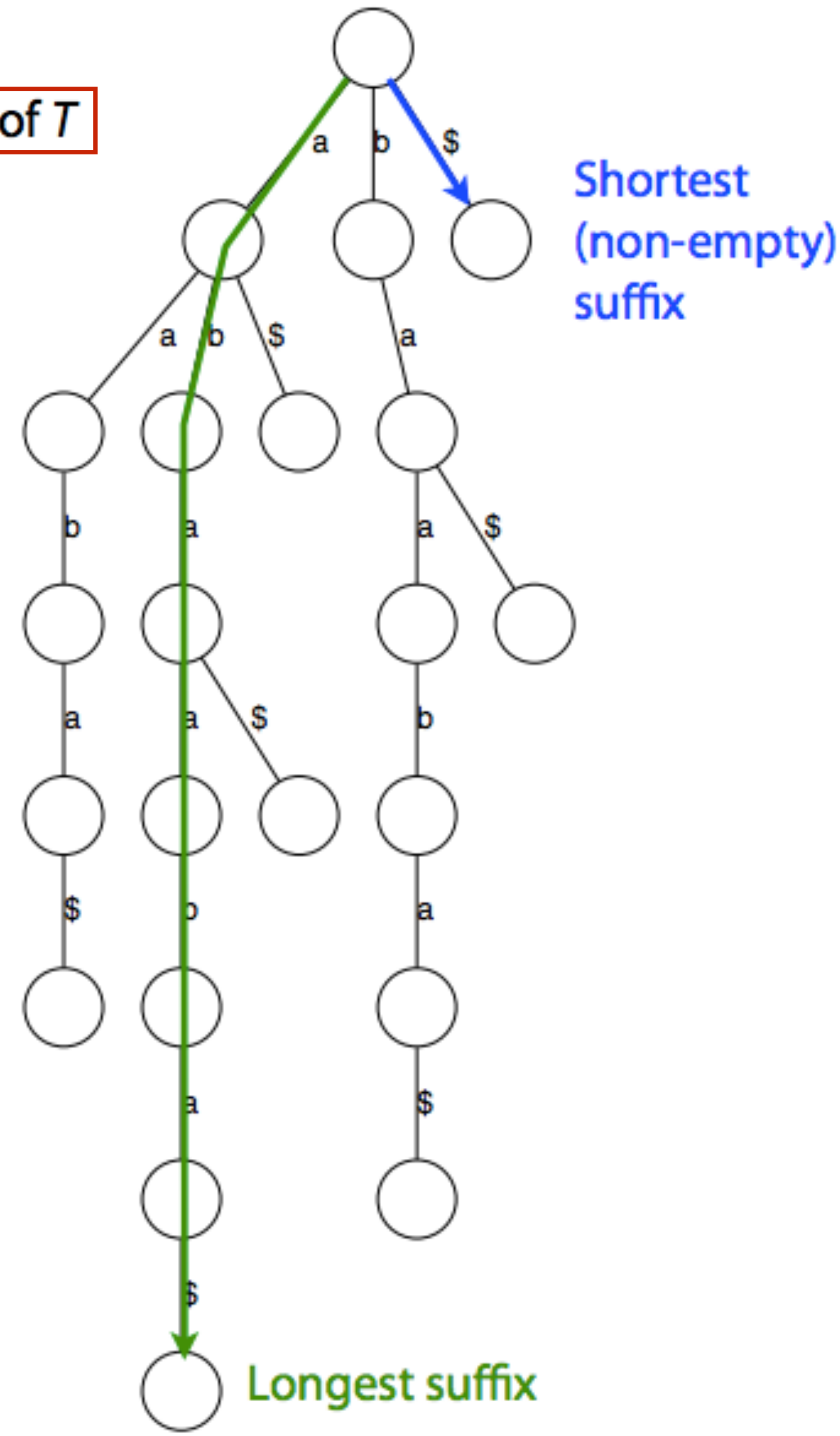
Example: suffix keyword tree

add special *terminal character* **$** to the end of *T*

T: abaaba *T***$***:* abaaba**$**

Each path from root to leaf represents a suffix; each suffix is represented by some path from root to leaf

Would this still be the case if we hadn't added **$**?

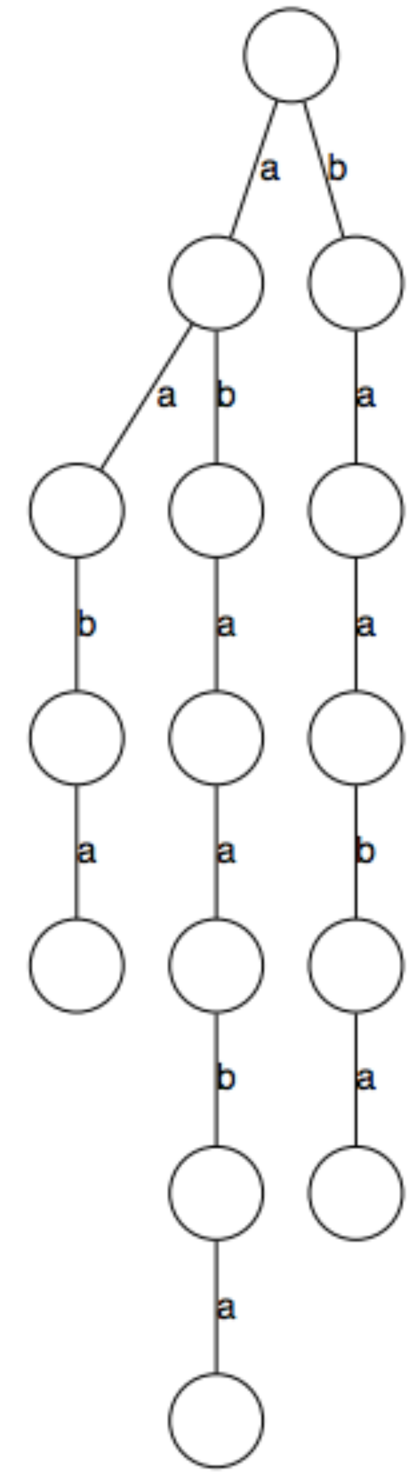


Example: suffix keyword tree

T: abaaba

Each path from root to leaf represents a suffix; each suffix is represented by some path from root to leaf

Would this still be the case if we hadn't added **$**? **No**



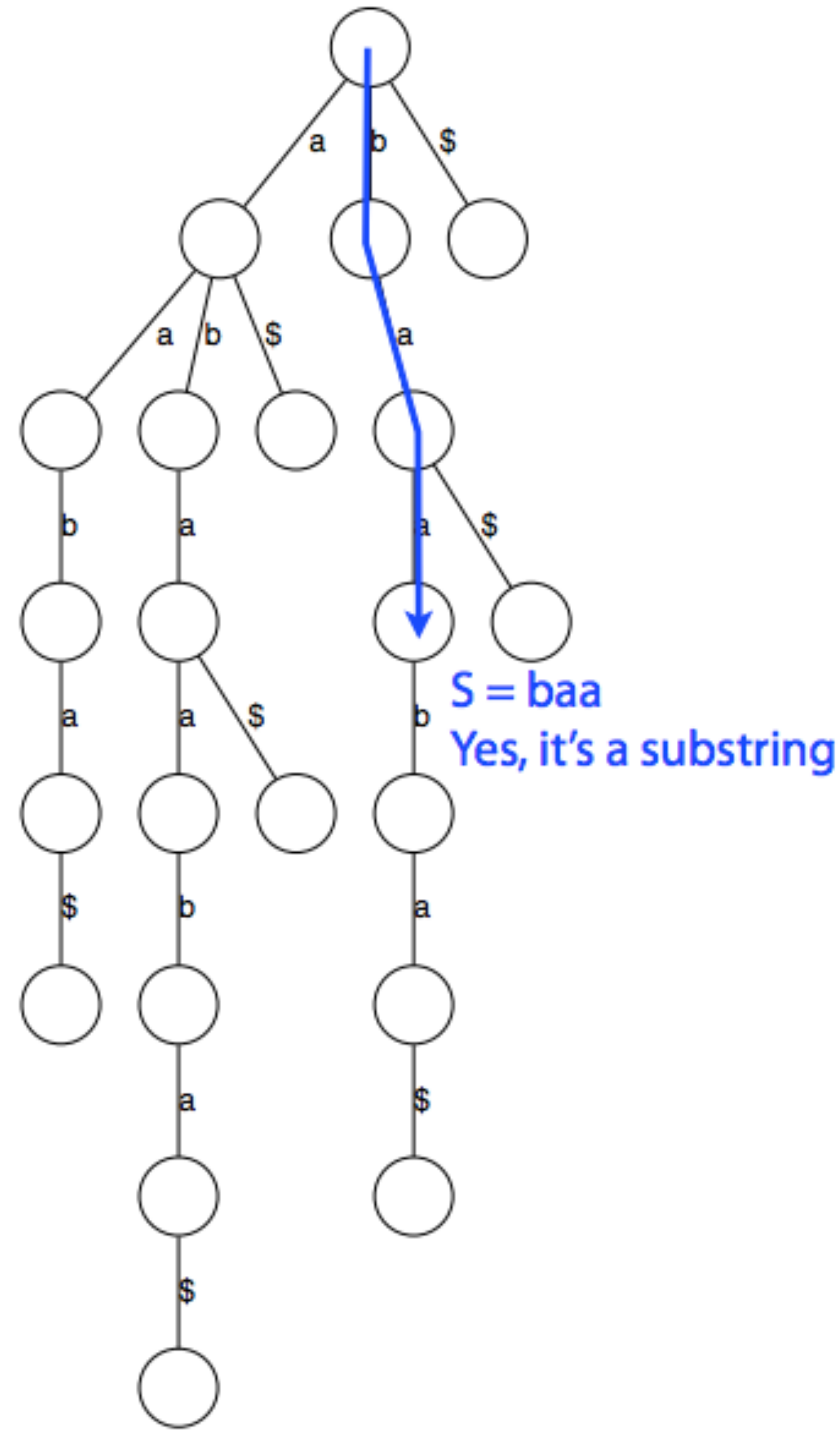
Example: suffix keyword tree

How do we check whether a string *S* is a substring of *T*?

Note: Each of *T*'s substrings is spelled out along a path from the root. I.e., every *substring* is a *prefix* of some *suffix* of T.

Start at the root and follow the edges labeled with the characters of *S*

- If we "fall off" the trie -- i.e. there is no outgoing edge for next character of *S*, then *S* is not a substring of *T*
- If we exhaust *S* without falling off, *S* is a substring of *T*



Example: suffix keyword tree

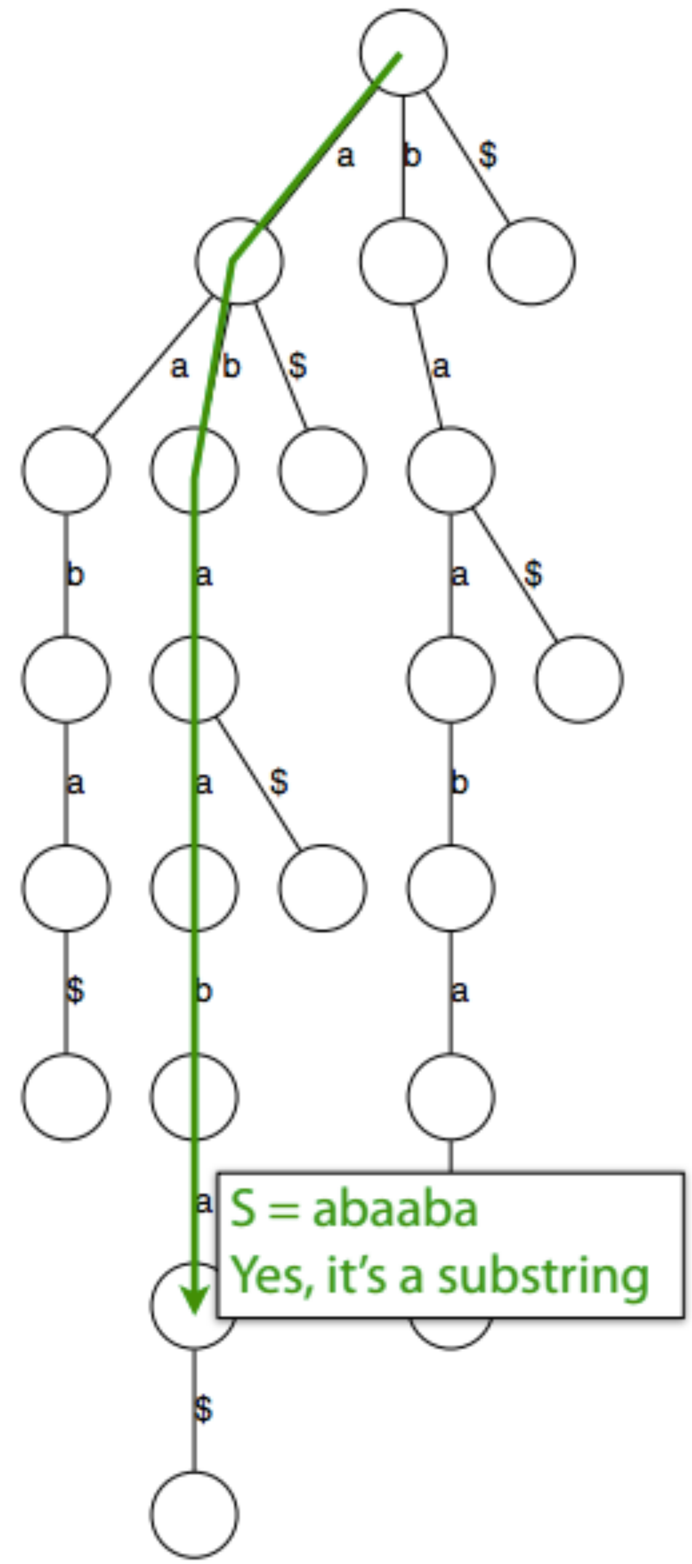
How do we check whether a string *S* is a substring of *T*?

Note: Each of *T*'s substrings is spelled out along a path from the root. I.e., every *substring* is a *prefix* of some *suffix* of T.

Start at the root and follow the edges labeled with the characters of *S*

If we "fall off" the trie -- i.e. there is no outgoing edge for next character of *S*, then *S* is not a substring of *T*

If we exhaust *S* without falling off, *S* is a substring of *T*



Example: suffix keyword tree

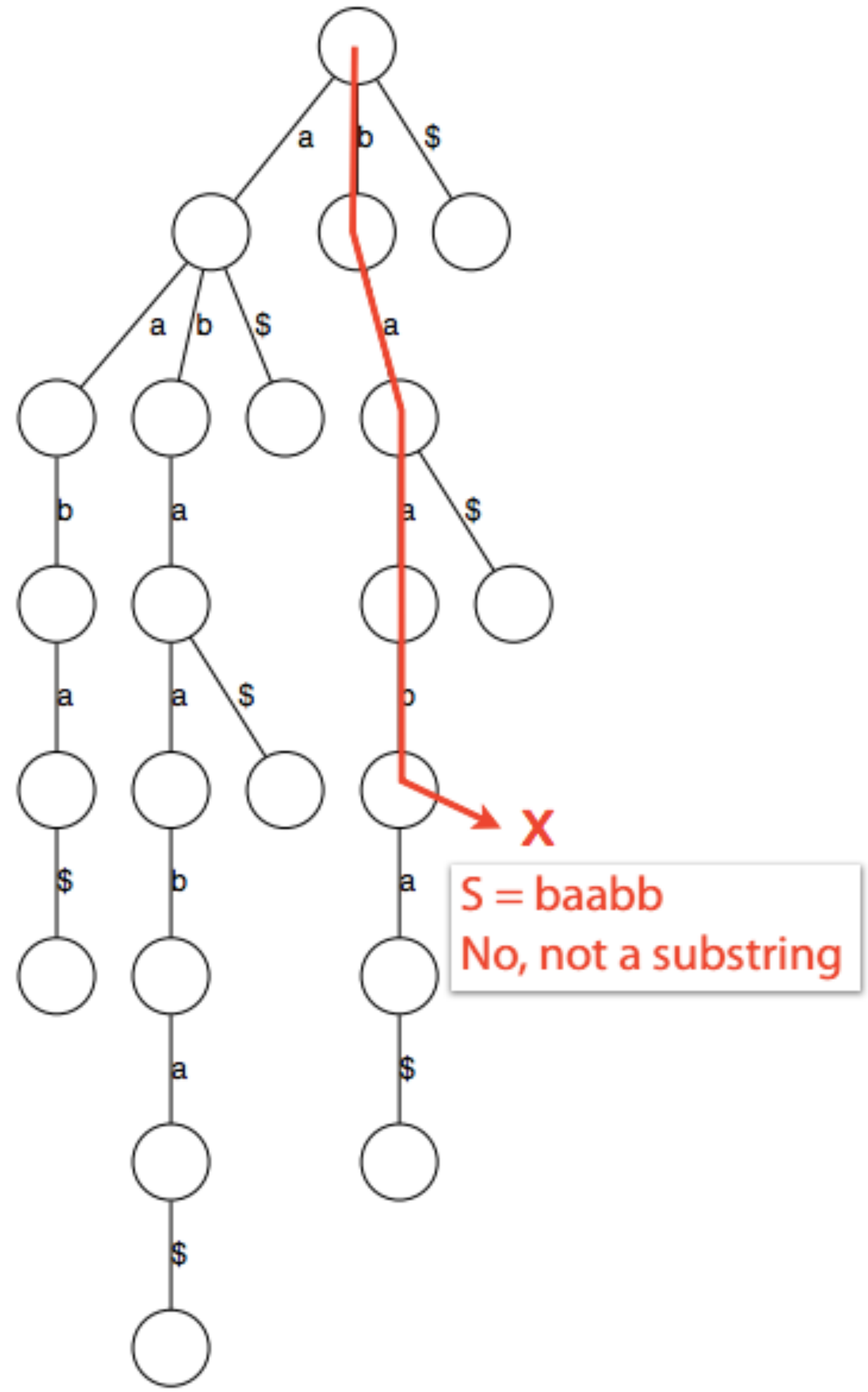
How do we check whether a string *S* is a substring of *T*?

Note: Each of *T*'s substrings is spelled out along a path from the root. I.e., every *substring* is a *prefix* of some *suffix* of T.

Start at the root and follow the edges labeled with the characters of *S*

If we "fall off" the trie -- i.e. there is no outgoing edge for next character of *S*, then *S* is not a substring of *T*

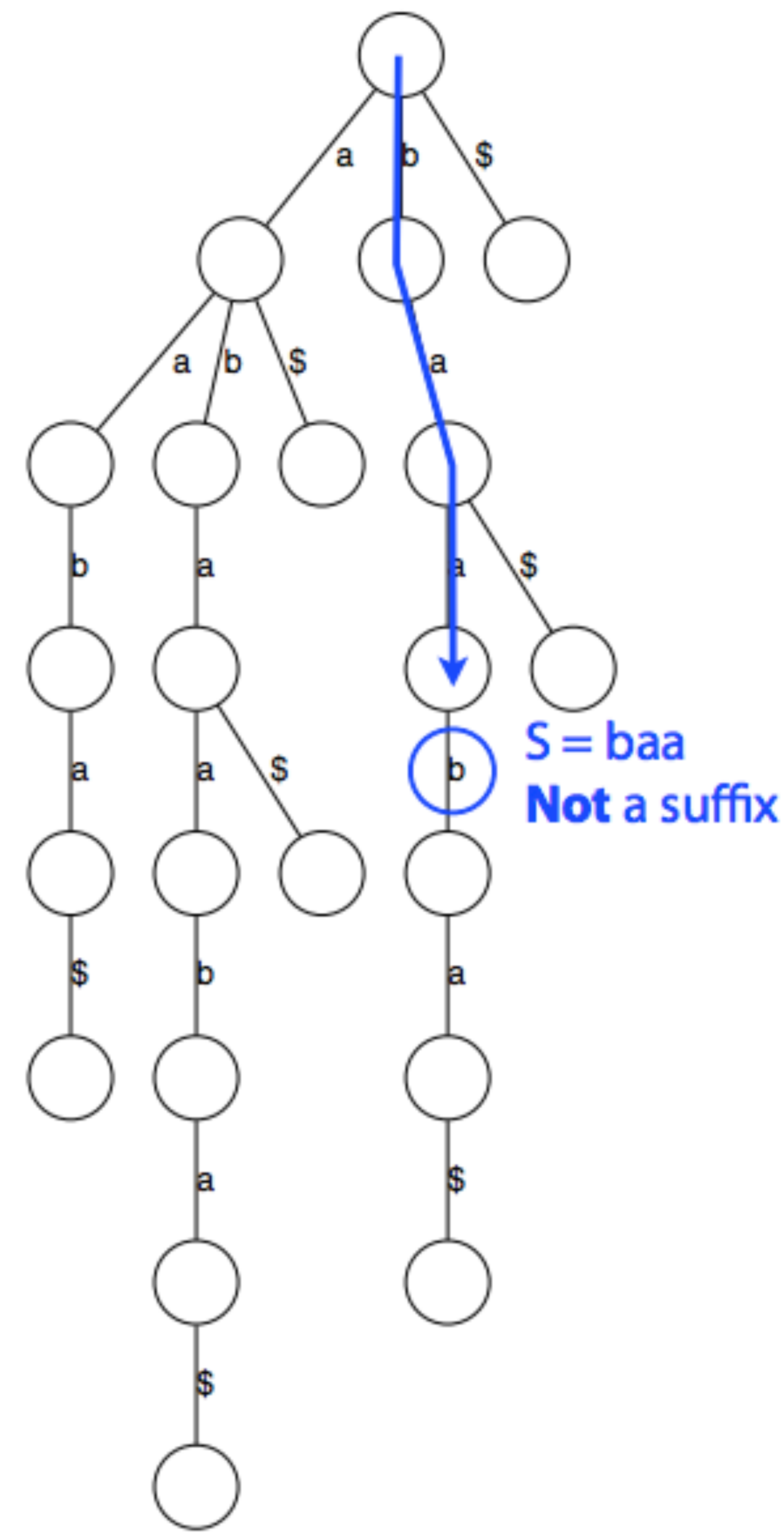
If we exhaust *S* without falling off, *S* is a substring of *T*



Example: suffix keyword tree

How do we check whether a string *S* is a **suffix** of *T*?

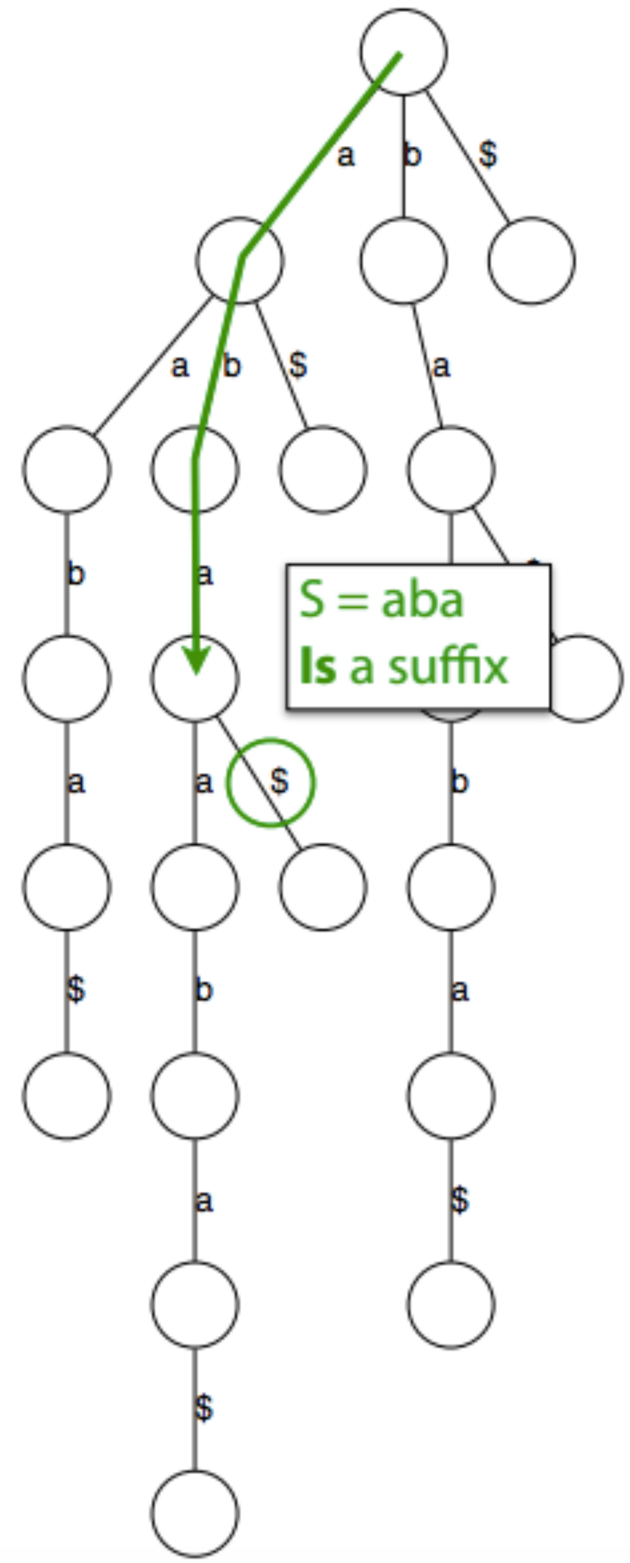
Same procedure as for substring, but additionally check whether the final node in the walk has an outgoing edge labeled **$**



Example: suffix keyword tree

How do we check whether a string *S* is a **suffix** of *T*?

Same procedure as for substring, but additionally check whether the final node in the walk has an outgoing edge labeled **$**

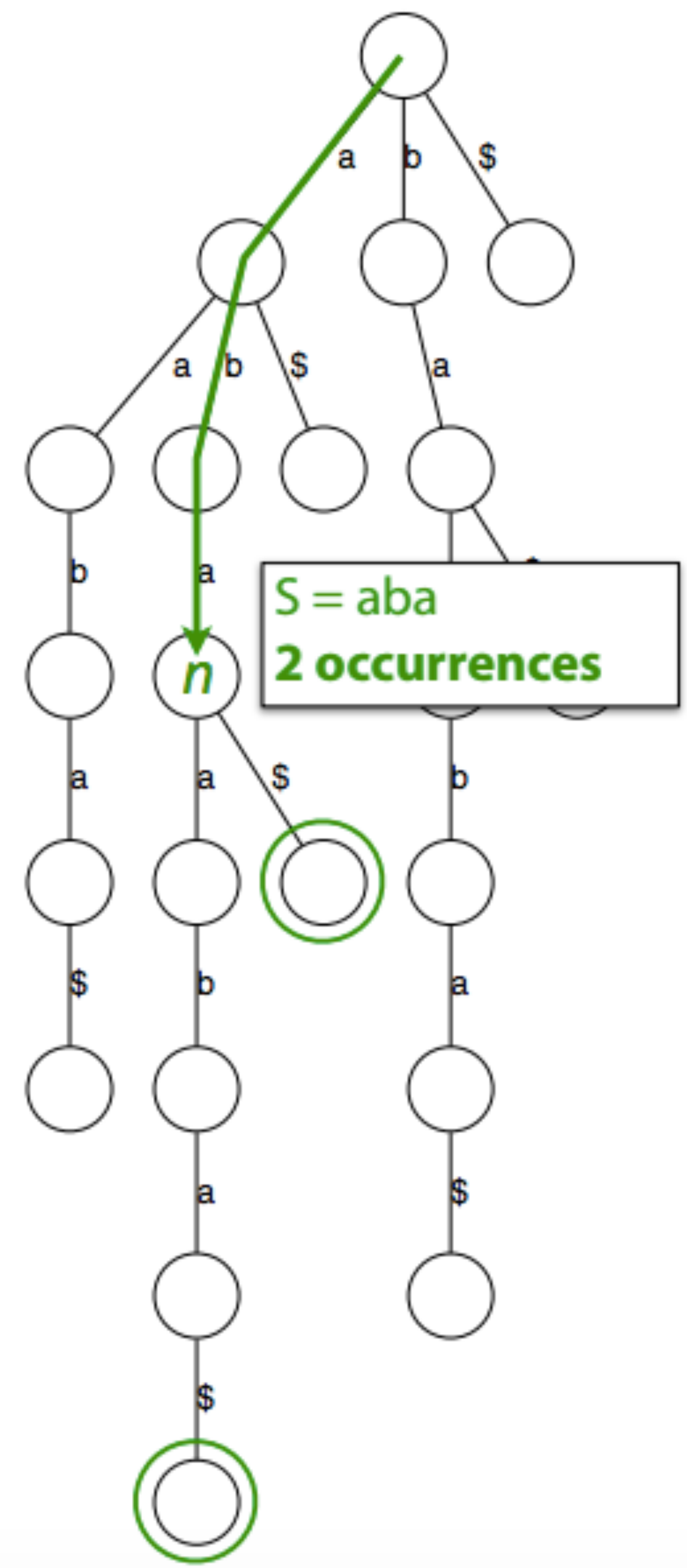


Example: suffix keyword tree

How do we count the **number of times** a string *S* occurs as a substring of *T*?

Follow path corresponding to *S*. Either we fall off, in which case answer is 0, or we end up at node *n* and the answer = # of leaf nodes in the subtree rooted at *n*.

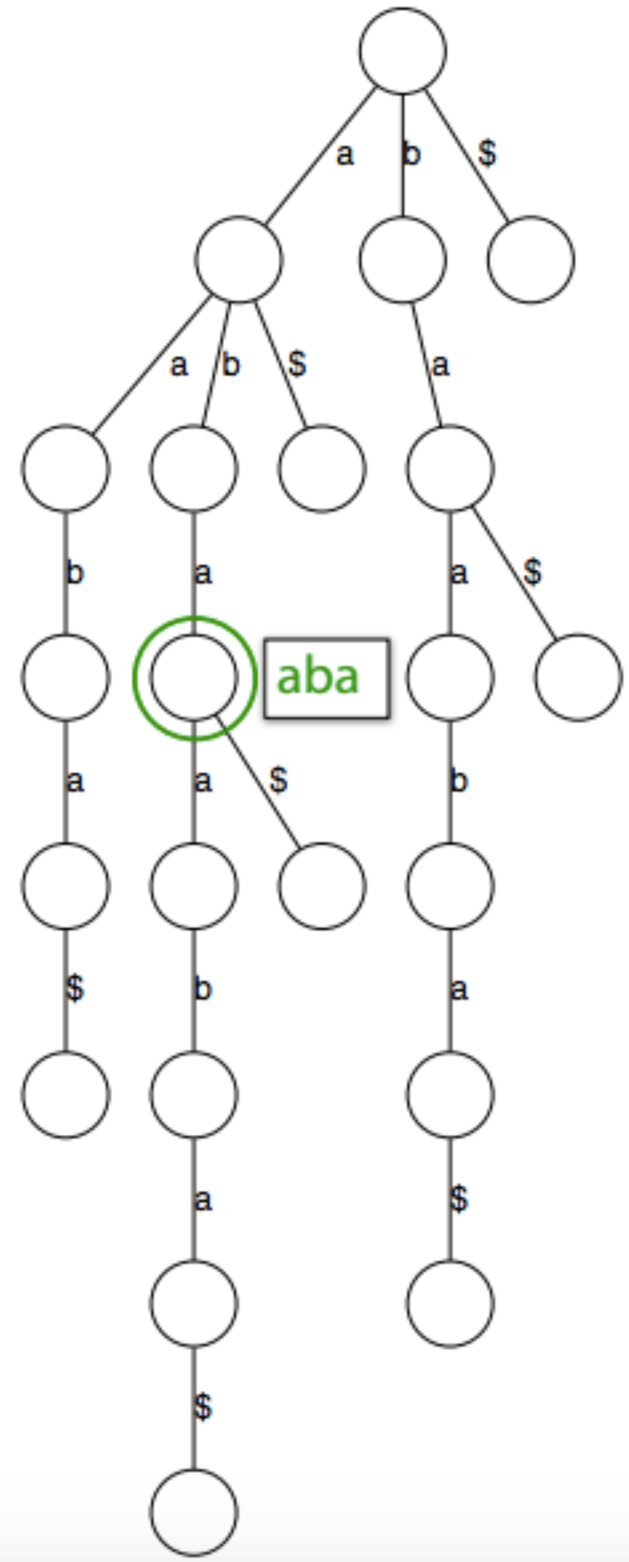
Leaves can be counted with depth-first traversal.



Example: suffix keyword tree

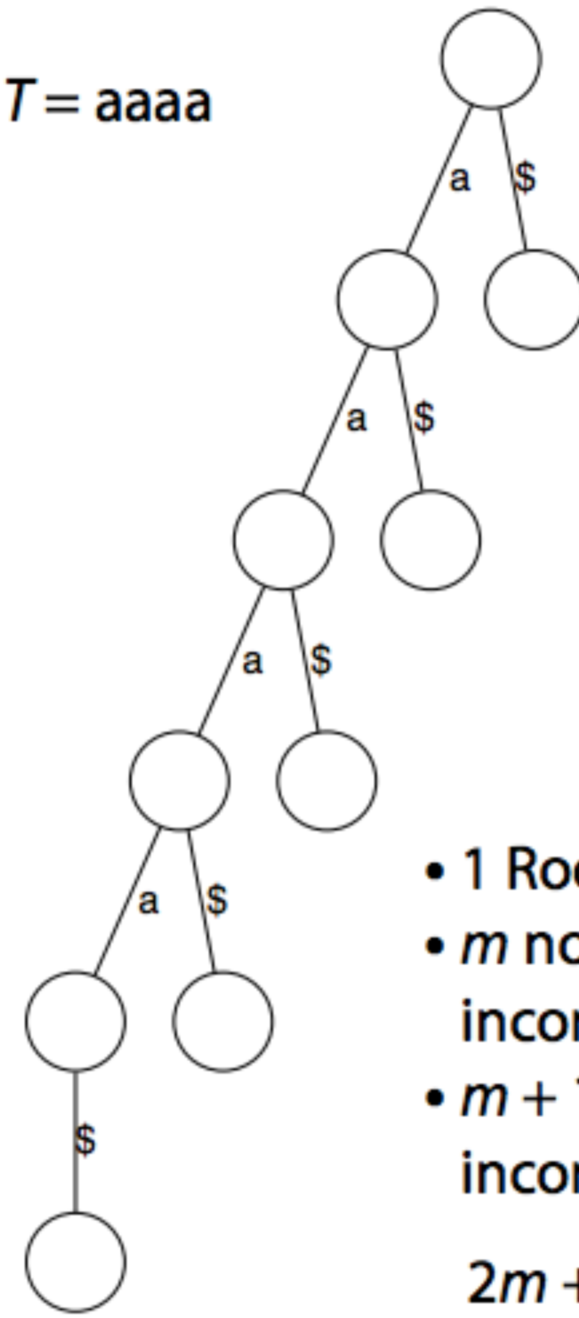
How do we find the **longest repeated substring** of *T*?

Find the deepest node with more than one child



How many nodes does a suffix keyword tree have?

$T = \text{aaaa}$



- 1 Root
- *m* nodes with incoming **a** edge
- *m* + 1 nodes with incoming **$** edge

$2m + 2$ nodes

How many nodes does a suffix keyword tree have?

```
T: GTTATAGCTGATCGCGGCGTAGCGG
   GTTATAGCTGATCGCGGCGTAGCGG
    TTATAGCTGATCGCGGCGTAGCGG
     TATAGCTGATCGCGGCGTAGCGG
      ATAGCTGATCGCGGCGTAGCGG
       TAGCTGATCGCGGCGTAGCGG
        AGCTGATCGCGGCGTAGCGG
         GCTGATCGCGGCGTAGCGG
          CTGATCGCGGCGTAGCGG
           TGATCGCGGCGTAGCGG
            GATCGCGGCGTAGCGG
             ATCGCGGCGTAGCGG
              TCGCGGCGTAGCGG
               CGCGGCGTAGCGG
                GCGGCGTAGCGG
                 CGGCGTAGCGG
                  GGCGTAGCGG
                   GCGTAGCGG
                    CGTAGCGG
                     GTAGCGG
                      TAGCGG
                       AGCGG
                        GCGG
                         CGG
                          GG
                           G
```

m(m+1)/2 chars

How many nodes does a suffix keyword tree have?

Is there a class of string where the number of nodes grows with m^2?

Yes: $a^n b^n$

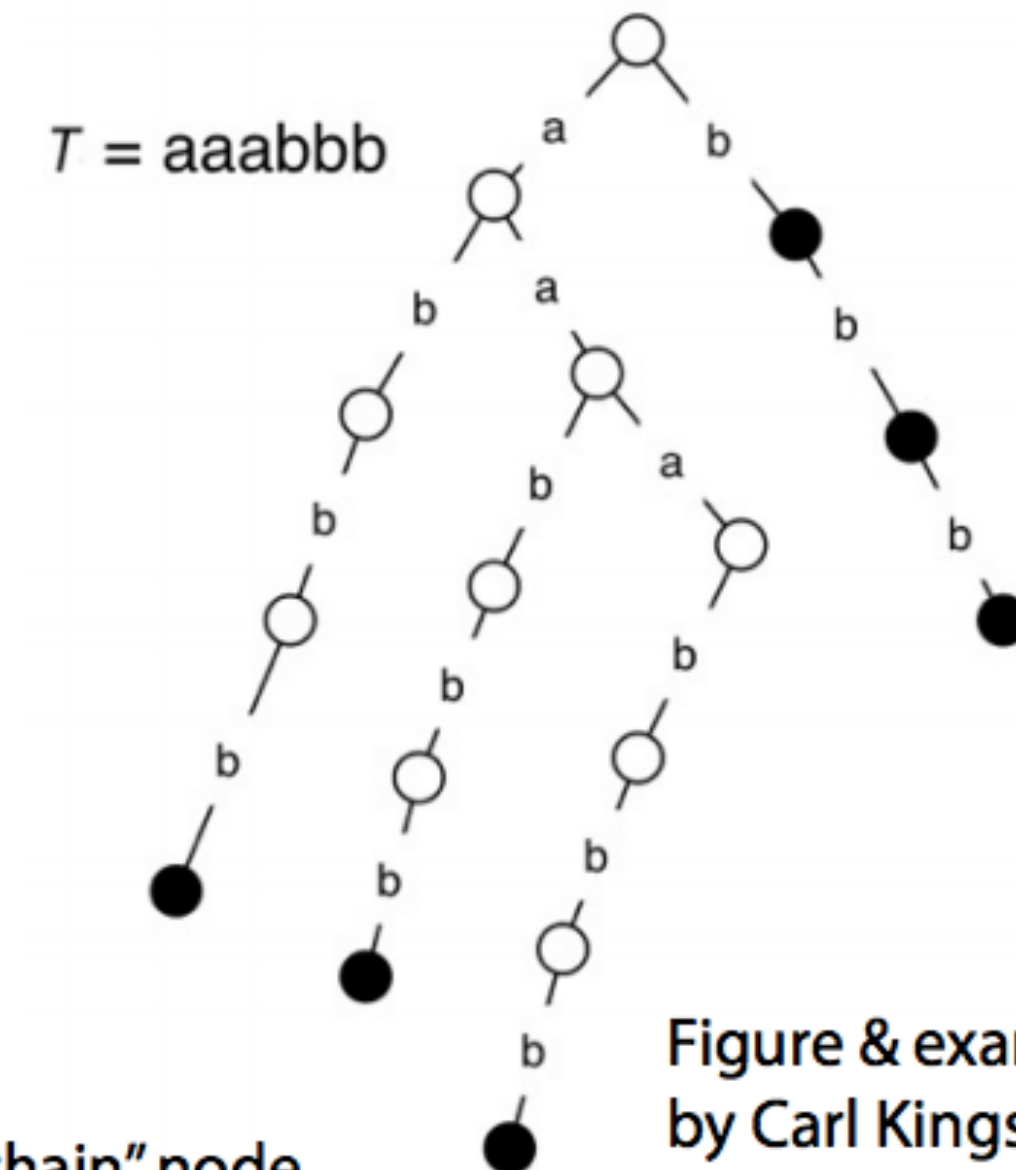


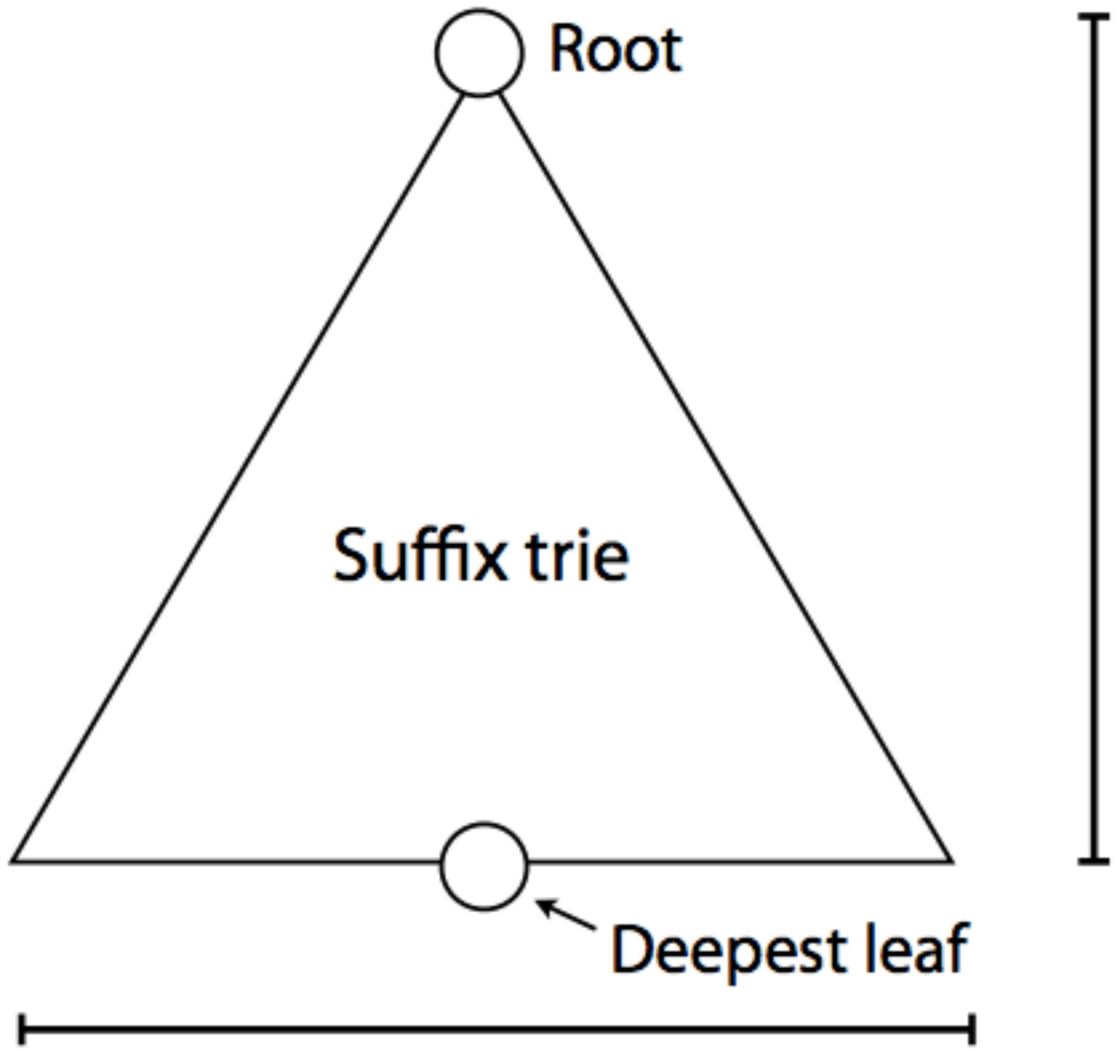
Figure & example by Carl Kingsford

- 1 root
- n nodes along "b chain," right
- n nodes along "a chain," middle
- n chains of n "b" nodes hanging off each "a chain" node
- $2n + 1$ **$** leaves (not shown)

$n^2 + 4n + 2$ nodes, where $m = 2n$

How many nodes does a suffix keyword tree have?

Could worst-case # nodes be worse than $O(m^2)$?



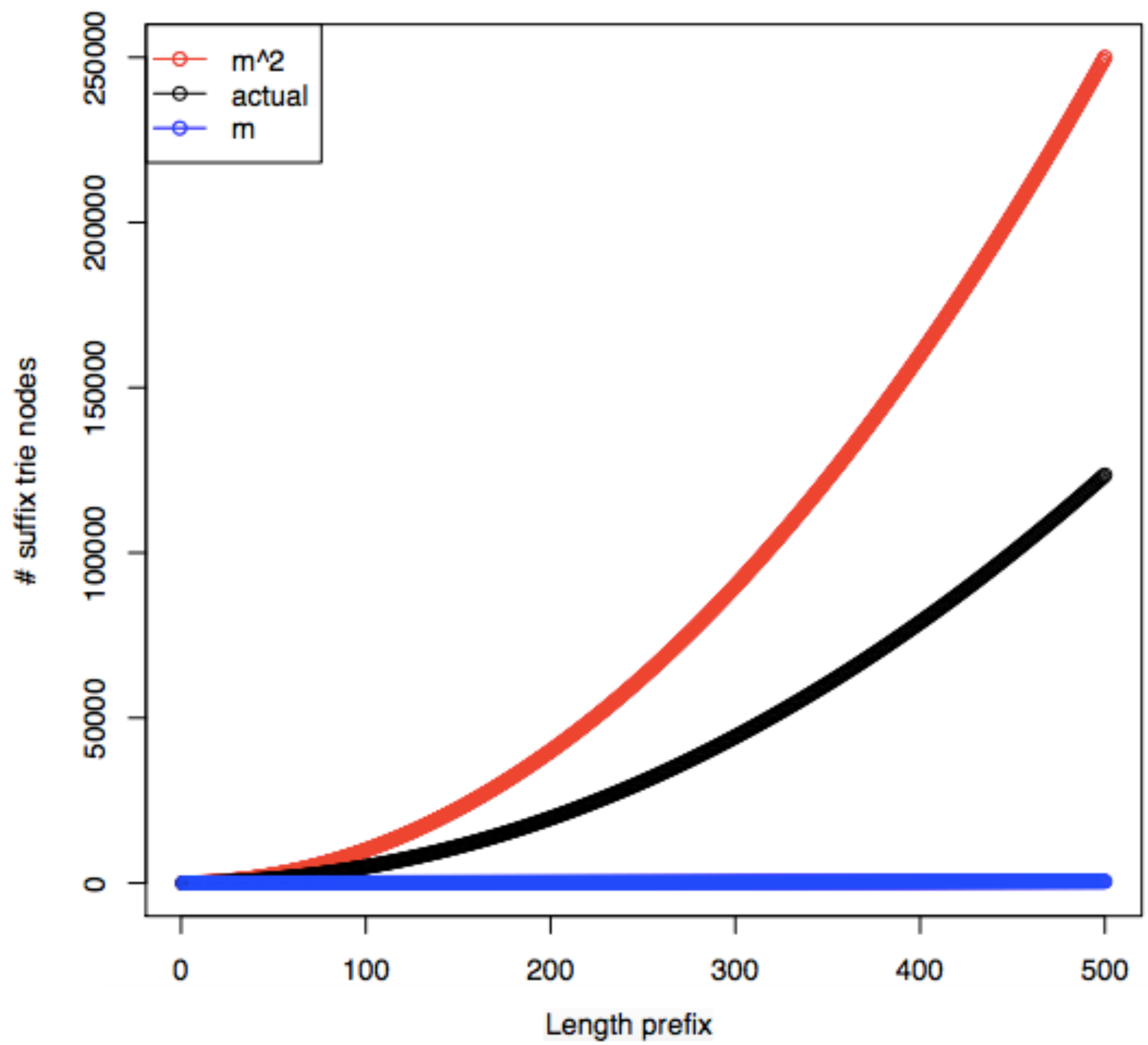
Max # nodes from *top to bottom*
= length of longest suffix + 1
= $m + 1$

Max # nodes from *left to right*
= max # distinct substrings of any length
$\leq m$

$O(m^2)$ is worst case

Actual growth: an example

Trees built using the first 500 prefixes of the lambda phage virus genome



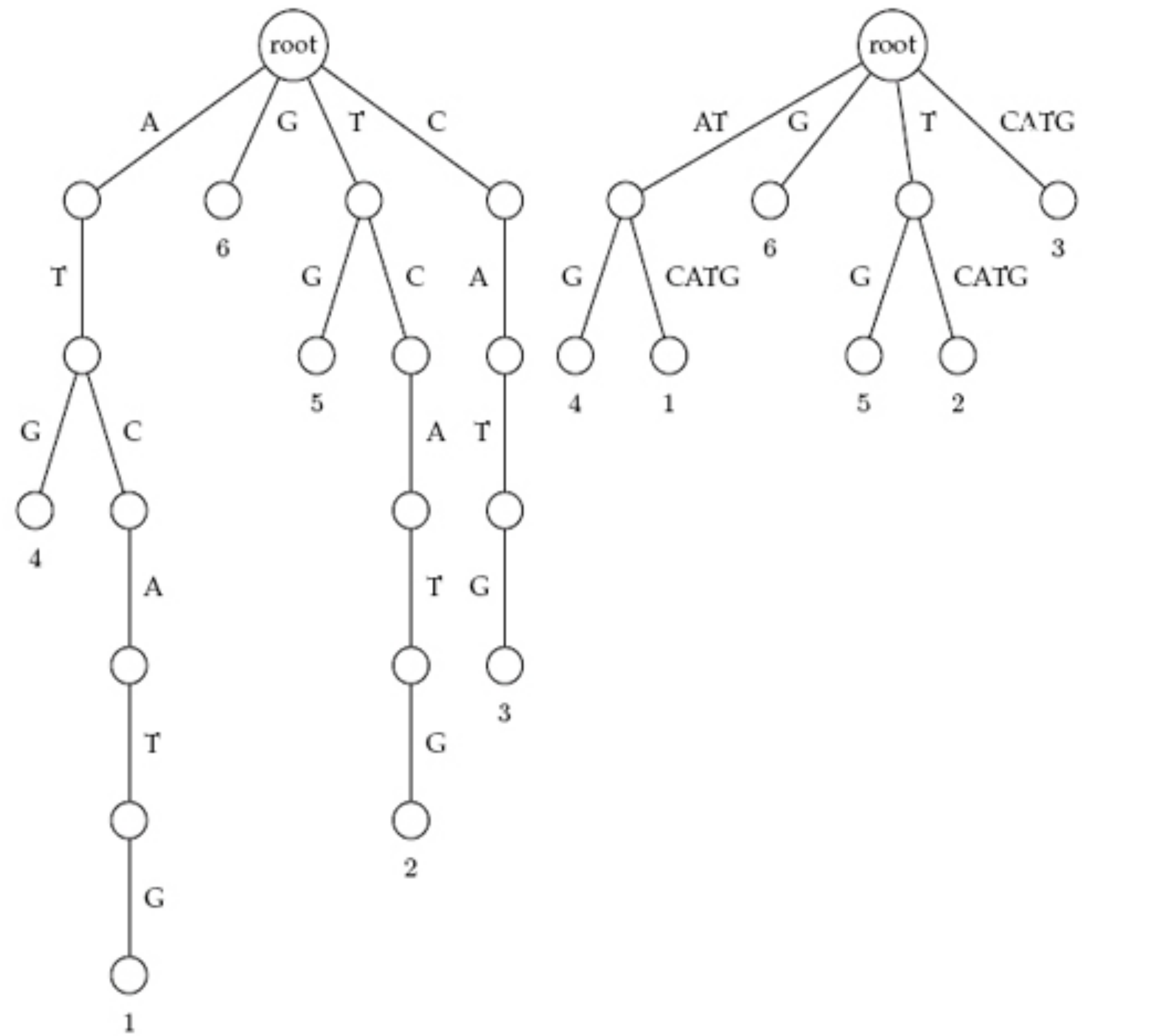
How to compress these trees?

Suffix tree = Collapsed Keyword Tree on Suffixes

Similar to keyword trees, except edges that form paths are collapsed

- Each edge is labeled with a *substring* of a text for less space
- All internal edges have at least two outgoing edges
- Leaves labeled by the location of the suffix on the text.

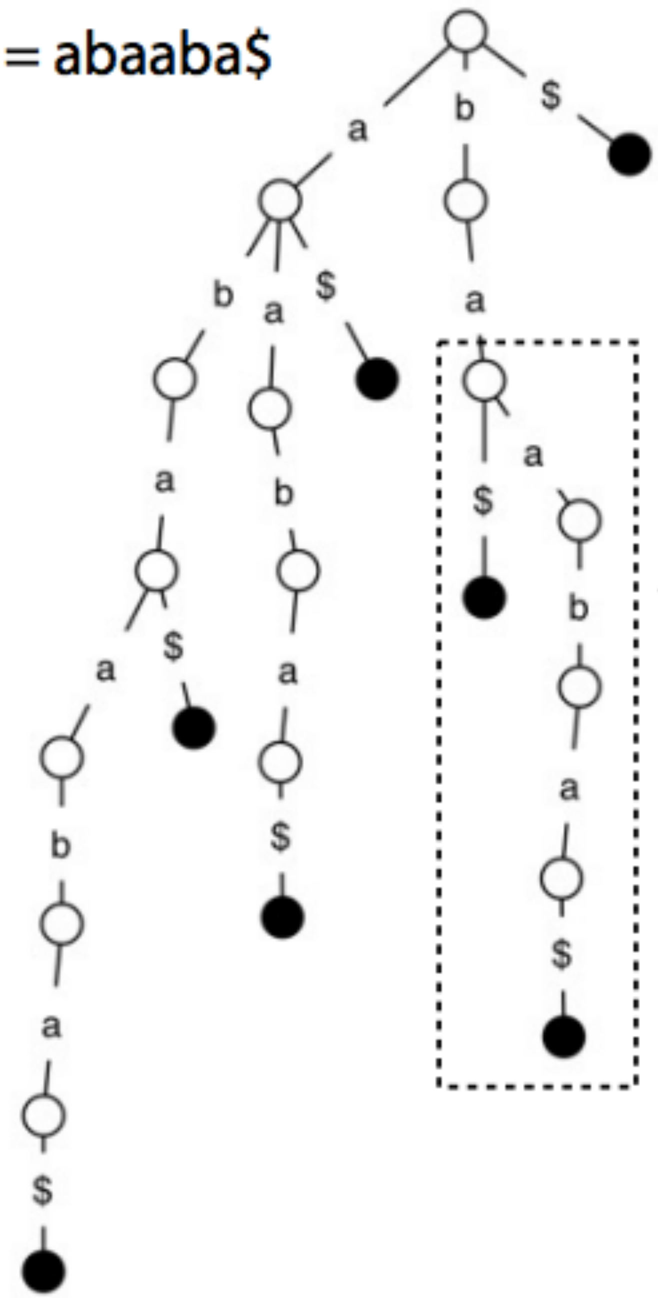
Text: ATCATG



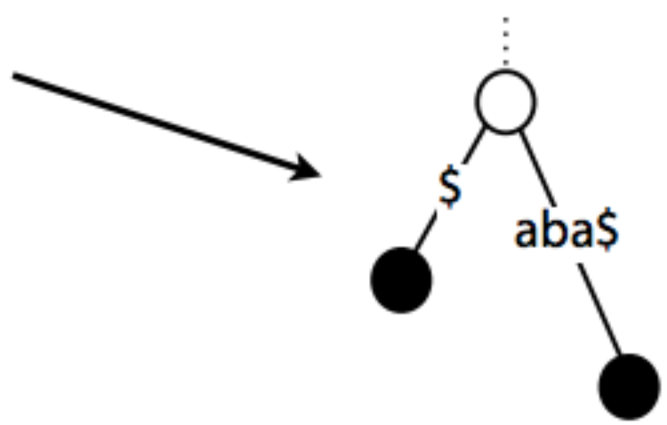
(a) Keyword tree (b) Suffix tree

Compression

T = abaaba$

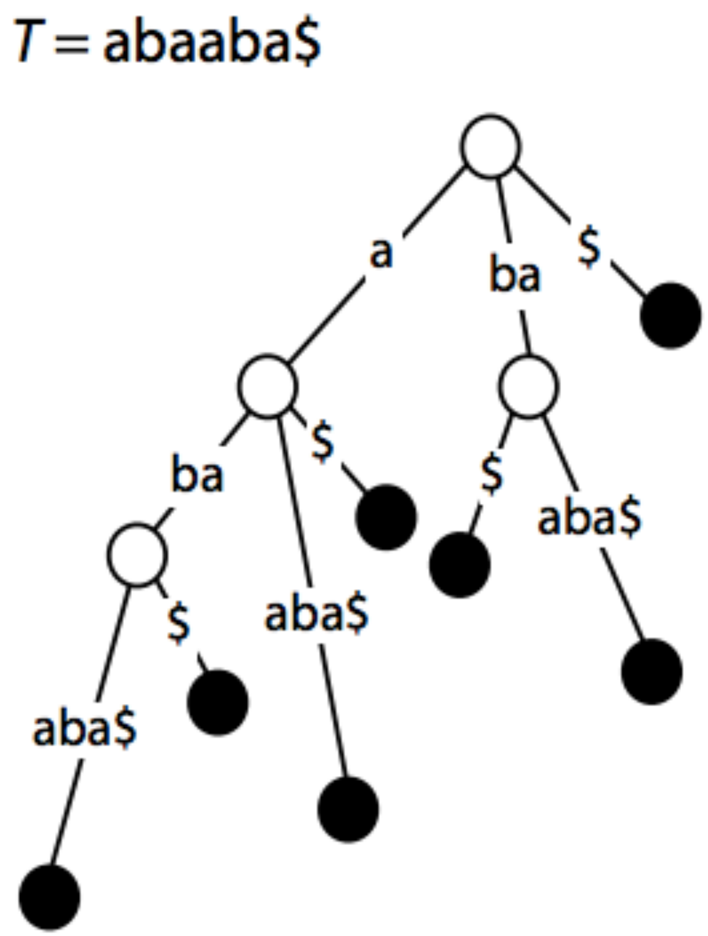


Idea 1: Coalesce non-branching paths into a *single edge* with a *string* label



Reduces # nodes, edges, guarantees internal nodes have >1 child

How many nodes does a suffix tree have?



L leaves, I internal nodes, E edges

$E = L + I - 1$

$E \geq 2I$ (each internal node branches)

$L + I - 1 \geq 2I \Rightarrow I \leq L - 1$

but

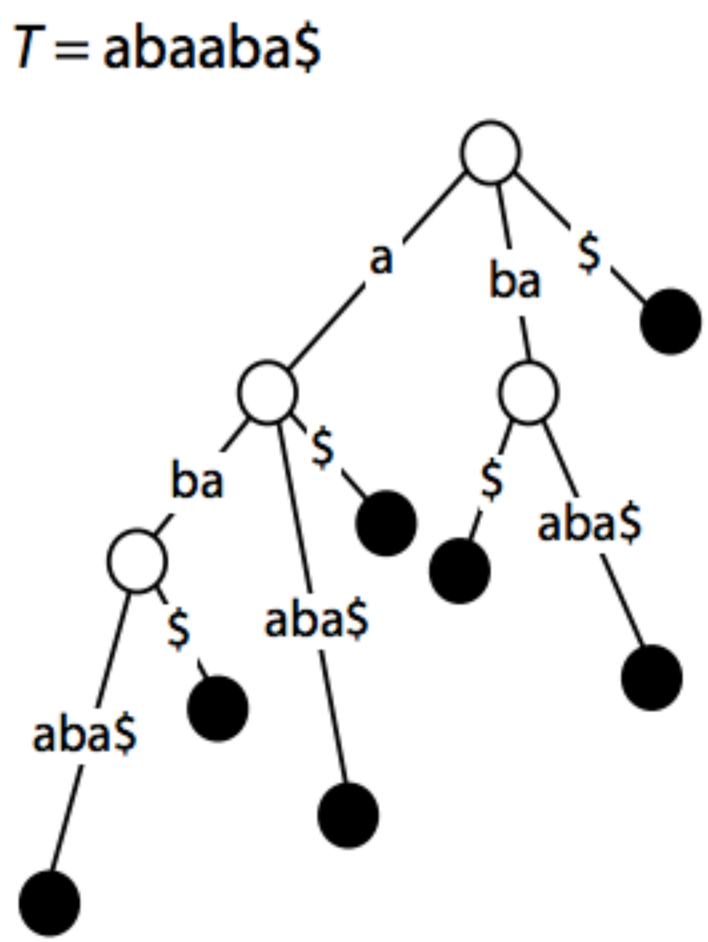
$L \leq m$ (at most m suffixes)

$I \leq m - 1$

$E = L + I - 1 \leq 2m - 2$

$E + L + I \leq 4m - 3 \in O(m)$

Compression



L leaves, I internal nodes, E edges

$E = L + I - 1$

$E \geq 2I$ (each internal node branches)

$L + I - 1 \geq 2I \Rightarrow I \leq L - 1$

but

$L \leq m$ (at most m suffixes)

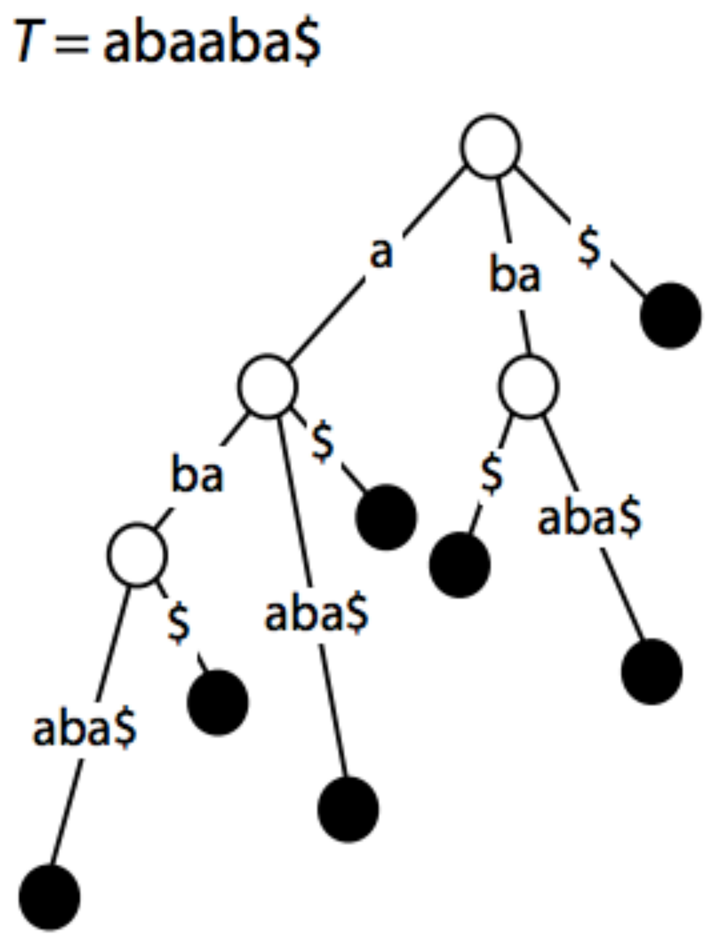
$I \leq m - 1$

$E = L + I - 1 \leq 2m - 2$

$E + L + I \leq 4m - 3 \in O(m)$

Is the total size O(m) now?

Compression



L leaves, I internal nodes, E edges

E = L + I -1

E ≥ 2I (each internal node branches)

L + I -1 ≥ 2I ⇒ I ≤ L - 1

but

L ≤ m (at most m suffixes)

I ≤ m -1

E = L + I - 1 ≤ 2m - 2

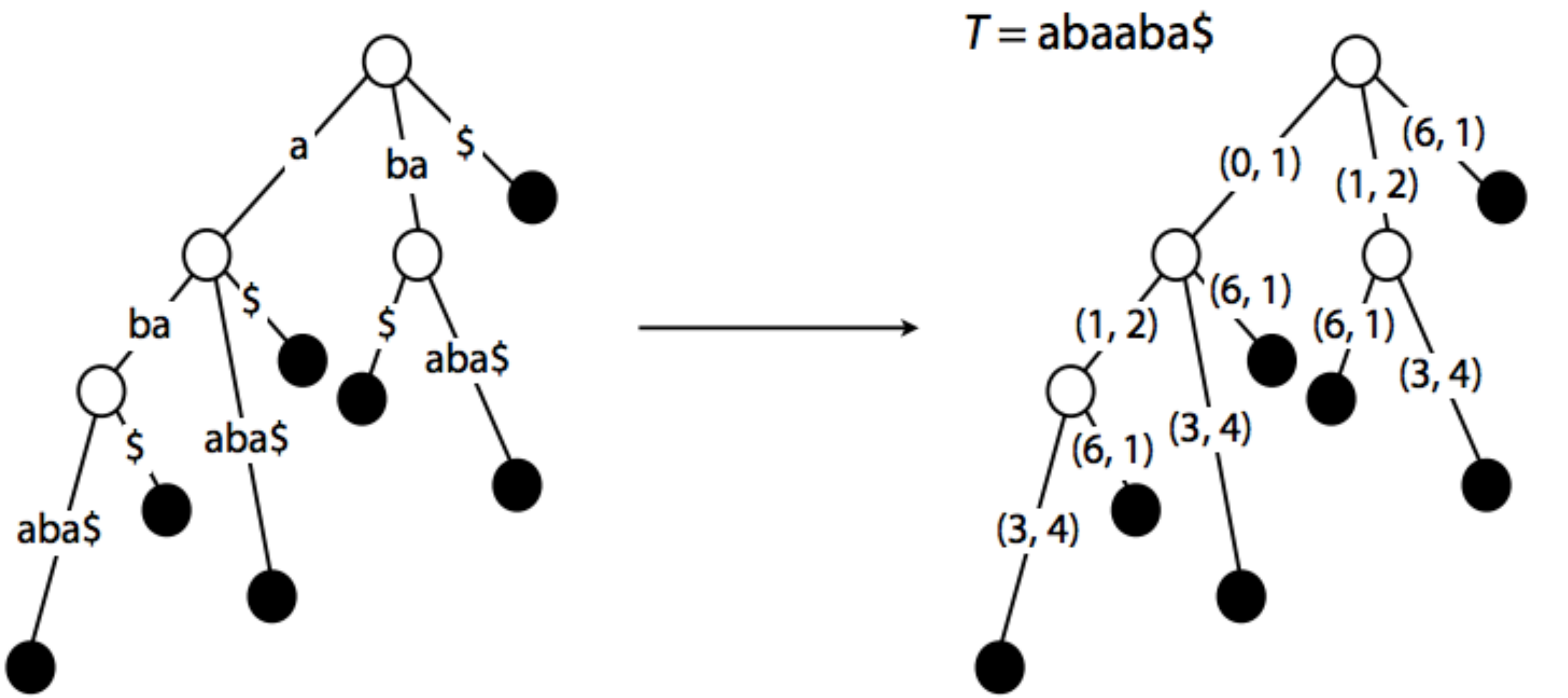
E + L + I ≤ 4m - 3 ∊ O(m)

Is the total size *O*(*m*) now?

No: total length of edge labels is quadratic in *m*

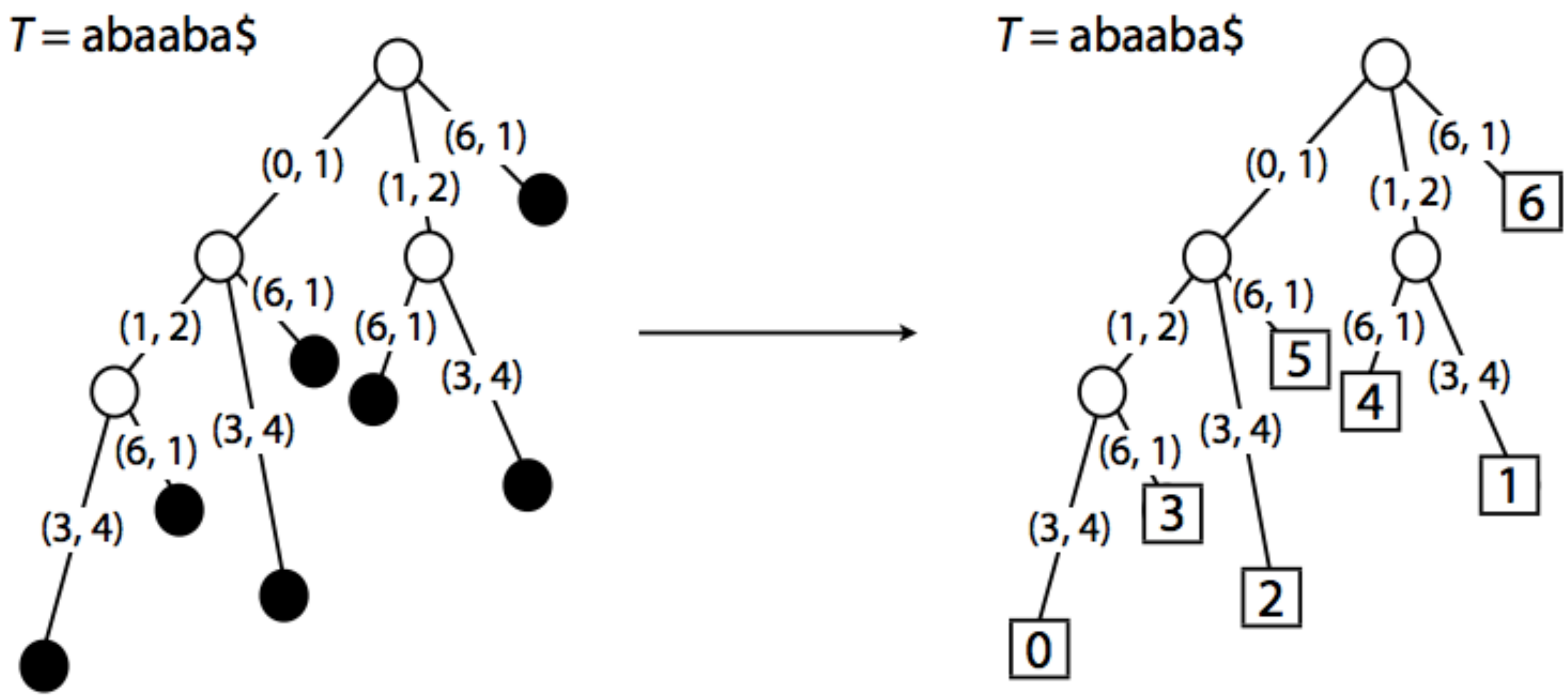
Space complexity

T = abaaba$ Idea 2: Store T itself in addition to the tree. Convert tree's edge labels to (offset, length) pairs with respect to T.

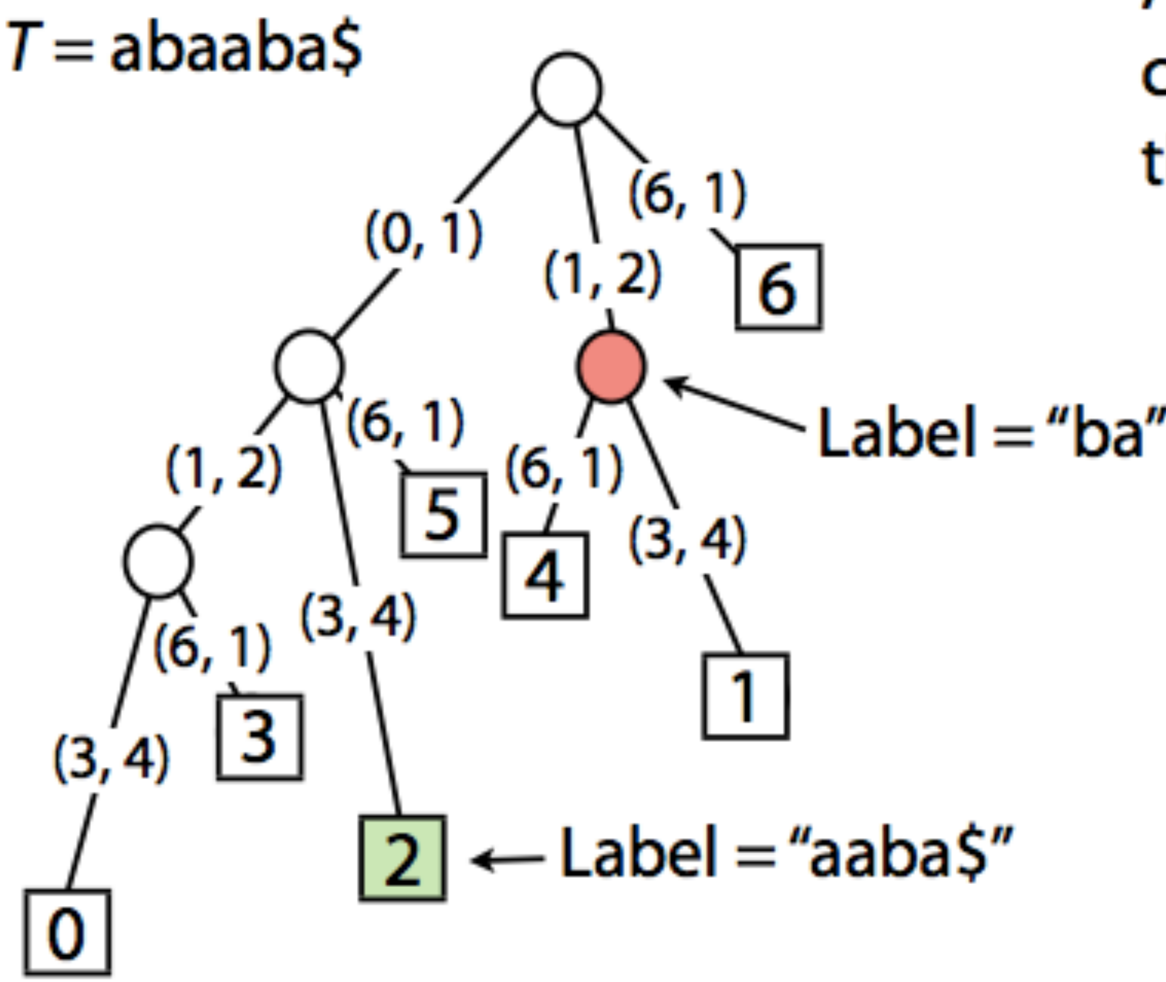


Space required for suffix tree is now $O(m)$

Add starting location/offset at each leaf node



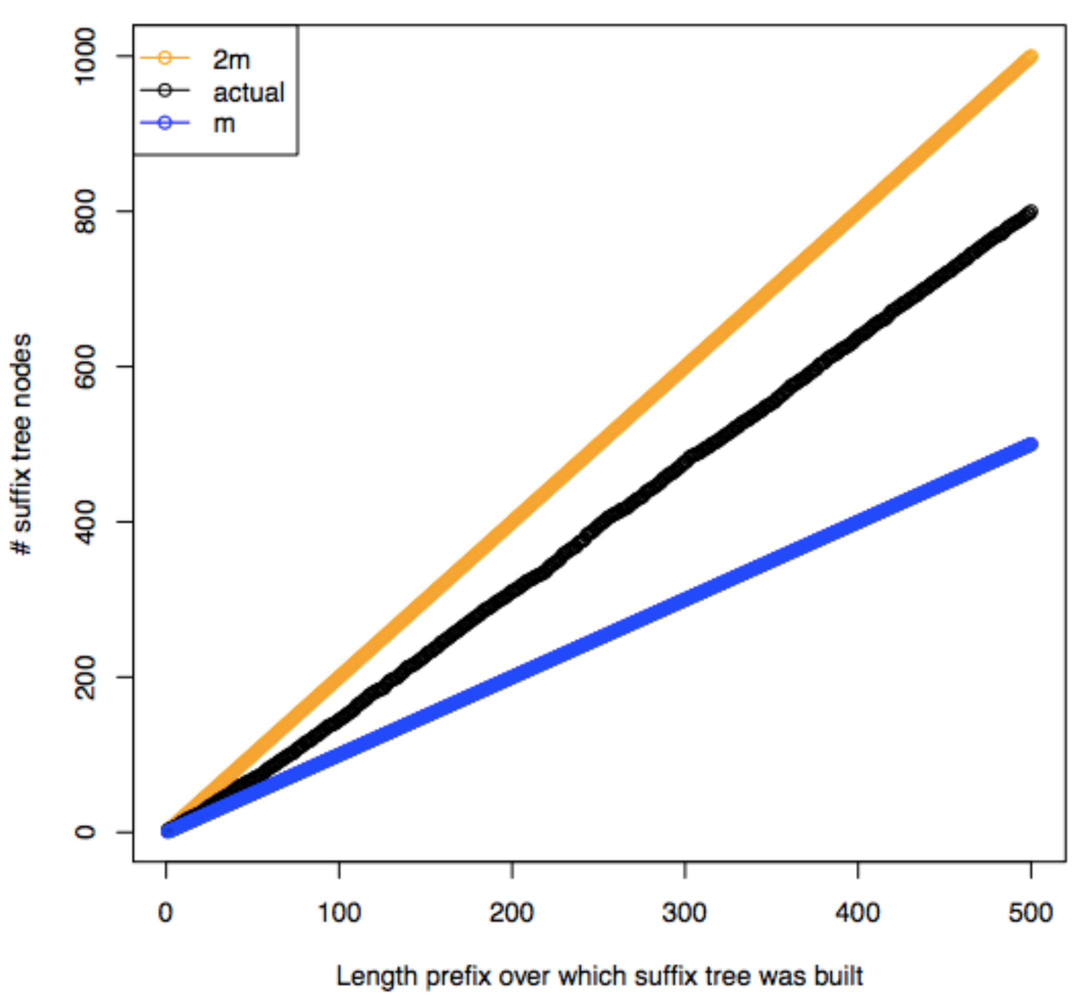
Retrieve substrings



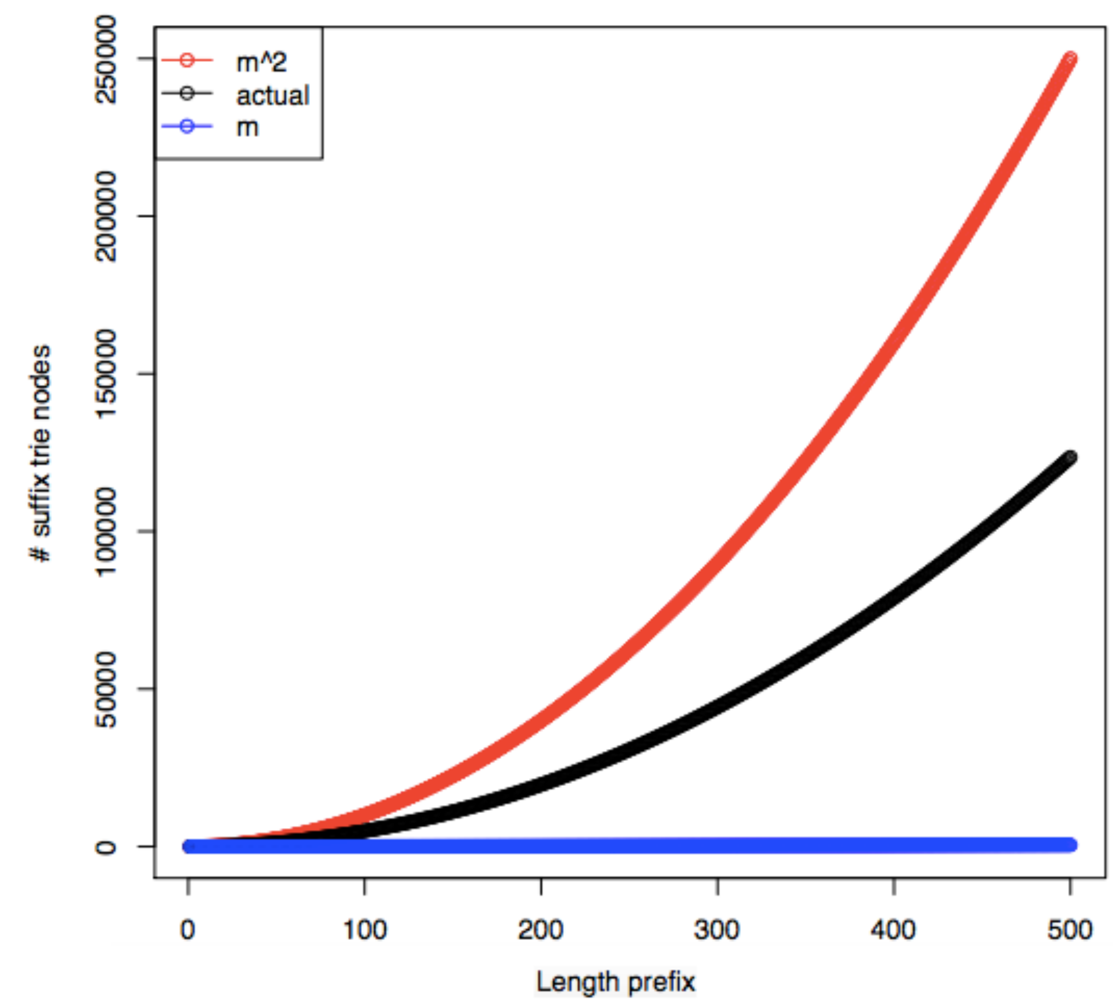
Again, each node's *label* equals the concatenated edge labels from the root to the node. These aren't stored explicitly.

Actual growth: comparison

Trees built using the first 500 prefixes of the lambda phage virus genome



suffix tree



keyword tree

Summary

- Keyword and suffix trees are used to find patterns in a text
- Keyword trees:
  - Build keyword tree of patterns, and thread text through it
  - Usage: checking a set of patterns within various texts
- Suffix trees:
  - Build suffix tree of text, and thread patterns through it
  - Usage: checking various patterns in the same text