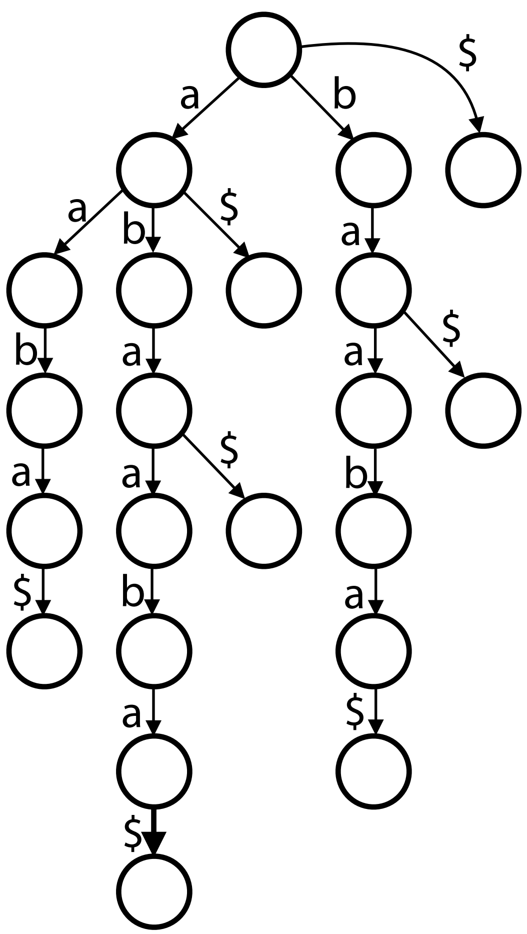
Suffix Trees: building

Ben Langmead

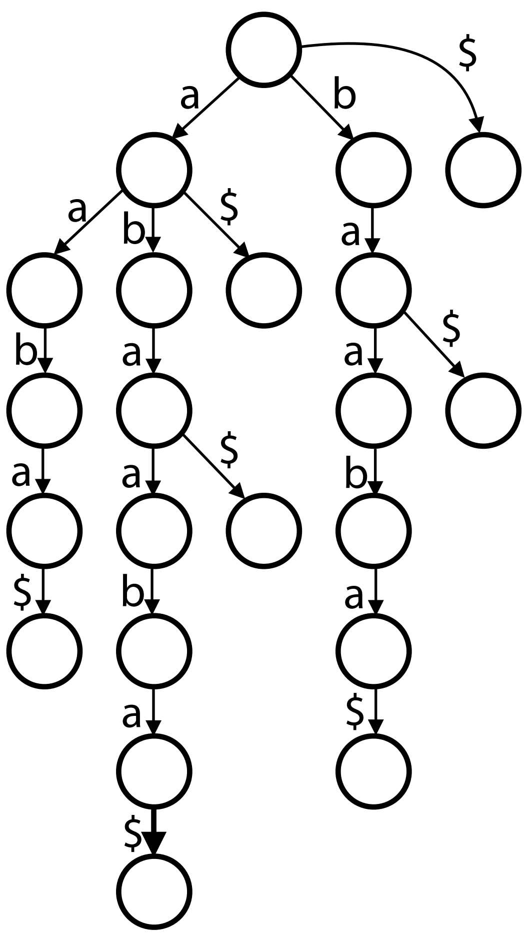
Suffix tree: building

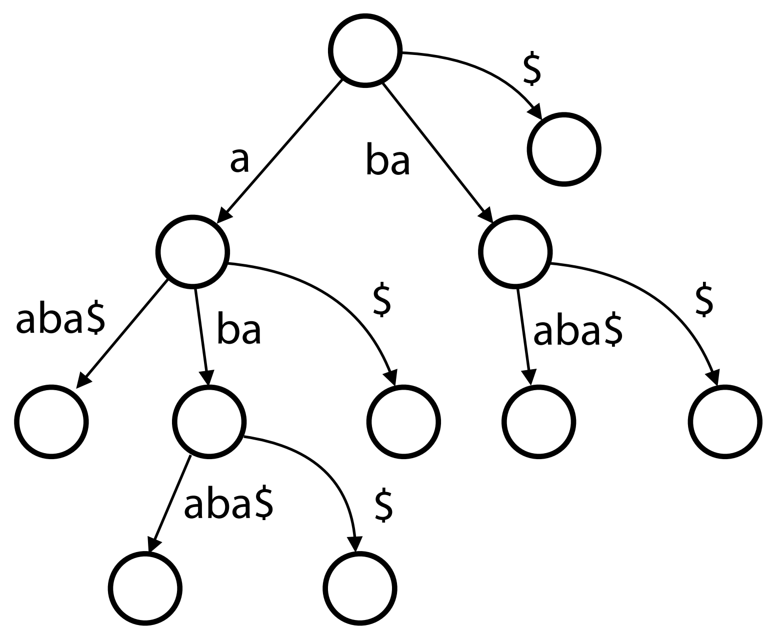
Method 1: build suffix trie



Suffix tree: building

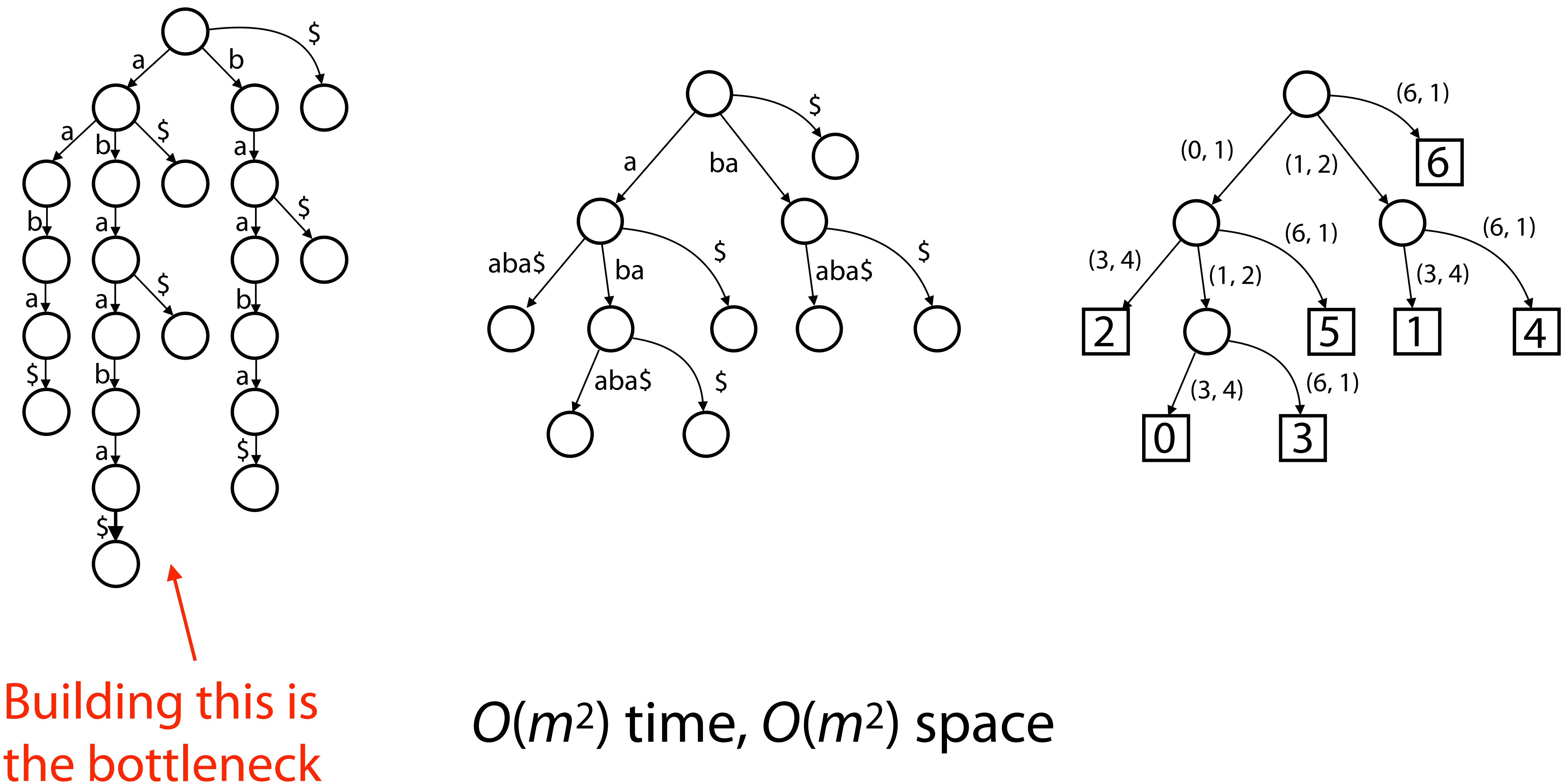
<u>Method 1</u>: build suffix trie, coalesce non-branching paths





Suffix tree: building

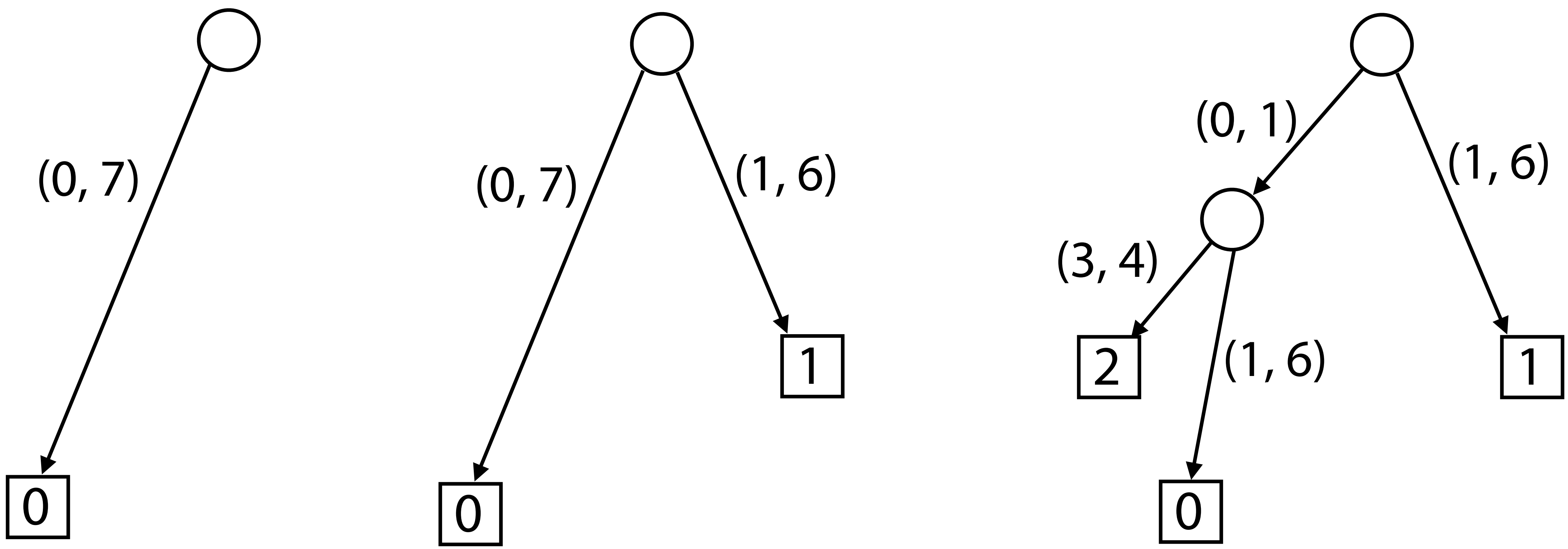
Method 1: build suffix trie, coalesce non-branching paths, label



$O(m^2)$ time, $O(m^2)$ space

Suffix tree: building

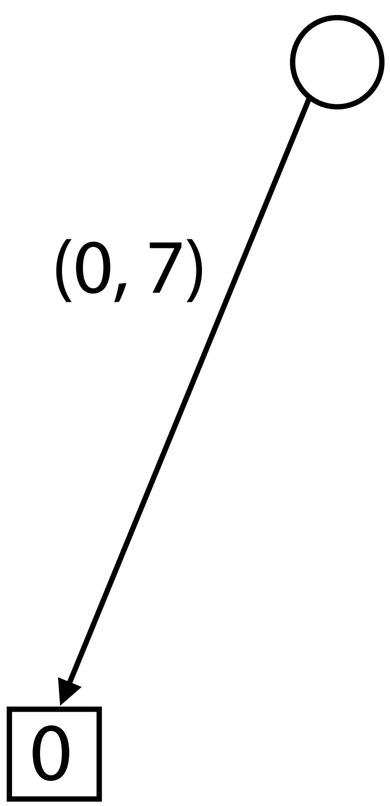
Instead of starting with a big trie and making it smaller, we can start from scratch and "grow" the tree



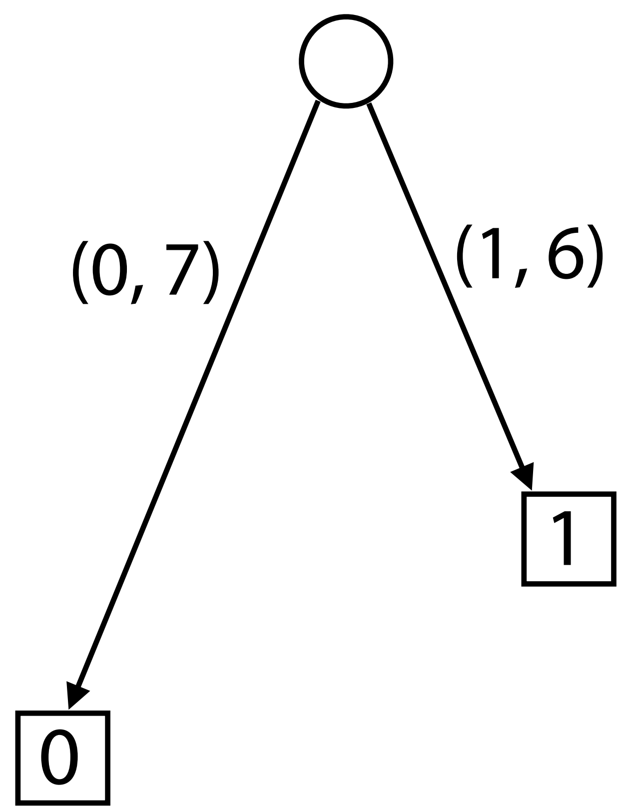
At no point do we have something larger than the final tree, so it must be O(*m*) at all points

Suffix tree: building

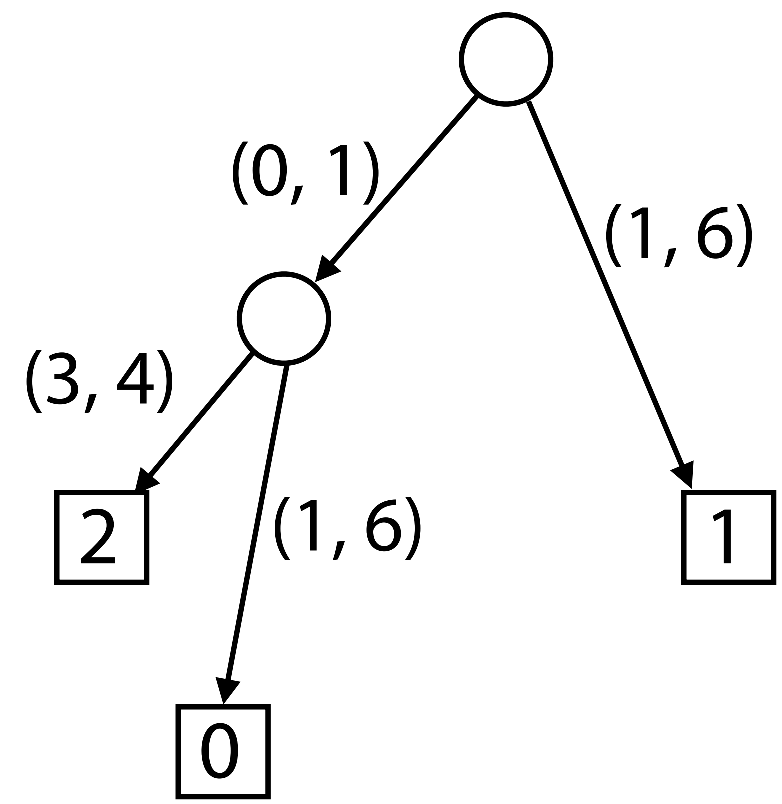
Build single-edge tree with longest suffix



Add 2nd-longest



Add 3rd-longest



etc

...

Suffix tree: building

Few steps of "method 2"

$T = abaaba\$$

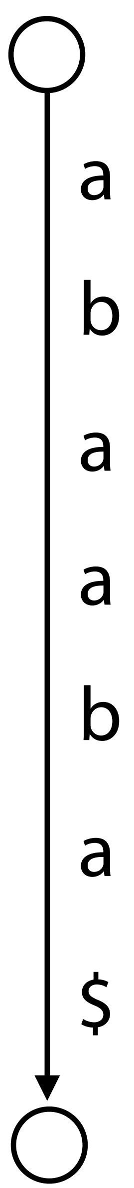
Just longest	Longest + 2nd	+ 3rd

Suffix tree: building

Few steps of "method 2"

T = abaaba$

Just longest	Longest + 2nd	+ 3rd

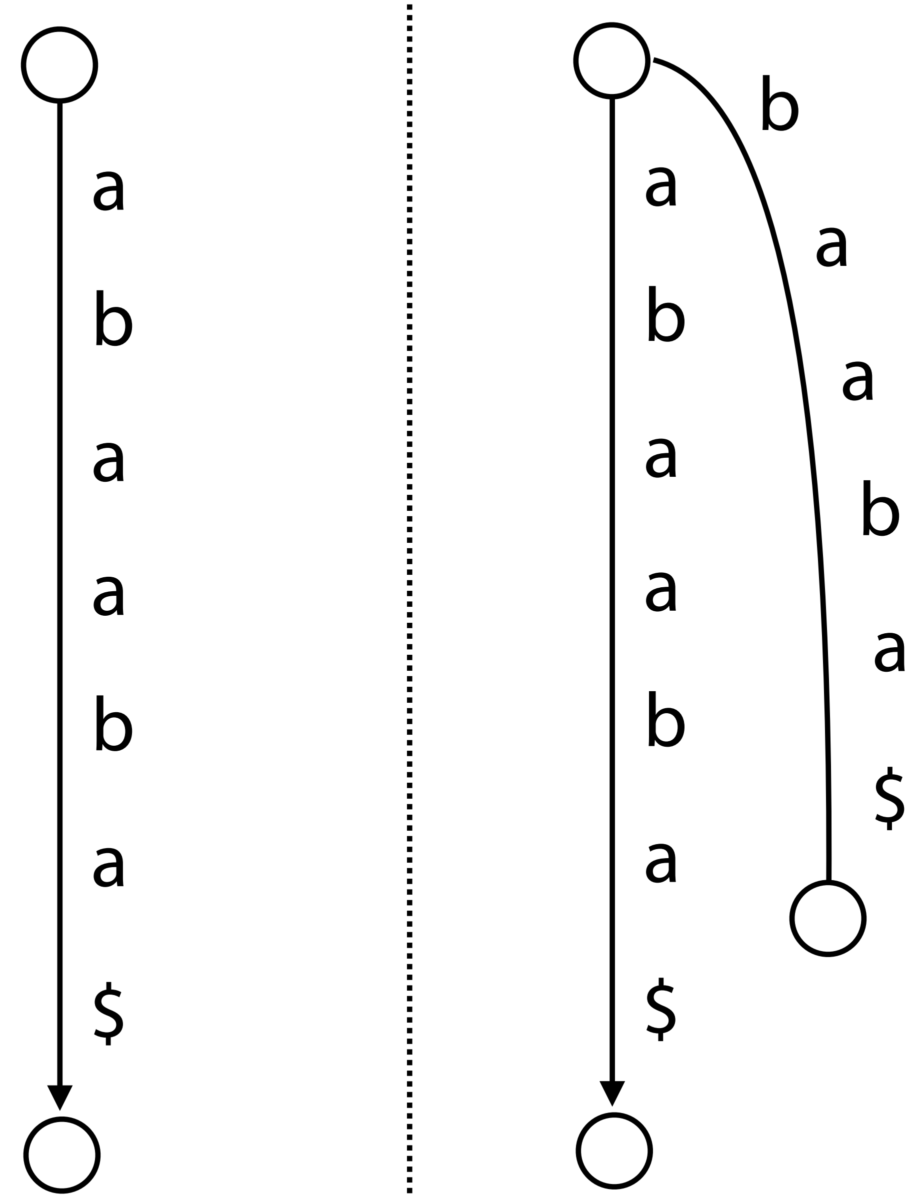


Suffix tree: building

Few steps of "method 2"

T = abaaba$

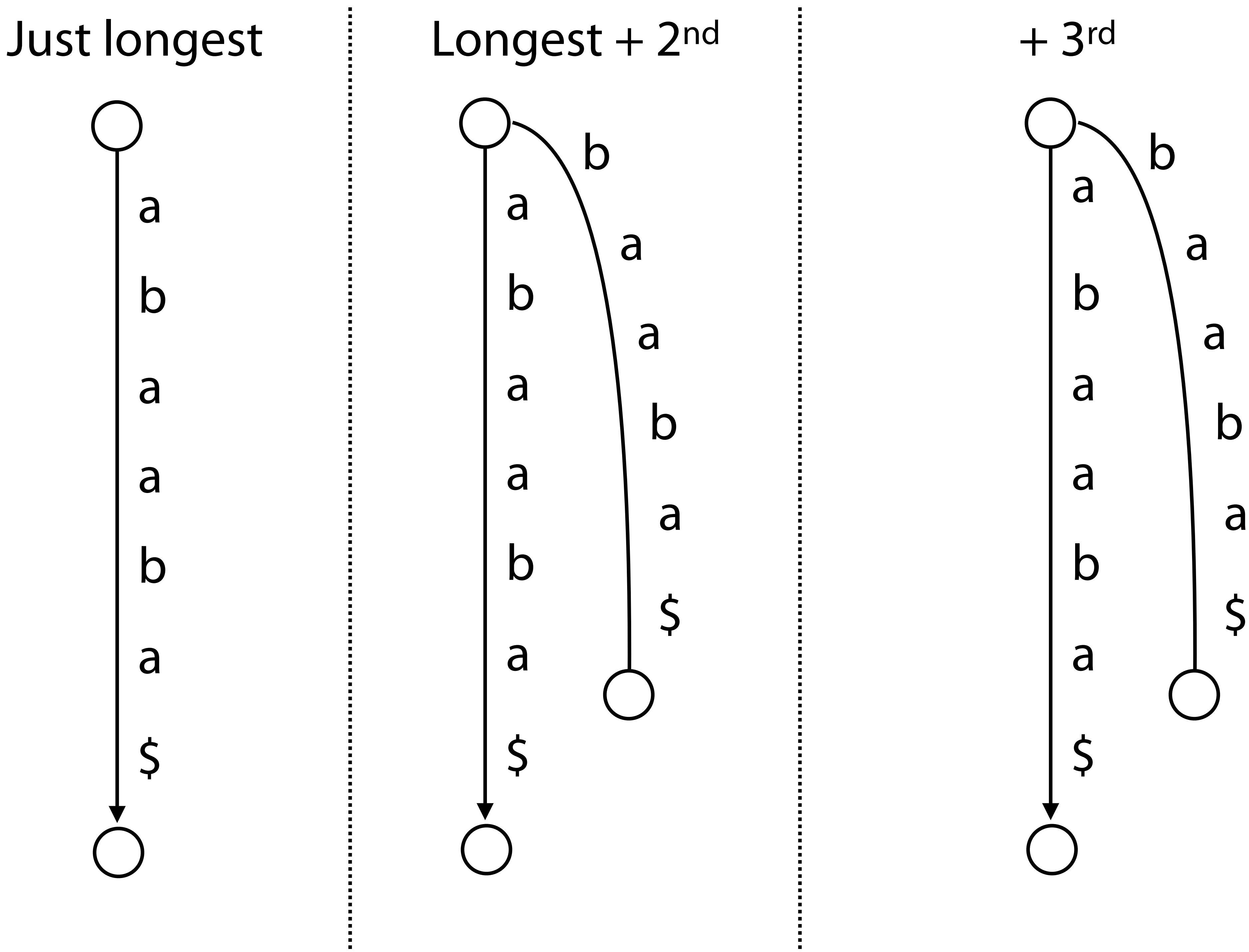
Just longest | Longest + 2nd | + 3rd



Suffix tree: building

Few steps of "method 2"

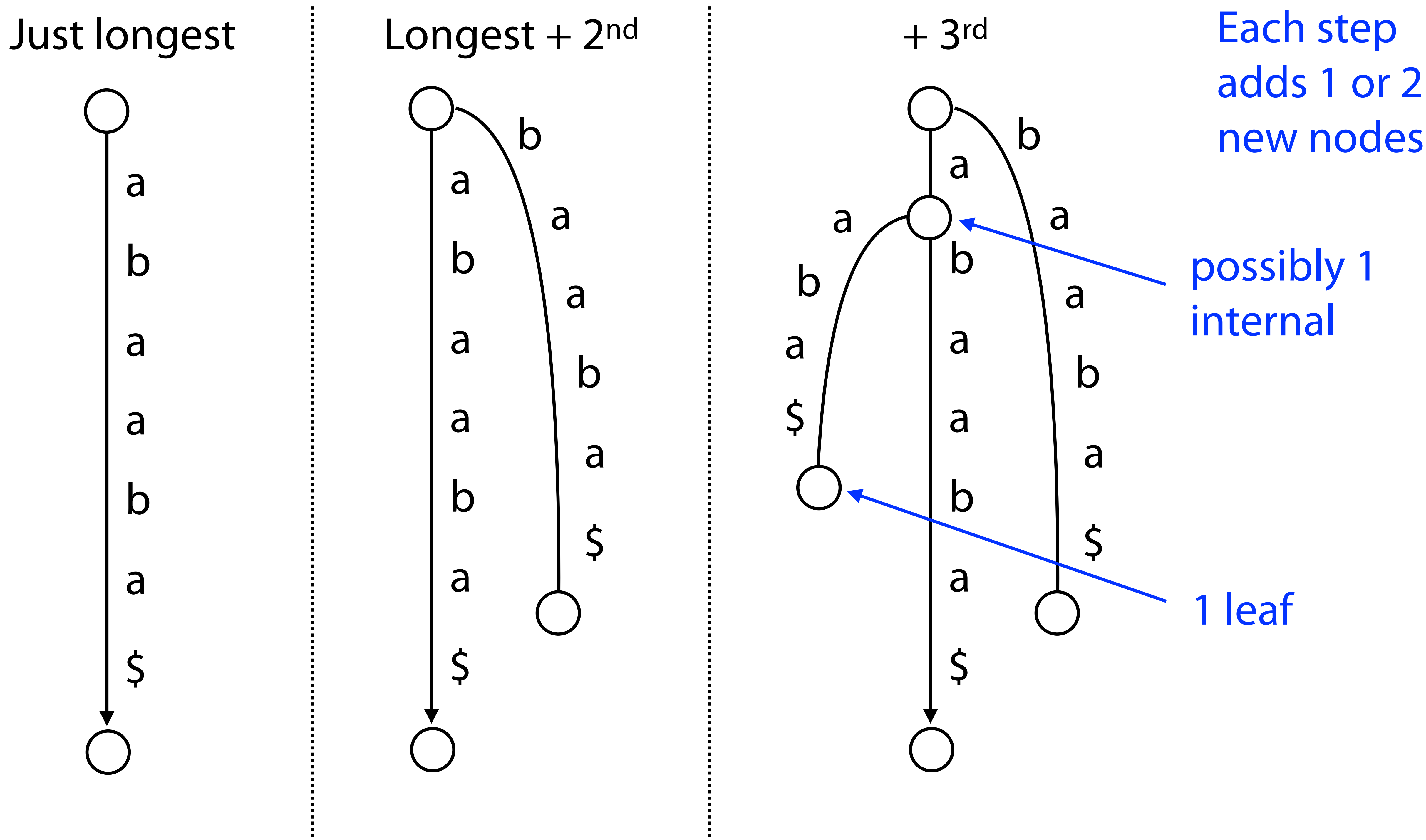
T = abaaba$



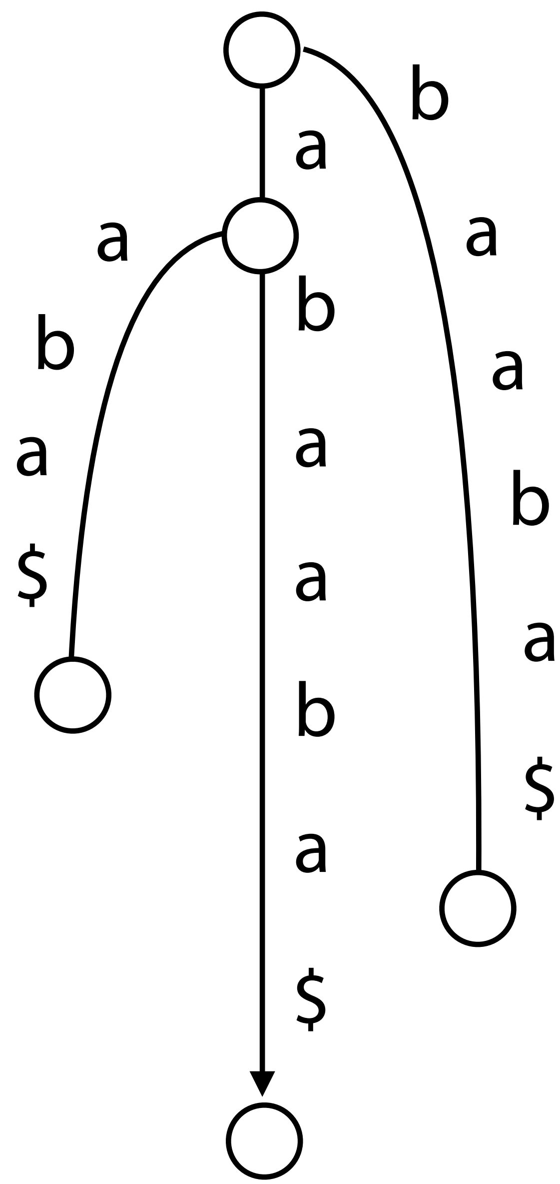
Suffix tree: building

Few steps of "method 2"

T = abaaba$



Suffix tree: building



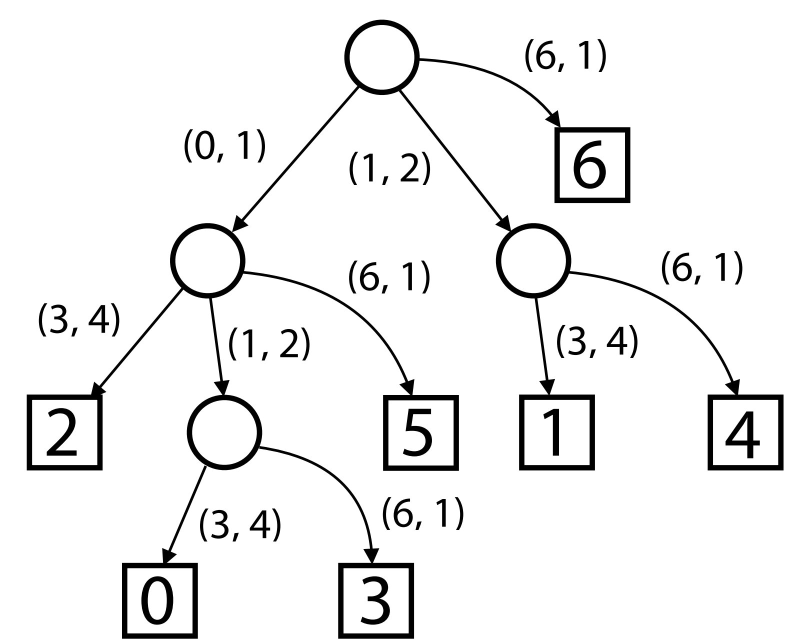
Though tree only grows by 1 or 2 nodes in each step, work is dominated by the need to "walk down" for each suffix

$O(m^2)$ time overall

Suffix tree: building

Method 2: build single-edge tree representing longest suffix, augment to include the 2nd-longest, augment to include 3rd-longest, etc

$O(m^2)$ time, $O(m)$ space



Ukkonen's algorithm

Algorithmica (1995) 14: 249–260

Algorithmica

On-Line Construction of Suffix Trees[1]

E. Ukkonen[2]

Abstract. An on-line algorithm is presented for constructing the suffix tree for a given string in time linear in the length of the string. The new algorithm has the desirable property of processing the string symbol by symbol from left to right. It always has the suffix tree for the scanned part of the string ready. The method is developed as a linear-time version of a very simple algorithm for (quadratic size) suffix *tries*. Regardless of its quadratic worst case this latter algorithm can be a good practical method when the string is not too long. Another variation of this method is shown to give, in a natural way, the well-known algorithms for constructing suffix automata (DAWGs).

Key Words. Linear-time algorithm, Suffix tree, Suffix trie, Suffix automaton, DAWG.

Figure from: Ukkonen, E. (1995). "On-line construction of suffix trees" Algorithmica. 14 (3): 249–260. doi:10.1007/BF01206331

Ukkonen's algorithm

Too complex to detail here, but you should know:

- $O(m)$ time and space
- It is the most widely used, though not the only algorithm with $O(m)$ time & space
- Also starts from scratch and builds up to full tree
- Uses and outputs "suffix links," to be discussed later

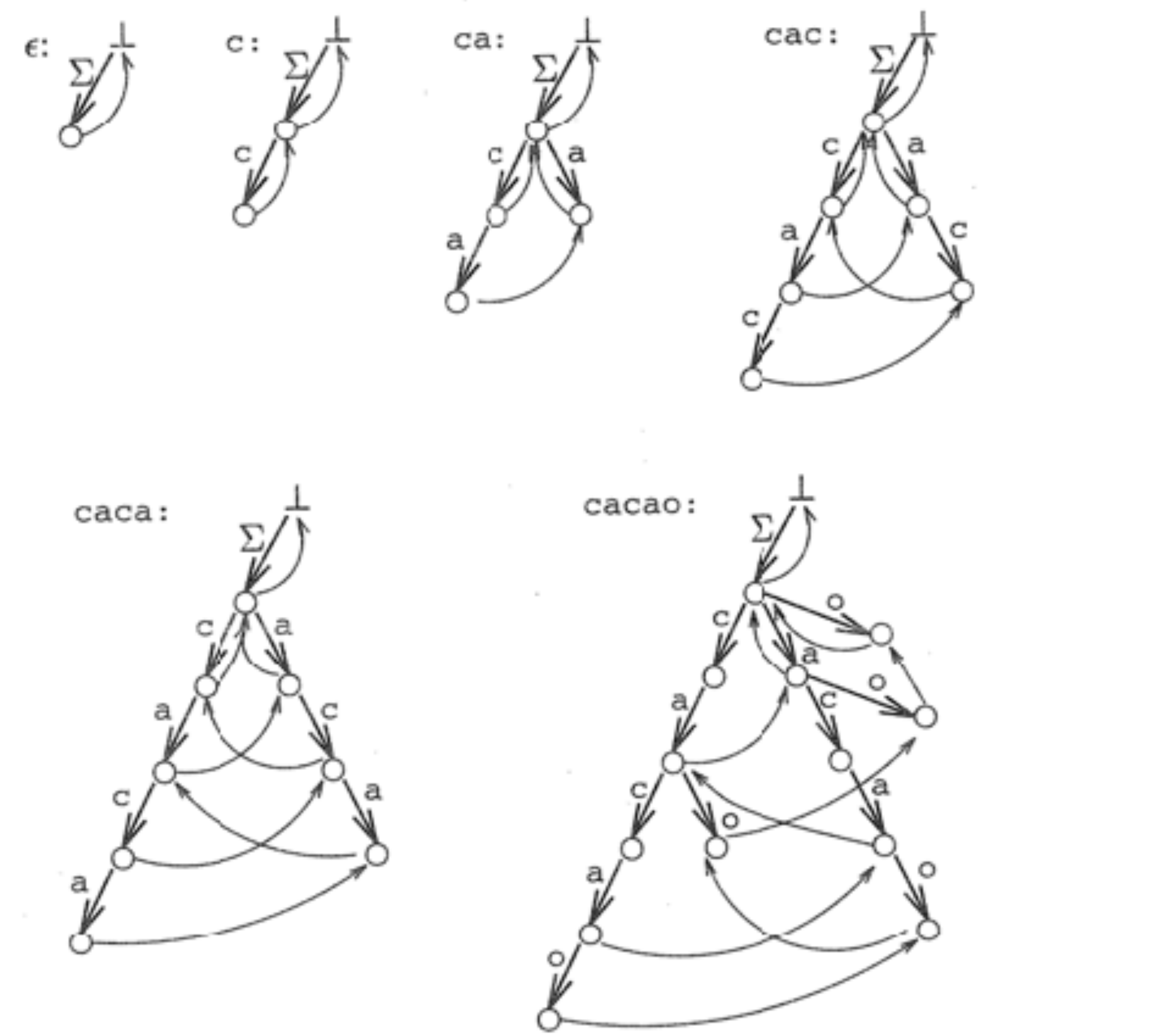


Fig. 1. Construction of *STrie*(cacao): state transitions shown in bold arrows, failure transitions in thin arrows. *Note:* Only the last two layers of suffix links shown explicitly.

Figure from: Ukkonen, E. (1995). "On-line construction of suffix trees" Algorithmica. 14 (3): 249–260. doi:10.1007/BF01206331

Building suffix trees: summary

Good methods exist that build the suffix tree incrementally from scratch, eventually reaching the $O(m)$-size tree

Our "Method 2" does this quite simply, but needs $O(m^2)$ time

Ukkonen's algorithm is the most widely used