The magic of suffix trees

Lecture 3

Pattern matching problem - continued

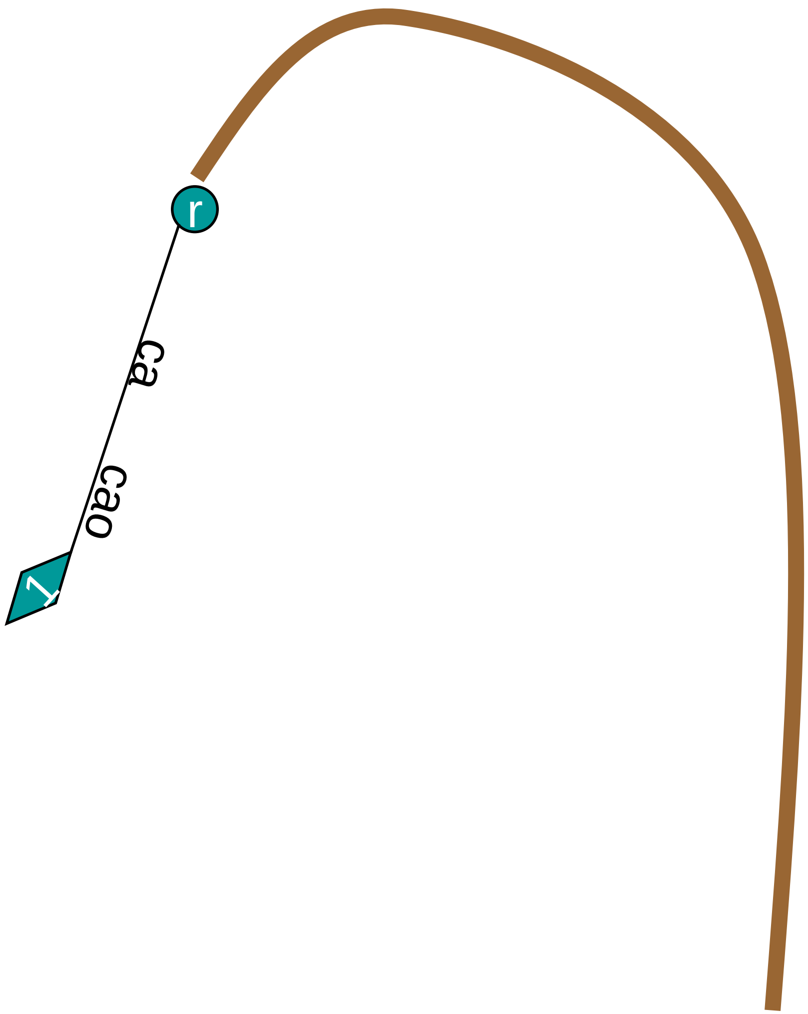
- KMP is an optimal linear-time algorithm for the patter-matching problem
- It works in a situation when the pattern is fixed and the text is streaming – the text is not known before the search starts
- The different setting:
  - text *T* is known first and it is kept fixed for some time
  - the new patterns are constantly arriving
  - the search for each pattern should be done as quick as possible

Suffix trees

- Suffix tree of *T* exposes the internal structure of this text
- Assuming that the text is re-written in a form of the suffix tree, the pattern matching problem can be performed in time O(M+k), where M is the length of a pattern, and k is the number of occurrences. The search time ***does not depend*** on the length of *T*
- In addition, suffix trees provide optimal (linear-time) solutions to numerous complex problems other than the pattern matching problem

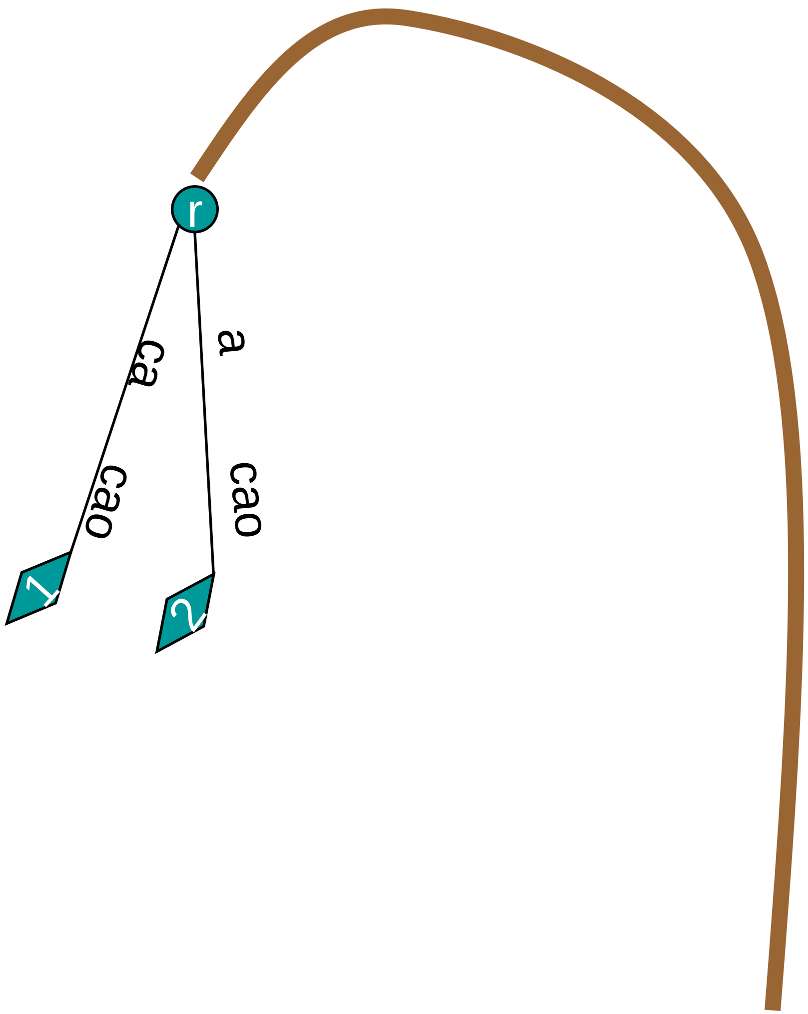
Tree branch with suffixes

T=*cacao*



Tree branch with suffixes

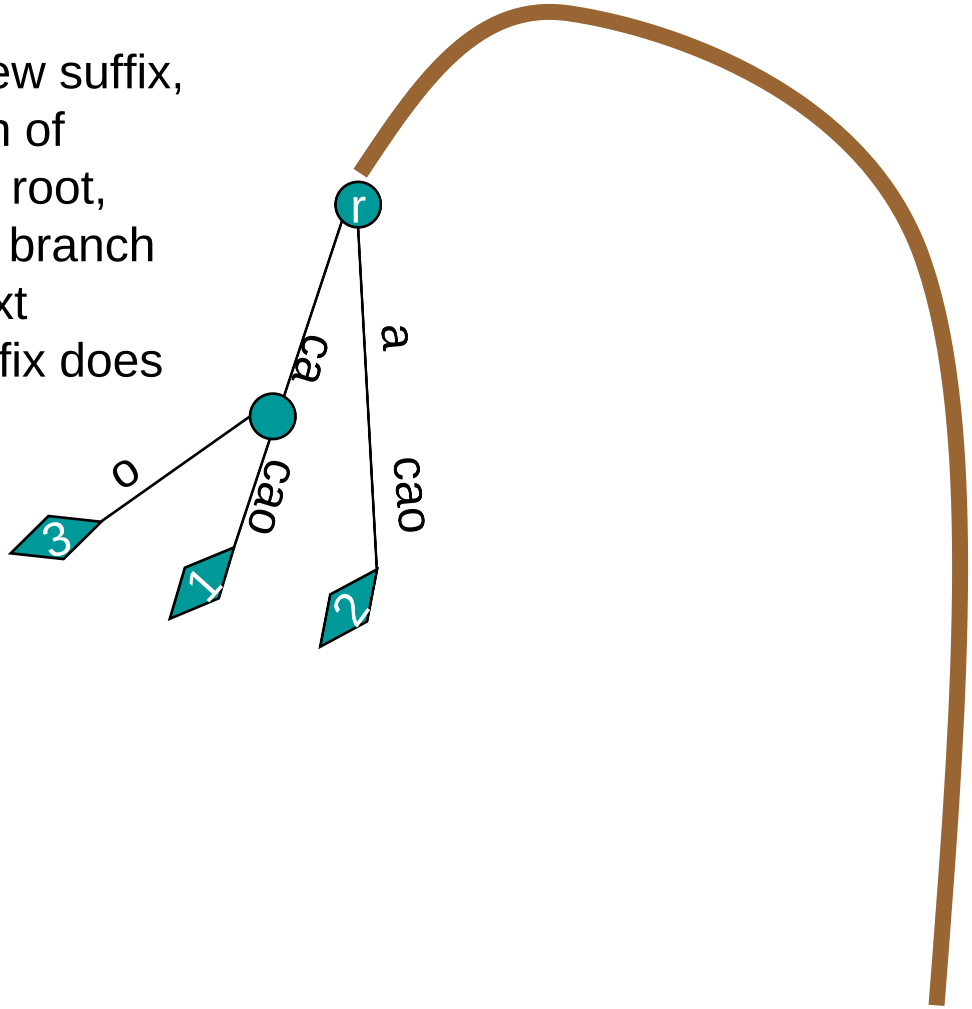
T=*c<u>acao</u>*



Tree branch with suffixes

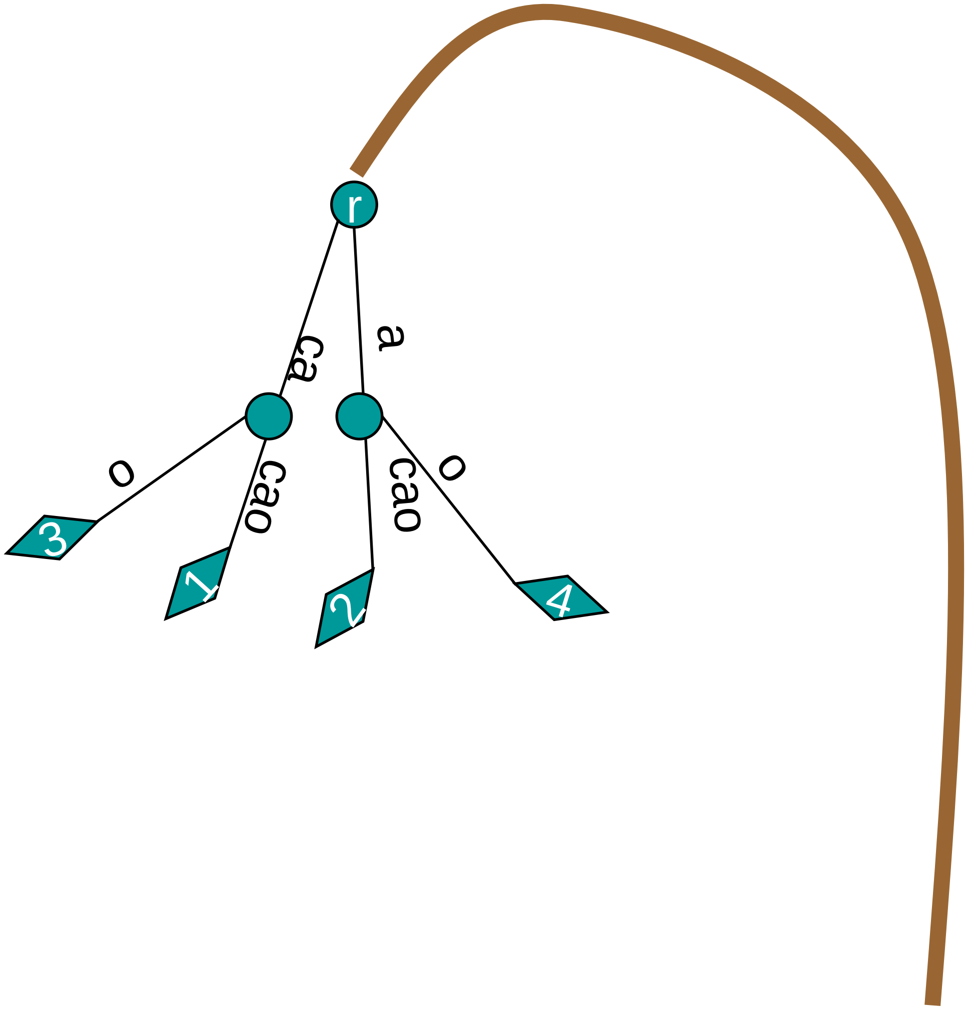
T=*ca<u>cao</u>*

While adding a new suffix, we follow the path of matches from the root, and create a new branch only when the next character of a suffix does not match



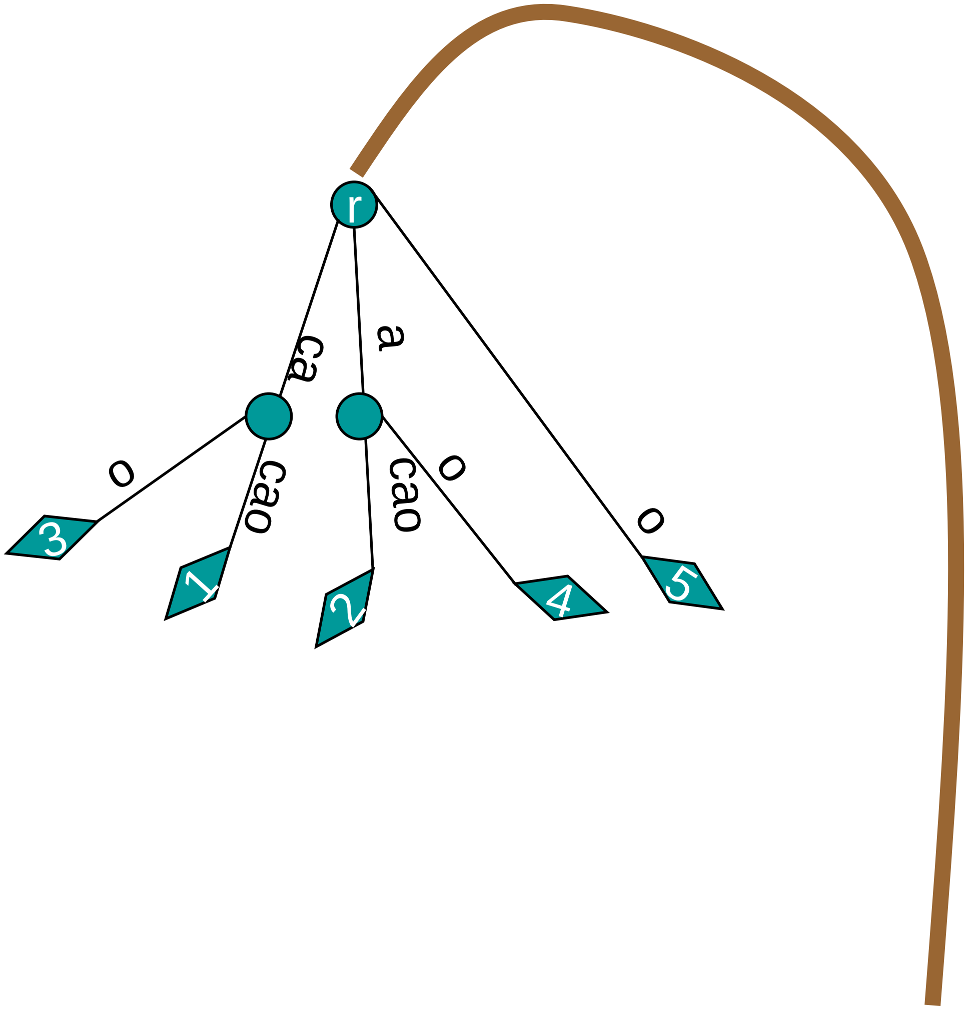
Tree branch with suffixes

T=*cacao*



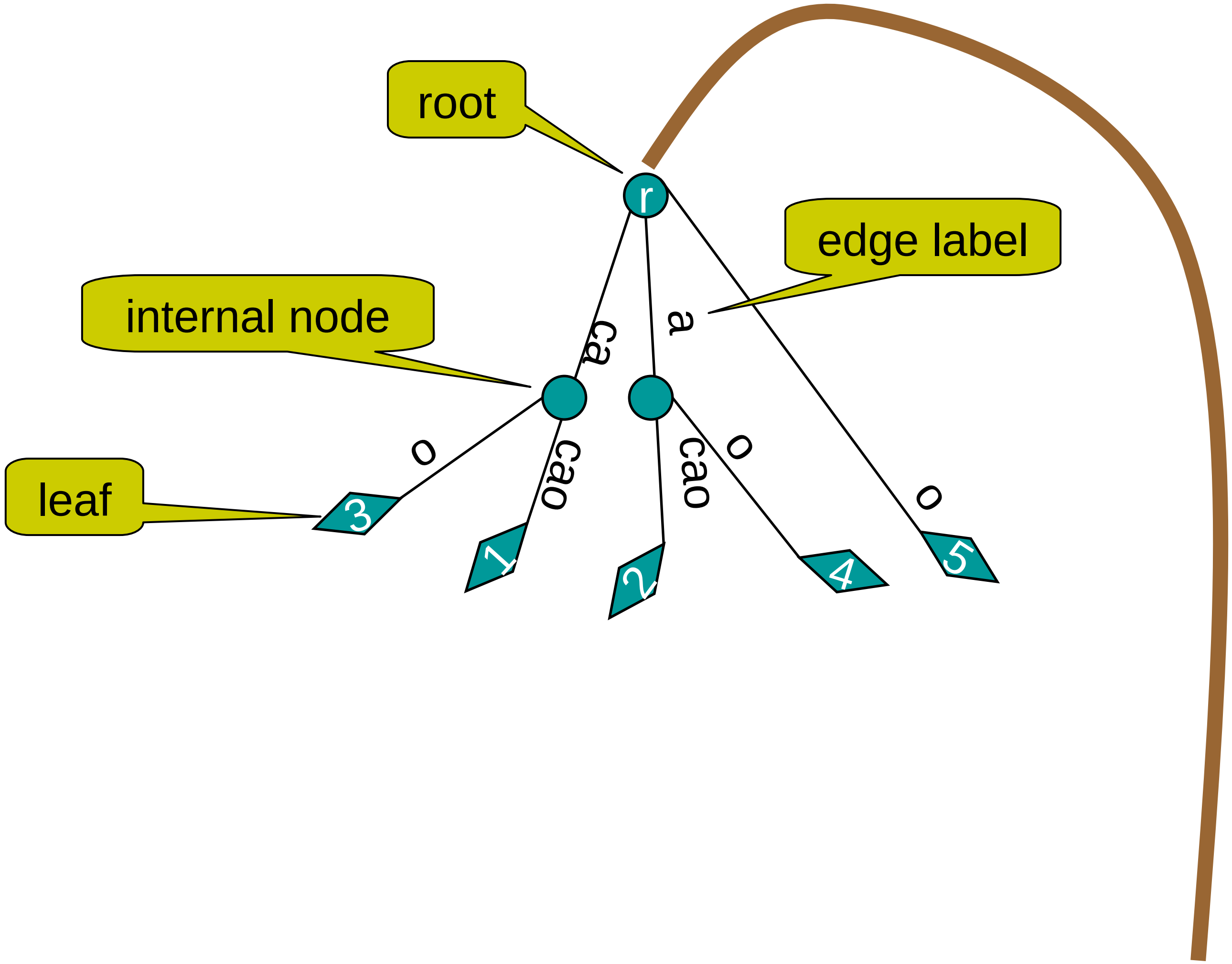
Tree branch with suffixes

T=*cacao*



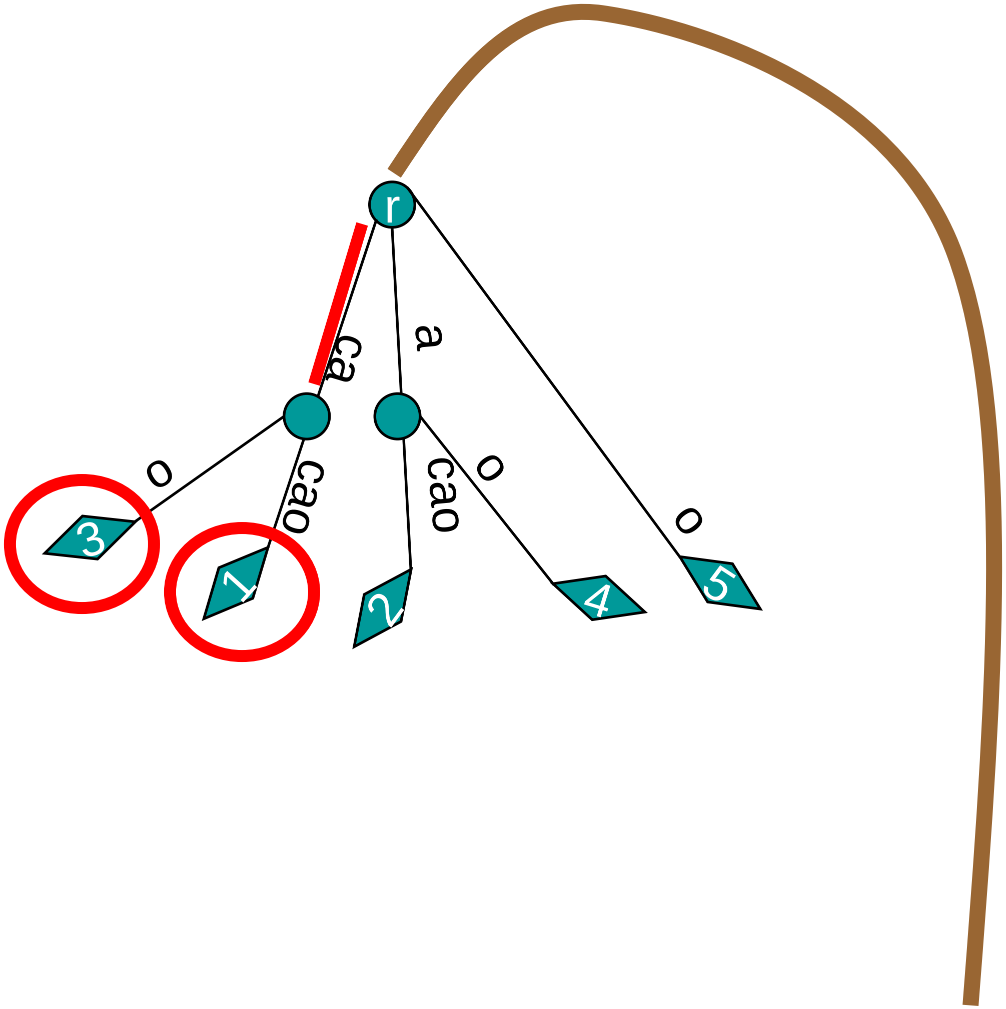
Suffix tree terminology

T=*cacao*



Search for pattern *ca*

T=*cacao*



Suffix tree - definition

- *A suffix tree* for string *T* (of length *N)* is a rooted directed tree with the following properties*:*
  - *N* leaves, numbered 1 to *N*.
  - Each internal node has at least two children.
  - No two edges out of a node have edge-labels beginning with the same character.
  - For any leaf *i*, the concatenation of the edge-labels on the path from root to leaf *i* spells out the suffix *T*[*i*..*N*] of *T.*

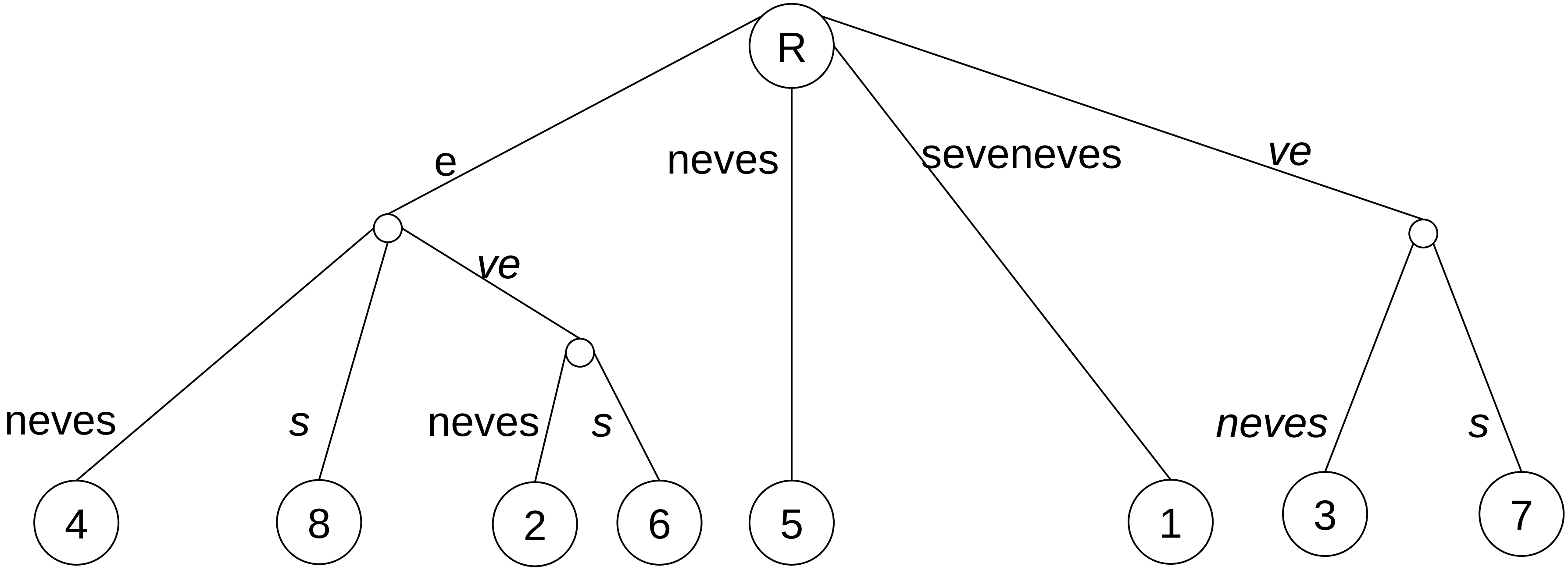
Full-text indexing

- All different substrings of *T* can be found in the suffix tree following the path from the root
- Build a tree for *T=bananas*

Another suffix tree

s e v e n e v e s

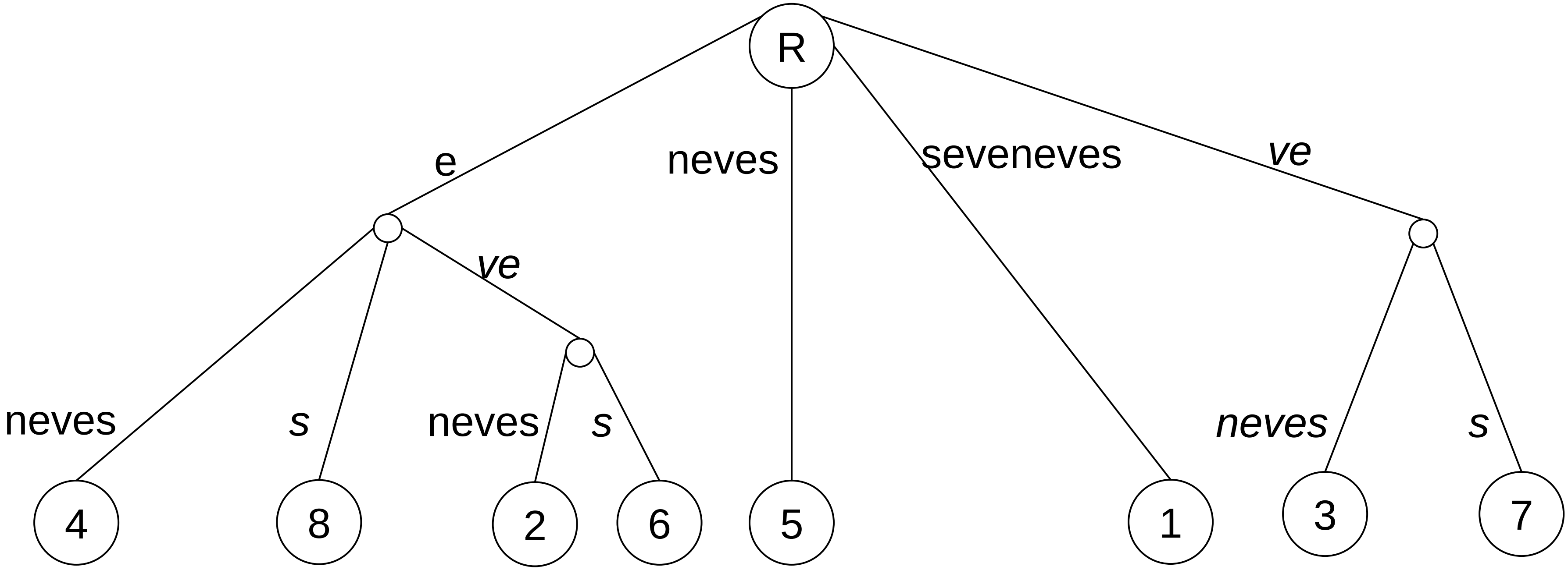
1 2 3 4 5 6 7 8 9



Another suffix tree

s e v e n e v e s

1 2 3 4 5 6 7 8 9

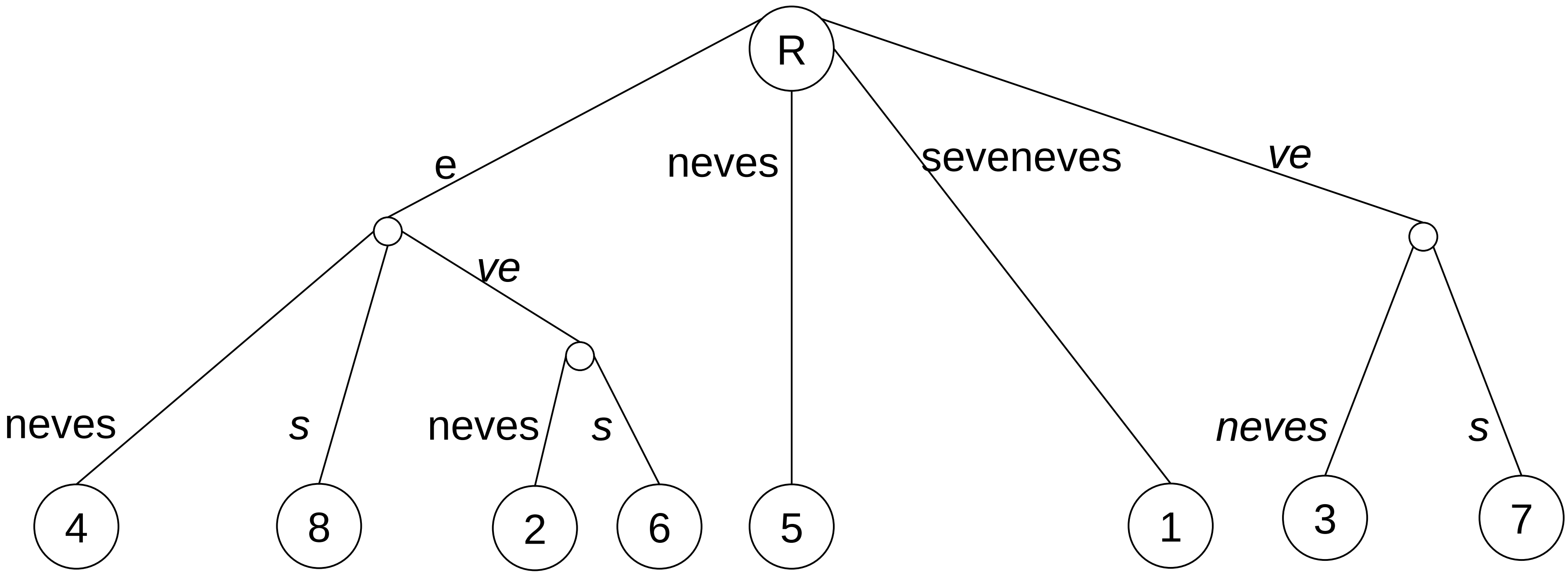


What suffix is missing?

Another suffix tree

s e v e n e v e s

1 2 3 4 5 6 7 8 9



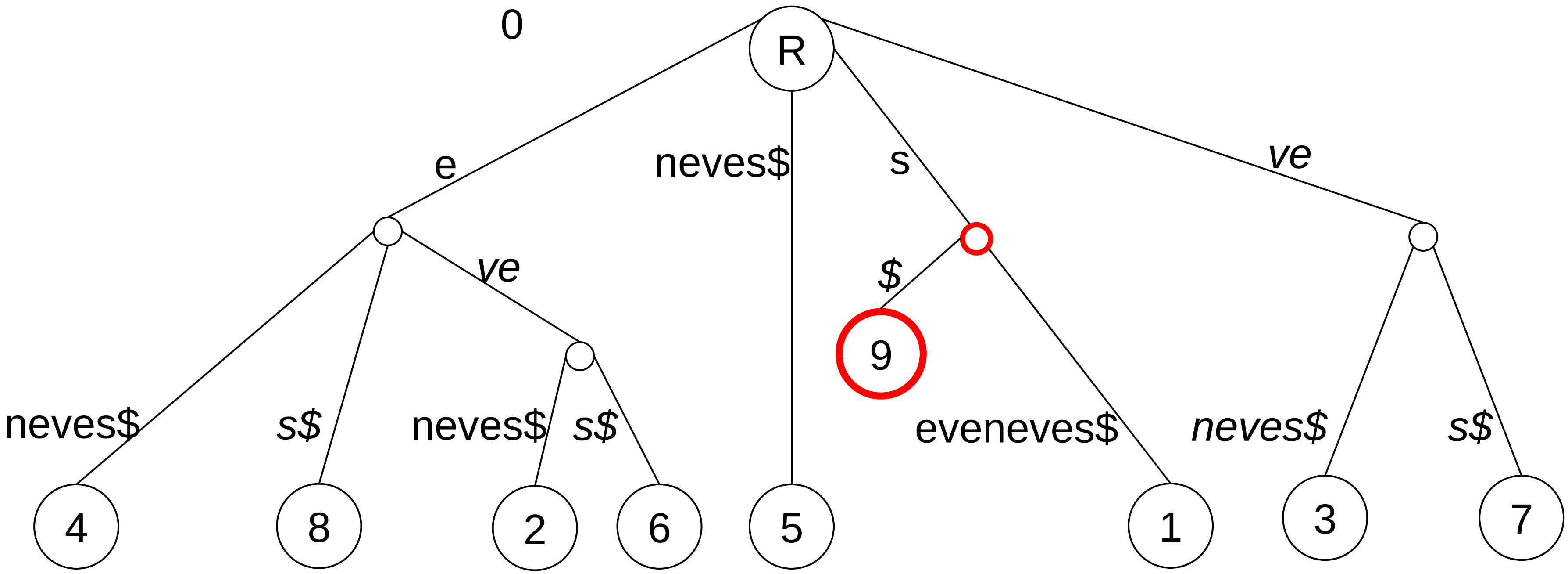
Where is the leaf for *T*[9…9]=*s*?

What if we search for pattern *P*=*s*?

Proper suffix tree

s e v e n e v e s $

1 2 3 4 5 6 7 8 9 10

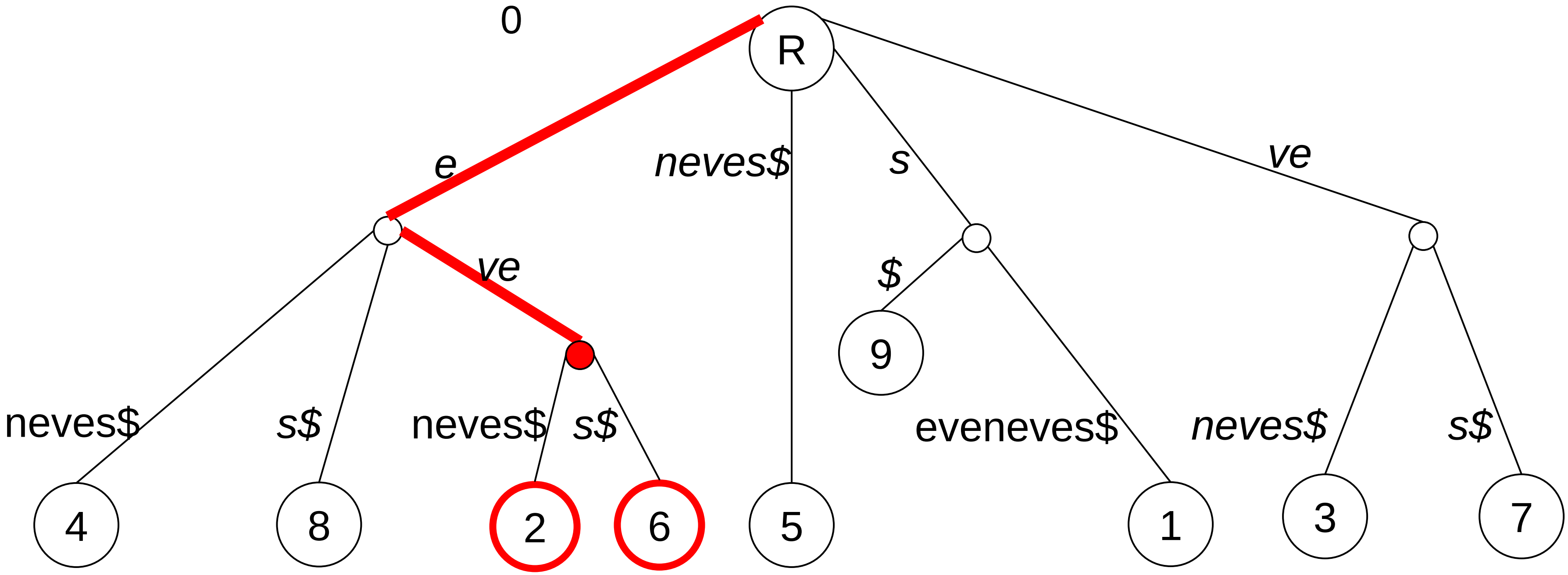


The *sentinel* $ does not occur in T

Search for *P=eve*

s e v e n e v e s $

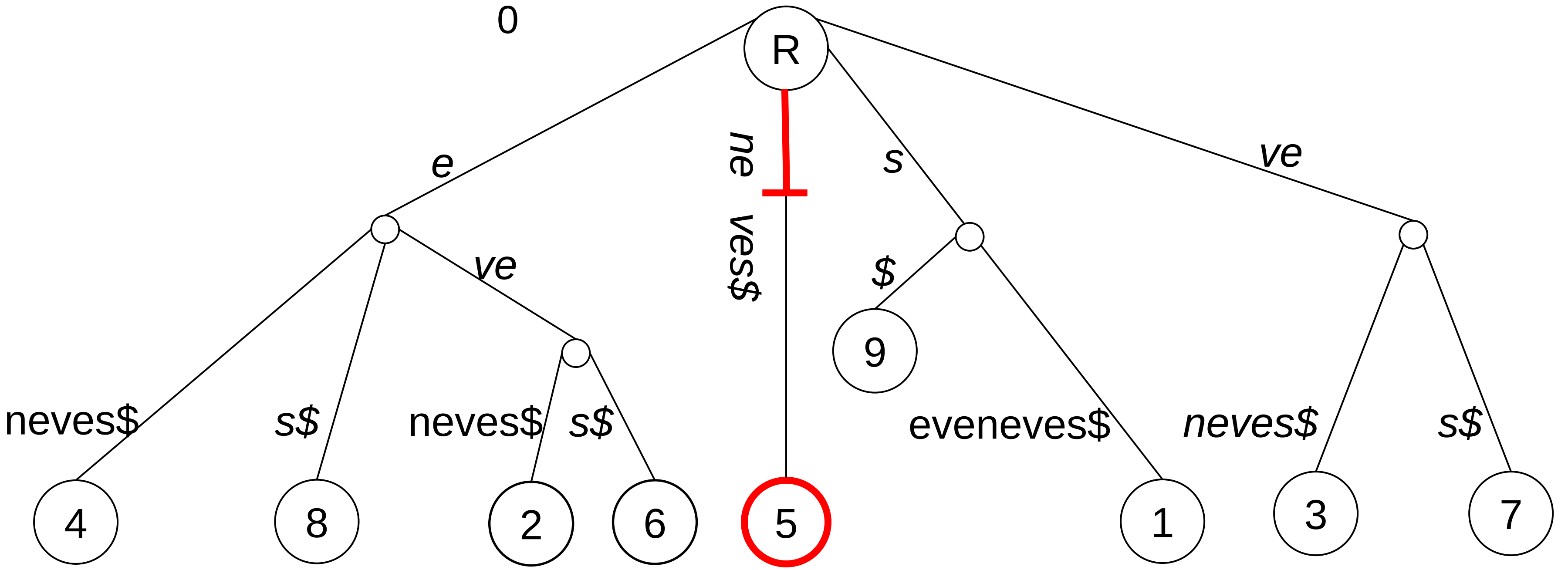
1 2 3 4 5 6 7 8 9 10



Search in time O(*M*+*k*)

Search for *P=ne*

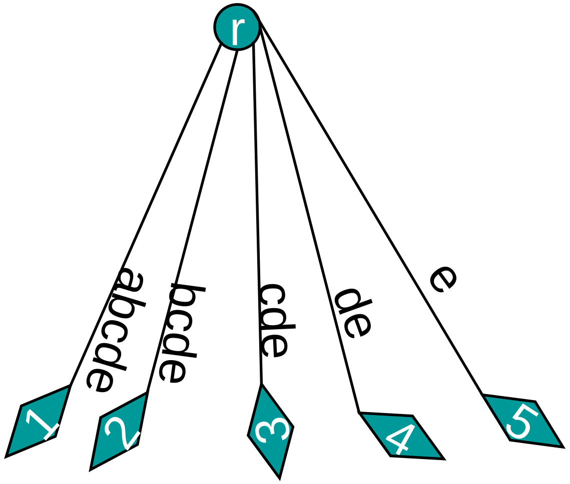
s	*e*	*v*	*e*	*n*	*e*	*v*	*e*	*s*	*$*
1	2	3	4	5	6	7	8	9	10



Search in time O(*M*+*k*)

Space

T=*abcde*



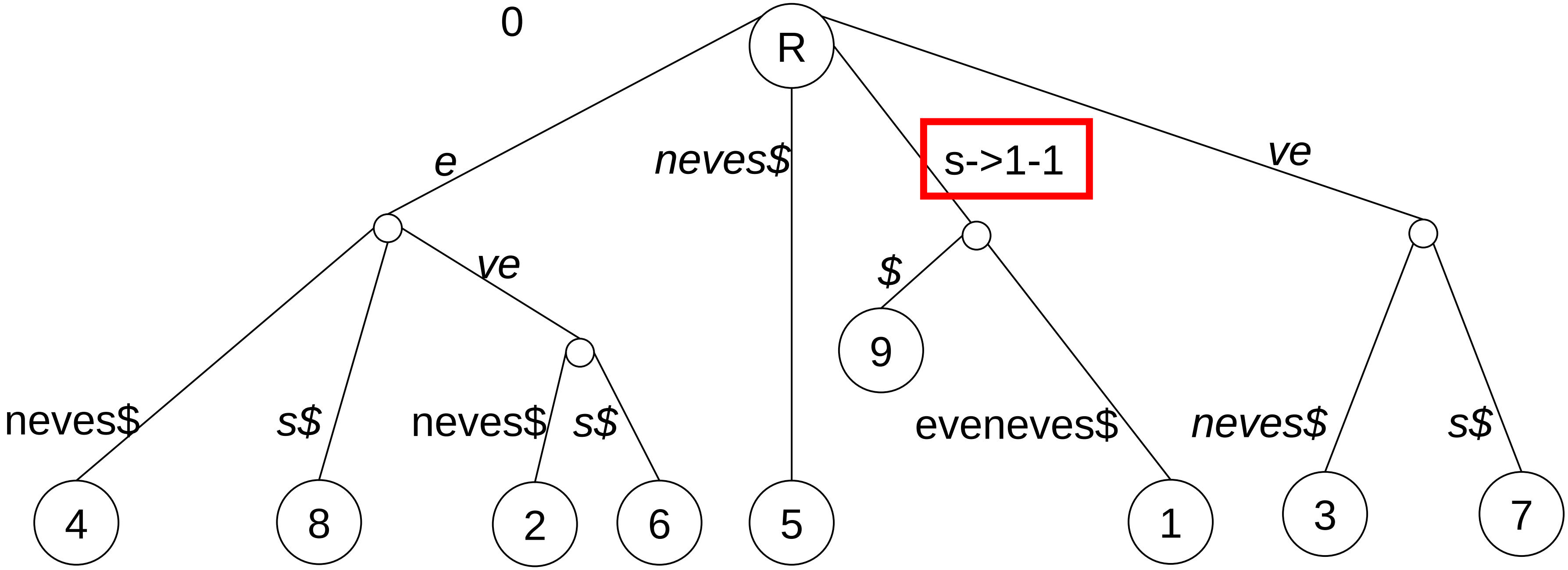
This tree occupies quadratic space

$1+2+3+\ldots.N=O(N^2)$

Trick – re-label the edges

s e v e n e v e s $

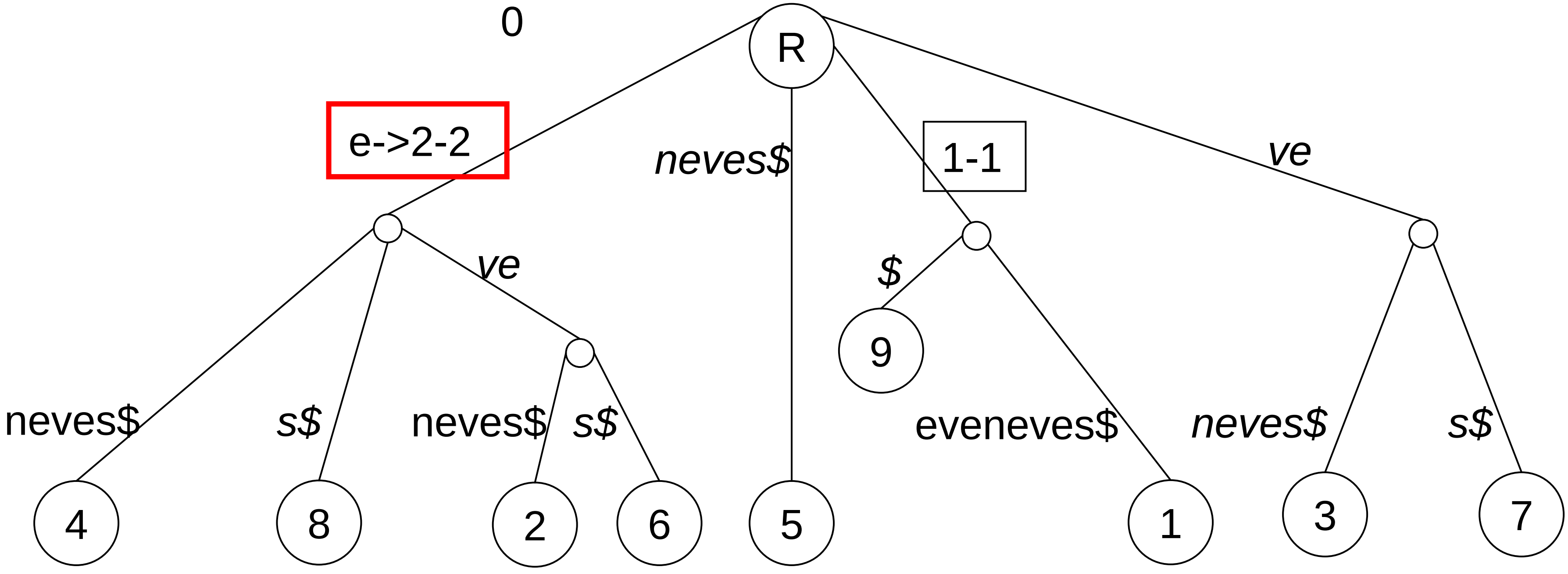
1 2 3 4 5 6 7 8 9 10



Trick – re-label the edges

s e v e n e v e s $

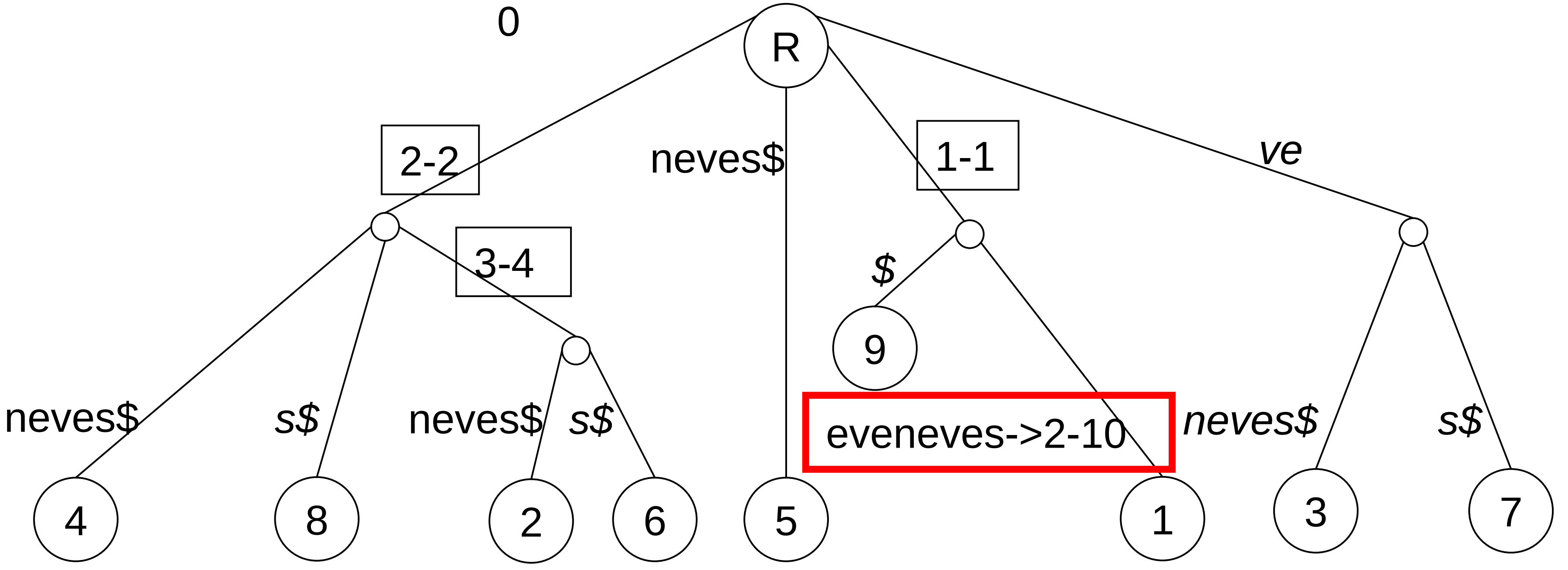
1 2 3 4 5 6 7 8 9 10



Trick – re-label the edges

s e v e n e v e s $

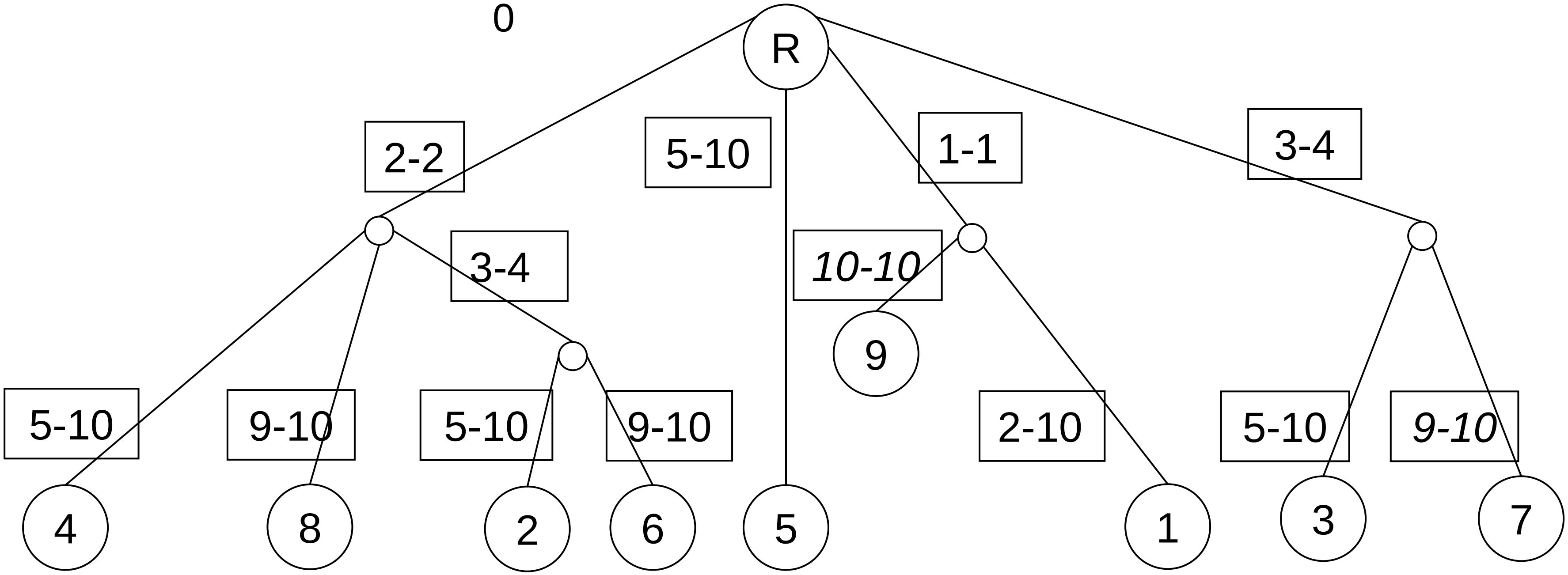
1 2 3 4 5 6 7 8 9 10



Linear space

s e v e n e v e s $

1 2 3 4 5 6 7 8 9 10



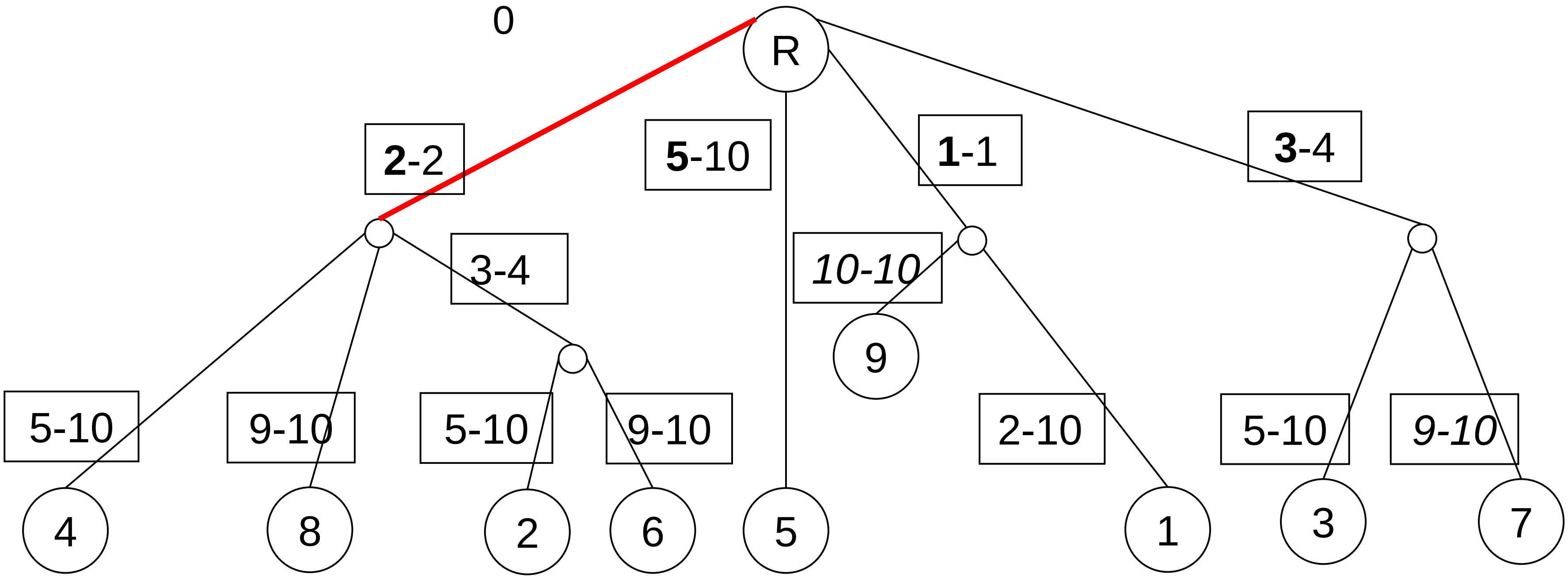
The total number of leaves is *N*, the total number of internal nodes is O(*N*)

With a constant storage space per node – the suffix tree can be stored in linear space

Search

s e v e n e v e s $

1 2 3 4 5 6 7 8 9 10



In order to find an outgoing edge which starts with *e*, we check which of T[2], T[5], T[1] or T[3] is *e*.

The search is as efficient as before, assuming a constant time access to each character of T

Summary of the search

- If we have preprocessed the text *T* into its suffix tree, we can answer a Boolean query of an occurrence of a pattern of length *M* by performing only *M* steps, <u>independently</u> of the length of the text *T*
- In order to report all *k* occurrences of a pattern, the traversal of a corresponding subtree is performed in O(*k*) steps

References

- http://marknelson.us/1996/08/01/suffix-trees/
- http://en.wikipedia.org/wiki/Suffix_tree
- http://www.allisons.org/ll/AlgDS/Tree/Suffix/