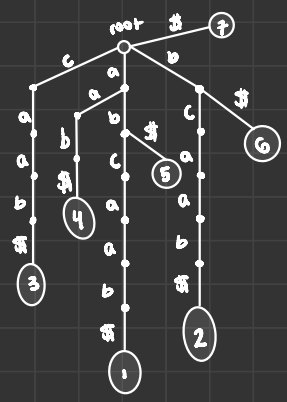
Suffix trees continued

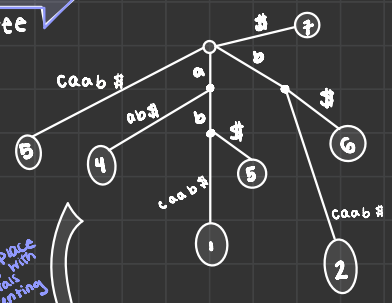
- Takes $O(n^2)$ time to construct a suffix tree for text X of length n
- after constructing a suffix tree for X, we can determine if a substring p is in X in optimal time
- Right now, our suffix tree takes quadratic space, but we can make it take linear space

X = a b c a a b

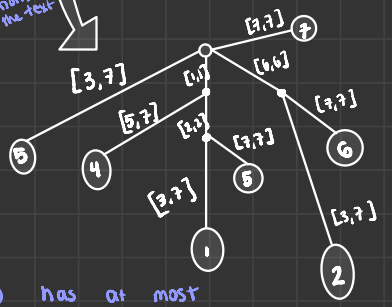
suffixes(x): a b c a a b $
b c a a b $
c a a b $
a a b $
a b $
b $
$



we can compress the branches of the tree



can replace strings with intervals representing positions in the text



X	a	b	c	a	a	b	#
i	1	2	3	4	5	6	7

This bound comes from properties of branching trees

✡ now we have a $O(n)$ space tree:

- a tree w/ n leaves (1 for each of the n suffixes) has at most $n-1$ internal nodes ⟶ $n + (n-1) = 2n-1$ total nodes
  ⟶ no more than $2n$ edges labeled w/ 2 numbers ⟶ linear space

Suffix tree Summary

- a suffix tree T is a data structure that takes a string X of length n as input. It contains all suffixes of X in an efficient way such that:
  1. T stores the starting position of each suffix of X. (@ the leaves)
  2. T stores each substring of X. Edges have strings (or intervals) as labels
  3. each suffix can be identified as a path label from the root to some leaf of T and vice versa: each leaf has 1 suffix as path label from the root to the leaf.
  4. every internal node has at least 2 successor nodes
  5. every edge leaving some node is labeled with substrings that have different 1st characters
  6. every leaf is labeled w/ the starting position of the suffix of X it has as path label.