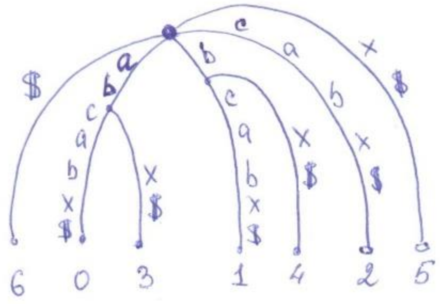
Merwyn Francis Quadras 2581654. **Assignment 2**

Shilova Liubov, no matriculation number yet.

ex. 1

1)



	r
$S\$ = \overset{0}{a}\overset{1}{b}\overset{2}{c}\overset{3}{a}\overset{4}{b}\overset{5}{x}\,\$$	1
bcabx $	3
cabx $	5
abx $	2
bx $	4
x $	6
$	0

P=ab

2) $L := \min[\{r \mid P \leq T[pos[r]\ldots]\} \cup \{n\}]$

$R := \max[\{r \mid P \geq T[pos[r]\ldots pos[r]+|P|\} \cup \{\overset{-1}{\cancel{\ }}\}]$

r	
0	$
1	$\underline{\underline{ab}}$cabx $
②	$\underline{\underline{ab}}$x $
3	bcabx $
4	bx $
5	cabx $
6	x $

L = 1 ; R = 2

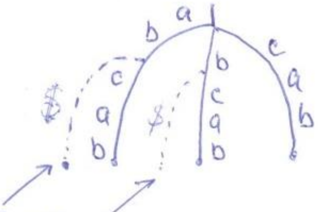
$pos = [\overset{0}{6}, \overset{1}{\underline{\underline{0}}}, \overset{2}{\underline{\underline{3}}}, \overset{3}{1}, \overset{4}{4}, \overset{5}{2}, \overset{6}{5}]$

$\underline{\underline{\overset{0}{a}\overset{1}{b}}}\overset{2}{c}\underline{\underline{\overset{3}{a}\overset{4}{b}}}\overset{5}{x}\ \$$

Pattern P=ab is in a string in positions 0 and 3, interval array (+ len(P)) : [0, 2), [3, 5)

3) Sentinel character → to mark the edge. If one suffix "ends" in the midde of the other, we would not recognize it:

S = abcab
bcab
cab
ab
b



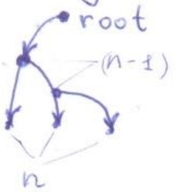
with $ we would have a leaf, which marks the suffix

4) Lemma: • S$ → n leaves; at most n-1 inner node, 2(n-1) edges

• With a $, as shown in (3), each suffix forms exactly one leaf. For every suffix we make a string every time shorter by 1 character => n leaves (Since len(s) = n)

- The inner node is formed, when 2 or more suffixes have the same beginning. When we build a tree, we compare the next suffix with the previous ones. Thus, all suffixes exept for the very first one, have a chance to make 1 node. The first one is not compared to anything. We have n suffixes => n-1 inner node (at most)

Clearly seen from a picture:



- Another explanation! each node has at least 2 child nodes in a compact suffix tree. => at most (n-1) inner nodes.

=> Overall we have 2n-1 nodes (n + (n-1)) and at most (2n-1) edges to connect them.

Exersise 2

Gusfield algorithm.

1) Make a string SS$ – "two circles" bind together, len (SS$) = 2n
2) Build a SuffixTree for the new string. and in each node find the lexicografically smallest edge.
3). Take the found edge, repeat search for the smallest child, take this path.
4) When a leaf, take the index and cut off right before this index.

```
SS = S + S + '$'  # make a string
SuffixTree (SS)
current_node = root
while current_node ≠ leaf:
    SmallestChild = FindTheSmallest (current_node)
    current_node = SmallestChild
Return current_node
```