Exercise 1 – Computational Biology

1 a) $T_n = \ldots\ldots \underbrace{bbbb}_{g_3}\underbrace{bba}_{g_2}\underbrace{ba}_{g_1}\$$

g_4	g_3	g_2	g_1	
bbbba	bbba	bba	ba	$\$$
1 7 7 7 7	7 4 4 4	4 2 2	2 1	0

ℓ_4 ℓ_3 ℓ_2 ℓ_1

f_k

$$\ell_k = 2 + \sum_{i=1}^{k-1} |g_i| = 2 + \sum_{i=1}^{k-1} (i+1) = 1 + \frac{k(k+1)}{2}$$

$$f_k = \ell_k + k \cdot \ell_{k-1} = 1 + \frac{3}{2}k + \frac{k^2}{2} = \Theta(k^3)$$

Let m be the biggest index of a group

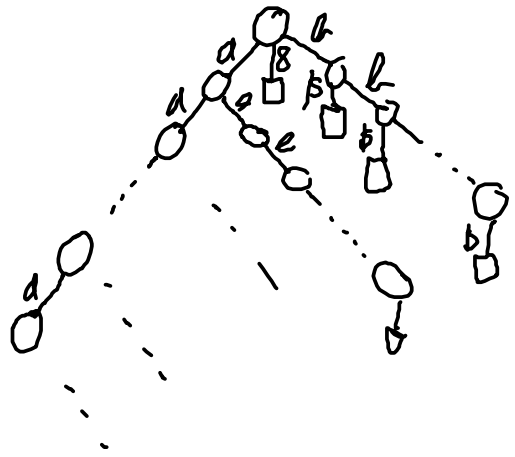
$$n = |T| = \sum_{i=1}^{m} |g_i| = \sum_{i=1}^{m} (i+1) = \Theta(m^2)$$

$$\Rightarrow m = \Theta(\sqrt{n})$$

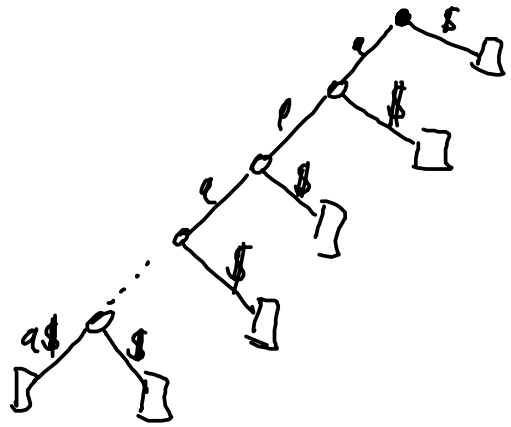
$$t_n = \sum_{i=1}^{m} f_i = \Theta(m^4) = \Theta(n^2)$$

Alternative example:

$T = a^n b^n \$$



1b) $T_n = a^{n-1}\$$



$k < n \rightarrow a^k\$ \sqsupset T_n$

$\rightsquigarrow$ no multiple characters on the edges

Number of inner nodes:

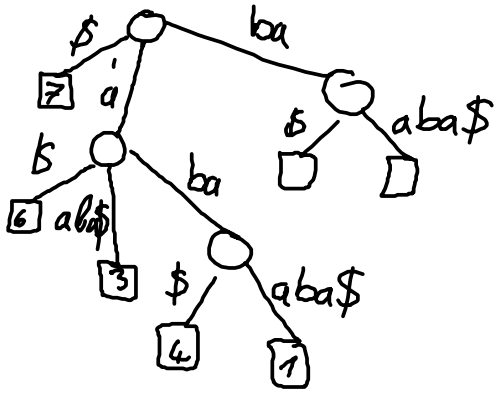
Every leaf except one has inner node as father

$\rightsquigarrow$ $n-1$ inner nodes

$\rightsquigarrow$ theoretical maximum $\rightsquigarrow$ Worst case

Problem 2

$T = ababa\$$



2-repeats: aba, ba, ab, b

Algorithm:

1) Construction B_S for S

2) Post-order traversal annotating every node with the subtree size

3) $m' := \infty$, $P' = \varepsilon$

4) Top-down traversal:
   - record: - current string depth m
     - path label P of current node
     - length l of last edge

   4.1) Check for every node:
   annotated with k and $m - l + 1 < m'$?
   yes: $m' = m - l + 1$
   $P' = P_{1, m-l+1}$
   no: do nothing

5) if $m' = \infty$, output "no k-repeat", otherwise P'

Construction time:

- construction of suffix tree $O(n)$
- 2 traversals : $O(n)$
- < constant number of assignments etc. : $O(1)$

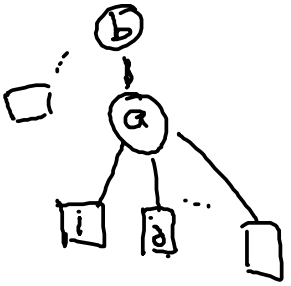
$O(n)$

Problem 3

Build implicit suffix tree $O(n)$

Adapt tree to form a compact suffix tree $O(n)$

Conduct a depth-first-search-like algo



compare T_{i-1}, T_{j-1} for every i,j

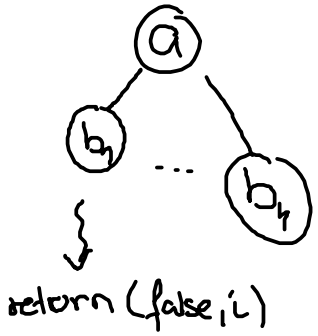
(1) $\exists\, i,j$ $T_{i-1} \neq T_{j-1} \rightsquigarrow a$ left diverse

(2) $\forall\, i,j$ $T_{i-1} = T_{j-1} \rightsquigarrow a$ ¬ left diverse

Information needed by father b:

(1) a is left diverse

(2) a is ¬ left diverse, i (arbitrary)



If b_i left dives for any i:

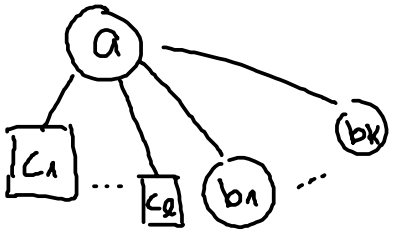
→ a also left dives (1)

else → a not left-div (2)

return: (1) true

(2) false, i (arbitrary)

↳ Same as in case before



If b_i left dives → a left dives (1)

$T_{c_{j-1}} \neq T_{c_{m-1}}$ → a left dives (1)

$T_{i_{b_j-1}} \neq T_{b_{j-1}}$ → "

$T_{i_{b_j-1}} \neq T_{c_i-1}$ → "

return as above

if true returned: output indices i, j that caused $T_{i-1} \neq T_{j-1}$

+ length of string of labels on path to node (can be passed down from father to child)

Problem 4

$$ov(S,T) = \max\left\{|y| \;\middle|\; y \in \Sigma^* \wedge \exists x,z \in \Sigma^+ : S = xy \wedge T = yz\right\}$$

$$\mathcal{T} = \{T^{(1)}, \dots, T^{(m)}\}, \quad \sum_{i=1}^{m} |T^{(i)}| = n$$

$$T = T^{(1)}\$_1 T^{(2)}\$_2 \dots T^{(m)}\$_m \qquad ov[i][j]$$

Algorithm:

1) Compute generalised suffixtree for T

2) For $i := 1$ to m:
   - Set $c = 0$
   - Travers along $T^{(i)}$
     - at every node: check for $j = 1$ to m
       if there is an edge labelled $\$_j$ and the leaf has not start index $j+1$
       yes. $ov[j][i] = c$
     - for every traversed edge, increase c by the edge label length

Runtime:
- suffix tree construction: $O(n)$
- traversal (in $O(m \cdot n)$): at most n nodes with m checks