

# Algorithms in Bioinformatics 2, SoSe2007

## Assignment sheet # 4

Toibas Kloepper

May 7, 2007

To solve this week's problems, please download

`www-ab.informatik.uni-tuebingen.de/teaching/ss07/abi2/java/assign04.zip` and modify the file `albi2.suffixtree.NaiveSuffixTree.java`. Please apply your program to the provided data files.

The goal of this assignment is to first get used to suffix trees and then implement it. The algorithm to be implemented is the naïve algorithm that builds a suffix tree for a text  $t = t_1t_2 \dots t_n$  in  $O(n^2)$  steps. The algorithm starts with a tree consisting of a single node. We process each suffix  $s_1 = t_1 \dots t_n$ ,  $s_2 = t_2 \dots t_n$ ,  $\dots$ ,  $s_{n+1} = \$$  in turn by matching as many characters of such a suffix  $s_i$  as possible into the existing tree and then adding a new leaf edge and leaf node at the appropriate position.

### 1 Construct a suffix tree (2 points)

Please construct by hand the suffix trees for *mississippi* and *atcgtcgaacg*.

### 2 Nodes and edges (1 points)

Please implement two private classes `Node` and `Edge` that can be used to encode the suffix tree.

Additionally, for debugging purposes, please implement a method that recursively prints out the suffix tree.

### 3 Building a suffix tree (4 points)

Implement a method that builds a suffix tree for a given text. The method should operate as follows. Given a suffix  $s_i = t_i \dots t_n$ . Match as long a prefix of  $s_i$  as possible into the tree. If the first mismatch occurs inside the label of an edge  $e$  from  $p$  to  $q$ , then insert a new node  $u$  between  $p$  and  $q$  and then add a new edge to  $u$  that leads to a new leaf node  $z$ . Set the edge labels of the edges  $(p, u)$ ,  $(u, q)$  and  $(u, z)$ . Set the suffix position of the leaf  $z$  to  $i$ . If the first mismatch occurrences at a node  $v$ , then add a new edge  $e$  from  $v$  to a new leaf  $w$ , set the edge label of  $e$  and set the suffix position of  $w$  to  $i$ .

### 4 Searching for all occurrences of a string (3 points)

For a given query  $q$ , attempt to match the whole query into the tree. If the query can be matched into the tree, collect all occurrences by visiting all nodes below the edge or node at which the query was matched. Print out suffix positions obtained from the nodes. Apply your code to the files *missip*, *mississippi* and *mississippimississippi*.

Assignments due: **Monday, May 14, 10am**