TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Fachbereich Informatik
AG Algorithmen und Komplexität

Issue Date: 29.10.2012
Version: 2012-10-29 12:39

Exercise Sheet 2
Computational Biology (Part 2), WS 12/13

Hand In: *Until Monday, 12.11.2012, 10:00 am, email to* `s_wild@cs...` *or in lecture.*

Exercise 5

3 Points

The *k-mismatch inexact string matching problem* consists in finding all "inexact occurrences" of search string $P \in \Sigma^m$ in a text $T \in \Sigma^n$ $(n > m)$ with up to k mismatches.

More precisely, the problem is to find all positions i in the text with

$$\Big|\{j \in [1..m] : P_j \neq T_{i+j-1}\}\Big| \leq k \ .$$

Give an efficient algorithm for solving the k-mismatch inexact string matching problem and analyze its running time.

The algorithm only needs to be efficient for $k \ll m$.

Exercise 6

2 + 5 Points

A generalization of the *k-**mismatch** inexact string matching problem* from Exercise 5 is the so-called *k-**Difference** Inexact String Matching Problem*: There, a subword $T_{i,j}$ of T is considered an occurrence of search string P iff $T_{i,j}$ and P have *edit distance*[1] $\leq k$.

a) Design a data structure based on compact suffix trees with which we can compute the *longest common extensions* of two positions in *two* words in constant time (as done for two positions in the *same* word in the lecture).

   Formally, we define for two words $u \in \Sigma^n$ and $v \in \Sigma^m$:

$$lce(i,j) := u_{i,i+\ell_{\max}} \quad \text{where} \quad \ell_{\max} := \max\{\ell \geq 0 : u_{i,i+\ell} = v_{j,j+\ell}\}$$

   Hint: Read/Review the section on the subword problem for a set texts, page 59f in the lecture notes.

b) Design an efficient algorithm for the k-difference inexact string matching problem and determine its running time.

   Hint: Use *lce*-queries (and the datastructure form a) to efficiently answer them) and dynamic programming.

[1] Find a definition of edit distance on page 66 of the lecture notes (last paragraph above "Globale Alignments").

Exercise 7

3 + 2 Points

In the lecture, we considered an algorithm to compute all tandom repeats.

a) Describe a method based on this algorithm to compute all *triple repeats*, i. e. all subwords of shape xxx in a text T.

b) Generalize your method to *higher order repeats*, i. e. subwords of the form x^k for arbitrary $k \geq 2$.