Technische Universität München
Department of Informatics
Chair for Efficient Algorithms
Prof. Dr. Ernst W. Mayr/Dr. Jens Ernst
Johannes Nowak

Summer Term 2005
Exercise Sheet 8
June 21, 2005

Selected Topics in Computational Biology

Due: 28.06.2005 **after** *the lecture*

Exercise 1 (10 points)

The NW-algorithm, which was described in the lecture, computes the (global) alignment distance of two sequences. In biological context we often have the situation that sequences with very different lengths are compared. The short sequence might have significant similarity to a relatively short subsequence of the longer sequence, which is not recognized by the global comparison. This motivates the notion of a *semi-global* alignemnt.
In a semi-global alignment, the costs of insertions and deletions at the left or right end are not considered in evaluating the alignment. Moreover, instead of an arbitrary cost function $w : \Sigma \times \Sigma \to \mathbb{R}$ we use the corresponding similiraty measure $s : \Sigma \times \Sigma \to \mathbb{R}$, where $s(a,b) = 1 - w(a,b)$ for all $a, b \in \overline{\Sigma}$. The similarity of an alignment $(\overline{s}, \overline{t})$ and a given cost function function w is given by

$$s(\overline{s}, \overline{t}) = \sum_{i=k}^{l} s(\overline{s}_i, \overline{t}_i)$$

where k and l are the positions of the first and last match or substitution.

a) Why is it useful to consider similarity instead of cost in the context of semi-global alignments?

b) Give an algorithm that computes the *semi-global alignment similarity* between two sequences s and t. That is, we are interested in the maximal similarity of s and t. (**Hint**: Modify the NW-algorithm accordingly.)

Exercise 2 (10 points)

Prove that the Hirschberg algorithm runs in time $O(nm)$. This can be done as follows.

- Let $T(n,m)$ be the time which is needed by the Hirschberg algorithm for two sequences of lengths n and m. Give a recursion formula for $T(n,m)$.
- Prove by induction that $T(n,m) = O(nm)$ is satisfied.

Programming Task

Implement a program that computes the alignment distance and returns an optimal alignment for two sequences s and t. The cost function is again given by:

$$\overline{w}(x,y) = \begin{cases} 0 & \text{if} \quad x = y \neq \{-\} \\ 1 & \text{else} \end{cases}$$

As in earlier programming tasks, the sequences are given in one file and are separated by the unique character \$.