# Algorithms for Bioinformatics (Autumn 2011)

## Exercise 3 (Thu 29.9, 10-12, BK107, Niko Välimäki)

1. **Correctness of improved breakpoint reversal sort.**

   Prove that the improved breakpoint reversal sort algorithm (page 28 at lecture slides) works correctly. That is, define concepts *increasing strip* and *decreasing strip* formally and then prove that if permutation $\pi$ contains one or more decreasing strips, then there is always a reversal that decreases the number of breakpoints $b(\pi)$ at least by one.

2. **Translocations via reversals.**

   Page 30 at lecture slides gives an example of a *translocation* simulated by three reversals. Define formally what translocation operation does for a permutation $\pi_1\pi_2\cdots\pi_n$ and prove that any translocation can be replaced by three reversal operations.

3. **Implementing improved breakpoint reversal sort.**

   Write a Python program that implements improved breakpoint reversal sort and analyse the running time of your implementation.

4. **Shortest approximate superstring.**

   Let $\mathbf{S} = S_1, S_2, \ldots, S_n \subseteq \Sigma^*$ be a set of strings from alphabet $\Sigma$. Given a threshold parameter $k$, an *approximate superstring* of $\mathbf{S}$ is defined as a string $T$ such that for each $S_i \in \mathbf{S}$ it holds $d_H(S_i, T[j_i \cdots j_i + |S_i| - 1]) \leq k$ for some $j_i$, where $d_H()$ denotes the Hamming distance.

   A greedy approximation algorithm for finding the *shortest approximate superstring* can be derived as follows. Let an *approximate overlap* of $A = \alpha\gamma, B = \gamma'\beta \in \mathbf{S}$ be pair of strings $(\gamma, \gamma')$ such that $d_H(\gamma, \gamma') \leq k$ and the length of the overlap $|\gamma| = |\gamma'|$ is maximum among all ways to to write $A$ and $B$ in parts $A = \alpha\gamma$ and $B = \gamma'\beta$. Iterate the following until there is only one string in set $\mathbf{S}$: (1) Choose $\alpha\gamma, B = \gamma'\beta \in \mathbf{S}$ with maximum approximate overlap; (2) remove $A$ and $B$ from $\mathbf{S}$ and insert $\alpha\gamma\beta$ into $\mathbf{S}$.

   Simulate the above greedy algorithm with $k = 1$ on the set {`ACACGATC, ATGACAAA,TAATAAGA,CAGGATCA`}.

   Is the solution of your simulation a valid approximate superstring? Does the algorithm always find a valid approximate superstring? If not, give a modification so that it does.

5. **Dynamic programming.**

   *Dynamic programming* or *tabulation* is a powerful algorithm design paradigm, providing a way to organize computation so that exponential time exhaustive search can, in some cases, be replaced by polynomial time computation. Simplest example was given at the first lecture; recursive fibonacci was replaced by array (tabulation) version of fibonacci computation. Recall the Change problem from

the same lecture. Show how dynamic programming can be used to solve it much faster than the exhaustive enumeration approach. *Hint.* Fill an array of size $M$ from left to right.