Algorithms for Biological Sequence Analysis (Midterm # 1)

Instructor: Kun–Mao Chao TA: Yi-Ching Chen

November 6, 2007

Problem 1 (15%): Suppose we are given a very long DNA sequence where the occurrence probabilities of nucleotides A (adenine), C (cytosine), G (guanine), T (thymine) are 0.1, 0.3, 0.4, and 0.2, respectively.

(a) (10%): Construct a Huffman code for them. You should work out the binary tree construction as well as the code assignment.

(b) (5%): By the above Huffman coding scheme, what is the binary string for a 10-nucleotide DNA sequence "GGGCTTCACG."

Problem 2 (15%): In class, we introduced an $O(n \log n)$-time algorithm for finding a longest increasing subsequence. Use $\langle 8, 2, 6, 4, 5, 7, 3, 1, 12, 9, 10\rangle$ to explain how the algorithm works.

Problem 3 (10%): Given a sequence of real numbers $A = \langle a_1, a_2, \ldots, a_n\rangle$, the maximum-sum segment problem is to find a consecutive subsequence, *i.e.*, a substring or segment, in A with the maximum sum. Let *prefix sum* $P[i] = \sum_{j=1}^{i} a_j$ be the sum of the first i elements. Explain how to use the prefix sum to deliver the maximum-sum segment in $O(n)$ time.

In the following, we are given two sequences $A = \langle a_1, a_2, \ldots, a_m\rangle$ and $B = \langle b_1, b_2, \ldots, b_n\rangle$. An alignment of A and B is obtained by introducing dashes into the two sequences such that the lengths of the two resulting sequences are identical and no column contains two dashes. Let Σ denote the input symbol alphabet. A score $\sigma(a, b)$ is defined for each $(a, b) \in \Sigma \times \Sigma$. The score of an alignment is the sum of σ scores of all columns with no dashes minus the penalties of the gaps.

Problem 4 (25%): In this problem, we employ a simple scoring scheme where each gap symbol is penalized by a nonnegative constant β. Let $S[i, j]$ denote the score of an optimal alignment between $\langle a_1, a_2, \ldots, a_i\rangle$ and $\langle b_1, b_2, \ldots, b_j\rangle$. With proper initializations, $S[i, j]$ can be computed by the following recurrences:

$$S[i,j] = \max \begin{cases} S[i-1,j] - \beta \\ S[i,j-1] - \beta \\ S[i-1,j-1] + \sigma(a_i, b_j) \end{cases}$$

(a) (15%): Write down a complete pseudo-code for computing $S[m, n]$ in $O(mn)$ time and $O(m + n)$ space. All initializations should be included in the pseudo-code.

(b) (10%): Assume that we allow at most three gaps in an alignment. Give a method (as efficient as possible) for computing the score of an optimal alignment.

Problem 5 (20%): In affine gap penalties, a gap of length k is penalized by $\alpha + k \times \beta$, where α and β are both nonnegative constants.

(a) (10%): Give the recurrence relations for computing the score of an optimal (global) alignment between A and B. Justify your recurrence relations and include all initializations.

(b) (10%): Give the recurrence relations for computing the score of an optimal *local* alignment between A and B. Explain your recurrence relations and include all initializations.

Problem 6 (15%): Consider the problem of computing all Δ-points of two sequences of lengths m and n, where $m \ll n$. Describe a method for computing all Δ-points that works in $O(mn)$ time and $O(m^{\frac{11}{10}} + n)$ working space.