Introduction to Bioinformatics Algorithms
Homework 3

Saad Mneimneh
Computer Science
Hunter College of CUNY

Problem 1: Concave penalty function
We have seen in class the following recurrence for alignment with a general penalty function:

$$A(i,j) = \max \begin{cases} A(i-1,j-1) + s(x_i, y_j) & \\ A(i-k,j) - \gamma(k) & k = 1, \ldots, i \\ A(i,j-k) - \gamma(k) & k = 1, \ldots, j \end{cases}$$

where $A(i,0) = -\gamma(i)$ and $A(0,j) = -\gamma(j)$, and $\gamma(k)$ is the penalty of a gap of length k.

(a) Show by a counter example that this algorithm fails to correctly find the optimal score. *Hint*: construct a γ that is not concave.

(b) Show that if $\gamma(0) = 0$ and $\gamma(k) - \gamma(k-1)$ is non-increasing in k, then we have (sub-additive property)

$$\gamma(k_1 + k_2) \leq \gamma(k_1) + \gamma(k_2)$$

Explain why this property makes the algorithm above correct.

(c) Modify the algorithm to work with any gap function.

Problem 2: Substitution matrices
Let's say that we would like to build a DNA substitution matrix (4x4 matrix) optimized for finding alignments among sequences that have 99% conservation (so mutation rate is 0.01, evolutionary distance 1). Assume that $p_i = 0.25$ for every nucleotide i (A, G, C, or T), and that all matches are equally probable, and all mismatches are equally probable (uniform model).

(a) Find the matrix Q, where Q_{ij} is the probability of seeing nucleotides i and j aligned.

(b) After constructing the matrix M, where $M_{ij} = Q_{ij}/p_i$, find the match and mismatch scores in the matrix S given by $S_{ij} = \log_2 M^k_{ij}/p_j$, for evolutionary distances $k = 1, 2, 5, 25, 50, 75, 100$. What is the percentage conservation in each case (e.g. for $k = 1$ it is 0.99)?

(c) [optional] Show that regardless of the probabilities p_i and Q_{ij}, S will always be symmetric for any evolutionary distance. *Hint*: $M = DQ$ for some diagonal matrix D (what is D?), and $S = \log_2[(DQ)^k D]$.

Problem 3: Longest increasing subsequence
Given a sequence $x_1, x_2, \ldots, x_n$, an increasing subsequence of length k is a sequence $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ such that $i_1 < i_2 < \ldots < i_k$ and $x_{i_1} < x_{i_2} < \ldots < x_{i_k}$. For example, in the sequence $8, 2, 1, 6, 5, 7, 4, 3, 9$, an increasing subsequence is $1, 5, 7, 9$. It has length 4, and it is a longest possible increasing subsequence.

(a) Describe a dynamic programming algorithm to find a longest increasing subsequence. *Hint:* you can think about alignment between the sequence and its sorted version (what scores will you assign for matches, mismatches, and gaps?).

(b) A 2-increasing subsequence is one that can be partitioned into two subsequences that are increasing. For example, while $2, 1, 6, 5, 7, 9$ is not an increasing subsequence, it is a 2-increasing subsequence because it can be partitioned into: $2, 6$ and $1, 5, 7, 9$. Show by a counter example that the longest 2-increasing subsequence cannot be obtained by a greedy strategy, i.e. removing the longest increasing sequence, then finding the longest increasing sequence among the remaining elements, and finally interleaving the two.

(c) Describe a dynamic programming algorithm to find the longest 2-increasing subsequence. *Hint:* you can think about aligning the sequence with two of its sorted versions. But this is not the standard multiple alignment because an alignment up to x_i, y_j, and z_k can end in one of 5 possibilities:

$$\begin{array}{ccccc} x_i & x_i & x_i & - & - \\ y_j & - & - & y_j & - \\ - & z_k & - & - & z_k \end{array}$$

Problem 4: Homodeletions
Do problem 6.40.