UC Berkeley, CS 176: Algorithms for Computational Biology (Fall 2014)

Problem Set 2

Instructor: Nir Yosef

Out: Oct 1, 2014 Due: Oct 14, 2014

1. **(FM-index and Exact Pattern Matching - 10pt)** Given a string S of size $|S| = n$, let $\Pi^{\text{sorted}}(S)$ denote the matrix containing the sorted cyclic permutations of S as rows, and let L denote the last column of that matrix (i.e., the BWT of S). The first column of $\Pi^{\text{sorted}}(S)$ is denoted by F. For σ a non-empty substring of S, we use $sp(\sigma)$ (respectively, $ep(\sigma)$) to denote the index of the first (respectively, last) row in $\Pi^{\text{sorted}}(S)$ such that σ occurs as a prefix of that row. (Recall that, in our convention, the row index takes value in $\{1, \ldots, n\}$.) Prove the following statements:

   (a) For all character c, if the concatenated string $c\sigma$ is a substring of S, then the following recursions hold:

$$\begin{aligned} sp(c\sigma) &= M[c] + \text{OCC}(c, sp(\sigma) - 1), \\ ep(c\sigma) &= M[c] + \text{OCC}(c, ep(\sigma)) - 1. \end{aligned}$$

   (b) If σ is a substring of S, and $sp(c\sigma)$ and $ep(c\sigma)$ are computed using the above recursions, then $sp(c\sigma) \leq ep(c\sigma)$ if and only if $c\sigma$ is a substring of S.

2. **(Optimal Substructure Property in NW - 10pt)** Given two strings X and Y, let $V(i, j)$ be the optimal score for an alignment of $X[1..i]$ $Y[1..j]$. In class, we mentioned that V has the optimal substructure property for both linear and affine gap penalties. Here, we will prove it.

   (a) (Linear Case) Show that V satisfies the recurrence

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + \sigma(x_i, y_j). \\ V(i-1, j) + \sigma(x_i, -). \\ V(i, j-1) + \sigma(-, y_j). \end{cases} \tag{1}$$

   (b) (Affine Case) Let G, E, F be defined as in class. Show that V, G, E, F satisfy the recurrences

$$\begin{aligned} G(i,j) &= V(i-1, j-1) + \sigma(x_i, y_i) \\ E(i,j) &= max\{E(i, j-1) + W_s, \quad V(i, j-1) + W_g + W_s\} \\ F(i,j) &= max\{F(i-1, j) + W_s, \quad V(i-1, j) + W_g + W_s\} \end{aligned} \tag{2}$$

$$V(i,j) = max\{E(i,j), \quad F(i,j), \quad G(i,j)\}$$

3. **(Hirschberg Algorithm Runtime - 10pt)** In class, we mentioned that Hirshberg's linear space algorithm is a constant factor slower than the Needleman-Wunsch algorithm. Estimate that constant; justify your estimation rigorously.

4. **(Counting Co-optimal Alignments - 10pt)** In the global alignment problem involving two sequences of lengths n and m, devise an $O(nm)$-time algorithm to compute the number of distinct co-optimal alignments.

5. **(Global Alignment of Circular DNA - 10pt)** Bacterial DNA is often organized into circular molecules. This motivates the following problem: Given two linear strings of lengths n and m, there are n circular shifts of the first string and m circular shifts of the second string, resulting in nm pairs of strings. We want to compute the global alignment of each of these nm pairs of strings. It turns out that this problem can be solved faster than the naive time bound $O(n^2m^2)$, which follows from applying the $O(nm)$-time dynamic programming algorithm to each pair of strings. Devise an $O(nm^2 + n^2m)$-time algorithm for finding an optimal global alignment for every pair of circular shifts.