CSC 2417 Algorithms in Molecular Biology
PS2: Due November 22 in Class

Don't Panic

This is an individual assignment. While you may discuss this assignment with classmates, please do not give away answers. You are NOT allowed to use the internet, besides to look up things not directly related to the assignment, such as a generic formula or a well-known algorithm. This homework may have bugs. If you spot something that looks wrong or is not clear please contact me.

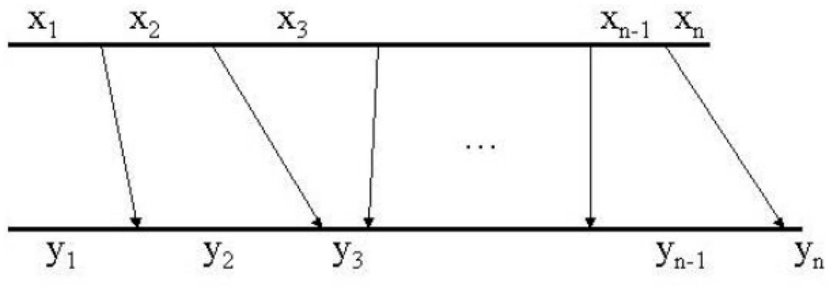
1. More Alignment

Recall the LCS to LIS reduction discussed in class. Here we will modify it in order to build the local alignment chaining algorithm used in LAGAN and similar tools. A local alignment can be thought of as a rectangle defined by its start and end points. It also has a score which show how well it is conserved. The requirement that we need to enforce is that the starting (top right) point of the next rectangle always has to be below and to the right of the end point of the previous one.

(a) Given two sequences, A, B with *n* matches between them as well as a score for every match show how to use a variation on the Longest Increasing Subsequence (LIS) algorithm to find the highest scoring common subsequence in time *O(n log n)*. Assume the LIS algorithm is know to the reader.

(b) Now consider the case of a multiple alignment, where we are aligning k sequences, each local alignment is a k-tuplet with coordinates from all sequences, and our goal is to construct the longest chain that is strictly increasing in all k sequences. Demonstrate an algorithm for this problem (algorithms faster than $O(n^2)$ will get a bonus).

Now let X be a sequence of length L_1 and Y be a sequence of length L_2. Let A_X and A_Y be two sequences of $(n - 1)$ anchors that will be used to align X and Y. Anchors in A_X and A_Y are ordered from left to right (anchor i of A_X will match anchor i of A_Y , and is located to the left of anchor j of A_X if and only if $i < j$).



Assume the lengths of the anchors themselves are negligible. Let x_i be the distance between anchors $i-1$ and i in A_X, and similarly for y_i. We also include the distances between the beginning of the sequence and the first anchor, and between the last anchor and the end; therefore, the sets $X_d = x_1, \ldots, x_n$ and Y_d both have cardinality n, 1 greater than the number of anchors.

We will globally align X and Y by fixing the alignments of corresponding anchors to each other, and using NW to align the portions of X and Y between neighboring anchors.

(c) Describe the locations of the n points on the alignment matrix that give the *worst* case running time, and how many cells in the Needleman-Wunsch algorithm will examine. locations of the n points on the alignment matrix that give the *best* case running time, and how many cells in the Needleman-Wunsch algorithm will examine.

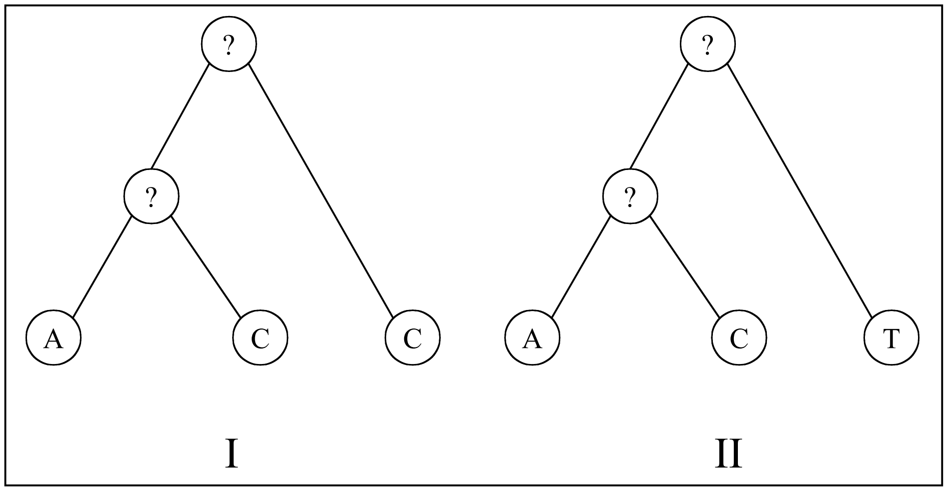
(d) Derive the general formula for the running time in terms of the lengths of the sequences, the number of anchor points and the covariance of the occurrences of the anchor points:

Definition. Given sequences of numbers $X_d = x_1,\ldots, x_n$, $Y_d = y_1,\ldots, y_n$, the covariance of X_d, Y_d is:

$$\mathrm{cov}(X_d, Y_d) = 1/n \, \Sigma_{\{i=1,\ldots,n\}}(x_i - x_{mean})(y_i - y_{mean})$$

2. Evolutionary Trees & Breakpoints

(a) Given the correct rooted evolutionary tree for a set of organisms and the value that a nucleotide has now for all extant species, we may ask what base pair did the ancestor of all of these have. The internal nodes of the tree correspond to ancestral sequences – organisms that are no longer alive but that lead to the current organisms. Develop an algorithm that for each internal node of the tree (labeled with "?") determines which nucleotides could have been in that particular ancestor under the "minimally parsimonious" scenario – that is the scenario that has the fewest overall mutations. If an ancestor could have had more than one letter at this position it should be labeled with all the possibilities. For example consider the two trees below. In case (I) both internal nodes must be labeled "C" under maximum parsimony, as this allows for only one mutation (C->A on the lower left branch). In case (II) the lower node is (A/C) and the root is (A/C/T).



(b) Multiple alignment algorithms which use the progressive technique often construct the phylogenetic tree using UPGMA rather than Neighbor Joining algorithms, and several studies claim that this gives better results. Speculate why this may be so.

(c) One way to measure rearrangement distances between genomes that we mentioned in class is by counting *breakpoints*. In the breakpoint median problem, given a set of 3 signed permutations we want to design a 4th signed permutation such that the total number of breakpoints between it and the other three is minimal. Show how to reduce this problem to the Traveling Salesman Problem. (Although TSP is NP-hard, it is very widely studied and there are many effective heuristics for it).

3. Multiple Alignment

Consider the following definitions:

Definition (1) Given a multiple alignment M of a set of strings S, the *consensus character* of column i of M is the character that minimizes the summed score between the character and all the characters in column i. (In case of ties, say by convention that we prefer A over C over G over T over 'gap'). The score of (gap, gap) is 0. Let d(i) denote that minimum sum in column i. (2) The *consensus string* S_M derived from alignment M is the concatenation of the consensus characters for each column of M. (3) The *alignment score* of S_M equals to the sum of column scores $d(S_M) = d(1) + \dots + d(m)$, where M has m columns. (4) The *optimal consensus multiple alignment* is a multiple alignment M for input set S whose consensus string S_M has smallest alignment error over all possible multiple alignments of S. (Definitions are adapted from Gusfield, p. 352.)

Example: S = {AGCC, ACC, TCC}, and match, mismatch, gap = +2, -2, -3. Consider the following alignments:

M_1: {AGCC, A-CC, T-CC}; S_{M1} = A-CC, and $d(S_{M1}) = (+2) + (-3) + (+6) + (+6) = 11$.

M_2: {AGCC, A-CC, -TCC}; S_{M1} = AGCC, and $d(S_{M1}) = (+1) + (-3) + (+6) + (+6) = 10$.

Definition (1) Given an input <u>rooted</u> binary tree T with a distinct string (from a set of strings S) written at each leaf, a *phylogenetic alignment* for T is an assignment of one string to each internal node of T. Note that the strings assigned to internal nodes need not be distinct and need not be from the input string S. (2) If strings S and S' are assigned to the endpoints of an edge (i, j), then that edge has *edge distance* D(S, S'), which is simply the edit distance between the two strings S and S'. The distance of a phylogenetic alignment is the total of all edge distances in the tree. (3) The *phylogenetic alignment* problem for T is to find an assignment of strings to internal nodes of T (one string to each node) that minimizes the distance of the alignment.

Note: it is also possible to define the problem where T is unrooted. If you prefer that definition, please go ahead and use it instead.

Example: S = {x = AGCC, y = ACC, z = TCC}, and match, mismatch, gap = +2, -2, -3. Let T = [V = {x, y, z, v_{yz}, v_{xyz}}; E = {(x, v_{xyz}), (v_{xyz}, v_{yz}), (v_{yz}, y), (v_{yz}, z)}, root = v_{xyz}], with leafs {x, y, z} and root v_{xyz}

Here is a phylogenetic alignment, which is just a labeling of the internal nodes: Label v_{yz} with "ACC", and v_{xyz} with "ACC". Then, the alignment score is D(x, v_{xyz}) + D(v_{xyz}, v_{yz}) + D(v_{yz}, y) + D(v_{yz}, z) = (+3) + (+6) + (+6) + (+2) = 17.

Show an example with three or more sequences where their optimal phylogenetic alignment, the optimal consensus multiple alignment, and the optimal sum-of-pairs multiple alignment all differ from one another. Assume a match, mismatch, and gap penalty of +2, -2, -3. Let the alphabet be {A,C,G,T}. You will get partial credit for finding sequences where two of the scoring schemes differ. In our solution it is pretty obvious that the alignment, is, in fact, optimal.