CSC 2427 Algorithms in Molecular Biology
PS1: Due February 16 before 5pm

DON'T PANIC

You may work with others on this homework assignment, but please submit your own writeup. You must acknowledge the contributions that other students make to your answers. Note: this homework may have bugs. If you spot something that looks wrong or is not clear please contact me, or post to the newsgroup. Acknowledgment: Some of these problems are from Stanford's CS 262 taught by Serafim Batzoglou.

1. Assembly & Overlaps

(a) There are 4^n different RNA molecules of length n. How many different DNA molecules of length n are there? Recall that a DNA molecule is double stranded. So a DNA molecule of length n will have 2n nucleotides in it, with n Watson-Crick base pairs.

(b) Assume that some genome has length G, and that N reads, each of length L, will be assembled to get this genome. The coverage of the genome is NL/G. Assume that G is much larger than L. The reads are assumed to be taken at random from the genome so that, ignoring end-effects, the position of the left-hand end of any read is uniformly distributed in [1,G]. A contig is a maximal covered subsequence in the genome.
i. What is the mean proportion of the genome covered by contigs? How much should the coverage be for 99% of the genome to be covered? How much should the coverage be for 99.9% of the genome to be covered?
ii. What is the mean number of contigs, in terms of N,L and G?
iii. What is the mean contig size?

(c) Literature analysis:
i. Pevzner and colleagues in their papers do not directly provide a method for dealing with edge directionality, but rather suggest that every k-mer be represented twice, and the resulting graph be divided into the two subgraphs, each representing the assembly of a particular strand. They could adopt their technique to use bi-directed edges. However in this case another of the assumptions they make does not hold. Which one?
ii. Myers' in his paper suggests using network flow to find potential tours of the string graph. This includes a method to reduce the network flow problem with lower bounds to an equivalent one without lower bounds (section 6). Can you spot a problem with the reduction? What are the potential problems in solving the network flow problem as he poses it, and not the Hamiltonian Path Problem?

(d) Finding overlaps:
Because in reality sequenced reads can have sequencing errors, we must allow for non-exact matches between reads. We may, for example, say that two reads overlap if they fewer than D discrepancies (substitutions, insertion or deletions: this is equivalent to edit distance < D). This is typically done using a simple variation on the standard Needleman-Wunsch algorithm, with running time $O(n^2)$ for two reads of length *n*. Demonstrate an algorithm that is asymptotically faster than this approach for a fixed D.

2. Linear Space Alignment

Recall Hirschberg's linear-space global alignment algorithm. In this problem, you will be asked to consider possible extensions of it to other alignments. For each part below, either extend the linear-space algorithm to the version of the problem, or discuss why this is not possible.

(a) Local alignment (Smith-Waterman), where you are asked to find a single optimal highest scoring local alignment.

(b) In the linear space alignment, the original problem of size mn is reduced to two subproblems of sizes km/2 and (n-k)m/2. In a fast, parallel implementation of sequence alignment, it is desirable to have a *balanced partitioning* that breaks the original problem into sub-problems of equal sizes. Design a linear space alignment algorithm with balanced partitioning.

3. Variations

(a) The score of a local alignment is not normalized over the length of the matching region. As a result, a local alignment with score 100 and length 100 will be chosen over a local alignment with score 99 and length 10, although the latter one is probably more important biologically. To reflect the length of the local alignment in scoring, the score F(I,J) of local alignment involving substrings I and J may be adjusted by dividing F(I,J) by the total length of the aligned regions: F(I,J)/(|I|+|J|). The *normalized local alignment problem* is to find substrings I and J that maximize F(I,J)/(|I|+|J|) among all substrings I and J with |I|+|J| >= k, where k is a threshold for the minimum overall length of I and J. Devise an algorithm for solving the normalized local alignment problem.

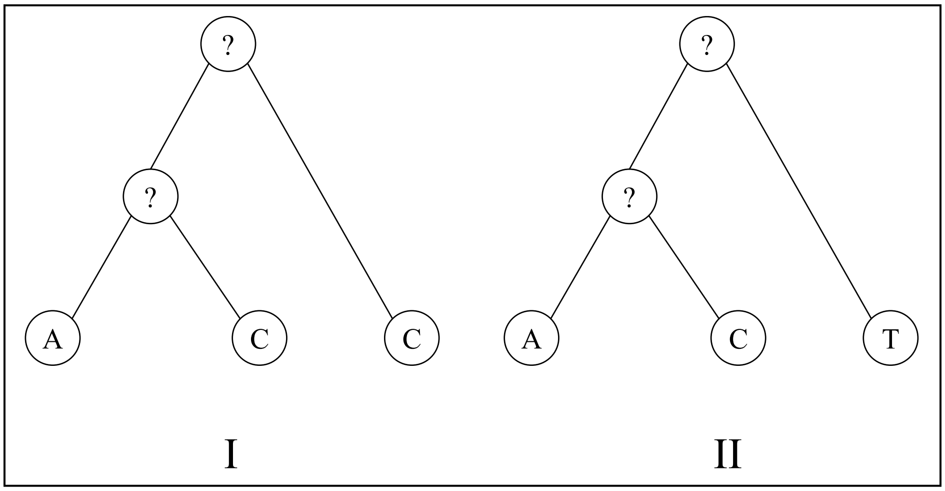
(b) Two sequences of length n & m may have more than one optimal alignment. Give an algorithm to compute the number of optimal alignments between two such strings in O(mn) time.

(c) Since the biological significance of the optimal alignment is sometimes uncertain, and optimality depends on the choice of (often disputed) weights, it is useful to efficiently produce or study a set of *suboptimal* (but close) alignments in addition to the optimal one. Given two strings X and Y (of lengths n and m) and a parameter δ, show how to construct the following matrix in O(nm) time: M(i,j) = 1 if and only if there is an alignment of X and Y in which characters X[i] and Y[j] are aligned with each other and the value of the alignment is within δ of the maximum value alignment of X and Y. That is, if F(n,m) is the is the value of the optimal alignment, then the best alignment that puts X[i] opposite Y[j] should have value at least F(n,m) – δ.

(d) Bacterial DNA is often organized into circular molecules. Given two strings x and y of length n and m, respectively, there are n circular shifts of x, and m circular shifts of y; therefore, there are nm pairs of circular shifts.
Build an efficient algorithm to find the best global alignment among the nm different pairs of circular shifts. The trivial $O(n^2m^2)$ algorithm is not good enough. A cubic time solution (i.e. $O(n^2m)$) will only get partial credit.

4. Evolutionary Trees

(a) [This problem may be difficult] 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.

5. Multiple Alignment

(a) Consensus multiple alignment versus sum-of-pairs multiple alignment.

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 $d(S_M) = d(1) + \ldots + d(m)$, 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.

Show an example with three or more sequences where their optimal multiple alignment according to the above model is different from the one according to a SP model. Assume a match, mismatch, and gap penalty of +2, -2, -3. Let the alphabet be {A,C,G,T}.

(b) Phylogenetic-tree–based alignment.

Definition (1) Given an input rooted 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}.