CSC 2427 Algorithms in Molecular Biology
PS2: Due March 25 in Class

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: ut.cdf.csc2427h. Note that questions on the newsgroup will be answered **before** those sent by e-mail.

1. More Alignment [25 pts]

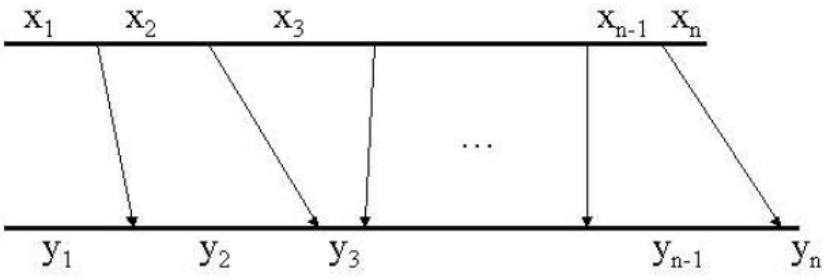
Recall the LCS to LIS reduction discussed in class. Here we will modify it in order to build the local alignment chaining algorithm used in LAGAN and similar tools. A local alignment can be thought of as a rectangle defined by its start and end points. It also has a score which show how well it is conserved. The requirement that we need to enforce is that the starting (top right) point of the next rectangle always has to be 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 modify the algorithm developed in part (a) to allow for chains of local alignments. [Hint: you will need to do some processing for every local alignment twice]

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 $i < j$).

Throughout this problem, 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:

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

2. Motifs & Rearrangements [25 pts]

(a) The sequence below is the RNA of a fake gene with exactly one intron. Where is the intron and what is the coding sequence?

ATGCAGTCTAGGTAA

(b) In the Gibbs Sampling examples discussed in class we were searching for ungapped motifs. It is actually not very well known how many – if any gaps transcription binding sites can tolerate. You are asked to come up with a Gibbs Sampling algorithm that will search for gapped motifs. Instead of returning a position weight matrix (PWM) of length K, your algorithm should return an alignment of length K. For simplicity we will treat the gap as a 5^{th} DNA character.

Describe how one may search for these gapped motifs in the one sequence we are currently leaving out. Also suggest a method to calculate the background probability of finding a certain alignment – this is not trivial as very little is know about the statistics of alignments with gaps

(c) The Gibbs Sampling algorithm we described in class (and many other related approaches) assume independence between adjacent positions. This is not always a valid assumption. How would you have to change the PWM in order to incorporate dependence between positions? What is the related complexity for a full model and a motif of length k?

(d) 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 4^{th} 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. Gene finding [50 pts]

In this problem you will develop and implement a simple Hidden Markov Model to search for bacterial genes. Bacterial genes do not have introns, and can usually be found by searching for Open Reading Frames (ORFs), sequences of amino acids that start with an ATG and go till a stop codon. For the coding sections you can use any language you wish, you should submit a printout of your code with your solution or by e-mail to the instructor.

(a) Write a program that given a genomic sequence searches for ORFs of length > K amino acids, where K is a parameter. Your program should take a genome in FASTA format (an example will be posted on the website) and find all these ORFs in all 6 frames. Report the length of the longest ORF, as well as the mean size of all ORFs ≥ 600 nucleotides.

(b) Using the ORFS ≥ 600 bp long, determine the frequency of the various codons (DNA triplets) in ORFs. Map the codons to the amino acids (so it is a 21 element frequency table) and include with your writeup. Likewise determine and submit the same emission table for the four residues {A,C,T,G} in the DNA outside of these ORFs (putatively non-coding DNA).

(c) Design a Hidden Markov Model that will search for genes in bacterial genomes. Your HMM should search for genes in 3 frames (it can be run separately on the forward and reverse strands). Your HMM should have a *begin* and an *end* state. It should model start & stop codons, and also take into account the emission probabilities you computed in section **b.** Explain how you computed the remaining parameters (there is more than one correct way to do this). Hand in the state diagram of your HMM with all transition edges labeled with your parameters.

(d) Implement the HMM you designed in section (c). Run it on the genome and use the Vitterbi algorithm to predict the genes in the sequence. Draw a histogram of ORF sizes that you find. What is the mean ORF size? What is the shortest ORF that you predict? For the shortest ORF you find use the NCBI BLAST program to find out if it is a real gene or a false positive: you should translate the ORF into a protein and use the blastp program available at http://www.ncbi.nlm.nih.gov/blast/ to search against the NR database of all known proteins. Hand in your ORF and your synopsis of results.