Project 2 suggestions

The following is a list of possible projects. If you decide to pick one I can give you the necessary details and point you to the relevant information that will allow you to complete the project. The broad rules (which will be presented in more detail on Monday) are:

- You can come up with a project other than those described below
- You can work in teams, though you'll have to convince me that the scope of the project is broad enough.
- Each of you doesn't have to pick a different project (e.g. it's OK if all of you decide to pick the same project)
- The project submission will include a 2-page write-up of the project, in addition to the usual README and code. Please allow sufficient time for working on the write-up as well.
- PICK A PROJECT YOU CAN COMPLETE IN THE TIME ALLOTTED!
- START WORKING ON THE PROJECT EARLY!

1. **Shotgun sequence overlapper.** Write a program that performs the pairwise comparisons between the set of sequences provided as input to a shotgun sequence assembler and report which sequences overlap. This project will require you to use a combination of exact matching (e.g. k-mer hashing) and inexact matching (Smith-Waterman) techniques.
2. **Aligner for 454 data.** Sequencing data produced by 454 Life Sciences contains frequent errors in homopolymer regions—sections of DNA where a single nucleotide is repeated (e.g. AAAA). This project involves modifying the Smith-Waterman aligner produced during the first project to allow for errors in such regions (i.e. gaps adjacent to a homopolymer are penalized less).
3. **SNP finder.** Given a multiple alignment produced by a genome assembler, identify columns in the alignment where there are high-quality discrepancies between overlapping reads and report these positions. This project will require you to learn how to interpret the output of a genome assembler, find a way to represent a multiple alignment and parse it, column by column to identify locations with discrepancies.
4. **TILLING (method for finding "interesting" mutations).** Given a set of DNA sequences derived from a gene, map them back to the gene sequence (using the Smith-Waterman algorithm implemented in project 1 or some other alignment program), identify the reading frame and categorize any mutation between the original and the mutated gene into one of: premature stop, aa change, frame-shift, silent to a rare codon.
5. **Protein secondary structure prediction.** Write a program that takes a protein sequence and attempts to predict the secondary structure elements in this protein. This project will require you to use a set of proteins with known secondary structures to compute structure propensities for amino-acids, then use this information to predict the structure of other proteins.

6. **RNA folding.** Write a program that computes the secondary structure of an RNA molecule and output this structure in a standard structure format. The RNA folding is a relatively straight-forward dynamic programming algorithm.
7. **Phylogenetic tree construction using neighbor-joining.** Starting with a distance matrix between a set of sequences, compute and display the phylogenetic tree for these sequences using the neighbor-joining algorithm.
8. **Transcription terminator finder.** Write a program that takes as input a set of DNA fragments representing the regions downstream from a set of genes and identifies potential transcription terminators. The terminators are characterized by a hairpin loop (which can be identified with a simple dynamic programming algorithm similar to that for RNA folding) flanked by regions with large concentrations of As upstream and Ts downstream from the hairpin.
9. **Simple multiple aligner.** Implement a simple multiple alignment program, extending the Smith-Waterman algorithm implemented in project 1 to allow implementing the progressive alignment approach.
10. **Simple gene finder.** Implement a simple gene finder for bacteria, specifically find stop codons, identify in-frame starts, then evaluate the ORFs, based on codon preferences in the organism being analyzed.
11. **Microarray clustering tool.** Develop a tool for clustering the expression data from a microarray experiment in order to identify correlations between genes and phenotype. This project will require you to parse publicly available microarray data and to implement a simple hierarchical clustering tool (e.g. UPGMA).
12. **Operon prediction.** Search the set of genes in publicly available genomes and count how often pairs of genes occur nearby each other and in the same orientation as well as how often they occur in opposite orientations. Genes that are most often found next to each other, in the same orientation, are likely to be members of operons.
13. **Non-redundify a protein database.** Take as input a public protein database that may contain multiple copies of a same protein (e.g. the same protein from different organisms) and remove duplicates, keeping track of the duplicated information.
14. **Simple annotation pipeline.** Run a gene finder on a genome, identify a set of genes, then compare these against GenBank using blast to determine their probable function. This project will require you to learn how to use the gene finder Glimmer, and how to run batch Blast jobs against GenBank.
15. **Evaluate protein-protein interaction network.** Given a network of interactions between proteins (e.g. yeast 2-hybrid) compute several characteristics of this network, e.g. diameter, clustering coefficient, etc. Use these parameters to compare networks determined through different experimental techniques. This project will require you to build a graph data-structure then traverse it to compute several characteristics of the structure.
16. **Simulate the shotgun sequencing process.** Given the sequence of a genome, and several parameters (e.g. coverage, library size, average read length), simulate a shotgun experiment and output the resulting reads in the AMOS format.
17. **LIS clustering for MUMmer.** Given a set of MUMmer exact alignments, attempt to cluster them using the LIS clustering algorithm (the two-dimensional

chaining described in class) in order to identify candidates for extension using the full Smith-Waterman algorithm.

18. **Simulate a protein 2D gel.** A protein 2D gel separates proteins in two dimensions according to isoelectric point on one dimension and mass on the other. Given a genome and all the predicted proteins, generate a picture that simulates the image you would get from a 2D gel electrophoresis experiment.
19. **GUI development.** Develop a GUI for one of the tools described above, or for the code you wrote in project 1.