Computational Genomics
© Benny Chor and Igor Ulitsky
School of Computer Science
Tel-Aviv University
April 11, 2009

Computational Genomics: Assignment No. 2
Due May 1st (May Day, the International Workers' Day) in Igor's mailbox

General Guidelines: This assignment is part of the final grade in the course. It should be done *independently*, either individually or in pairs, without any help from others. Duplicated and copied works will be given zero grade. Using articles, books, or web sites is perfectly acceptable as long as you include the reference in your relevant answer. Exceptionally original answers will be rewarded with bonus points. If a questions requires a description of an algorithm, you must prove its correctness and analyze its time and space complexity.

Credit: Solve task 8, plus six questions out of the first seven for full credit. Solving all the questions will result in bonus points. **Task 8 can be submitted separately by May 10th**.

1. **Repeated prefix.** A substring α is called a *repeated-prefix* of string S if α is a prefix of S and has the form $\beta\beta$ for some string β. Give a linear time algorithm to find the longest repeated-prefix of an input string S.

2. **Longest inverted repeats.** In biological applications, we are often interested in inverted repeats, i.e., two substrings which are inverted copy of the other. For example, the string *abcdelmnedcblmnk* contains the substring *bcde* and its inverted repeat. The substring *lmn* appears twice, but not as inverted repeat. Give a linear time algorithm to find the longest inverted repeat in a string, in two versions:

   (a) When the two inverted copies may overlap.

   (b) When overlap between the two copies is *not* allowed.

3. **DNA contamination.** The Sequenced DNA of a certain organism may contain external contaminations, i.e., segments of foreign DNA coming from the technician or other known sources. Given a string S and a set of known possible contaminations $T_1, T_2, ..., T_k$, we are interested in finding all the substrings of any T_i that appear as substrings of S, and whose length exceeds a given threshold l. Give an efficient algorithm for this problem and analyze its complexity.

4. You are given k sequences. Let us denote by $l(i)$ the length of the longest continuous subsequence common to at least i sequences. Give an efficient algorithm to compute $l(2), ..., l(k)$.

5. Suppose you are given a rooted binary phylogenetic tree with n leaves labeled with species. Suppose also that distances $D[i,j]$ between all pairs of species are given. We wish to order the species in a linear order $p(1),...,p(n)$ so that (1) for every rooted subtree, its leaves appear consecutively in the order, and (2) the order minimizes the sum of neighbor distances. Show that the problem can be solved in polynomial time. Hint: use dynamic programming.

6. In the neighbor joining algorithm given in class, one has to maintain $\{r_i\}_{i=1}^{n}$, where $r_i = \sum_j D_{ij}$ is the sum of distances from taxon (or "super taxon") i to all the other taxa (or "super taxa"). Show how this can be done in $O(n^2)$ time in total, for all the $n-1$ iterations.

7. **Maximum parsimony and Vertex cover:**
   (a) Show that the vertex cover problem on undirected graphs (VC) is polynomial time reducible to the vertex cover problem on undirected, *triangle free* graphs (Δ-VC).

   (b) Consider the reduction showed in class from Δ-VC to BIG maximum parsimony. Let u and v be two "reduction strings" (each encodes an edge in the original graph). Suppose they are both connected to w, an internal node in a labeled tree T. The node w is labeled by some string from $\{0,1\}^n$. Show that if u and v do not share a 1 in the same position, than either $d(u,w) \geq 2$ or $d(v,w) \geq 2$, where d is the number of changes.

8. In this assignment, you will implement a pairwise alignment algorithm, followed by Neighbor Joining, in order to build phylogenetic trees. You program should have the following functionality:

   - **Reading a sequence file:** You program will read the sequence from a file named input.txt, which contains multiple sequences in a FASTA format (http://www.ebi.ac.uk/help/formats_frame.html). Basically, a FASTA file contains two lines for every sequence: a comment line (starting with a " >″) containing the name of the sequence and then several lines containing the sequence itself. Two input files with sequences, in several species, of the genes GBA (glucocerebrosidase, mutated in Gaucher disease) and SMC3 (structural component of the kinetochore complex) can be found in
     http://www.cs.tau.ac.il/~ulitskyi/cg/SMC3.fasta.txt and
     http://www.cs.tau.ac.il/~ulitskyi/cg/GBA.fasta.txt , respectively.
   - **Reading a substitution matrix:** Use the PAM250 substitution matrix which can be found in: http://www.cs.tau.ac.il/~ulitskyi/cg/pam250.txt. The * row and column contain the scores for gap. Use constant gap penalty with this score (basically -8 for a space).

- **Generating a phylogenetic tree:** Your program should use the Needleman-Wunch global alignment algorithm (that you learned in the course, possibly without crediting the designers :-) to compute the pairwise alignments between every pair of sequences in each input file separately (only the score of the alignment is required). The scores of the global pairwise alignments should be used as an input to the Neighbor joining algorithm, which will generate a phylogenetic tree. The tree should be written to a file out.tree in a NEWICK format (http://evolution.genetics.washington.edu/phylip/newicktree.html). There is no need to write the branch lengths. A sample tree (generated using ClustalW) is available in http://www.cs.tau.ac.il/~ulitskyi/cg/GBA.tree.txt.
- **Inspection of the trees** Use the DrawGram server (available at http://mobyle.pasteur.fr/cgi-bin/portal.py?form=drawgram) to draw your phylogenetic tree. Turn off the use of branch lengths. Visualize both of the trees you obtain for the two sample sequence files. What are the similarities and the differences between the two trees (note that the two files don't contain sequences from exactly the same species). Run ClustalW (http://www.ebi.ac.uk/Tools/clustalw2/index.html, use default parameters) on both files, and use DrawGram to view the trees you obtain (also turning off branch lengths). What are the similarities/differences between the trees obtained using your implementation and ClustalW?
- **Technical notes:**
  - You can use C,C++,C# or Java programming languages.
  - Readability and comments in your code will be a factor in your grade. There is no need to comment every line of code, but every file and every function that is not self-explanatory function should be explained.
  - The final project (including all source files) must be placed in you home directory in a directory named cgTree with a short documentation file, containing: (a) Short description of all the source files; (b) Detailed explanation on how to execute your program; (c) Listing of deviations between your implementation and what has been requested in the exercise;
  - Everything must be executable on the Linux machines in the school OR on a standard Windows XP. Indicate which one on your submitted exercise
  - Dont forget to give 705 permissions to the directory and its contents (chmod 705 *).
  - The documentation file has to also be printed and submitted in class. There is no need to print any source code.