Elements of Bioinformatics (autumn 2010)

Lecturers: Veli Mäkinen and Ilari Scheinin

Exercise 2

Tue 16.11, 16-18, C222, Esa Pitkänen

Choose any 5 assignments from below (each assignment gives 1 point, 5 points is maximum for each week; you can do more for better learning, of course).

1. **FASTA and BLAST.**

   Give an example of a good local alignment that will not be found with FASTA but will be found by BLAST using the same value for parameter k.

2. **Descending suffix walk.**

   Simulate the descending suffix walk algorithm given at the lectures on the strings $Q =$ `ACA` and $D =$ `CATACT`, to find out the longest common substring of Q and D.[1]

3. **MEMs, MUMs, and MAMs.**

   Use `MUMMER` software (`http://mummer.sourceforge.net/`) to learn the following concepts and to do the following tasks:

   a) Find all *maximal exact matches (MEMs)* of $Q =$ `ACA` and $D =$ `CATACT` of length at least 2.

   b) Find all *maximal unique matches (MUMs)* of $Q =$ `ACA` and $D =$ `CATACTAC` of length at least 2.

   c) Find all *maximal almost-unique matches (MAMs)* of $Q =$ `CACAC` and $D =$ `CATACTAC` of length at least 2, that is, maximal matches that are unique in one sequence (here D).

4. **Biodatabases and SQL I.**

   Write a python program that takes a gene name as user input, finds the corresponding Ensembl Gene ID using SQL, and outputs this ID back to the user.

[1] Note that the algorithm is described for *implicit suffix tree* a.k.a. *suffix trie*, which is like suffix tree, but each edge label spelling more than one symbol is replaced by a unary path of nodes linked by edges each spelling one symbol. These added nodes are called *implicit nodes* in suffix tree, whereas the real nodes are *explicit nodes*. Descending suffix walk works identically in suffix trie and in suffix tree, except that suffix links are only stored for explicit nodes in suffix tree. However, each implicit suffix link can be computed through following nearest preceding explicit suffix link. Such computation is no longer constant time per step, but can be shown to be *amortized constant time*, that is, the number of steps taken to compute *all* implicit suffix links encountered during the algorithm is bounded by the length in Q; for details, compare to the analyses of suffix tree construction algorithms to be given at the String Processing Algorithms course. For this assignment, you can assume that suffix links are available for both explicit and implicit nodes.

5. **Biodatabases and SQL II.**

   Write a python program that takes an Ensembl Gene ID as input, and prints out all the corresponding transcript and translation IDs.

6. **Biodatabases and SQL III.**

   Write a python program that takes a gene name as user input, and prints out the length of the gene.

7. **Combining SQL and sequence analysis.**

   Complete the template python program at `http://www.cs.helsinki.fi/u/vmakinen/elements10/ex2_7.py` to do the following task: Given input gene names, use SQL to query for the IDs of corresponding transcripts that also have a protein translation, read the corresponding entries from the FASTA file containing all transcripts as cDNA, convert the cDNAs into protein sequences, do pair-wise local alignment for each pair of proteins encoded by a different gene, and output the score of the best local alignment for each pair of genes.

   The motivation for the pipeline is for example a *gene expression study* that indicates a set of genes is active in a certain experimental setting, and one would be interested in finding out if they may share a common function based on sequence similarity.