Problem set

Cpt S 471/571: Computational Genomics

School of EECS, Washington State University

Spring 2019

Updated: Tuesday 9th April, 2019

1 Special instructions

- Homework problems will be posted here as the course progresses.

- "Active" problems are those which are currently due. "Retired" problems are those which are already past due.

- All problems are intended to be solved individually unless otherwise specified (as "collaborative").

- For "collaborative" problems, you are allowed/encouraged to discuss the questions and possible approaches with one or more student colleagues from *your class*[1]. If you do so, then please acknowledge the participants in the discussion(s) you had by simply listing their names as collaborators in your submission. However, **all** writing in the final answer should be entirely yours. Never show or share your solution in any form. Never look for solutions on the web unless the question itself points you to some specific web resources for guidance/information. If there is remote evidence to the contrary, the student will be subject to the academic dishonesty code.

- For those in Pullman, I encourage you to submit your homeworks as hardcopies in class, on the due date. If you cannot make it to the class, you can submit via Blackboard dropbox as backup. But that also will have to happen by the class time on the due date.

- For those of you located outside Pullman, please use the Blackboard dropbox.

- For all Blackboard submissions, only PDF format is allowed.

[1] Discussions are allowed only with the current batch of students in the class.

2 Active problems

Due date: April 18, 2019

5. (10 points) — collaborative

   A string s is said to be *periodic* with a *period* α, if s is α^k for some $k \geq 2$. (Note that the notation α^k means the string formed by concatenating k copies of α.) A DNA sequence s is called a *tandem repeat* if it is periodic. Given a DNA sequence s, provide an efficient suffix tree-based algorithm to determine if it is periodic, and if so, the values for α and k. Note that there could be more than one period for a periodic string. In such a case, you need to report the shortest period.

   For example, $acgacgacgacg$ is a tandem repeat with two distinct periods: i) $\alpha = acg$ and $k = 4$; and ii) $\alpha = acgacg$ and $k = 2$. Your algorithm should output the shortest period (which here is $\alpha = acg$).

   Hint: you can reformulate the problem into one of using LCA queries.

6. (10 points) — collaborative

   A non-empty string β is called a *repeat prefix* of a string s if $\beta\beta$ is a prefix of s. Give a linear time algorithm to find the longest repeat prefix of s.
   Hint: Think of using LCA queries.

3 Retired problems

1. *(2 points; Individual)* Give the *reverse complement* of the following DNA sequence: *accgtagccggatatac*.

2. *(8 points; Individual)* This problem is a bit of search expedition. For this question, you will use online Genome resources.

   The NCBI GenBank is one of the most accessed databases in bioinformatics. Biologists and bioinformaticians use this site and all its resources like "Google" for the internet — i.e., searching for genes, browing through genomes, understanding what genes express what proteins, and so on.

   As part of the Genbank, NCBI maintains the Human Genome Resources, and its URL is `http://www.ncbi.nlm.nih.gov/genome/guide/human/`.

   For this assignment you will use primarily the NCBI Human Genome Resources to answer the following questions:

   i) By browsing through the Human Genome Resources page, please collect and report the following statistics:
   a) How many chromosomes are there in the human genome?
   b) What is the length of every chromosome (measured in nucleotide base pairs)? This is same as the number of characters along a DNA sequence. We will abbreviate "base pairs" as "bp".
   c) What is the number of genes in each of those chromosomes?

   Report answers to the above questions in the form of a simple table - i.e.,

Chromosome id.	Chromosome length (in bp)	Number of genes
Chr. 1		
Chr. 2		
...		
Full Genome (total)		

   The last row should simply be the sum of the respective columns.

   Using the table constructed above, construct two charts (using any of Excel, Matlab, GNUplot, etc.) — one that lists the Chromosome ids along the X-axis (i.e., 1, 2, ...), and along the Y-axis, their respective chromosome lengths (in bp). The second chart plots the number of genes along the Y-axis. It will be nice if you juxtapose both charts or even overlay them on top of one another, so that its easy to observe any relation (if any) between the length of a chromosome and the number of genes on it.

Your solution should basically have this table, and two charts.

Hint: Please spend some time browsing this website to get used to the interface and to understand what information it has to offer. You may find the NCBI Map Viewer visual interface helpful to browse each chromosome.

ii) As the second part to this assignment, I'd like you to search for a specific gene. You are welcome to use Google for this, to guide any of your initial searches. (But beware that not all websites may give you the correct information.) For a search to be correct, you should see consistent evidence from the NCBI GenBank webpage: `https://www.ncbi.nlm.nih.gov/genbank/`.

Find the main breast cancer gene in humans. There will be a few genes you will find but when you search one or two genes should come to the top of the list. Your task is to find the name of this gene, search for this gene in the NCBI GenBank, and collect the following basic statistics about it:

The name of this gene, the human chromosome in which it is present, the exact coordinates in which it is present (i.e., from where to where), the gene's length (in bp); and its main protein product's length (in amino acid residues).

Note: There is no need to print out the gene sequence itself as that could be quite long. But at least be sure that you know where from on the NCBI page to find the sequence information if need be.

3. (8 points) - Collaborative

For global alignments, there can possibly exist more than one optimal global alignment path between a pair of strings. Give an efficient algorithm for *counting the number of such co-optimal paths* between two strings s_1 and s_2 of lengths m and n respectively. Your algorithm's runtime should not exceed $O(m \times n)$, which also implies that you cannot afford to explicitly enumerate all possible optimal paths and then count them, since there could be exponential such paths.

4. (7 points) — collaborative

How will you modify the linear space Hirschberg technique to work for optimal *local* alignment computation in linear space (both opt. local alignment score & traceback path)? Explain the main steps of your new (modified) algorithm. No need to expand on parts that are identical to the version of the algorithm discussed for global alignment computation in class.