CSCI3220 Algorithms for Bioinformatics 2018-2019 First Term
The Chinese University of Hong Kong

Assignment 2

Due date: 28 October 2017 (Sun) 23:59 Full mark: 100
Expected time spent: 3-6 hours

Aims:
1. Get familiar with the details of some data structures and algorithms for short read alignment and sequence assembly.
2. Get deeper understandings about the relationships between different data structures.
3. Establish scientific rigor by cross-checking your results using multiple means.

Description:
In this assignment, you will manually construct different data structures for short read alignment based on a sample sequence. For some structures, you will use multiple methods to construct them such that you can cross-check your answers. You will then use BWT to search for a query from the sample sequence and answer a question about a property of BWT. Finally, you will write a program to construct a directed graph from short reads and find Eulerian path(s) from it.

Questions:

1. This question is about various data structures for short read alignment. You are given the DNA sequence s=`GTAACTGTAGTG$`.

   (a) Construct the suffix trie of s. (10%)

   (b) Construct the suffix tree of s. Each non-root node should be labeled with the starting and ending positions of the corresponding sub-sequence in s (first position counted as 1). If the sub-sequence appears multiple times in s, use the starting and ending positions of the first appearance. (5%)

   (c) Using the suffix tree constructed in Part b, construct the suffix array of s. Show clearly in every step how you use the suffix tree to construct the suffix array. (10%)

   (d) Use another method to construct the suffix array without assisted by the suffix trie or suffix tree. Show your steps clearly. Do not forget to compare your results in Parts c and d. (5%)

   (e) Construct the BWT of s, denoted as b, using the suffix array. Show every step clearly. (5%)

   (f) Use another way to deduce the b from s without assisted by the suffix trie, suffix tree or suffix array. Do not forget to compare your results in Parts e and f. (5%)

   (g) Use b to determine the starting positions of all appearances of query q=`GTA` in s, if there is any. Construct any auxiliary data structures of the FM index that you need. Show every step. You should not use the input sequence s or its suffix trie/tree directly. (15%)

   (h) Suppose you are given a random DNA sequence drawn from the set of all possible DNA sequences of length n with equal probability for each of the 4^n sequences. Now you append the symbol `$` to the end of the sequence and perform BWT. Does the end-of-sequence symbol `$` have equal probability to appear in each of the n+1 positions in the BWT output? Explain why. (5%)

2. Write a computer program called **EP** in C, C++, Java or Python for finding an Eulerian path from the graph constructed from a set of input DNA short reads. Your program should take a list of DNA short reads in the form of upper-case strings from stdin (**not** from a file), each on a different line. The input ends with an empty line.

   You can assume that the input list contains **distinct** DNA sequences **all of the same length** in no particular order. You do not need to check for errors in the inputs.

   Your program should then construct a directed graph based on the input short reads. Specifically, each read (of length *k*) should become an edge of the graph connecting a node representing the length *k*-1 prefix of it to a node representing the length *k*-1 suffix of it. For example, the read `CAT` should become an edge connecting node `CA` to node `AT`.

   Your program should then find an Eulerian path and output it to the screen (i.e., stdout) in the form of a text string. For example, suppose an Eulerian path involves the traversal from node `CA` to `AT` and then from `AT` to `TT`, the output should be the text string `CATT`.

   If there are multiple Eulerian paths in the graph, you can output any one of them. If the graph does not contain an Eulerian path, your program should output nothing.

   The non-comment portion of your program is expected to contain no more than 250 lines of code.

   Here is an expected screen shot when a Java program is run on an example from the lecture notes:

```
>java EP
ACA
ATA
ATT
CAT
TAC
TAG
TAT
TTA

TATTACATAG
```

   Your program will be graded based on i) whether it can be compiled/run successfully, ii) whether it follows the input/output formats as specified above, iii) its accuracy on a number of test cases and iv) whether the program is well-documented with appropriate comments added to explain the meaning of the code. In view of the large size of our class, for easy grading, you may get zero score if your program cannot be compiled or it does not conform to the input/output formats specified. (40%)

   [Optional] Output all Eulerian paths in the graph. Each Eulerian path should occupy a separate line in the output in any order.

   Here is a screen shot when a sample Java program was run on the same example from the lecture notes:

```
>java EP
ACA
```

```
ATA
ATT
CAT
TAC
TAG
TAT
TTA

TACATATTAG
TACATTATAG
TATTACATAG
TATACATTAG
```

If your program outputs multiple Eulerian paths, the first one will determine your score for the non-bonus part (40% of the assignment score) while all the output paths together will determine your score for the bonus part. Therefore, if you decide to take this bonus part, please make sure that the first Eulerian path you output is correct. (bonus 10%)

Submission:

For the programming question, you should first submit your program to our **online judge system**. Please see tutorial notes set 1 for explanations.

For the non-programming question, give all your answers in a single file named <ID>_asmt2.<ext>, where <ext> is either doc, docx or pdf. We prefer pdf files because it has better portability. Then put your files for both the programming and non-programming questions in a zip file named <ID>_asmt2.zip and submit it to Blackboard.

Both your written and source code files should contain the following header. Contact Kevin before submitting the assignment if you have anything unclear about the guidelines on academic honesty.

```
CSCI3220 2018-19 First Term Assignment 2

I declare that the assignment here submitted is original except for source
material explicitly acknowledged, and that the same or closely related material
has not been previously submitted for another course. I also acknowledge that I
am aware of University policy and regulations on honesty in academic work, and
of the disciplinary guidelines and procedures applicable to breaches of such
policy and regulations, as contained in the following websites.

University Guideline on Academic Honesty:
http://www.cuhk.edu.hk/policy/academichonesty/

Student Name: <fill in your name>
Student ID  : <fill in your ID>
```

Marking Scheme and Notes:

1. Remember to submit your assignment by 23:59pm of the due date. We may not accept late submissions.

2. For the written part, if you submit multiple times, **ONLY** the content and time-stamp of the **latest** one before the submission deadline will be considered. For the program, the version submitted to the online judge system with the highest score will be graded.

University Guideline for Plagiarism

Please pay attention to the university policy and regulations on honesty in academic work, and the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details can be found at http://www.cuhk.edu.hk/policy/academichonesty/. With each assignment, students will be required to submit a statement that they are aware of these policies, regulations, guidelines and procedures.