COMP 7295/8295 – Course Project

Due: 12/05/2017

The goal of the project is to create a Python toolkit that identifies microbes in a metagenomics sample. At the core of your methodology is a fast substring matcher based on the FM-index, which we discussed extensively in class. To ensure that you will complete this project successfully, we will break it down into several concrete tasks.

Here are the main tasks:

1. Review and experiment with the code that implements the FM Index.
2. Create a utility function that generates a random genome and save it to a file.
3. Create a utility function that generates random reads from a given genome and save the reads to a file. The input is a reference genome, read length and a coverage number. Reads are random substring of the reference genome. Example, genome has length 1000; read length is 100; and coverage is 3. Then, you must generate 30 random reads. Coverage is defined as: **number of reads** multiplied by **read lengths** and divided by **genome length**.
4. Create a Python class called Aligner, which consists of an FM index. The process consists of two distinct phases. In the first phase, you build indexes for all reference genomes. In the second phase, you align reads to the genomes (using the indexes) to figure which reads belong to which genomes. The first phase is done only once for a set of reference genome. The second phase can be repeated multiple times for new reads.

   The Aligner class has the following methods:
   a. **BuildIndexes**, which takes as input a list of reference genomes and create an FM index for each genome. The input should be a directory that consists of individual files. Each file stores a reference genome in the FASTA format.
   b. **AlignExactly**, which takes as input a read and returns the id of the genome that most likely contains the read. You should store genome ids when you process reference genomes in FASTA files when you build indexes.
5. Create 3 synthetic datasets (without read errors). All reference genomes should have lengths between 50,000 and 1,000,000. Read length should be between 200 and 300.
   a. **Dataset 1** consists of 2 reference genomes and 3 sets of reads. The first set of reads consists of 1x coverage of genome 1 and 2x coverage of genome 2. The second set of reads consists of 2x coverage of genome 1 and 4x coverage of genome 2.
   b. **Dataset 2** consists of 8 reference genomes and 3 sets of reads. Generate your 3 read datasets in a similar way as you did for dataset 1.
   c. **Dataset 3** consists of 64 reference genomes and 3 sets of reads. Generate your 3 read datasets in a similar way as you did for dataset 1.

6. Report results of AlignExactly on these these 3 datasets. For each dataset, report (i) running time, (ii) true positive rate and (iii) false positive rates.
7. Modify your function that generate reads to introduce few errors. Your reads should have 1% of errors. In other words, a read of length 200 should have 2 errors at random positions.
8. Create 3 synthetic datasets with 1% read error. Other than errors, our synthetic datasets should be generated in the same way was the previous 3 datasets.
9. Report results of AlignExactly on these these 3 datasets. For each dataset, report (i) running time, (ii) true positive rate and (iii) false positive rates.
10. Create a new method for the Aligner class called **AlignApproximately**. This method is similar to AlignExactly. The difference is that in AlignExactly, the starting search position starts at the end of the input read. In AlignApproximately, the starting search position starts at a random location and stops when there is match to only one genome.
11. Report results of AlignApproximately on these these 3 datasets. For each dataset, report (i) running time, (ii) true positive rate and (iii) false positive rates.
12. BONUS (extra 10%): implement a new method that can outperform AlignApproximately either in (i) running time or (ii) accuracy in terms of true positive rate and false positive rate.
13. BONUS (extra 10%): report the results of AlignExactly and AlignApproximately on real datasets. The TA will provide a few real datasets.

How to turn in your project.

- Place your code and data in a DropBox folder. Send me (vphan@memphis.edu) the link to that folder. Share your folder with me as soon as you start working on your project and writing your project report.
- Write a report that addresses each task (listed above). For each task, indicate (i) if you successfully complete it; (ii) if not, indicate what you have not been able to do; (iii) the percentage of the code created by you and the percentage of the code created by someone else. You must indicate very clearly the sources and which parts of your code was written by you and which parts were not. Each function that is not written by you must has a comment on top of it indicating the author(s) of that piece of code.
- The report must include the results of AlignExactly and AlignApproximately on the synthetic datasets as listed above.
- Your report should be written progressively. Do not wait until the last week to do the project. You won't be able to finish it. Do not wait until the last week to write the report. Update the report as soon as you complete each task.
- Although this is an individual project, you should collaborate with your classmates. If you let your classmate use your code, insist that he or she acknowledge it in his/her report. If you use someone's code (your classmate's or somebody's else), you must

acknowledge it very clearly. If you use someone's code and fail to acknowledge it, your project will get a score of 0.

- Project grade is based on (i) how much you accomplish, (ii) how much you use someone else's code, (iii) how much you help your classmates, and (iv) how properly you acknowledge someone's contribution and report the results.