1 Overview

In this project, you will need to implement a bioinformatics tool. The objective is to gain experience in developing quality tools which are efficient and accurate.

The problem is as follows. You are given a reference DNA sequence, which is usually very long (say the whole genome of some species). Then you are given a large number of DNA sequences, each of which is much shorter (say 30 nucleotides). Presumably, these shorter sequences (called reads) are the output of a modern DNA sequencing machine. Thus, each read can be considered as a (sometimes inexact) copy of some short segment of the reference sequence. Your program is, given the reference sequence and the reads, find where the reads are originated in the reference sequence. Remember that this is not so easy, due to the following main difficulties: the reference sequence is long, the number of reads is large, and the reads are noisy.

There are a lot of more complexity in developing a real working tool for this problem. In this project, you are going to focus on the above model. Your tool needs to find reads where there are a small number of substitutions and/or single base insertions and deletions. You may also assume the number of such copying errors is relatively small in each read. However, it may be considered for bonus points if your program is robust enough to handle more general case (say multiple base insertion/deletion). Note that there may be difference sources for the noise. For now, let us assume the errors are due to the sequencing. That is, we assume the reads are indeed taken from the reference sequence but then some noise is imposed on the reads.

2 Tasks

You need to write a program tool, which takes a reference sequence and a large number of reads and find out where these reads originate in the reference. The reference sequence is more or less constant, which means you can (and to speedup computation, I believe you should) perform some types of preprocessing of this reference sequence for faster searching. The reads, on the other hand, are much more variable. Thus, your program should be able to handle any set of reads that the program has not seen before. Now, here is what I suggest you to do.

1. Plan. You need to decide a general strategy on how to perform fast searching. For example, you can use suffix tree or suffix array. Or you can use some types of hashing. You are free to choose whatever techniques you want to use. In any case, remember that your method should be able to handle the noise (substitution and indels) and also fast.

2. Coding and testing. You need to implement your method. You can use any programming language you like. Keep in mind that your program needs to be fast. Test your code early.

I will post several simple examples (with the reference genome and a small number of reads). This may help you to develop and debug your program. Later, I will post some larger test data so that you can evaluate the performance of your program.

3 Hand ins

A written report including the following: (a) Problem description and overview, (b) description of your method, (c) implementation and simulation results. This includes results (running time, how many of the reads are properly located, how many substitutions and insertions/deletions found) on the provided test data and any data you simulate yourself. Please format your outputs to resemble the provided simulation format. This helps me to automatically compare your results with mine.