Programming Assignment 3 - 10 kilobase Sequence Assembly

BIOINFO M260

Due: February 22nd at 11:59 pm

This programming assignment is designed to teach you about sequence assembly.

Overview

In this assignment, you are given paired-end reads from an unknown donor sequence, and 10 percent of the reads are generated randomly to mimic contamination with another genetic source. These reads are formatted as two 50 bp-long ends, which are separated by a 90-110 bp-long separator.

Your job will be to reconstruct contigs–the longest sequences of the donor that you are confident that you have assembled correctly. The output format is simple, and described on the cm124 site. The approximate length of the donor sequence is 10,000 bases.

Starter Code

Starter code for the project is available at `https://github.com/michaelbilow/BIOINFO_M260B`. Remember to start a new branch so you can pull in any changes that I make to the code. Tou should read the content of PA3, and see if you can understand what it is doing. You should also look to see where your input/output is going to go. This will generate an alignment file in the data folder from which it was executed.

Download the datasets from the cm124 site, and edit the functions in `simple_de_bruijn.py`.

This code takes reads, generates a spectrum from those reads, and follows an Eulerian path through the deBruijn graph to generate contigs.

I/O Details

`https://cm124.herokuapp.com/ans_file_doc` should handle most of your questions on reading and writing output.

Spectrum

To generate the de Bruijn Graph, you need to first generate a spectrum from the reads; the spectrum is the set of shorter sequences present. You will want to generate a spectrum with a sufficiently large number of bases, so that most of the sequences in the spectrum are not covered.

de Bruijn Graph

The de Bruijn graph using a spectrum of size `k` is by taking each read, and drawing a directed edge from the chunk of the read from position `j` to position `j+k`, to the chunk from `j+1` to position `j+k+1` for all `j`. Some filtering of this graph needs to be done in the case that the errors have reads, and to filter out data from the "contaminating" sequence.

Other Assembly Algorithms

There are other assembly algorithms (for example, an $O(N^2)$ greedy algorithm) that only work in easy cases. As this is an easy case, this algorithm will work (and in fact, it will likely work better than the deBruijn graph algorithm you are given starter code for), but it is not necessary to implement this algorithm to achieve a full-credit score.