Homeworks 5A&B

Due: Please read

1 Reading

Sundquist et al. *Whole-Genome Sequencing and Assembly with High-Throughput, Short-Read Technologies.* 2007 PLoS One.
Zerbino and Birney. *Velvet: Algorithms for de novo short read assembly using de Bruijn graphs.* 2008 Genome Research

2 Overview

Your job is to simulate the process of sampling random pieces of a long piece of DNA and assembling the fragments to reconstruct the original sequence. In the read generating phase, you will need to create a set of random pieces cut from a given DNA sequence. The number of reads generated should follow the Lander-Waterman statistics given the mean read length, the size of the genome, and the coverage. Given these fragments, you will attempt to reconstruct the original sequence and compare your reconstruction with the original. Reads should start at a random index on the genome – you may handle edge effects (i.e. if genome size is 1000bp and you generate a read starting at base 995bp) in a manner of your choosing.

The real-world problem is very challenging because technical factors and uncertainty affect the accuracy of the assembly process. In the scope of our class, we assume that all random pieces have the same orientation.

The following tasks can be used to assemble the random pieces into longer contigs. Note that it is not always possible to reconstruct the original DNA sequence.

Your job is to *propose a detailed algorithm* (ie. write a plan) for above steps and *implement the genome assembly process.* Your program should read in a file containing a DNA sequence (or optionally a set of reads) and print the assembly to a file. Experiment with your program by changing the parameters and *discuss your findings.* You will use an Eulerian path algorithm as your approach, please do not implement Celera or PHRAP. Implement some variation on Velvet and/or Sundquist et al (*Reading*, above).

3 Parameterization

Your program should accept a number of parameters on the command line. The TAs should be able to change these without modifying the source code of your program.

- a, Coverage: The number of genomes that your program should fragment and sample from.
- l, Mean Read Length. Actual reads should vary in length following a normal distribution with a standard deviation of 3.
- k , length of k-mer fragments (see Readings)

- e, error rate in $[0, 1]$. This is the chance of a sequencing error occuring at any given base-pair in the read. Due to the infinite sites assumption, treat all errors as simple mismatches (i.e. there are only two possible states at any given base).
- $r \in \{0, 1\}$, Reverse Complement: if 1, put all reverse complements of the reads into the input; if 0, do not.

4 I/O Specification

Input

Read a genome from a FASTA file format OR read a (multi)set of premade reads, in which case l, r are not specified. You can specify a command-line option for this.

(The benefit of inputting premade reads, or an initial seed value for the psuedorandom number generator, is to allow debugging.)

For the first week, you should test your program on the first 10000bp of the artificial bacterial genome we gave you for RNAseq and see how it goes. By the second week we will have provided you with reads from a larger genome, which you are to assemble and turn in – part of your grade will be how well the assembly matches this genome (within the capacity of the parameters used to generate the reads).

Output

The assembler should output the contigs to a file. They should be newline delimited. If a genome is used as input, your program should also be able to output the randomly generated reads to a file as newline delimited strings.

Additionally, we require that you programmatically generate two visualizations on completion of the assembly: (1) the de Bruijn graph you generate, with the Eulerian path(s) annotated, preferably using color; (2) a graph of the contigs in which the overlapping DNA fragments which constitute each contig are shown and the contigs are separate "islands". These visualizations should help you recognize problems with repeats, et cetera.

I highly recommend that you look further into Graphviz for (1). A very good intro to Graphviz is at `http://linuxdevcenter.com/pub/a/linux/2004/05/06/graphviz_dot.html`
Once you've written your Graphviz file then `dot graphviz.file -Teps -o output.eps`
Outputs to eps, which can be put into LaTeX, or via `epstopdf` into PdfLaTeX. For (2), you can use your rectangular skills that you've picked up from dot plots.

5 Logistics

Due Tuesday March 11 is (1) your algorithmic plan (typeset), (2) some semi-functional code, and (3) a sample output from the first 10000bp of the artificial bacterial genome for some reasonable parametrization. The quality of your output will not be graded at this time, and in fact you might assemble better contigs in your final version.

This first week check-in is graded out of 20 points, and students who complete the above will likely get all possible points. The remaining 80 points are assessed on the final version.

Due Thursday March 20 is (1) completed code; (2) a set {reads, assembly, de Bruijn graph, and contig map} for one run of the assembler on the first 10000bp of artificial bacterial genome with settings $a = 12$, $r = 0$, $l = 50$, $e = .01$; (3) a set {assembly and contig map} for a run of the assembler on the reads we provide in the second week with settings $a = 12$, $r = 1$, $l = 50$, $e = .01$ (we would ask for the de Bruijn graph except it will probably be too large; if you cannot create a contig map either that is okay); (4) estimate the value of the parameter a (genomic coverage) that is necessary to get an assembly with c contigs other parameters set to values given in (2); (5) lastly, in a brief write-up:

- give a summary of your experience with playing with parameter values on the reads we provide
- summarize any changes from your first week algorithmic plan
- describe how you corrected errors
- suggest reasons why the reads from the genome we gave you in the second week is more difficult to assemble
- calculate the assembly percent coverage of the 10000bp from part (2), in a brief writeup