Adam Weiner
CMPB 5002 - Hajirasouliha
11/07/19

CMPB 5002 Homework 2

Programming Questions

Problem 1-1.

A dot file of full graph and fasta file of edges can be built by changing the `reads_fn` variable to a file name of choice at the bottom of the `assembly.py` file. Nodes connecting non-branching edges are collapsed to create a condensed graph. Edges in the output fasta file are formatted so that each edge and its average coverage are separated by a comma in each line. Coverage in this graph was defined as the number of reads per 55 bp window. These non-simplified graphs produce the output files with the suffixes `_normal_db.dot` and `_normal.edges.fasta`. The dot file can be converted to a png file with the command `dot -Tpng s6_normal_db.dot > s6_normal_db.png`. The graph produced for `s_6.first1000.fastq` can be seen in Figure 1. Note that the process of condensing the graph is time intensive and scales polynomially with the number of edges in the graph; therefore, even though building a condensed de Bruijn graph takes approx. 1 sec for `s_6.first1000.fastq`, it is estimated to take approx. 24 hrs (exact time depends on machine) for `MG1655-K12.fasta`. While run-time increases polynomially here, the use of dictionary and Counter objects means that the memory requirements scales linearly (less than 1 GB of RAM used for all samples).

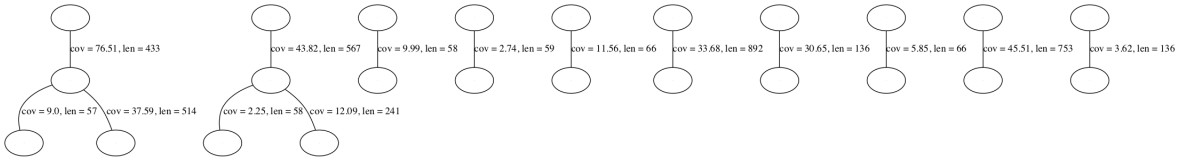

Figure 1: Condensed De Bruijn graph for reads in `s_6.first1000.fastq`. Edge lengths and average coverage are denoted by "cov" and "len", respectively.

Problem 1-2.

(a) Tips were removed in the graph by traversing through all nodes in the condensed graph and removing any edges from branch nodes (forks) that did not rise above specified coverage (`tip_cov`) and edge length (`tip_len`) thresholds. The user can specify `tip_cov` and `tip_len` for the arguemnts in functions `plot_db_tip_removal` and `fasta_edges_tip_removal`; these parameters were set to 10 and 100, respectively, in all reported analysis. The graph was intentionally not condensed again after tip removal in order to highlight the specific locations where edges were removed from the `_normal_db` file. All tip-less graphs produce the output files with the suffixes `_tip_removal.dot` and `_tip_removal.edges.fasta`. The tip-less graph produced for `s_6.first1000.fastq` can be seen in Figure 2. This

approach allowed us to eliminate low-coverage branches from the two leftmost subgraphs; however, this tip removal method did nothing to eliminate the low-coverage one-edge subgraphs on the right-hand side of Figure 2.

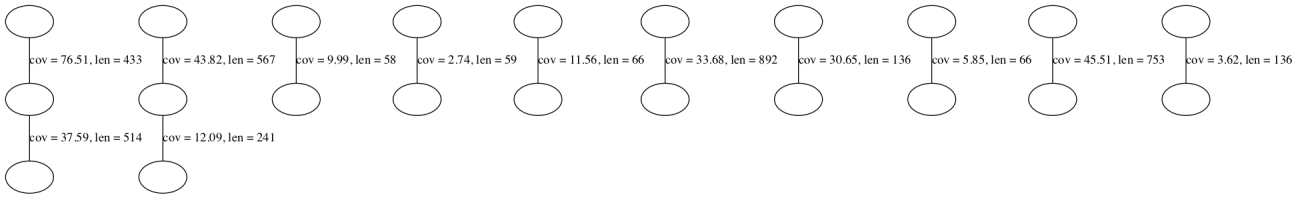

Figure 2: Tip-less De Bruijn graph for reads in `s_6.first1000.fastq` with `tip_cov` = 10 and `tip_len` = 100.

(b) Edges in the graph that were short or had low coverage were determined to be of low quality and thus were discarded. All edges in these "high quality" graphs must have edge lengths greater than the user specified `min_len` and `min_cov`. The graph was intentionally not condensed again after removing low quality edges in order to highlight the specific locations where edges were removed from the `_normal_db` file. All high quality graphs produce the output files with the suffixes `_high_quality.dot` and `_high_quality.edges.fasta`. The tip-less graph produced for `s_6.first1000.fastq` can be seen in Figure 3. This approach allowed us to eliminate all of the low-coverage and short edges in the graph – not just the tips off larger subgraphs. The elimination of the low quality one-edge subgraphs seen on the right-hand side of Figure 2 is a significant advantage of using global coverage and edge length thresholds instead of constraints restricted to branch points (tips).

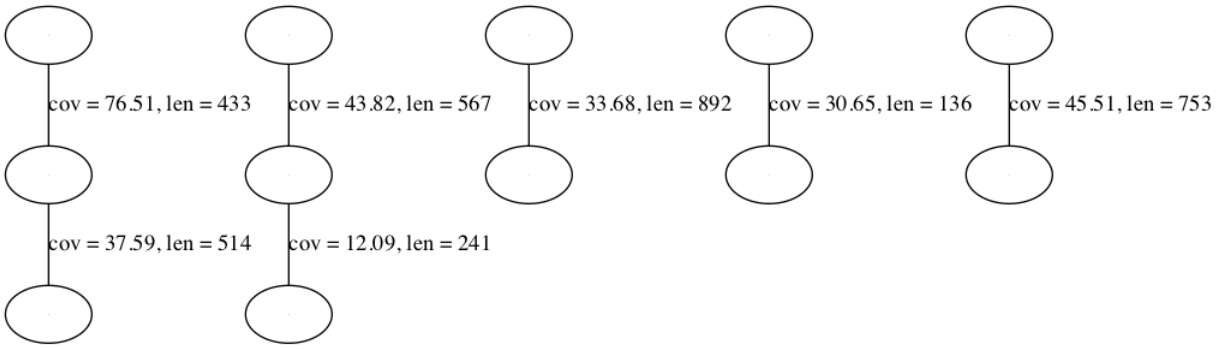

Figure 3: High quality de Bruijn graph for reads in `s_6.first1000.fastq` with `min_cov` = 10 and `min_len` = 100.

Problem 1-3. Web-based assembly tools from the Center of Genomic Epidemiology (https://cge.cbs.dtu.dk/services/) were used to obtain contigs from the `s_6.first1000.fastq` file. The first assembly method was a Velvet-based approach that cross-referenced a database of

bacterial reference genomes in order to help assemble short genomes that might be non-human (doi: 10.1128/JCM.06094-11). The second assembly method was a bareboanes SPAdes approach similar to the one covered in class (https://doi.org/10.1007/978-3-642-37195-0_13). After generating contigs for both assembly algorithms, their performances were evaluated by QUAST (http://quast.bioinf.spbau.ru/). The QUAST reports for the Velvet and SPAdes assemblers can be seen in the PDF files titled `Velvet_report.pdf` and `SPAdes_report.pdf`, respectively. Since the genome for `s_6.first1000.fastq` was reduced to 1 kb, both assemblers were able to reduce all the reads into a single contig of length 1 kb. While these two assembly algorithms behaved identically in this example, they are fundamentally different and would generate unique contigs for a larger genome such as that in `MG1655-K12.fasta`; however, assembling a genome of said size was not supported by the CGE web-server.