Attached is a list of the projects for the course. Please email me (christinaboucher@ufll.edu) the name of your project members and your selected project (1, 2, 3, or 4) by September 8, 2017 (this is also on the website). After September 8th, I will just assign the remaining students a group and project. Only one project member needs to send the email but they should cc the other team members.

There should be 4 people for each group (no more or less).

There is a maximum of 3 groups for each project. The projects will be assigned on a first come, first serve basis.

Project 1: Reusing Colors in Rainbowfish

Rainbowfish is a succinct representation of the colored de Bruijn graph. It compactly stores the colors of the graph. The storage of the colors of the graph can be further reduced by reducing the number of colors used. In order to do this we would like to reuse colors in the construction of the graph. Here is one sketch of a potential color-reuse algorithm: for each read R in turn, first tentatively insert into the graph G all the k-mers from R assigning them some special temporary color t, and make a list of all the colors of their neighboring k-mers and themselves if they're already in the graph (remembering each k-mer can have several colors, and if we're inserting a k-mer from R that's already in the graph then we just add t to its list of colors). If there's some color c that's already used in the graph but which no neighboring k-mer has or k-mer from R already has, then we undo our insertion of the k-mers from R with color t, and re-insert them with color c. Only if the neighboring vertices and k-mers from R already have every color currently in the graph do we introduce a new color x, undoing our insertion of the k-mers from R with color t and reinserting them with the new color x. For this project you will implement this color-reuse algorithm in Rainbowfish and determine the amount of memory that is saved.

Project 2: Succinct DBG for AMR

Antimicrobial resistance (AMR), or more commonly referred to as antibiotic resistance, is an important issue that potentially will have a severe impact on public health. Antimicrobial resistant (AMR) genes are the genomic elements that make bacteria resistant to antibiotics. Recently, databases have been developed to keep track of all known genes. The goal of this project is to put together a computational pipeline that takes as input reads that aligned to the AMR database, and reads that did not align from metagenomics dataset, and then uses the AMR-aligned reads as a "seed" and attempts to "extend" the alignment with partially-overlapping unaligned reads. Lastly, these regions could be annotated to identify flanking region. This would give us interesting information about the genomic context of the AMR gene, such as host bacteria or plasmid (or other nearby AMR genes that may be "traveling" together).

The motivation for this project is that currently we do not have an idea of the genomic context of antimicrobial resistant genes within metagenomic data—we know that they're somewhere

but we don't know which bacteria/plasmids are carrying them, or whether they're chromosomal or "mobile". This information is very important biologically because AMR genes that are on mobile elements are more likely to be passed horizontally between different bacteria, which is a risk factor for AMR dissemination. Alternatively, an AMR gene that is on the bacterial chromosome and NOT flanked by any mobile elements is very unlikely to be transmitted horizontally, and thus is "lower risk". This is basically a way for us to start to understand the risk associated with different AMR genes in a metagenomic sample. The goal of this project is implement the following algorithm in a memory and time efficient way so that the AMR genes can be associated with specific bacteria and/or plasmids.

Project 3: Dynamic DBG: Adding Dynamic Vertices

Bioinformaticians define the kth-order de Bruijn graph for a string or set of strings to be the directed graph whose nodes are the distinct k-tuples in those strings and in which there is an edge from u to v if there is a (k + 1)-tuple somewhere in those strings whose prefix of length k is u and whose suffix of length k is v. These graphs have many uses in bioinformatics, including de novo assembly, read correction and pan-genomics. The datasets in these applications are massive and the graphs can be even larger, however, so pointer-based implementations are impractical. Researchers have suggested several approaches to representing de Bruijn graphs compactly, the two most popular of which are based on Bloom filters and the Burrows-Wheeler Transform, respectively. In a recent paper entitled “Fully Dynamic de Bruijn Graphs” a new approach to the storage is described that is based on minimal perfect hash functions. It is similar to using Bloom filters but has better theoretical bounds when the number of connected components in the graph is small, and is fully dynamic: i.e., we can both insert and delete nodes and edges efficiently. The goal of this project is to implement the approach in this paper for dynamic vertices.

Project 4: Dynamic DBG: Adding Dynamic Edges

Bioinformaticians define the kth-order de Bruijn graph for a string or set of strings to be the directed graph whose nodes are the distinct k-tuples in those strings and in which there is an edge from u to v if there is a (k + 1)-tuple somewhere in those strings whose prefix of length k is u and whose suffix of length k is v. These graphs have many uses in bioinformatics, including de novo assembly, read correction and pan-genomics. The datasets in these applications are massive and the graphs can be even larger, however, so pointer-based implementations are impractical. Researchers have suggested several approaches to representing de Bruijn graphs compactly, the two most popular of which are based on Bloom filters and the Burrows-Wheeler Transform, respectively. In a recent paper entitled “Fully Dynamic de Bruijn Graphs” a new approach to the storage is described that is based on minimal perfect hash functions. It is similar to using Bloom filters but has better theoretical bounds when the number of connected components in the graph is small, and is fully dynamic: i.e., we can both insert and delete nodes and edges efficiently. The goal of this project is to implement the approach in this paper for dynamic edges.