Bioinformatics Homework 2.1

De Bruijn Graph from K-mer Polymers

Problem Statement: Given a set of k-mers, construct the adjacency list of the corresponding De Bruijn graph, which *could* then be used to form a De Bruijn sequence by finding an Eulerian path in the respective graph.

Algorithm Description:

```
  KmerEdgeList(Kmers)
      n  <- | Kmers |
      // Adjacency list
      DB <- graph where every k-mer in Kmers is has edge between
      its prefix and suffix of size k-1. Graph resulting from
      gluing all nodes in DB with identical labels

      // Iterating through each k-mer
      for every integer i between 0 and n-1
             prefix = first k-1 characters of the kmer
             suffix = last k-1 characters of the kmer
             add edge in DB from prefix to suffix

// Returning the finalized adjacency list
      return DB
```

This algorithm takes a set of k-mers, and creates a representation of the graph that could be used to do string reconstruction using the Eulerian Paths algorithm. To create the graph, we take each k-mer and add an edge in our graph from the node representing the first k-1 letters of the k-mer to the node representing the final k-1 letters in our k-mer. The resulting edge represents the full, original k-mer. By finding an Eulerian path on the resulting graph, we can ensure that all the k-mers will be used in the final string resulting from the path.

Time Analysis: Since this algorithm goes through each k-mer (O(n)), and adds a substring of those k-mers to a graph (O(k)), this algorithm runs in O(nk) time. Since we are only adding one string of size k-1 to a list k times, the space complexity is also O(nk)

Implementation: **#1082623**

https://cogniterra.org/submissions/202076/1082623?unit=45309

Discussion: The algorithms in the program for creating the graph are simple and straight forward. Because we need to look through each letter in each k-mer, the best runtime for this algorithm is going to be O(nk) which is what we get with this algorithm. Depending on the implementation, there might be a log factor because of the graph edge insertion function.