CSE 5307 Bioinformatics Homework 2

Name: Venkat Ankit Gundala

Student ID: 1002069069

1 Shortest Common String Time complexity

Shortest Common String Time complexity

The original brute force solution implemented above had a very undesirable time complexity of O(n!), where n is the number of strings in the input set. This is because it involved generating all possible permutations of the input set and then iterating through each permutation to compute the overlap between adjacent strings. The improved solution implemented in the second code snippet has a much better time complexity of O(n$^2 * k$, where n is the number of strings in the input set and k is the length of the longest string in the set. This is because the solution pairs up adjacent strings and computes their overlap, then merges them together to create a new string. This process is repeated until there is only one string left in the set. Additionally, the use of while loops instead of for loops contributes to the improved efficiency of the solution.
The time complexity of your improved solution implemented in the scsfast.py file is, O(n$^2 * k$).

2 De Brujin Graphs

2.1 Generating Your Own Unique Data

To generate a unique data set using the code provided in datasetGenerator_hw2.py, you should run the program and input your student ID (in this case, 1002069069). This can be done by using the command line and typing python3 datasetGenerator_hw2.py –ID 1002069069.
Once the program has run, a new text document with the name 1002069069.txt will be created, containing the unique reads generated by the program. In summary, the process involves running a program with your student ID as input to create a new text document with unique data.

2.2 Generating K-Mers

Calculated K-mers with k=2 and generated the strings in new file named kmers.txt

2.3 Generating a De Bruijn Graph

Taking the available 1002069069.txt as input, a De Brujin graph is generated with the available string in the text file with k=2.

2.4 Eulerian Cycle

Since the De Bruijn graph does follow the Eulerian walk i.e it cover all the edges, my De Bruijn graph has an Eulerian cycle.

2.5 Genome Assembly

Since my De Brujin graph has an Eulerian cycle there is no need to add any reads.
The Assembled Genome sequence is tcgagacgcgt

3 Difficulty Adjustment

It took 20 plus hours for me to complete the assignment.
writing the code for the shortest common superstring and identifying which file to use to generate De Bruijn's Graph.