Final Project

INFR 2820U: Algorithms and Data Structures
Dr. Kalil El-Khatib
April 9th

Ryan Manzie (100743508)

Noah Morin (100740090)

Blake Whiting (100743587)

Academic Integrity Statement

As a member of the Ontario Tech University community, I share our community's commitment to the highest standards of academic integrity and excellence in all dimensions of our work.

I therefore promise that I will not lie, cheat, or use any unauthorized aids or assistance to complete any of my essays, assignments, and exams. I further promise that I will not offer any unauthorized assistance to any of my fellow students, and I promise that I will not ask any of my fellow students for unauthorized assistance. I promise that the work I submit is my own and that where I have drawn on the work of others, I have included proper attribution for my sources.

Signed,

Ryan Manzie, Noah Morin, Blake Whiting April 9th, 2021

Our Project is a program written in Python that simulates a network employing the Routing Information Protocol (RIP), which is a distance-vector protocol that uses the Bellman-Ford Algorithm to compute the best path to a destination network. Furthermore, the program implements Kruskal's minimum spanning tree algorithm to create a minimum spanning tree in the network.

Our program begins by taking the input of a file named network.txt which contains the raw data for the graph of the network, stored in a text format. This data is taken in through an in line for loop that takes node names and their respective values and writes them into a two dimensional array. By storing our graph in this data structure, we are able to efficiently store edge information and quickly access it through for loop iterations.

From this point the user is presented with a number of optional actions. They can modify the graph in a number of ways, such as changing an edge, removing an edge, or removing an entire node. Should the user decide to update any of the graph's information, the text file can be overwritten to reflect the changes. From this same menu, the user can also choose to print the graph, where it will display in a text table format, listing all the relevant edge node names and values. There are a number of other options as well, such as the ability to view our academic integrity statement, or write any changes made into the file for future use.

When the user decides to run the computational algorithms portion of the program, it will begin by entering the BFPrep function, which prepares the graph data to be run through the Bellman-Ford algorithm. It does so by passing the appropriate data into the necessary variables, such as vertices and edges. From here it will pass the data into the Bellman-Ford function, where we use the Bellman-Ford algorithm to calculate the shortest path to each destination

node in the network. For the case of this program, we selected Oshawa as the source node to calculate the costs to each destination.

The user will also have the option to calculate the minimum spanning tree using Kruskal's MST algorithm, which is an algorithm that computes the minimum spanning tree on a given graph including a list of edges and vertices. To begin the process, much like the Bellman-Ford algorithm before it, certain values and variables must be prepared. To do this it goes into the KruskalPrep function, where it assigns vertices and formats the data to be processed. Once that is completed the data is passed into the Kruskal function, where it uses the algorithm to find the minimum spanning tree for the given graph input into it. This results in the lowest cost for travelling to every single node on the graph. In the context of a real world network using Kruskal's algorithm with RIP, a higher bandwidth would be preferred. To reflect this, the minimum spanning tree is reversed to reflect the preference towards higher bandwidth links (higher bandwidth, lower cost).

Works Cited

A. Chaudhary, “Reading a file into a multidimensional array with Python,” *Stack Overflow*, 27-Sep-2013. [Online]. Available: https://stackoverflow.com/a/19056145. [Accessed: 03-Apr-2021].

“Bellman–Ford algorithm,” *Wikipedia*, 14-Mar-2021. [Online]. Available: https://en.wikipedia.org/wiki/Bellman–Ford_algorithm. [Accessed: 03-Apr-2021].

D. Getz, “How to do ‘hit any key’ in python?,” *Stack Overflow*, 08-Sep-2009. [Online]. Available: https://stackoverflow.com/a/1395006. [Accessed: 09-Apr-2021].

“Kruskal's algorithm,” *Wikipedia*, 05-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/Kruskal's_algorithm. [Accessed: 03-Apr-2021].

“Kruskal's Algorithm,” *Programiz*. [Online]. Available: https://www.programiz.com/dsa/kruskal-algorithm. [Accessed: 09-Apr-2021].

M. Sambol, *YouTube*, 10-Jun-2015. [Online]. Available: https://www.youtube.com/watch?v=obWXjtg0L64. [Accessed: 01-Apr-2021].

M. Sambol, *YouTube*, 25-Nov-2012. [Online]. Available: https://www.youtube.com/watch?v=71UQH7Pr9kU. [Accessed: 09-Apr-2021].

YouTube, 16-Feb-2018. [Online]. Available: https://www.youtube.com/watch?v=FtN3BYH2Zes&t=132s. [Accessed: 09-Apr-2021].