FINAL PROJECT REPORT

Algorithms and Data Structures (INFR 2820)

Faseeh Ahmed (100730111)
Neroshan Manoharan(100581847)

04.09.2020

Dr. Khalil El-Khatib

Academic Integrity is a central value of this university. Please read and sign the following statement:

Students are expected to be familiar with and abide by UOIT's regulations on Academic Conduct which sets out the kinds of actions that constitute academic misconduct, including plagiarism, copying or allowing one's own work to copied, use of unauthorized aids in examinations and tests, submitting work prepared in collaboration with another student when such collaboration has not been authorized, among other academic offences. The regulations also describe the procedures for dealing with allegations, and the sanctions for any finding of academic misconduct, which can range from a resubmission of work to a failing grade to permanent expulsion from the university. A lack of familiarity with UOIT's regulations on academic conduct does not constitute a defense against its application.

I/We am/are the sole author(s) of all the work and solutions for this project.

Name and Signature(s): ___FASEEH AHMED 100730111_____

_____NEROSHAN MANOHARAN 100581847_____

PROJECT

The objective of this project is to create a computer program that will automatically create a graph showcasing the different routers in each city, across Canada. This program will take the input of a single "network.txt" file which will list an edge, along with its corresponding vertices and weight on that edge which is connecting the two verticies. The vertices will be the city which we will be referring to, in string format, and the weight will be the distance, expressed as an integer.

As a deliverable, we need to show the shortest distance from one vertex to all others. In this case, we have used Vancouver as the source node for which all other vertices were calculated from. This can easily be changed in the following snippet from our code:

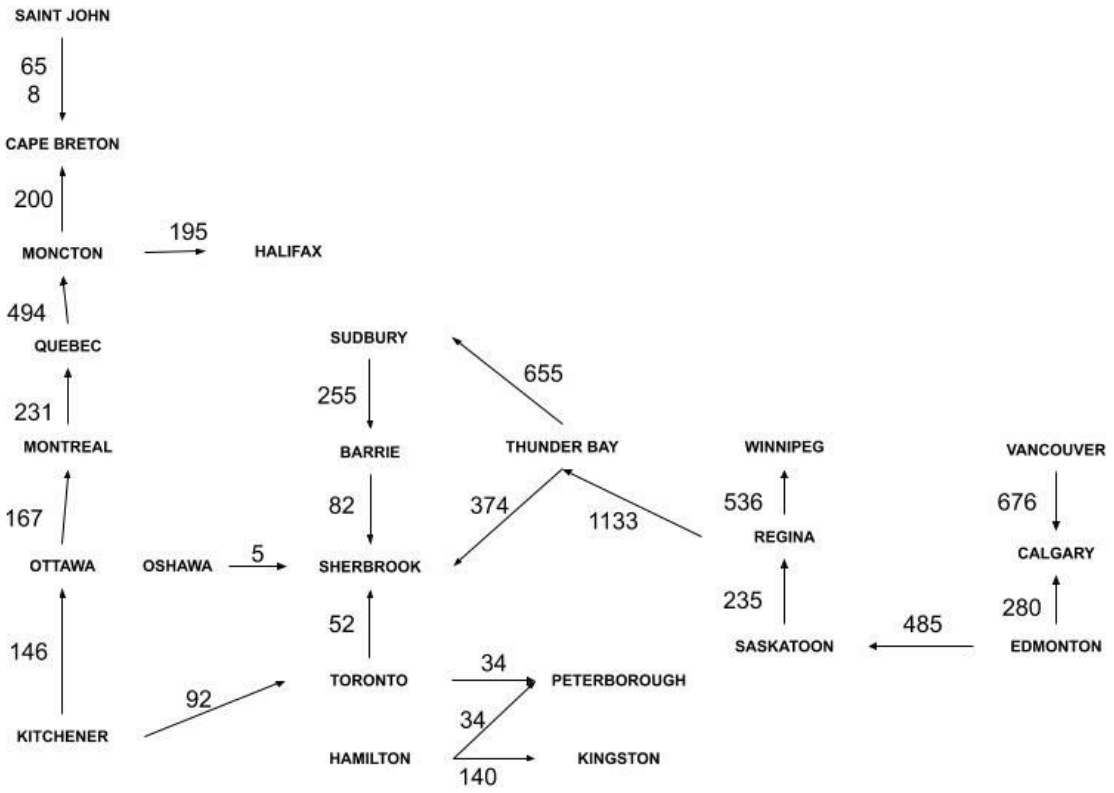
```
g.bellman_ford('Vancouver')
```

PROGRAM

The program is initiated when a file is called up with the creation of a Graph, in this case our “network.txt” file. The default constructor would automatically call the create_graph function and feed the list to this function. This will then create a 2D graph and return an adjacency list for it.

The Bellman_ford function uses the Bellman-Ford algorithm to find the shortest path from a vertex to all other vertices in the graph. The Bellman-Ford algorithm uses for loops to calculate the shortest path between edges.The algorithm starts at the vertex given, in our program that would be Vancouver, and compares it to other vertices, while updating shorter paths as they are found. It iterates for a total of (V-1) times, which is basically the number of vertices -1. The nested for loop iterates over all the u, v edges relaxing each one if a shorter distance is found. One issue to consider with this for loop is the existence of negative cycles. This was an issue our team faced in the early stage of the project as weights can sum to a negative number. To resolve this, we used an if statement that detects when a negative cycle has occurred and notifies the user.

Kruskal’s algorithm was used to compute the minimal spanning tree for the graph. The function implemented for this is named mst_kruskal. Minimal spanning tree is computed by first sorting all the edges by edge weight in ascending order, and then adding the smallest edges to an array named mst. The algorithm also considers the rank of each edge as it creates the minimum spanning tree. This ranking is created in the union function which keeps track of the entire path weight from the root of the parent down to leaf. The output would then consist of a list of all vertices that create a cycle free network. The minimum spanning tree created by kruskal's algorithm is depicted below.



DELIVERABLES

The deliverables for part b of the project, which were functions that need to be able to carry out various actions for the graph were implemented in the beginning of our code. These functions include update_weight, remove_edge and remove_node. These functions do exactly as their names suggest. Once the graph is updated, the bellman ford algorithm is executed to display the shortest path between the intended source (on line 138) and every destination. The dictionary consists of each city and its respective distance to the source. Lastly, the Kruskal algorithm is executed to create the minimum spanning tree out of each node in the network.

INSTRUCTIONS

1. Ensure network.txt file is located in root project folder
2. Open main.py
3. Ensure the name of the .txt file on line 137 corresponds with the .txt file name in project root folder

4. Set source city for Bellman Ford algorithm on line 138
5. Execute/run program