Coursework for DataStructures and Algorithms

A screenshot of a computer

Description automatically generated

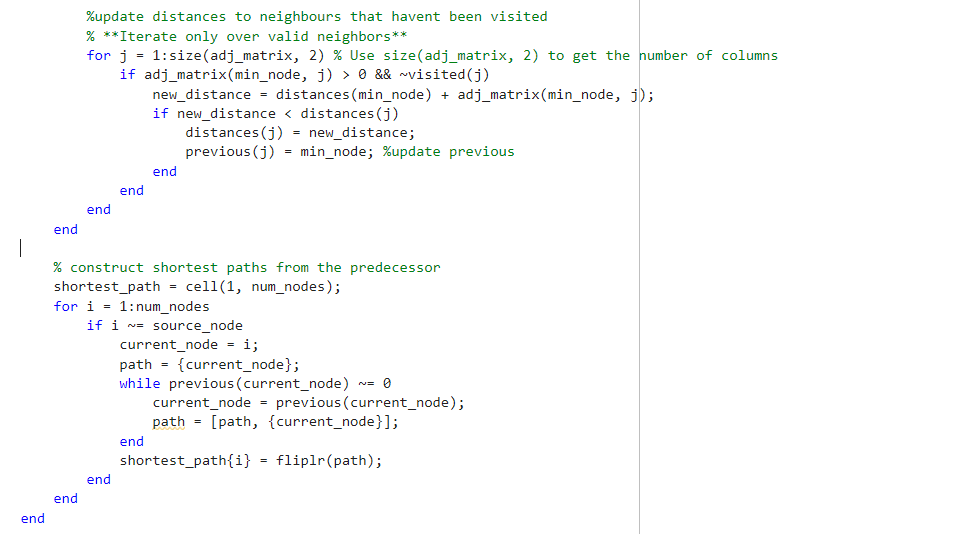
Above is the data that I was provided for my student number and I have spent time creating a way to plot this in a graph. I first looked at how to draw the graph and decided with Digraph, since this is a directed graph. I then went onto plotting the graph and this is what I was given.

A diagram of a triangle with points and numbers

Description automatically generated I then went onto created Dijkstra’s shortest path algorithm, which looks like this in MATLAB code. The code is full of comments which explains how it works.

A screenshot of a computer program

Description automatically generatedThe greedy strategy within shortest path algorithms is a method in which you find the shortest path by using the shortest available options at one time, even if the next value is much higher than the other. It is a method of approximating what will be the shortest path to any node at any given time. This does mean that greedy strategies never can guarantee the best possible solution, there are solutions where at first choosing a suboptimal choice can be better later. Greedy algorithms are especially useful in scheduling problems, where you need to fit a set of tasks in within a given time. For example, going to a theme park, you might always choose the ride with the shortest wait time. However, if you take the travelling salesman problem, the problem that asks the question: “Given a list of cities and te distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?”. Dijkstra’s algorithm cannot solve this well as it is only coded to find the shortest path between one city, not multiple. Dijkstra’s algorithm is aimed towards minimising the edge weights, along one given path, whereas the TSP aims to minimise the entire route.

Dijkstra’s works by reading the values which are next on a graph, while a node is unknown, the value is set to infinity, as this is how Dijkstra’s works. It needs to do this because any finite value is less than infinity. When it reaches a node, it counts the node as visited and tracks the distance, it then repeats this process until it has gotten to the node it needs to get to. Once it has reached the value, it backtracks to find the parents of the node, to check the distance and the node it is on.