Max Chi
ECE 368 Project 3

Milestone 1

For Project 3, we have been assigned the task of implementing the algorithm known as Dijkstra's shortest path algorithm. The Dijkstra algorithm is used to help find the shortest paths between nodes in a graph, which can represent a map of rode networks. This is applicable in today's world in that similar algorithms are used in geographic information systems and GPS navigational systems such as MapQuest or Google Maps.

To start off, the first step is to understanding how Dijkstra's shortest path algorithm works. Based on the Wikipedia link provided and other sources not listed, it seems that the performing this algorithm can be split into multiple parts. First it is important to analyze the input files, the first which represents the map, and determines city locations that will be represented by nodes, as well as edges, which are used to represent the roads connected the cities together. The input file is constructed so that the first line will provide information of the number of nodes and roads there are in the map, followed by lines listing the nodes in an x- y coordinate fashion, then finally listing the the roads in terms of pairs of nodes, representing the path between two nodes. The second input file has on its first line the number of queries that is to have the shortest paths found for the cities listed on the respective lines that follow.

After analyzing the input files, I will attempt to create an algorithm that given input files, will output information pertaining to adjacent nodes and edges to certain designated nodes/cities. This would help complete milestone 2, but at the same time help implement my solution of the Dijkstra algorithm. In theory, the adjacent algorithm would continue to collect information on the adjacent nodes and the lengths of their edges and add and compare them and collect multiple path lengths, and then compare them once more to obtain the shortest possible path between two given nodes. I am thinking of implementing a version of linked list in order to have a map layout of all the nodes depending on the input files. Each list would be comprised of the current location, and adjacent nodes along with their distances from the respective origin nodes. Finally, when the apparent shortest path has been found, it is necessary to back trace or reverse the path from the destination to the origin in order to confirm the shortest path. If the path distance of this is equal to that of the initial path distance, then the shortest path will be confirmed true.