**Dijkstra’s Algorithm ReadMe**

**Introduction:**

In this code, I implemented Dijkstra’s algorithm, an algorithm that determines the shortest path from one node to any other node in a graph. Dijkstra’s algorithm happens to be a greedy algorithm, which means that it makes the optimal choice at every point in an attempt to find the correct solution. It is also possible to prove that Dijkstra’s algorithm always works.

There are a couple initial conditions for Dijkstra’s algorithm. First, three ArrayLists are initialized. The visited ArrayList, which stores whether or not a node has been visited, is initialized to false. The distance ArrayList, which stores the distance from the source node, is initialized to infinity for every node except the start node, which has a distance of zero. The vertex ArrayList stores the previous node in the shortest path to the specified node, and is initialized to null.

Every iteration of Dijkstra’s algorithm chooses the closest node to the source that hasn’t been visited, and then updates all three ArrayLists. The algorithm ends when every node has been visited.

This project contains two classes, the Dijkstra class and the Graph class

**How This Fulfills Specifications:**

This project fulfills the expectations of the assignment. It successfully implements Dijkstra’s algorithm, which returns the correct shortest path distances and shortest paths. However, there is some discrepancy between the wording of the assignment and the way I implemented Dijkstra’s algorithm. The assignment suggests implementing a method that has arguments of an adjacency matrix and a start node, which returns a sequence of vertices that corresponds to the shortest path. However, this did not make sense, since the shortest path also needs a representation of the end node. Thus, I implemented Dijkstra’s algorithm in the constructor of the object. Getters can then access the shortest paths and distances calculated. This is much more efficient than performing Dijkstra’s algorithm multiple times for the same start node, since Dijkstra’s algorithm solves the shortest path for all end nodes of the graph.

**Explanations of Current Errors:**

From the testing that I have performed so far, there are no outstanding errors present in this code. For all of the graphs that I have tested, all the shortest paths have been calculated without any problem.

**Overview of the Code:**

This is provided in the introduction.

**Discussion of Major Challenges:**

First, it was hard to design a good selectMinNode method, since it requires checking both the distance ArrayList and the visited ArrayList. In addition, a problem came up using Integer.MAX\_VALUE to represent unconnected nodes. As it turns out, Integer.MAX\_VALUE + 1 returns Integer.MIN\_VALUE. Because Dijkstra's algorithm deals with minimums, this fact caused many problems in my code that I eventually did solve

**Acknowledgments:**

I would like to thank MIT OpenCourseware for an extremely detailed and thorough explanation of Dijkstra’s algorithm, which I needed to fully understand the specifics of the algorithm