The first thing I did was create a class for the edge and the vertex so that they can be used later. I also set a variable V to 50 since this is the maximum number of vertices. In the edge class, s constructor is set up with the origin the destination and weight. The vertex class has a constructor with a name only. It also contains a method to add an edge.

Insert Vertex: O(n) this method creates a method however if it finds that the vertex already exists it won’t do this.

Insert Edge: O(n) creates an edge with the appropriate weight however if the vertex isn’t real it will add the vertices and then the edges.

Get weight: O(n) the first thing this method does is create a small path from the starting vertex to the ending one. When this is done it runs through the vertices to get the weight from the edges. If any of the vertices don’t exist, the method returns zero.

Is Edge: O(n) this method runs through the vertices to check for the edges and their properties if it doesn’t exist then it returns false. If it does exist, the method returns true.

Get Adjacent: O(n) first this method locates the vertex that is needed. After this it searches all of the edges attached to this vector and obtains the end of each edge attached to the appropriate vertex.

Print Dijkstra: O(n^2) goes through a loop in which it finds the edge with the smallest value in the direction of the destination and updates the distance. If a shorter path is found it will update the path.

Print Graph: O(n^2) method initially collects all of the vertices from the graph and sorts them. It the proceeds to get the adjacent vertices and sorts them. Finally, it prints them out on the display.

What I learned from this project is how algorithms (such as this one) can be used to find the shortest route to a destination and how applicable something like this is to real life. Ex. Navigation systems. If I were to do this again I would set it up as a navigation system and compare it to one to see how my algorithm preformed and see what I could change.