CPSC 331
TA: Harnoor K.

Shaheryar Syed
UCID: 30052162

When considering my implementation for Dijkstra's algorithm, and in general, we find that the time complexity is related to the vertices. Namely, for my code the time complexity is O(V^2) where v is the number of vertices, which is equal to the warehouse(s) + delivery locations. The reason this is the case is because the algorithm scans through all the vertices in order to locate the minimum edge distance which is O(V), however we must remember that the algorithm has to do this for every single vertex that we provided it, therefore we get O(V) * O(V) = O(V^2).

The space complexity of this algorithm is also O(V^2), this is because we use a 2D matrix for the code and the matrix handles the storage of the distance between all the pairs of vertices we enter into the program. This means even if some of the roads aren't connected, they are still stored in the matrix impacting space complexity.

If we consider the implementation for a much higher scale, we will find that the quadratic approach to the complexity is very difficult to store and maintain as it would require lots of storage space and take a long time to execute. Therefore, one thing we can do is to partition the graph into smaller graphs, similar to the way we would approach a merge sort problem. By doing so, we can effectively split each pair of vertices and its weight, then use multi threading to visit multiple locations from a single location at once. At the end we can compare which distance is smaller to a particular location by comparing the edge weight. This would reduce the time and space required because we are using smaller graphs meaning the time complexity will be smaller, and the space complexity.