Task1:

Using Dijkstra algorithm to find the shortest path from one node to all other nodes. Using min-heap as discovered list to make time complexity for searching the smallest node in the current discovered list O (log V).

First create the graph, initialize the discovered list to be containing only start node and distance list to be all 0s. initialize previous list and finalized list to be empty. Loop until all nodes are finalized: get the node with smallest distance in the discovered list, for every node connected to this smallest node: if the node is already finalized, ignore it. Else if the node is not in the discovered list, change the distance in the distance list as the weight of this node from the current node plus the distance of current node, append it into discovered list and rearrange the heap. Else if the node is in the discovered list, compare the distance of this node and the sum of weight of this node to current node and the distance of current node. If the last one is smaller, update the distance of this node in the distance list and rearrange the list as the distance list is already changed. After all nodes are finalized. Return the previous list and the distance list. The element having index end in the distance list will be the distance from start to end. Loop through the previous list by changing current node to previous [current node] will give us the path from start to the end. When creating the graph, we go through all the edges, therefore, the time complexity of creating graph will be O (E). while the first loop, we go through all the vertices until all vertices are finalized, which means V times, inside each loop, we find the smallest node which take O (log V) time, and go through all the edges connected to that node which means E times, for each time go through an edge, update the distance or insert element into the heap take O (log V) time. Therefore, overall need O (V*log V + E*log V) time, because E >= V, the time complexity is O (E log V). the program need to store every edge so need E space. Discovered list and finalized list and previous list all need V space. Overall, need O (E + V) space.

Task 2:

Run Dijkstra algorithm twice, first time from start to end, second time from end to start. Then distance from start to all customers and distance from end to all customers are all known. This take O (E log V) time. Then load customers into a list. This need O (C) time. Add two part of distance together and get the total distance for every customer. Get the minimum one and get the corresponding customer. Using the same method get the path. This need O (C) time. Overall, time complexity is O (C + E log V) and because C <= E then O (E log V). there is only an extra list to store the distance of each customers then the space complexity is still O (E + V).