# Task 1

Task 1 was completed by representing the vertices and edges of the graph in a minHeap. The optimal path was obtained by applying Dijkstra's Algorithm to the minHeap structure in order to satisfy the complexity requirements. The graph is represented as a list of lists of size V. It is initialised first as [0] * V. Then at each index is the lists of edges and the distance from that node.

The following illustrates the representation of the graph:

graph = [[0,[1,2.3],[3,6.3]],[1,[5,1.2],[3,2.4]],…….]

So at graph[0], the node "0" is connected to the node "1" with distance 2.3 and is also connected to the node "3" with distance 6.3.

**Time complexity: O(Elog(V))**

V = # of vertices
E = # of edges

To traverse through the heap structure will take O(log(V)) time since the heap has size V, and at each step the edges within the graph are updated, thus giving the total time complexity of O(Elog(V)).

**Space complexity: O(E + V)**

Given that the graph will have V number of vertices and E number of edges, then the total space it will take up in memory is O(E + V)

# Task 2

Task 2 was completed using the same implementation as Task 1, with the exception that the nodes and edges at which there was either a camera or a toll, they were replaced with a null character thus, indicating that these nodes/edges must not be traversed.

**Time complexity: O(Elog(V))**

V = # of vertices
E = # of edges

To traverse through the heap structure will take O(log(V)) time since the heap has size V, and at each step the edges within the graph are updated, thus giving the total time complexity of O(Elog(V)).

**Space complexity: O(E + V)**

Given that the graph will have V number of vertices and E number of edges, then the total space it will take up in memory is O(E + V)

# Task 3

To understand how Task 3 was completed consider the following example:
Suppose we are given a positive weighted graph containing 100 nodes. Within this graph, 10 nodes consist of service points and we are trying to find the quickest detour path from node 0 to 100.

In order to resolve this problem, first compute the shortest path between all pairs of the twelve special nodes (nodes 1, 100, and all ten of the service point nodes), and use these as edge

lengths in a new graph consisting only of these twelve nodes. Now, a graph containing twelve nodes has been created, Dijkstra's Algorithm is applied to this simplified graph in order to find the quickest detour path.
More specifically, first Dijkstra's algorithm is applied to the source node to find the optimal path from the source node to the service points, and then Dijkstra's algorithm is applied to the target node, to find the optimal path from the tservice point to the target node. These two paths are combined to find the overall quickest detour path from source node to target node.

**Time complexity: O(Elog(V))**

V = # of vertices
E = # of edges

To traverse through the heap structure will take O(log(V)) time since the heap has size V, and at each step the edges within the graph are updated. This is repeated for the target node as well thus giving a total time complexity of:
O(ElogV + ElogV) = O(ElogV)

**Space complexity: O(E + V)**

Given that the graph will have V number of vertices and E number of edges, then the total space it will take up in memory is O(E + V)