Assignment 4

FIT2004 Pavle Spasic 29691931

PreTask

The graph class main structure is built using only adjacency lists, it takes the file of length E, inserts the edges to which vertex they start is a dynamic list, dynamic as in always changing its size if the vertices are too big. It takes O(E) time complexity and uses O(E+V) space complexity.

Task 1

The quickest path function used Dijkstra's algorithm to find the shortest path within a graph and 2 vertices in O(ElogV) time complexity. This time complexity is achieved as each edge is iterated, O(E), and during each iteration, a min heap is used to know which the minimum node used next, the rebalancing of the min heap requires O(logV) time complexity. There are many lists used, ranging from O(E) and O(V) space complexity, never being greater. The path from source to target can then be found from backtracking from the source as each vertex stored which node they connected to for the shortest path, this does not increase the functions space or time complexity.

Task 2

First part of this task was the function augmentGraph. The function gets 2 files from input and stores them into a list under __init__ in the class. These functions have space complexity of their own but will not ever reach E or V in terms or size and time.

The next part is the function quickestSafePath. This was achieved using the same code as the quickestPath function from Task 1, but it has been altered to factor in the bad edges and vertices. Firstly, the red lights are considered, this is simply done by having a copy data structure of the graph and changing the nodes and the index of the light to None, as if they had no vertices from the node. The toll roads are checked during the Dijkstra's algorithm loop, when it gets to an edge, it checks if in another list that was created prior, if the edge had a certain value and was not changed, it it did, it did not go through that edge, and continued to the next. The time complexity of the red light elimination was O(n) where n is the number of red lights, this will never be greater than E, so, the worst case time complexity remains O(E + V).

Task 3

The first part, like task 2, was the function addService. This function simply got the input of a file, filename_service, and save it to a list in the class.

Next is the main function, quickestDetourPath, the way this function works is it uses the function quickestPath from task 1, and it iterates through the list of service stops, checks if it can go from source to stop, if it can check if it can then go from stop to target, if it can good. As it goes through each stop it saves the minimum cost and its path, so at the end you have the quickest path. The worst-case time complexity is still O(ElogV) as found in task 1. Even though it is iterating a list of, let's say size X, X is usually very small in comparison to E and, in fact, is a constant, so it will not affect the time Complexity. No data structure was made where its size complexity was in terms of E or V, so the space complexity remains O(E+V).