GRAPH

Time Complexity Analysis

Therefore, the time complexity of my Dijkstras algorithm is O(v^2), where v is the number of vertices (the location which is inclusive and contains the warehouse).

The key steps and their time complexities are:

Initialization:

Initializing the distance array D and the visited array V: O(v)

Main Loop:

Finding the vertex with the minimum distance: O(v) using the minimumDistance function.

Up to updating the distances for the adjacent vertices: O(v^2), since the algorithm has to do this for each visited vertex and it loops through all the vertices.

Calculating the Number of Routes:

Each location in the routes will take O(v^2) time, since in this algorithm, for each location, it needs to iterate over the vertices and so the route list.

Time complexity of the implemented functions, in the given context, therefore, is overall O(v^2), where v is the number of locations (including the warehouse).

Space Complexity Analysis

On its space complexity, Dijkstra's algorithm observes O(v^2) since v represents the number of vertices (locations in the graph, in this case, it is the warehouse).

The key data structures and their space complexities are:

Data Structures:

Distance array D: O(v)

Visited array V: O(v)

Route array route: O(v^2), as each location can have multiple routes, and each route can have up to v vertices.

Auxiliary Variables:

The minimumDistance function uses a few integer variables, which can be considered as O(1) space.

The main function uses a few integer variables for input, which can be considered as O(1) space.

Thus, the space complexity involved in implementing D其中ijkstra's algorithm in the given context is O(v^2), where v is the number of locations (including the warehouse).

Scalability and Optimization If executed with a priority queue, the time complexity of the algorithm is O(V^2), operated with an adjacency matrix that can further be reduced. Space complexity: O(V) with the matrix and O(V+E) with the list. Generally, the retrieval of a vertex is more time-efficient and uses a list over a matrix representation, which is space-efficient