GIT Department of Computer Engineering

CSE 222/505 - Spring 2023 Homework #8 Report

Pelin Erdoğan

210104004266

1. PROBLEM SOLUTION APPROACH

In this homework the first thing I had to figure out was how to read the file and store the map. I took every line in the file and took integers in the line and stored it in 2d arraylist. Except from first 2 line, I keep them separately as start and end points. After whole map is stored in arraylist I check If start and end point is in unoccupied region. After that I created Graph. I had searched through the map and created vertexes from 0's. Then I created edges with a nested loop. First loop chooses the main vertex then searches for its neighbors. The right corner neighbor is the last one so the second loop breaks after it finds the right corner neighbor. When it finds the neighbor, it creates an edge and adds it to the edges list of main vertex.

For BFS:

I created visited(Boolean) and parent(Vertex) arraylists. And set their initial values. False for visited and null for parent. Then I found the vertex at start point. After setting the start point to true in visited, I added the start point to queue. Then search through the edges list of vertex that is polled from the queue. If neighbor is not visited we set it to visited and add it to the queue and set the parent of neighbor to current vertex. When queue is empty method calls get path method which returns arraylist of strings that stores path in it.

For Dijkstra:

I created visited(Boolean) ,distances(Integer) and predecessors(Vertex) arraylists. And set their initial values. False for visited and null for parent. Then I found the vertex at start point. After setting the start point to true in visited and distance to 0. Current vertex is returned from getMinDistanceVertex. Which returns the vertex that is not visited and has minimum distance.while current vertex is not null, I set visited to true for current vertex and for every neighbor of it if their distance is bigger than distance of current index plus one, I set the predecessors of that neighbor to current vertex and set their distance to their new distance. After all I call getPath method which returns path as a string arraylist. Get Path set current vertex to the finish vertex and add the predecessor of it to path and until current vertex is null.

2. TIME ANALYSİS

Dijkstra has a while loop that calls getMinDistanceVertex and that method has a loop that iterates number of vertexes and for now lets call it V. After that method it has another loop that iterates number of current vertexes edge count. Lets call that e for now. Since while loop iterates until current vertex is null. We can say it iterates the same amount of vertex number because getMinDistanceVertex is not returning null until every vertex is visited.

Even though every vertex doesn't have same amount of edges we can say that it has an equation V* (V + e) which equals to V^2 + Ve+1 and if we say amount of all edges are E it is V^2 + E+V and that is represented as O(V^2).

In my program it took 49741300 ns (0,497 seconds) for map 2

Bfs has a while loop that iterates number of connected vertexes. Because every vertex's neighbor is added to queue if it is not visited. Inside the while loop it has a for loop that iterates number of edges. So it has equation V(e + 1) which can be shown as V+E so it has complexity O(V+E).

In my program it took 31241000 ns (0,312 seconds) for map 2

V: Number of vertexes.

e: Number of current vertex's edges

E: Number of all edges