Computer Engineering
CSE 222/505 - Spring 2023
Homework # Report

Name: Ahmet Furkan Ekinci
Student Number: 200104004063

1. Introduction

Purpose of Document

This document is report for this project. The report is going to explain you what stages the homework was going through.

Purpose of Project

The purpose of this assignment is to create our own graph class by reading from a txt file and to implement methods including bfs and dijkstra algorithms to find a way to go between two points on this graph.

Note: First number is 'x' , second number is 'y'.

2. Time Complexity Analysis

1. BFS Algorithm : O(n * e)

n is the size of the queue and e is the number of vertex edges.

```
while (!queue.isEmpty())
{
    Vertex currentVertex = queue.poll();

    ArrayList<Edge> neighbors = currentVertex.getEdges();

    for (int i = 0; i < neighbors.size(); i++)
    {
        if (!visited[(neighbors.get(i).getX()*size) + neighbors.get(i).getY()])
        {
            visited[(neighbors.get(i).getX() * size) + neighbors.get(i).getY()] = true;
            parent[(neighbors.get(i).getX() * size) + neighbors.get(i).getY()] = currentVertex;
            queue.add(findNumberOfVertex((neighbors.get(i).getX() * size) + neighbors.get(i).getY()));

            // Hedef düğüme ulaşıldıysa, yolunuzu döndürebilirsiniz
            if (((neighbors.get(i).getX() * size) + neighbors.get(i).getY()) == target)
            {
                return constructPath(parent, start, target);
            }
        }
    }
}
```

2. Dijkstra Algorithm : O(n * x)

x is the sum of the number of vertex edges and the number of vertexes. n is the amount of vertex until the vertex is no longer connected with anyone.

3. Running Time Analysis

The working time of the bfs method is 20 seconds, while the djikstra method takes about 2 minutes and these times may differ for different maps.

I defined a list<vertex> type variable inside the graph structure and created my vertex class. In addition, I defined a list<edge> type variable inside the vertex class and also created my own edge class. Since I create a real graph structure myself using my own classes and their methods, the time is a little longer, but because the maps are too large, the code's running time is also very long.

There are 4 lines that I have commented in the TestCases class, if you open these comments, it prints map i as 0 and 1, it also reads the beginning and end indexes from txt and prints it to the screen, and the line in the last comment prints all the vertexes in my graph and their edge indexes, that is, it proves that I created the graph.

4. Comparing Algorithms

BFS (Breadth-First Search) and Dijkstra's algorithms are search algorithms used in graph-based problems. Both are used to find a path from a starting node to a target node or other nodes. However, they have different purposes and constraints, which results in different

outcomes.

The BFS algorithm performs a breadth-first search and explores the graph step by step. It starts from the initial node and discovers all its neighbors, then expands to the neighbors' neighbors, and continues this process. The BFS algorithm explores the nodes in a sequential manner, gradually expanding as the shortest path. Therefore, the path found with BFS is the one with the minimum number of steps from the starting node to the target node. However, BFS algorithm does not produce accurate results in weighted graphs or graphs with negative weights since it simply explores all neighbors.

On the other hand, Dijkstra's algorithm is used to find the shortest paths from the starting node to all other nodes. Unlike BFS, Dijkstra's algorithm works on a weighted graph and calculates the total cost from the starting node to each node. It selects the node with the lowest cost and updates the costs of its neighbors. This process is repeated to find the shortest costs from the starting node to all other nodes. Dijkstra's algorithm produces accurate results in weighted graphs but does not work with graphs that have negative weights.

In conclusion, the BFS algorithm focuses on finding the shortest path in terms of the minimum number of steps, while Dijkstra's algorithm focuses on finding the shortest path in terms of minimum cost. Therefore, the results of these two algorithms can be different. If only the shortest path is required, BFS can be used, while Dijkstra's algorithm is preferred for weighted graphs or situations where costs are taken into account.