GTU Department of Computer Engineer
CSE 222/505 - Spring 2023
Homework #8 Report

AHMET HAKAN SEVİNÇ
200104004047

Graph Traversal Algorithms

Introduction: The objective of this project is to develop and analyze two essential graph traversal algorithms: Breadth-First Search (BFS) and Dijkstra's algorithm. These algorithms are implemented using the classes CSE222Graph, CSE222MAP, CSE222Dijkstra, and BFSAlgorithm. The primary purpose is to apply these algorithms to solve pathfinding problems in a graph representation.

All Classes

CSE222Graph: The CSE222Graph class serves as a graph data structure using an adjacency list implementation. It manages the vertices and their adjacent vertices within the graph. Key functionalities include adding vertices, adding edges, retrieving adjacent vertices, and obtaining the start and end vertices.

CSE222MAP: The CSE222MAP class is responsible for reading a map from a file and generating a 2D array representation of the map. It stores the start and end coordinates of the map and provides methods to access the map as well as retrieve the start and end coordinates. Additionally, it offers features to convert the map to a PNG image and draw a path on the image.

BFSAlgorithm: The BFSAlgorithm class implements the Breadth-First Search algorithm for determining the shortest path in a graph. It takes an instance of the CSE222Graph class as input and performs the BFS traversal to find the shortest path from the start vertex to the end vertex. It includes methods to retrieve the shortest path and throws an exception if the path is not found.

CSE222Dijkstra: The CSE222Dijkstra class implements Dijkstra's algorithm for finding the shortest path in a graph. It takes an instance of the CSE222Graph class as input and executes Dijkstra's algorithm to compute the shortest path from a given start vertex to all other vertices. The class provides methods to retrieve the shortest path from the start vertex to a specified end vertex.

Time Complexity Analysis:

Breadth-First Search (BFS) Algorithm:

The time complexity of the BFS algorithm is O(V + E), where V represents the number of vertices and E denotes the number of edges in the graph. Within the BFSAlgorithm class, the breadthFirstSearch method performs the BFS traversal, visiting each vertex and its adjacent vertices exactly once. The overall time complexity depends on the size of the graph and the number of edges.
Real time is measured for given first map is **14 seconds.**

Dijkstra's Algorithm:

The time complexity of Dijkstra's algorithm depends on the implementation. In our implementation, the time complexity is O(V^2), where V represents the number of vertices in the graph. The algorithm iterates over all vertices and their adjacent vertices in each iteration. The getSmallestVertex method, called within the algorithm, has a time complexity of O(V).
Real time is measured for given first map is **240 seconds**.

Conclusion: This project demonstrates the implementation and utilization of the BFS and Dijkstra's algorithms for graph traversal and pathfinding purposes. The BFSAlgorithm class enables the discovery of the shortest path using the BFS algorithm, while the CSE222Dijkstra class implements Dijkstra's algorithm to find the shortest path from a start vertex to all other vertices. Both algorithms have distinct time complexities, with BFS having a time complexity of O(V + E) and Dijkstra's algorithm having a time complexity of O(V^2) in the given implementation. The selection of the appropriate algorithm depends on specific requirements and the graph's size.

By incorporating these graph traversal algorithms, this project offers a valuable solution for analyzing and finding optimal paths within a given graph. Such applications can prove beneficial in various domains, including route planning, network analysis, and resource allocation.