Ariel Saadon 315434902

Graph Algorithms Project

This project implements a graph data structure along with several graph algorithms in C++. The project provides an undirected weighted graph implementation using an adjacency list representation and includes implementations of the graph algorithms: BFS, DFS, Dijkstra (shortest path), Prim(mst) and Kruskal(mst).

Edge Class

Represents an edge in the graph. Its fields are: the destination vertex (the source vertex is implied by the row in the adjacency list), and a pointer to the next edge in the adjacency list.

Implemented in graph.hpp and graph.cpp

Graph Class

Represents an undirected weighted graph using adjacency lists, supports operations for adding edges, removing edges, and printing the graph.

Each vertex has a linked list of edges ,because it's undirected graph, each edge (u, v) is stored twice: once in u's list and once in v's list

Implemented in graph.hpp and graph.cpp

Algorithms Class

Includes BFS, DFS, Dijkstra's algorithm, Prim's algorithm, and Kruskal's algorithm

Bfs: Uses a queue to track vertices to visit next, marks vertices as visited to avoid cycles, builds a new graph representing the BFS tree.

Dfs: Traverses the graph in depth-first order, returns a new graph representing the DFS forest, implementation uses recursion through the dfs_visit helper function.

Dijkstra: Finds the shortest paths from a source vertex to all other vertices, only works on graphs with non-negative edge weights, returns a graph representing the shortest path tree, uses a priority queue to select the next vertex to process to, use the priority queue to select the next vertex with the minimum distance, Throws an exception if the graph contains negative edge weights

Prim: Finds a minimum spanning tree (MST) of the graph, returns a graph representing the MST, uses a priority queue to select the next edge to add to the MST, builds the MST by connecting vertices with minimum weight edges.

Kruskal: Finds a minimum spanning tree (MST) of the graph, returns a graph representing the MST, uses a Union-Find data structure to detect cycles , processes edges in order of increasing weight, sorts all edges in non-decreasing order of weight, adds edges to the MST if they don't create a cycle

Queue Class

A simple queue implementation used by the BFS algorithm

Implemented in queue.hpp and queue.cpp

PriorityQueue Class

A priority queue implementation based on a min-heap, used by Dijkstra's and Prim's algorithms

Implemented in priority_queue.hpp and priority_queue.cpp

unionFind Class

Implements the Union-Find data structure (disjoint set).

Implemented in union_find.hpp and union_find.cpp

The project uses namespaces to organize the code

All classes are in the graph namespace