Minimum Spanning Tree (MST) Project using Prim's and Kruskal's Algorithms

1 Introduction

This project implements Prim's and Kruskal's algorithms to find the Minimum Spanning Tree (MST) of an undirected, weighted graph. The project consists of six main components organized across separate files, as specified in the assignment. Each component follows a modular design for ease of use and understanding.

2 File Structure

The following files are included in this project:

- `GraphADT.h` and `GraphADT.cpp`: Implements the Graph Abstract Data Type (ADT).
- `Heap.h` and `Heap.cpp`: Implements a Heap for efficient edge selection in Prim's algorithm.
- `Prims.h` and `Prims.cpp`: Contains the function for Prim's algorithm, which takes a `Graph` instance as input and returns an MST.
- `Union.h` and `Union.cpp`: Implements the Union-Find data structure for efficient edge selection in Kruskal's algorithm.
- `Kruskal.h` and `Kruskal.cpp`: Contains the function for Kruskal's algorithm, which uses the `UnionFind` and `Heap` data structures.
- `main.cpp`: The top-level program that takes as input a cost matrix, initializes a graph, and calls both Prim's and Kruskal's algorithms. The results are printed in the specified format along with total costs and runtimes.

3 Input Format

The input is read from a text file named `input.txt`. The file should contain only the adjacency matrix of the graph. Each row represents a node, and the columns represent the weights of edges to other nodes. For non-existent edges or no connection between nodes, use the value `-1` or `0`.

The following is an example of the contents of `input.txt`:

```
0 4 -1 -1 -1 5
4 0 8 -1 -1 -1
-1 8 0 2 -1 -1
-1 -1 2 0 1 -1
-1 -1 -1 1 0 1
5 -1 -1 -1 1 0
```

4 Setup, Compilation, and Running the Program

4.1 Requirements

To compile and run the project, you need:

- A C++ compiler supporting C++11 or later (e.g., `g++`).
- The Standard Library for C++.

4.2 Compilation

To compile all files, use the following command:

```
g++ main.cpp GraphAdt.cpp Heap.cpp Kruskal.cpp Prims.cpp Union.cpp -o assignment
```

This command generates an executable named `assignment.exe`.

4.3 Running the Program

To run the program, use the following command:

```
./assignment
```

The program will:

- Run Prim's algorithm on the graph and print the MST, total cost, and runtime.
- Run Kruskal's algorithm on the graph and print the MST, total cost, and runtime.

5 Output Format

The output format includes:

- `MST via Prim's Algorithm`: A list of edges in the MST with their respective weights.
- `Total Cost of MST via Prim's Algorithm`: The sum of the weights of all edges in the MST.
- `Runtime of Prim's Algorithm`: The time taken to compute the MST.
- `MST via Kruskal's Algorithm`: A list of edges in the MST with their respective weights.
- `Total Cost of MST via Kruskal's Algorithm`: The sum of the weights of all edges in the MST.
- `Runtime of Kruskal's Algorithm`: The time taken to compute the MST.

6 Example

Suppose you have an input file named `input.txt` with the following content:

```
0 4 -1 -1 -1 5
4 0 8 -1 -1 -1
-1 8 0 2 -1 -1
-1 -1 2 0 1 -1
-1 -1 -1 1 0 1
5 -1 -1 -1 1 0
```

Run the following command:

./assignment

The output will display the MSTs generated by Prim's and Kruskal's algorithms, the total cost of each MST, and their respective runtimes. Sample output:

```
Prim's algorithm MST:
Total cost- 13
(2 - 1)
(3 - 4)
(4 - 5)
(5 - 6)
(6 - 1)
Prim's Algorithm running time: 0.00408 seconds

Kruskal's algorithm MST:
Total cost- 13
(4 - 5)
(5 - 6)
(3 - 4)
(1 - 2)
(1 - 6)
Kruskal's Algorithm running time: 0.0015055 seconds
```