Assignment 3 : Minimum Spanning Tree

1 Problem statement

This assignment involves implementing the Prim and Kruskal algorithms for finding a Minimum Spanning Tree (MST) of an undirected, weighted graph

2 Introduction

This document describes the implementation of Prim's and Kruskal's algorithms to find the Minimum Spanning Tree (MST) of an undirected, weighted graph represented by the following cost matrix:

$$\begin{bmatrix} 0 & 6 & 1 & 5 & -1 & -1 \\ 6 & 0 & 5 & -1 & 3 & -1 \\ 1 & 5 & 0 & 5 & 6 & 4 \\ 5 & -1 & 5 & 0 & -1 & 2 \\ -1 & 3 & 6 & -1 & 0 & 6 \\ -1 & -1 & 4 & 2 & 6 & 0 \end{bmatrix}$$

Here, a value of -1 represents the absence of an edge between nodes.

3 Classes and Functions

This implementation uses several classes and functions:

- **Edge Structure**: Represents each edge in the graph with source, destination, and weight attributes.
- **Node Structure**: Used for the linked list-based priority queue in Prim's algorithm, storing a vertex and key.
- **PriorityQueue Class**: Supports operations for managing nodes by key values in Prim's algorithm:
  - `insert`: Adds a node to the queue.
  - `deleteMin`: Removes and returns the node with the minimum key.
  - `decreaseKey`: Updates the key of a specified node.

- **Graph Class**: Represents the undirected, weighted graph and provides functions to add edges and access adjacency lists.
- **UnionFind Class**: Implements the Union-Find data structure to avoid cycles in Kruskal's algorithm. It includes:
  - `find`: Finds the root of a node.
  - `unionSets`: Unites two sets based on rank.
- **printMST Function**: Displays the MST, total weight, and runtime for both algorithms.

4 Prim's Algorithm

Prim's algorithm is a greedy approach that constructs the MST by adding the minimum weight edge from the set of edges that connect nodes already in the MST with those outside the MST. The following steps outline the process:

1. **Initialization**: Start with an arbitrary node (node 0) and set its key value to 0. Insert it into a priority queue based on key values.
2. **Edge Selection**: Extract the minimum key node from the priority queue and add it to the MST.
3. **Key Update**: For each adjacent node of the selected node, if the edge weight is less than the current key of the adjacent node, update the key in the priority queue.
4. **Add Edge to MST**: After selecting the node, add the corresponding edge to the MST if it connects to a previously unvisited node.
5. **Repeat** steps 2-4 until all nodes are included in the MST.

This implementation used a linked-list-based priority queue supporting operations like `insert`, `decreaseKey`, and `deleteMin` to efficiently manage the nodes and keys.

5 Kruskal's Algorithm

Kruskal's algorithm constructs the MST by iteratively adding edges in increasing order of weight while avoiding cycles. The steps are as follows:

1. **Edge Sorting**: Sort all edges in ascending order of weight.
2. **Union-Find Initialization**: Use the Union-Find data structure to manage connected components.
3. **Edge Selection and Addition**: For each edge in the sorted list, check if the source and destination nodes belong to different components.

4. **Cycle Check**: Use the `find` operation to check if adding the edge would create a cycle.

5. **Add Edge to MST**: If the nodes are in different components, add the edge to the MST and merge the components using the `union` operation.

6. **Repeat** steps 3-5 until the MST contains exactly $(n - 1)$ edges.

This implementation used a Union-Find data structure to efficiently track components and avoid cycles.

6 Output

The output of the implemented Prim's and Kruskal's algorithms for the given cost matrix is as follows:

6.1 Prim's Algorithm Output

```
Minimum Spanning Tree using Prim's Algorithm:
Edges in the MST:
(1,3) with weight 1
(2,3) with weight 5
(2,5) with weight 3
(3,6) with weight 4
(4,6) with weight 2
Total Weight of MST: 15
```

6.2 Kruskal's Algorithm Output

```
Minimum Spanning Tree using Kruskal's Algorithm:
Edges in the MST:
(1,3) with weight 1
(2,5) with weight 3
(2,3) with weight 5
(3,6) with weight 4
(4,6) with weight 2
Total Weight of MST: 15
```

6.3 Execution Time

```
Execution time for Prim's Algorithm: 17 seconds
Execution time for Kruskal's Algorithm: 41 seconds
```