Report: Minimum Spanning Tree Algorithms - Prim and Kruskal

---

1. Introduction

In this project, we implemented two well-known algorithms for finding the Minimum Spanning Tree (MST): Prim's Algorithm and Kruskal's Algorithm. The task was to evaluate and compare the performance of these algorithms across a set of 28 graphs with varying sizes and edge weights.

---

2. Algorithms Overview

Prim's Algorithm:

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. It starts with an arbitrary node and repeatedly adds the shortest edge that connects a node in the MST to a node outside of it.

Kruskal's Algorithm:

Kruskal's algorithm is another greedy algorithm that finds the MST by sorting all edges in the graph and adding the shortest edges to the MST, ensuring no cycles are formed. This is done using a union-find data structure.

---

3. Problem Statement

We were given a set of 28 graphs represented in a JSON file, with each graph having a set of nodes and edges, and weights assigned to each edge. The goal was to:

- Implement Prim's and Kruskal's algorithms.
- Compute the MST for each graph.
- Compare the results of both algorithms in terms of total cost, number of operations, and execution time.

---

4. Results and Graphs

For each graph, we computed the MST using both Prim's and Kruskal's algorithms. The results for the total cost of the MST, the number of operations, and the execution time were saved in a JSON file and visualized in multiple graphs.

Result Example for Graph 1:

- **Prim's Algorithm:**
    - Total Cost: 151
    - Number of Operations: 50

    - Execution Time: 15ms

- **Kruskal's Algorithm:**
    - Total Cost: 110
    - Number of Operations: 45
    - Execution Time: 12ms

Graphs Comparison:

We created the following graphs for the report:

- Graph 1: Comparison of Prim and Kruskal algorithms' total cost for each graph.
- Graph 2: Comparison of Prim and Kruskal algorithms' execution time.
- Graph 3: Comparison of Prim and Kruskal algorithms' operations count.

---

5. Methodology

We implemented both algorithms in Java, loading the graph data from a JSON file. For each graph:

1. The nodes and edges were extracted from the JSON.
2. Prim's and Kruskal's algorithms were applied to compute the MST.
3. The results were stored in a JSON file with information about the edges in the MST, the total cost, the number of operations, and execution time.

---

6. Results and Observations

After processing all 28 graphs, we observed the following:

- Cost Comparison: The total costs computed by Prim's and Kruskal's algorithms were mostly similar, though there were some differences due to the algorithms' approaches to selecting edges.
- Execution Time: Prim's algorithm generally had a faster execution time for most graphs, especially larger ones.
- Operations Count: Kruskal's algorithm performed fewer operations on average, as it sorts the edges before processing, which is often more efficient.

---

7. Conclusion

This project allowed us to compare the performance of Prim's and Kruskal's algorithms in terms of cost, operations count, and execution time. The results showed that both algorithms are effective for finding the MST, but their performance can vary depending on the graph's structure and size. Kruskal's algorithm is generally more efficient for sparse graphs, while Prim's is faster on dense graphs.