Assignment 3: Optimization of a City Transportation Network (Minimum Spanning Tree)

The goal of this assignment was to apply Prim's and Kruskal's algorithms to find the Minimum Spanning Tree (MST) of a city's transportation network. The MST ensures that all areas of the city are connected using the least total cost without forming any cycles. Both algorithms were implemented in Java, tested with sample inputs, and analyzed for their performance and efficiency.

1. Summary of Input Data and Algorithm Results

Three different test graphs were used to compare the performance of Prim's and Kruskal's algorithms. Each graph varied in terms of nodes and edge density. The results were stored in 'ComparisonSummary.csv', while the input and output were managed through JSON files. The summary is shown below:

Graph ID	Nodes	Edges	Prim Cost	Kruskal Cost	Same Cost	Valid MST	Prim Time (ms)	Kruskal Time (ms)	Observation
1	4	5	6	6	Yes	Yes	7.483	3.281	Kruskal faster
2	6	9	13	13	Yes	Yes	0.137	0.105	Kruskal faster
3	10	14	25	25	Yes	Yes	0.218	0.151	Kruskal faster

Both algorithms produced the same MST cost for all test cases, confirming correctness. However, Kruskal's algorithm consistently executed faster, likely due to its simpler structure and fewer heap operations.

2. Theoretical and Practical Comparison

Prim's Algorithm

- Works by growing the MST one vertex at a time.
- It starts from a node and repeatedly adds the cheapest edge connecting a visited node to an unvisited one.
- Best suited for dense graphs where many edges exist.
- In this implementation, a PriorityQueue (Min-Heap) was used to pick minimum edges efficiently.
- Operation count increases due to frequent updates in the priority queue.

Kruskal's Algorithm

- Works by sorting all edges by weight and adding them one by one, avoiding cycles using Union-Find.
- Best suited for sparse graphs because it mainly depends on the number of edges.
- Simpler to implement and often faster for smaller datasets.
- Fewer operations since the sorting is done once, and the Union-Find checks are efficient.

Practical Results

- For all test cases, Kruskal's algorithm completed faster and required fewer operations.
- Both algorithms produced identical MST total costs, showing they are both correct and reliable.
- Time differences were small but consistent, showing Kruskal's advantage in practical performance.

3. Conclusions

Based on both theory and experimental results, the following conclusions can be made:

- Kruskal's algorithm showed better performance in all test cases in terms of time.
- Prim's algorithm is more suitable for dense graphs where adjacency lists are used.
- Both algorithms produced identical MST costs, confirming that both are correct and reliable.
- Kruskal's algorithm is simpler and easier to code for smaller datasets.
- Prim's algorithm can be optimized for large graphs using adjacency lists and heaps.
- Real-world applications like road network optimization and utility mapping can use either algorithm depending on the graph structure.

4. References

1. GeeksforGeeks – 'Prim's and Kruskal's Algorithm for Minimum Spanning Tree'
2. Class lecture notes and assignment guidelines
3. Java documentation for PriorityQueue and Union-Find classes