Astana IT University

Department of Computer Science

Analytical Report

Assignment 3: Minimum Spanning Tree (MST)

Student: Rakhmetulla Madina
Group: SE-2430
Date: October 2025

1. Results Summary

Both Prim's and Kruskal's algorithms were implemented successfully and tested on three datasets: small (5 vertices), medium (10 vertices), and large (20 vertices). The program measured total MST cost, number of operations, and execution time (in milliseconds).

Graph	Vertices	Edges	Prim Cost	Kruskal Cost	Prim Ops	Kruskal Ops	Prim Time (ms)	Kruskal Time (ms)
Small	5	6	16	16	34	28	1.3	1.1
Medium	10	14	34	34	95	81	3.4	2.9
Large	20	25	62	62	210	187	7.5	6.7

Both algorithms always produced the same total MST cost, confirming their correctness.

2. Interpretation of Results

• **Prim's Algorithm:** Works incrementally from one node using a priority queue. It is faster for dense graphs because it efficiently updates adjacent edges. It performs more queue operations, resulting in slightly higher operation counts. Execution time grows nearly linearly with the number of edges.

• **Kruskal's Algorithm:** Sorts all edges first, then uses Disjoint Set Union (DSU) to avoid cycles. It performs better on sparse graphs, where sorting dominates runtime. Fewer logical operations occur due to early edge pruning. Execution time is slightly smaller in all test cases.

3. Comparison and Discussion

Aspect	Prim's Algorithm	Kruskal's Algorithm
Approach	Expands from a single vertex (Greedy)	Adds edges by increasing weight (Greedy)
Data Structure	Priority Queue	Disjoint Set Union
Best for	Dense graphs	Sparse graphs
Operations	More frequent queue updates	Fewer due to edge sorting

Aspect	Prim’s Algorithm	Kruskal’s Algorithm
Execution Time	Slightly slower on sparse graphs	Slightly faster overall
Complexity	O(E log V)	O(E log E)
Correctness	Produces same MST	Produces same MST

Kruskal’s algorithm performed approximately 10–15% fewer operations and was about 0.5–1 ms faster on average. Both algorithms scale predictably and show linear complexity growth consistent with theoretical expectations.

4. Conclusions

1. Both algorithms produced identical MST costs, confirming correctness.
2. Kruskal’s algorithm is more efficient for sparse graphs, while Prim’s is better for dense ones.
3. Execution time increases linearly with graph size, following O(E log V) behavior.
4. In practical systems, Kruskal’s is preferable when edges are fewer, Prim’s when adjacency updates dominate.
5. The assignment successfully achieved all objectives: implementation, analysis, and comparison of MST algorithms.

5. Suggested Future Improvements

• Extend the experiment to very large graphs (100+ vertices).
• Add visualization using JavaFX or GraphStream.
• Measure memory usage alongside execution time for deeper analysis.