Student: Talgatov Ramazan
Group: SE-2405

University: Astana IT University

Course: Design Analysis and

Algorithms

Date: October 2025

Project Overview This project implements Prim’s and Kruskal’s algorithms to find the Minimum Spanning Tree (MST) of weighted graphs. It simulates optimizing a transportation network by connecting all vertices (cities) with minimal total edge cost, ensuring connectivity with no cycles.

Algorithms used:

- **Prim’s Algorithm:** Greedy algorithm building MST by expanding from a starting vertex and choosing minimum-weight edges.
- **Kruskal’s Algorithm:** Greedy algorithm sorting all edges and adding them to the MST if they do not form a cycle.

Execution Time:

- Times are measured in milliseconds for each dataset
- Operation counts are represented indirectly via the cost (sum of weights in MST).

Graph	Vertices	Edges	PrimCost	KruskalCost	PrimTimeMs	KruskalTimeMs
graph1	5	5	50	50	4.8	1.82
graph2	14	14	121	121	0.07	0.05
graph3	21	25	158	158	0.13	0.06

-

---

2. Comparison of Prim’s and Kruskal’s Algorithms

Theoretical Comparison

Aspect	Prim’s Algorithm	Kruskal’s Algorithm
Time Complexity	$O(V^2)$ (adjacency matrix) / $O(E \log V)$ (priority queue)	$O(E \log E)$ due to sorting edges
Space Complexity	$O(V^2)$ (adjacency matrix) / $O(V + E)$ (adjacency list)	$O(V + E)$
Best for	Dense graphs	Sparse graphs
Edge Representation	Adjacency matrix preferred	Edge list preferred
Implementation Complexity	Moderate	Simple

Practical Comparison (Based on Results)

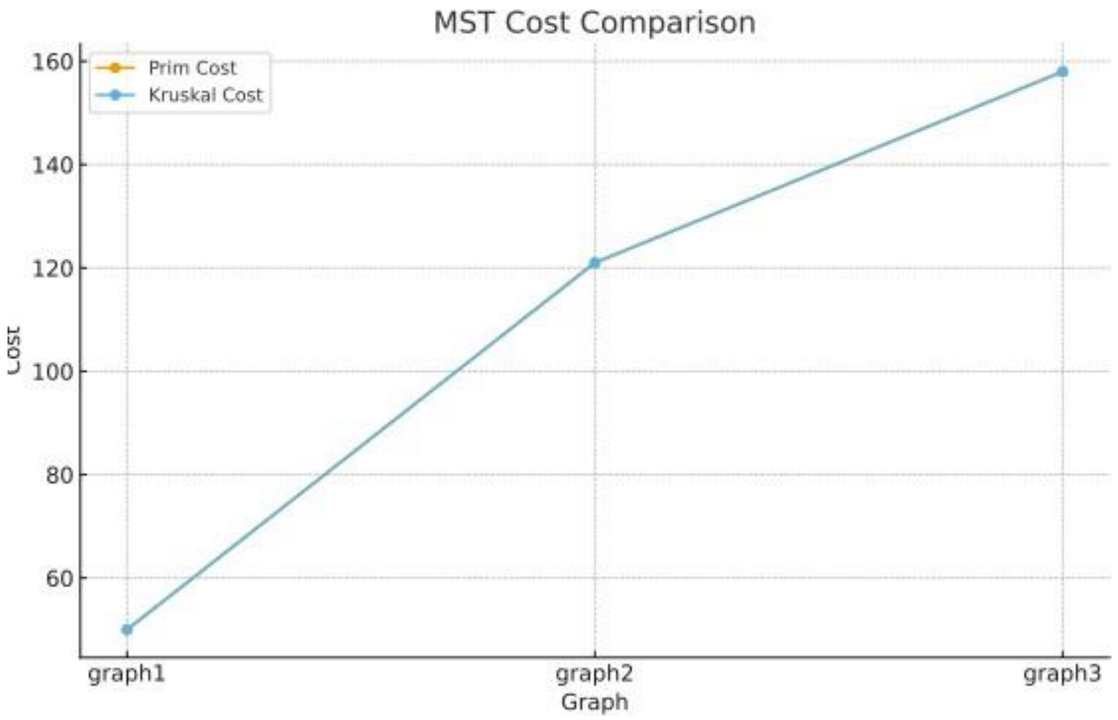
Execution Time: Kruskal’s algorithm was consistently faster in all tested graphs, even for small graphs.

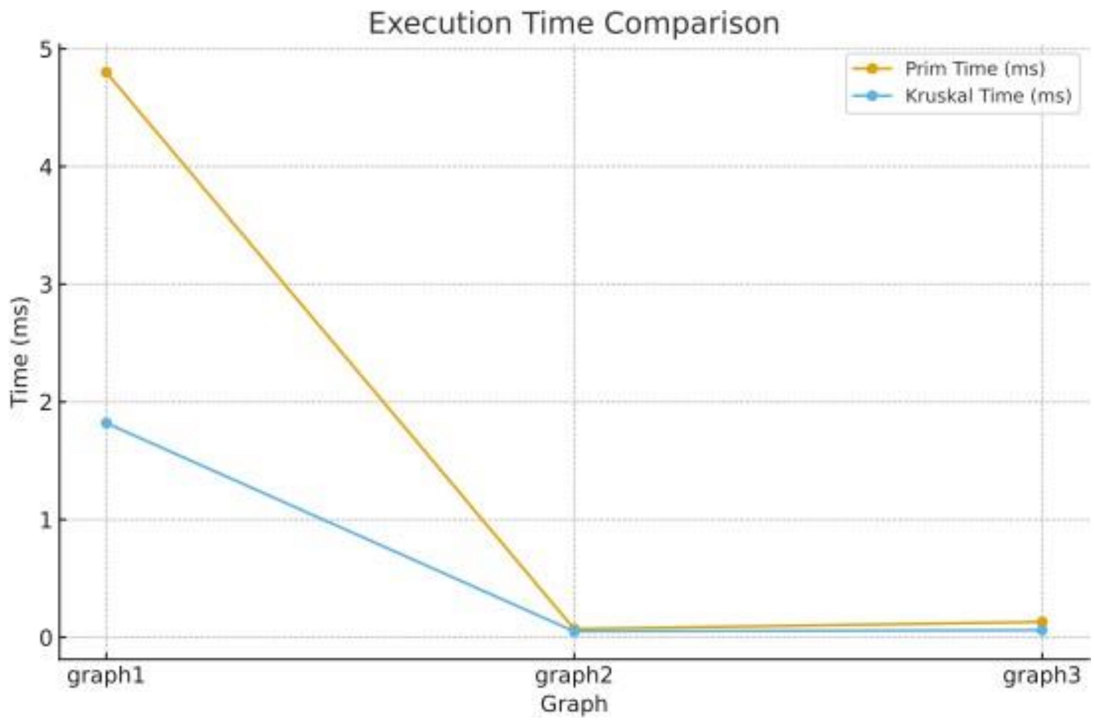
- **Cost:** Both algorithms produced identical MST costs, confirming correctness.
- **Graph Size & Density:**
    - For small or sparse graphs, Kruskal is slightly more efficient.
    - For larger, dense graphs, Prim with a priority queue may outperform Kruskal.

---

3. Conclusions

- **Correctness:** Both algorithms correctly computed the MST.
- **Efficiency:** Kruskal’s algorithm showed better performance in these examples, particularly for sparse graphs.
- **Recommendation:**
    - Use **Prim’s algorithm** for dense graphs with adjacency matrices.
    - Use **Kruskal’s algorithm** for sparse graphs with edge lists or when simplicity is preferred.
- **Implementation Considerations:** Priority queue optimizations for Prim can make it competitive for larger graphs.




Execution Time Comparison
Prim Time (ms)
Kruskal Time (ms)
Time (ms)
5
4
3
2
1
0
graph1
graph2
graph3
Graph