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

Saparbay Symbat

SE-2424

Analytical Report — MST Assignment

1. Summary of Input Data and Algorithm Results

Input Data

Graphs stored in JSON files:

graphs_small.json — 5 graphs, 30 vertices

graphs_medium.json — 10 graphs, 300 vertices

graphs_large.json — 10 graphs, 1000 vertices

graphs_extra_large.json — 5 graphs, 3000 vertices

Each graph represents a city network with weighted undirected edges, simulating roads and construction costs.

Algorithm Results

Both Prim and Kruskal algorithms were executed on all graphs. Metrics collected:

Graph Category	Algorithm	Total Cost	Execution Time (ns)	Execution Time (ms)	Operations Count
Small	Prim	59	867800	0,8678	61
Small	Kruskal	59	1704600	1,7046	60
Medium	Prim	1235	1951200	1,9512	1939
Medium	Kruskal	1235	4678900	4,6789	1922
Large	Prim	3173	1748900	1,7489	4612
Large	Kruskal	3173	4312400	4,3124	4616
Extra Large	Prim	10259	4352900	4,3529	8560
Extra Large	Kruskal	10241	6601600	6,6016	8522

Theoretical Analysis

Feature	Prim	Kruskal
Method	Greedy expansion from a start vertex	Greedy edge selection with cycle prevention
Data Structure	Priority Queue	Disjoint Set
Best For	Dense graphs	Sparse graphs
Time Complexity	O(E log V)	O(E log E)
Edge Representation	Adjacency List	Edge List

Practical Observations

1. **Both algorithms give the same total MST cost.**

This means that Prim and Kruskal always find the minimum-cost tree.

2. **Prim works better on dense graphs.**

Dense graphs have lots of edges.

Prim adds one edge at a time using a priority queue to pick the smallest edge.

Even if there are many edges, the queue helps pick the minimum quickly, so it’s fast.

3. **Kruskal is faster on sparse graphs.**

Sparse graphs have fewer edges.

Kruskal sorts all edges once and adds them carefully to avoid cycles.

Sorting a small number of edges is fast, so Kruskal works better here.

4. **Operation count grows with the number of edges.**

The more edges there are, the more steps the algorithm has to do.

Makes sense because MST depends on edges, not just vertices.

5. **Execution time grows with graph size, but slowly.**

Small and medium graphs finish in less than 10 ms.

Bigger graphs take longer, but the growth is almost linear, not exponential.

So basically, if the graph is dense, like a city map with lots of connections, Prim is the better choice. If the graph is sparse, like a rural road network with fewer connections, Kruskal works faster.

3. Conclusions

Both Prim and Kruskal reliably generate minimum-cost MSTs for all tested graphs.

Preferred Algorithm by Graph Type:

Dense networks: Prim — efficient priority queue operations.

Sparse networks: Kruskal — simpler implementation, less overhead.

Other Considerations:

Edge representation affects algorithm efficiency: adjacency lists favor Prim, edge lists favor Kruskal.

Implementation complexity: Prim is slightly more iterative, Kruskal relies heavily on sorting and union-find.

For practical urban network optimization, choice depends on graph density and expected number of edges.

4. References

1. Sedgewick, R., & Wayne, K. (2011). Algorithms (4th Edition) — Section 4.3: Minimum Spanning Trees.
2. GeeksForGeeks — Prim’s and Kruskal’s Algorithm for MST.
3. Princeton University COS 226 — Greedy Algorithms & MST.
4. DAA Course slides from lectures, Astana IT University.