Design and Analysis of Algorithms | Assignment 3

Test Graphs Overview

Graph ID	Size Category	Vertices	Edges	Description
1	Small	6	9	Simple network for correctness verification
2	Medium	15	24	Moderate complexity network
3	Large	30	54	Complex network for scalability testing

Performance Comparison Results

Summary Table

Graph	Algorithm	MST Cost	Operations	Execution Time (ms)	MST Edges
Graph 1 (Small)	Prim's	25.0	65	0.058	5
	Kruskal's	25.0	65	1.173	5
Graph 2 (Medium)	Prim's	100.0	406	0.207	14
	Kruskal's	100.0	151	0.080	14
Graph 3 (Large)	Prim's	185.0	1,864	0.840	29
	Kruskal's	185.0	317	0.463	29

Detailed Analysis by Graph

Graph 1 - Small Graph (6 vertices, 9 edges)

Metric	Prim's Algorithm	Kruskal's Algorithm
MST Total Cost	25.0	25.0
Operations Count	65	65
Execution Time	0.058 ms	1.173 ms
Edges in MST	5	5

Metric	Prim's Algorithm	Kruskal's Algorithm
Performance	Faster execution	More operations overhead

Key Observations: - Both algorithms produce the same optimal MST cost - Prim's executes **20× faster** despite same operation count - Small graph favors Prim's simpler implementation

---

Graph 2 - Medium Graph (15 vertices, 24 edges)

Metric	Prim's Algorithm	Kruskal's Algorithm
MST Total Cost	100.0	100.0
Operations Count	406	151
Execution Time	0.207 ms	0.080 ms
Edges in MST	14	14
Performance	More operations	Faster execution

Key Observations: - Kruskal's performs **62.8% fewer operations** (406 vs 151) - Kruskal's executes **2.6× faster** (0.207 ms vs 0.080 ms) - Efficiency crossover point appears between small and medium graphs

---

Graph 3 - Large Graph (30 vertices, 54 edges)

Metric	Prim's Algorithm	Kruskal's Algorithm
MST Total Cost	185.0	185.0
Operations Count	1,864	317
Execution Time	0.840 ms	0.463 ms
Edges in MST	29	29
Performance	Slower scaling	Better scalability

Key Observations: - Kruskal's performs **83.0% fewer operations** (1,864 vs 317) - Kruskal's executes **1.8× faster** (0.840 ms vs 0.463 ms) - Performance gap widens significantly with graph size

---

Algorithm Complexity Analysis

Operations Count Growth

Graph Size	Prim's Ops	Kruskal's Ops	Ratio (Prim/Kruskal)
Small (6V, 9E)	65	65	1.0×
Medium (15V, 24E)	406	151	2.7×
Large (30V, 54E)	1,864	317	5.9×

Trend: Prim's operation count grows much faster (quadratic behavior)

Execution Time Growth

Graph Size	Prim's Time (ms)	Kruskal's Time (ms)	Speedup
Small (6V, 9E)	0.058	1.173	0.05× (slower)
Medium (15V, 24E)	0.207	0.080	2.59×
Large (30V, 54E)	0.840	0.463	1.81×

Key Findings

Correctness

- **Both algorithms produce identical MST costs** across all test cases
- All graphs connected successfully with V-1 edges

Performance Patterns

Prim's Algorithm: - Better for very small graphs (< 6 vertices) - O(E · V) complexity visible in operation counts - Operations grow: 65 → 406 → 1,864 (quadratic)

Kruskal's Algorithm: - Better for medium to large graphs (15 vertices) - O(E log E) complexity advantage clear - Operations grow: 65 → 151 → 317 (logarithmic) - Consistently fewer operations at scale

Recommendations

Graph Size	Recommended Algorithm	Reason
Small (< 10 vertices)	**Prim's**	Simpler, faster on tiny graphs

Graph Size	Recommended Algorithm	Reason
Medium (10-20 vertices)	**Kruskal's**	Better operation efficiency
Large (20+ vertices)	**Kruskal's**	Significantly faster, scales better

---

Implementation Notes

- **Prim's Implementation:** Basic O(E · V) without priority queue optimization
- **Kruskal's Implementation:** Uses Union-Find with path compression
- All tests run on Java 11 with Maven
- Results averaged from multiple runs for consistency

Conclusion

The analysis demonstrates that **Kruskal's algorithm scales significantly better** than the basic Prim's implementation for graphs with more than 10 vertices. While Prim's shows better performance on tiny graphs, Kruskal's superior O(E log E) complexity with Union-Find optimization makes it the clear choice for real-world applications involving moderate to large networks.

Author

Student: Nurzhan Izimbetov **Group**: SE-2436 **Github**: nurzhqn0
