**Minimum Spanning Tree Algorithms Analytical Report**

**Executive Summary**

This report analyzes Prim's and Kruskal's algorithms for finding Minimum Spanning Trees (MST) in weighted undirected graphs, highlighting their correctness, efficiency, and practical applications—especially in transportation networks. Both algorithms are implemented and evaluated based on multiple performance metrics and real-world scenarios.

**Algorithm Implementations**

- Prim's Algorithm

- Time Complexity: O(E log V) using binary heap

- Space Complexity: O(V)

- Approach: Greedy, expands MST by adding minimum weight edge connecting known MST to unknown nodes

- Kruskal's Algorithm

- Time Complexity: O(E log E) due to sorting

- Space Complexity: O(V) for union-find data structure

- Approach: Greedy, adds next-lightest edge that does not form a cycle using edge sorting and union-find

**Experimental Results**

Test Case 1 (5 vertices, 7 edges):

- Prim's: MST Cost = 16.00, Operations = 47, Time = 2.97 ms

- Kruskal's: MST Cost = 16.00, Operations = 36, Time = 0.33 ms

Test Case 2 (4 vertices, 5 edges):

- Prim's: MST Cost = 6.00, Operations = 35, Time = 0.02 ms

- Kruskal's: MST Cost = 6.00, Operations = 26, Time = 0.01 ms

Both algorithms generated identical MST costs in tested graphs, demonstrating correct MST properties: full connectivity, no cycles, minimum cost.

**Performance Analysis**

- Operations Count: Kruskal's tends to be faster with fewer operations due to early stopping when MST found

- Prim's: More operations, especially in large/dense graphs, from priority queue management

- Execution Time: Kruskal's is generally quicker for the same graphs, though exact timings vary by graph structure

- Scalability: Prim's better for dense graphs, Kruskal's for sparse graphs

**Applications & Recommendations**

- Urban and transportation network design: Both algorithms optimize costs and ensure all nodes/districts are connected

- Economic Impact: Lower infrastructure cost yields substantial savings for large-scale networks

- Recommendations:

- Use Kruskal's for sparse graphs or when early termination is useful

- Use Prim's for dense networks or memory-constrained environments

**Technical Implementation Notes**

- Key optimization methods: Path compression, union by rank, and binary heaps are critical for efficient scaling

- Ongoing improvements focus on parallelization and adapting to temporal and incremental graph changes