**Report.**  
**Assignment 3.**

Sadvokassova Aliya

SE – 2405

**1. Summary of Input Data and Algorithm Results**

Two graph datasets were analyzed to determine the Minimum Spanning Tree (MST) using **Prim’s** and **Kruskal’s** algorithms.  
Each graph represents a city transportation network where vertices are districts, edges are possible roads, and weights indicate road construction costs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Graph ID** | **Vertices** | **Edges** | **Algorithm** | **Total Cost** | **Operations Count** | **Execution Time (ms)** |
| **1** | 5 | 7 | Prim | 16 | 28 | 1.1767 |
|  |  |  | Kruskal | 16 | 7 | 0.616 |
| **2** | 4 | 5 | Prim | 6 | 15 | 0.0108 |
|  |  |  | Kruskal | 6 | 5 | 0.0115 |

**Interpretation:**

* For both graphs, Prim’s and Kruskal’s algorithms produced identical MST costs, confirming correctness.
* Kruskal’s algorithm generally required fewer operations and executed slightly faster, particularly for Graph 1 (denser network).
* The difference in performance between the two algorithms becomes more noticeable as the number of edges increases.

**2. Detailed MST Results (from output.json)**

Graph 1

Vertices: 5  Edges: 7

Prim’s Algorithm MST Edges:

A – C (weight 3)

B – C (weight 2)

B – D (weight 5)

D – E (weight 6)  
Total Cost: 16  Operations: 28  Execution Time: 1.1767 ms

Kruskal’s Algorithm MST Edges:

B – C (weight 2)

A – C (weight 3)

B – D (weight 5)

D – E (weight 6)  
Total Cost: 16  Operations: 7  Execution Time: 0.616 ms

Graph 2

Vertices: 4  Edges: 5

Prim’s Algorithm MST Edges:

A – B (weight 1)

B – C (weight 2)

C – D (weight 3)  
Total Cost: 6  Operations: 15  Execution Time: 0.0108 ms

Kruskal’s Algorithm MST Edges:

A – B (weight 1)

B – C (weight 2)

C – D (weight 3)  
Total Cost: 6  Operations: 5  Execution Time: 0.0115 ms

**3. Comparison Between Prim’s and Kruskal’s Algorithms**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Prim’s Algorithm** | **Kruskal’s Algorithm** |
| **Approach** | Expands from a starting vertex, adding the smallest edge connecting the tree to a new vertex. | Sorts all edges and adds the smallest edges while avoiding cycles. |
| **Main Data Structures** | Visited set and edge list (can be optimized with a priority queue). | Sorted edge list and Disjoint Set (Union-Find). |
| **Time Complexity** | O(V²) with adjacency list; O(E log V) using a heap. | O(E log E) due to sorting and union–find operations. |
| **Efficiency in Tests** | More operations (28 and 15). Slightly slower on denser graphs. | Fewer operations (7 and 5). Generally faster. |
| **Memory Usage** | Based on vertex representation. | Based on edge representation. |
| **Best For** | Dense graphs or when adjacency structure is used. | Sparse graphs or when edges are easily sorted. |
| **Implementation Complexity** | Moderate. | Simple and clear. |

**Interpretation:**

* Kruskal’s algorithm tends to be more efficient on sparse networks because it avoids repeated vertex-based searches.
* Prim’s algorithm can be more efficient on dense graphs, especially when implemented with a min-heap (priority queue) structure.
* In both tests, Kruskal’s method achieved lower operation counts, demonstrating better performance for the given data.

**4. Conclusions**

1. Both algorithms successfully produced Minimum Spanning Trees with identical total costs, proving their correctness.
2. Kruskal’s algorithm was generally more efficient in terms of operations and execution time for the tested graphs.
3. Prim’s algorithm may perform better on dense graphs or when using optimized data structures like heaps.
4. In practical applications:
   1. Use Kruskal when the graph is sparse or represented as an edge list.
   2. Use Prim when the graph is dense or represented as an adjacency matrix/list.
5. Both algorithms remain fundamental tools for network optimization and infrastructure planning problems.

**5. References**

[Difference between Prim's and Kruskal's algorithm for MST - GeeksforGeeks](https://www.geeksforgeeks.org/dsa/difference-between-prims-and-kruskals-algorithm-for-mst/)[Kruskal's and Prim's Algorithms for Minimum Spanning Trees | Abdul Wahab Junaid](https://awjunaid.com/algorithm/kruskals-and-prims-algorithms-for-minimum-spanning-trees/)[Prim’s Algorithm for Minimum Spanning Tree (MST) - GeeksforGeeks](https://www.geeksforgeeks.org/dsa/prims-minimum-spanning-tree-mst-greedy-algo-5/)

[Kruskal’s Minimum Spanning Tree (MST) Algorithm - GeeksforGeeks](https://www.geeksforgeeks.org/dsa/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/)