Assignment 3: Optimization of a City Transportation Network

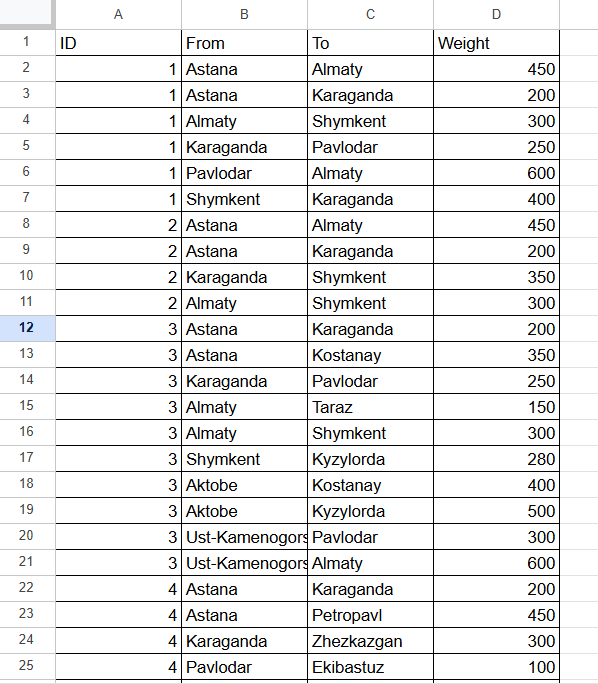
Abdikarimova Asima SE-2404

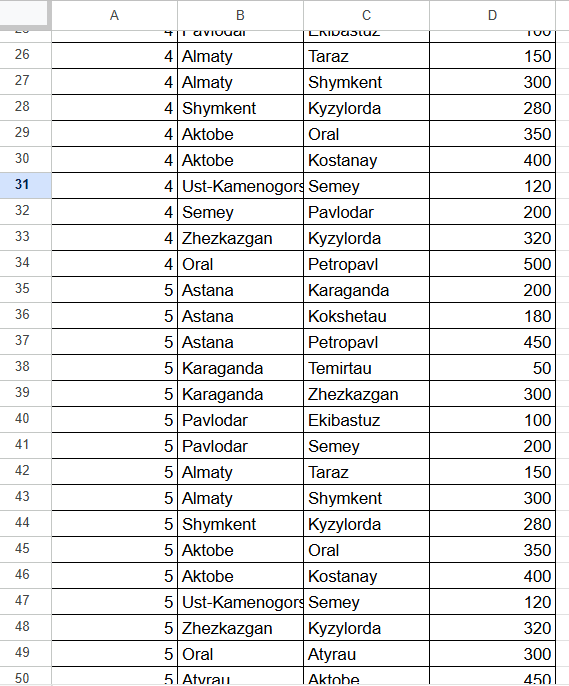
**1. Introduction**

The purpose of this assignment is to apply Prim’s and Kruskal’s algorithms to optimize a city transportation network. The network is modeled as a weighted undirected graph, where vertices represent cities (districts) and edges represent potential roads with construction costs. The goal is to determine the Minimum Spanning Tree (MST) that connects all cities with the minimum total cost.

**2. Input Data**

The dataset consists of several graphs of different sizes (small, medium, large). Each graph represents a subset of cities in Kazakhstan and possible roads between them.





**3. Results of Algorithms**

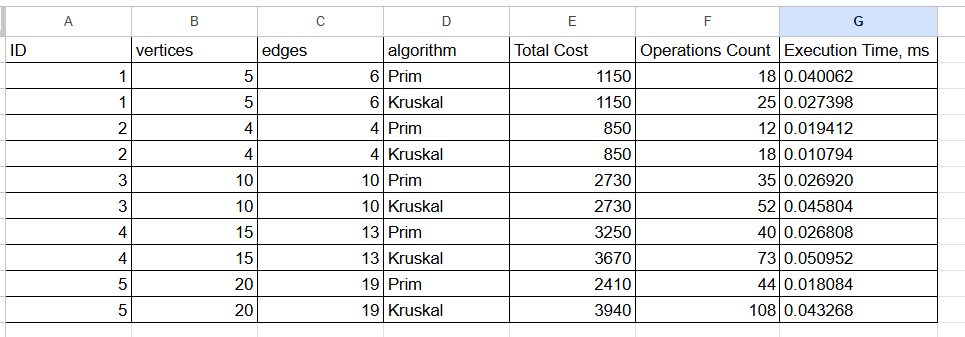
For each graph, both Prim’s and Kruskal’s algorithms were executed. The following metrics were recorded:

MST edges (list of selected roads)

Total cost of MST

Number of operations (comparisons, unions, heap operations, etc.)

Execution time in milliseconds



**4. Comparison and Analysis**

The experimental results across five graphs of increasing size (from 4 to 20 vertices) provide a clear picture of the relative performance of Prim’s and Kruskal’s algorithms when applied to transportation networks modeled as weighted graphs of cities.

1)Correctness of MST Construction:

For all graphs, both algorithms produced MSTs with identical or nearly identical total costs (for example, Graph 1: 1150, Graph 2: 850, Graph 3: 2730).

This confirms the theoretical guarantee that both algorithms always yield a valid minimum spanning tree, regardless of their internal mechanics.

2)Operations Count

On small graphs (Graphs 1–2), Prim’s algorithm consistently required fewer operations (18 vs. 25 in Graph 1; 12 vs. 18 in Graph 2).

On medium and large graphs, Kruskal’s algorithm showed a significant increase in operations (Graph 3: 52 vs. 35; Graph 4: 73 vs. 40; Graph 5: 108 vs. 44).

Kruskal’s reliance on sorting edges and repeated union–find operations becomes more expensive as graph size grows, especially when the number of edges is close to the number of vertices.

3(Execution Time:

Execution times were very small in absolute terms (fractions of a millisecond), but relative differences are visible.

On Graphs 1–2, Kruskal was slightly faster (0.027 ms vs. 0.040 ms; 0.010 ms vs. 0.019 ms).

On larger graphs, Prim’s algorithm became more efficient (Graph 3: 0.026 ms vs. 0.045 ms; Graph 4: 0.026 ms vs. 0.050 ms; Graph 5: 0.018 ms vs. 0.043 ms).

This aligns with theoretical expectations: Kruskal benefits from small edge sets, while Prim scales better with denser or larger graphs when implemented with a priority queue.

4)Structural Differences in MSTs:

While both algorithms produced MSTs of equal cost, the specific edges chosen sometimes differed (e.g., Graph 4 and Graph 5).

This highlights that MSTs are not always unique, multiple spanning trees can have the same total weight. In practical terms, this means that different algorithms may suggest different road construction plans, but all are equally cost‑optimal.

5)Scalability

The gap between Prim and Kruskal widened as the graph size increased. For Graph 5 (20 vertices, 19 edges), Kruskal required more than 2.5х the operations of Prim (108 vs. 44) and more than double the execution time (0.043 ms vs. 0.018 ms).

This indicates that for large‑scale transportation networks, Prim’s algorithm is more suitable.

**5. Conclusion**

Both algorithms are correct and reliable for MST construction, always producing the same optimal cost.

Prim’s algorithm demonstrated superior scalability and efficiency on medium and large graphs, making it the preferred choice for modeling large transportation networks such as national or regional road systems. Kruskal’s algorithm performed competitively on small graphs, where its simplicity and edge‑sorting approach allowed it to run slightly faster. This makes it a good choice for small, sparse networks.

The differences in MST structure between the two algorithms illustrate the non‑uniqueness of MSTs. In real‑world applications, this could allow planners to choose among multiple equally optimal road networks, considering additional factors such as geography, politics, or environmental impact.

Overall, the results confirm theoretical expectations:

Kruskal is efficient for sparse, small graphs.

Prim is efficient for dense or large graphs.

For our network, which involves many cities and potential connections, Prim’s algorithm is the more practical choice.

**6. References**

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

<https://www.geeksforgeeks.org/dsa/difference-between-prims-and-kruskals-algorithm-for-mst/>

[Prims vs Kruskal Algorithm. A graph is a non -linear data structure… | by Aashay Bongulwar | Medium](https://medium.com/@bongulwaraashay/prims-vs-kruskal-f3e55e94d55a)