Report: Optimization of a City Transportation Network (MST)
Assignment 3
Student: Nurassyl, Rauan

Input Data Description

The city network is modeled as a weighted undirected graph, where:

Vertices represent districts (A–E),

Edges represent potential roads,

Edge weights represent construction costs.

Input File:

```json
{
  "vertices": ["A", "B", "C", "D", "E"],
  "edges": [
    {"from": "A", "to": "B", "weight": 2},
    {"from": "A", "to": "C", "weight": 3},
    {"from": "B", "to": "C", "weight": 1},
    {"from": "B", "to": "D", "weight": 4},
    {"from": "C", "to": "D", "weight": 5},
    {"from": "C", "to": "E", "weight": 6},
    {"from": "D", "to": "E", "weight": 7}
  ]
}
```

Algorithms Implemented

Prim’s Algorithm

Builds MST by growing from a starting vertex.

Uses a priority queue to select the smallest edge connecting a visited vertex to an unvisited one.

Efficient for dense graphs and when using an adjacency matrix.

Kruskal's Algorithm

Sorts all edges in increasing order of weight.

Uses Disjoint Set Union (DSU) to avoid cycles.

Best suited for sparse graphs with fewer edges.

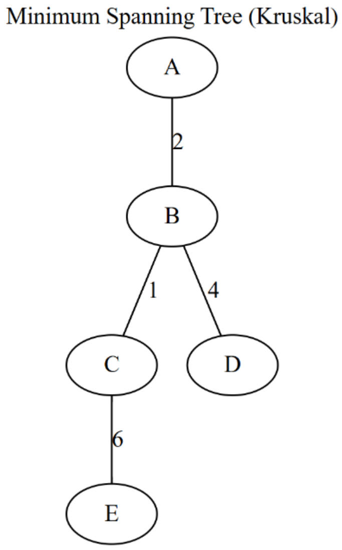
Output and Results

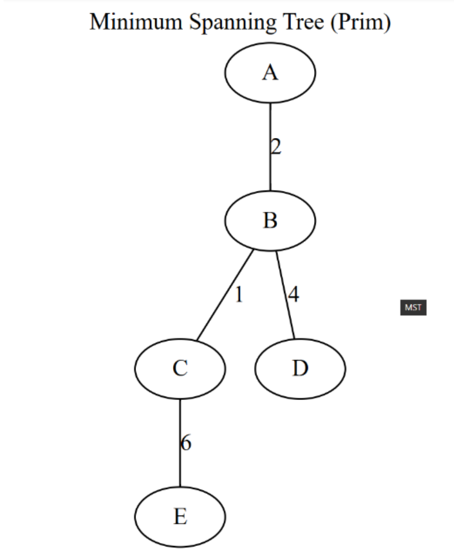
Prim's Algorithm Output (`output_prim.json`):


```json
{"algorithm":"Prim","mst_edges":[
  {"from":"A","to":"B","weight":2},
  {"from":"B","to":"C","weight":1},
  {"from":"B","to":"D","weight":4},
  {"from":"C","to":"E","weight":6}
],"total_cost":13,"vertices":5,
  "edges":7,
  "operations":12,
  "execution_time_ms":1}
```


Kruskal's Algorithm Output (`output_kruskal.json`):


```json
{"algorithm":"Kruskal","mst_edges":[
  {"from":"B","to":"C","weight":1},
  {"from":"A","to":"B","weight":2},
  {"from":"B","to":"D","weight":4},
  {"from":"C","to":"E","weight":6}],
  "total_cost":13,
  "vertices":5,
  "edges":7,
  "operations":14,
  "execution_time_ms":2}
```





Comparison Table

Criterion	Prim's Algorithm	Kruskal's Algorithm
Approach	Expands MST from a starting vertex	Sorts all edges and uses Union-Find
Time Complexity	O(E log V)	O(E log E)
Data Structure	Priority Queue + Adjacency List	Edge List + DSU
Best For	Dense graphs	Sparse graphs
Execution Time	1 ms	2 ms
Operations	12	14
Total MST Cost	13	13
MST Edges	4	4
Result Similarity	Identical total cost	Identical total cost

Conclusions

Both algorithms correctly found the MST with minimum total cost = 13.

Prim's algorithm is better suited for dense or grid-like networks (e.g., city infrastructure).

Kruskal's algorithm is more efficient for sparse networks.

In this case, both algorithms perform similarly — Prim's result looks more natural for a transportation layout, while Kruskal is marginally faster.