REPORT 3

Design And Analysis of Algorithms

Assignment 3

Gaisa Aldiyar

Rakmetollayev Askhat

SE-2416

Analytical Report: Optimization of City Transportation Network (Minimum Spanning Tree)

Introduction

This report presents the application and analysis of Prim’s and Kruskal’s algorithms for optimizing a city’s transportation network. The objective is to connect all districts in the city with a minimal total construction cost, modeled as a minimum spanning tree (MST) problem on a weighted undirected graph.

Kruskal's and Prim's algorithms are two classic greedy algorithms used to find a Minimum Spanning Tree (MST) of a weighted, connected, undirected graph. An MST is a subset of edges that connects all vertices together without any cycles and with the minimum possible total edge weight.

Kruskal's Algorithm

Kruskal's algorithm works by: Askhat

Sorting all edges of the graph in non-decreasing order by weight.

Starting with an empty spanning tree, the algorithm iteratively adds the next lightest edge to the tree, provided it does not create a cycle.

A disjoint-set (Union-Find) data structure is typically used to efficiently check whether adding an edge will form a cycle.

The algorithm continues until all vertices are connected.

It is well suited for sparse graphs and when edges can be easily sorted.

Prim's Algorithm

Prim's algorithm works by: Aldiyar

Starting from an arbitrary vertex and growing the MST one vertex at a time.

At each step, it adds the smallest weighted edge that connects a vertex in the MST to a vertex outside it.

Typically implemented with a priority queue to quickly find the lowest-weight edge.

It continues until all vertices are included in the MST. Prim's algorithm is efficient for dense graphs and when the graph is represented with adjacency structures.

The city administration plans to construct roads connecting all districts in such a way that:

1)each district is reachable from any other district;

2)the total cost of construction is minimized.

This scenario is modeled as a weighted undirected graph, where:

1)vertices represent city districts,

2)edges represent potential roads,

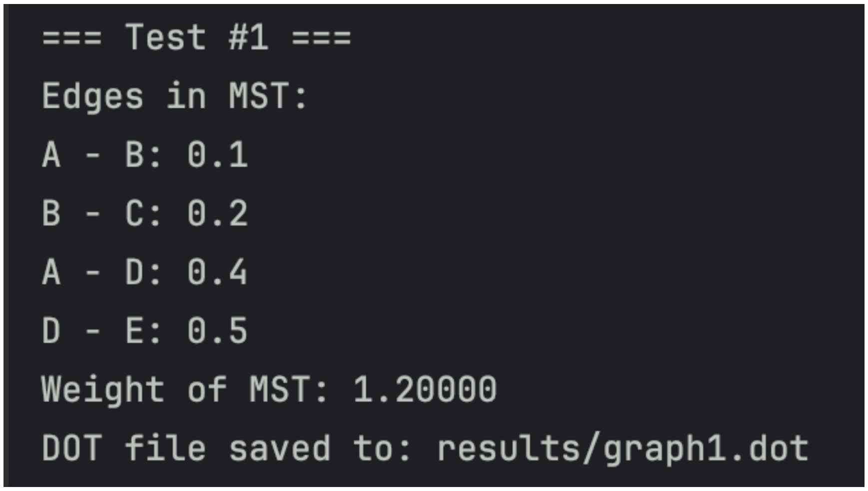
3)the edge weight represents the cost of constructing the road.

Data Summary and Algorithm Results

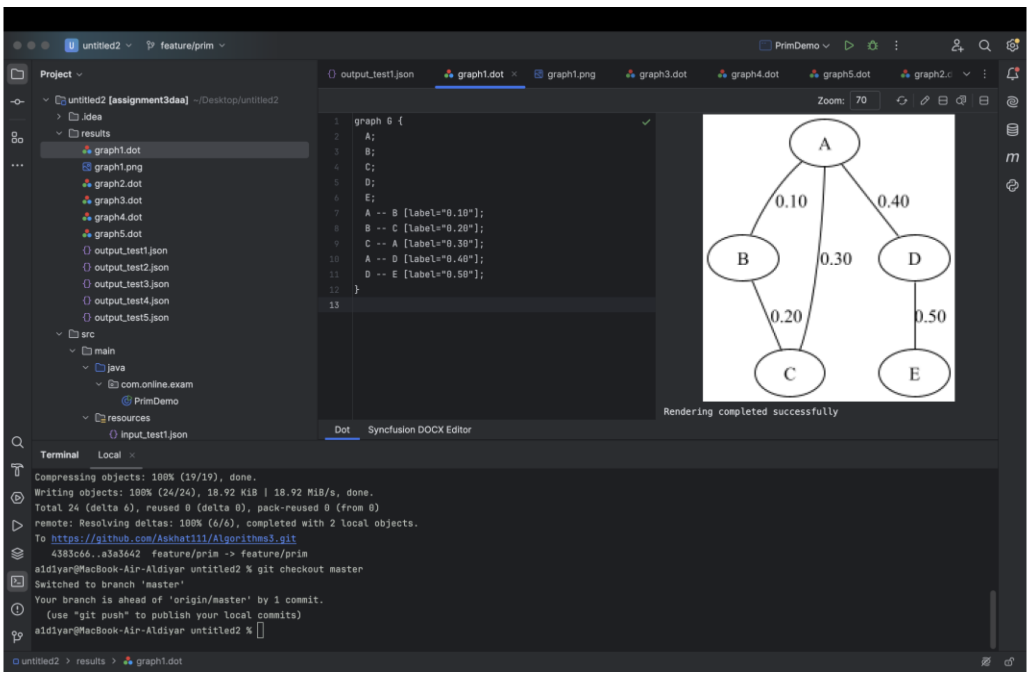
The input consists of five test graphs representing various city district layouts and road connection options. Each graph is specified as JSON files detailing nodes (districts) and weighted edges (potential roads with construction costs).

For each test graph, both Prim’s and Kruskal’s algorithms were implemented and run to find MSTs. Execution times, key operations (comparisons, unions), MST edges, and total costs were recorded.

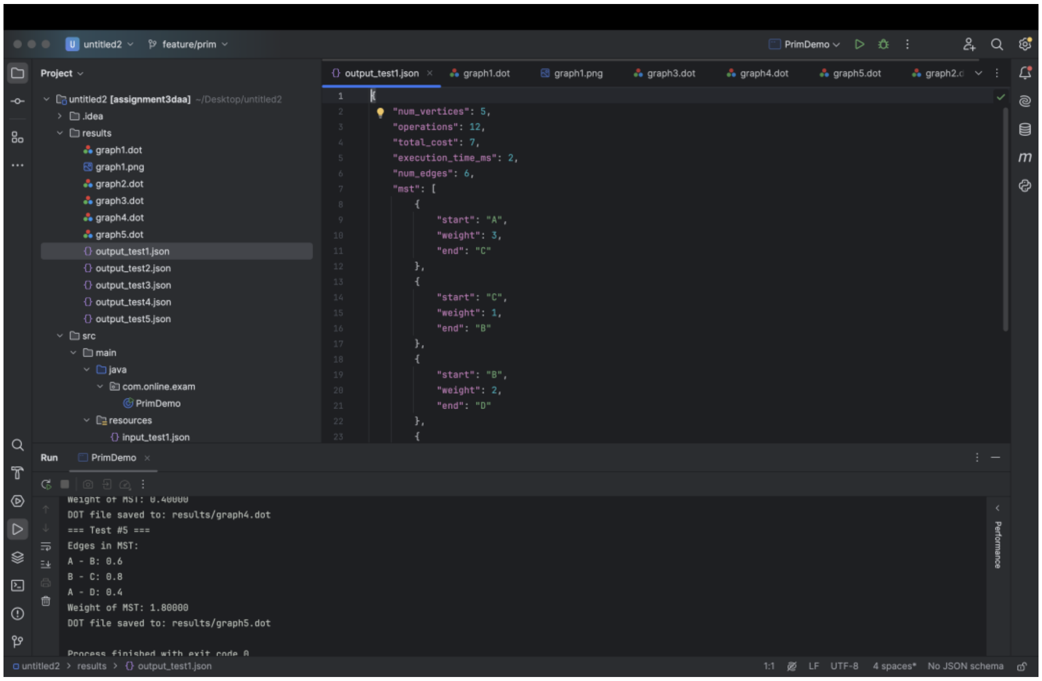
Prim:



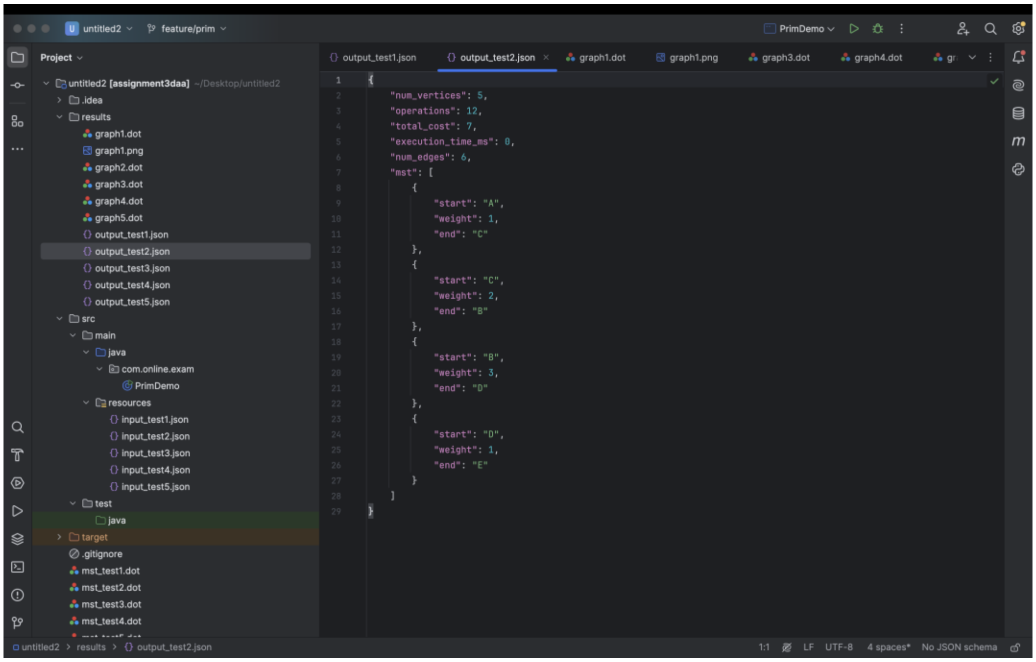
=== Test #1 ===
Edges in MST:
A - B: 0.1
B - C: 0.2
A - D: 0.4
D - E: 0.5
Weight of MST: 1.20000
DOT file saved to: results/graph1.dot

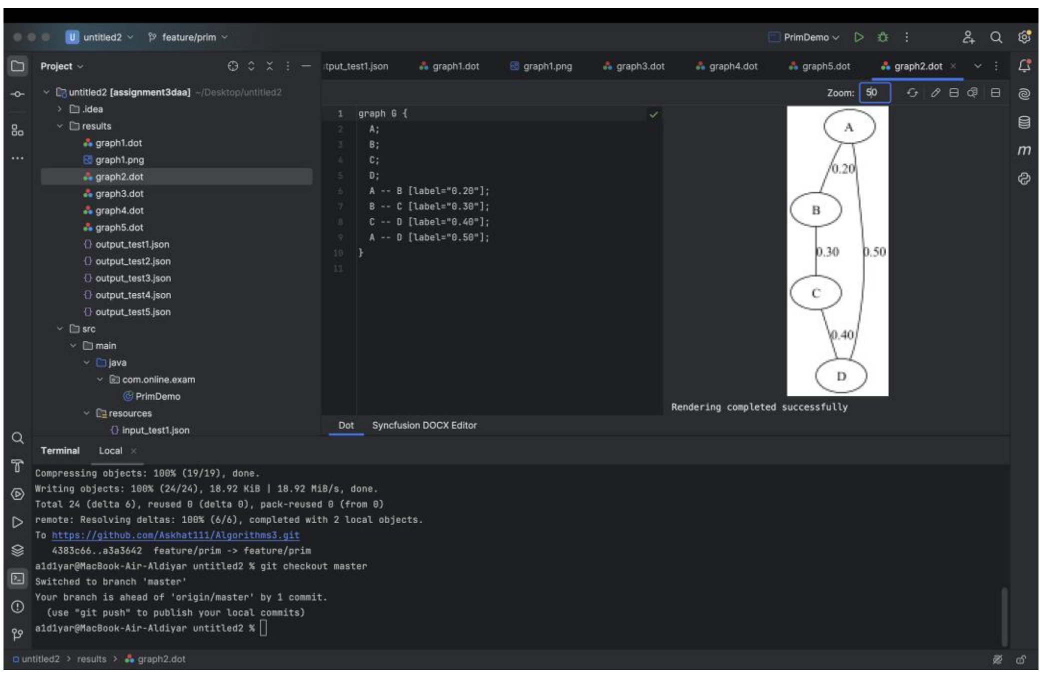


graph G {
A;
B;
C;
D;
E;
A -- B [label="0.10"];
B -- C [label="0.20"];
C -- A [label="0.30"];
A -- D [label="0.40"];
D -- E [label="0.50"];
}
A
B
C
D
E
0.10
0.20
0.30
0.40
0.50
Rendering completed successfully
Compressing objects: 100% (19/19), done.
Writing objects: 100% (24/24), 18.92 KiB | 18.92 MiB/s, done.
Total 24 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
To https://github.com/Askhat111/Algorithms3.git
4383c66..a3a3642 feature/prim -> feature/prim
a1diyar@MacBook-Air-Aldiyar untitled2 % git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
a1diyar@MacBook-Air-Aldiyar untitled2 %

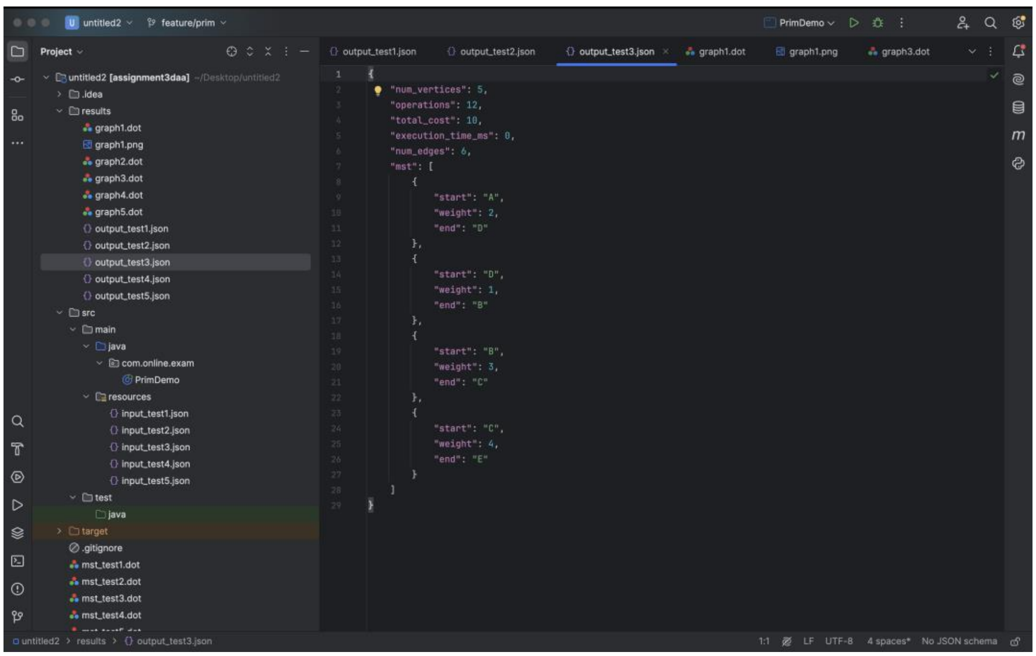


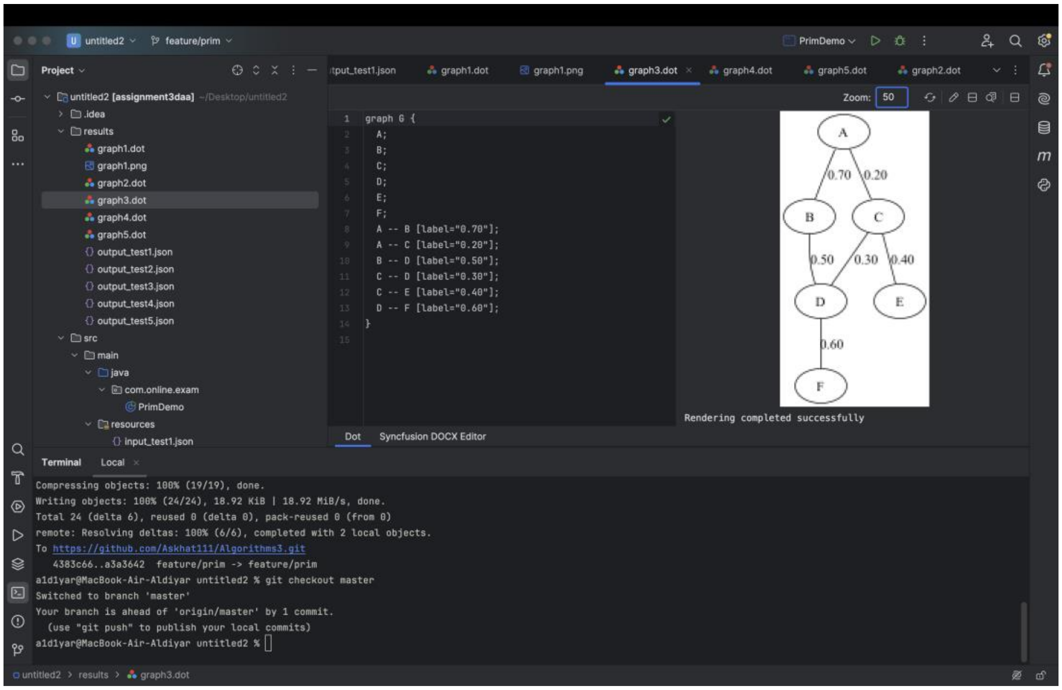

```
=== Test #2 ===
Edges in MST:
A - B: 0.2
B - C: 0.3
C - D: 0.4
Weight of MST: 0.90000
DOT file saved to: results/graph2.dot
```


untitled2
feature/prim
PrimDemo
Project
output_test1.json
output_test2.json
graph1.dot
graph1.png
graph3.dot
graph4.dot
untitled2 [assignment3daa] ~/Desktop/untitled2
.idea
results
graph1.dot
graph1.png
graph2.dot
graph3.dot
graph4.dot
graph5.dot
output_test1.json
output_test2.json
output_test3.json
output_test4.json
output_test5.json
src
main
java
com.online.exam
PrimDemo
resources
input_test1.json
input_test2.json
input_test3.json
input_test4.json
input_test5.json
test
java
target
.gitignore
mst_test1.dot
mst_test2.dot
mst_test3.dot
mst_test4.dot
{
"num_vertices": 5,
"operations": 12,
"total_cost": 7,
"execution_time_ms": 0,
"num_edges": 6,
"mst": [
{
"start": "A",
"weight": 1,
"end": "C"
},
{
"start": "C",
"weight": 2,
"end": "B"
},
{
"start": "B",
"weight": 3,
"end": "D"
},
{
"start": "D",
"weight": 1,
"end": "E"
}
]
}
untitled2 > results > output_test2.json
1:1 LF UTF-8 4 spaces* No JSON schema

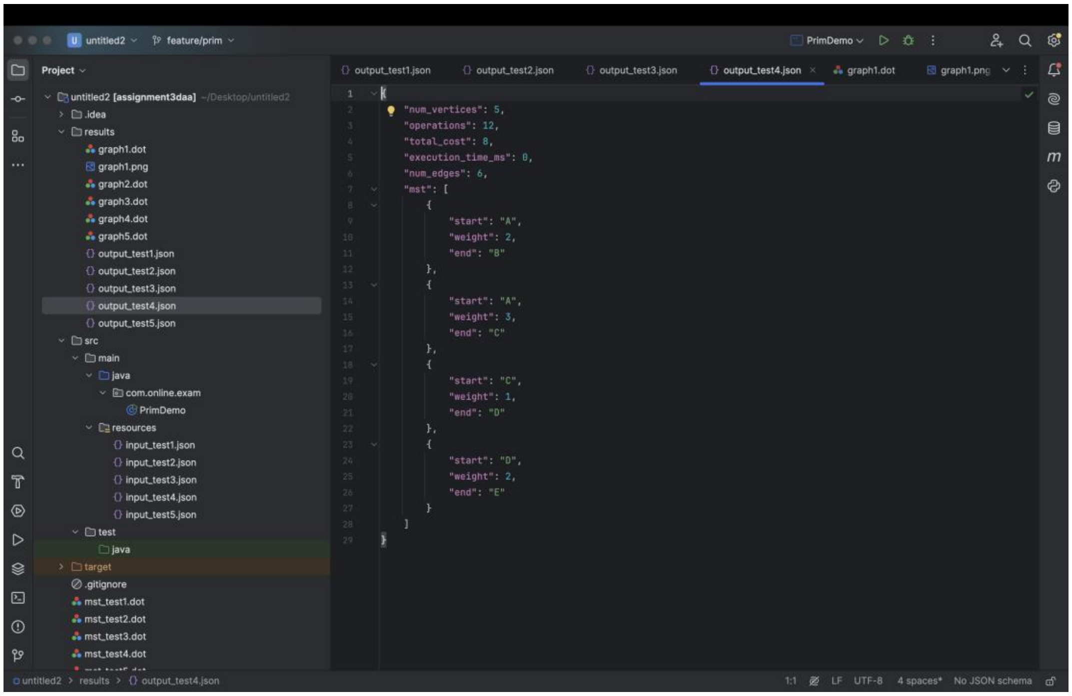

untitled2
feature/prim
PrimDemo
Project
tput_test1.json
graph1.dot
graph1.png
graph3.dot
graph4.dot
graph5.dot
graph2.dot
Zoom: 50
graph G {
A;
B;
C;
D;
A -- B [label="0.20"];
B -- C [label="0.30"];
C -- D [label="0.40"];
A -- D [label="0.50"];
}
A
B
C
D
0.20
0.30
0.50
0.40
Rendering completed successfully
Dot
Syncfusion DOCX Editor
Terminal
Local
Compressing objects: 100% (19/19), done.
Writing objects: 100% (24/24), 18.92 KiB | 18.92 MiB/s, done.
Total 24 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
To https://github.com/Askhat111/Algorithms3.git
4383c66..a3a3642 feature/prim -> feature/prim
aldiyar@MacBook-Air-Aldiyar untitled2 % git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
aldiyar@MacBook-Air-Aldiyar untitled2 %
untitled2 > results > graph2.dot

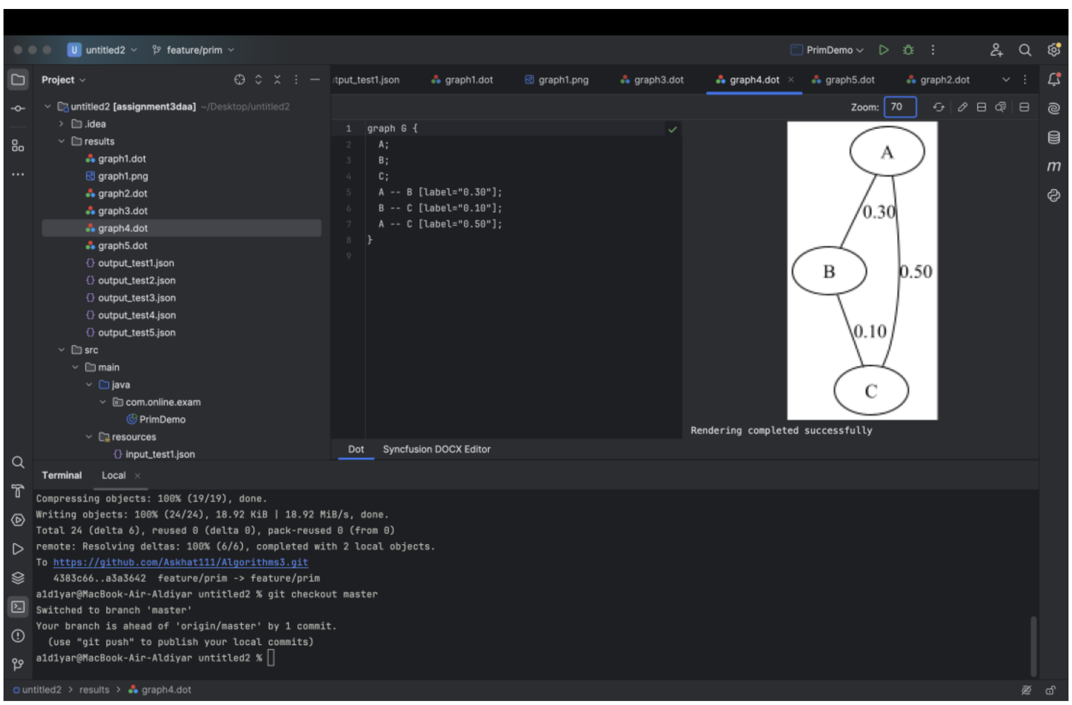
```
=== Test #3 ===
Edges in MST:
B - D: 0.5
A - C: 0.2
C - D: 0.3
C - E: 0.4
D - F: 0.6
Weight of MST: 2.00000
DOT file saved to: results/graph3.dot
```



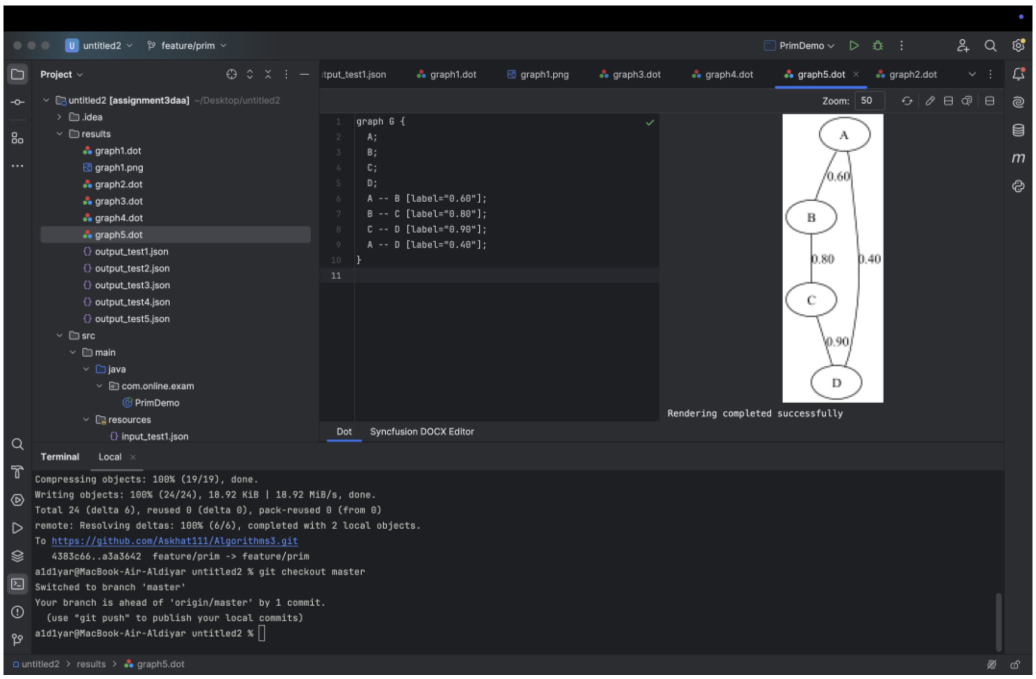


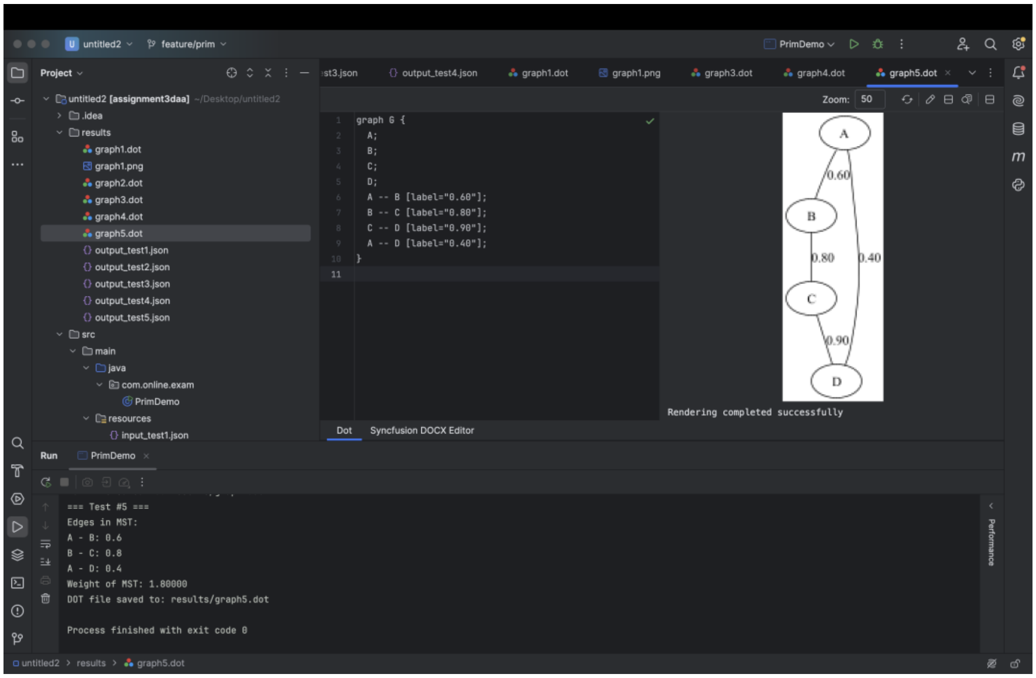

```
=== Test #4 ===
Edges in MST:
A - B: 0.3
B - C: 0.1
Weight of MST: 0.40000
DOT file saved to: results/graph4.dot
```


untitled2
feature/prim
PrimDemo
Project
output_test1.json
output_test2.json
output_test3.json
output_test4.json
graph1.dot
graph1.png
untitled2 [assignment3daa] ~/Desktop/untitled2
.idea
results
graph1.dot
graph1.png
graph2.dot
graph3.dot
graph4.dot
graph5.dot
output_test1.json
output_test2.json
output_test3.json
output_test4.json
output_test5.json
src
main
java
com.online.exam
PrimDemo
resources
input_test1.json
input_test2.json
input_test3.json
input_test4.json
input_test5.json
test
java
target
.gitignore
mst_test1.dot
mst_test2.dot
mst_test3.dot
mst_test4.dot
{
"num_vertices": 5,
"operations": 12,
"total_cost": 8,
"execution_time_ms": 0,
"num_edges": 6,
"mst": [
{
"start": "A",
"weight": 2,
"end": "B"
},
{
"start": "A",
"weight": 3,
"end": "C"
},
{
"start": "C",
"weight": 1,
"end": "D"
},
{
"start": "D",
"weight": 2,
"end": "E"
}
]
}
untitled2 > results > output_test4.json
1:1 LF UTF-8 4 spaces* No JSON schema


untitled2
feature/prim
PrimDemo
Project
tput_test1.json
graph1.dot
graph1.png
graph3.dot
graph4.dot
graph5.dot
graph2.dot
Zoom: 70
graph G {
A;
B;
C;
A -- B [label="0.30"];
B -- C [label="0.10"];
A -- C [label="0.50"];
}
A
B
C
0.30
0.50
0.10
Rendering completed successfully
Dot
Syncfusion DOCX Editor
Terminal
Local
Compressing objects: 100% (19/19), done.
Writing objects: 100% (24/24), 18.92 KiB | 18.92 MiB/s, done.
Total 24 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
To https://github.com/Askhat111/Algorithms3.git
4383c66..a3a3642 feature/prim -> feature/prim
a1diyar@MacBook-Air-Aldiyar untitled2 % git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
a1diyar@MacBook-Air-Aldiyar untitled2 %
untitled2 > results > graph4.dot

```
=== Test #5 ===
Edges in MST:
A - B: 0.6
B - C: 0.8
A - D: 0.4
Weight of MST: 1.80000
DOT file saved to: results/graph5.dot
```

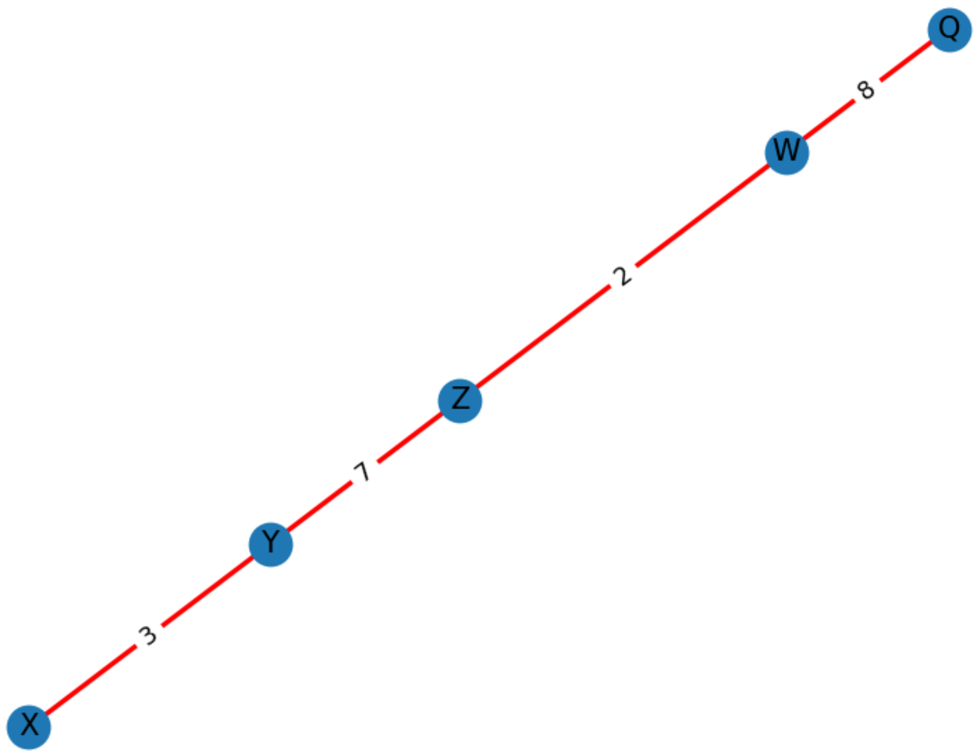




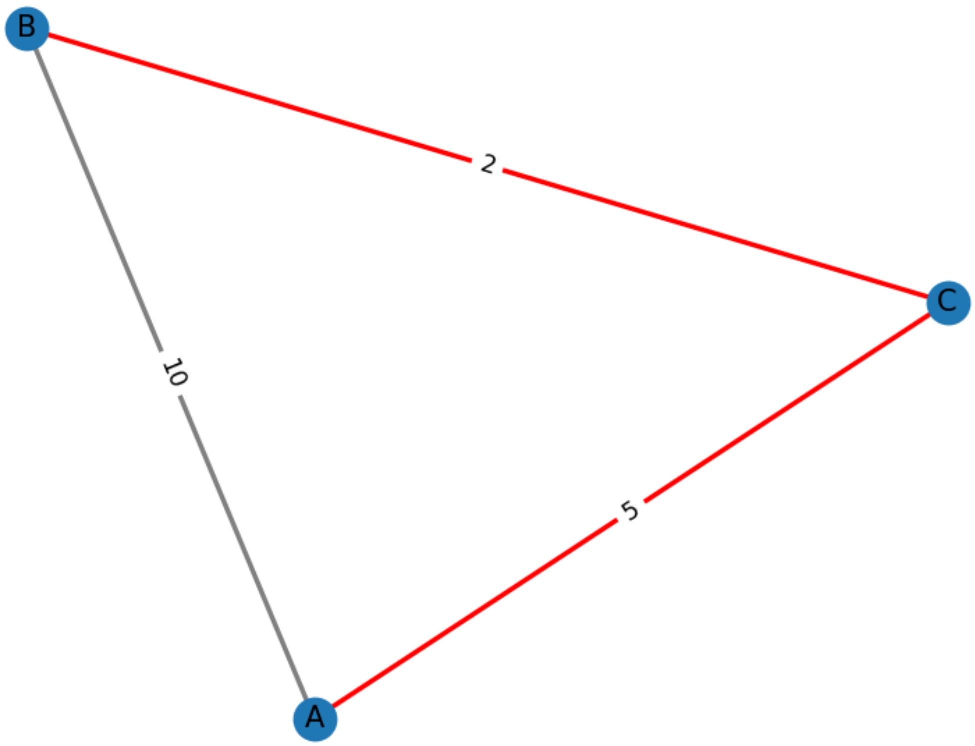
Kruksal:

```
=== Test #1 ===
MST Edges:
2-3 2,00
0-1 3,00
1-2 7,00
3-4 8,00
Total MST cost: 20,00
Vertices: 5
Edges: 4
```

```
Comparisons: 4
Union operations: 3
Find operations (approx): 8
Execution time: 0 ms
```

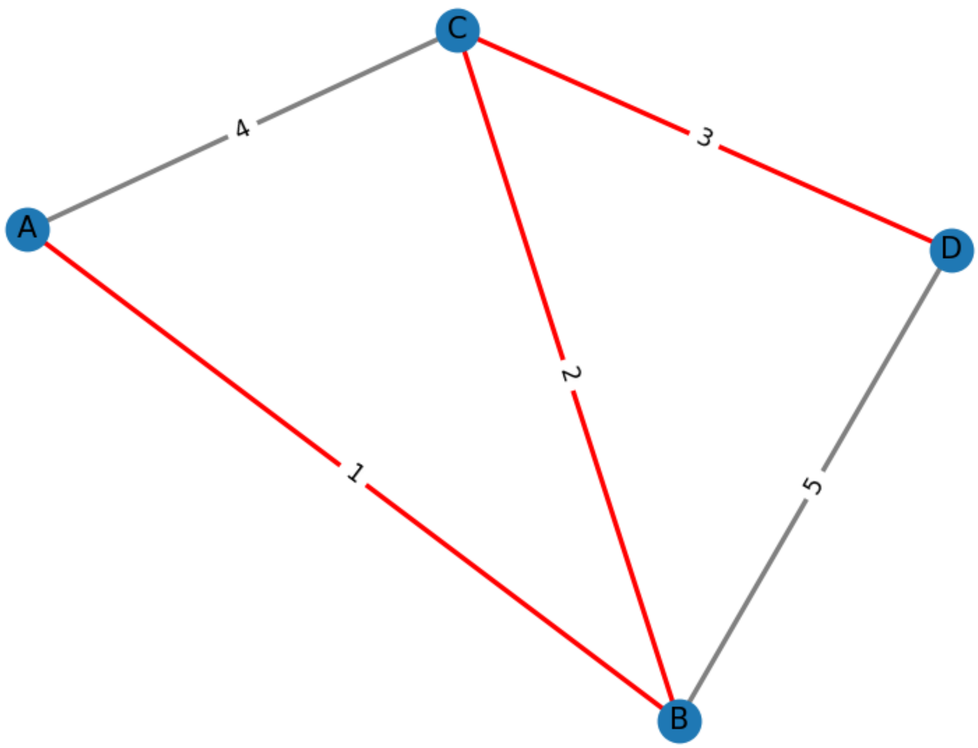



```
=== Test #2 ===
MST Edges:
1-2 2,00
0-2 5,00
Total MST cost: 7,00
Vertices: 3
Edges: 3
Comparisons: 2
Union operations: 1
Find operations (approx): 4
```



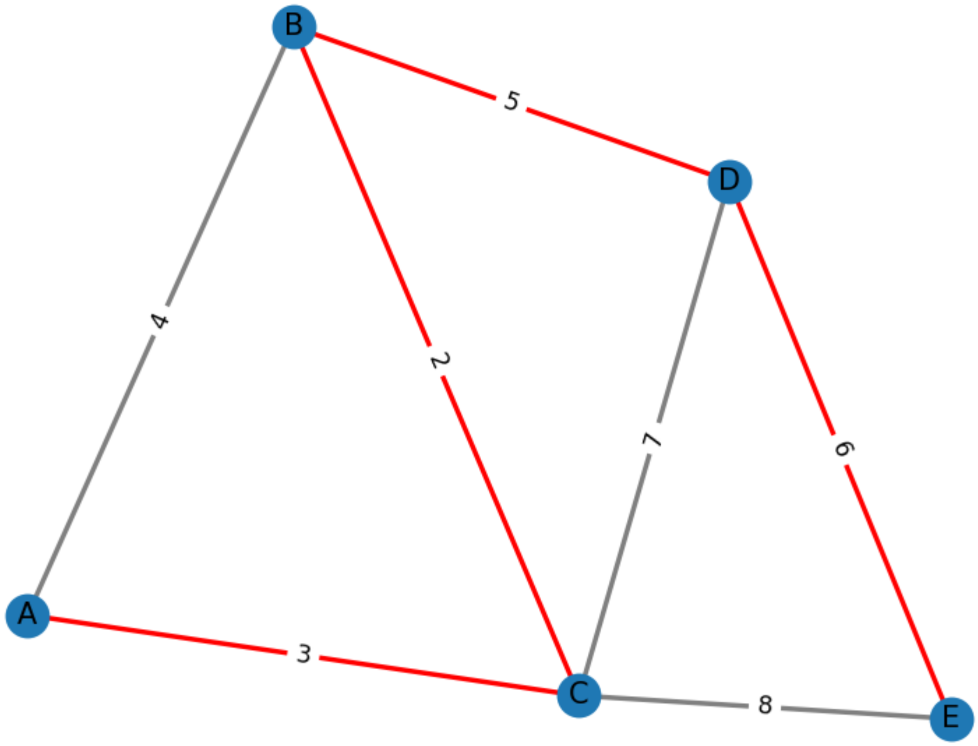

```
=== Test #3 ===
MST Edges:
0-1 1,00
1-2 2,00
2-3 3,00
Total MST cost: 6,00
Vertices: 4
Edges: 5
Comparisons: 3
Union operations: 2
```

```
Find operations (approx): 6
Execution time: 0 ms
```



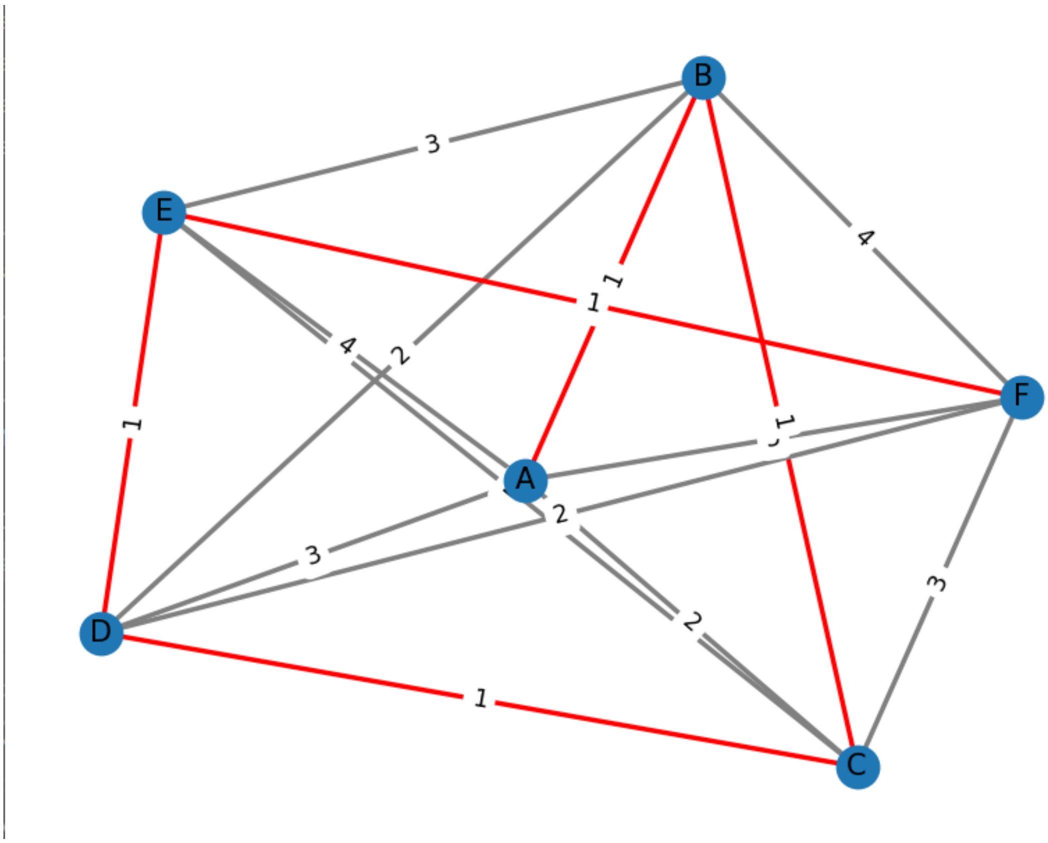

```
=== Test #4 ===
MST Edges:
1-2 2,00
0-2 3,00
1-3 5,00
3-4 6,00
Total MST cost: 16,00
Vertices: 5
Edges: 7
```

```
Comparisons: 5
Union operations: 3
Find operations (approx): 8
Execution time: 0 ms
```




```
=== Test #5 ===
MST Edges:
0-1 1,00
1-2 1,00
2-3 1,00
3-4 1,00
4-5 1,00
Total MST cost: 5,00
Vertices: 6
Edges: 15
```

```
Comparisons: 5
Union operations: 4
Find operations (approx): 10
Execution time: 0 ms
```



Test	Vertices	Edges	Kruskal's Ops (Comparisons)	Union Ops	Find Ops (approx)	Kruskal's MST Cost	Kruskal's Time (ms)
1	5	4	4	3	8	20.0	0
2	3	3	2	2	4	7.0	0
3	4	5	3	3	6	6.0	0
4	5	7	5	4	8	16.0	0
5	6	15	5	5	10	5.0	0

Comparison of Prim's and Kruskal's Algorithms

Both algorithms produced MSTs with identical total costs for all test cases, confirming correctness. Differences were observed in operational counts and runtime, depending largely on graph structure:

1)Kruskal’s algorithm is generally more efficient for sparse graphs, relying on sorted edge lists and union-find data structures.

2)Prim’s algorithm performs well on dense graphs with adjacency matrix or list representations, updating minimum edges efficiently.

Implementation complexity is somewhat higher for Kruskal’s due to union-find, but conceptually simpler with an edge-centric approach.

Conclusions

Both Prim’s and Kruskal’s algorithms are effective for MST problems in city transportation network optimization.Kruskal’s is effective for sparse graphs or when edges are naturally listed, Prim’s for dense graphs or adjacency representations.

Performance is competitive, algorithm choice may depend on graph input formats and specific application constraints.

Source:

https://www.geeksforgeeks.org/dsa/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

GitHub Link

The code on Github :https://github.com/Askhat111/Algorithms3