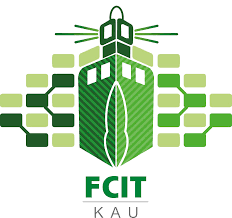
****

**KING ABDUL AZIZ UNIVERSITY**

**COLLEGE OF COMPUTING AND INFORMATION**

**TECHNOLOGY DEPARTMENT OF INFORMATION SYSTEM**

**CPCS 324: Algori6thms and Data Structures (II)**

**Spring 2021**

**Group Project: Phase 2**

**Group 2 students:**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Section** |
| **Safia Aljahdali** | **1807806** | **DAR** |
| **Shahad Shamsan (Leader)** | **1808718** | **DAR** |
| **Suha Shafi** | **1806004** | **DAR** |

**Instructor: Dr. Bassma Alsulami**

Table of Contents

[Introduction 3](#_Toc68298833)

[Screenshots of the outputs 3](#_Toc68298834)

[Minimum Spanning Tree Algorithms output 3](#_Toc68298835)

[Dijkstra output 6](#_Toc68298836)

[Comparisons of the running times 13](#_Toc68298837)

[Kruskal’s algorithm and Priority queue-based Prim’s algorithm 13](#_Toc68298838)

[Priority-queue based Prim’s algorithm and Minheap based Prim’s algorithm 13](#_Toc68298839)

[Code implementation 14](#_Toc68298840)

[Difficulties faced during the phase design 14](#_Toc68298841)

[Conclusion 15](#_Toc68298842)

[References 15](#_Toc68298843)

# Introduction

A spanning tree of a connected undirected graph is the subgraph that is a tree and connects all the vertices together, many spanning trees can be found in one graph, and a minimum spanning tree (MST), is a spanning tree with the minimum possible total weight without containing any cycles, in a connected, weighted, undirected graph.

There are many algorithms to find the minimum spanning tree of a graph, such as Prim’s and Kruskal algorithms, both are greedy algorithms, but the Prim’s algorithm operates by building the tree one vertex at a time from a random starting vertex, adding the cheapest possible connection from the tree to another vertex, this algorithm can be implemented using a minheap or a priority queue, when the latter adds the next lowest weighted edge that will not form a cycle in each step to the minimum spanning forest.

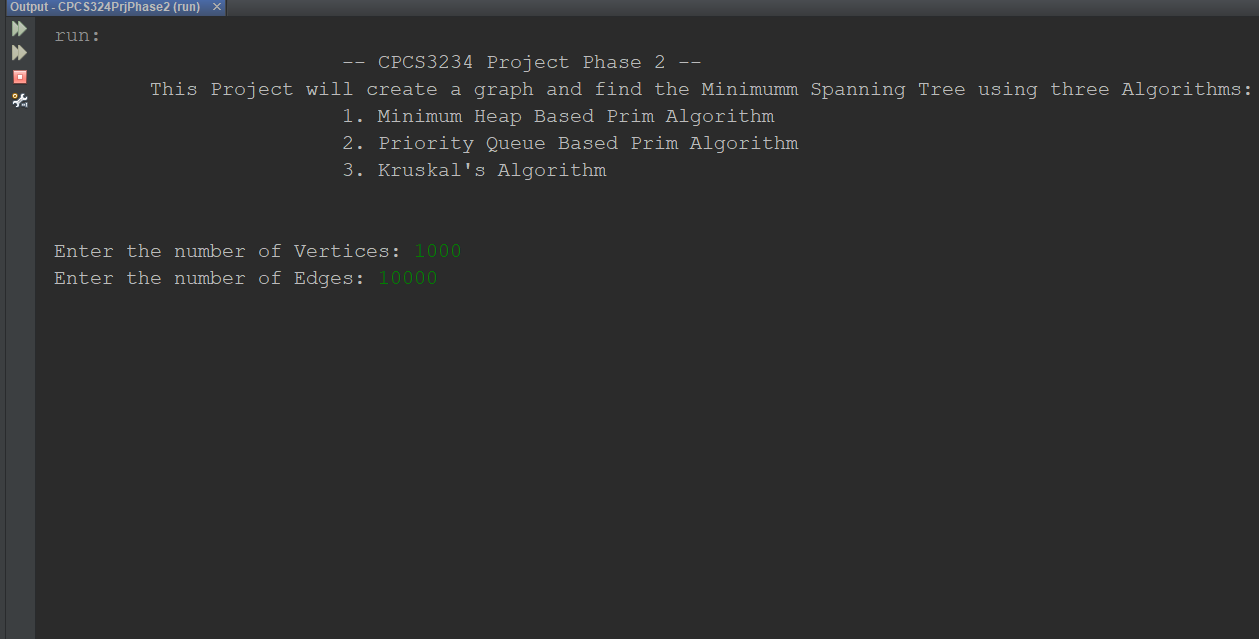
Our goal in this phase is to perform and experimental comparison of the running time of the algorithms, one between the Prim’s priority queue implementation and minheap implementation, and the other between Kruskal algorithm and priority-queue based Prim.

# Screenshots of the outputs

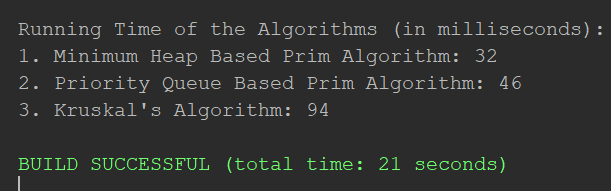
## Minimum Spanning Tree Algorithms output

Kruskal and Minheap and Priority-queue based Prim algorithms were each run 10 time as follows:

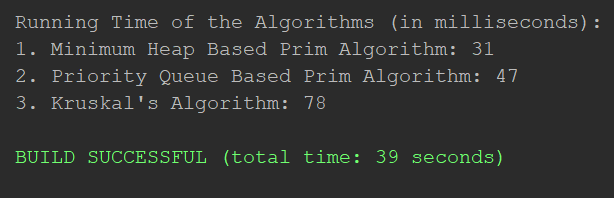
First, the program asks user to enter number of vertices and edges to perform each algorithm



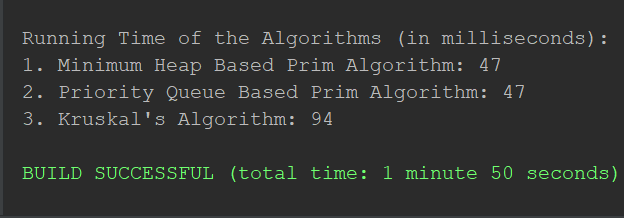
1. **n=1000, m=10000**



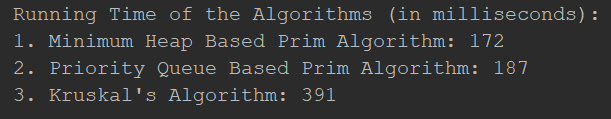
1. **n=1000, m=15000**



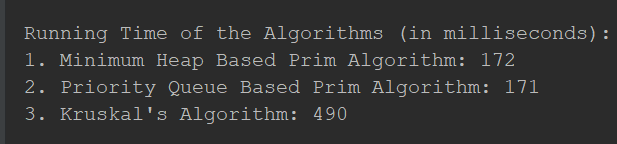
1. **n=1000, m=25000**



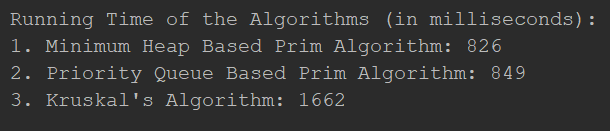
1. **n=5000, m=15000**



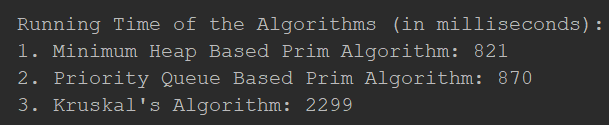
1. **n=5000, m=25000**



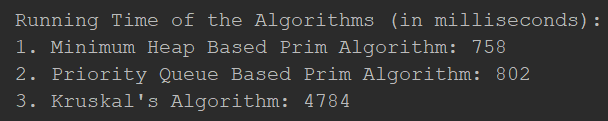
1. **n=10000, m=15000**



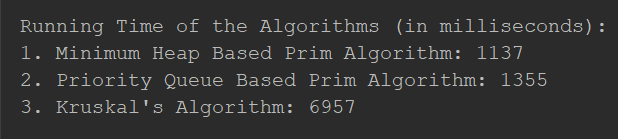
1. **n=10000, m=25000**



1. **n=20000, m=200000**

****

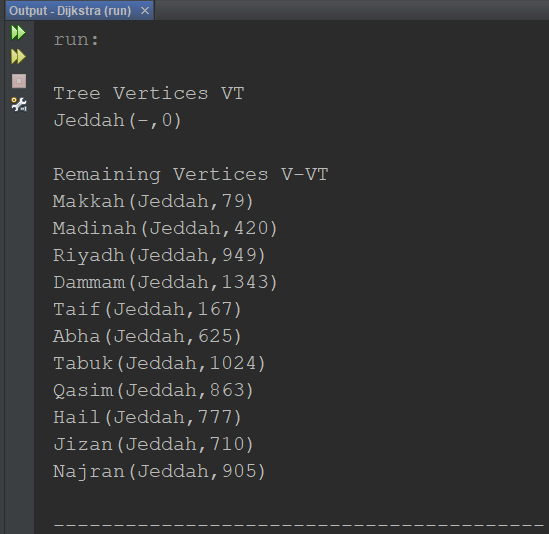
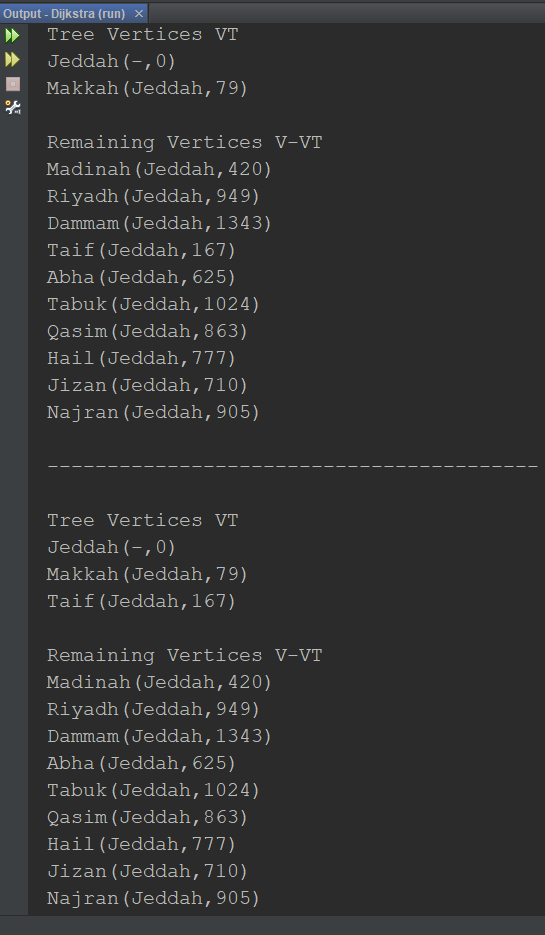
1. **n=20000, m=300000**

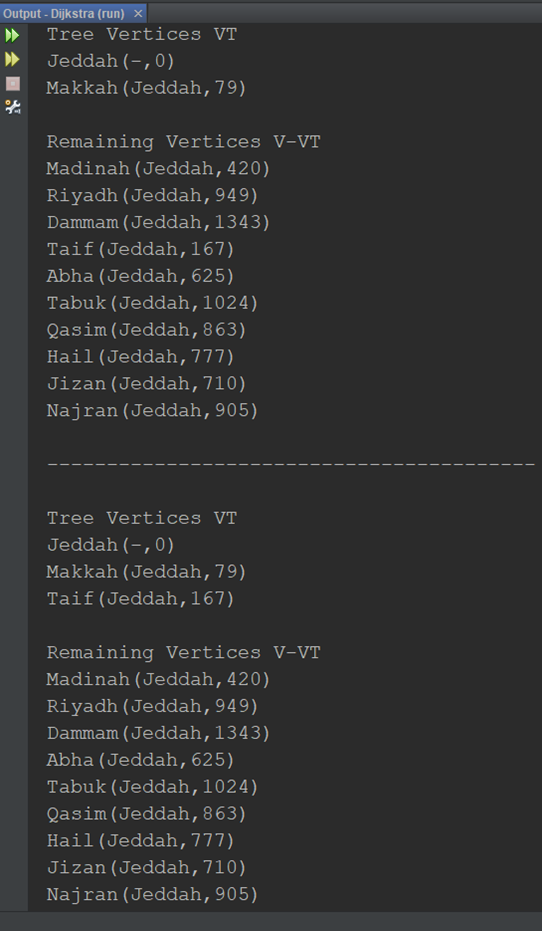
****

1. **n=50000, m=1000000**

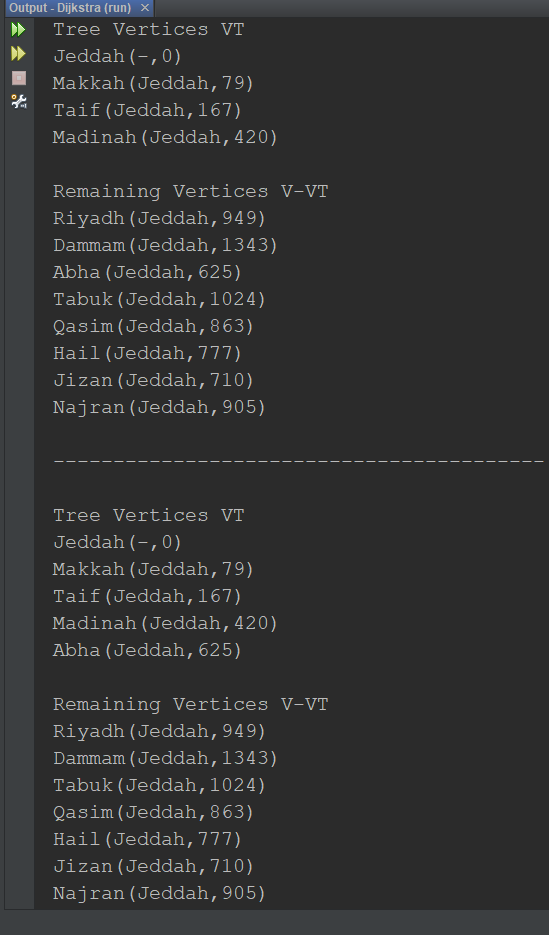
The instance is too big that the program took more than 3 hours and still no result.

## Dijkstra output

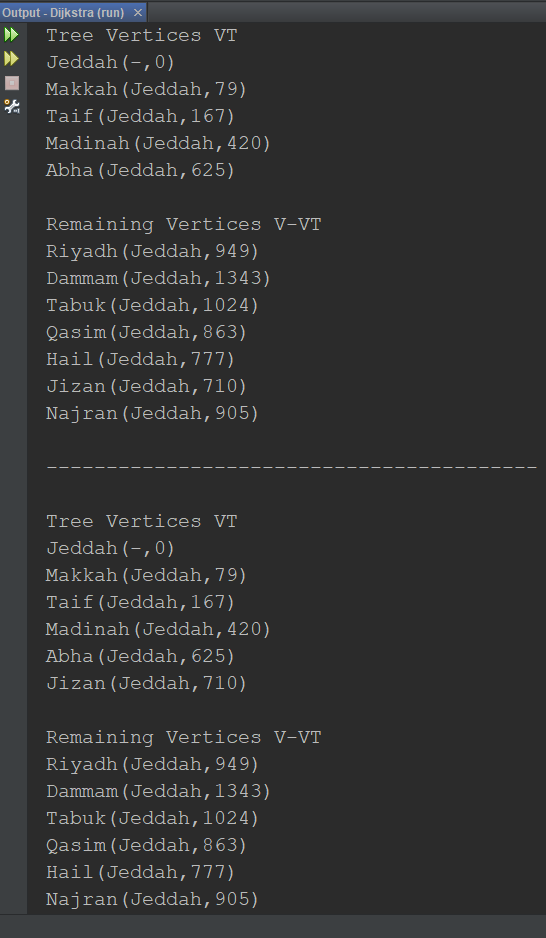
1. Starting with the given source Jeddah, it is added to the Tree verities list and all other vertices are on the remaining vertices list
2. Makkah is added to the Tree verities list since it has the lowest edge weight, then Taif and all other vertices are in the remaining list
3. Taif is added to the Tree verities list since it has the lowest edge weight, and all other vertices are in the remaining list.



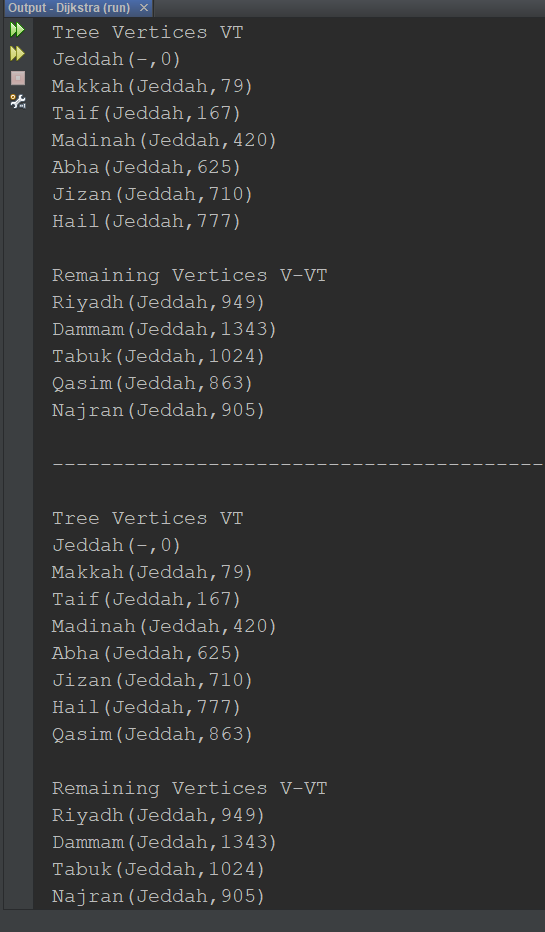
1. Madinah is added to the Tree verities list since it has the lowest edge weight, and all other vertices are in the remaining list.



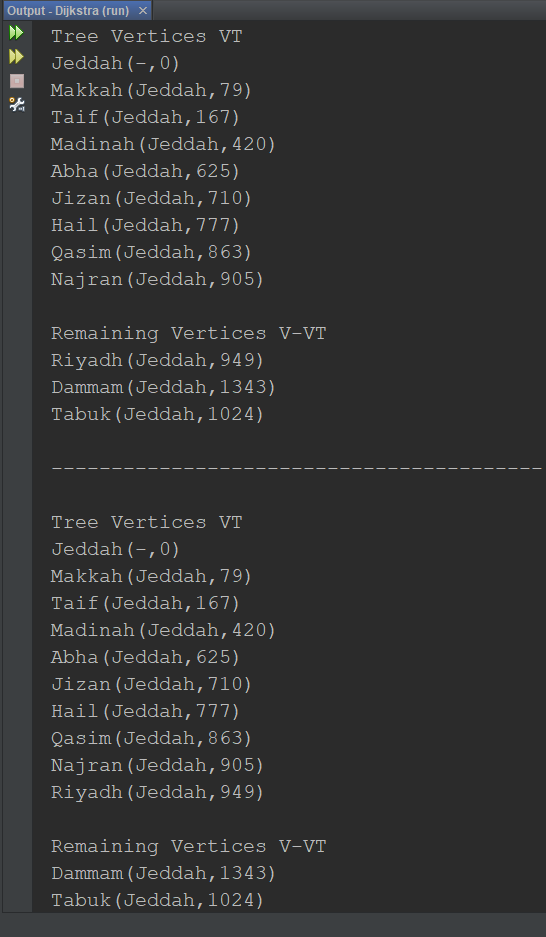
1. Abha is added to the Tree verities list since it has the lowest edge weight, then Jizan, and all other vertices are in the remaining list.



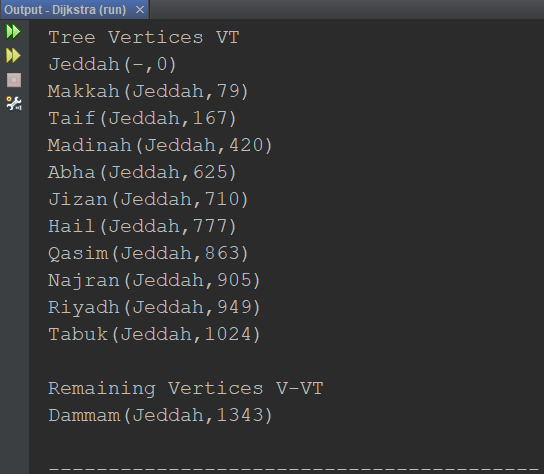
1. Hail is added to the Tree verities list since it has the lowest edge weight, then Qasim, and all other vertices are in the remaining list.



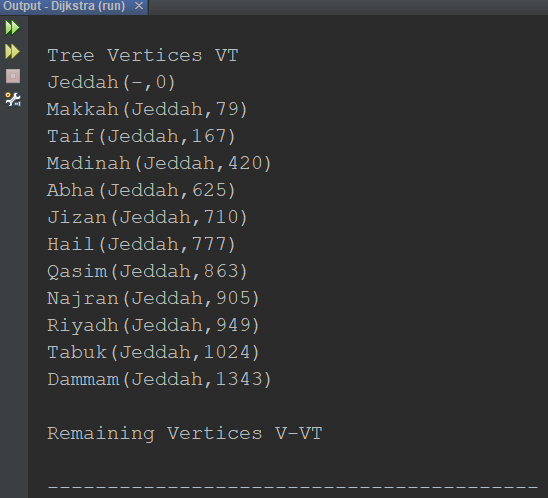
1. Najran is added to the Tree verities list since it has the lowest edge weight, then Riyadh, and all other vertices are in the remaining list.



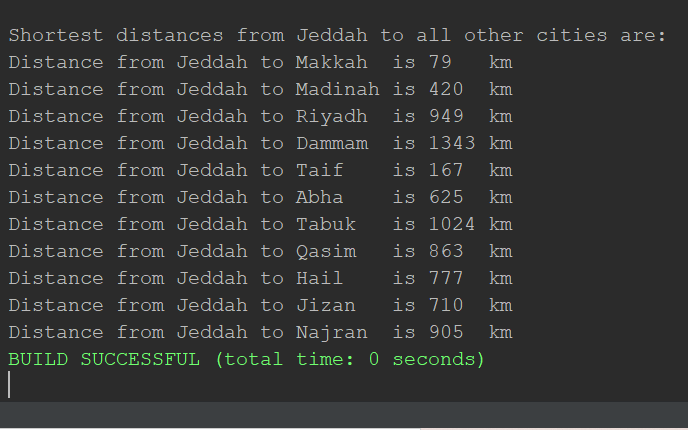
1. Tabuk is added to the Tree verities list since it has the lowest edge weight, and only one other vertex in the remaining list.



1. Dammam is added to the Tree verities list since it has the lowest edge weight, hence the remaining vertices list becomes empty



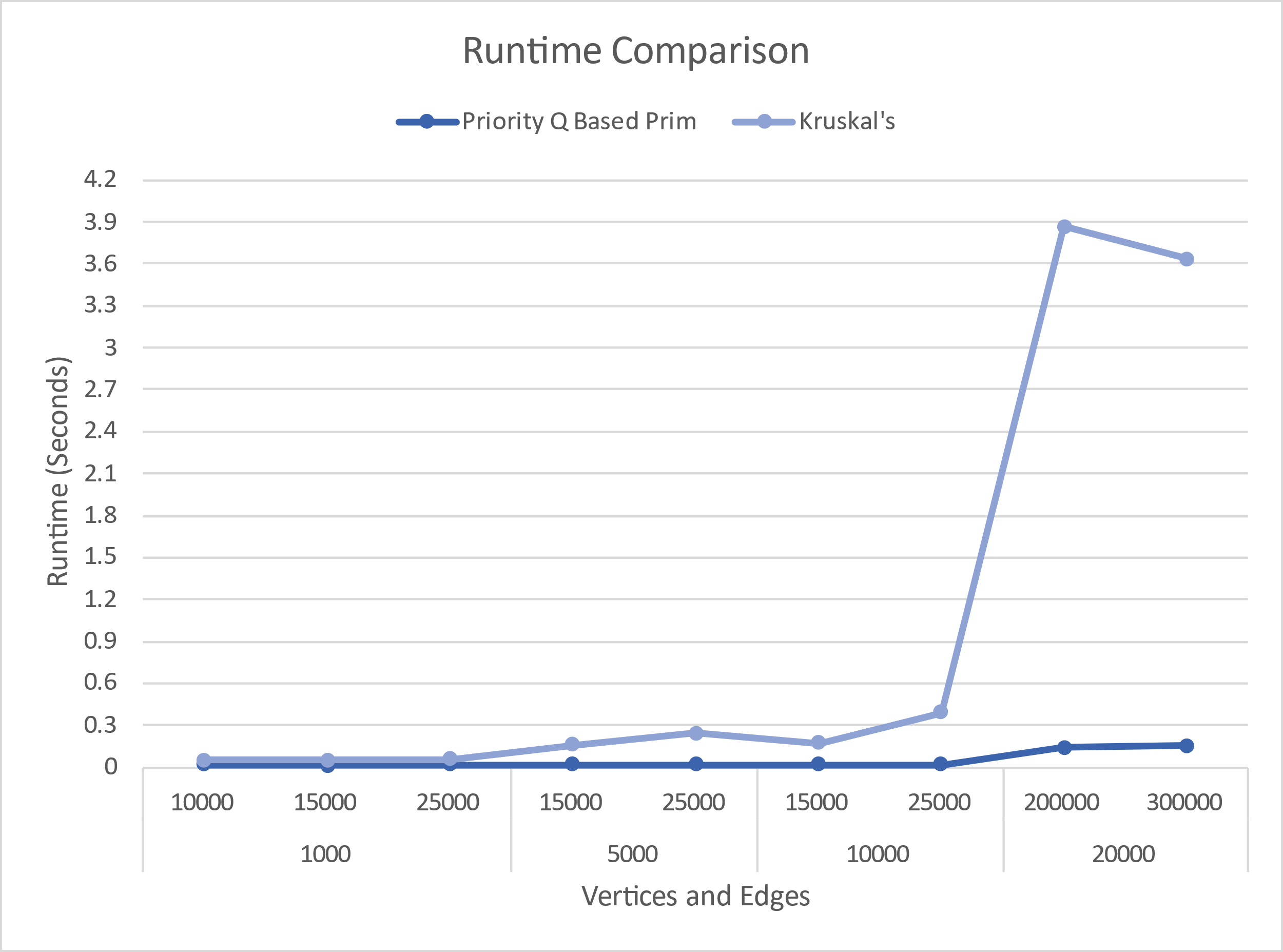
1. Finally, print the shortest distance from Jeddah (source) to all other cities in km



# Comparisons of the running times

## Kruskal’s algorithm and Priority queue-based Prim’s algorithm

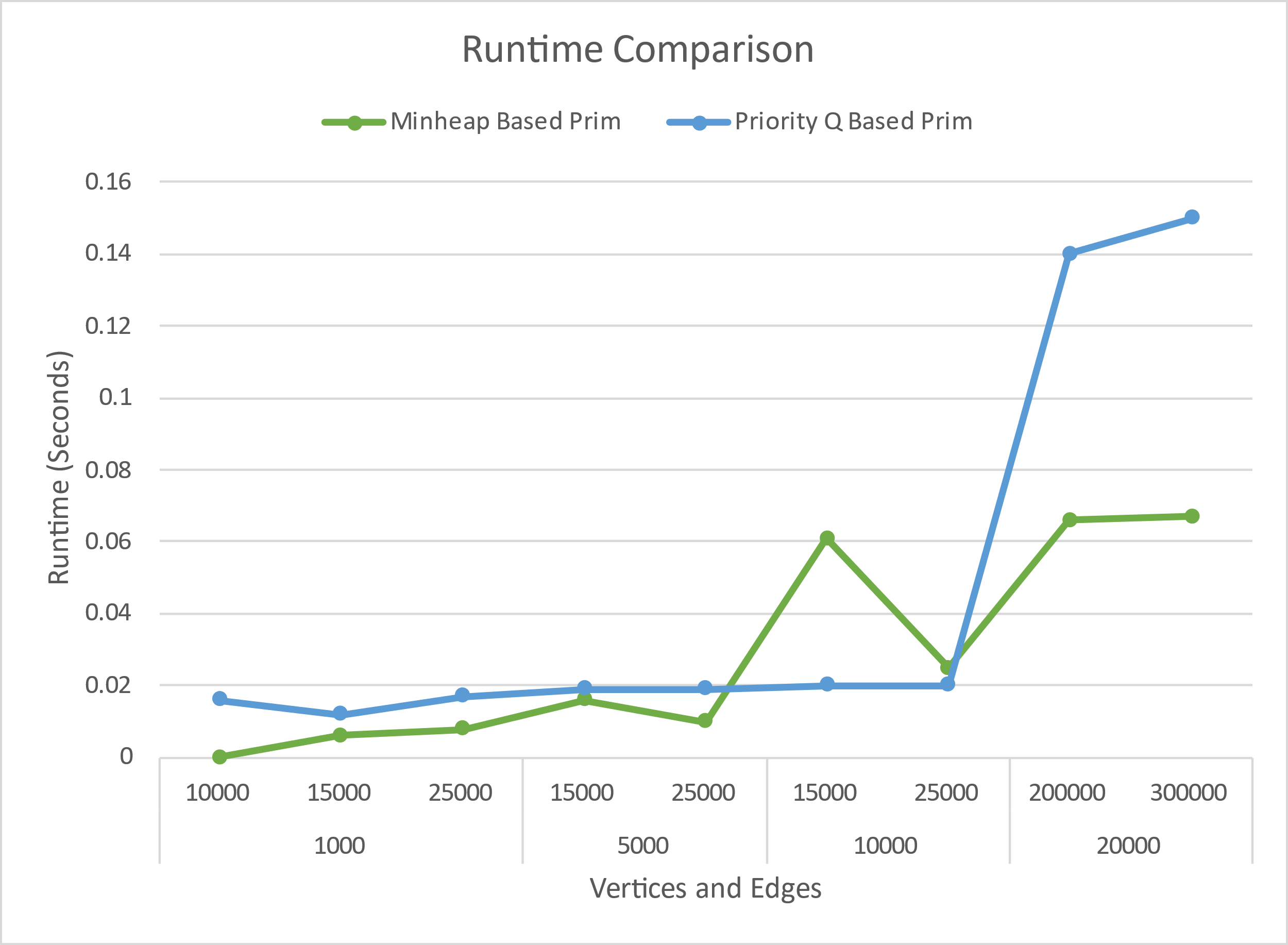
On the first test run with vertices n, and edges m, equals to n= 1000 and m= {10000, 15000, 25000} the running times for the two algorithms are close, then on the second test run with n= 5000 and m={15000, 25000}, the Kruskal’s algorithm takes more time than the priority queue-based Prim’s algorithm but the difference is not that big, after that the test run increased to n= 10000 and m={15000, 25000} in which the Kruskal’s algorithm’s running time is also slightly more than Prim’s, then, n=20000 and m={200000, 300000} and a big change in the Kruskal’s running time occurs, while the Prim’s only changes a little. Lastly, n=50000 and m=1000000 the running time for both algorithms took longer than 3 hours so we could not track it. Overall, the running time increases with increasing instances, but Prim’s running time stays stable throughout the experiment while Kruskal’s noticeably rises.



1: Runtime of Kruskal and pq based Prim

## Priority-queue based Prim’s algorithm and Minheap based Prim’s algorithm

Starting with n= 1000 and m= {10000, 15000, 25000} the running times for the two implementation are close with the Priority-queue based Prim taking longer, the same thing happened on the second test run with n= 5000 and m={15000, 25000}, then when n= 10000 and m={15000, 25000} the minheap Prim’s running time had a sudden rise, while the Priority-queue based Prim stayed consistent, after that, n=20000 and m={200000, 300000} the Priority-queue based Prim increased a lot and Minheap increased gradually. Lastly, n=50000 and m=1000000 the running time for both algorithms also took longer than 3 hours so we could not track it as well. Overall, the running time increases with increasing instances, but Minheap Prim’s running time is steadier than Priority-queue based Prim which takes a lot longer time on bigger instances.



2: Runtime of minheap and pq based Prim

# Code implementation

All minimum spanning tree algorithms were referenced from websites individually, and then gathered on the same project by us, using the make-graph method that constructs all graphs randomly when given the number of vertices and edges, which then were used to perform all the three algorithms.

Dijkstra’s algorithm was also referenced from a website then we edited the matrix to the 12-city dataset (Saudi Cities Distances) as required and making Jeddah the source as instructed.

# Difficulties faced during the phase design

Each algorithms implementation had its own difficulties while designing them, some algorithms take a lot of time with larger instances thus when it had to be modified and run again the process was time consuming.

# Conclusion

To summarize the results, the Prim’s algorithm was found to be more efficient than Kruskal’s algorithm, and the Minheap implementation of Prim’s algorithm is more efficient than Priority-queue based Prim implementation.

The time complexity for Kruskal’s algorithm is O (E log E), and for both Minheap implementation and Priority-queue based Prim’s algorithm is O (E log V).

# References

1. <https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/>
2. Cormen, Thomas; Charles E Leiesron, Ronald L Rivest, Clifford Stein (2009). Introduction to Algorithms (Third ed.). MIT Press.
3. <https://en.wikipedia.org/wiki/Minimum_spanning_tree#cite_ref-1>
4. Kruska’ls algorithm  
   <https://algorithms.tutorialhorizon.com/kruskals-algorithm-minimum-spanning-tree-mst-complete-java-implementation/>
5. Prim’s algorithm (priority queue-based)  
   <https://algorithms.tutorialhorizon.com/prims-minimum-spanning-tree-mst-using-adjacency-list-and-priority-queue-without-decrease-key-in-oelogv/>
6. Prim’s algorithm (minheap-based)  
   <https://algorithms.tutorialhorizon.com/prims-minimum-spanning-tree-mst-using-adjacency-list-and-min-heap/>
7. Dijkstra’s algorithm  
   <https://www.programiz.com/dsa/dijkstra-algorithm>