King Abdulaziz University
Faculty of Computing and Information Technology
Computer Science Department
Spring 2023
CPCS324 - Algorithms and Data Structures (II)
Group Project – Part II

Dijkstra Algorithm Empirical Analysis

Group#37

Team Members

Student Name	Student ID	Section
Rima Ali Alghamdi	2005617	B8
Layla Zuhair Alsulaimani	2005800	B0A
Zinab Adnan Alsaggaf	2006531	B8
Hadeel Hassan Althbiti	1906441	B8

Course Instructor:
Dr. Basmah Alsulami
Dr. Sidra Qurashi

Table of Contents

Table of Figures

Table of Tables

1. Introductions

Greedy Algorithms are known to be efficient in most problems, and the reason for that is that they always make optimal decisions which are called “Greedy Choice Property”, and the second reason is “Optimal Structure” which contains solutions to sub-problems of an optimal solution, one of these greedy algorithms is Dijkstra's algorithm.

Although the efficiency of Dijkstra's Algorithm is mathematically proved, theoretical results and experimental results are usually different, since theoretical results estimate the experimental outcomes, often leaving a huge gap between the two.

In this report, Empirical Analysis is used to compare the theoretical efficiency and experimental physical time of Dijkstra's greedy algorithm which is used to compute the length of the shortest paths from the source location to every other location, and in our implementation every location will get the chance to be the source location.

1.1. Purpose

The goal of this study is to use empirical analysis to compare the theoretical time Efficiency of Dijkstra to its Physical Time.

1.2 Study Design

1.2.1 Efficiency Matric

In algorithms analysis there are two ways to measurements:

1. Physical time.
2. Count of basic operation.

Both measurements are valid, but the analysis in this project we are using physical time as the efficiency metric, and while physical time is not accurate, since it is dependent on the CPU (Central Processing Unit), our goal of this experiment is to compare the two algorithms, independently of the CPU.

1.2.2 Characteristics of The Input Sample

The algorithm will be put to the test using graphs that were constructed at random, where n stands for the number of nodes and m for the number of edges:

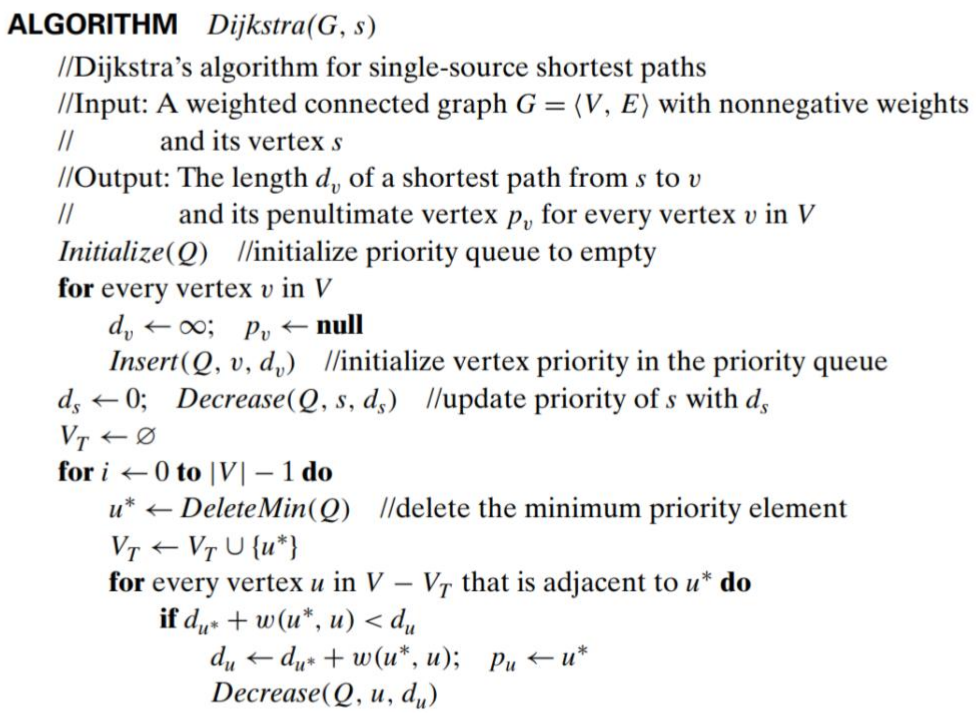
n = 2000 m = 10000
n = 3000 m = 15000
n = 4000 m = 20000
n = 5000 m = 25000
n = 6000 m = 3000

1.3 Analysis tools

- Visual Studio & NetBeans, used to implement and run the algorithms in java code.
- Microsoft Excel, used to create analysis tables and graphs that represent the collected data.
- Git & GitHub, to store the work.

2 The Pseudocode of Dijkstra

The Dijkstra Algorithm is Obtained from textbook (1).



```
ALGORITHM  Dijkstra(G, s)
    //Dijkstra's algorithm for single-source shortest paths
    //Input: A weighted connected graph $G = \langle V, E \rangle$ with nonnegative weights
    //         and its vertex $s$
    //Output: The length $d_v$ of a shortest path from $s$ to $v$
    //           and its penultimate vertex $p_v$ for every vertex $v$ in $V$
    Initialize(Q)   //initialize priority queue to empty
    for every vertex $v$ in $V$
        $d_v \leftarrow \infty$;   $p_v \leftarrow$ null
        Insert($Q, v, d_v$)   //initialize vertex priority in the priority queue
    $d_s \leftarrow 0$;   Decrease($Q, s, d_s$)   //update priority of $s$ with $d_s$
    $V_T \leftarrow \varnothing$
    for $i \leftarrow 0$ to $|V| - 1$ do
        $u^* \leftarrow$ DeleteMin($Q$)   //delete the minimum priority element
        $V_T \leftarrow V_T \cup \{u^*\}$
        for every vertex $u$ in $V - V_T$ that is adjacent to $u^*$ do
            if $d_{u^*} + w(u^*, u) < d_u$
                $d_u \leftarrow d_{u^*} + w(u^*, u)$;   $p_u \leftarrow u^*$
                Decrease($Q, u, d_u$)
```

3 Analysis

3.1 Recording data

We analyzed the running time duration for five randomly generated graphs. We commuted the total time in milliseconds and did ten experiments for each graph as shown in Table 1, to get more accurate picture of how the algorithm performs on average.

Dijkstra Based All Source Shortest Path Algorithm															
Input Size		Iterations										Total	Best	Average	Worst
n	m	1	2	3	4	5	6	7	8	9	10				
2000	10000	6165	6144	6280	6157	6424	6194	6116	6398	6189	6449	62516	6116	6251.6	6449
3000	15000	22210	22708	22217	22682	22778	22658	22796	22299	22558	22790	225696	22210	22569.6	22796
4000	20000	52831	52089	52819	52137	51999	52254	52153	53047	52618	57765	529712	51999	52971.2	57765
5000	25000	100201	100541	100294	99437	100849	102102	100479	100226	100555	99056	1003740	99056	100374	102102
6000	30000	171586	171042	171318	170434	170914	176325	174021	174383	173221	172147	1725391	170434	172539.1	176325
Total		352993	352524	352928	350847	352964	359533	355565	356353	355141	358207	3547055	349815	354705.5	365437

Table 1: runtime computed in millisecond for Dijkstra algorithm.

After recording the data in Table 1, we draw a graph that represents the best, average, and worst time efficiency (Figure 1), and from the figure, we can conclude the time efficiency of the algorithm which is $\theta(n^2)$.

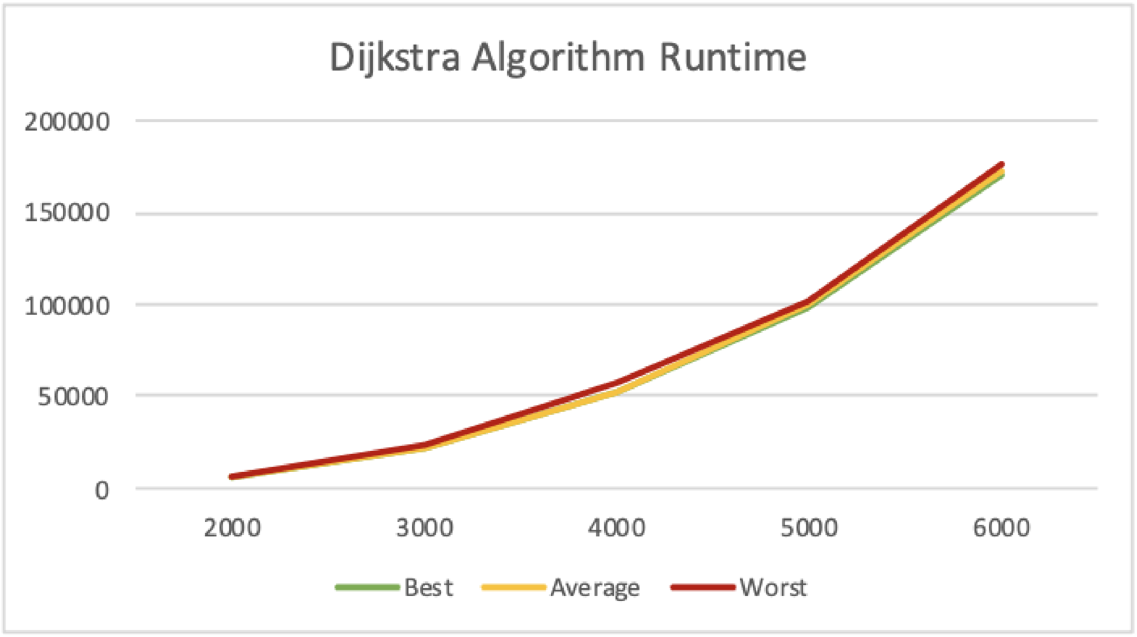


Figure 1: Dijkstra runtime graph

3.2 Comparing Empirical And Theoretical Expected Efficiencies

To compare between the empirical and theoretical efficiencies in each of the five random graphs, we will use the average empirical runtime time in milliseconds corresponding to its Expected theoretical time which is $\theta(|V|^2)$ for single path, because of implementing the priority queue as an unordered array, and after finding all shortest paths it becomes $\theta(\boldsymbol{n}(|\boldsymbol{V}|^{\mathbf{2}}))$.

n	Running time of Algorithm in ms	Expected theoretical time efficiency $\boldsymbol{n}(|\boldsymbol{V}|^{\mathbf{2}})$
2000	6252	8000000000
3000	22570	27000000000
4000	52971	64000000000
5000	100374	125000000000
6000	172539	216000000000

Table 2 comparison between empirical runtime and theoretical expected runtime

To be able to compare between them, we will take the ratio value.

	Ratio of running time	Ratio of expected time efficiency
Time of 3000/ Time of 2000	3.61004478567	3.375
Time of 4000/ Time of 3000	2.34696499778	2.37037037037
Time of 5000/ Time of 4000	1.89488588095	1.953125
Time of 6000/ Time of 5000	1.71896108554	1.728

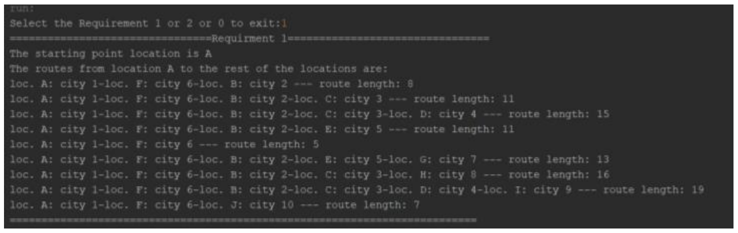
Table 3 ratio value for the empirical runtime and theoretical expected runtime

Table 3 shows that the two approaches are very close to each other, and the efficiency runtime for both gets smaller when the input gets larger.

4 Screenshots of Runtime

4.1 requirement#1 Screenshots

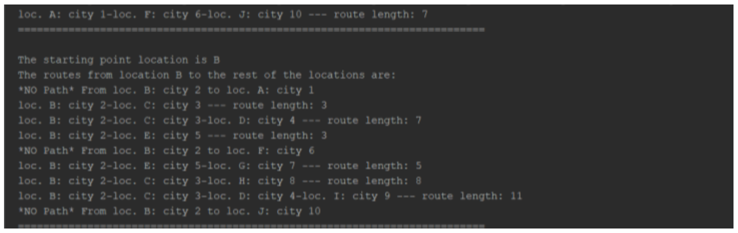
As shown in the following figure (Figure 2), the user shall enter the requirement number, in this example the user started with requirement #1. As can be concluded, the Location class generated a graph with nine locations. As can be seen, the Dijkstra algorithm started with the vertex “A” as a source showing all the optimal paths to all other vertices with their values.



```
run:
Select the Requirement 1 or 2 or 0 to exit:1
================================Requirment 1================================
The starting point location is A
The routes from location A to the rest of the locations are:
loc. A: city 1-loc. F: city 6-loc. B: city 2 --- route length: 8
loc. A: city 1-loc. F: city 6-loc. B: city 2-loc. C: city 3 --- route length: 11
loc. A: city 1-loc. F: city 6-loc. B: city 2-loc. C: city 3-loc. D: city 4 --- route length: 15
loc. A: city 1-loc. F: city 6-loc. B: city 2-loc. E: city 5 --- route length: 11
loc. A: city 1-loc. F: city 6 --- route length: 5
loc. A: city 1-loc. F: city 6-loc. B: city 2-loc. E: city 5-loc. G: city 7 --- route length: 13
loc. A: city 1-loc. F: city 6-loc. B: city 2-loc. C: city 3-loc. H: city 8 --- route length: 16
loc. A: city 1-loc. F: city 6-loc. B: city 2-loc. C: city 3-loc. D: city 4-loc. I: city 9 --- route length: 19
loc. A: city 1-loc. F: city 6-loc. J: city 10 --- route length: 7
==========================================================================
```

Figure 2: Requirement#1 - optimal paths from the source "A" to all other locations

After iterating all the possible paths that start with “A” as the source, Dijkstra algorithm will do so to the other locations in this order: “A” -> B -> “D” -> “E” -> “F” -> “G” -> “H” -> “I” -> “J” , finding the optimal path from each location as the source to all other locations. The following figure (Figure 3) shows all optimal paths from location “B” to all other locations.

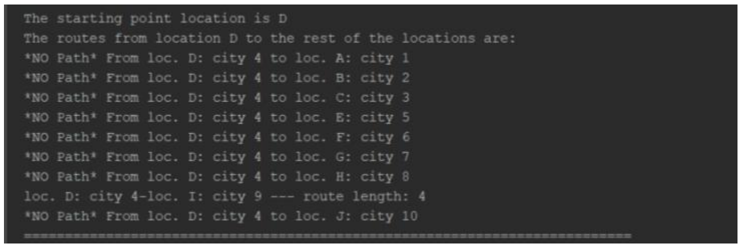


```
loc. A: city 1-loc. F: city 6-loc. J: city 10 --- route length: 7
==========================================================================

The starting point location is B
The routes from location B to the rest of the locations are:
*NO Path* From loc. B: city 2 to loc. A: city 1
loc. B: city 2-loc. C: city 3 --- route length: 3
loc. B: city 2-loc. C: city 3-loc. D: city 4 --- route length: 7
loc. B: city 2-loc. E: city 5 --- route length: 3
*NO Path* From loc. B: city 2 to loc. F: city 6
loc. B: city 2-loc. E: city 5-loc. G: city 7 --- route length: 5
loc. B: city 2-loc. C: city 3-loc. H: city 8 --- route length: 8
loc. B: city 2-loc. C: city 3-loc. D: city 4-loc. I: city 9 --- route length: 11
*NO Path* From loc. B: city 2 to loc. J: city 10
==========================================================================
```

Figure 3: Requirement#1 - optimal paths from the source "B" to all other locations

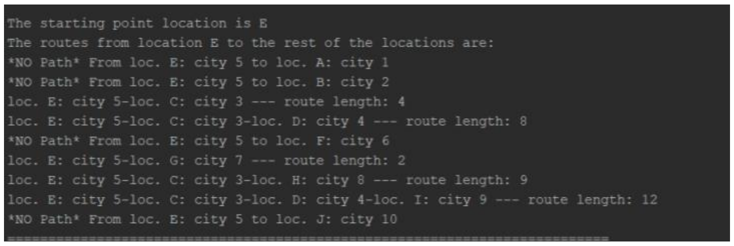
The following figure (Figure 4) shows the optimal paths from location “D” to all other locations of the graph.



```
The starting point location is D
The routes from location D to the rest of the locations are:
*NO Path* From loc. D: city 4 to loc. A: city 1
*NO Path* From loc. D: city 4 to loc. B: city 2
*NO Path* From loc. D: city 4 to loc. C: city 3
*NO Path* From loc. D: city 4 to loc. E: city 5
*NO Path* From loc. D: city 4 to loc. F: city 6
*NO Path* From loc. D: city 4 to loc. G: city 7
*NO Path* From loc. D: city 4 to loc. H: city 8
loc. D: city 4-loc. I: city 9 --- route length: 4
*NO Path* From loc. D: city 4 to loc. J: city 10
==========================================================================
```

Figure 4: Requirement#1 - optimal paths from the source "D" to all other locations

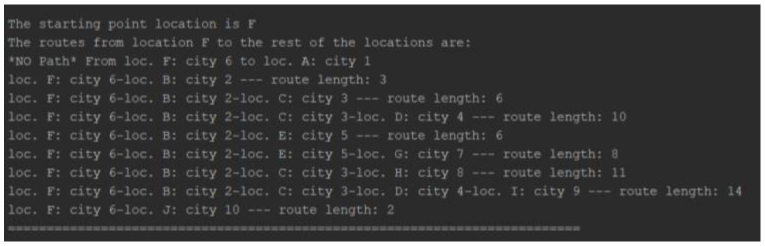
The following figure (Figure 5) shows the optimal paths from location “E” to all other locations of the graph.



```
The starting point location is E
The routes from location E to the rest of the locations are:
*NO Path* From loc. E: city 5 to loc. A: city 1
*NO Path* From loc. E: city 5 to loc. B: city 2
loc. E: city 5-loc. C: city 3 --- route length: 4
loc. E: city 5-loc. C: city 3-loc. D: city 4 --- route length: 8
*NO Path* From loc. E: city 5 to loc. F: city 6
loc. E: city 5-loc. G: city 7 --- route length: 2
loc. E: city 5-loc. C: city 3-loc. H: city 8 --- route length: 9
loc. E: city 5-loc. C: city 3-loc. D: city 4-loc. I: city 9 --- route length: 12
*NO Path* From loc. E: city 5 to loc. J: city 10
==========================================================================
```

Figure 5: Requirement#1 - optimal paths from the source "E" to all other locations

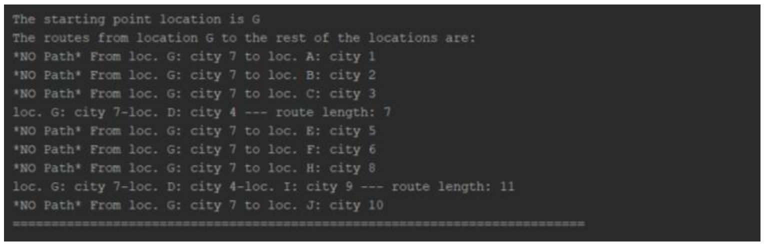
The following figure (Figure 6) shows the optimal paths from location “F” to all other locations of the graph.



```
The starting point location is F
The routes from location F to the rest of the locations are:
*NO Path* From loc. F: city 6 to loc. A: city 1
loc. F: city 6-loc. B: city 2 --- route length: 3
loc. F: city 6-loc. B: city 2-loc. C: city 3 --- route length: 6
loc. F: city 6-loc. B: city 2-loc. C: city 3-loc. D: city 4 --- route length: 10
loc. F: city 6-loc. B: city 2-loc. E: city 5 --- route length: 6
loc. F: city 6-loc. B: city 2-loc. E: city 5-loc. G: city 7 --- route length: 8
loc. F: city 6-loc. B: city 2-loc. C: city 3-loc. H: city 8 --- route length: 11
loc. F: city 6-loc. B: city 2-loc. C: city 3-loc. D: city 4-loc. I: city 9 --- route length: 14
loc. F: city 6-loc. J: city 10 --- route length: 2
==========================================================================
```

Figure 6: Requirement#1 - optimal paths from the source "F" to all other locations

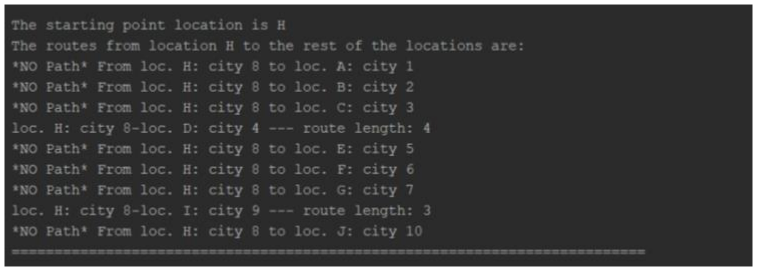
The following figure (Figure 7) shows the optimal paths from location “G” to all other locations of the graph.



```
The starting point location is G
The routes from location G to the rest of the locations are:
*NO Path* From loc. G: city 7 to loc. A: city 1
*NO Path* From loc. G: city 7 to loc. B: city 2
*NO Path* From loc. G: city 7 to loc. C: city 3
loc. G: city 7-loc. D: city 4 --- route length: 7
*NO Path* From loc. G: city 7 to loc. E: city 5
*NO Path* From loc. G: city 7 to loc. F: city 6
*NO Path* From loc. G: city 7 to loc. H: city 8
loc. G: city 7-loc. D: city 4-loc. I: city 9 --- route length: 11
*NO Path* From loc. G: city 7 to loc. J: city 10
==========================================================================
```

Figure 7: Requirement#1 - optimal paths from the source "G" to all other locations

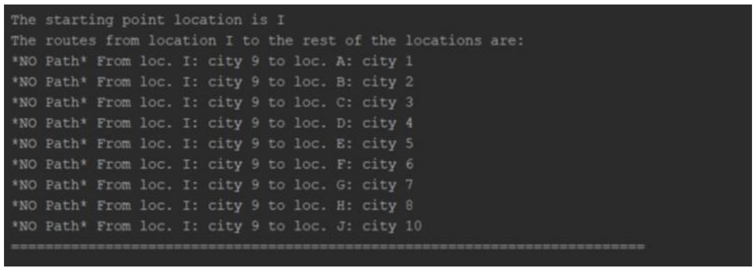
The following figure (Figure 8) shows the optimal paths from location “H” to all other locations of the graph.



```
The starting point location is H
The routes from location H to the rest of the locations are:
*NO Path* From loc. H: city 8 to loc. A: city 1
*NO Path* From loc. H: city 8 to loc. B: city 2
*NO Path* From loc. H: city 8 to loc. C: city 3
loc. H: city 8-loc. D: city 4 --- route length: 4
*NO Path* From loc. H: city 8 to loc. E: city 5
*NO Path* From loc. H: city 8 to loc. F: city 6
*NO Path* From loc. H: city 8 to loc. G: city 7
loc. H: city 8-loc. I: city 9 --- route length: 3
*NO Path* From loc. H: city 8 to loc. J: city 10
==========================================================================
```

Figure 8: Requirement#1 - optimal paths from the source "H" to all other locations

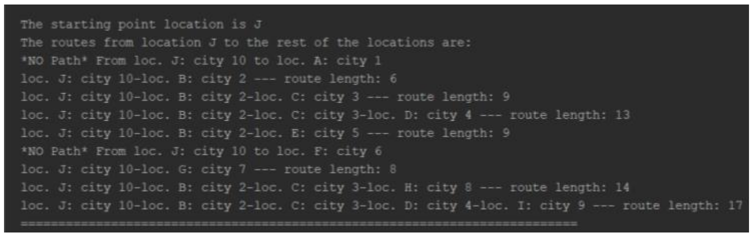
The following figure (Figure 9) shows the optimal paths from location “I” to all other locations of the graph.



```
The starting point location is I
The routes from location I to the rest of the locations are:
*NO Path* From loc. I: city 9 to loc. A: city 1
*NO Path* From loc. I: city 9 to loc. B: city 2
*NO Path* From loc. I: city 9 to loc. C: city 3
*NO Path* From loc. I: city 9 to loc. D: city 4
*NO Path* From loc. I: city 9 to loc. E: city 5
*NO Path* From loc. I: city 9 to loc. F: city 6
*NO Path* From loc. I: city 9 to loc. G: city 7
*NO Path* From loc. I: city 9 to loc. H: city 8
*NO Path* From loc. I: city 9 to loc. J: city 10
==========================================================================
```

Figure 9: Requirement#1 - optimal paths from the source "I" to all other locations

The following figure (Figure 10) shows the optimal paths from location “J” to all other locations of the graph.

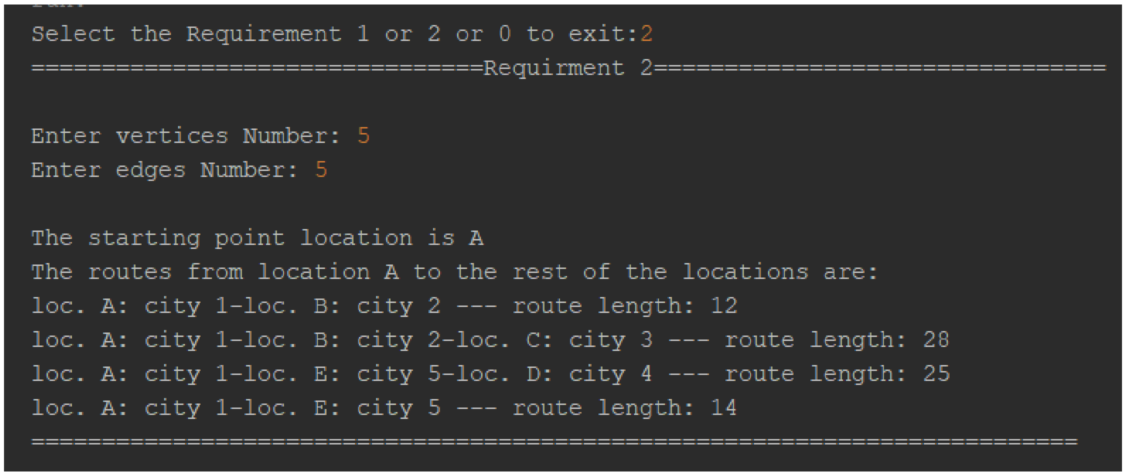


```
The starting point location is J
The routes from location J to the rest of the locations are:
*NO Path* From loc. J: city 10 to loc. A: city 1
loc. J: city 10-loc. B: city 2 --- route length: 6
loc. J: city 10-loc. B: city 2-loc. C: city 3 --- route length: 9
loc. J: city 10-loc. B: city 2-loc. C: city 3-loc. D: city 4 --- route length: 13
loc. J: city 10-loc. B: city 2-loc. E: city 5 --- route length: 9
*NO Path* From loc. J: city 10 to loc. F: city 6
loc. J: city 10-loc. G: city 7 --- route length: 8
loc. J: city 10-loc. B: city 2-loc. C: city 3-loc. H: city 8 --- route length: 14
loc. J: city 10-loc. B: city 2-loc. C: city 3-loc. D: city 4-loc. I: city 9 --- route length: 17
==========================================================================
```

Figure 10: Requirement#1 - optimal paths from the source "J" to all other locations

4.2 requirement#2 Screenshots

In the following figure (Figure 11) the user asked for requirement #2, in this requirement the user also should enter the number of vertices and edges, in this example they were both set to 5. As can be seen, the Dijkstra algorithm started with the vertex "A" as a source showing all the optimal paths to all of other vertices with their values.



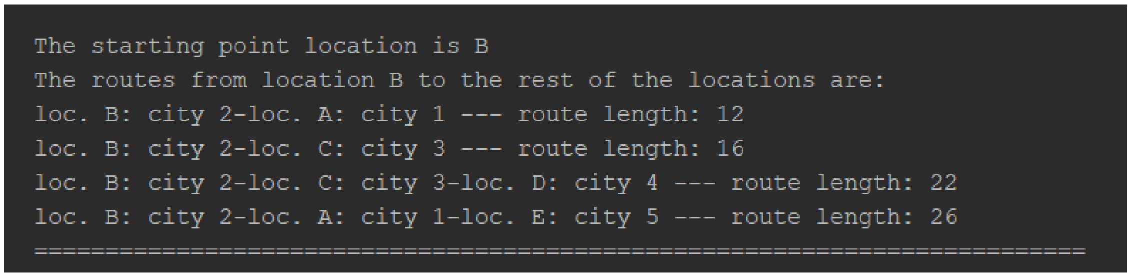
```
Select the Requirement 1 or 2 or 0 to exit:2
===============================Requirment 2===============================

Enter vertices Number: 5
Enter edges Number: 5

The starting point location is A
The routes from location A to the rest of the locations are:
loc. A: city 1-loc. B: city 2 --- route length: 12
loc. A: city 1-loc. B: city 2-loc. C: city 3 --- route length: 28
loc. A: city 1-loc. E: city 5-loc. D: city 4 --- route length: 25
loc. A: city 1-loc. E: city 5 --- route length: 14
==========================================================================
```

Figure 11: Requirement#2 - optimal paths from the source "A" to all other locations

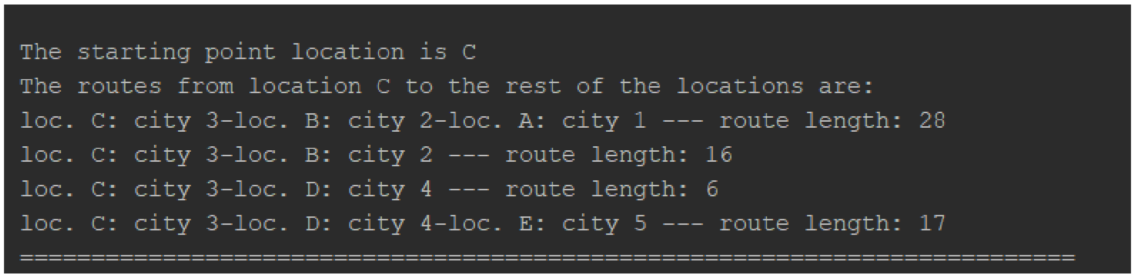
After iterating all the possible paths that start with "A" as the source, Dijkstra algorithm will do so to the other locations in this order: "A" -> B -> "C" -> "D" -> "E", finding the optimal path from each location as the source to all other locations. The following figure (Figure 12) shows all optimal paths from location "B" to all other locations.



```
The starting point location is B
The routes from location B to the rest of the locations are:
loc. B: city 2-loc. A: city 1 --- route length: 12
loc. B: city 2-loc. C: city 3 --- route length: 16
loc. B: city 2-loc. C: city 3-loc. D: city 4 --- route length: 22
loc. B: city 2-loc. A: city 1-loc. E: city 5 --- route length: 26
==========================================================================
```

Figure 12: Requirement#2 - optimal paths from the source "B" to all other locations

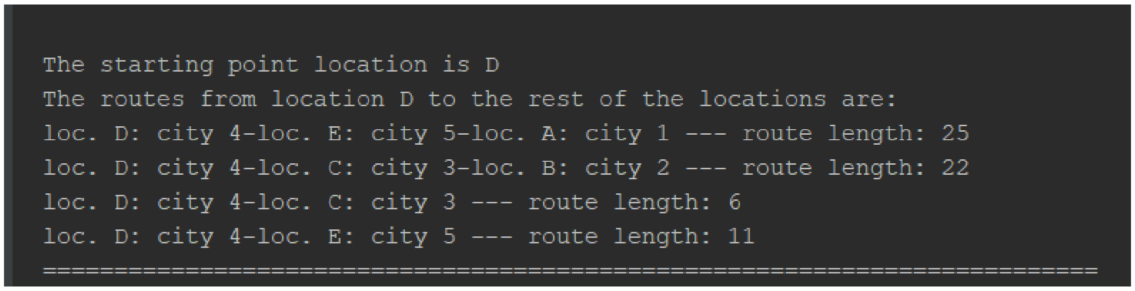
The following figure (Figure 13) shows the optimal paths from location "C" to all other locations of the graph.



```
The starting point location is C
The routes from location C to the rest of the locations are:
loc. C: city 3-loc. B: city 2-loc. A: city 1 --- route length: 28
loc. C: city 3-loc. B: city 2 --- route length: 16
loc. C: city 3-loc. D: city 4 --- route length: 6
loc. C: city 3-loc. D: city 4-loc. E: city 5 --- route length: 17
==========================================================================
```

Figure 13: Requirement#2 - optimal paths from the source "C" to all other locations

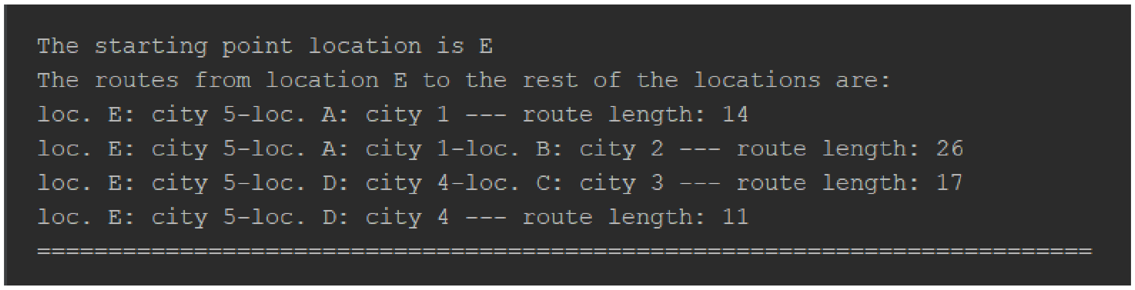
The following figure (Figure 14) shows the optimal paths from location “D” to all other locations of the graph.



```
The starting point location is D
The routes from location D to the rest of the locations are:
loc. D: city 4-loc. E: city 5-loc. A: city 1 --- route length: 25
loc. D: city 4-loc. C: city 3-loc. B: city 2 --- route length: 22
loc. D: city 4-loc. C: city 3 --- route length: 6
loc. D: city 4-loc. E: city 5 --- route length: 11
==========================================================================
```

Figure 14: Requirement#2 - optimal paths from the source "D" to all other locations

The following figure (Figure 15) shows the optimal paths from location “E” to all other locations of the graph.



```
The starting point location is E
The routes from location E to the rest of the locations are:
loc. E: city 5-loc. A: city 1 --- route length: 14
loc. E: city 5-loc. A: city 1-loc. B: city 2 --- route length: 26
loc. E: city 5-loc. D: city 4-loc. C: city 3 --- route length: 17
loc. E: city 5-loc. D: city 4 --- route length: 11
==========================================================================
```

Figure 15: Requirement#2 - optimal paths from the source "E" to all other locations

5 Difficulties

There were few difficulties that we have faced as team while working on this project, which are:

- Code Integration: integrating different pieces of code was a bit of a challenge which we learned from how to code as a team.
- Running algorithms with large inputs: since we are all using personal PC`s with simple CPU`s, running algorithms with large data took a lot of time.
- Accomplishing the analysis within a short time.

6 Conclusion

To summarize, we Implemented Dijkstra algorithm to compute the length of the shortest paths from all locations to the rest of the locations. We also generated randomly graphs to use it in analyzing the efficiency. After analyzing efficiency, we concluded that, the empirical time efficiency was nearly the same as the expected theoretical time efficiency.

7 Reference

(1) Levitin, A. (2011). *Introduction to the Design and Analysis of Algorithms*. Pearson Higher Ed.

8 Source Code

The Link of GitHub of the source code: https://github.com/Zinab0/CPCS324-Part2-Dijkstra-Algorithm-Empirical-Analysis