King Abdulaziz University

Faculty of Computing and Information Technology

Computer Science Department

Empirical Analysis of Floyed and Warshall and Digkstra Algorithms

Algorithms and data structure 2 (CPCS324)

PHASE 2

S2 1443H – Spring 2022

Lujain Alsefri – 1805544

Sara Alahmari – 1905503

Shaima Bashammakh - 1914892

Section: DAR

Table of Contents:

Illustrations:

Tables:

Figures:

1. Introduction

In the second phase of CPCS324 project, we will find and compute the shortest distance (path) between any two vertices of a graph by using two algorithms; Floyed and warshall algorithm or Dijkstra algorithm. In our project we will use both the Floyed and warshall and the Dijkstra algorithms to find the shortest distance. Floyed and warshall algorithm is one of the algorithms that follows the dynamic programming approach while the Dijkstra algorithm follows the greedy approach.

This project includes two requirements. The first requirement is reading the graph from a file and generating a graph based on the data that is read from that file. The second requirement is to generate a random graph based on a specific number of vertices and edges.

2. Algorithms

2.1 Floyed and Warshall Algorithm

ALGORITHM *Floyd*$(W[1..n, 1..n])$

```
//Implements Floyd's algorithm for the all-pairs shortest-paths problem
//Input: The weight matrix W of a graph with no negative-length cycle
//Output: The distance matrix of the shortest paths' lengths
D ← W  //is not necessary if W can be overwritten
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            D[i, j] ← min{D[i, j], D[i, k] + D[k, j]}
return D
```

Figure1: Floyed and Warshall

ALGORITHM *Dijkstra*(G, s)

for every vertex v in V
    $d_v \leftarrow \infty$; $p_v \leftarrow$ **null**
    Insert(Q, v, d_v)
$d_s \leftarrow 0$; *Decrease*(Q, s, d_s)
$V_T \leftarrow \varnothing$
for $i \leftarrow 0$ **to** $|V| - 1$ **do**
    $u^* \leftarrow$ *DeleteMin*(Q)
    $V_T \leftarrow V_T \cup \{u^*\}$
    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)

Figure2: Dijkstra Algorithm

3. Runtimes of the Algorithm

3.1 Floyed and Warshall Algorithm (Requirement 1)

n	m	1st run	2nd run	3rd run	4th run	5th run	total	best	average	worst
10	16	56	26	133	14	42	271	14	54.2	133

Table 1: The runtimes of requirement 1 (n=10 and m=16)

The average run time of n = 10 and m = 16 is: 54.2

The expected theoretical time efficiency is: n^3 which is 10^3 = 1000

3.2 Floyed and Warshall Algorithm (Requirement 2)

The first run time:

n	Running time of Algorithm	Expected theoretical time efficiency (n^3)
2000	23293	8000000000 = 8*10^9
3000	80505	27000000000 = 2.7*10^10
4000	195265	64000000000 = 6.4*10^10
5000	437313	125000000000 = 1.25*10^11
6000	717167	216000000000 = 2.16*10^11
	Ratio of running time	Ratio of expected time efficiency

Time of 3000/ Time of 2000	3.4562	3.375
Time of 4000/ Time of 3000	2.4255	2.3704
Time of 5000/ Time of 4000	2.2396	1.9531
Time of 6000/ Time of 5000	1.6399	1.728

Table 2: The first runtime of requirement 2

The second run time:

n	Running time of Algorithm	Expected theoretical time efficiency (n^3)
2000	23841	8000000000 = 8*10^9
3000	80095	27000000000 = 2.7*10^10
4000	212401	64000000000 = 6.4*10^10
5000	398050	125000000000 = 1.25*10^11
6000	733847	216000000000 = 2.16*10^11
	Ratio of running time	**Ratio of expected time efficiency**
Time of 3000/ Time of 2000	3.3595	3.375
Time of 4000/ Time of 3000	2.6519	2.3704
Time of 5000/ Time of 4000	1.8740	1.9531
Time of 6000/ Time of 5000	1.8436	1.728

Table 3: The second runtime of requirement 2

The third run time:

n	Running time of Algorithm	Expected theoretical time efficiency (n^3)
2000	23510	8000000000 = 8*10^9
3000	80187	27000000000 = 2.7*10^10
4000	216570	64000000000 = 6.4*10^10
5000	401037	125000000000 = 1.25*10^11
6000	738104	216000000000 = 2.16*10^11
	Ratio of running time	**Ratio of expected time efficiency**
Time of 3000/ Time of 2000	3.4108	3.375
Time of 4000/ Time of 3000	2.7008	2.3704
Time of 5000/ Time of 4000	1.8518	1.9531
Time of 6000/ Time of 5000	1.8405	1.728

Table 4: The third runtime of requirement 2

The fourth run time:

n	Running time of Algorithm	Expected theoretical time efficiency (n^3)
2000	25660	8000000000 = 8*10^9
3000	81503	27000000000 = 2.7*10^10
4000	220410	64000000000 = 6.4*10^10
5000	402523	125000000000 = 1.25*10^11
6000	710948	216000000000 = 2.16*10^11
	Ratio of running time	**Ratio of expected time efficiency**
Time of 3000/ Time of 2000	3.1763	3.375
Time of 4000/ Time of 3000	2.7043	2.3704
Time of 5000/ Time of 4000	1.8262	1.9531
Time of 6000/ Time of 5000	1.7662	1.728

Table 5: The fourth runtime of requirement 2

The fifth run time:

n	Running time of Algorithm	Expected theoretical time efficiency (n^3)
2000	23390	8000000000 = 8*10^9
3000	81390	27000000000 = 2.7*10^10
4000	214641	64000000000 = 6.4*10^10
5000	382764	125000000000 = 1.25*10^11
6000	701760	216000000000 = 2.16*10^11
	Ratio of running time	**Ratio of expected time efficiency**
Time of 3000/ Time of 2000	3.4797	3.375
Time of 4000/ Time of 3000	2.6372	2.3704
Time of 5000/ Time of 4000	1.7833	1.9531
Time of 6000/ Time of 5000	1.8334	1.728

Table 6: The fifth runtime of requirement 2

n	m	First Run	Second Run	Third Run	Fourth Run	Fifth Run	Total	Best	Average	Worst
2000	10000	23293	23841	23510	25660	23390	**119694**	23293	23938.8	25660
3000	15000	80505	80095	80187	81503	81390	**403680**	80095	80736	81503
4000	20000	195265	212401	216570	220410	214641	**1059287**	195265	211857.4	220410
5000	25000	437313	398050	401037	402523	382764	**2021687**	382764	404337.4	437313
6000	30000	717167	733847	738104	710948	701760	**3601826**	701760	720365.2	738104

Table 7: All runtimes of requirement 2 (Floyed and Warshall algorithm)

The average run time:

n	Running time of Algorithm (average)	Expected theoretical time efficiency (n^3)
2000	119694	8000000000 = 8*10^9
3000	403680	27000000000 = 2.7*10^10
4000	1059287	64000000000 = 6.4*10^10
5000	2021687	125000000000 = 1.25*10^11
6000	3601826	216000000000 = 2.16*10^11
	Ratio of running time	Ratio of expected time efficiency
Time of 3000/ Time of 2000	3.3726	3.375
Time of 4000/ Time of 3000	2.6241	2.3704
Time of 5000/ Time of 4000	1.9085	1.9531
Time of 6000/ Time of 5000	1.7816	1.728

Table 8: The average runtime of requirement 2 (Floyed and Warshall algorithm)

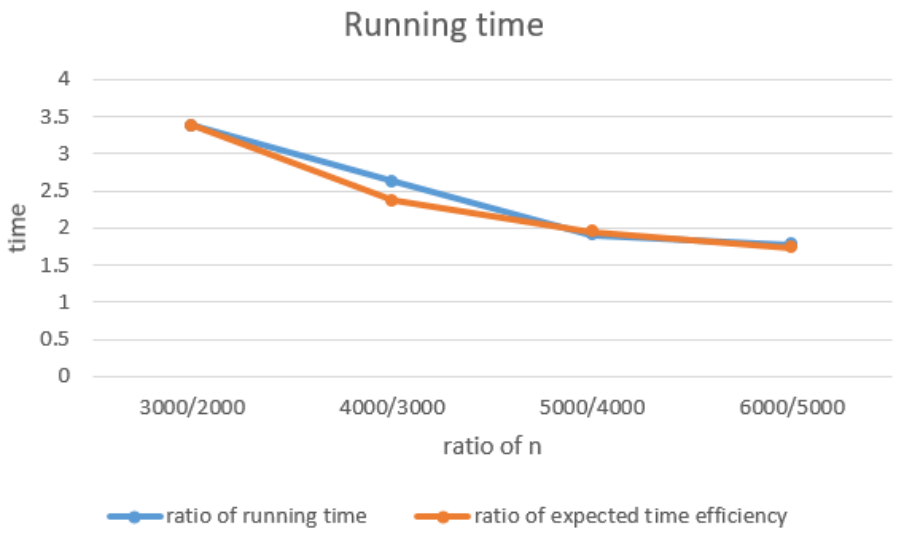

Figure3: The average running time of Floyed and Warshall algorithm

3.3 Dijkstra Algorithm (Requirement 1)

The run time of n = 10 and m = 16 is: 2.0

The expected theoretical time efficiency is: n^2 which is 10^2 = 100

3.4 Dijkstra Algorithm (Requirement 2)

n	m	First Run	Second Run	Third Run	Total	Best	Average	Worst
2000	10000	1258	745	748	**2751**	745	917	1258
3000	15000	1602	385	614	**2601**	385	867	1602
4000	20000	1966	1982	1948	**5896**	1948	1965.333	1982
5000	25000	6127	4514	7050	**17691**	4514	5897	7050
6000	30000	5601	3183	10492	**19276**	3183	6425.333	10492

Table 9: All runtimes of requirement 2 (Dijkstra algorithm)

The average run time:

n	Running time of Algorithm (average)	Expected theoretical time efficiency (n^2)
2000	917	4000000
3000	867	9000000
4000	1965.333	16000000
5000	5897	25000000
6000	6425.333	36000000
	Ratio of running time	**Ratio of expected time efficiency**
Time of 3000/ Time of 2000	0.9455	2.25
Time of 4000/ Time of 3000	2.2668	1.7778
Time of 5000/ Time of 4000	3.0005	1.5625
Time of 6000/ Time of 5000	1.0896	1.44

Table 10: The average runtime of requirement 2 (Dijkstra algorithm)

4. Empirical Analysis and Time efficiency

4.1 Floyed and Warshall Algorithm

Floyed and warshall algorithm is one of the algorithms that finds and computes the shortest distance (path) between every pair of vertices, where n is a number of vertices |V| and m is a number of edges |E|. Floyed and warshall algorithm has three for loops so the time efficiency will be Θ(n^3).

In this project we have 5 cases:

1- n=2000, m=10000
2- n=3000, m=15000
3- n=4000, m=20000
4- n=5000, m=25000
5- n=6000, m=30000

We did five experimental tests (five runs) for each case. According to the class efficiency of the algorithm and according to the ratios between running time and expected time efficiency results in Table8 and also based on the Figure 2, we can say that the running time (which is empirical analysis) of the Floyed and warshall algorithm and expected theoretical time efficiency of the Floyed and warshall algorithm have the same results approximately and they are extremely close.

The empirical analysis of the Floyed and Warshall algorithm proves that the theoretical time efficiency is correct.

4.2 Digkstra Algorithm

Digkstra algorithm is one of the algorithms that finds and computes the shortest distance (path) between every pair of vertices, where n is a number of vertices |V| and m is a number of edges |E|. Digkstra algorithm has two for loops so the time efficiency will be Θ(n^2).

In this project we have 5 cases:

1- n=2000, m=10000
2- n=3000, m=15000
3- n=4000, m=20000
4- n=5000, m=25000
5- n=6000, m=30000

We did three experimental tests (three runs) for each case. According to the class efficiency of the algorithm and according to the ratios between running time and expected time efficiency results in Table10 we can say that the running time (which is empirical analysis) of the Digkstra algorithm and expected theoretical time efficiency of the Digkstra algorithm algorithm have the different results. The empirical analysis of the Digkstra algorithm does not prove that the theoretical time efficiency because the run may be affected by the CPU of the device and by the programming environment.

5. Tools of the analysis

The tools that used in our project are:

1. NetBeans IDE 8.2: Used to run the code.
2. Java Language: Used to write and implement the code.
3. Microsoft Excel: Used to represent the data and to draw the plot.

6. Difficulties faced during the phase design

We have faced some difficulties, here are the major of them:

- Running the code for a large number of vertices and edges takes a lot of time and sometimes it comes up with an error.

7. Conclusion

In the end, we used a Floyed and warshall and a Digkstra algorithm to implement the two requirements.

According to the previous results and the empirical analysis, we found that the Digkstra algorithm is faster than the Flyoed algorithm and take a smallest running time while the Floyed and Warshall algorithm is the slowest.

Finally, the solution or the result that we got from the empirical analysis of Floyed and Warshall algorithm is considered to be true to a large extent, although there are many influencing factors like the speed of the computer, the CPU and the device that is used. But the Digkstra algorithm is not considered to be true because there are differences in the results.

8. Appendix

8.1 Floyed and Warshall -> Screenshots of the outputs (requirement one)

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 1

Vertex [A] connected with vertex [B] by edge has weight: 10
Vertex [A] connected with vertex [F] by edge has weight: 5

Vertex [B] connected with vertex [C] by edge has weight: 3
Vertex [B] connected with vertex [E] by edge has weight: 3

Vertex [C] connected with vertex [D] by edge has weight: 4
Vertex [C] connected with vertex [H] by edge has weight: 5

Vertex [D] connected with vertex [I] by edge has weight: 4

Vertex [E] connected with vertex [C] by edge has weight: 4
Vertex [E] connected with vertex [G] by edge has weight: 2

Vertex [F] connected with vertex [B] by edge has weight: 3
Vertex [F] connected with vertex [J] by edge has weight: 2

Vertex [G] connected with vertex [D] by edge has weight: 7

Vertex [H] connected with vertex [D] by edge has weight: 4
Vertex [H] connected with vertex [I] by edge has weight: 3


Vertex [J] connected with vertex [B] by edge has weight: 6
Vertex [J] connected with vertex [G] by edge has weight: 8

The matrix (graph) bafore Floyed Warshal algorithm.
   A   B   C   D   E   F   G   H   I   J
A  0   10  INF INF INF 5   INF INF INF INF
B  INF 0   3   INF 3   INF INF INF INF INF
C  INF INF 0   4   INF INF INF 5   INF INF
D  INF INF INF 0   INF INF INF INF 4   INF
E  INF INF 4   INF 0   INF 2   INF INF INF
F  INF 3   INF INF INF 0   INF INF INF 2
G  INF INF INF 7   INF INF 0   INF INF INF
H  INF INF INF 4   INF INF INF 0   3   INF
I  INF INF INF INF INF INF INF INF 0   INF
J  INF 6   INF INF INF INF 8   INF INF 0
```

D(0)

	A	B	C	D	E	F	G	H	I	J
A	0	10	INF	INF	INF	5	INF	INF	INF	INF
B	INF	0	3	INF	3	INF	INF	INF	INF	INF
C	INF	INF	0	4	INF	INF	INF	5	INF	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	INF	0	INF	2	INF	INF	INF
F	INF	3	INF	INF	INF	0	INF	INF	INF	2
G	INF	INF	INF	7	INF	INF	0	INF	INF	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	INF	INF	INF	INF	8	INF	INF	0

D(1)

	A	B	C	D	E	F	G	H	I	J
A	0	10	INF	INF	INF	5	INF	INF	INF	INF
B	INF	0	3	INF	3	INF	INF	INF	INF	INF
C	INF	INF	0	4	INF	INF	INF	5	INF	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	INF	0	INF	2	INF	INF	INF
F	INF	3	INF	INF	INF	0	INF	INF	INF	2
G	INF	INF	INF	7	INF	INF	0	INF	INF	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	INF	INF	INF	INF	8	INF	INF	0

D(2)

	A	B	C	D	E	F	G	H	I	J
A	0	10	13	INF	13	5	INF	INF	INF	INF
B	INF	0	3	INF	3	INF	INF	INF	INF	INF
C	INF	INF	0	4	INF	INF	INF	5	INF	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	INF	0	INF	2	INF	INF	INF
F	INF	3	6	INF	6	0	INF	INF	INF	2
G	INF	INF	INF	7	INF	INF	0	INF	INF	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	INF	9	INF	8	INF	INF	0

D(3)

	A	B	C	D	E	F	G	H	I	J
A	0	10	13	17	13	5	INF	18	INF	INF
B	INF	0	3	7	3	INF	INF	8	INF	INF
C	INF	INF	0	4	INF	INF	INF	5	INF	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	INF	INF
F	INF	3	6	10	6	0	INF	11	INF	2
G	INF	INF	INF	7	INF	INF	0	INF	INF	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	INF	0

D(4)

	A	B	C	D	E	F	G	H	I	J
A	0	10	13	17	13	5	INF	18	21	INF
B	INF	0	3	7	3	INF	INF	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	INF	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

D(5)

	A	B	C	D	E	F	G	H	I	J
A	0	10	13	17	13	5	15	18	21	INF
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

D(6)

	A	B	C	D	E	F	G	H	I	J
A	0	8	11	15	11	5	13	16	19	7
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

D(7)

	A	B	C	D	E	F	G	H	I	J
A	0	8	11	15	11	5	13	16	19	7
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

D(8)

	A	B	C	D	E	F	G	H	I	J
A	0	8	11	15	11	5	13	16	19	7
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

D(9)

	A	B	C	D	E	F	G	H	I	J
A	0	8	11	15	11	5	13	16	19	7
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

D(10)

	A	B	C	D	E	F	G	H	I	J
A	0	8	11	15	11	5	13	16	19	7
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

The matrix (graph) after Floyed Warshal algorithm.

	A	B	C	D	E	F	G	H	I	J
A	0	8	11	15	11	5	13	16	19	7
B	INF	0	3	7	3	INF	5	8	11	INF
C	INF	INF	0	4	INF	INF	INF	5	8	INF
D	INF	INF	INF	0	INF	INF	INF	INF	4	INF
E	INF	INF	4	8	0	INF	2	9	12	INF
F	INF	3	6	10	6	0	8	11	14	2
G	INF	INF	INF	7	INF	INF	0	INF	11	INF
H	INF	INF	INF	4	INF	INF	INF	0	3	INF
I	INF	INF	INF	INF	INF	INF	INF	INF	0	INF
J	INF	6	9	13	9	INF	8	14	17	0

```
The matrix (graph) after Floyed Warshal algorithm.
   A   B   C   D   E   F   G   H   I   J
A  0   8   11  15  11  5   13  16  19  7
B  INF 0   3   7   3   INF 5   8   11  INF
C  INF INF 0   4   INF INF INF 5   8   INF
D  INF INF INF 0   INF INF INF INF 4   INF
E  INF INF 4   8   0   INF 2   9   12  INF
F  INF 3   6   10  6   0   8   11  14  2
G  INF INF INF 7   INF INF 0   INF 11  INF
H  INF INF INF 4   INF INF INF 0   3   INF
I  INF INF INF INF INF INF INF INF 0   INF
J  INF 6   9   13  9   INF 8   14  17  0


----------------------------------------
The Runtime (in MilliSecond) of n= 10 and m= 16 :116.0
```

Figure 4: The sample output of n=10 and m=16 (requirement 1 in Flyed and Warshall algorithm)

8.2 Floyed and Warshall -> Screenshots of the outputs (requirement two)

Sample output 1:

```
----------------------------------------------------------------------
                    CPCS325 Project Phase 2
                  --Floyed Warshal Algorithm--
----------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 1
----------------------------------------
The Runtime (MilliSecond)of n=2000, m=10000: 23293.0
```

Figure 5: The sample output of case 1 (n= 2000 and m= 10000) in Flyed and Warshall algorithm

Sample output 2:

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
                  --Floyed Warshal Algorithm--
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 2
---------------------------------------
The Runtime (MilliSecond)of n=3000, m=15000: 80095.0
```

Figure 6: The sample output of case 2 (n= 3000 and m= 15000) 1 in Flyed and Warshall algorithm

Sample output 3:

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
                  --Floyed Warshal Algorithm--
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 3
---------------------------------------
The Runtime (MilliSecond)of n=4000, m=20000: 212401.0
```

Figure 7: The sample output of case 3 (n= 4000 and m= 20000) in Flyed and Warshall algorithm

Sample output 4:

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
                  --Floyed Warshal Algorithm--
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 4
---------------------------------------
The Runtime (MilliSecond)of n=5000, m=25000: 398050.0
```

Figure 8: The sample output of case 4 (n= 5000 and m= 25000) in Flyed and Warshall algorithm

Sample output 5:

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
                  --Floyed Warshal Algorithm--
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 5
---------------------------------------
The Runtime (MilliSecond) of n=6000, m=30000: 733847.0
```

Figure 9: The sample output of case 5 (n= 6000 and m= 30000) in Floyed and Warshall algorithm

8.3 Digkstra -> Screenshots of the outputs (requirement one)

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 1
Choose one of the following Algorithm:
 1- Floyd-Warshall algorithm.
 2- Dijkstra algorithm.
Enter your choice: 2

Vertex [A] connected with vertex [B] by edge has weight: 10
Vertex [A] connected with vertex [F] by edge has weight: 5

Vertex [B] connected with vertex [C] by edge has weight: 3
Vertex [B] connected with vertex [E] by edge has weight: 3

Vertex [C] connected with vertex [D] by edge has weight: 4
Vertex [C] connected with vertex [H] by edge has weight: 5

Vertex [D] connected with vertex [I] by edge has weight: 4

Vertex [E] connected with vertex [C] by edge has weight: 4
Vertex [E] connected with vertex [G] by edge has weight: 2

Vertex [F] connected with vertex [B] by edge has weight: 3
Vertex [F] connected with vertex [J] by edge has weight: 2

Vertex [G] connected with vertex [D] by edge has weight: 7
```

```
****************** Dijkstra Algorithm ******************
All the shortest distances from the source vertex to other vertices in graph
Shortest Distance from 'A' to 'A' is 0, the Path: A -(0)->  A and the lenght is 0
Shortest Distance from 'A' to 'B' is 8, the Path: A -(0)-> F -(5)-> B -(3)->  B and the lenght is 8
Shortest Distance from 'A' to 'C' is 11, the Path: A -(0)-> F -(5)-> B -(3)-> C -(3)->  C and the lenght is 11
Shortest Distance from 'A' to 'D' is 15, the Path: A -(0)-> F -(5)-> B -(3)-> C -(3)-> D -(4)->  D and the lenght is 15
Shortest Distance from 'A' to 'E' is 11, the Path: A -(0)-> F -(5)-> B -(3)-> E -(3)->  E and the lenght is 11
Shortest Distance from 'A' to 'F' is 5, the Path: A -(0)-> F -(5)->  F and the lenght is 5
Shortest Distance from 'A' to 'G' is 13, the Path: A -(0)-> F -(5)-> B -(3)-> E -(3)-> G -(2)->  G and the lenght is 13
Shortest Distance from 'A' to 'H' is 16, the Path: A -(0)-> F -(5)-> B -(3)-> C -(3)-> H -(5)->  H and the lenght is 16
Shortest Distance from 'A' to 'I' is 19, the Path: A -(0)-> F -(5)-> B -(3)-> C -(3)-> D -(4)-> I -(4)->  I and the lenght is 19
Shortest Distance from 'A' to 'J' is 7, the Path: A -(0)-> F -(5)-> J -(2)->  J and the lenght is 7

---------------------------------------
The Runtime (in MilliSecond) of n= 10 and m= 16 :2.0
```

Figure 10: The sample output of n=10 and m=16 (requirement 1 in Digkstra algorithm)

8.4 Digkstra -> Screenshots of the outputs (requirement two)

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 1
Choose one of the following Algorithm:
 1- Floyd-Warshall algorithm.
 2- Dijkstra algorithm.
Enter your choice: 2

***************** Dijkstra Algorithm *****************
All the shortest distances from the source vertex to other vertices in graph
---------------------------------------
The Runtime (MilliSecond)of n=2000, m=10000: 757.0
```

Figure 11: The sample output of case 1 (n= 2000 and m= 10000) in Digkstra algorithm

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 2
Choose one of the following Algorithm:
 1- Floyd-Warshall algorithm.
 2- Dijkstra algorithm.
Enter your choice: 2

***************** Dijkstra Algorithm *****************
All the shortest distances from the source vertex to other vertices in graph
---------------------------------------
The Runtime (MilliSecond)of n=3000, m=15000: 2760.0
```

Figure 12: The sample output of case 2 (n= 3000 and m= 15000) in Digkstra algorithm

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 3
Choose one of the following Algorithm:
 1- Floyd-Warshall algorithm.
 2- Dijkstra algorithm.
Enter your choice: 2

****************** Dijkstra Algorithm ******************
All the shortest distances from the source vertex to other vertices in graph
---------------------------------------
The Runtime (MilliSecond)of n=4000, m=20000: 4009.0
```

Figure 13: The sample output of case 3 (n= 4000 and m= 20000) in Digkstra algorithm

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 4
Choose one of the following Algorithm:
 1- Floyd-Warshall algorithm.
 2- Dijkstra algorithm.
Enter your choice: 2

****************** Dijkstra Algorithm ******************
All the shortest distances from the source vertex to other vertices in graph
---------------------------------------
The Runtime (MilliSecond)of n=5000, m=25000: 7826.0
```

Figure 14: The sample output of case 4 (n= 5000 and m= 25000) in Digkstra algorithm

```
---------------------------------------------------------------------
                    CPCS325 Project Phase 2
---------------------------------------------------------------------
Choose one of the following options:
 1- Read the graph vertecies and edges from the file.
 2- Choose a specific number of vertecies and edges.
 3- End the program.
Enter your choice: 2
Choose the number of vertecies and edges from the following list:
 (n) is number of vertecies, (m) is number of edges
 1- n=2000 , m=10000
 2- n=3000 , m=15000
 3- n=4000 , m=20000
 4- n=5000 , m=25000
 5- n=6000 , m=30000

Enter your choice: 5
Choose one of the following Algorithm:
 1- Floyd-Warshall algorithm.
 2- Dijkstra algorithm.
Enter your choice: 2

***************** Dijkstra Algorithm *****************
All the shortest distances from the source vertex to other vertices in graph
--------------------------------------
The Runtime (MilliSecond) of n=6000, m=30000: 12260.0
```

Figure 15: The sample output of case 5 (n= 6000 and m= 30000) in Digkstra algorithm

8.5 GitHub

https://github.com/lujainAAlsefri/CPCS324-Project-Phase2