G. H. RAISONI COLLEGE OF ENGG., NAGPUR
(An Autonomous Institute under UGC Act 1956)
Department of Computer Science & Engg.

Date: 20/08/2020

Practical Subject: Design and Analysis of Algorithms
Session: 2020-21

Student Details:

Roll Number	58
Name	Shivam Tawari
Semester	3
Section	A
Branch	Artificial Intelligence

Practical Details: Practical Number- 8

Practical Aim	To implement and analyze time complexity of All pair shortest path algorithm.
Theory & Algorithm	**Theory:** ***All Pair Shortest Path Algorithm:*** Floyd-Warshall's algorithm is for finding shortest paths in a weighted graph with positive or negative edge weights. A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between all pair of vertices. With a little variation, it can print the shortest path and can detect negative cycles in a graph. Floyd-Warshall is a Dynamic-Programming algorithm. This algorithm works for both the directed and undirected weighted graphs. But it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

	Algorithm:
	
	Step 1: START
	Step 2- floydWarshal(cost), Input is The cost matrix of given Graph.
	Step 3- Begin
	for k := 0 to n, do
	for i := 0 to n, do
	for j := 0 to n, do
	if cost[i,k] + cost[k,j] < cost[i,j], then
	cost[i,j] := cost[i,k] + cost[k,j]
	done
	done
	done
	Step 4- Display the current cost matrix
	Step 5: STOP

Complexity	***Time Complexity:*** O(n3)

Program	main.cpp
	1 #include <iostream>
	2 #include <conio.h>
	3 using namespace std;
	4 void floyds(int b[][7])
	5 {
	6 int i, j, k;
	7 for (k = 0; k < 7; k++)
	8 {
	9 for (i = 0; i < 7; i++)
	10 {
	11 for (j = 0; j < 7; j++)
	12 {
	13 if ((b[i][k] * b[k][j] != 0) && (i != j))
	14 {
	15 if ((b[i][k] + b[k][j] < b[i][j]) || (b[i][j] == 0))
	16 {
	17 b[i][j] = b[i][k] + b[k][j];
	18 }
	19 }
	20 }
	21 }
	22 }
	23 for (i = 0; i < 7; i++)
	24 {
	25 cout<<"\nMinimum Cost With Respect to Node:"<<i<<endl;
	26 for (j = 0; j < 7; j++)
	27 {
	28 cout<<b[i][j]<<"\t";
	29 }

```
30
31      }
32  }
33  int main()
34  {
35      int b[7][7];
36      cout<<"\n Name: Shivam Tawari";
37      cout<<"\n Roll no: A-58 Semester: 3";
38      cout<<"\n ENTER VALUES OF ADJACENCY MATRIX\n\n";
39      for (int i = 0; i < 7; i++)
40      {
41          cout<<"enter values for "<<(i+1)<<" row"<<endl;
42          for (int j = 0; j < 7; j++)
43          {
44              cin>>b[i][j];
45          }
46      }
47      floyds(b);
48      getch();
49  }
```

Output

```
Name: Shivam Tawari
Roll no: A-58 Semester: 3
ENTER VALUES OF ADJACENCY MATRIX

enter values for 1 row
2
3
6
1
2
3
6
enter values for 2 row
2
6
5
5
9
4
11
enter values for 3 row
2
6
3
1
0
5
6
enter values for 4 row
12
13
5
6
2
3
1
enter values for 5 row
0
6
6
```

```
1
2
2
enter values for 6 row
6
1
5
65
5
2
9
enter values for 7 row
4
7
6
2
9
8
1
```

```
Minimum Cost With Respect to Node:0
2       3       6       1       2       3       2
Minimum Cost With Respect to Node:1
2       6       5       3       4       4       4
Minimum Cost With Respect to Node:2
2       5       3       1       3       4       2
Minimum Cost With Respect to Node:3
5       4       5       6       2       3       1
Minimum Cost With Respect to Node:4
5       3       6       4       1       2       2
Minimum Cost With Respect to Node:5
3       1       5       4       5       2       5
Minimum Cost With Respect to Node:6
4       6       6       2       4       5       1
```