UCS1403 Design and Analysis of Algorithms
AY: 2021-22

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

DAA Lab – Assignment 8

Dynamic Programming

1. To Implement Warshall's Algorithm Transitive Closure using DP.

Code:

```
# Python program to implement the Warshall algorithm for Transitive Closure
using Dynamic Programming

INF = 99

def warshall(G):
      v = len(G)
      distance = [el for el in [row for row in G]]

      for k in range(v):
                  for i in range(v):
                              for j in range(v):
                                          distance[i][j] = max(distance[i][j],
distance[i][k] * distance[k][j])
      print_solution(distance)

def print_solution(distance):
      print("Solution: ")
      v = len(distance)
      for i in range(v):
                  for j in range(v):
                              if(distance[i][j] == INF):
                                          print("INF", end=" ")
                              else:
                                          print(distance[i][j], end="  ")
                  print(" ")

G = []
n = int(input("Enter no. of vertices: "))

print("Enter adj matrix: ")
for i in range(n):
      row = list(map(int,input().split()))
      if (len(row) != n):
            print("Invalid no. of columns entered. Enter again.")
            i-=1
      else:
            G.append(row)
print()

warshall(G)
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

Output:

```
~/DAA-Exercise8$ python3 warshall.py
Enter no. of vertices: 4
Enter adj matrix:
0 1 0 0
0 0 0 1
0 0 0 0
1 0 1 0

Solution:
1  1  1  1
1  1  1  1
0  0  0  0
1  1  1  1
~/DAA-Exercise8$ python3 warshall.py
Enter no. of vertices: 4
Enter adj matrix:
0 1 0 0
0 0 1 0
0 0 0 1
0 0 0 0

Solution:
0  1  1  1
0  0  1  1
0  0  0  1
0  0  0  0
~/DAA-Exercise8$
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

2. To Implement Floyd's Algorithm for all pair shortest path using DP.

Code:

```
# Program to implement Floyd's Algorithm for all pair shortest path using
Dynamic Programming

INF = 99

def floyd(G):
      v = len(G)
      distance = [el for el in [row for row in G]]

      for k in range(v):
                  for i in range(v):
                              for j in range(v):
                                          distance[i][j] = min(distance[i][j],
distance[i][k] + distance[k][j])
      print_solution(distance)

def print_solution(distance):
      print("Solution: ")
      v = len(distance)
      for i in range(v):
                  for j in range(v):
                              if(distance[i][j] == INF):
                                          print("INF", end=" ")
                              else:
                                          print(distance[i][j], end="  ")
                  print(" ")

G = []
n = int(input("Enter no. of vertices: "))

print("Enter adj matrix: ")
for i in range(n):
      row = list(map(int,input().split()))
      if (len(row) != n):
            print("Invalid no. of columns entered. Enter again.")
            i-=1
      else:
            G.append(row)
print()

floyd(G)
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

Output:

```
~/DAA-Exercise8$ python3 floyd.py
Enter no. of vertices: 4
Enter adj matrix:
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0

Solution:
0  10  3  4
2  0  5  6
7  7  0  1
6  16  9  0
~/DAA-Exercise8$ python3 floyd.py
Enter no. of vertices: 5
Enter adj matrix:
0 2 99 1 8
6 0 3 2 99
99 99 0 4 99
99 99 2 0 3
3 99 99 99 0

Solution:
0  2  3  1  4
6  0  3  2  5
10  12  0  4  7
6  8  2  0  3
3  5  6  4  0
```