TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

Dharan-8

A Lab Report On: Implementing Warshall's Algorithm

Submitted By

Name: Dhiraj KC
Roll No. : PUR077BEI014
Faculty: Electronics, Communication and Information
Group: A

Submitted To

Department Of Electronics and Computer Engineering
Checked By: Pukar Karki

TITLE: IMPLEMENTING WARSHALL'S ALGORITHM

THEORY

Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. 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

1. Start
2. Initialize the shortest paths between any 2 vertices with Infinity.
3. Find all pair shortest paths that use 0 intermediate vertices, then find the shortest paths that use 1 intermediate vertex and so on.. until using all N vertices as intermediate nodes.
4. Minimize the shortest paths between any 2 pairs in the previous operation.
5. For any 2 vertices (i,j) , one should actually minimize the distances between this pair using the first K nodes, so the shortest path will be: min(dist[i][k]+dist[k][j],dist[i][j]).
6. Stop

PROGRAM

```
# Floyd Warshall Algorithm in python
# The number of vertices
nV = 4
INF = 999
# Algorithm implementation
def floyd_warshall(G):
  distance = list(map(lambda i: list(map(lambda j: j, i)), G))
  # Adding vertices individually
  for k in range(nV):
    for i in range(nV):
      for j in range(nV):
        distance[i][j] = min(distance[i][j], distance[i][k] + distance[k][j])
  print_solution(distance)

# Printing the solution
def print_solution(distance):
  for i in range(nV):
    for j in range(nV):
      if (distance[i][j] == INF):
        print("INF", end=" ")
      else:
        print(distance[i][j], end="  ")
    print(" ")

G = [[0, 3, INF, 5],
     [2, 0, INF, 4],
     [INF, 1, 0, INF],
     [INF, INF, 2, 0]]

floyd_warshall(G)
```

OUTPUT

0 3 7 5
2 0 6 4
3 1 0 5
5 3 2 0