**TITLE 32**

Write a program to implement a single source shortest path algorithm. Either Bellman-Ford or Dijkstra’s algorithm

**OBJECTIVE:**

By the end of the above C program we will be writing a program to implement a single source shortest path algorithm of Bellman-Ford

**PROBLEM STATEMENT:**

In this problem we write a Bellman-Ford algorithm for the shortest path. Input from user:

Enter no. of vertices:

Once the input is collected and stored, the output is printed.

**ALGORITHM:**

START

Define variables: i , u, v, k, distance, parent, S, flag

INPUT: Read from the keyboard

COMPUTATION: Computing the shortest path algorithm

DISPLAY: Displaying the shortest path program using Bellman-Ford algorithm

STOP

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

int Bellman\_Ford(int G[20][20] , int V, int E, int edge[20][2])

{

int i,u,v,k,distance[20],parent[20],S,flag=1;

for(i=0;i<V;i++)

distance[i] = 1000 , parent[i] = -1 ;

printf("Enter source: ");

scanf("%d",&S);

distance[S-1]=0 ;

for(i=0;i<V-1;i++)

{

for(k=0;k<E;k++)

{

u = edge[k][0] , v = edge[k][1] ;

if(distance[u]+G[u][v] < distance[v])

distance[v] = distance[u] + G[u][v] , parent[v]=u ;

}

}

for(k=0;k<E;k++)

{

u = edge[k][0] , v = edge[k][1] ;

if(distance[u]+G[u][v] < distance[v])

flag = 0 ;

}

if(flag)

for(i=0;i<V;i++)

printf("Vertex %d -> cost = %d parent = %d\n",i+1,distance[i],parent[i]+1);

return flag;

}

int main()

{

int V,edge[20][2],G[20][20],i,j,k=0;

printf("BELLMAN FORD\n");

printf("Enter no. of vertices: ");

scanf("%d",&V);

printf("Enter graph in matrix form:\n");

for(i=0;i<V;i++)

for(j=0;j<V;j++)

{

scanf("%d",&G[i][j]);

if(G[i][j]!=0)

edge[k][0]=i,edge[k++][1]=j;

}

if(Bellman\_Ford(G,V,k,edge))

printf("\nNo negative weight cycle\n");

else printf("\nNegative weight cycle exists\n");

return 0;

}

**CONCLUSION:**

By the simulation of the above C program I learnt how to create a program for shortest path using Bellman-Ford algorithm.

**OUTPUT:**

BELLMAN FORD

Enter no. of vertices: 5

Enter graph in matrix form:

0 1 3 4 5

9 6 5 0 8

1 3 7 8 2

4 9 2 0 7

9 6 2 1 8

Enter source: 1

Vertex 1 --> cost = 0 parent = 0

Vertex 2 --> cost = 3 parent = 1

Vertex 3 --> cost = 4 parent = 1

Vertex 4 --> cost = 4 parent = 1

Vertex 5 --> cost = 5 parent = 1

No negative weight cycle