**EXP 8**

Code:

#include <stdio.h>

#include <limits.h>

// Define the maximum number of vertices in the graph

#define MAX\_VERTICES 100

// Define structure for representing edges

struct Edge {

int source, destination, weight;

};

// Define structure for representing a graph

struct Graph {

int vertices, edges;

struct Edge edge[MAX\_VERTICES];

};

// Initialize the graph

void initGraph(struct Graph \*graph, int vertices, int edges) {

graph->vertices = vertices;

graph->edges = edges;

}

// Bellman-Ford algorithm

void bellmanFord(struct Graph \*graph, int source) {

int distance[MAX\_VERTICES];

// Initialize distances from source to all other vertices as INFINITY

int i, j;

for (i = 0; i < graph->vertices; i++) {

distance[i] = INT\_MAX;

}

distance[source] = 0; // Distance from source to itself is 0

// Relax all edges for V-1 times

for (i = 1; i <= graph->vertices - 1; i++) {

for (j = 0; j < graph->edges; j++) {

int u = graph->edge[j].source;

int v = graph->edge[j].destination;

int weight = graph->edge[j].weight;

if (distance[u] != INT\_MAX && distance[u] + weight < distance[v]) {

distance[v] = distance[u] + weight;

}

}

}

// Check for negative weight cycles

for (i = 0; i < graph->edges; i++) {

int u = graph->edge[i].source;

int v = graph->edge[i].destination;

int weight = graph->edge[i].weight;

if (distance[u] != INT\_MAX && distance[u] + weight < distance[v]) {

printf("Graph contains negative weight cycle\n");

return;

}

}

// Print the distances

printf("Vertex Distance from Source\n");

for (i = 0; i < graph->vertices; i++) {

printf("%d \t\t %d\n", i, distance[i]);

}

}

int main() {

struct Graph graph;

int vertices, edges, source;

printf("Enter number of vertices and edges: ");

scanf("%d %d", &vertices, &edges);

initGraph(&graph, vertices, edges);

printf("Enter source vertex: ");

scanf("%d", &source);

printf("Enter edges (source destination weight):\n");

for (int i = 0; i < edges; i++) {

scanf("%d %d %d", &graph.edge[i].source, &graph.edge[i].destination, &graph.edge[i].weight);

}

bellmanFord(&graph, source);

return 0;

}