NAME- Aniket Kalbhor

PRN- 12210601

ROLL- CSB-48

CODE- 10a**)Implement Dijkstras Algorithm to generate MST using Adjacency Matrix.**

#include <stdio.h>

#include <stdlib.h>

int findMinDist(int dist[10], int visited[10], int vertices)

{

int min = 9999;

int index;

for (int i = 0; i < vertices; i++)

if (visited[i] == 0 && dist[i] < min)

{

min = dist[i];

index = i;

}

return index;

}

void display(int dist[10], int vertices)

{

printf("Vertex\t Dist from source\n");

for (int i = 0; i < vertices; i++)

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

}

void Dijkstras(int graph[10][10], int source, int vertices)

{

int dist[10], visited[10];

int u;

for (int i = 0; i < vertices; i++)

{

dist[i] = 9999;

visited[i] = 0;

}

dist[source] = 0;

for (int i = 0; i < vertices - 1; i++)

{

u = findMinDist(dist, visited, vertices);

visited[u] = 1;

for (int i = 0; i < vertices; i++)

if (visited[i] != 1 && graph[u][i] && dist[u] != 9999 && dist[u] + graph[u][i] < dist[i])

dist[i] = dist[u] + graph[u][i];

}

display(dist, vertices);

}

int main()

{

int graph[10][10], numVertices, numEdges, i, j, u, v, weight, source;

printf("Enter number of vertices: ");

scanf("%d", &numVertices);

for (int i = 0; i < numVertices; i++)

for (int j = 0; j < numVertices; j++)

graph[i][j] = 999;

printf("Enter the number of edges: ");

scanf("%d", &numEdges);

for (int i = 0; i < numEdges; i++)

{

printf("Enter edge: ");

scanf("%d %d", &u, &v);

printf("Enter weight: ");

scanf("%d", &weight);

graph[u][v] = graph[v][u] = weight;

}

for (int k = 0; k < numVertices; k++)

{

for (int l = 0; l < numVertices; l++)

printf("%d \t", graph[k][l]);

printf("\n");

}

printf("Enter starting vertex: ");

scanf("%d", &source);

Dijkstras(graph, source, numVertices);

}

10b**)Implement Dijkstras Algorithm to generate MST using Adjacency List.**

#include <stdio.h>

#include <stdlib.h>

#define INFINITY 9999

struct Node {

int vertex;

int weight;

struct Node\* next;

};

struct Node\* a[10];

int findMinDist(int dist[10], int vset[10], int v) {

int min = INFINITY;

int index = -1;

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

if (vset[i] == 0 && dist[i] < min) {

min = dist[i];

index = i;

}

}

return index;

}

void display(int dist[10], int v) {

printf("Vertex\tDistance from source\n");

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

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

}

}

void Dijkstras(int v, int src) {

int dist[10], vset[10];

int u;

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

dist[i] = INFINITY;

vset[i] = 0;

}

dist[src] = 0;

for (int i = 0; i < v - 1; i++) {

u = findMinDist(dist, vset, v);

if (u == -1) {

break;

}

vset[u] = 1;

struct Node\* p = a[u];

while (p != NULL) {

int vert = p->vertex;

int weight = p->weight;

if (!vset[vert] && dist[u] + weight < dist[vert]) {

dist[vert] = dist[u] + weight;

}

p = p->next;

}

}

display(dist, v);

}

int main() {

struct Node\* p;

struct Node\* newV;

int u, v, vertex,e, i;

int weight, src;

printf("Enter the number of vertices: ");

scanf("%d", &vertex);

for (int i = 0; i < vertex; i++)

a[i] = NULL;

printf("Enter the number of edges: ");

scanf("%d",&e);

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

{

printf("Enter edge: ");

scanf("%d",&u);

scanf("%d",&v);

printf("Enter weight: ");

scanf("%d",&weight);

newV = (struct Node\*)malloc(sizeof(struct Node));

newV->vertex = v;

newV->weight = weight;

newV->next = NULL;

p = a[u];

if (p == NULL)

a[u] = newV;

else

{

while (p->next != NULL)

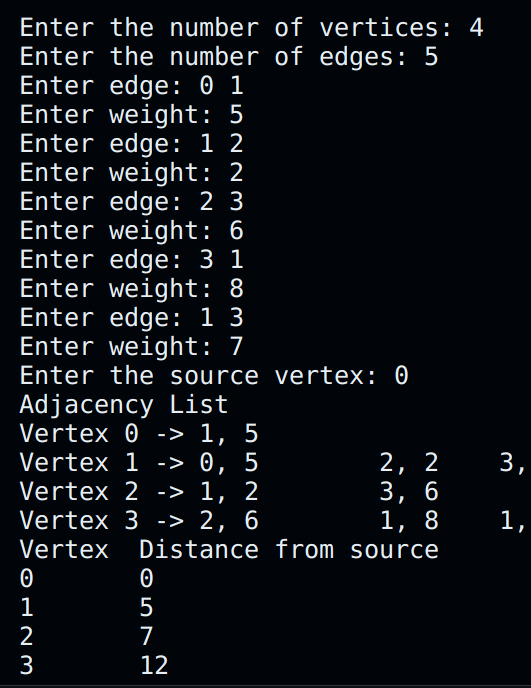
p = p->next;

p->next = newV;

}

newV = (struct Node\*)malloc(sizeof(struct Node));

newV->vertex = u;

newV->weight = weight;

newV->next = NULL;

p = a[v];

if (p == NULL)

a[v] = newV;

else

{

while (p->next != NULL)

p = p->next;

p->next = newV;

}

}

printf("Enter the source vertex: ");

scanf("%d", &src);

printf("Adjacency List\n");

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

printf("Vertex %d -> ", i);

p = a[i];

while (p != NULL) {

printf("%d, %d\t", p->vertex, p->weight);

p = p->next;

}

printf("\n");

}

Dijkstras(vertex, src);

return 0;

}