**ADA LAB WEEK 7**

**Zayd Ahmed**

**1BM21CS254**

**01/08/23**

**1) Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s and Kruskal’s algorithm.**

#include <stdio.h>

int parent[10];

int a[10][10];

int n;

int t[10][2];

void kruskals(int cost[10][10]){

int count=0;

int k=0;

int u,v;

int i,j,sum=0;

while(count!=n-1){

int min=999;

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

parent[i]=i;

}

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

for(int j=0;j<n;j++){

if(cost[i][j]<min && cost[i][j]!=0){

min=cost[i][j];

u=i;

v=j;

}

}

}

i=find(u);

j=find(v);

if(i!=j){

t[k][0]=u;

t[k][1]=v;

k++;

count++;

sum+=cost[i][j];

unionn(i,j);

}

cost[u][v]=cost[v][u]=999;

}

printf("the minimal spannning tree is:\n");

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

printf("%d->%d\n",t[i][0],t[i][1]);

}

printf("optimal solution: %d",sum);

}

int find(int a){

while(parent[a]!=a){

a=parent[a];

}

return a;

}

void unionn(int a,int b){

if(a<b){

parent[b]=a;

}

else{

parent[a]=b;

}

}

int main(){

printf("enter the number of vertices:\n");

scanf("%d",&n);

printf("enter the cost matrix:\n");

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

for(int j=0;j<n;j++){

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

}

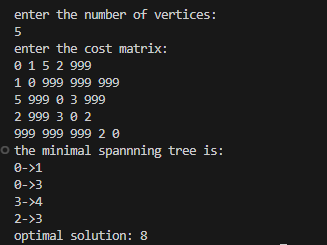
}

kruskals(a);

return 0;

}

**Output:**

****

**Prims Algorithm:**

#include <stdio.h>

#include <stdbool.h>

#define INF 999

#define V 5

int n;

int G[V][V];

int main()

{

int sum=0;

printf("enter the number of vertices:\n");

scanf("%d", &n);

printf("enter the cost matrix:\n");

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

{

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

{

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

}

}

int no\_edge;

int selected[V];

memset(selected, false, sizeof(selected));

no\_edge = 0;

selected[0] = true;

int x;

int y;

printf("Edge : Weight\n");

while (no\_edge < V - 1)

{

int min = INF;

x = 0;

y = 0;

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

{

if (selected[i])

{

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

{

if (!selected[j] && G[i][j])

{

if (min > G[i][j])

{

min = G[i][j];

x = i;

y = j;

}

}

}

}

}

printf("%d - %d : %d\n", x, y, G[x][y]);

sum+=G[x][y];

selected[y] = true;

no\_edge++;

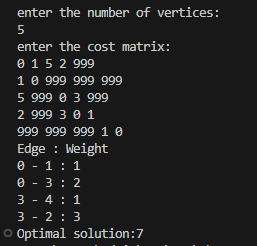
}

printf("Optimal solution:%d",sum);

return 0;

**}**

**OUTPUT:**

****

**2) From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm.**

#include <stdio.h>

#define INFINITY 999

#define MAX 10

void Dijkstra(int Graph[MAX][MAX], int n, int start);

void Dijkstra(int Graph[MAX][MAX], int n, int start) {

int cost[MAX][MAX], distance[MAX], pred[MAX];

int visited[MAX], count, mindistance, nextnode, i, j;

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

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

if (Graph[i][j] == 0)

cost[i][j] = INFINITY;

else

cost[i][j] = Graph[i][j];

for (i = 0; i < n; i++) {

distance[i] = cost[start][i];

pred[i] = start;

visited[i] = 0;

}

distance[start] = 0;

visited[start] = 1;

count = 1;

while (count < n - 1) {

mindistance = INFINITY;

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

if (distance[i] < mindistance && !visited[i]) {

mindistance = distance[i];

nextnode = i;

}

visited[nextnode] = 1;

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

if (!visited[i])

if (mindistance + cost[nextnode][i] < distance[i]) {

distance[i] = mindistance + cost[nextnode][i];

pred[i] = nextnode;

}

count++;

}

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

if (i != start) {

printf("\nDistance from source to %d: %d", i, distance[i]);

}

}

int main() {

int Graph[MAX][MAX], i, j, n, u;

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

scanf("%d",&n);

printf("Enter the adjacency matrix:\n");

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

{

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

{

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

}

}

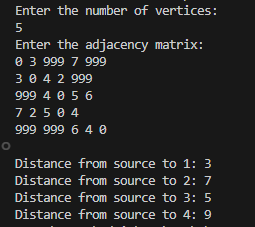
u = 0;

Dijkstra(Graph, n, u);

return 0;

}

**Output:**

****