Lab 4

1.Prims algorithm

#include <stdio.h>

#include <limits.h>

#include <stdbool.h>

#define MAX 100

int findMinKey(int key[], bool mstSet[], int n) {

int min = INT\_MAX, min\_index = -1;

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

if (!mstSet[v] && key[v] < min) {

min = key[v];

min\_index = v;

}

}

return min\_index;

}

void primMST(int graph[MAX][MAX], int n, int start) {

int parent[MAX];

int key[MAX];

bool mstSet[MAX];

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

key[i] = INT\_MAX;

mstSet[i] = false;

}

key[start] = 0;

parent[start] = -1;

for (int count = 0; count < n - 1; count++) {

int u = findMinKey(key, mstSet, n);

mstSet[u] = true;

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

if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {

parent[v] = u;

key[v] = graph[u][v];

}

}

}

int totalWeight = 0;

printf("\nMinimum Spanning Tree Edges:\n");

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

if (i != start && parent[i] != -1) {

printf("%d - %d (weight %d)\n", parent[i], i, graph[i][parent[i]]);

totalWeight += graph[i][parent[i]];

}

}

printf("Total weight of MST: %d\n", totalWeight);

}

int main() {

int n, m, start;

int graph[MAX][MAX] = {0};

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

scanf("%d %d", &n, &m);

printf("Enter %d edges (u v weight):\n", m);

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

int u, v, w;

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

graph[u][v] = w;

graph[v][u] = w;

}

printf("Enter the starting vertex (0 to %d): ", n - 1);

scanf("%d", &start);

if (start < 0 || start >= n) {

printf("Invalid start vertex!\n");

return 1;

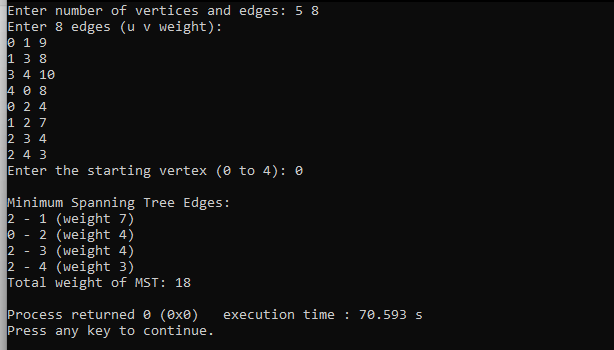
}

primMST(graph, n, start);

return 0;

}

Output:



2.Kruskals algorithm

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

#define INF 9999

int parent[MAX];

int find(int i) {

while (parent[i] != i)

i = parent[i];

return i;

}

void union\_set(int i, int j) {

int a = find(i);

int b = find(j);

parent[a] = b;

}

int main() {

int n;

int cost[MAX][MAX];

int i, j, a, b, u, v;

int ne = 0, mincost = 0, min;

printf("Enter number of vertices: ");

scanf("%d", &n);

printf("Enter the adjacency matrix (0 if no edge):\n");

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

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

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

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

cost[i][j] = INF;

}

}

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

parent[i] = i;

printf("Edge \tWeight\n");

while (ne < n - 1) {

min = INF;

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

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

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

min = cost[i][j];

a = u = i;

b = v = j;

}

}

}

union\_set(u, v);

printf("%d - %d\t%d\n", a, b, min);

mincost += min;

ne++;

}

printf("Minimum cost = %d\n", mincost);

return 0;

}

Output:

