Aim: write a program in c to find MST using Kruskal algorithm

Algorithm:

1. Sort the edges of the graph in ascending order of their weights.

2. Initialize an empty result set and an empty subset (disjoint sets).

3. Iterate through the sorted edges:

a. If adding the edge to the result set doesn't form a cycle, add it to the result set and update the disjoint sets.

b. If adding the edge forms a cycle, skip it.

4. Repeat the steps 3a and 3b until there are V-1 edges in the result set or all edges have been considered.

5. The result set is the minimum spanning tree.

Source Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 100

typedef *struct* edge {

*int* source, dest, weight;

} Edge;

typedef *struct* graph {

*int* V, E;

    Edge\* edge;

} Graph;

Graph\* createGraph(*int* V, *int* E) {

    Graph\* graph = (Graph\*) malloc(sizeof(Graph));

    graph->V = V;

    graph->E = E;

    graph->edge = (Edge\*) malloc(E \* sizeof(Edge));

    return graph;

}

*int* find(*int* parent[], *int* i) {

    if (parent[i] == -1)

        return i;

    return find(parent, parent[i]);

}

*void* union1(*int* parent[], *int* x, *int* y) {

*int* xset = find(parent, x);

*int* yset = find(parent, y);

    parent[xset] = yset;

}

*int* isCycle(Graph\* graph) {

*int* parent[MAX];

    memset(parent, -1, sizeof(parent));

    for (*int* i = 0; i < graph->E; i++) {

*int* x = find(parent, graph->edge[i].source);

*int* y = find(parent, graph->edge[i].dest);

        if (x == y)

            return 1;

        union1(parent, x, y);

    }

    return 0;

}

*int* cmp(const *void*\* a, const *void*\* b) {

    Edge\* a1 = (Edge\*)a;

    Edge\* b1 = (Edge\*)b;

    return a1->weight > b1->weight;

}

*void* KruskalMST(Graph\* graph) {

*int* V = graph->V;

    Edge result[V];

*int* e = 0;

*int* i = 0;

    qsort(graph->edge, graph->E, sizeof(graph->edge[0]), cmp);

*int* subset[MAX];

    memset(subset, -1, sizeof(subset));

    while (e < V - 1) {

        Edge next\_edge = graph->edge[i++];

*int* x = find(subset, next\_edge.source);

*int* y = find(subset, next\_edge.dest);

        if (x != y) {

            result[e++] = next\_edge;

            union1(subset, x, y);

        }

    }

    printf("Following are the edges in the constructed MST\n");

    for (i = 0; i < e; ++i)

        printf("%d - %d: %d\n", result[i].source, result[i].dest, result[i].weight);

    return;

}

*int* main() {

*int* V, E, i, s, d, w;

    printf("Enter number of vertices: ");

    scanf("%d", &V);

    printf("Enter number of edges: ");

    scanf("%d", &E);

    Graph\* graph = createGraph(V, E);

    printf("Enter source, destinationand weight for each edge:\n");

    for (i = 0; i < E; i++) {

        scanf("%d %d %d", &s, &d, &w);

        graph->edge[i].source = s;

        graph->edge[i].dest = d;

        graph->edge[i].weight = w;

    }

    KruskalMST(graph);

    return 0;

}

Output:

Enter number of vertices: 4

Enter number of edges: 5

Enter source, destinationand weight for each edge:

0 1 10

0 2 6

0 3 5

1 3 15

2 3 4

Following are the edges in the constructed MST

2 - 3: 4

0 - 3: 5

0 - 1: 1