#include <stdio.h>

#include <stdlib.h>

#define MAX 100

// Structure to represent an edge

struct Edge {

int src, dest, weight;

};

// Compare function for qsort (to sort edges by weight)

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

return ((struct Edge\*)a)->weight - ((struct Edge\*)b)->weight;

}

// Find parent of a node (with path compression)

int findParent(int parent[], int i) {

if (parent[i] != i)

parent[i] = findParent(parent, parent[i]);

return parent[i];

}

// Union of two sets

void unionSets(int parent[], int rank[], int x, int y) {

int xroot = findParent(parent, x);

int yroot = findParent(parent, y);

if (rank[xroot] < rank[yroot])

parent[xroot] = yroot;

else if (rank[xroot] > rank[yroot])

parent[yroot] = xroot;

else {

parent[yroot] = xroot;

rank[xroot]++;

}

}

void kruskal(struct Edge edges[], int V, int E) {

struct Edge result[MAX]; // Store result

int parent[MAX], rank[MAX];

int i, e = 0; // e is number of edges in result

// Sort edges by weight

qsort(edges, E, sizeof(struct Edge), compareEdges);

// Initialize disjoint sets

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

parent[i] = i;

rank[i] = 0;

}

i = 0; // Index for sorted edges

while (e < V - 1 && i < E) {

struct Edge nextEdge = edges[i++];

int x = findParent(parent, nextEdge.src);

int y = findParent(parent, nextEdge.dest);

// If including this edge doesn't cause cycle, include it

if (x != y) {

result[e++] = nextEdge;

unionSets(parent, rank, x, y);

}

}

// Print the MST

printf("Edges in the Minimum Spanning Tree:\n");

int totalWeight = 0;

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

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

totalWeight += result[i].weight;

}

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

}

int main() {

int V, E;

struct Edge edges[MAX];

printf("Enter number of vertices: ");

scanf("%d", &V);

printf("Enter number of edges: ");

scanf("%d", &E);

printf("Enter each edge (src dest weight):\n");

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

scanf("%d %d %d", &edges[i].src, &edges[i].dest, &edges[i].weight);

}

kruskal(edges, V, E);

return 0;

}

A screenshot of a computer program

AI-generated content may be incorrect.