29.Minimum Spanning Tree Kruskal Algorithm

#include <stdio.h>

#include <stdlib.h>

#define V 6

typedef struct edge {

int u;

int v;

int weight;

} edge;

edge edges[V];

int parent[V];

int rank[V];

// Function prototype for the compare function

int compare(const void \*a, const void \*b);

void init() {

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

parent[i] = i;

rank[i] = 0;

}

}

int find(int x) {

if (parent[x] == x) {

return x;

} else {

return parent[x] = find(parent[x]);

}

}

void unionSets(int x, int y) {

int x\_parent = find(x);

int y\_parent = find(y);

if (rank[x\_parent] < rank[y\_parent]) {

parent[x\_parent] = y\_parent;

} else if (rank[x\_parent] > rank[y\_parent]) {

parent[y\_parent] = x\_parent;

} else {

parent[x\_parent] = y\_parent;

rank[y\_parent]++;

}

}

void kruskal() {

edge mst[V - 1];

int mst\_size = 0;

int n\_edges;

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

scanf("%d", &n\_edges);

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

int u, v, weight;

printf("Enter the starting vertex of edge %d: ", i + 1);

scanf("%d", &u);

printf("Enter the ending vertex of edge %d: ", i + 1);

scanf("%d", &v);

printf("Enter the weight of edge %d: ", i + 1);

scanf("%d", &weight);

edges[i] = (edge) { u, v, weight };

}

qsort(edges, n\_edges, sizeof(edges[0]), compare);

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

int u = edges[i].u;

int v = edges[i].v;

int weight = edges[i].weight;

if (find(u) != find(v)) {

mst[mst\_size++] = edges[i];

unionSets(u, v);

}

}

for (int i = 0; i < mst\_size; i++) {

printf("(%d, %d, %d)\n", mst[i].u, mst[i].v, mst[i].weight);

}

}

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

edge \*edge1 = (edge \*) a;

edge \*edge2 = (edge \*) b;

return edge1->weight - edge2->weight;

}

int main() {

init();

kruskal();

return 0;

}

OUTPUT

