#include <stdio.h>

#include <stdlib.h>

struct Edge {

int src, dest, weight;

};

struct Graph {

int V, E;

struct Edge\* edge;

};

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

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

graph->V = V;

graph->E = E;

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

return graph;

}

int find(int parent[], int i);

void Union(int parent[], int rank[], int x, int y);

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

void KruskalMST(struct Graph\* graph);

void KruskalMST(struct Graph\* graph) {

int V = graph->V;

struct Edge result[V];

int e = 0;

int i = 0;

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

int \*parent = (int\*)malloc(V \* sizeof(int));

int \*rank = (int\*)malloc(V \* sizeof(int));

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

parent[v] = v;

rank[v] = 0;

}

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

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

int x = find(parent, next\_edge.src);

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

if (x != y) {

result[e++] = next\_edge;

Union(parent, rank, 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].src, result[i].dest, result[i].weight);

}

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

struct Edge\* a1 = (struct Edge\*)a;

struct Edge\* b1 = (struct Edge\*)b;

return a1->weight > b1->weight;

}

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

if (parent[i] == i)

return i;

return find(parent, parent[i]);

}

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

int xroot = find(parent, x);

int yroot = find(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]++;

}

}

int main() {

int V = 4;

int E = 5;

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

graph->edge[0].src = 0;

graph->edge[0].dest = 1;

graph->edge[0].weight = 10;

graph->edge[1].src = 0;

graph->edge[1].dest = 2;

graph->edge[1].weight = 6;

graph->edge[2].src = 0;

graph->edge[2].dest = 3;

graph->edge[2].weight = 5;

graph->edge[3].src = 1;

graph->edge[3].dest = 3;

graph->edge[3].weight = 15;

graph->edge[4].src = 2;

graph->edge[4].dest = 3;

graph->edge[4].weight = 4;

KruskalMST(graph);

return 0;

}

A screenshot of a computer

Description automatically generated