```c
/**
 *   TRABALHO DE GRAFOS - ENGENHARIA DE COMPUTAÇÃO
 *
 *   Implementação do Algoritmo de Kruskal
 *
 *   EQUIPE:
 *       - Cristiano Oliveira
 *       - Gilmaicon Leandro
 *       - Leonardo de Holanda Costa
 *       - Lucas Maia
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>


#define vertex int
// variáveis necessarias para o union-find (valores arbitrários)
#define MAXN 50500
#define MAXM 200200
//A estrutura edge representa a floresta geradora
typedef struct {
    vertex v, w;
    int cst;
} edge;

// 'n' número de vértices e 'm' número de arestas
int n, m;

//Adjacências
edge graph_edges[MAXM];

// Para o union find
int parents[MAXN];//Estutura de chefes fazendo com que cada vértice seja o seu próprio chefe
int weight[MAXN];//Estrutura de pesos

// Árvore geradora de custo mínimo
edge mst[MAXM];

//Comparador usado no sort
int comparator(edge *v, edge *w){
    //Função auxiliar usada para organizar as arestas em ordem crescente de custos
    if ( v->cst < w->cst ){
        return -1;
    }
    else if (v->cst > w->cst ){
        return 1;
    }
    else {return 0;}
}


// funções do union find
int find(int x){
    //Devolve o chefe da componente conexa
    if(parents[x] == x) {
        return x;
    }
    return parents[x] = find(parents[x]);
}


void join(int v, int w){
    //Faz a união das componentes cujos chefes são v e w respectivamente
    v = find(v);
    w = find(w);

    if(weight[v] < weight[w]) {
        parents[v] = w;
```

```
    }
    else if(weight[w] < weight[v]) {
        parents[w] = v;
    }
    else{
        parents[v] = w;
        weight[w]++;
    }

}


//A função kruskal é responsável pelo crescimento de uma floresta geradora até que ela se torne co
void kruskal(FILE *file){
    int i,v,w,cst;
    int cst_total=0;
    int size = 0;

    //ler os dados do arquivo e popula a estrutura graph_edges com sua lista de adjacências
    fscanf(file,"%d %d",&n, &m);
    for(i = 0;i <= m;i++){
        fscanf(file, "%d %d %d", &v, &w , &cst);
        graph_edges[i].v = v;
        graph_edges[i].w = w;
        graph_edges[i].cst = cst;
    }
    fclose(file); //Fecha o arquivo

    // inicializar os pais para o union-find
    for(i = 0;i <= n-1;i++) {
        parents[i] = i;
    }
    // ordenar as arestas
    qsort((void *) &graph_edges, m, sizeof(edge), (const void *) comparator);
    //Compondo os subconjuntos da estrutura do union-find
    for(i = 0;i <= m-1;i++){
        if(find(graph_edges[i].v) != find(graph_edges[i].w) ){
            // se estiverem em componentes distintas
            join(graph_edges[i].v, graph_edges[i].w);
            mst[size] = graph_edges[i];
            size++;
        }

    }

    //Resultado Final
    printf("Arvore geradora de custo minimo: ");
    for(i = 0;i < n-1;i++){
        printf("(");
        printf("%d",mst[i].v);
        printf(",");
        printf("%d",mst[i].w);
        printf(")");
        cst_total += mst[i].cst;
    }
    printf("\nCusto: %d",cst_total);
}

FILE *read_file(){
    FILE *file;
    char *nomeArquivo;
    nomeArquivo = malloc(sizeof(char)*35); /* Aloca string com até 35 caracteres (valor arbitrário

    if(nomeArquivo == NULL){
        printf("[x] Falha! Não há memória disponível no sistema");
        free(nomeArquivo);
        return NULL;
    }

    printf("Digite o nome do arquivo de texto (SEM SUA EXTENSÃO): ");
    scanf(" %s", nomeArquivo);   //Lê o nome do arquivo;
    strcat(nomeArquivo,".txt\0"); //Adiciona a extensão '.txt' mais caractere '\0' (final da strin
```

```
144     file = fopen(nomeArquivo,"r"); //Abre arquivo no modo de leitura;
145
146     if(file == NULL){ /* Arquivo não encontrado */
147         free(nomeArquivo);
148         printf("\n\n[!] Arquivo não localizado");
149         printf("\n[i] Verifique se o arquivo existe ou se você digitou o nome corretamente");
150         printf("\n\n[i] Pressione <ENTER> para voltar...");
151         setbuf(stdin,NULL);
152         getchar();
153
154         return NULL;
155     }
156     else {
157         printf("Arquivo: %s\t\n", nomeArquivo);
158         return file;
159     }
160 }
161
162 int main() {
163     FILE *file = read_file();
164     kruskal(file);
165 }
```