10

Problema 25

PAREDES SÁNCHEZ, MICHAEL STEVEN (TAIS74)

ID envio	Usuario/a	Hora envío	Veredicto
59529	TAIS74	2021-10-19 10:33	AC
59519	TAIS74	2021-10-19 10:31	AC
59455	TAIS74	2021-10-19 10:08	WA
59451	TAIS74	2021-10-19 10:07	WA

Fichero Source.cpp

```
*
* Nombre y Apellidos: IVAN GONZALEZ / TAIS50
*                     MICHAEL PAREDES / TAIS74
*


 Primero hacemos un grafovalorado y una priorityqueue de aristas(ordenadas de menor a mayor)
    luego hacemos el algoritmo de Kruskal con ese grafo
 hacemos que se unan los conjuntos solo si el valor de hacer la carretera es menor al de
    construir un aeropuerto. en el punto en el que las carreteras son mas
 caras que los aeropuertos paramos(o si ya estan unidas todas las ciudades numconjuntos ==1);
 Tambien miramos los conjuntos que nos quedan ya que son los aeropuertos que hay que poner.

 la priority queue(añadir los elementos) tiene un coste de A log A siendo A las aristas.
 el algoritmo de kruskal tiene un coste A log A siendo A las aristas.
 Poner todas las arista en el grafo valorado es de A siendo A las aristas.
 Coste del problema es de A log A siendo A las aristas


/*bool operator> (const Arista<int> a, const Arista<int> b) {
    return a.valor() > b.valor();
}*/

pair<int, int> resuelve(GrafoValorado<int> const& g, priority_queue<Arista<int>, vector<
    Arista<int>>, greater<Arista<int>>> pq, int A) {
    ConjuntosDisjuntos cjtos(g.V());
    int coste = 0;
    while (!pq.empty()) {
        auto a = pq.top(); pq.pop();
        int v = a.uno(), w = a.otro(v);
        if (!cjtos.unidos(v, w)&& a.valor()<A) {
            cjtos.unir(v, w);
            coste += a.valor();
        }
        if (a.valor() >= A)break;
        if (cjtos.num_cjtos() == 1)break;
    }
    coste += cjtos.num_cjtos() * A;
    return {coste, cjtos.num_cjtos()};
}
```

```
bool resuelveCaso() {

    // leer los datos de la entrada
    int N, M, A;
    cin >> N >> M >> A;
    if (!std::cin)  // fin de la entrada
        return false;


    // resolver el caso posiblemente llamando a otras funciones
    int aux1, aux2, coste;
    GrafoValorado<int> grafo(N);
    priority_queue<Arista<int>, vector<Arista<int>>, greater<Arista<int>>> pq;
    for (int i = 0; i < M; i++)
    {
        cin >> aux1 >> aux2 >> coste;
        Arista<int> arista(aux1-1, aux2-1, coste);
        grafo.ponArista(arista);
        pq.push(arista);
    }
    // escribir la solución
    pair<int, int> sol = resuelve(grafo, pq, A);
    cout << sol.first << "␣" << sol.second << '\n';
    return true;
}
```