9

Problema 25

OTEGUI MARÍN, JOSÉ (TAIS73)

ID envio	Usuario/a	Hora envío	Veredicto
59403	TAIS73	2021-10-19 09:48	AC
59387	TAIS73	2021-10-19 09:38	AC

Fichero prob-25.cpp

```
*
* Indicad el nombre completo y usuario del juez de quienes habéis hecho esta solución:
* Estudiante 1: Alberto Chaves, TAIS20
* Estudiante 2: Jose Otegui, TAIS73
*

Al leer los datos se crea una cola de prioridad en las que se almacena cada posible carretera
     con su origen y destino orenadas de menor a mayor coste de construcción.
La resolución comienza en el peor caso posible, en el que se construya un aeropuerto para
    cada localidad.
A partir de ahí, se extrae la carretera con menor coste, si el coste de construirla es menor
    que el de mantener dos
aeropuertos en lugar de solo uno, ambas regiones se consideran ahora unidas, se descuenta el
    coste de un aeropuerto y se incrementa con el coste de la carretera.
Cuando no quedan carreteras que evaluar, muestra el coste total y el número de conjuntos
    disjuntos que han quedado, que coresponden con el número de aeropuertos.

El coste del algoritmo sería de O(N M) ya que la operacion Unir de hace N veces.

struct Carretera {
  int origen;
  int fin;
  int coste;
  //si redefino el operador menor puedo usarlo como comparador de la cola de prioridad
  bool operator<(const Carretera& otro) const {
    return coste < otro.coste;
  }
};

void resolver(int localidades, PriorityQueue<Carretera>& cola, int costeAeropuertos) {

  int total = 0; //el coste de las carreteras
  int aeropuertos = localidades; //cuantos aeropuertos se necesitan, realmente coincide con
    el número de conjuntos
  ConjuntosDisjuntos conjuntos(localidades); //al principio, se considera 1 aeropuerto por
    localidad, esto es, que no hacen falta careteras

  while (!cola.empty()) {
    Carretera menor = cola.top();
    cola.pop();
    //solo si el coste de comunicar las dos localidades que une la carretera mediante la
    misma es menor que mantener el aeropuerto
```

```
    //y además las dos regiones no estaban unidas previamente, se unen y se incrementa el
    coste de carreteras
    if (menor.coste < costeAeropuertos && !conjuntos.unidos(menor.origen, menor.fin)) {
      conjuntos.unir(menor.origen, menor.fin);
      aeropuertos--;
      total += menor.coste;
    }
  }

  cout << (total + aeropuertos * costeAeropuertos) << "␣" << conjuntos.num_cjtos() << endl;
}

bool resuelveCaso() {

  // leemos la entrada
  int N, M, A;
  cin >> N >> M >> A;
  if (!cin)
    return false;

  // leer el resto del caso y resolverlo

  PriorityQueue<Carretera> cola;
  for (int i = 0; i < M; i++) {//se leen todas las carreteras posibles
    Carretera aux;
    cin >> aux.origen >> aux.fin >> aux.coste;
    aux.origen = aux.origen - 1;
    aux.fin = aux.fin - 1;
    cola.push(aux);
  }

  resolver(N, cola, A);

  return true;
}
```