9

Problema 16

GONZÁLEZ RAMIRO, IVÁN (TAIS50)

ID envio	Usuario/a	Hora envío	Veredicto
57786	TAIS50	2022-10-05 10:23	AC
57769	TAIS50	2022-10-05 10:15	AC
57743	TAIS50	2022-10-05 09:57	AC
57737	TAIS50	2022-10-05 09:53	TLE

Fichero Source.cpp

```
 *
 * Indicad el nombre completo y usuario del juez de quienes habéis hecho esta solución:
 * Estudiante 1:IVÁN GONZÁLEZ RAMIRO TAIS 50
 * Estudiante 2:MICHAEL PAREDES SANCHEZ TAIS 74
 *
 guardamos las conexiones en un grafo.h
 Tenemos un priorityqueue ordenados por precios mas bajos, tambien tenemos un vector para
    poder acceder mas facilmente a las personas(con su precio y vertice).
 Vamos haciendo dfs a las personas que vamos sacando del priorityqueue y sumamos su precio a
    la variable precio, comprobando que no hayan sido vistas antes con el array de bool
    visitados.
 Con la variable global de vistos comprobamos que ya hayamos visto a todas las personas y asi
     ahorrar tiempo.

 Guardar los elementos en el grafo N+M en el vector es de N mientras que en el priorityqueue
    es de N log N. Siendo N el numero de vertices y M el de aristas.

 En el bucle donde se calcula el precio sacar el top de la priorityqueue es constante
 y el dfs recorre solo una vez cada vertice gracias al array de visitados haciendo que sea N+
    M, siendo N el numero de vertices y M el de aristas
 Coste total NlogN + (N+M) siendo N el numero de vertices y M el de aristas
struct gente {
    int precio;
    int vertice;
};
bool operator<(gente a, gente b) {
    return a.precio > b.precio || (a.precio == b.precio && a.vertice<b.vertice);
}
bool visitados[25002];
int vistos=0;

void dfs(gente g, const Grafo &grafo,const vector<gente> &v) {
    vistos++;
    visitados[g.vertice] = true;
    for (int w:grafo.ady(g.vertice)) {
        if (!visitados[w]) dfs(v[w],grafo,v);
    }
}
bool resuelveCaso() {
```

no es necesario (sobrecoste)

No sigue el patrón de diseño

```
    // leemos la entrada
    int N, M;
    cin >> N >> M;

    if (!cin)
        return false;
    int precio;
    priority_queue<gente> aldeanos;
    vector<gente> personas;
    vistos = 0;
    for (int i = 0; i < N; i++) {
        cin >> precio;
        aldeanos.push({ precio,i });
        personas.push_back({ precio,i });
        visitados[i] = false;
    }
    Grafo grafo(N);
    int aux1, aux2;
    for (int i = 0; i < M; i++) {
        cin >> aux1 >> aux2;
        grafo.ponArista(aux1-1, aux2-1);
    }
    precio=0;
    while (vistos < grafo.V()) {
        gente aux=aldeanos.top();
        aldeanos.pop();
        if (!visitados[aux.vertice]) {
            precio += aux.precio;
            dfs(aux, grafo, personas);
        }

    }
    cout << precio << "\n";
    // leer el resto del caso y resolverlo



    return true;
}
```