

Problema de laboratorio (Semana 4)

Peaje a la sombra

AC

Enunciado

AC

Todos

7

Nota

📄 4-L.cpp

FALTA explicación de la solución y del coste

class CaminosDFS en realidad es BFS

```
{
protected:
    vector<bool> visit; // visit[v] = ¿hay camino de s a v?
    vector<int> ant; // ant[v] = último vértice antes de llegar a v
```

Si no hace falta recuperar el camino es mejor que elimines el atributo ant.

```
    vector<int> dist;
    int s; // vertice origen

    void dfs(Grafo const& g)
    {
        queue<int> q;
        dist[s] = 0;
        visit[s] = true;
        q.push(s);
        while (!q.empty())
        {
            const int v = q.front(); q.pop();
            for (int w : g.ady(v))
            {
                if (!visit[w]) {
                    ant[w] = v;
                    dist[w] = dist[v] + 1;
                    visit[w] = true;
                    q.push(w);
                }
            }
        }
    }
}
```

```
public:
    CaminosDFS(Grafo const& g, int s) : visit(g.V(), false), ant(g.V()), s(s),
    dist(g.V(), g.V())
    {
        dfs(g);
    }
```

```
/// Caminos
// ¿hay camino del origen a v?
```

```

bool hayCamino(int v) const
{
    return visit[v];
}

using Camino = deque<int>; // para representar caminos
// devuelve un camino desde el origen a v (debe existir)
Camino camino(int v) const

```

se puede copiar código de las diapositivas, pero si copias de más, cosas que no hacen falta para este problema, no demuestras que sabes bien lo que haces

```

{
    if (!hayCamino(v))
        throw domain_error("No existe camino.");

    Camino cam;
    // recuperamos el camino retrocediendo
    for (int x = v; x != s; x = ant[x])
        cam.push_front(x);
    cam.push_front(s);
    return cam;
}

/// Problema
int distancia(int v) const
{
    return dist[v];
}
};

bool resuelveCaso() {
    int N, C, A, L, T;
    cin >> N >> C >> A >> L >> T;
    int minCoste = C;

    if (!cin)
        return false;

    Grafo g(N);
    while (C--)
    {
        int v, w;
        cin >> v >> w;
        g.ponArista(v - 1, w - 1);
    }

    const CaminosDFS a(g, A - 1);
    const CaminosDFS l(g, L - 1);
    const CaminosDFS t(g, T - 1);

    for (int i = 0; i < g.V(); i++)
    {
        if (a.hayCamino(i) && l.hayCamino(i) && t.hayCamino(i))
            minCoste = min(minCoste, a.distancia(i) + l.distancia(i) + t.distancia(i));
    }

    cout << minCoste << endl;

    return true;
}

```

