

Problema 29

SOTILLO SOTILLO, CLARA (TAIS96)

ID envio	Usuario/a	Hora envío	Veredicto
60324	TAIS96	2022-10-26 10:18	AC
60311	TAIS96	2022-10-26 10:10	AC
60273	TAIS96	2022-10-26 09:56	AC
60244	TAIS96	2022-10-26 09:43	TLE

Fichero Source.cpp

*
 * Indicad el nombre completo y usuario del juez de quienes habéis hecho esta solución:
 * Estudiante 1: Carlos Gómez TAIS45
 * Estudiante 2: Clara Sotillo TAIS96
 *

El coste de la función en el caso peor es de $O(P \log N)$ siendo P el número de aristas del grafo y N número de vértices.
 Este coste se debe al uso del algoritmo de Dijkstra el cual calcula los caminos mínimos desde el origen al resto de vértices en un tiempo $O(P \log N)$ y con un espacio adicional $O(N)$

Resolvemos el problema usando un grafo valorado invirtiendo las aristas proporcionadas, ya que queremos que se recorran los caminos de la salida al resto de vértices.
 Usamos Dijkstra para hallar los caminos mínimos desde la salida a cada uno de los vértices y luego comprobamos que esa distancia es menor igual al T proporcionado en cada caso

```
class laberinto {
public:
    laberinto(DigrafoValorado<int> const& d, int o) : dist(d.V(), INF), pq(d.V()), origen(o) {
        Dijkstra(d);
    }

    bool hayCamino(int s) const {
        return dist[s] != INF;
    }

    int distancia(int s) const {
        return dist[s];
    }


private:
```

```

const int INF = std::numeric_limits<int>::max();
int origen;
vector<int> dist;
IndexPQ<int> pq;
int cont;
void relajar(AristaDirigida<int> a) {
    int v = a.desde(), w = a.hasta();
    if (dist[w] > dist[v] + a.valor()) {
        dist[w] = dist[v] + a.valor();
        pq.update(w, dist[w]);
    }
}

void Dijkstra(DigrafoValorado<int> const& d) {
    dist[origen] = 0;
    pq.push(origen, 0);
    while (!pq.empty()) {
        int v = pq.top().elem; pq.pop();
        for (auto a : d.ady(v))
            relajar(a);
    }
}
};

```



```

bool resuelveCaso() {

    // leemos la entrada
    int N, S, T, P;
    cin >> N >> S >> T >> P;

    if (!cin)
        return false;

    DigrafoValorado<int> d(N);


    int a, b, val, cont=0;

    for (int i = 0; i < P; i++) {
        cin >> a >> b >> val;
        d.ponArista({ b - 1, a - 1, val });
    }

    laberinto l(d, S - 1);

    for (int i = 0; i < N; i++) {
        if (i != S - 1) {
            if (l.hayCamino(i) && l.distancia(i) <= T) {
                cont++;
            }
        }
    }
}

```



```
    }  
}  
  
cout << cont << "\n";  
  
return true;  
}
```