

Estructuras de Datos no Lineales

2.2. Caminos de coste mínimo

José Fidel Argudo Argudo
José Antonio Alonso de la Huerta
M^a Teresa García Horcajadas



Versión 1.0

Caminos de coste mínimo

Algoritmo de Dijkstra

```
template <typename tCoste>
vector<tCoste> Dijkstra(const GrafoP<tCoste>& G,
                       typename GrafoP<tCoste>::vertice origen,
                       vector<typename GrafoP<tCoste>::vertice>& P);
```

Calcula los caminos de coste mínimo entre **origen** y todos los vértices del grafo **G**.

Salida:

- Un **vector de costes mínimos** de tamaño **G.numVert()**.
- **P**, un **vector de vértices** de tamaño **G.numVert()**, tal que **P[i]** es el vértice anterior a **i** en el camino de coste mínimo desde **origen** hasta **i**.

```

// Suma de costes
template <typename tCoste>
tCoste suma(tCoste x, tCoste y)
{
    const tCoste INFINITO = GrafoP<tCoste>::INFINITO;

    if (x == INFINITO || y == INFINITO)
        return INFINITO;
    else
        return x + y;
}

template <typename tCoste>
vector<tCoste> Dijkstra(const GrafoP<tCoste>& G,
                      typename GrafoP<tCoste>::vertice origen,
                      vector<typename GrafoP<tCoste>::vertice>& P)
{
    typedef typename GrafoP<tCoste>::vertice vertice;
    vertice v, w;
    const size_t n = G.numVert();
    vector<bool> S(n, false);           // Conjunto de vértices vacío.
    vector<tCoste> D;                   // Costes mínimos desde origen.

    // Iniciar D y P con caminos directos desde el vértice origen.
    D = G[origen];
    D[origen] = 0;                       // Coste origen-origen es 0.
    P = vector<vertice>(n, origen);

```

```

// Calcular caminos de coste mínimo hasta cada vértice.
S[origen] = true; // Incluir vértice origen en S.
for (size_t i = 1; i <= n-2; i++)
{
    // Localizar vértice w no incluido en S con menor coste desde origen
    tCoste costeMin = GrafoP<tCoste>::INFINITO;
    for (v = 0; v <= n-1; v++)
        if (!S[v] && D[v] <= costeMin)
        {
            costeMin = D[v];
            w = v;
        }
    S[w] = true; // Incluir vértice w en S.
    // Recalcular coste hasta cada v no incluido en S, a través de w.
    for (v = 0; v <= n-1; v++)
        if (!S[v])
        {
            tCoste Owv = suma(D[w], G[w][v]);
            if (Owv < D[v])
            {
                D[v] = Owv;
                P[v] = w;
            }
        }
    }
return D;
}

```

```

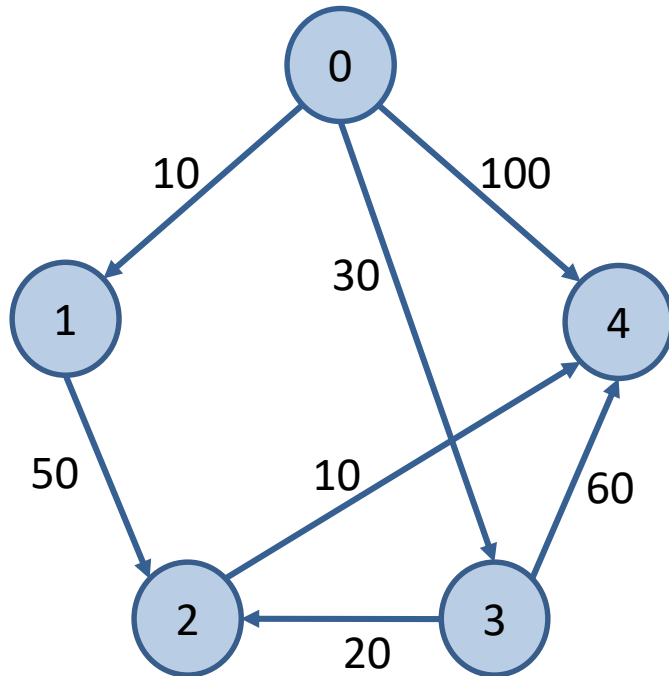
#include "listaenla.h"
template <typename T> class GrafoP {
public:
    typedef Lista<vertice> tCamino;
    // ...
};

template <typename tCoste> typename GrafoP<tCoste>::tCamino
camino(typename GrafoP<tCoste>::vertice orig,
        typename GrafoP<tCoste>::vertice v,
        const vector<typename GrafoP<tCoste>::vertice>& P)
// Devuelve el camino de orig a v a partir de un vector
// P obtenido mediante la función Dijkstra().
{
    typename GrafoP<tCoste>::tCamino C;

    C.insertar(v, C.primer());
    do {
        C.insertar(P[v], C.primer());
        v = P[v];
    } while (v != orig);
    return C;
}

```

Ejemplo de Dijkstra



Matriz de Costes

	0	1	2	3	4
0	∞	10	∞	30	100
1	∞	∞	50	∞	∞
2	∞	∞	∞	∞	10
3	∞	∞	20	∞	60
4	∞	∞	∞	∞	∞

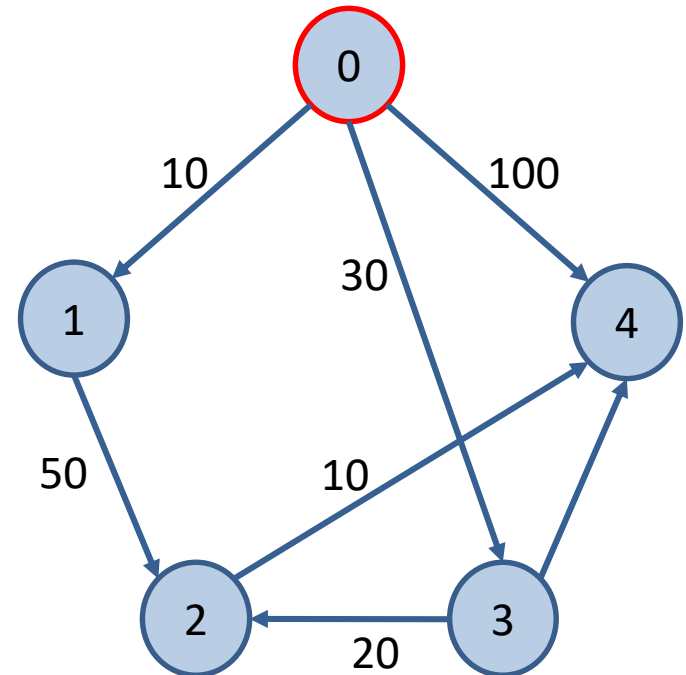
numVert = 5

Inicializamos tomando como origen el vértice 0

	0	1	2	3	4
S	T	F	F	F	F

	0	1	2	3	4
D	0	10	∞	30	100

	0	1	2	3	4
P	0	0	0	0	0



Hacemos las iteraciones

1ª) $i = 1$

$\text{Min } \{D[1], D[2], D[3], D[4]\} = \text{Min } \{10, \infty, 30, 100\} = 10$

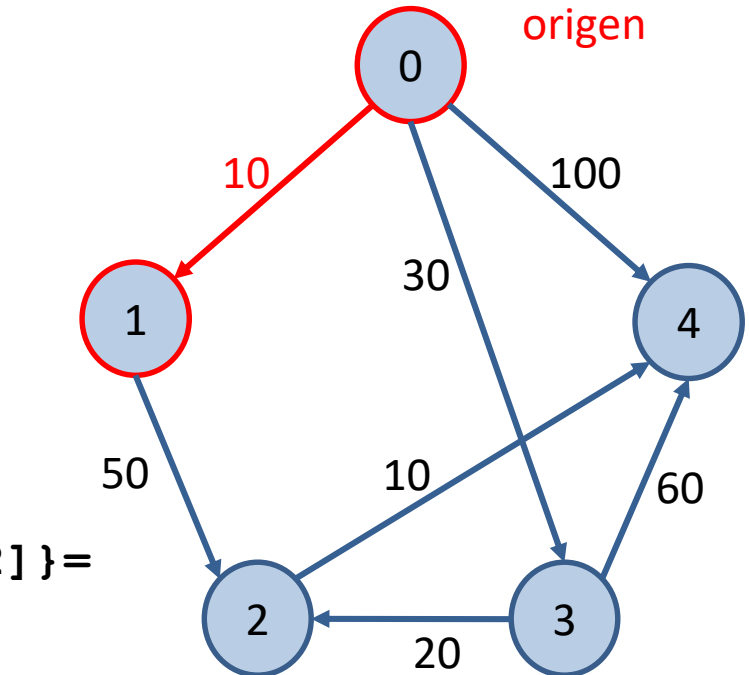
$w = 1$

	0	1	2	3	4
S	T	T	F	F	F

Recalculamos los costes a través
del nuevo vértice

$D[2] = \min\{D[2], D[1] + G \rightarrow \text{Costes}[1, 2]\} =$
 $\min\{\infty, 10 + 50\} = 60$

	0	1	2	3	4
D	0	10	60	30	100



	0	1	2	3	4
P	0	0	1	0	0

$$D[3] = \min\{D[3], D[1] + G \rightarrow \text{Costes}[1,3]\} = \min\{30, 10 + \infty\} = 30$$

D y P siguen igual.

$$D[4] = \min\{D[4], D[1] + G \rightarrow \text{Costes}[1,4]\} = \min\{100, 10 + \infty\} = 100$$

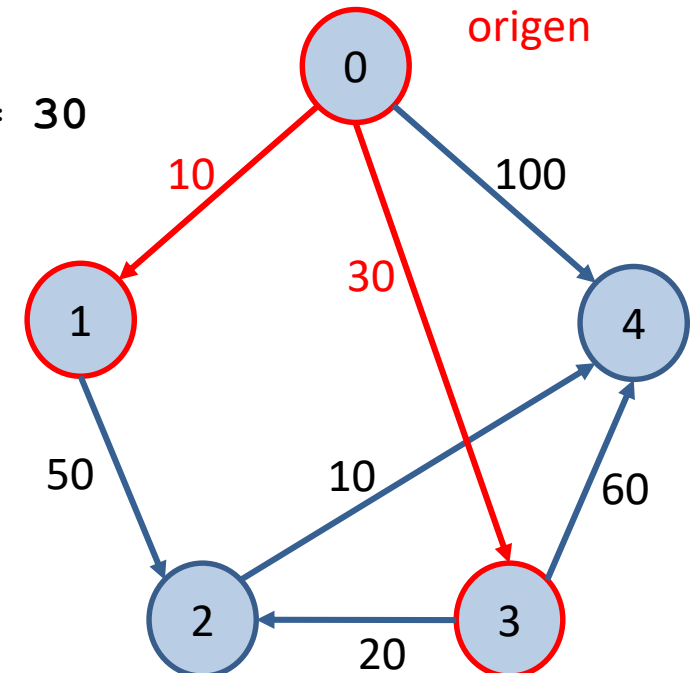
D y P siguen igual

2ª) $i = 2$

$$\text{Min } \{D[2], D[3], D[4]\} = \text{Min } \{60, 30, 100\} = 30$$

w = 3

	0	1	2	3	4
S	T	T	F	T	F



Recalculamos las distancias a través del nuevo vértice

$$D[2] = \min\{D[2], D[3] + G \rightarrow \text{Costes}[3, 2]\} = \min\{60, 30 + 20\} = 50$$

	0	1	2	3	4
D	0	10	50	30	100
	0	1	2	3	4
P	0	0	3	0	0

$$D[4] = \min\{D[4], D[3] + G \rightarrow \text{Costes}[3, 4]\} = \min\{100, 30 + 60\} = 90$$

	0	1	2	3	4
D	0	10	50	30	90
	0	1	2	3	4
P	0	0	3	0	3

3ª) $i = 3$

$\text{Min } \{D[2], D[4]\} = \text{Min } \{50, 90\} = 50$

$w = 2$

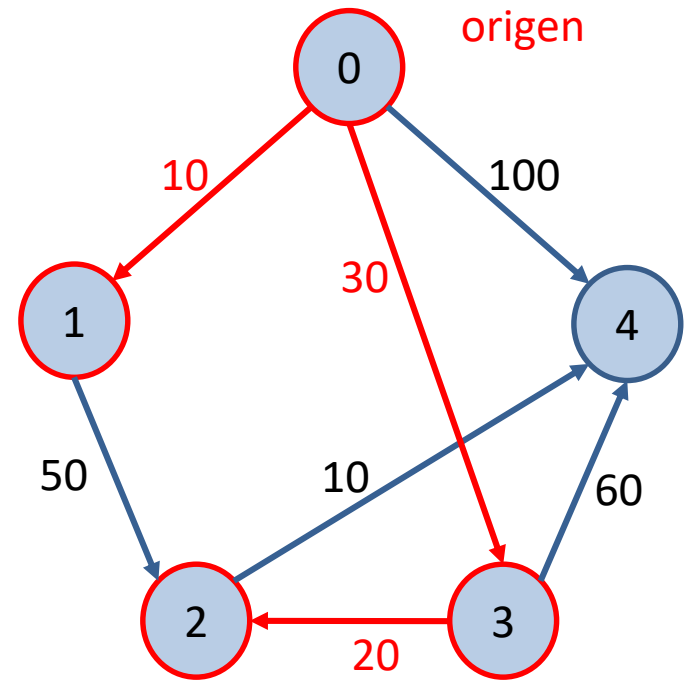
	0	1	2	3	4
S	T	T	T	T	F

**Recalculamos las distancias a través
del nuevo vértice**

$D[4] = \min\{D[4], D[2] + G \rightarrow \text{Costes}[2, 4]\} =$
 $\min\{90, 50 + 10\} = 60$

	0	1	2	3	4
D	0	10	50	30	60

	0	1	2	3	4
P	0	0	3	0	2



Resultado

	0	1	2	3	4
D	0	10	50	30	60

	0	1	2	3	4
P	0	0	3	0	2

