

Tema 8. Caminos Mínimos.

8.1 Notación y Formalización

8.2 Existencia de Caminos Mínimos

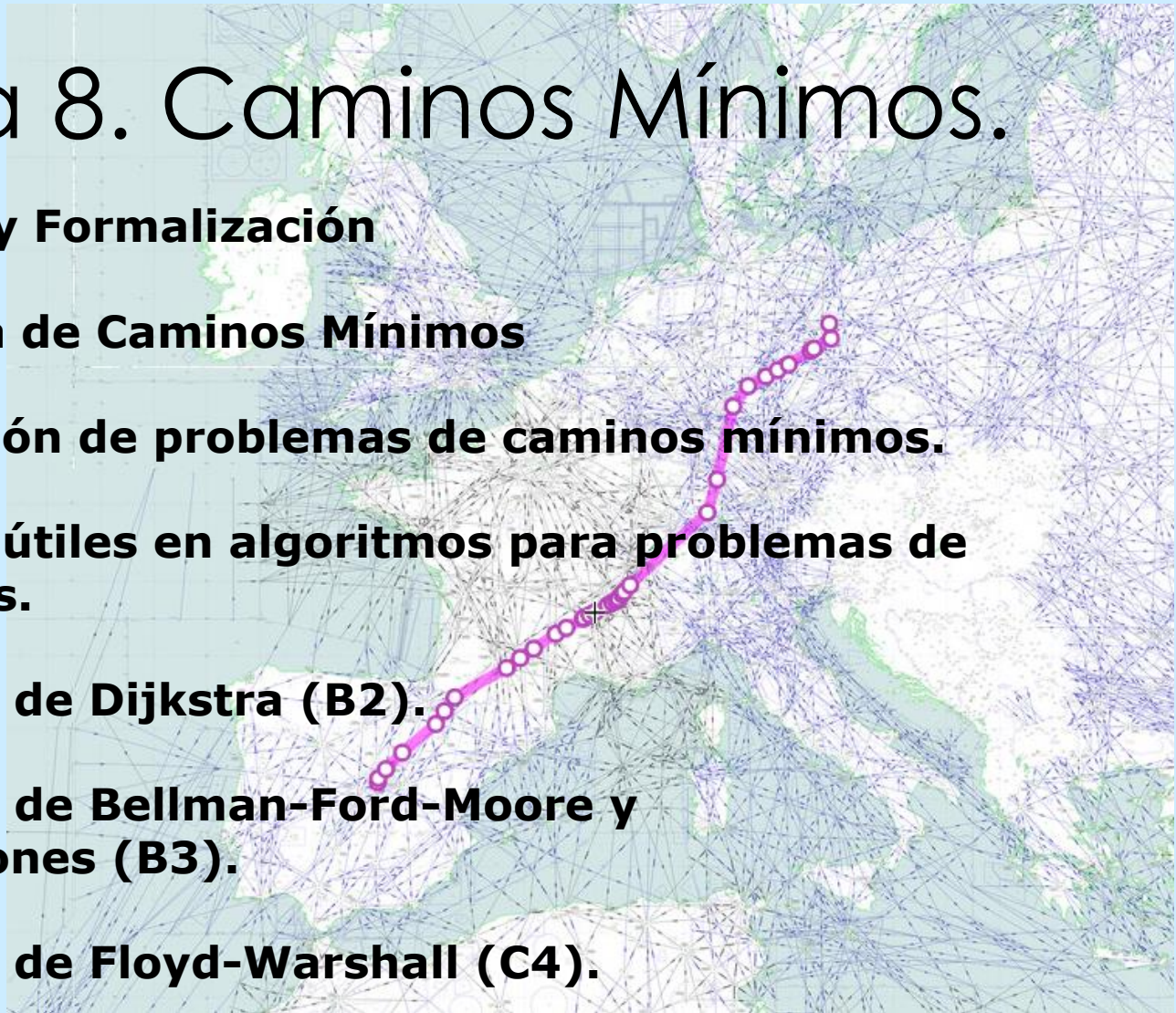
8.3 Clasificación de problemas de caminos mínimos.

8.4 Etiquetas útiles en algoritmos para problemas de uno a todos.

8.5 Algoritmo de Dijkstra (B2).

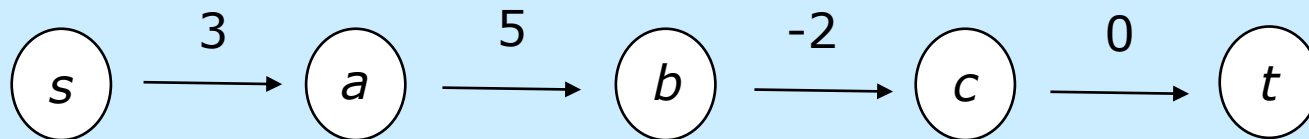
8.6 Algoritmo de Bellman-Ford-Moore y modificaciones (B3).

8.7 Algoritmo de Floyd-Warshall (C4).



8.1 Notación y Formalización

- Sea $G=(V, A)$ un Grafo dirigido,
- Todo arco (i, j) tiene asociado un coste real c_{ij} ,
- Un nodo origen distinguido s y un nodo destino t .
- Un camino P de s a t es una secuencia de arcos dirigidos de s a t .
- Definimos por \mathbf{P} el conjunto de todos los caminos de s a t en G .
- La longitud/coste de un camino $c(P)$ coincide con la suma de los costes de los arcos en el camino, es decir: $c(P) = \sum_{(i,j) \in P} c_{ij}$

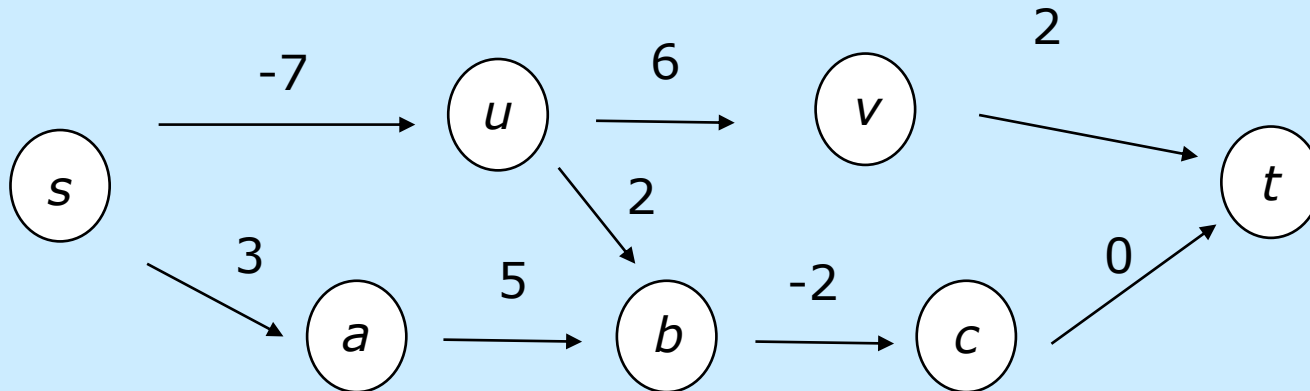


$$c(P) = 3 + 5 - 2 + 0 = 6$$

8.1 Notación y Formalización

Planteamiento: El problema del camino mínimo de s a t consiste en determinar el camino P en \mathbf{P} con menor coste, es decir,

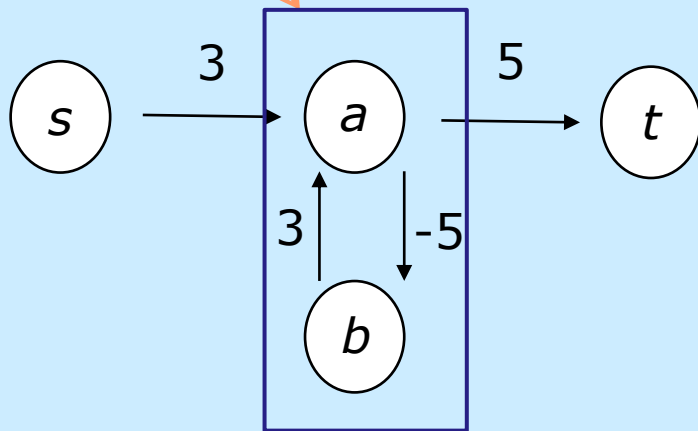
$$\min_{P \in \mathbf{P}} c(P) = \min_{P \in \mathbf{P}} \sum_{(i,j) \in P} c_{ij}$$



8.2 Existencia de camino mínimo

Nuestro problema tendrá solución óptima si es factible y acotado. Esto se traduce en las siguientes 2 condiciones de existencia:

- 1. Existe un camino P de s a t en G (factibilidad).
- 2. No existe ningún camino P de \mathbf{P} que contenga un circuito de coste negativo (acotación).



$$P = s \rightarrow a \rightarrow t, \quad c(P) = 8$$

$$W = a \rightarrow b \rightarrow a, \quad c(W) = -2$$

$$P' = P + W = s \rightarrow a \rightarrow b \rightarrow a \rightarrow t, \quad c(P') = 6$$

La solución óptima consiste en dar $+\infty$ vueltas en W (o equivalentemente $x_{sa}=1$, $x_{ab}=+\infty$, $x_{ba}=+\infty$ y $x_{at}=1$).

En esta situación decimos que no existe camino mínimo.

Luego si existe camino mínimo, buscamos sólo aquellos que sean elementales (sin repetir nodos).

8.3 Clasificación de Problemas de Caminos Mínimos

Si atendemos a las **características del Grafo**:

- **1.** Todos los costes iguales y positivos, es decir, $c_{ij} = k > 0 \quad \forall (i, j) \in A$.
- **2.** Todos los costes no-negativos, es decir, $c_{ij} \geq 0 \quad \forall (i, j) \in A$.
- **3.** Costes arbitrarios (reales), pero G sin circuitos de coste negativo.
- **4.** Costes y G arbitrarios (podría contener circuitos de longitud negativa)

Si atendemos al **tipo de problema**:

- **A.** Camino mínimo entre un nodo origen s y nodo destino t (uno-a-uno). Aquí hay que calcular 1.
- **B.** Caminos mínimos entre un nodo origen s y el resto $V - \{s\}$ (uno-a-todos). Aquí hay que calcular $n-1$.
- **C.** Caminos mínimos entre todos los pares de nodos (todos-a-todos). Aquí hay que calcular $n(n-1)$.

8.3 Clasificación de Problemas de Caminos Mínimos

Notar que resolver **A** puede implicar tener que resolver **B**, dado que el camino mínimo entre s y t podría pasar por todos los nodos. Así que nos centraremos en los problemas **B** y **C** con las características 1-4. Más específicamente en los siguientes:

- **Problema B1.** Determinar los caminos mínimos de uno-a-todos con costes iguales positivos. Notar que aquí lo que hay que minimizar es el número de arcos. Ya hemos visto un algoritmo que alcanza a cada nodo (si es posible) usando el menor número de arcos y es un **recorrido en Amplitud**. La complejidad es $O(m)$. Lo damos aquí por visto.
- **Problema B2.** Determinar los caminos mínimos de uno-a-todos con costes no negativos. El algoritmo del-estado-del-arte es el **algoritmo de Dijkstra** que con una estructura de datos adecuada corre $O(m + n \log n)$.

8.3 Clasificación de Problemas de Caminos Mínimos

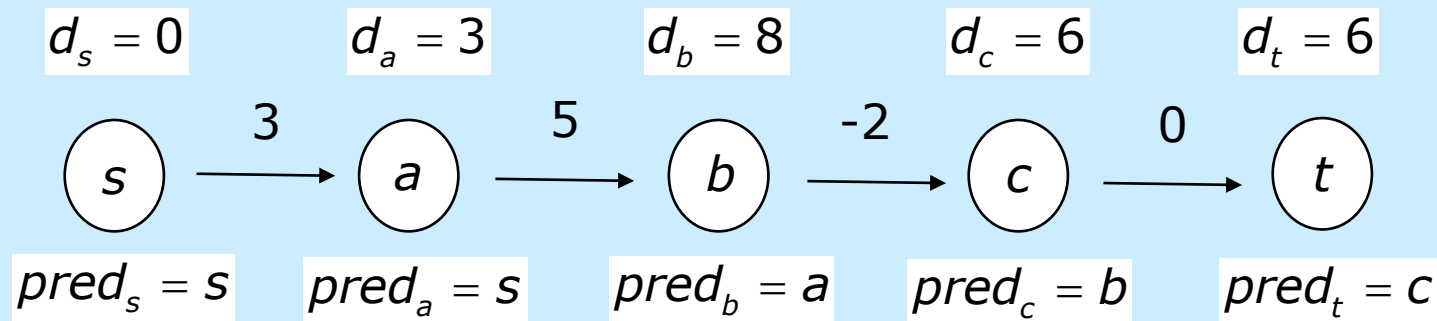
- **Problema B3.** Caminos mínimos de uno-a-todos donde los costes arbitrarios (reales), pero sabemos que G no contiene circuitos de coste negativo. En este caso podemos acelerar en la práctica el **algoritmo de Bellman-Ford-Moore**. Este algoritmo necesita un tiempo $O(nm)$.
- **Problema C4.** Caminos mínimos entre todos los pares de nodos donde los costes y G son arbitrarios. Este es el problema más general y se resuelve (entre otros) con el **algoritmo de Floyd y Warshall**. La complejidad de este algoritmo es $O(n^3)$.

Notar que las complejidades de los algoritmos aumenta al aumentar la dificultad del problema. No podemos matar moscas a cañonazos, debemos usar el algoritmo más eficiente para cada situación. En otro caso sólo veríamos el algoritmo de Floyd-Warshall.

8.4 Etiquetas útiles en algoritmos para problemas de uno a todos

Dado un camino P de s a t se usan dos etiquetas para cada nodo:

- **Etiqueta distancia** $d_i \forall i \in V$ que coincide con el coste del camino de s a i . Al final del correspondiente algoritmo deben ser óptimas, es decir, coincidir con el coste del camino mínimo de s a i .



- **Etiqueta predecesor** $pred_i \forall i \in V$ que almacena el nodo anterior al nodo i en el camino de s a i . Por convenio el nodo anterior a s es s , es decir, $pred_s = s$

8.4 Etiquetas útiles en algoritmos para problemas de uno a to

La Etiqueta predecesor permite identificar un camino de s a t de manera inversa empezando en el nodo destino t y aplicando el operador $pred$ hasta alcanzar al nodo s . Por ejemplo usando el procedimiento **MostrarCamino** siguiente:

```
Procedimiento MostrarCamino( $s, t, pred$ ) {  
     $i = t$ ;  
    Mientras ( $i \neq s$ ) hacer{  
        Imprimir " $i \leftarrow$ ";  
         $i = pred_i$ ;  
    }  
    Imprimir " $s$ ";  
}
```

El resultado para el anterior camino es: $t \leftarrow c \leftarrow b \leftarrow a \leftarrow s$
cuando llamamos a *MostrarCamino*($s, t, pred$)

8.4 Etiquetas útiles en algoritmos para problemas de uno a to

Si queremos imprimir el camino directo (no al revés), lo más sencillo es usar el siguiente procedimiento recursivo:

```
Procedimiento MostrarCamino_(s, i, &pred) {  
    Si (i ≠ s) entonces {  
        MostrarCamino_(s, pred[i], pred);  
        Imprimir "-> i";  
    }  
  
    En otro caso  
        Imprimir "s";  
}
```

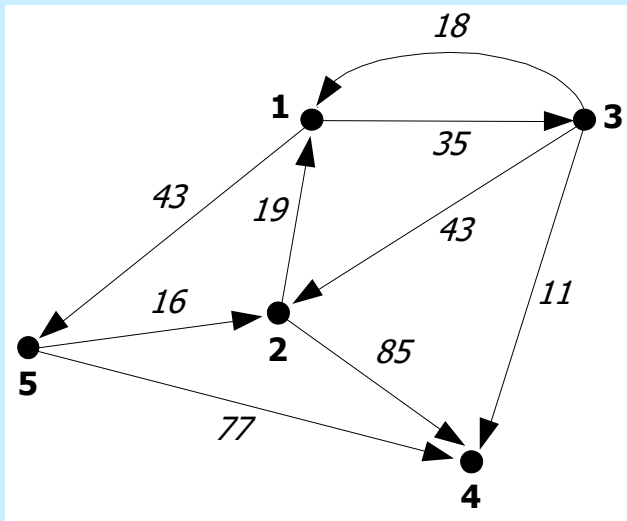
El resultado para el anterior camino es: *s*-> *a*-> *b*-> *c*-> *t* cuando llamamos a *MostrarCamino_*(*s*, *t*, *pred*).

Note que aquí *pred* se pasa por referencia con el objetivo de que no se hagan copias de todo el vector *pred* en la pila de recursión en cada una de las llamadas recursivas.

8.5 Algoritmo de Dijkstra (B2).

Problema B2. Determinar los caminos mínimos de uno-a-todos con costes no negativos. El algoritmo del-estado-del-arte es el **algoritmo de Dijkstra**.

Dado el siguiente grafo queremos calcular los caminos mínimos desde $s=1$ al resto de nodos.



El algoritmo de Dijkstra comienza inicializando las etiquetas distancia y de predecesores de la forma siguiente

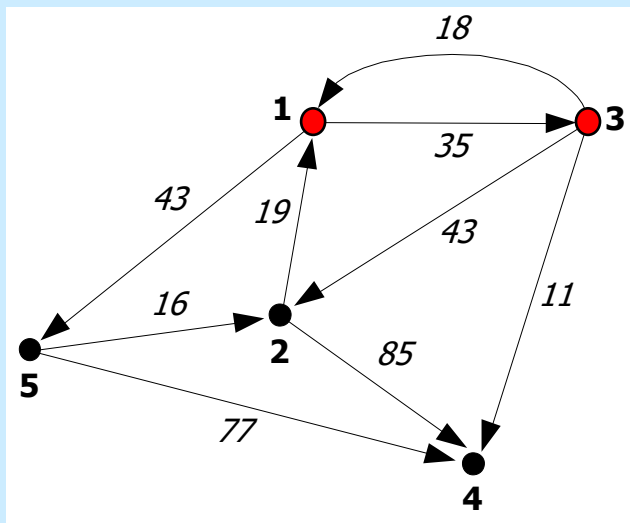
i	1	2	3	4	5
d_i	0	∞	∞	∞	∞
$pred_i$	1	0	0	0	0

Un valor de d_i igual a ∞ ($pred_i$ igual a 0) indica que no se conoce camino de s a i .

En cada iteración toma una decisión: selecciona el nodo con menor etiqueta distancia indicando que es óptima y permanente.

8.5 Algoritmo de Dijkstra (B2).

La primera vez la toma sobre el nodo $s=1$, dado que d_1 es 0 y no hay costes negativos, por tanto óptima.



i	1	2	3	4	5
d_i	0	∞	∞	∞	∞
$pred_i$	1	0	0	0	0

A continuación examina los nodos sucesores del nodo 1 para ver si podemos encontrar un camino mejor que el actual. En este caso, para el nodo 3 y el nodo 5.

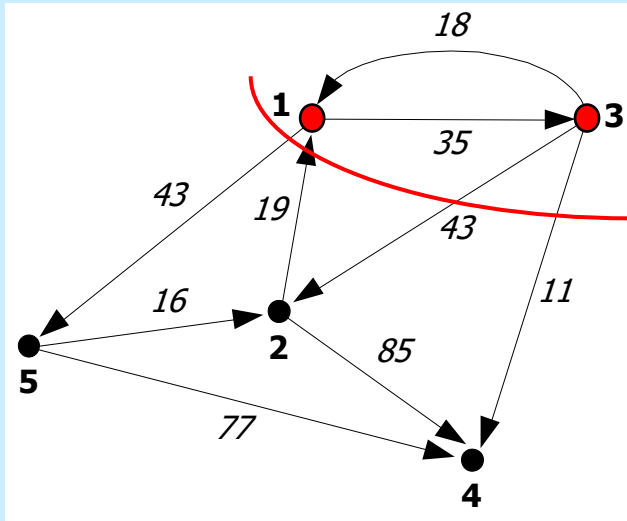
$d_3 > d_1 + 35$? Si, dado que $\infty > 35 \rightarrow d_3 = d_1 + 35 = 35$ y $pred_3=1$. Así, obtenemos:

i	1	2	3	4	5
d_i	0	∞	35	∞	43
$pred_i$	1	0	1	0	1

Terminada está iteración, **comienza la siguiente seleccionando la etiqueta distancia del nodo 3 (por ser la menor) como óptima (permanente)**

8.5 Algoritmo de Dijkstra (B2).

¿qué observamos?



En toda iteración el algoritmo de Dijkstra particiona el conjunto de nodos V en dos subconjuntos:

S el conjunto de nodos con etiqueta óptima que se denomina conjunto de los nodos con etiqueta permanente (no se puede cambiar ya que la decisión fue tomada) y su conjunto complementario **$V - S$** .

Desde que la selección se realiza sobre los nodos en $V-S$, es decir, se decide sobre el nodo con menor etiqueta distancia en $V-S$, es este el conjunto que debe almacenarse.

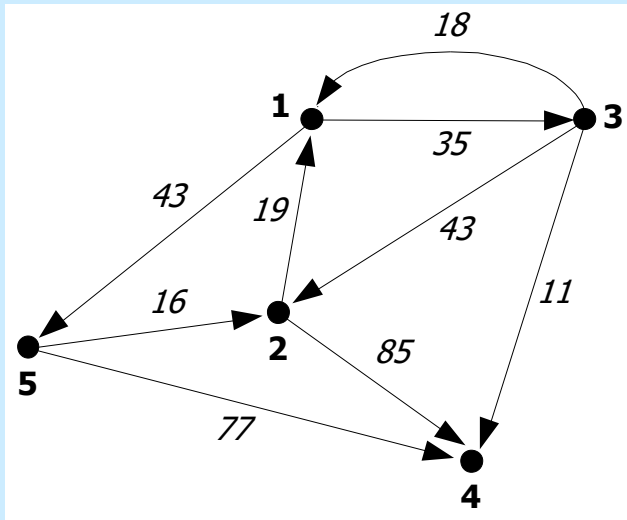
Usamos una estructura genérica denominada aquí **Almacén donde los nodos con etiqueta no permanente son almacenados.**

Así ya estamos en condiciones de presentar el siguiente pseudocódigo:

8.5 Algoritmo de Dijkstra (B2). Pseudocódigo.

```
Algoritmo Dijkstra( $G$ ,  $s$ , var  $d$ , Var  $pred$ ) {  
    //inicialización  
     $Almacén = \emptyset$ ;  
    Para todo  $i = 1$  hasta  $n$  {  
         $d_i = +\infty$ ;  $pred_i = 0$ ;  $Almacén += \{i\}$ ;  
    }  
     $d_s = 0$ ;  $pred_s = s$ ;  
    //bucle  
    Mientras ( $Almacén \neq \emptyset$ ) Hacer {  
        Extraer del  $Almacén$  el nodo  $i$  con menor  $d_i$ ;  
        //aquí  $i$  tiene distancia óptima y permanente  
        //A continuación propagamos etiqueta de  $i$   
        Para todo nodo  $j$  sucesor del nodo  $i$  Hacer  
            Si ( $d_j > d_i + c_{ij}$ ) y ( $j$  en  $Almacén$ ) entonces {  
                 $d_j = d_i + c_{ij}$ ;  $pred_j = i$ ;  
            }  
        }  
    }  
}
```

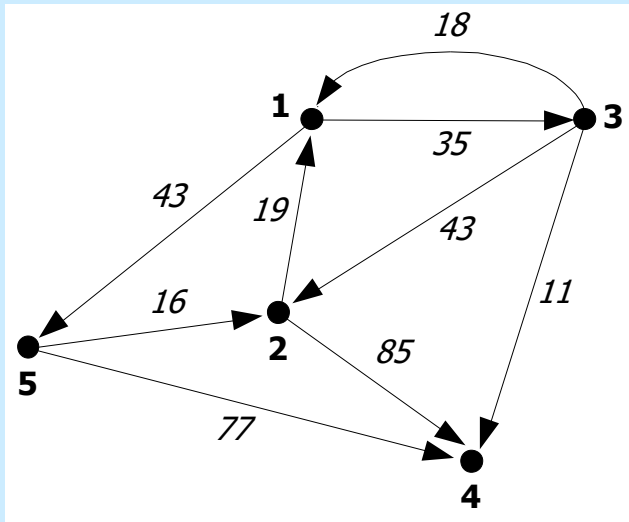
8.5 Algoritmo de Dijkstra (B2). Traza.



La traza del algoritmo de Dijkstra es una **tabla** donde la primera columna (iter), refleja la iteración; la segunda columna el estado del almacén; la tercera columna el nodo i extraído del almacén y luego hay tantas columnas como nodos que reflejan el par (d, pred) . Cada fila refleja la ejecución entera de una iteración. **Se señalan en rojos las etiquetas permanentes y en negrita cuando mejoran.** El origen es $s=1$.

Iter	Almacén	i	d, pred				
			1	2	3	4	5
Ini	1, 2, 3, 4, 5		0, 1	$+\infty, 0$	$+\infty, 0$	$+\infty, 0$	$+\infty, 0$
1	2, 3, 4, 5	1	0, 1	$+\infty, 0$	35, 1	$+\infty, 0$	43, 1
2	2, 4, 5	3	0, 1	78, 3	35, 1	46, 3	43, 1
3	2, 4	5	0, 1	59, 5	35, 1	46, 3	43, 1
4	2	4	0, 1	59, 5	35, 1	46, 3	43, 1
5		2	0, 1	59, 5	35, 1	46, 3	43, 1


8.5 Algoritmo de Dijkstra (B2). Traza.



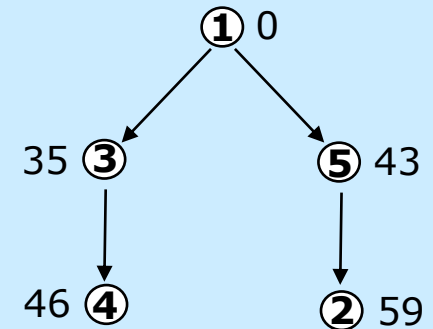
Por tanto, los caminos mínimos de $s=1$ al resto son identificados por las etiquetas d y $pred$ de la última iteración.

$d, pred$				
1	2	3	4	5
0, 1	59, 5	35, 1	46, 3	43, 1

Si quisiéramos identificar todos los caminos podríamos llamar al procedimiento MostrarCamino con t variando de 2 a 4.

Alternativamente podemos construir **el árbol de caminos mínimos** resultante que consiste en un grafo que contiene todos los arcos $(pred_i, i)$ para todo i en $V - \{s\}$ y todos los nodos, donde s es la raíz:  Al lado de los nodos aparecen las etiquetas distancias óptimas.

El único camino en el árbol desde s a un nodo i es el camino óptimo.

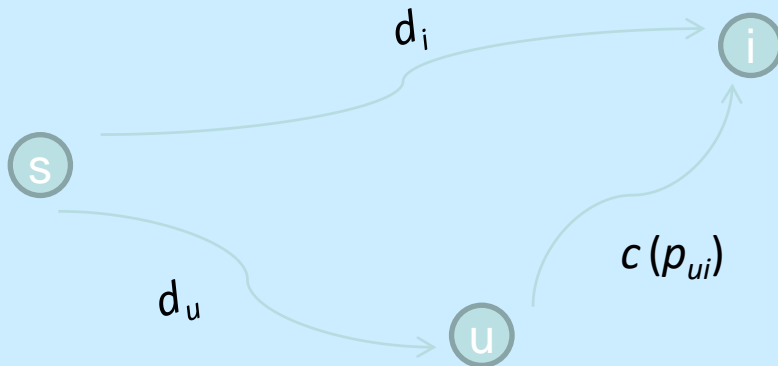


8.5 Algoritmo de Dijkstra (B2). Exactitud.

¿Por qué funciona Dijkstra? Hay que demostrar que cada decisión que toma es correcta, es decir, cada vez que extrae un nodo del almacén su etiqueta distancia es óptima.

Lema. El nodo extraído del almacén en cada iteración tiene etiqueta óptima.

Demostración: Se demuestra por inducción. Por ejemplo, $d_s=0$ es óptima desde que no hay costes negativos. Suponemos cierto para la iteración k y probamos para la siguiente. Supongamos que en la iteración $k+1$ se extrae el nodo i del almacén. Si d_i no es óptima, debe existir un camino alternativo \mathbf{p} de menor coste (ver figura). Este camino debe pasar por al menos un nodo u no extraído del almacén, es decir, un camino aún no explorado.



El coste del camino alternativo \mathbf{p} es $c(\mathbf{p}) = d_u + c(p_{ui})$

Sabemos que:

- 1) Como u está en el Almacén, $d_i \leq d_u$.
- 2) $c(p_{ui}) \geq 0$

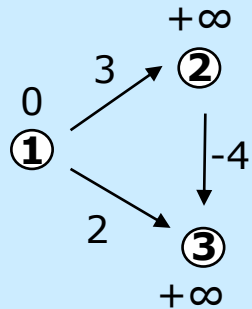
Por tanto, $c(\mathbf{p}) = d_u + c(p_{ui}) \geq d_i + 0 = d_i$.

Luego, d_i es óptima.

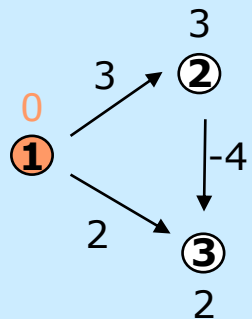
Sea \mathbf{u} es el primer nodo en el camino alternativo \mathbf{p} que está en Almacén, pero su nodo sucesor en \mathbf{p} no está en el almacén.

8.5 Algoritmo de Dijkstra (B2). Exactitud.

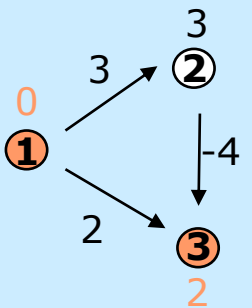
¿Por qué no funciona Dijkstra con costes negativos? Esto lo vemos con la ejecución de Dijkstra en el siguiente ejemplo.



Iter	Almacen	i	d, pred		
			1	2	3
Ini	1, 2, 3		0, 1	$+\infty$, 0	$+\infty$, 0



1	2, 3	1	0, 1	3, 1	2, 1
---	------	---	------	------	------



2	2	3	0, 1	3, 1	2, 1
---	---	---	------	------	------

Aquí **se toma una decisión errónea**, asegurando que $d_3=2$ es óptima.

Notar que en la siguiente iteración, cuando se decida sobre el nodo 2, no se podrá cambiar la distancia del nodo 3 ya que 3 no está en el *Almacén*.

8.5 Algoritmo de Dijkstra (B2). Final.

¿qué ocurre con Dijkstra cuando hay nodos no accesibles desde el de partida? En ese caso, quedan en el Almacén con su etiqueta distancia como fue inicializada, esto es, $+\infty$. Modifiquemos la parada para que en ese caso, finalice.

```
Algoritmo Dijkstra( $G$ ,  $s$ , var  $d$ , Var  $pred$ ) {  
    //inicialización  
    Almacén =  $V$ ;  
    Para todo  $i = 1$  hasta  $n$  { $d_i = +\infty$ ;  $pred_i = \_$ ;}  
     $d_s = 0$ ;  $pred_s = s$ ;  
    //bucle  
    Mientras (Almacén tenga nodos con etiqueta distancia  
finita) hacer {  
        Extraer del Almacén el nodo  $i$  con menor  $d_i$  finita  
        //aquí  $i$  tiene distancia óptima y permanente  
        //A continuación propagamos etiqueta de  $i$   
        Para todo nodo  $j$  sucesor del nodo  $i$  Hacer  
            Si ( $d_j > d_i + c_{ij}$ ) y ( $j$  en Almacén) entonces {  
                 $d_j = d_i + c_{ij}$ ;  $pred_j = i$ ;  
            }  
        }  
    }  
}
```