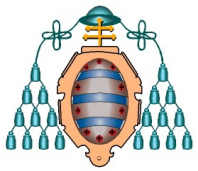


Departamento de Informática
UNIVERSIDAD DE OVIEDO

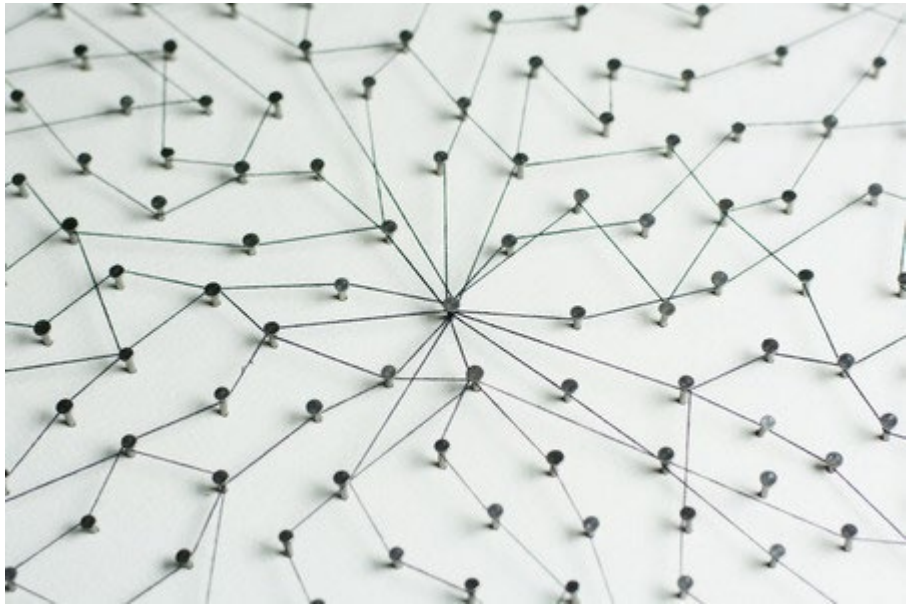
ESTRUCTURAS EN RED

Estructura de Datos

2023-2024

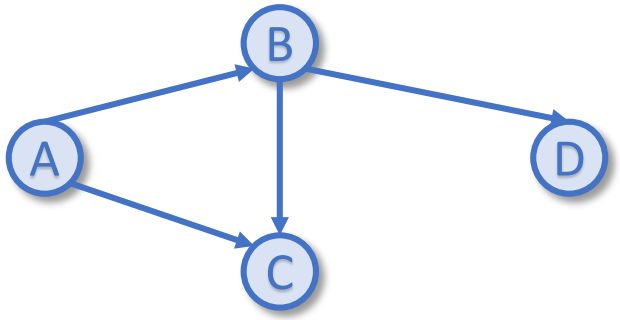
Objetivo

- Modelar relaciones entre objetos
  - Redes sociales
  - Redes de comunicaciones
  - Redes de transporte



Grafo

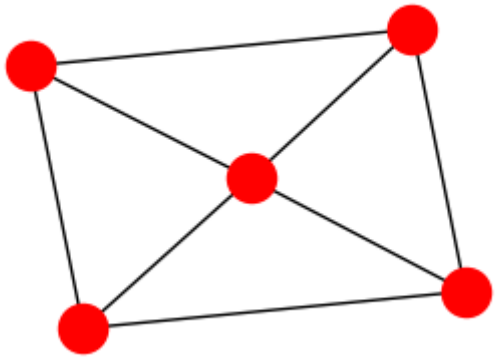
- Permite representar relaciones entre objetos
- Desde el punto de vista formal
  - Conjunto finito de vértices o nodos → V
    - V = {V_1, v_2, v3, …., V_n}
  - Conjunto finito de ejes → E
    - Pares de elementos (v,w) pertenecientes a V que representan las relaciones entre v y w
    - E = {(v_1, v_2), (v_4, v_3),….)
- Ejemplo



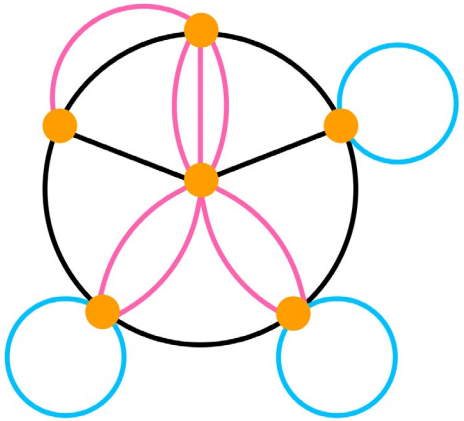
V = {A,B, C, D}
E = {(A,C), (B,A), (B,C), (B,D)}

Tipos de grafos

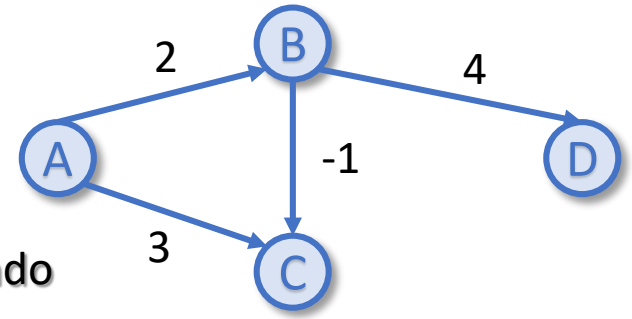
- Grafo simple → aceptan una única arista entre dos vértices
- Multigrafo → aceptan mas de un eje entres dos vértices
- Grafo dirigido → pares (v,w) ordenador (arcos)
- Grafo no dirigido → pares (v,w) no ordenador (aristas)
- Grafo etiquetado → cada arista o arcos tiene una etiqueta



Grafo simple no dirigido



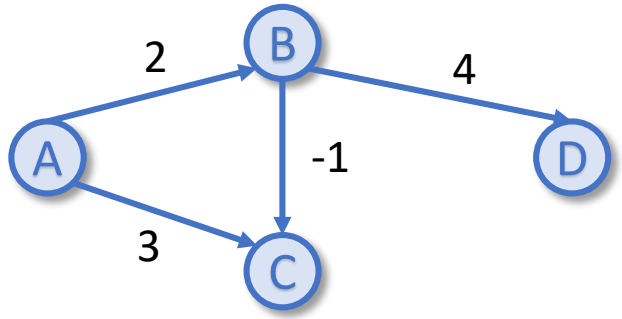
Multigrafo no dirigido



Grafo simple dirigido y etiquetado

Grafos etiquetados

- Formado por un trio (V,E,W) denotado por G(V,E,W)
  - V → Conjunto finito de vértices
  - E → Conjunto finito de arcos o aristas
  - W → Conjunto finito de etiquetas
    - Cada arco o arista dispone de una
    - $W = \{W_1, W_2, W_3, \ldots\ldots, W_n\}$
  - Tipos de etiquetas
    - Números → pesos que representan costes o beneficios
    - Caracteres o cadenas de caracteres



Grafo simple dirigido y etiquetado

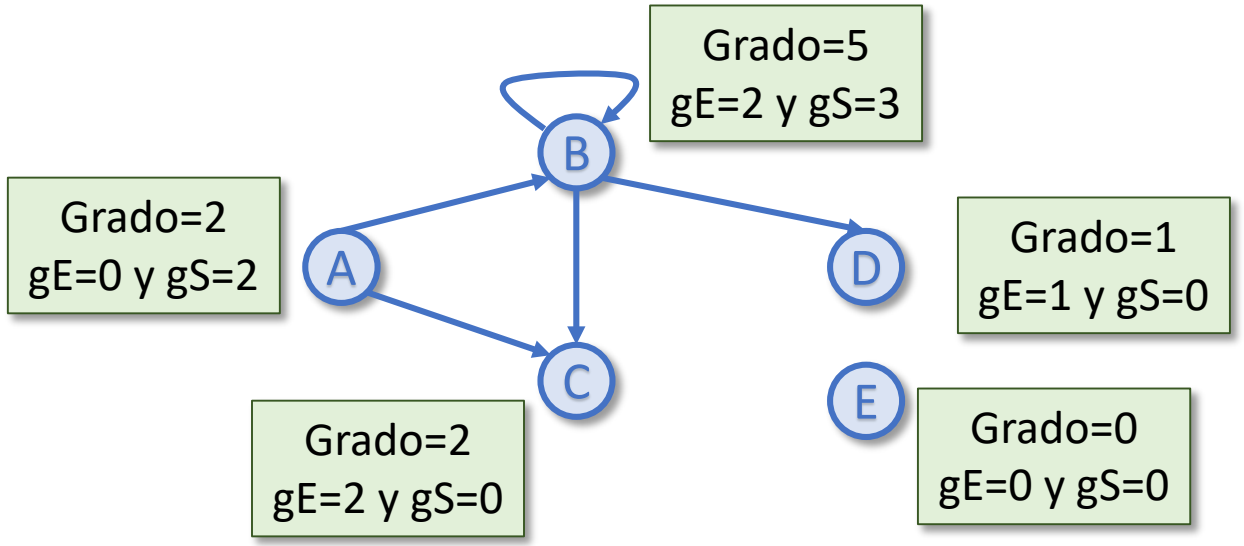
V = { A, B, C, D }

E = { (A,B), (A,C), (B,C), (B,D) }

W = { 2, 3, -1, 4 }

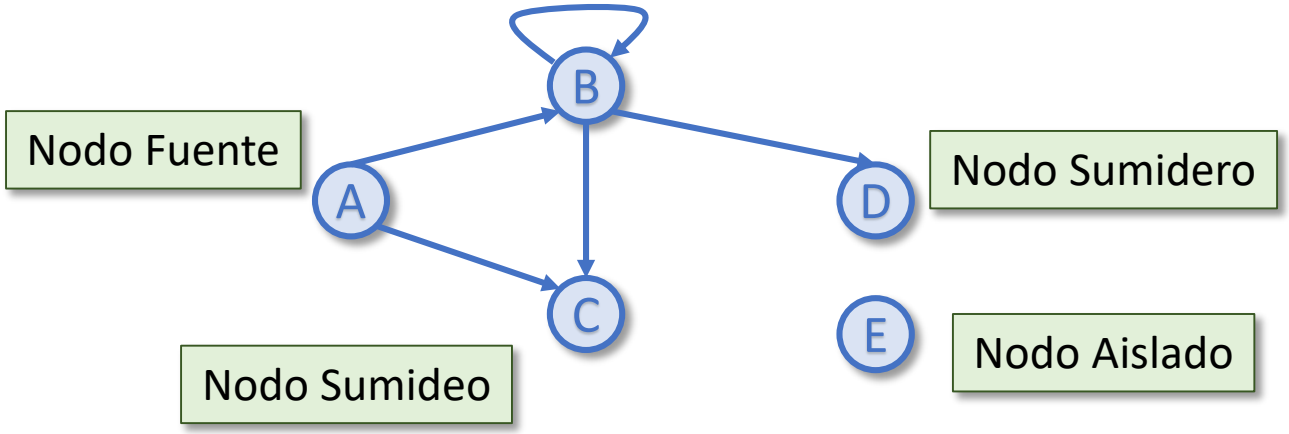
Concepto básicos

- Bucle → arco o arista con igual origen que destino
- Grado de un nodo -> número de arcos o aristas conectados al nodo
  - Grado de entrada (gE) de un nodo
    - Número de arcos o aristas que llegan al nodo
  - Grado de salida (gS) de un nodo
    - Número de arcos o aristas que salen de un nodo



Concepto básicos

Tipo nodo	Grado Salida	Grado Entrada
Nodo Fuente	>0	=0
Nodo Sumidero	=0	>0
Nodo Aislado	=0	=0



Capacidad de un grafo

- Cardinalidad de un grafo → número de vértices del grafo (n)
  - Se utiliza para calcular la eficiencia
- Número de arcos o arista mínimo es cero → $A_{min}(n) = 0$
- Número de arcos o aristas máximo
  - No dirigido
    - Sin bucles → $A_{max}(n) = n * (n-1) = n^2 - n$
    - Con bucles → $A_{max}(n) = n^2 - n + n = n^2$
  - Dirigido
    - Sin bucles → $A_{max}(n) = \frac{n*(n-1)}{2} = \frac{n^2-n}{2}$
    - Con bucles → $A_{max}(n) = \frac{n^2-n}{2} + n = \frac{n^2+n}{2}$
- Grafo completo → Número máximo de ejes

Densidad de un grafo

- Densidad de un grafo → proporción de ejes que posee
- Sea G = (V, E) un grafo simple, **m** el número de ejes y **n** el número vértices, se define la densidad como

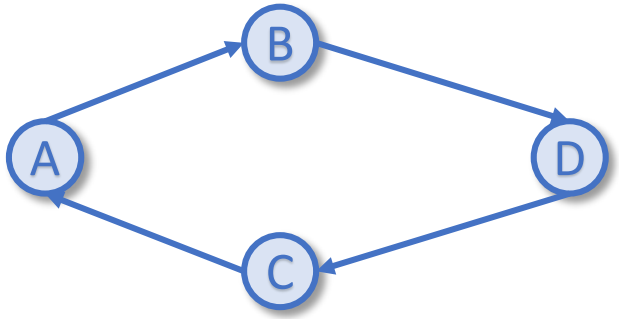
$$d = \frac{2m}{n\,(n-1)} \quad \Rightarrow \quad 0 < d < 1$$

  - Si d=0 todos los vértices son aislados
  - Si d=1 se trata de un grafo completo

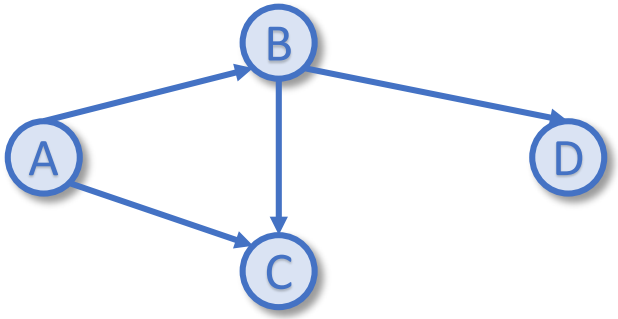
- Representación en memora
  - Matriz de adyacencias
  - Lista de adyacencias

Grafo fuertemente conexo

- Nodo fuertemente conexo
  - Desde dicho nodo se puede acceder al resto de los nodos y viceversa
- Grafo fuertemente conexo
  - Si todos los nodos del grafo son fuertemente conexos
- Ejemplos:



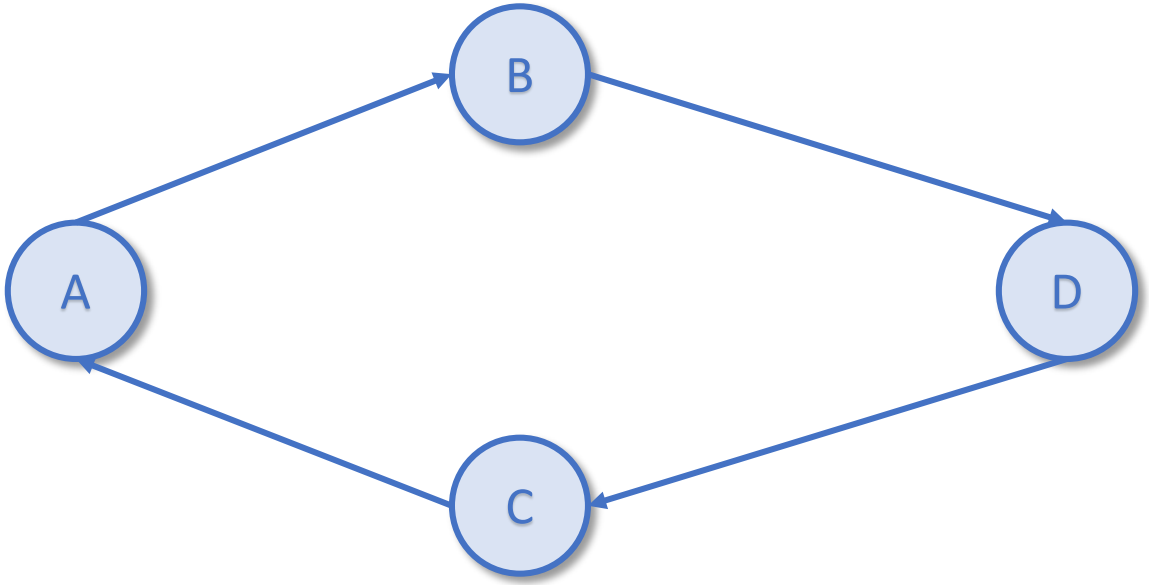
Grafo fuertemente conexo



Grafo **NO** fuertemente conexo

Ciclo sobre un nodo

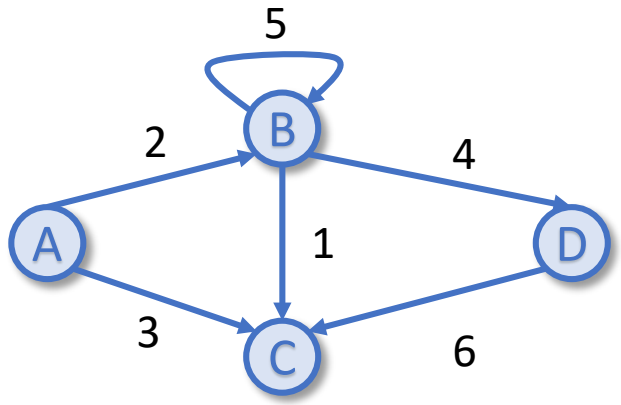
- Camino desde el nodo hasta si mismo
- Ejemplo:
  - A, B, D, C, A → Ciclo para A



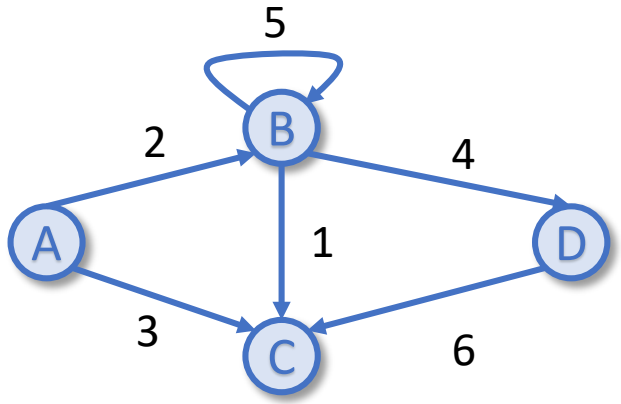
Implementación de un Grafo

La estructura de datos para definir un grafo

- Clase genérica para definir un grafo con sus atributos y sus métodos mediante una **matriz de adyacencias**
- Atributos que definen un grafo
  - El **número de nodos** insertados hasta el momento
  - Un **vector** donde se almacenarán los nodos que serán de tipo genérico
  - Una **matriz** donde se almacenarán valores booleanos e indicará para cada par de nodos si existe un eje o no
  - Una **matriz** donde se almacenarán los pesos del eje entre dos nodos si existe el eje
- Métodos. Todos aquellos necesarios para gestionar un grafo



Matriz de adyacencias



Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	F	T	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	∞	6	∞	
4					

Podemos suponer que el peso para un par de nodos no conectado (sin eje) es infinito (∞)

Matriz de adyacencias - Eficiencia

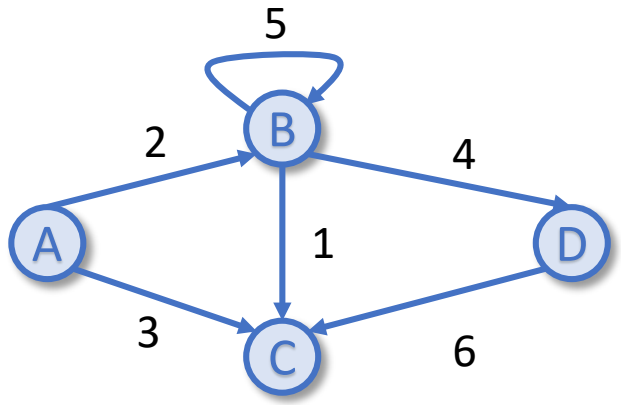
VENTAJAS
Acceso instantáneo a un elemento o a la información de cualquier elemento de cualquier matriz

DESVENTAJAS
Dificultad para determinar el tamaño inicial de la matriz. Debería de ser lo mas cercado al número de nodos
Desaprovechamiento de memoria en grafos ligeros

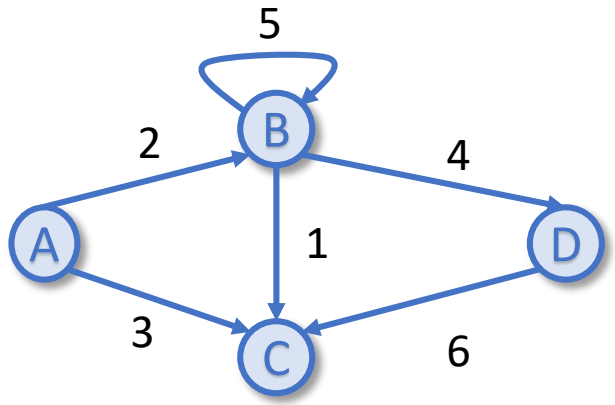
Indicada para grafos densos

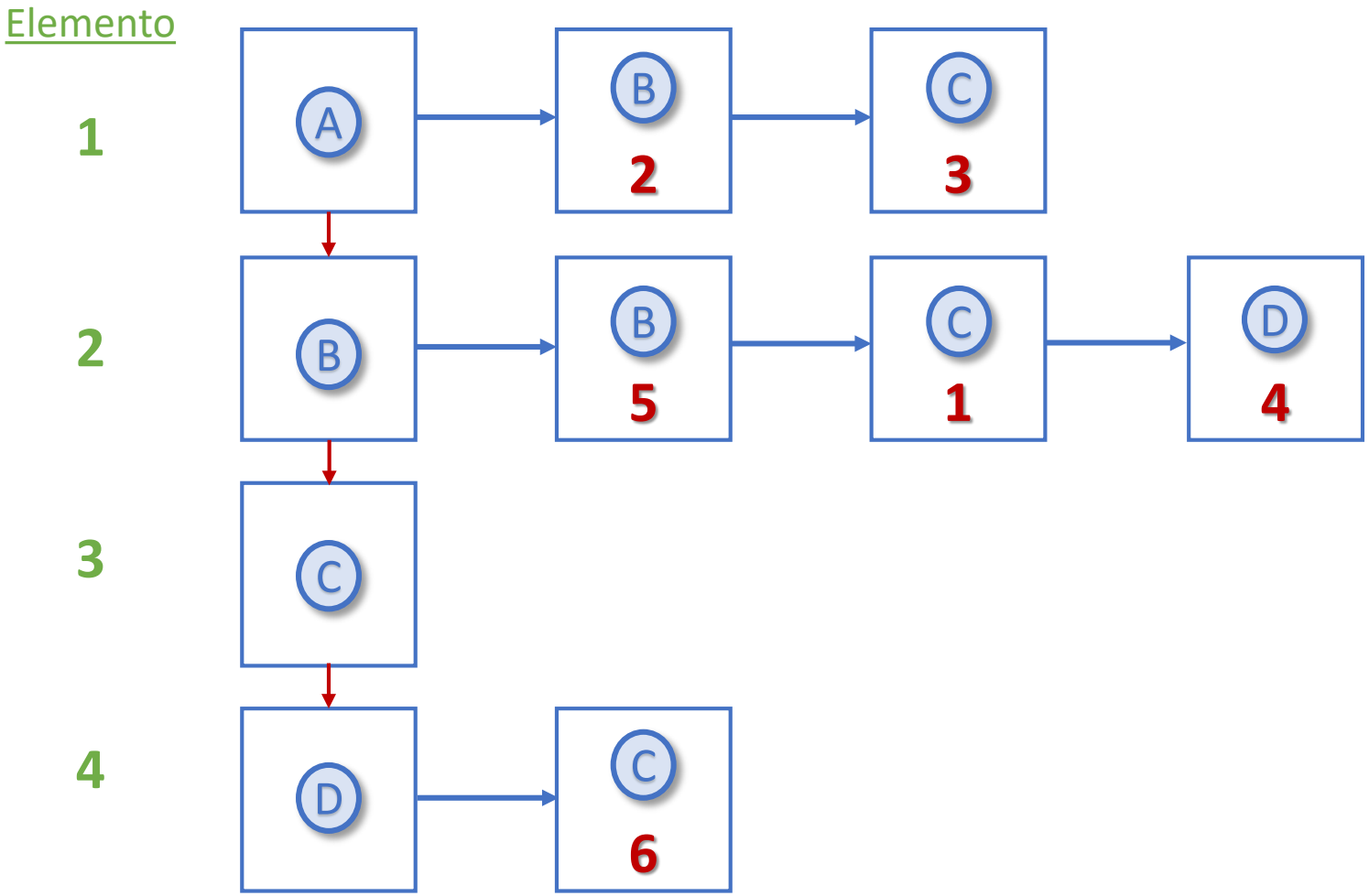
La estructura de datos para definir un grafo

- Otra opción → **lista de adyacencias**
- Clase para definir un eje
  - El **peso** del eje
  - El **nodo destino**
- Clase para definir un nodo
  - La **información** del nodo de tipo genérica
  - Una **lista de ejes** con los que conecta ese nodo
- Clase grafo con una propiedad que será la **lista enlazada** de nodos del grafo

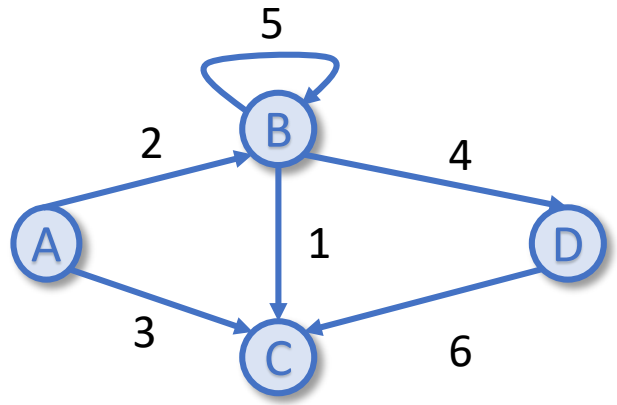


Lista de adyacencias





Lista de adyacencias



Elemento

1 → A → B (2) → C (3)

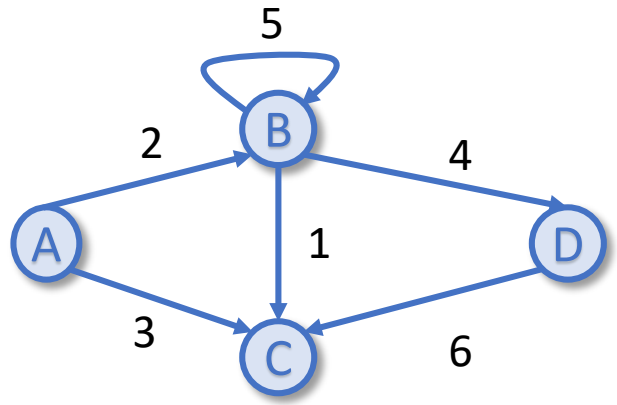
2 → B → B (5) → C (1) → D (4)

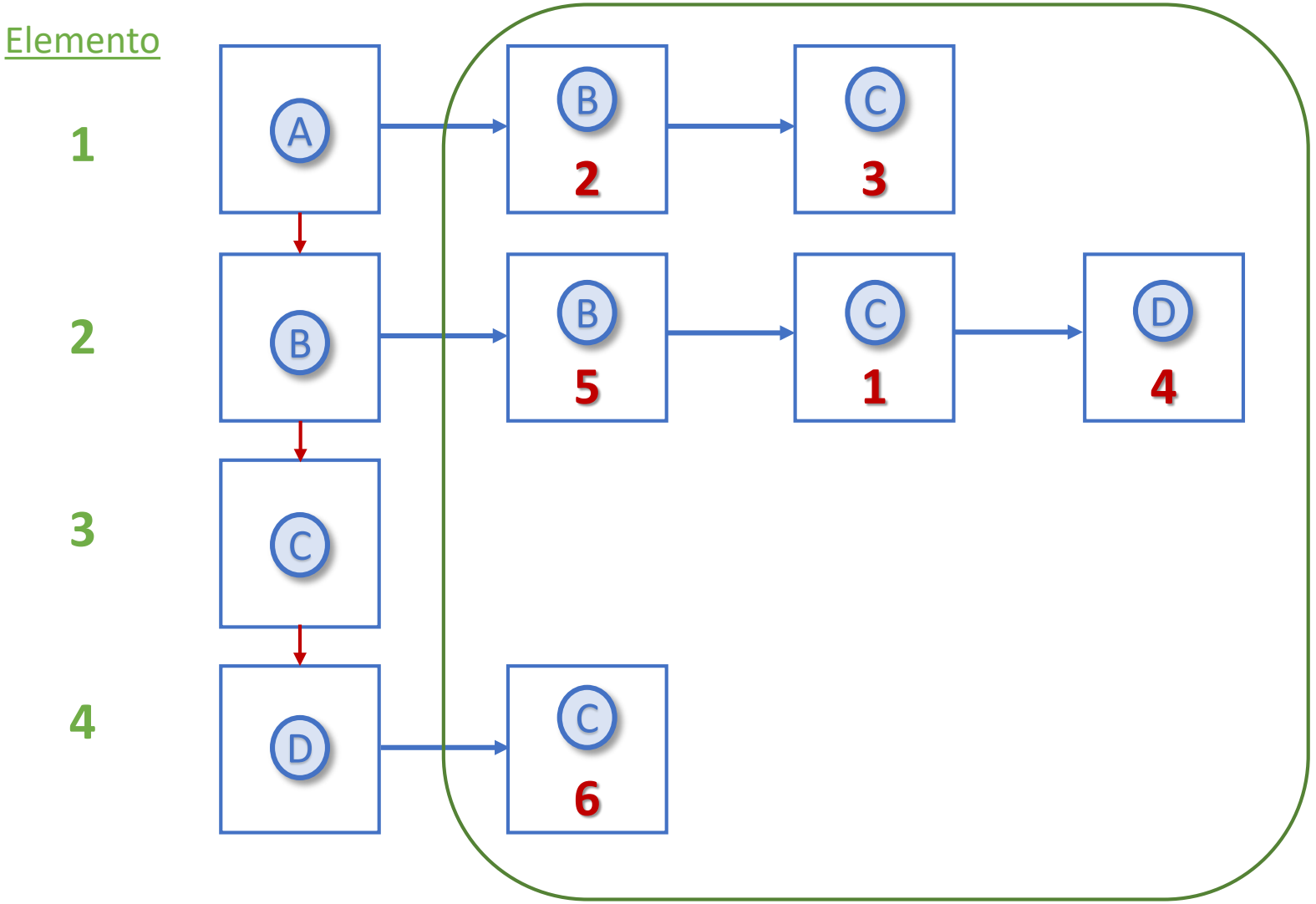
3 → C

4 → D → C (6)

Lista enlazada de nodos

Lista de adyacencias





Conjunto de listas de ejes por nodo

Lista de adyacencias - Eficiencia

VENTAJAS
Memoria consumida en función del número de nodos y del número de aristas reales

DESVENTAJAS
Es necesario realizar búsquedas secuenciales complejas en las listas
Si el grafo es denso se desaprovecha gran cantidad de memoria en las referencias necesarias para mantener las listas. El grado máximo de desaprovechamiento de memoria se alcanza con el grafo completo.

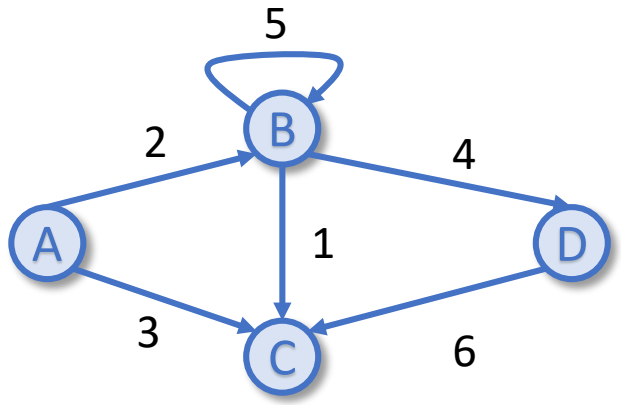
Indicada para grafos ligeros

Grafos – Matriz de adyacencia

Método	Complejidad
graph (constructor)	O(1)
getNode	O(n)
addNode	O(n)
removeNode	O(n)
existEdge	O(n)
addEdge	O(n)
removeEdge	O(n)
print	$O(n^2)$

La estructura de datos para definir un grafo

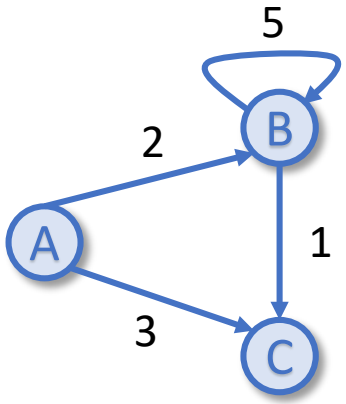
- Clase genérica para definir un grafo con sus atributos y sus métodos mediante una **matriz de adyacencias**
- Atributos que definen un grafo
  - El **número de nodos** insertados hasta el momento
  - Un **vector** donde se almacenarán los nodos que serán de tipo genérico
  - Una **matriz** donde se almacenarán valores booleanos e indicará para cada par de nodos si existe un eje o no
  - Una **matriz** donde se almacenarán los pesos del eje entre dos nodos si existe el eje
- Métodos. Todos aquellos necesarios para gestionar un grafo



Grafos - Métodos

- graph (constructor) → O(1)
  - Se inicializa el vector y las matrices con el tamaño pasado como parámetro
  - Se inicializa el número de elementos a cero
- getNode → O(n)
  - Se recorre el vector para devolver la posición del valor de un nodo que se pasa como parámetro
- existEdge → O(n)
  - Se recorre la fila del nodo origen pasado como parámetro para ver si existe un eje hasta el nodo destino especificado como parámetro
- toString → $O(n^2)$
  - Muestra el vector de nodos y las matrices de pesos y ejes

Grafos – Métodos - addNode



- Añade un nuevo nodo al grafo con la información que se pasa como parámetro
- Inicializa la fila y columna de la matriz de pesos a **Infinito**
- Inicializa la fila y la columna de la matriz de ejes a **False**
- Incrementar el número de nodos insertados

Nodos

Posición	0	1	2	3	4
Valor	A	B	C		

NumeroNodos = 3

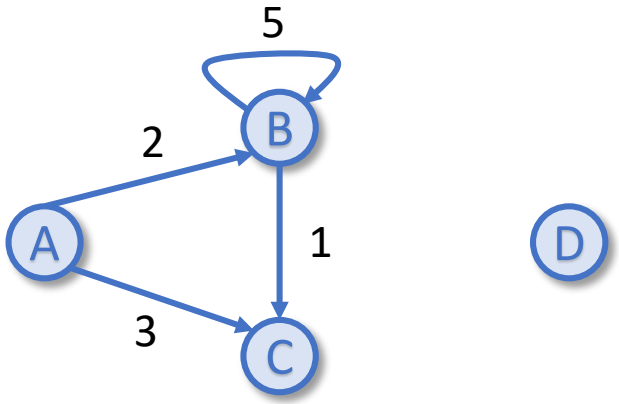
Ejes

	0	1	2	3	4
0	F	T	T		
1	F	T	T		
2	F	F	F		
3					
4					

Pesos

	0	1	2	3	4
0	∞	2	3		
1	∞	5	1		
2	∞	∞	∞		
3					
4					

Grafos – Métodos - addNode



- Añade un nuevo nodo al grafo con la información que se pasa como parámetro
- Inicializa la fila y columna de la matriz de pesos a **infinito**
- Inicializa la fila y la columna de la matriz de ejes a **False**
- Incrementar el número de nodos insertados

Insertar el **nodo D**

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

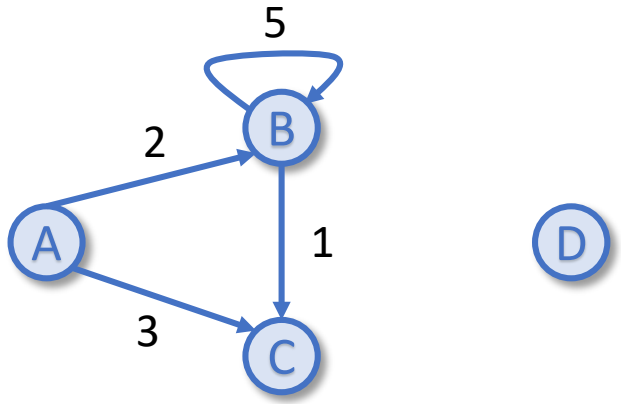
	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	F	
2	F	F	F	F	
3	F	F	F	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	∞	
2	∞	∞	∞	∞	
3	∞	∞	∞	∞	
4					

Grafos – Métodos - addEdge

- Añade un nuevo eje entre dos nodos que se pasan como parámetro
- Almacena el peso pasado como parámetro en la posición correspondiente de la matriz de pesos
- Almacena el valor True en la posición correspondiente de la matriz de ejes



Insertar el **eje de B a D** con **peso 4**

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

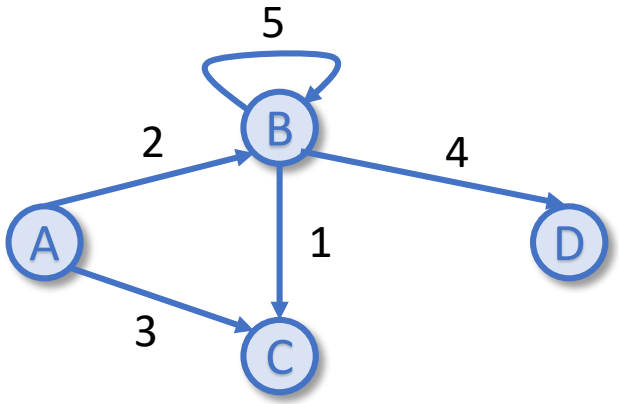
	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	F	
2	F	F	F	F	
3	F	F	F	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	∞	
2	∞	∞	∞	∞	
3	∞	∞	∞	∞	
4					

Grafos – Métodos - addEdge

- Añade un nuevo eje entre dos nodos que se pasan como parámetro
- Almacena el peso pasado como parámetro en la posición correspondiente de la matriz de pesos
- Almacena el valor True en la posición correspondiente de la matriz de ejes



Insertar el **eje de B a D** con **peso 4**

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

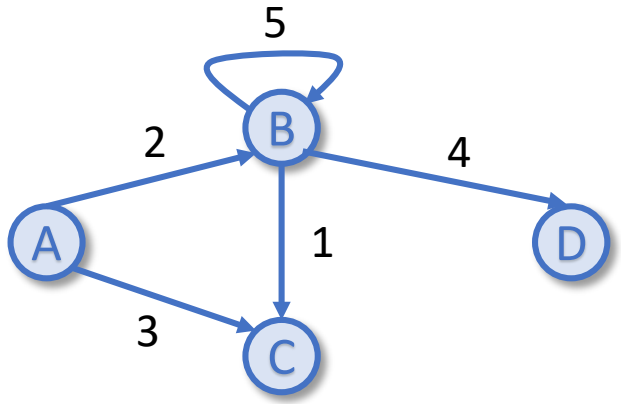
	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	F	F	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	∞	∞	∞	
4					

Grafos – Métodos - addEdge

- Añade un nuevo eje entre dos nodos que se pasan como parámetro
- Almacena el peso pasado como parámetro en la posición correspondiente de la matriz de pesos
- Almacena el valor True en la posición correspondiente de la matriz de ejes



Insertar el **eje de B a D** con **peso 4**
Insertar el **eje de D a C** con **peso 6**

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

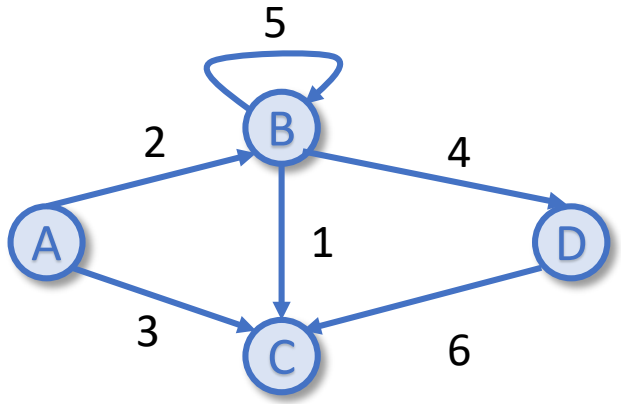
	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	F	F	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	∞	∞	∞	
4					

Grafos – Métodos - addEdge

- Añade un nuevo eje entre dos nodos que se pasan como parámetro
- Almacena el peso pasado como parámetro en la posición correspondiente de la matriz de pesos
- Almacena el valor True en la posición correspondiente de la matriz de ejes



Insertar el **eje de B a D** con **peso 4**
Insertar el **eje de D a C** con **peso 6**

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

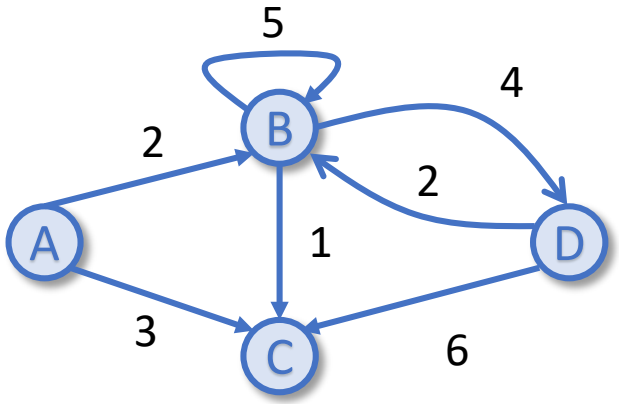
Ejes

	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	F	T	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	∞	6	∞	
4					

Grafos – Métodos – removeNode



- Borra el nodo que se pasa como parámetro
- Copia el último nodo almacenado en la posición del nodo a borrar
- Copia la última fila y columna de la matrices a la fila y columna del nodo a borrar

Borrar el **nodo B**
pos = 1

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

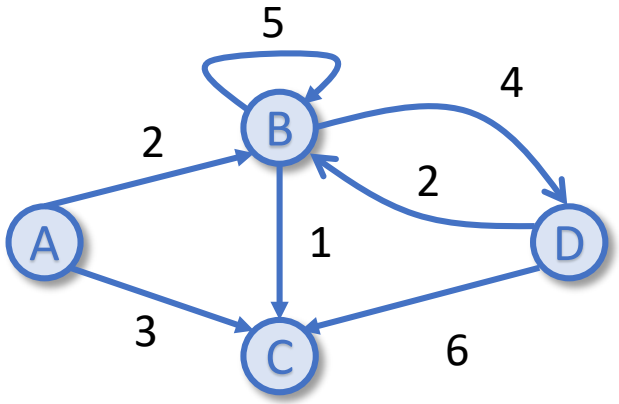
Ejes

	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	T	T	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	2	6	∞	
4					

Grafos – Métodos – removeNode



- Borra el nodo que se pasa como parámetro
- Copia el último nodo almacenado en la posición del nodo a borrar
- Copia la última fila y columna de la matrices a la fila y columna del nodo a borrar

Borrar el **nodo B**
pos = 1

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

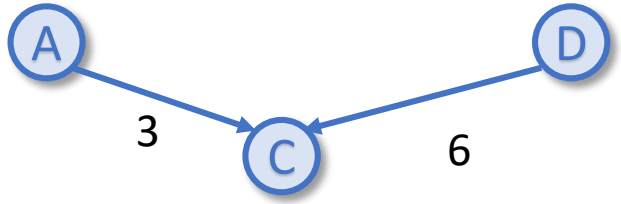
	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	T	T	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	2	6	∞	
4					

Grafos – Métodos – removeNode

- Borra el nodo que se pasa como parámetro
- Copia el último nodo almacenado en la posición del nodo a borrar
- Copia la última fila y columna de la matrices a la fila y columna del nodo a borrar



Borrar el **nodo B**
pos = 1

Nodos

Posición	0	1	2	3	4
Valor	A	D	C	??	

NumeroNodos = 4

Ejes

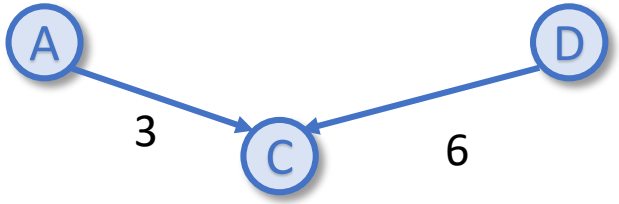
	0	1	2	3	4
0	F	F	T	??	
1	F	T	T	??	
2	F	F	F	??	
3	??	??	??	F	
4					

Pesos

	0	1	2	3	4
0	∞	∞	3	??	
1	∞	2	6	??	
2	∞	∞	∞	??	
3	∞	2??	??	∞	
4					

Grafos – Métodos – removeNode

- Borra el nodo que se pasa como parámetro
- Copia el último nodo almacenado en la posición del nodo a borrar
- Copia la última fila y columna de la matrices a la fila y columna del nodo a borrar
- El elemento de la esquina inferior derecha pasa a ser el elemento (pos,pos)



Borrar el **nodo B**
pos = 1

Nodos

Posición	0	1	2	3	4
Valor	A	D	C	??	

NumeroNodos = 4

Ejes

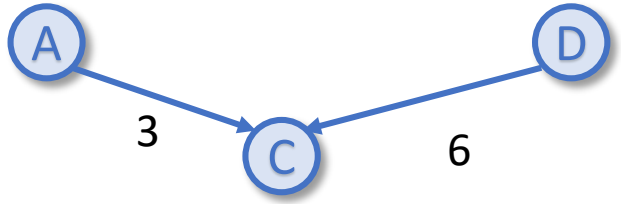
	0	1	2	3	4
0	F	F	T	??	
1	F	T	T	??	
2	F	F	F	??	
3	??	??	??	F	
4					

Pesos

	0	1	2	3	4
0	∞	∞	3	??	
1	∞	2	6	??	
2	∞	∞	∞	??	
3	∞	2??	??	∞	
4					

Grafos – Métodos – removeNode

- Borra el nodo que se pasa como parámetro
- Copia el último nodo almacenado en la posición del nodo a borrar
- Copia la última fila y columna de la matrices a la fila y columna del nodo a borrar
- El elemento de la esquina inferior derecha pasa a ser el elemento (pos,pos)
- Decrementar el número de nodos insertados



Borrar el **nodo B**
pos = 1

Nodos

Posición	0	1	2	3	4
Valor	A	D	C		

NumeroNodos = 3

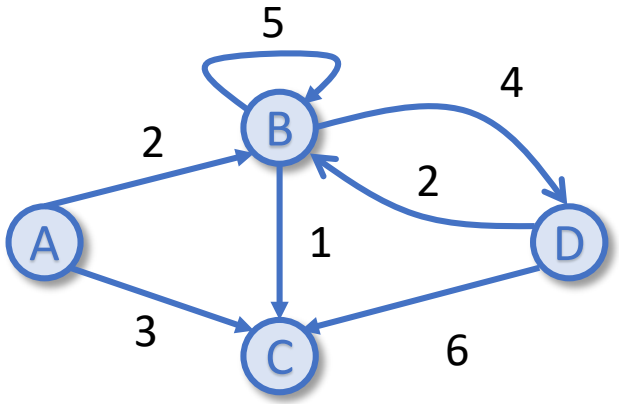
Ejes

	0	1	2	3	4
0	F	F	T		
1	F	F	T		
2	F	F	F		
3					
4					

Pesos

	0	1	2	3	4
0	∞	∞	3		
1	∞	∞	6		
2	∞	∞	∞		
3					
4					

Grafos – Métodos – removeEdge



- Borra el nodo que va desde un nodo a otro que se pasan como parámetro
- Poner la posición correspondiente de la matriz de ejes a **False**
- Poner la posición correspondiente de la matriz de pesos a **Infinito**

Borrar el eje de B a C
pos (1,2)

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

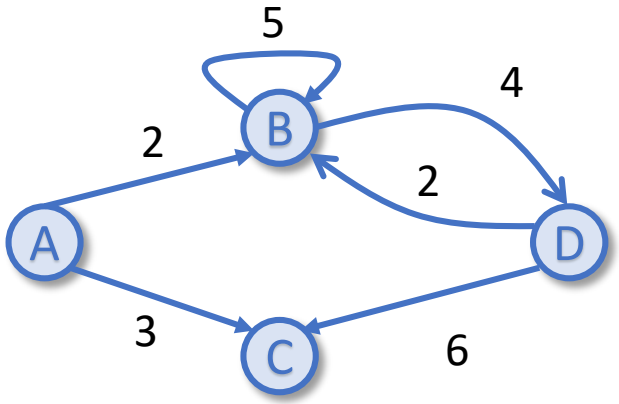
Ejes

	0	1	2	3	4
0	F	T	T	F	
1	F	T	T	T	
2	F	F	F	F	
3	F	T	T	F	
4					

Pesos

	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	1	4	
2	∞	∞	∞	∞	
3	∞	2	6	∞	
4					

Grafos – Métodos – removeEdge



- Borra el nodo que va desde un nodo a otro que se pasan como parámetro
- Poner la posición correspondiente de la matriz de ejes a **False**
- Poner la posición correspondiente de la matriz de pesos a **Infinito**

Borrar el eje de B a C
pos (1,2)

Nodos

Posición	0	1	2	3	4
Valor	A	B	C	D	

NumeroNodos = 4

Ejes

	0	1	2	3	4
0	F	T	T	F	
1	F	T	**F**	T	
2	F	F	F	F	
3	F	T	T	F	
4					

Pesos

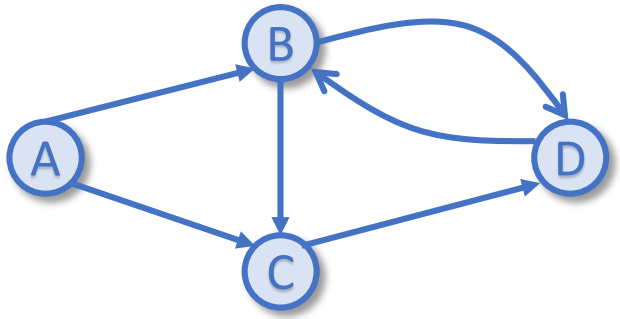
	0	1	2	3	4
0	∞	2	3	∞	
1	∞	5	**∞**	4	
2	∞	∞	∞	∞	
3	∞	2	6	∞	
4					

Grafos – Métodos Avanzados

Método	Complejidad
Dijkstra	$O(n^2)$
Floyd	$O(n^3)$
Recorrido en Profundidad	$O(n^2)$
Prim / Warshall	$O(n^2)$

Grafos – Conceptos básicos

- **Camino** entre dos nodos distintos
  - Secuencia de nodos con sus respectivos ejes que permiten acceder desde un nodo a otro
  - Pueden existir muchos caminos distintos

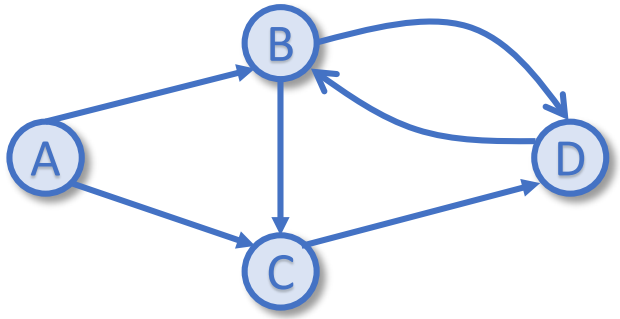


Caminos para ir del nodo A al nodo D

- A, C, D
- A, B, D
- A, B, C, D

Grafos – Conceptos básicos

- **Longitud** de un camino entre dos nodos distintos
  - Número de ejes empleados para llegar de un nodo a otro
  - Número de nodos del camino menos uno
  - Puede haber mas de un camino



Longitud del camino para ir del nodo A al nodo D

- A, C, D → 3 – 1 = 2
- A, B, D → 3 – 1 = 2
- A, B, C, D → 4 – 1 = 3

Grafos – Conceptos básicos

- **Camino simple** entre dos nodos distintos
  - Es aquel camino en el que **no se repite** ningún nodo para acceder desde un nodo a otro

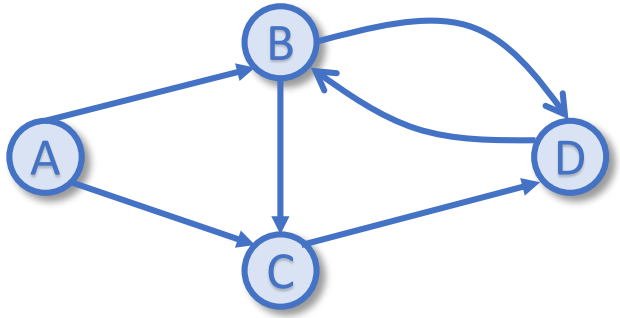
Teorema del Camino Simple

Si existe algún camino entre un par de nodos origen y destino, entonces existe al menos un camino simple entre ellos

- Es posible eliminar los ciclos de un camino para convertirlo en un **camino simple**.

Grafos – Conceptos básicos

- **Camino de longitud mínima** entre dos nodos distintos
  - Es aquel camino que implique pasar por el menor número de ejes para acceder desde un nodo a otro
  - El camino de longitud mínima es simple



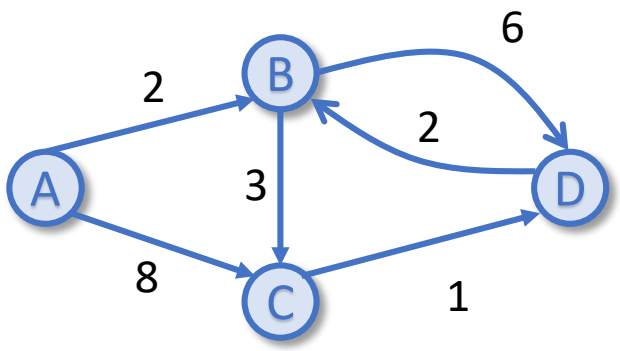
Camino de longitud mínima para ir del nodo A al nodo D

- Camino1 → A, C, D (longitud 2)
- Camino2 → A, B, D (longitud 2)
- Camino3 → A, B, C, D (longitud 3)

Camino de longitud mínima **A, C, D** o **A, B, D**

Grafos – Conceptos básicos

- **<u>Camino de coste mínimo</u>** entre dos nodos distintos
  - Es aquel que implica pasar por ejes cuya suma de pesos es mínima
  - El camino de coste mínimo no necesariamente se corresponde con el camino de longitud mínima



Camino de coste mínimo para ir del nodo A al nodo D

- Camino1 → A, C, D (coste 9)
- Camino2 → A, B, D (coste 8)
- Camino3 → A, B, C, D (coste 6)

Camino de coste mínimo **A, B, C, D** con **coste 6**

Grafos – Algoritmo de Dijkstra

- Algoritmo de caminos mínimos
- Algoritmos de búsqueda
- Determinar la ruta más corta, desde el nodo origen, hasta cualquier nodo del grafo
- Desarrollado por el holandés Edger Dijkstra en 1956
- Ejemplos de uso
  - ¿Cuál es la ruta más corta en tiempo para ir de Oviedo a Madrid?
  - ¿Cuál es la ruta mas corta en Km para ir de Oviedo a Madrid?

Grafos – Algoritmo de Dijkstra

Se parte de un nodo origen que se pasa como parámetro
El coste de ir de un nodo a si mismo es cero
Todos los costes son positivos

Inicializaciones

- **Inicializar** un vector **S** unidimensional con el nodo desde el cual si quiere calcular la ruta mas corta a cualquier otro nodo
- **Inicializar** el vector **D** unidimensional con el coste de ir de el nodo origen al resto de los nodos

Obtiene

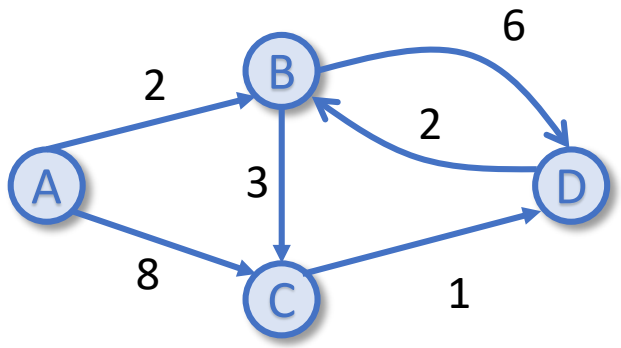
- Un vector **D** unidimensional (Costes Mínimos) con los costes de ir del nodo origen al resto de los nodos del grafo
- Un vector **P** unidimensional (Rutas de Coste Mínimo) con las ruta de coste mínimo para ir del nodo origen al resto de los nodos del grafo

El algoritmo devuelve el vector D

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos

Paso1: Inicialización
S = {A}



nodos = {A, B, C, D}

Matriz de Pesos

	A	B	C	D
A	∞	2	8	∞
B	∞	∞	3	6
C	∞	∞	∞	1
D	∞	2	∞	∞

Matriz de Ejes

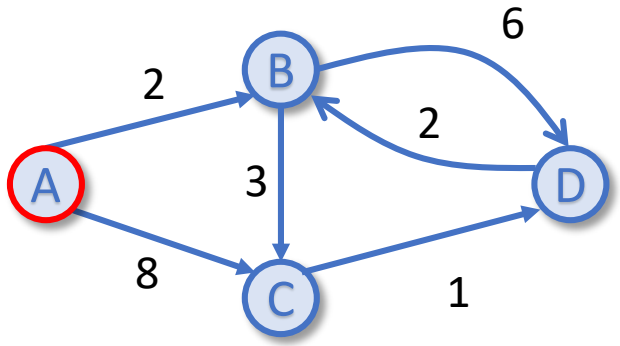
	A	B	C	D
A	F	T	T	F
B	F	F	T	T
C	F	F	F	T
D	F	T	F	F

D	A	B	C	D
	0	2	8	∞

P	A	B	C	D
		A	A	

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A}
nodos = {A, B, C, D}

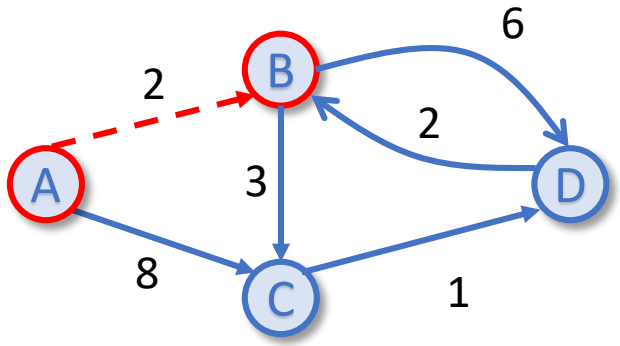
Paso2: Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
Agregar w al conjunto solución

D	A	B	C	D
	0	2	8	∞

P	A	B	C	D
		A	A	

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A}
nodos = {A, B, C, D}

Paso2: Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
Agregar w al conjunto solución

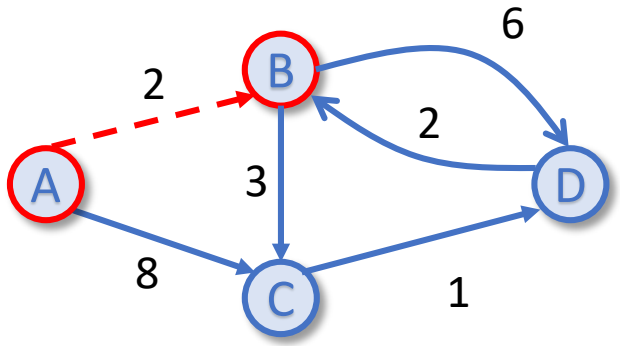
Mínimo = B → w = B
S = {A, B}

D	A	B	C	D
	0	2	8	∞

P	A	B	C	D
		A	A	

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A,B}
w = B
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {C, D} hacer D[v] = min(D[v], D[w]+Coste[w, v])

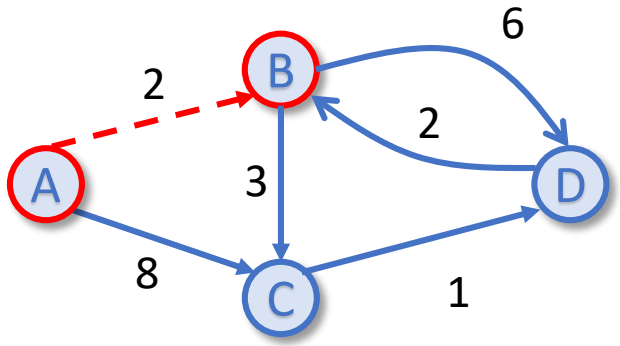
Para v = C
D[C] = min(D[C], D[B]+C[B, C]) = min(8, 2+3) = 5 → Mejora

D	A	B	C	D
	0	2	8	∞

P	A	B	C	D
		A	A	

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A,B}
w = B
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {C, D} hacer D[v] = min(D[v], D[w]+Coste[w, v])

Para v = C
D[C] = min(D[C], D[B]+Coste[B, C]) = min(8, 2+3) = 5 → Mejora

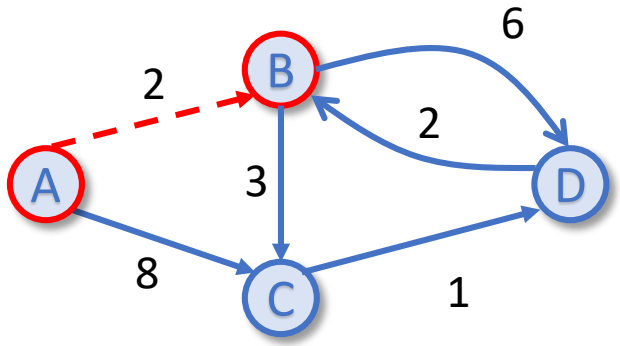
Para v = D
D[D] = min(D[D], D[B]+Coste[B, D]) = min(∞, 2+6) = 8 → Mejora

D	A	B	C	D
	0	2	**5**	∞

P	A	B	C	D
		A	B	

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A,B}
w = B
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {C, D} hacer D[v] = min(D[v], D[w]+Coste[w, v])

Para v = C
D[C] = min(D[C], D[B]+Coste[B, C]) = min(8, 2+3) = 5 → Mejora

Para v = D
D[D] = min(D[D], D[B]+Coste[B, D]) = min(∞, 2+6) = 8 → Mejora

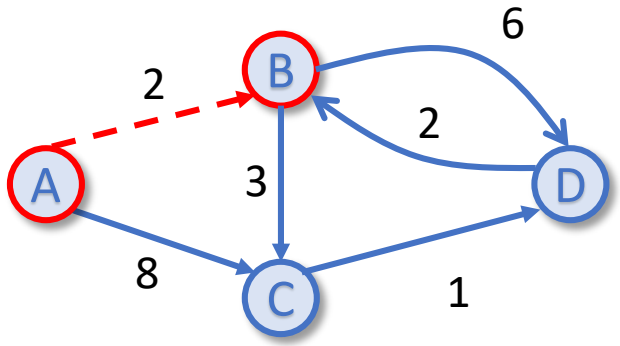
D	A	B	C	D
	0	2	5	8

P	A	B	C	D
		A	B	B

Realizar Paso2 y Paso 3 hasta que nodos-S sea vacío

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B}
nodos = {A, B, C, D}

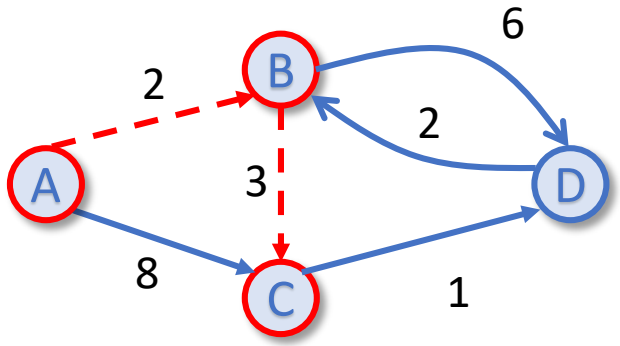
Paso2: Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
Agregar w al conjunto solución

D	A	B	C	D
	0	2	5	8

P	A	B	C	D
		A	B	B

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B}
nodos = {A, B, C, D}

Paso2: Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
Agregar w al conjunto solución

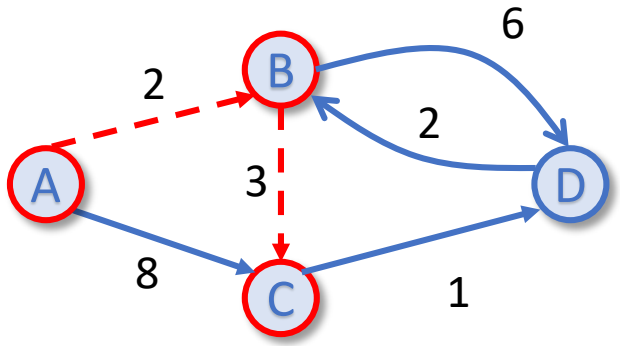
Mínimo = C → w = C
S = {A, B, C}

D	A	B	C	D
	0	2	5	8

P	A	B	C	D
		A	B	B

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B, C}
w = C
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {D} hacer D[v] = min(D[v], D[w]+Coste[w, v])

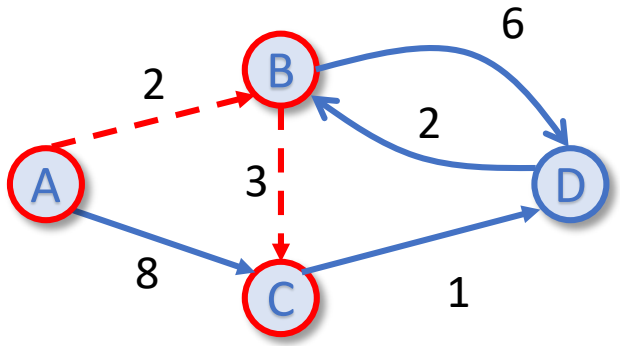
Para v = D
D[D] = min(D[D], D[C]+Coste[C, D]) = min(8, 5+1) = 6 →
Mejora

D	A	B	C	D
	0	2	5	8

P	A	B	C	D
		A	B	B

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B, C}
w = C
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {D} hacer D[v] = min(D[v], D[w]+Coste[w, v])

Para v = D
D[D] = min(D[D], D[C]+Coste[C, D]) = min(8, 5+1) = 6 →
Mejora

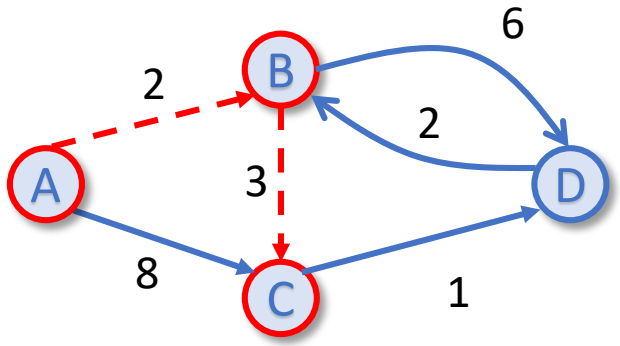
D	A	B	C	D
	0	2	5	6

P	A	B	C	D
		A	B	C

Realizar Paso2 y Paso 3 hasta que nodos-S sea vacío

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B, C}
nodos = {A, B, C, D}

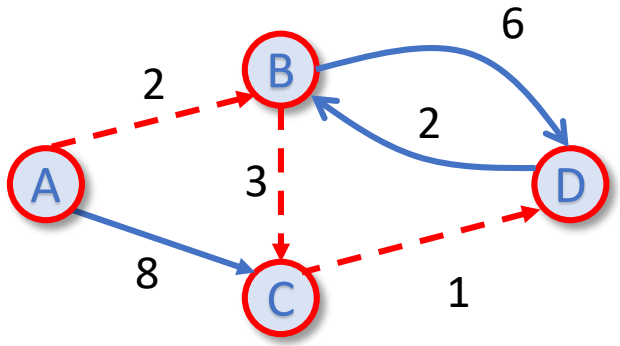
Paso2: Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
Agregar w al conjunto solución

D	A	B	C	D
	0	2	5	6

P	A	B	C	D
		A	B	C

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B, C}
nodos = {A, B, C, D}

Paso2: Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
Agregar w al conjunto solución

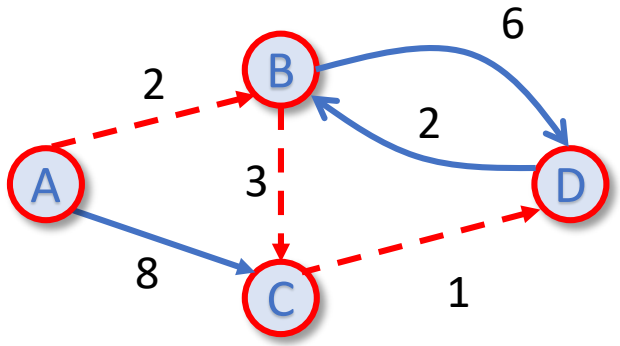
Mínimo = D → w = D
S = {A, B, C, D}

D	A	B	C	D
	0	2	5	6

P	A	B	C	D
		A	B	C

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B, C, D}
w = D
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {} hacer D[v] = min(D[v], D[w]+Coste[w, v])

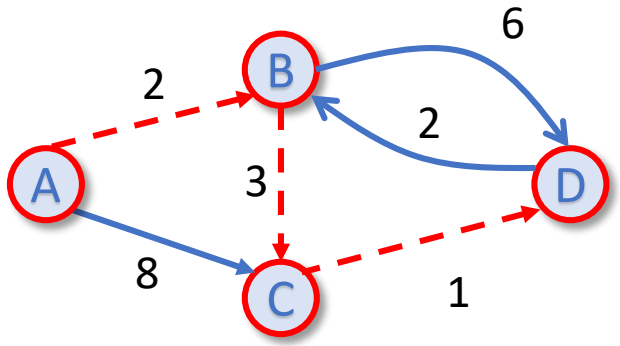
Como ya no hay nodos → Se finaliza el proceso

D	A	B	C	D
	0	2	5	6

P	A	B	C	D
		A	B	C

Grafos – Algoritmo de Dijkstra - Ejemplo

Encontrar el camino de coste mínimo desde A al resto de los nodos



S = {A, B, C, D}
w = D
nodos = {A, B, C, D}

Paso3: Para cada v ∈ {} hacer D[v] = min(D[v], D[w]+Coste[w, v])

Como ya no hay nodos → Se finaliza el proceso

El algoritmo devuelve D
Coste del camino mínimo para ir del nodo A al resto d e los nodos

D	A	B	C	D
	0	2	5	6

Con P se puede calcular el camino que va desde A hasta cualquiera de los modos

P	A	B	C	D
		A	B	C

Grafos – Algoritmo de Dijkstra - Resumiendo

Encontrar el camino de coste mínimo desde A al resto de los nodos

nodos → Conjunto de nodos
S → Nodo origen
P → Vector costes mínimos. Coste desde el nodo origen hasta cada uno de los nodos
D → Vector de rutas de coste mínimo
mientras (nodos – S) no sea un conjunto vacío **hacer**
    Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
    Agregar w al conjunto solución (S)
    para cada v ∈ (nodos – S) **hacer**
        D[v] = min(D[v], D[w]+Coste[w, v])
        Actualizar P si procede
    fin_para
fin_mientras

Grafos – Algoritmo de Dijkstra - Resumiendo

Encontrar el camino de coste mínimo desde A al resto de los nodos

```
nodos → Conjunto de nodos
S → Nodo origen
P → Vector costes mínimos. Coste desde el nodo origen hasta cada uno de los nodos
D → Vector de rutas de coste mínimo
mientras (nodos – S)  no sea un conjunto vacío hacer
        Elegir un nodo w ∈ (nodos – S) tal que D[w] sea mínima
        Agregar w al conjunto solución (S)
        para cada v ∈ (nodos – S) hacer
                D[v] = min( D[v], D[w]+Coste[w, v] )
                Actualizar P si procede
        fin_para
fin_mientras
```

n (bucle interno) — **n** (bucle externo)

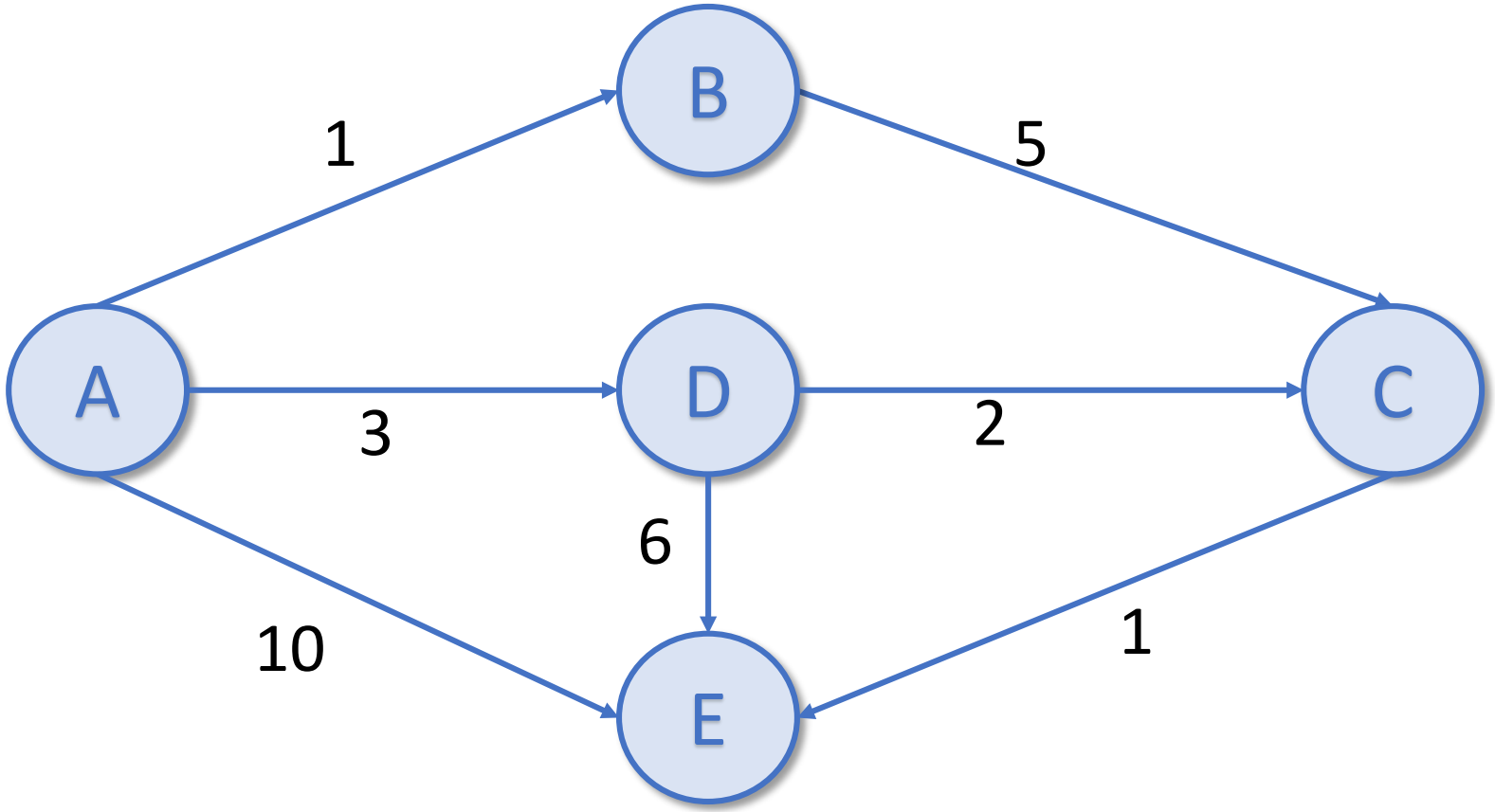
$O(n^2)$

Grafos – Algoritmo de Dijkstra - Importante

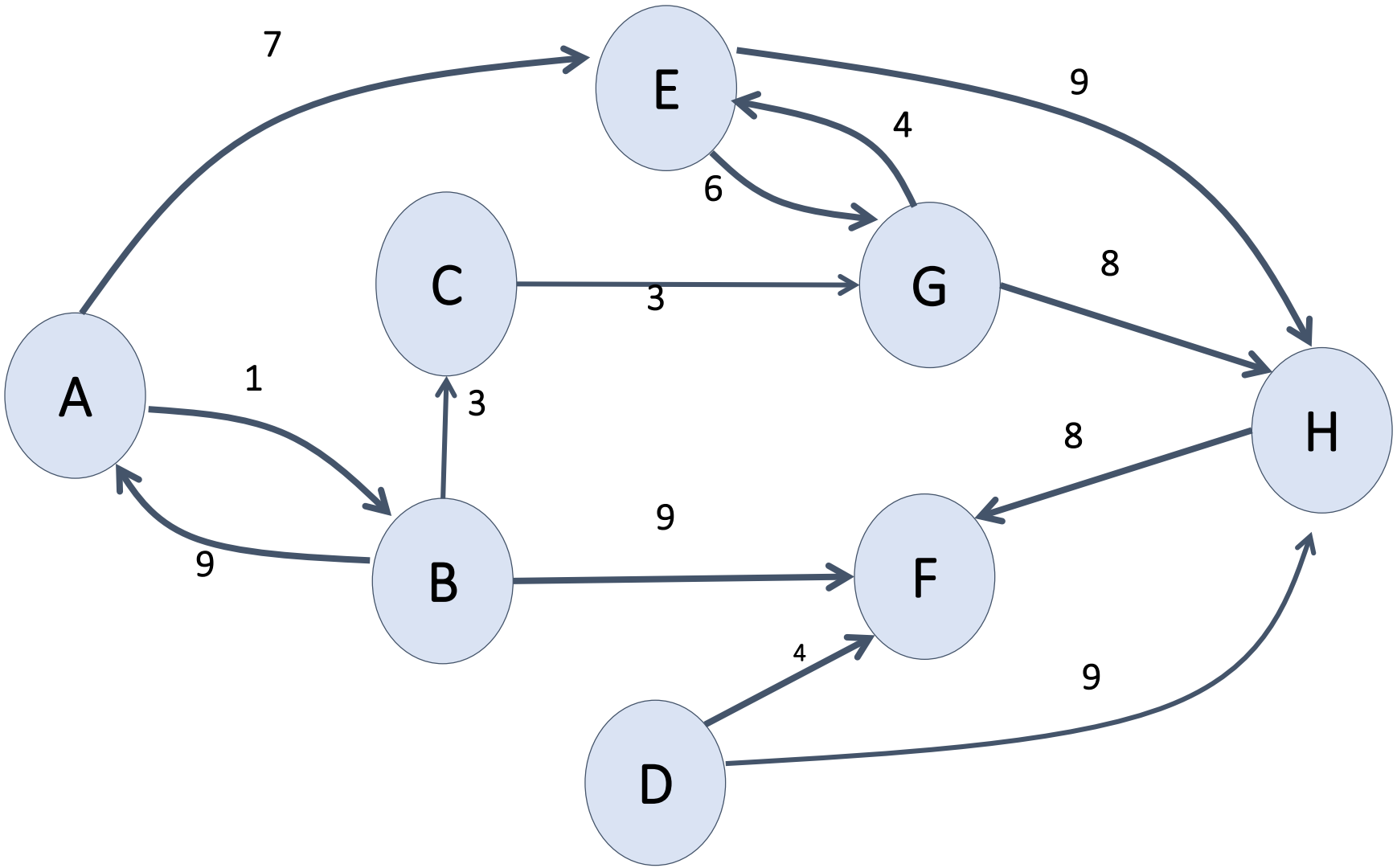
Encontrar el camino de coste mínimo desde A al resto de los nodos

El camino de longitud mínima se obtiene sustituyendo los costes de cada eje por un peso de 1

Ejercicio – Aplicar Dijkstra desde el nodo B



Ejercicio – Aplicar Dijkstra desde el nodo B



Grafos – Algoritmo de Floyd-Warshall

- Algoritmo de caminos mínimos
- Algoritmos de búsqueda
- Determinar la ruta más corta entre cualquier par de nodos del grafo
- Desarrollado por los estadounidenses Robert Floyd y Stephen Warshall en 1962
- Ejemplos de uso
  - ¿ Cuál es la ruta más barata para llegar a Madrid desde Oviedo, Sevilla o Burgos?

Grafos – Algoritmo de Floyd-Warshall

El coste de ir de un nodo a si mismo es cero
Todos los costes son positivos

Inicializaciones

- **Inicializar** la matriz **A** de Coste Mínimo
  - Copia de todos los valores de una matriz de pesos
  - Modifica la matriz para que aparezcan ceros en la diagonal. El coste de ir de un nodo a si mismo se considera nulo

Obtiene

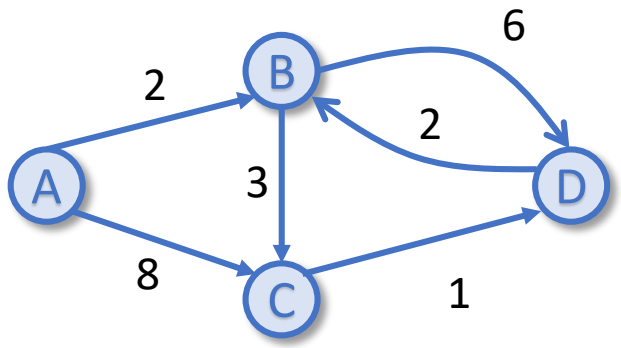
- La matriz **A** de Coste mínimo que almacenará el coste mínimo de ir desde cualquier nodo a cada uno de los restantes nodos del grafo
- La matriz **P** o de Rutas de Coste Mínimo, que almacena la secuencia de nodos que forman todos los caminos de coste mínimo

El algoritmo devuelve la matriz A

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso1: Inicialización



nodos = {A, B, C, D}

Matriz de Pesos

	A	B	C	D
A	∞	2	8	∞
B	∞	∞	3	6
C	∞	∞	∞	1
D	∞	2	∞	∞

A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

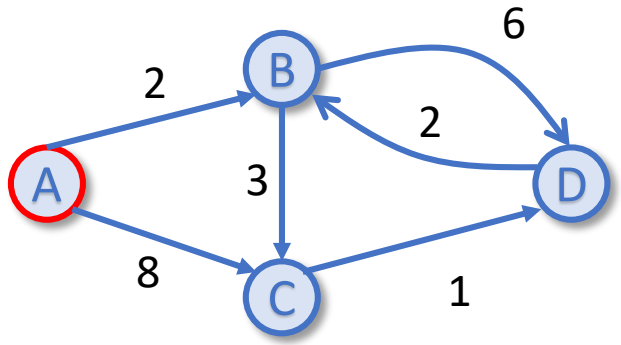
P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])



v = A → posición = 0

A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

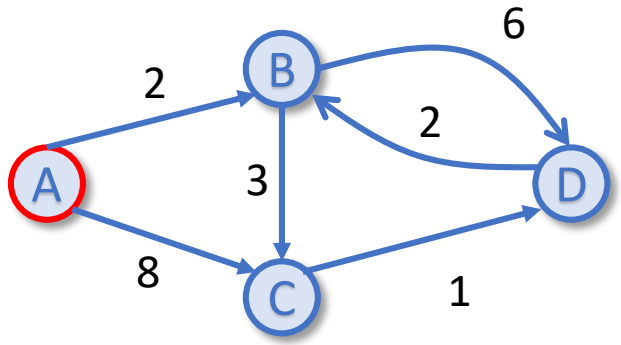
A[1][2] = min(A[1][0] + A[0][2], A[1][2]) = min (∞+2, 3) = 3

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = A → posición = 0



A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

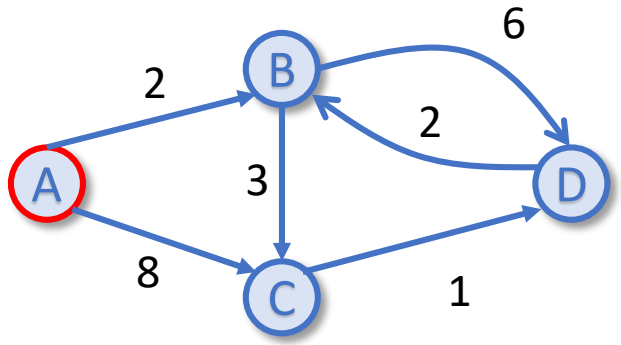
A[1][2] = min(A[1][0] + A[0][2], A[1][2]) = min (∞+2, 3) = 3
A[1][3] = min(A[1][0] + A[0][3], A[1][3]) = min (∞+∞, 6) = 6

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = A → posición = 0



A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[1][2] = min(A[1][0] + A[0][2], A[1][2]) = min (∞+2, 3) = 3
A[1][3] = min(A[1][0] + A[0][3], A[1][3]) = min (∞+∞, 6) = 6

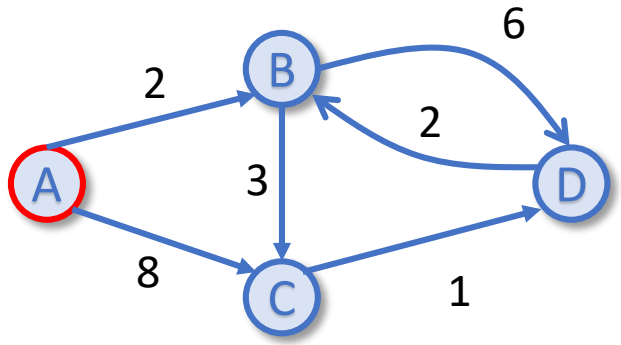
A[2][1] = min(A[2][0] + A[0][1], A[2][1]) = min (∞+2, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = A → posición = 0



A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[1][2] = min(A[1][0] + A[0][2], A[1][2]) = min (∞+2, 3) = 3
A[1][3] = min(A[1][0] + A[0][3], A[1][3]) = min (∞+∞, 6) = 6

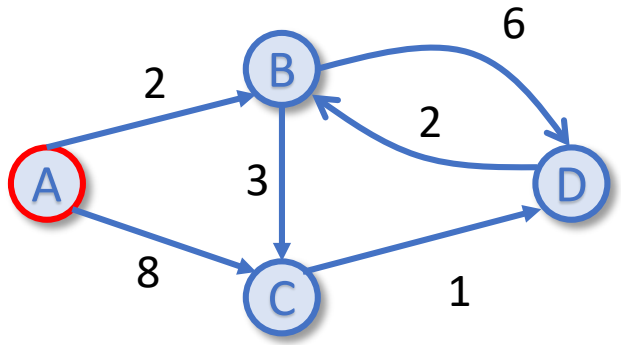
A[2][1] = min(A[2][0] + A[0][1], A[2][1]) = min (∞+2, ∞) = ∞
A[2][3] = min(A[2][0] + A[0][3], A[2][3]) = min (∞+∞, 1) = 1

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = A → posición = 0



A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[1][2] = min(A[1][0] + A[0][2], A[1][2]) = min (∞+2, 3) = 3
A[1][3] = min(A[1][0] + A[0][3], A[1][3]) = min (∞+∞, 6) = 6

A[2][1] = min(A[2][0] + A[0][1], A[2][1]) = min (∞+2, ∞) = ∞
A[2][3] = min(A[2][0] + A[0][3], A[2][3]) = min (∞+∞, 1) = 1

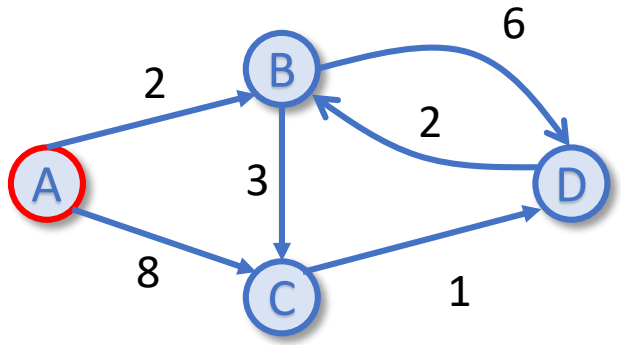
A[3][1] = min(A[3][0] + A[0][1], A[3][1]) = min (∞+2, 2) = 2

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = A → posición = 0



A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[1][2] = min(A[1][0] + A[0][2], A[1][2]) = min (∞+2, 3) = 3
A[1][3] = min(A[1][0] + A[0][3], A[1][3]) = min (∞+∞, 6) = 6

A[2][1] = min(A[2][0] + A[0][1], A[2][1]) = min (∞+2, ∞) = ∞
A[2][3] = min(A[2][0] + A[0][3], A[2][3]) = min (∞+∞, 1) = 1

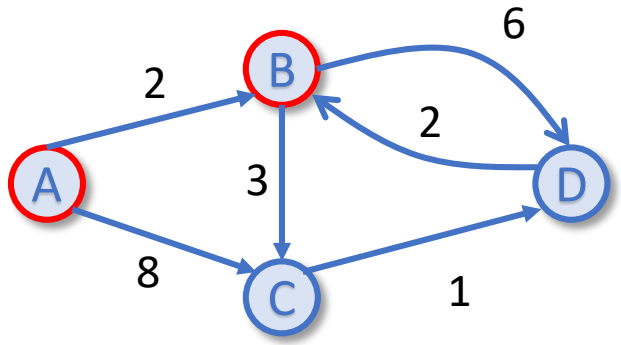
A[3][1] = min(A[3][0] + A[0][1], A[3][1]) = min (∞+2, 2) = 2
A[3][2] = min(A[3][0] + A[0][2], A[3][2]) = min (∞+8, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	8	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	-	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

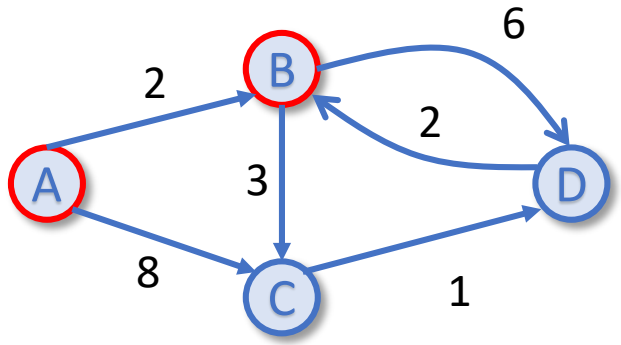
A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	**5**	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	**B**	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

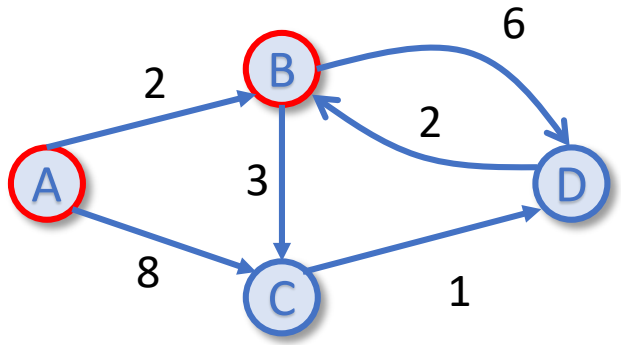
A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	∞
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	B	-
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

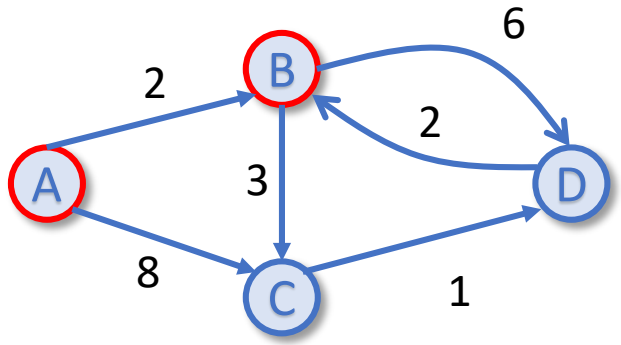
A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	**8**
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	B	**B**
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

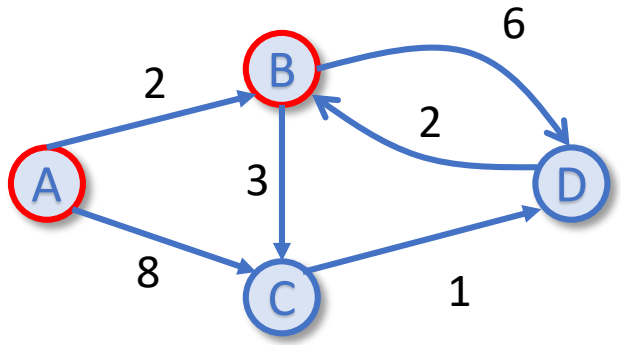
A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

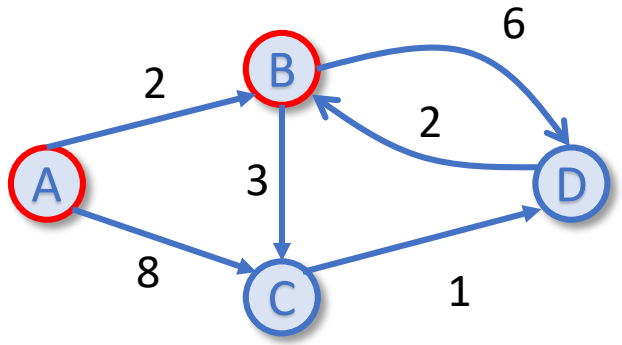
A[2][0] = min(A[2][1] + A[1][0], A[2][0]) = min (∞+∞, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

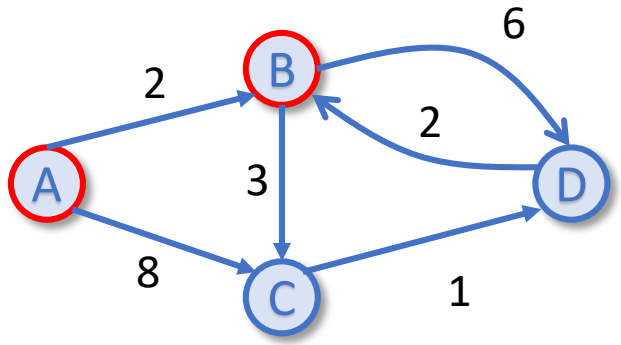
A[2][0] = min(A[2][1] + A[1][0], A[2][0]) = min (∞+∞, ∞) = ∞
A[3][0] = min(A[3][1] + A[1][0], A[3][0]) = min (2+∞, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

A[2][0] = min(A[2][1] + A[1][0], A[2][0]) = min (∞+∞, ∞) = ∞
A[3][0] = min(A[3][1] + A[1][0], A[3][0]) = min (2+∞, ∞) = ∞

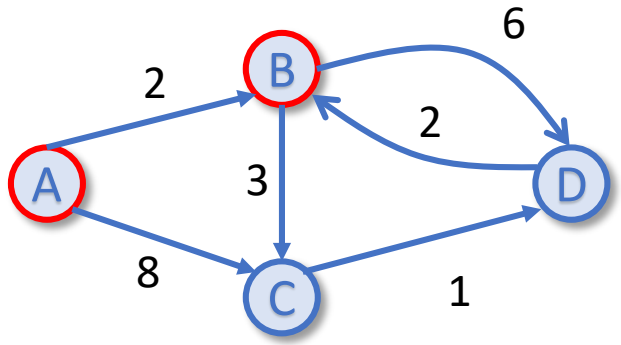
A[2][3] = min(A[2][1] + A[1][3] , A[2][3]) = min (∞+6, 1) = 1

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	∞	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	-	-

A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

A[2][0] = min(A[2][1] + A[1][0], A[2][0]) = min (∞+∞, ∞) = ∞
A[3][0] = min(A[3][1] + A[1][0], A[3][0]) = min (2+∞, ∞) = ∞

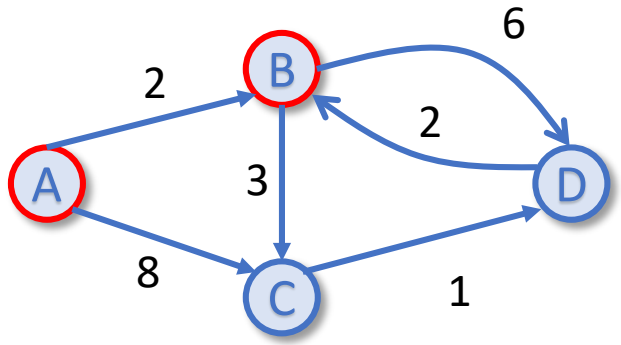
A[2][3] = min(A[2][1] + A[1][3] , A[2][3]) = min (∞+6, 1) = 1
A[3][2] = min(A[3][1] + A[1][2], A[3][2]) = min (2+3, ∞) = 5

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = B → posición = 1



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	**5**	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	**B**	-

A[0][2] = min(A[0][1] + A[1][2], A[0][2]) = min (2+3, 8) = 5
A[0][3] = min(A[0][1] + A[1][3], A[0][3]) = min (2+6, ∞) = 8

A[2][0] = min(A[2][1] + A[1][0], A[2][0]) = min (∞+∞, ∞) = ∞
A[3][0] = min(A[3][1] + A[1][0], A[3][0]) = min (2+∞, ∞) = ∞

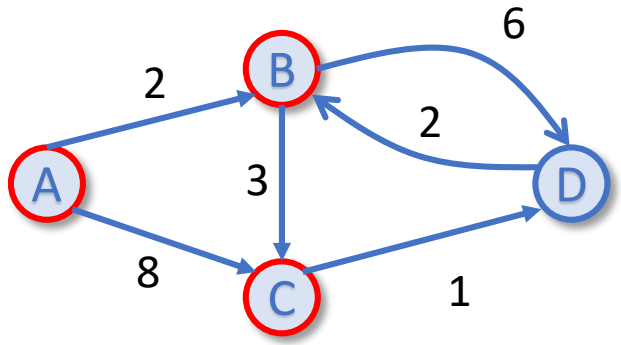
A[2][3] = min(A[2][1] + A[1][3] , A[2][3]) = min (∞+6, 1) = 1
A[3][2] = min(A[3][1] + A[1][2], A[3][2]) = min (2+3, ∞) = 5

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

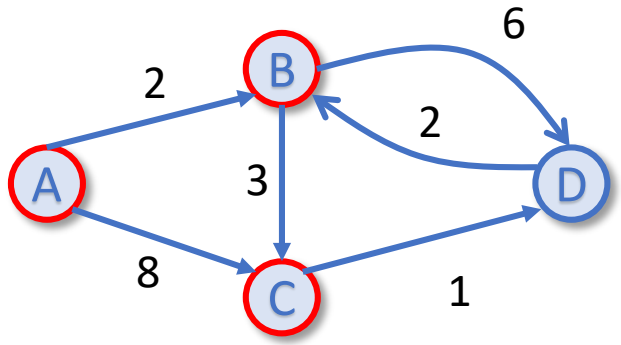
A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

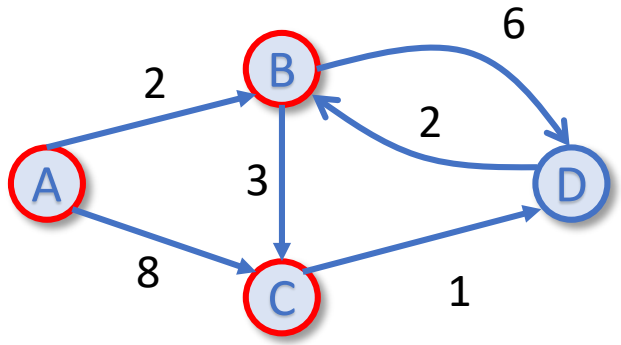
A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

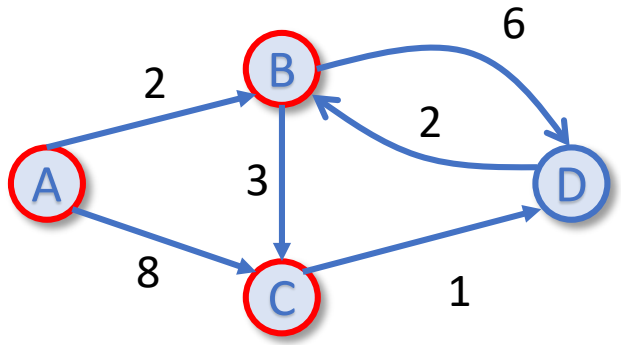
A[3][0] = min(A[3][2] + A[2][0], A[3][0]) = min (5+∞, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

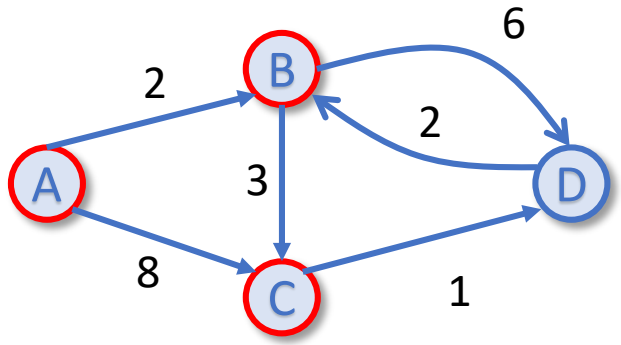
A[3][0] = min(A[3][2] + A[2][0], A[3][0]) = min (5+∞, ∞) = ∞
A[3][1] = min(A[3][2] + A[2][1], A[3][0]) = min (5+∞, 2) = 2

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	8
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	B
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

A[3][0] = min(A[3][2] + A[2][0], A[3][0]) = min (5+∞, ∞) = ∞
A[3][1] = min(A[3][2] + A[2][1], A[3][0]) = min (5+∞, 2) = 2

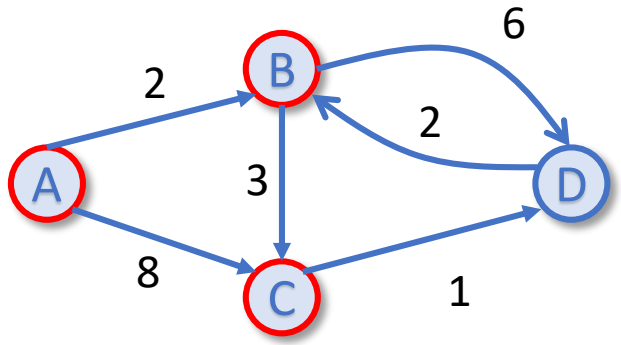
A[0][3] = min(A[0][2] + A[2][3], A[0][3]) = min (5+1, 8) = 6

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	**6**
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	**C**
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

A[3][0] = min(A[3][2] + A[2][0], A[3][0]) = min (5+∞, ∞) = ∞
A[3][1] = min(A[3][2] + A[2][1], A[3][0]) = min (5+∞, 2) = 2

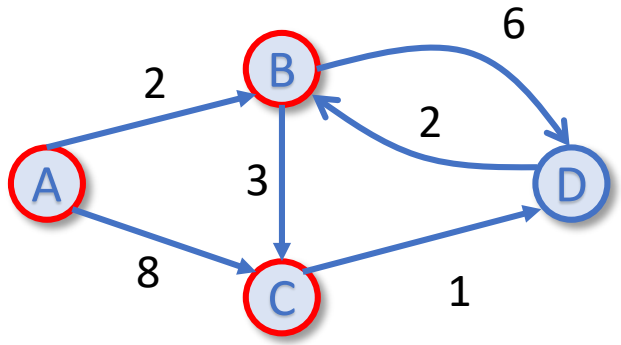
A[0][3] = min(A[0][2] + A[2][3], A[0][3]) = min (5+1, 8) = 6

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	6
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	-
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

A[3][0] = min(A[3][2] + A[2][0], A[3][0]) = min (5+∞, ∞) = ∞
A[3][1] = min(A[3][2] + A[2][1], A[3][0]) = min (5+∞, 2) = 2

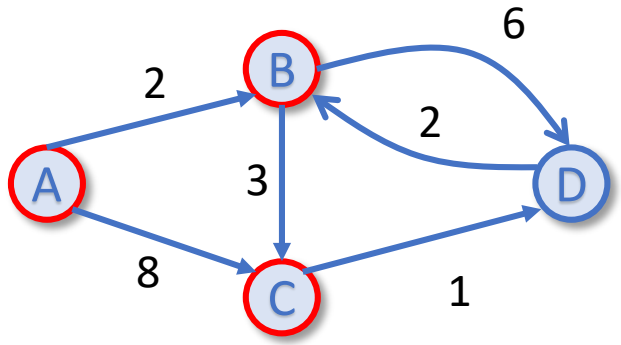
A[0][3] = min(A[0][2] + A[2][3], A[0][3]) = min (5+1, 8) = 6
A[1][3] = min(A[1][2] + A[2][3], A[1][3]) = min (3+1, 6) = 4

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = C → posición = 2



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	**4**
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	**C**
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][2] + A[2][1], A[0][1]) = min (5+∞, 2) = 2
A[1][0] = min(A[1][2] + A[2][0], A[0][1]) = min (3+∞, ∞) = ∞

A[3][0] = min(A[3][2] + A[2][0], A[3][0]) = min (5+∞, ∞) = ∞
A[3][1] = min(A[3][2] + A[2][1], A[3][0]) = min (5+∞, 2) = 2

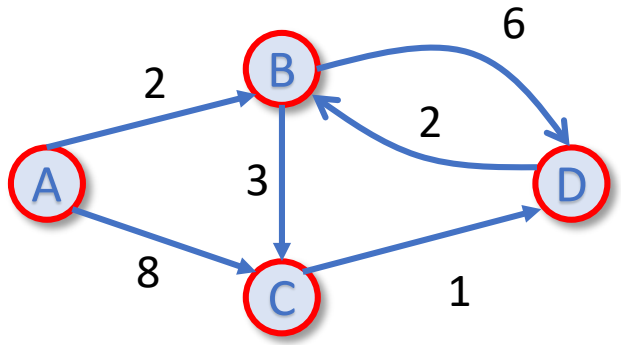
A[0][3] = min(A[0][2] + A[2][3], A[0][3]) = min (5+1, 8) = 6
A[1][3] = min(A[1][2] + A[2][3], A[1][3]) = min (3+1, 6) = 4

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	-	-	-
D	-	-	B	-

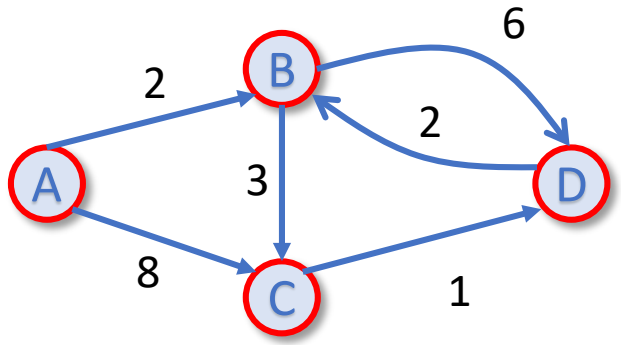
A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	-	-	-
D	-	-	B	-

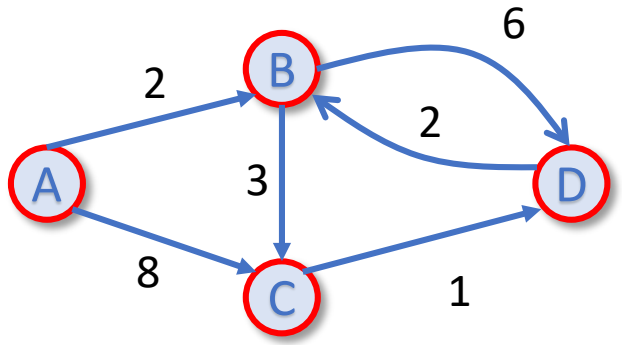
A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2
A[0][2] = min(A[0][3] + A[3][2], A[0][2]) = min (6+5, 5) = 5

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2
A[0][2] = min(A[0][3] + A[3][2], A[0][2]) = min (6+∞, 5) = 5

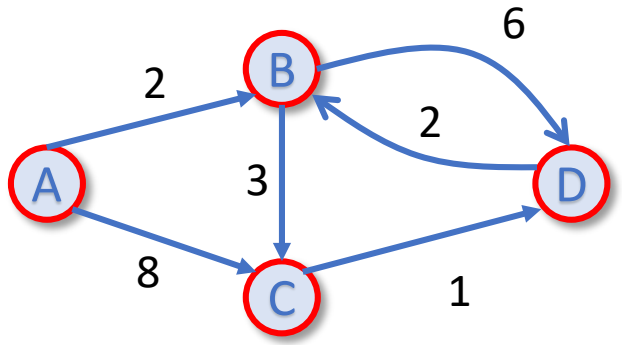
A[1][0] = min(A[1][3] + A[3][0], A[1][0]) = min (4+∞, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2
A[0][2] = min(A[0][3] + A[3][2], A[0][2]) = min (6+∞, 5) = 5

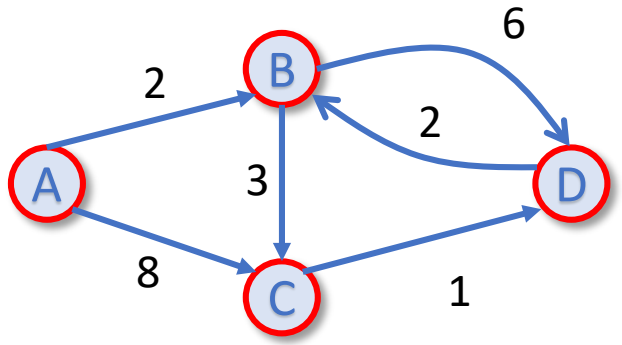
A[1][0] = min(A[1][3] + A[3][0], A[1][0]) = min (4+∞, ∞) = ∞
A[1][2] = min(A[1][3] + A[3][2], A[1][2]) = min (4+5, 3) = 3

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2
A[0][2] = min(A[0][3] + A[3][2], A[0][2]) = min (6+∞, 5) = 5

A[1][0] = min(A[1][3] + A[3][0], A[1][0]) = min (4+∞, ∞) = ∞
A[1][2] = min(A[1][3] + A[3][2], A[1][2]) = min (4+∞, 3) = 3

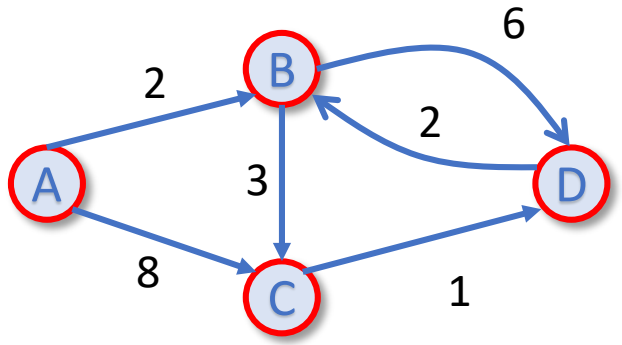
A[2][0] = min(A[2][3] + A[3][0], A[2][0]) = min (1+∞, ∞) = ∞

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	∞	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	-	-	-
D	-	-	B	-

A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2
A[0][2] = min(A[0][3] + A[3][2], A[0][2]) = min (6+∞, 5) = 5

A[1][0] = min(A[1][3] + A[3][0], A[1][0]) = min (4+∞, ∞) = ∞
A[1][2] = min(A[1][3] + A[3][2], A[1][2]) = min (4+∞, 3) = 3

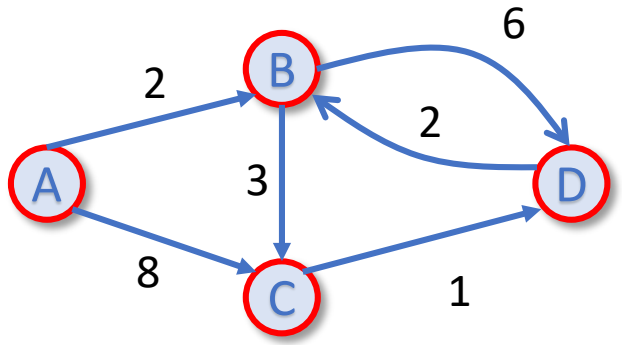
A[2][0] = min(A[2][3] + A[3][0], A[2][0]) = min (1+∞, ∞) = ∞
A[2][1] = min(A[2][3] + A[3][1], A[2][0]) = min (1+2, ∞) = 3

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

Paso2: Para cada v ∈ nodos hacer A[i][j] = min(A[i][v] + A[v][j], A[i][j])

v = D → posición = 3



A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	**3**	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	**D**	-	-
D	-	-	B	-

A[0][1] = min(A[0][3] + A[3][1], A[0][1]) = min (6+2, 2) = 2
A[0][2] = min(A[0][3] + A[3][2], A[0][2]) = min (6+∞, 5) = 5

A[1][0] = min(A[1][3] + A[3][0], A[1][0]) = min (4+∞, ∞) = ∞
A[1][2] = min(A[1][3] + A[3][2], A[1][2]) = min (4+∞, 3) = 3

A[2][0] = min(A[2][3] + A[3][0], A[2][0]) = min (1+∞, ∞) = ∞
A[2][1] = min(A[2][3] + A[3][1], A[2][0]) = min (1+2, ∞) = 3

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

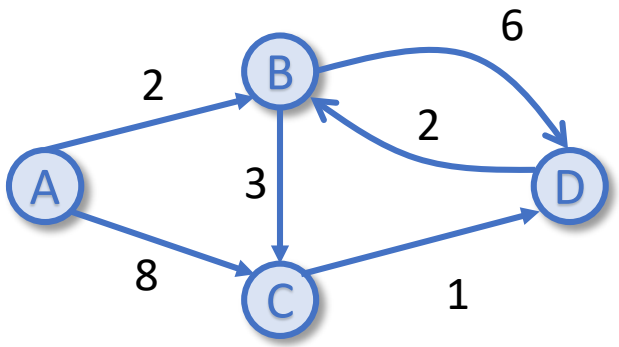
Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-



Para cualquier par de nodos se obtiene el coste del camino mínimo
Con la matriz P se obtiene el camino de coste mínimo para ir de un nodo a otro

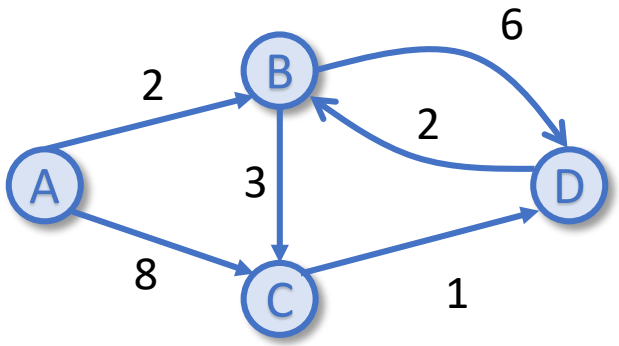
Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D

Camino (**A**→D)



Camino: A

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

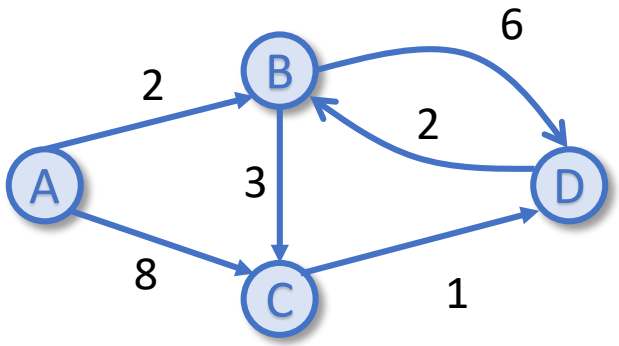
	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D

Camino (**A**→D)

↓

C



<u>Camino</u>: A

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D

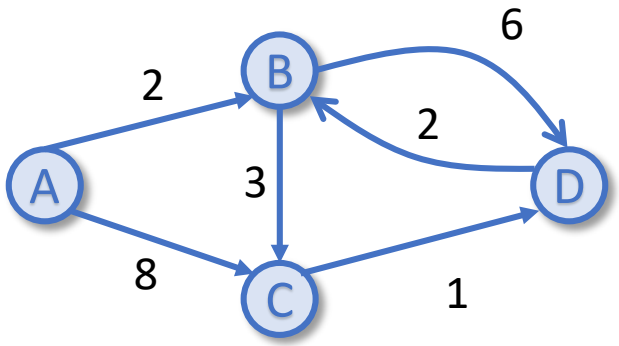
Camino (A→D)

↓

C

↙

Camino (A→C)



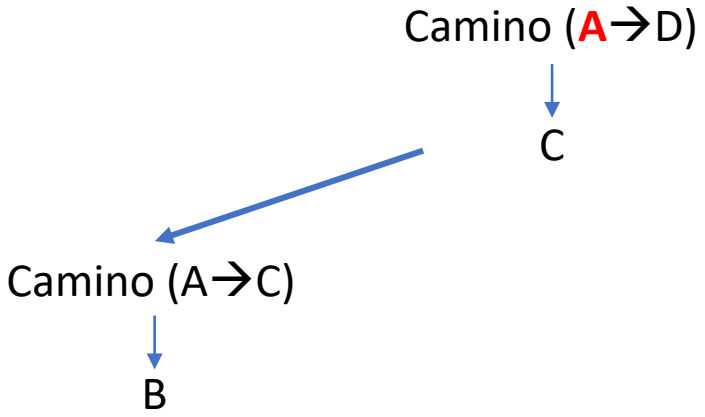
Camino: A

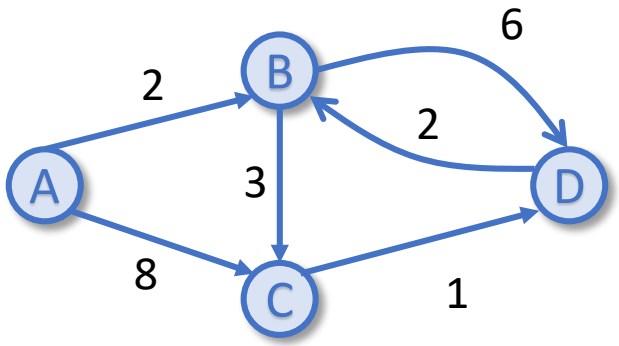
Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D





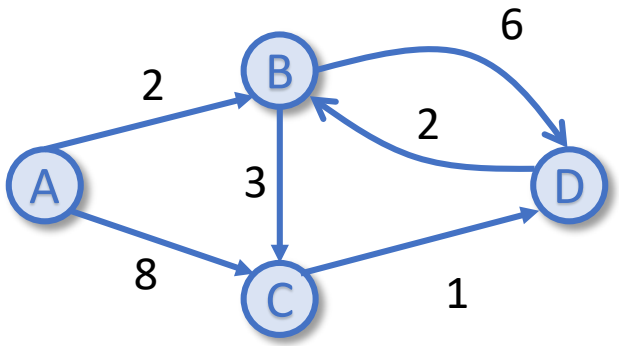
Camino: A

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (A→D)

C

Camino (A→C)

B

Camino (A→B)

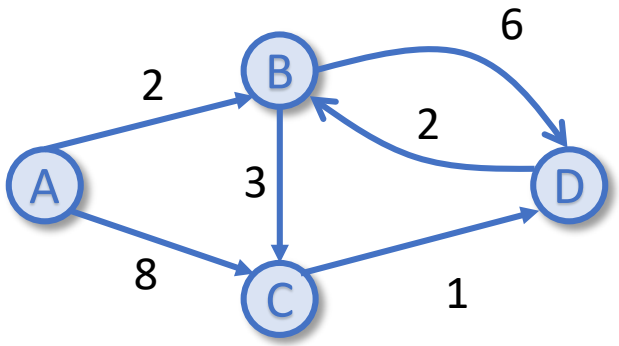
Camino: A

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

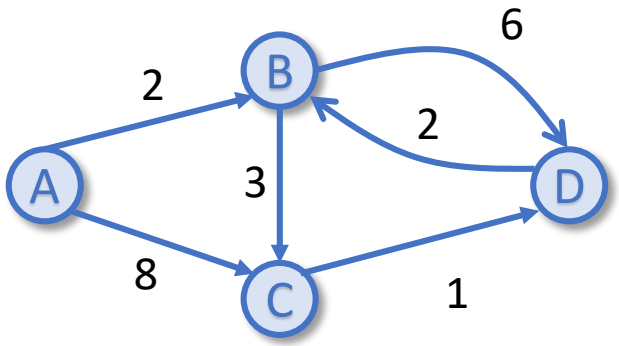
Camino: A

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

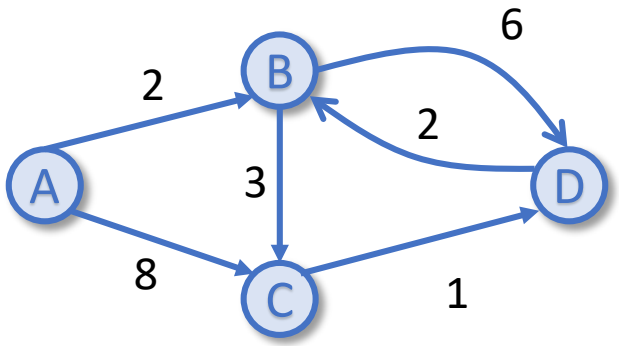
Camino: A B

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C)

↓

B

Camino (A→B) → **null**

Camino (B→C)

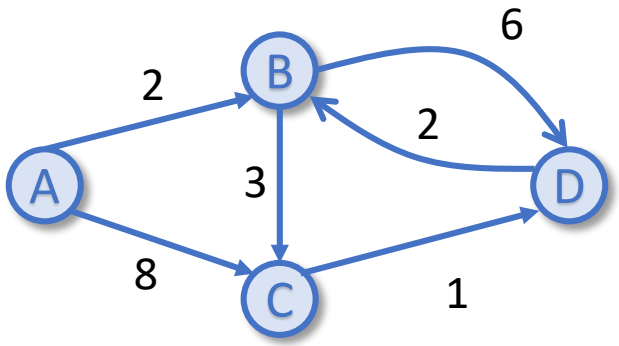
Camino: A B

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

Camino (B→C)

↓

null

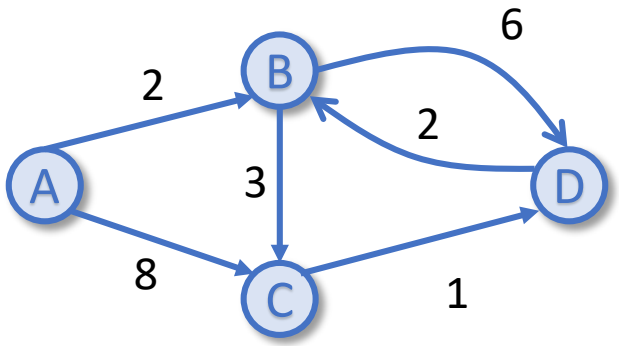
Camino: A B

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

Camino (B→C)

↓

null

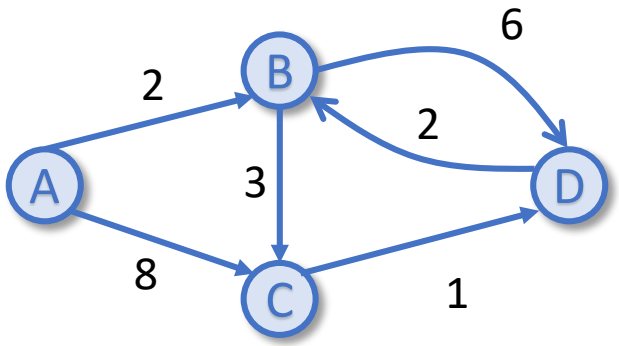
Camino: A B C

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C) — Camino (C→D)

↓

B

Camino (A→B) — Camino (B→C)

↓ ↓

null **null**

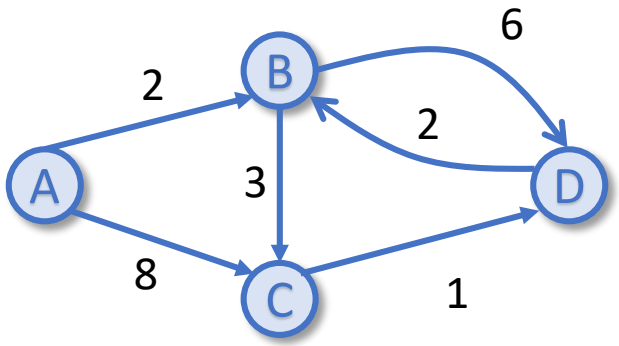
Camino: A B C

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→D)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

Camino (B→C)

↓

null

Camino (C→D)

↓

null

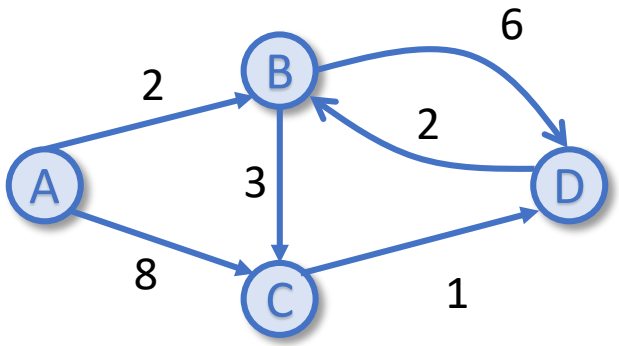
Camino: A B C

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→**D**)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

Camino (B→C)

↓

null

Camino (C→D)

↓

null

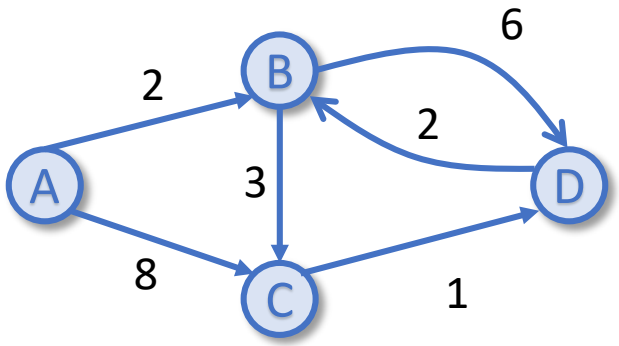
Camino: A B C D

Grafos – Algoritmo de Floyd-Warshall - Ejemplo

P

	A	B	C	D
A	-	-	B	C
B	-	-	-	C
C	-	D	-	-
D	-	-	B	-

Camino de coste mínimo para ir de A a D



Camino (**A**→**D**)

↓

C

Camino (A→C)

↓

B

Camino (A→B)

↓

null

Camino (B→C)

↓

null

Camino (C→D)

↓

null

A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0

Camino: A B C D **Coste**: 6

Grafos – Algoritmo de Floyd-Warshall - Resumiendo

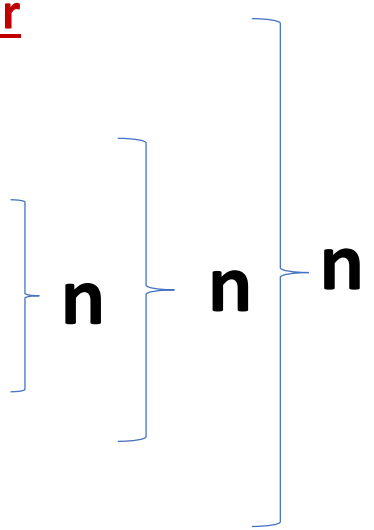
Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

```
A → Matriz de pesos con la diagonal principal a cero
P → Vector de rutas de coste mínimo
mientras  no se hayan tratado todos los nodos hacer
        Seleccionar el nodo pivote → w
        para cada nodo en la fila (i) hacer
                para cada nodo en la columna (j) hacer
                    Actualizar A si se mejora el coste
                    Actualizar P si se mejora el coste
                fin_para
        fin_para
fin_mientras
```

Grafos – Algoritmo de Floyd-Warshall - Resumiendo

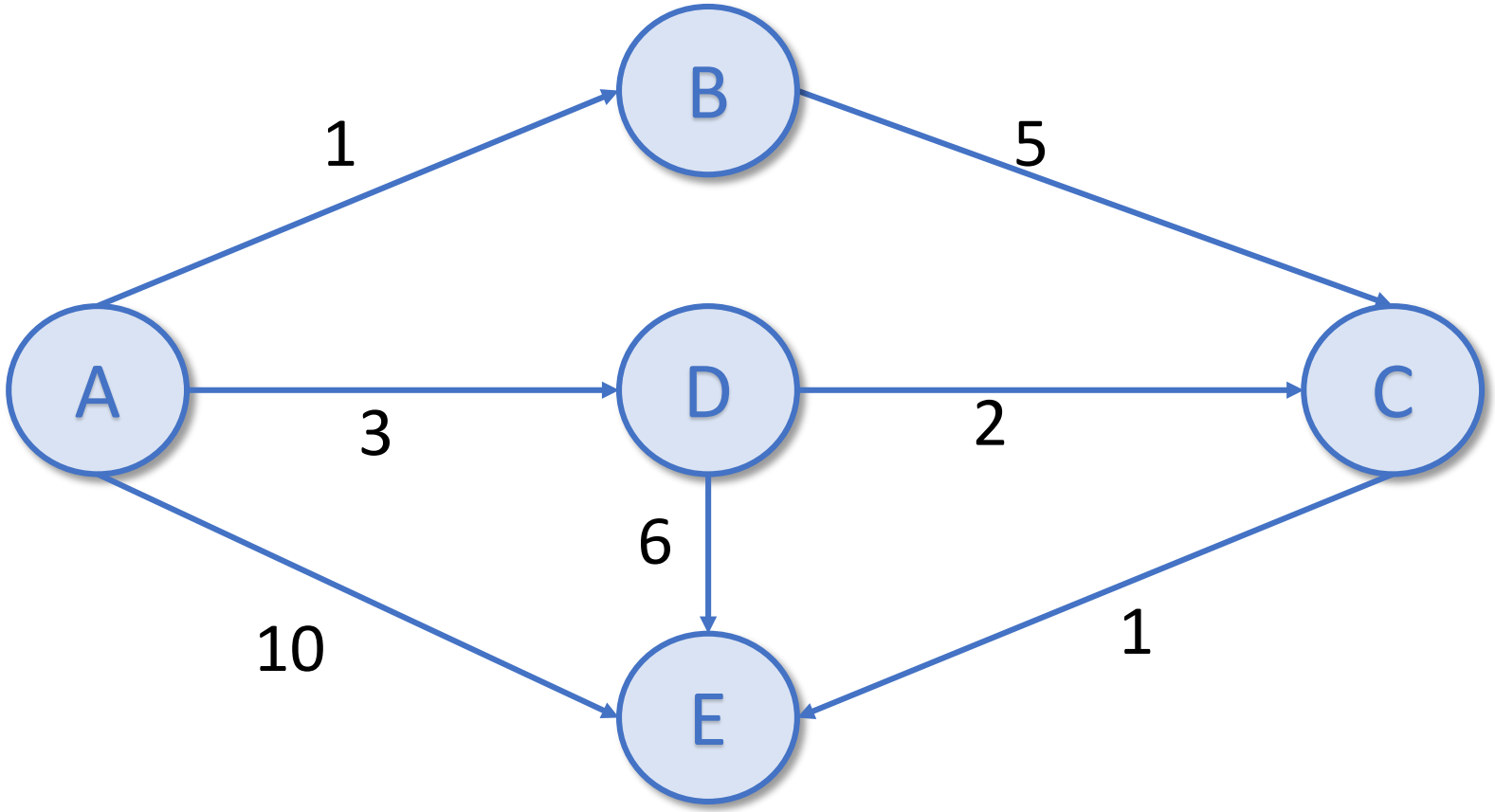
Encontrar el camino de coste mínimo desde cualquier nodo al resto de los nodos

```
A → Matriz de pesos con la diagonal principal a cero
P → Vector de rutas de coste mínimo
mientras  no se hayan tratado todos los nodos hacer
        Seleccionar el nodo pivote → w
        para cada nodo en la fila (i) hacer
                para cada nodo en la columna (j) hacer
                        Actualizar A si se mejora el coste
                        Actualizar P si se mejora el coste
                fin_para
        fin_para
fin_mientras
```



$O(n^3)$

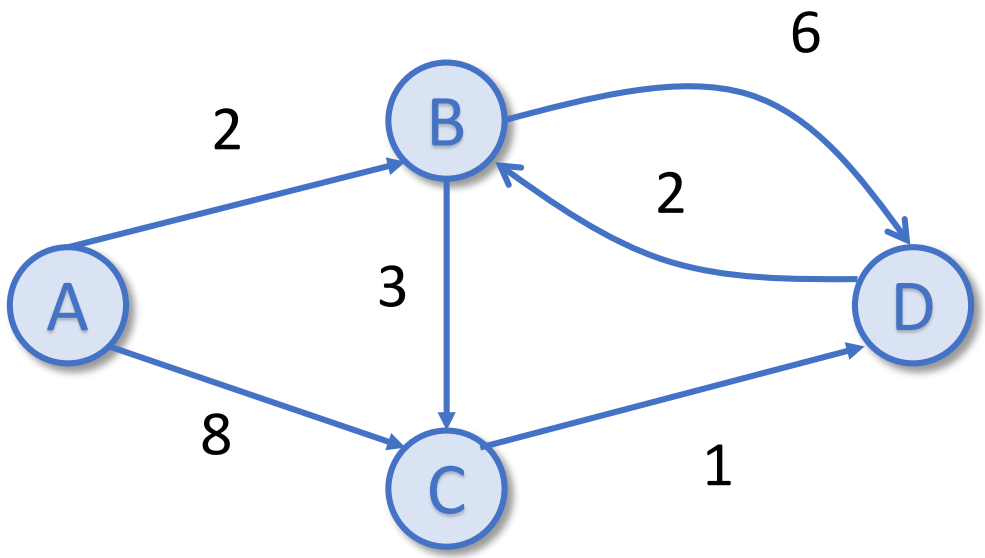
Ejercicio – Aplicar Floyd-Warshall



Grafos –Excentricidad y centro de un grafo

- Se aplica Floyd → distancias mínimas de cada nodo al resto
- Excentricidad
  - La excentricidad de un vértice v es la distancia desde dicho vértice al mas lejano
    - ***exc(v) = max{d(s,v)}*** tal que s pertenece al conjunto de vértices
- Centro de un grafo
  - Es el nodo de mínima excentricidad
- Utilidad
  - ¿Dónde ubicamos un hospital en una determinada región?

Grafos –Excentricidad y centro de un grafo



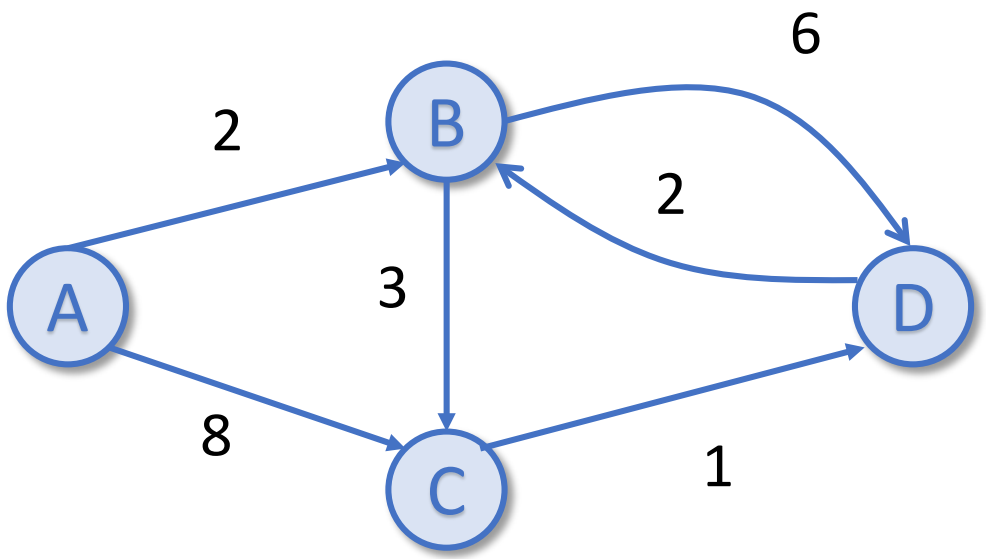
A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0

Máximo				

Grafos –Excentricidad y centro de un grafo

Se selecciona la distancia mas lejana al nodo A



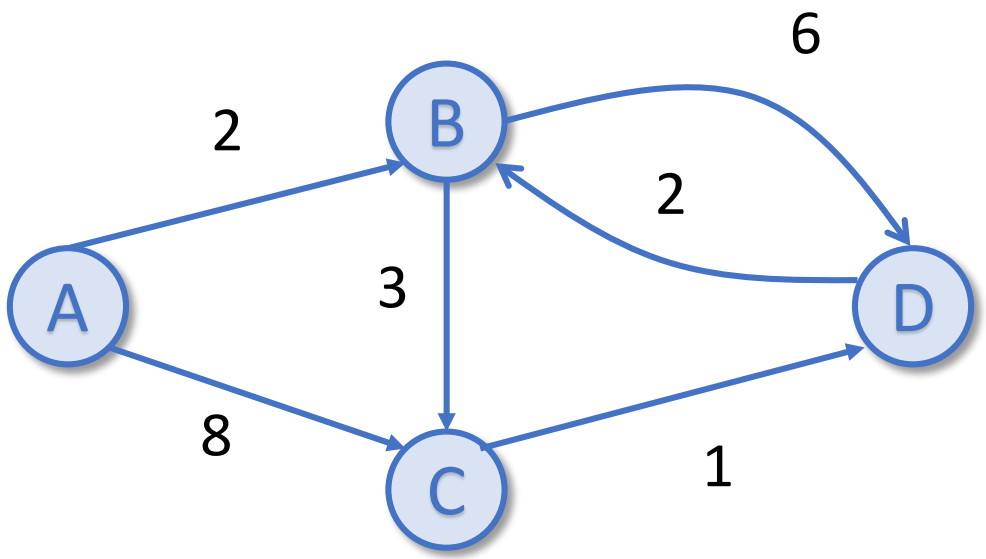
A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0

Máximo	∞			

Grafos –Excentricidad y centro de un grafo

Se selecciona la distancia mas lejana al nodo B



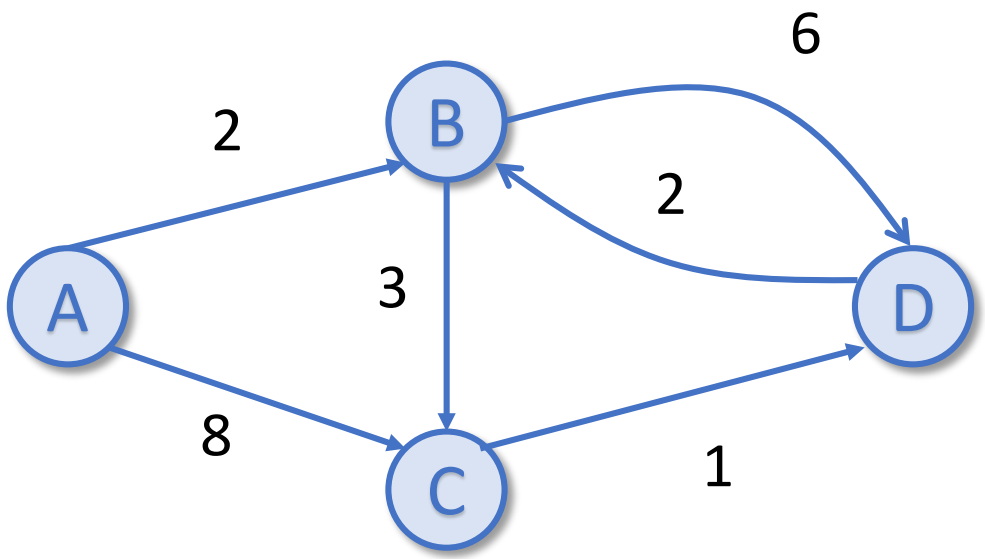
A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	**3**	0	1
D	∞	2	5	0

Máximo	∞	3		

Grafos –Excentricidad y centro de un grafo

Se selecciona la distancia mas lejana al nodo C



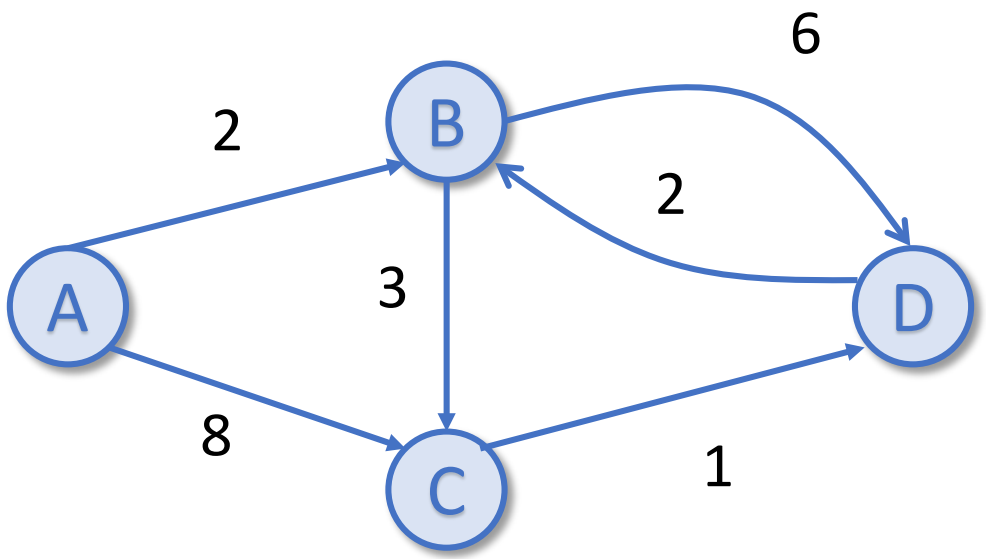
A

	A	B	C	D
A	0	2	**5**	6
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0

Máximo	∞	3	5	

Grafos –Excentricidad y centro de un grafo

Se selecciona la distancia mas lejana al nodo D

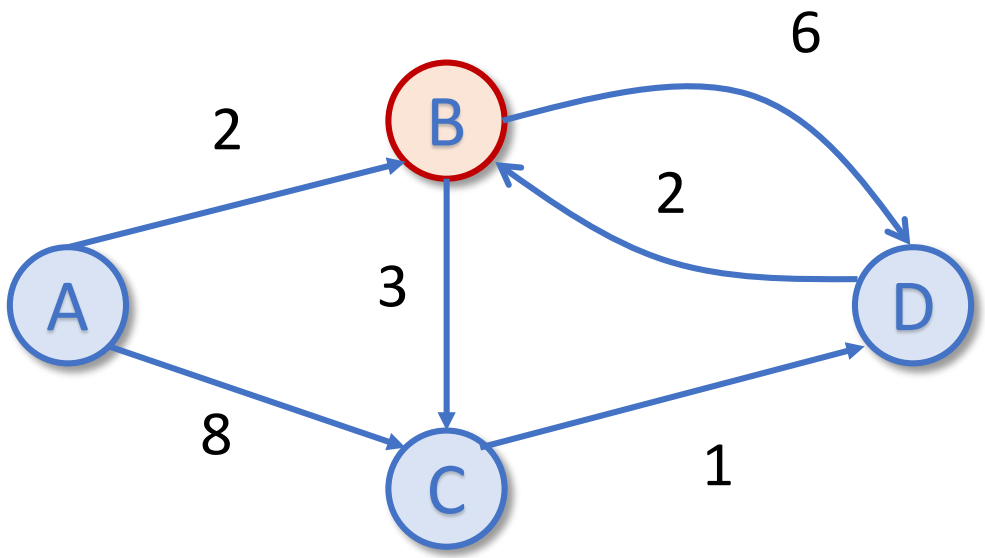


A

	A	B	C	D
A	0	2	5	**6**
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0

Máximo	∞	3	5	6

Grafos –Excentricidad y centro de un grafo

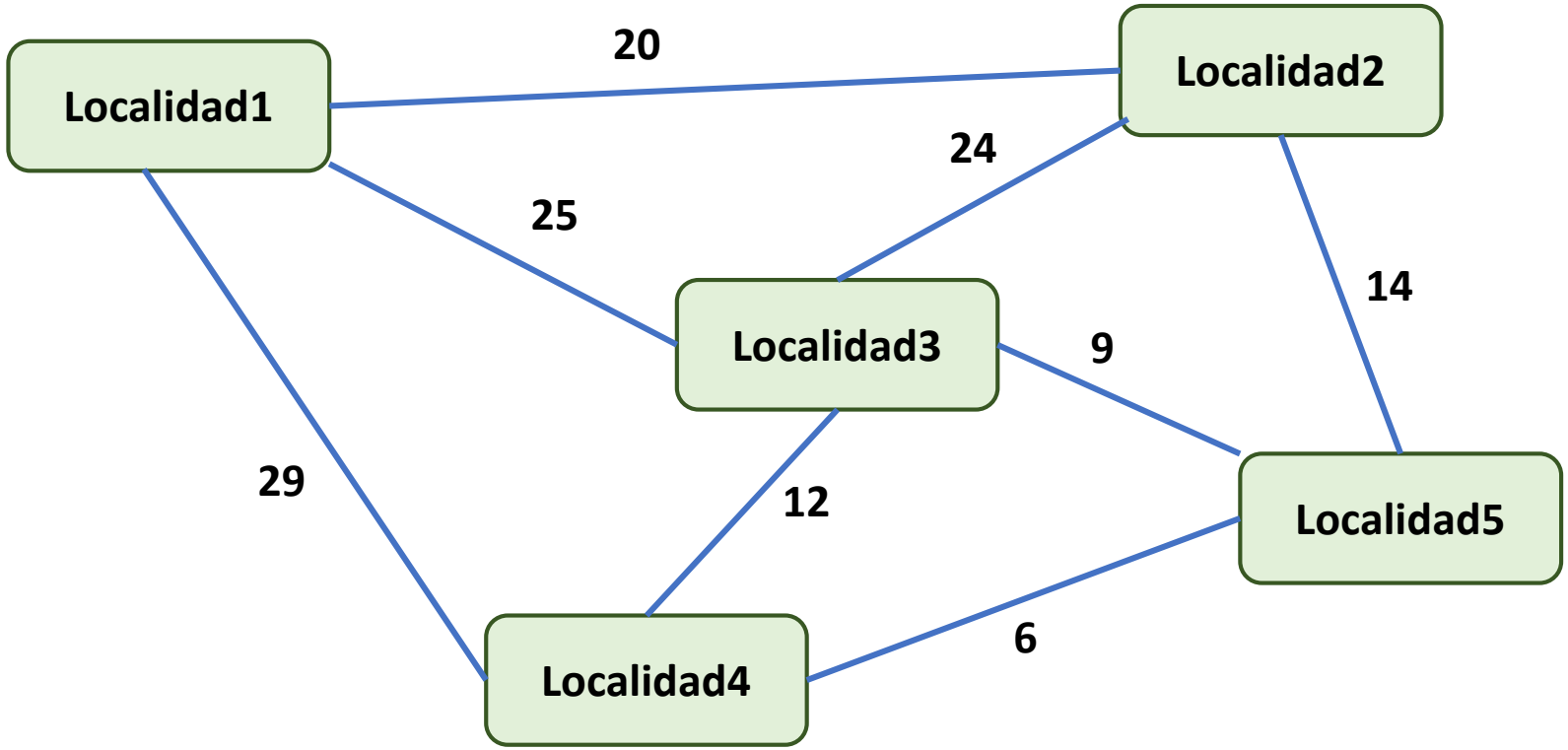


A

	A	B	C	D
A	0	2	5	6
B	∞	0	3	4
C	∞	3	0	1
D	∞	2	5	0
Máximo	∞	**3**	5	6

Se selecciona el nodo con la excentricidad mínima

Grafos –Excentricidad y centro de un grafo



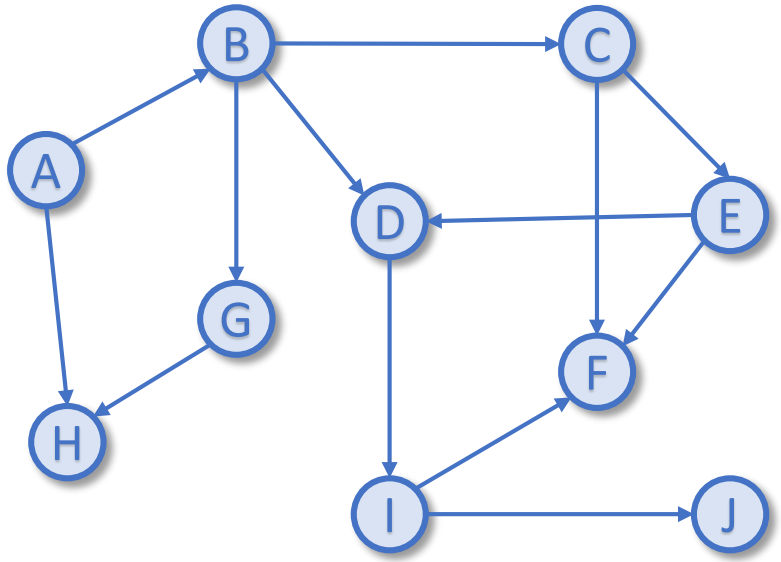
Grafos – Recorrido en profundidad (DFPrint)

- Algoritmo que permite recorrer todos los nodos de un grafo
- Visitar primero los hijos y luego los hermanos
- Partiendo de un nodo, explorar cada camino que salga de él
- Un camino deja de explorarse cuando se llega a un vértice ya visitado
- Hay que llevar un recuento de los nodos visitados
- Pueden existir nodos inalcanzables
- Complejidad $O(n^2)$
  - Se evalúan todos los nodos y para cada nodo sus hijos

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso1: Inicialización



S = { }
Candidatos = {A}

Visitados

A	B	C	D	E	F	G	H	I	J
F	F	F	F	F	F	F	F	F	F

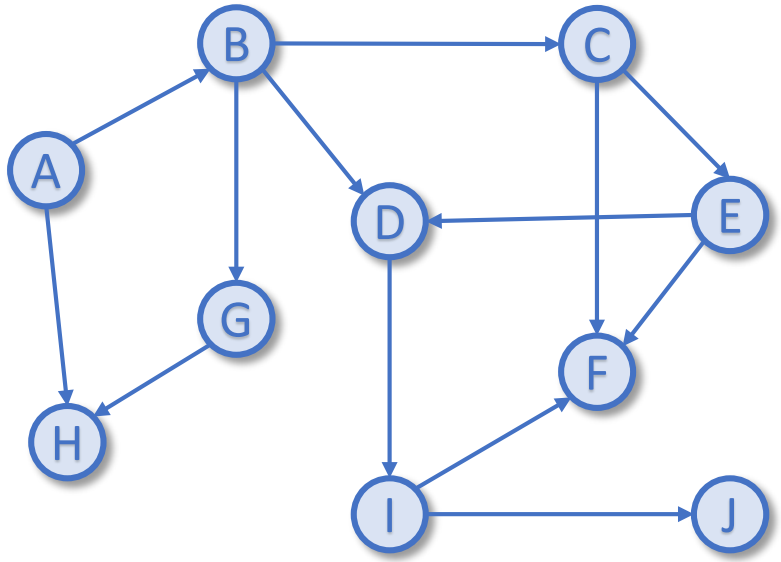
Matriz de Ejes

	A	B	C	D	E	F	G	H	I	J
A	F	T	F	F	F	F	F	T	F	F
B	F	F	T	T	F	F	T	F	F	F
C	F	F	F	F	T	T	F	F	F	F
D	F	F	F	F	F	F	F	F	T	F
E	F	F	F	T	F	T	F	F	F	F
F	F	F	F	F	F	F	F	F	F	F
G	F	F	F	F	F	F	F	T	F	F
H	F	F	F	F	F	F	F	F	F	F
I	F	F	F	F	F	T	F	F	F	T
J	F	F	F	F	F	F	F	F	F	F

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

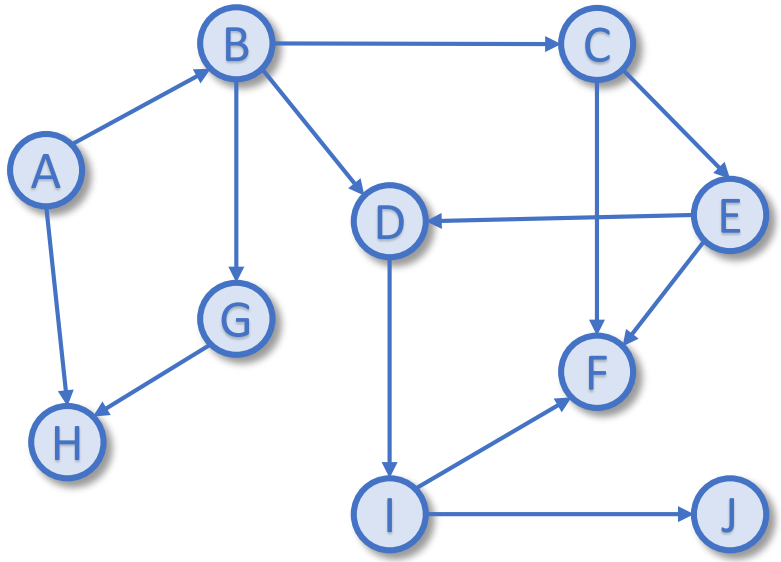
A	B	C	D	E	F	G	H	I	J
F	F	F	F	F	F	F	F	F	F

Iteración	S	Candidatos

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

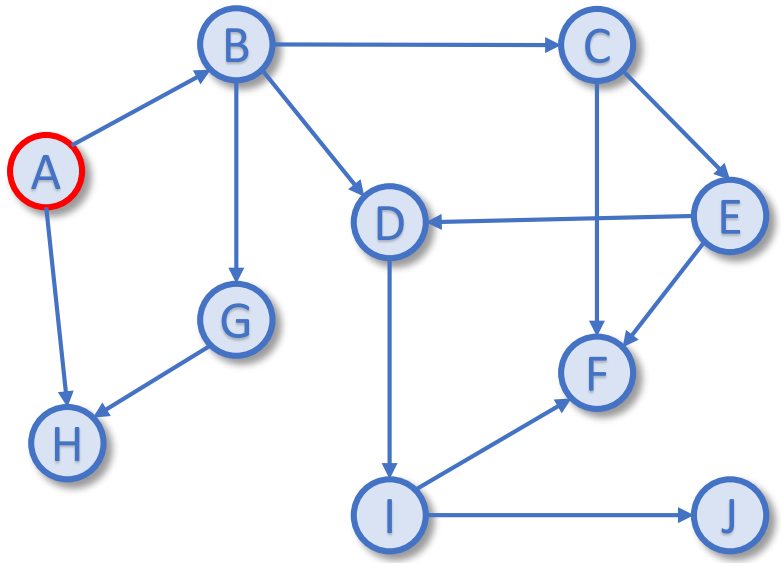
A	B	C	D	E	F	G	H	I	J
F	F	F	F	F	F	F	F	F	F

Iteración	S	Candidatos
1	{ }	{A}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

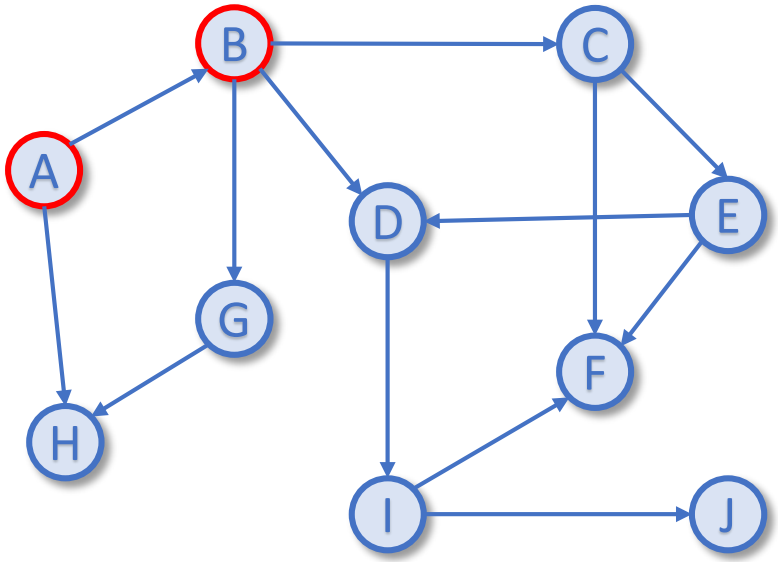
A	B	C	D	E	F	G	H	I	J
T	F	F	F	F	F	F	F	F	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

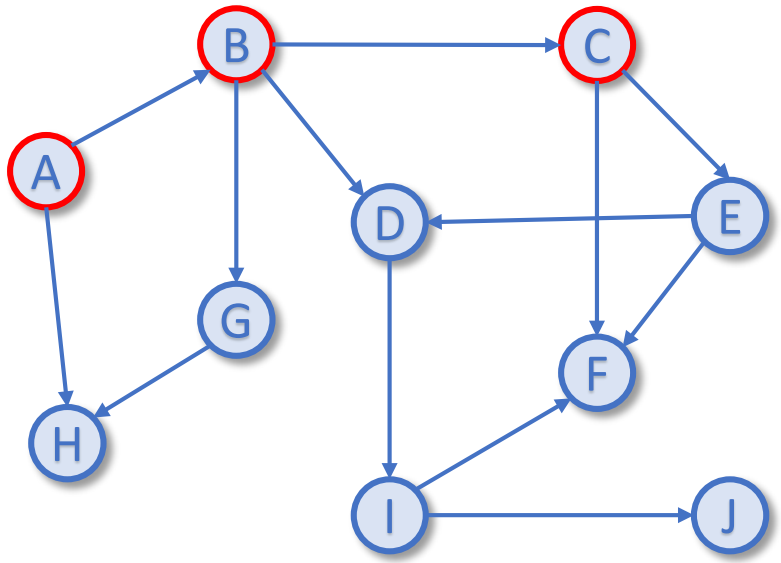
A	B	C	D	E	F	G	H	I	J
T	T	F	F	F	F	F	F	F	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

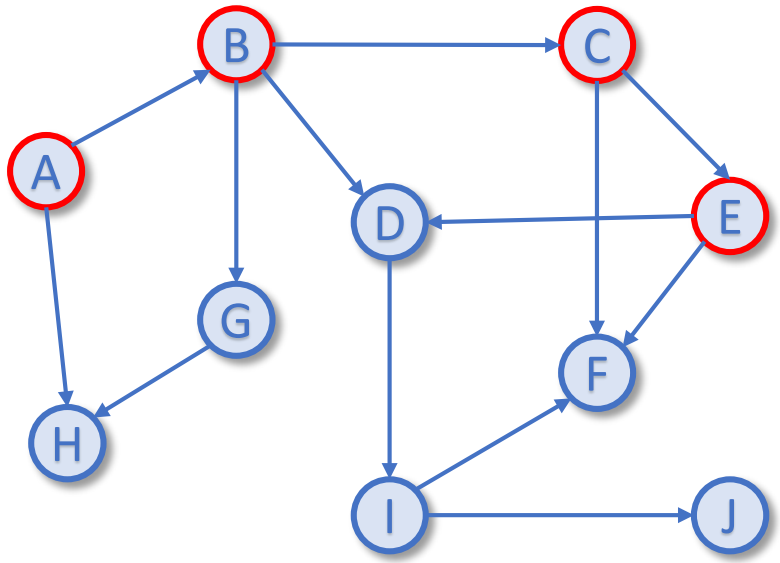
A	B	C	D	E	F	G	H	I	J
T	T	T	F	F	F	F	F	F	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

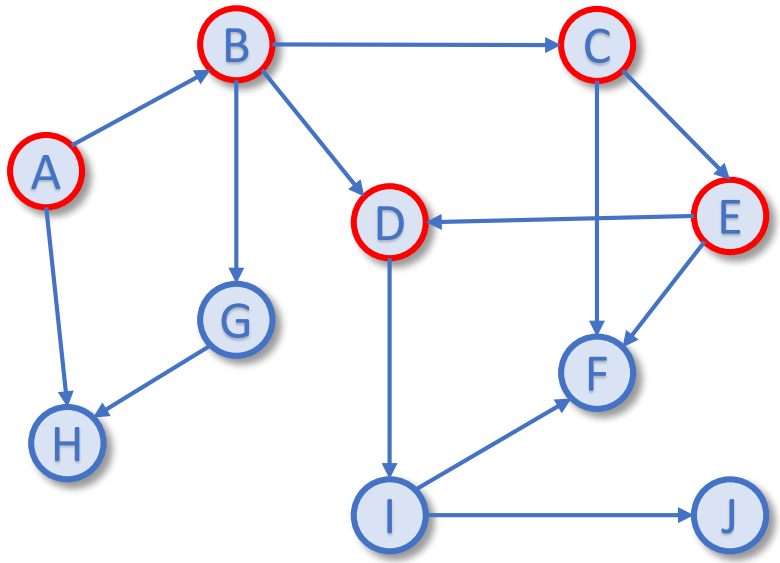
A	B	C	D	E	F	G	H	I	J
T	T	T	F	T	F	F	F	F	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

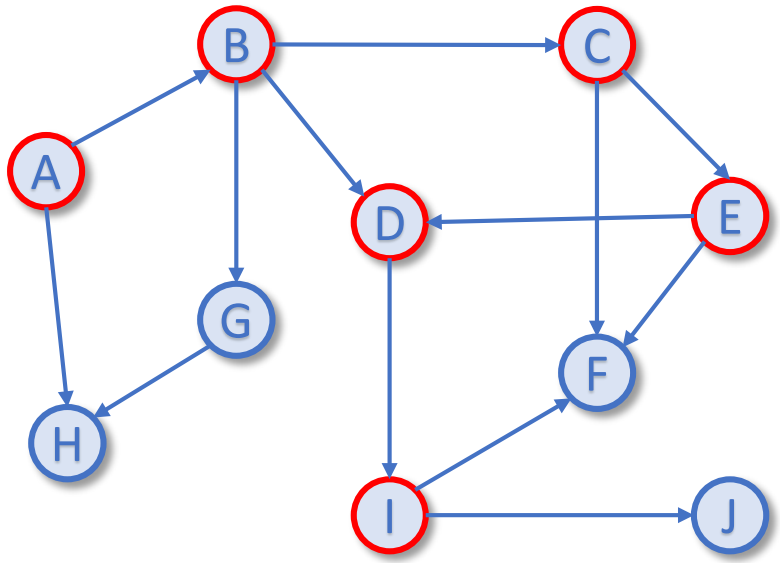
A	B	C	D	E	F	G	H	I	J
T	T	T	T	T	F	F	F	F	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}
6	{A, B, C, E, D}	{I, F, G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

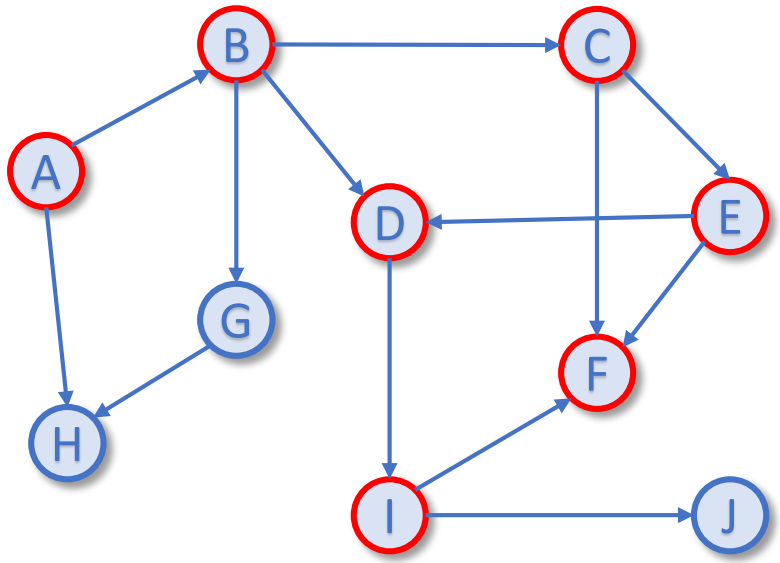
A	B	C	D	E	F	G	H	I	J
T	T	T	T	T	F	F	F	T	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}
6	{A, B, C, E, D}	{I, F, G, H}
7	{A, B, C, E, D, I}	{F, J, G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

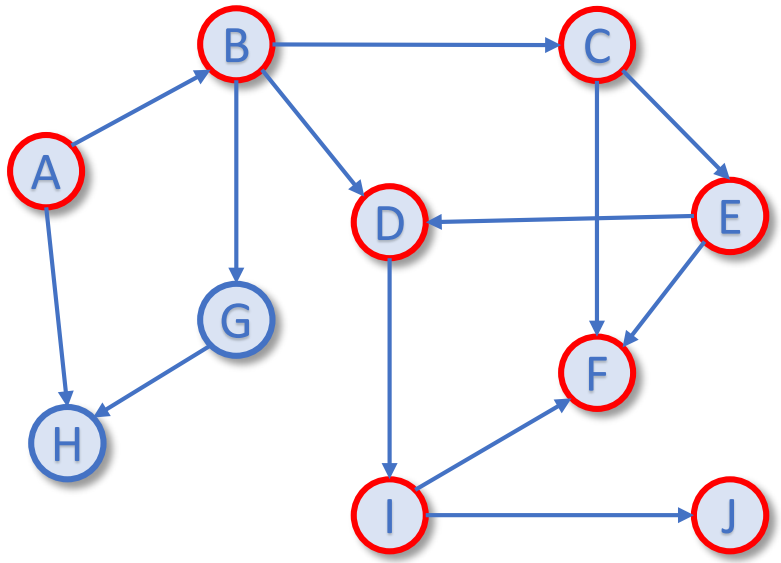
A	B	C	D	E	F	G	H	I	J
T	T	T	T	T	T	F	F	T	F

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}
6	{A, B, C, E, D}	{I, F, G, H}
7	{A, B, C, E, D, I}	{F, J, G, H}
8	{A, B, C, E, D, I, F}	{J, G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

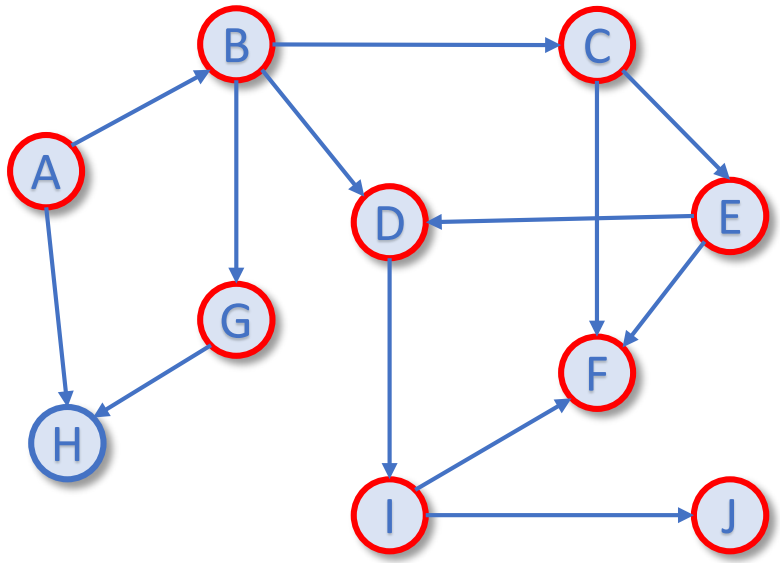
A	B	C	D	E	F	G	H	I	J
T	T	T	T	T	T	F	F	T	T

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}
6	{A, B, C, E, D}	{I, F, G, H}
7	{A, B, C, E, D, I}	{F, J, G, H}
8	{A, B, C, E, D, I, F}	{J, G, H}
9	{A, B, C, E, D, I, F, J}	{G, H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A



Visitados

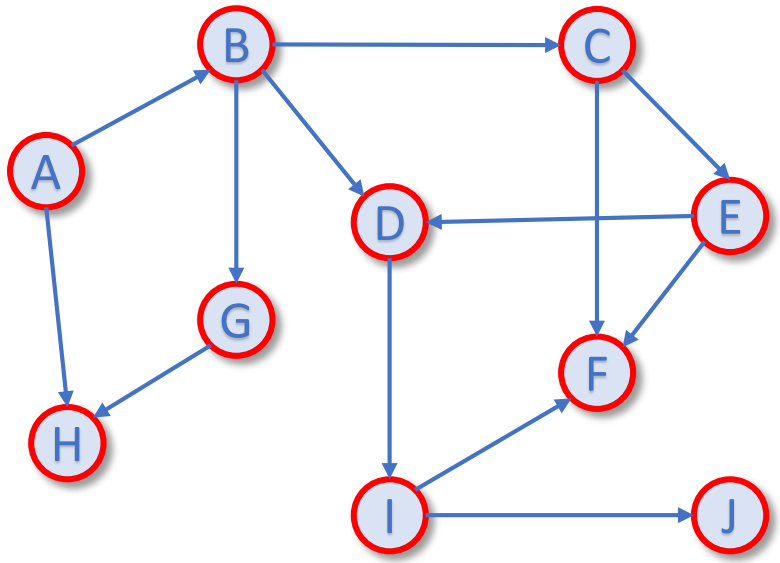
A	B	C	D	E	F	G	H	I	J
T	T	T	T	T	T	T	F	T	T

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}
6	{A, B, C, E, D}	{I, F, G, H}
7	{A, B, C, E, D, I}	{F, J, G, H}
8	{A, B, C, E, D, I, F}	{J, G, H}
9	{A, B, C, E, D, I, F, J}	{G, H}
10	{A, B, C, E, D, I, F, J, G}	{H}

Grafos – Algoritmo DFPrint – Ejemplo

Recorrer todos los nodos de un grafo a partir de un nodo inicial (por ejemplo desde A)

Paso2: Ir visitando los nodos desde A

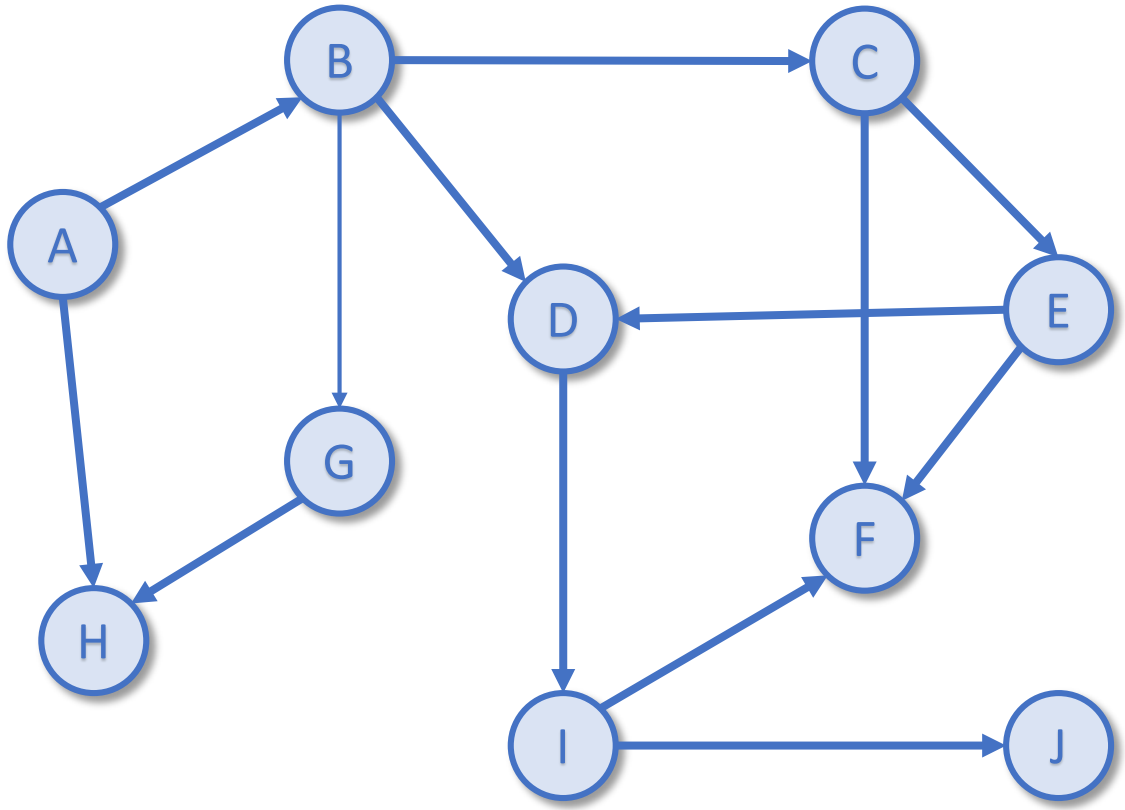


Visitados

A	B	C	D	E	F	G	H	I	J
T	T	T	T	T	T	T	T	T	T

Iteración	S	Candidatos
1	{ }	{A}
2	{A}	{B, H}
3	{A, B}	{C, D, G, H}
4	{A, B, C}	{E, F, D, G, H}
5	{A, B, C, E}	{D, F, G, H}
6	{A, B, C, E, D}	{I, F, G, H}
7	{A, B, C, E, D, I}	{F, J, G, H}
8	{A, B, C, E, D, I, F}	{J, G, H}
9	{A, B, C, E, D, I, F, J}	{G, H}
10	{A, B, C, E, D, I, F, J, G}	{H}
11	{A, B, C, E, D, I, F, J, G, H}	{ }

Ejercicio – Aplicar DFPrint desde el nodo D

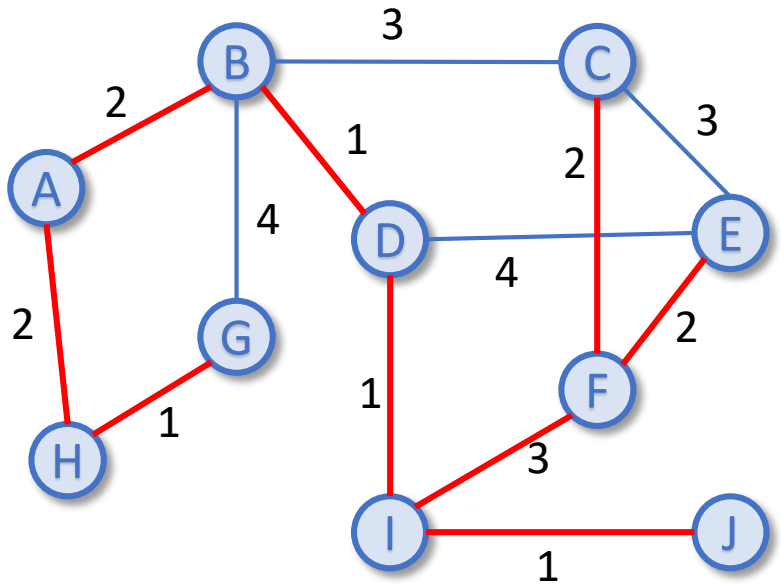


Algoritmo de Prim

- Encontrar un subconjunto de aristas con todos los vértices tal que la suma de los pesos sea mínima
- El conjunto de vértices forman un árbol recubridor mínimo
- Desarrollado por el estadounidense Robert C. Prim en 1957
- Ejemplos de uso
  - ¿Qué carreteras se deben construir para conectar todas las ciudades de Europa de la forma más barata?
  - ¿Cómo se pueden conectar todos los ordenadores de una red con la menor longitud de cable?

Algoritmo de Prim – Conceptos básicos

- Árbol
  - Grafo conexo sin ciclos con numNodos-1 aristas
  - Para cualquier par de nodos siempre existe un camino simple
- Árbol abarcador
  - Conecta todos los nodos del grafo
- Árbol libre abarcador de coste mínimo
  - La suma de los pesos de las arista es la mínima posible



Grafos – Algoritmo de Prim

Obtener el árbol libre abarcador de coste mínimo

Inicializaciones

- **Inicializar** el vector **T** a vacío
  - Se irán almacenando las aristas que formarán parte del Árbol Libre Abarcador de coste mínimo.
- **Inicializar** el vector **U** con un nodo cualquiera del grafo
  - Se irán al almacenando los nodos avaluados

Obtiene

- El vector **T** con los arcos que forman el árbol libre abarcador de coste mínimo

El algoritmo devuelve el vector T

Grafos – Algoritmo de Prim – Ejemplo

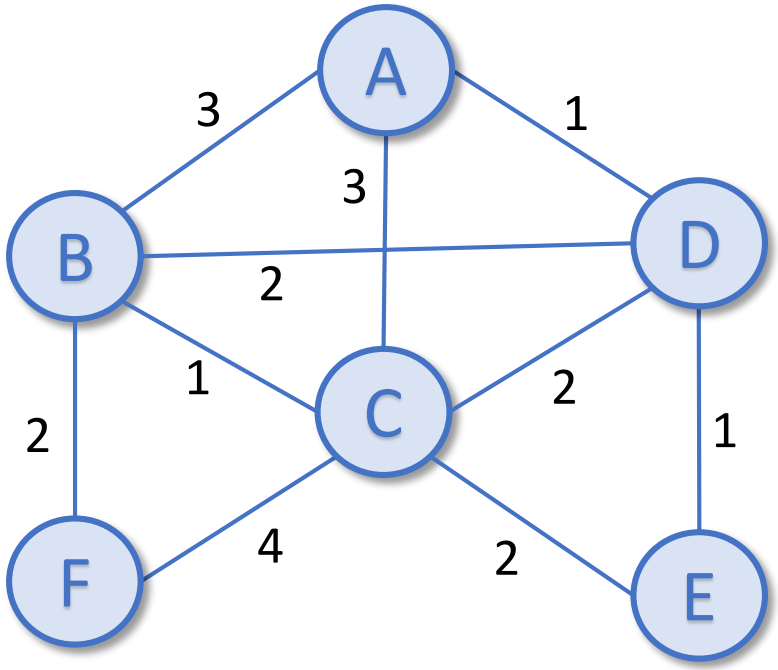
Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso1: Inicialización

U = {A}

T = { }

V = {A ,B, C, D, E, F ,G}

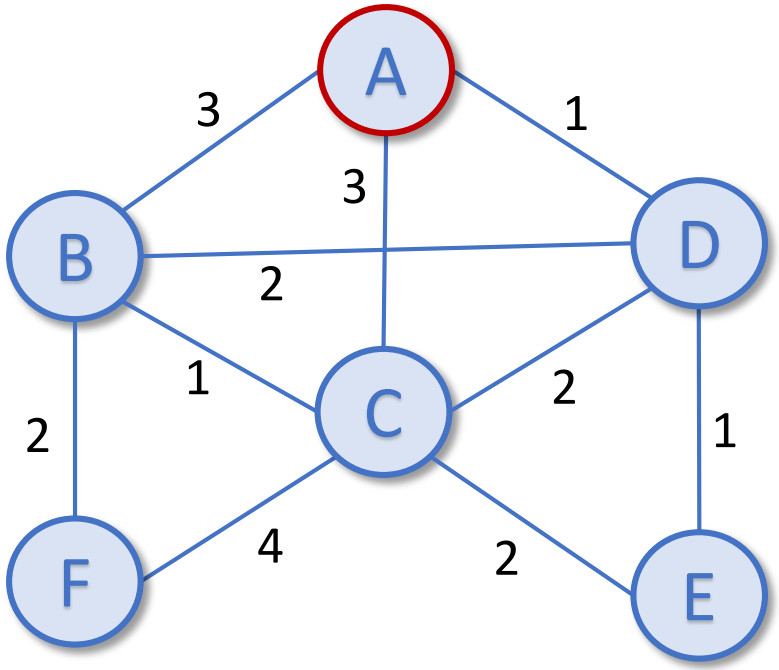


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	
2		
3		
4		
5		
6		

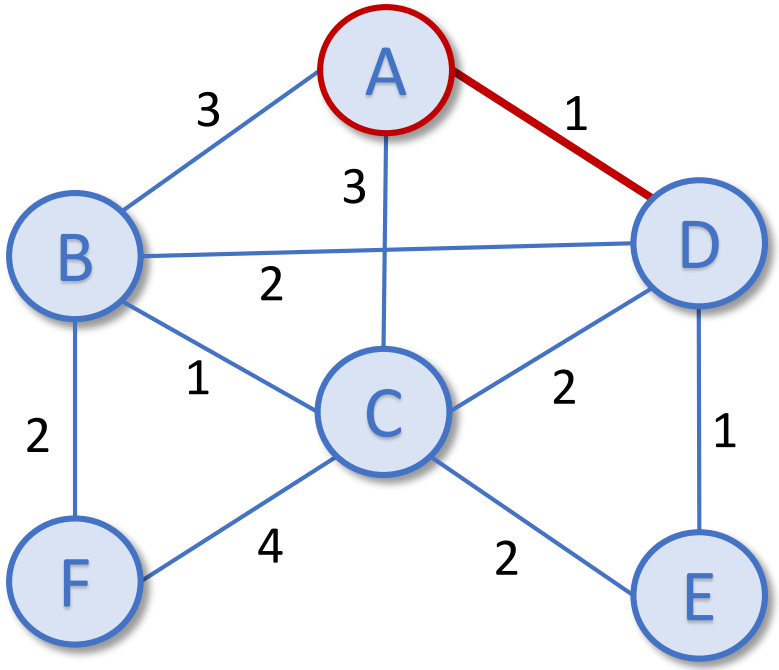


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2		
3		
4		
5		
6		

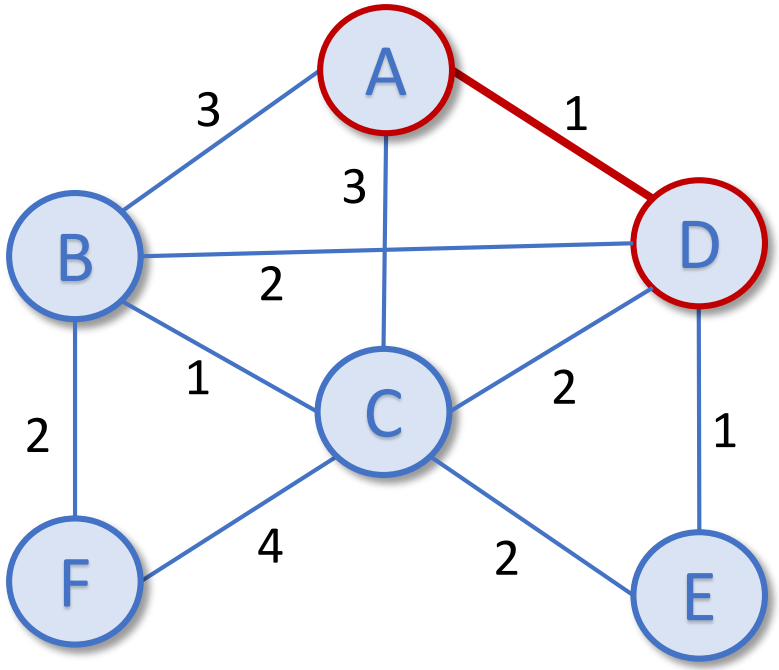


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, …..
3		
4		
5		
6		

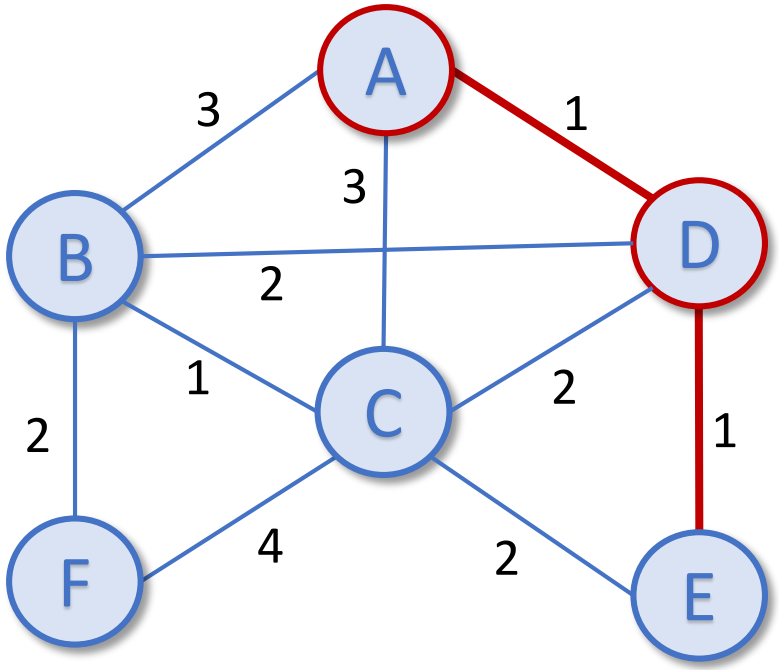


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3		
4		
5		
6		

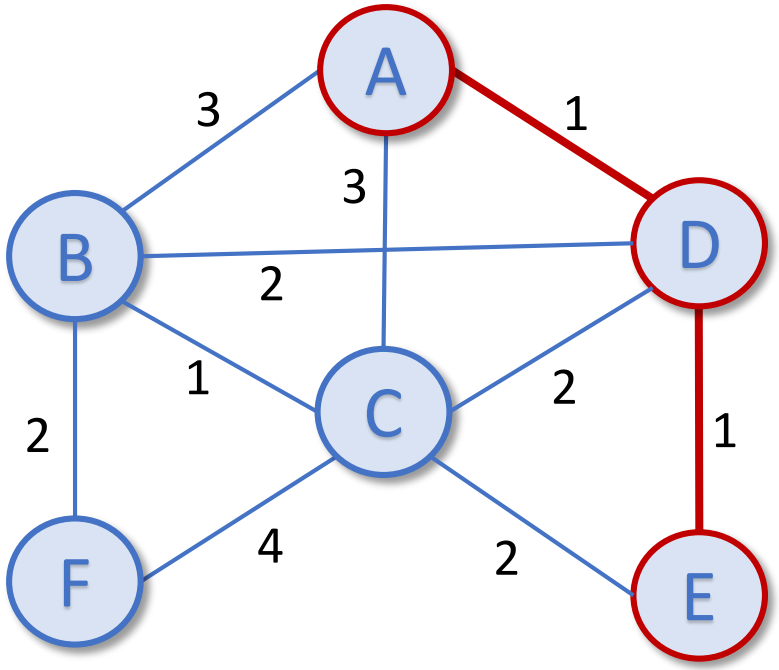


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, ….
4		
5		
6		

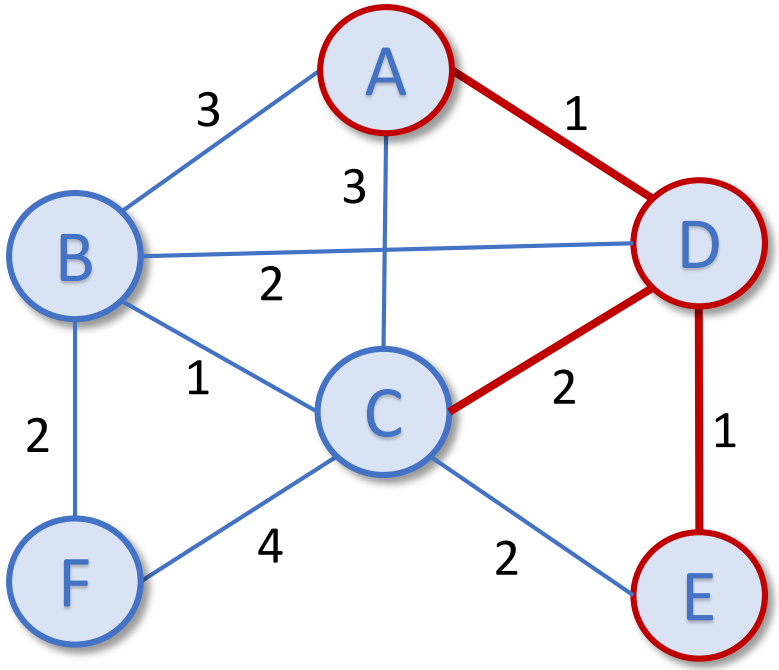


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, DC}
4		
5		
6		

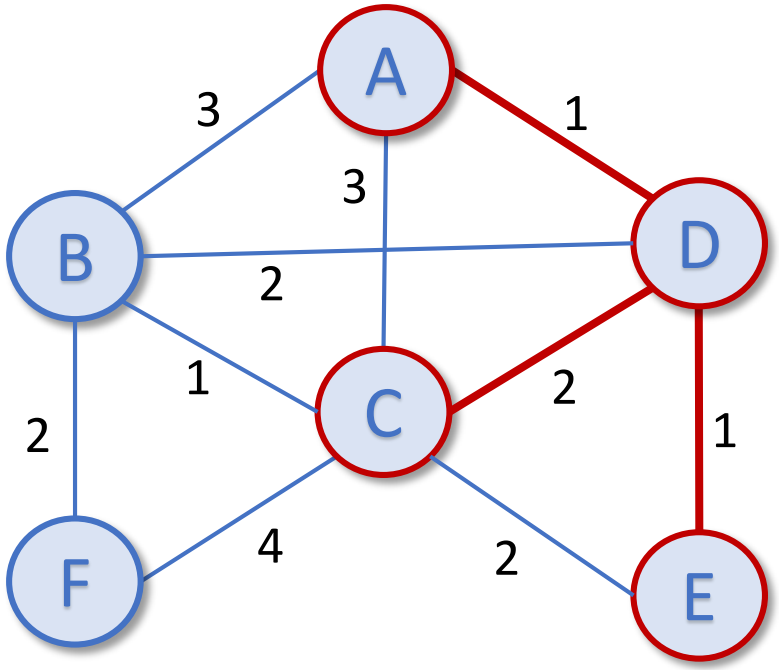


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, DC}
4	{A, D, E, C}	{AD, DE, DC, ….
5		
6		

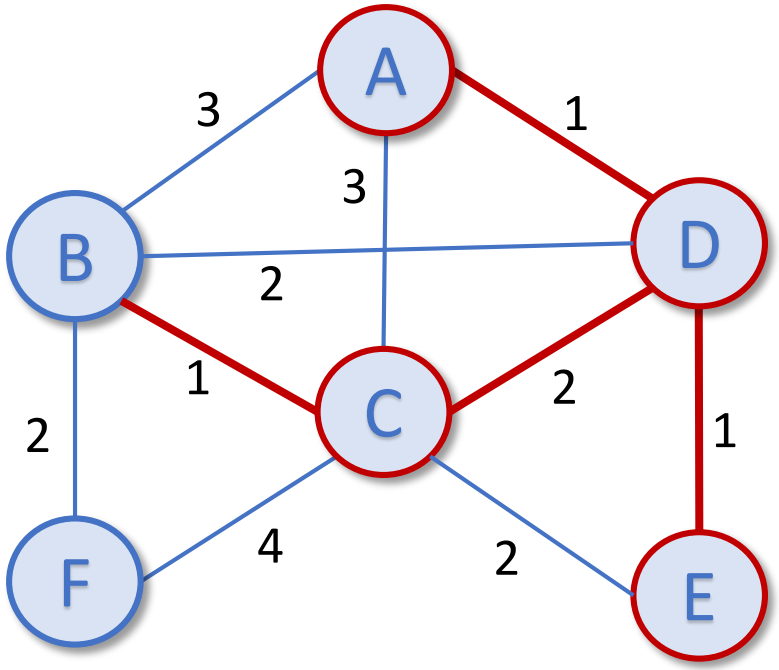


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, DC}
4	{A, D, E, C}	{AD, DE, DC, CB}
5		
6		

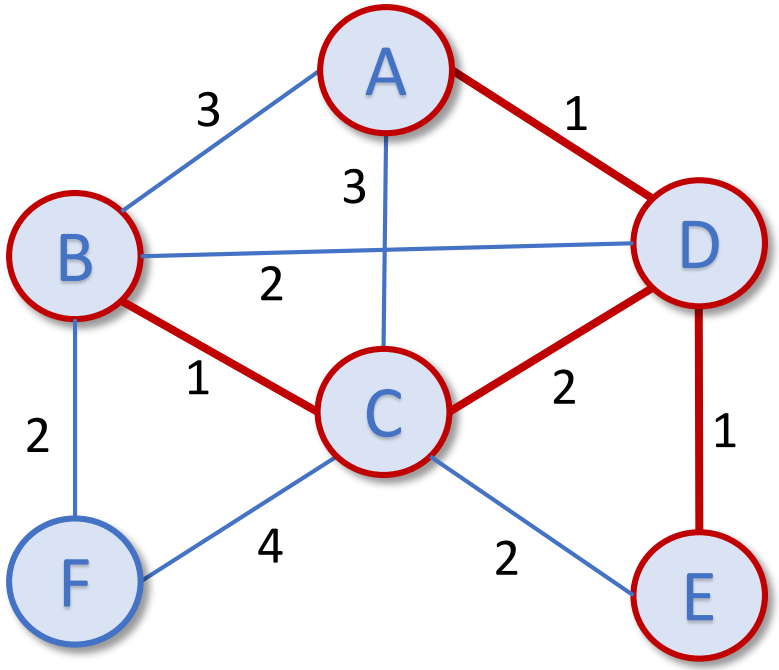


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, DC}
4	{A, D, E, C}	{AD, DE, DC, CB}
5	{A, D, E, C, B}	{AD, DE, DC, CB, ….
6		

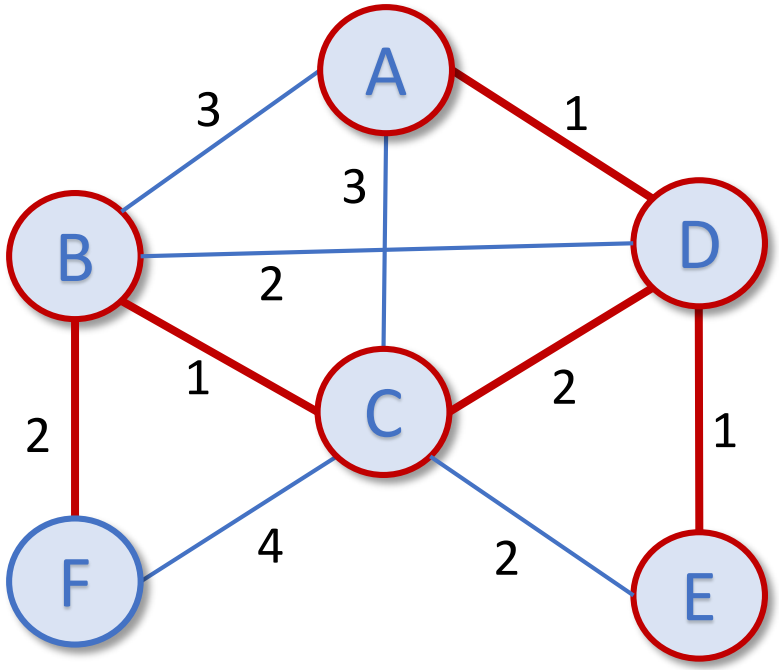


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, DC}
4	{A, D, E, C}	{AD, DE, DC, CB}
5	{A, D, E, C, B}	{AD, DE, DC, CB, BF}
6		

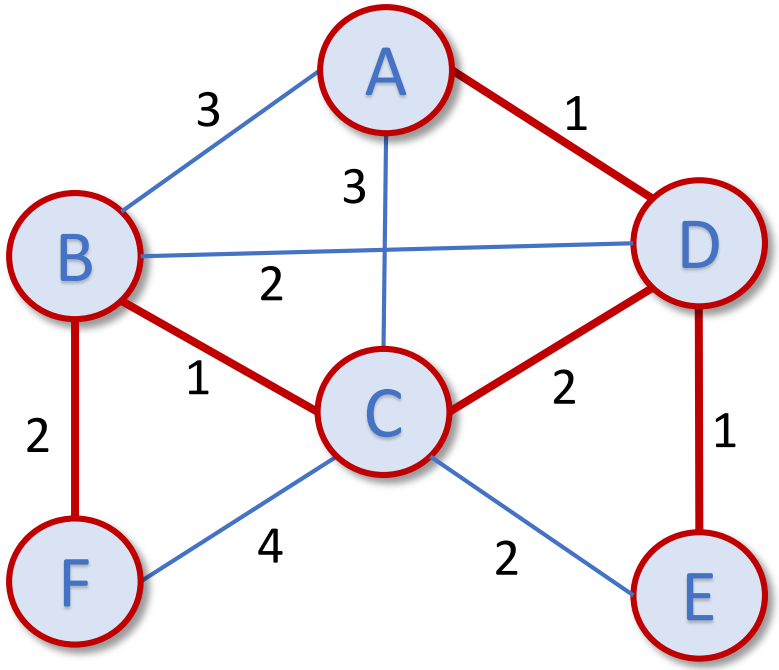


Grafos – Algoritmo de Prim – Ejemplo

Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso2: Partiendo de un nodo seleccionar ele eje de conjunto de nodos evaluados con menor coste

Iteración	U	T
1	{A}	{AD}
2	{A, D}	{AD, DE}
3	{A, D, E}	{AD, DE, DC}
4	{A, D, E, C}	{AD, DE, DC, CB}
5	{A, D, E, C, B}	{AD, DE, DC, CB, BF}
6	{A, D, E, C, B, F}	



Ejercicio – Aplicar el algoritmo de Prim desde A

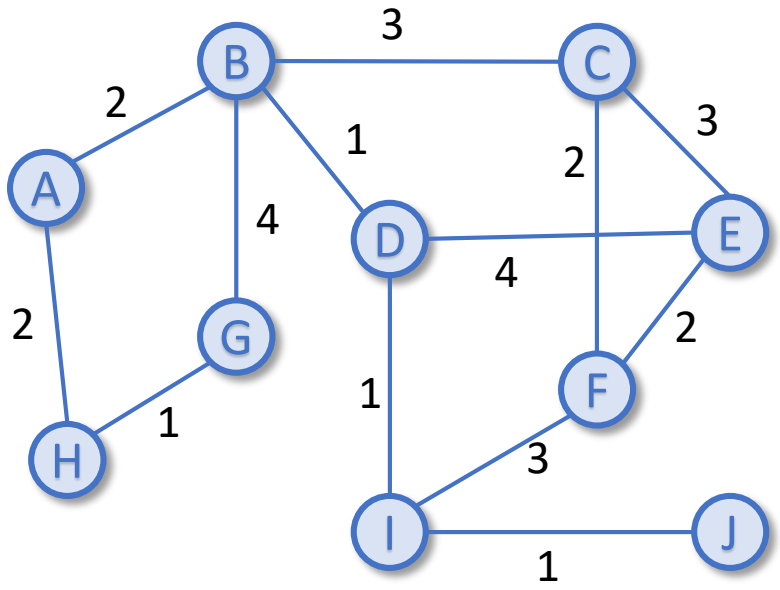
Obtener el árbol libre abarcador de coste mínimo a partir de A

Paso1: Inicialización

U = {A}

T = { }

V = {A ,B, C, D, E, F ,G, H, I, J}



Algoritmo de Prim - Conclusiones

- El árbol resultante depende de:
  - Nodo de partida
  - Selección de la arista de coste mínimo en cada iteración
    - Puede existir más de una con el coste más pequeño

- Optimización
  - Utilizar vectores auxiliares ordenados para elegir la arista de menor coste, reduciendo la complejidad
    - Mayor velocidad a costa de mayor consumo de memoria