Camino mínimo en grafos (II)

Técnicas de Diseño de Algoritmos (Ex Algoritmos y Estructuras de Datos III)

Segundo cuatrimestre 2024

Algoritmos matriciales

Sea $G = (\{1, \ldots, n\}, X)$ un digrafo y $l : X \to R$ una función de longitud/peso para las aristas de G. Definimos las siguientes matrices:

Algoritmos matriciales

Sea $G = (\{1, \ldots, n\}, X)$ un digrafo y $l : X \to R$ una función de longitud/peso para las aristas de G. Definimos las siguientes matrices:

- $L \in R^{n \times n}$, donde los elementos l_{ij} de L se definen como:

$$l_{ij} = \begin{cases} 0 & \text{si } i = j \\ l(i \to j) & \text{si } i \to j \in X \\ \infty & \text{si } i \to j \notin X \end{cases}$$

Algoritmos matriciales

Sea $G = (\{1, \ldots, n\}, X)$ un digrafo y $l : X \to R$ una función de longitud/peso para las aristas de G. Definimos las siguientes matrices:

- $L \in R^{n \times n}$, donde los elementos l_{ij} de L se definen como:

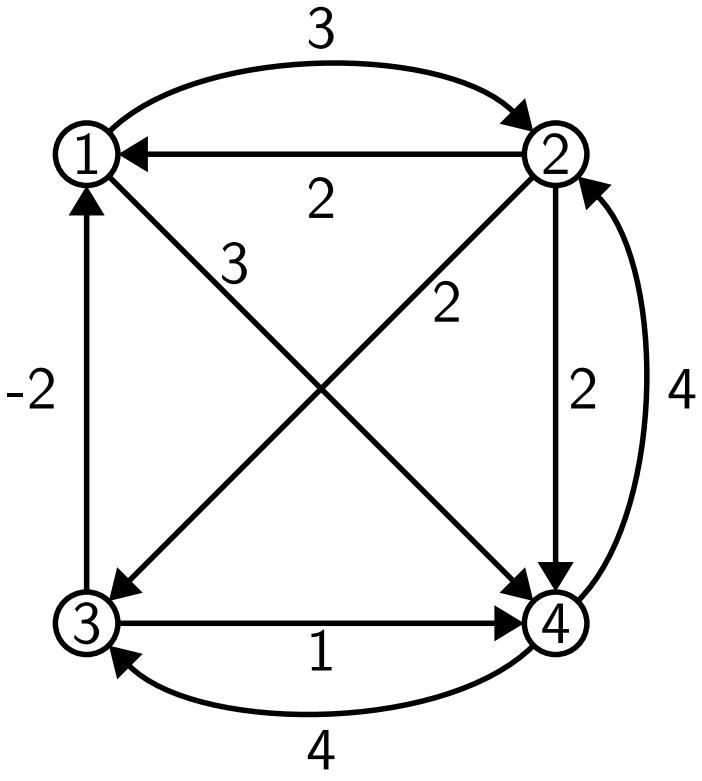
$$l_{ij} = \begin{cases} 0 & \text{si } i = j \\ l(i \to j) & \text{si } i \to j \in X \\ \infty & \text{si } i \to j \notin X \end{cases}$$

- $D \in R^{n \times n}$, donde los elementos d_{ij} de D se definene como:

$$d_{ij} = \begin{cases} \text{longitud del camino mínimo orientado de } i \text{ a } j & \text{si existe alguno} \\ \infty & \text{si no} \end{cases}$$

D es llamada matriz de distancias de G.

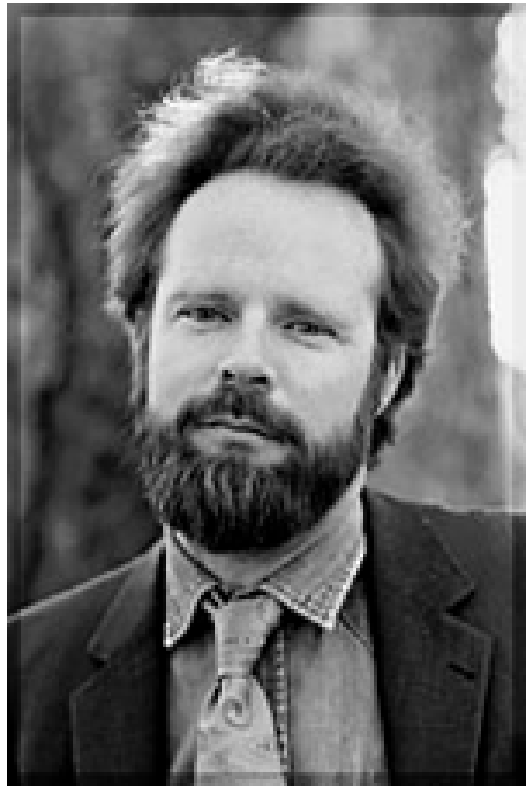
Algoritmos matriciales



$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Floyd (1962)



Robert Floyd (1936–2001)

Algoritmo de Floyd (1962)

Llamamos $v_1, \ldots, v_n$ a los nodos de G. El algoritmo de Floyd se basa en lo siguiente:

Algoritmo de Floyd (1962)

Llamamos $v_1, \ldots, v_n$ a los nodos de G. El algoritmo de Floyd se basa en lo siguiente:

1. Si $L^0 = L$ y calculamos L^1 como

$$l_{ij}^1 = \text{mín}(l_{ij}^0, l_{i1}^0 + l_{1j}^0)$$

l_{ij}^1 es la longitud de un camino mínimo de i a j con nodo intermedio v_1 o directo.

Algoritmo de Floyd (1962)

Llamamos $v_1, \ldots, v_n$ a los nodos de G. El algoritmo de Floyd se basa en lo siguiente:

1. Si $L^0 = L$ y calculamos L^1 como

$$l_{ij}^1 = \text{mín}(l_{ij}^0, l_{i1}^0 + l_{1j}^0)$$

   l_{ij}^1 es la longitud de un camino mínimo de i a j con nodo intermedio v_1 o directo.

2. Si calculamos L^k a partir de L^{k-1} como

$$l_{ij}^k = \text{mín}(l_{ij}^{k-1}, l_{ik}^{k-1} + l_{kj}^{k-1})$$

   l_{ij}^k es la longitud de un camino mínimo de i a j cuyos nodos intermedios están en $\{v_1, \ldots, v_k\}$.

Algoritmo de Floyd (1962)

Llamamos $v_1, \ldots, v_n$ a los nodos de G. El algoritmo de Floyd se basa en lo siguiente:

1. Si $L^0 = L$ y calculamos L^1 como

$$l_{ij}^1 = \text{mín}(l_{ij}^0, l_{i1}^0 + l_{1j}^0)$$

   l_{ij}^1 es la longitud de un camino mínimo de i a j con nodo intermedio v_1 o directo.

2. Si calculamos L^k a partir de L^{k-1} como

$$l_{ij}^k = \text{mín}(l_{ij}^{k-1}, l_{ik}^{k-1} + l_{kj}^{k-1})$$

   l_{ij}^k es la longitud de un camino mínimo de i a j cuyos nodos intermedios están en $\{v_1, \ldots, v_k\}$.

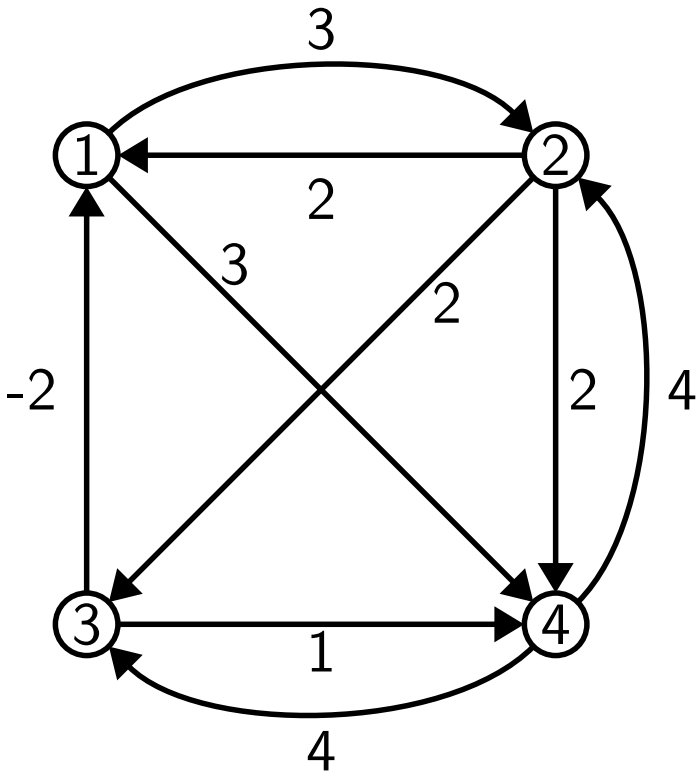
3. $D = L^n$

Algoritmo de Floyd (1962)

Asumimos que el grafo es orientado y que no hay ciclos de longitud negativa.

$L^0 := L$
para k **desde** 1 **a** n **hacer**
  para i **desde** 1 **a** n **hacer**
    para j **desde** 1 **a** n **hacer**
      $l_{ij}^k := \text{mín}(l_{ij}^{k-1}, l_{ik}^{k-1} + l_{kj}^{k-1})$
    fin para
  fin para
fin para
retornar L^n

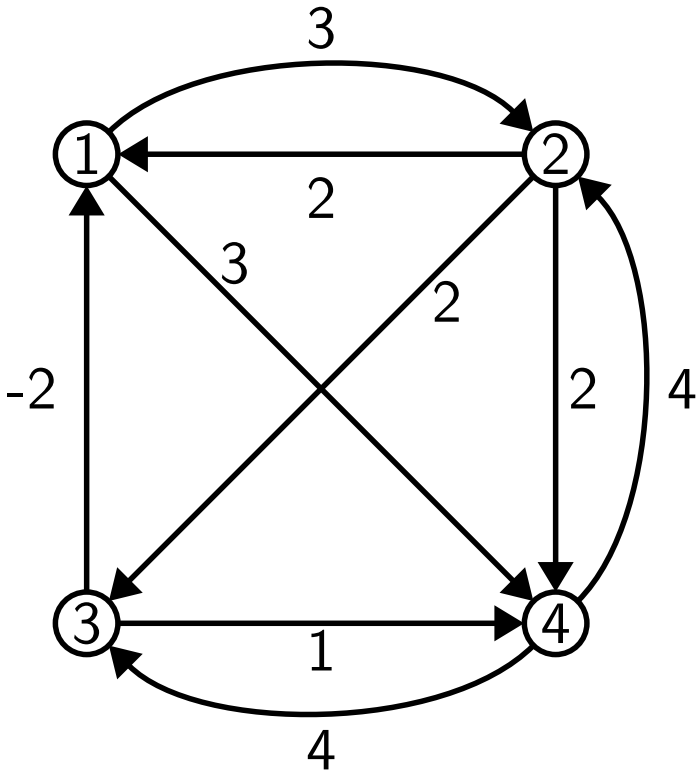
Algoritmo de Floyd (1962) - Ejemplo



$L^0 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Floyd (1962) - Ejemplo



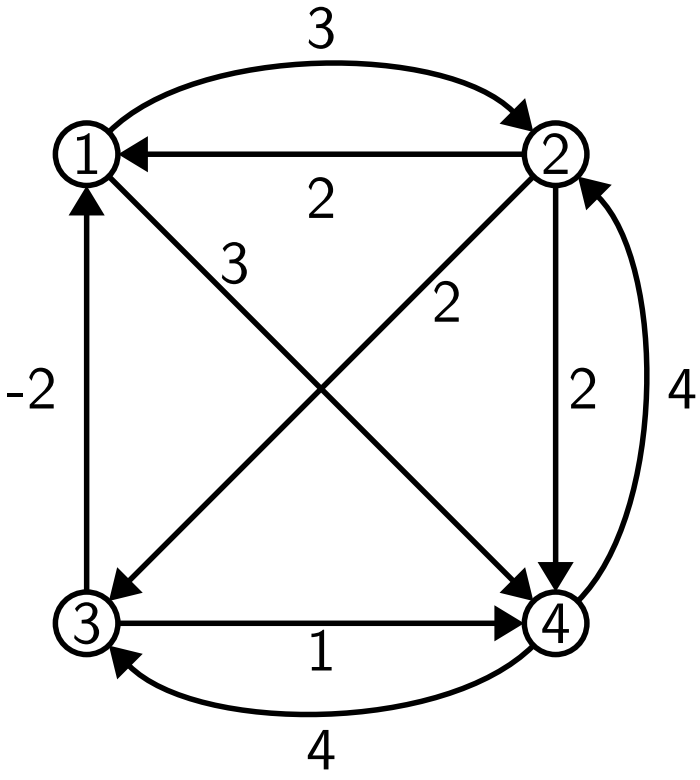
$L^0 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

$L^1 =$

	1	2	3	4
1				
2				
3				
4				

Algoritmo de Floyd (1962) - Ejemplo



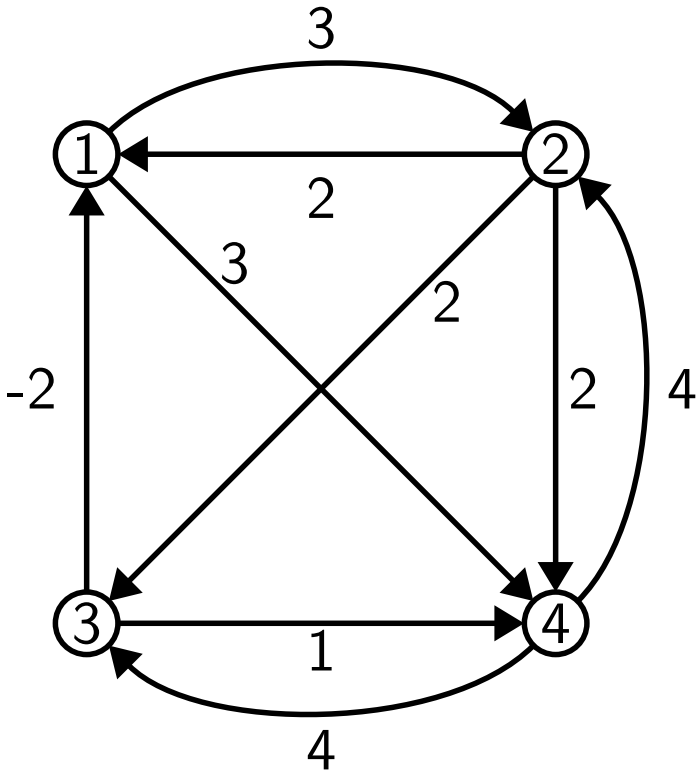
$L^0 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2			
4				

Algoritmo de Floyd (1962) - Ejemplo



$k = 1$ $i = 3$ $j = 2$

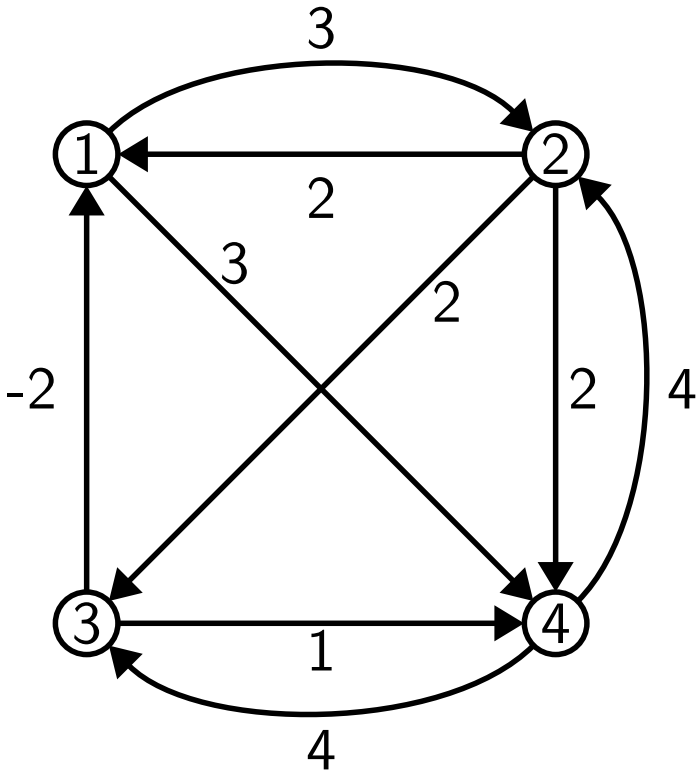
$L^0 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2			
4				

Algoritmo de Floyd (1962) - Ejemplo



$L^0 =$

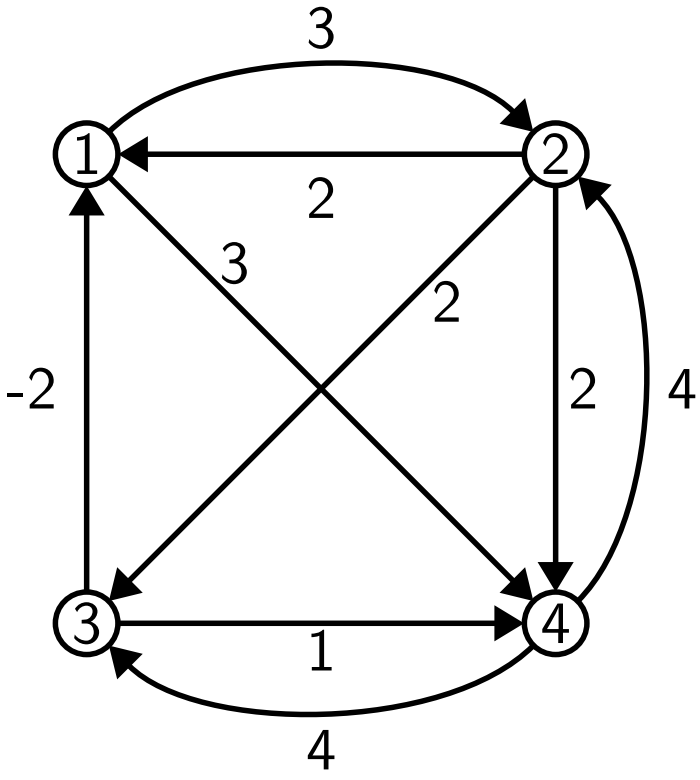
	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1		
4				

$k = 1 \qquad i = 3 \qquad j = 2$

Algoritmo de Floyd (1962) - Ejemplo



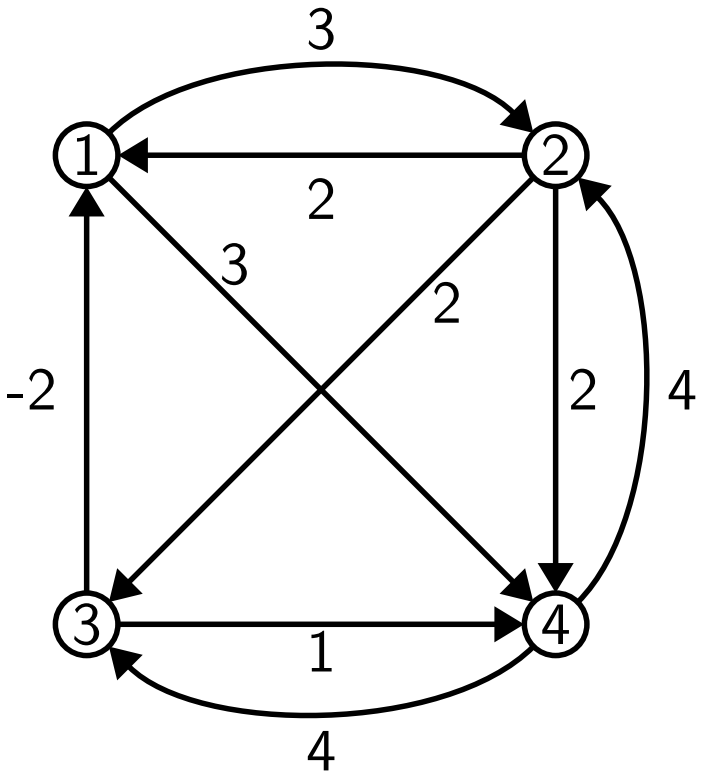
$L^0 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

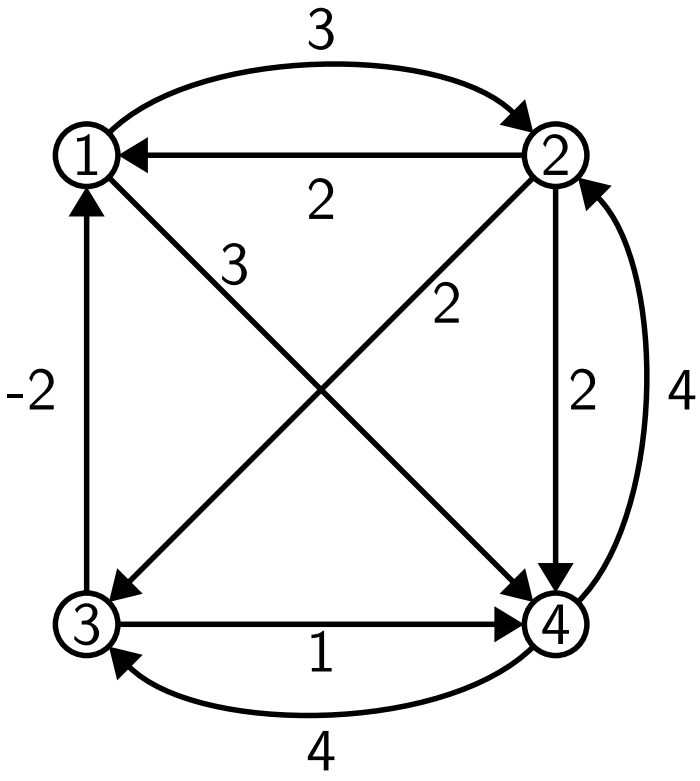
Algoritmo de Floyd (1962) - Ejemplo



$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

Algoritmo de Floyd (1962) - Ejemplo



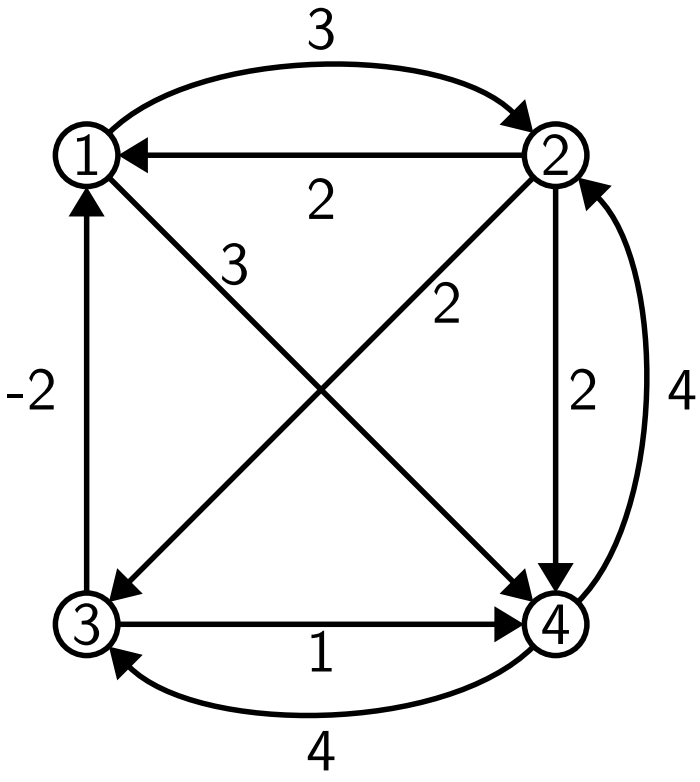
$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3		
2				
3				
4				

Algoritmo de Floyd (1962) - Ejemplo



$L^1 =$

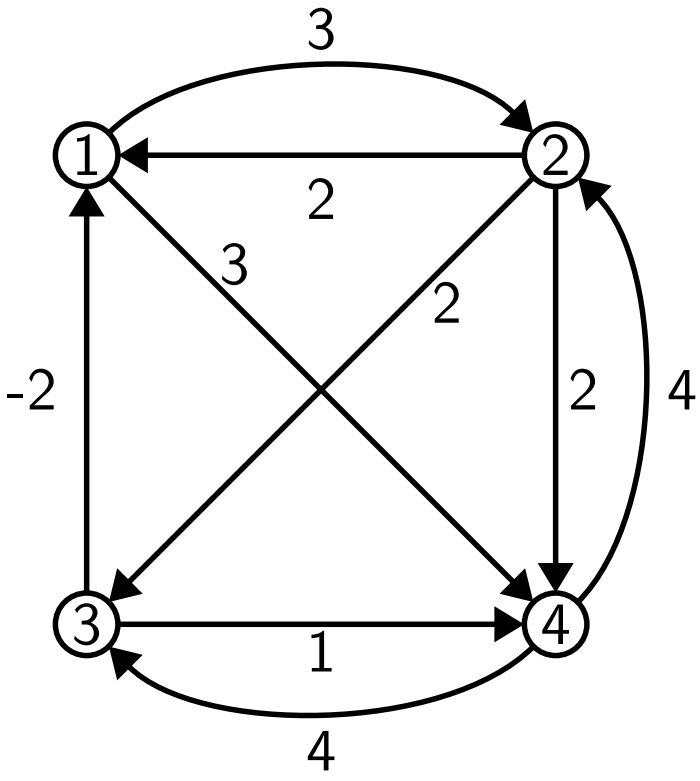
	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3		
2				
3				
4				

$k = 2 \qquad i = 1 \qquad j = 3$

Algoritmo de Floyd (1962) - Ejemplo



$L^1 =$

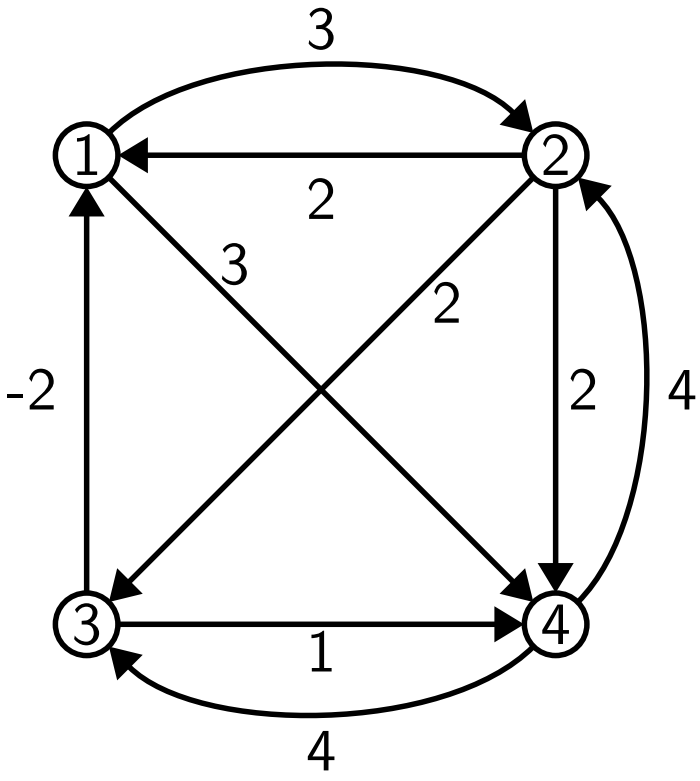
	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3	5	
2				
3				
4				

$k = 2 \qquad i = 1 \qquad j = 3$

Algoritmo de Floyd (1962) - Ejemplo



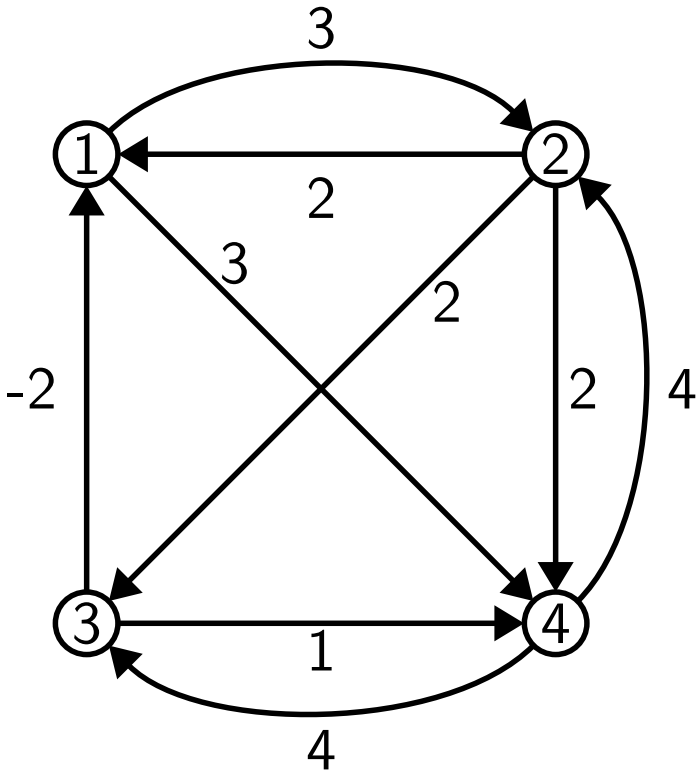
$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4				

Algoritmo de Floyd (1962) - Ejemplo



$L^1 =$

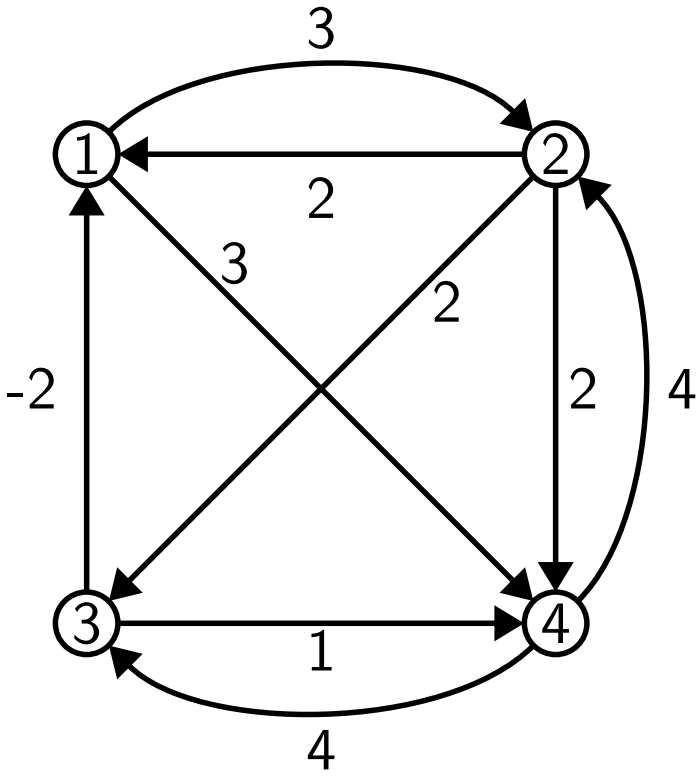
	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4				

$k = 2 \qquad i = 4 \qquad j = 1$

Algoritmo de Floyd (1962) - Ejemplo



$k = 2$ $i = 4$ $j = 1$

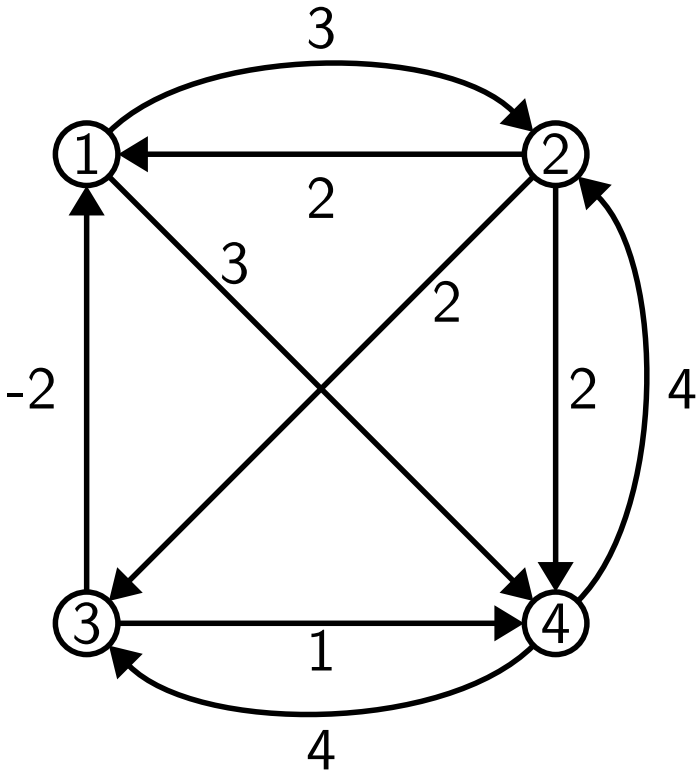
$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6			

Algoritmo de Floyd (1962) - Ejemplo



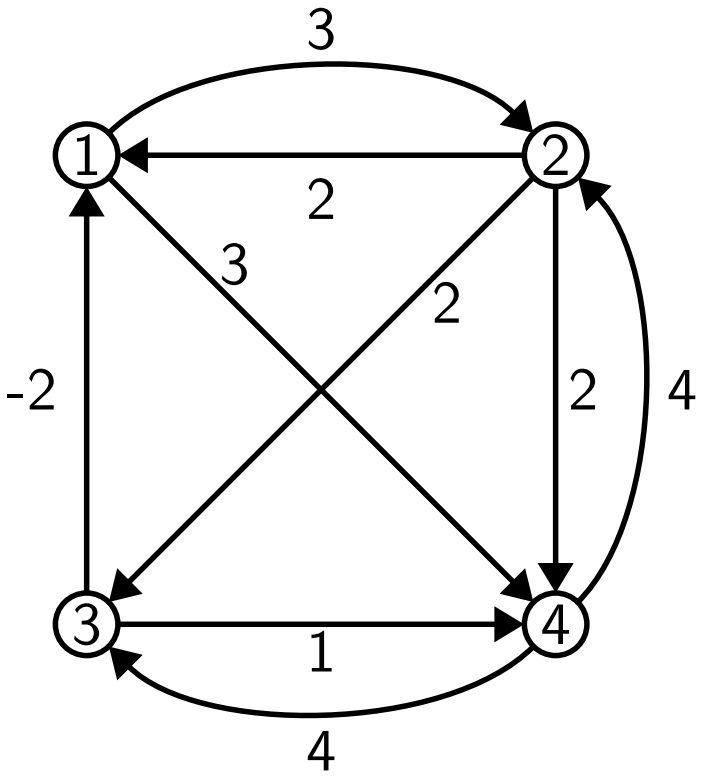
$L^1 =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	1	0	1
4	∞	4	4	0

$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

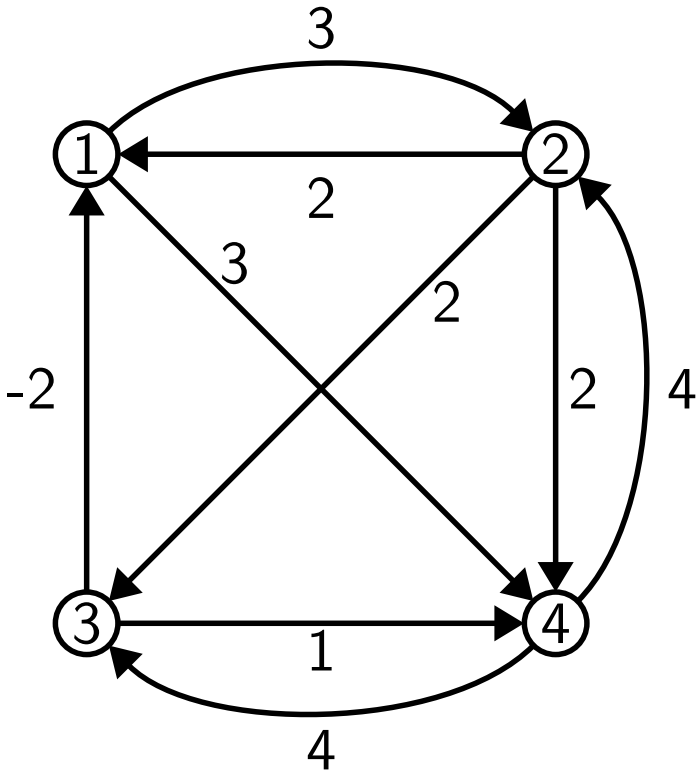
Algoritmo de Floyd (1962) - Ejemplo



$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

Algoritmo de Floyd (1962) - Ejemplo



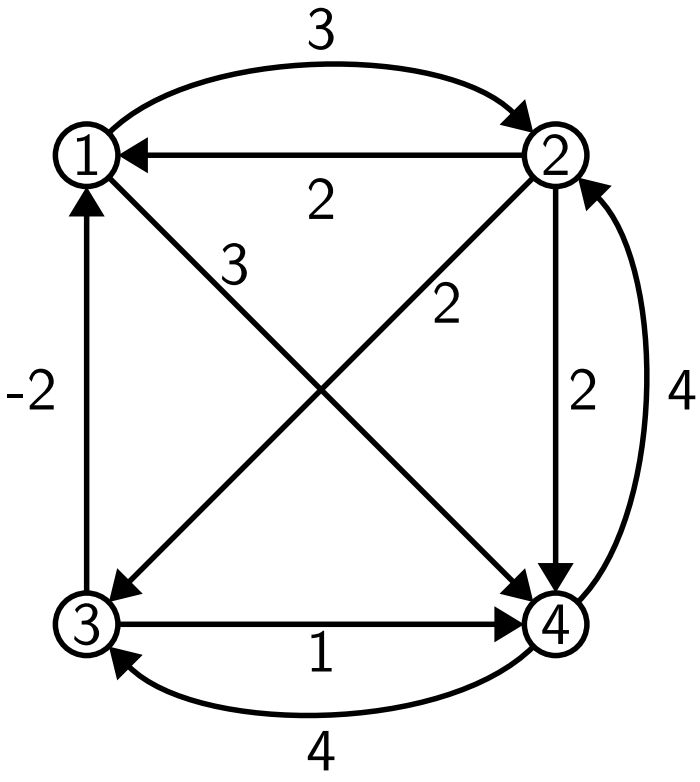
$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

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

Algoritmo de Floyd (1962) - Ejemplo



$L^2 =$

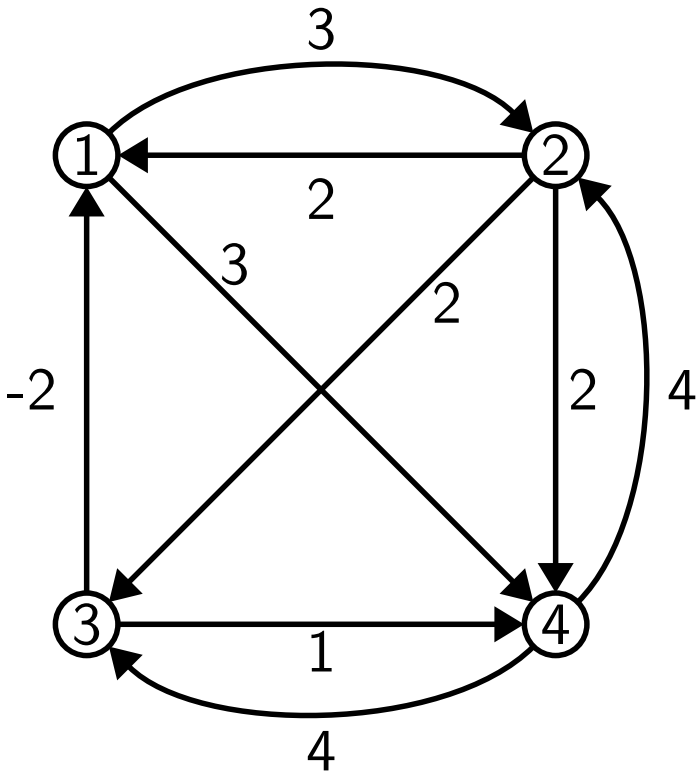
	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

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

$k = 3 \qquad i = 2 \qquad j = 1$

Algoritmo de Floyd (1962) - Ejemplo



$L^2 =$

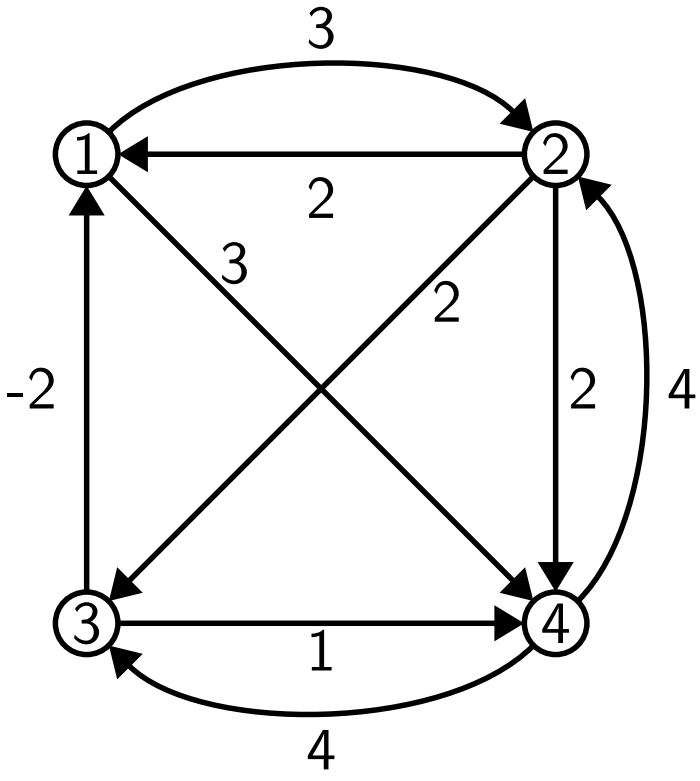
	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

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

$k = 3 \qquad i = 2 \qquad j = 1$

Algoritmo de Floyd (1962) - Ejemplo



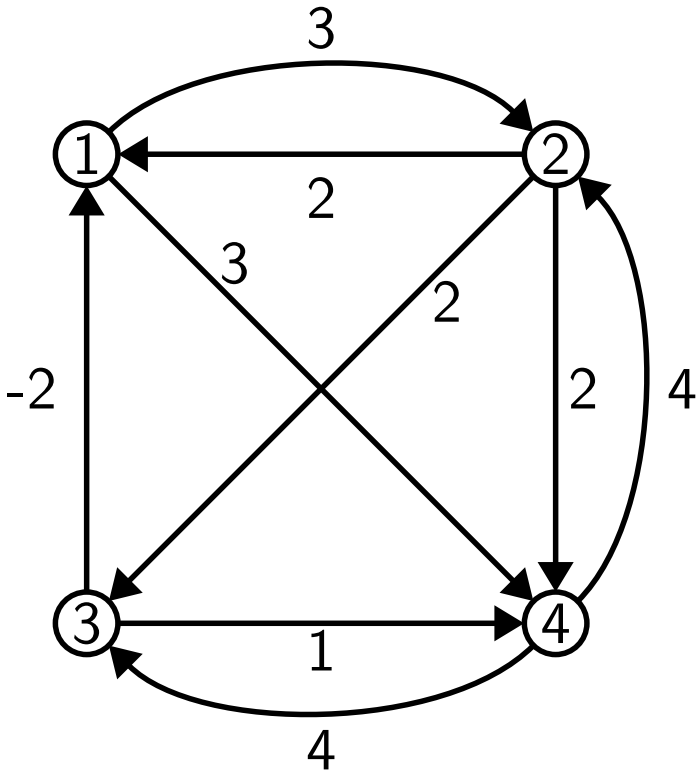
$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4				

Algoritmo de Floyd (1962) - Ejemplo



$L^2 =$

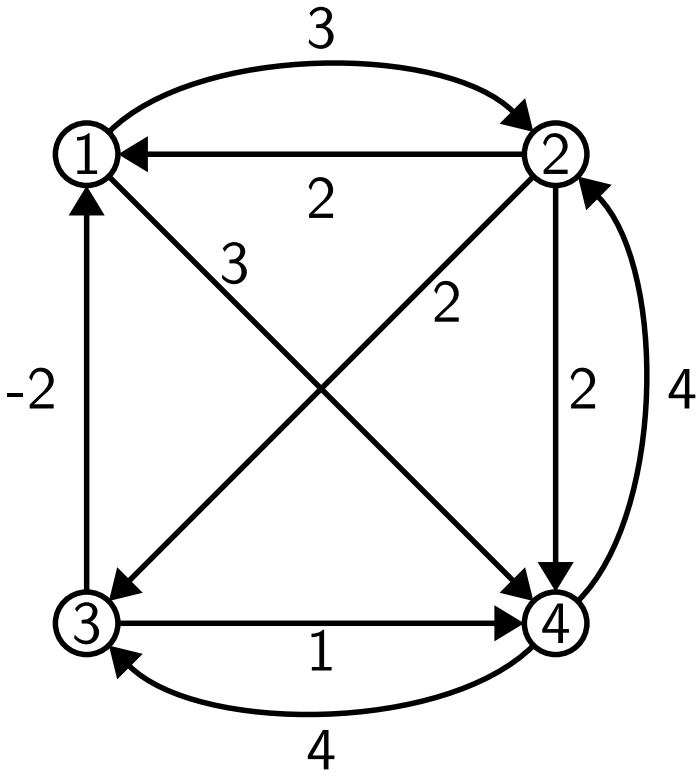
	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4				

$k = 3 \qquad i = 4 \qquad j = 1$

Algoritmo de Floyd (1962) - Ejemplo



$k = 3$ $i = 4$ $j = 1$

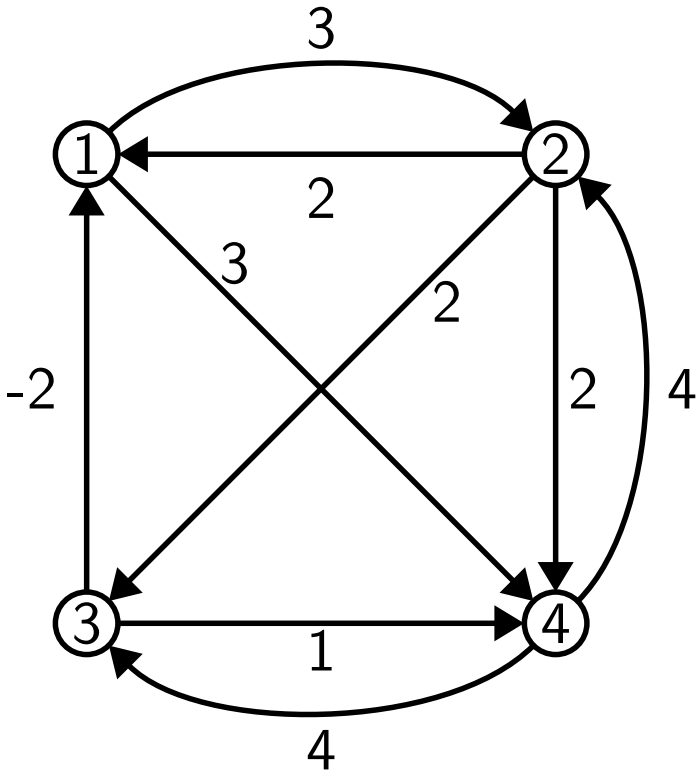
$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2			

Algoritmo de Floyd (1962) - Ejemplo



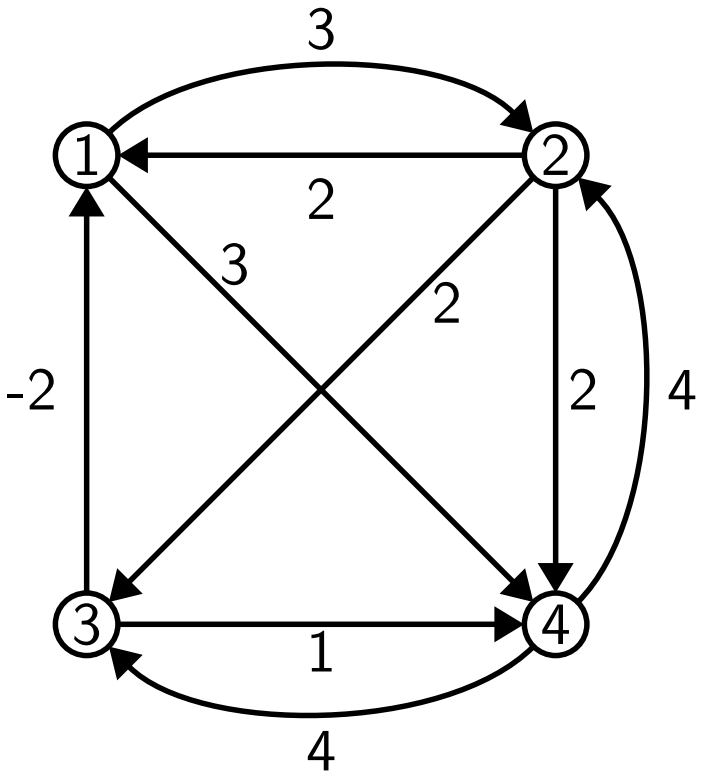
$L^2 =$

	1	2	3	4
1	0	3	5	3
2	2	0	2	2
3	-2	1	0	1
4	6	4	4	0

$L^3 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

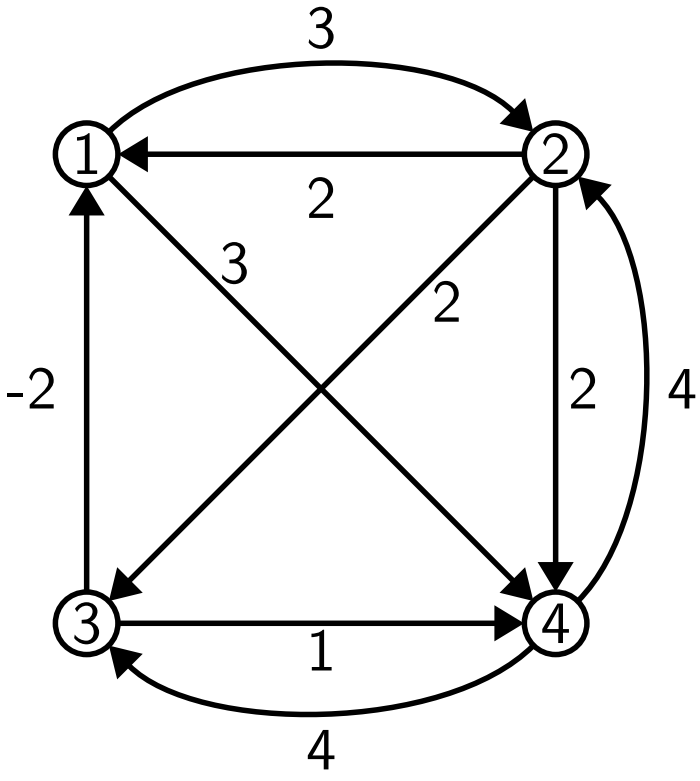
Algoritmo de Floyd (1962) - Ejemplo



$L^3 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

Algoritmo de Floyd (1962) - Ejemplo



$L^3 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

$D = L^4 =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

Algoritmo de Floyd (1962)

Lema: Al finalizar la iteración k del algoritmo de Floyd, l_{ij} es la longitud de los caminos mínimos desde v_i a v_j cuyos nodos intermedios son elementos de $V_k = \{v_1, \ldots, v_k\}$, si no existe ciclo de longitud negativa con todos sus vértices en V_k

Algoritmo de Floyd (1962)

Lema: Al finalizar la iteración k del algoritmo de Floyd, l_{ij} es la longitud de los caminos mínimos desde v_i a v_j cuyos nodos intermedios son elementos de $V_k = \{v_1, \ldots, v_k\}$, si no existe ciclo de longitud negativa con todos sus vértices en V_k

Teorema: El algoritmo de Floyd determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos negativos.

Algoritmo de Floyd (1962)

Lema: Al finalizar la iteración k del algoritmo de Floyd, l_{ij} es la longitud de los caminos mínimos desde v_i a v_j cuyos nodos intermedios son elementos de $V_k = \{v_1, \ldots, v_k\}$, si no existe ciclo de longitud negativa con todos sus vértices en V_k

Teorema: El algoritmo de Floyd determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos negativos.

- ¿Cuál es la complejidad de algoritmo de Floyd?

Algoritmo de Floyd (1962)

Lema: Al finalizar la iteración k del algoritmo de Floyd, l_{ij} es la longitud de los caminos mínimos desde v_i a v_j cuyos nodos intermedios son elementos de $V_k = \{v_1, \ldots, v_k\}$, si no existe ciclo de longitud negativa con todos sus vértices en V_k

Teorema: El algoritmo de Floyd determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos negativos.

- ¿Cuál es la complejidad de algoritmo de Floyd?
- ¿Cuánta memoria requiere?

Algoritmo de Floyd (1962)

Lema: Al finalizar la iteración k del algoritmo de Floyd, l_{ij} es la longitud de los caminos mínimos desde v_i a v_j cuyos nodos intermedios son elementos de $V_k = \{v_1, \ldots, v_k\}$, si no existe ciclo de longitud negativa con todos sus vértices en V_k

Teorema: El algoritmo de Floyd determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos negativos.

- ¿Cuál es la complejidad de algoritmo de Floyd?
- ¿Cuánta memoria requiere?
- ¿Cómo podemos hacer si además de las longitudes queremos determinar los caminos mínimos?

Algoritmo de Floyd (1962)

Lema: Al finalizar la iteración k del algoritmo de Floyd, l_{ij} es la longitud de los caminos mínimos desde v_i a v_j cuyos nodos intermedios son elementos de $V_k = \{v_1, \ldots, v_k\}$, si no existe ciclo de longitud negativa con todos sus vértices en V_k

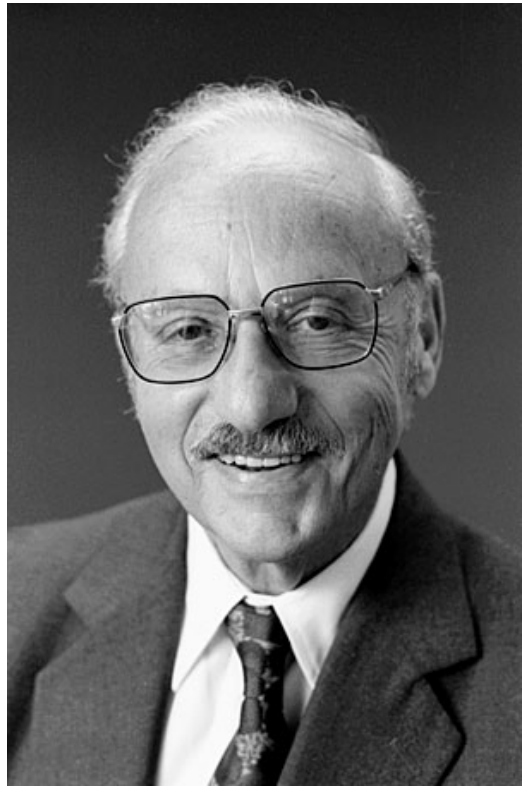
Teorema: El algoritmo de Floyd determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos negativos.

- ¿Cuál es la complejidad de algoritmo de Floyd?
- ¿Cuánta memoria requiere?
- ¿Cómo podemos hacer si además de las longitudes queremos determinar los caminos mínimos?
- ¿Cómo se puede adaptar para detectar si el grafo tiene ciclos de longitud negativa?

Algoritmo de Floyd (1962)

$L^0 := L$
para k **desde** 1 **a** n **hacer**
    para i **desde** 1 **a** n **hacer**
        si $l_{ik}^{k-1} \neq \infty$ **entonces**
            si $l_{ik}^{k-1} + l_{ki}^{k-1} < 0$ **entonces**
                retornar `"Hay ciclos negativos."`
            fin si
            para j **desde** 1 **a** n **hacer**
                $l_{ij}^{k} := \text{mín}(l_{ij}^{k-1}, l_{ik}^{k-1} + l_{kj}^{k-1})$
            fin para
        fin si
    fin para
fin para
retornar L

Algoritmo de Dantzig (1966)



George Dantzig (1914–2005)

Algoritmo de Dantzig (1966)

Al finalizar la iteración $k-1$, el algoritmo de Dantzig genera una matriz de $k \times k$ de caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Algoritmo de Dantzig (1966)

Al finalizar la iteración $k-1$, el algoritmo de Dantzig genera una matriz de $k \times k$ de caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Calcula la matriz L^{k+1} a partir de la matriz L^k para $1 \leq i, j \leq k$ como:

- $L^{k+1}_{i,k+1} = \text{mín}_{1 \leq j \leq k}(L^k_{i,j} + L^k_{j,k+1})$
- $L^{k+1}_{k+1,i} = \text{mín}_{1 \leq j \leq k}(L^k_{k+1,j} + L^k_{j,i})$
- $L^{k+1}_{i,j} = \text{mín}(L^k_{i,j}, L^k_{i,k+1} + L^k_{k+1,j})$

Algoritmo de Dantzig (1966)

Al finalizar la iteración $k-1$, el algoritmo de Dantzig genera una matriz de $k \times k$ de caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Calcula la matriz L^{k+1} a partir de la matriz L^k para $1 \leq i, j \leq k$ como:

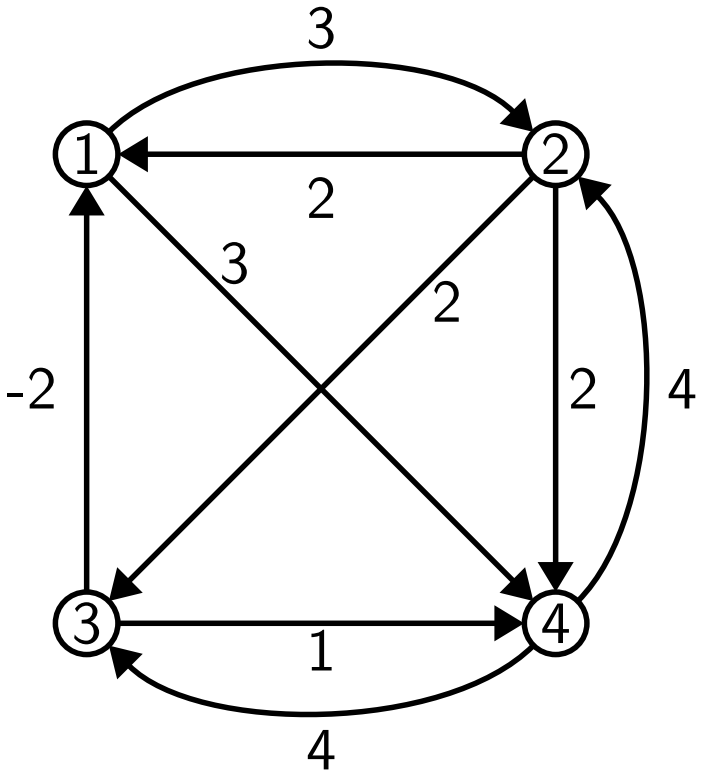
- $L^{k+1}_{i,k+1} = \text{mín}_{1 \leq j \leq k}(L^k_{i,j} + L^k_{j,k+1})$
- $L^{k+1}_{k+1,i} = \text{mín}_{1 \leq j \leq k}(L^k_{k+1,j} + L^k_{j,i})$
- $L^{k+1}_{i,j} = \text{mín}(L^k_{i,j}, L^k_{i,k+1} + L^k_{k+1,j})$

Asumimos que el grafo es orientado. Detecta si hay ciclos de longitud negativa.

Algoritmo de Dantzig (1966)

para k **desde** 1 **a** $n-1$ **hacer**
  para i **desde** 1 **a** k **hacer**
    $L_{i,k+1} := \text{mín}_{1 \leq j \leq k}(L_{i,j} + L_{j,k+1})$
    $L_{k+1,i} := \text{mín}_{1 \leq j \leq k}(L_{k+1,j} + L_{j,i})$
  fin para
  $t := \text{mín}_{1 \leq i \leq k}(L_{k+1,i} + L_{i,k+1})$
  si $t < 0$ **entonces**
    retornar "Hay ciclos de longitud negativa"
  fin si
  para i **desde** 1 **a** k **hacer**
    para j **desde** 1 **a** k **hacer**
      $L_{i,j} := \text{mín}(L_{i,j}, L_{i,k+1} + L_{k+1,j})$
    fin para
  fin para
fin para
retornar L

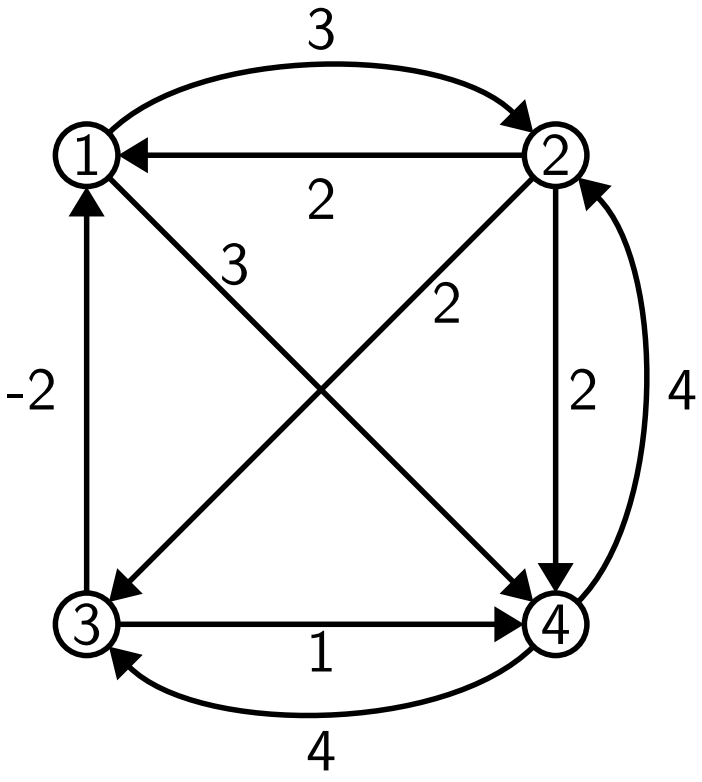
Algoritmo de Dantzig (1966) - Ejemplo



$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

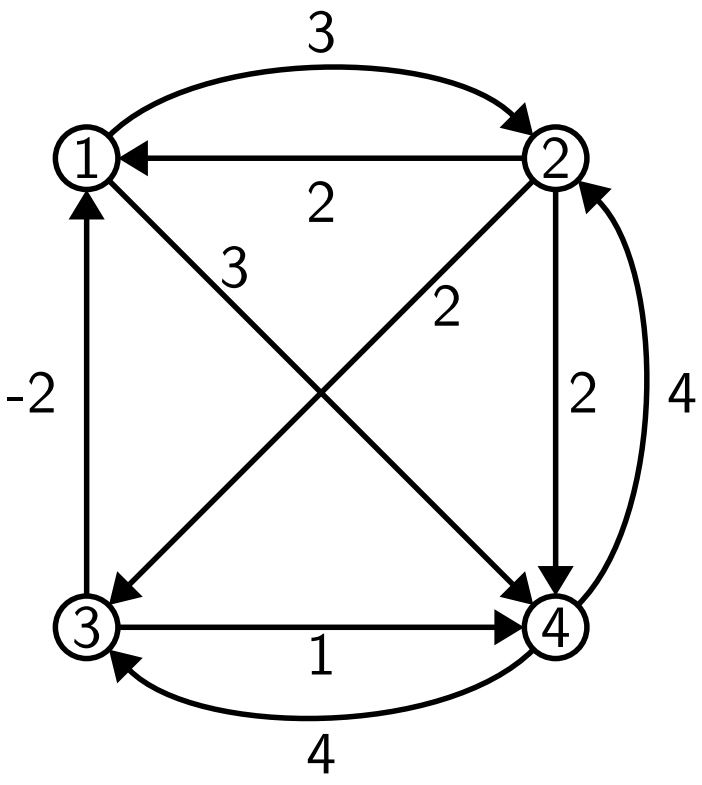


$k = 1$

$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

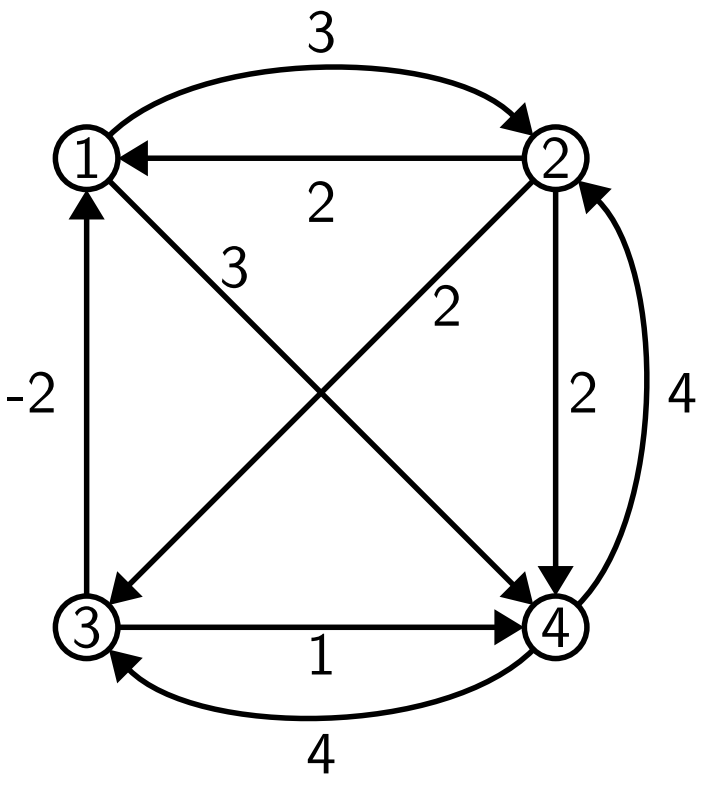


$k = 1$

$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

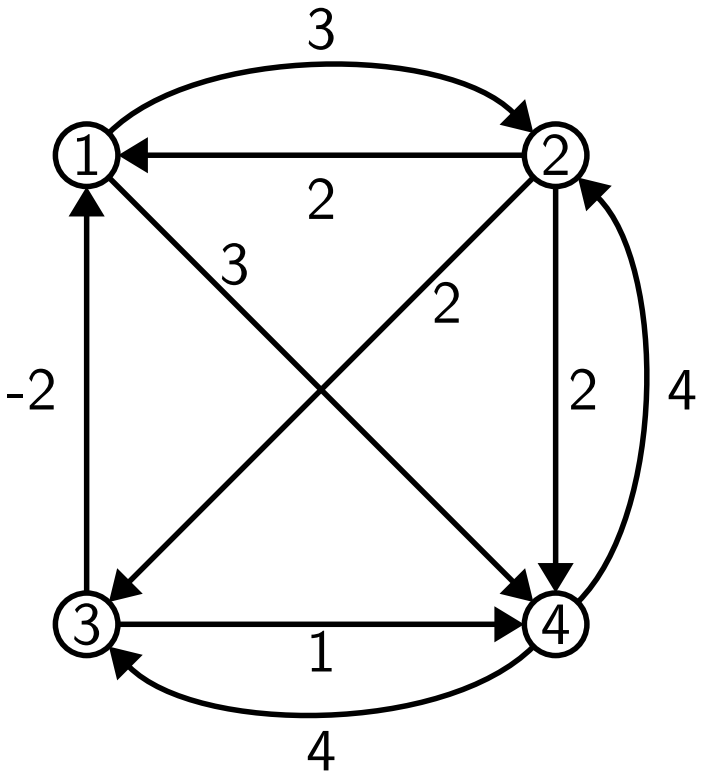


$k = 1$

$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

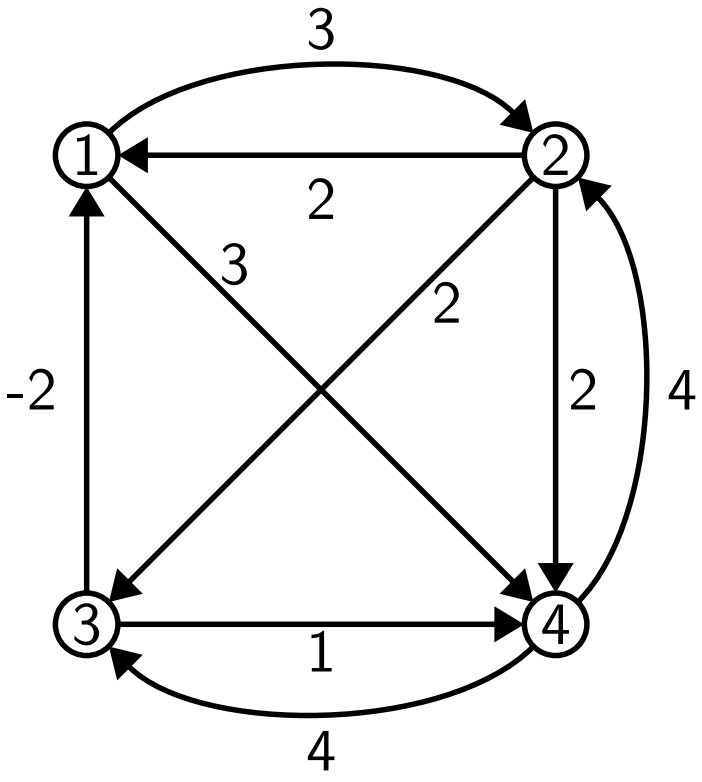


$k = 2$

$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

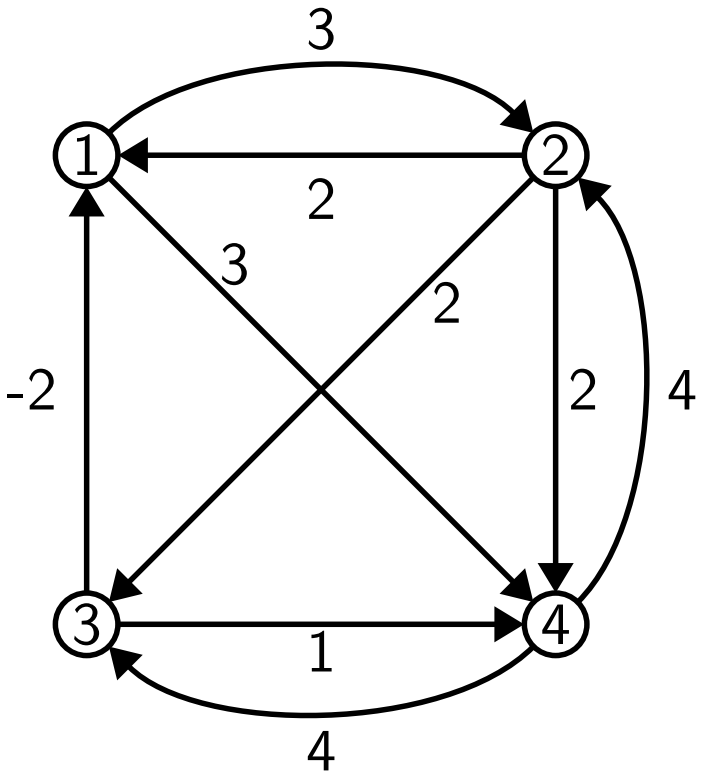


$k = 2$

$L =$

	1	2	3	4
1	0	3	∞	3
2	2	0	2	2
3	-2	∞	0	1
4	∞	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

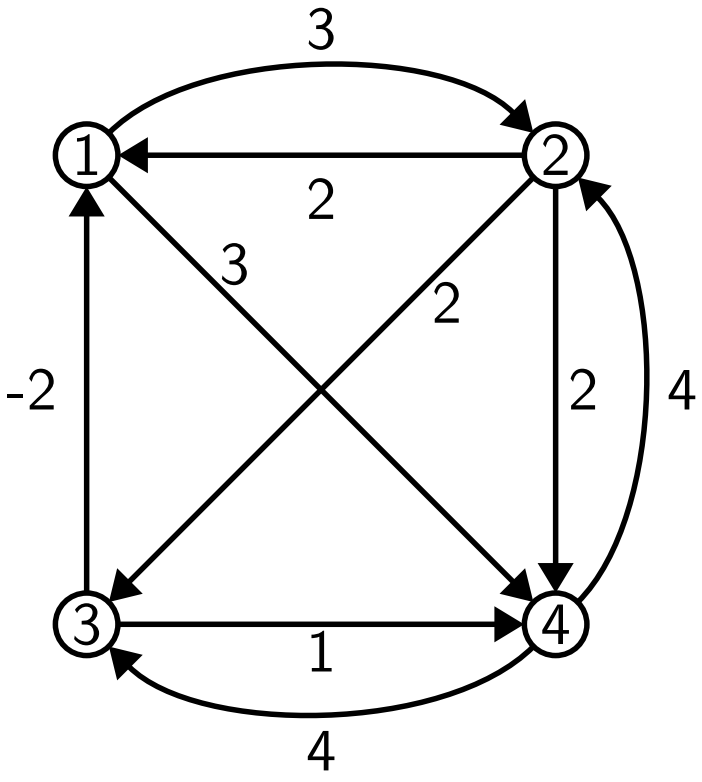


$k = 2$

$L =$

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

Algoritmo de Dantzig (1966) - Ejemplo

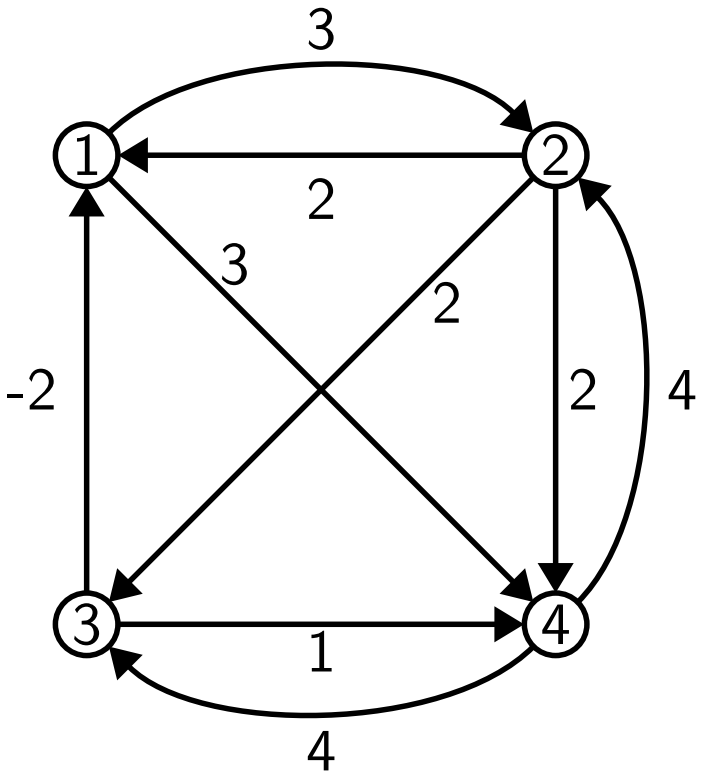


$k = 2$

$L =$

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

Algoritmo de Dantzig (1966) - Ejemplo

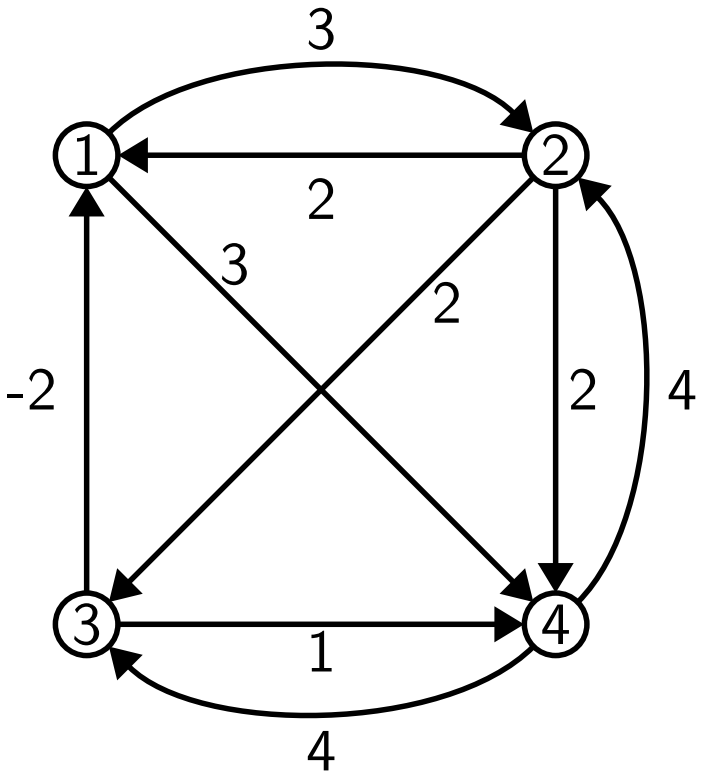


$k = 3$

$L =$

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

Algoritmo de Dantzig (1966) - Ejemplo

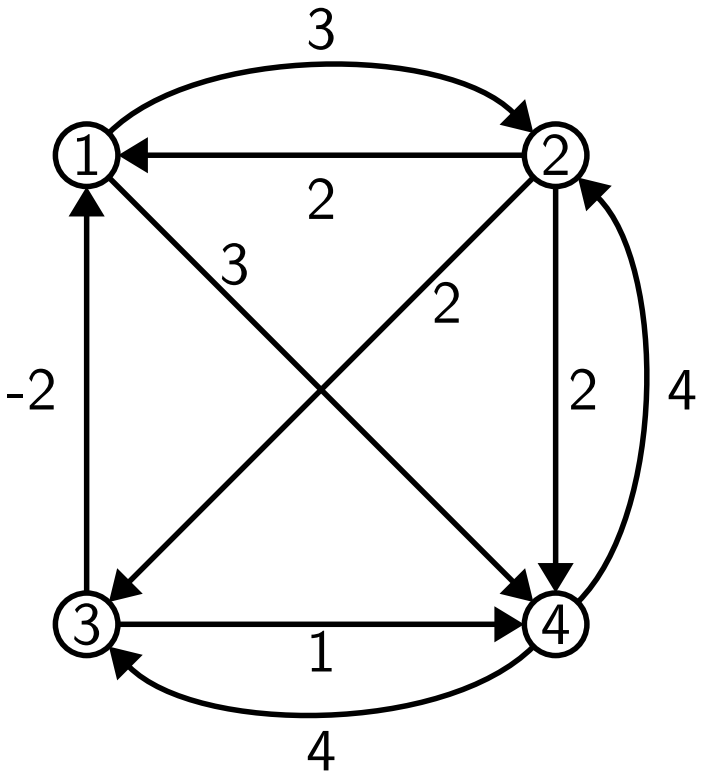


$k = 3$

$L =$

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

Algoritmo de Dantzig (1966) - Ejemplo

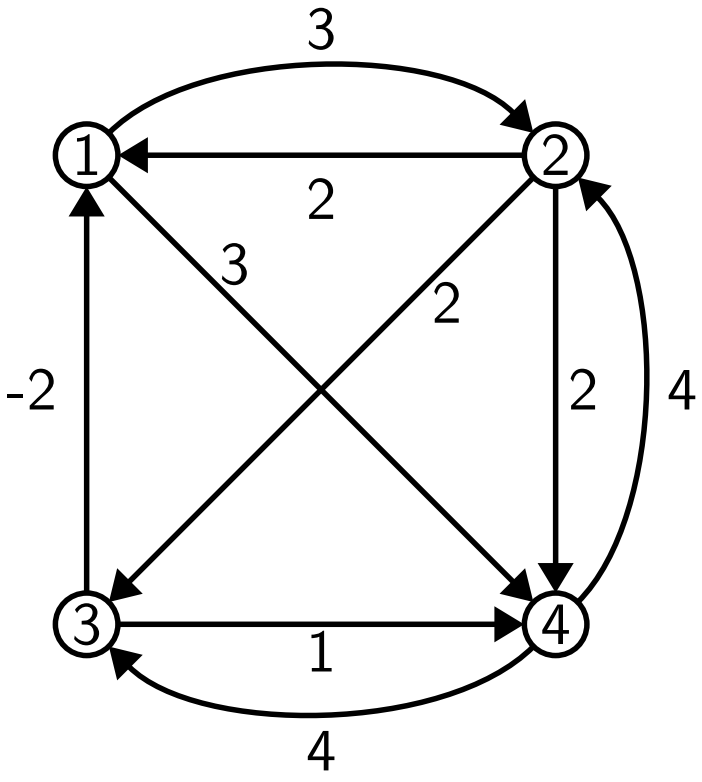


$k = 3$

$L =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo

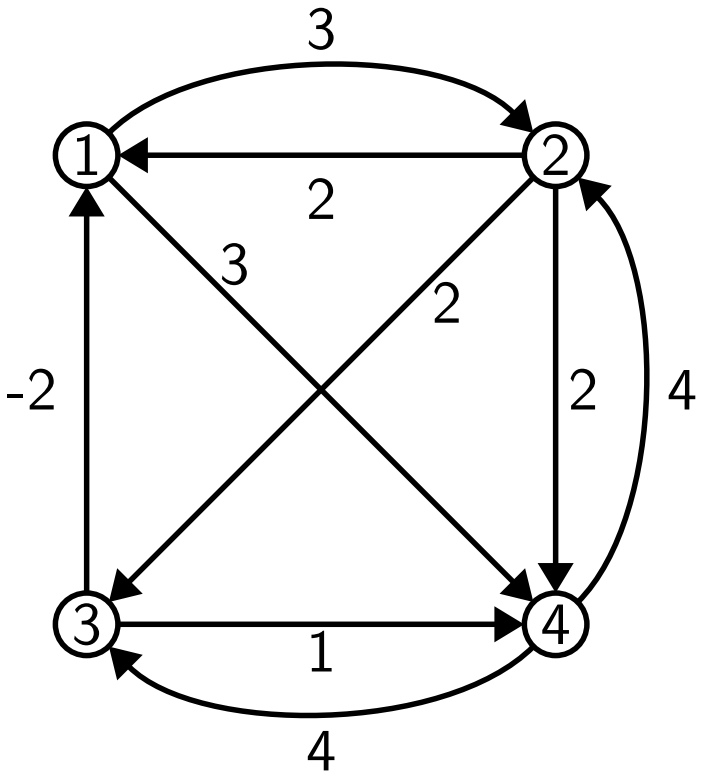


$k = 3$

$L =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

Algoritmo de Dantzig (1966) - Ejemplo



$k = 4$

$L =$

	1	2	3	4
1	0	3	5	3
2	0	0	2	2
3	-2	1	0	1
4	2	4	4	0

Algoritmo de Dantzig (1966)

Lema Al finalizar la iteración $k-1$ del algoritmo de Dantzig, la matriz de $k \times k$ generada contiene la longitud de los caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Algoritmo de Dantzig (1966)

Lema Al finalizar la iteración $k-1$ del algoritmo de Dantzig, la matriz de $k \times k$ generada contiene la longitud de los caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Teorema: El algoritmo de Dantzig determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos.

Algoritmo de Dantzig (1966)

Lema Al finalizar la iteración $k-1$ del algoritmo de Dantzig, la matriz de $k \times k$ generada contiene la longitud de los caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Teorema: El algoritmo de Dantzig determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos.

- ¿Cuál es la complejidad del algoritmo de Dantzig?

Algoritmo de Dantzig (1966)

Lema Al finalizar la iteración $k-1$ del algoritmo de Dantzig, la matriz de $k \times k$ generada contiene la longitud de los caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Teorema: El algoritmo de Dantzig determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos.

- ¿Cuál es la complejidad del algoritmo de Dantzig?
- ¿Qué diferencia tiene con el algoritmo de Floyd?

Algoritmo de Dantzig (1966)

Lema Al finalizar la iteración $k-1$ del algoritmo de Dantzig, la matriz de $k \times k$ generada contiene la longitud de los caminos mínimos en el subgrafo inducido por los vértices $\{v_1, \ldots, v_k\}$.

Teorema: El algoritmo de Dantzig determina los caminos mínimos entre todos los pares de nodos de un grafo orientado sin ciclos.

- ¿Cuál es la complejidad del algoritmo de Dantzig?
- ¿Qué diferencia tiene con el algoritmo de Floyd?
- ¿Qué ventajas o desventajas tiene?