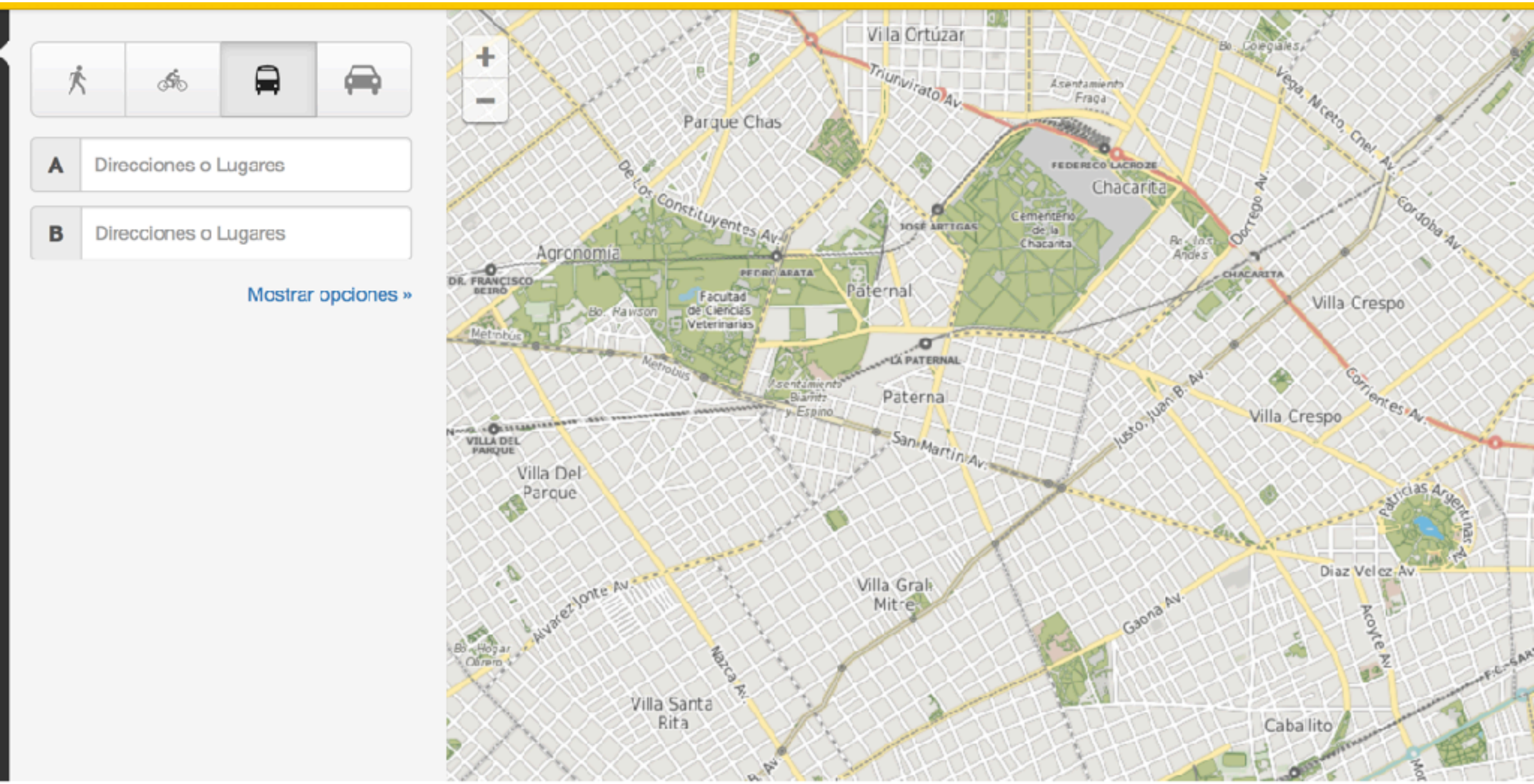
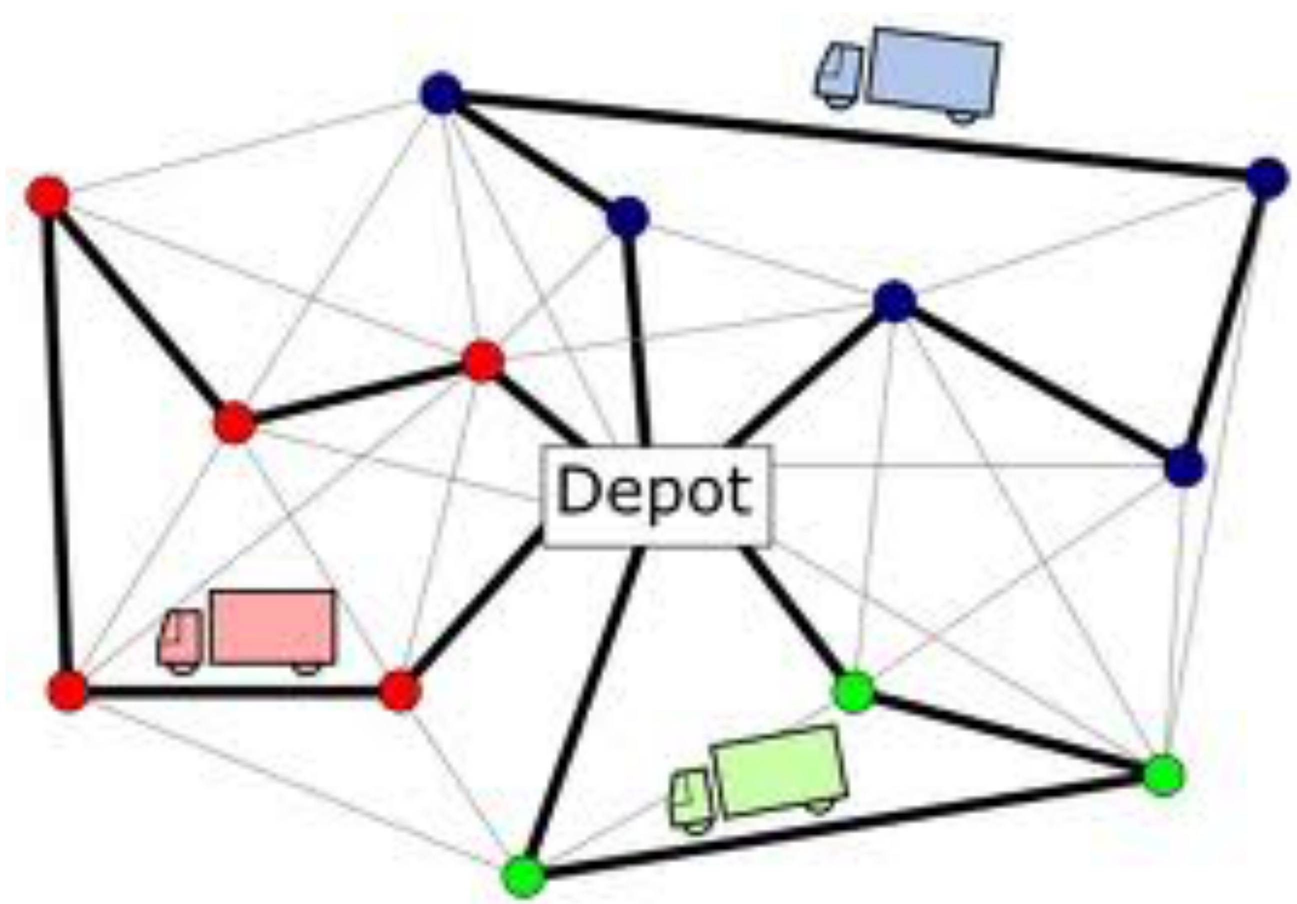
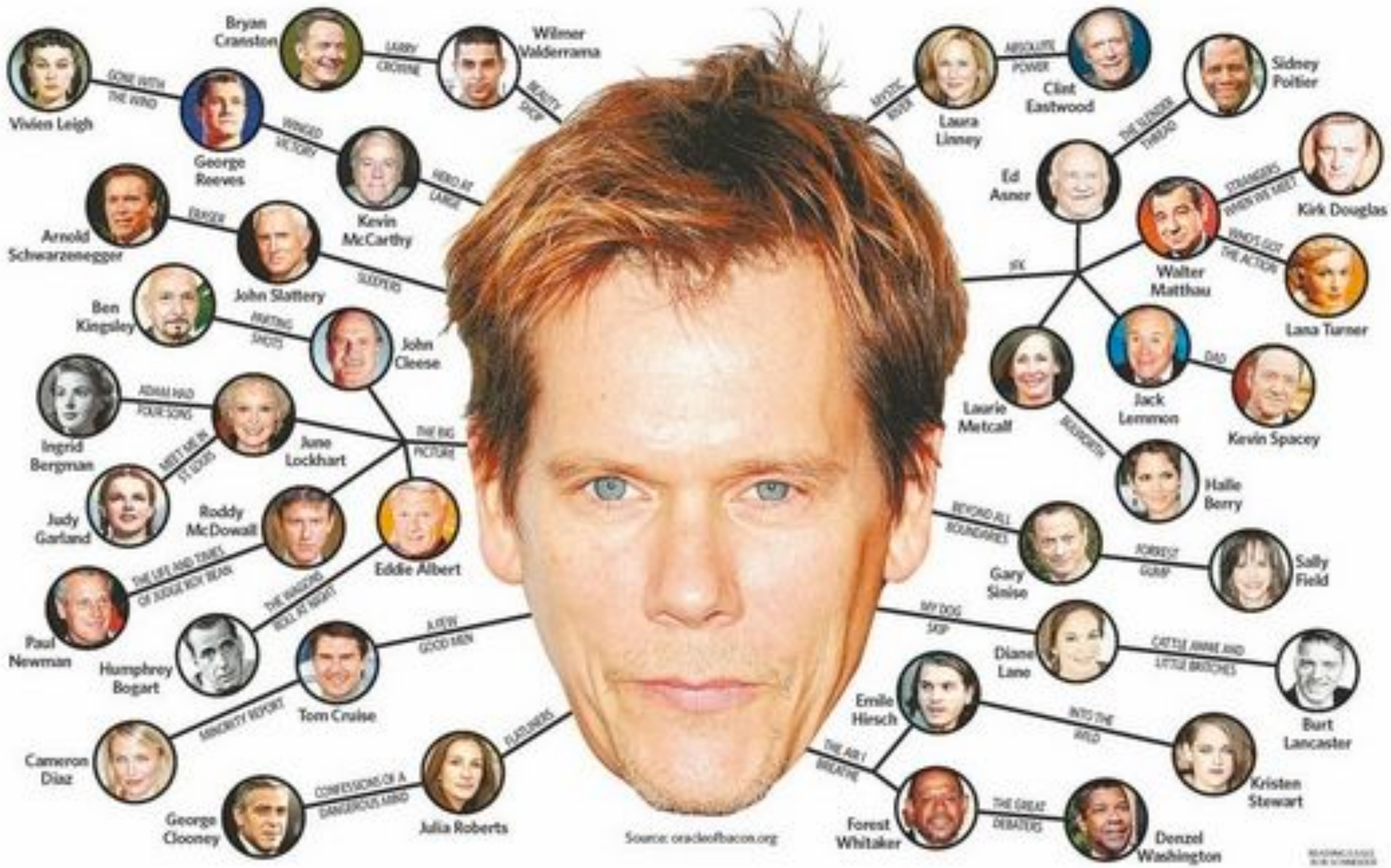
Camino(s) mínimo(s)

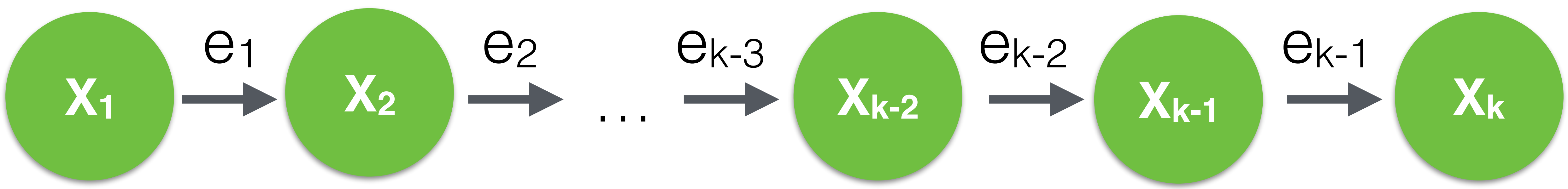

A
Direcciones o Lugares
B
Direcciones o Lugares
Mostrar opciones »
Villa Ortúzar
Triunvirato Av.
Asentamiento Fraga
Bo. Colegiales
Vega, Niceto, Cnel. Av.
Parque Chas
De los Constituyentes Av.
FEDERICO LACROZE
Chacarita
Cementerio de la Chacarita
JOSE ARTIGAS
Dorrego Av.
Córdoba Av.
Agronomía
PEDRO ARATA
DR. FRANCISCO BEIRO
Facultad de Ciencias Veterinarias
Bo. Rawson
Paternal
CHACARITA
Villa Crespo
Metrobus
LA PATERNAL
Asentamiento Biarritz y Espino
Paternal
Justo, Juan B. Av.
Corrientes Av.
Villa Crespo
VILLA DEL PARQUE
San Martín Av.
Villa Del Parque
Patricias Argentinas Av.
Díaz Velez Av.
Alvarez Jonte Av.
Villa Gral. Mitre
Gaona Av.
Acoyte Av.
Bo. Hogar Obrero
Nazca Av.
Villa Santa Rita
Caballito

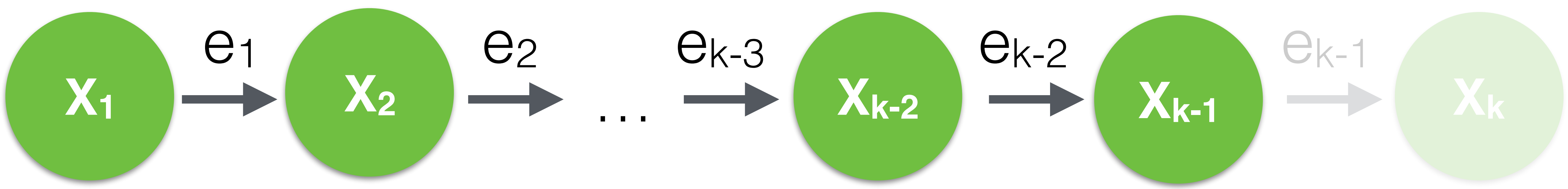

HIT FARM WITH SWORD
FARM CATCHES FIRE
MEMEBASE.com


Depot

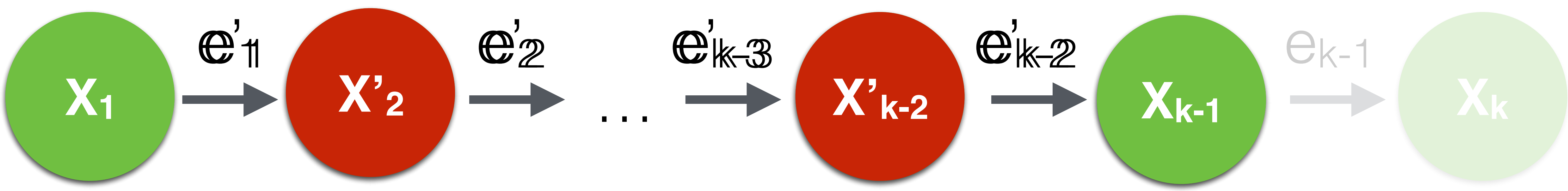

Vivien Leigh
GONE WITH THE WIND
George Reeves
Bryan Cranston
LARRY CROWNE
Wilmer Valderrama
BEAUTY SHOP
WINGED VICTORY
Kevin McCarthy
HERO AT LARGE
Arnold Schwarzenegger
ERASER
John Slattery
SLEEPERS
Ben Kingsley
PARTING SHOTS
John Cleese
Ingrid Bergman
ADAM HAD FOUR SONS
June Lockhart
MEET ME IN ST. LOUIS
Judy Garland
THE BIG PICTURE
Roddy McDowall
Eddie Albert
Paul Newman
THE LIFE AND TIMES OF JUDGE ROY BEAN
Humphrey Bogart
THE WAGONS ROLL AT NIGHT
Tom Cruise
A FEW GOOD MEN
Cameron Diaz
MINORITY REPORT
George Clooney
CONFESSIONS OF A DANGEROUS MIND
Julia Roberts
FLATLINERS
Source: oracleofbacon.org
MYSTIC RIVER
Laura Linney
ABSOLUTE POWER
Clint Eastwood
Sidney Poitier
THE SLENDER THREAD
Ed Asner
JFK
Kirk Douglas
STRANGERS WHEN WE MEET
Walter Matthau
WHO'S GOT THE ACTION
Lana Turner
Laurie Metcalf
Jack Lemmon
DAD
Kevin Spacey
Halle Berry
BEYOND ALL BOUNDARIES
Gary Sinise
FORREST GUMP
Sally Field
MY DOG SKIP
Diane Lane
CATTLE ANNIE AND LITTLE BRITCHES
Burt Lancaster
Emile Hirsch
INTO THE WILD
Kristen Stewart
THE AIR I BREATHE
Forest Whitaker
THE GREAT DEBATERS
Denzel Washington

Principio de optimalidad

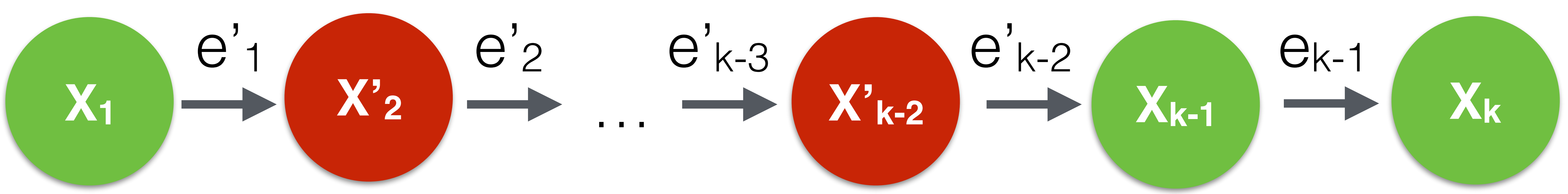

x_1
e_1
x_2
e_2
...
e_{k-3}
x_{k-2}
e_{k-2}
x_{k-1}
e_{k-1}
x_k



x_1
e_1
x_2
e_2
...
e_{k-3}
x_{k-2}
e_{k-2}
x_{k-1}
e_{k-1}
x_k

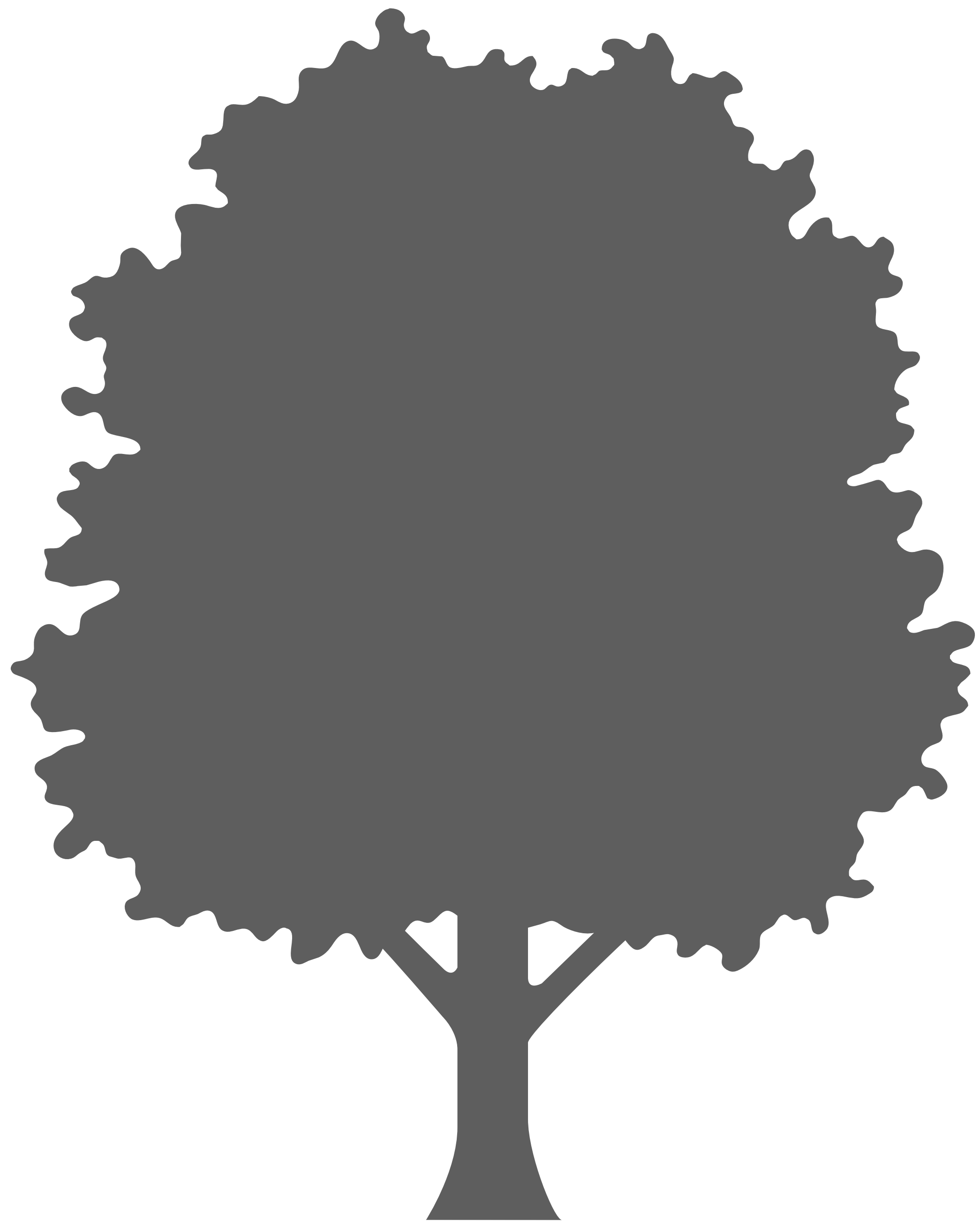


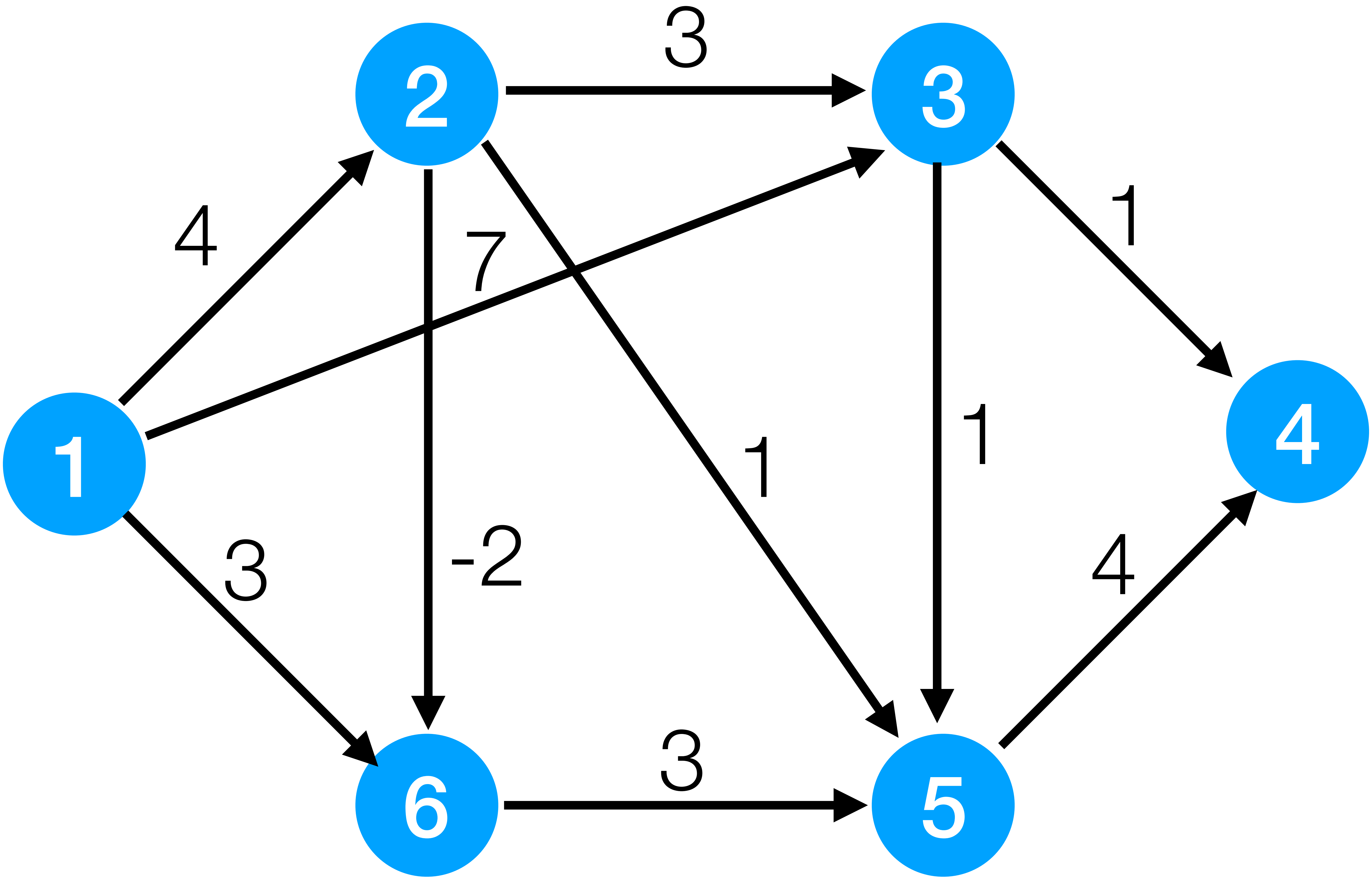
X1
e'1
X'2
e'2
...
e'k-3
X'k-2
e'k-2
Xk-1
ek-1
Xk



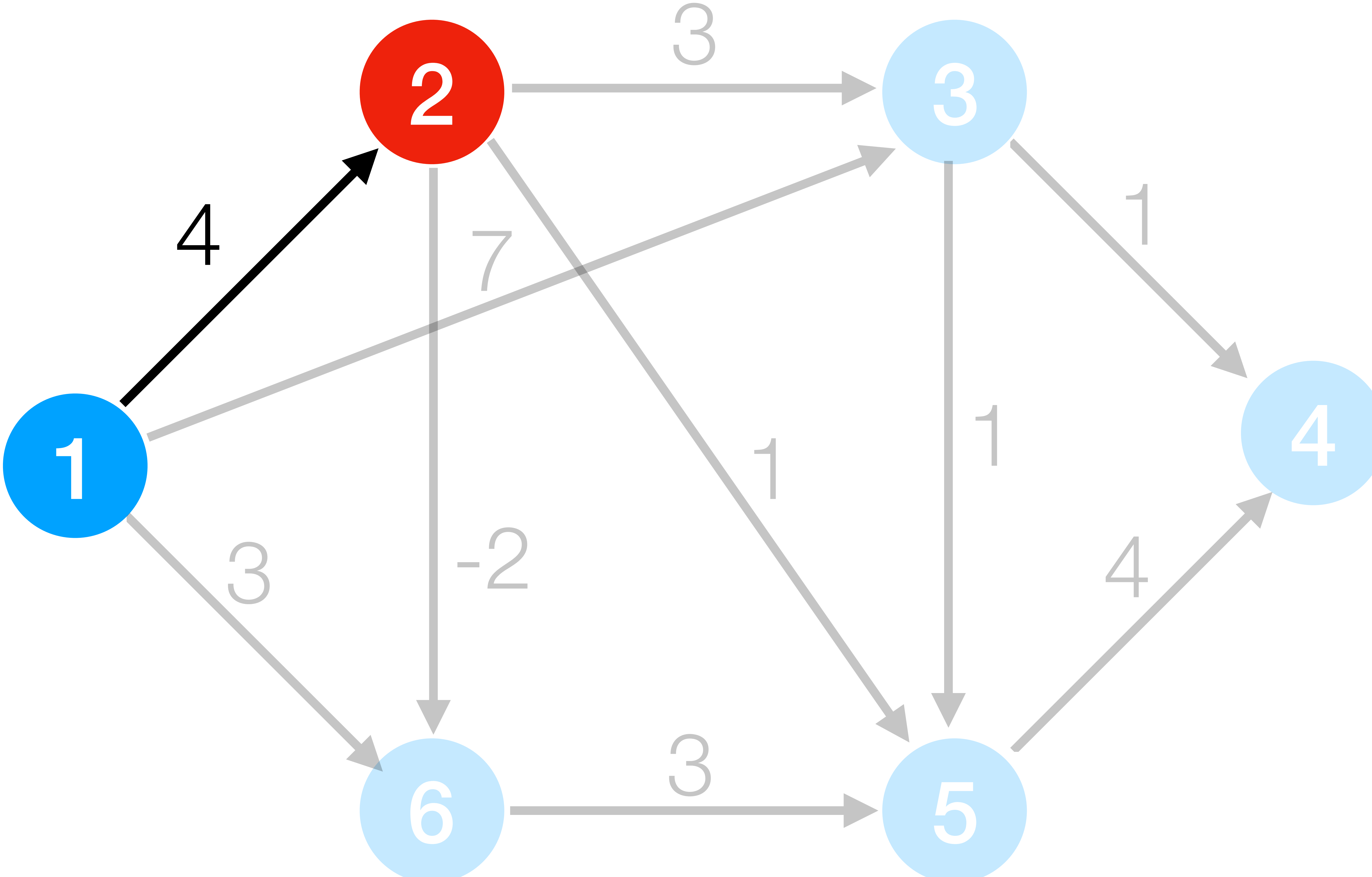
X1
e'1
X'2
e'2
…
e'k-3
X'k-2
e'k-2
Xk-1
ek-1
Xk

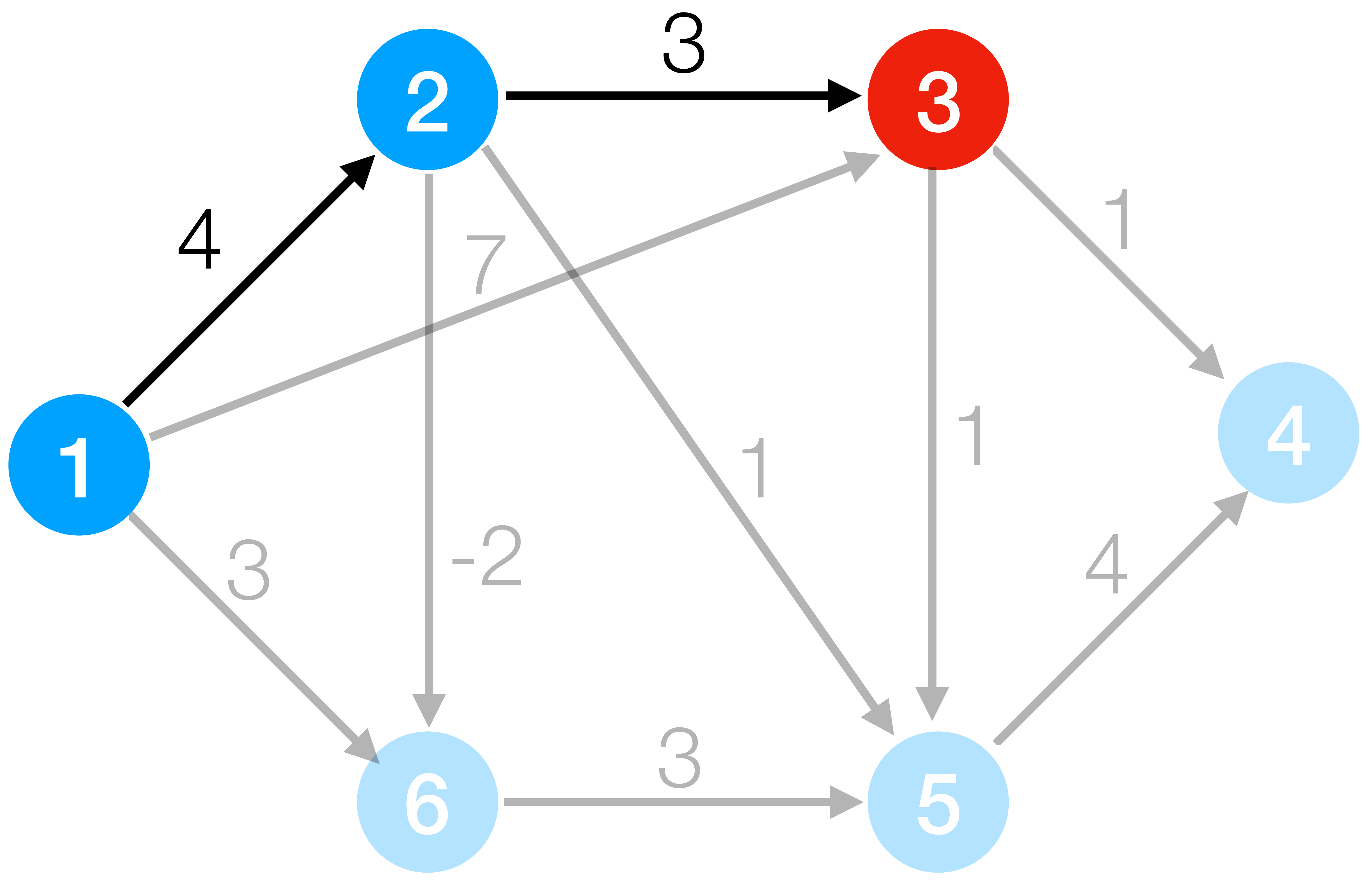
Árboles de caminos mínimos

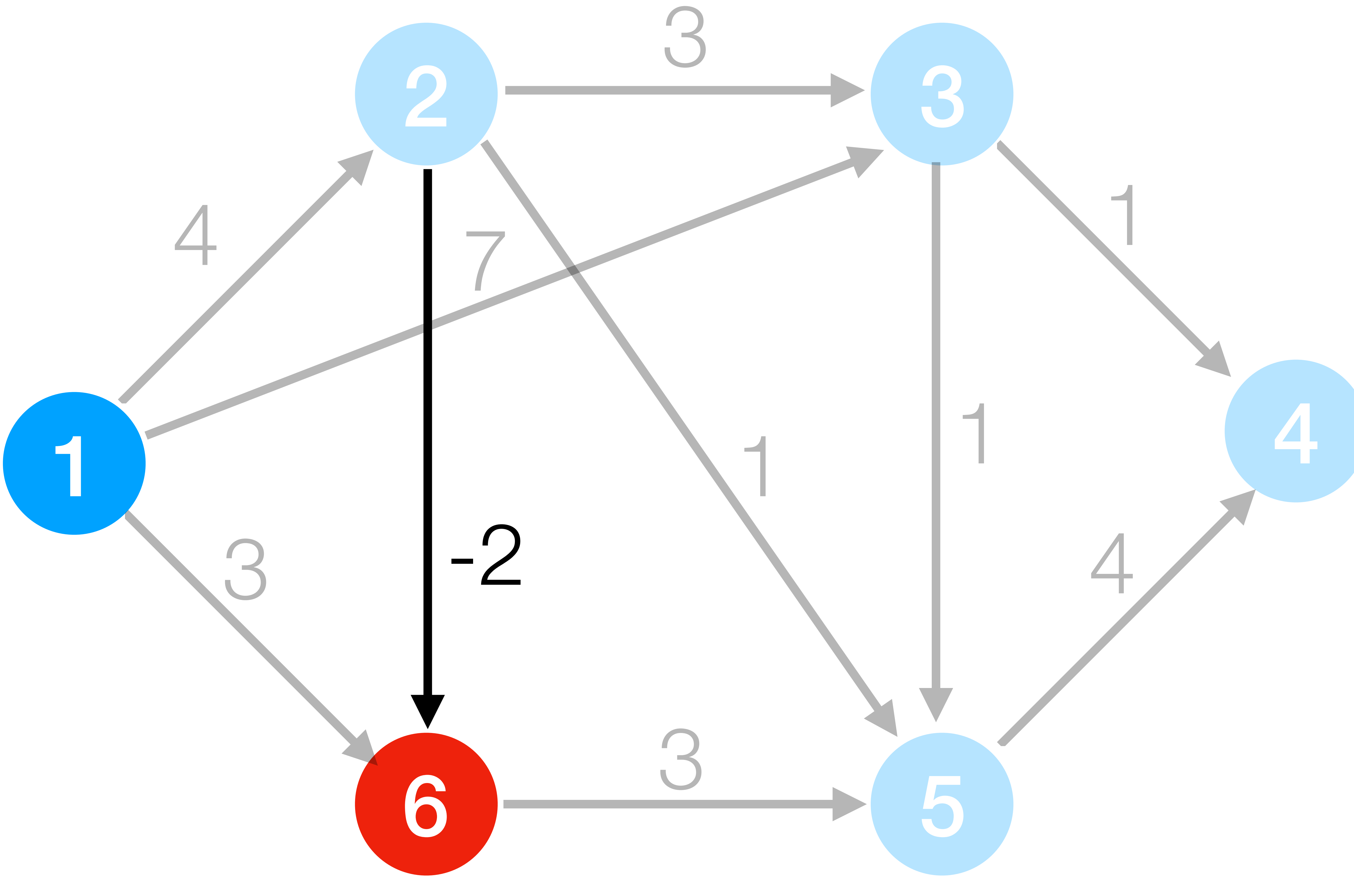



1
2
3
4
5
6
4
3
7
-2
3
3
1
1
1
4

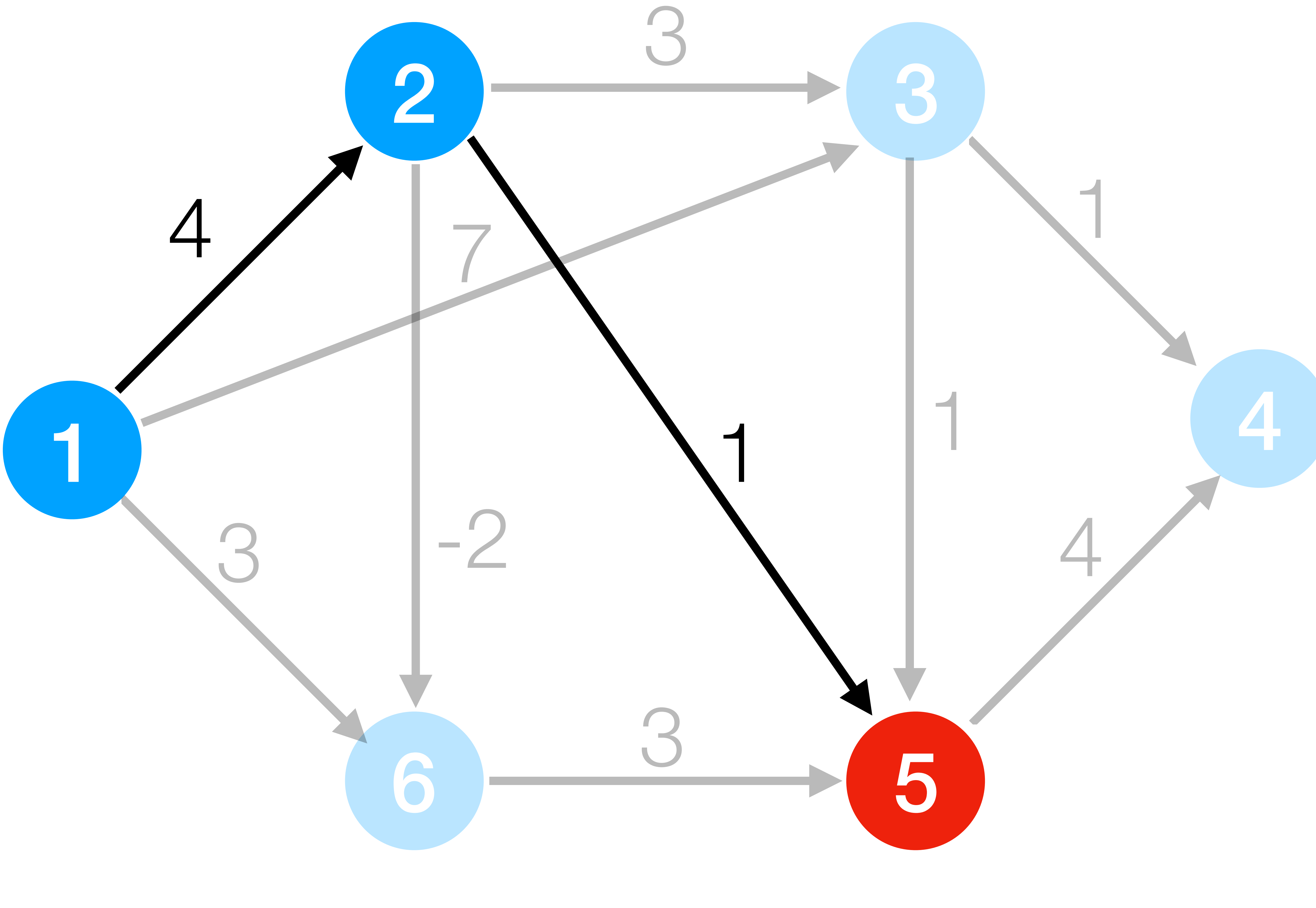
¿Cuál es el camino mínimo desde 1 hasta…?

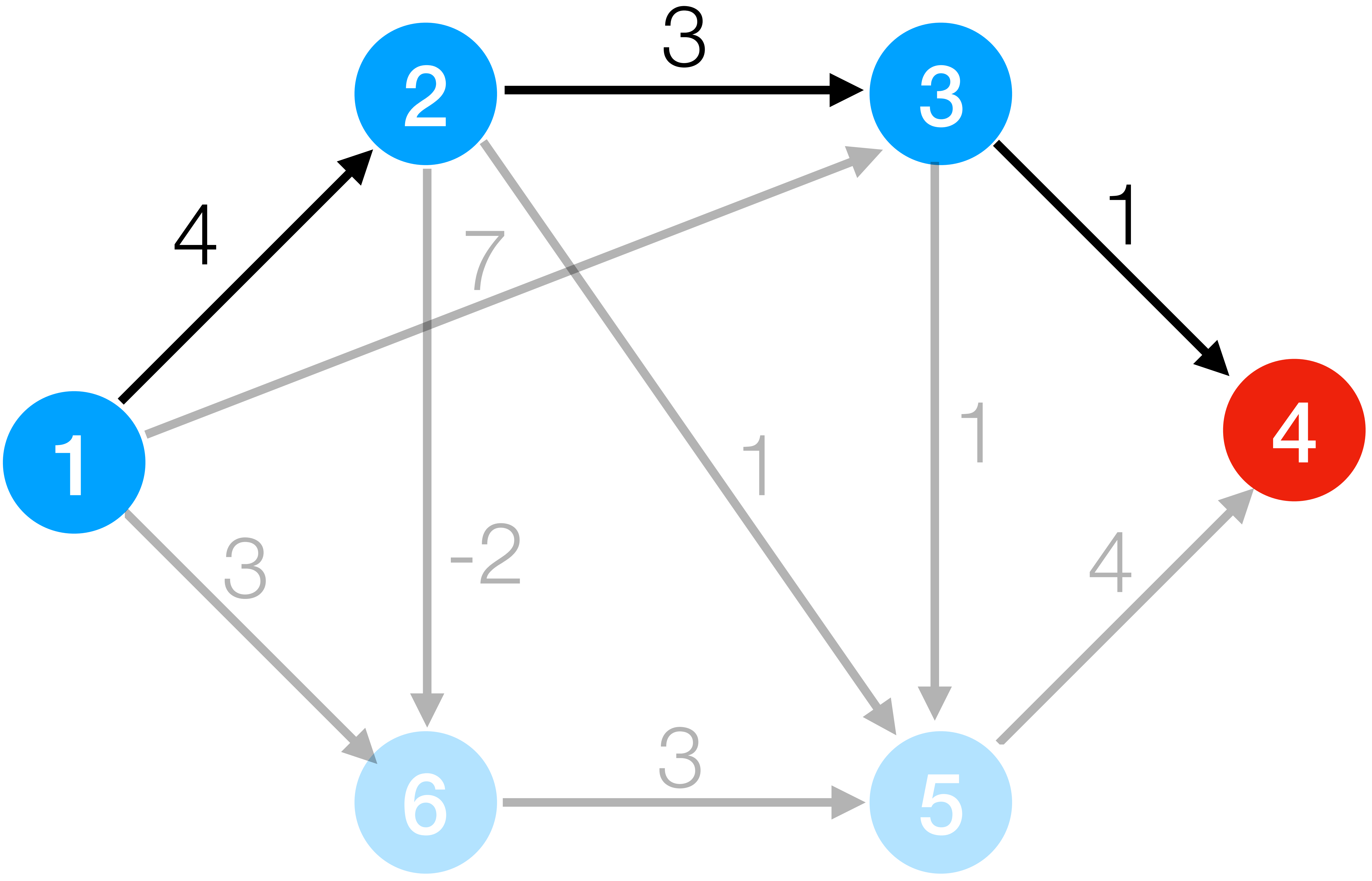

1
2
3
4
5
6
4
3
7
-2
1
1
1
4
3
3


1
2
3
4
5
6
4
3
7
-2
1
1
1
4
3
3



1
2
3
4
5
6
4
3
3
7
-2
1
1
1
4
3

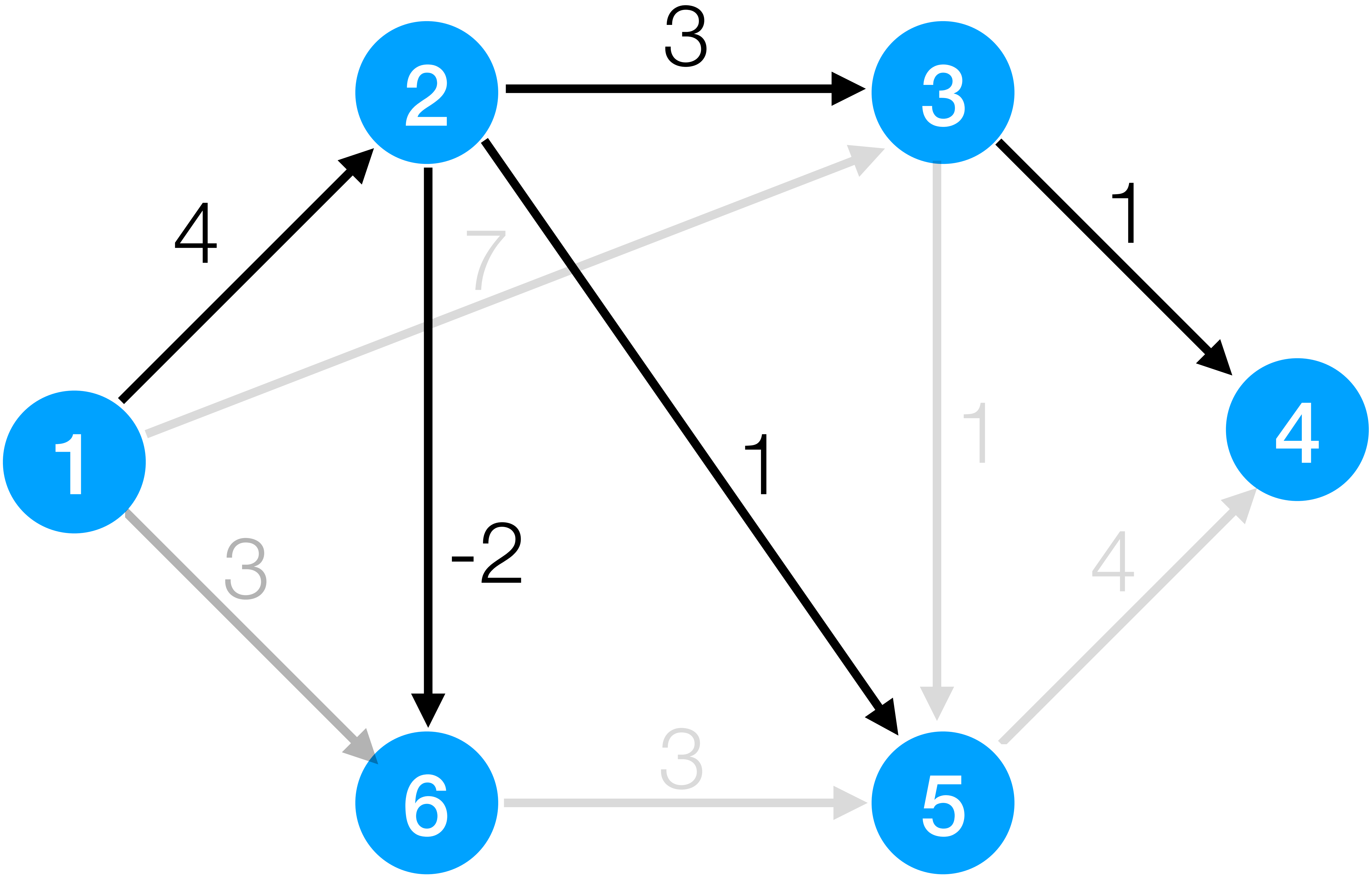

1
2
3
4
5
6
4
3
7
1
1
1
-2
3
4
3



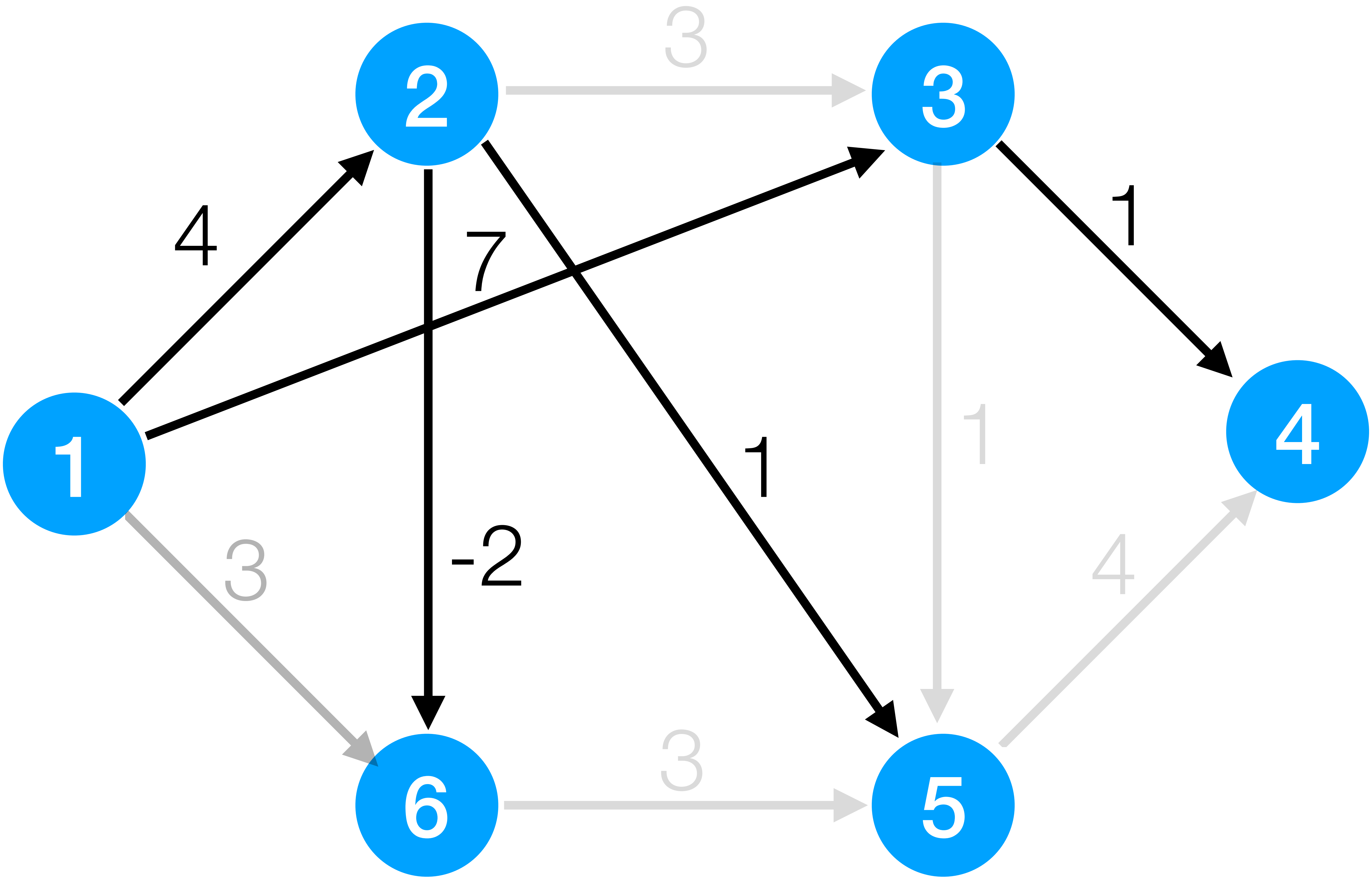
1
2
3
4
5
6
4
3
1
7
-2
1
1
3
3
4

¿Y si dejamos solo esos caminos?

Árbol de caminos mínimos



Pero… ¿es único el
árbol de caminos mínimos?



1
2
3
4
5
6
4
3
7
-2
3
1
1
1
4
3

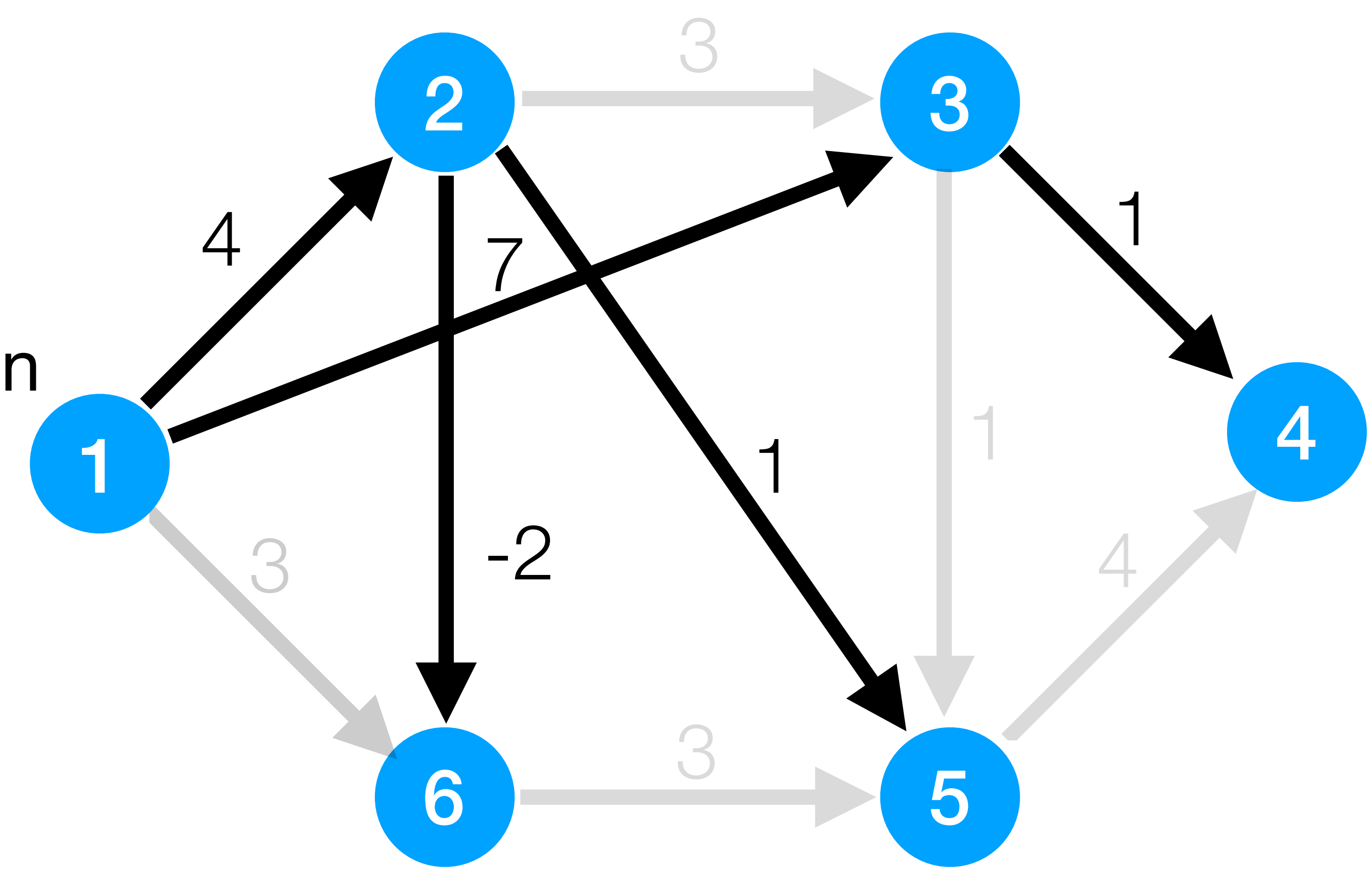
¿Cómo lo representamos?

Índice de predecesores

- Por cada vértice nos guardamos quién es su predecesor en el árbol de caminos mínimos. Notar que es único porque es un árbol.

- El parent del vértice raíz es un valor especial que no sea ningún vértice (e.g. 0).

- parent[1] = 0 parent[2] = 1
  parent[3] = 1 parent[4] = 3
  parent[5] = 2 parent[6] = 2



parent[1] = 0 parent[2] = 1 parent[3] = 1
parent[4] = 3 parent[5] = 2 parent[6] = 2

- Sea P = 1…v el camino mínimo desde 1 hasta el vértice v.

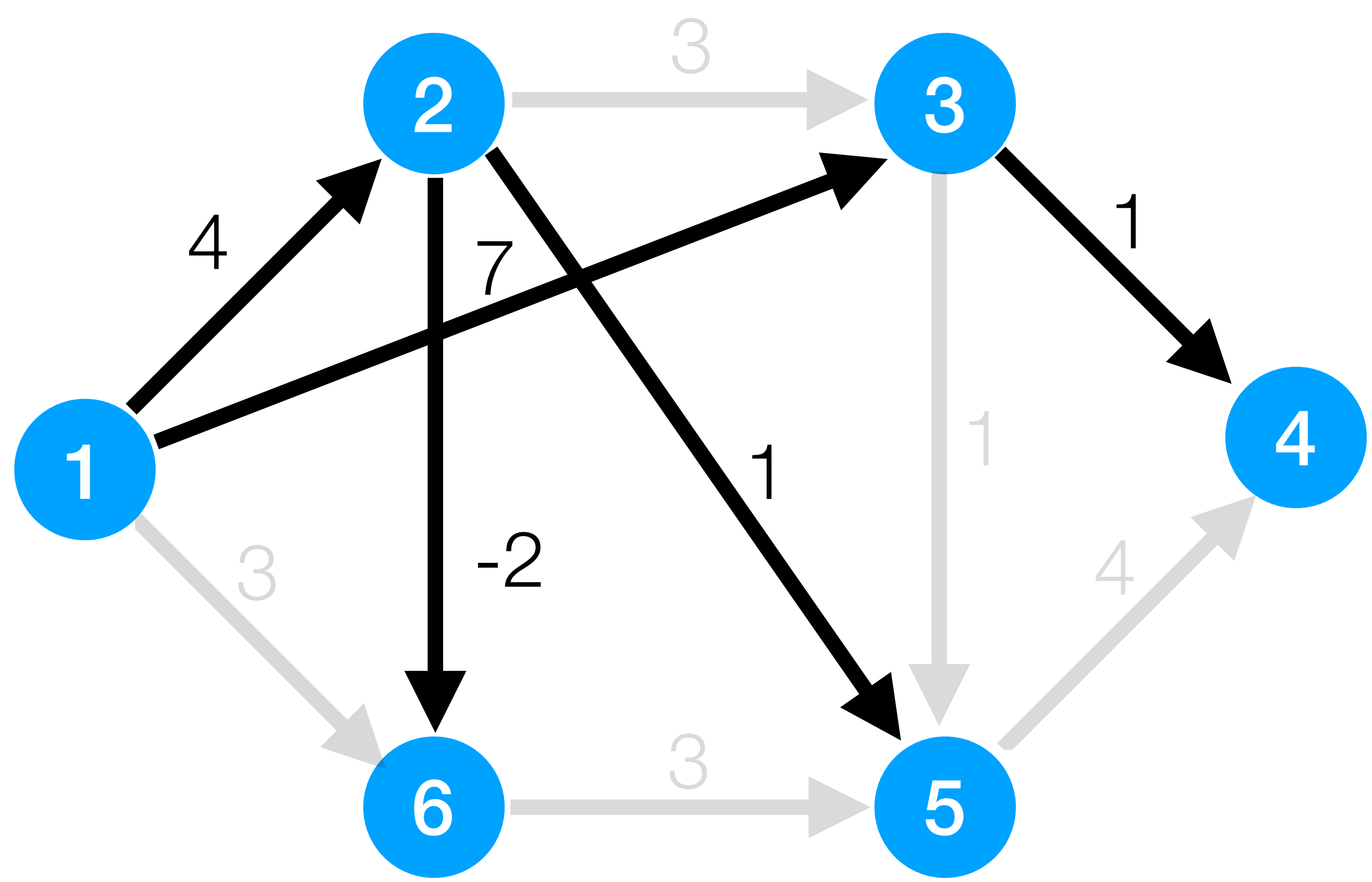
- ¿Qué complejidad tiene reconstruir P?

- # O(|P|)

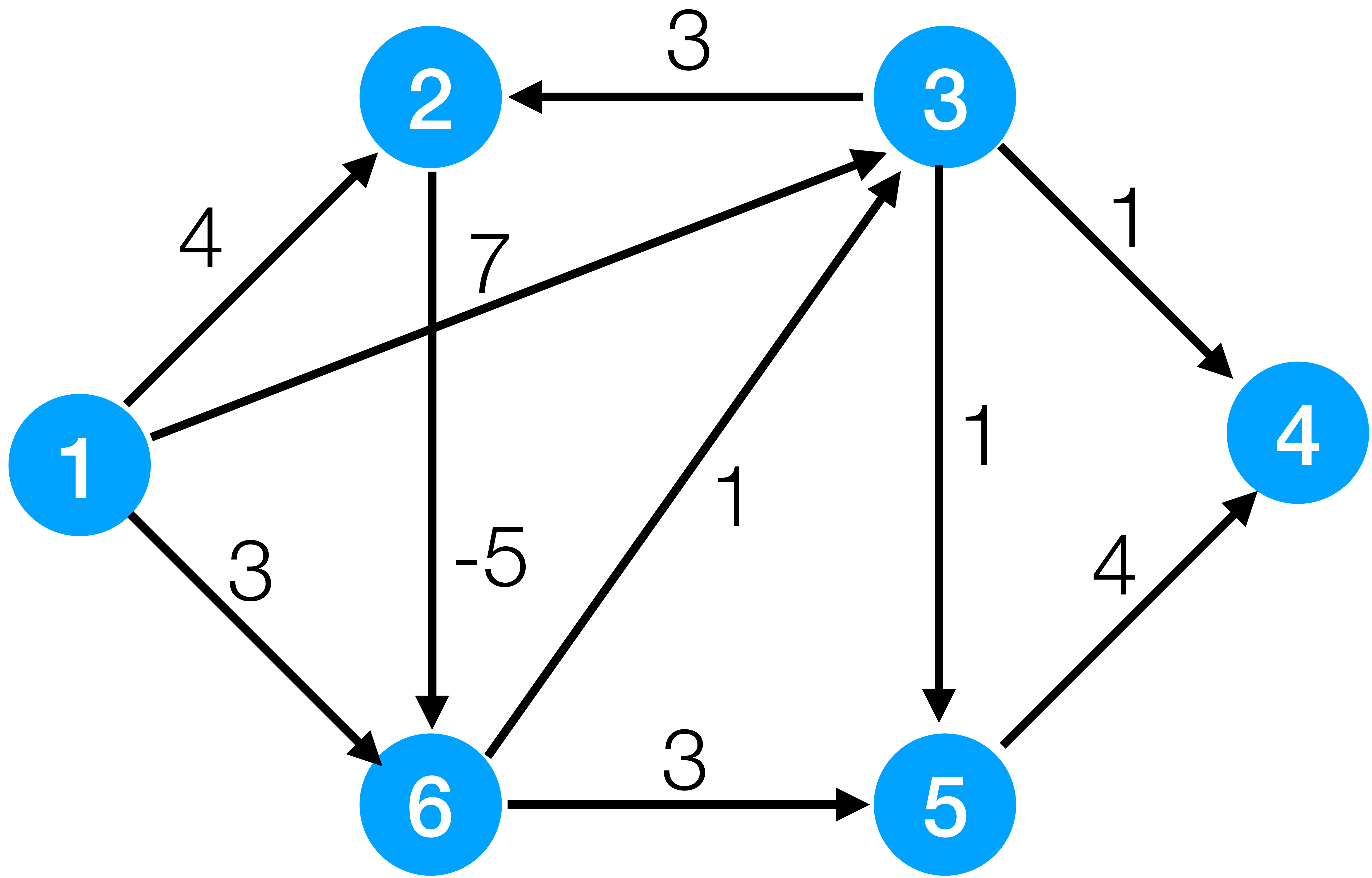
- ¿Qué complejidad tiene calcular la distancia de P?

- # O(|P|) ó O(|P|+m)

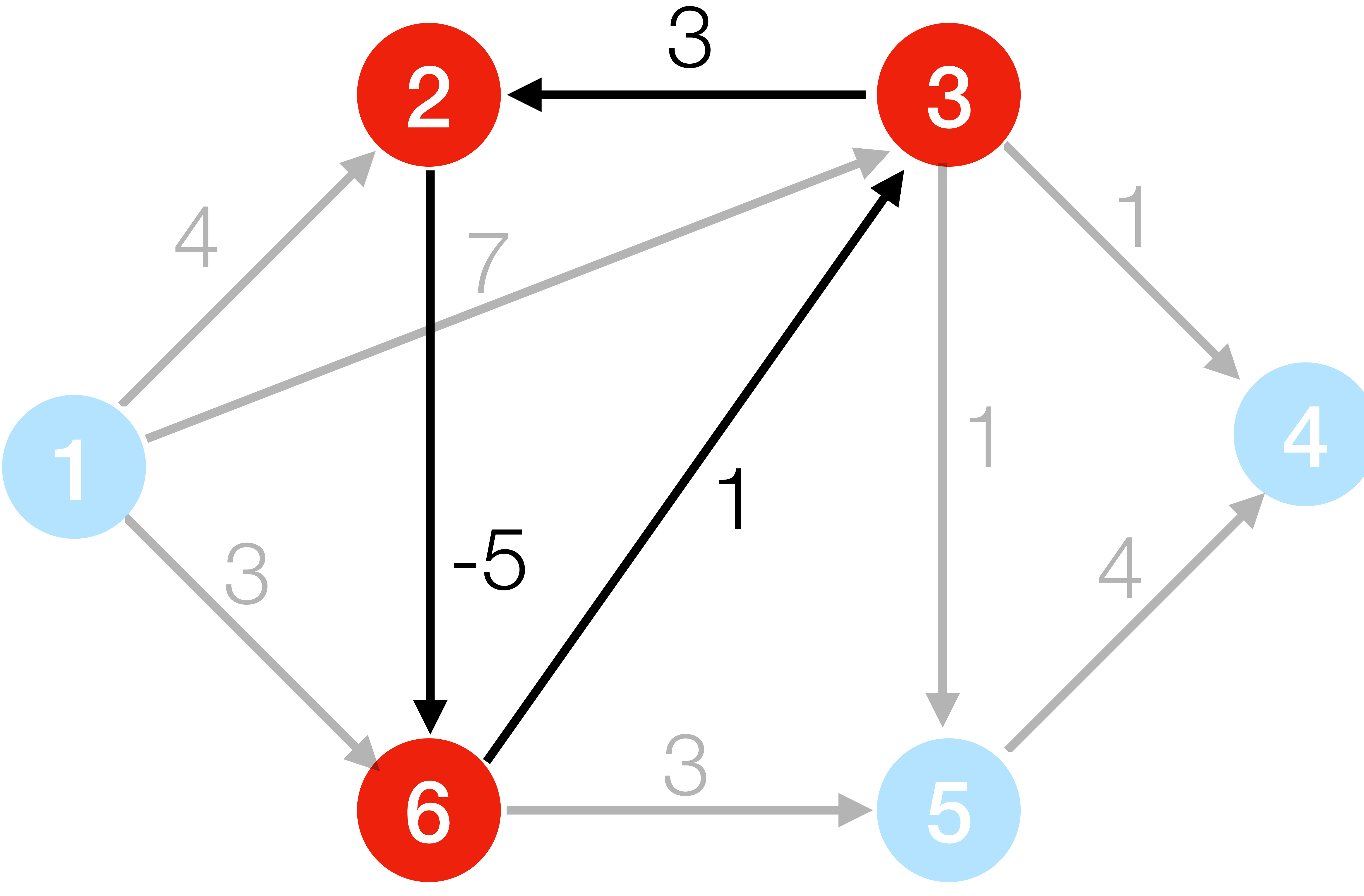
Matriz de adyacencia   Listas de adyacencia



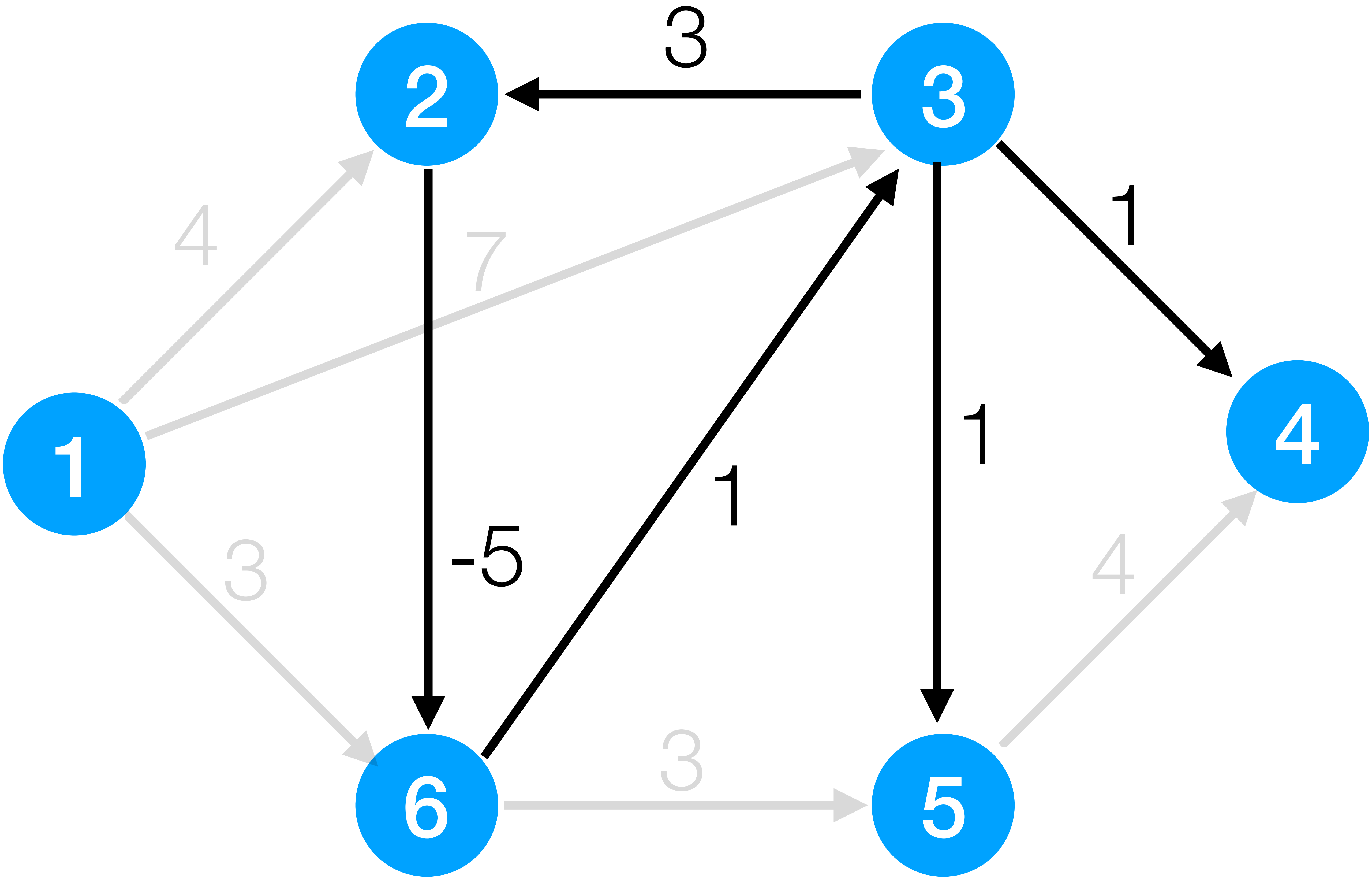
¿Y si hay ciclos negativos qué pasa?



1
2
3
4
5
6
4
3
7
-5
3
1
1
1
4
3



1
2
3
4
5
6
4
3
7
-5
1
1
1
4
3
3



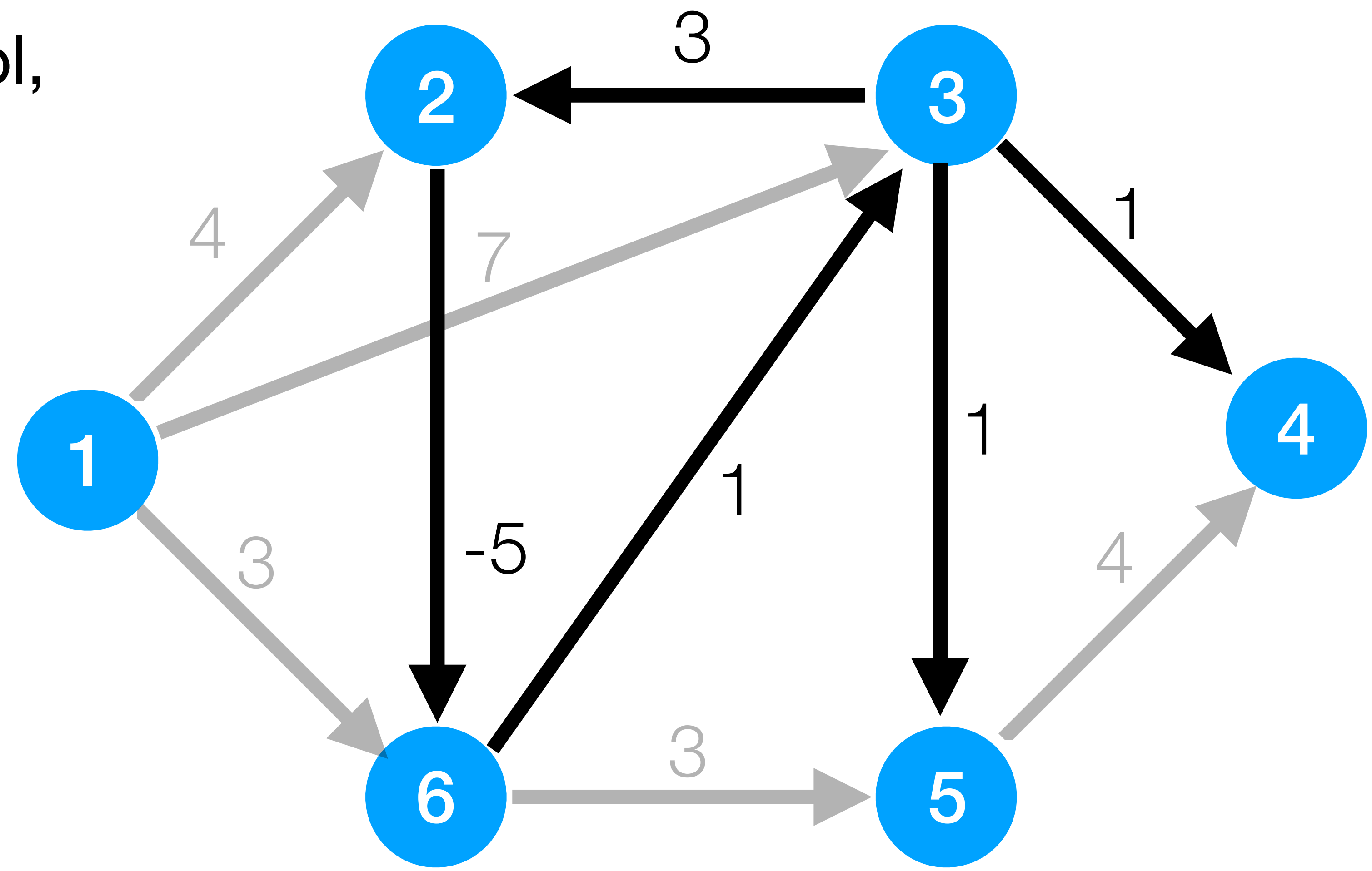
1
2
3
4
5
6
3
4
7
1
1
1
-5
3
4
3

parent[1] = 0 parent[2] = 3 parent[3] = 6
parent[4] = 3 parent[5] = 3 parent[6] = 2

- Ahora parent no forma un árbol, pero si nos permite identificar los ciclos.

- ¿Qué complejidad tiene encontrar un ciclo C que contiene a un vértice v?

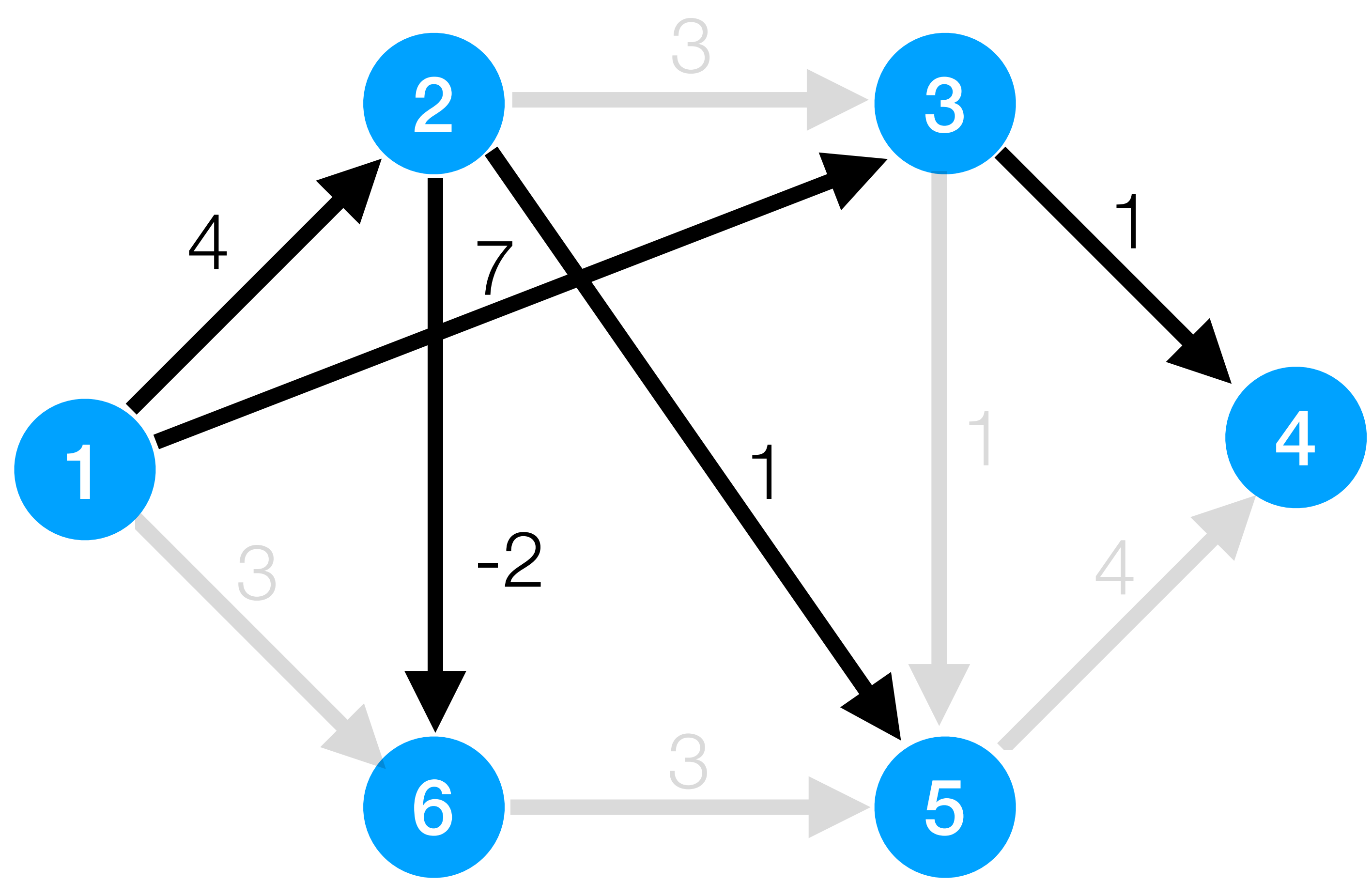
- O(|C|)



Índice de distancias

- Por cada vértice nos guardamos cuál es la distancia mínima desde la raíz hasta él.
- d[1] = 0 d[2] = 4 d[3] = 7
  d[4] = 8 d[5] = 5 d[6] = 2
- Sea P el camino mínimo desde 1 al vértice v.
- ¿Cuál es la complejidad de obtener la distancia de P?

- O(1)



d[1] = 0 d[2] = 4 d[3] = 7 d[4] = 8 d[5] = 5 d[6] = 2

- Sea P el camino mínimo desde 1 al vértice v.

- ¿Cuál es la complejidad de obtener la distancia de P?

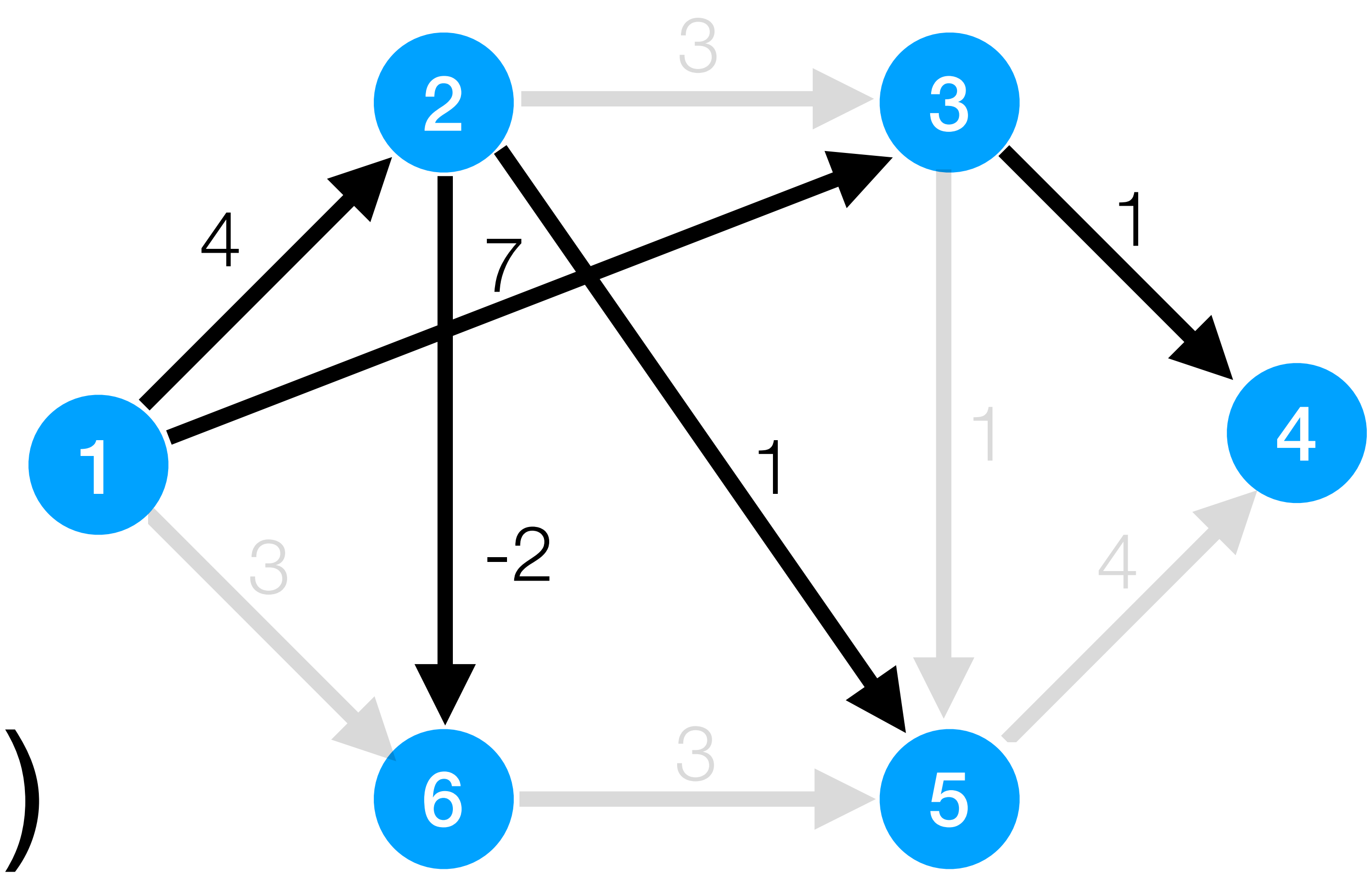
- O(1)

- ¿Cuál es la complejidad de reconstruir el camino P?

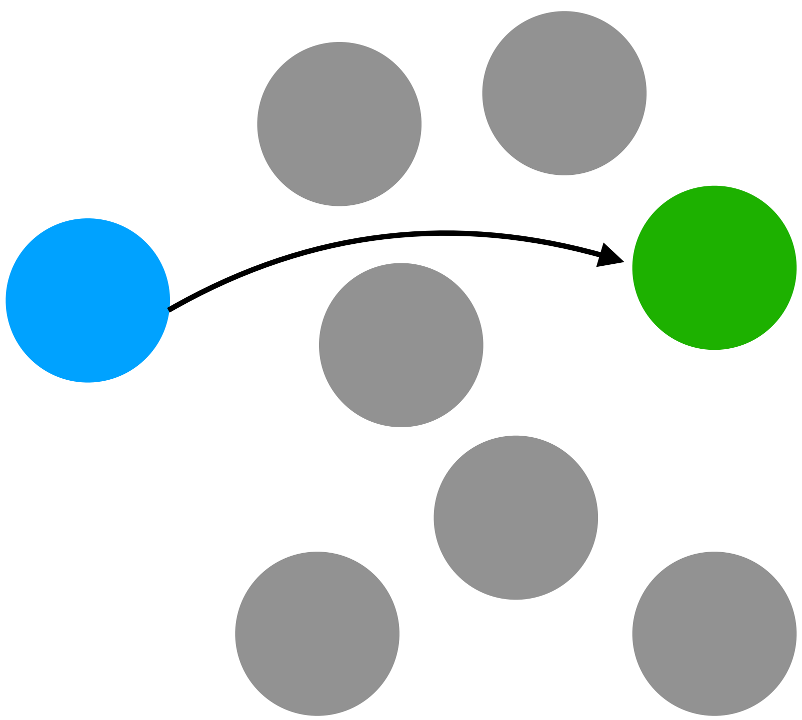
- $O(\sum_{v \in P} d_v)$ ó $O(n|P|)$

Listas de adyacencia

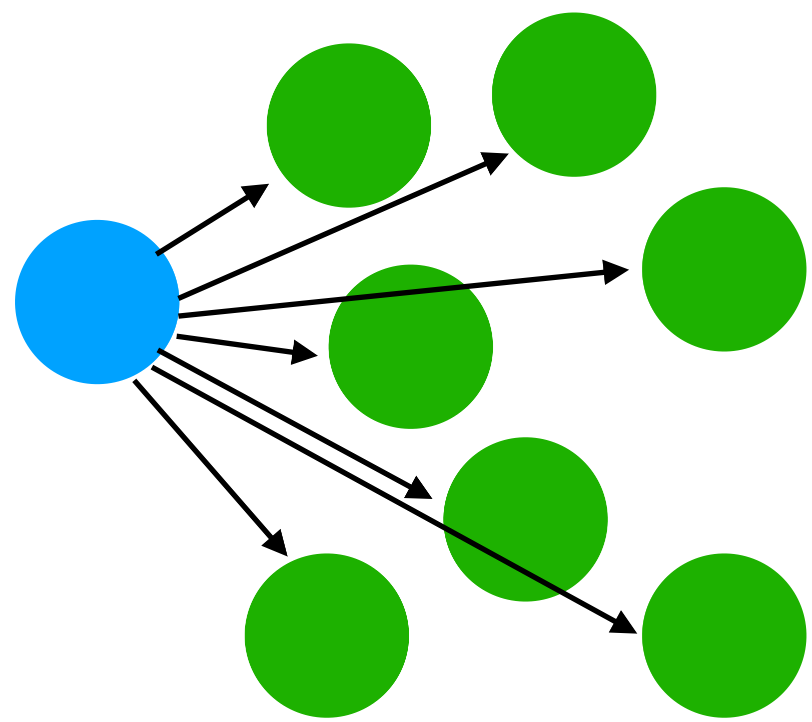
Matriz de adyacencia



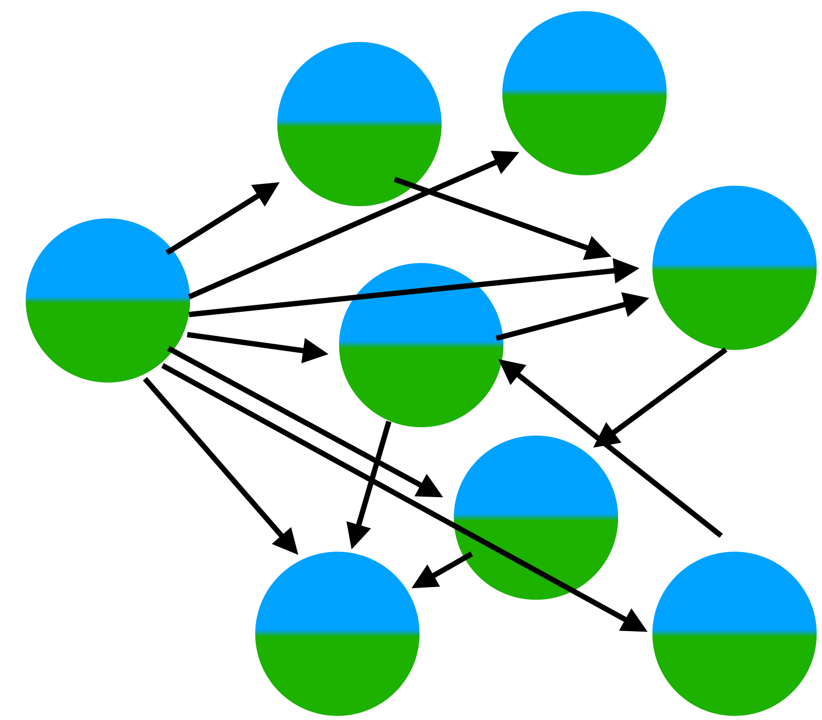
Caminos mínimos



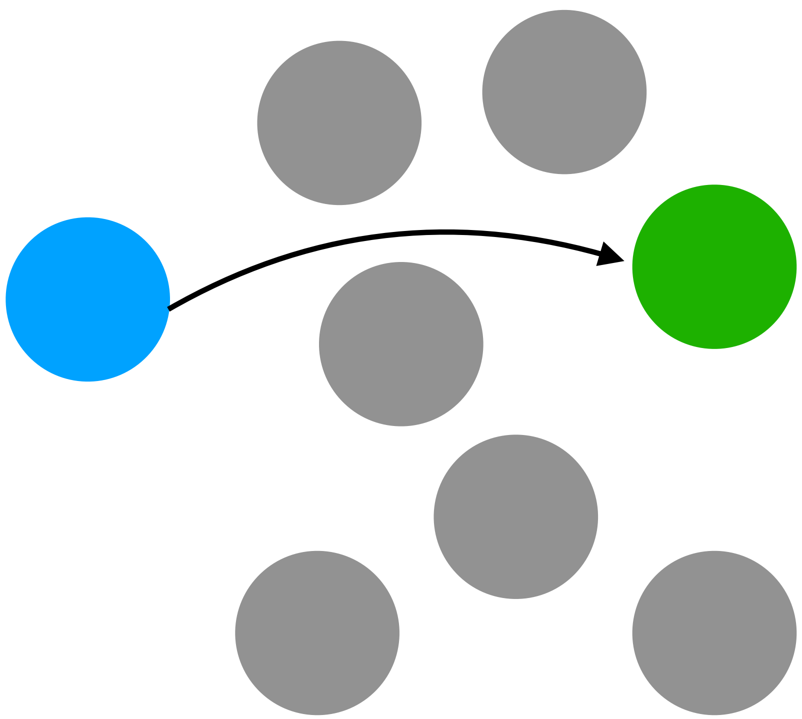
Uno a uno



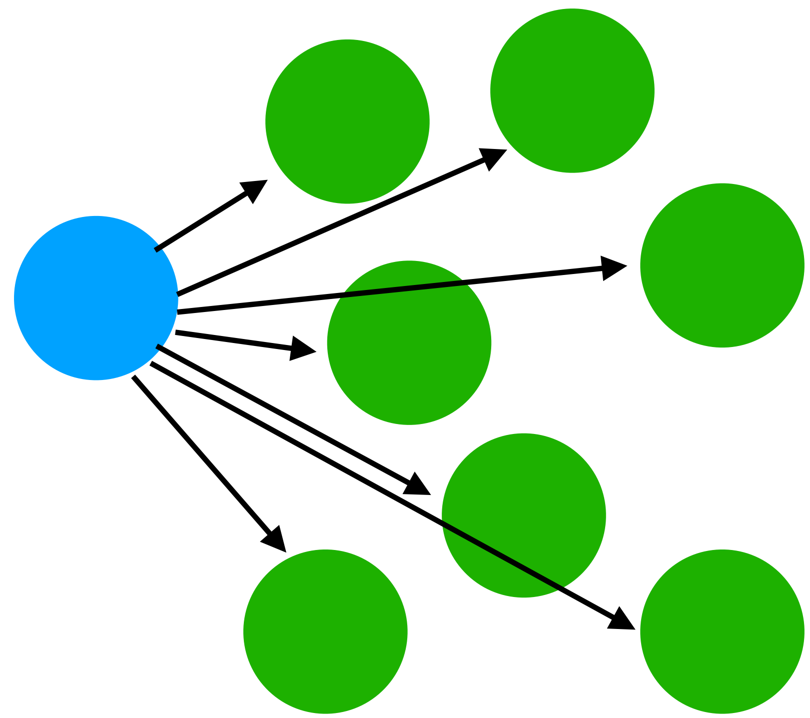
Uno a muchos



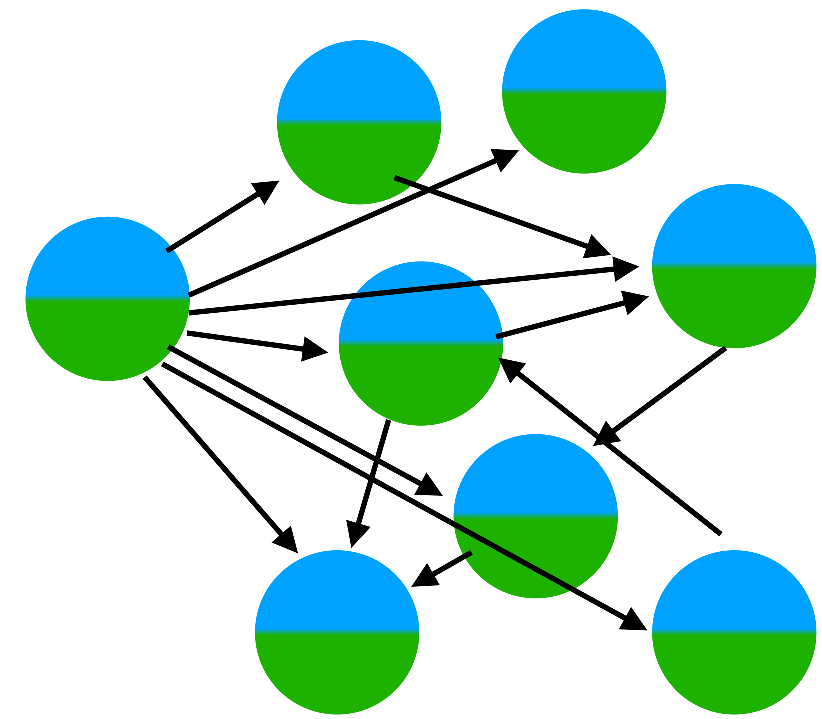
Muchos a muchos



A*



Dijkstra
Bellman-Ford
BFS



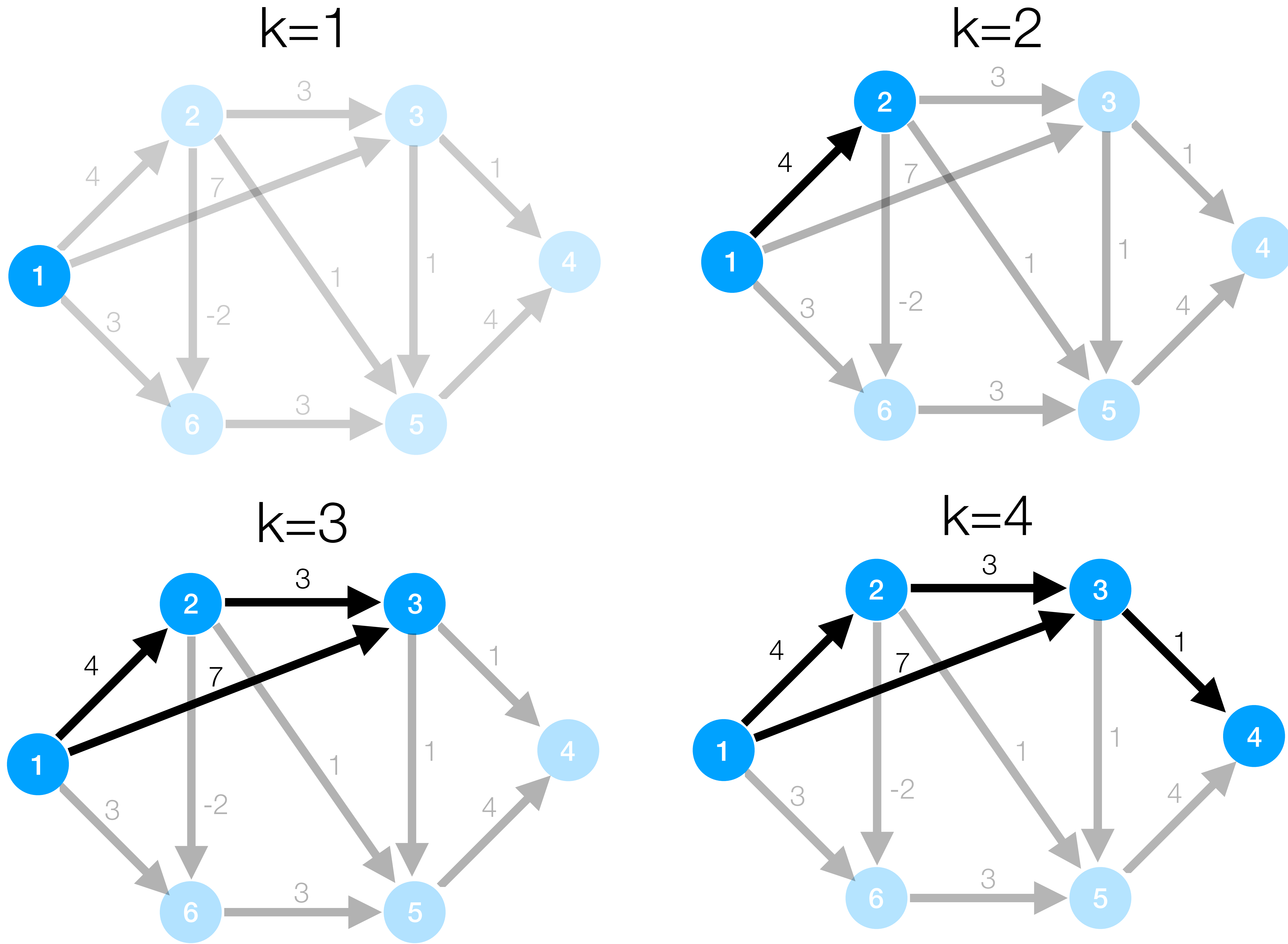
Dantzig
Floyd-Warshall

George Dantzig

- Persona importante en la historia de la computación.
- Padre de la Programación Lineal.
- Físico y matemático.
- **Creador de un algoritmo de caminos mínimos.**

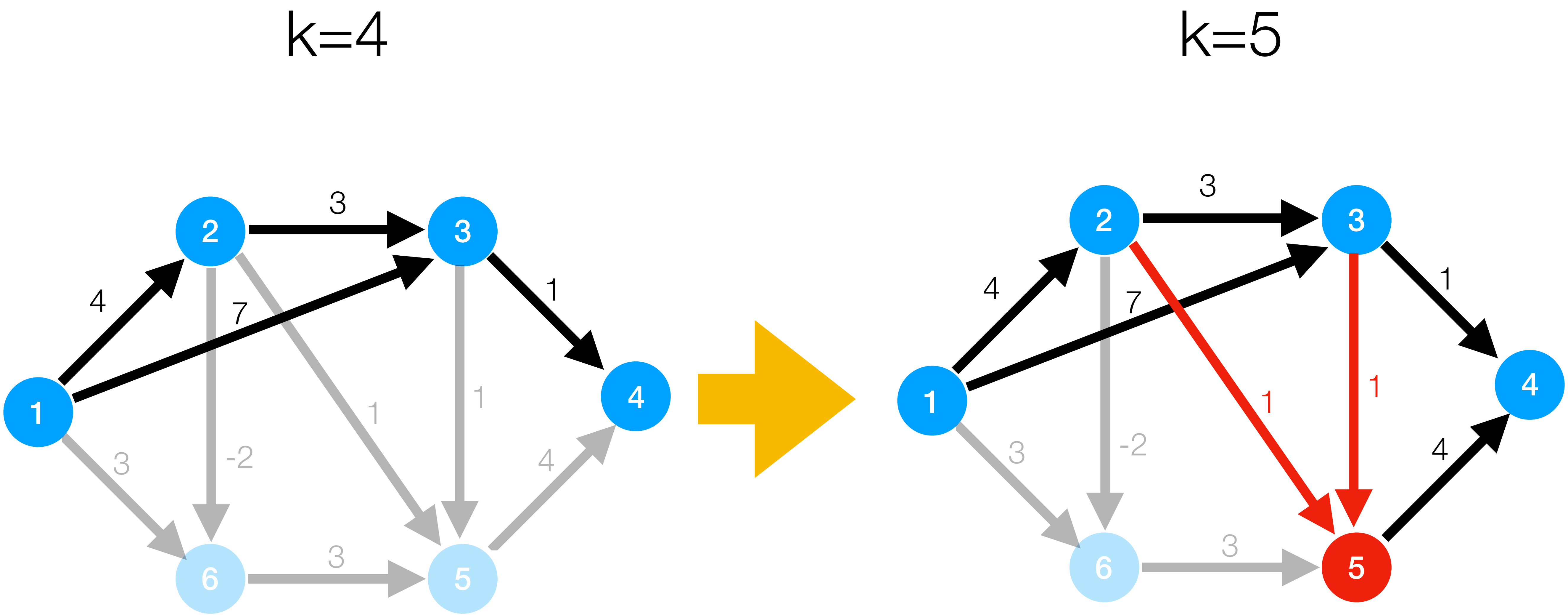
Puedo ir agregando de una responsabilidad a la vez.

Puedo ir resolviendo el problema con un subgrafo más grande cada vez.

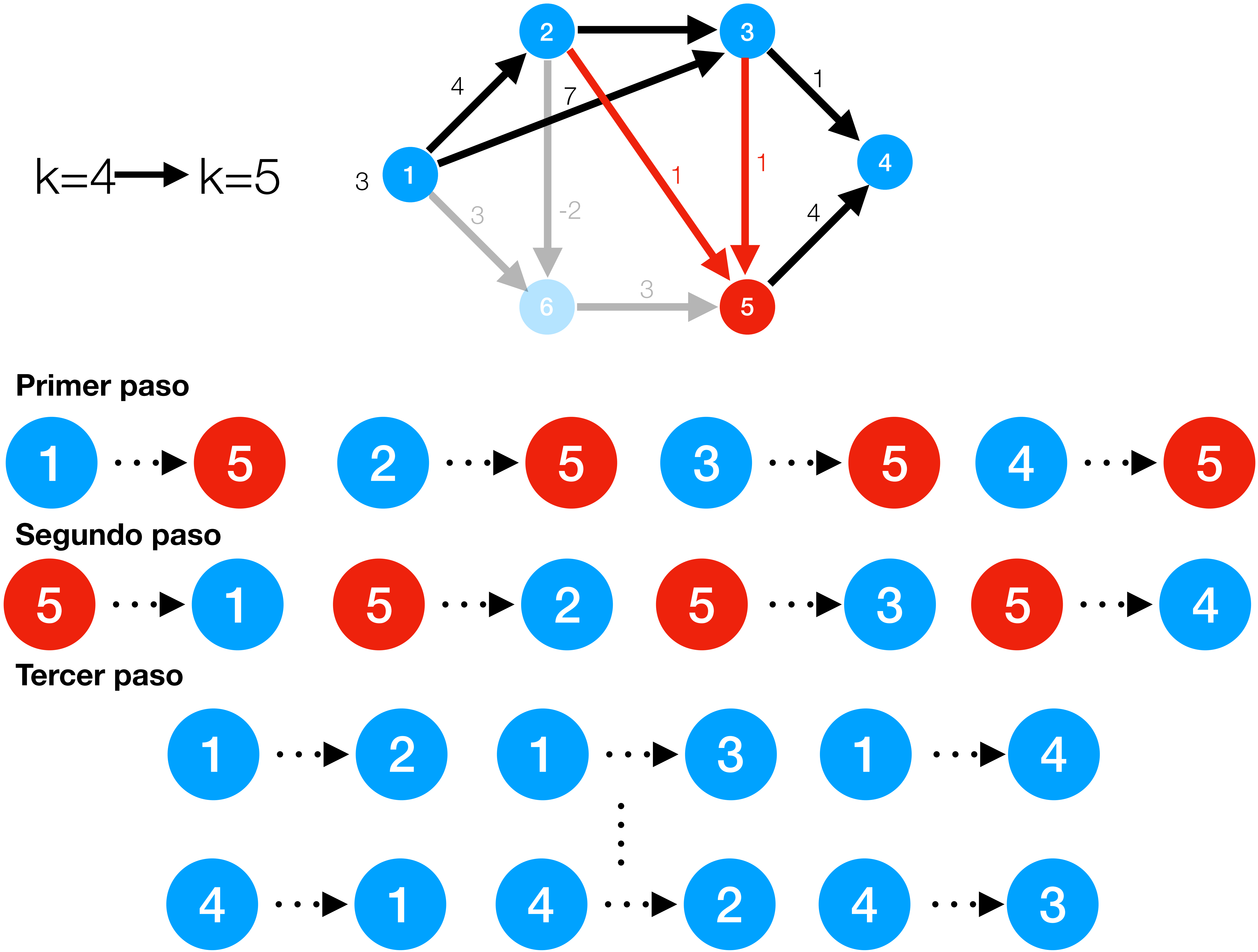


k=1
1
2
3
4
5
6
4
3
7
3
-2
1
1
1
4
3
k=2
1
2
3
4
5
6
4
3
7
3
-2
1
1
1
4
3
k=3
1
2
3
4
5
6
4
3
7
3
-2
1
1
1
4
3
k=4
1
2
3
4
5
6
4
3
7
3
-2
1
1
1
4
3

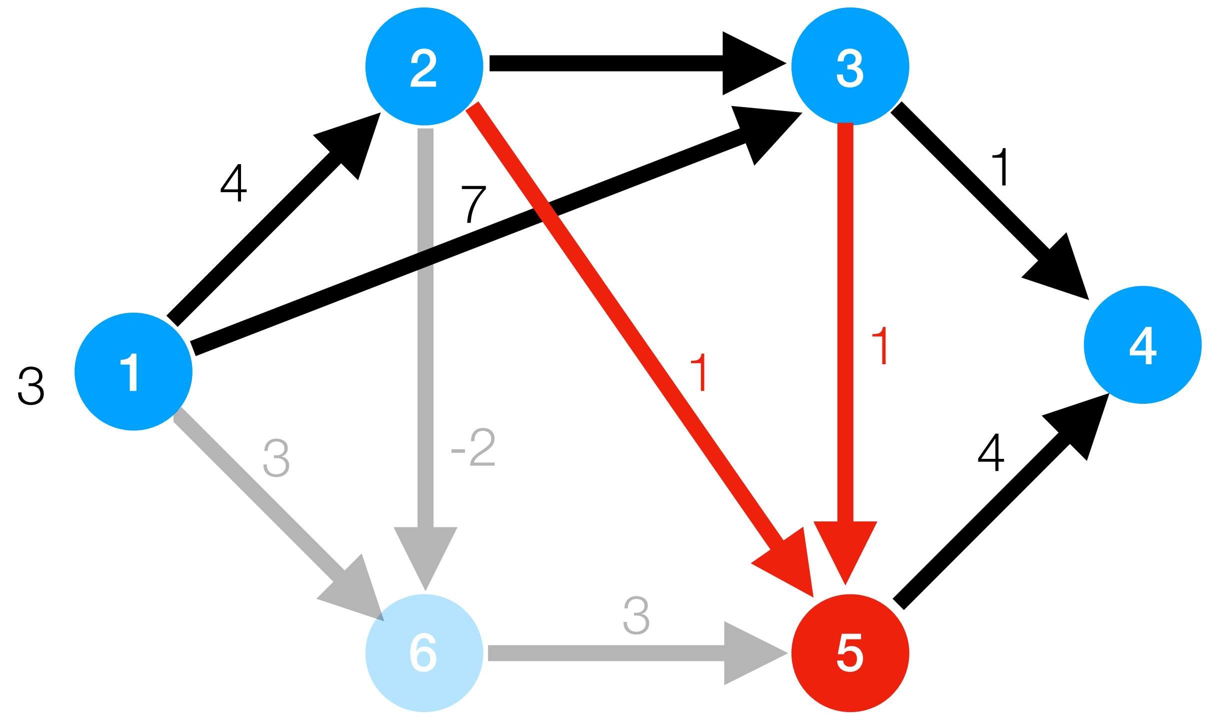
Si ya se los caminos mínimos entre todos los vértices de G_k…
¿Puedo saber los de G_{k+1}?



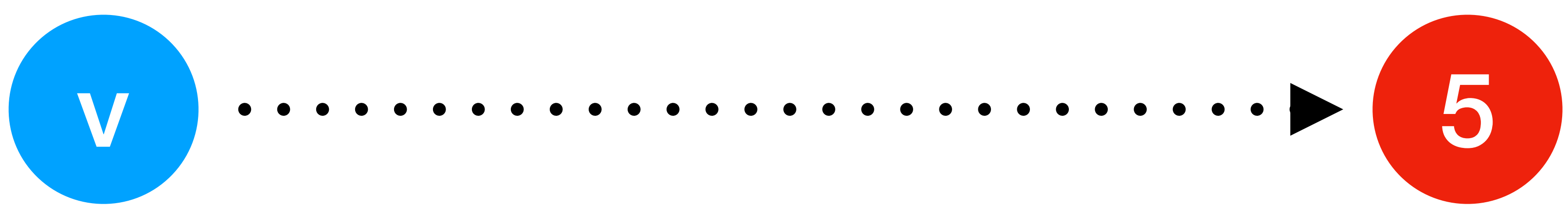
k=4
k=5
1
2
3
4
5
6
4
3
7
1
3
-2
1
1
4
3


k=4 → k=5
Primer paso
Segundo paso
Tercer paso

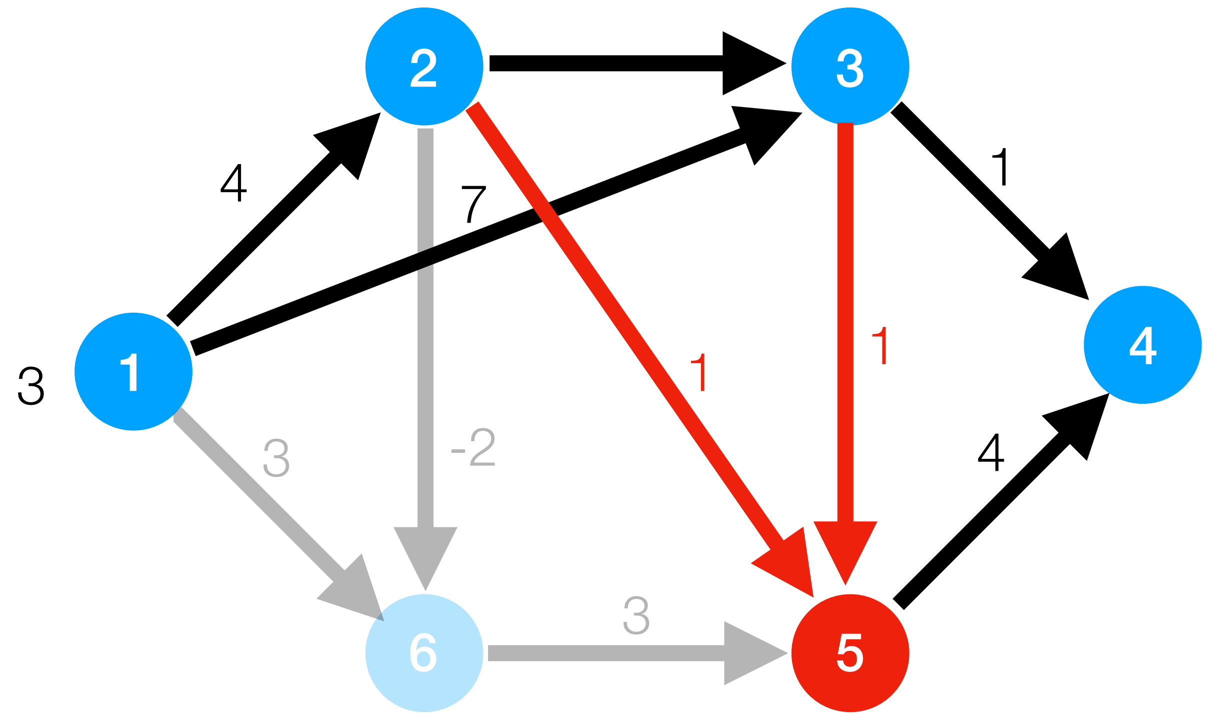
$k=4 \rightarrow k=5$



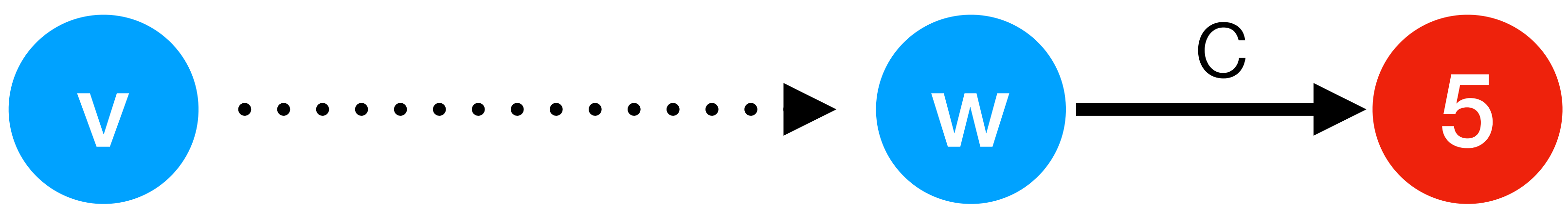
Primer paso



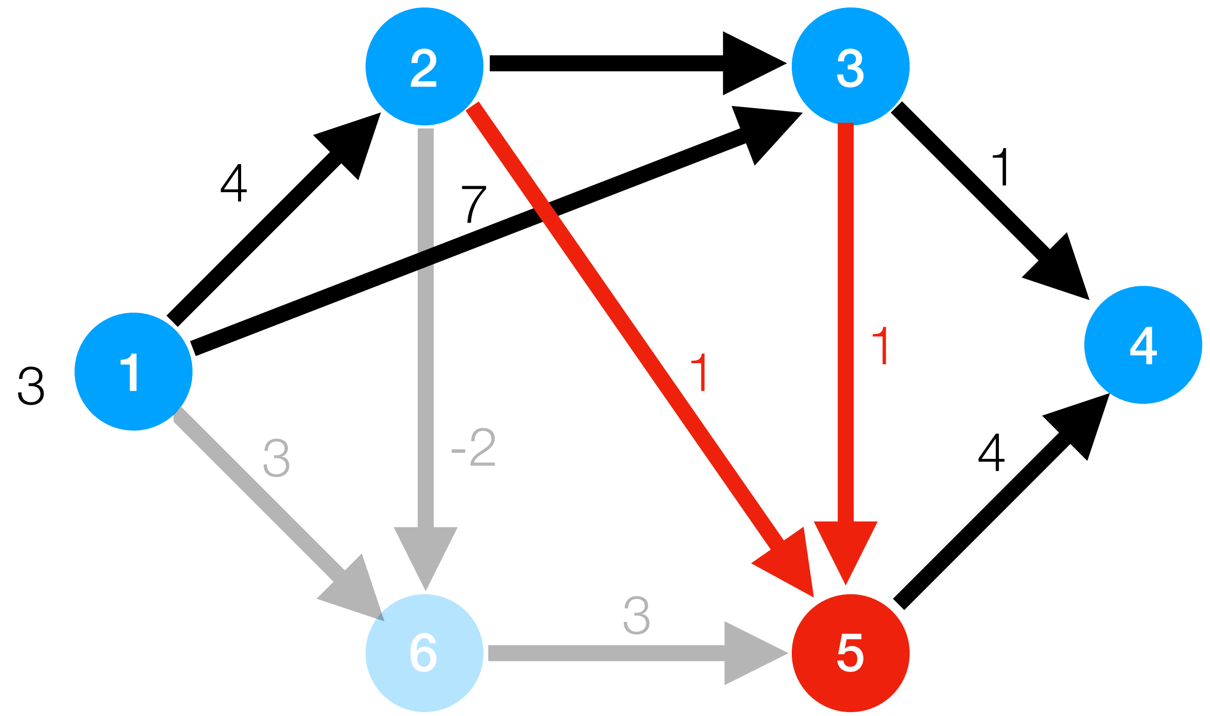
$k=4 \rightarrow k=5$



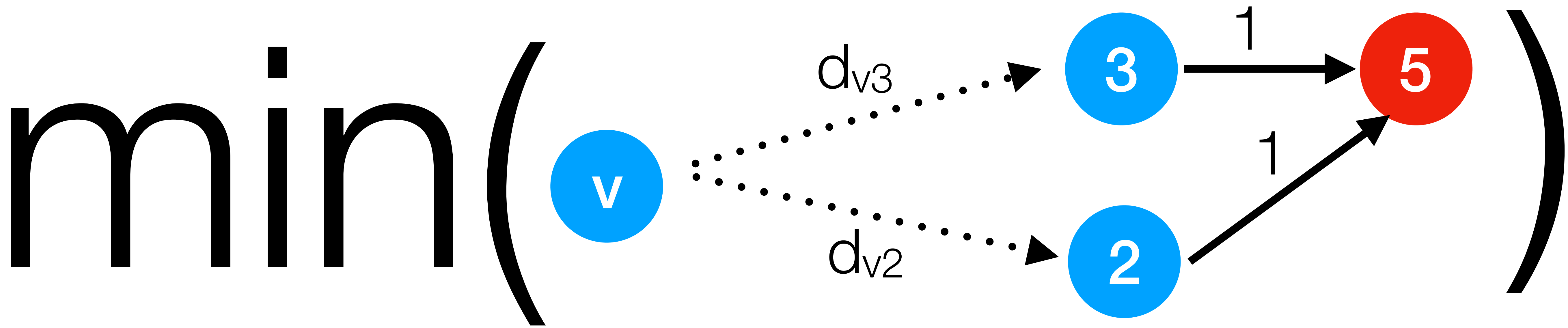
Primer paso



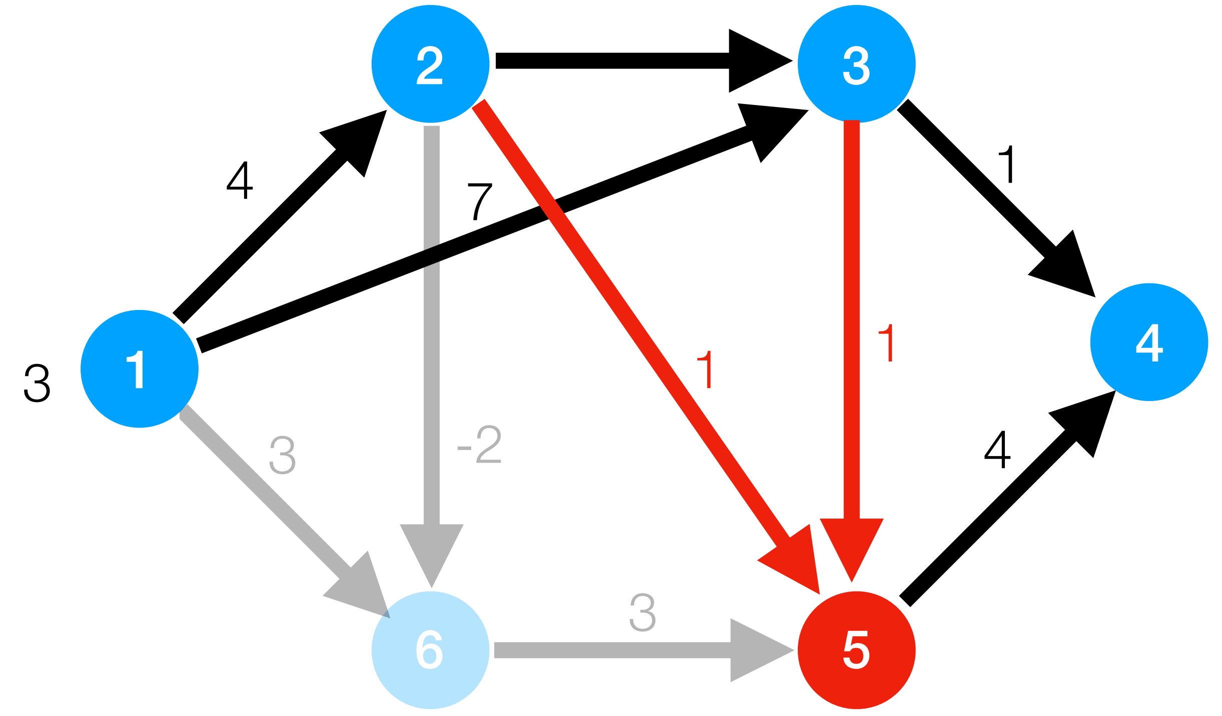
$k=4 \longrightarrow k=5$



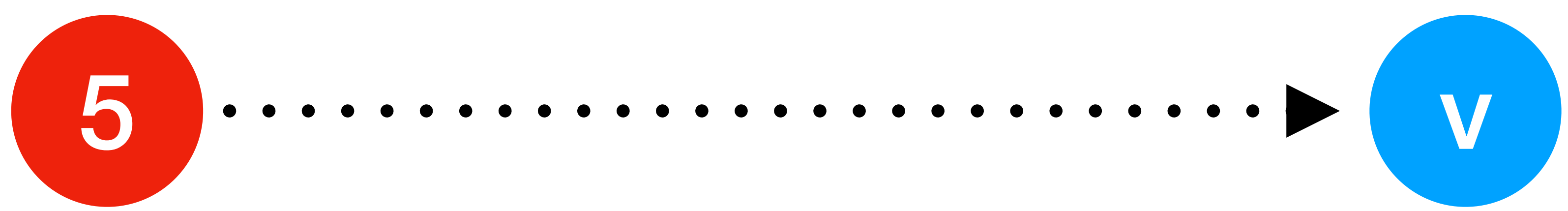
Primer paso



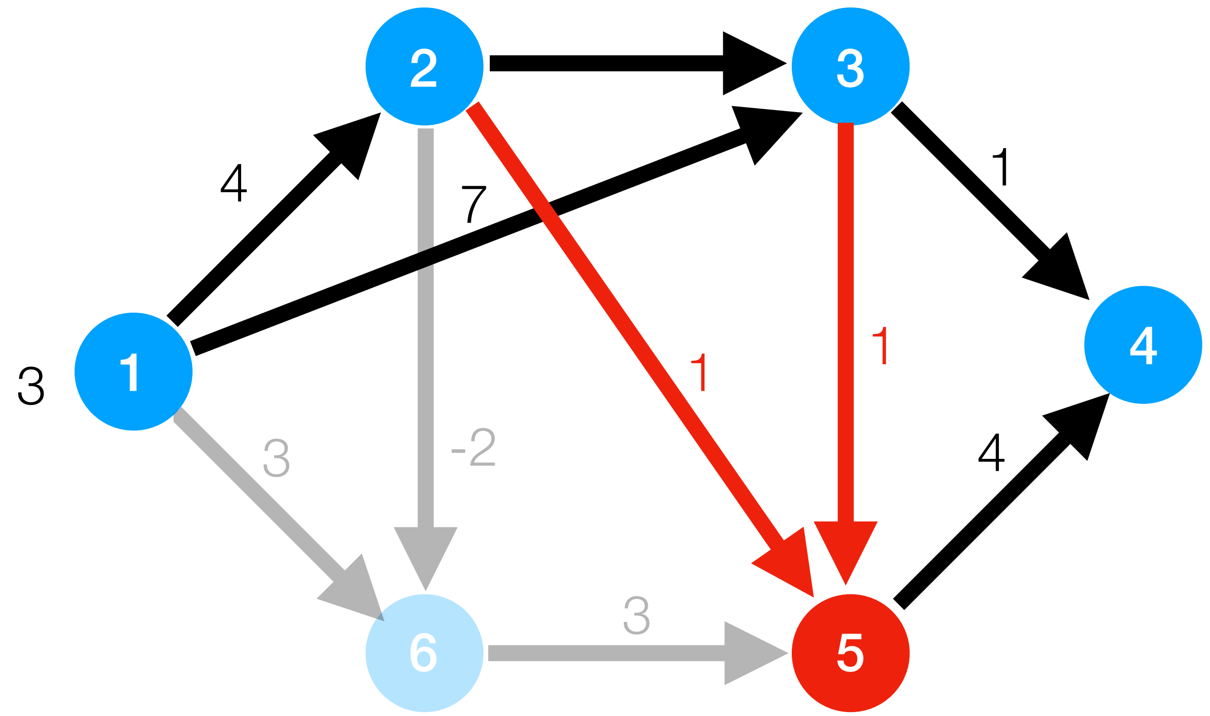
k=4 → k=5



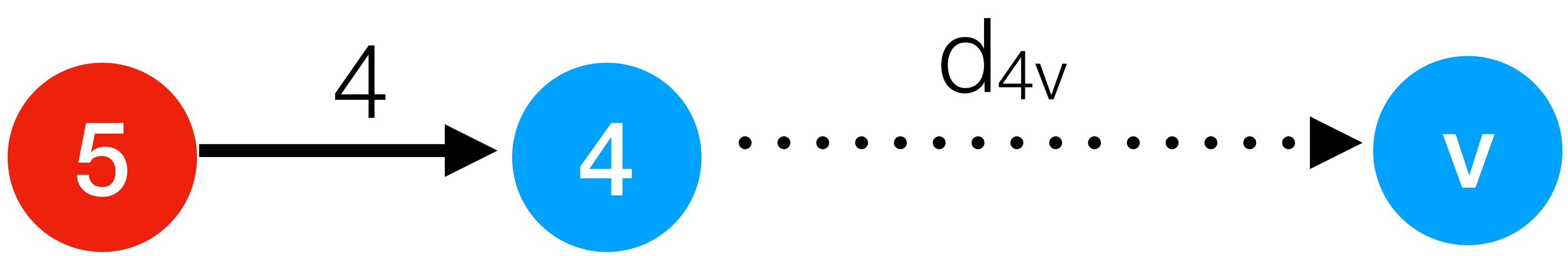
Segundo paso



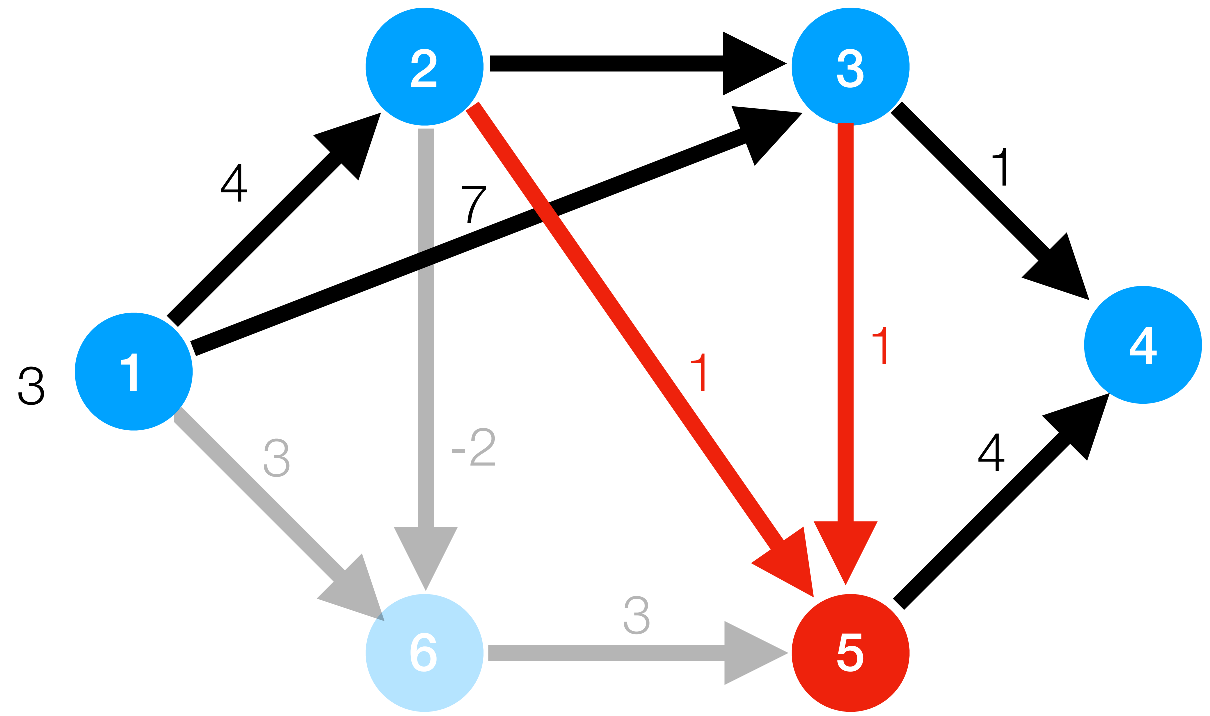
$k=4 \rightarrow k=5$



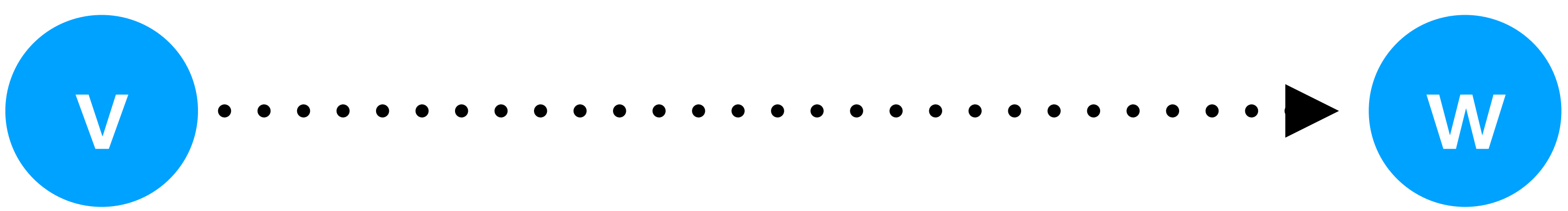
Segundo paso



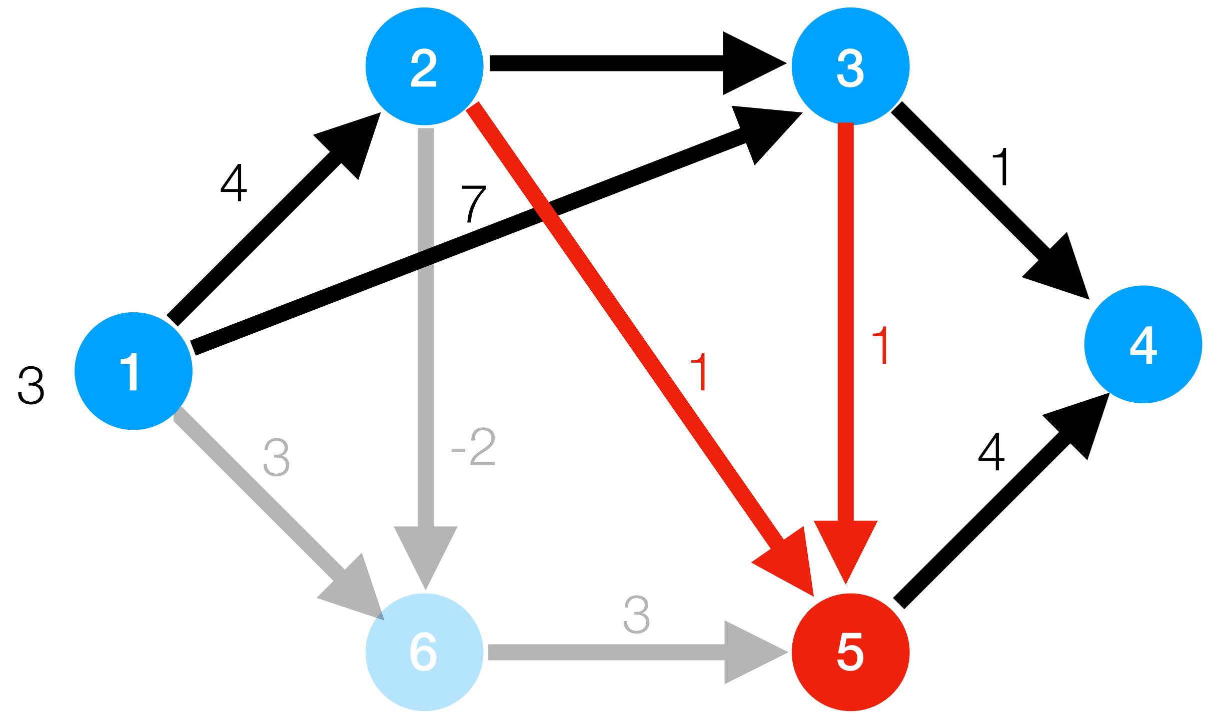
k=4 → k=5



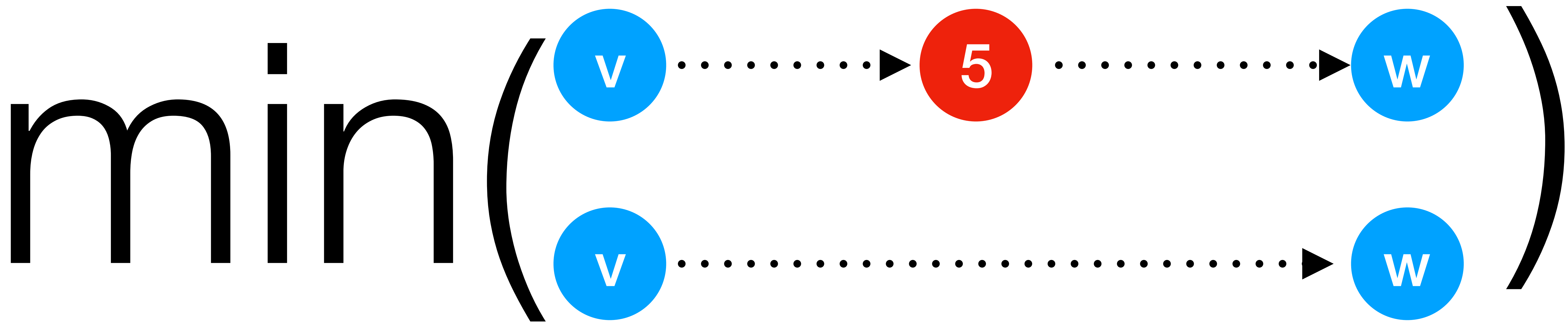
Tercer paso



k=4 → k=5



Tercer paso



Pseudocódigo

function DANTZIG(D, w)

$d_{ij} \leftarrow \infty \ \forall ij,\ d_{ii} \leftarrow 0 \ \forall i$ O(n²)

for $k \in 1 \ldots n$ **do**

for $i \in 1 \ldots k-1$ **do** $d_{ik} \leftarrow min_{1 \leq j \leq k-1} d_{ij} + w_{jk}$ O(n²)

for $i \in 1 \ldots k-1$ **do** $d_{ki} \leftarrow min_{1 \leq j \leq k-1} w_{ki} + d_{ij}$ O(n²)

$d_{kk} \leftarrow min(0, min_{1 \leq i \leq k-1} d_{ki} + d_{ik})$ O(n)

for $i \in 1 \ldots k-1$ **do**

for $j \in 1 \ldots k-1$ **do**

$d_{ij} \leftarrow min(d_{ij}, d_{ik} + d_{kj})$

O(n²)

O(n³)

return d

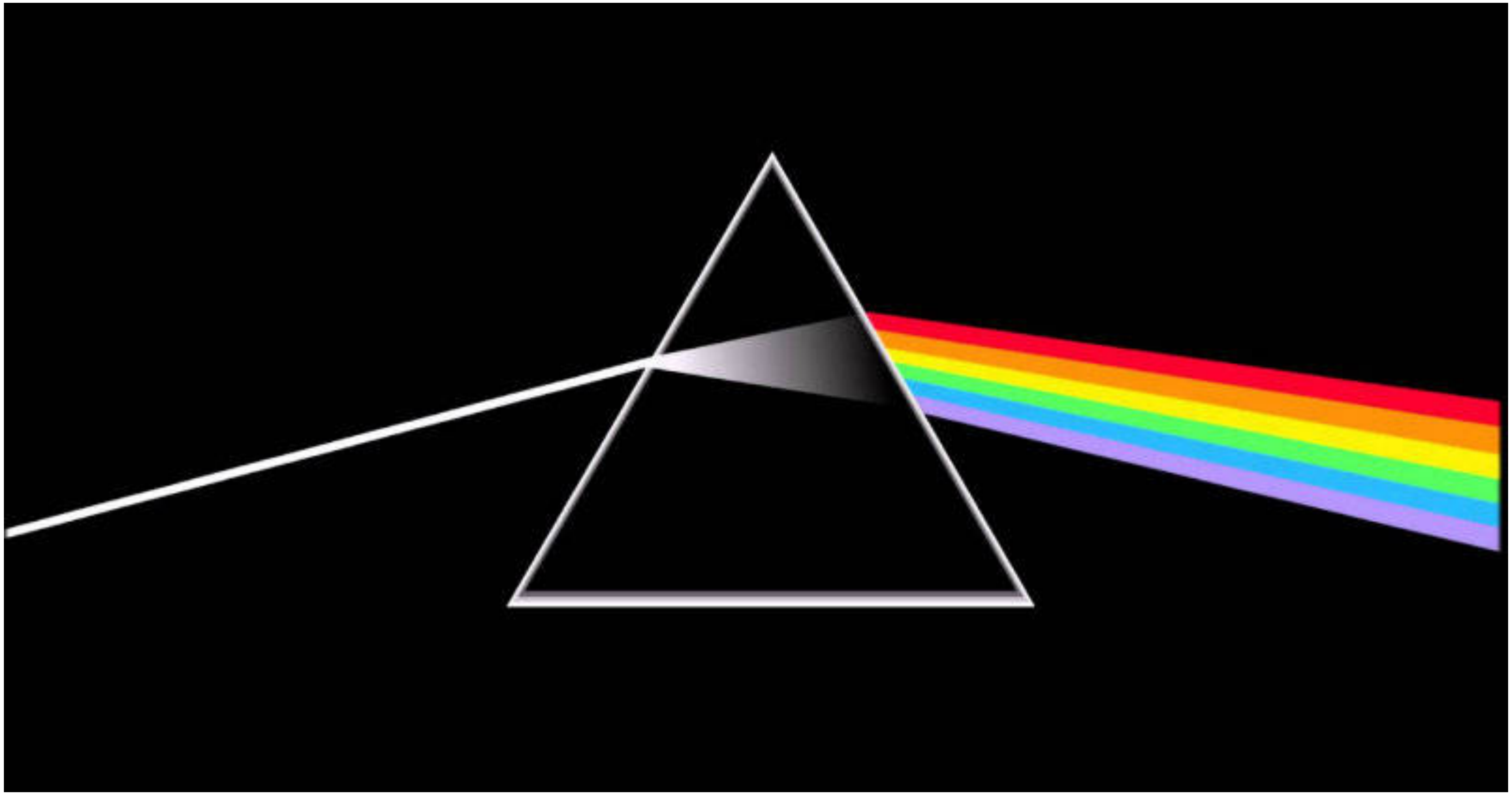
Globalizando

Se tienen n ciudades numeradas de 1 a n en las cuales hay aeropuertos. Hay vuelos desde algunas ciudades a otras con un cierto costo. Cada vuelo se puede representar con su ciudad de salida, de llegada, y su costo (i, j, c).

Monti viaja mucho por todo el mundo y tiene calculada la manera más barata de viajar entre cada par de ciudades (i, j). Sin embargo, un nuevo aeropuerto se ha descubierto en la ciudad de *La Plata* y ahora hay nuevos vuelos desde y hacia ella.

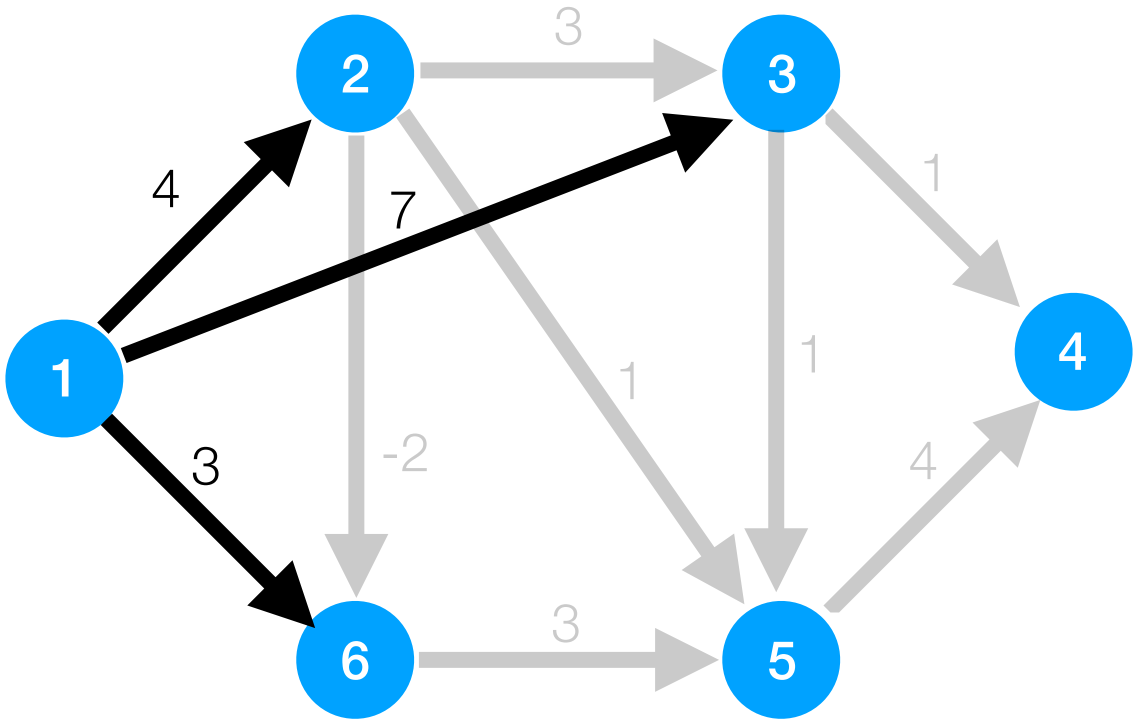
Monti quiere seguir sabiendo la mejor manera de viajar entre cada par de ciudades considerando esta nueva ciudad. ¿Cómo puede hacer para actualizar su información?

Floyd-Washall

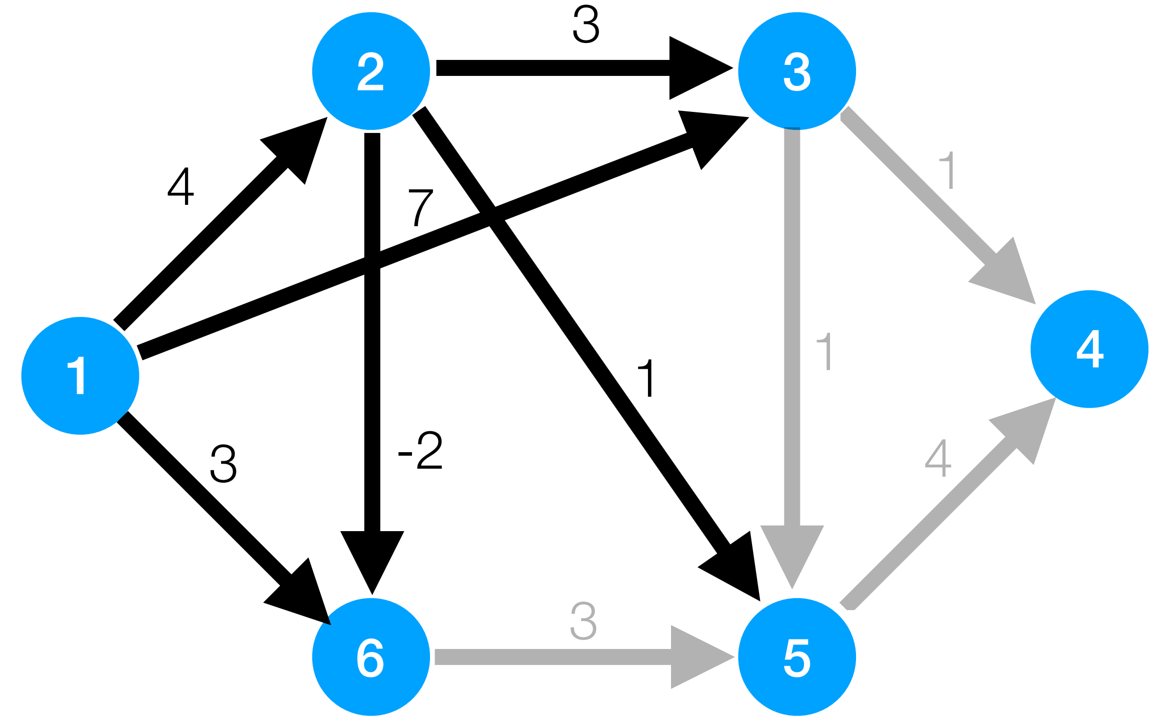


Puedo ir resolviendo el problema considerando caminos que usen cada vez más vértices diferentes

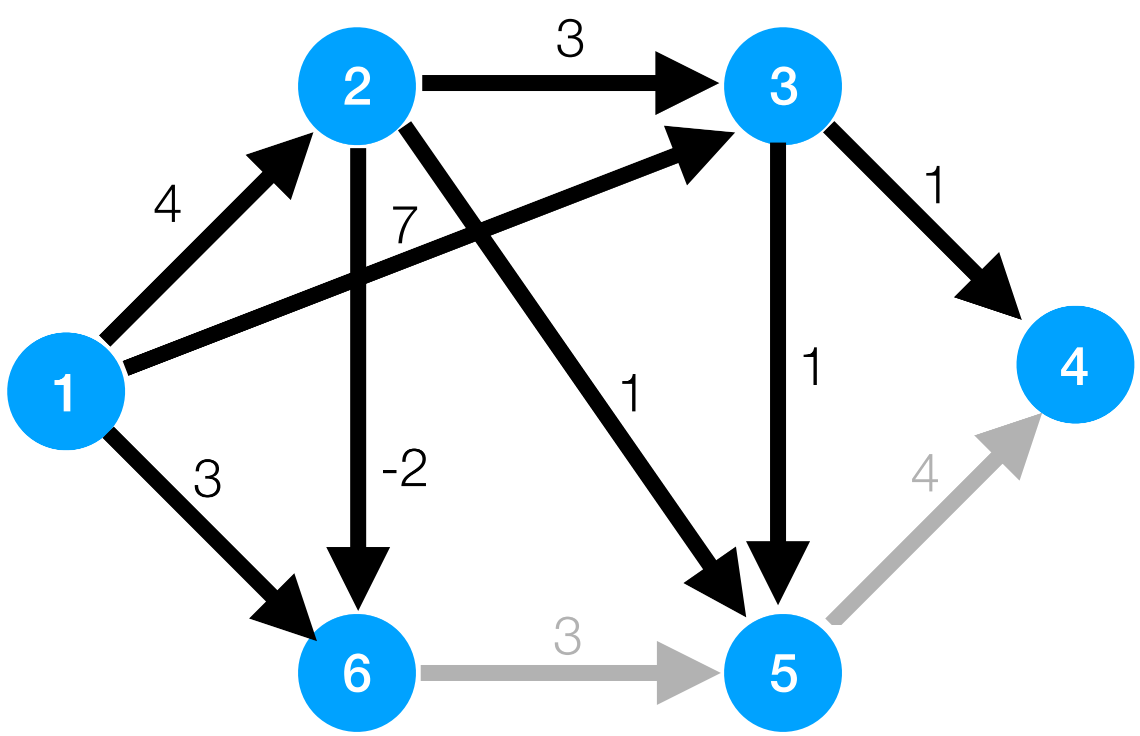
k=1



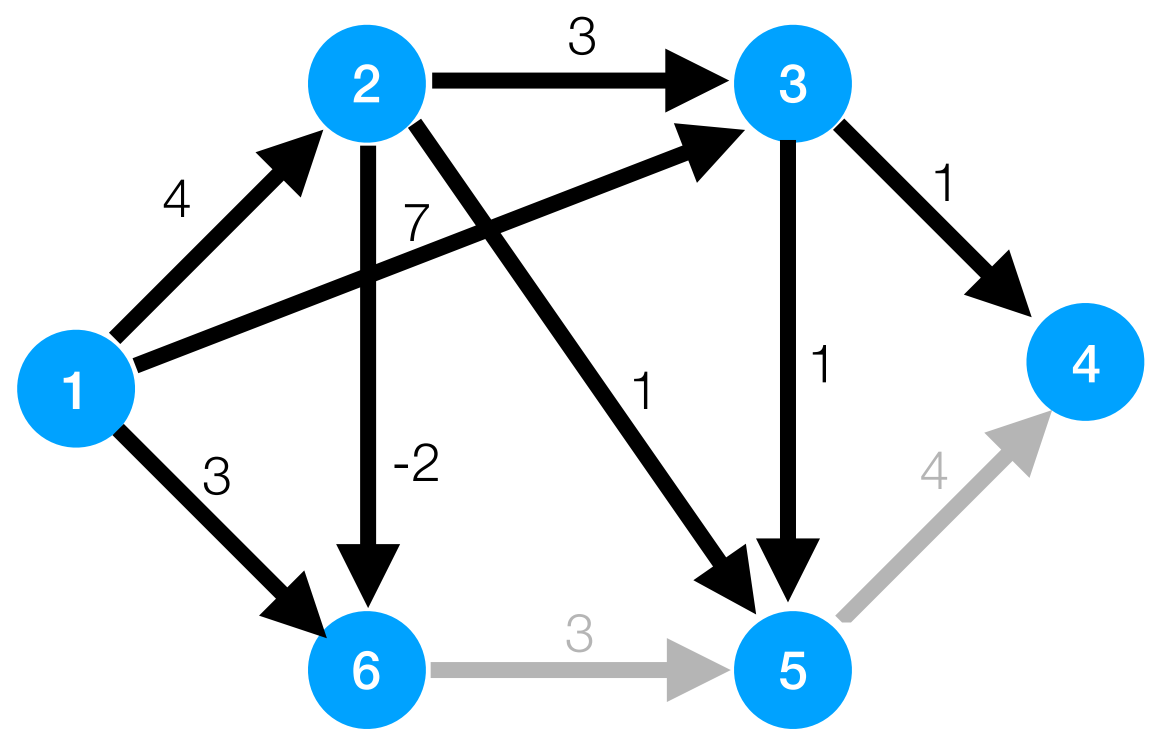
k=2



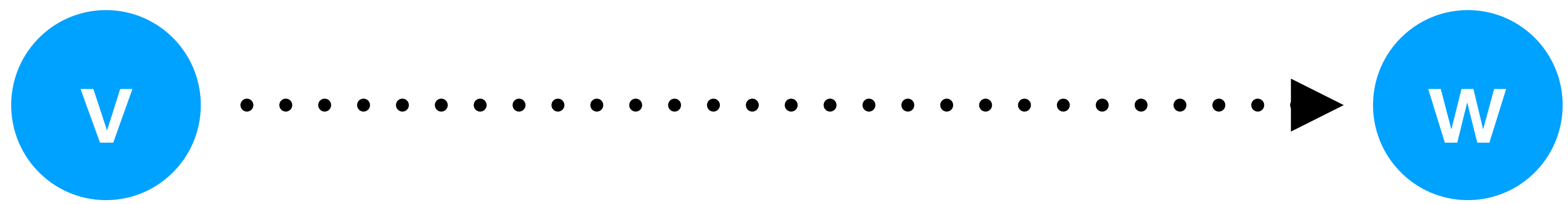
k=3



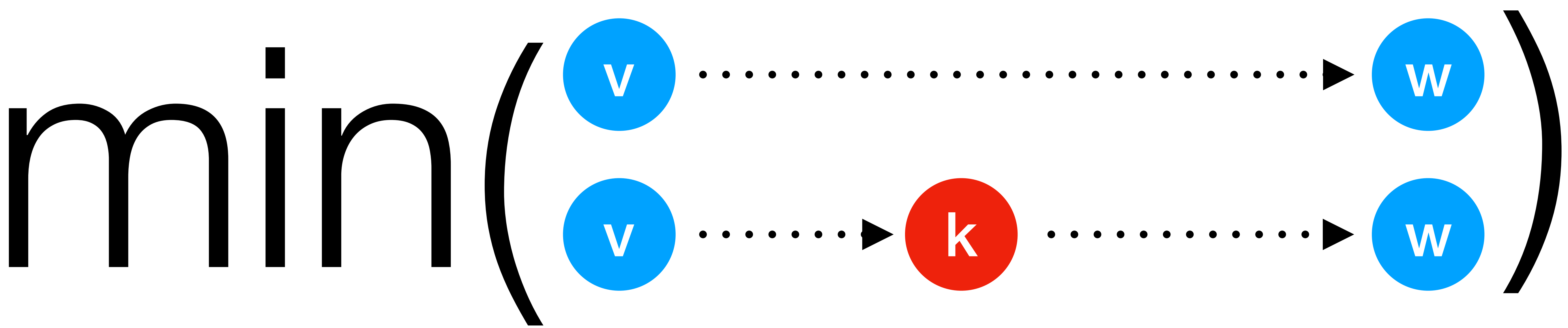
k=4



Camino mínimo considerando los primeros k-1 vértices como intermedios



Camino mínimo considerando los primeros k vértices como intermedios



Pseudocódigo

function FLOYD-WARSHALL(D, w)

$d_{ii} \leftarrow 0 \; \forall i \in V(D), \;\; d_{ij} \leftarrow \infty \; \forall ij \notin E(D), d_{ij} \leftarrow w_{ij} \; \forall ij \in E(D)$ O(n^2)

$$\left.\begin{array}{l}
\textbf{for } k \in 1 \ldots n \textbf{ do} \\
\quad \textbf{for } i \in 1 \ldots n \textbf{ do} \\
\quad\quad \textbf{for } j \in 1 \ldots n \textbf{ do} \\
\quad\quad\quad d_{ij} \leftarrow min(d_{ij}, d_{ik} + d_{kj})
\end{array}\right\} O(n^3)$$

return d

¿Y si hay ciclos negativos?

function FLOYD-WARSHALL(D, w)

$d_{ii} \leftarrow 0 \; \forall i \in V(D),\;\; d_{ij} \leftarrow \infty \; \forall ij \notin E(D), d_{ij} \leftarrow w_{ij} \; \forall ij \in E(D)$

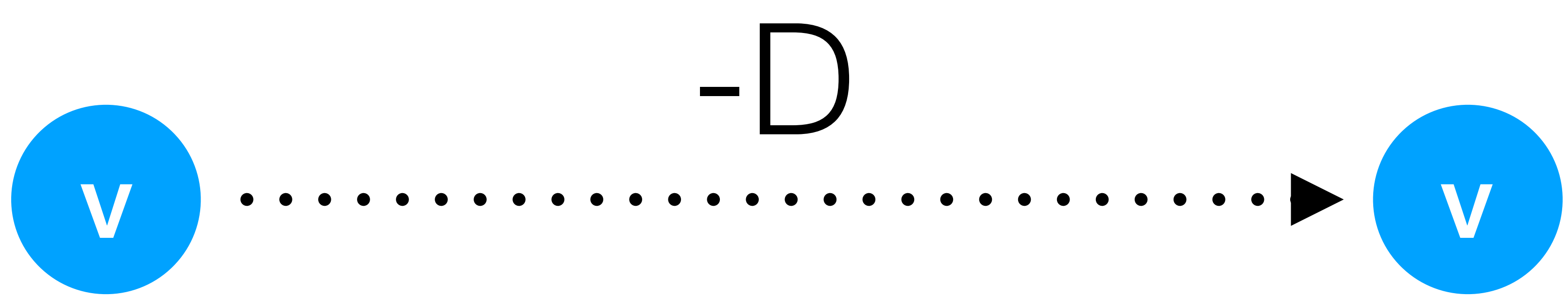
for $k \in 1 \ldots n$ **do**

for $i \in 1 \ldots n$ **do**

for $j \in 1 \ldots n$ **do**

$d_{ij} \leftarrow min(d_{ij}, d_{ik} + d_{kj})$

return d

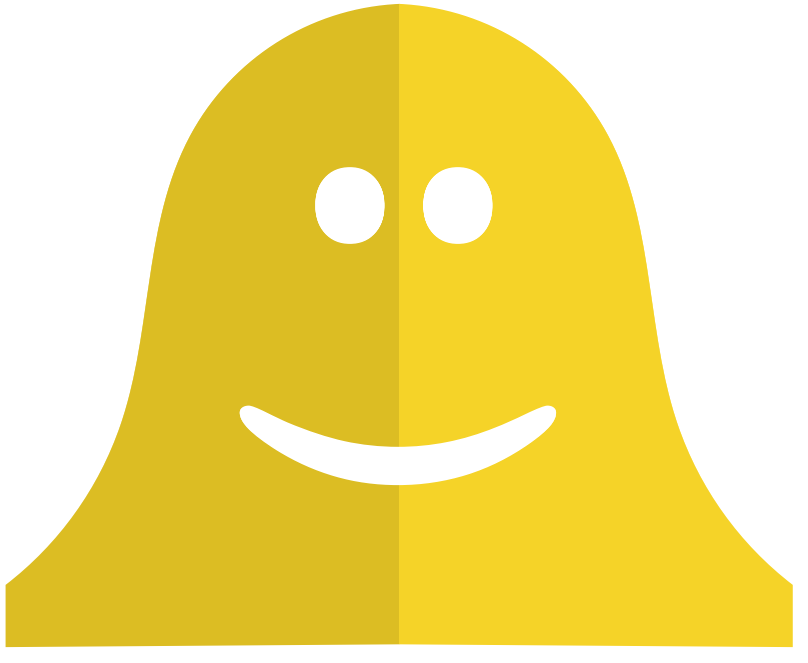
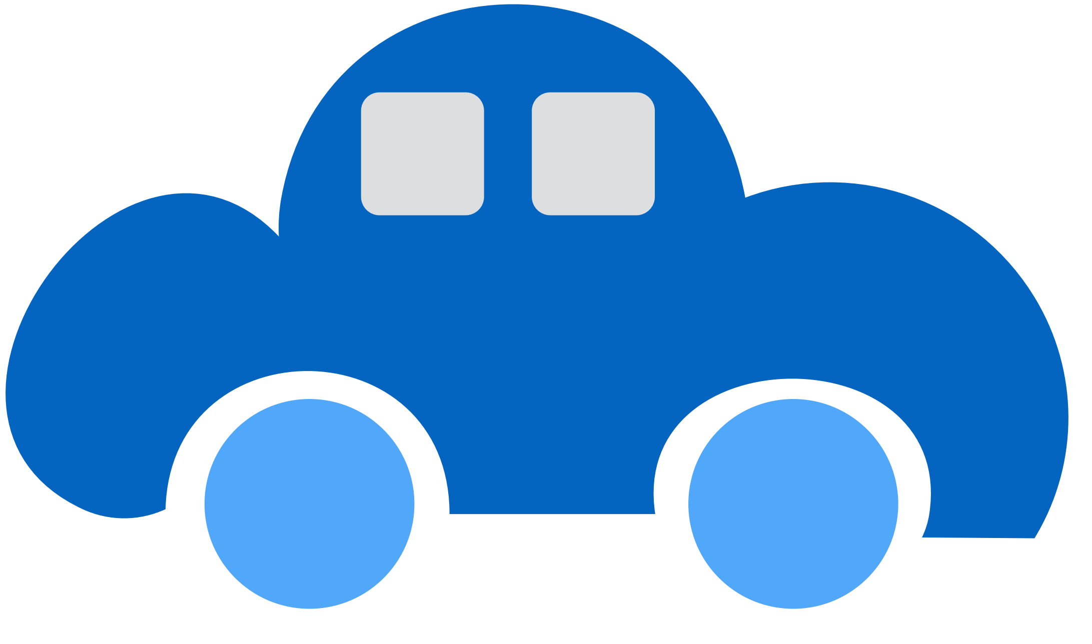


Clausura geodésica

Sea $G = \langle V, E \rangle$ un grafo y $v, w \in V$ dos vértices cualquiera. Definimos la clausura geodésica de G con respecto a v y w como

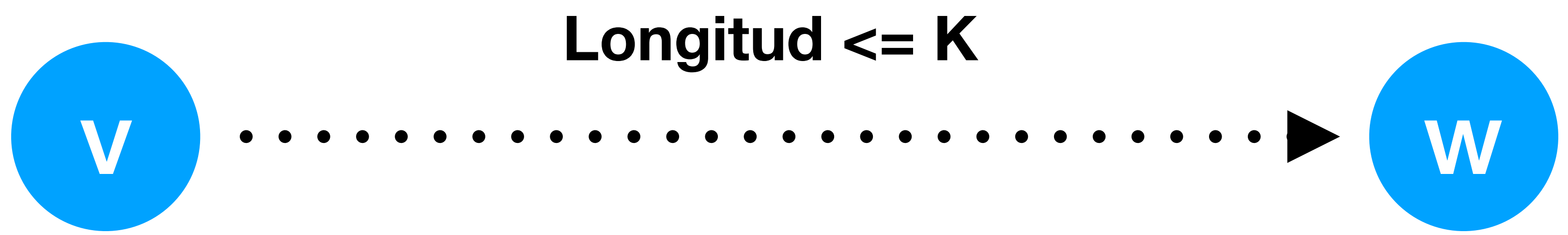
$$G[v, w] = \{u \in V \mid u \text{ pertenece a algún camino mínimo de } v \text{ a } w\}$$

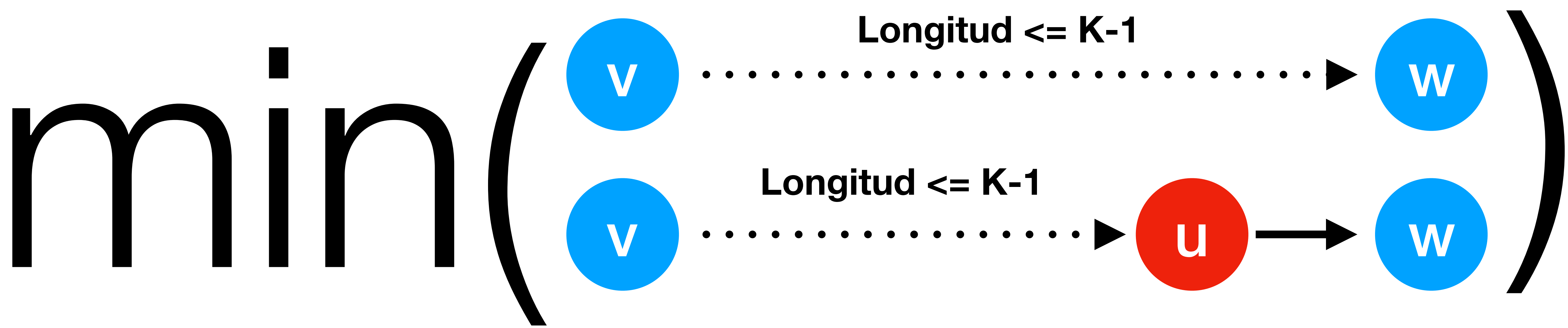
Se desea determinar para qué par de vértices v, w vale que $G[v, w] = G$.

Puedo ir resolviendo el problema considerando caminos cada vez más largos

Camino mínimo de longitud a lo sumo k de v—>w





Pseudocódigo

Listas de adyancencia

function BELLMAN-FORD(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D), \ d_s \leftarrow 0$ O(n)

for $k \in 1 \ldots n-1$ **do**

for $i \in 1 \ldots n$ **do**

for $j \in N(i)$ **do**

$d_j \leftarrow min(d_j, d_i + w_{ij})$

O(n+m)

O(n(n+m))

return d

Pseudocódigo
Matriz de adyancencia

function BELLMAN-FORD(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D), \ d_s \leftarrow 0$ O(n)

for $k \in 1 \ldots n-1$ **do**

for $i \in 1 \ldots n$ **do** O(n³)

for $j \in N(i)$ **do** O(n²)

$d_j \leftarrow min(d_j, d_i + w_{ij})$

return d

function BELLMAN-FORD(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D), \ d_s \leftarrow 0$

for $k \in 1 \ldots n-1$ **do**

for $i \in 1 \ldots n$ **do**

for $j \in N(i)$ **do**

$d_j \leftarrow min(d_j, d_i + w_{ij})$

return d

¡Si en la iteración k no hubo cambios, en la k+1 tampoco!

function BELLMAN-FORD-OPTIMIZADO(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D),\ d_s \leftarrow 0$

for $k \in 1 \ldots n-1$ **do**

for $i \in 1 \ldots n$ **do**

for $j \in N(i)$ **do**

$d_j \leftarrow min(d_j, d_i + w_{ij})$

if no cambió d_i para ningún i **then parar**

return d

¿Cómo detectamos ciclos negativos?

function BELLMAN-FORD-OPTIMIZADO(D, w, s)

$d_i \leftarrow \infty \; \forall i \in V(D), \; d_s \leftarrow 0$

for $k \in 1 \dots n-1$ **do**

  for $i \in 1 \dots n$ **do**

    for $j \in N(i)$ **do**

      $d_j \leftarrow min(d_j, d_i + w_{ij})$

  if no cambió d_i para ningún i **then parar**

return d

¿Cómo detectamos ciclos negativos?

function BELLMAN-FORD-OPTIMIZADO(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D), \ d_s \leftarrow 0$

for $k \in 1 \dots n$ **do**

for $i \in 1 \dots n$ **do**

for $j \in N(i)$ **do**

$d_j \leftarrow min(d_j, d_i + w_{ij})$

if no cambió d_i para ningún i **then** $k \leftarrow k - 1$, **parar**

$\boxed{\textbf{if } k = n \textbf{ then } \text{hay ciclos negativos}}$

return d

¡No puede haber un camino simple de longitud mayor a n-1!

function BELLMAN-FORD-OPTIMIZADO(D, w, s)

$d_i \leftarrow \infty \; \forall i \in V(D), \; d_s \leftarrow 0$

for $k \in 1 \ldots n$ **do**

for $i \in 1 \ldots n$ **do**

for $j \in N(i)$ **do**

$d_j \leftarrow min(d_j, d_i + w_{ij})$

si d_i no cambió en la iteración anterior entonces no va a actualizar nada

if no cambió d_i para ningún i **then** $k \leftarrow k - 1$, **parar**

if $k = n$ **then** hay ciclos negativos

return d

function BELLMAN-FORD-MASOPTIMIZADO(D, w, s)

$d_i \leftarrow \infty \; \forall i \in V(D), \;\; d_s \leftarrow 0$

$Q_0 \leftarrow \{s\}$

for $k \in 1 \ldots n$ **do**

for $i \in 1 \ldots Q_{k-1}$ **do**

Solo reviso los que cambiaron en la iteración anterior

for $j \in N(i)$ **do**

if $d_j > d_i + w_{ij}$ **then**

$d_j \leftarrow d_i + w_{ij}$

$push(Q_k, (d_j, j))$

if no cambió d_i para ningún i **then** $k \leftarrow k - 1$, **parar**

return d

function BELLMAN-FORD-LABELING(D, w, s)

$d_i \leftarrow \infty \; \forall i \in V(D), \;\; d_s \leftarrow 0$

$Q \leftarrow \{(0, s)\}$

No hago un Q por cada k, reuso el mismo e ignoro los caminos que ya están obsoletos

while $Q \neq \emptyset$ **do**

$(c, i) \leftarrow pop(Q)$

if $c > d_i$ **then continue**

for $j \in N(i)$ **do**

if $d_j > d_i + w_{ij}$ **then**

$d_j \leftarrow d_i + w_{ij}$

$push(Q, (d_j, j))$

return d

Conexiones

Se tienen n ciudades numeradas de 1 a n en las cuales hay aeropuertos. Además se sabe que hay vuelos desde algunas ciudades a otras con un cierto costo. Cada vuelo se puede representar con su ciudad de salida, de llegada, y su costo (i, j, c).

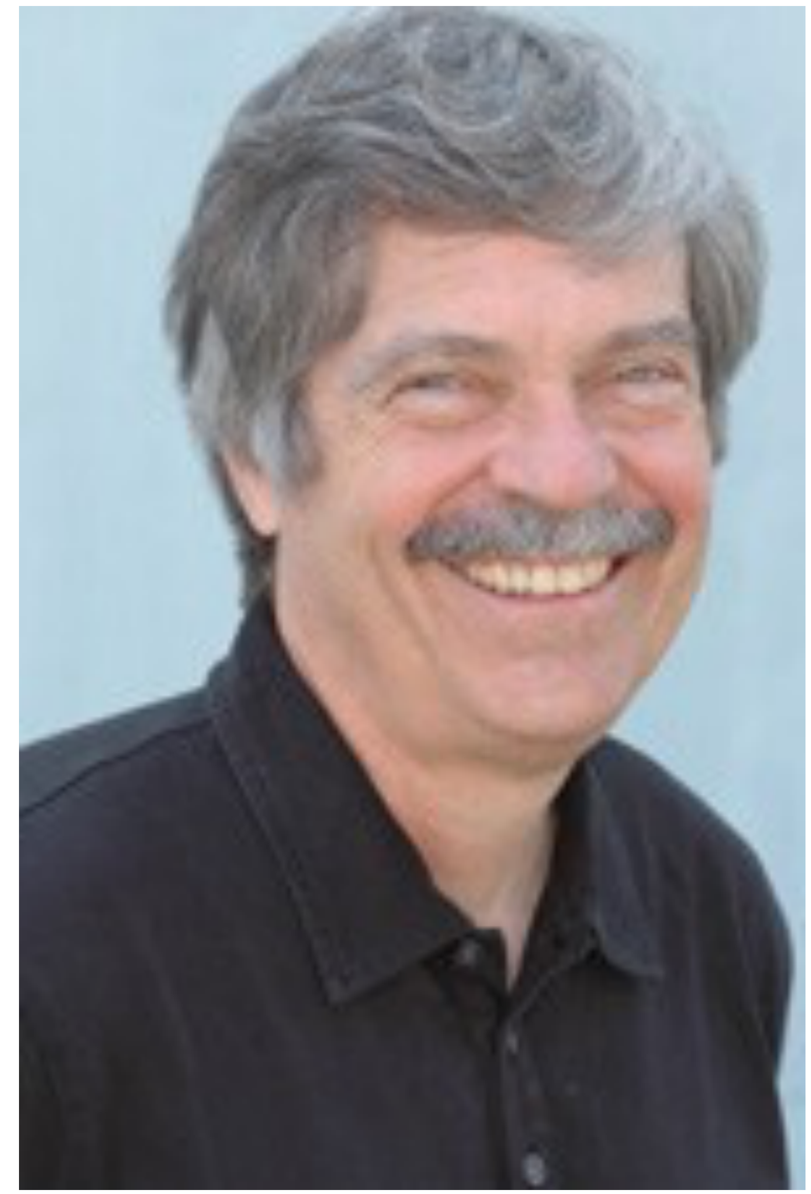
Monti quiere viajar desde la ciudad 1 a la n haciendo a lo sumo k conexiones de vuelos y gastar la menor cantidad de dinero posible. ¿Es posible lograrlo? ¿Cuál es la forma más barata?

Edsger Dijkstra

- Persona importante en la historia de la computación.
- Creador de varios algoritmos, entre ellos el de caminos mínimos.
- Muy riguroso y fomentador de la verificación formal.

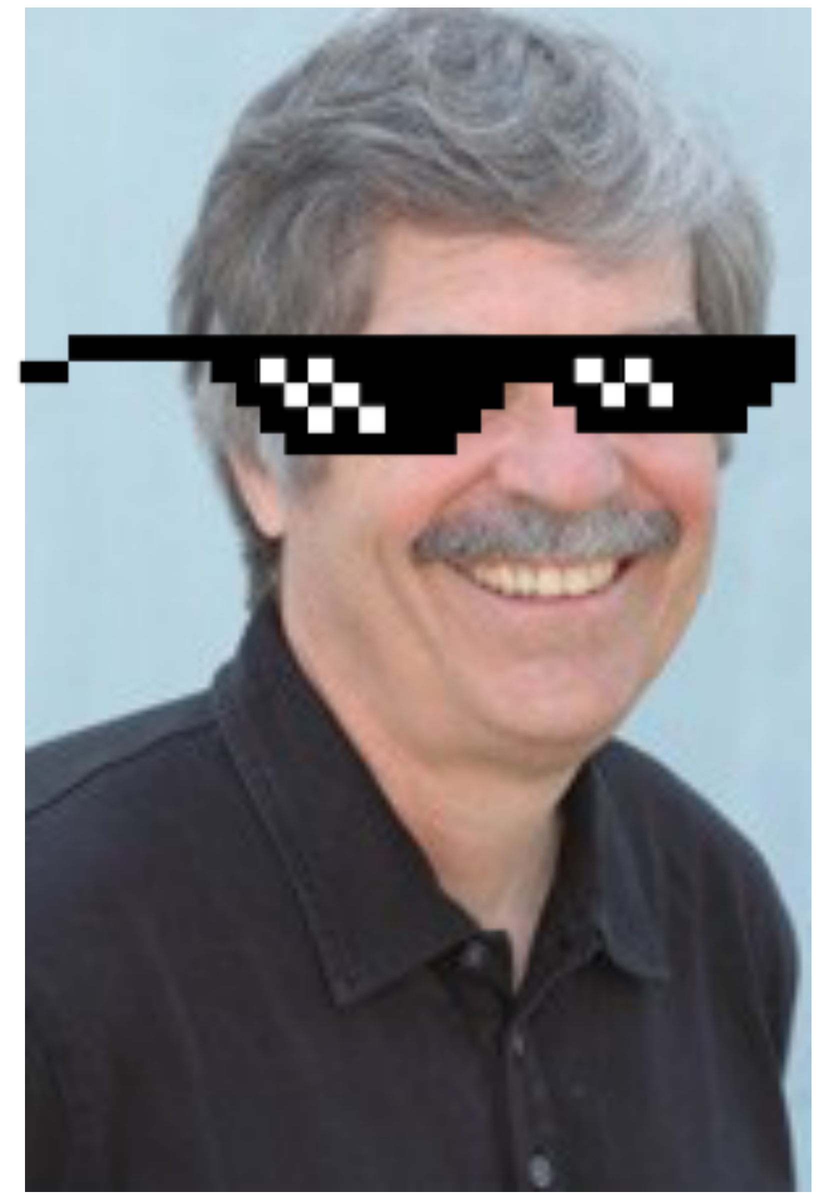
Alan Kay

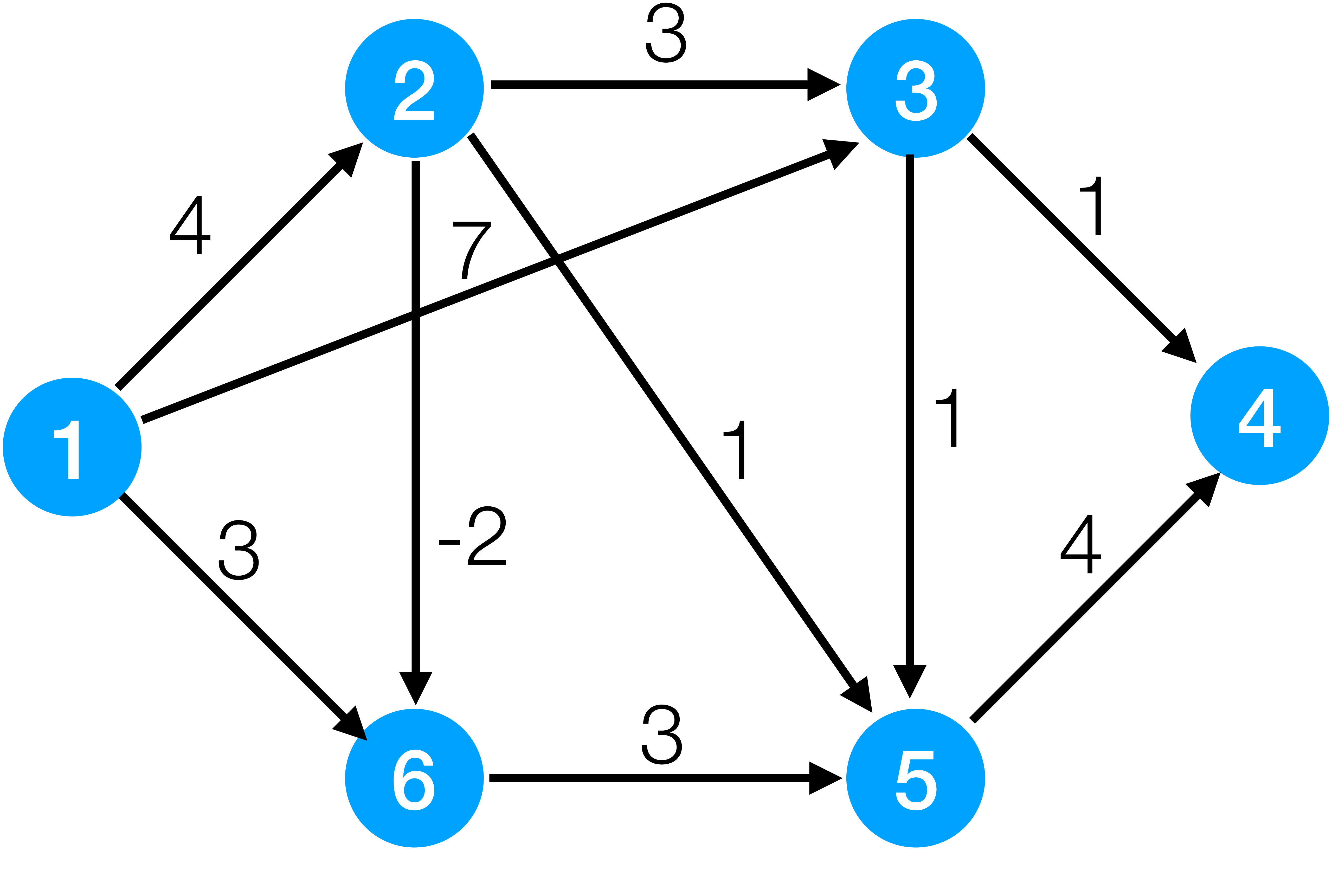
"I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras"

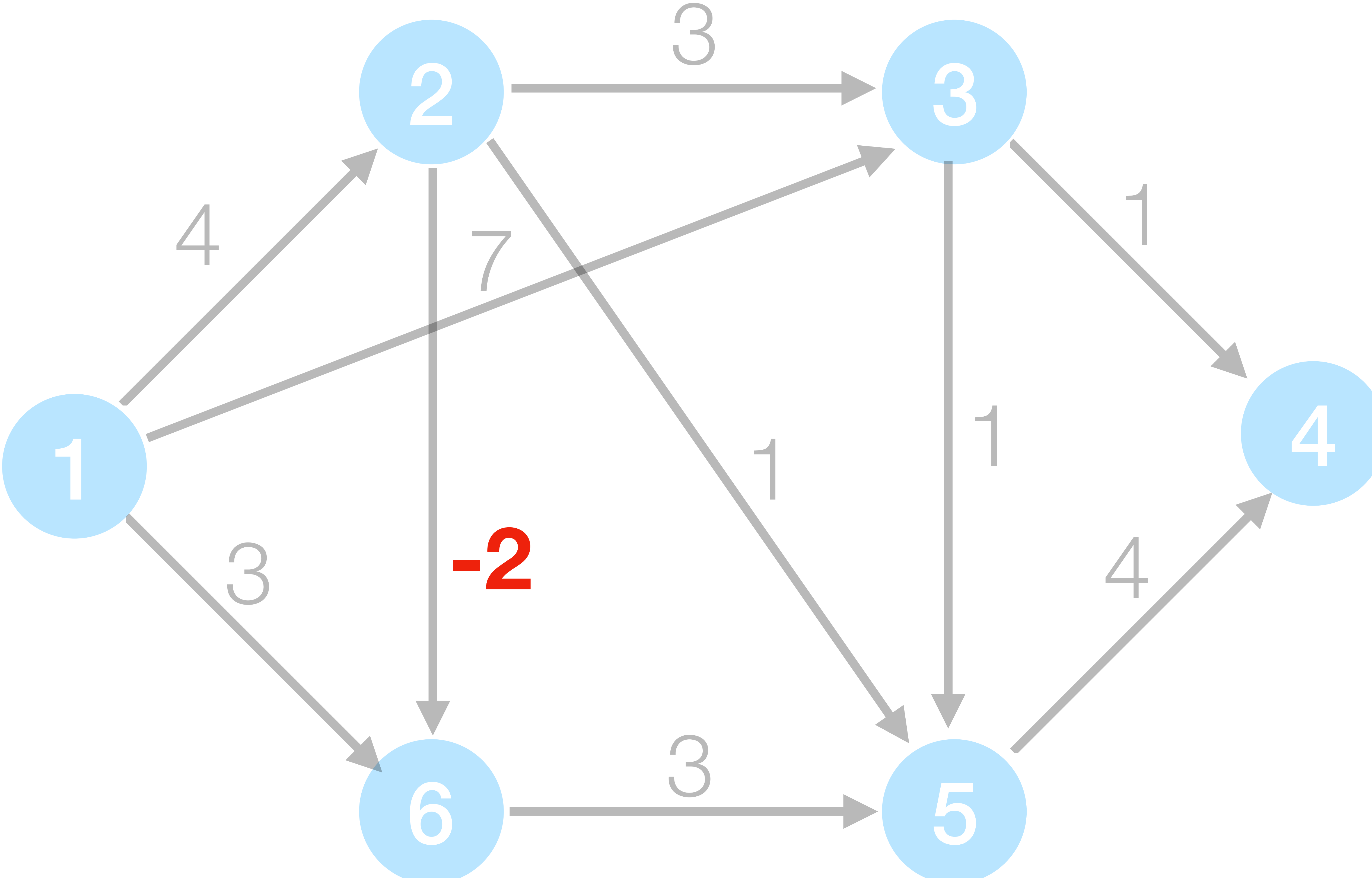


Alan Kay

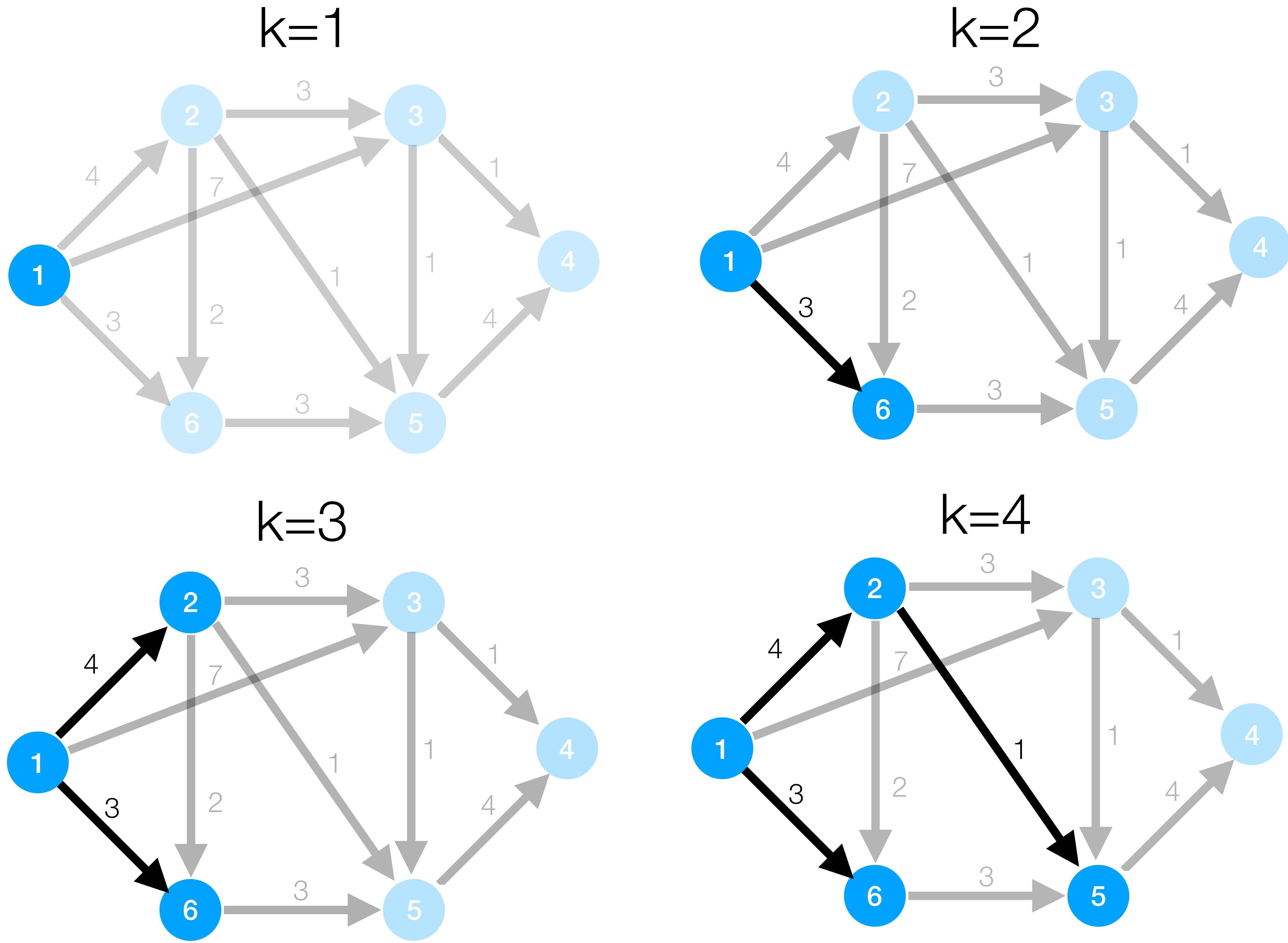
“I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras”

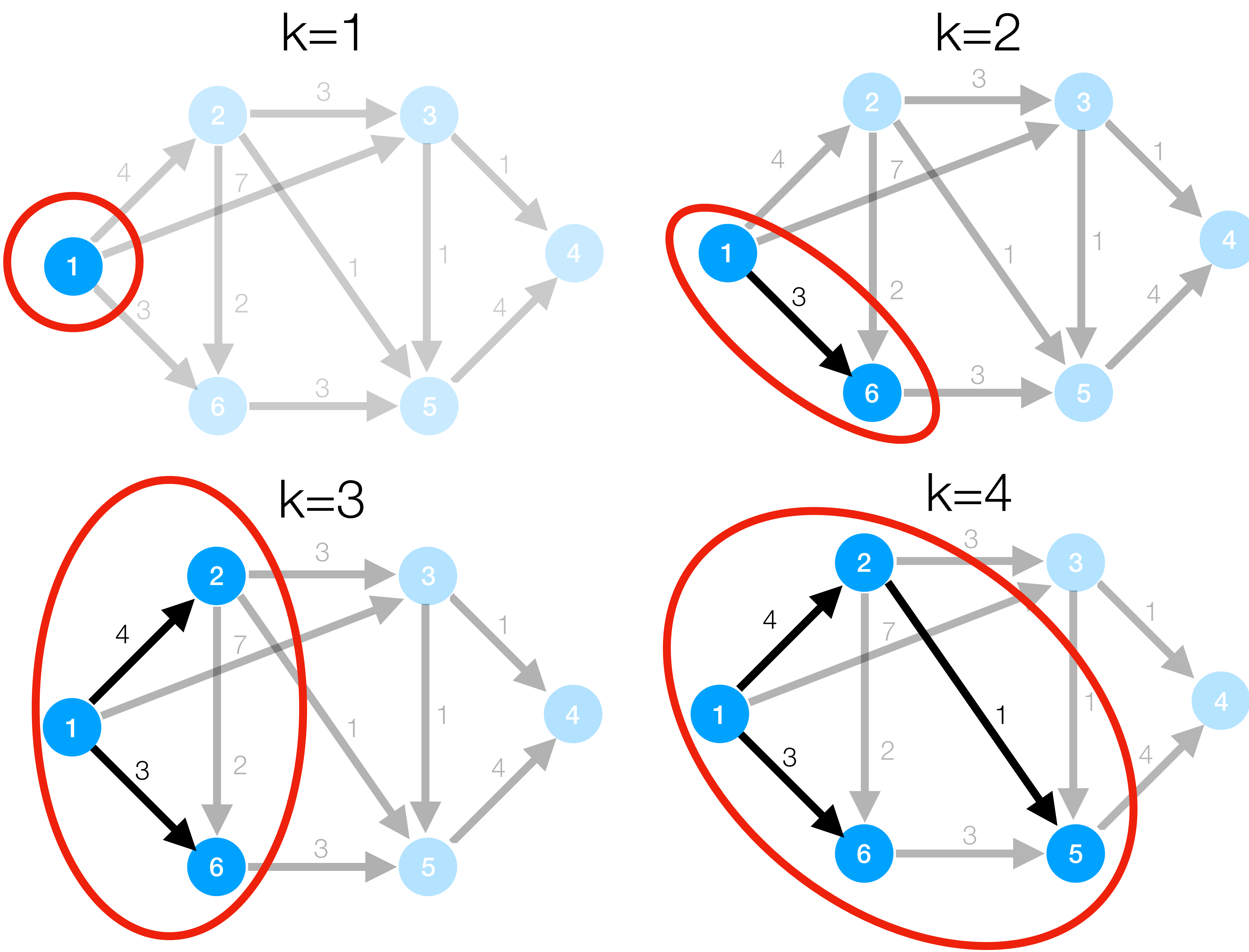



1
2
3
4
5
6
4
3
3
7
-2
1
1
1
3
4



1
2
3
4
5
6
4
3
3
7
1
1
1
-2
4
3


k=1
k=2
k=3
k=4


k=1
k=2
k=3
k=4

Pseudocódigo

Listas de adyancencia

function DIJKSTRA(D, w, s)

$d_i \leftarrow \infty \; \forall i \in V(D),\; d_s \leftarrow 0$ O(n)

$Q \leftarrow \{(d_s, s)\}$ Cola de prioridad <

while $Q \neq \emptyset$ **do**

$(c, i) \leftarrow pop(Q)$ O(pop)

if $c > d_i$ **then continue**

for $j \in N(i)$ **do**

if $d_j > d_i + w_{ij}$ **then**

$d_j \leftarrow d_i + w_{ij}$

$push(Q, (d_j, j))$ O(push)

return d

O(m+n(pop+push))

	Heap	Secuencia
pop	O(log(n))	O(n)
push	O(log(n))	O(1)

Pseudocódigo

Matriz de adyancencia

function DIJKSTRA(D, w, s)

$d_i \leftarrow \infty \; \forall i \in V(D), \; d_s \leftarrow 0$ O(n)

$Q \leftarrow \{(d_s, s)\}$ Cola de prioridad <

while $Q \neq \emptyset$ **do**

$(c, i) \leftarrow pop(Q)$ O(pop)

if $c > d_i$ **then continue**

for $j \in N(i)$ **do**

if $d_j > d_i + w_{ij}$ **then**

$d_j \leftarrow d_i + w_{ij}$

$push(Q, (d_j, j))$ O(push)

return d

$O(n^2+n(pop+push))$

	Heap	Secuencia
pop	O(log(n))	O(n)
push	O(log(n))	O(1)

Mucho tráfico

Se tienen n esquinas en una ciudad $(1, \ldots, n)$ y calles que conectan algunas esquinas (i, j) con algun tiempo de viaje t_{ij}. Juan quiere llegar desde la esquina v a la w. Como mucha gente toma el camino mínimo entre esas esquinas entonces Juan quiere viajar únicamente por calles que no estén en ningún camino mínimo.

¿Podés determinar qué calles están en algún camino mínimo entre v y w para que Juan no las use?

Breadth First Search

Pseudocódigo

Listas de adyancencia

function BFS(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D), \ d_s \leftarrow 0$ O(n)

$Q \leftarrow \{(d_s, s)\}$ Cola FIFO

while $Q \neq \emptyset$ **do**

$(c, i) \leftarrow pop(Q)$ O(pop)

if $c > d_i$ **then continue**

for $j \in N(i)$ **do**

if $d_j > d_i + w_{ij}$ **then**

$d_j \leftarrow d_i + w_{ij}$

$push(Q, (d_j, j))$ O(push)

O(m+n)

return d

Pseudocódigo

Matriz de adyancencia

function BFS(D, w, s)

$d_i \leftarrow \infty \ \forall i \in V(D), \ d_s \leftarrow 0$ O(n)

$Q \leftarrow \{(d_s, s)\}$ Cola FIFO

while $Q \neq \emptyset$ **do**

$(c, i) \leftarrow pop(Q)$ O(pop)

if $c > d_i$ **then continue**

for $j \in N(i)$ **do**

if $d_j > d_i + w_{ij}$ **then**

$d_j \leftarrow d_i + w_{ij}$

$push(Q, (d_j, j))$ O(push)

O(n²)

return d

Fin