Algorithmique des graphes

David Pichardie

18 Avril 2018

Bilan du CM7

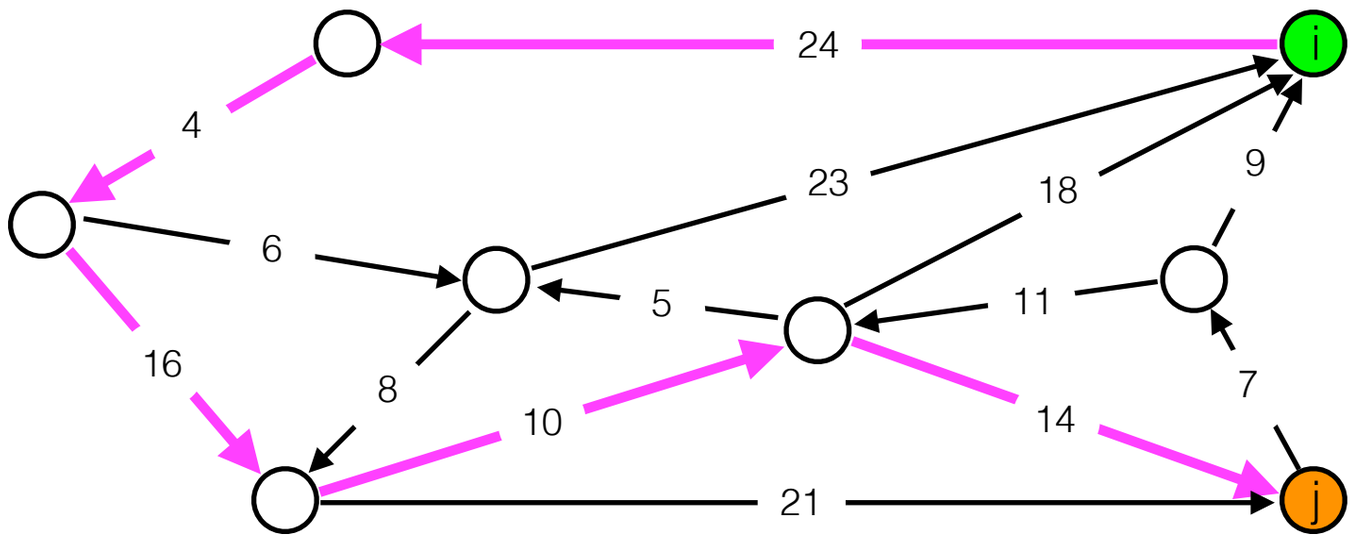
- Arbres couvrants minimaux (ACM)
  - Propriété de la coupure
  - Algorithme glouton abstrait
  - Algorithme de Prim
  - Algorithme de Kruskal

Longueur d'un chemin

Définition

Dans un graphe G orienté pondéré, la *longueur* d'un chemin est la somme des poids des arcs qui composent ce chemin.

Exemple



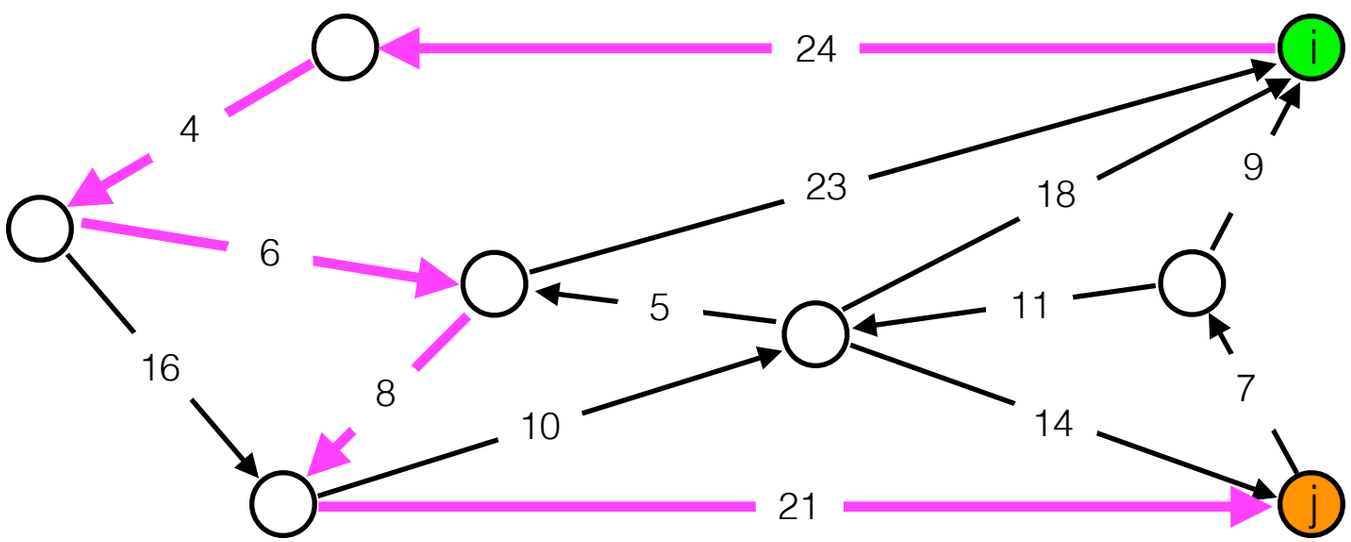
chemin de i à j de longueur 68=24+4+16+10+14

Distance minimum

Définition

Dans un graphe G orienté pondéré, la *distance minimum δ(i,j)* entre deux sommets i et j est le minimum ($\in \mathbb{Z} + \{-\infty, +\infty\}$) des longueurs des chemins entre i et j.

Exemple



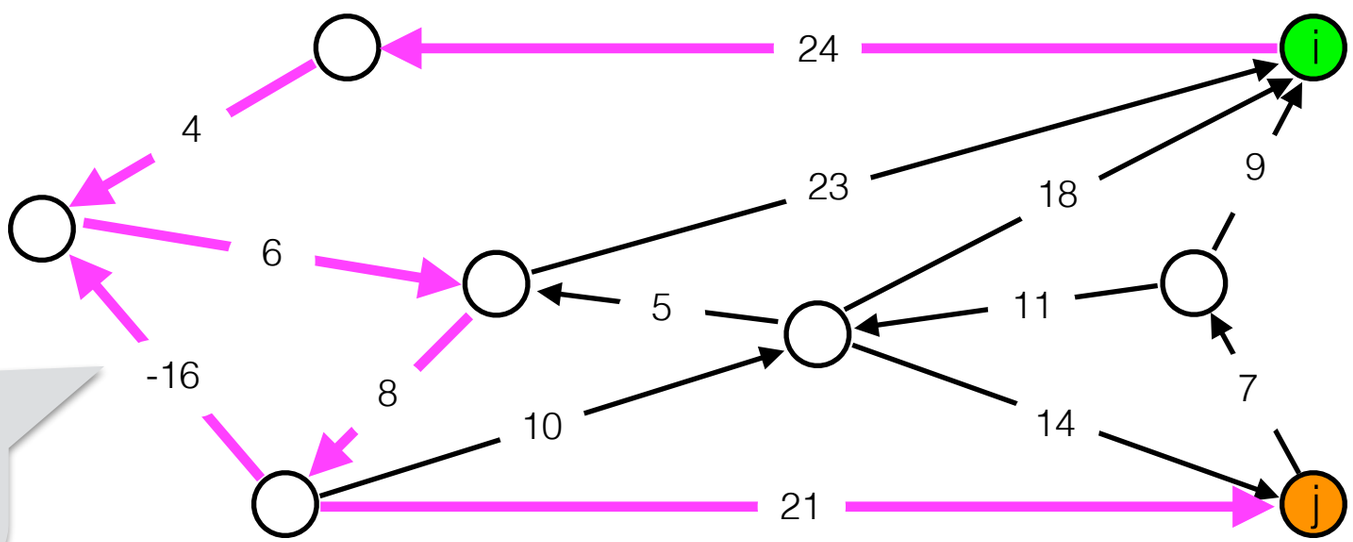
δ(i,j) = 63

Distance minimum

Définition

Dans un graphe G orienté pondéré, la *distance minimum δ(i,j)* entre deux sommets i et j est le minimum ($\in \mathbb{Z} + \{-\infty, +\infty\}$) des longueurs des chemins entre i et j.

Exemple



cycle de poids négatif

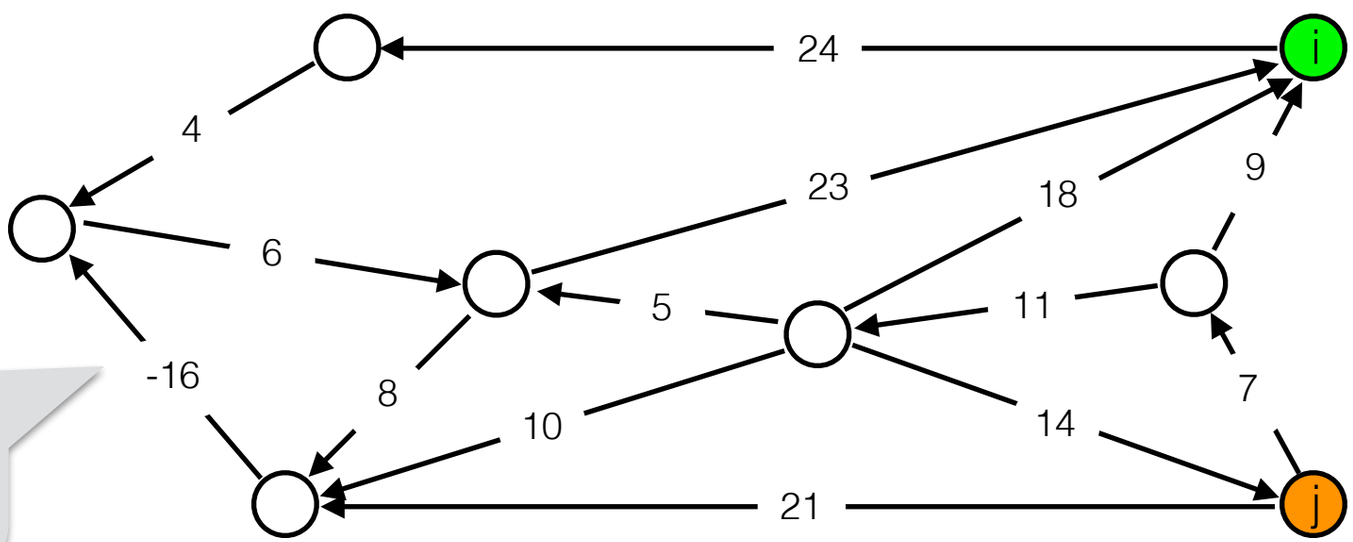
$\delta(i,j) = -\infty$

Distance minimum

Définition

Dans un graphe G orienté pondéré, la *distance minimum δ(i,j)* entre deux sommets i et j est le minimum ($\in \mathbb{Z} + \{-\infty, +\infty\}$) des longueurs des chemins entre i et j.

Exemple



pas de chemin entre i et j

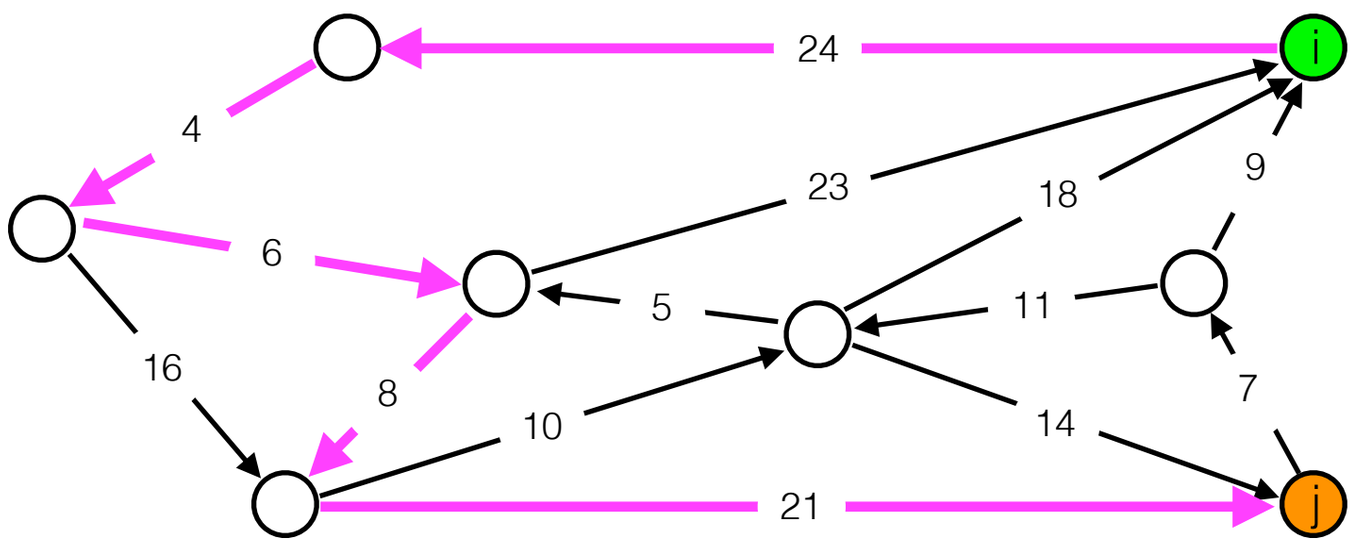
δ(i,j) = +∞

Plus courts chemins

Définition

Dans un graphe G orienté pondéré, un *plus court chemin* entre deux sommets i et j est **un** chemin de i à j dont la longueur ($\in \mathbb{Z}$) est la distance minimum entre i et j.

Exemple



Question

Définition

Dans un graphe G orienté pondéré, un *plus court chemin* entre deux sommets i et j est **un** chemin de i à j dont la longueur ($\in \mathbb{Z}$) est la distance minimum entre i et j.

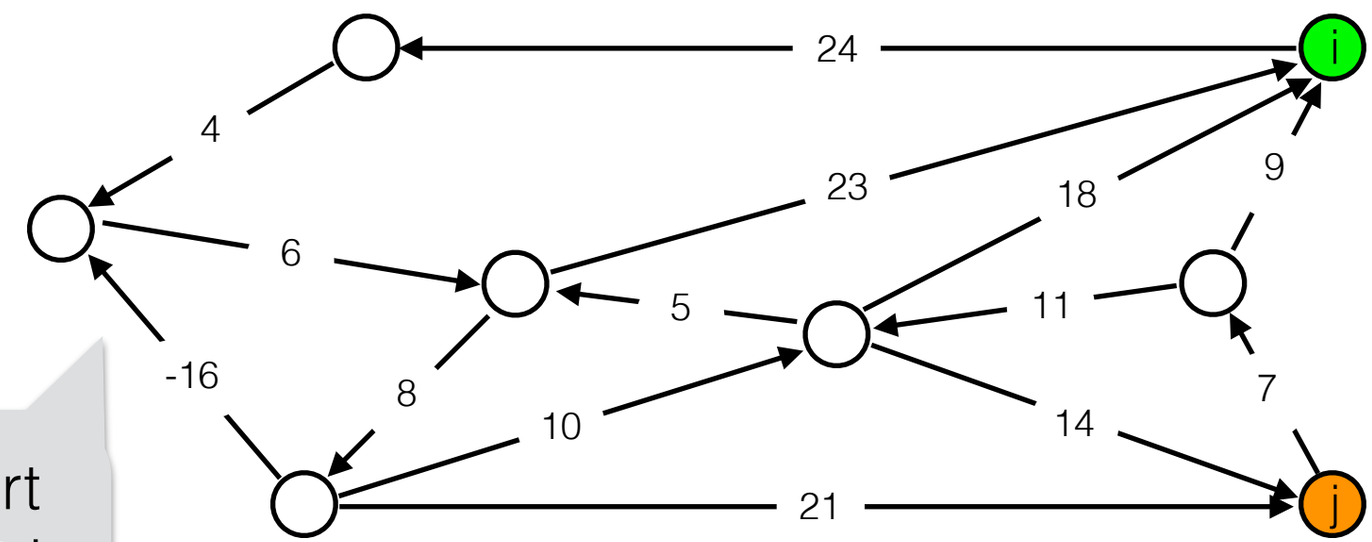
Si j est accessible depuis i, est-ce qu'il y a forcément un plus court chemin entre i et j ?

Plus courts chemins

Définition

Dans un graphe G orienté pondéré, un *plus court chemin* entre deux sommets i et j est **un** chemin de i à j dont la longueur ($\in \mathbb{Z}$) est la distance minimum entre i et j.

Contre-exemple



pas de plus-court chemin entre i et j

Calcul des plus courts chemins

- Applications
  - Calcul d’itinéraire sur une carte
  - Ordonnancement de tâche

Calcul des plus courts chemins

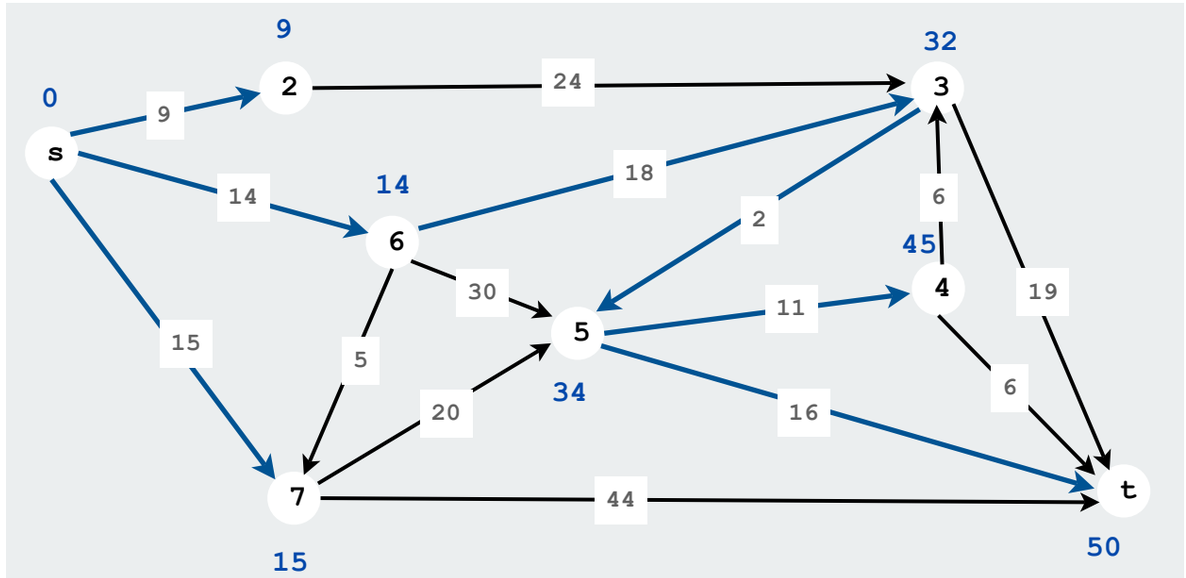
- Plusieurs versions
  - chemins entre deux sommets i et j
  - chemins à partir d'une origine unique
  - chemins entre tous couples de sommets

on s'appuie sur le calcul de tous les chemins à partir i

Calcul des plus courts chemins à partir d'une origine

- Les plus courts chemins, à partir de s, forment un **arbre**

on suppose qu'il n'y a pas de cycle de poids négatif et que tous les sommet sont accessibles depuis s



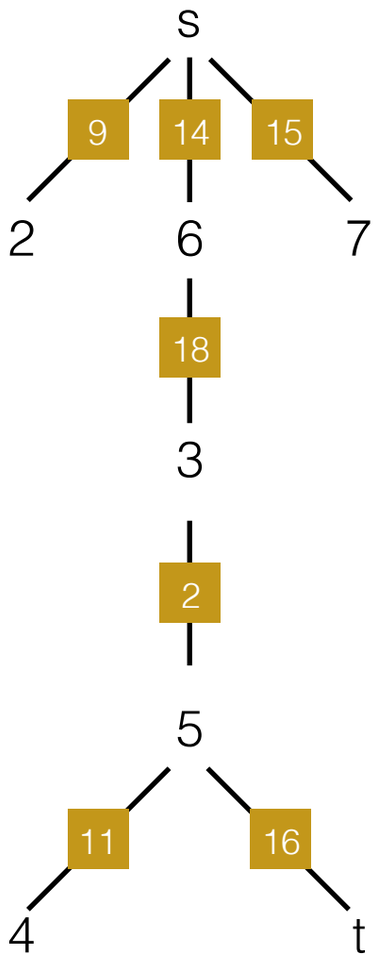
- On souhaite calculer deux tableaux
  - dist[i] : distance minimal de s à i
  - pred[i] : prédécesseur de i dans un chemin minimal de s à i

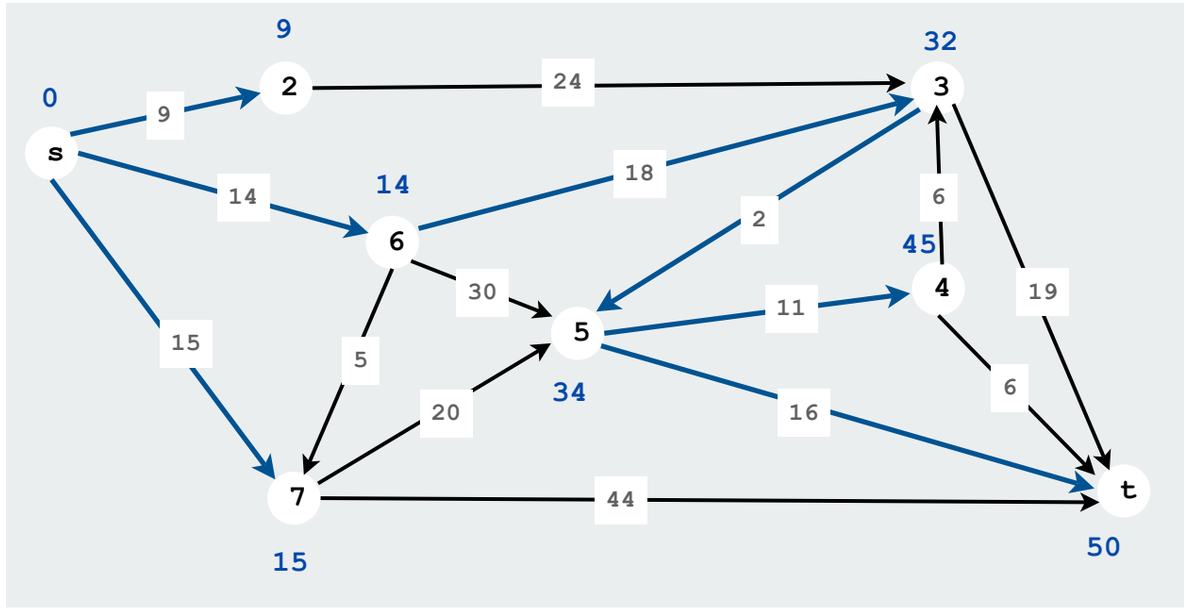
v	s	2	3	4	5	6	7	t
dist[ ]	0	9	32	45	34	14	15	50
pred[ ]	s	s	6	5	3	s	s	5

on impose pred[s]=s

i doit être adjacent à pred[i]

Calcul des plus courts chemins à partir d'une origine



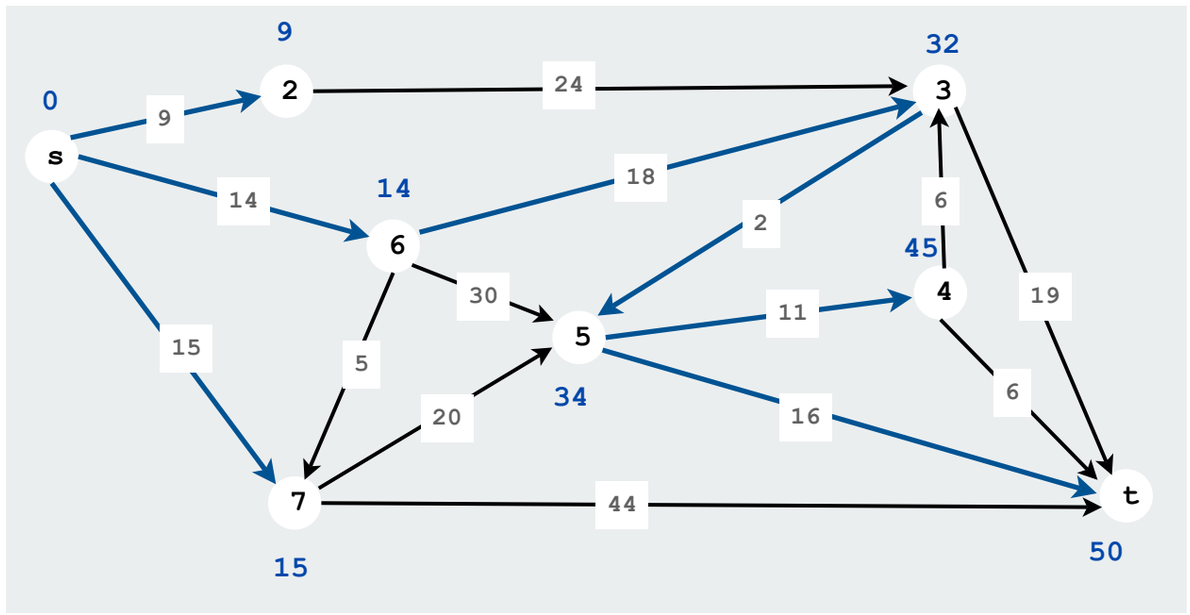


s = 0 !!

v	s	2	3	4	5	6	7	t
dist[ ]	0	9	32	45	34	14	15	50
pred[ ]	s	s	6	5	3	s	s	5

Théorème

Si j est un prédécesseur de i dans un chemin minimal de s à i, alors $\delta(s,i) = \delta(s,j)+w(j,i)$



v	s	2	3	4	5	6	7	t
dist[ ]	0	9	32	45	34	14	15	50
pred[ ]	s	s	6	5	3	s	s	5

Théorème

Si j est un prédécesseur de i dans un chemin minimal de s à i, alors $\delta(s,i) = \delta(s,j)+w(j,i)$

Preuve

- Supposons que j est un prédécesseur de i dans un chemin minimal de s à i, et montrons l'égalité.
  - Considérons un tel chemin qui va de s à i en passant par j puis l'arc (j,i). Il est de longueur $\delta(s,i)$.
  - Ce chemin de s à j est minimal sinon on pourrait construire un chemin de s à i strictement plus court que $\delta(s,i)$.
  - Le chemin considéré (de s à i) a donc pour longueur $\delta(s,j) +w(j,i)$.
  - Donc $\delta(s,i)=\delta(s,j)+w(j,i)$

Théorème

on suppose qu'il n'y a pas de cycle de poids négatif et que tous les sommet sont accessibles depuis s

Le graphe formé par la relation pred est un arbre

Preuve

- Montrons que ce graphe est acyclique.
  - S'il existait un cycle passant par un sommet i, cela signifierait qu'il existe un plus court chemin de s à i, passant par i, et même par s lui même (un sommet a un unique prédécesseur choisi)
  - C'est impossible car s est son propre prédécesseur.
- Montrons que le graphe est connexe.
  - Pour l'instant nous savons que c'est une forêt.
  - Mais si un arbre de cette forêt n'avait pas s pour racine, cela signifierai qu'il n'y a pas de chemin minimal de s à cette racine dans le graphe de départ. C'est en contradiction avec notre hypothèse.

Algorithme ordinal

- on se restreint à un graphe acyclique
- on parcours le graphe selon un ordre topologique
- à chaque étape, on examine le sommet i, mais on a déjà calculé dist[j] pour tous les prédécesseurs j de i dans le graphe.

$$dist[i] = \min \{ dist[j] + w(j,i) \mid j \text{ prédécesseur de } i \}$$

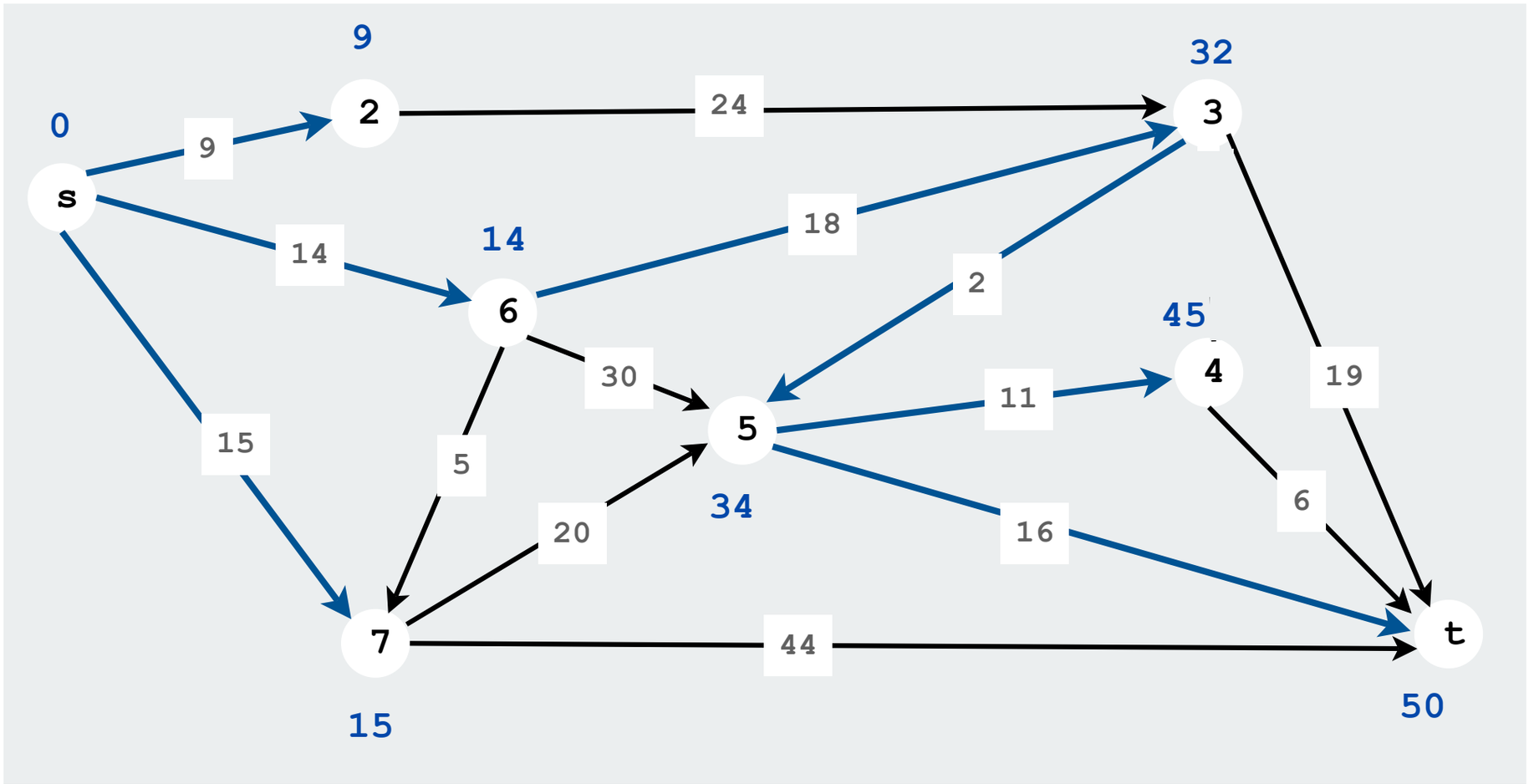
Correct d'après les théorèmes précédents

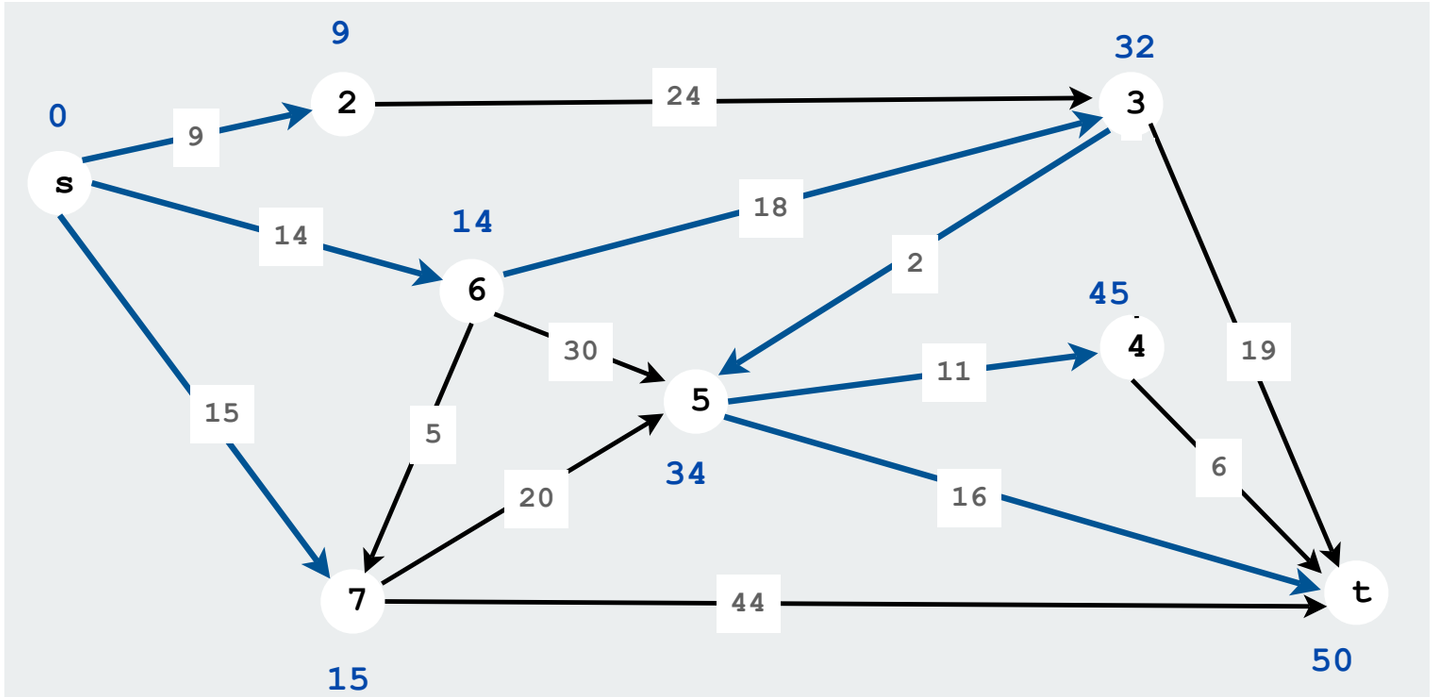
Algorithme ordinal

Version en arrière

```
ORDINAL(G,s)=
  n ⟵ nombre de sommets de G
  ord ⟵ TRI_TOPOLOGIQUE(G)
  dist ⟵ [+oo, …, +oo ]
  pour k=0 à n-1
     i ⟵ ord[k]
     si i=s alors dist[i] ⟵ 0
     sinon
        pour tout j∈Adj^-1[i]
          si dist[j]+w(j,i)<dist[i]
          alors
             dist[i] ⟵ dist[j]+w(j,i)
             pred[i] ⟵ j
```

Algorithme ordinal





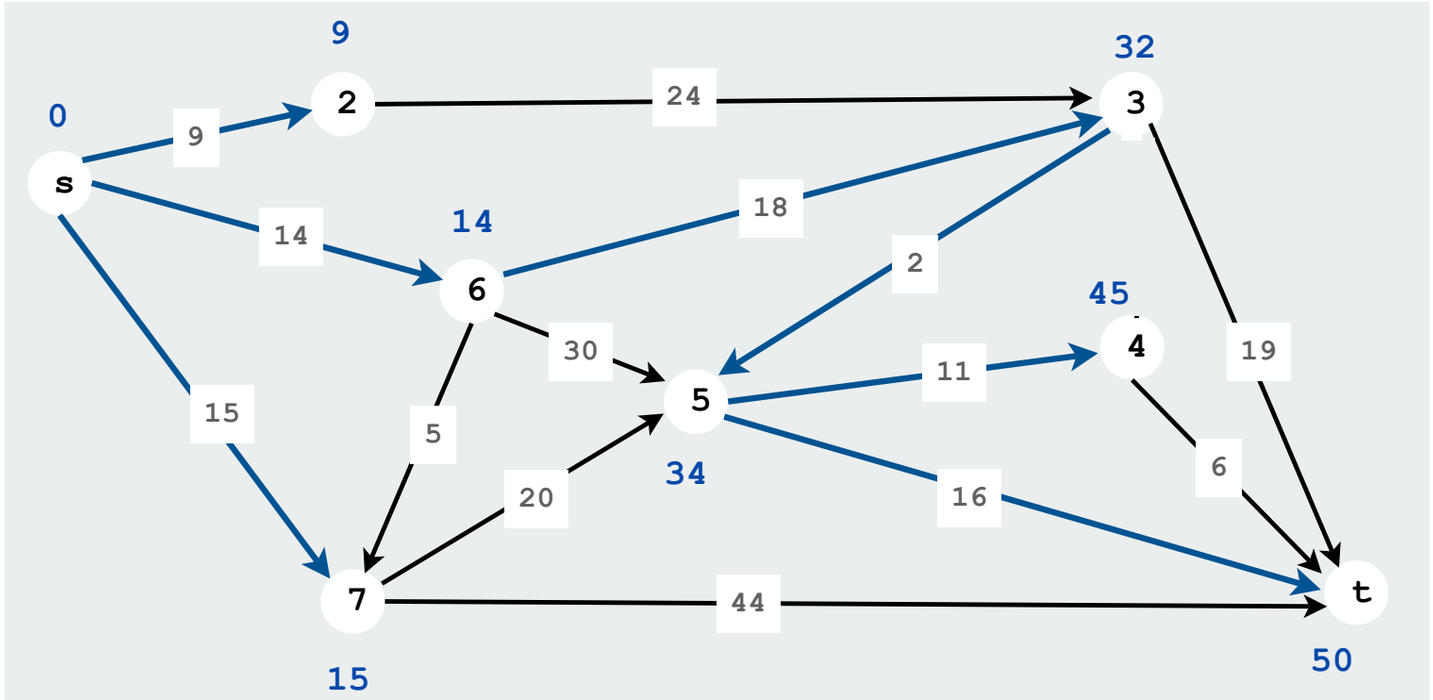
k	ord[k]	dist[s]	dist[2]	dist[6]	dist[7]	dist[3]	dist[5]	dist[4]	dist[t]
0	s	0	+∞	+∞	+∞	+∞	+∞	+∞	+∞
1	2	0	9	+∞	+∞	+∞	+∞	+∞	+∞
2	6	0	9	14	+∞	+∞	+∞	+∞	+∞
3	7	0	9	14	15	+∞	+∞	+∞	+∞
4	3	0	9	14	15	32	+∞	+∞	+∞
5	5	0	9	14	15	32	34	+∞	+∞
6	4	0	9	14	15	32	34	45	+∞
7	t	0	9	14	15	32	34	45	50

Algorithme ordinal

Version en avant

```
ORDINAL(G,s)=
  n ⟵ nombre de sommets de G
  ord ⟵ TRI_TOPOLOGIQUE(G)
  dist ⟵ [+oo, …, +oo ]
  dist[s] ⟵ 0
  pour k=0 à n-1
     i ⟵ ord[k]
     pour tout j∈Adj [i]
        si dist[i]+w(i,j)<dist[j]
        alors
           dist[j] ⟵ dist[i]+w(i,j)
           pred[j] ⟵ i
```

Correction : lorsqu'on traite i, sa distance est déjà correctement calculée



k	ord[k]	dist[s]	dist[2]	dist[6]	dist[7]	dist[3]	dist[5]	dist[4]	dist[t]
0	s	0	9	14	15	+∞	+∞	+∞	+∞
1	2	0	9	14	15	33	+∞	+∞	+∞
2	6	0	9	14	15	32	44	+∞	+∞
3	7	0	9	14	15	32	35	+∞	59
4	3	0	9	14	15	32	34	+∞	51
5	5	0	9	14	15	32	34	35	50
6	4	0	9	14	15	32	34	35	50
7	t	0	9	14	15	32	34	45	50

Algorithme de Dijkstra

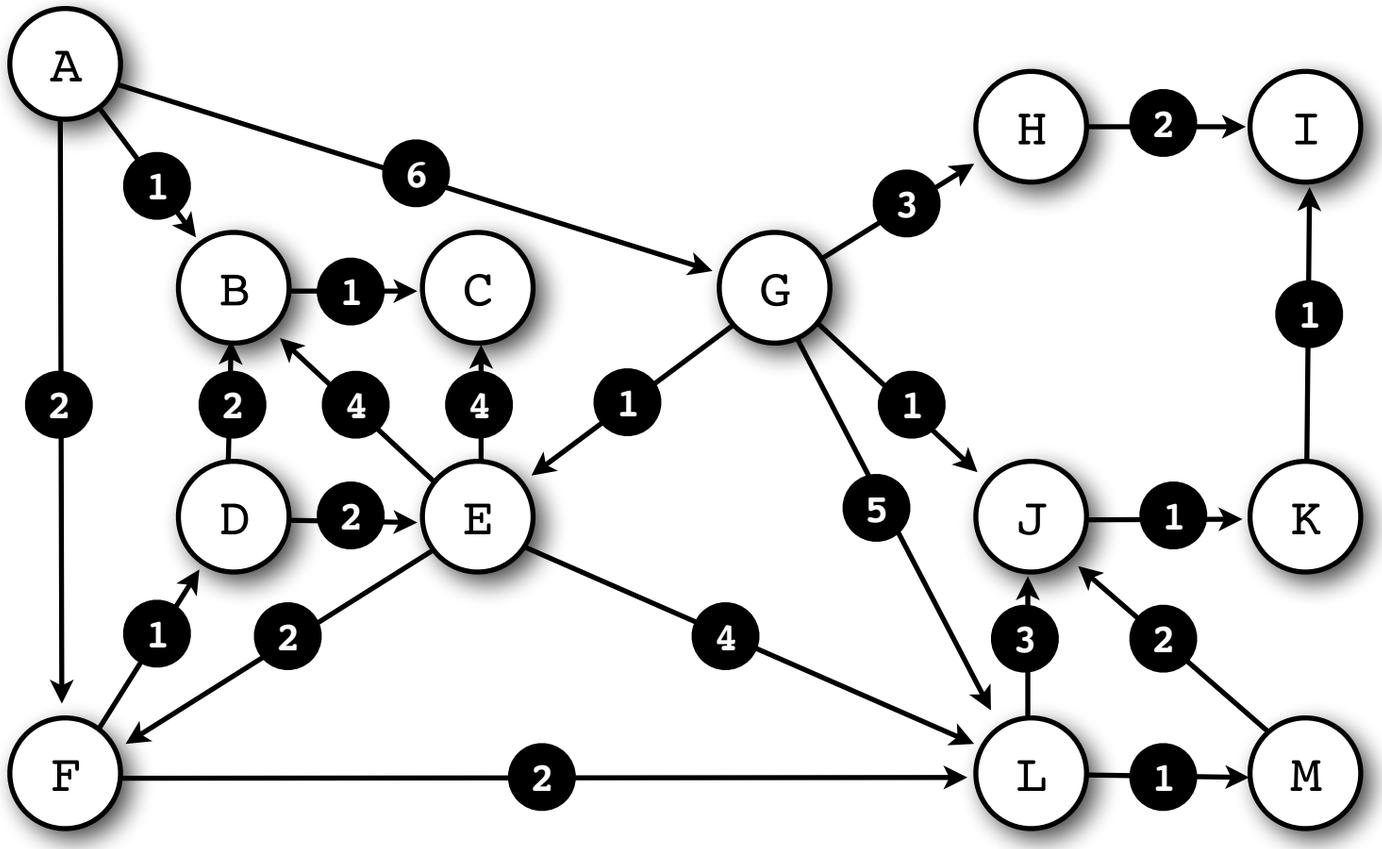
- on se restreint à un graphe sans poids négatifs

```
DIJKSTRA(G,s)=
   dist ⟵ [+oo, …, +oo ]
   dist[s] ⟵ 0
   S ⟵ {}
   S’ ⟵ tous les sommets de G
   tant que S’ non vide
      i ⟵ choisir i∈S’, tel que dist[i] minimal
      S’ ⟵ S’-{i}
      S ⟵ S+{i}
      pour tout j∈Adj [i]
           si dist[i]+w(i,j)<dist[j]
           alors
                dist[j] ⟵ dist[i]+w(i,j)
                pred[j] ⟵ i
```

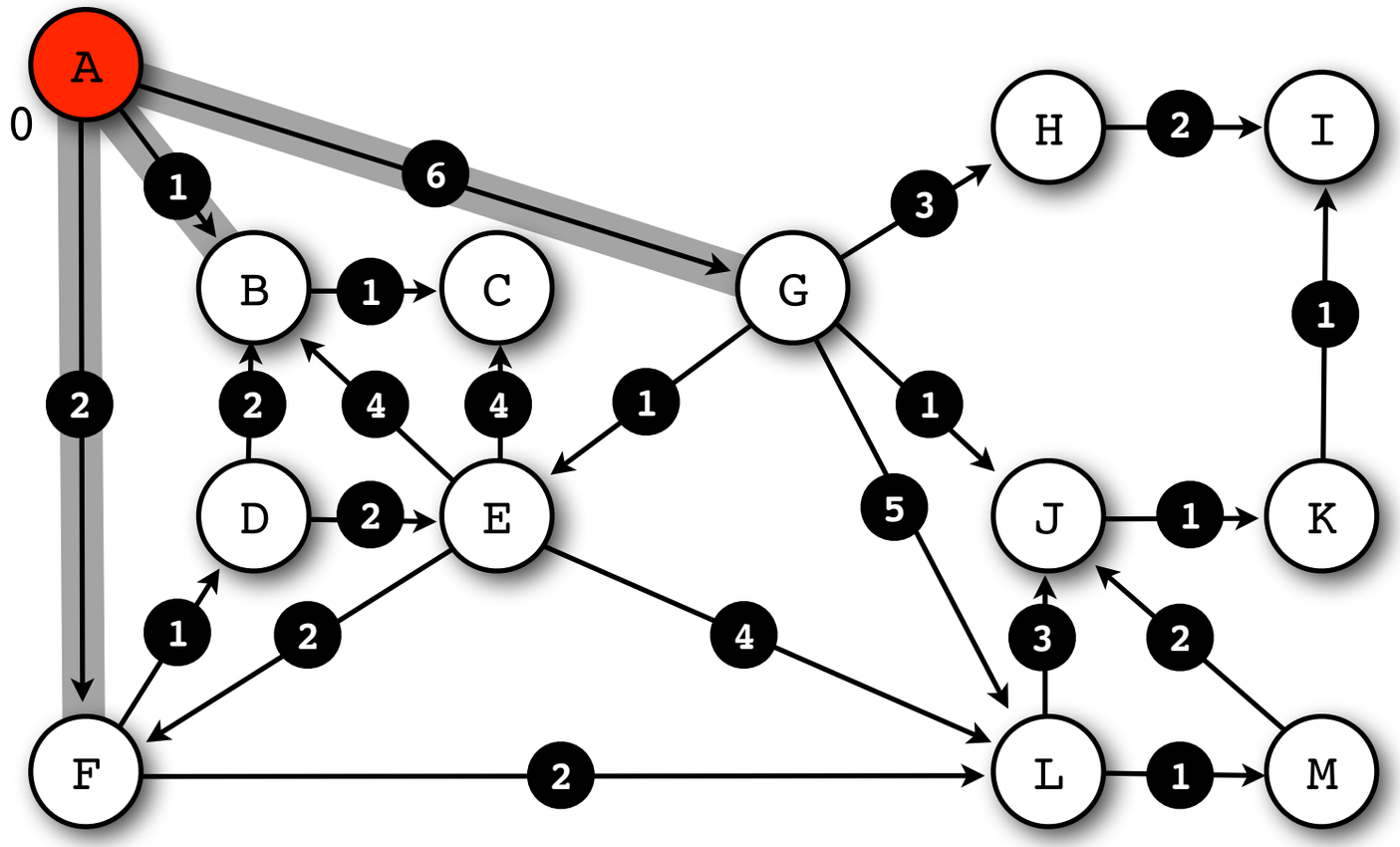
Idée : S va toujours contenir les sommets dont nous avons déjà correctement calculé la distance minimale.

La correction n’est plus aussi évidente que pour l’algorithme ordinal car le graphe peut contenir des cycles

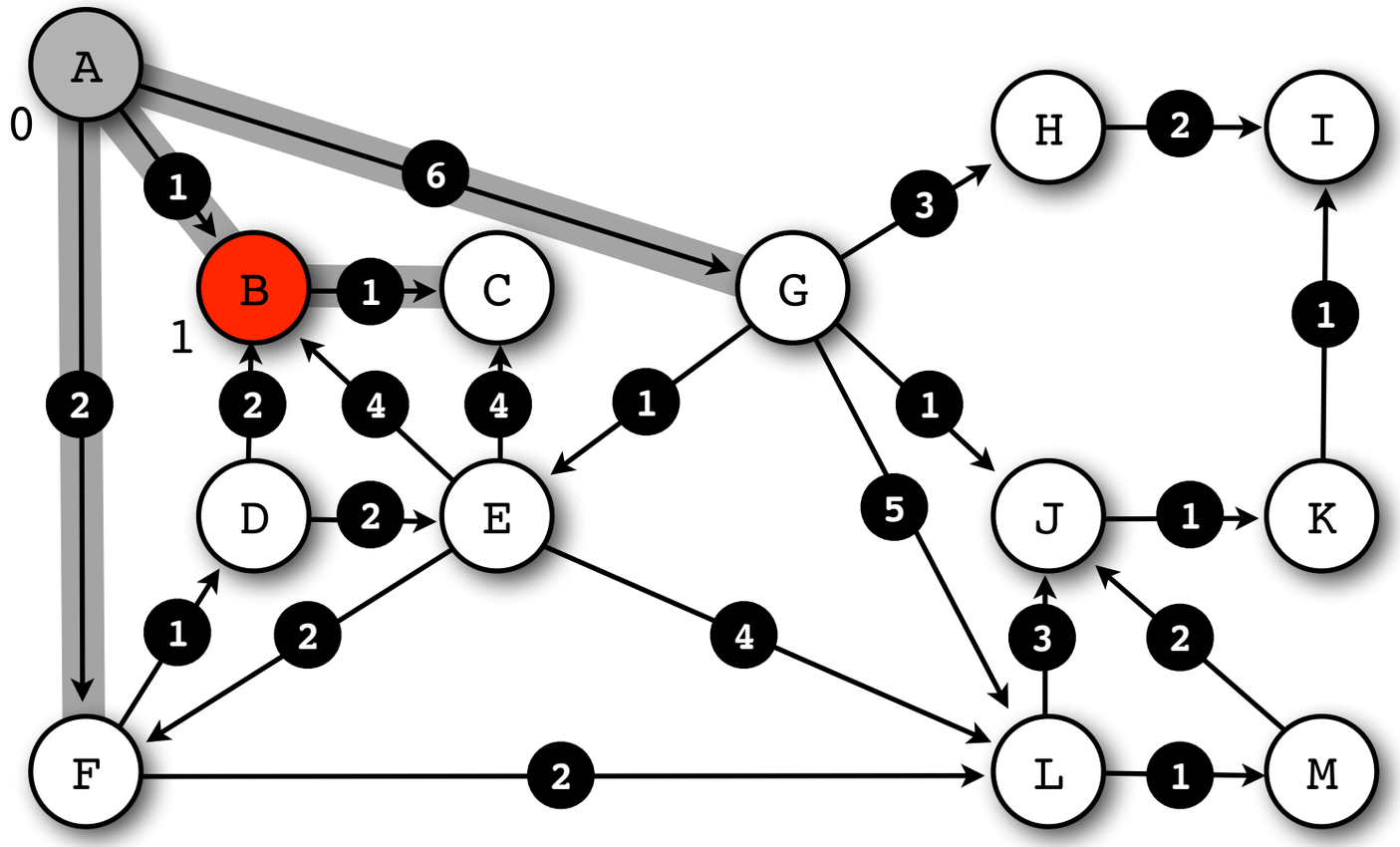
A	0
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞
H	∞
I	∞
J	∞
K	∞
L	∞
M	∞



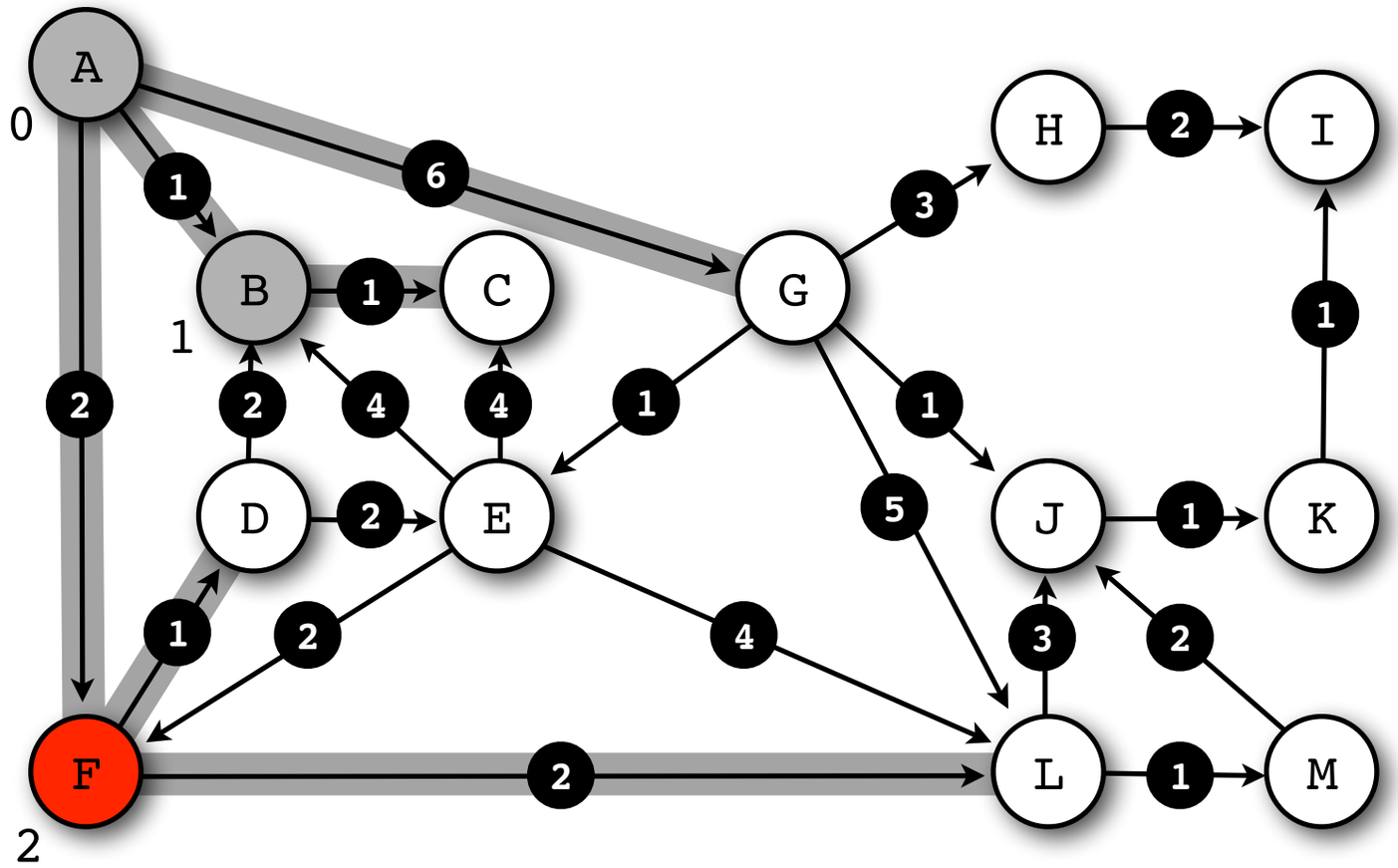
B	1
F	2
G	6
E	∞
C	∞
D	∞
H	∞
I	∞
J	∞
K	∞
L	∞
M	∞



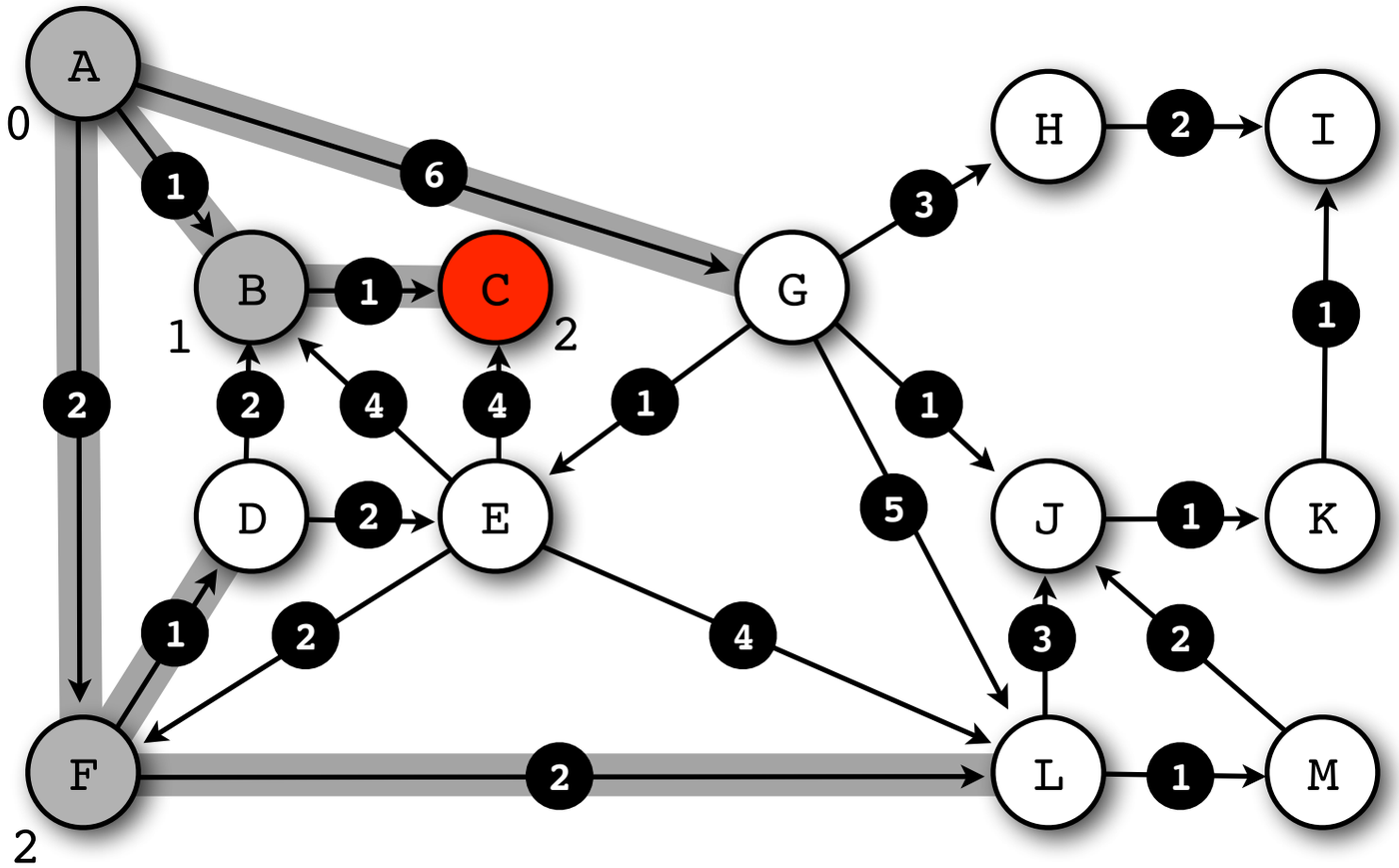
F	2
C	2
G	6
E	∞
D	∞
H	∞
I	∞
J	∞
K	∞
L	∞
M	∞



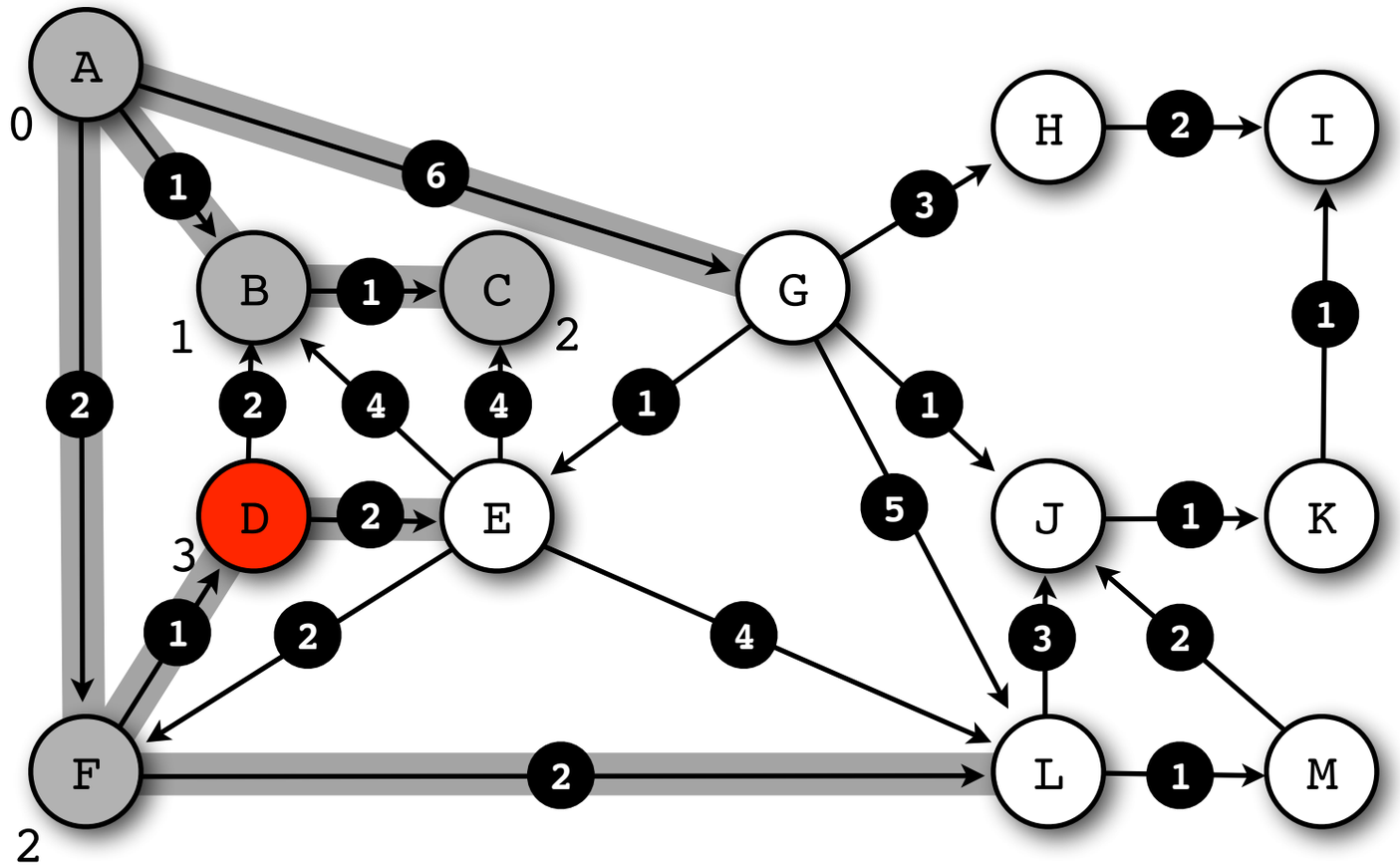
C	2
D	3
L	4
G	6
H	∞
I	∞
J	∞
K	∞
E	∞
M	∞

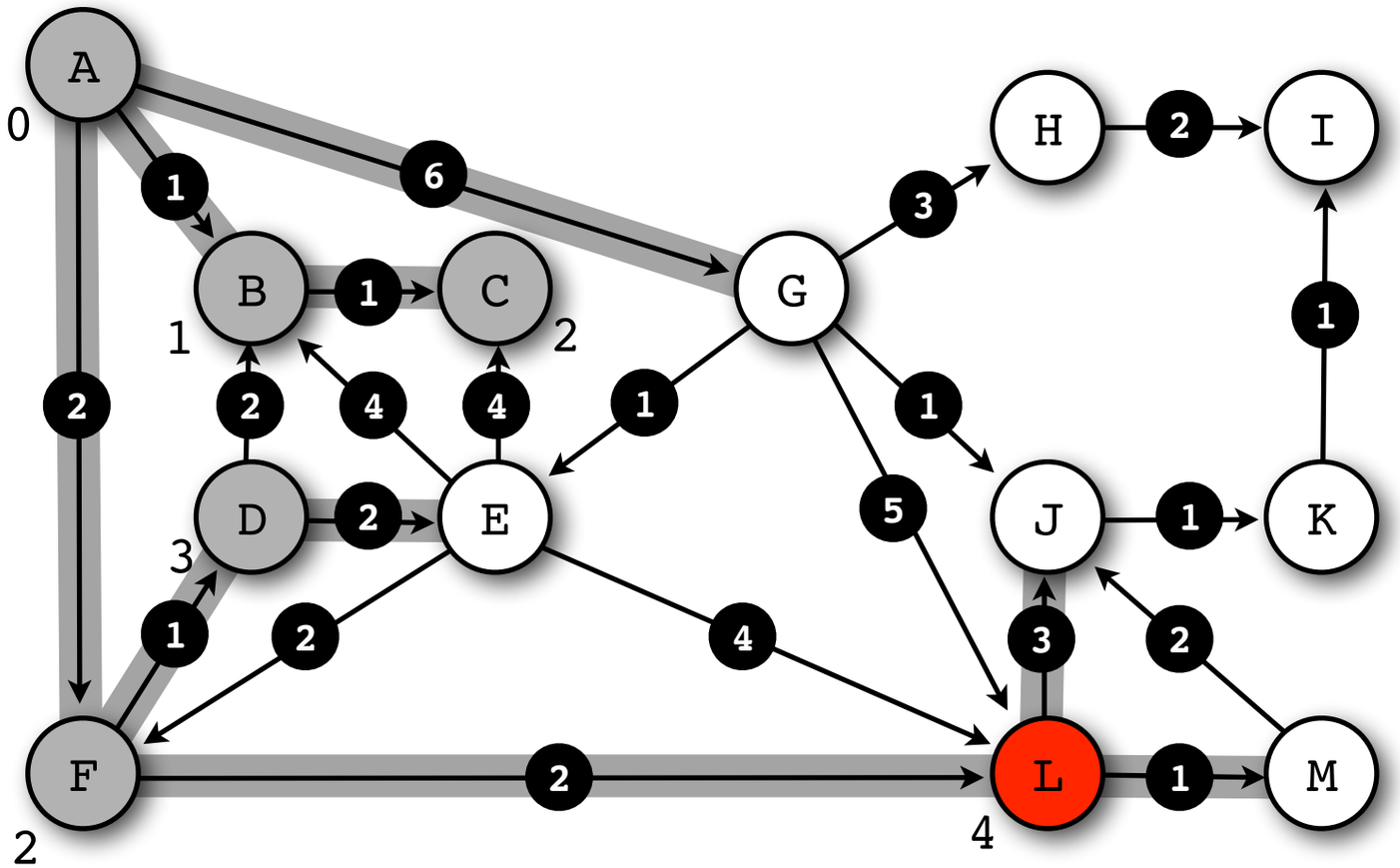


D	3
L	4
G	6
H	∞
I	∞
J	∞
K	∞
E	∞
M	∞

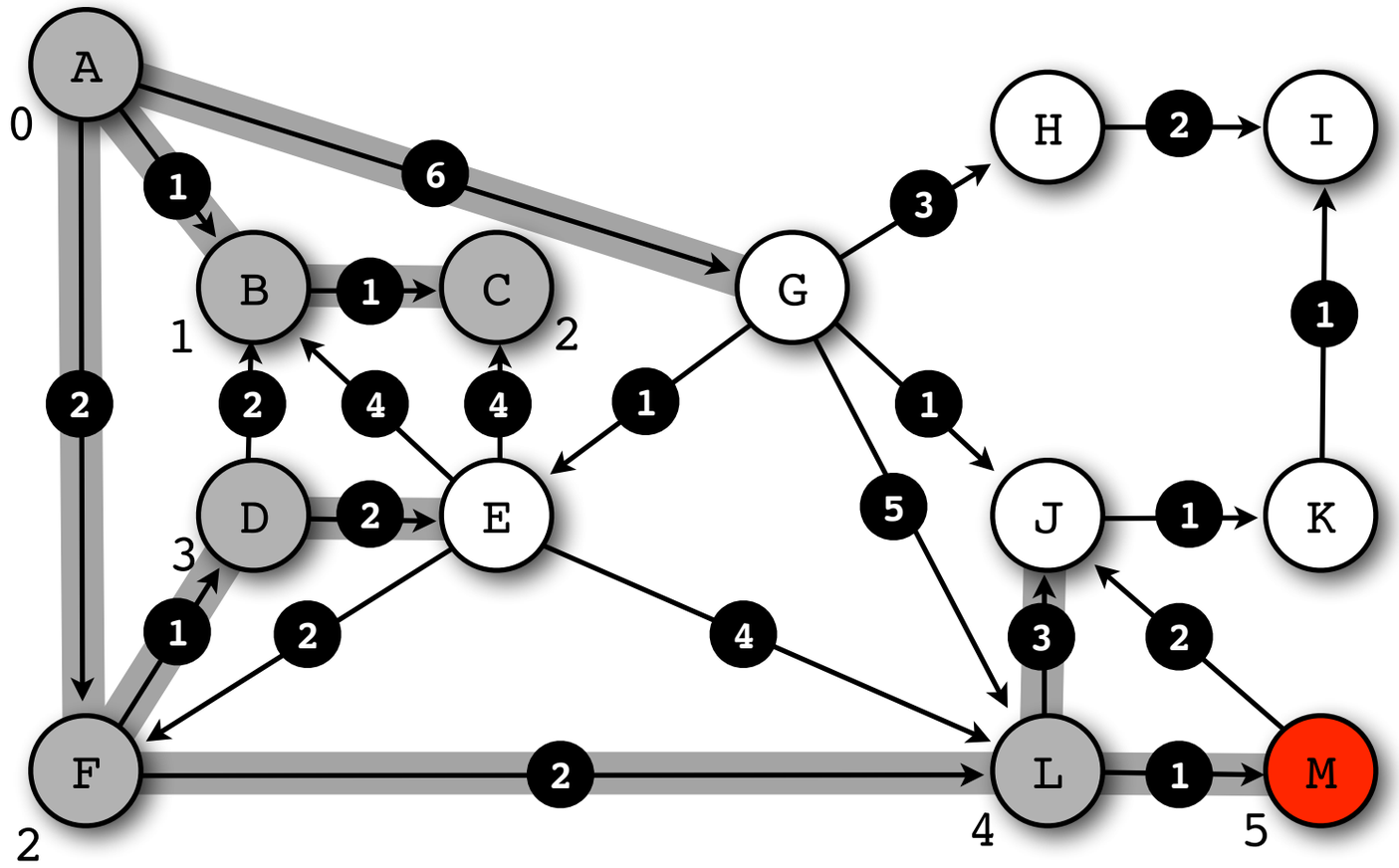


L	4
E	5
G	6
I	∞
J	∞
K	∞
H	∞
M	∞

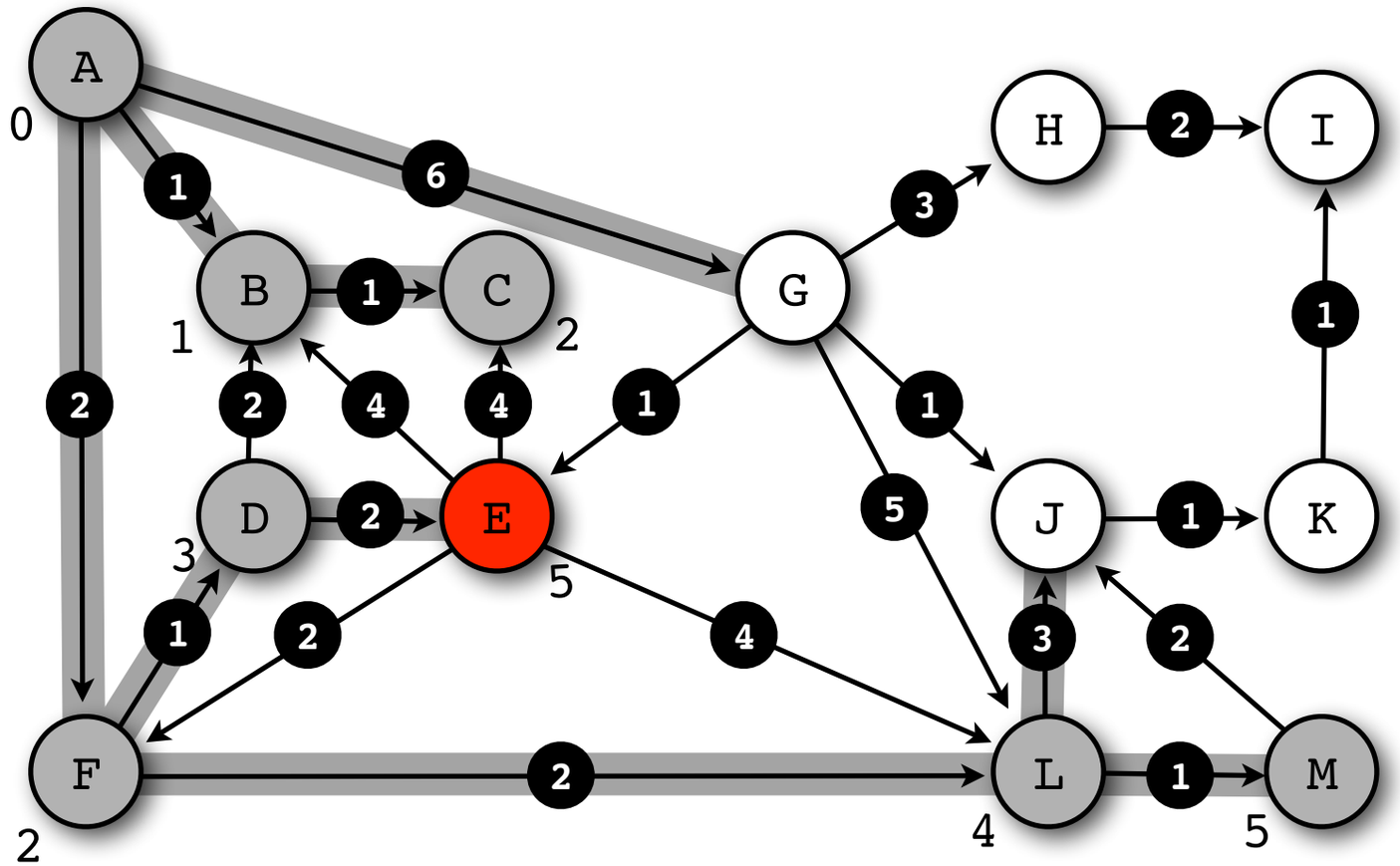




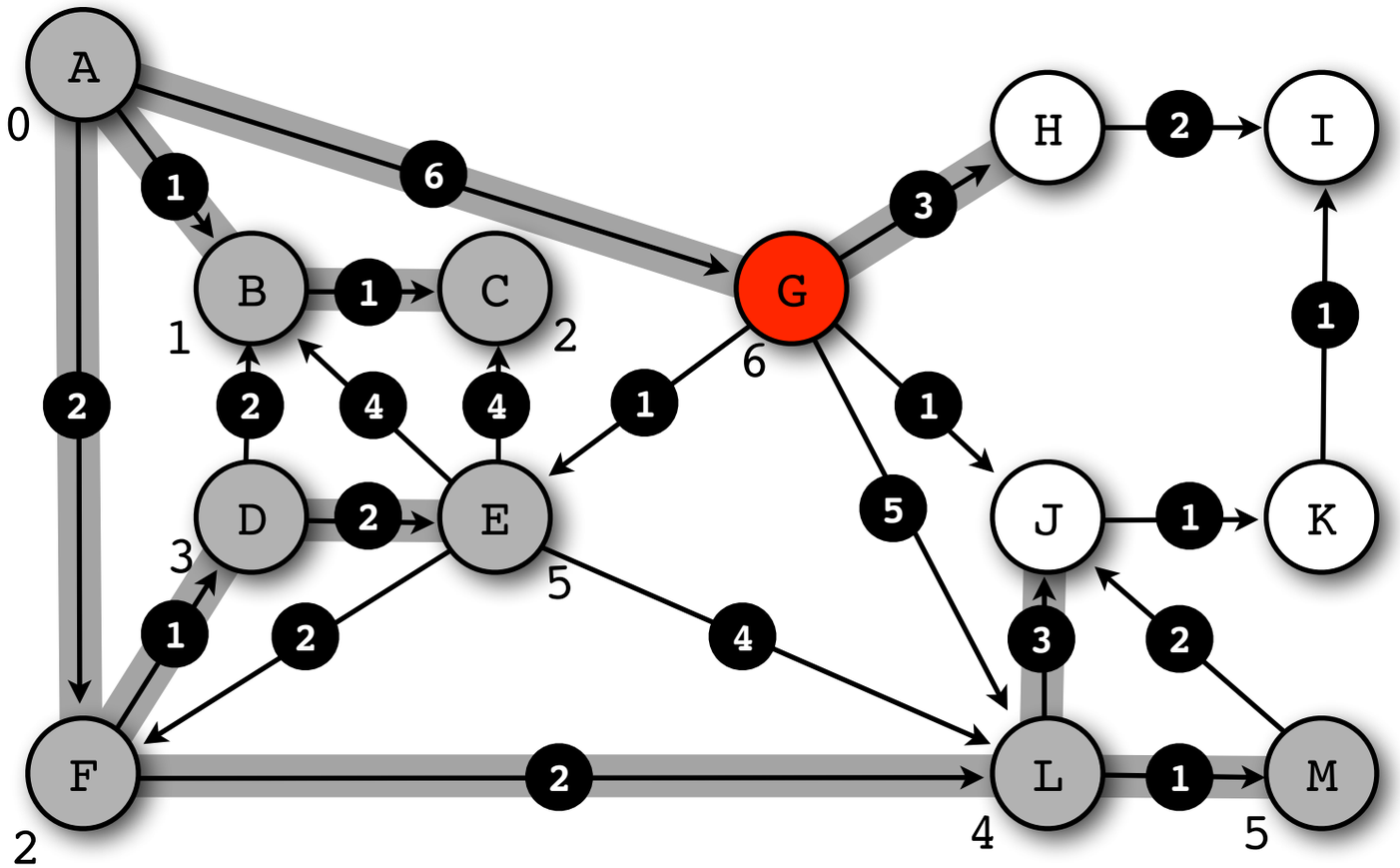
M	5
E	5
G	6
J	7
K	∞
H	∞
I	∞

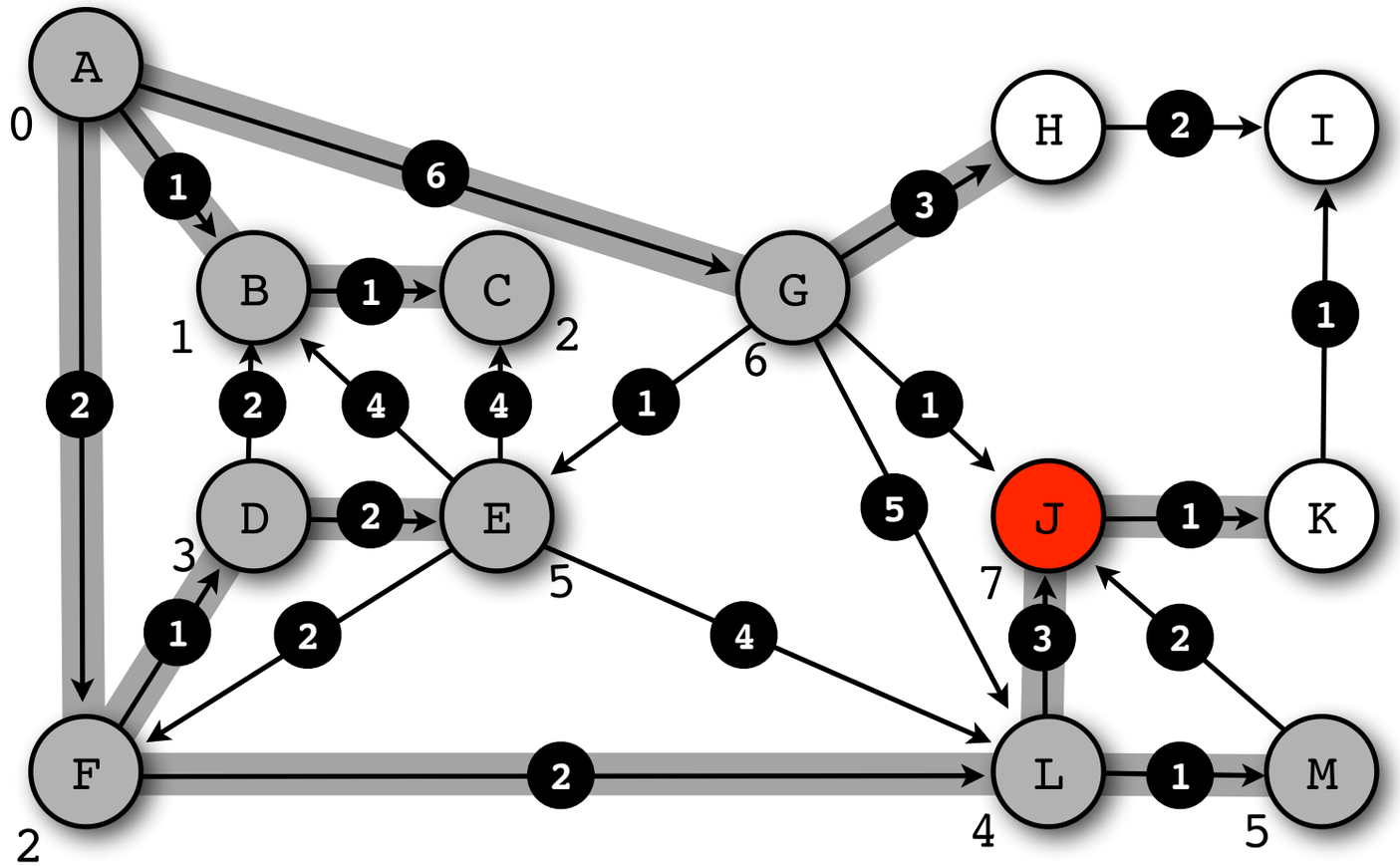


E	5
G	6
J	7
K	∞
H	∞
I	∞

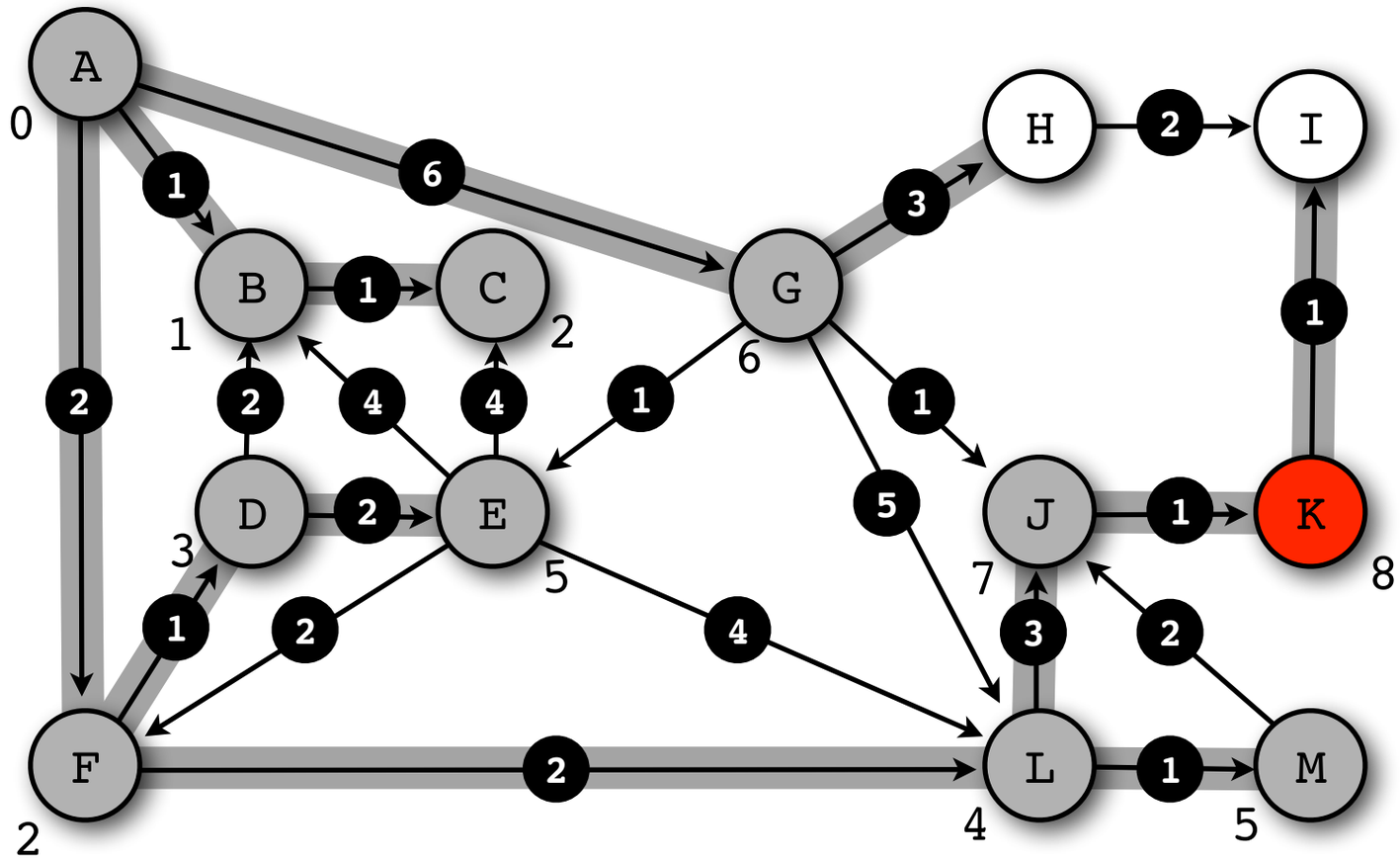


G	6
J	7
K	∞
H	∞
I	∞

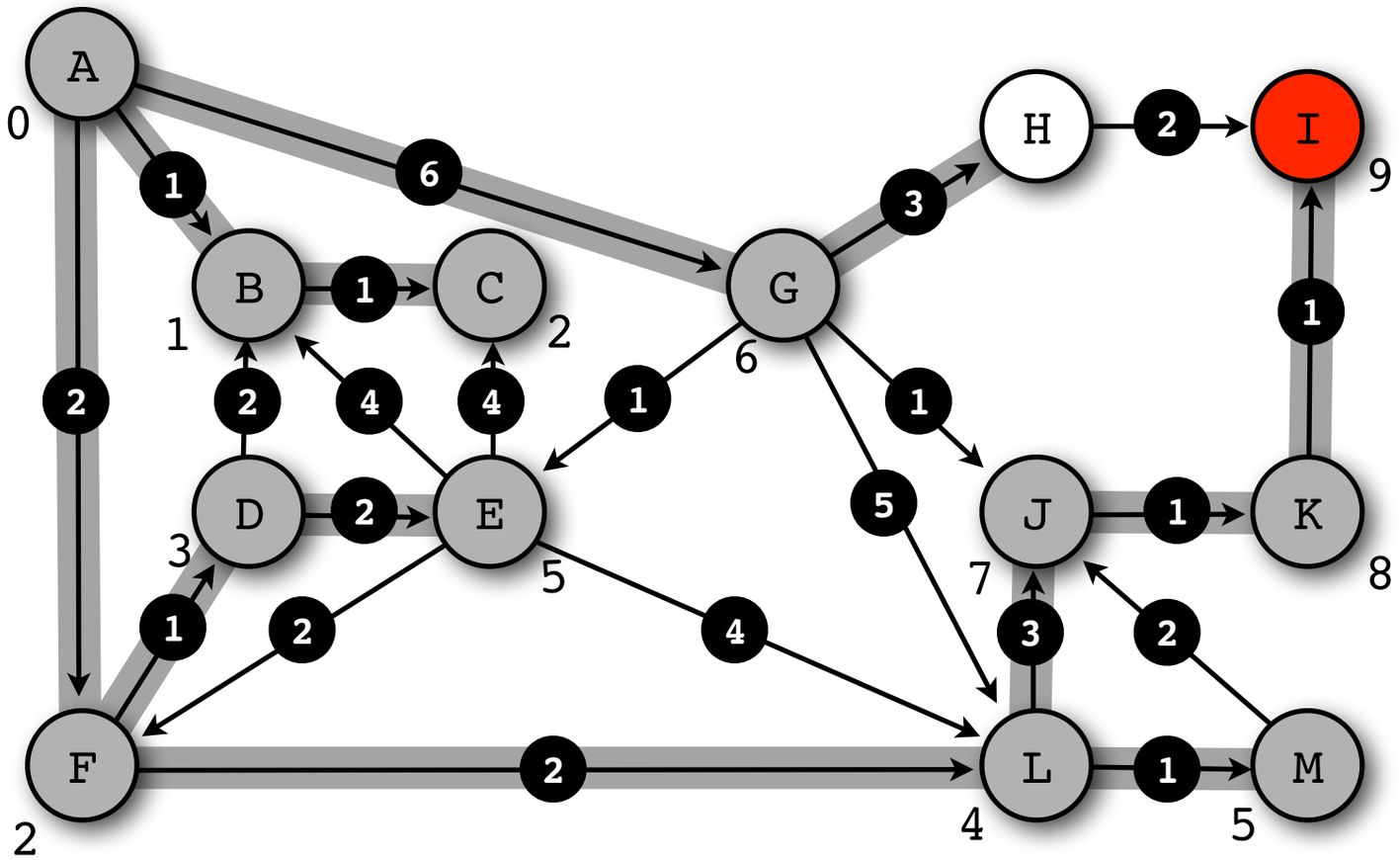

A
0
B
1
C
2
D
3
E
5
F
2
G
6
H
9
I
∞
J
7
K
∞
L
4
M
5
1
6
2
3
1
2
4
4
1
1
5
2
1
2
4
3
2
1
2



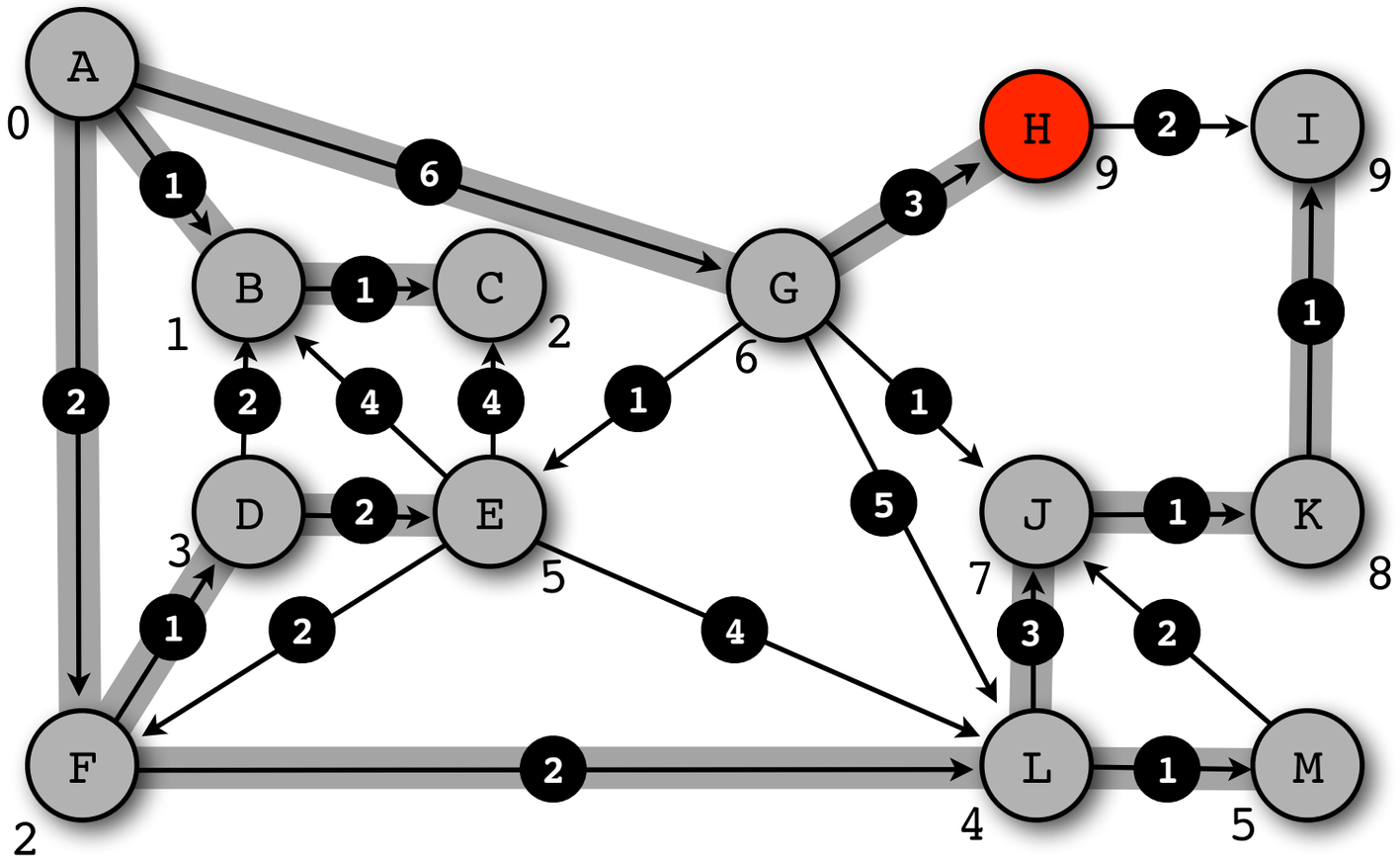
K	8
H	9
I	∞

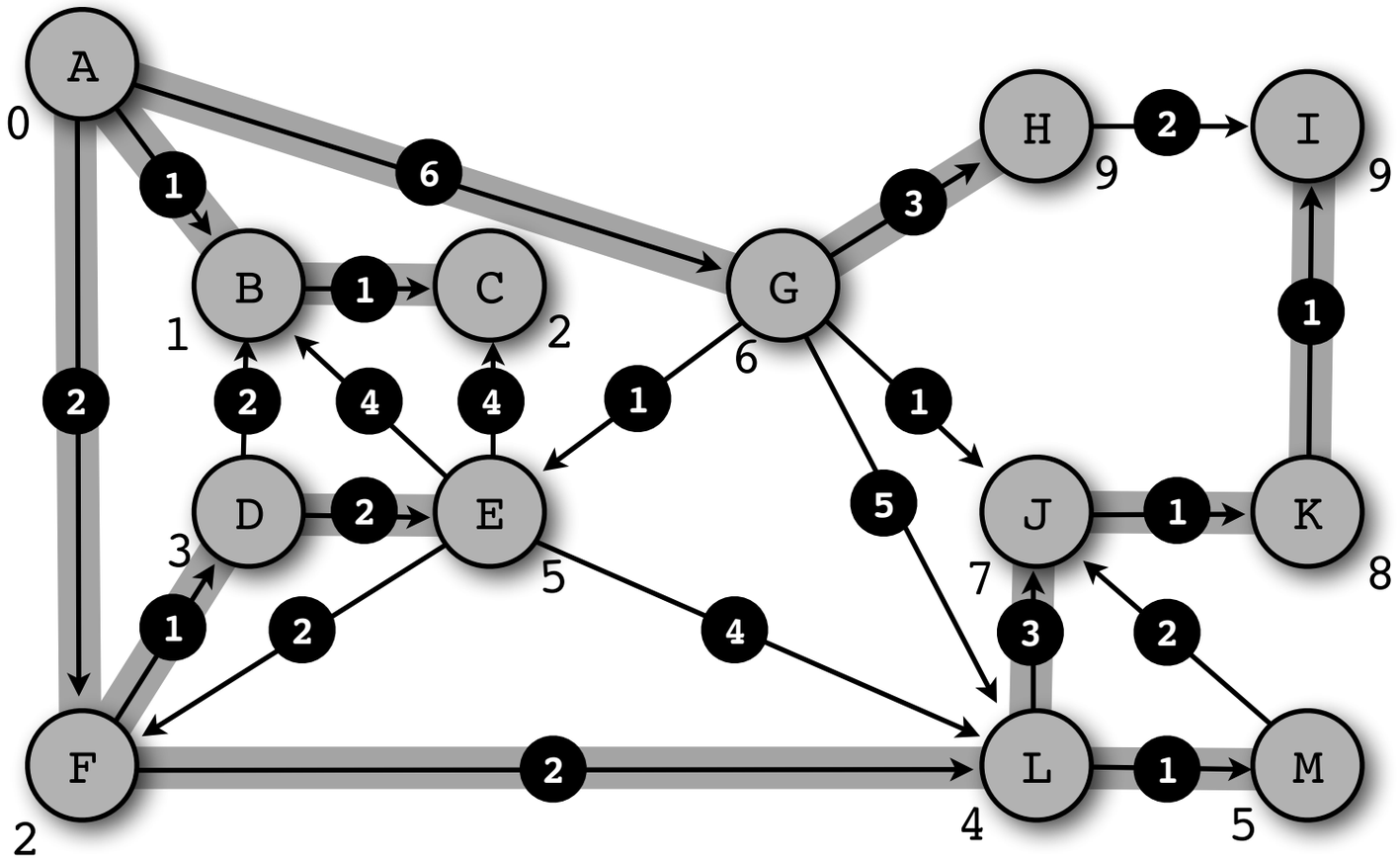


I	9
H	9



H 9


A
0
B
1
C
2
D
3
E
5
F
2
G
6
H
9
I
9
J
7
K
8
L
4
M
5
1
6
2
1
2
4
4
1
3
1
5
1
2
4
3
2
2
1
2


A
0
B
1
C
2
D
3
E
5
F
2
G
6
H
9
I
9
J
7
K
8
L
4
M
5
1
6
2
1
2
4
4
1
3
1
5
1
2
4
3
2
2
1
1
2