Plus courts chemins

Olivier Baudon

Université de Bordeaux

24 février 2021

Plus courts chemins

Problématiques

Soit G un graphe orienté et ω une fonction de poids sur les arcs de G.
Si G est un graphe non orienté sans boucle, on peut considérer chaque arête comme une paire d'arcs symétriques.

1. Quel est le plus court chemin pour aller de u à v ?
2. Quel est le plus court chemin d'un sommet u à tous les sommets du graphe ?
3. Quel est le plus court chemin entre toute paire de sommets ?

Plus courts chemins

Théorème

Il existe un plus court chemin de u à v dans (G, w) si et seulement si il existe un chemin de u à v et aucun chemin de u à v ne contient de circuit de longueur totale strictement négative.

Arborescence

L'ensemble des plus courts chemins à partir d'un sommet s forme une arborescence de racine s.
On notera par $d(v)$ la distance trouvée de s à v et par $\pi(v)$ le prédécesseur de v sur le chemin de s à v de longueur $d(v)$.

Plus courts chemins

Principe du relâchement

Soient
d la valeur du plus court chemin trouvé à l'instant t entre un sommet s et les sommets de G,
π les prédécesseurs de chaque sommet sur les plus courts chemins trouvés à l'instant t depuis le sommet s,
uv un arc de G,
Relacher(G, ω, u, v**)** :

1: **si** $d(u) + \omega(uv) < d(v)$ **alors**
2: $\quad d(v) \leftarrow d(u) + \omega(uv)$
3: $\quad \pi(v) \leftarrow u$
4: **fin si**

Algorithme de Dijkstra

Utilisation

On peut utiliser l'algorithme de Dijkstra à condition que la fonction ω de pondération des arcs soit positive.

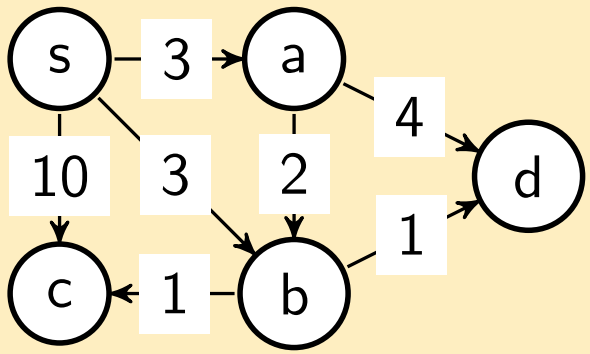
Algorithme de Dijkstra

G : graphe orienté,
$w : E(G) \to \mathbb{R}^+$,
$s \in V(G)$

1: **pour tout** v de $V(G)$ **faire**
2: $d(v) \leftarrow \infty$
3: $pere(v) \leftarrow NIL$
4: $couleur(v) \leftarrow BLANC$
5: **fin pour**
6: $d(s) \leftarrow 0$
7: $F \leftarrow FILE\_PRIORITE(\{s\}, d)$
8: **tant que** $F \neq \emptyset$ **faire**
9: $pivot \leftarrow EXTRAIRE\_MIN(F)$
10: **pour tout** $e = (pivot, v)$ arc sortant de $pivot$ **faire**
11: **si** $couleur(v) = BLANC$ **alors**
12: **si** $d(v) = \infty$ **alors**
13: INSERER(F, v)
14: **fin si**
15: **si** $d[v] > d[pivot] + w(e)$ **alors**
16: $d[v] \leftarrow d[pivot] + w(e)$
17: $pere[v] \leftarrow pivot$
18: **fin si**
19: **fin si**
20: **fin pour**
21: $couleur[pivot] \leftarrow NOIR$
22: **fin tant que**

Algorithme de Dijkstra

Exemple



Graphe G

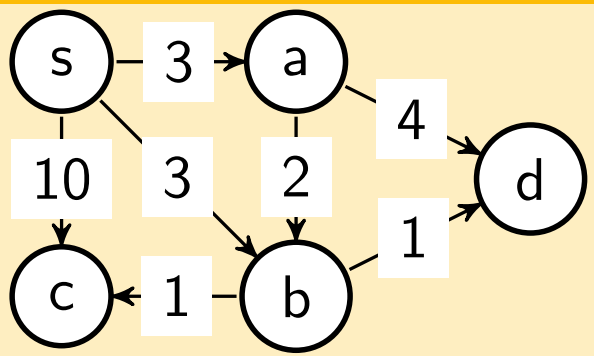
pivot	*s*	*a*	*b*	*c*	*d*
	0	∞	∞	∞	∞
s	*X*	3	3	10	∞
a		*X*			7
b			*X*	4	4
c				*X*	
d					*X*

Dijkstra(G, ω, s)

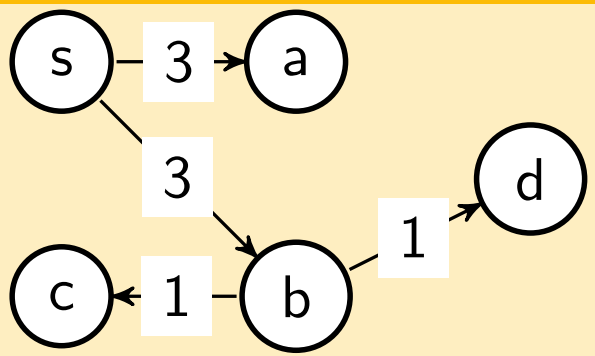
Remarque : $\pi(s) = \text{NIL}$ et $\pi(v)$ est le pivot quand $d(v)$ a été modifié pour la dernière fois.
Ici : $\pi(s) = NIL, \pi(a) = s, \pi(b) = s, \pi(c) = b, \pi(d) = b$.

Algorithme de Dijkstra

Exemple - Arborescence des plus courts chemins



Graphe G



Arborescence

Algorithme de Dijkstra

Complexité

Si la file de priorité est implémentée sous la forme d'une liste, alors la complexité de l'algorithme de Dijkstra est en $O(n^2 + m)$. Si G ne possède pas d'arcs parallèles, cela nous donne du $O(n^2)$.
Si le nombre d'arcs est faible par rapport à n^2, on a intérêt à implémenter la file de priorité à l'aide d'un tas binaire. Dans ce cas, la complexité de Dijkstra devient $O((n + m) \times \log(n))$.

Algorithme de Dijkstra

Validité

On note par $\delta(u)$ la plus courte distance entre s et u dans G.
On va montrer que $d(u) = \delta(u)$ à chaque fois qu'un sommet u est extrait de la file de priorité F.
Supposons que ce ne soit pas le cas. Soit u le premier sommet extrait de F tel que $d(u) \neq \delta(u)$ à cet instant.
u ne peut pas être égal à s, car s est le premier sommet extrait de F et $d(s) = \delta(s) = 0$.
De plus, $d(u) \neq \infty$, sinon, u n'aurait pas été inséré dans F. Donc il existe lors de l'extraction de u un chemin de s à u de longueur $d[u] \neq \infty$ et donc il existe un plus court chemin dans G de longueur $\delta(u) \neq \infty$. Soit p ce chemin et soit y le premier sommet de p non extrait de F quand on extrait u. y existe, sinon on aurait $d(u) = \delta(u)$. Soit x le prédécesseur de y dans p. Comme x a été extrait de F avant u, cela signifie que $d(y) = \delta(y)$ quand on extrait u et comme $\delta(y) < \delta(u) < d[u]$, y aurait du être extrait de F avant u. Donc u n'existe pas !

Algorithme de Bellman

Utilisation

On peut utiliser l'algorithme de Bellman à condition que le graphe G soit sans circuit.
Dans ce cas, on peut obtenir un tri topologique sur les sommets de G, c'est à dire un ordre tel que si un arc va du sommet de rang i vers le sommet de rang j, alors $i < j$.

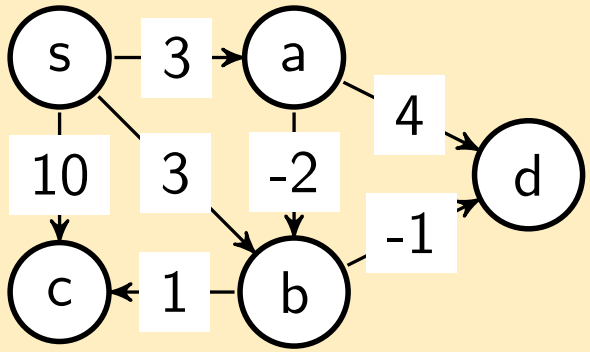
Algorithme de Bellman

G : graphe orienté,
$w : E(G) \to \mathbb{R}$
$s \in V(G)$

1: **pour tout** v de $V(G)$ **faire**
2: $d[v] \leftarrow \infty$
3: $pere[v] \leftarrow NIL$
4: $npred[v) \leftarrow deg^-[v]$
5: **si** $npred(v) = 0$ **alors**
6: INSERER_FILE(F, v)
7: **fin si**
8: **fin pour**
9: $d[s] \leftarrow 0$
10: **tant que** F non vide **faire**
11: $u \leftarrow TETE\_FILE(F)$
12: DEFILER(F)
13: **pour** $v \in Adj(u)$ **faire**
14: **si** $d[v] > d[u] + w(u, v)$ **alors**
15: $d[v] \leftarrow d[u] + w[u, v]$
16: $pere[v] \leftarrow u$
17: **fin si**
18: $npred(v) \leftarrow npred[v] - 1$
19: **si** $npred(v) = 0$ **alors**
20: INSERER_FILE(F, v)
21: **fin si**
22: **fin pour**
23: **fin tant que**

Algorithme de Bellman

Exemple



Graphe G

u	s	a	b	c	d	*File*
	0	∞	∞	∞	∞	s
s	0	3	3	10	∞	a
a	0	3	1	10	7	b
b	0	3	1	2	0	cd
c	0	3	1	2	0	d
d	0	3	1	2	0	$\emptyset$

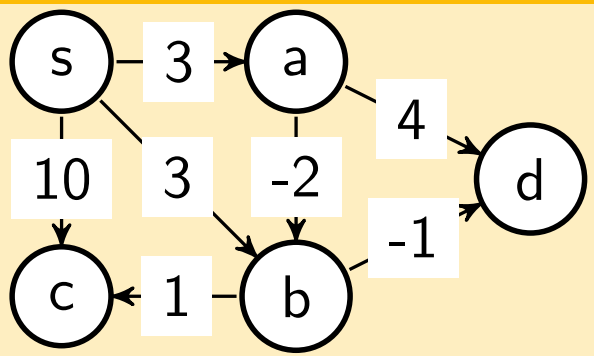
Bellman(G, ω, s)

Remarque : $\pi(s) =$ NIL et $\pi(v)$ est la tête de liste quand $d(v)$ a été modifié pour la dernière fois.

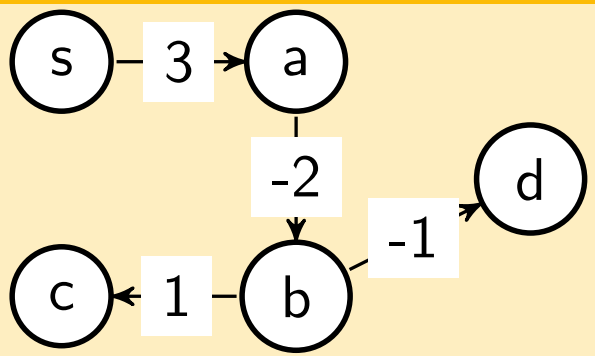
Ici : $\pi(s) = NIL, \pi(a) = s, \pi(b) = a, \pi(c) = b, \pi(d) = b$.

Algorithme de Bellman

Exemple - Arborescence des plus courts chemins



Graphe G



Arborescence

Algorithme de Bellman

Complexité

La complexité de l'algorithme de Bellman est en $O(n+m)$.

Validité

On a bien $d(s) = \delta(s)$
Si $v_1 = s, v_2, \dots v_k$ est l'ordre obtenu par le tri topologique, il est clair que si pour tout $j < i$, $d(v_j) = \delta(v_j)$, alors $d(v_i) = \delta(v_i)$.
Donc par récurrence, on a bien $\forall i, 1 \leq i \leq k, d(v_i) = \delta(v_i)$.

Algorithme de Ford

Objectifs

L'algorithme de Bellman-Ford, que nous appellerons simplement Ford (pour éviter les confusions), permet de calculer les plus courts chemins entre un sommet s et tous les autres, en particulier même si le graphe possède un ou des circuits et/ou des arcs de poids négatifs.
Si le graphe possède un circuit négatif, l'algorithme renverra Faux, sinon Vrai.

Algorithme de Ford

Principe

Le principe de l'algorithme est basé sur la constatation suivante : s'il existe un plus court chemin entre deux sommets u et v dans un graphe à n sommets, il existe un plus court chemin de longueur au plus $n-1$.
On va donc calculer les plus courts chemins depuis un sommet s ayant successivement des longueurs (au moins) de 1 à $n-1$. Puis on va calculer les plus courts chemins depuis un sommet s ayant une longueur (au moins) n. Si la longueur des plus courts chemins de longueur (au moins) n diminue par rapport à ceux de longueur (au moins) $n-1$, alors on pourra en déduire que le graphe possède un circuit négatif.

Algorithme de Ford

Principe

A chaque itération de 1 à n, on effectue un relâchement sur chaque arc uv pour voir si il est opportun d'utiliser cet arc pour avoir un plus court chemin entre s et v.

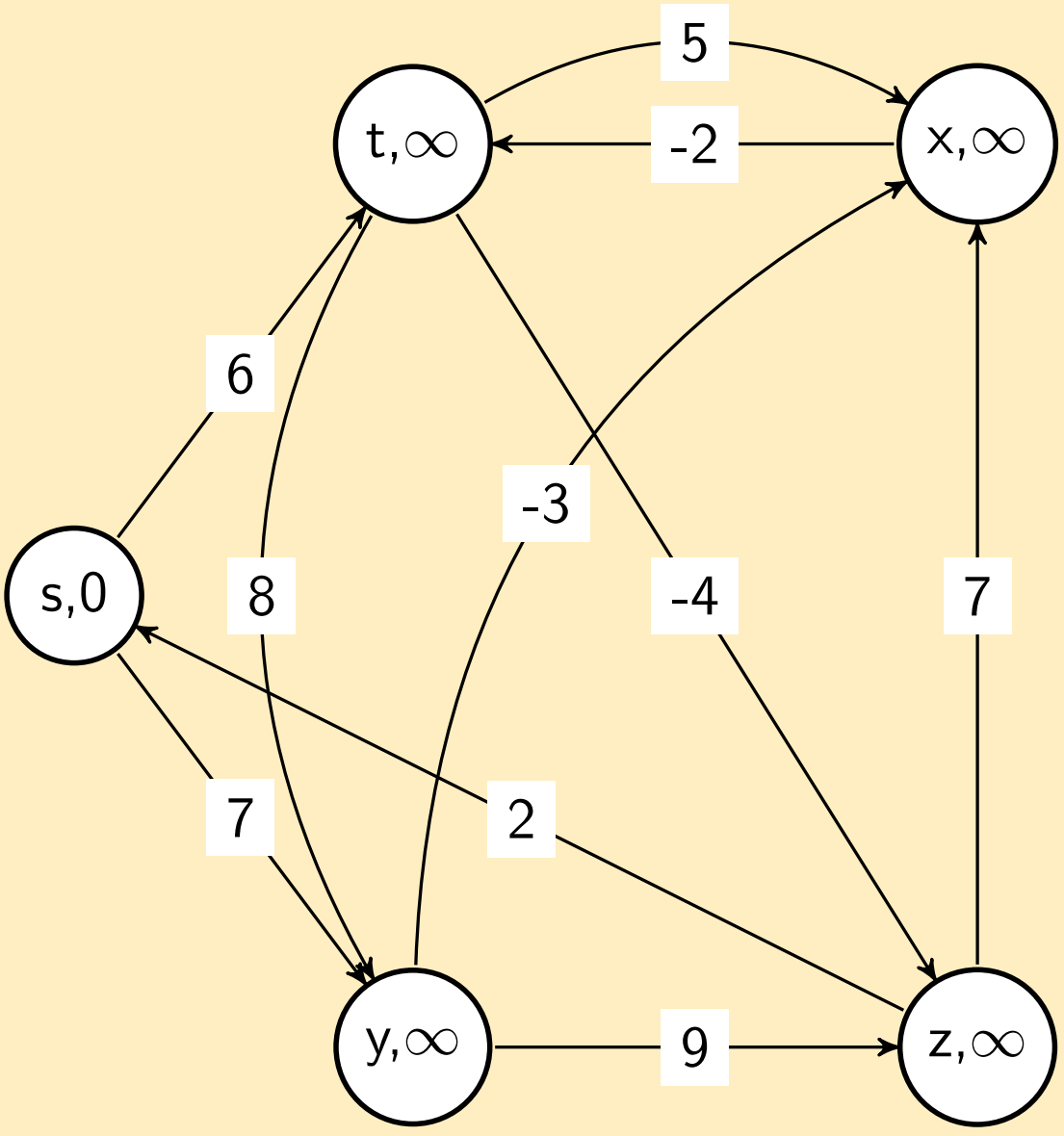
Algorithme de Ford

G : graphe orienté,
$w : E(G) \to \mathbb{R}^+$,
$s \in V(G)$
L'algorithme renvoie vrai si le graphe G est sans circuit négatif.

1: **pour tout** v de $V(G)$ **faire**
2: $\quad d[v] \leftarrow \infty$
3: $\quad pere[v] \leftarrow NIL$
4: **fin pour**
5: $d[s] \leftarrow 0$
6: **pour** i de 1 à $n-1$ **faire**
7: $\quad$ **pour** tout arc $e = (u, v) \in E(G)$ **faire**
8: $\quad\quad$ **si** $d[v] > d[u] + w(u, v)$ **alors**
9: $\quad\quad\quad d[v] \leftarrow d[u] + w[u, v]$
10: $\quad\quad\quad pere[v] \leftarrow u$
11: $\quad\quad$ **fin si**
12: $\quad$ **fin pour**
13: **fin pour**
14: **pour** tout arc $e = (u, v) \in E(G)$ **faire**
15: $\quad$ **si** $d[v] > d[u] + w(u, v)$ **alors**
16: $\quad\quad$ **retourner** FAUX
17: $\quad$ **fin si**
18: **fin pour**
19: **retourner** VRAI

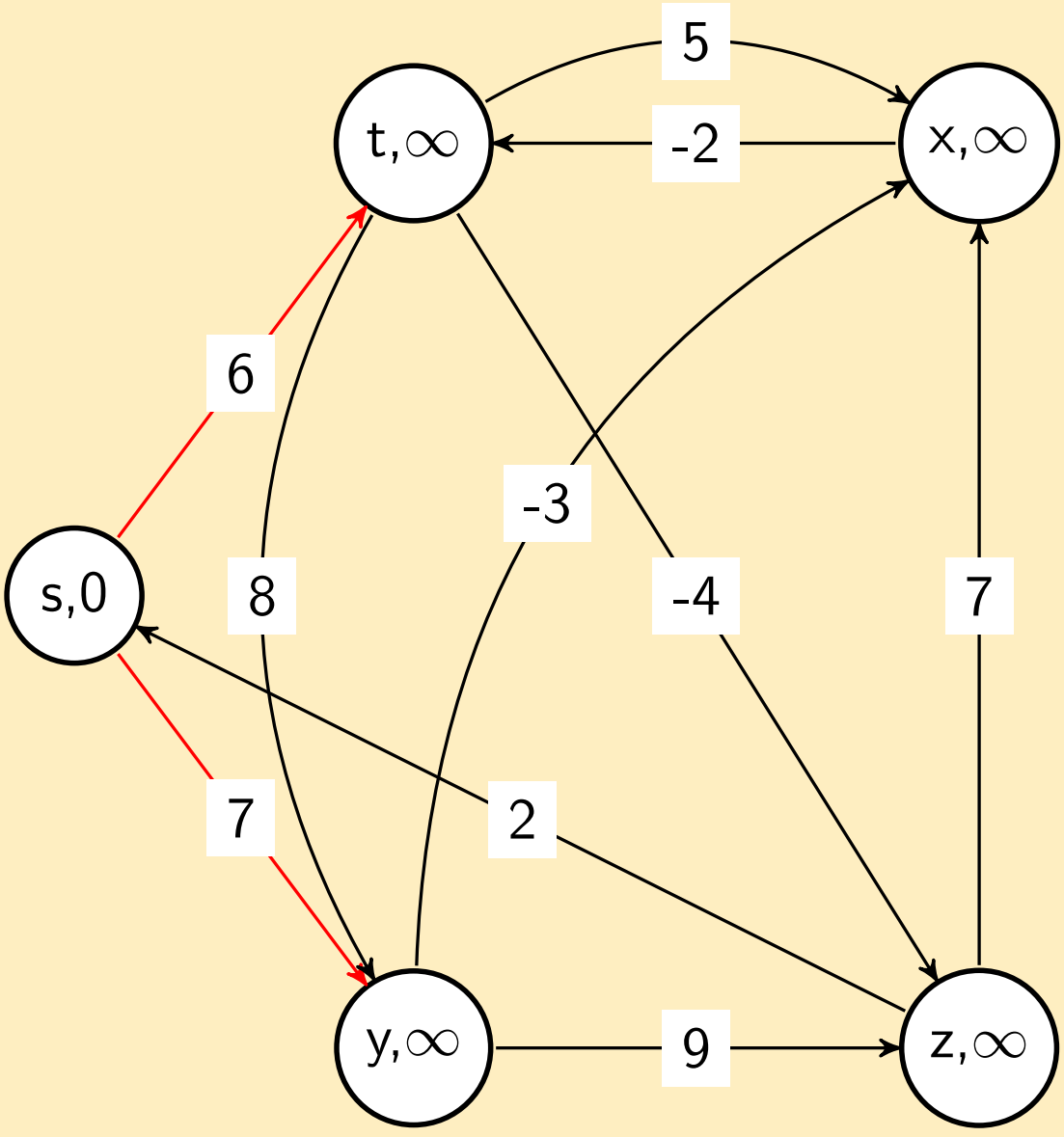
Algorithme de Ford

Exemple



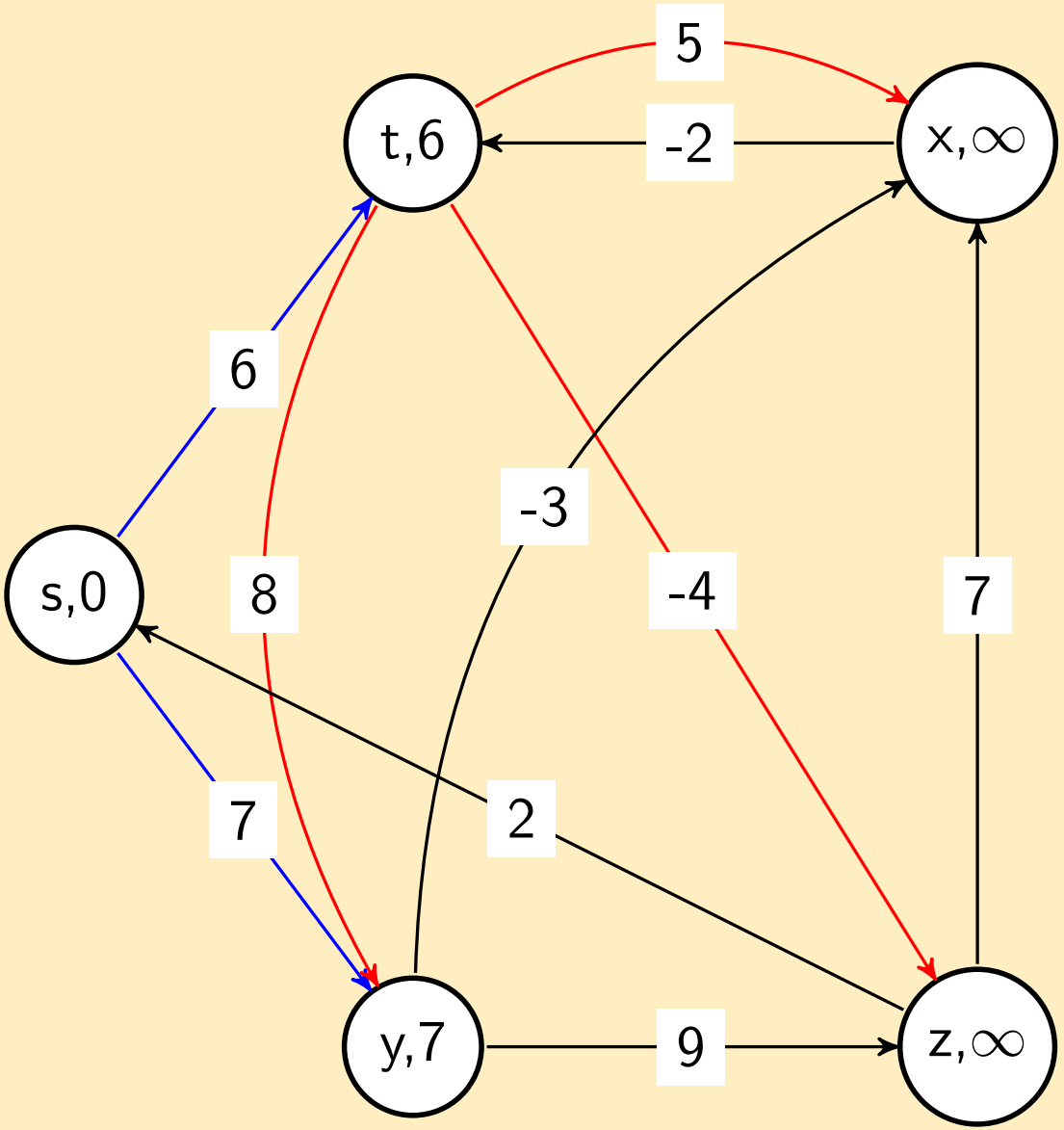
Algorithme de Ford

Exemple : $i = 1$, arcs sortants de s



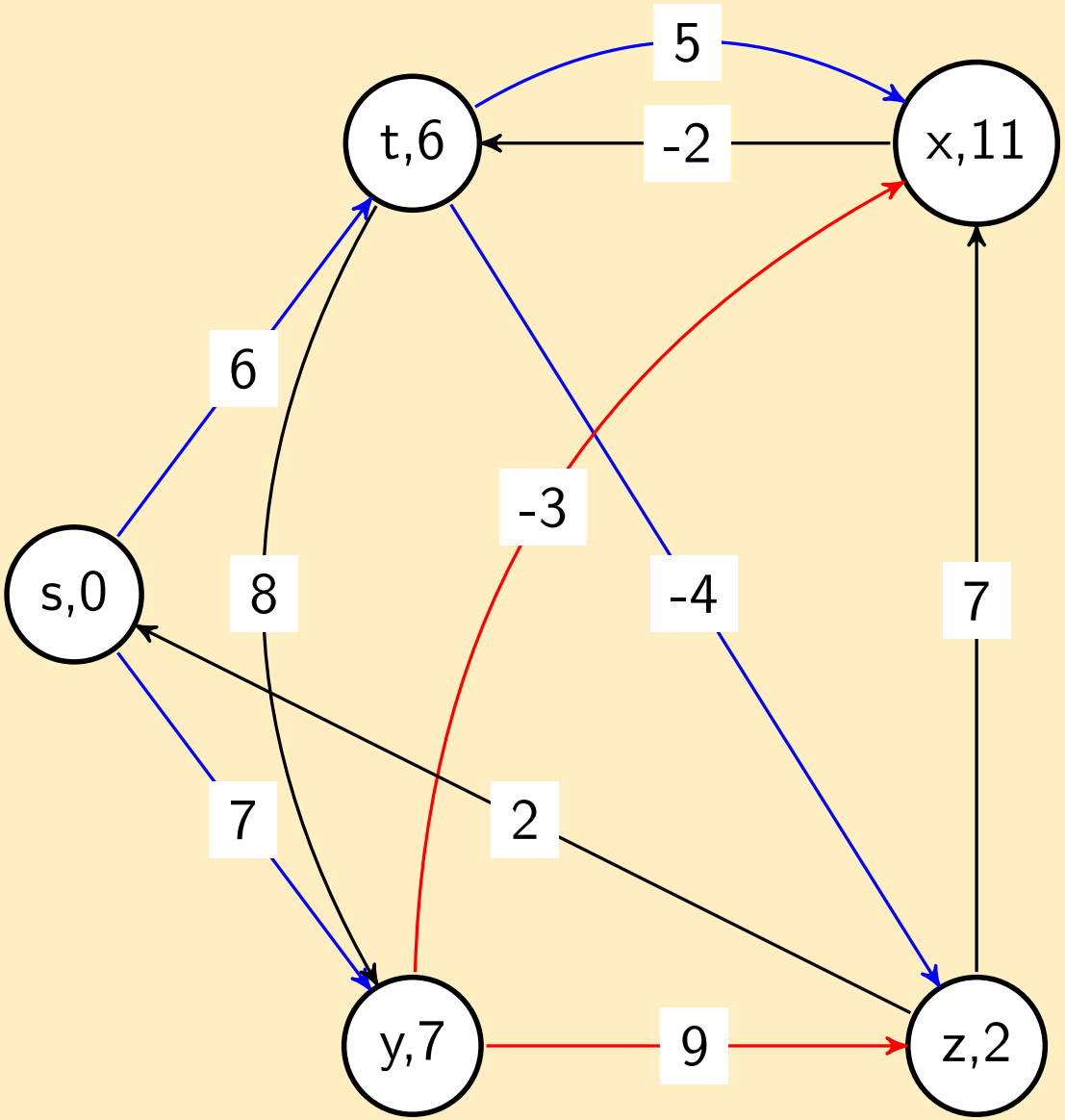
Algorithme de Ford

Exemple : $i = 1$, arcs sortants de t



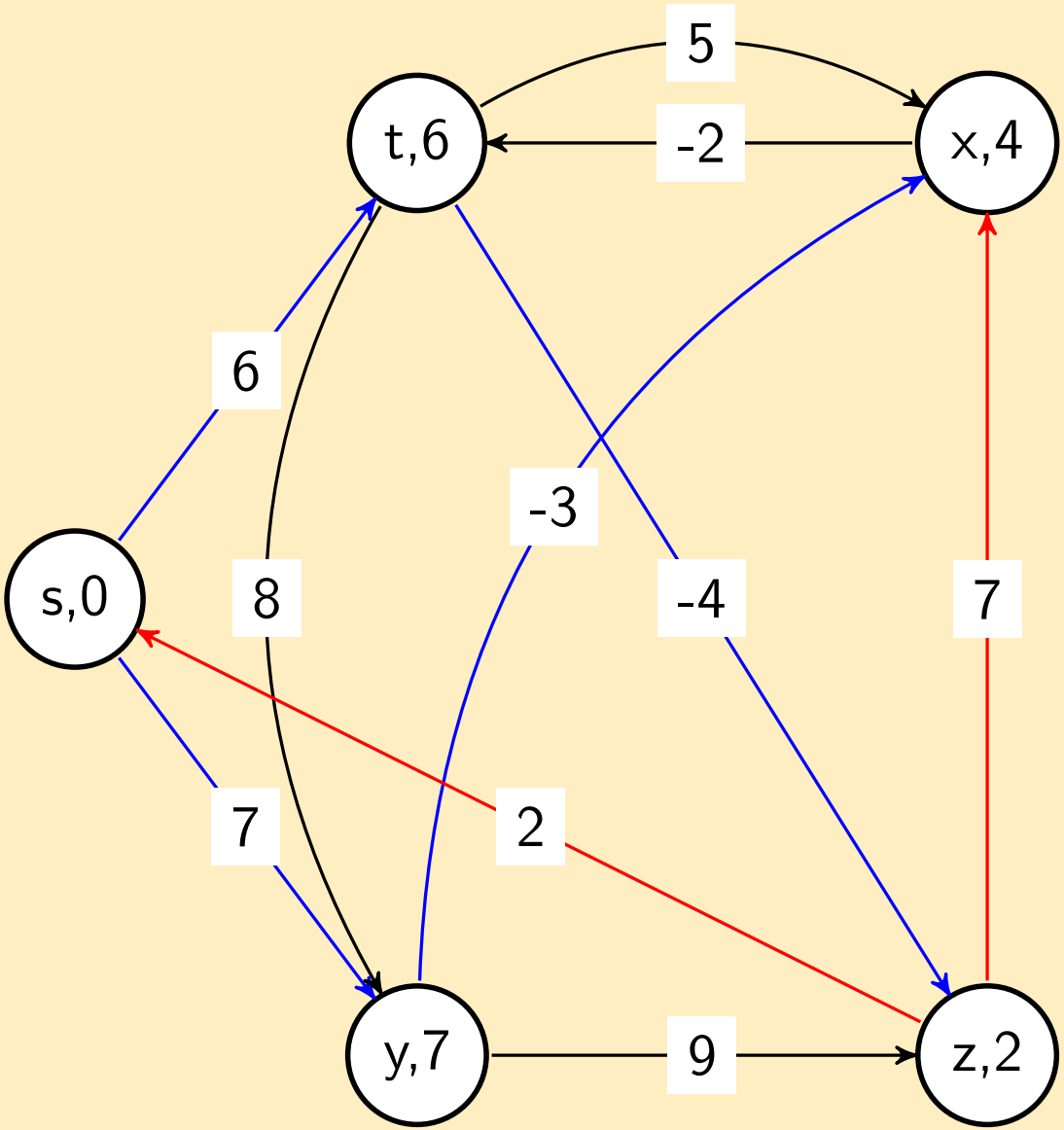
Algorithme de Ford

Exemple : $i = 1$, arcs sortants de y



Algorithme de Ford

Exemple : $i = 1$, arcs sortants de z



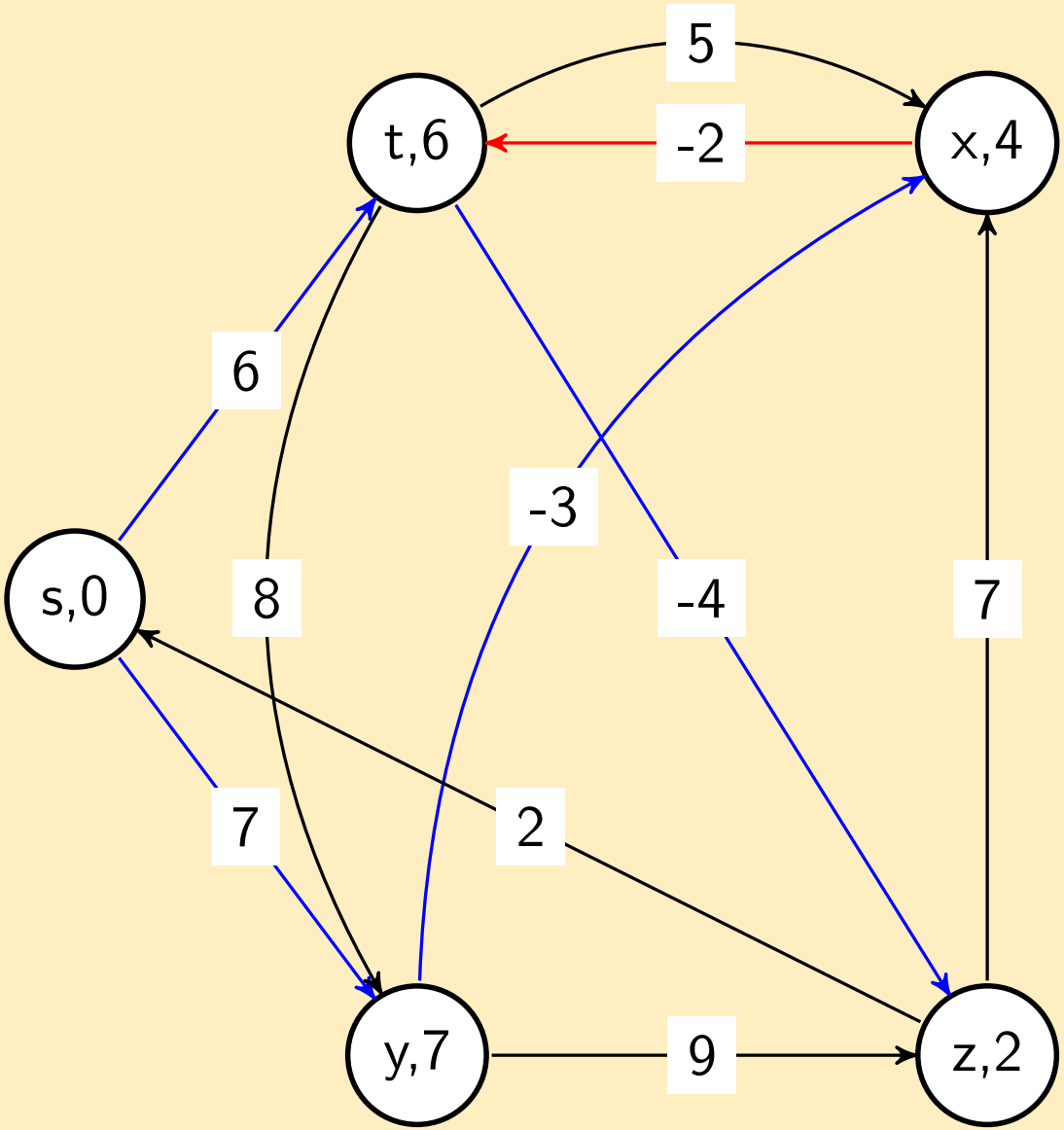
Algorithme de Ford

Exemple : $i = 2$

Remarque : pour s, t, y et z, leur distance à s n'a pas été modifiée après avoir traité leurs arcs sortants avec $i = 1$.

Algorithme de Ford

Exemple : $i = 2$, arcs sortants de x



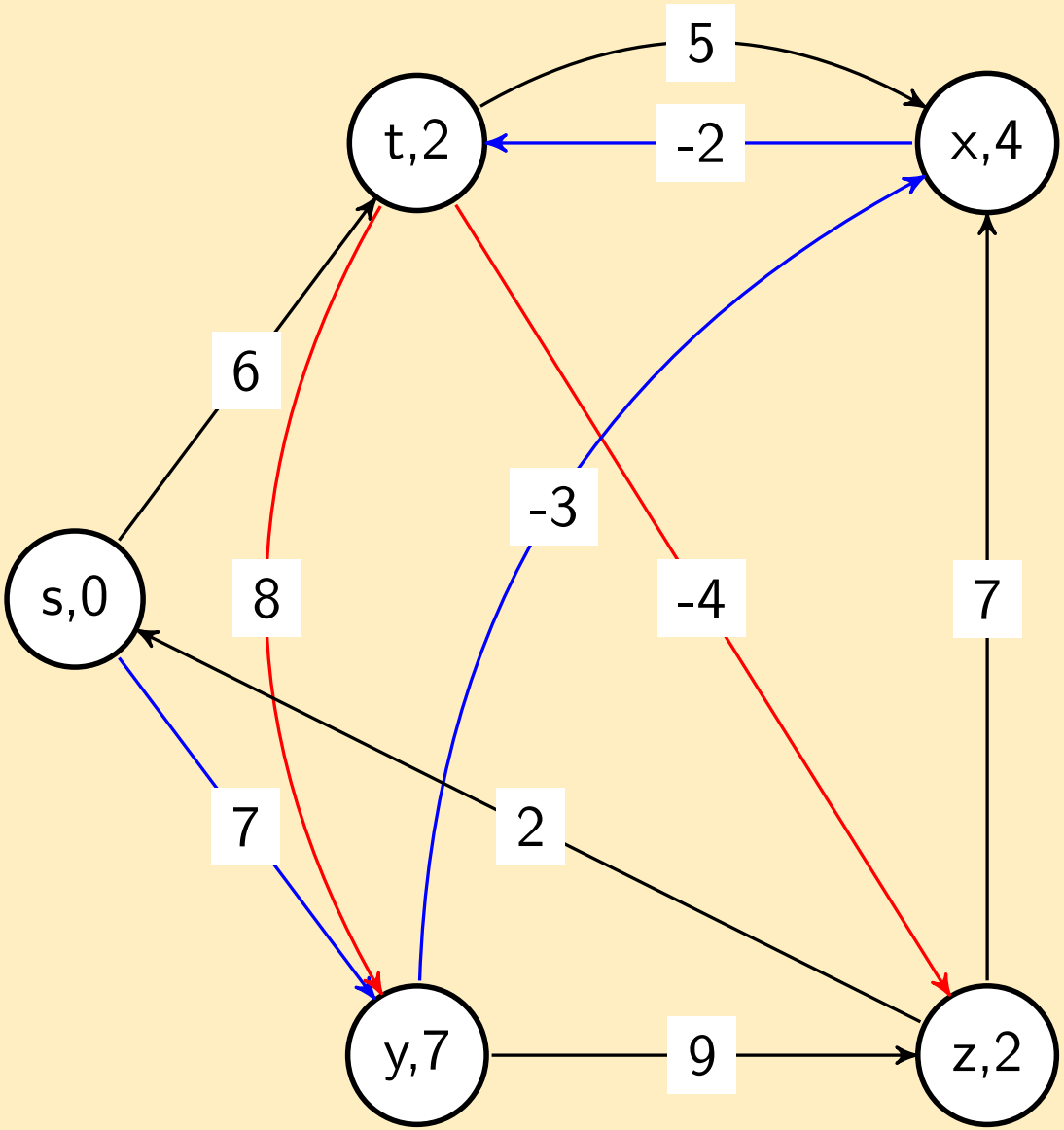
Algorithme de Ford

Exemple : $i = 3$

Remarque : pour s, x, y et z, leur distance à s n'a pas été modifiée après avoir traité leurs arcs sortants avec $i = 2$.

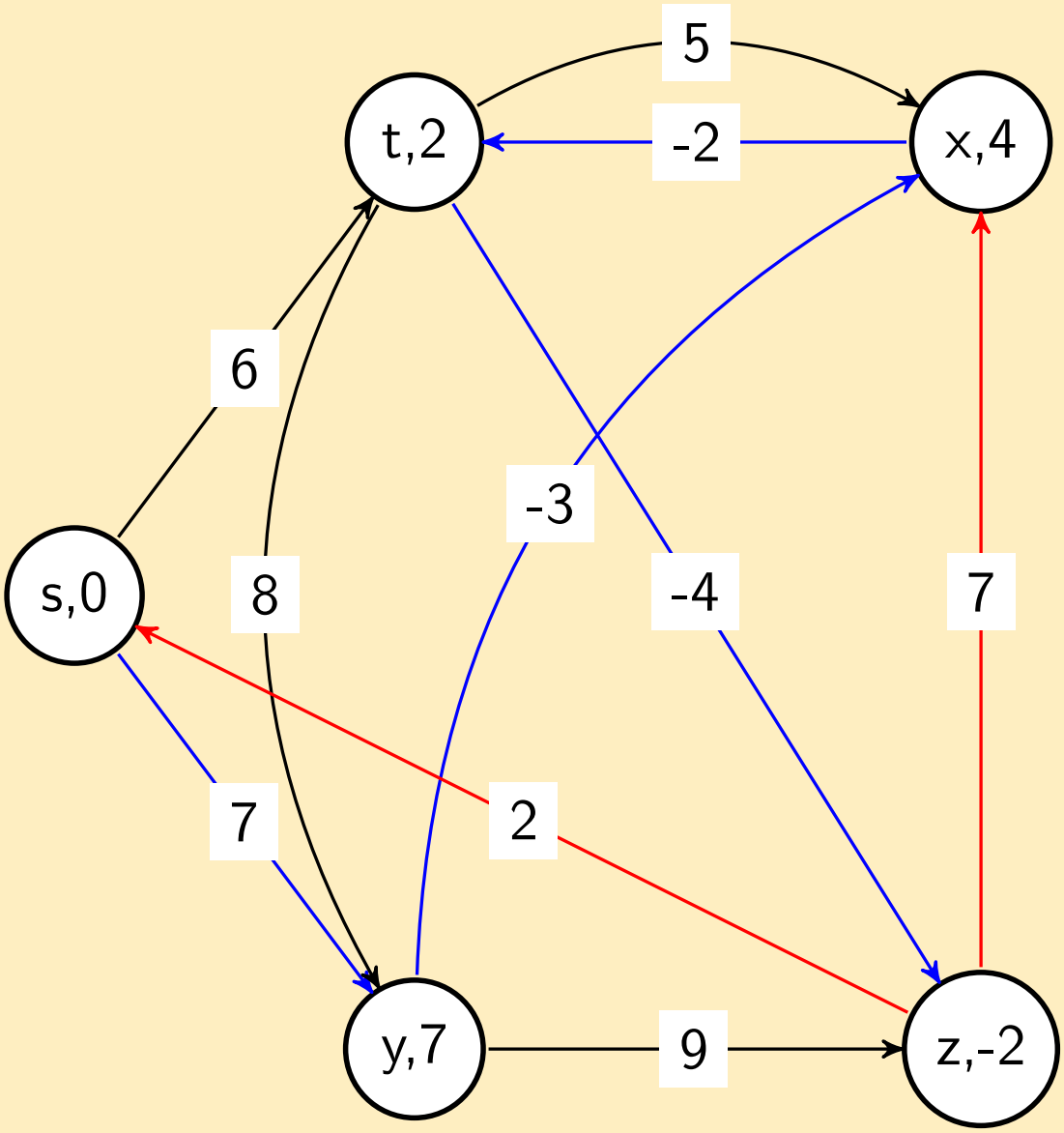
Algorithme de Ford

Exemple : $i = 3$, arcs sortants de t



Algorithme de Ford

Exemple : $i = 3$, arcs sortants de z



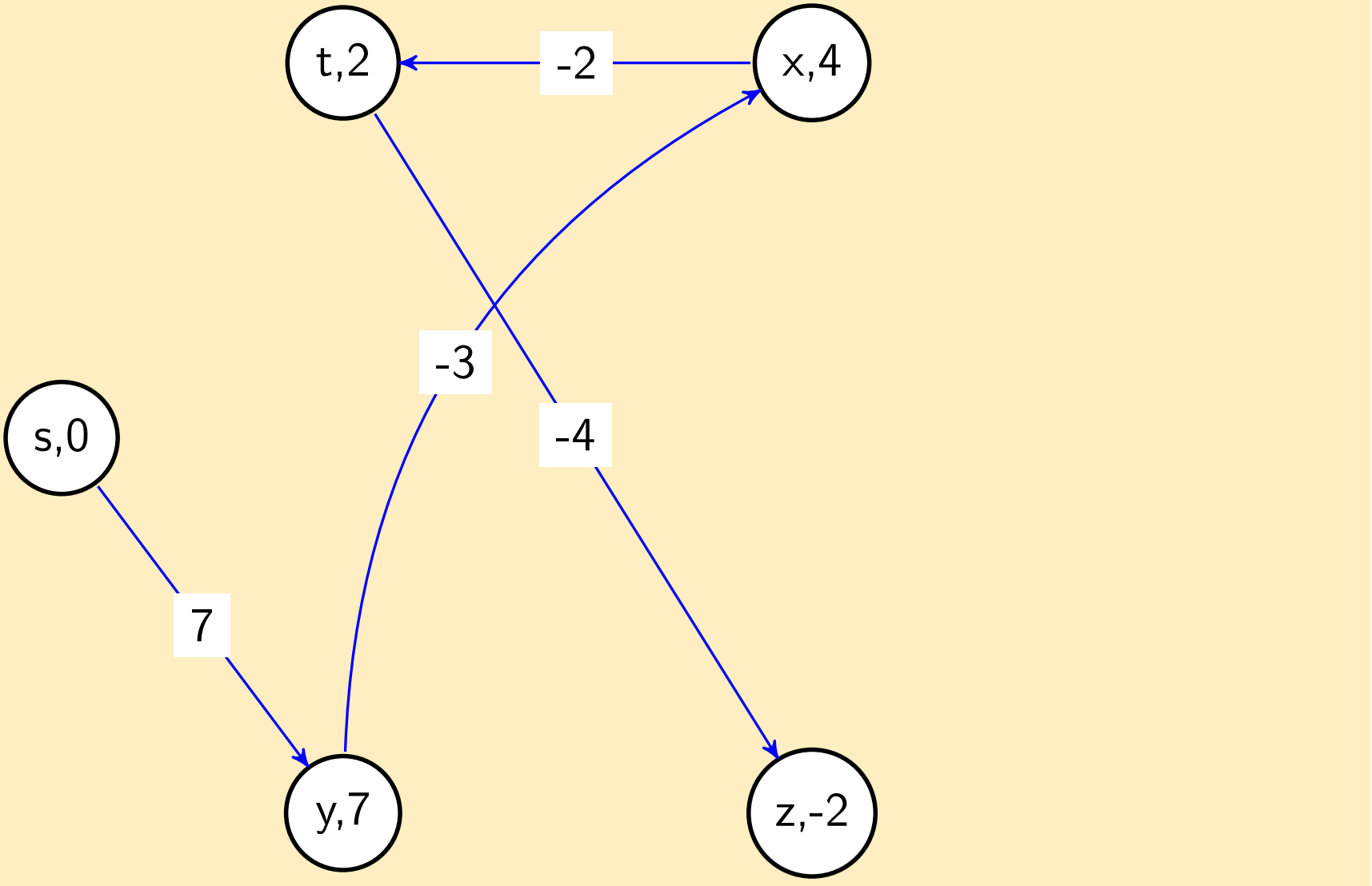
Algorithme de Ford

Exemple : $i = 4$

Remarque : Aucun sommet n'a eu sa distance à s modifiée après avoir traité ses arcs sortants avec $i = 3$. L'algorithme va donc terminer en renvoyant Vrai.

Algorithme de Ford

Arborescence



Algorithme de Ford

Complexité

L'algorithme de Ford est en $O(n \times m)$.

Plus court chemin entre toutes paires de sommets

Utilisation de Dikstra

Si le graphe n'a que des poids positifs, on peut utiliser n fois Disjkstra (1 fois à partir de chaque sommet). On obtient ainsi un algorithme en $O(n^2 \times log(n) + n \times m)$ en utilisant un tas de Fibonacci pour coder la file de priorité.

Plus court chemin entre toutes paires de sommets

Calcul matriciel

On considère une matrice W avec $W[i,j] =$

- 0 si $i = j$
- $\omega(i,j)$ si $ij \in E(G)$
- ∞ sinon.

Plus court chemin entre toutes paires de sommets

Calcul matriciel

On calcule ensuite une suite de matrice $D^{(1)} \dots D^{(n-1)}$. On pose $D^{(1)} = W$ La valeur $D_{ij}^{(k)}$ contiendra la longueur d'un plus court chemin de i à j de longueur au moins k. $D^{(k+1)}$ est calculée à partir de $D^{(k)}$ à l'aide de l'algorithme suivant :

1: **pour** i de 1 à n **faire**
2: **pour** j de 1 à n **faire**
3: $D_{ij}^{(k+1)} \leftarrow \infty$
4: **pour** a de 1 à n **faire**
5: $D_{ij}^{(k+1)} \leftarrow min(D_{ij}^{(k+1)}, D_{ia}^{(k)} + W_{aj})$
6: **fin pour**
7: **fin pour**
8: **fin pour**
9: **retourner** $D^{(k+1)}$

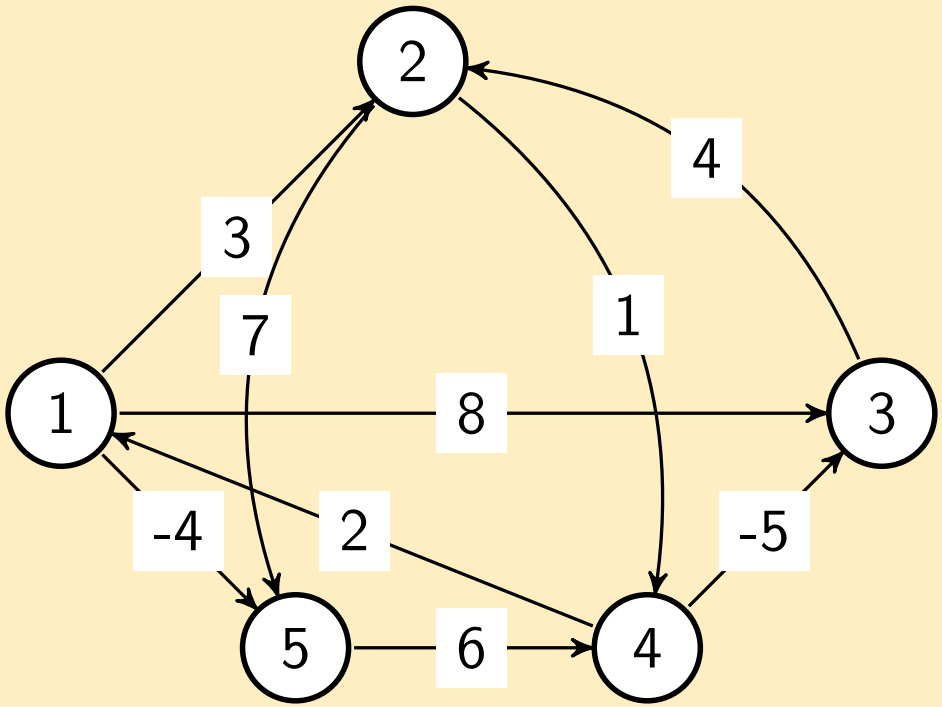
Plus court chemin entre toutes paires de sommets

Calcul matriciel

Si $D^{(n-1)}$ contient dans sa diagonale une valeur négative, disons en $D_{ii}^{(n-1)}$, alors le graphe possède un circuit négatif passant par le sommet d'indice i.

Plus court chemin entre toutes paires de sommets

Calcul matriciel - Exemple



Graphe G

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Matrice W

Calcul matriciel - Exemple

$$D^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

Plus court chemin entre toutes paires de sommets

Calcul matriciel - Exemple

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{pmatrix}
0 & 3 & -3 & 2 & -4 \\
3 & 0 & -4 & 1 & -1 \\
7 & 4 & 0 & 5 & 11 \\
2 & -1 & -5 & 0 & -2 \\
8 & 5 & 1 & 6 & 0
\end{pmatrix}
$$

$D^{(3)}$

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{pmatrix}
0 & 1 & -3 & 2 & -4 \\
3 & 0 & -4 & 1 & -1 \\
7 & 4 & 0 & 5 & 3 \\
2 & -1 & -5 & 0 & -2 \\
8 & 5 & 1 & 6 & 0
\end{pmatrix}
$$

$D^{(4)}$

Plus court chemin entre toutes paires de sommets

Calcul matriciel - accélération

On peut diminuer le nombre de calculs en calculant directement, à partir de $D^{(k)}$ la matrice $D^{(2k)}$ en remplaçant W par $D^{(k)}$. On ne calculera ainsi que les matrices D^k avec k puissance de 2, jusqu'à avoir $k \geq n-1$.

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Principe

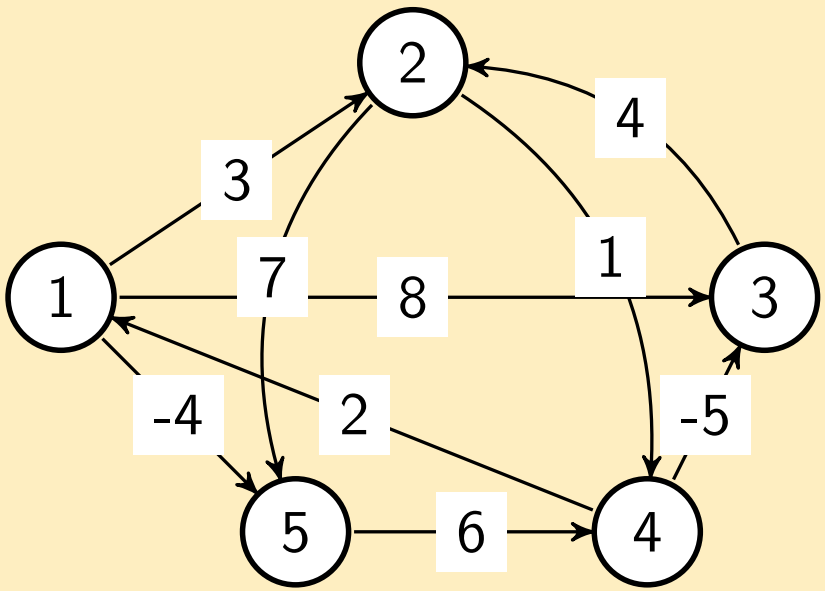
L'algorithme de Floyd (souvent appelé Floyd-Warshall) s'appuie sur le fait que l'on ne va à chaque calcul de la nouvelle matrice considérer que les k premiers sommets comme sommets intermédiaires potentiels. Comme il y a n boucles, on aura bien à la fin considéré tous les sommets comme sommet intermédiaire potentiel.
On calculera en parallèle une matrice $Pred$ où $Pred_{ij}$ sera l'indice du sommet précédent j dans le plus court chemin de i à j.

Algorithme de Floyd

G est un graphe orienté, munis d'une fonction de poids sur les arcs w.
Les sommets sont numérotés de 1 à n.
$W_{i,j}$ contiendra la plus courte distance entre le sommet i et le sommet j, $Pred_{i,j}$ le sommet prédécesseur de j sur un plus court chemin de i à j.

```
1: pour i de 1 à n faire
2:     pour j de 1 à n faire
3:         si i = j alors
4:             W(i, j) ← 0
5:             Pred(i, j) ← NIL
6:         sinon si ij ∈ E(G) alors
7:             W(i, j) ← w(i, j)
8:             Pred(i, j) ← i
9:         sinon
10:            W(i, j) ← ∞
11:            Pred(i, j) ← NIL
12:        fin si
13:    fin pour
14: fin pour
15: pour k de 1 à n faire
16:    pour i de 1 à n faire
17:        pour j de 1 à n faire
18:            si W(i, k) + W(k, j) < W(i, j) alors
19:                W(i, j) ← W(i, k) + W(k, j)
20:                Pred(i, j) ← Pred(k, j)
21:            fin si
22:        fin pour
23:    fin pour
24: fin pour
```

Algorithme de Floyd - Exemple



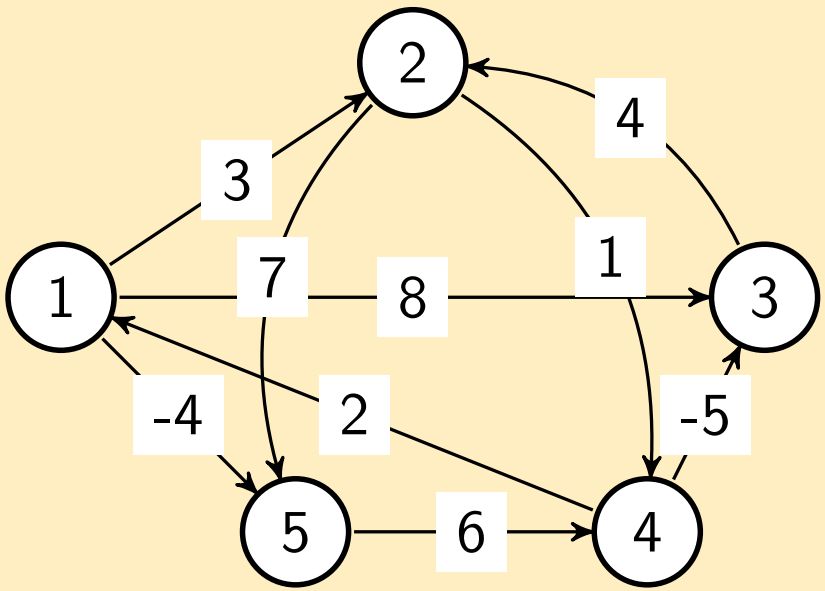
Graphe G

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Matrice W

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Exemple



Graphe G

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} NIL & 1 & 1 & NIL & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & NIL & NIL \\ 4 & NIL & 4 & NIL & NIL \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

Matrice Pred

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Exemple avec $k = 1$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Matrice W

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} NIL & 1 & 1 & NIL & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & NIL & NIL \\ 4 & 1 & 4 & NIL & 1 \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

Matrice Pred

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Exemple avec $k = 2$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Matrice W

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} NIL & 1 & 1 & 2 & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & 2 & 2 \\ 4 & 1 & 4 & NIL & 1 \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

Matrice Pred

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Exemple avec $k = 3$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Matrice W

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} NIL & 1 & 1 & 2 & 1 \\ NIL & NIL & NIL & 2 & 2 \\ NIL & 3 & NIL & 2 & 2 \\ 4 & 3 & 4 & NIL & 1 \\ NIL & NIL & NIL & 5 & NIL \end{pmatrix}$$

Matrice Pred

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Exemple avec $k = 4$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Matrice W

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} NIL & 1 & 4 & 2 & 1 \\ 4 & NIL & 4 & 2 & 1 \\ 4 & 3 & NIL & 2 & 1 \\ 4 & 3 & 4 & NIL & 1 \\ 4 & 3 & 4 & 5 & NIL \end{pmatrix}$$

Matrice Pred

Plus court chemin entre toutes paires de sommets

Algorithme de Floyd - Exemple avec $k = 5$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Matrice W

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{pmatrix} NIL & 3 & 4 & 5 & 1 \\ 4 & NIL & 4 & 2 & 1 \\ 4 & 3 & NIL & 2 & 1 \\ 4 & 3 & 4 & NIL & 1 \\ 4 & 3 & 4 & 5 & NIL \end{pmatrix}$$

Matrice Pred

Fermeture transitive - Algorithme de Warshall

Principe

L'algorithme de Floyd peut être adapté pour calculé la matrice d'adjacence de la fermeture transitive d'un graphe G orienté, c'est à dire du graphe G' tel que $ij \in E(G') \Leftrightarrow$ il existe un chemin de i à j dans G.

Fermeture transitive - Algorithme de Warshall

G est un graphe orienté dont les sommets sont numérotés de 1 à n.
$W_{i,j}$ vaudra 1 s'il existe un chemin de i à j dans le graphe G, 0 sinon.

```
1:  pour i de 1 à n faire
2:      pour j de 1 à n faire
3:          si i = j || ij ∈ E(G) alors
4:              W(i, j) ← 1
5:          sinon
6:              W(i, j) ← 0
7:          fin si
8:      fin pour
9:  fin pour
10: pour k de 1 à n faire
11:     pour i de 1 à n faire
12:         pour j de 1 à n faire
13:             W(i, j) ← (W(i, k) & W(k, j)) || W(i, j)
14:         fin pour
15:     fin pour
16: fin pour
```