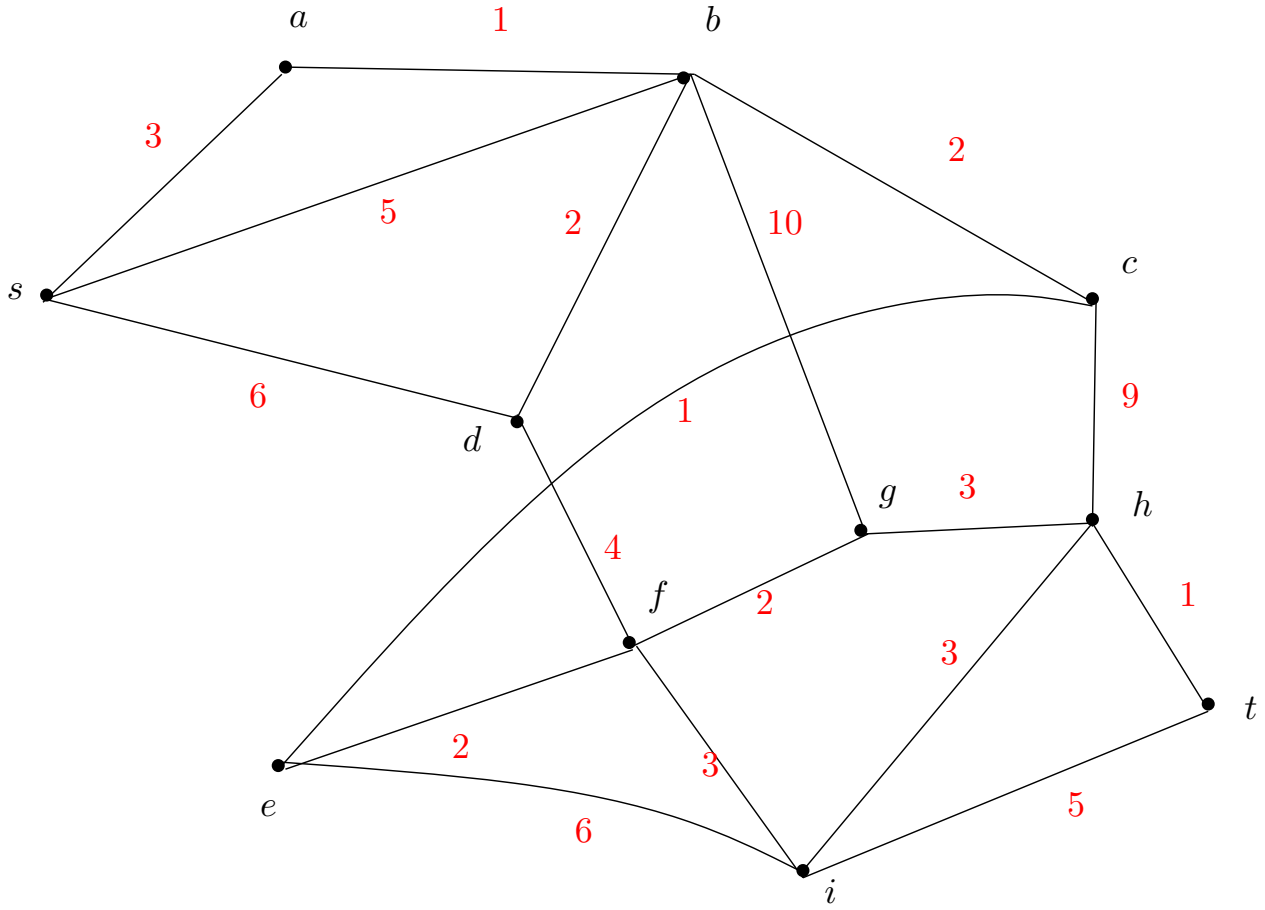
Plus courts chemins et programmation dynamique discrète

Maël Forcier

7 octobre 2020

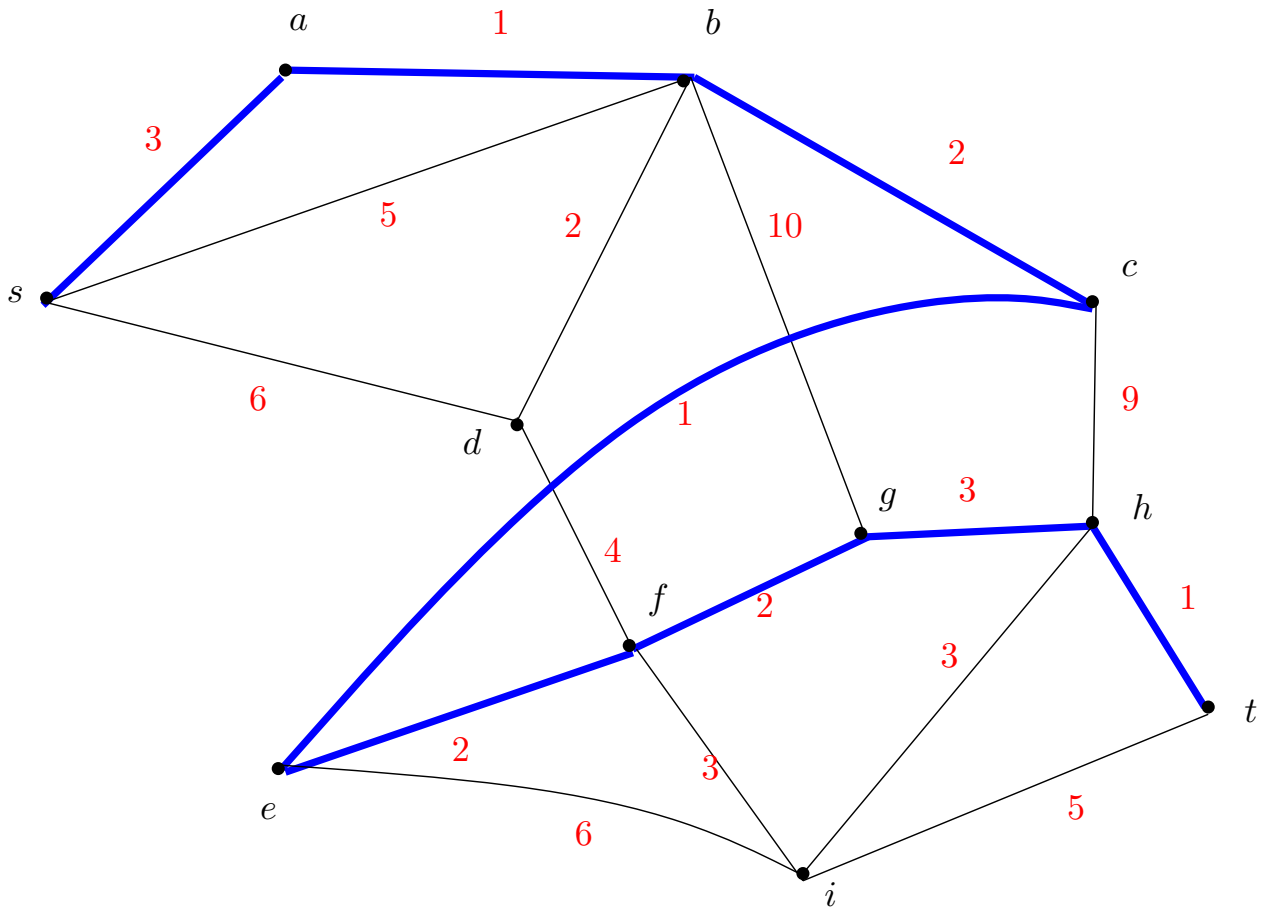
École des Ponts, France

Un exemple



Trouver le chemin le plus court dans ce réseau,
i.e. le ***s*-*t*** chemin de coût minimal dans ce graphe.

Un exemple



Trouver le chemin le plus court dans ce réseau,
i.e. le ***s*-*t*** chemin de coût minimal dans ce graphe.
Solution en bleu et coût minimal = 15

Définition

Problème du plus court chemin :

Étant donné un graphe $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ et une fonction de coût $\boldsymbol{c} : \boldsymbol{E} \to \mathbb{Q}$,
Trouver un $\boldsymbol{o}$-$\boldsymbol{d}$ chemin $\boldsymbol{P}$ de coût minimal $\sum_{\boldsymbol{e} \in \boldsymbol{P}} \boldsymbol{c}(\boldsymbol{e})$

Rappel :

- un chemin simple visite chaque arc/arête au plus une fois
- un chemin élémentaire visite chaque sommets au plus une fois

Définition

Problème du plus court chemin :

Étant donné un graphe $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ et une fonction de coût $\boldsymbol{c} : \boldsymbol{E} \to \mathbb{Q}$,
Trouver un $\boldsymbol{o}$-$\boldsymbol{d}$ chemin simple $\boldsymbol{P}$ de coût minimal $\sum_{\boldsymbol{e} \in \boldsymbol{P}} \boldsymbol{c}(\boldsymbol{e})$

Rappel :

- un chemin simple visite chaque arc/arête au plus une fois
- un chemin élémentaire visite chaque sommets au plus une fois

Définition

Problème du plus court chemin :

Étant donné un graphe $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ et une fonction de coût $\boldsymbol{c} : \boldsymbol{E} \to \mathbb{Q}$,
Trouver un $\boldsymbol{o}$-$\boldsymbol{d}$ chemin élémentaire $\boldsymbol{P}$ de coût minimal $\sum_{\boldsymbol{e} \in \boldsymbol{P}} \boldsymbol{c}(\boldsymbol{e})$

Rappel :

- un chemin simple visite chaque arc/arête au plus une fois
- un chemin élémentaire visite chaque sommets au plus une fois

Cas général : Difficile !

Théorème

Trouver le poids minimal d'un o-d *chemin élémentaire lorsque le graphe* $G = (V, E)$ *a des poids quelconques est* **NP***-difficile.*

Preuve : Réduire au problème du plus court chemin élementaire.

Réduire au problème de chemin hamiltonien :

Prendre $c(e) = -1$ pour tout $e \in E$

Il existe un chemin hamiltonien
ssi le coût minimal est égal à $-(\text{card}(V) - 1)$

Cas général : Difficile !

Théorème

Trouver le poids minimal d'un o-d *chemin élémentaire lorsque le graphe* $G = (V, E)$ *a des poids quelconques est* **NP***-difficile.*

Preuve : Réduire au problème du plus court chemin élementaire.

Réduire au problème de chemin hamiltonien :

Prendre $c(e) = -1$ pour tout $e \in E$

Il existe un chemin hamiltonien
ssi le coût minimal est égal à $-(\text{card}(V) - 1)$

Cas général : Difficile !

Théorème

Trouver le poids minimal d'un o-d *chemin élémentaire lorsque le graphe* $G = (V, E)$ *a des poids quelconques est* **NP***-difficile.*

Preuve : Réduire au problème du plus court chemin élementaire.

Réduire au problème de chemin hamiltonien :

Prendre $c(e) = -1$ pour tout $e \in E$

Il existe un chemin hamiltonien
ssi le coût minimal est égal à $-(\text{card}(V) - 1)$

Cas général : Difficile !

Théorème

Trouver le poids minimal d'un o-d *chemin élémentaire lorsque le graphe* $G = (V, E)$ *a des poids quelconques est* **NP***-difficile.*

Preuve : Réduire au problème du plus court chemin élementaire.

Réduire au problème de chemin hamiltonien :

Prendre $c(e) = -1$ pour tout $e \in E$

Il existe un chemin hamiltonien
ssi le coût minimal est égal à $-(\text{card}(V) - 1)$

Cas général : Difficile !

Théorème

Trouver le poids minimal d'un o-d *chemin élémentaire lorsque le graphe* $G = (V, E)$ *a des poids quelconques est* **NP***-difficile.*

Preuve : Réduire au problème du plus court chemin élementaire.

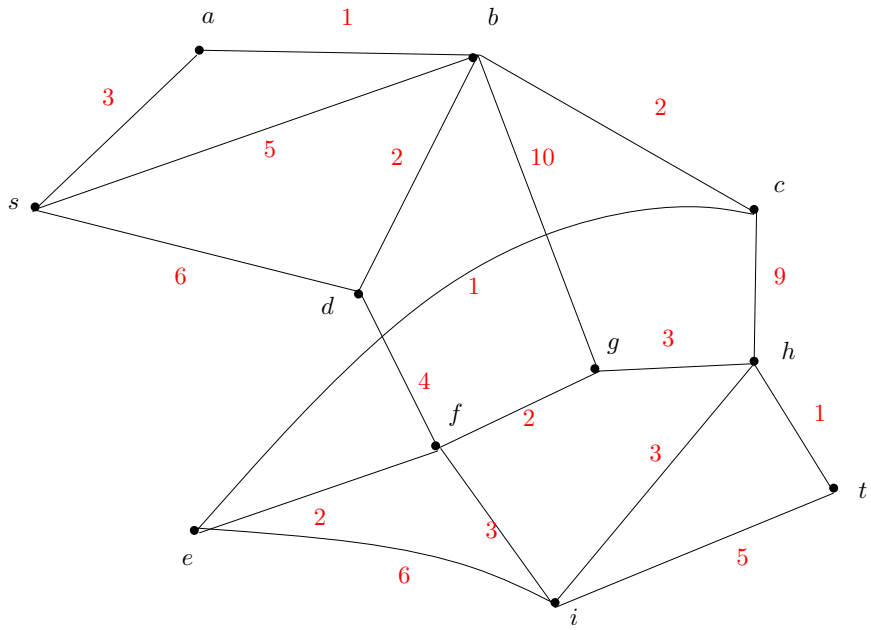
Réduire au problème de chemin hamiltonien :

Prendre $c(e) = -1$ pour tout $e \in E$

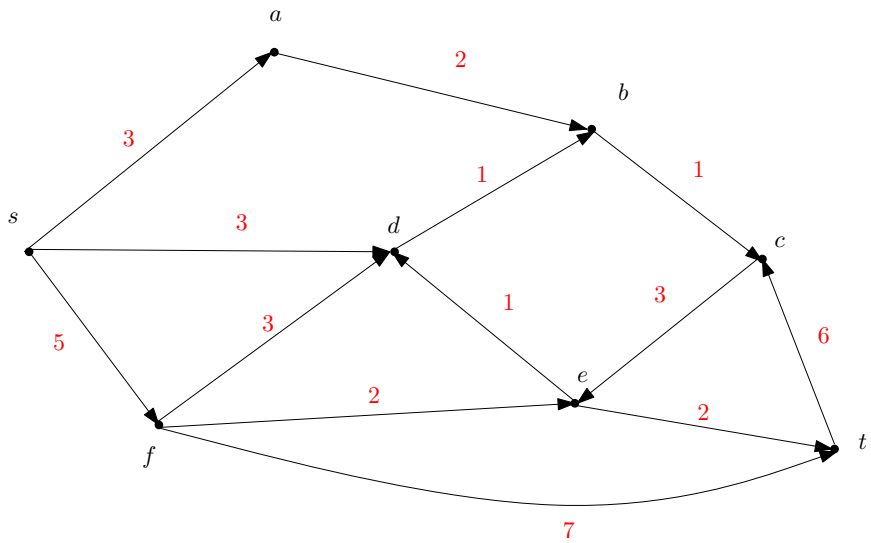
Il existe un chemin hamiltonien
ssi le coût minimal est égal à $-(\mathrm{card}(V) - 1)$

Graphe non-orienté / orienté

graphe non-orienté

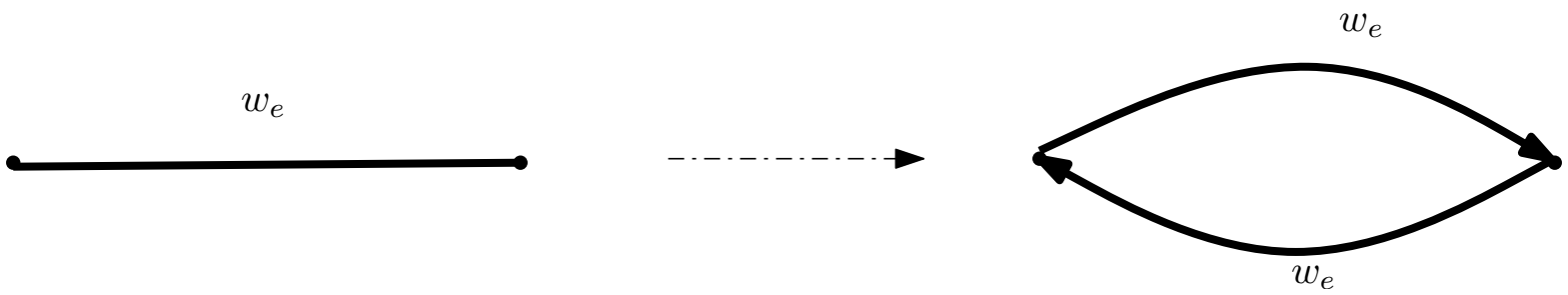


graphe orienté



Réduction d'orienté à non-orienté

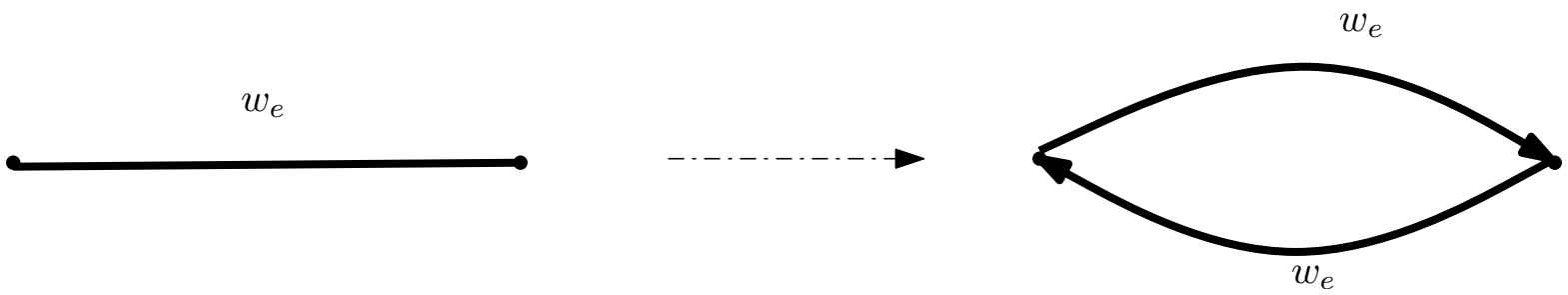
Réduction :



Peut permettre de d'utiliser des algorithmes de graphes orientés au cas non orienté

Réduction d'orienté à non-orienté

Réduction :



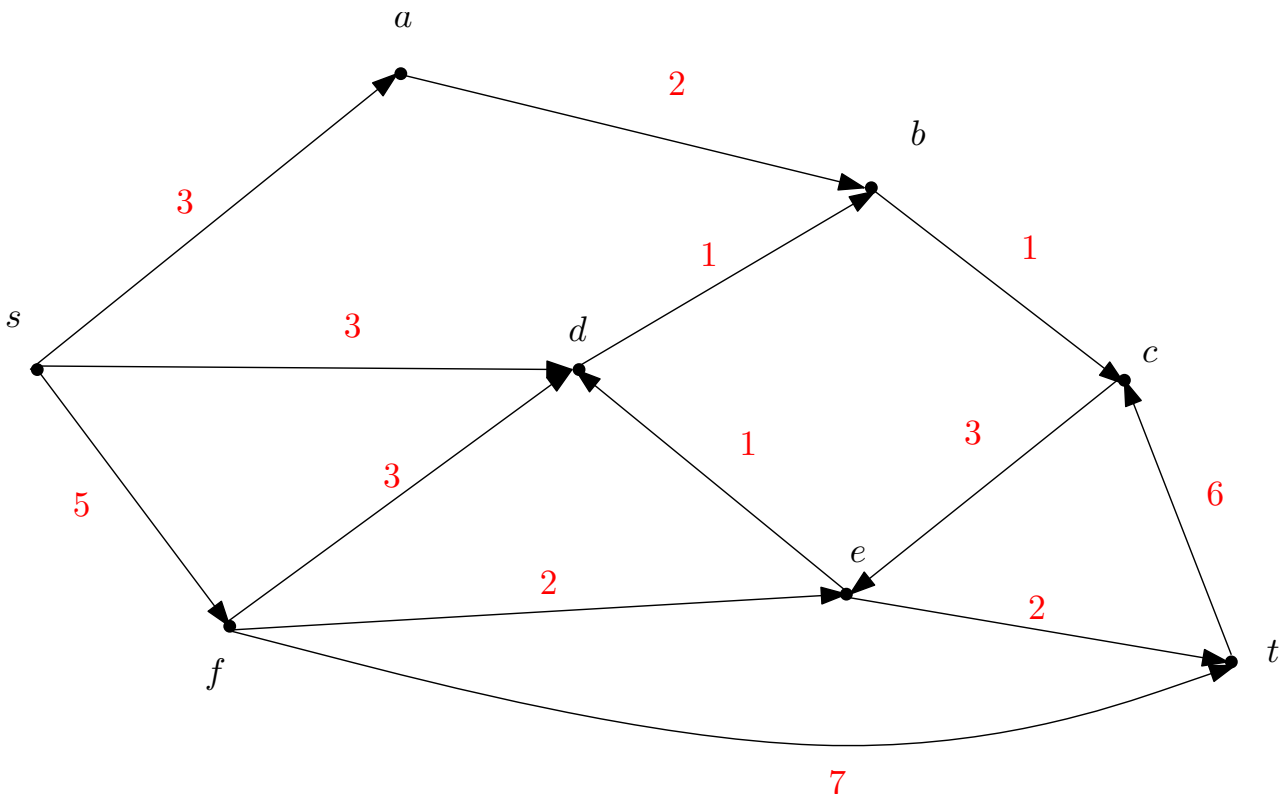
Peut permettre de d'utiliser des algorithmes de graphes orientés au cas non orienté

Plan

Table of Contents

Tous les coûts sont $\geq$ **0**

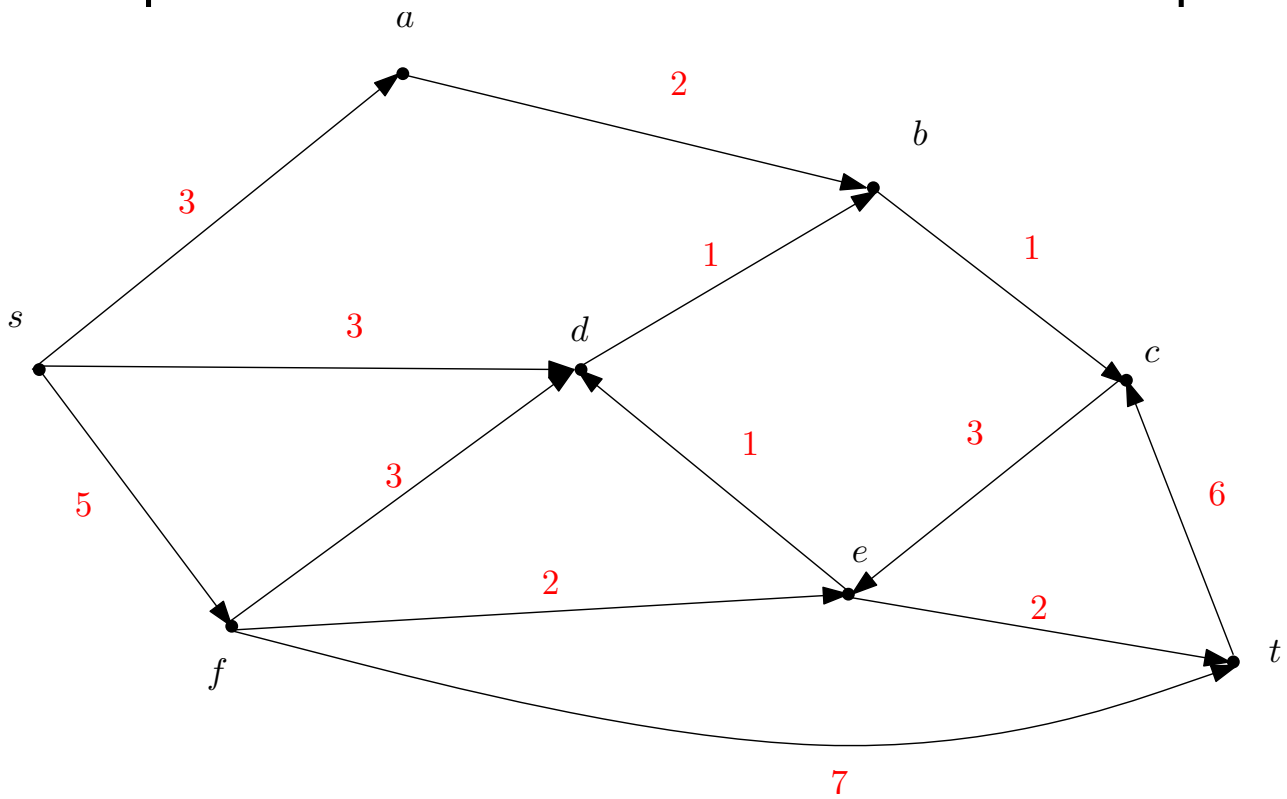
Cas le plus naturel : tous les poids sont $\geq$ **0**.



Tous les coûts sont $\geq$ **0**

Cas le plus naturel : tous les poids sont $\geq$ **0**.

Exemple : Modélisation d'un réseau de transport.

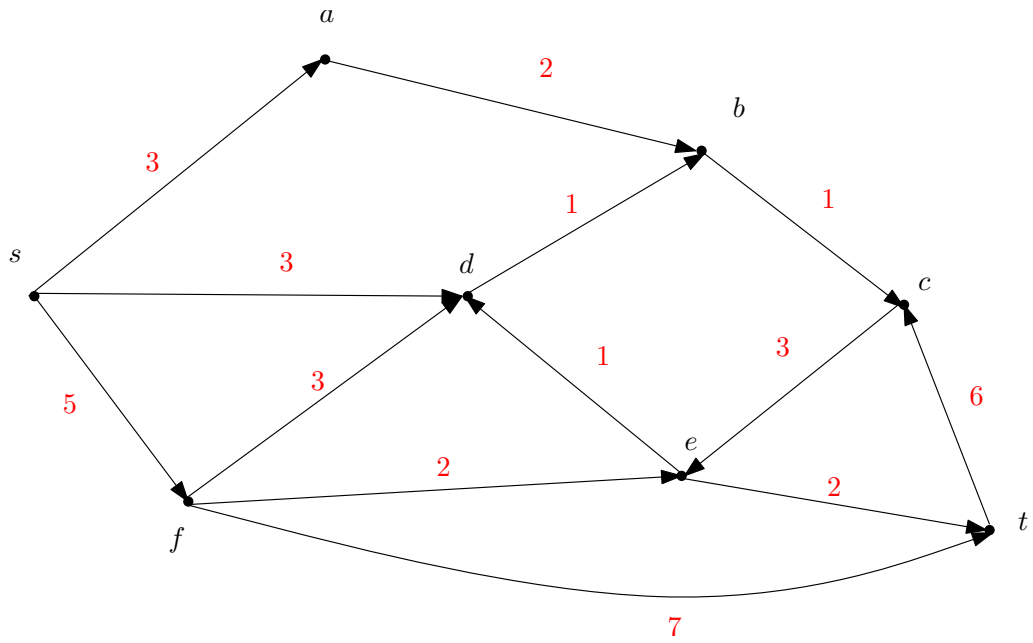


L'algorithme de Dijkstra

- Initialisation : $\boldsymbol{U} := \emptyset$, $\boldsymbol{d}(\boldsymbol{o}) := \mathbf{0}$ et $\boldsymbol{d}(\boldsymbol{v}) = +\infty$ pour tout $\boldsymbol{v} \neq \boldsymbol{o}$.

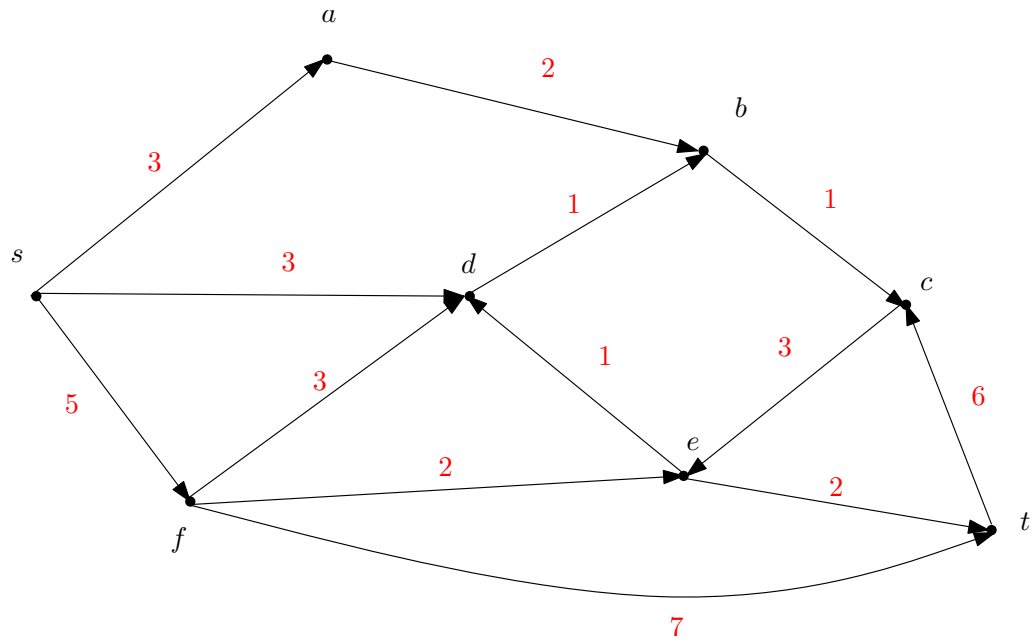
- Tant que $\boldsymbol{V} \backslash \boldsymbol{U} \neq \emptyset$
  1. Choisir $\boldsymbol{v}$ minimisant $\boldsymbol{d}(\boldsymbol{v})$ dans $\boldsymbol{V} \backslash \boldsymbol{U}$.
  2. $\boldsymbol{U} := \boldsymbol{U} \cup \{\boldsymbol{v}\}$.
  3. Pour chaque $\boldsymbol{a} = (\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{A}$
     - $\boldsymbol{d}(\boldsymbol{v}) := \min[\boldsymbol{d}(\boldsymbol{v}), \boldsymbol{d}(\boldsymbol{u}) + \boldsymbol{c}(\boldsymbol{a})]$.

Exemple : Dijkstra



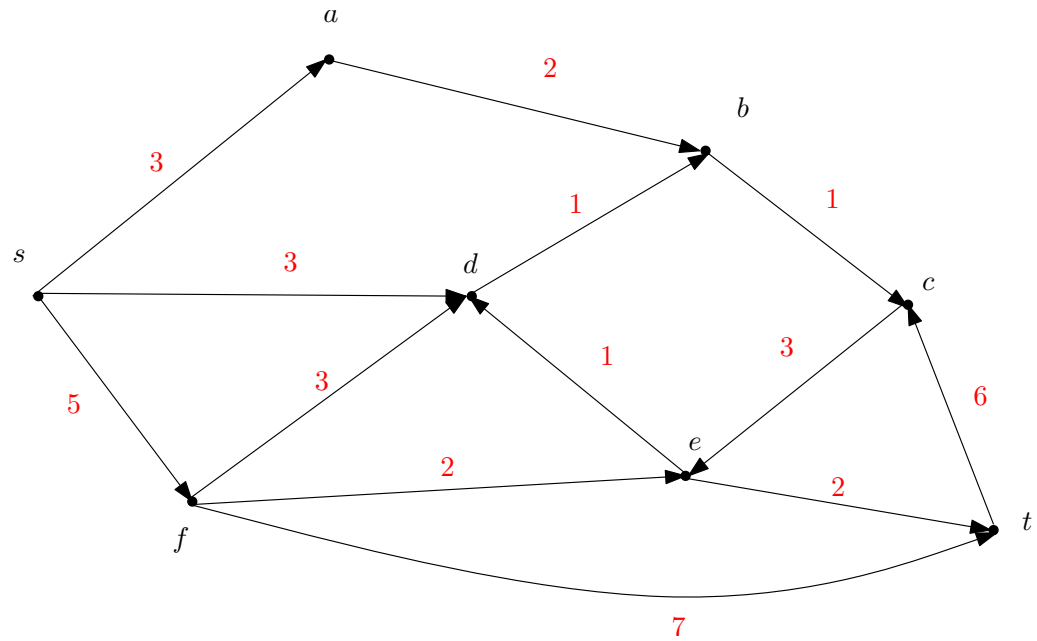
s	***a***	***b***	***c***	***d***	***e***	***f***	***t***
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)

Exemple : Dijkstra



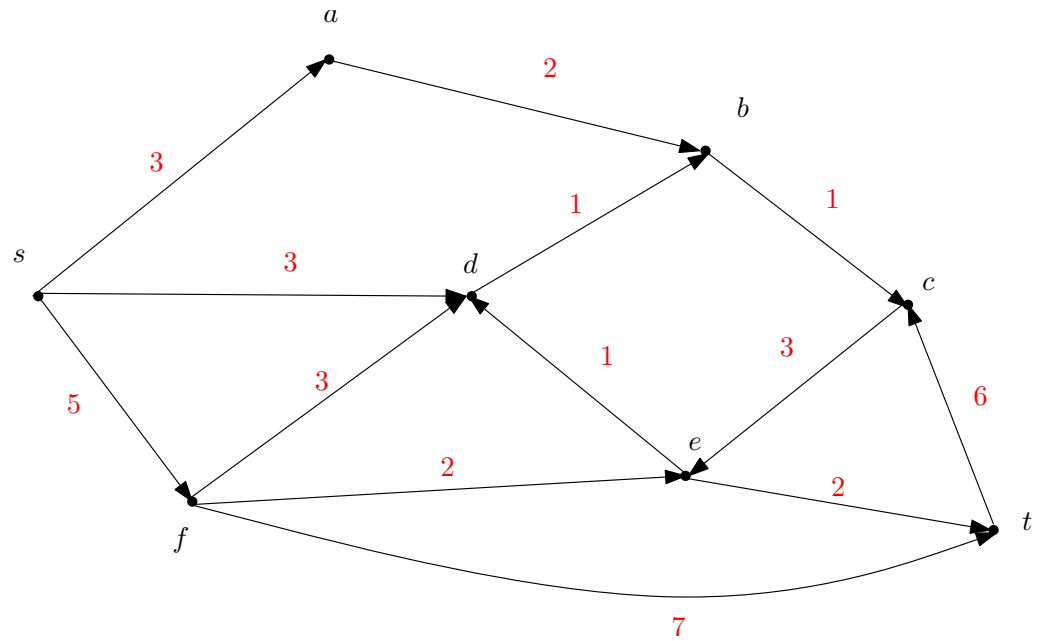
s	***a***	***b***	***c***	***d***	***e***	***f***	***t***
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	**(3)**	(∞)	(∞)	**(3)**	(∞)	**(5)**	(∞)

Exemple : Dijkstra



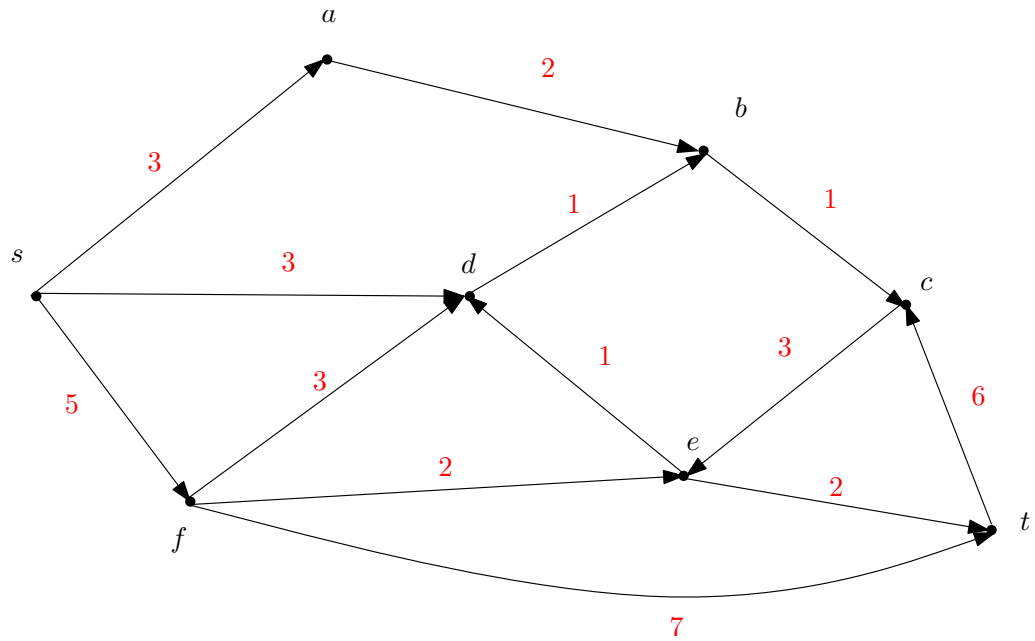
s	*a*	*b*	*c*	*d*	*e*	*f*	*t*
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	**(3)**	(∞)	(∞)	**(3)**	(∞)	**(5)**	(∞)
0	**3**	**(5)**	(∞)	**(3)**	(∞)	**(5)**	(∞)

Exemple : Dijkstra



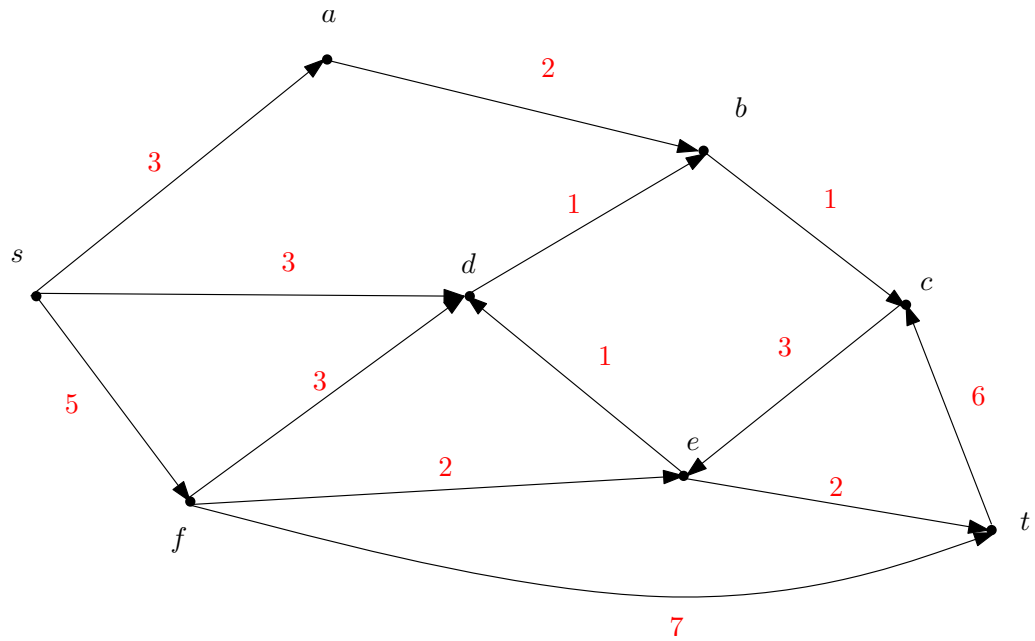
s	*a*	*b*	*c*	*d*	*e*	*f*	*t*
(**0**)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(**3**)	(∞)	(∞)	(**3**)	(∞)	(**5**)	(∞)
0	**3**	(**5**)	(∞)	(**3**)	(∞)	(**5**)	(∞)
0	**3**	(**4**)	(∞)	**3**	(∞)	(**5**)	(∞)

Exemple : Dijkstra



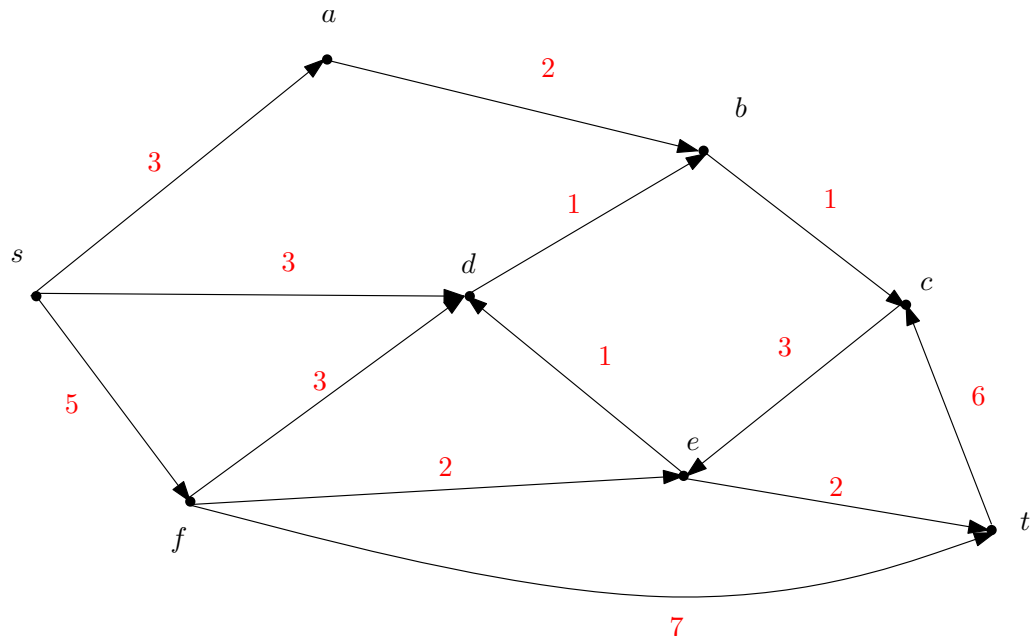
s	**a**	**b**	**c**	**d**	**e**	**f**	**t**
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	**(3)**	(∞)	(∞)	**(3)**	(∞)	**(5)**	(∞)
0	**3**	**(5)**	(∞)	**(3)**	(∞)	**(5)**	(∞)
0	**3**	**(4)**	(∞)	**3**	(∞)	**(5)**	(∞)
0	**3**	**4**	**(5)**	**3**	(∞)	**(5)**	(∞)

Exemple : Dijkstra



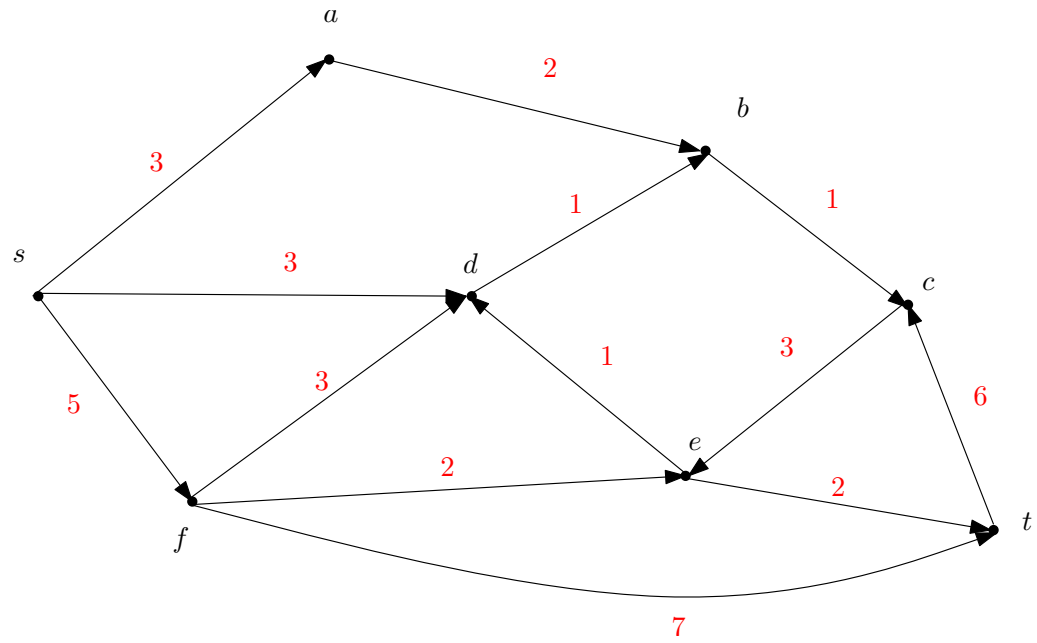
s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)
0	3	4	5	3	(8)	(5)	(∞)

Exemple : Dijkstra



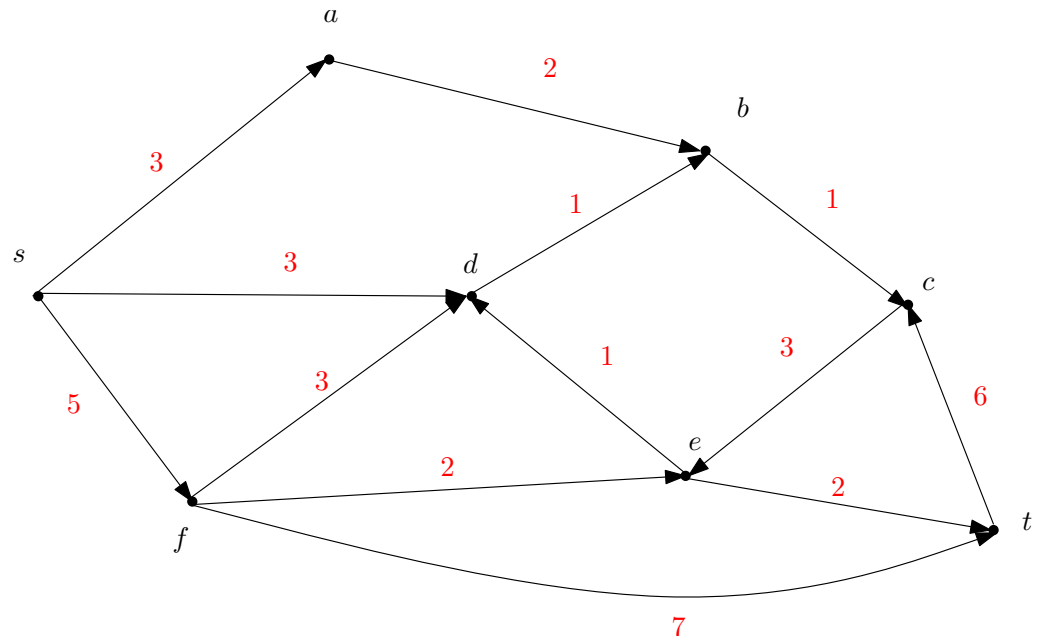
s	**a**	**b**	**c**	**d**	**e**	**f**	**t**
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	**(3)**	(∞)	(∞)	**(3)**	(∞)	**(5)**	(∞)
0	**3**	**(5)**	(∞)	**(3)**	(∞)	**(5)**	(∞)
0	**3**	**(4)**	(∞)	**3**	(∞)	**(5)**	(∞)
0	**3**	**4**	**(5)**	**3**	(∞)	**(5)**	(∞)
0	**3**	**4**	**5**	**3**	**(8)**	**(5)**	(∞)
0	**3**	**4**	**5**	**3**	**(7)**	**5**	**(12)**

Exemple : Dijkstra



s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)
0	3	4	5	3	(8)	(5)	(∞)
0	3	4	5	3	(7)	5	(12)
0	3	4	5	3	7	5	(9)

Exemple : Dijkstra



s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)
0	3	4	5	3	(8)	(5)	(∞)
0	3	4	5	3	(7)	5	(12)
0	3	4	5	3	7	5	(9)
0	3	4	5	3	7	5	9

L'algorithme de Dijkstra

Peut être facilement adapté pour calculer le chemin lui-même (et pas seulement le coût minimum)

Fonctionne sur les graphes non-orientés (cf réduction précédente)

Converge en $O(m + nlog(n))$ avec
n : nombre de sommets
m : nombres d'arcs

L'algorithme de Dijkstra

Peut être facilement adapté pour calculer le chemin lui-même (et pas seulement le coût minimum)

Fonctionne sur les graphes non-orientés (cf réduction précédente)

Converge en $O(m + n\log(n))$ avec
n : nombre de sommets
m : nombres d'arcs

L'algorithme de Dijkstra

Peut être facilement adapté pour calculer le chemin lui-même (et pas seulement le coût minimum)

Fonctionne sur les graphes non-orientés (cf réduction précédente)

Converge en $\boldsymbol{O}(\boldsymbol{m} + \boldsymbol{n}\boldsymbol{log}(\boldsymbol{n}))$ avec
$\boldsymbol{n}$: nombre de sommets
$\boldsymbol{m}$: nombres d'arcs

Table of Contents

Table of Contents

L'idée de Bellman

Principe d'optimalité de Bellman

La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.2)

Soit P un o-v chemin avec k arcs et Q son sous-o-u chemin, pù u iest le sommet avant v dans P. Si P est le plus petit $o - v$ chemin parmie ceux à k arcs, alors Q est le plus petit o-u parmi ceux à $k - 1$ arcs.

Preuve :

- Par l'absurde soit Q' o-u chemin avec $k - 1$ arcs tel que $c(Q') < c(Q)$.
- $P' = Q' \cup (u, v)$ est un o-v chemin avec k arcs.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.2)

*Soit **P** un **o**-**v** chemin avec **k** arcs et **Q** son sous-**o**-**u** chemin, pù **u** iest le sommet avant **v** dans **P**. Si **P** est le plus petit **o** − **v** chemin parmie ceux à **k** arcs, alors **Q** est le plus petit **o**-**u** parmi ceux à **k** − **1** arcs.*

Preuve :

- Par l'absurde soit Q' o-u chemin avec $k-1$ arcs tel que $c(Q') < c(Q)$.
- $P' = Q' \cup (u, v)$ est un o-v chemin avec k arcs.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.2)

Soit P un o-v chemin avec k arcs et Q son sous-o-u chemin, pù u iest le sommet avant v dans P. Si P est le plus petit $o - v$ chemin parmie ceux à k arcs, alors Q est le plus petit o-u parmi ceux à $k - 1$ arcs.

Preuve :

- Par l'absurde soit Q' o-u chemin avec $k - 1$ arcs tel que $c(Q') < c(Q)$.
- $P' = Q' \cup (u, v)$ est un o-v chemin avec k arcs.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.2)

Soit P un o-v chemin avec k arcs et Q son sous-o-u chemin, pù u iest le sommet avant v dans P. Si P est le plus petit $o - v$ chemin parmie ceux à k arcs, alors Q est le plus petit o-u parmi ceux à $k - 1$ arcs.

Preuve :

- Par l'absurde soit Q' o-u chemin avec $k - 1$ arcs tel que $c(Q') < c(Q)$.
- $P' = Q' \cup (u, v)$ est un o-v chemin avec k arcs.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.2)

Soit P un o-v chemin avec k arcs et Q son sous-o-u chemin, pù u iest le sommet avant v dans P. Si P est le plus petit $o - v$ chemin parmie ceux à k arcs, alors Q est le plus petit o-u parmi ceux à $k - 1$ arcs.

Preuve :

- Par l'absurde soit Q' o-u chemin avec $k - 1$ arcs tel que $c(Q') < c(Q)$.
- $P' = Q' \cup (u, v)$ est un o-v chemin avec k arcs.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

Algorithme de Ford-Bellman

Le coût minimal $f(v, k)$ d'un o-v chemin à k arcs vérifie l'équation de Bellman

$$f(v, k+1) = \min_{u \in N^-(v)} f(u, k-1) + c(u, v)$$

$$f(v, 0) = \begin{cases} 0 & \text{if } v = o \\ +\infty & \text{otherwise} \end{cases}$$

On peut calculer récursivement à partir de $k = 0$. Critère d'arrêt ?

Solution :
Besoin d'ajouter une hypothèse : le graphe G n'a pas de cycles négatifs.

S'il G n'a pas de cycles négatifs, il y a un plus court chemin élémentaire de longueur au plus $n - 1$.

Algorithme de Ford-Bellman

Le coût minimal $f(v, k)$ d'un o-v chemin à k arcs vérifie l'équation de Bellman

$$f(v, k+1) = \min_{u \in N^-(v)} f(u, k-1) + c(u, v)$$

$$f(v, 0) = \begin{cases} 0 & \text{if } v = o \\ +\infty & \text{otherwise} \end{cases}$$

On peut calculer récursivement à partir de $k = 0$. Critère d'arrêt ?

Solution :
Besoin d'ajouter une hypothèse : le graphe G n'a pas de cycles négatifs.

S'il G n'a pas de cycles négatifs, il y a un plus court chemin élémentaire de longueur au plus $n - 1$.

Algorithme de Ford-Bellman

Le coût minimal $f(v, k)$ d'un o-v chemin à k arcs vérifie l'équation de Bellman

$$f(v, k+1) = \min_{u \in N^-(v)} f(u, k-1) + c(u, v)$$

$$f(v, 0) = \begin{cases} 0 & \text{if } v = o \\ +\infty & \text{otherwise} \end{cases}$$

On peut calculer récursivement à partir de $k = 0$. Critère d'arrêt ?

Solution :
Besoin d'ajouter une hypothèse : le graphe G n'a pas de cycles négatifs.

S'il G n'a pas de cycles négatifs, il y a un plus court chemin élémentaire de longueur au plus $n - 1$.

Algorithme de Ford-Bellman

Le coût minimal $f(v, k)$ d'un o-v chemin à k arcs vérifie l'équation de Bellman

$$f(v, k+1) = \min_{u \in N^-(v)} f(u, k-1) + c(u, v)$$

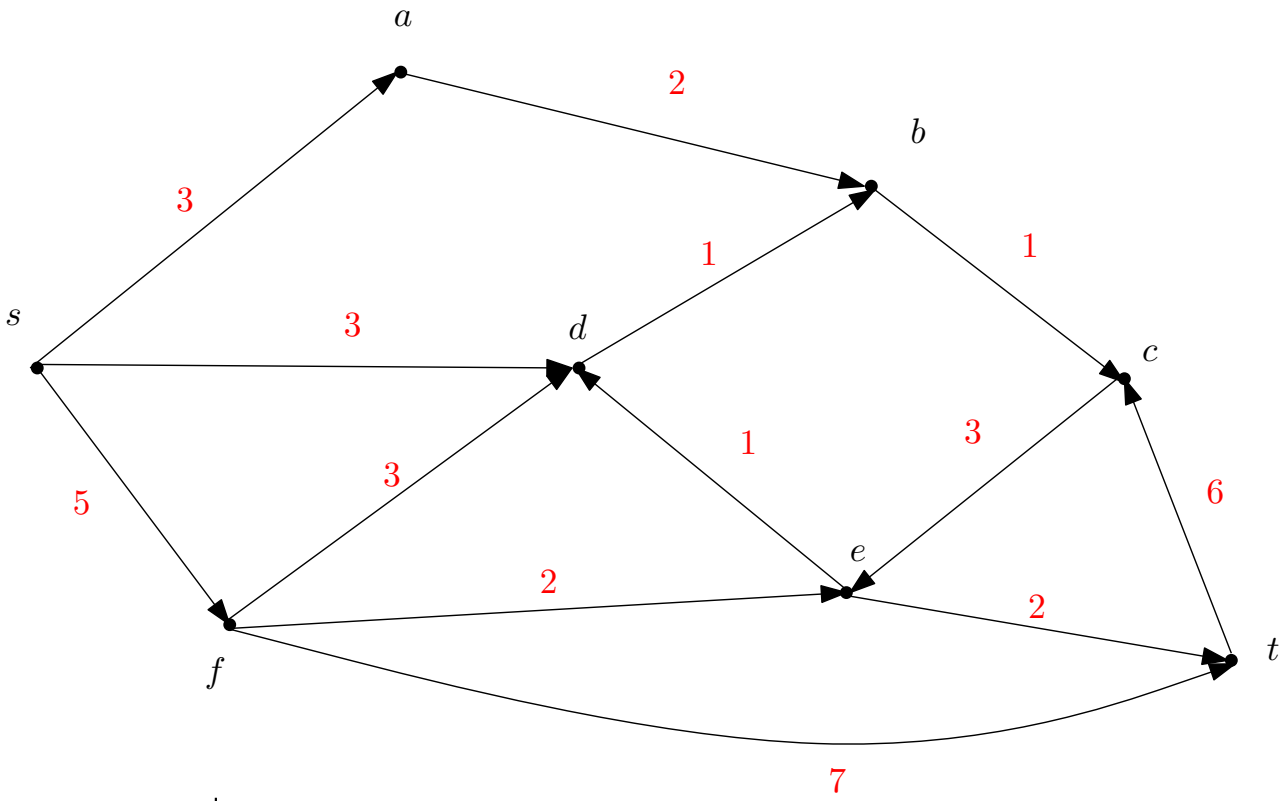
$$f(v, 0) = \begin{cases} 0 & \text{if } v = o \\ +\infty & \text{otherwise} \end{cases}$$

On peut calculer récursivement à partir de $k = 0$. Critère d'arrêt ?

Solution :
Besoin d'ajouter une hypothèse : le graphe G n'a pas de cycles négatifs.

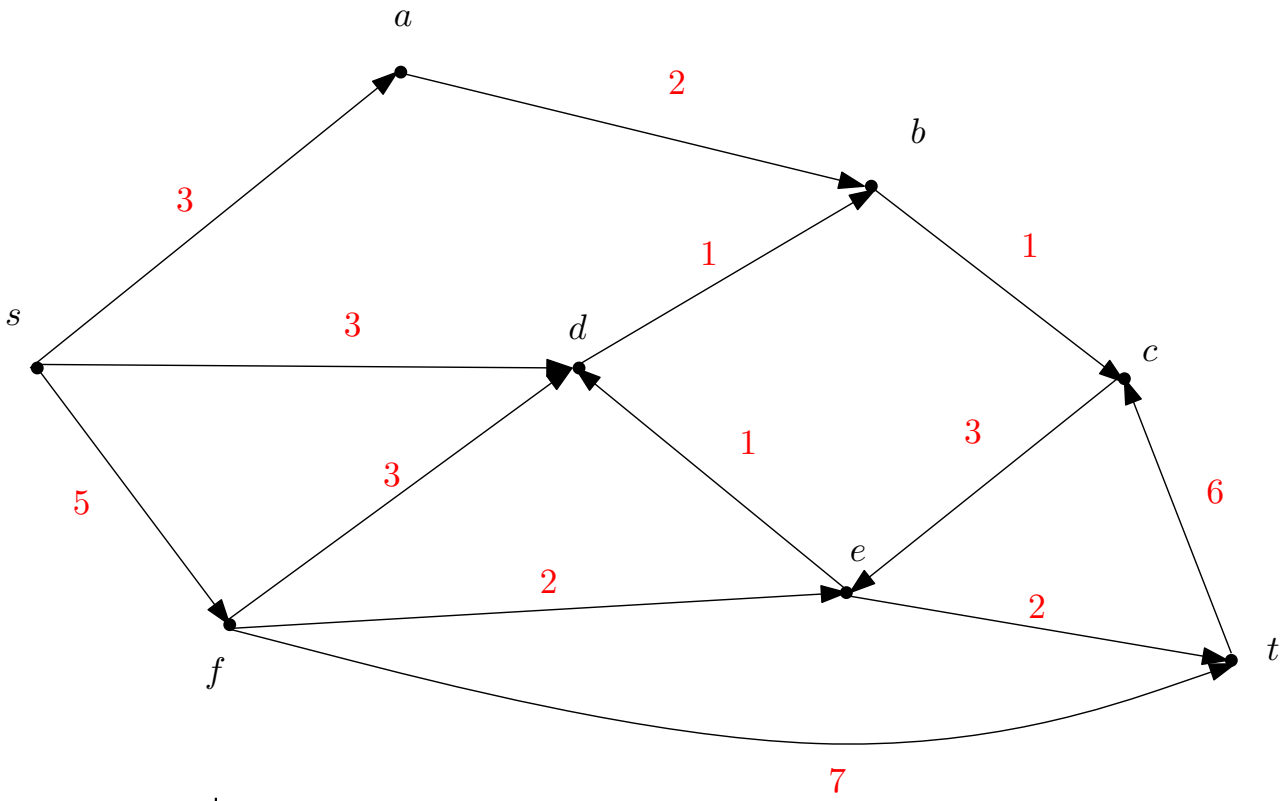
S'il G n'a pas de cycles négatifs, il y a un plus court chemin élémentaire de longueur au plus $n - 1$.

Exemple : Ford-Bellman



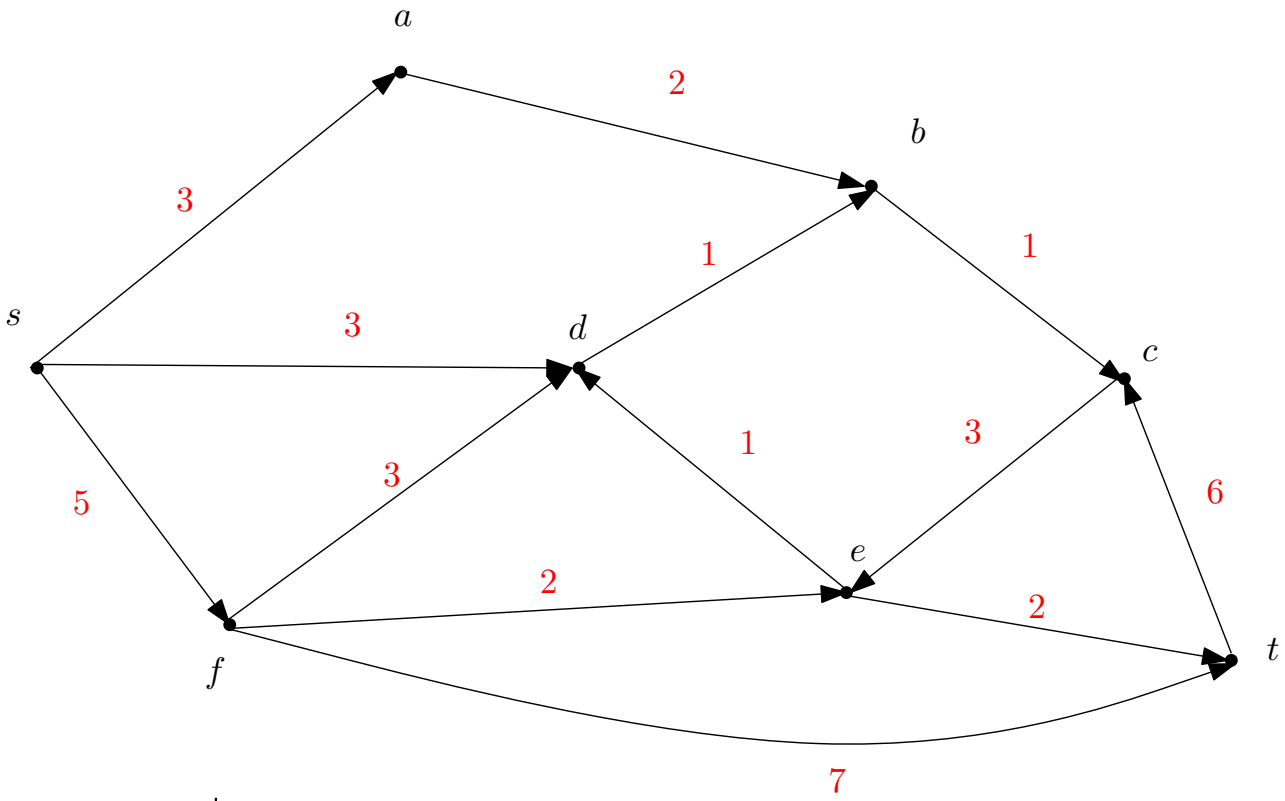
k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞

Exemple : Ford-Bellman



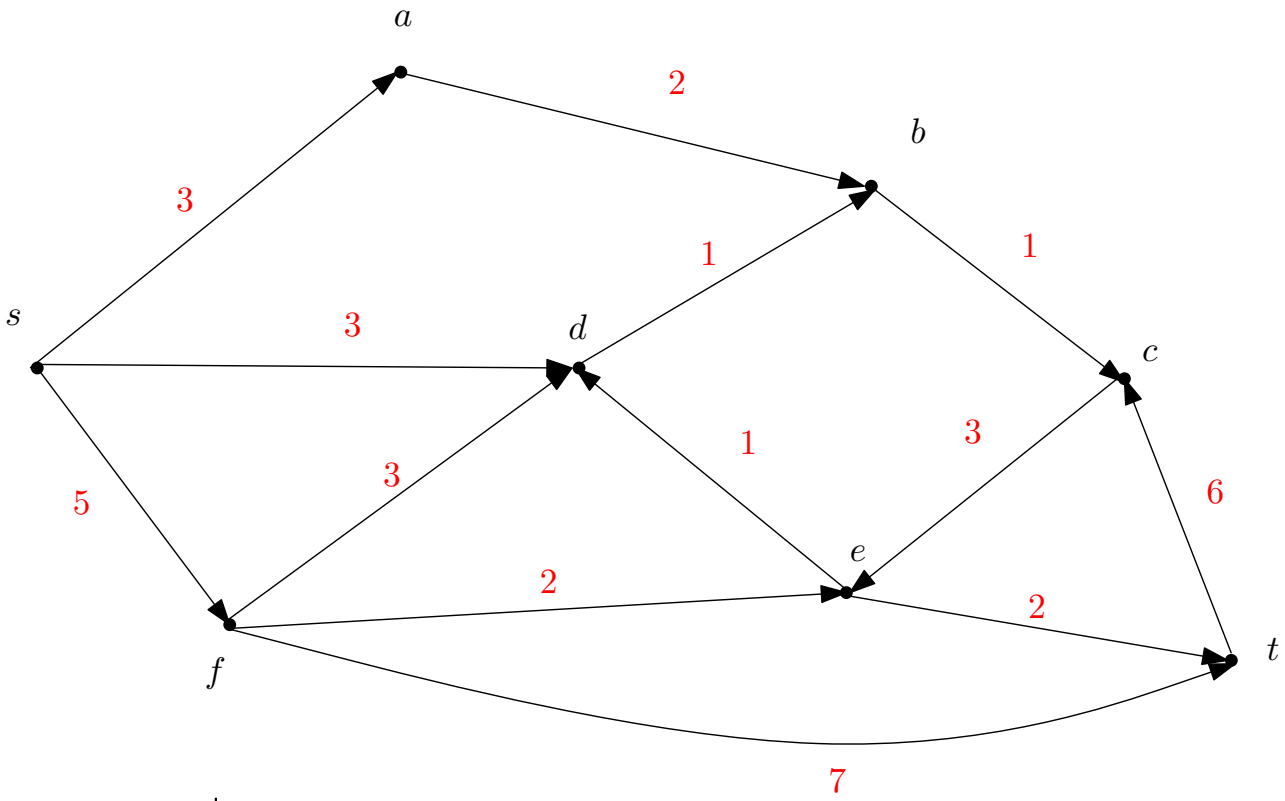
k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞

Exemple : Ford-Bellman



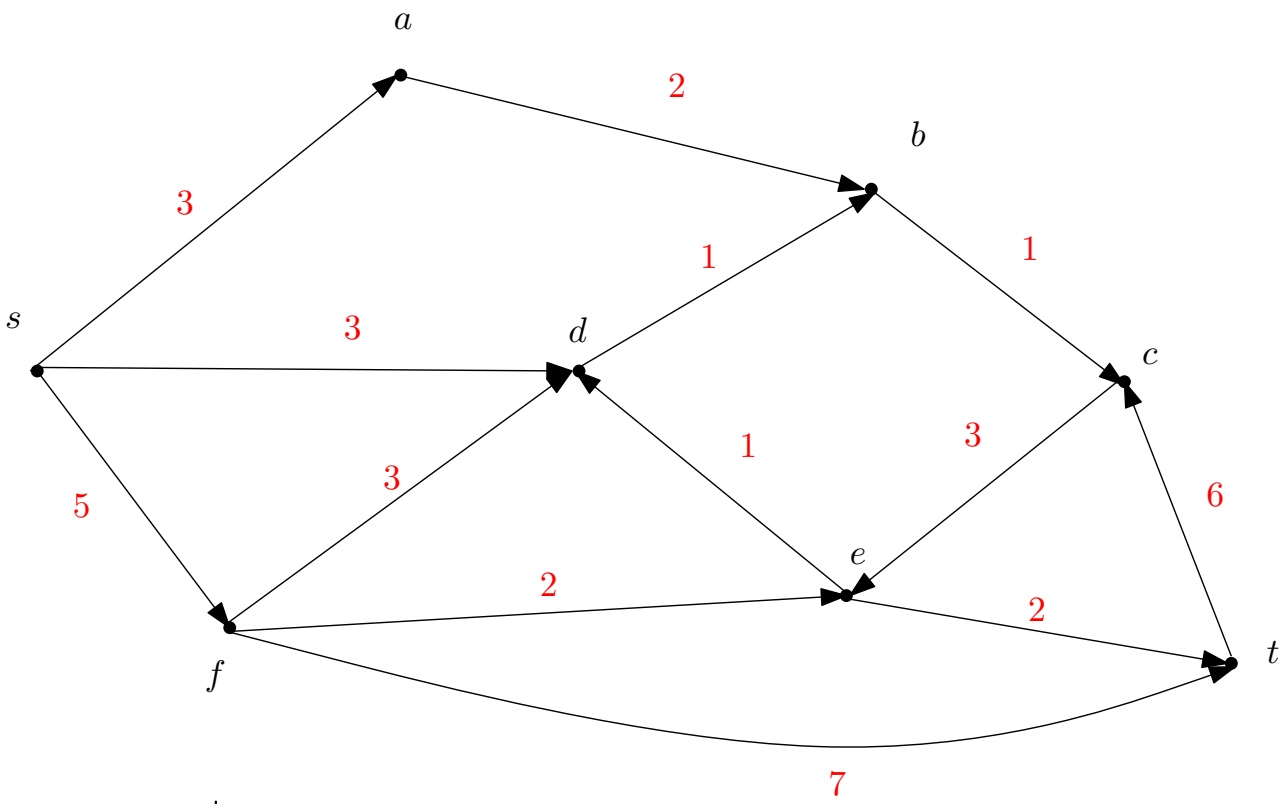
k	*s*	*a*	*b*	*c*	*d*	*e*	*f*	*t*
0	**0**	∞	∞	∞	∞	∞	∞	∞
1	∞	**3**	∞	∞	**3**	∞	**5**	∞
2	∞	∞	**5**	∞	**8**	**7**	∞	∞

Exemple : Ford-Bellman



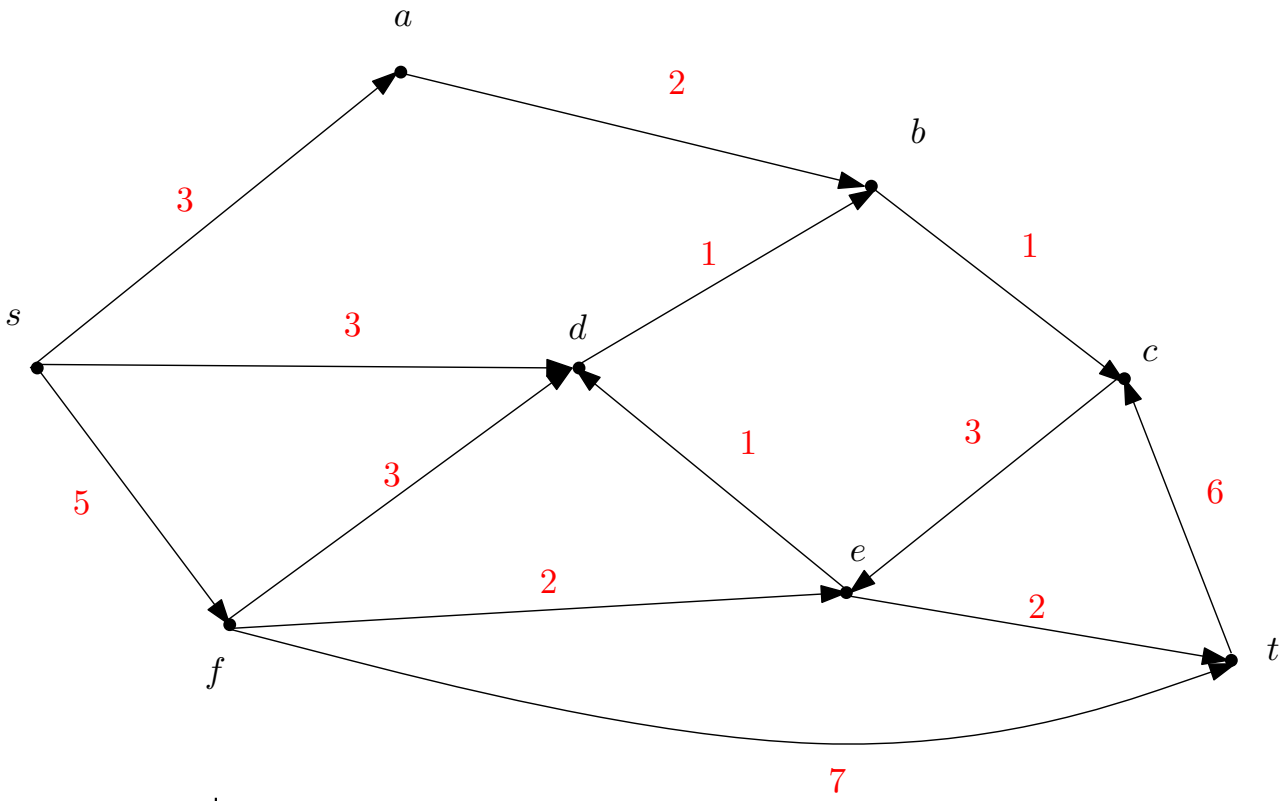
k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞
2	∞	∞	5	∞	8	7	∞	∞
3	∞	∞	9	5	8	∞	∞	9

Exemple : Ford-Bellman



k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞
2	∞	∞	5	∞	8	7	∞	∞
3	∞	∞	9	5	8	∞	∞	9
4	∞	∞	9	6	∞	8	∞	9

Exemple : Ford-Bellman



k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞
2	∞	∞	5	∞	8	7	∞	∞
3	∞	∞	9	5	8	∞	∞	9
4	∞	∞	9	6	∞	8	∞	9
5	∞	∞	9	10	9	10	∞	11

L'algorithme de Ford-Bellman

Peut être facilement adapté pour calculer le chemin lui-même (et pas seulement le coût minimum)

Pas de généralisation triviale aux graphes non-orientés (la réduction crée des cycles négatifs)

Converge en $O(mn)$ avec
n : nombre de sommets
m : nombres d'arcs

L'algorithme de Ford-Bellman

Peut être facilement adapté pour calculer le chemin lui-même (et pas seulement le coût minimum)

Pas de généralisation triviale aux graphes non-orientés (la réduction crée des cycles négatifs)

Converge en $O(mn)$ avec
n : nombre de sommets
m : nombres d'arcs

L'algorithme de Ford-Bellman

Peut être facilement adapté pour calculer le chemin lui-même (et pas seulement le coût minimum)

Pas de généralisation triviale aux graphes non-orientés (la réduction crée des cycles négatifs)

Converge en $\boldsymbol{O}(\boldsymbol{mn})$ avec
$\boldsymbol{n}$: nombre de sommets
$\boldsymbol{m}$: nombres d'arcs

Table of Contents

L'idée de Bellman

Principe d'optimalité de Bellman

La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.4)

Soit G un graphe orienté acylique, P un o-v chemin et Q son sous-o-u chemin, où u est le sommet avant v dans P.
Si P est le plus petit o-v chemin, alors Q est le plus petit o-u chemin.

L'équation de Bellman devient $f(v) = \min_{u \in N^-(v)} f(u) + c(u, v)$

Calculer tous les $f(v)$ par récurrence :
C'est possible selon un ordre dit topologique $\preceq$ tel que
$(u, v) \in A \Rightarrow u \preceq v$

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.4)

Soit G un graphe orienté acylique, P un o-v chemin et Q son sous-o-u chemin, où u est le sommet avant v dans P.
Si P est le plus petit o-v chemin, alors Q est le plus petit o-u chemin.

L'équation de Bellman devient $f(v) = \min_{u \in N^-(v)} f(u) + c(u, v)$

Calculer tous les $f(v)$ par récurrence :
C'est possible selon un ordre dit topologique $\preceq$ tel que
$(u, v) \in A \Rightarrow u \preceq v$

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.4)

Soit G un graphe orienté acylique, P un o-v chemin et Q son sous-o-u chemin, où u est le sommet avant v dans P.
Si P est le plus petit o-v chemin, alors Q est le plus petit o-u chemin.

L'équation de Bellman devient $f(v) = \min_{u \in N^-(v)} f(u) + c(u, v)$

Calculer tous les $f(v)$ par récurrence :
C'est possible selon un ordre dit topologique $\preceq$ tel que
$(u, v) \in A \Rightarrow u \preceq v$

L'idée de Bellman

Principe d'optimalité de Bellman
La sous-trajectoire d'une trajectoire optimale est encore optimale :

Proposition (5.4)

Soit G un graphe orienté acylique, P un o-v chemin et Q son sous-o-u chemin, où u est le sommet avant v dans P.
Si P est le plus petit o-v chemin, alors Q est le plus petit o-u chemin.

L'équation de Bellman devient $f(v) = \min_{u \in N^-(v)} f(u) + c(u, v)$

Calculer tous les $f(v)$ par récurrence :
C'est possible selon un ordre dit topologique $\preceq$ tel que
$(u, v) \in A \Rightarrow u \preceq v$

Programmation dynamique

On a

1. système dynamique à temps discret $x_{k+1} = f_k(x_k, u_k), \quad k = 1, 2, \ldots, N$
2. fonction de coût additive dans le temps.

x_0 : état initial.

x_k : état au début de la période k.

u_k : décision pour la période k.

$c_k(x_k, x_{k+1})$: coût de la transition de x_k à x_{k+1} sur la période k.

Trouver la trajectoire $x_0, x_1, \ldots, x_N$ de coût total $\sum_{k=1}^{N} c_k(x_k, x_{k+1})$ minimal : peut se résoudre par la programmation dynamique.

Table of Contents

Programmation dynamique

$\boldsymbol{V}(\boldsymbol{t}, \boldsymbol{x}) :=$ coût minimal future lorsqu'on est dans l'état $\boldsymbol{x}$ à l'étape $\boldsymbol{t}$.

Programmation dynamique : équation de Bellman pour tout $\boldsymbol{y}$

$$\boldsymbol{V}(\boldsymbol{t}, \boldsymbol{x}) = \min_{\boldsymbol{y} \in \boldsymbol{X}_{\boldsymbol{k}+1}} (\boldsymbol{c}_{\boldsymbol{t}}(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{V}(\boldsymbol{t} + \mathbf{1}, \boldsymbol{y}))$$

Algorithme : calculer de proche en proche

Gestion de stock

Dynamique d'un stock : $x_{t+1} = x_t - d_t + u_t$,

d_t : demande pour la période t (supposée connue)
$x_t \in \mathbb{Z}$: nombre d'unités disponibles en début de période t
u_t : nombre d'unités commandées (et reçues immédiatement) en début de période t
K : capacité maximale de stockage
$c(u_t) + g(x_{t+1})$: coût de gestion de stock pour la période t en

- $c(u_t)$ coût de réapprovisionnement
- $g(x_{t+1})$ coût de stockage ou coût de pénurie

Objectif. Minimiser $\sum_{t=0}^{T-1} c(u_t) + g(x_{t+1})$.

Gestion de stock

Modélisation par la programmation dynamique

périodes : périodes $t = 1, \ldots, T$

états : valeurs possibles x_t de niveau de stock,

transitions : $x_t \to x_{t+1}$ tq x_{t+1} satisfasse simultanément $x_{t+1} \in \mathbb{Z}$, $x_{t+1} + d_t \leq K$ et $x_{t+1} \geq x_t - d_t$

coût de la transition : $x_t \to x_{t+1}$ est $c(x_{t+1} - x_t + d_t) + g(x_{t+1})$

Objectif. Minimiser $\sum_{t=0}^{N-1} c(x_{t+1} - x_t + d_t) + g(x_{t+1})$ (critère additif = somme des coûts des transitions de la trajectoire).

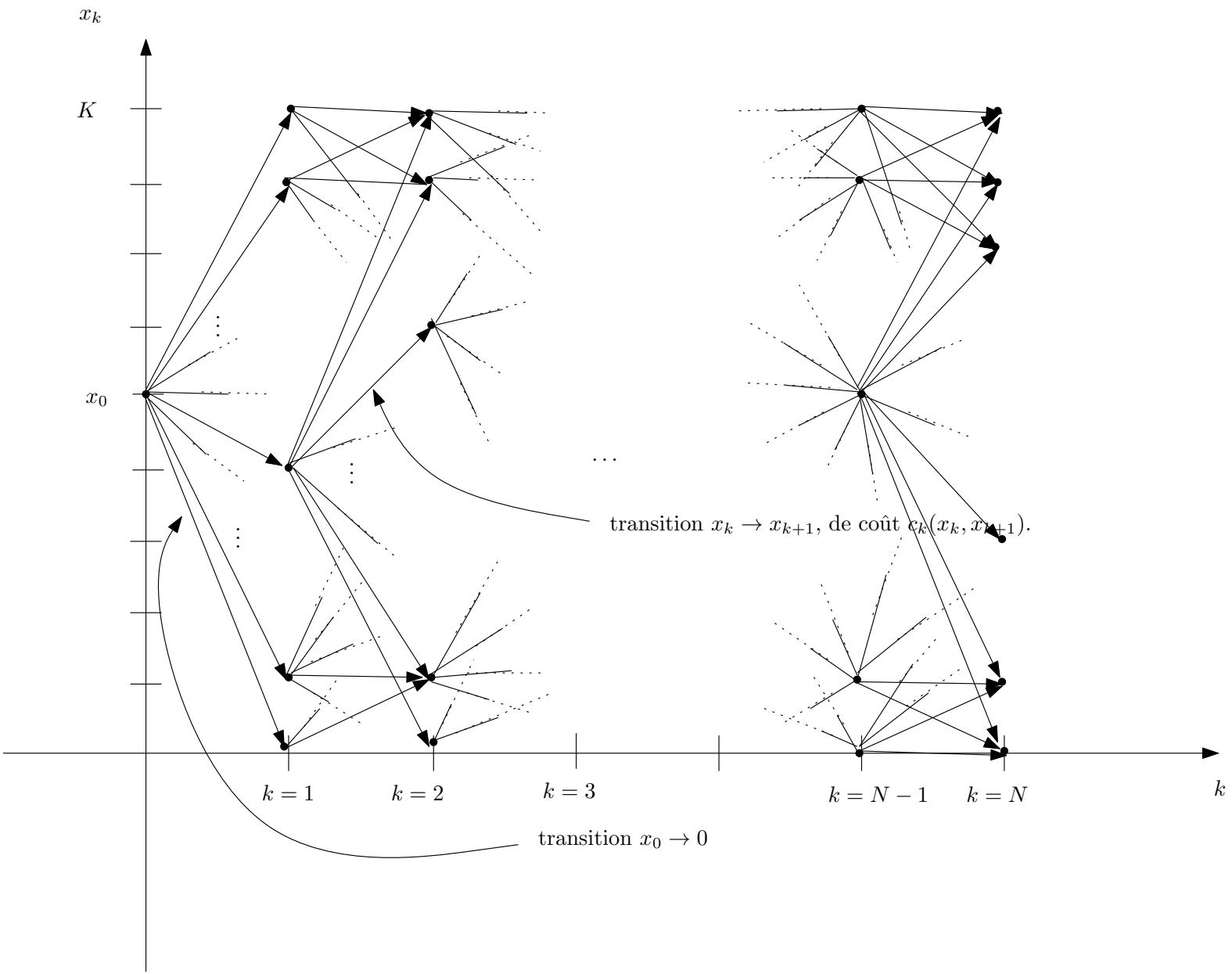
Programmation dynamique = plus court chemin

Programmation dynamique = plus court chemin dans un graphe acircuitique

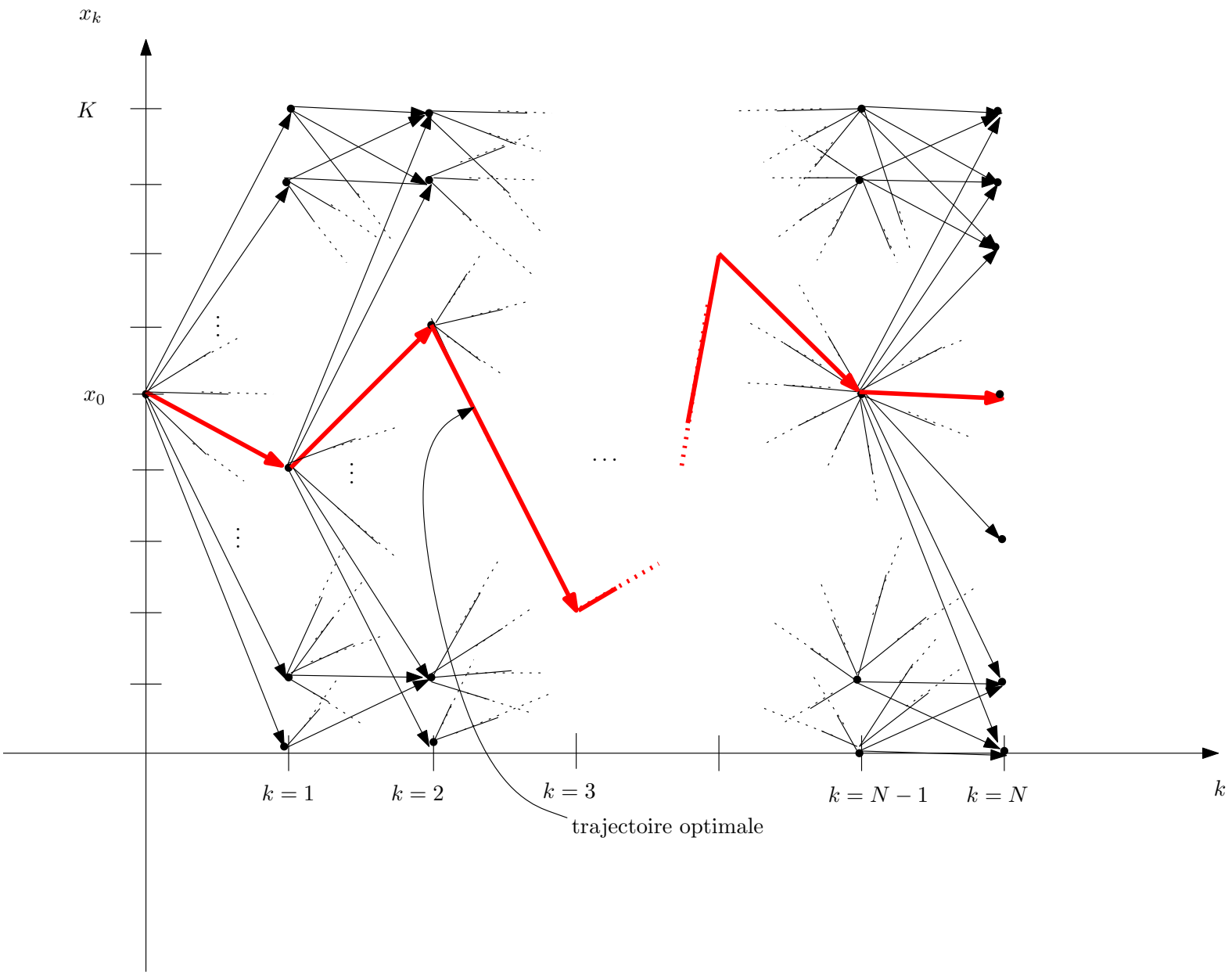
Prog. dyn.	Plus court ***o***-***d*** chemin
états $\times$ périodes	sommets
transitions	arcs
coût de la transition	longueur l'arc
trajectoire	chemin
trajectoire optimale	plus court chemin

Si plusieurs états de départ ou plusieurs états à l'arrivée, on peut ajouter des sommets fictifs ***o*** et ***d***, et des transitions de coût $=$ **0**.

Programmation dynamique = plus court chemin



Programmation dynamique = plus court chemin



Programmation dynamique = plus court chemin

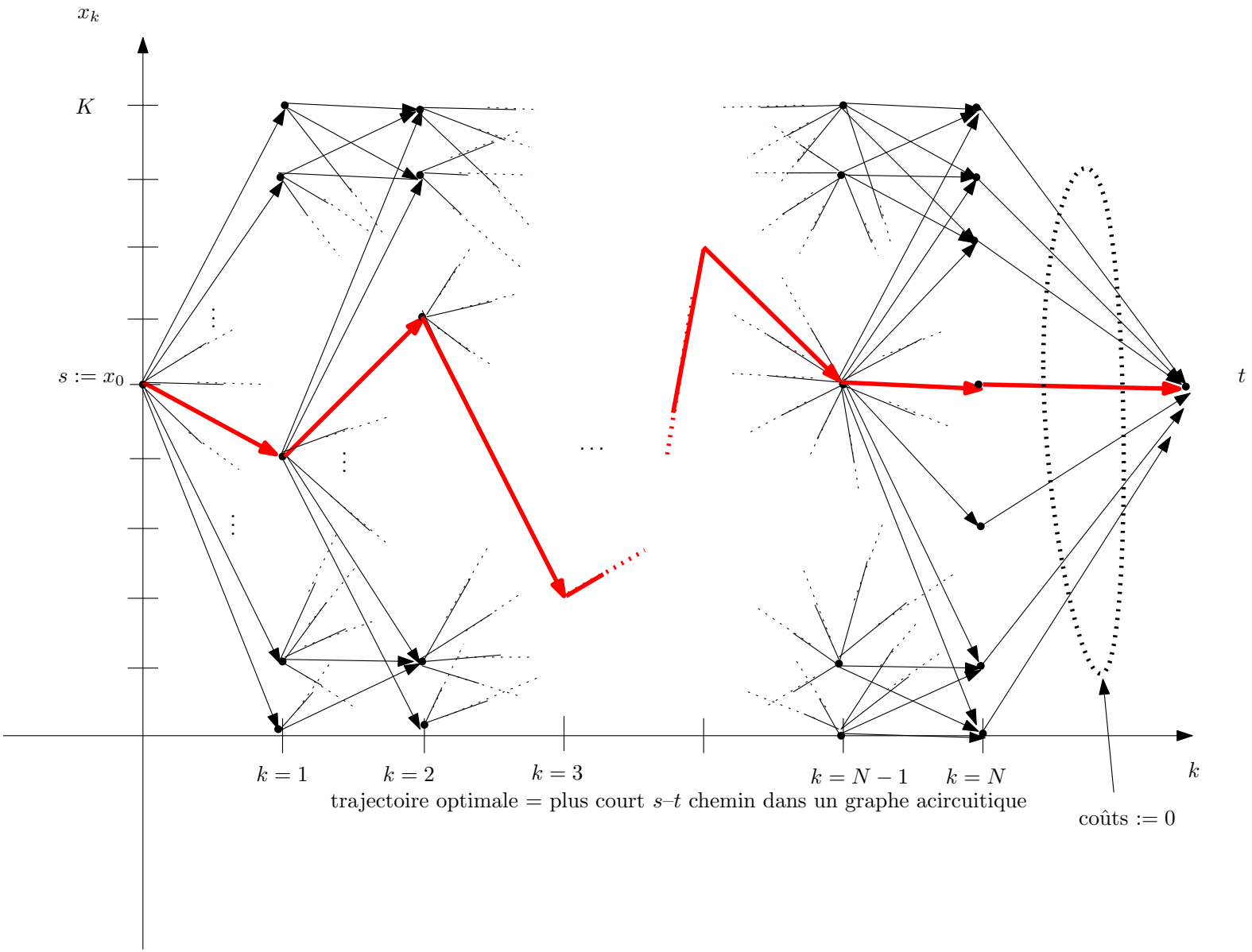


Table of Contents

Complexité : Résumé

	Complexité
Graphe orienté, $\boldsymbol{c(a) \geq 0}$	$\boldsymbol{O(m + nlog(n))}$ (Dijkstra)
Graphe orienté, acyclique	$\boldsymbol{O(m)}$ (Programmation dynamique)
Graphe orienté sans cycle absorbant	$\boldsymbol{O(nm)}$ (Programmation dynamique, Ford-Bellman)
Graphe orienté	**NP**-difficile
Graphe non-orienté, $\boldsymbol{c(a) \geq 0}$	$\boldsymbol{O(m + nlog(n))}$ (Dijkstra)
Graphe non-orienté, sans cycle absorbant	$\boldsymbol{O(n^3)}$ (plus compliqué)
Graphe non-orienté	**NP**-difficile