

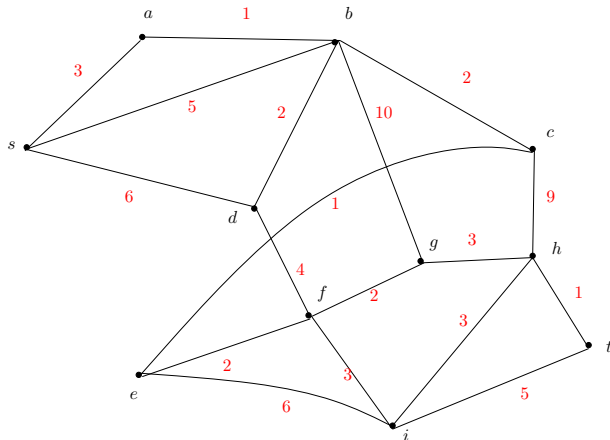
Plus courts chemins et programmation dynamique discrète

Maël Forcier

7 octobre 2020

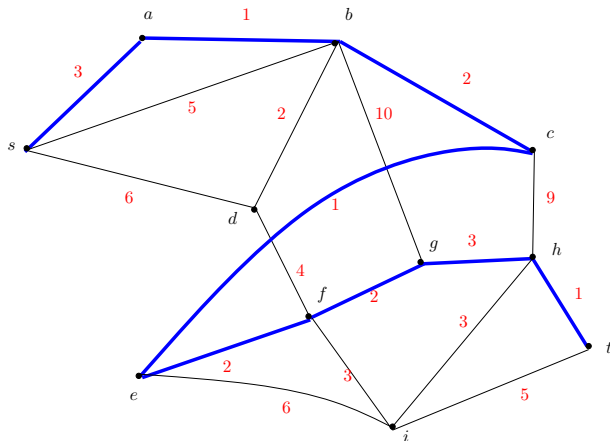
École des Ponts, France

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

Problème du plus court chemin :

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

Rappel :

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

Problème du plus court chemin :

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

Rappel :

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

Problème du plus court chemin :

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

Rappel :

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

Théorème

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

Preuve : Réduire au problème du plus court chemin élémentaire.

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 $-(\text{card}(\mathbf{V}) - 1)$

Théorème

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

Preuve : Réduire au problème du plus court chemin élémentaire.

Réduire au problème de chemin hamiltonien :

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

Il existe un chemin hamiltonien

ssi le coût minimal est égal à $-(\text{card}(\mathbf{V}) - 1)$

Théorème

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

Preuve : Réduire au problème du plus court chemin élémentaire.

Réduire au problème de chemin hamiltonien :

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

Il existe un chemin hamiltonien

ssi le coût minimal est égal à $-(\text{card}(\mathbf{V}) - 1)$

Théorème

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

Preuve : Réduire au problème du plus court chemin élémentaire.

Réduire au problème de chemin hamiltonien :

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

Il existe un chemin hamiltonien

ssi le coût minimal est égal à $-(\text{card}(\mathbf{V}) - 1)$

Théorème

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

Preuve : Réduire au problème du plus court chemin élémentaire.

Réduire au problème de chemin hamiltonien :

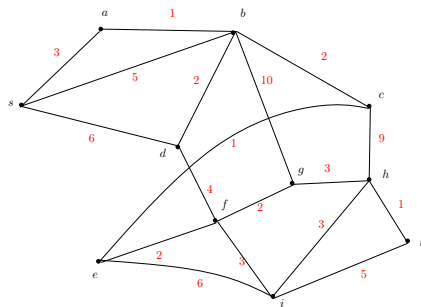
Prendre $\mathbf{c}(\mathbf{e}) = -1$ pour tout $\mathbf{e} \in \mathbf{E}$

Il existe un chemin hamiltonien

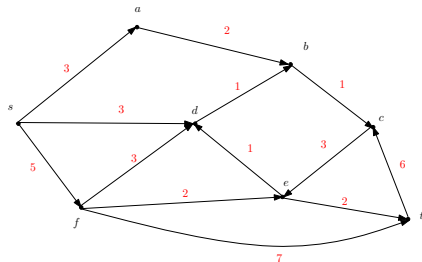
ssi le coût minimal est égal à $-(\text{card}(\mathbf{V}) - 1)$

Graphe non-orienté / orienté

graphe non-orienté

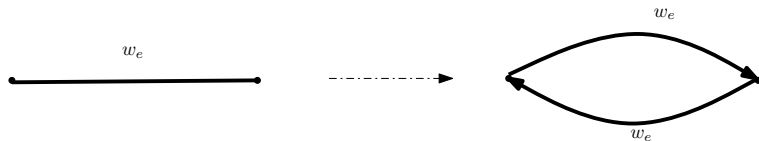


graphe orienté



Réduction d'orienté à non-orienté

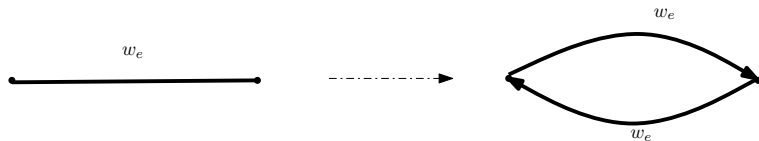
Réduction :



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

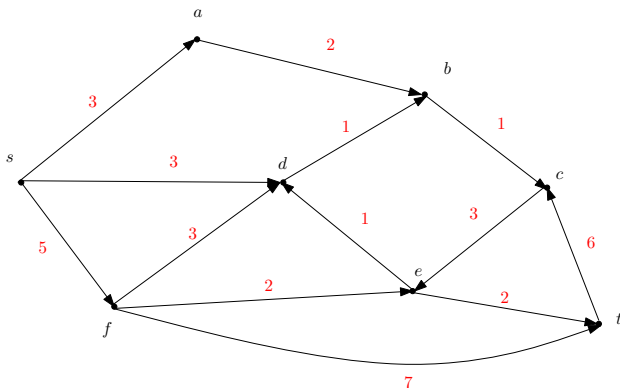
- 1 Coûts positifs : Dijkstra
- 2 Programmation Dynamique Ford-Bellman
 - L'algorithme de Ford-Bellman
 - Graphes acycliques : ordre topologique
- 3 Complexité résumé

Table of Contents

- 1 Coûts positifs : Dijkstra
- 2 Programmation Dynamique Ford-Bellman
 - L'algorithme de Ford-Bellman
 - Graphes acycliques : ordre topologique
- 3 Complexité résumé

Tous les coûts sont ≥ 0

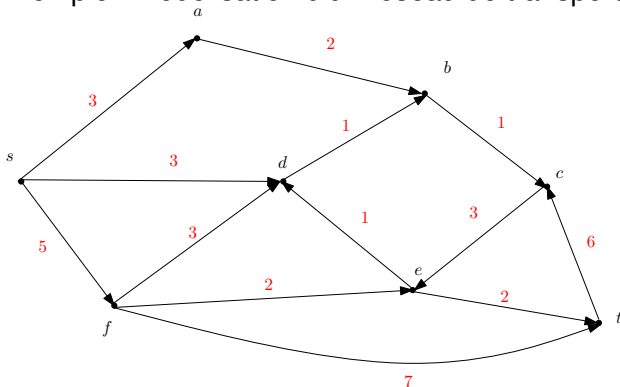
Cas **le plus naturel** : tous les poids sont ≥ 0 .



Tous les coûts sont ≥ 0

Cas **le plus naturel** : tous les poids sont ≥ 0 .

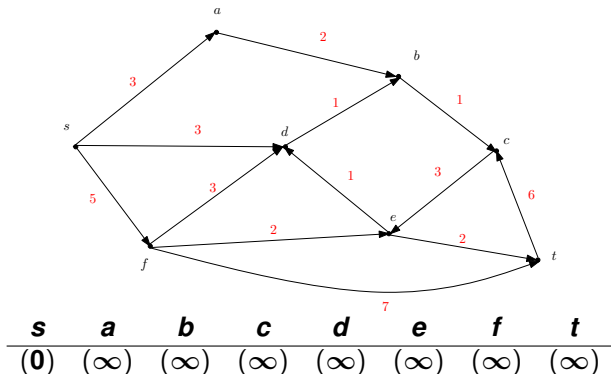
Exemple : Modélisation d'un réseau de transport.



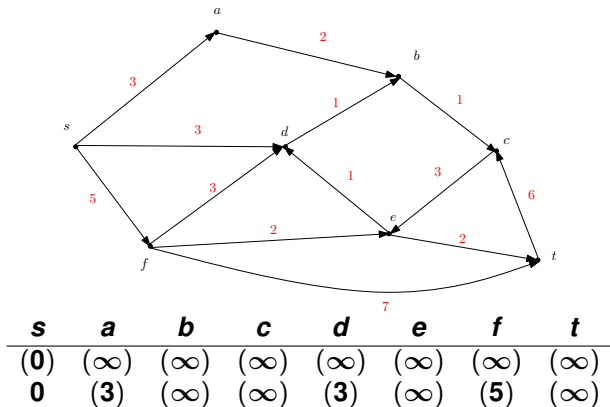
L'algorithme de Dijkstra

- Initialisation : $U := \emptyset$, $d(o) := 0$ et $d(v) = +\infty$ pour tout $v \neq o$.
- Tant que $V \setminus U \neq \emptyset$
 - 1 Choisir v minimisant $d(v)$ dans $V \setminus U$.
 - 2 $U := U \cup \{v\}$.
 - 3 Pour chaque $a = (u, v) \in A$
 - $d(v) := \min[d(v), d(u) + c(a)]$.

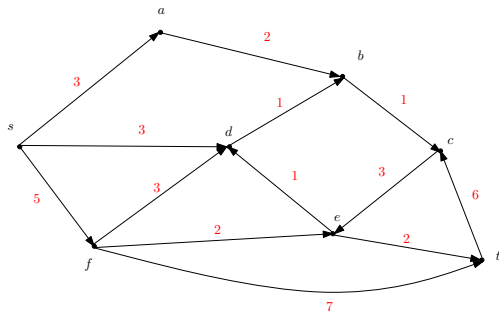
Exemple : Dijkstra



Exemple : Dijkstra

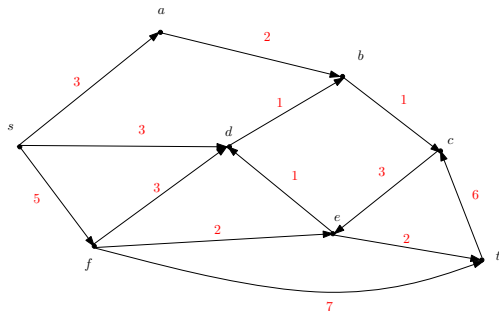


Exemple : Dijkstra



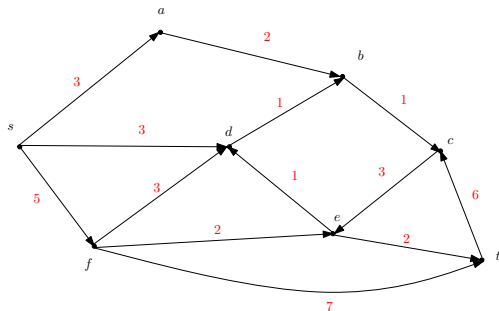
s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)

Exemple : Dijkstra



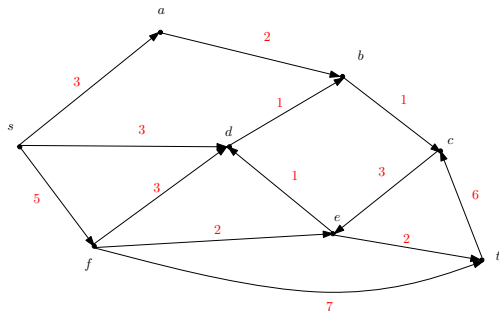
s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)

Exemple : Dijkstra



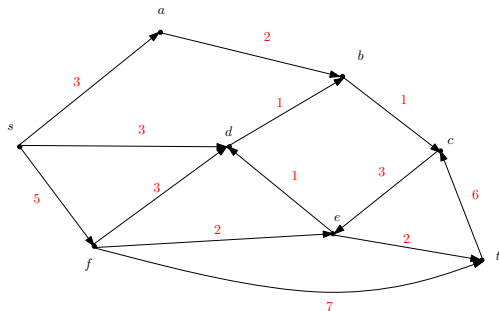
s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)

Exemple : Dijkstra



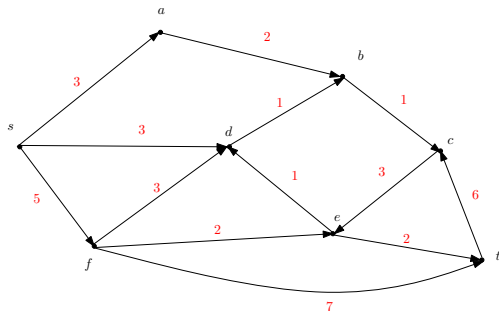
s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)
0	3	4	5	3	(8)	(5)	(∞)

Exemple : Dijkstra



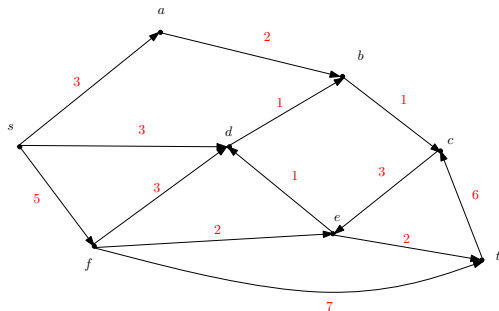
s	a	b	c	d	e	f	t
(0)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)	(∞)
0	(3)	(∞)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(5)	(∞)	(3)	(∞)	(5)	(∞)
0	3	(4)	(∞)	3	(∞)	(5)	(∞)
0	3	4	(5)	3	(∞)	(5)	(∞)
0	3	4	5	3	(8)	(5)	(∞)
0	3	4	5	3	(7)	5	(12)

Exemple : Dijkstra



<i>s</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>t</i>
(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



<i>s</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>t</i>
(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 $O(m + n \log(n))$ 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 $O(m + n \log(n))$ 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 $O(m + n \log(n))$ avec

n : nombre de sommets

m : nombres d'arcs

Table of Contents

1 Coûts positifs : Dijkstra

2 Programmation Dynamique Ford-Bellman

- L'algorithme de Ford-Bellman
- Graphes acycliques : ordre topologique

3 Complexité résumé

- 1 Coûts positifs : Dijkstra
- 2 Programmation Dynamique Ford-Bellman
 - L'algorithme de Ford-Bellman
 - Graphes acycliques : ordre topologique
- 3 Complexité résumé

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, où u est le sommet avant v dans P . Si P est le plus petit $o-v$ chemin parmi 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.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

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, où u est le sommet avant v dans P . Si P est le plus petit $o-v$ chemin parmi 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.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

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, où u est le sommet avant v dans P . Si P est le plus petit $o-v$ chemin parmi 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.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

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, où u est le sommet avant v dans P . Si P est le plus petit $o-v$ chemin parmi 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.
- $c(P') = c(Q') + c(u, v) < c(Q) + c(u, v) = c(P)$.

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, où u est le sommet avant v dans P . Si P est le plus petit $o-v$ chemin parmi 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 $o \rightarrow v$ chemin à k arcs vérifie l'équation de Bellman

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

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

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 $o \rightarrow v$ chemin à k arcs vérifie l'équation de Bellman

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

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

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

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

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

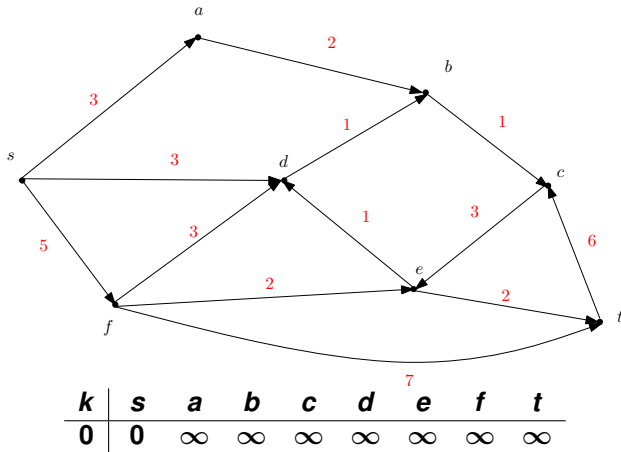
On peut calculer récursivement à partir de $k = 0$. Critère d'arrêt ?

Solution :

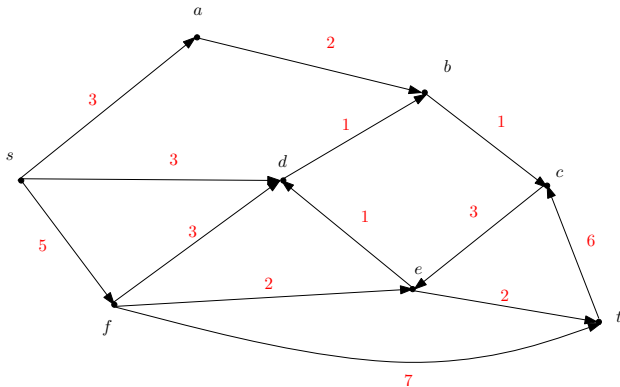
Besoin d'ajouter une hypothèse : le graphe G n'a pas de cycles négatifs.

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

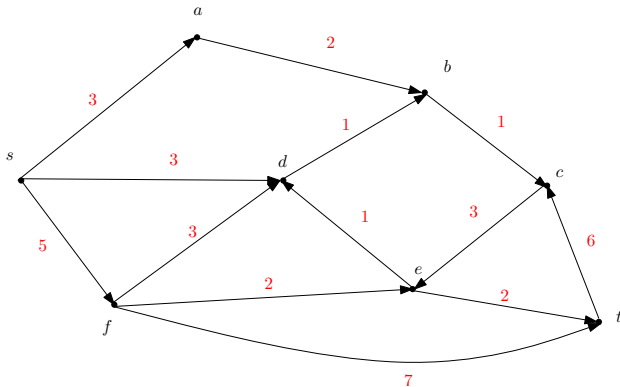


Exemple : Ford-Bellman



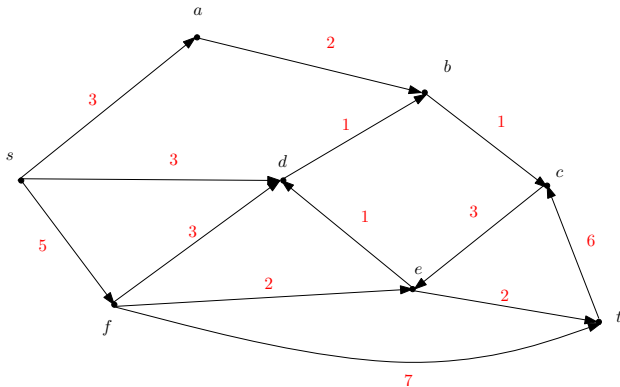
k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞

Exemple : Ford-Bellman



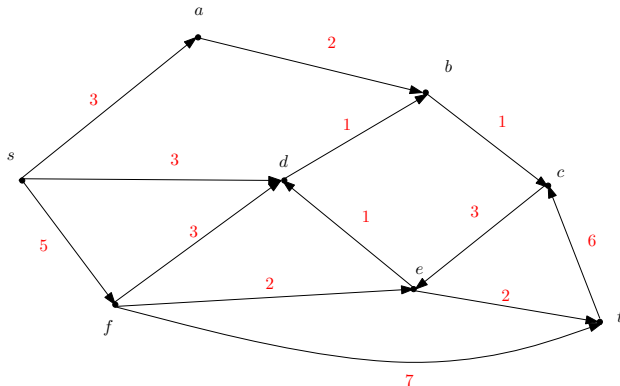
k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞
2	∞	∞	5	∞	8	7	∞	∞

Exemple : Ford-Bellman



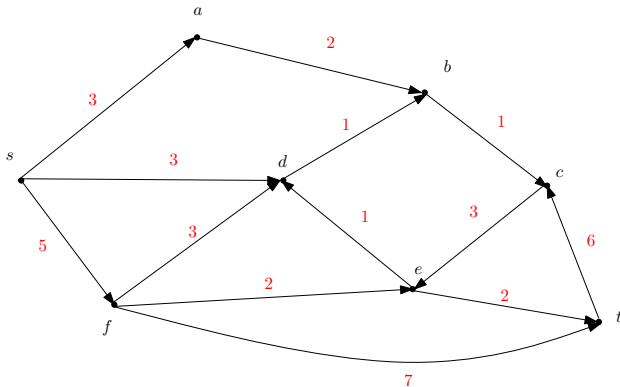
k	s	a	b	c	d	e	f	t
0	0	∞	∞	∞	∞	∞	∞	∞
1	∞	3	∞	∞	3	∞	5	∞
2	∞	∞	5	∞	8	7	∞	∞
3	∞	∞	9	5	8	∞	∞	9

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

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

- 1 Coûts positifs : Dijkstra
- 2 Programmation Dynamique Ford-Bellman
 - L'algorithme de Ford-Bellman
 - Graphes acycliques : ordre topologique
- 3 Complexité résumé

Principe d'optimalité de Bellman

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

Proposition (5.4)

Soit G un graphe orienté acyclique, 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

$$(u, v) \in A \Rightarrow u \preceq v$$

Principe d'optimalité de Bellman

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

Proposition (5.4)

Soit G un graphe orienté acyclique, 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 acyclique, 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 \mathbf{G} un graphe orienté acyclique, \mathbf{P} un $\mathbf{o}-\mathbf{v}$ chemin et \mathbf{Q} son sous- $\mathbf{o}-\mathbf{u}$ chemin, où \mathbf{u} est le sommet avant \mathbf{v} dans \mathbf{P} .

Si \mathbf{P} est le plus petit $\mathbf{o}-\mathbf{v}$ chemin, alors \mathbf{Q} est le plus petit $\mathbf{o}-\mathbf{u}$ chemin.

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

Calculer tous les $\mathbf{f}(\mathbf{v})$ par récurrence :

C'est possible selon un ordre dit topologique \preceq tel que

$(\mathbf{u}, \mathbf{v}) \in \mathbf{A} \Rightarrow \mathbf{u} \preceq \mathbf{v}$

Programmation dynamique

On a

- 1 système dynamique à temps discret $\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k)$, $k = 1, 2, \dots, N$
- 2 fonction de coût additive dans le temps.

\mathbf{x}_0 : état initial.

\mathbf{x}_k : état au début de la période k .

\mathbf{u}_k : décision pour la période k .

$\mathbf{c}_k(\mathbf{x}_k, \mathbf{x}_{k+1})$: coût de la transition de \mathbf{x}_k à \mathbf{x}_{k+1} sur la période k .

Trouver la trajectoire $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N$ de coût total $\sum_{k=1}^N \mathbf{c}_k(\mathbf{x}_k, \mathbf{x}_{k+1})$ minimal :
peut se résoudre par la programmation dynamique.

$V(\mathbf{k}, \mathbf{x})$:= coût minimal de la trajectoire amenant le système dans l'état \mathbf{x} en \mathbf{k} étapes.

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

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

Algorithme : calculer de proche en proche

Dynamique d'un stock : $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{d}_k + \mathbf{u}_k$,

\mathbf{d}_k : demande pour la période k (supposée connue)

$\mathbf{x}_k \in \mathbb{Z}$: nombre d'unités disponibles en début de période k

\mathbf{u}_k : nombre d'unités commandées (et reçues immédiatement) en début de période k

K : capacité maximale de stockage

$\mathbf{c}(\mathbf{u}_k) + \mathbf{g}(\mathbf{x}_{k+1})$: coût de gestion de stock pour la période k en

- $\mathbf{c}(\mathbf{u}_k)$ coût de réapprovisionnement
- $\mathbf{g}(\mathbf{x}_{k+1})$ coût de stockage ou coût de pénurie

Objectif. Minimiser $\sum_{k=0}^{N-1} \mathbf{c}(\mathbf{u}_k) + \mathbf{g}(\mathbf{x}_{k+1})$.

Modélisation par la programmation dynamique

périodes : périodes $k = 1, \dots, N$

états : valeurs possibles x_k de niveau de stock,

transitions : $x_k \rightarrow x_{k+1}$ tq x_{k+1} satisfasse simultanément $x_{k+1} \in \mathbb{Z}$,
 $x_{k+1} + d_k \leq K$ et $x_{k+1} \geq x_k - d_k$

coût de la transition : $x_k \rightarrow x_{k+1}$ est $c(x_{k+1} - x_k + d_k) + g(x_{k+1})$

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

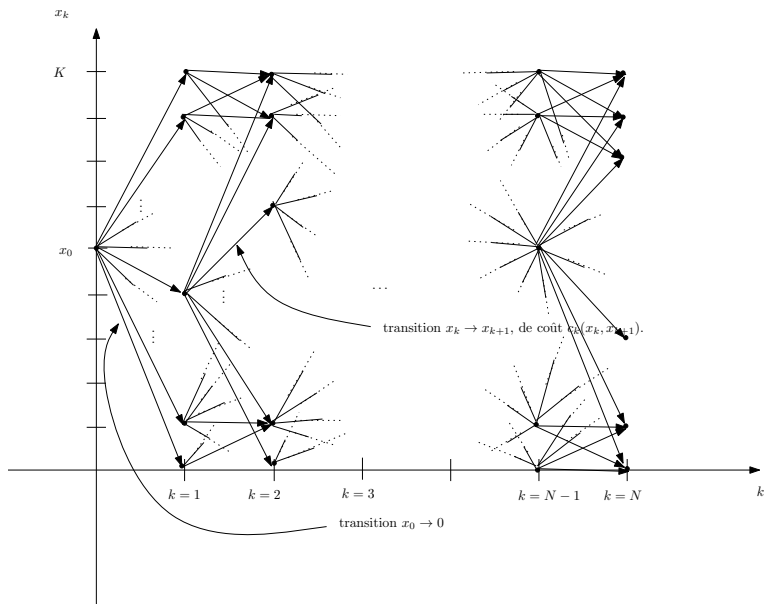
Programmation dynamique = plus court chemin

Programmation dynamique = plus court chemin dans un graphe acircuitique

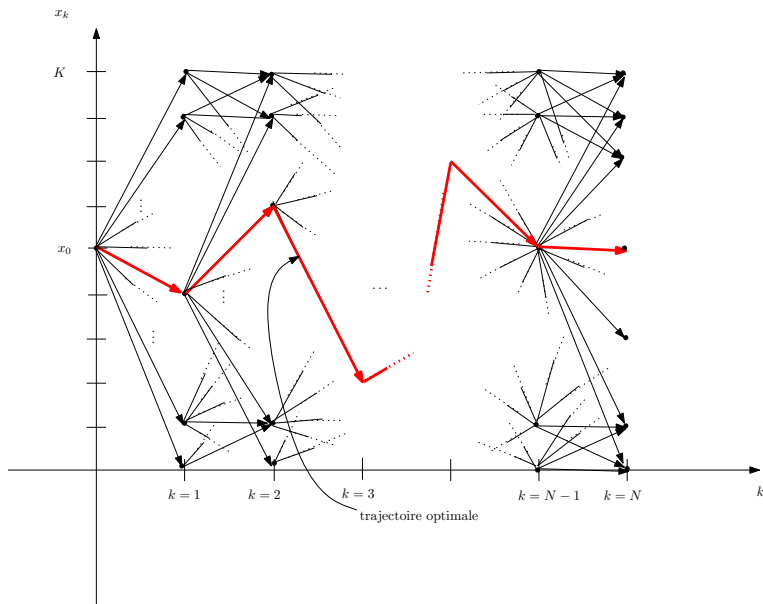
Prog. dyn.	Plus court s-t chemin
états \times périodes	sommets
transitions	arcs
coût de la transition	longueur l'arc
trajectoire	chemin
trajectoire optimale	plus court chemin

Si plusieurs états de départ ou plusieurs états à l'arrivée, on peut ajouter des sommets fictifs **s** et **t**, et des transitions de coût = **0**.

Programmation dynamique = plus court chemin



Programmation dynamique = plus court chemin



Programmation dynamique = plus court chemin

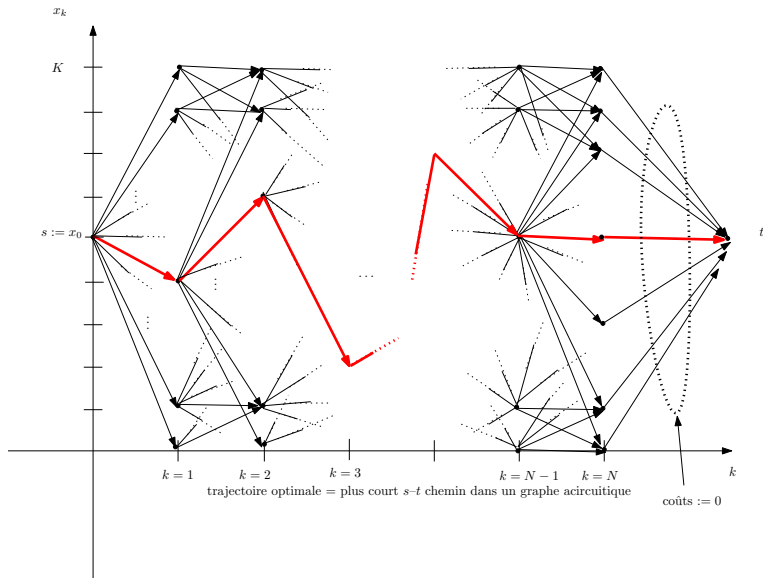


Table of Contents

1 Coûts positifs : Dijkstra

2 Programmation Dynamique Ford-Bellman

- L'algorithme de Ford-Bellman
- Graphes acycliques : ordre topologique

3 Complexité résumé

Complexité : Résumé

	Complexité
Graphe orienté, $c(a) \geq 0$	$O(m + n \log(n))$ (Dijkstra)
Graphe orienté, acyclique	$O(m)$ (Programmation dynamique)
Graphe orienté sans cycle absorbant	$O(nm)$ (Programmation dynamique, Ford-Bellman)
Graphe orienté	NP -difficile
<hr/>	
Graphe non-orienté, $c(a) \geq 0$	$O(m + n \log(n))$ (Dijkstra)
Graphe non-orienté, sans cycle absorbant	$O(n^3)$ (plus compliqué)
Graphe non-orienté	NP -difficile