# RESOLUTION DE PROBLEMES DE PLUS COURT CHEMIN

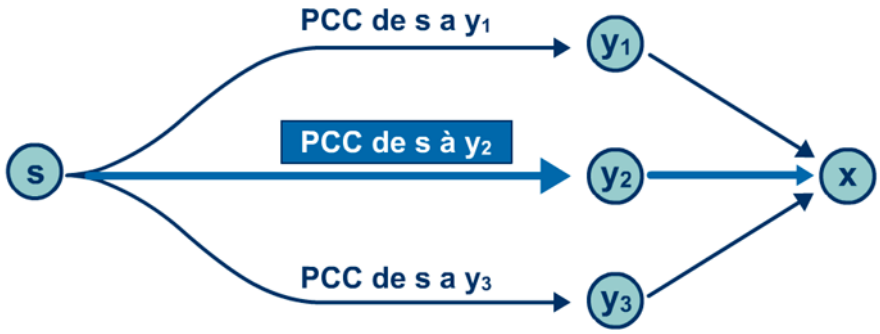
Dans la leçon précédente, nous avons vu comment résoudre le problème du plus court chemin dans un graphe valué par des nombres positifs.
Dans cette leçon, nous commençons par le cas des graphes sans circuit comme, par exemple, celui rencontré dans l'exemple de recherche de la politique optimale de remplacement de véhicule automobile.
Puis nous traiterons le cas d'un graphe quelconque.

## I Algorithme de détermination des plus courts chemins : cas des graphes sans circuit

**Principe de l'algorithme**
Soient :

s un sommet à partir duquel on cherche les plus courts chemins
x un autre sommet
Pred(x) l'ensemble des prédécesseurs de x

Le résultat que l'on va utiliser est le suivant :
Supposons connu un plus court chemin de s à chacun des éléments y de Pred(x), on peut déterminer le plus court chemin de s à x en comparant les chemins constitués d'un plus court chemin de s à y complété par l'arc (y,x).




y1, y2, y3 sont 3 prédécesseurs de x. PCC de s à $y_1$ est un plus court chemin de s à $y_1$, de même PCC de s à $y_2$ et PCC de s à $y_3$ représentent des plus courts chemins de s aux autres prédécesseurs de x.

Pour déterminer le plus court chemin de s à x, on compare les longueurs des 3 chemins :
Plus court chemin de s à y1 complété par l'arc (y1, x)
Plus court chemin de s à y2 complété par l'arc (y2, x)
Plus court chemin de s à y3 complété par l'arc (y3, x)

Et on prend …le plus court d'entre eux !

Notons qu'on obtient ainsi la longueur de ce plus court chemin mais aussi le dernier arc de ce chemin.

La mise en œuvre de ce résultat pour déterminer le plus court chemin de s à x, nécessite de connaître le plus court chemin de s à tous les prédécesseurs de x. Il faut donc que le graphe soit sans circuit. Dans ce cas, les sommets peuvent être examinés dans l'ordre du tri topologique.

**Algorithme de Bellman**

*Données*

- Un graphe valué sans circuit dont les sommets sont numérotés dans l'ordre du tri topologique
- Le sommet de départ s est racine du graphe et il est numéroté 1
- On connaît pour chaque sommet x ses prédécesseurs Pred(x).

*Initialisation*

Poser $\lambda(1) = 0$

*Corps de l'algorithme*

POUR x de 2 à n
    FAIRE (*on examine les prédécesseurs du sommet x*)
        Calculer $\lambda(x) = \min(\lambda(y) + l(y,x))$ pour y dans Pred(x) (*on parcourt les prédécesseurs de x, on calcule* $(\lambda(y) + l(y,x)$ *et on prend le minimum de ces quantités)*
        POSER père(x) = y avec y prédécesseur y de x pour lequel ce minimum est atteint
    FINFAIRE
FINPOUR
FIN

*Résultats*

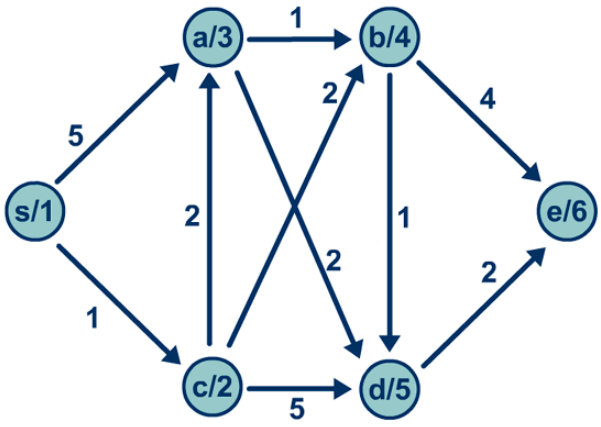
$\lambda(x)$ est égale à la longueur d'un plus court chemin de s à x.
Père(x) détermine le sommet prédécesseur de x sur le plus court chemin de s à x.

***Exemple***

On reprend l'exemple de la leçon précédente auquel on avait appliqué l'algorithme de Moore-Dijkstra.

Il s'agit d'abord de numéroter les sommets du graphe dans l'ordre du tri topologique.

Num(s) = 1 num(c) = 2 num(a) = 3 num(b) = 4 num(d) = 5 num(e) = 6
Pred(s) = ∅ Pred(c) = {s} Pred(a) = {s, c} Pred(b) = {a, c} Pred(d) = {a, b, c} Pred(e) = {b, d}




On calcule successivement :
$\lambda(s) = 0$

$\lambda(c) = \lambda(s) + l(s, c) = 0 + 1 = 1$  père(c) = s

$\lambda(a) = \text{Min}(\lambda(s) + l(s, a),\ \lambda(c) + l(c, a)) = \min(0 + 5, 1 + 2) = 3$ père(a) = c

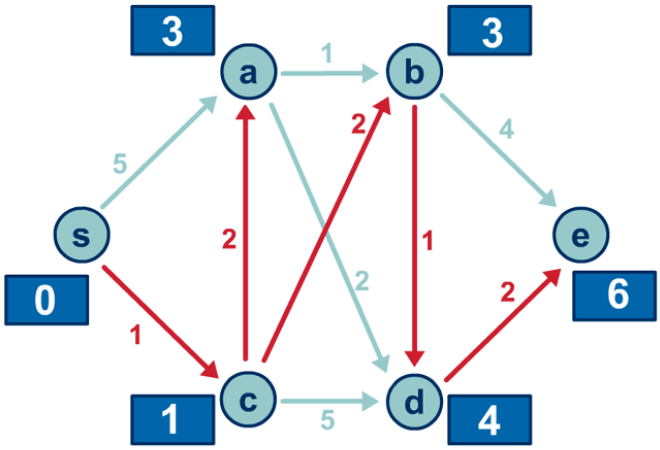
$\lambda(b) = \text{Min}(\lambda(a) + l(a, b),\ \lambda(c) + l(c, b)) = \min(3 + 1, 1 + 2) = 3$ père(b) = c

$\lambda(d) = \text{Min}(\lambda(a) + l(a, d),\ \lambda(b) + l(b, d),\ \lambda(c) + l(c, d)) = \min(3 + 2, 3 + 1, 1 + 5) = 4$ père(d ) = b

$\lambda(e) = \text{Min}(\lambda(b) + l(b, e),\ \lambda(d) + l(d, e)) = \min(3 + 4, 4 + 2) = 6$ père(e) = d

Ces résultats sont reportés sur le graphique suivant :
Les longueurs des plus courts chemins sont dans les étiquettes et les arcs en gras montrent pour chaque sommet l'arc le reliant à son père.

Pour retrouver les plus courts chemins de la racine à un sommet donné, on part de ce sommet et de père en père on remonte à la racine.
On obtient par exemple, pour le plus court chemin de s à e : e a pour père d, qui a pour père b, qui a pour père c, qui a pour père s.
Le plus court chemin de s à e est de longueur 6 ; c'est le chemin s c b d e.




*Commentaires*
1 - Comme dans l'algorithme de Moore-Dijkstra, dès qu'un sommet est étiqueté, il a sa marque définitive.
Si on cherche le plus court chemin jusqu'à un sommet donné, on peut arrêter l'algorithme dès que ce sommet a été examiné.

2 - Sur l'exemple, le plus court chemin du sommet s est unique, il est facile d'envisager des situations où ceci n'est pas le cas.
Si dans l'exemple précédent on change la longueur de l'arc (b, e) de 4 en 3, lors du calcul de l'étiquette du sommet e, les 2 termes dont on cherche le minimum sont égaux à 6. Le sommet "e" possède alors 2 pères !
Les chemins s c b d e et s c b e ont tous deux pour longueur 6.

3 - Efficacité de l'algorithme de Bellman
L'algorithme de Bellman est un algorithme très efficace.
Le temps de calcul croît comme m, le nombre d'arcs du graphe.

Lorsqu'il est nécessaire de numéroter auparavant les sommets par un tri topologique, nous avons vu que le temps de calcul de l'opération de numérotation croissait également comme le nombre d'arcs du graphe.

4 - Pourquoi plusieurs algorithmes ?
L'algorithme de Moore-Dijkstra n'est applicable que si les longueurs sont positives.
L'algorithme de Bellman n'est applicable que si le graphe est sans circuit.

Dans l'exemple traité, le graphe possède ces 2 propriétés.
On peut constater que le nombre de calculs à faire est plus faible dans le cas de l'algorithme de Bellman. Quand la taille du graphe augmente, le nombre d'opérations à effectuer croît comme m, le nombre d'arcs du graphe, alors que pour l'algorithme de Moore-Dijkstra ce nombre croît comme $n^2$; on a généralement m nombre d'arcs (beaucoup) plus petit que $n^2$, carré du nombre de sommets. Il vaut donc mieux dans ce cas utiliser l'algorithme de Bellman.

## II Algorithme de détermination des plus courts chemins : cas général

### Algorithme de Ford - Bellman

**Principe**
Une itération de l'algorithme de Ford - Bellman consiste à passer en revue systématiquement tous les sommets et pour chacun d'eux à examiner, par la même méthode que celle utilisée dans l'algorithme de Bellman, son étiquette à partir de celles de ses prédécesseurs.

Si, au cours de cet examen, l'étiquette d'un sommet est modifiée, on **recommence** et ceci tant qu'un des sommets aura eu son étiquette modifiée.
Cet algorithme est donc une généralisation de l'algorithme de Bellman. 0n devrait plutôt dire que l'algorithme de Bellman est un cas particulier de cet algorithme général !

**Algorithme de Ford-Bellman : Enoncé**
*Données*
- un graphe valué dont les sommets sont numérotés de 1 à n,
- le sommet de départ s est racine du graphe ; il est numéroté 1,
- on connaît pour chaque sommet x ses prédécesseurs pred(x).

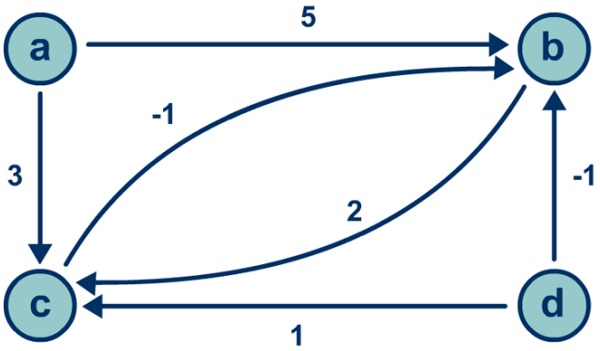
*Initialisation*
Poser $\lambda(1) = 0$
POUR x de 2 à n
  Poser $\lambda(x) = +\infty$ et père(x) = vide
FINPOUR
Poser fin : = FAUX

*Corps de l'algorithme*
TANTQUE fin = FAUX
  POUR x de 2 à n *(on passe en revue tous les sommets)*
    FAIRE *(on examine les prédécesseurs du sommet x)*
      Calculer min ( $\lambda(y)$ + l(y,x)) pour y dans Pred(x)
      SI $\lambda(x)$ > min ( $\lambda(y)$ + l(y,x)) ALORS
        Poser $\lambda(x)$ = min ( $\lambda(y)$ + l(y,x))
        Poser père(x) = y avec y prédécesseur y de x pour lequel ce minimum est atteint
    FINFAIRE
  FINPOUR
SI aucune étiquette modifiée ALORS Poser fin : = VRAI
FINTANTQUE
FIN

Par rapport à l'algorithme de Bellman, on ajoute une variable qui indique la nécessité de recommencer l'examen de l'ensemble des sommets si au moins l'un d'eux a eu son étiquette modifiée.

***Exemple***

On recherche les plus courts chemins à partir du sommet a.




Ce graphe possède un circuit, ainsi que des arcs de longueur négative. On ne peut donc appliquer ni l'algorithme de Bellman ni l'algorithme de Moore-Dijkstra.

| | a<br>pred(a) = Ø | b<br>pred(b) = {a, c, d} | c<br>pred(c) ={a, b} | d<br>pred(d) = {c} |
|---|---|---|---|---|
| *Initialisation* | 0 | + ∞ | + ∞ | + ∞ |
| *Itération 1* | // | 5 (a) | 3 (a) | 4 (c) |
| *Itération 2* | | 2 (c) | // | // |
| *Itération 3* | | // | // | // |

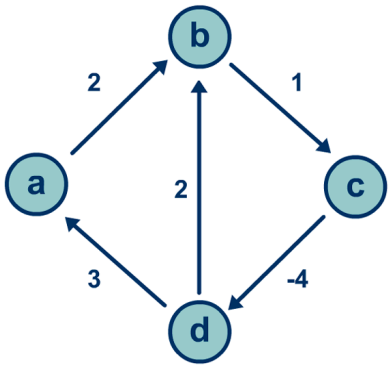
A chaque itération on parcourt tous les sommets ; pour chacun d'eux on compare son étiquette à celles de ses prédécesseurs.
Par exemple, à l'itération 2, b ayant comme prédécesseurs les 3 sommets a, c et d, on calcule :
$\lambda(b) = \min(\lambda(a) + l(a, b),\ \lambda(c) + l(c, b),\ \lambda(d) + l(d, b)) = \min(0 + 5, 3 - 1, 4 - 1) = \min(5, 2, 3) = 2$ avec père(b) = c.

A l'itération 2, il existe au moins un sommet dont l'étiquette a encore été modifiée.
On recommence l'examen des 3 sommets b puis c puis d.
Il n'y a plus de changement. Donc on arrête l'algorithme.

On a ainsi les longueurs des plus courts chemins.
Pour avoir les plus courts chemins eux-mêmes, on procède comme dans les algorithmes déjà vus en remontant de père en père.
*Plus court chemin de a à b* : on part de b puis c puis a donc a c b de longueur 2.
*Plus court chemin de a à c* : a c de longueur 3.
*Plus court chemin de a à d* : d puis c puis a donc plus court chemin a c d de longueur 4.

**Commentaires sur l'algorithme de Ford-Bellman**
*Convergence de l'algorithme de Ford-Bellman*

***Exemple***
Considérons l'exemple suivant.
On recherche les plus courts chemins à partir du sommet a.




| | a<br>pred(a) = d | b<br>pred(b) = {a, d} | c<br>pred(c) = {b} | d<br>pred(d) = {c} |
|---|---|---|---|---|
| *Initialisation* | 0 | + ∞ | + ∞ | + ∞ |
| *Itération 1* | // | 2 (a) | 3 (b) | -1 (c) |
| *Itération 2* | | 1 (d) | 2 (b) | -2 (c) |
| *Itération 3* | | 0 (d) | 1 (b) | -3 (c) |

On s'aperçoit ici qu'on ne pourra jamais avoir les marques inchangées. Les marques des sommets b, c et d diminuent de 1 à chaque itération.

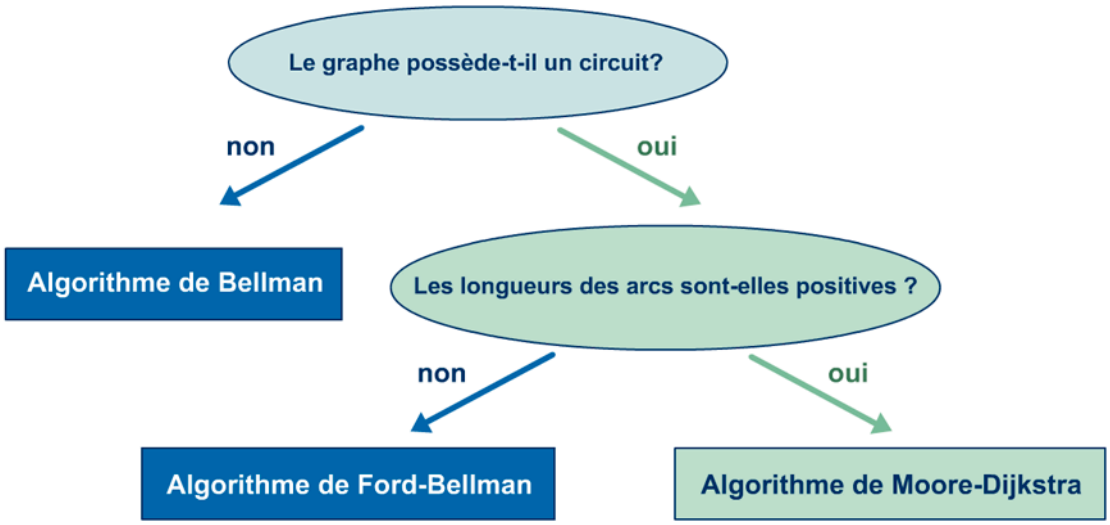
Ceci est du à la présence du circuit b, c, d, b de longueur - 1 donc négative. C'est un circuit dit "absorbant".
En empruntant ce circuit autant de fois que l'on veut, la longueur des chemins peut tendre vers moins l'infini !

Dans ce cas, l'algorithme de Ford-Bellman ne converge pas.
On peut démontrer que si, après avoir fait un nombre d'itérations de l'algorithme de Ford-Bellman égal au nombre de sommets du graphe, il y a encore des modifications, il faut arrêter le déroulement de l'algorithme ; il existe sûrement un circuit de longueur négative dans le graphe et le problème du plus court chemin n'a pas de solution.

*Efficacité de l'algorithme de Ford-Bellman*
Chaque itération de l'algorithme de Ford-Bellman est identique à l'unique itération de l'algorithme de Bellman : le nombre d'opérations correspondant croît donc comme le nombre d'arcs du graphe.
Quoiqu'il arrive, d'après ce qu'on vient de voir au point précédent, on ne fait pas plus de n itérations avec n égal au nombre de sommets du graphe.
Globalement, le nombre d'opérations se comportera donc comme la fonction m*n lorsque la taille du problème deviendra "grande".

## III CONCLUSION

**Quel algorithme choisir pour un problème de plus court chemin ?**




Ce diagramme situe les algorithmes que nous venons de voir. Le critère de choix est lié à leur efficacité en fonction des propriétés potentielles du graphe considéré.

Signalons qu'il existe d'autres algorithmes pour résoudre ce type de problèmes, par exemple des algorithmes de type matriciel qui permettent d'obtenir les plus courts chemins entre n'importe quel couple de sommets.

**Le problème du plus long chemin dans un graphe**
Jusqu'à présent on ne s'est intéressé qu'à la détermination de plus courts chemins.
Certains problèmes sont modélisés par le problème de recherche de plus long chemin dans un graphe comme on le verra à la leçon suivante.
La modification des algorithmes précédents est généralement immédiate. Par exemple, dans l'algorithme de Bellman, il suffit de remplacer les min par max.

Dans le cas général, on modifie l'algorithme de Ford-Bellman de la manière suivante :

Calculer min ( $\lambda(y) + l(y, x)$) pour y dans Pred(x)
SI $\lambda(x) >$ min ( $\lambda(y) + l(y, x)$) ALORS
Poser $\lambda(x) =$ min ( $\lambda(y) + l(y, x)$)

est remplacé par :

Calculer **max** ( $\lambda(y) + l(y, x)$) pour y dans Pred(x)
SI $\lambda(x) <$ **max** ( $\lambda(y) + l(y, x)$) ALORS
Poser $\lambda(x) =$ min ( $\lambda(y) + l(y, x)$)

mais attention s'il existe un circuit de longueur **positive** il n'y a pas de solution.

***Exemple***

On reprend le graphe précédent. Il est sans circuit. On peut déterminer les plus longs chemins par l'algorithme de Bellman adapté à un problème de plus long chemin.
On obtient :