

## CM 02 - Algorithmes de plus courts chemins (DIJKSTRA et BELLMAN-FORD)

---

De nombreux problèmes concrets peuvent se modéliser par la recherche de plus courts chemins dans un graphe valué. Par exemple :

- déterminer l'itinéraire le plus rapide entre deux villes (voiture),
- entre deux stations (métro),
- entre deux arrêts (bus)
- ou bien effectuer le routage dans des réseaux de télécommunication.

### Définitions.

- Soit  $G = (S, A)$  un graphe orienté (ou non) valué par une fonction de poids  $w : A \rightarrow \mathbb{R}$ .

- **Le poids d'un chemin**  $P = (s_0, s_1, s_2 \dots s_k)$  est égal à la somme des poids des arcs/arêtes composant le chemin/chaîne,

$$\text{i.e. } w(P) = \sum_{i=1}^k w(s_{i-1}, s_i)$$

### Existence.

Etant donnés deux sommets  $s$  et  $t$ , trois cas peuvent se présenter :

- il n'y a pas de chemins de  $s$  à  $t$
- il existe des chemins de  $s$  à  $t$  mais pas de plus court
- il existe un ou plusieurs plus courts chemins de  $s$  à  $t$

S'il existe un plus court chemin de  $s$  à  $t$ , alors le poids de ce chemin est défini par

$$\text{pcc}(s, t) = \min \{w(P) \mid P \text{ chemin de } s \text{ à } t\}$$

*Différentes variantes du problème.*

- Origine unique  $s$

Trouver le plus court chemin d'un sommet source  $s$  à tous les autres sommets

→ algorithme de DIJKSTRA / algorithme de BELLMAN-FORD

- Couple unique  $(s, t)$

Trouver un plus court chemin entre deux sommets donnés  $s$  et  $t$

→ algorithme  $A^*$

→ résoudre le problème "origine unique" et en déduire le PCC de  $s$  à  $t$

- Tous couples (calcul d'un distancier)

Trouver les plus courts chemins entre tous les couples de sommets de  $S$

→ algorithme de FLOYD-WARSHALL

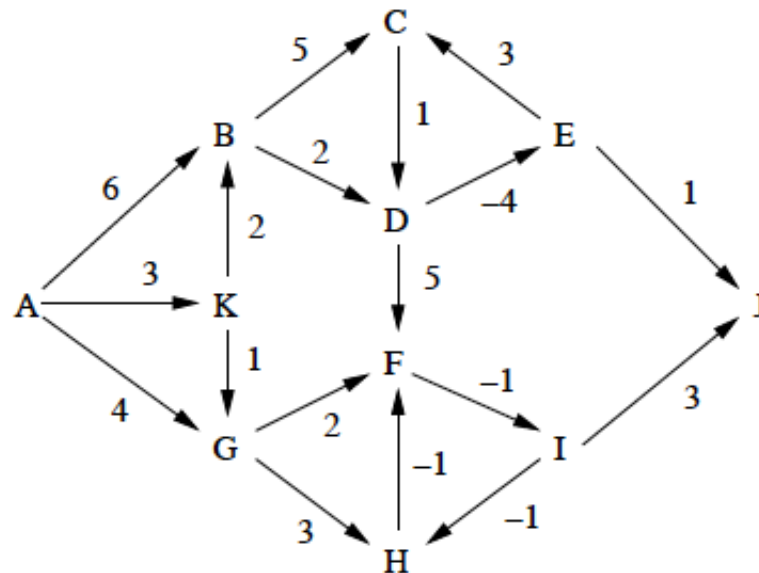
---

## **Plan de la présentation**

- Existence d'un ou plusieurs plus courts chemins entre deux sommets
  - Origine unique : principes communs à DIJKSTRA et BELLMAN-FORD
  - Algorithme de DIJKSTRA
  - Algorithme de BELLMAN-FORD
  - Arborescence des plus courts chemins issus d'un sommet source
-

## Existence d'au moins un plus court chemin entre deux sommets

*Exemple #1.*

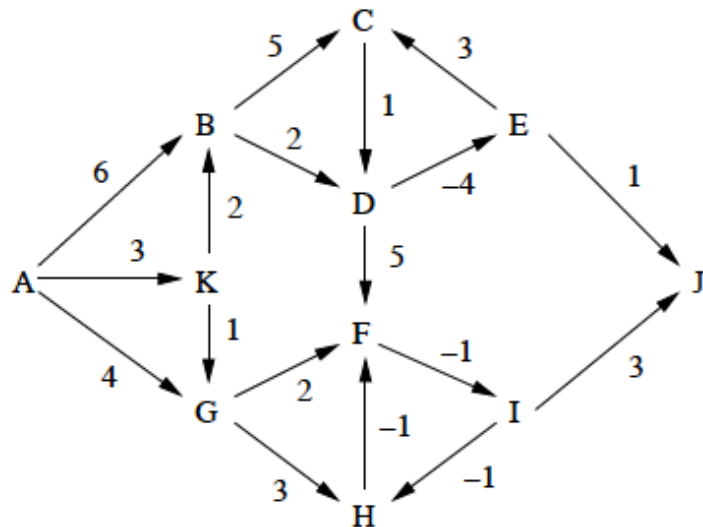


De  $A$  à  $B$  : il existe un unique plus court chemin  $(A, K, B)$ .

De  $A$  à  $G$  : il existe deux plus courts chemins  $(A, K, G)$  et  $(A, G)$ .

De  $E$  à  $A$  : il n'existe pas de chemins, donc pas de plus courts chemins.

**Exemple 2.**



De  $A$  à  $E$  : il existe une infinité de plus courts chemins :  $(A, K, B, D, E)$ ,  $(A, K, B, D, E, C, D, E)$ ,  $(A, K, B, D, E, C, D, E, \dots, C, D, E), \dots$

De  $A$  à  $J$  : il existe des chemins mais pas de plus court : les chemins  $(A, G, H, F, I, H, F, I, \dots, H, F, I, J)$  sont arbitrairement courts.

**Définition.** Un **circuit absorbant** est un circuit de poids strictement négatif.

Si un graphe possède un circuit absorbant, alors il n'existe pas de plus courts chemins entre certains de ses sommets.

En effet,  
si deux sommets  $s$  et  $t$  appartiennent à un circuit de poids strictement négatif, alors  $w(s, t)$  n'est pas défini.

**Théorème.**

Soit  $G$  un graphe orienté (ou pas) n'ayant pas de circuit absorbant.

Soient  $s$  et  $t$  deux sommets de  $G$ .

S'il existe un chemin allant de  $s$  à  $t$ , alors il existe (au moins) un plus court chemin de  $s$  à  $t$ .

## Origine unique : principes communs à DIJKSTRA et BELLMAN-FORD

Deux algorithmes permettent de résoudre les problèmes de recherche de plus courts chemins à origine unique :

- l'algorithme de DIJKSTRA lorsque tous les poids sont positifs ou nuls,
- l'algorithme de BELLMAN-FORD lorsque les poids sont positifs, nuls ou négatifs, sous réserve qu'il n'y ait pas de circuit absorbant (de poids strictement négatif).

Ces deux algorithmes procèdent de manière voisine. L'idée est d'associer à chaque sommet  $s_i$  de  $S$  une valeur  $\text{DistSource}[s_i]$  qui représente un *majorant* du poids d'un plus court chemin entre  $s$  et  $s_i$ .

(1) Au départ,

$\text{DistSource}[s]$  vaut 0 ;

$\text{DistSource}[s_i]$  vaut  $+\infty$  pour tout sommet  $s_i \neq s$  ;

Le sommet source  $s$  ne possède pas de prédécesseur.

(2) L'algorithme diminue alors progressivement les valeurs  $\text{DistSource}[s_i]$  associées aux différents sommets  $s_i$ ,

(3) jusqu'à ce qu'on ne puisse plus les diminuer, i.e.  $\text{DistSource}[s_i] = \text{pcc}(s, s_i)$ .

Pour diminuer les valeurs, on va itérativement examiner chaque arc  $(s_i \rightarrow s_j)$  du graphe, et regarder si, en passant par  $s_i$ , on ne peut diminuer la valeur de  $\text{DistSource}[s_j]$ .

Pour diminuer les valeurs, on va itérativement examiner chaque arc ( $s_i \rightarrow s_j$ ) du graphe, et regarder si, en passant par  $s_i$ , on ne peut diminuer la valeur de  $\text{DistSource}[s_j]$ .

Cette opération de diminution du majorant est appelée “relâchement” de l’arc ( $s_i \rightarrow s_j$ )

---

```
procédure Relacher ( $s_i \rightarrow s_j$  : arc) ;
début
    si  $\text{DistSource}[s_i] + w(s_i, s_j) < \text{DistSource}[s_j]$ 
        alors      % mieux vaut passer par  $s_i$  pour aller en  $s_j$ 
             $\text{DistSource}[s_j] := \text{DistSource}[s_i] + w(s_i, s_j)$  ;
             $\text{PRED}[s_j] := s_i$ 
        fin si
    fin
```

---

Les algorithmes de DIJKSTRA et BELLMAN-FORD procèdent tous les deux par relâchements successifs d’arcs, mais :

- pour DIJKSTRA, chaque arc est relâché une fois et une seule (*algorithme à fixation d’étiquettes*),
- pour BELLMAN-FORD, chaque arc peut être relâché plusieurs fois (*algorithme à correction d’étiquettes*).

## Algorithme de DIJKSTRA

- origine unique : trouver les plus courts chemins d'un sommet  $s$  à tous les autres sommets de  $S$
- la fonction de poids  $w$  est à valeurs positives (et donc absence de circuit/cycle absorbant)

### *Pseudo-code.*

---

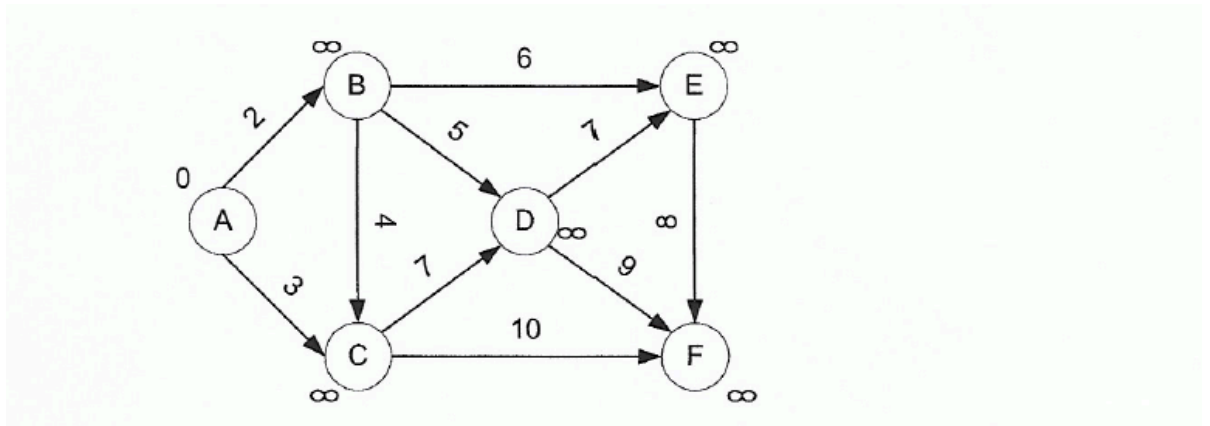
```
Procédure DIJKSTRA (G : graphe valué, s : sommet source) ;  
début  
  DistSource[s] := 0 ;  
  pour tout sommet  $u \neq s$  faire DistSource[u] := + inf ;  
  pour tout sommet  $u$  faire PRED[u] := nil ;  
  
  tantque les sommets ne sont pas tous marqués faire  
    soit  $t_0$  un sommet non marqué tq DistSource[ $t_0$ ] soit minimale ;  
    marquer le sommet  $t_0$  ;  
    pour tout successeur non marqué  $t$  de  $t_0$  faire Relacher( $t_0 \rightarrow t$ ) ;  
  fttque  
  
  fin  
  
procédure Relacher (arc  $u \rightarrow v$ ) ;  
début  
  si DistSource[v] > DistSource[u] +  $w(u, v)$   
    alors  
      DistSource[v] := DistSource[u] +  $w(u, v)$  ;  
      PRED[v] := u ;  
  fsi  
  
  fin
```

---



**Exemple.**

Soit **A** le sommet source.



Rappel du pseudo-code de l'initialisation

```

DistSource[s] := 0 ;
pour tout sommet u ≠ s faire DistSource[u] := + inf ;
pour tout sommet u faire PRED[u] := nil ;

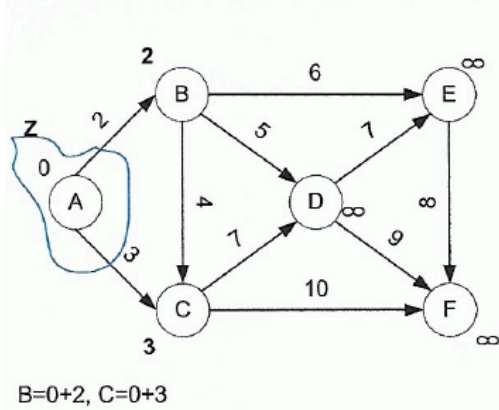
```

Tableau DistSource noté D

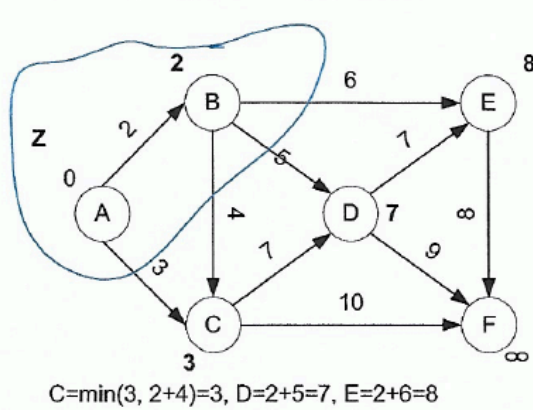
Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
PRED	A	B	C	D	E	F
	nil	nil	nil	nil	nil	nil

### Exemple (suite 1)

On absorbe A avec une mise à jour de B et C



On absorbe B, et on marque C, D, et E



(à gauche) Sélection du sommet **A** qui a les sommets **B** et **C** comme successeurs

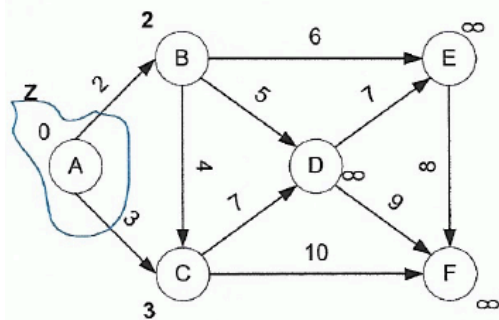
Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
<b>{A}</b>	0	<b>2</b>	<b>3</b>	inf	inf	inf

PRED	A	B	C	D	E	F
	nil	nil	nil	nil	nil	nil
	nil	<b>A</b>	<b>A</b>	nil	nil	nil

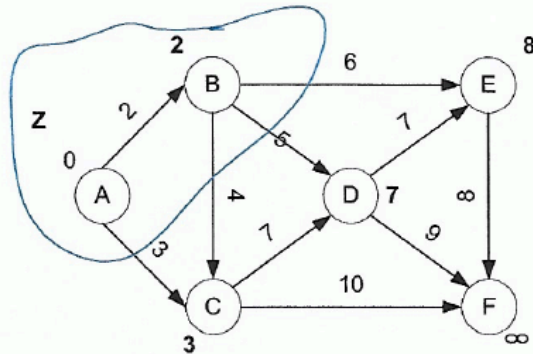
## Exemple (suite 2)

On absorbe A avec une mise à jour de B et C



$B=0+2$ ,  $C=0+3$

On absorbe B, et on marque C, D, et E



$C=\min(3, 2+4)=3$ ,  $D=2+5=7$ ,  $E=2+6=8$

(à

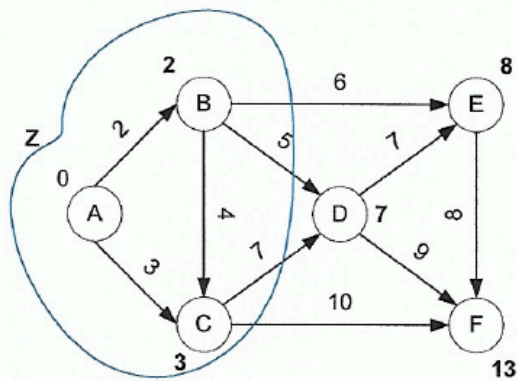
(à droite) Sélection du sommet **B** ayant les sommets **C**, **D** et **E** comme successeurs

Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
{A}	0	2	3	inf	inf	inf
<b>{A, B}</b>	0	2	<b>3*</b>	<b>7</b>	<b>8</b>	inf

PRED	A	B	C	D	E	F
nil	nil	nil	nil	nil	nil	nil
nil	A	A	nil	nil	nil	nil
nil	A	A	<b>B</b>	<b>B</b>	nil	nil

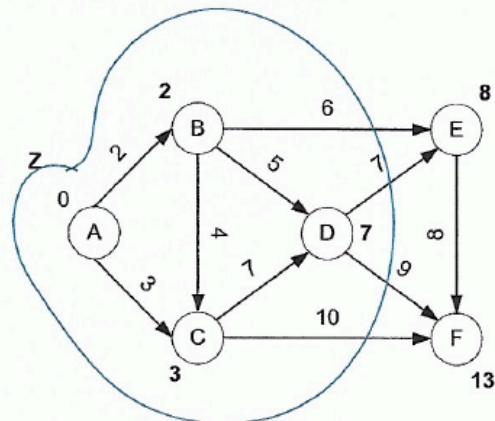
### Exemple (suite 3)

On absorbe C et on marque D et F



$$D = \min(7, 3+7) = 7, F = 3+10 = 13$$

On absorbe D et on marque E et F



$$E = \min(8, 7+7) = 8, F = \min(13, 7+9) = 13$$

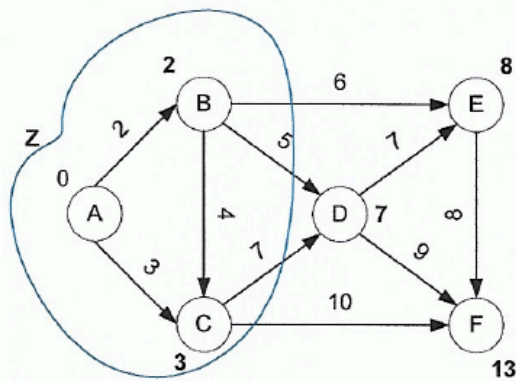
(à gauche) Sélection du sommet C ayant les sommets D et F comme successeurs

Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
{A}	0	2	3	inf	inf	inf
{A, B}	0	2	3	7	8	inf
{A, B, C}	0	2	3	7 *	8	13

PRED	A	B	C	D	E	F
	nil	nil	nil	nil	nil	nil
	nil	A	A	nil	nil	nil
	nil	A	A	B	B	nil
	nil	A	A	B	B	C

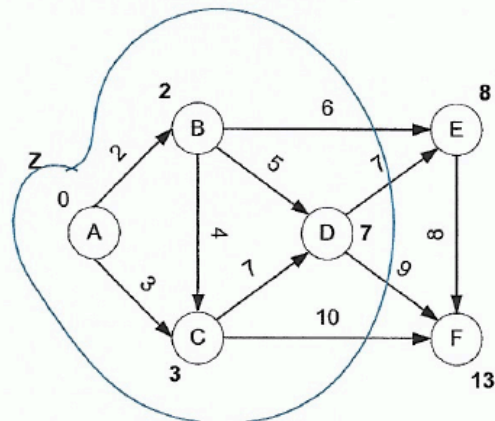
### Exemple (suite 4)

On absorbe C et on marque D et F



$$D = \min(7, 3+7) = 7, F = 3+10 = 13$$

On absorbe D et on marque E et F



$$E = \min(8, 7+7) = 8, F = \min(13, 7+9) = 13$$

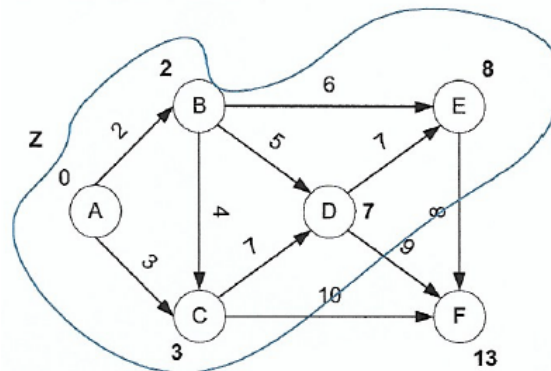
(à droite) Sélection du sommet **D** ayant les sommets **E** et **F** comme successeurs

Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
{A}	0	2	3	inf	inf	inf
{A, B}	0	2	3	7	8	inf
{A, B, C}	0	2	3	7	8	13
<b>{A, B, C, D}</b>	0	2	3	7	<b>8 *</b>	<b>13 *</b>

PRED	A	B	C	D	E	F
	nil	nil	nil	nil	nil	nil
	nil	A	A	nil	nil	nil
	nil	A	A	B	B	nil
	nil	A	A	B	B	C

### Exemple (suite 5)

On absorbe E et on marque F



$$F = \min(13, 8+8) = 13$$

Sélection du sommet **E** ayant le sommet **F** comme successeur

Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
{A}	0	2	3	inf	inf	inf
{A, B}	0	2	3	7	8	inf
{A, B, C}	0	2	3	7	8	13
{A, B, C, D}	0	2	3	7	8	13
<b>{A, B, C, D, E}</b>	0	2	3	7	8	<b>13 *</b>

PRED	A	B	C	D	E	F
	nil	nil	nil	nil	nil	nil
	nil	A	A	nil	nil	nil
	nil	A	A	B	B	nil
	nil	A	A	B	B	C

La sélection du sommet **F** n'ayant pas de successeurs ne change rien.

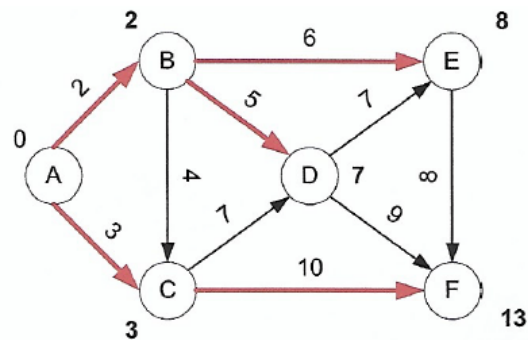
### Exemple (suite et fin)

Valeurs finales :

Visités	D[A]	D[B]	D[C]	D[D]	D[E]	D[F]
{}	0	inf	inf	inf	inf	inf
{A}	0	2	3	inf	inf	inf
{A, B}	0	2	3	7	8	inf
{A, B, C}	0	2	3	7	8	13
{A, B, C, D}	0	2	3	7	8	13
{A, B, C, D, E}	0	2	3	7	8	13
{A, B, C, D, E, F}	0	2	3	7	8	13

PRED	A	B	C	D	E	F
	nil	nil	nil	nil	nil	nil
	nil	A	A	nil	nil	nil
	nil	A	A	B	B	nil
	nil	A	A	B	B	C

### Arbre des plus courts chemins issus de A

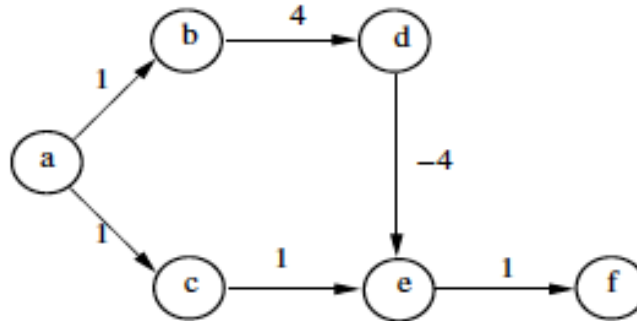


De manière intuitive,

- Pourquoi les PCC obtenus par DIJKSTRA forment-ils un arbre ?
- Cet arbre est-il unique ?

### Quid de DIJKSTRA en présence de poids négatifs ?

On considère le graphe orienté valué  $G$  ci-dessous dont les sommets sont  $\{a, b, c, d, e, f\}$ .



- Appliquer l'algorithme de DIJKSTRA au graphe  $G$  avec comme sommet source  $a$ .
- Quel est le chemin (obtenu par l'algorithme de DIJKSTRA) entre le sommet source  $a$  et le sommet  $f$  ? Est-il optimal ? Justifier.

→ Pour aller de  $a$  à  $f$ , DIJKSTRA donne  $a$ - $c$ - $e$ - $f$  de poids 3

→ Le PCC de  $a$  à  $f$  est  $a$ - $b$ - $d$ - $e$ - $f$  de poids 2



## Algorithme de BELLMAN-FORD

L'algorithme de BELLMAN-FORD permet de trouver les plus courts chemins à origine unique dans le cas où le graphe contient des arcs dont le poids est négatif, sous réserve que le graphe ne contienne pas de circuit absorbant (dans ce cas, l'algorithme détecte l'existence de circuits absorbants).

*De manière analogue à DIJKSTRA*, on associe à chaque sommet  $u$  une valeur  $\text{DistSource}[u]$  qui représente un majorant du poids du plus court chemin entre  $s$  et  $u$ . L'algorithme diminue de manière progressive ces valeurs en relâchant les arcs.

*Contrairement à DIJKSTRA*, chaque arc va être relâché plusieurs fois : on relâche une première fois tous les arcs ; tous les plus courts chemins de longueur 1, issus de  $s$ , sont trouvés. On relâche alors une 2<sup>de</sup> fois tous les arcs ; tous les plus courts chemins de longueur 2, issus de  $s$ , sont trouvés, et ainsi de suite ...

- Si le graphe ne comporte pas de circuit absorbant, un plus court chemin sera de longueur inf. ou égale à  $(n-1)$ . Donc, au bout de  $(n-1)$  passages, on aura trouvé tous les plus courts chemins issus de  $s$ .

- Si le graphe possède un circuit absorbant, après  $(n-1)$  passages, il restera au moins un arc ( $u \rightarrow v$ ) dont le relâchement permettrait de diminuer la valeur de  $\text{DistSource}[v]$ . L'algorithme utilise cette propriété pour détecter la présence de circuits absorbants.

### *Pseudo-code (v1)*

---

Procédure BELLMAN-FORD\_v1 ( $G=(S, A)$  : graphe valué,  $s$  : sommet source) ;

début

$\text{DistSource}[s] := 0$  ;

pour tout sommet  $u \neq s$  faire  $\text{DistSource}[u] := +\infty$  ;

pour tout sommet  $u$  faire  $\text{PRED}[u] := \text{nil}$  ;

pour  $k$  allant de 1 à  $(n-1)$  faire

pour tout arc ( $u \rightarrow v$ ) de  $A$  faire Relacher( $u \rightarrow v$ ) ;

fpour

si existeCycleAbsorbant( $G$ ) alors Afficher(« existence d'un circuit absorbant »)

fin

---

## Algorithme de BELLMAN-FORD

### *Pseudo-code (v1)*

---

```
Procédure BELLMAN-FORD_v1 (G : graphe valué, s : sommet source) ;  
  début  
    DistSource[s] := 0 ;  
    pour tout sommet u ≠ s faire DistSource[u] := + inf ;  
    pour tout sommet u faire PRED[u] := nil ;  
    pour k allant de 1 à (n-1) faire  
      pour tout arc (u -> v) de A faire Relacher(u -> v) ;  
    fpour  
    si existeCycleAbsorbant(G) alors Afficher(« existence d'un circuit absorbant »)  
  fin
```

```
Fonction existeCycleAbsorbant(G : graphe) → Booléen ;  
  début  
    Existe := False ;  
    pour tout arc (u -> v) de A faire  
      si DistSource [v] > DistSource [u] + w(u, v) alors Existe := True  
    fpour  
    retourner Existe  
  fin
```

---

(1) Justifier la définition de la fonction `existeCycleAbsorbant(G : graphe) → Booléen` ;

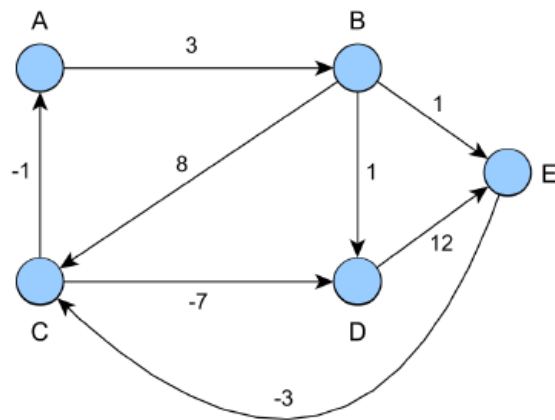
### *(2) Améliorations possibles.*

**(v2)** Mémoriser, lors de chaque itération, l'ensemble des sommets  $t$  pour lesquels  $\text{DistSource}[t]$  a été modifiée, afin de ne relâcher que les arcs partant de ces sommets lors de l'itération suivante.

**(v3)** Arrêter l'algorithme dès qu'aucune valeur  $\text{DistSource}[t]$  n'a été modifiée lors d'une itération.

## Exemple #2 : BELLMAN-FORD sans circuit absorbant

On considère le graphe orienté G ci-dessous avec A comme sommet source.



(init) On effectue la phase d'initialisation

(Iter=1) On envisage les chemins de longueur 1 issus du sommet source A.

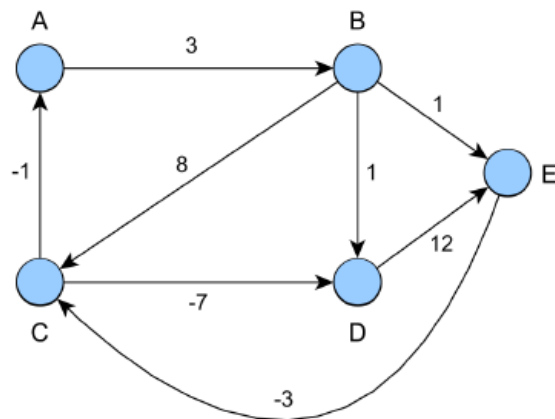
**DistSource [1..5]**

	A	B	C	D	E
init	0	inf	inf	inf	inf
1		3			
2					
3					
4					

**PRED [1..5]**

	A	B	C	D	E
init	nil	nil	nil	nil	nil
1		A			
2					
3					
4					

## Exemple #2 : BELLMAN-FORD sans circuit absorbant (suite)



(Iter=2)

On continue avec les chemins de longueur 2 issus du sommet source A.

Pour cela, on envisage les successeurs de B, i.e. C, puis D, puis E.

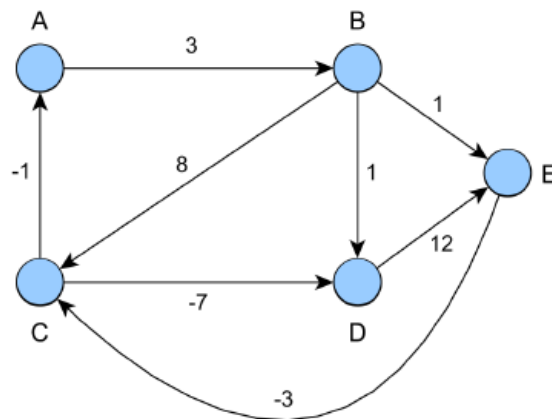
**DistSource [1..5]**

	A	B	C	D	E
init	0	inf	inf	inf	inf
1		3			
2			<b>11</b>	<b>4</b>	<b>4</b>

**PRED [1..5]**

	A	B	C	D	E
init	nil	nil	nil	nil	nil
1		A			
2			<b>B</b>	<b>B</b>	<b>B</b>

## Exemple #2 : BELLMAN-FORD sans circuit absorbant (suite)



(Iter=3) On continue avec les chemins de longueur 3 issus du sommet source A.

Pour cela, on envisage successivement

- les successeurs de C (sommet D) → ABCD poids 4
- puis les successeurs de D (sommet E) → ABDE poids 16
- puis les successeurs de E (sommet C) → ABEC poids 1 (seule amélioration)

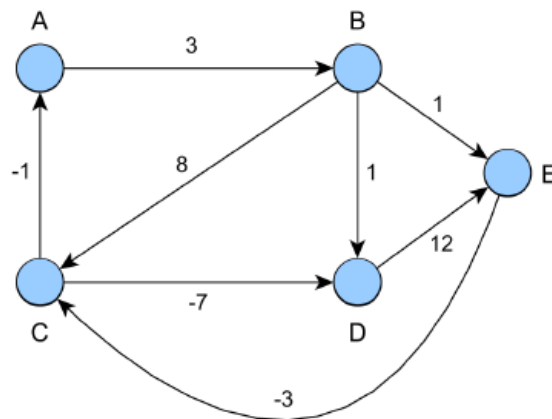
**DistSource [1..5]**

	A	B	C	D	E
init	0	inf	inf	inf	inf
1		3			
2			11	4	4
3			1		

**PRED [1..5]**

	A	B	C	D	E
init	nil	nil	nil	nil	nil
1		A			
2			B	B	B
3			E		

## Exemple #2 : BELLMAN-FORD sans circuit absorbant (suite)



(Iter=4) On termine avec les chemins de longueur 4 issus de A  $\rightarrow$  ABECD de poids -6 (seule amélioration)

**DistSource [1..5]**

	A	B	C	D	E
init	0	inf	inf	inf	inf
1		3			
2			11	4	4
3			1		
4				-6	

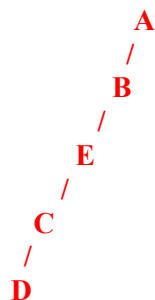
**PRED [1..5]**

	A	B	C	D	E
init	nil	nil	nil	nil	nil
1		A			
2			B	B	B
3			E		
4				C	

Il existe toujours un PCC entre 2 sommets car le graphe ne possède aucun circuit absorbant.

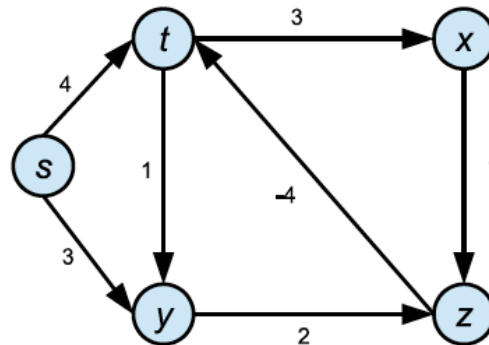
Donc, les PCC issus de la source forment un arbre.

Ici, il se trouve qu'ici l'arbre est un peu « particulier » car chaque nœud (sauf la racine A) possède un unique fils.



### Exemple #3 : BELLMAN-FORD en présence d'un circuit absorbant

On considère le graphe orienté valué suivant :



1. Montrer que ce graphe possède un circuit absorbant

2. Appliquer l'algorithme de BELLMAN-FORD sur ce graphe

Contrairement à DIJKSTRA, chaque arc va être relâché plusieurs fois : on relâche une première fois tous les arcs ; tous les plus courts chemins de longueur 1, issus de s, sont trouvés.

On relâche alors une 2<sup>de</sup> fois tous les arcs ; tous les plus courts chemins de longueur 2, issus de s, sont trouvés, et ainsi de suite .../...

3. Ce graphe possède-t-il un arbre de PCC ?

#### Eléments de solution

(1) Iter=0 représente la phase d'initialisation.

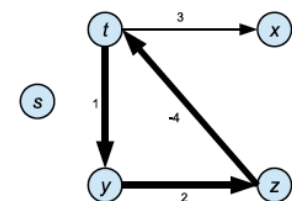
(2) Evolution des 2 tableaux au fur et à mesure des 4 (5-1) itérations

Iter	s	t	y	x	z
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0	4	3	$+\infty$	$+\infty$
2	0	4	3	7	5
3	0	1	3	7	5
4	0	1	2	4	5

DistSource[u]

s	t	y	x	z
-1	z	t	t	y

PRED[u]



Circuit absorbant (poids -1)

## Arbres des plus courts chemins issus d'un sommet source

Tableau PRED[1..n] des prédécesseurs :

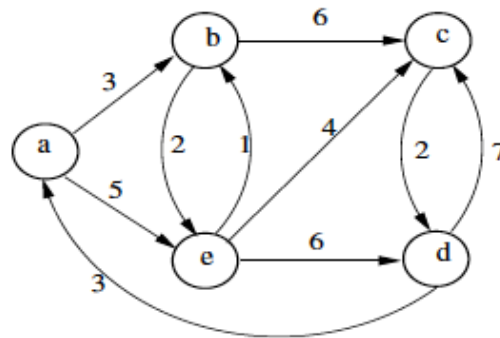
PRED[s] = nil ;

PRED[s<sub>j</sub>] = s<sub>i</sub> ssi l'arc (s<sub>i</sub> -> s<sub>j</sub>) appartient à l'arborescence.

Remarque

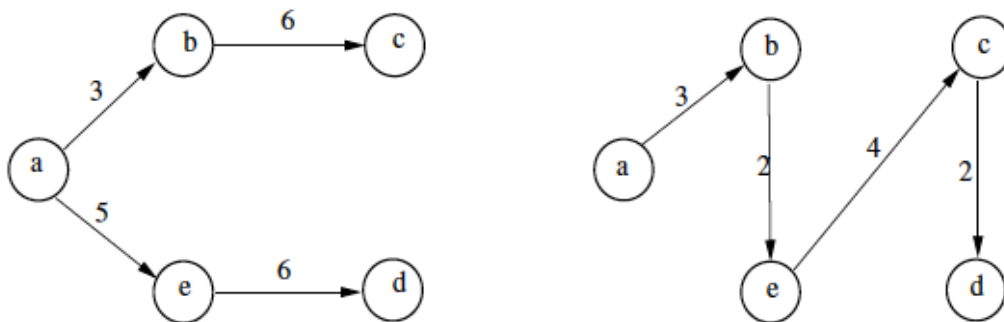
$$pcc(s, t) = pcc(s, PRED[t]) + w(PRED[t], t)$$

L'arbre des plus courts chemins issus d'un sommet source s n'est pas forcément unique, comme le montre le graphe orienté valué ci-dessous en prenant comme sommet source a.



Graphe orienté valué

Il y a 2 PCC différents entre a et e ; considérer l'arc (a -> e) et le chemin (a -> b -> e)



Deux arbres distincts des plus courts chemins issus du sommet source a



## Arbres des plus courts chemins issus d'un sommet source

Tableau  $PRED[1..n]$  des prédécesseurs :

$PRED[s] = nil$  ;

$PRED[s_j] = s_i$  ssi l'arc  $(s_i \rightarrow s_j)$  appartient à l'arborescence.

Remarque

$$pcc(s, t) = pcc(s, PRED[t]) + w(PRED[t], t)$$

Pour connaître le plus court chemin entre le sommet source  $s$  et un sommet  $t$  donné, il faut “remonter” du sommet  $t$  jusqu'au sommet  $s$  en utilisant l'algorithme suivant :

---

Procédure AfficherPlusCourtChemin( $t$  sommet de  $G$ ) ;

début

    si  $t = s$

        alors afficher( $t$ )

    sinon

        AfficherPlusCourtChemin( $Pred[t]$ ) ;

        Afficher( $t$ )

    fsi

fin

---

## Synthèse

### (1) Tous les poids sont $\geq 0$

S'il existe (au moins) un chemin entre 2 sommets, alors il existe (au moins) un PCC entre ces 2 sommets. (cf exemple page 24)

Utiliser l'algorithme de DIJKSTRA

### (2) les poids sont quelconques

Le graphe possède un circuit absorbant  $\rightarrow$  il n'existe pas de PCC

Le graphe ne possède aucun circuit absorbant  $\rightarrow$  si il existe un chemin entre 2 sommets, alors il existe un PCC entre ces 2 sommets.

Utiliser l'algorithme de BELLMAN-FORD

## Questions

1. Si tous les poids sont  $\geq 0$ , peut-on utiliser BELLMAN-FORD ? Si oui, quel est l'intérêt de DIJKSTRA ?
2. Si le graphe possède un circuit absorbant, pourquoi BELLMAN-FORD ne boucle-t-il pas ?
3. Peut-il y avoir une infinité de PCC ?

## Consignes pour le TD 02