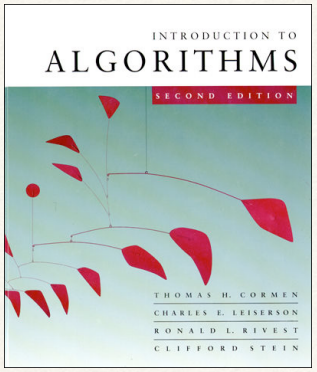
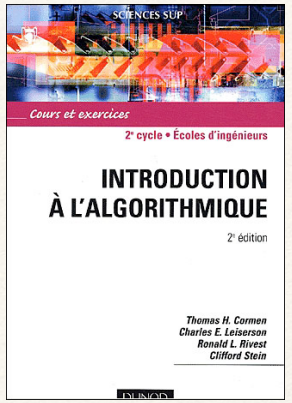




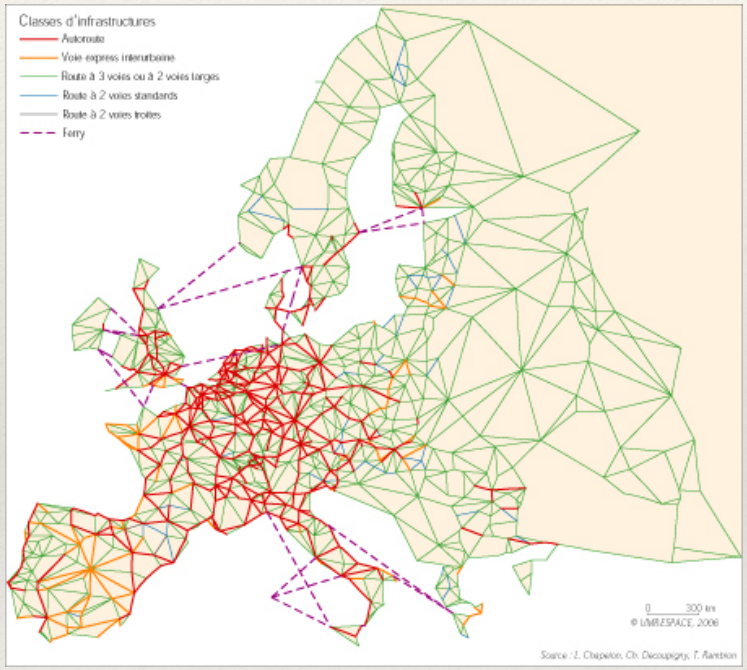
Plus courts chemin

Théorie des graphes

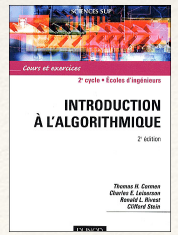
Cours de Lélia Blin

3eme année de Licence

Plus court chemin

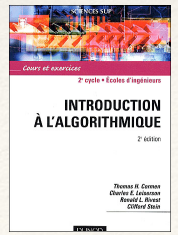


- Graphes valués (Pondérés)
- Chaque arête e est associée à une valeur, appelée souvent son poids
- Notation: w(e).



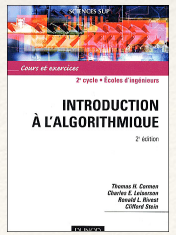
Modélisation

- La valuation des arêtes peut représenter
  - Des coûts de transit,
  - Des distances kilométriques,
  - Des temps de transit,
  - ...



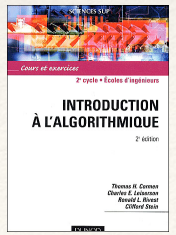
Généralisation

- Le graphe ci-contre peut être valué avec les temps de voyage en voitures entre 2 villes.



Définition: Graphe valué

- On note G=(V,A,w) un graphe simple non orienté valué. V est l'ensemble des sommets du graphe, A l'ensemble des arcs et w une fonction de pondération A→**R** qui fait correspondre à chaque arc un poids à valeur réelle.

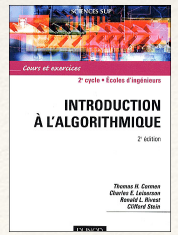


Longueur d'un chemin

- La ***longueur*** du chemin $p = \langle v_0, v_1, \ldots, v_k \rangle$ est la somme des poids (longueurs) des arcs qui le constituent

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \ .$$

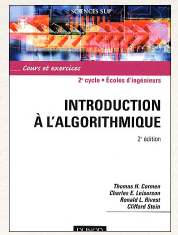
Dans ce qui suit nous appellerons le poids d'un chemin sa longueur.



Plus court chemin

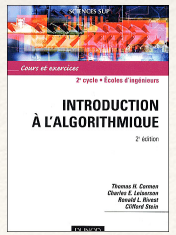
- Le plus court chemin entre 2 sommets u et v est défini comme le chemin de plus faible poids reliant u et v.

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{s'il existe un chemin de } u \text{ à } v \text{ ,} \\ \infty & \text{sinon .} \end{cases}$$



Formalisation

- Soit G=(V,A,w) un graphe orienté avec des coûts (poids) sur ses arcs.
- On note w_{ij} le coût de l'arc (i,j)
- **Définition**:
  - Le coût ou longueur d'un chemin est la somme du coût de ses arcs
- **Définition**:
  - Un chemin est dit élémentaire s'il ne passe pas deux fois par le même sommet

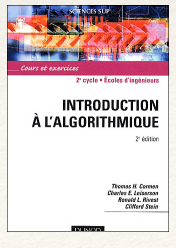


Problèmes

A. Etant donné deux sommets s et t, trouver un plus court chemin (chemin de coût minimal) de s à t.

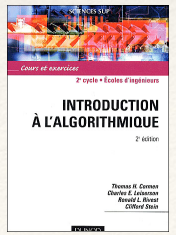
B. Etant donné un sommet de départ s, trouver un plus court chemin de s vers tout autre sommet

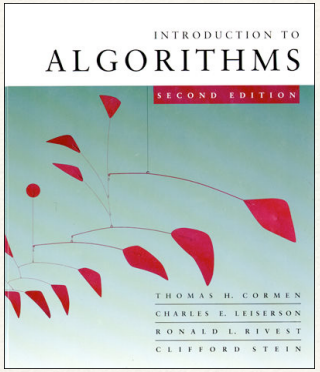
C. Trouver un plus court chemin entre tout couple de sommets

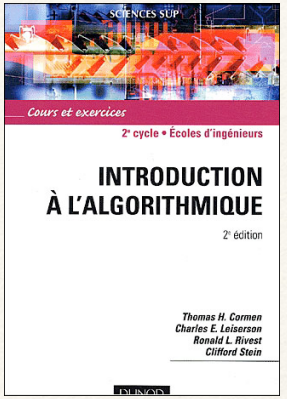


Remarques

- Un algorithme pour le problème A peut servir à résoudre
  - le problème B et le problème C
- le problème B peut servir à résoudre
  - le problème A et le problème C







Lélia Blin

Plus courts chemin à origine unique

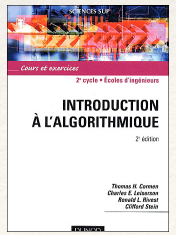
Théorie des graphes

Cours de Lélia Blin

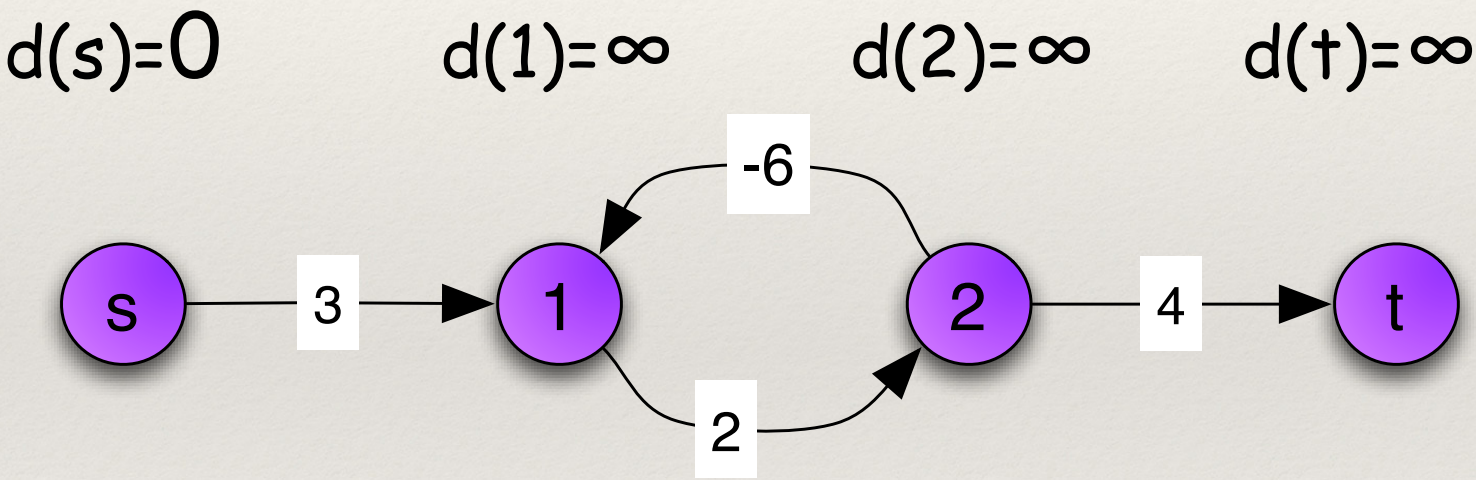
3eme année de Licence

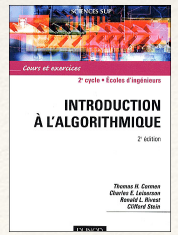
Circuit absorbant

- **Définition**:
  - On appelle circuit absorbant un circuit de coût strictement négatif
- **Proposition**:
  - Il existe un plus court chemin depuis un sommet s jusqu'à tout sommet k si et seulement si il n'existe pas de circuits absorbants dans le graphe.

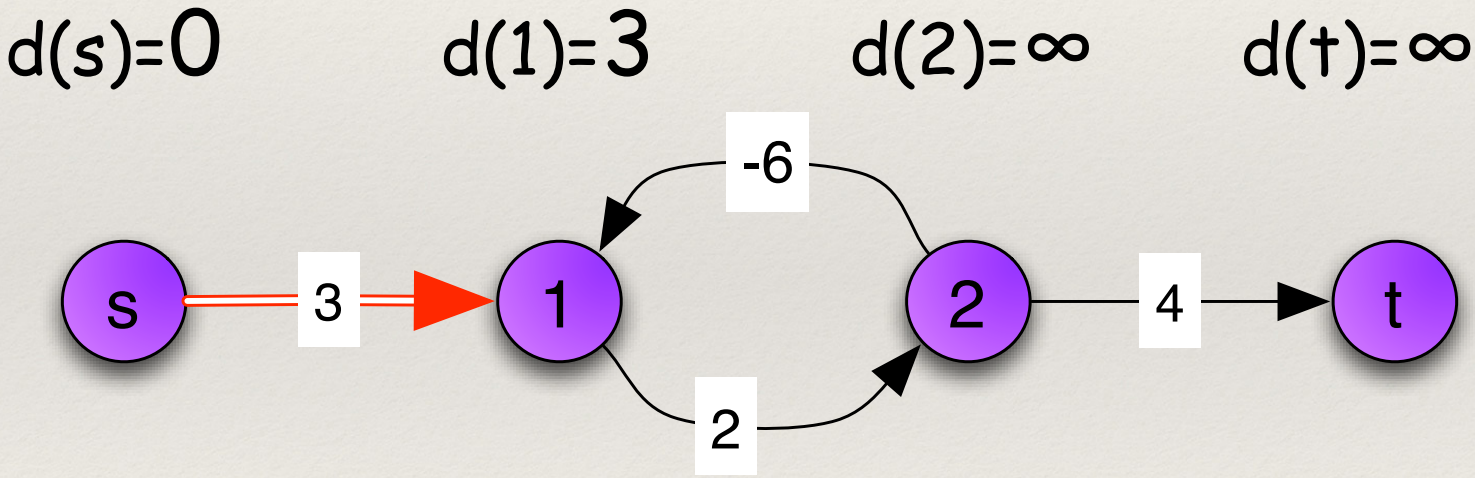


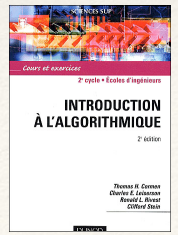
Circuit absorbant exemple



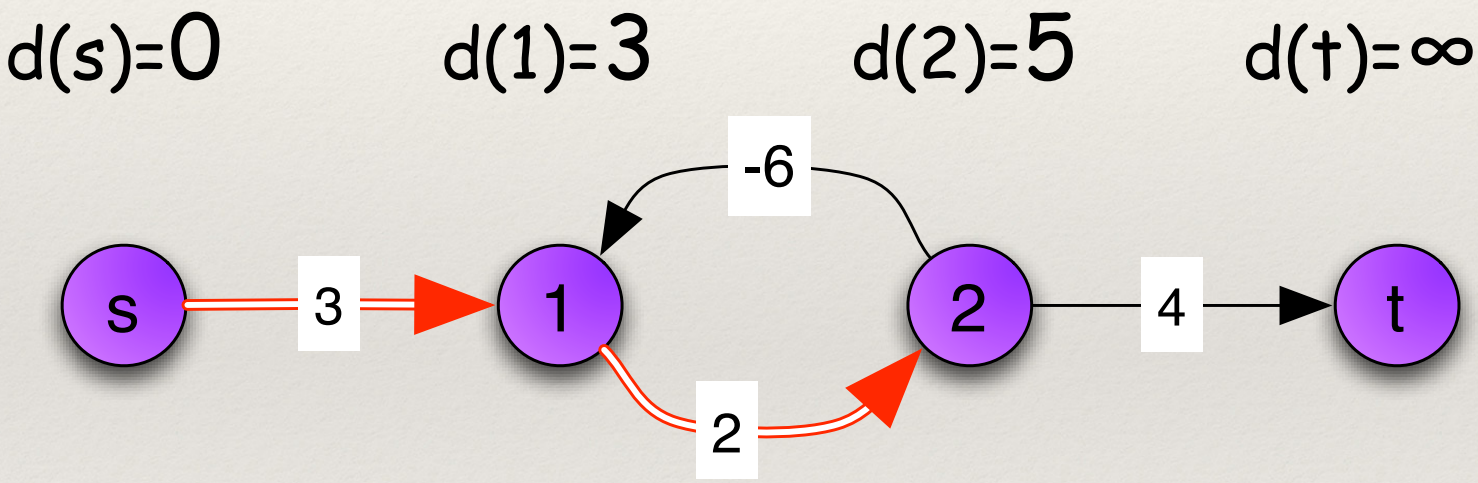


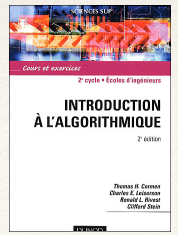
Circuit absorbant: Exemple



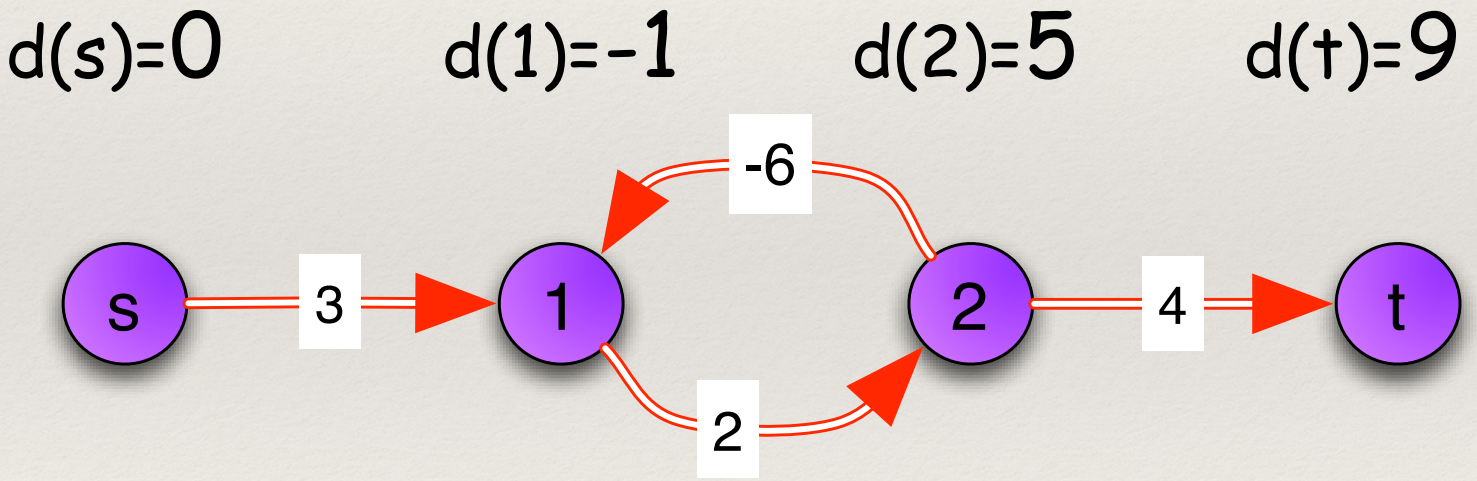


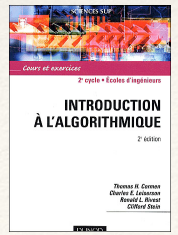
Circuit absorbant: Exemple



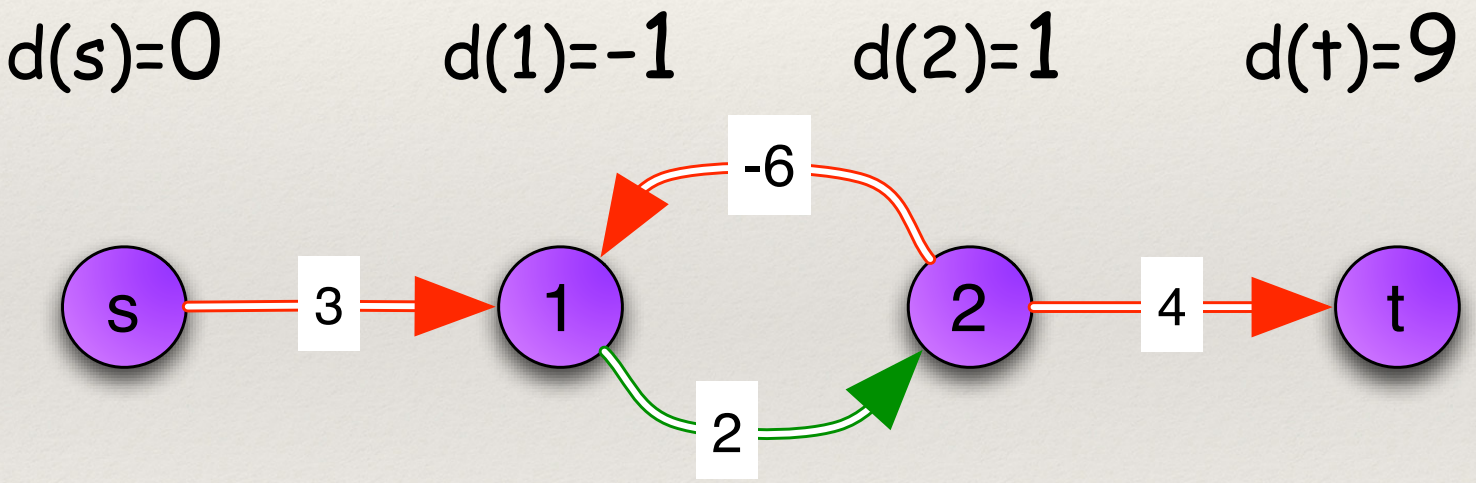


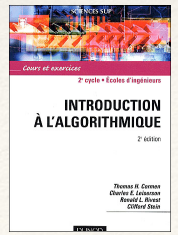
Circuit absorbant: Exemple



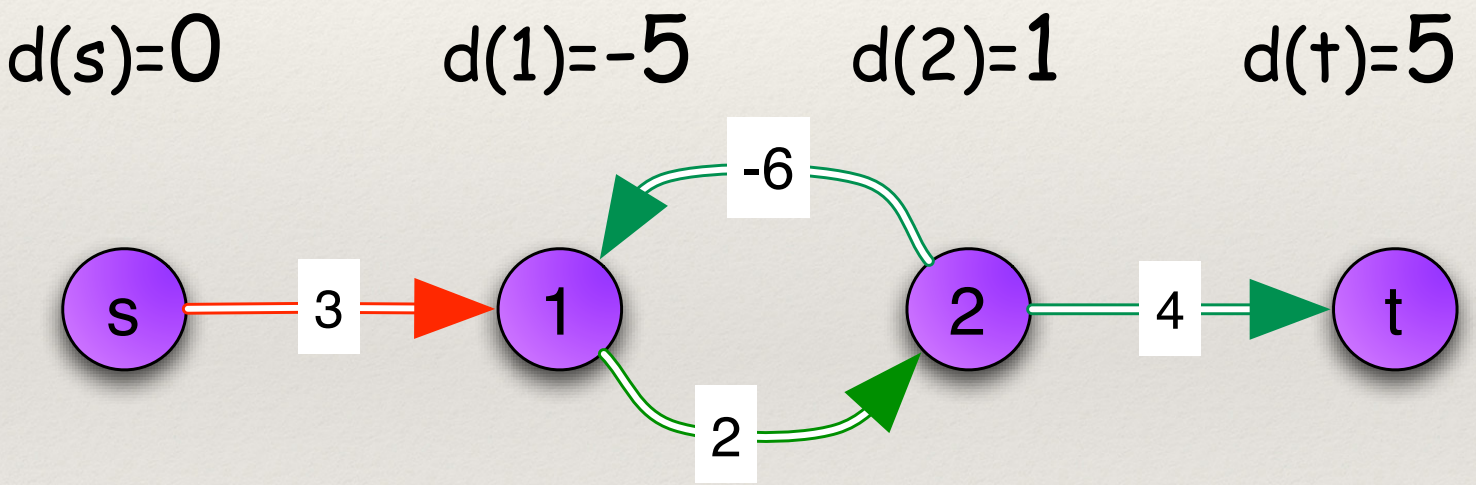


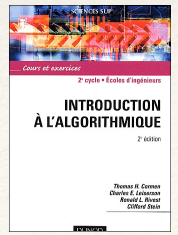
Circuit absorbant: Exemple



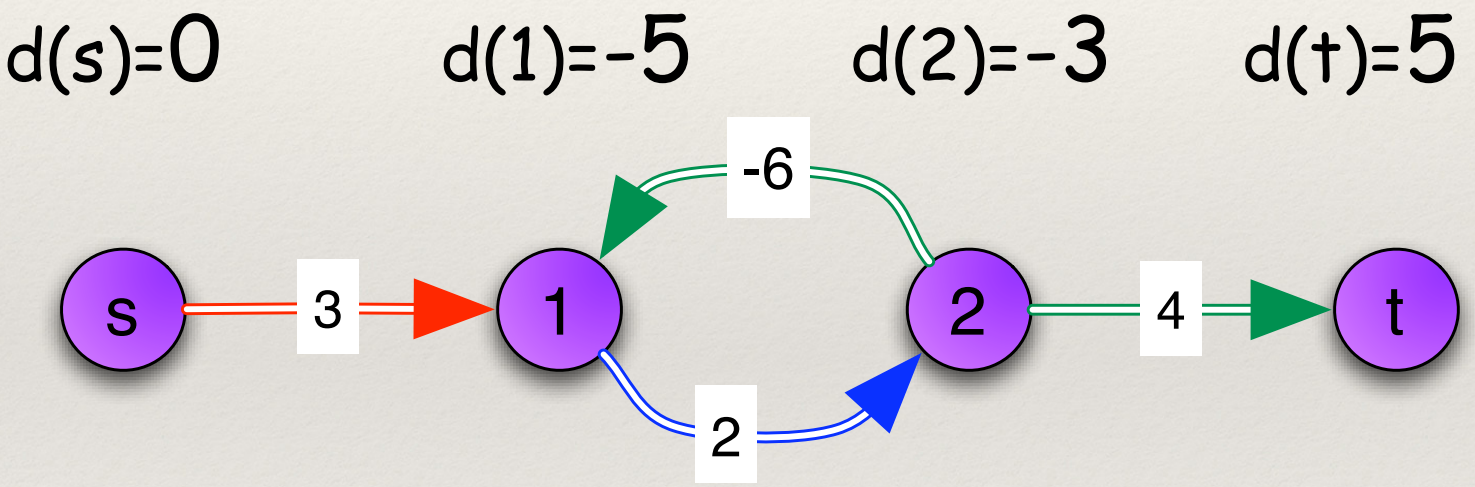


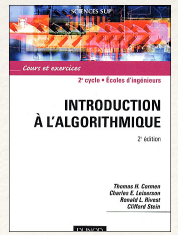
Circuit absorbant: Exemple



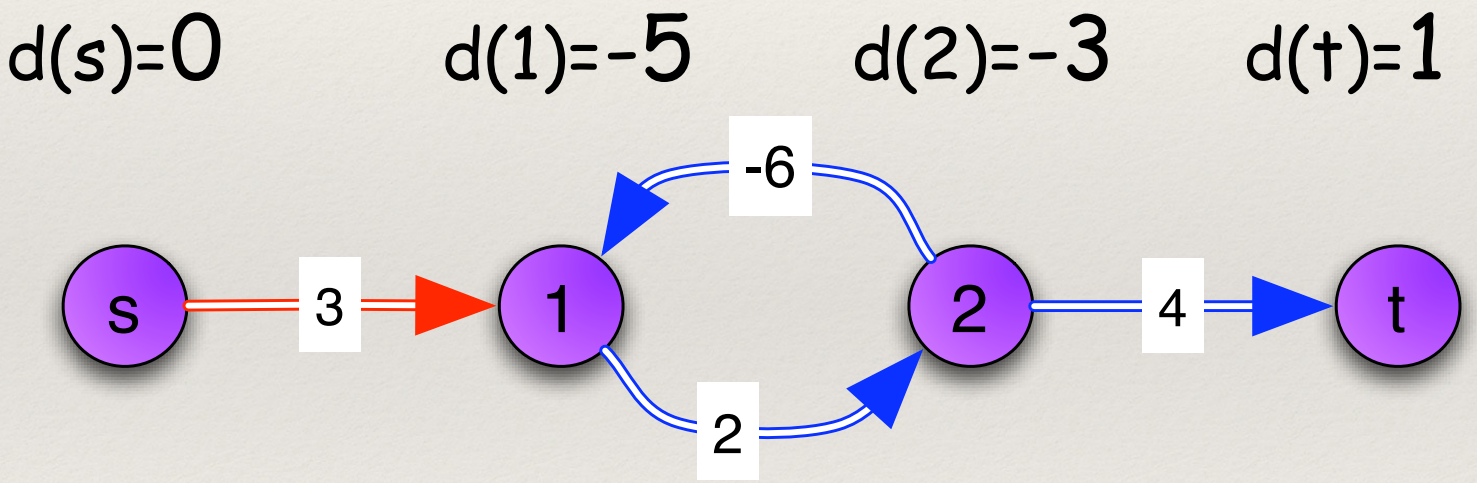


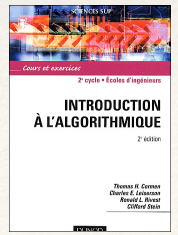
Circuit absorbant: Exemple





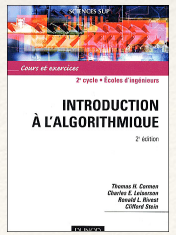
Circuit absorbant: Exemple





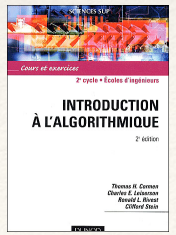
Représentation des plus courts chemins

- On souhaite:
  - Calculer les poids des plus courts chemins,
  - Donner les sommets présents sur ces plus courts chemins
- Étant donné un graphe G = (S, A,w),
- On gère pour chaque sommet v ∈ S un prédécesseur π[v]
  - π[v] est soit un autre sommet, soit NIL.
- On utilisera la fonction IMPRIMER-CHEMIN pour afficher tout les sommets:



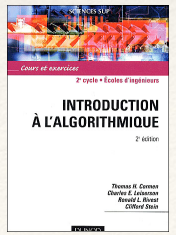
Représentation des plus courts chemins

```
IMPRIMER-CHEMIN(G, s, v)
1  si v = s
2      alors imprimer s
3      sinon si π[v] = NIL
4                alors imprimer " il n'existe aucun chemin de " s " à " v
5                sinon IMPRIMER-CHEMIN(G, s, π[v])
6                      imprimer v
```



Relâchement

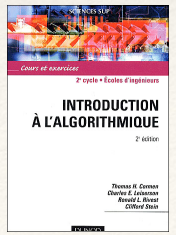
- Pour chaque sommet $v \in S$, on gère un attribut **d[v]**
- d[v] est un majorant de la longueur d'un plus court chemin entre l'origine s et v.
- On appelle d[v] une estimation de plus court chemin.
- On initialise les estimations et les prédécesseurs via la procédure SOURCE-UNIQUE-INITIALISATION(G, s) Procédure en temps $\Theta(S)$.



Relâchement

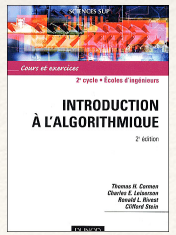
SOURCE-UNIQUE-INITIALISATION(G, s)

1 **pour** chaque sommet $v \in S[G]$
2 **faire** $d[v] \leftarrow \infty$
3 $\pi[v] \leftarrow$ NIL
4 $d[s] \leftarrow 0$



Relâchement

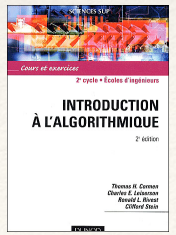
- Le processus de relâchement d'un arc (u, v) consiste à
  - Tester si l'on peut améliorer le plus court chemin vers v trouvé jusqu'ici en passant par u,
  - Si tel est le cas, actualisé d[v] et π[v].
- Une étape de relâchement peut
  - Diminuer la valeur de l'estimation de plus court chemin d[v]
  - Mettre à jour le champ prédécesseur π[v] de v.

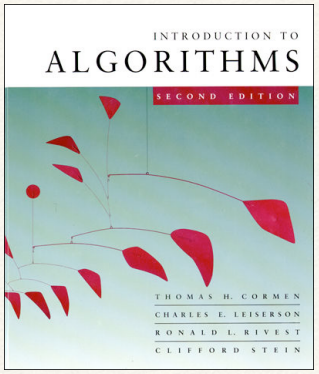


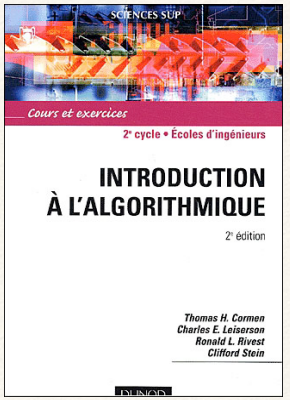
Relâchement

RELÂCHER(u, v, w)

1 **si** $d[v] > d[u] + w(u, v)$
2 **alors** $d[v] \leftarrow d[u] + w(u, v)$
3 $\pi[v] \leftarrow u$







Lélia Blin

L'algorithme de Bellman-Ford

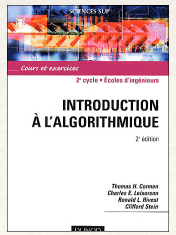
Théorie des graphes

Cours de Lélia Blin

3eme année de Licence

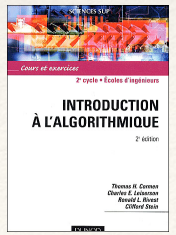
Algorithme de Bellman-Ford

- L'algorithme de Bellman-Ford résout le problème des **plus courts chemins à origine unique** dans le cas général où les poids d'arc peuvent avoir des **valeurs négatives**.
- Étant donné un graphe orienté pondéré G = (S, A,w), et une origine s.
- L'algorithme de Bellman-Ford retourne une valeur booléenne indiquant s'il existe ou non un circuit de longueur strictement négative accessible à partir de s.
- Si un tel circuit n'existe pas, l'algorithme donne les plus courts chemins ainsi que leurs poids.



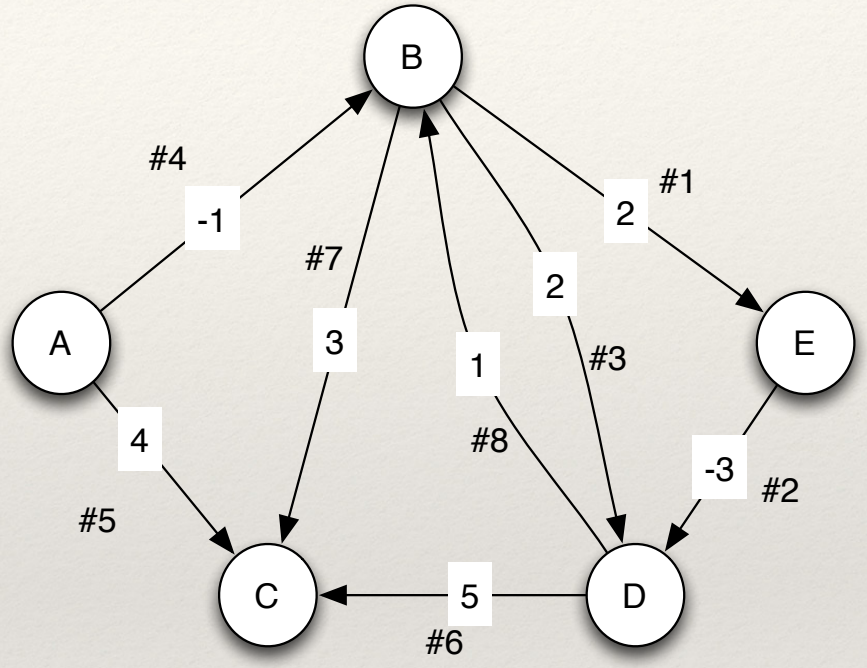
Algorithme de Ford-Bellman

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



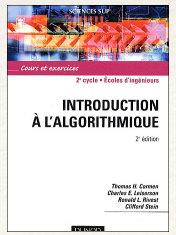
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



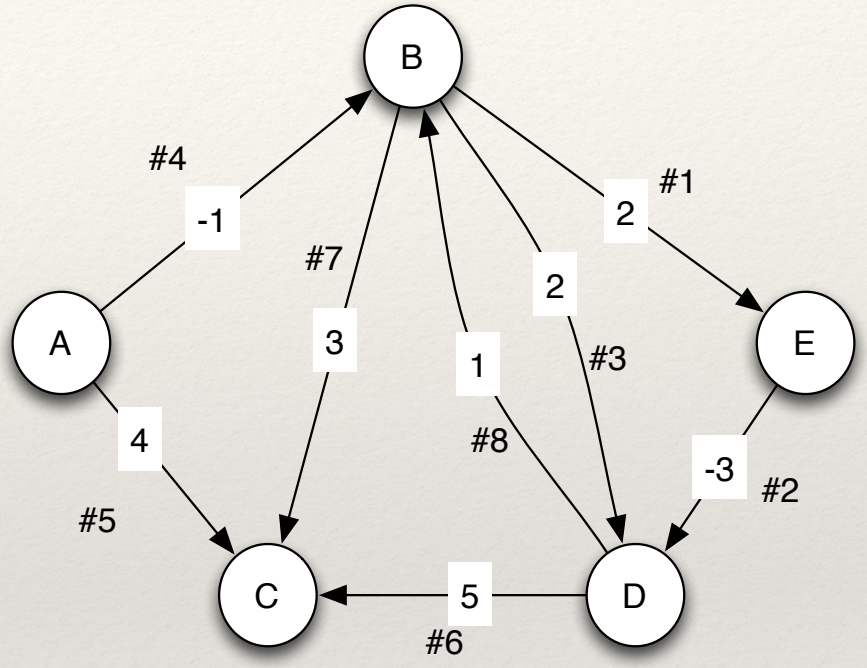
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2					
#3					
#4					
#5					
#6					
#7					
#8					



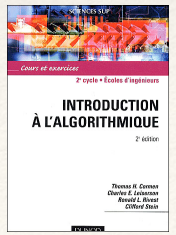
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



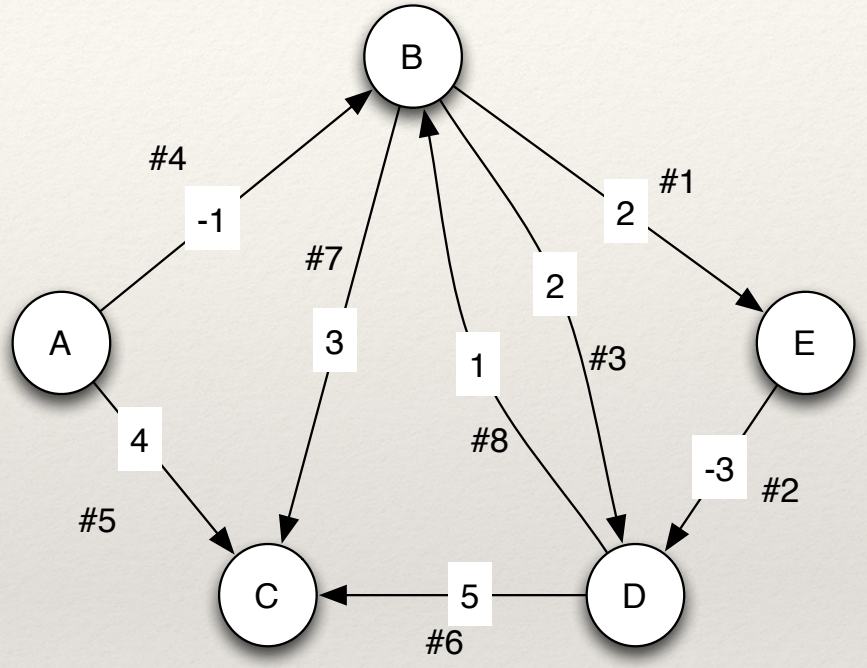
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2	0	∞	∞	∞	∞
#3					
#4					
#5					
#6					
#7					
#8					



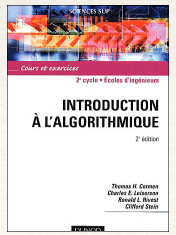
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3       faire pour chaque arc (u, v) ∈ A[G]
4            faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6       faire si d[v] > d[u] + w(u, v)
7            alors retourner FAUX
8  retourner VRAI
```



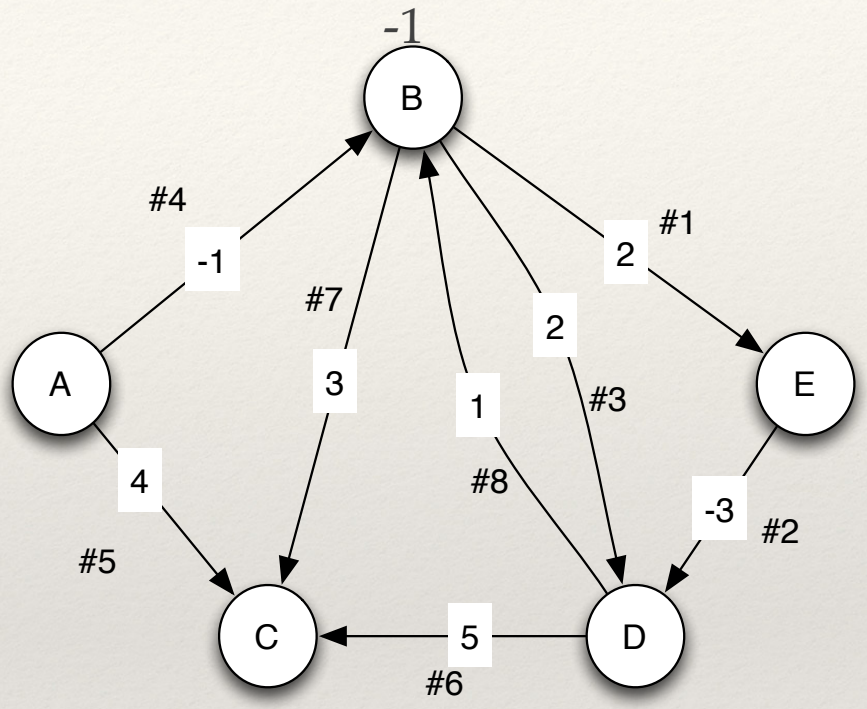
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2	0	∞	∞	∞	∞
#3	0	∞	∞	∞	∞
#4					
#5					
#6					
#7					
#8					



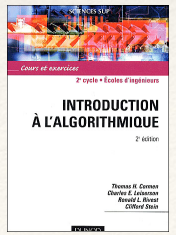
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



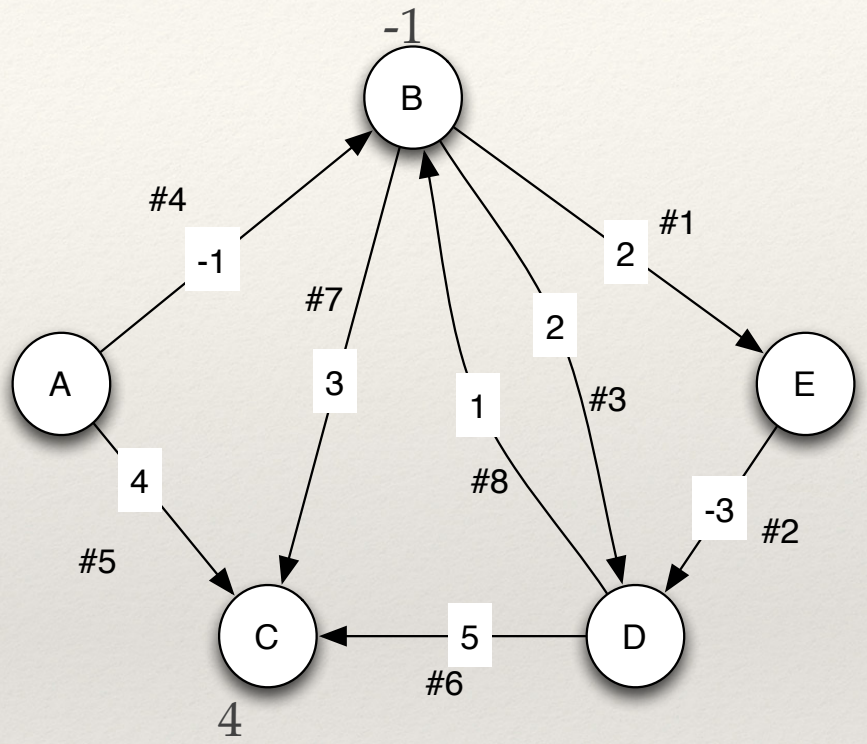
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2	0	∞	∞	∞	∞
#3	0	∞	∞	∞	∞
#4	0	-1	∞	∞	∞
#5					
#6					
#7					
#8					



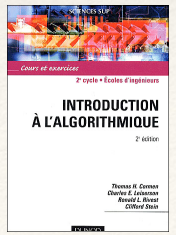
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



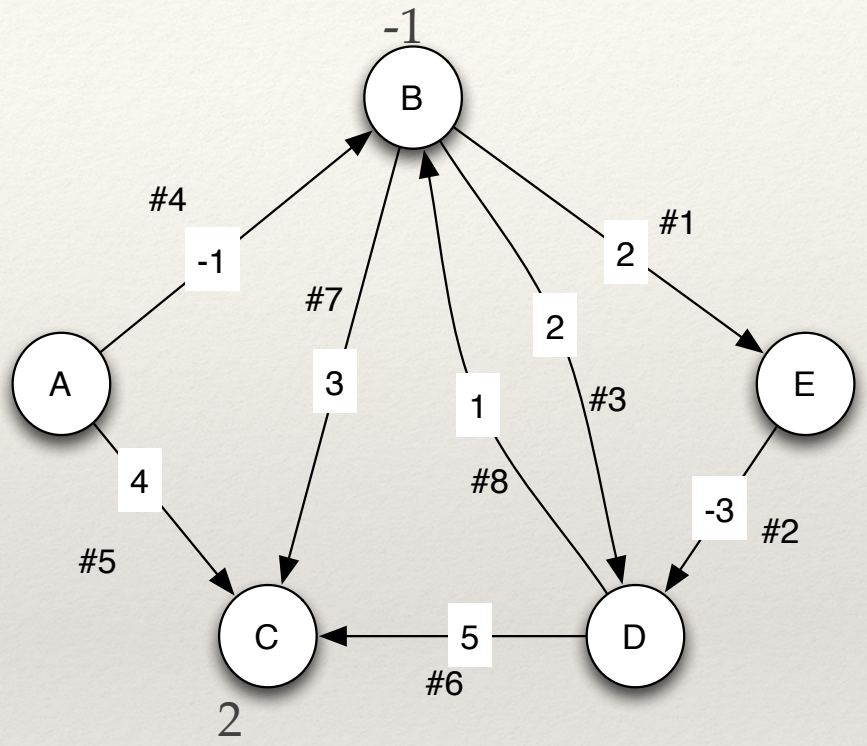
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2	0	∞	∞	∞	∞
#3	0	∞	∞	∞	∞
#4	0	-1	∞	∞	∞
#5	0	-1	4	∞	∞
#6					
#7					
#8					



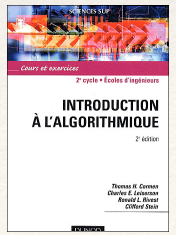
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



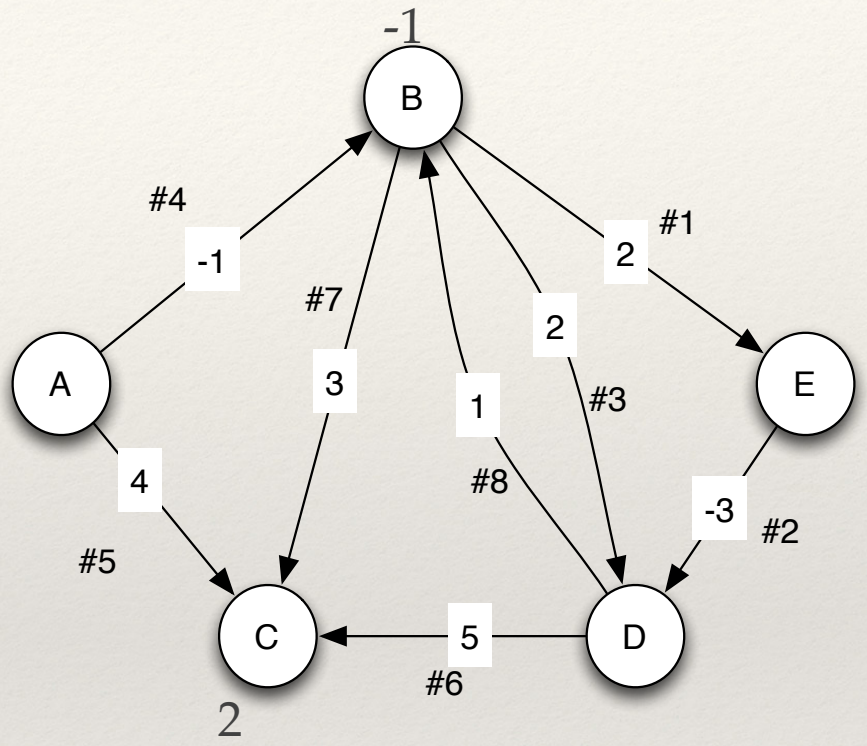
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2	0	∞	∞	∞	∞
#3	0	∞	∞	∞	∞
#4	0	-1	∞	∞	∞
#5	0	-1	4	∞	∞
#6	0	-1		∞	∞
#7					
#8					



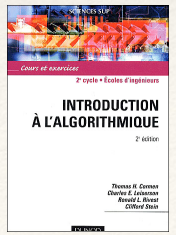
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



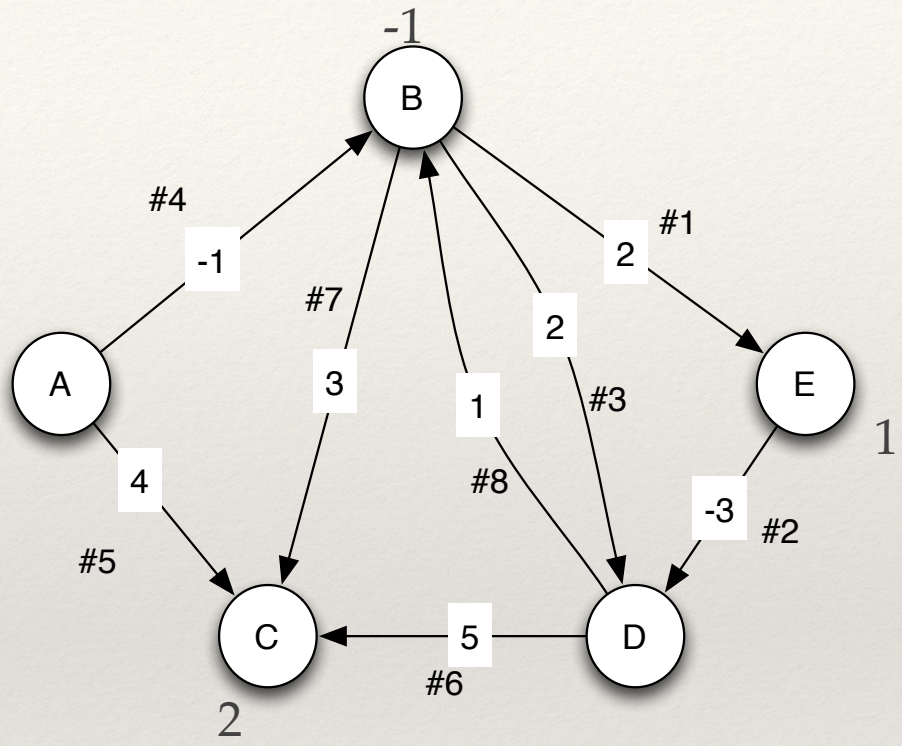
i=1

	A	B	C	D	E
	0	∞	∞	∞	∞
#1	0	∞	∞	∞	∞
#2	0	∞	∞	∞	∞
#3	0	∞	∞	∞	∞
#4	0	-1	∞	∞	∞
#5	0	-1	4	∞	∞
#6	0	-1	4	∞	∞
#7	0	-1	2	∞	∞
#8	0	-1	2	∞	∞



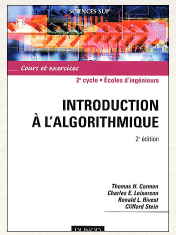
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2					
#3					
#4					
#5					
#6					
#7					
#8					

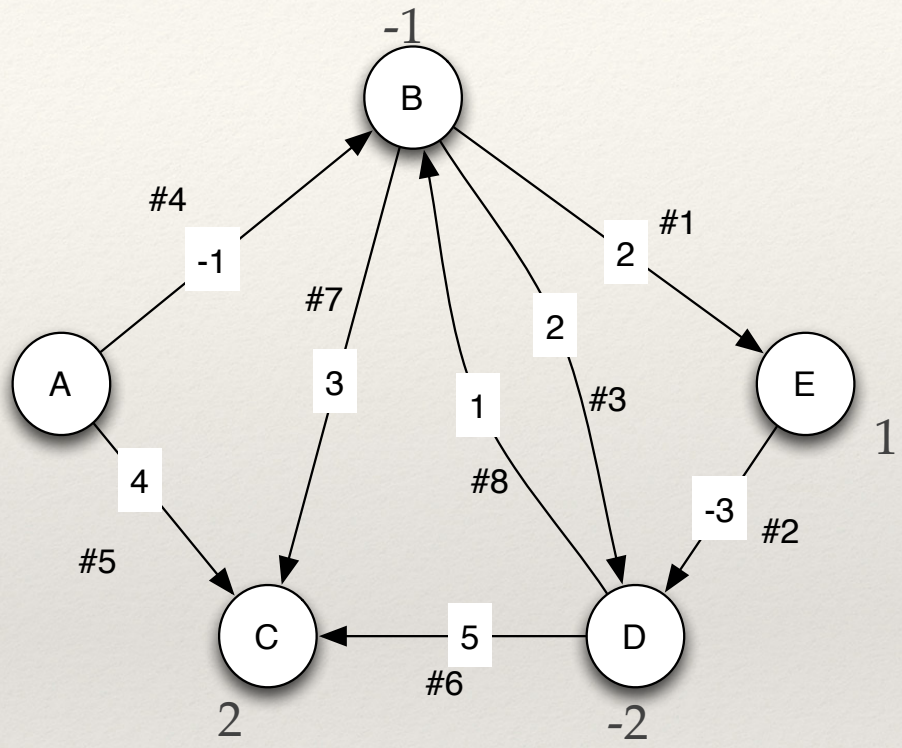


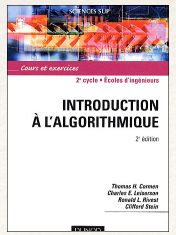
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```

i=2

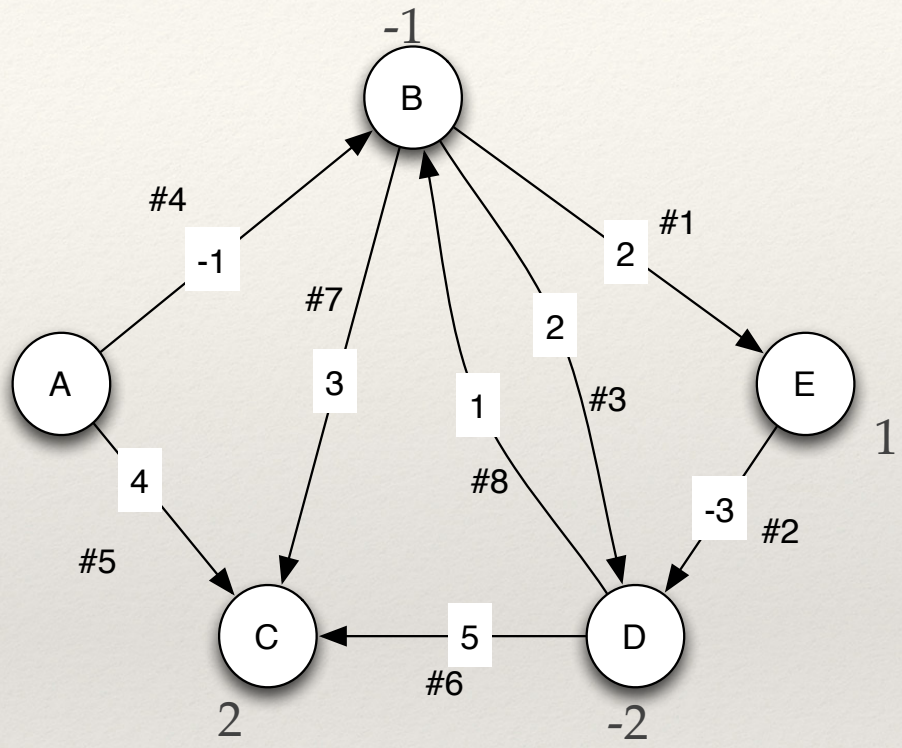
	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3					
#4					
#5					
#6					
#7					
#8					





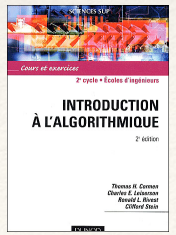
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3       faire pour chaque arc (u, v) ∈ A[G]
4            faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6       faire si d[v] > d[u] + w(u, v)
7            alors retourner FAUX
8  retourner VRAI
```



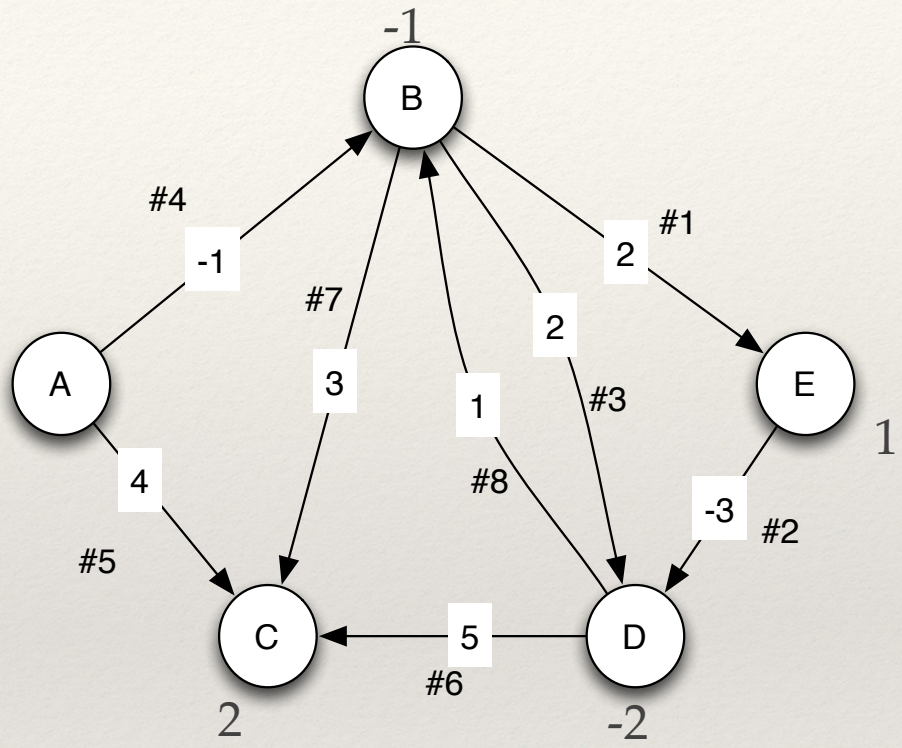
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4					
#5					
#6					
#7					
#8					



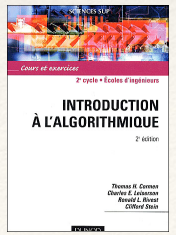
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



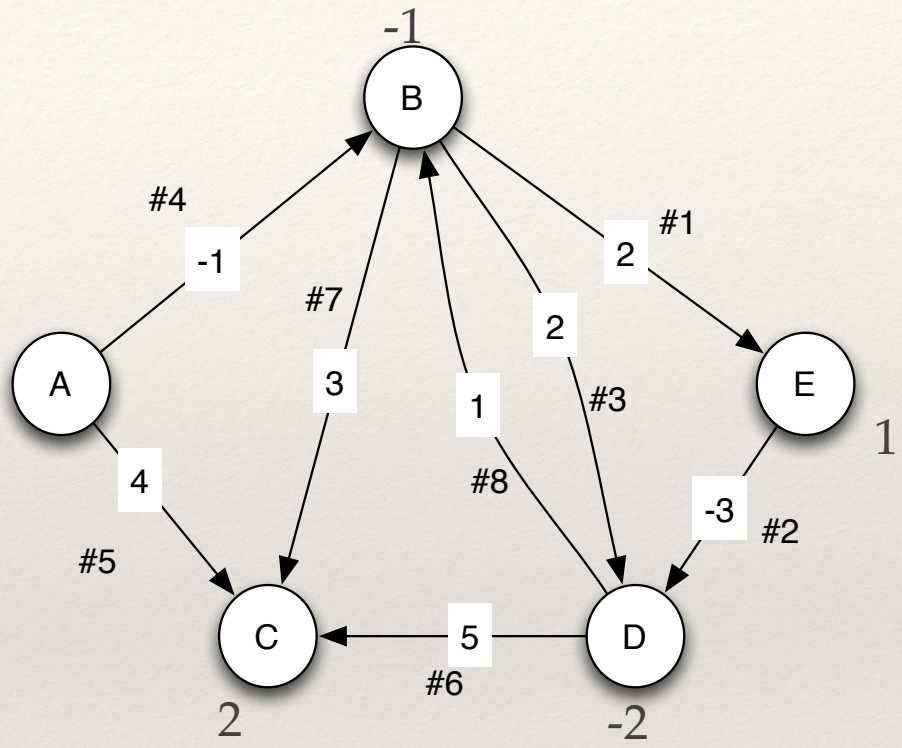
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4					
#5					
#6					
#7					
#8					



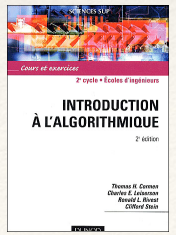
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



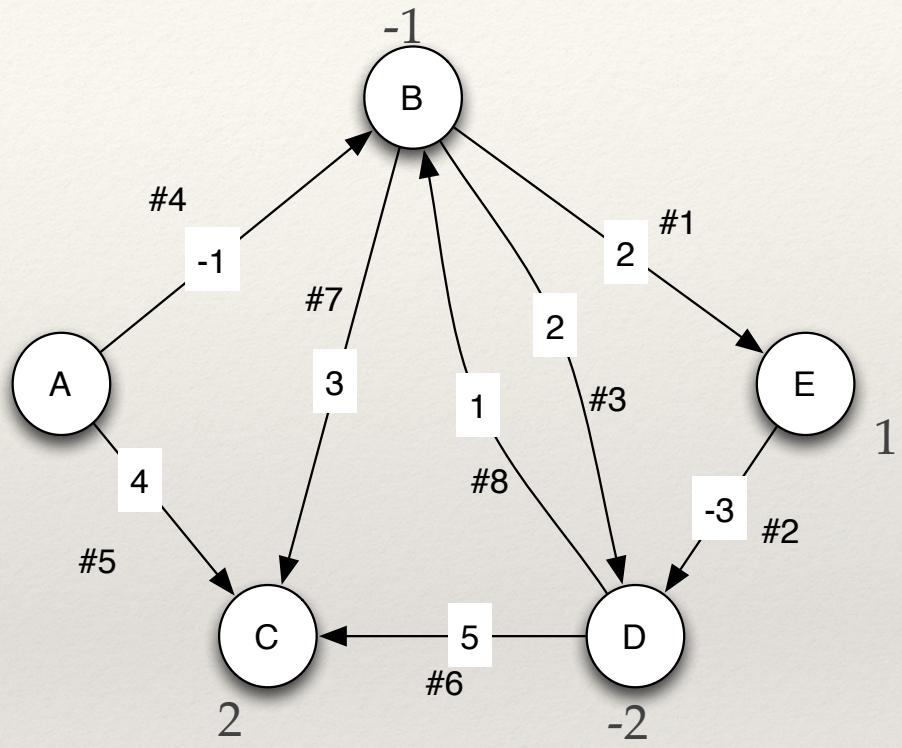
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4	0	-1	2	-2	1
#5					
#6					
#7					
#8					



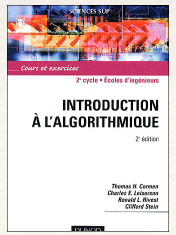
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



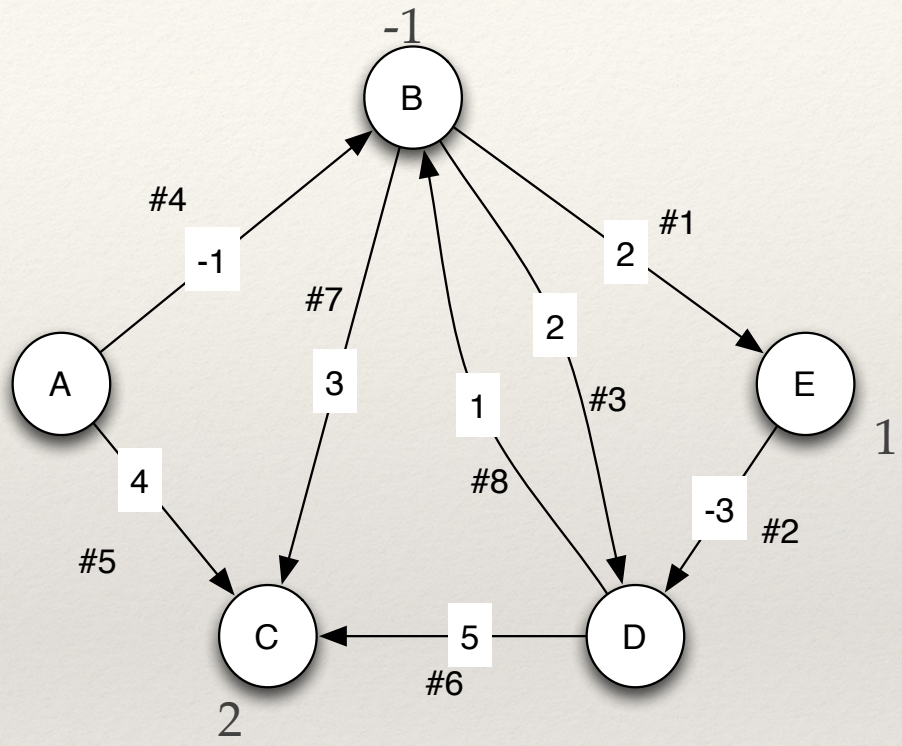
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4	0	-1	2	-2	1
#5	0	-1	2	-2	1
#6					
#7					
#8					



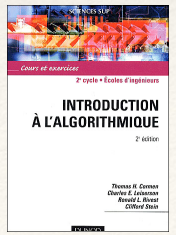
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3       faire pour chaque arc (u, v) ∈ A[G]
4            faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6       faire si d[v] > d[u] + w(u, v)
7            alors retourner FAUX
8  retourner VRAI
```



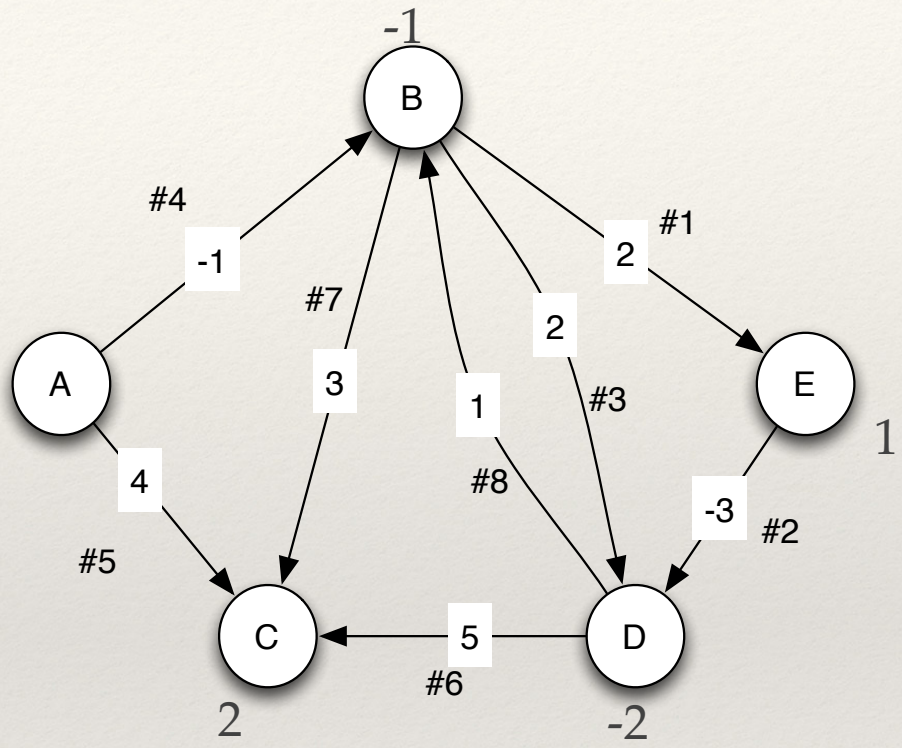
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4	0	-1	2	-2	1
#5	0	-1	2	-2	1
#6	0	-1	2	-2	1
#7					
#8					



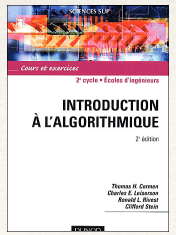
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



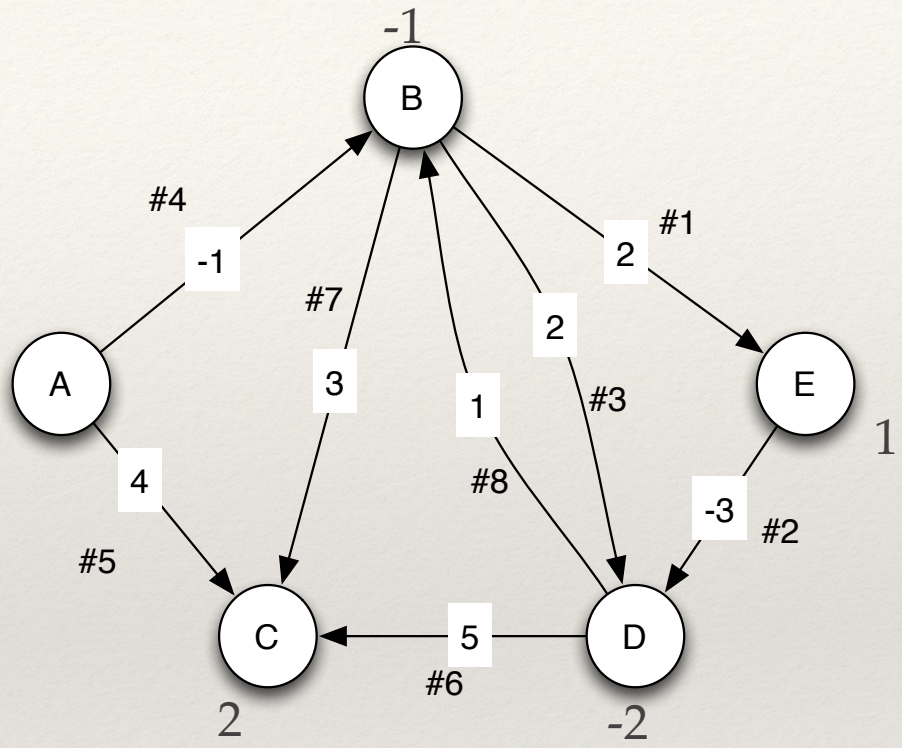
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4	0	-1	2	-2	1
#5	0	-1	2	-2	1
#6	0	-1	2	-2	1
#7	0	-1	2	-2	1
#8					



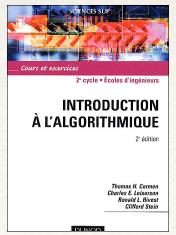
Exemple

```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```



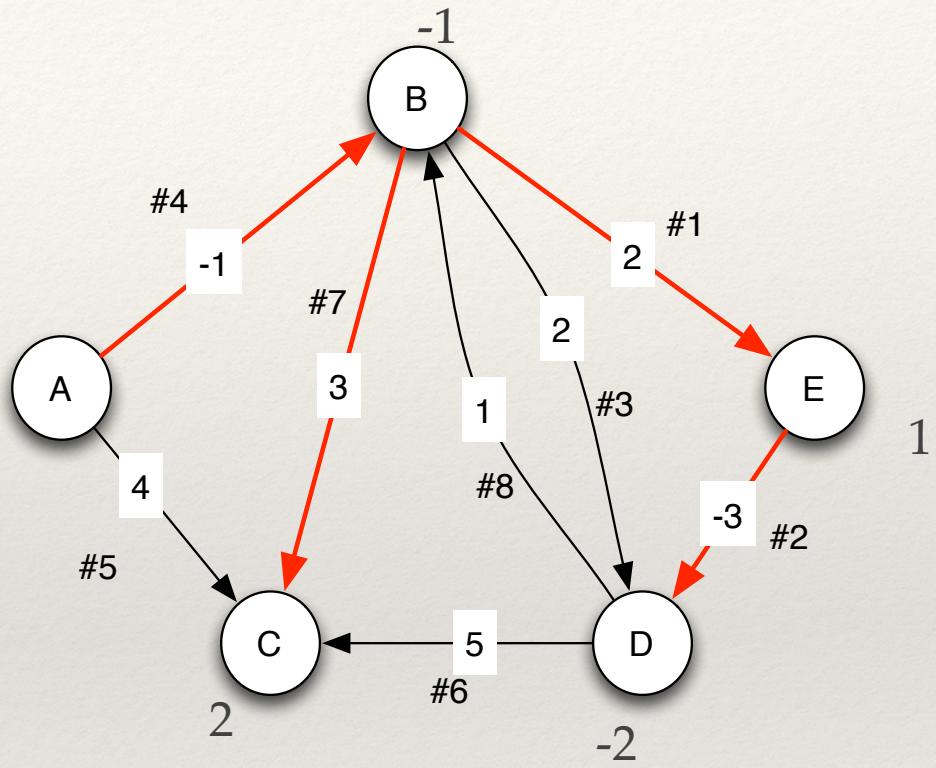
i=2

	A	B	C	D	E
	0	-1	2	∞	∞
#1	0	-1	2	∞	1
#2	0	-1	2	-2	1
#3	0	-1	2	-2	1
#4	0	-1	2	-2	1
#5	0	-1	2	-2	1
#6	0	-1	2	-2	1
#7	0	-1	2	-2	1
#8	0	-1	2	-2	1

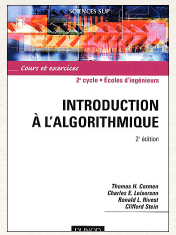


Exemple

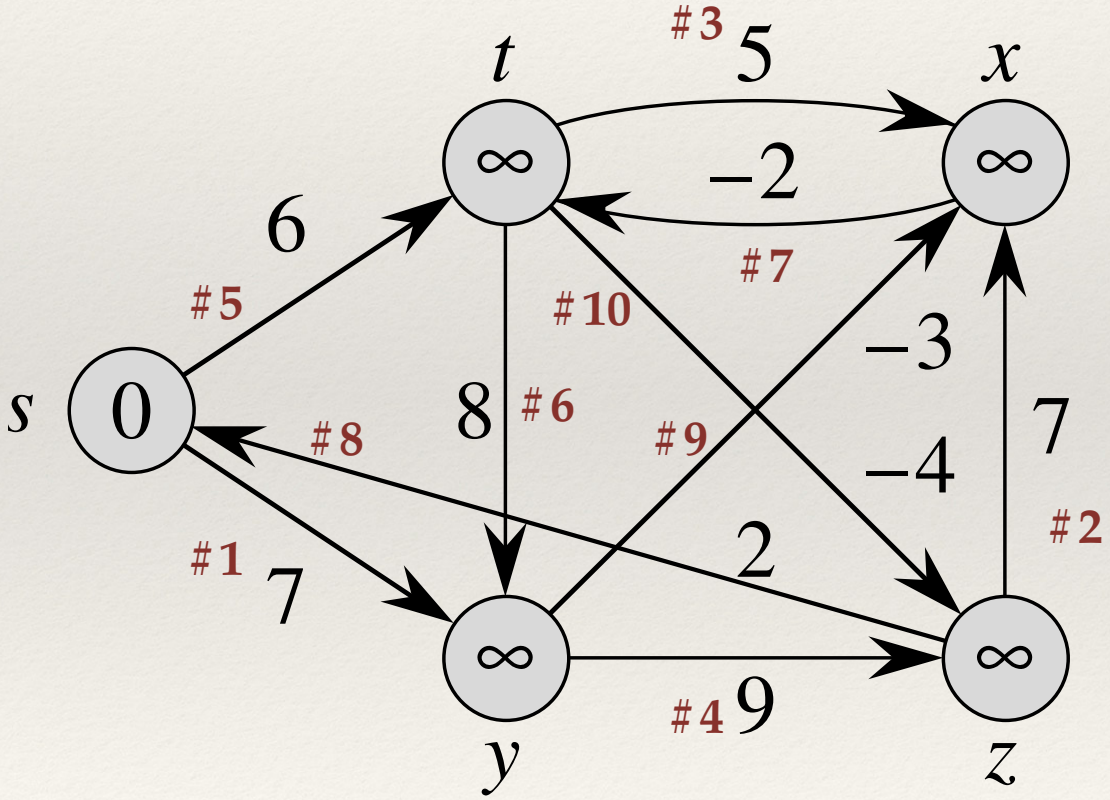
```
BELLMAN-FORD(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  pour i ← 1 à |S[G]| − 1
3      faire pour chaque arc (u, v) ∈ A[G]
4          faire RELÂCHER(u, v, w)
5  pour chaque arc (u, v) ∈ A[G]
6      faire si d[v] > d[u] + w(u, v)
7          alors retourner FAUX
8  retourner VRAI
```

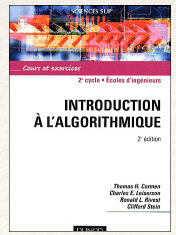


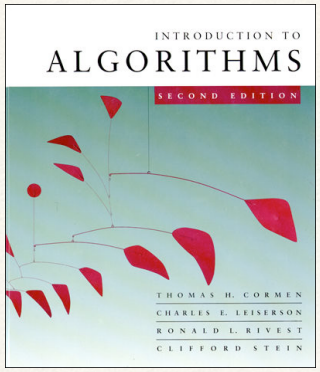
il faut recommencer pour
i=3 et i=4
Mais rien ne change

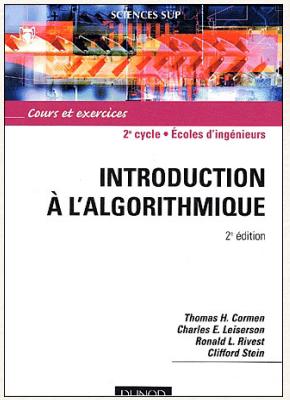


Autre exemple









Lélia Blin

Algorithme de Dijkstra

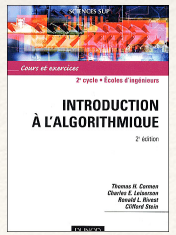
Théorie des graphes

Cours de Lélia Blin

3eme année de Licence

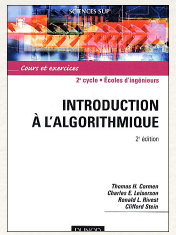
Remarques

- L'algorithme de Dijkstra est un autre algorithme de recherche de distance et de plus court chemin.
- Il est plus efficace que Bellman-Ford.
  - Mais ne fonctionne que dans le cas où **toutes les valuations des arcs sont positives**.



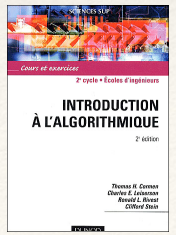
Principe

- On construit petit à petit, à partir de {x0}, un ensemble M de sommets marqués.
- Pour tout sommet marqués , l'estimation d(s) est égale à la distance δ(x, s).
- A chaque étape, on sélectionne le (un) sommet non marqué x dont la distance estimé d (x) est la plus petite parmi tous les sommets non marqué.
- On marque alors x (on rajoute x à M), puis on met à jour à partir de x les distances estimées des successeurs non marqués de x
- On recommence, jusqu'à épuisement des sommets non marqués



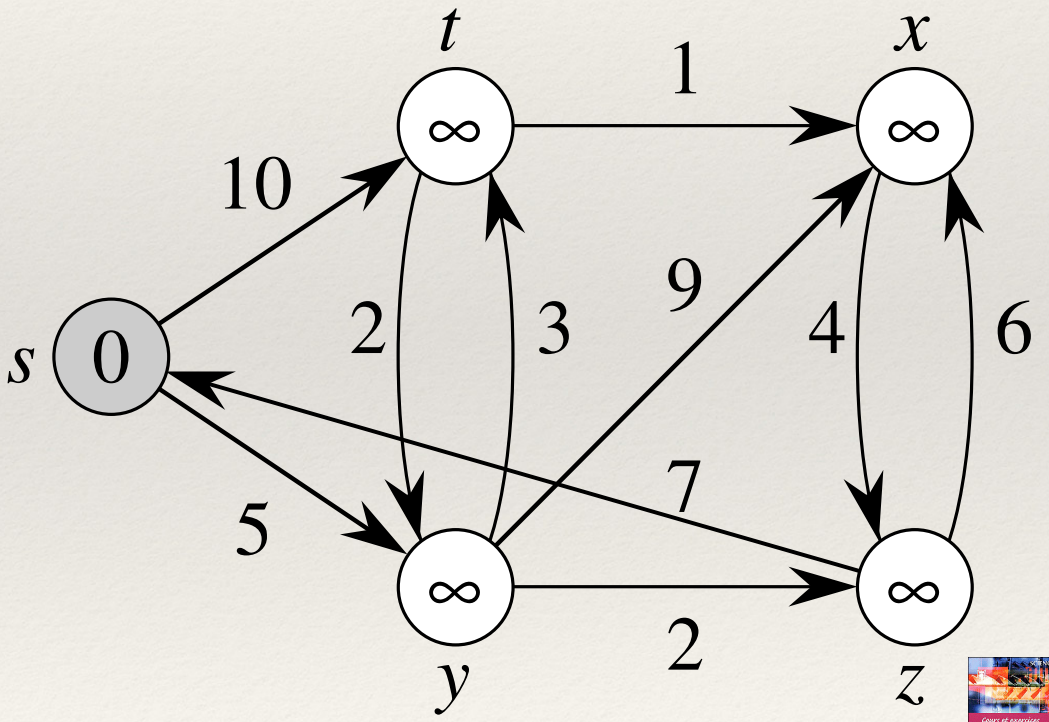
Algorithme de Dijkstra

```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5      faire u ← EXTRAIRE-MIN(F)
6           E ← E ∪ {u}
7           pour chaque sommet v ∈ Adj[u]
8               faire RELÂCHER(u, v, w)
```

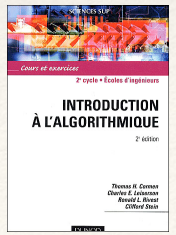


Exemple

```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6             E ← E ∪ {u}
7             pour chaque sommet v ∈ Adj[u]
8                  faire RELÂCHER(u, v, w)
```



F={s,t,x,y,z}
E={}



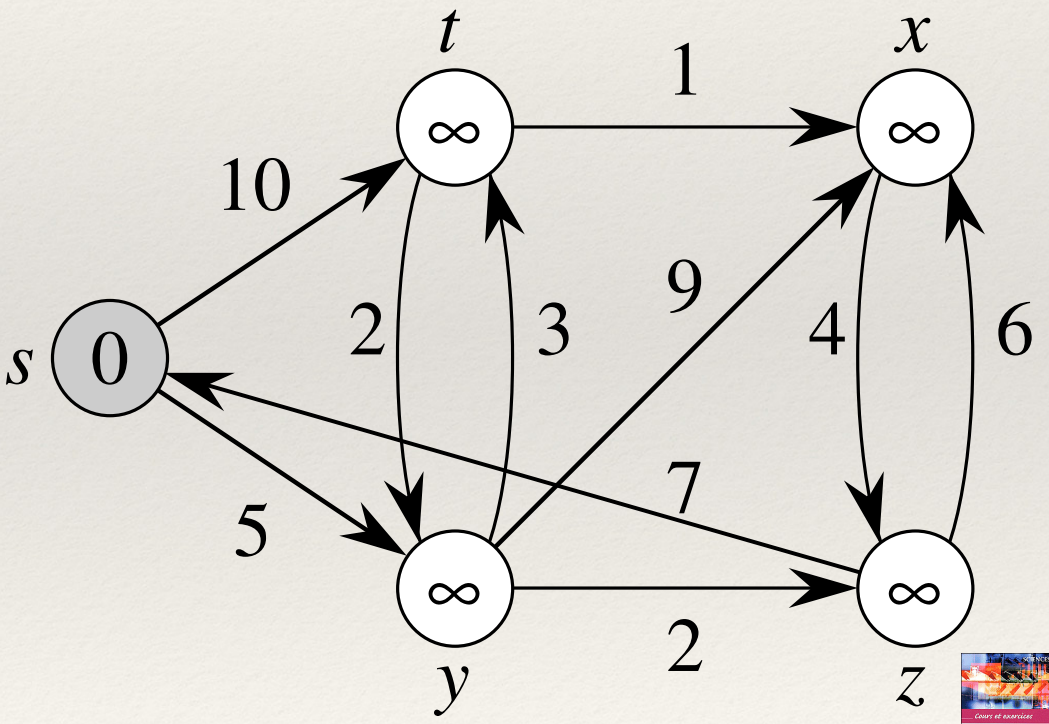
Exemple

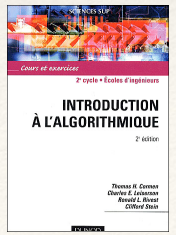
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞

F={t,x,y,z}

E={s}





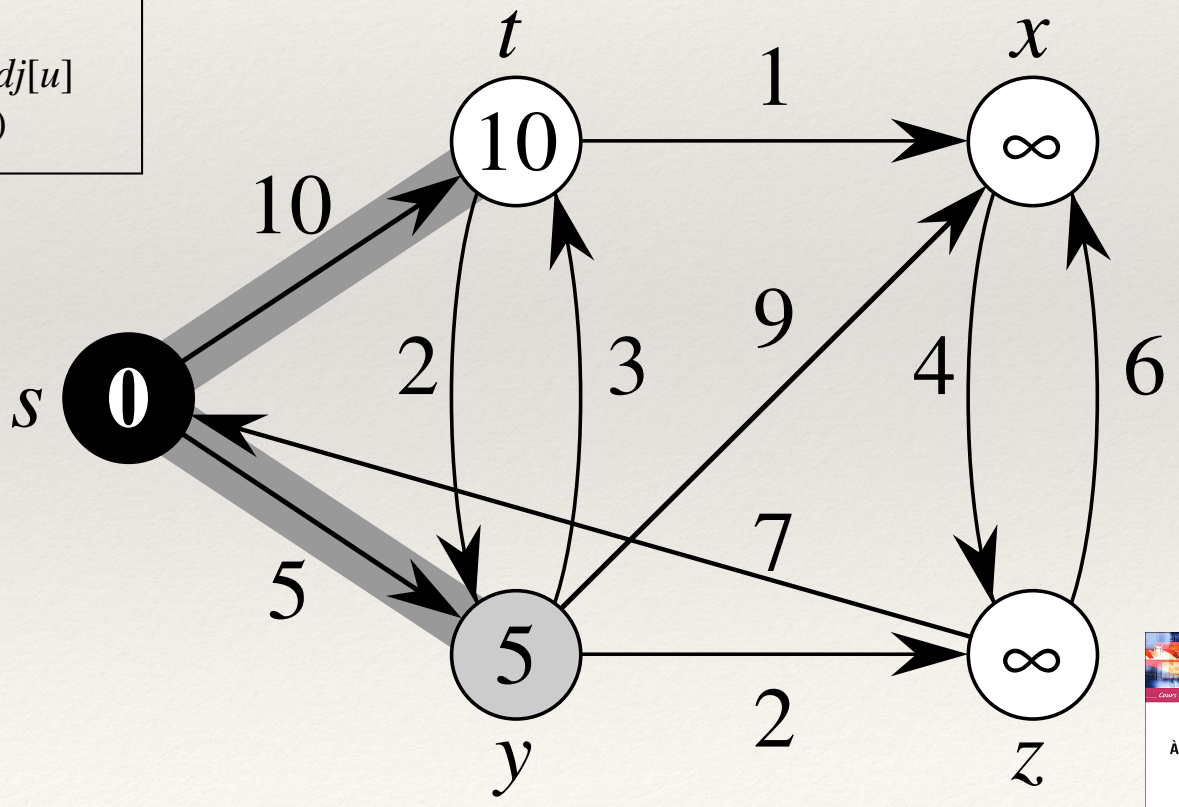
Exemple

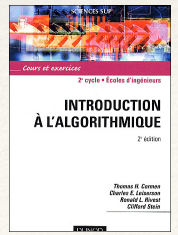
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6             E ← E ∪ {u}
7             pour chaque sommet v ∈ Adj[u]
8                  faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞

F={t,x,y,z}

E={s}





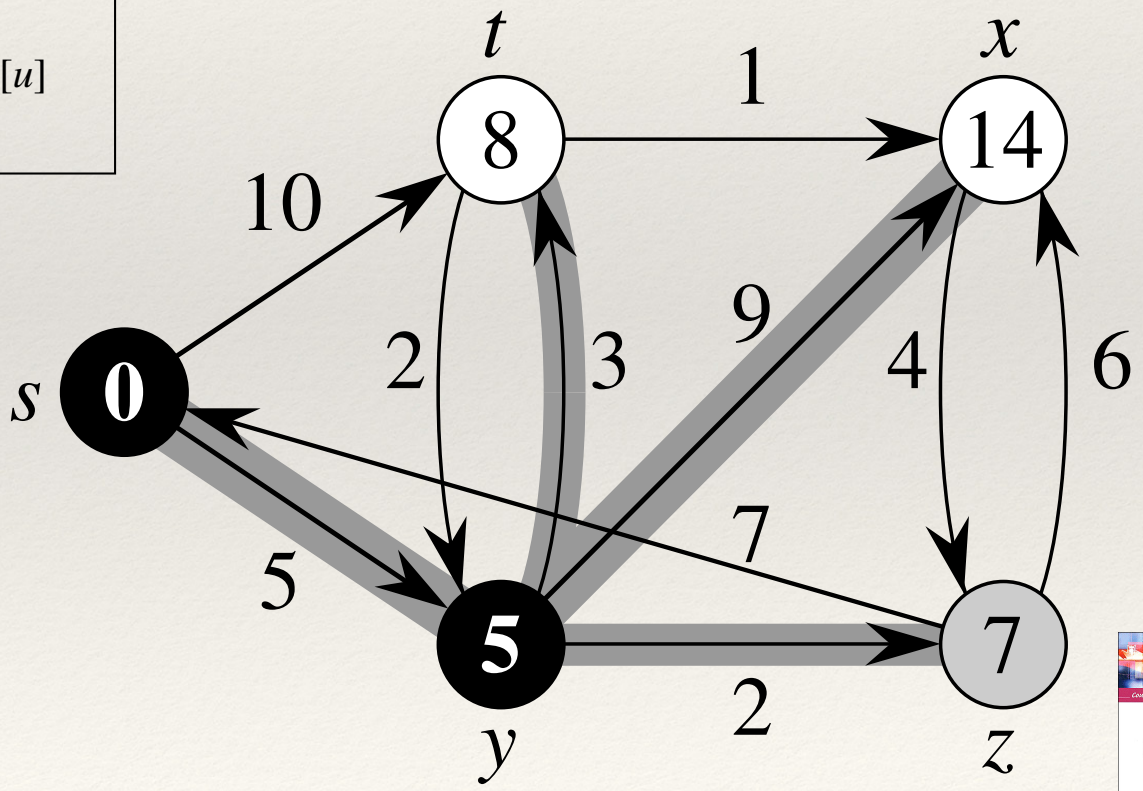
Exemple

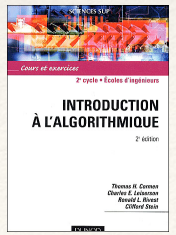
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y

F={t,x,z}

E={s,y}

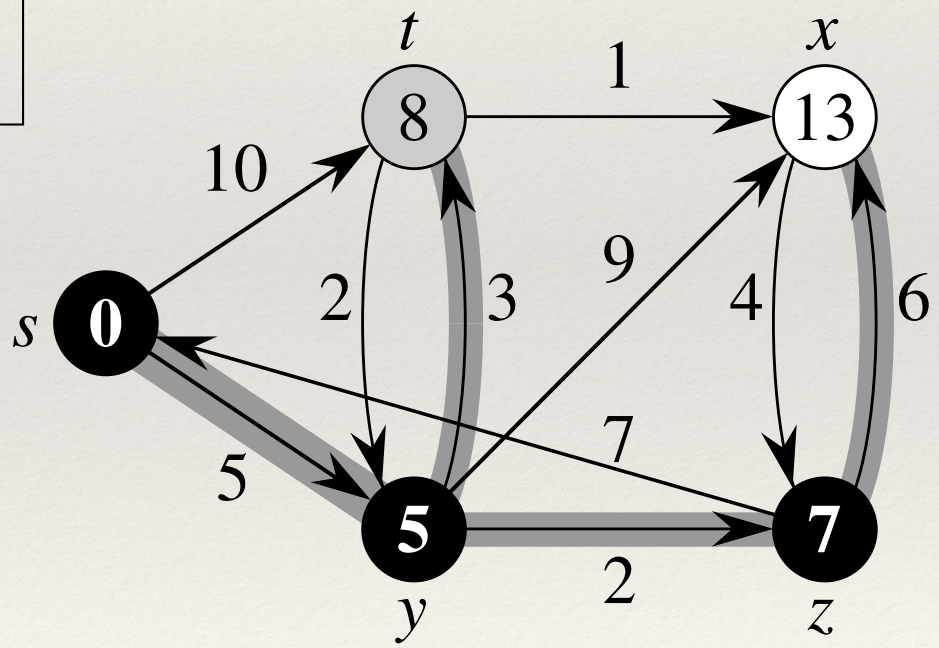




Exemple

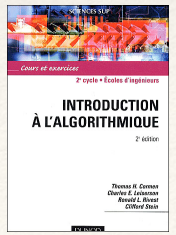
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y
	8y	13z		



F={t,x}

E={s,y,z}



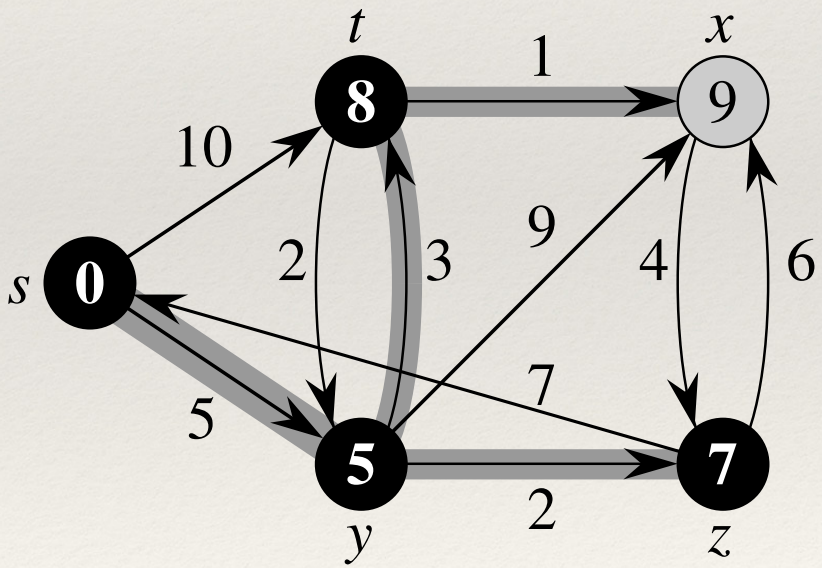
Exemple

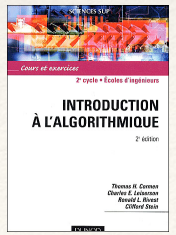
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y
	8y	13z		
		9t		

F={x}

E={s,y,z,t}





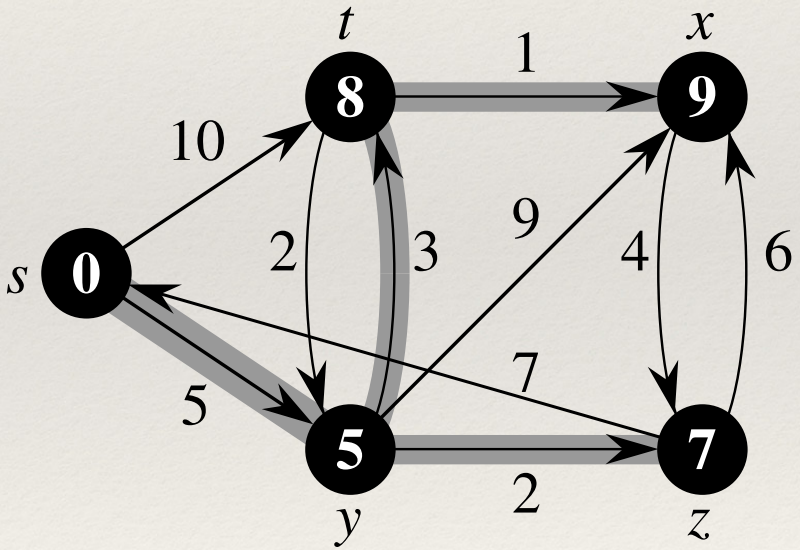
Exemple

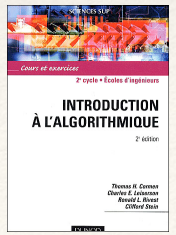
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y
	8y	13z		
		9t		

F={}

E={s,y,z,t,x}





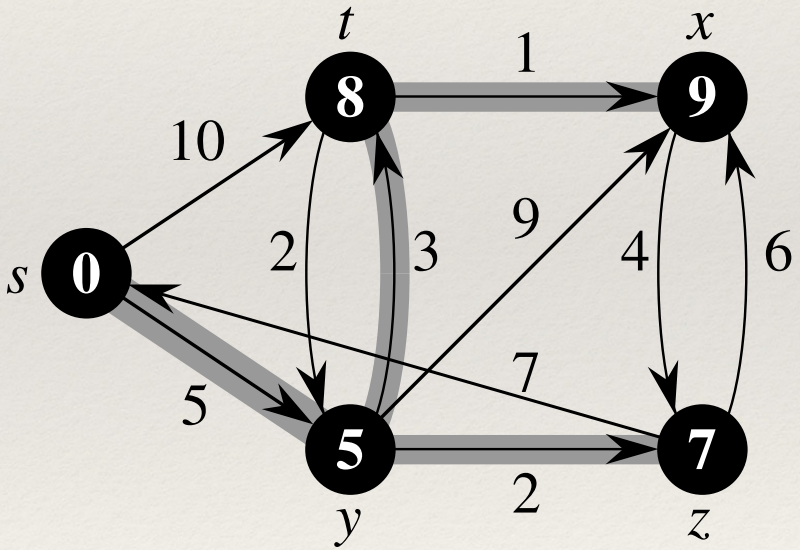
Exemple

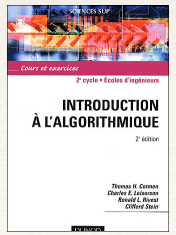
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y
	8y	13z		
		9t		

F={}

E={s,y,z,t,x}

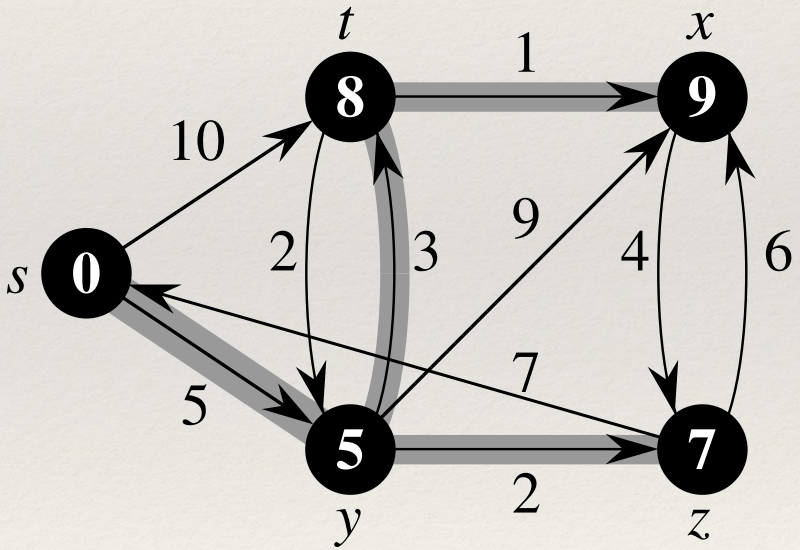




Exemple

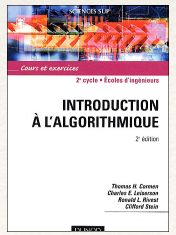
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y
	8y	13z		
		9t		



F={}

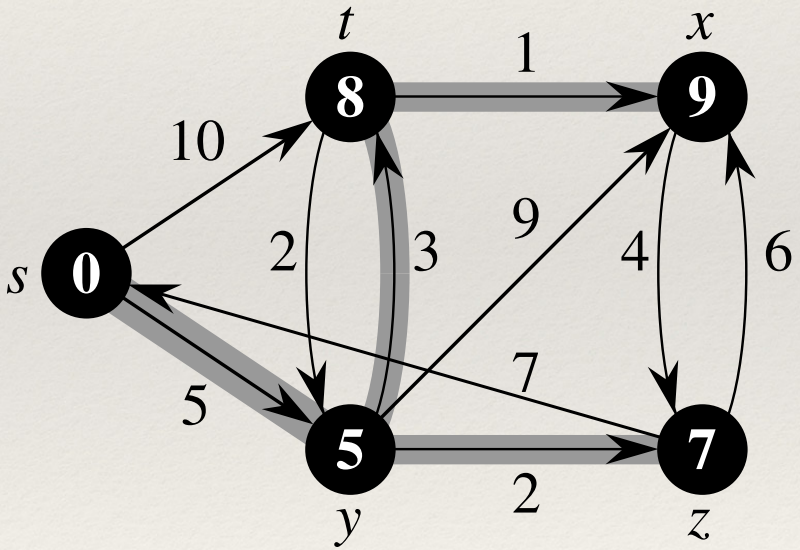
E={s,y,z,t,x}



Exemple

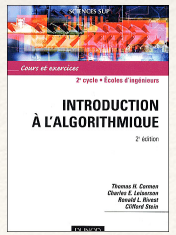
```
DIJKSTRA(G, w, s)
1  SOURCE-UNIQUE-INITIALISATION(G, s)
2  E ← ∅
3  F ← S[G]
4  tant que F ≠ ∅
5       faire u ← EXTRAIRE-MIN(F)
6            E ← E ∪ {u}
7            pour chaque sommet v ∈ Adj[u]
8                 faire RELÂCHER(u, v, w)
```

s	t	x	y	z
0	∞	∞	∞	∞
	10s	∞	5s	∞
	8y	14y		7y
	8y	13z		
		9t		

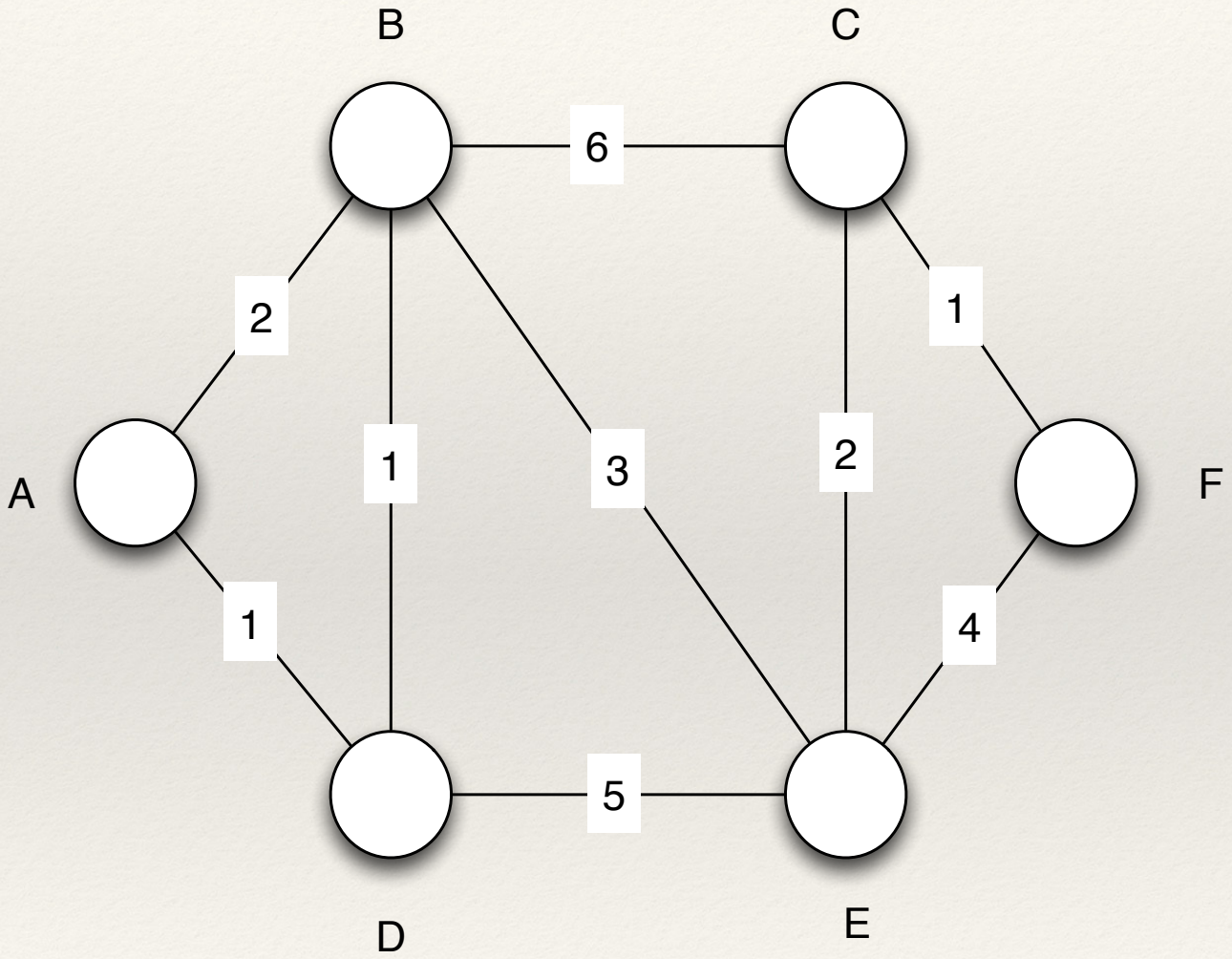


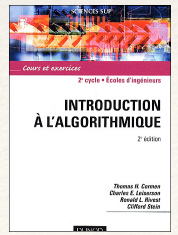
F={}

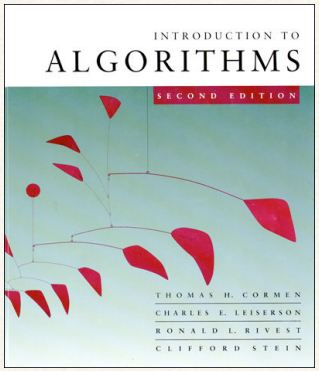
E={s,y,z,t,x}

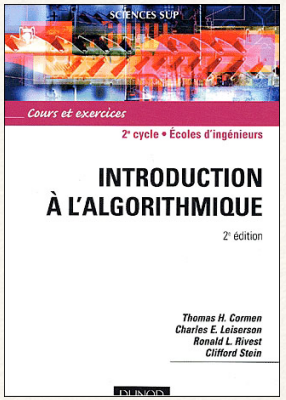


Autre exemple









Lélia Blin

Propriétés des plus court chemins

Théorie des graphes

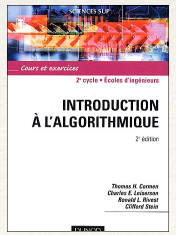
Cours de Lélia Blin

3eme année de Licence

Inégalités triangulaires

- **Propriété**:
  - Pour tout arc $(u,v) \in A$, on a $\delta(s,v) \leq \delta(s,u) + w(u,v)$.

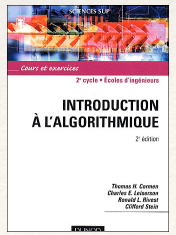
- **Preuve par l'absurde:**
  - Hypothèse: $\delta(s,v)$ est le plus court chemin de s au sommet u.
  - Si il existe $\delta'(s,v)=\delta(s,u) + w(u,v)$ tel que $\delta(s,v) ' < \delta(s,v)$
  - Alors l'hypothèse est fausse



Propriété du majorant

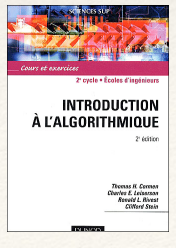
- **Propriété:**
  - On a toujours d[v] ≥ δ(s, v) pour tous les sommets v ∈ S

- **Preuve par récurrence:**
  - **Cas de base:** à l'initialisation d[v]=∞ si G est connexe (ou fortement connexe) alors il existe un chemin de longues l ∈**R** si G n'est pas connexe δ(s, v)=∞.
  - **Hypothèse de récurrence:** Au relâchement i concernant v on a d[v] ≥ δ(s, v).



Propriété du majorant

- **Hypothèse de récurrence:** Au relâchement i concernant v on a d[v] ≥ δ(s, v).
- **Prouvons qu'au relâchement i+1 concernant v on a toujours d[v] ≥ δ(s, v).**
  - Puisque l'on peut faire un relâchement i+1 c'est que l'on a un voisin u de v tel que: d[v]> d[u]+w(u,v).
  - On a d[v]> d[u]+w(u,v)≥ δ(s, v)

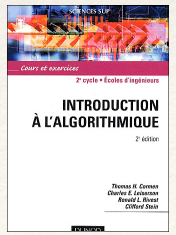


Propriété de convergence

- **Propriété:**
  - Si il n'existe plus aucun relâchement possible alors $\forall v \in S$ on a $d[v]=\delta(s, v)$.

- **Preuve**
  - Trivial

- **Remarque:**
  - Un plus court chemin est constitué de plus court chemin.



Propriété de sous-graphe prédécesseurs

- Une fois que d[v] = δ(s, v) pour tout v $\in$ S, le sous-graphe des prédécesseur est une arborescence de plus courts chemins de racine s.