Université de Msila
Faculté Mathématiques et Informatique
Département d'Informatique

Cours de Théorie des graphes

2eme année Informatique

Dr Nasser Eddine MOUHOUB

2015 / 2016

Chapitre IV

Le Problème du Plus court chemin

Problème du Plus court chemin

Étant donné un Graphe valué: $G = (S,A,v)$ avec valuation

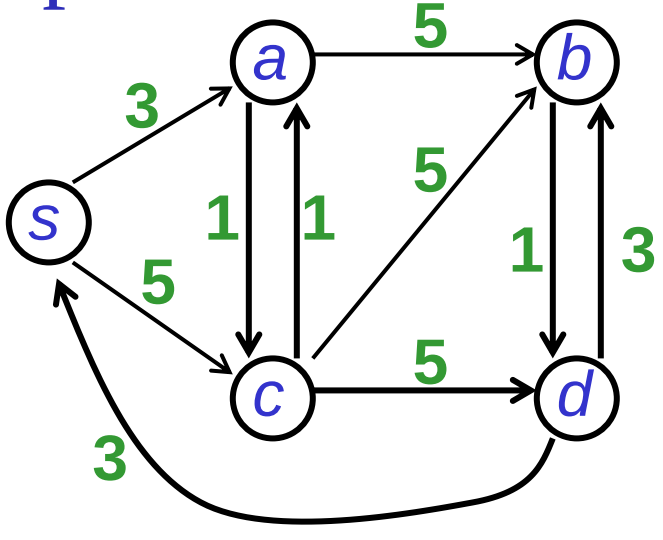
$$v : A \rightarrow \mathbf{R}$$

Source du graphe : $s \in S$

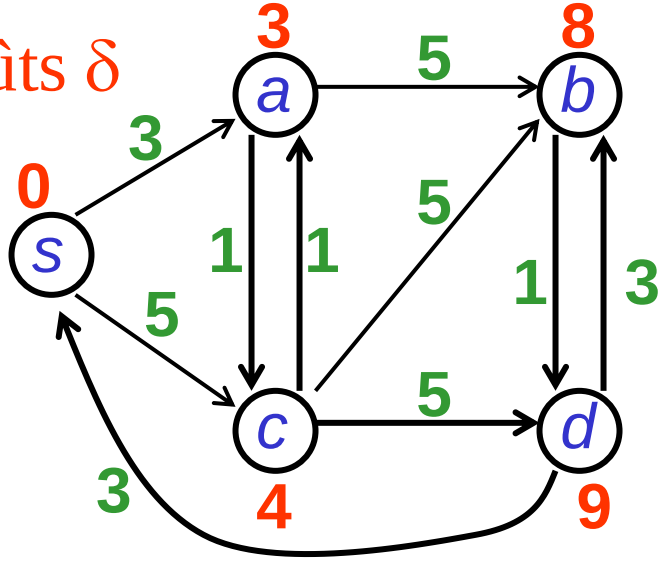
Problème: pour tout $t \in S$ calculer

$$\delta(s, t) = \min \{ v(c) ; c \text{ chemin de } s \text{ à } t \} \cup \{+\infty\}$$

Exemple:

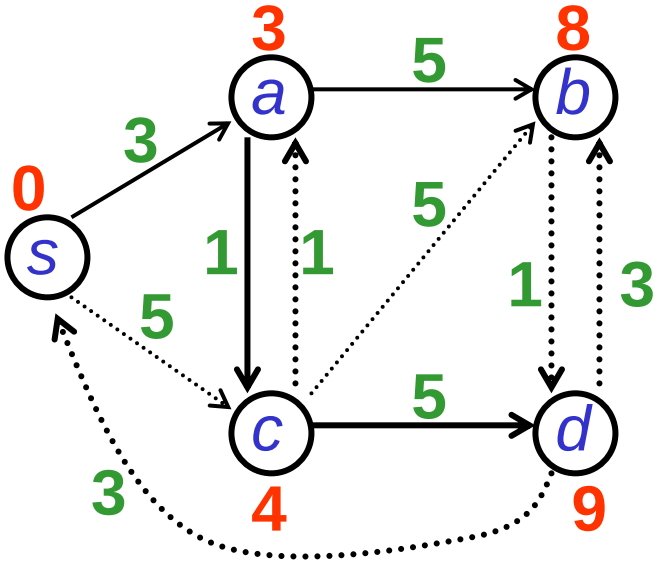


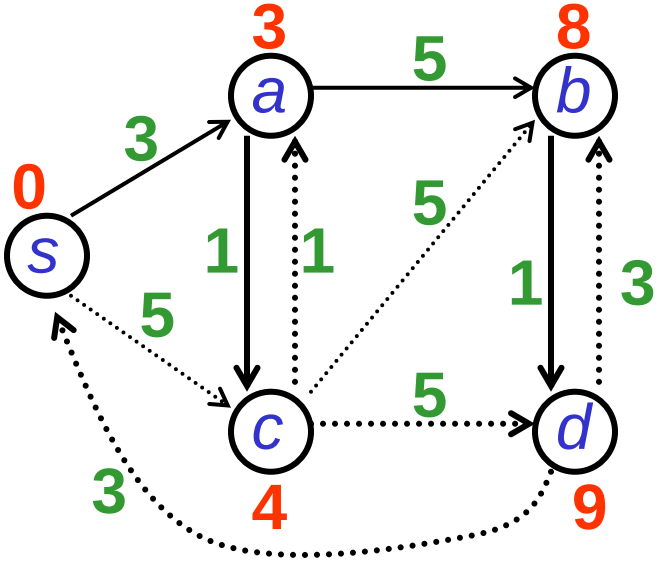
Coûts δ



Arbres du plus court chemin

Arbres de racine *s*
les branches sont des chemins de coût minimal





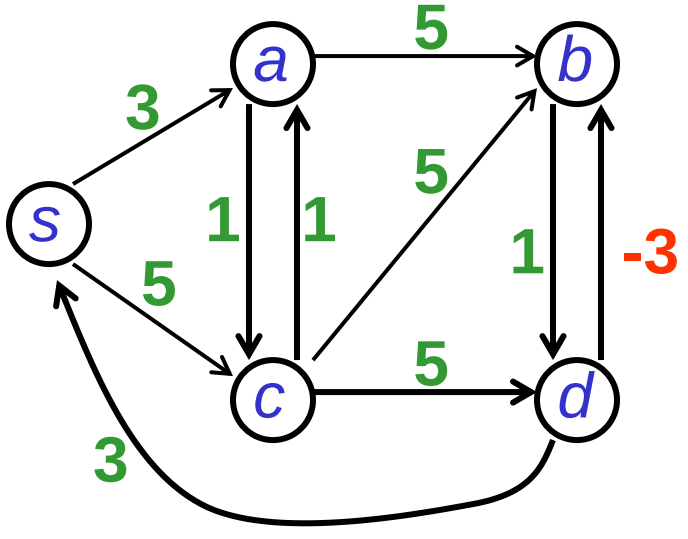
Existence

Proposition

pour tout $t \in S \quad \delta(s, t) > -\infty$

ssi

le graphe n'a pas de circuit de coût < 0 accessible depuis s

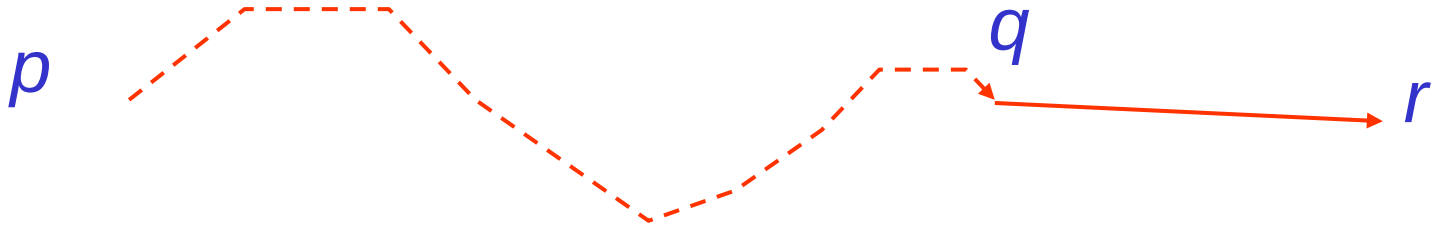


Propriétés de base

Propriété 1 : $G = (S, A, v)$

soit c un plus court chemin de p à r dont l'avant-dernier sommet est q

$$\text{Alors } \delta(p, r) = \delta(p, q) + v(q, r)$$



Propriété 2 : $G = (S, A, v)$

soit c un chemin de p à r dont l'avant-dernier sommet est q

$$\text{Alors } \delta(p, r) \leq \delta(p, q) + v(q, r)$$

Relaxation

Calcul des $\delta(s, t)$ par approximations successives $t \in S$

d(t) = estimation de δ(s, t)

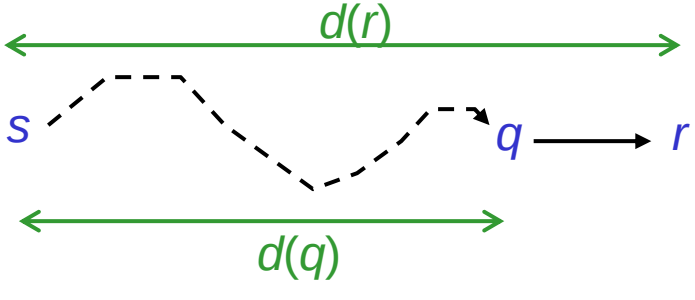
π(t) = prédécesseur de t : avant-dernier sommet d'un chemin de s à t ayant pour coût d(t)

Initialisation de d et π

```
INIT
    pour chaque t ∈ S faire
    { d(t) ← ∞ ;  π(t) ← nil; }
      d(s) ← 0;
```

Relaxation de l'arc (q, r)

```
RELAX(q, r)
    si d(q) + v(q, r) < d(r) alors
      { d(r) ← d(q) + v(q, r) ;  π(r) ← q ; }
```

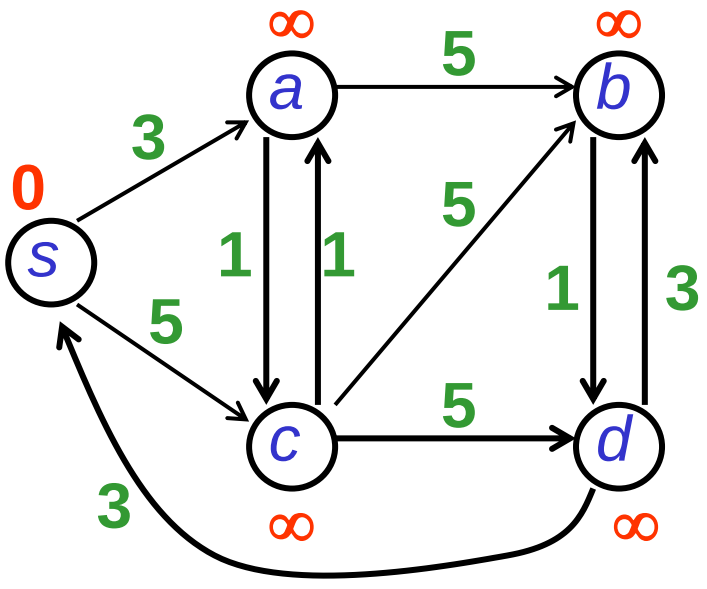


Algorithme de Dijkstra

Condition : v(p, q) ≥ 0 pour tout arc (p, q)

```
début
    INIT;
    Q ← S ;        // l'ensemble de Sommets
    tant que Q ≠ ∅ faire {
        q ← MIN_d(Q) ; Q ← Q - {q} ;
        pour chaque r successeur de q faire
            RELAX(q, r) ;
    }
fin
```

Exemple 1:

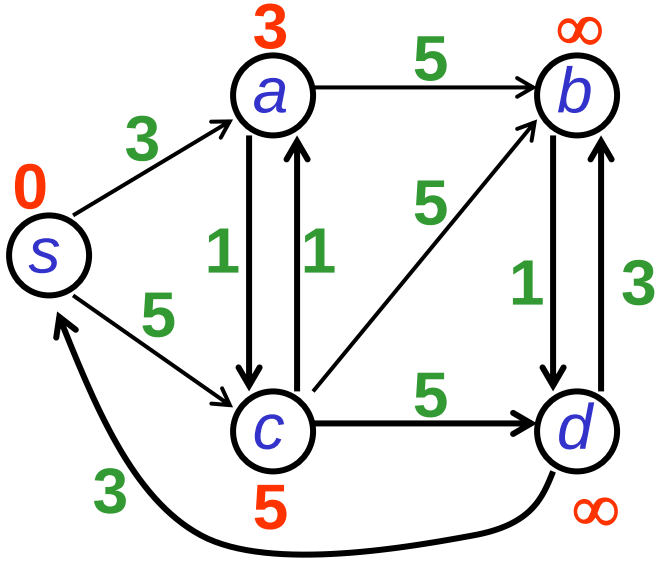


$S = \{ s , a , b , c , d \}$

$Q = \{ s , a , b , c , d \}$

$d = \{0 , \infty , \infty , \infty , \infty \}$

$\pi = \{nil , nil , nil , nil , nil \}$



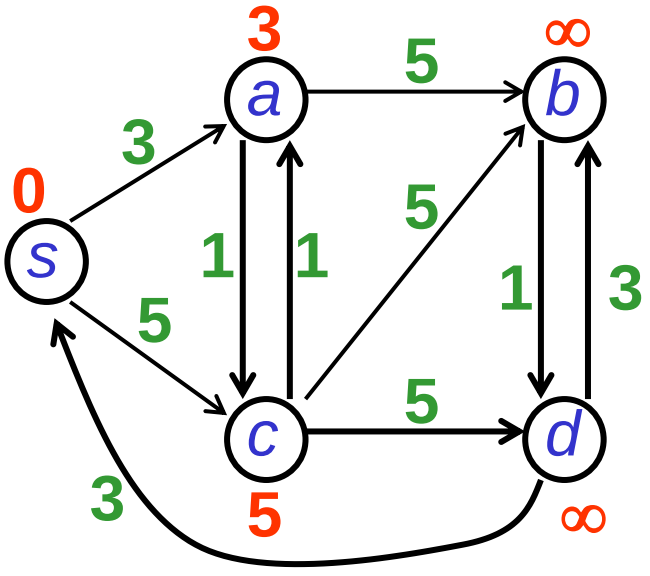
$S = \{ s , a , b , c , d \}$

$Q = \{ a , b , c , d \}$

$d = \{0, 3, \infty , 5 , \infty \}$

$\pi = \{nil , s , nil , s , nil \}$

Exemple 1 suite:

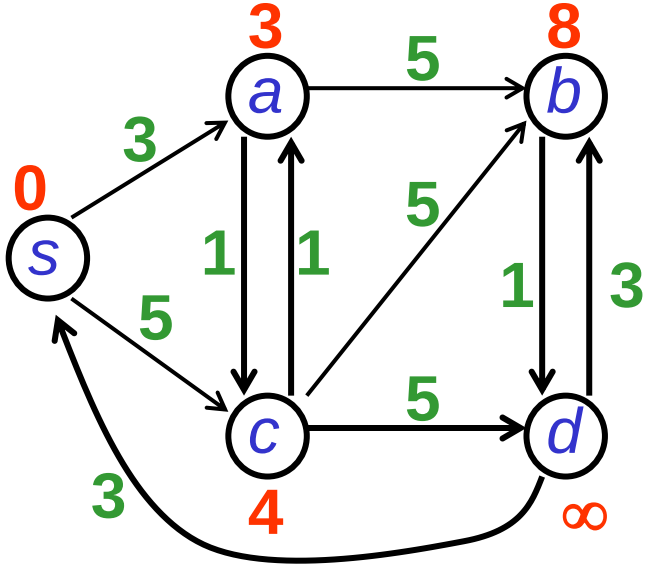


$S = \{ s , a , b , c , d \}$

$Q = \{ a , b , c , d \}$

$d = \{0, 3, \infty, 5 , \infty \}$

$\pi = \{\text{nil} , s , \text{nil} , s , \text{nil} \}$



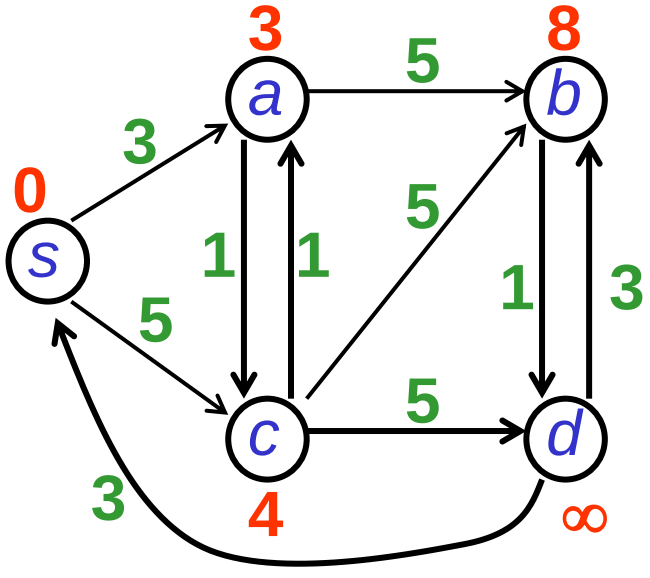
$S = \{ s , a , b , c , d \}$

$Q = \{ b , c , d \}$

$d = \{0, 3, 8 , 4 , \infty \}$

$\pi = \{\text{nil} , s , a , a , \text{nil} \}$

Exemple 1 suite:

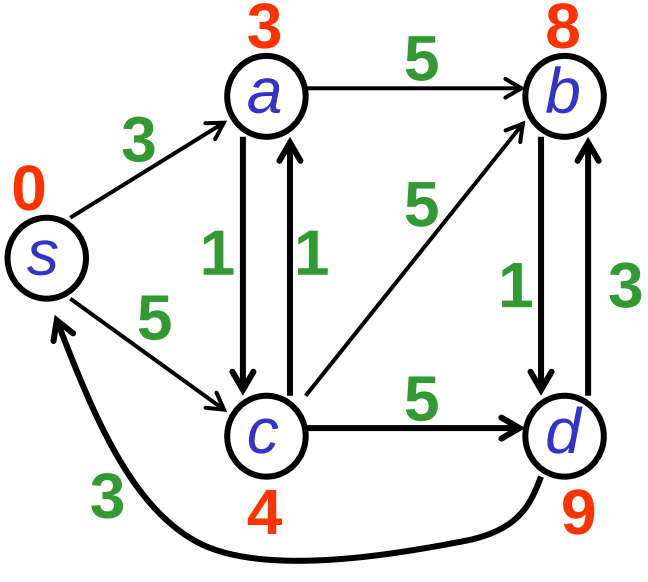


$S = \{ s , a , b , c , d \}$

$Q = \{ \quad b , c , d \}$

d = {0, 3, 8 , 4 , ∞ }

π = {nil , s , a , a , nil }



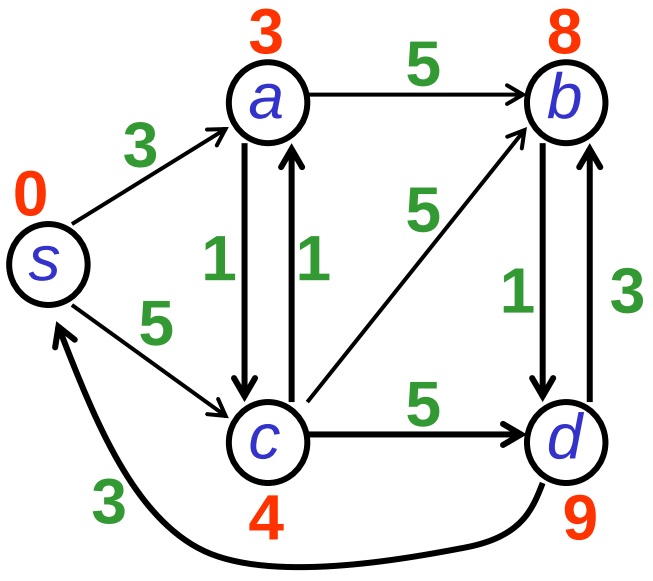
$S = \{ s , a , b , c , d \}$

$Q = \{ \quad b , \quad d \}$

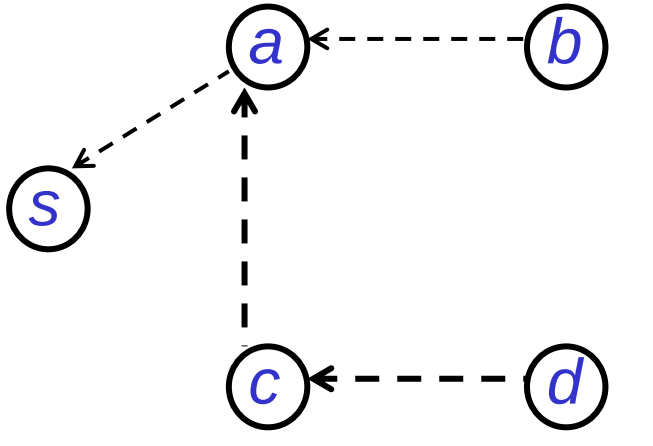
d = {0, 3 , 8 , 4 , 9 }

π = {nil , s , a , a , c }

Exemple 1 suite:



$S = \{ s , a , b , c , d \}$
$Q = \{ b , d \}$
d = {0, 3 , 8 , 4 , 9 }
π = {nil , s , a , a , c }



$S = \{ s , a , b , c , d \}$
$Q = \{ d \}$
d = {0, 3 , 8 , 4 , 9 }
π = {nil , s , a , a , c }

puis $Q = \varnothing$

Implémentation

Si $|S| = n$ et $|V| = m$

Par matrice d'adjacence

temps global $O(n^2)$

Par listes de successeurs

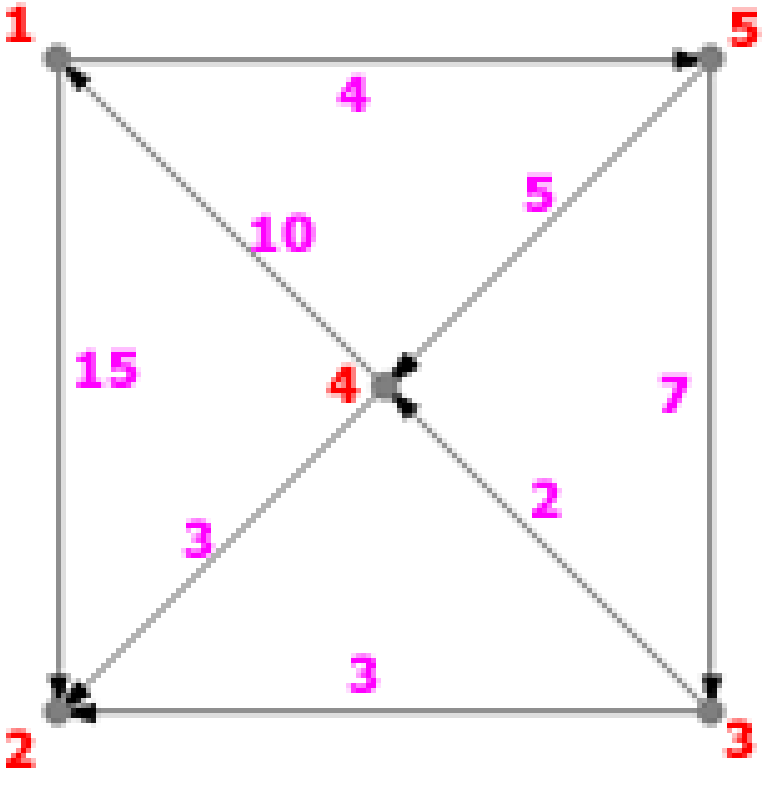
Q : file de priorité (tas)

n opérations $\mathbf{MIN}_d$: O(n . log (n))

m opérations **RELAX** : O((m). log (n))

temps global $O((n + m) . \log (n))$

Exemple 2



Algorithme de Bellman-Ford

L'algorithme de DIJKSTRA MOORE ne permet pas de considérer les arcs négatifs, car une fois qu'un sommet est marqué on ne peut changer ce marquage lors des itérations suivantes. **fixation d'étiquettes**.

On considère donc ici un algorithme qui permet un marquage qui n'est pas définitif tant que le programme n'est pas déterminé (le marquage est modifié itérativement). **correction d'étiquettes.**

L' algorithme de BELLMAN-FORD est valable pour des graphes sans circuit, valués par des longueurs quelconques.

Notations :

S = L'ensemble de n sommets est numéroté de 1 à n.

$\pi(p)$ = Sommet précédant p sur le plus court chemin de l'origine à p.

d = Plus courte distance de l'origine aux autre sommets.

Initialisation : n = nombre de sommets de G

π = tableau initialisé à 0

d = tableau des distances initialisé à $+\infty$ (sauf d(s) = 0)

W = matrice des poids des arcs (∞ si l'arc n'existe pas)

Traitement : k = 1

tant que $k \leq n$ et il y a eu des modifications à l'étape précédente

pour tout sommet x faire

pour tout y successeur de x faire

si d(x) +W(x, y) < d(y) alors

d(y) = d(x) +W(x, y)

π(y) = x

fin

fin

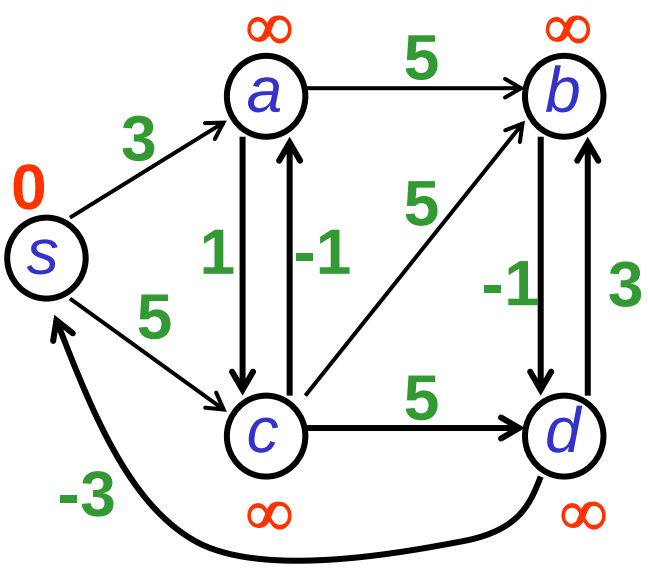
fin

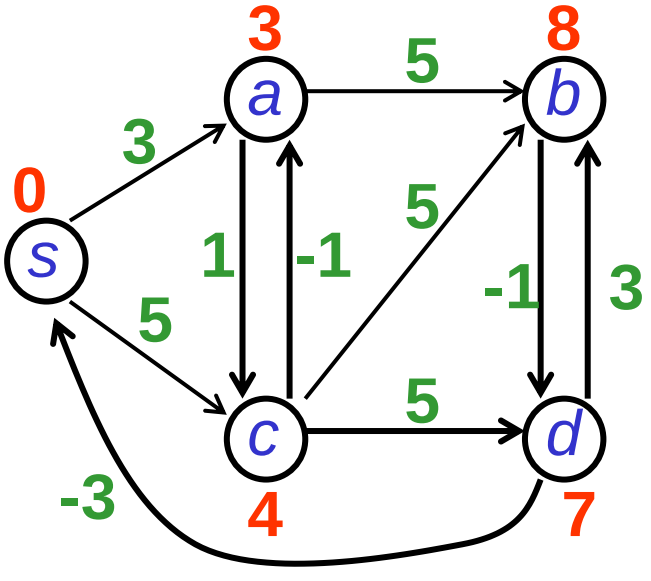
k = k + 1

Fin

Temps : O(n . m)

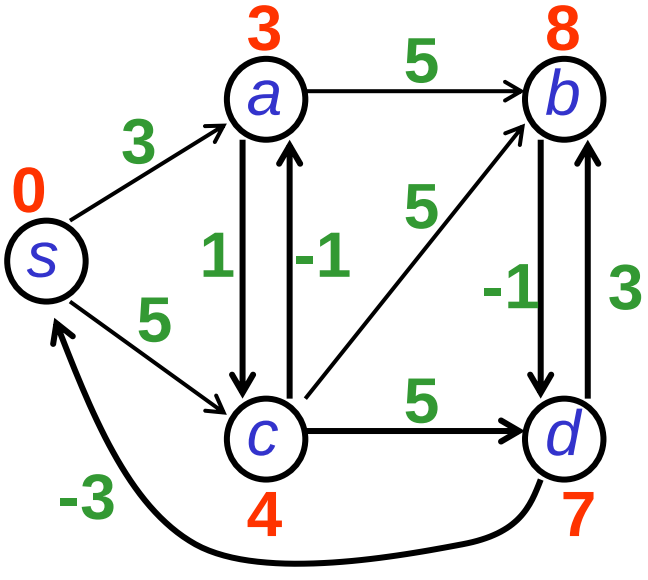
Exemple 1

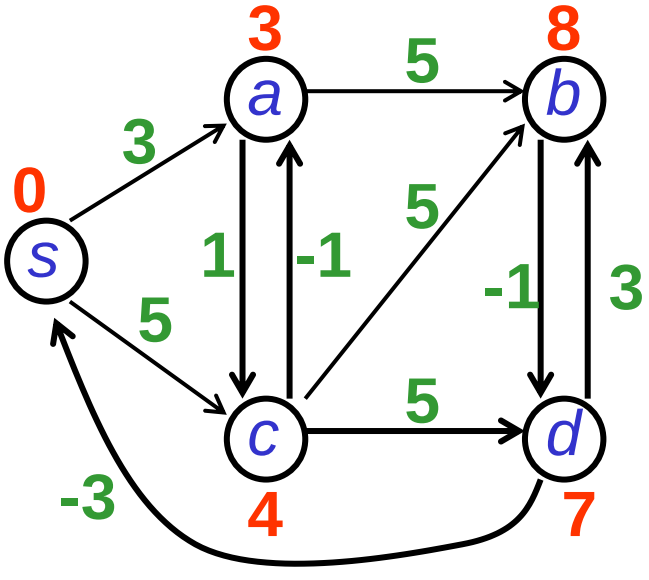




Étape 1 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Exemple 1 (suite)

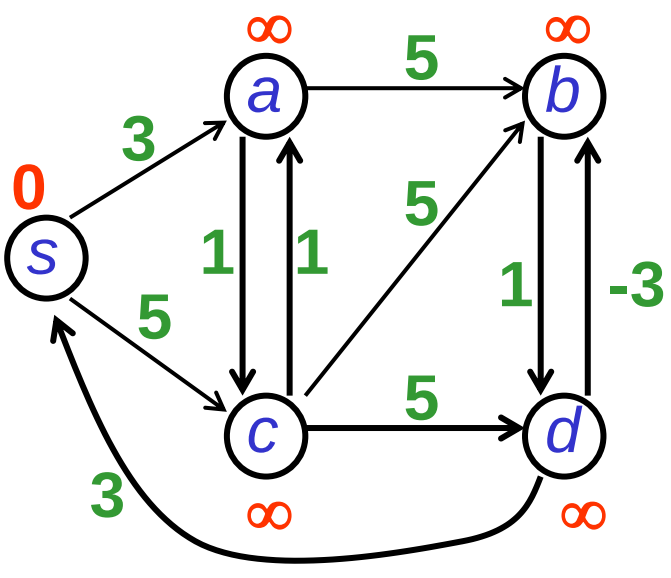


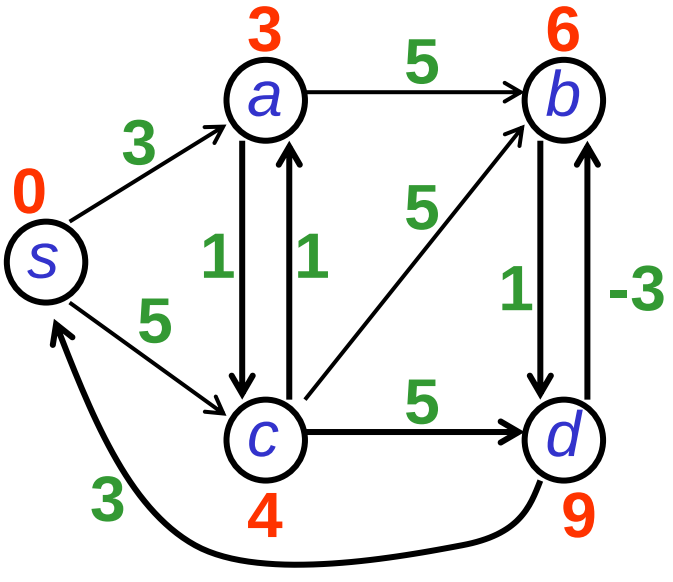


Étape 2 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

pas de réduction possible : coûts corrects

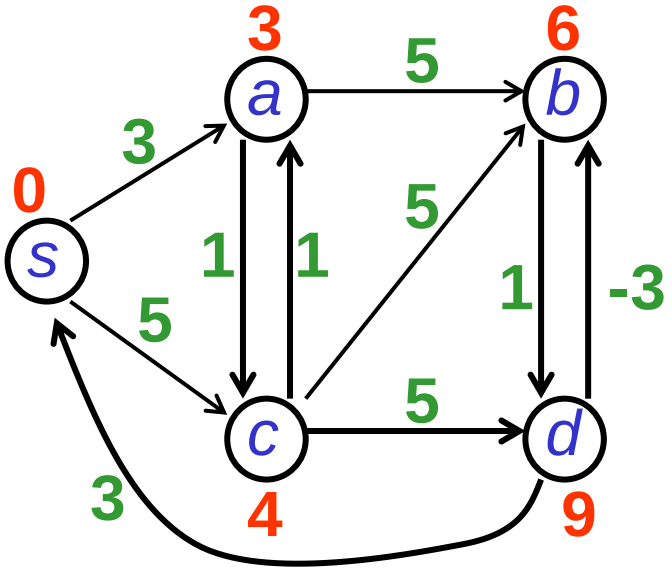
Exemple 2

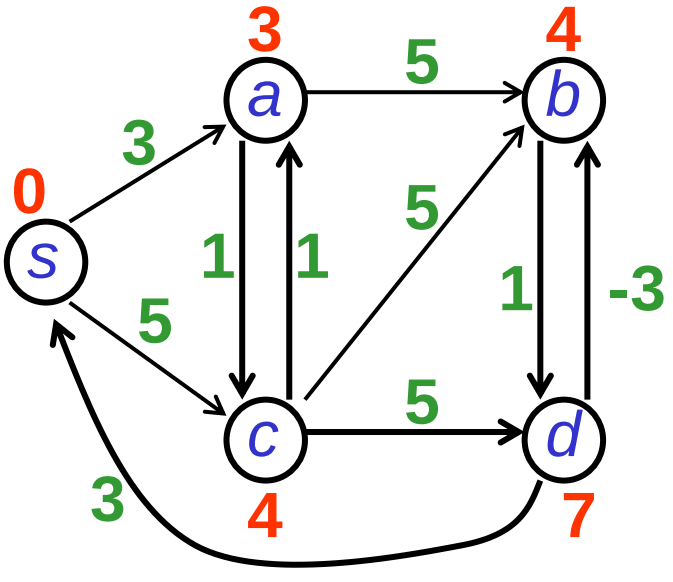




Étape 1 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

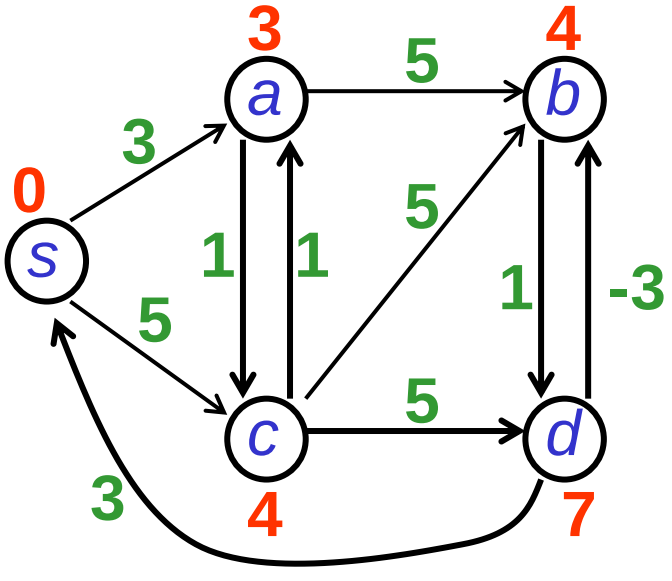
Exemple 2 (suite)

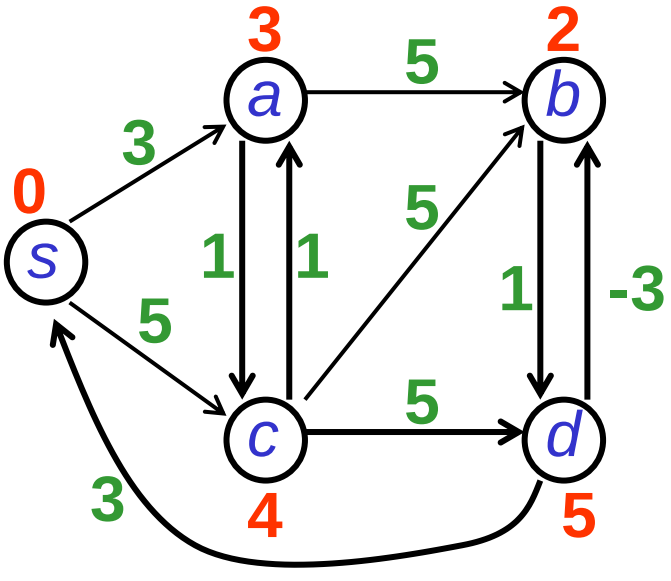




Étape 2 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

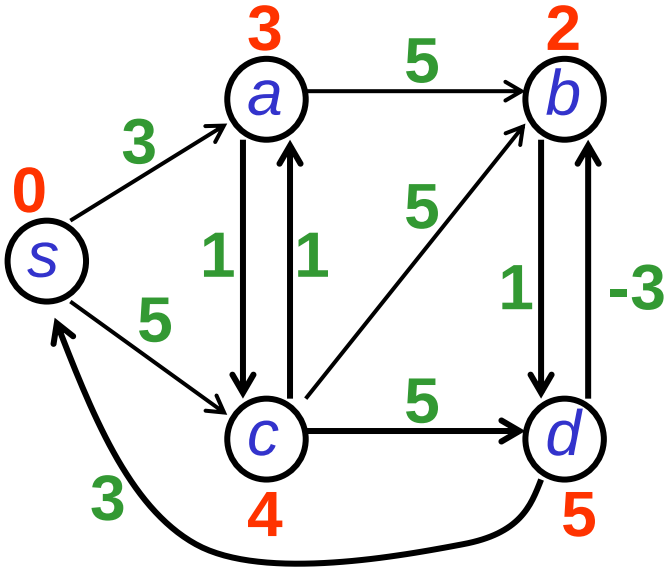
Exemple 2 (suite)

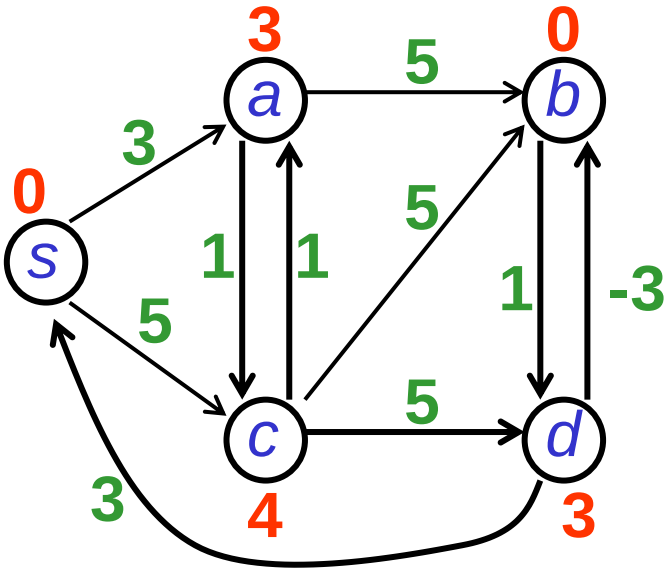




Étape 3 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

Exemple 2 (suite)





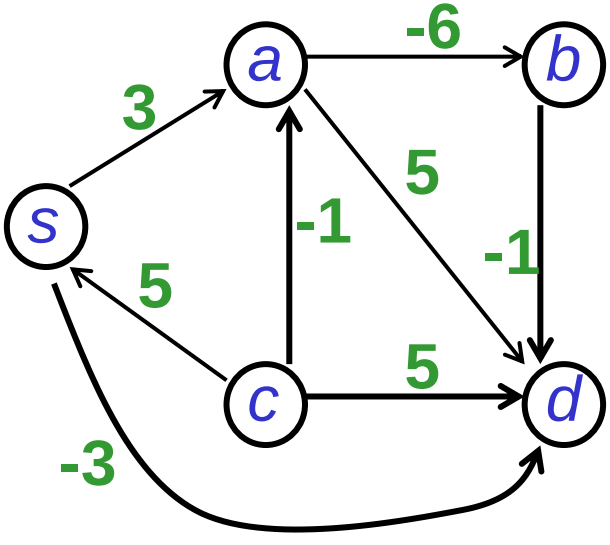
Étape 4 relaxation de tous les arcs dans l'ordre :
(*s*,*a*) (*s*,*c*) (*a*,*b*) (*a*,*c*) (*b*,*d*) (*c*,*a*) (*c*,*b*) (*c*,*d*) (*d*,*b*) (*d*,*s*)

réduction possible : cycle de coût négatif

Graphes acycliques

Tri topologique

- Théorème: Soit G(X; A) un graphe orienté sans circuit. G possède un tri topologique, c'est à dire un ordre $(x_1; x_2; \ldots ; x_n)$ sur les sommets tel que tous les arcs de G sont orientés dans le sens croissant de l'ordre :
- pour tout arc $(x_i , x_j) \subset A$,
  on a $x_i < x_j$.



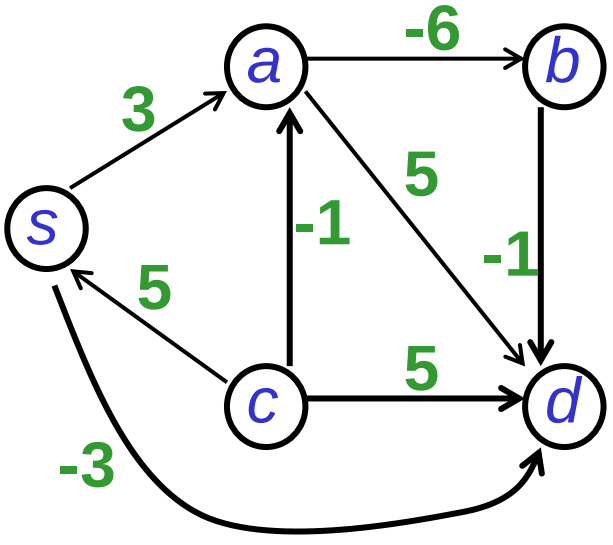
Graphes acycliques

Aucune condition : pour tout arc (p, q), $v\,(p, q) \in \mathbf{R}$
Calcul après ordre topologique

```
début
      INIT;
      pour chaque q ∈ S en ordre topologique faire
            pour chaque r successeur de q faire
                  RELAX(q, r) ;
Fin
```

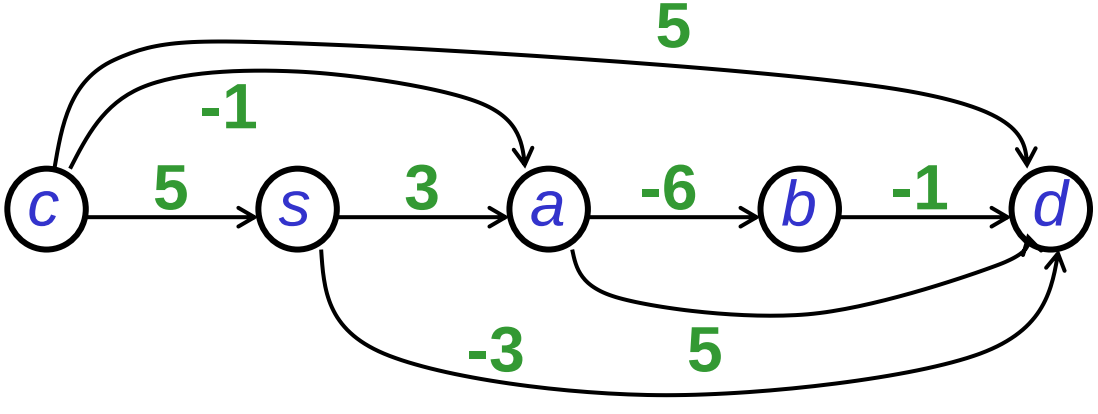
Temps : $O(|S| + |A|)$
chaque sommet et chaque arc est examiné une fois

Exemple

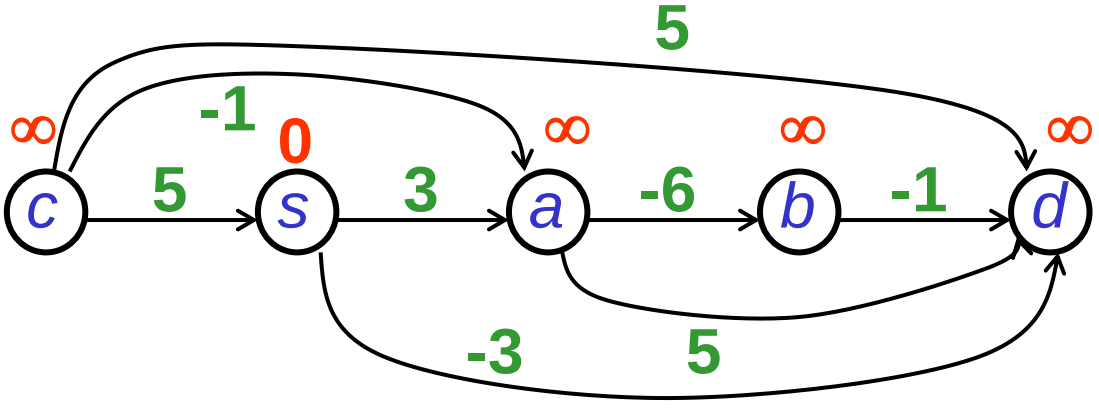


Ordre topologique

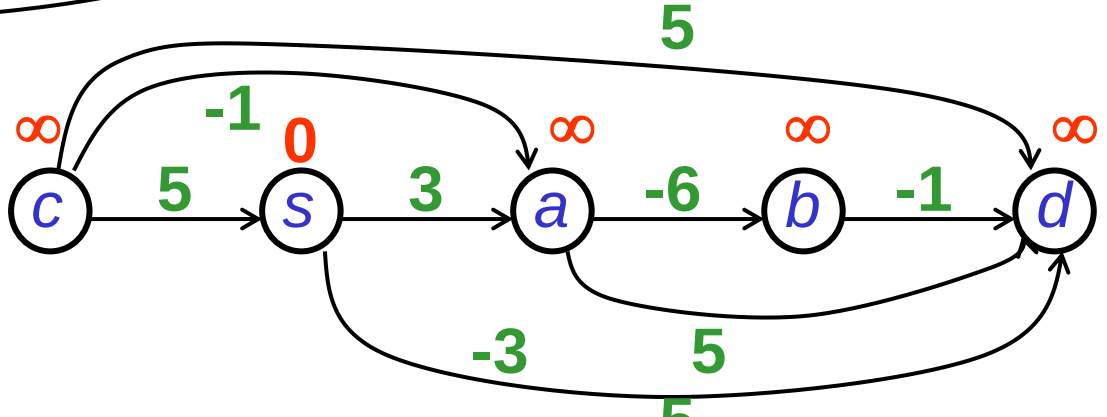
c, *s*, *a*, *b*, *d*



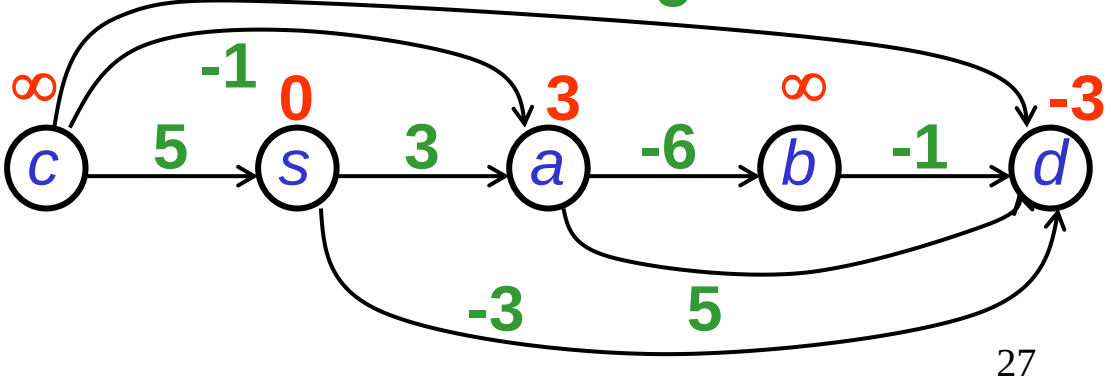
Calcul des coûts



Examen de *c*

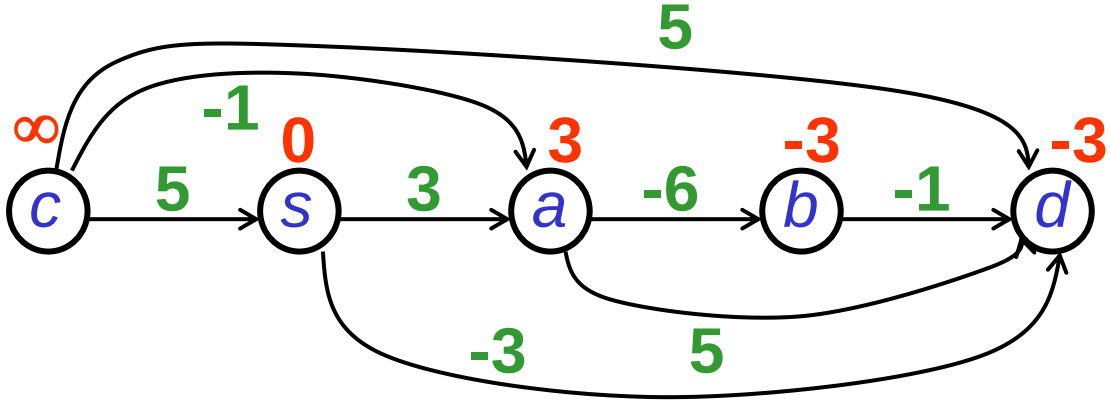


Examen de *s*

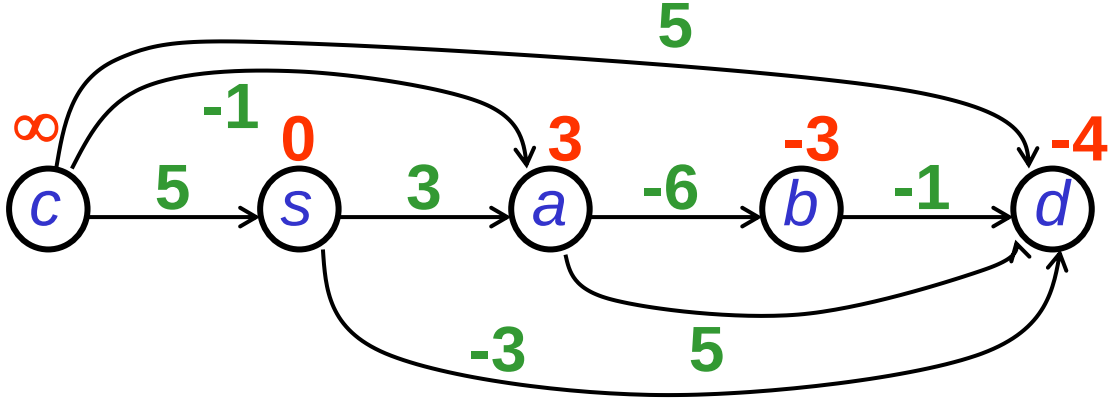


Calcul des coûts (suite)

Examen de *a*



Examen de *b*



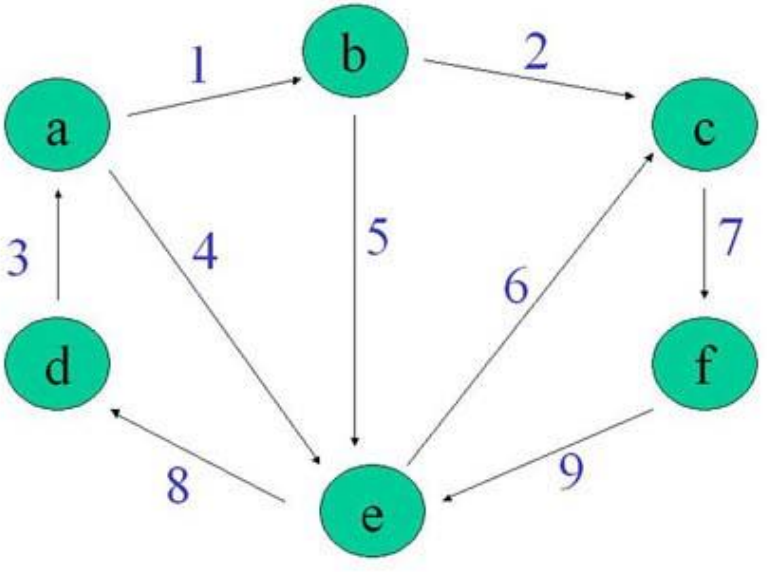
Examen de *d* inutile

Cycle et circuit

- Un *cycle* est une chaîne telle que :
  - le même arc ne figure pas deux fois dans la séquence ;
  - les deux sommets aux extrémités de la chaîne coïncident.
- Un *cycle élémentaire* est un cycle ne rencontrant pas deux fois le même sommet (excepté le sommet initial qui coïncide nécessairement avec le sommet final).

Exemples

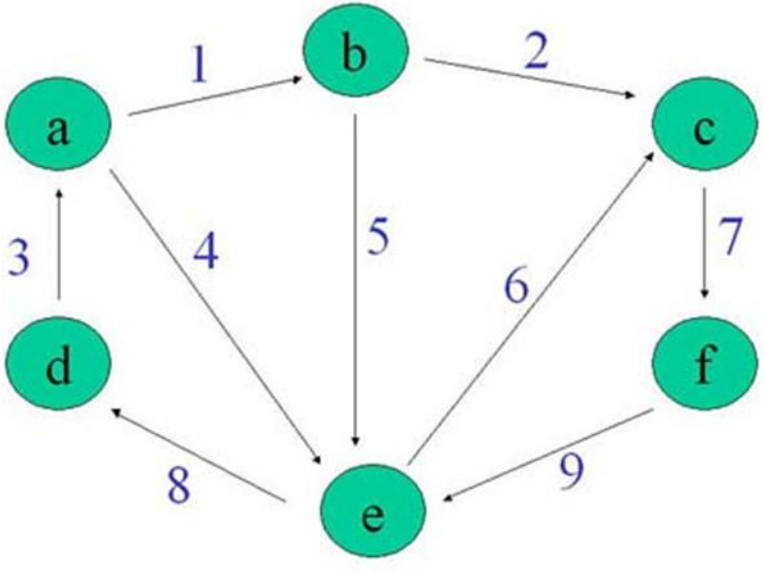
- C1 = (1,5,4) est un cycle **élémentaire** de longueur 3
- C2 = (1,2,7,9,8,3) est un cycle **élémentaire** de longueur 5
- (1,5,6,2,5,4) n'est **pas un cycle**
- C3 = (3, 4, 9, 7, 6. 8) est un cycle de longueur 7 qui n'est **pas élémentaire**



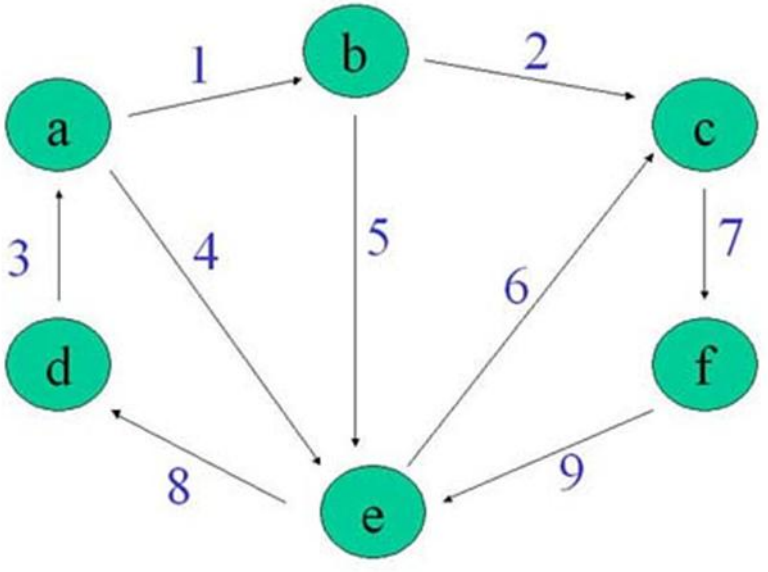
Notation Vectorielle

- Les arcs du graphe étant numérotés de 1 à m, on peut faire correspondre à tout cycle un m-uplet (un "vecteur") composé de -1, 1 et 0 de la manière suivante :
- Si l'arc n'appartient pas au cycle, on met un "0"
- Si l'arc appartient au cycle et est parcouru dans le bon sens (on le qualifie alors de "**direct**"), on met un +1
- Si l'arc appartient au cycle mais parcouru dans le mauvais sens (on le qualifie alors de "**inverse**"), on met un -1

- Exemples :
- C1 = (1,5,4)
- **C1 = (1, 0, 0, -1, 1, 0, 0, 0, 0)**
- C2 = (1,2,7,9,8,3)
- **C2 = (1, 1, 1, 0, 0, 0, 1, 1, 1)**
- C3 = (3, 4, 9, 7, 6. 8)
- et **C3 = (0, 0, 1, 1, 0, -1, -1, 1, -1)**



- On peut donc assimiler un cycle à un vecteur et définir la somme (vectorielle) de 2 ou plusieurs cycles.



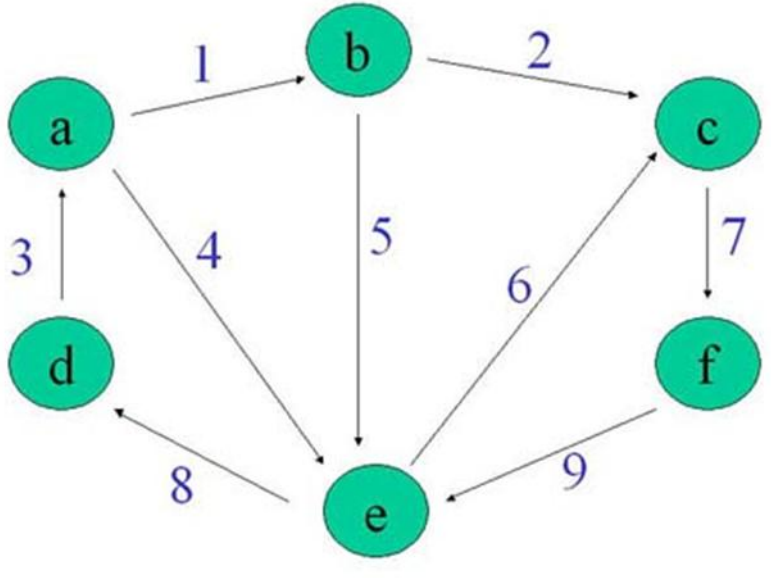
- *Exemple :* C3 = (3, 4, 9, 7, 6. 8)
- est la somme du cycle (3,4,8) et du cycle (9,7,6).
- (0,0,1,1,0,0,0,1,0) + (0,0,0,0,0,-1,-1,0,-1) = (0,0,1,1,0,-1,-1,1,-1)
- **Propriétés:**
  - ✓ Tout cycle est la somme de cycles élémentaires sans arc commun.
  - ✓ Un cycle est élémentaire si et seulement s'il est minimal.

- On peut donc définir une **base de cycles** (au sens vectoriel) comme étant
- une famille de cycles **libre** (tous les cycles indépendants, aucun ne peut se définir comme combinaison linéaire des autres)
- et **génératrice** (tout cycle peut s'écrire comme combinaison linéaire des cycles de la famille).
- Le Cardinal (Nombre d'éléments) commun de toutes les bases de cycles sera appelé **nombre cyclomatique** et noté μ.

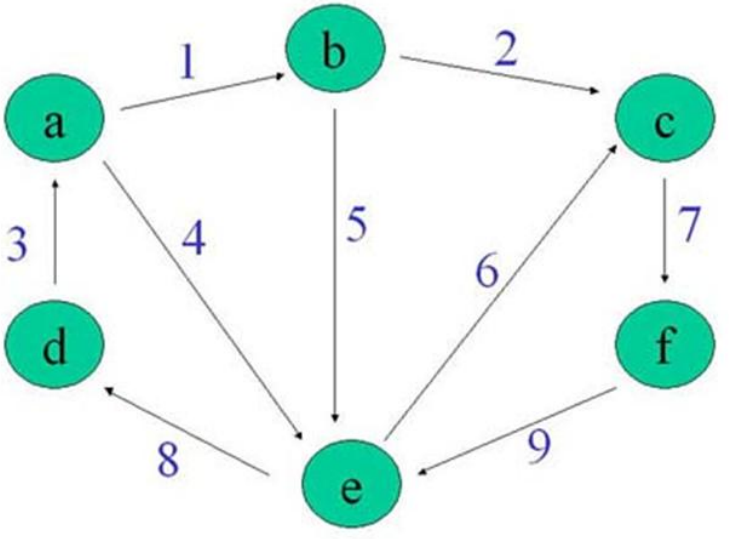
- **Propriété :** Soit un graphe G d'ordre n (n sommets) avec m arcs et p composantes connexes: **$\mu(G) = m - n + p$.**
- **Définition:** Soit A un ensemble de sommets du graphe, on appellera **cocycle** associé à A, qu'on notera **$\omega(A)$**, l'ensemble des arcs **incidents** à A, ceux qui quittent A seront notés positivement et ceux qui pointent vers A seront notés négativement.
- Un cocycle c'est donc l'ensemble des arcs qui relient A aux autres sommets du graphe; si on enlève les arcs du cocycle on "déconnecte" A.
    - **Cycle et cocycle sont des notions duales.**

- **Exemple** : En réutilisant le graphe ci-dessus, si A = {b, e}
  ω(A) = {-1 , +2 , -4 , +6 , +8 , -9}

- On peut, bien sûr appliquer la notation vectorielle définie pour les cycles, aux cocycles;
- on obtiendrait pour l'exemple ci-dessus le vecteur :
- (-1,1,0,-1,0,1,0,1,-1).



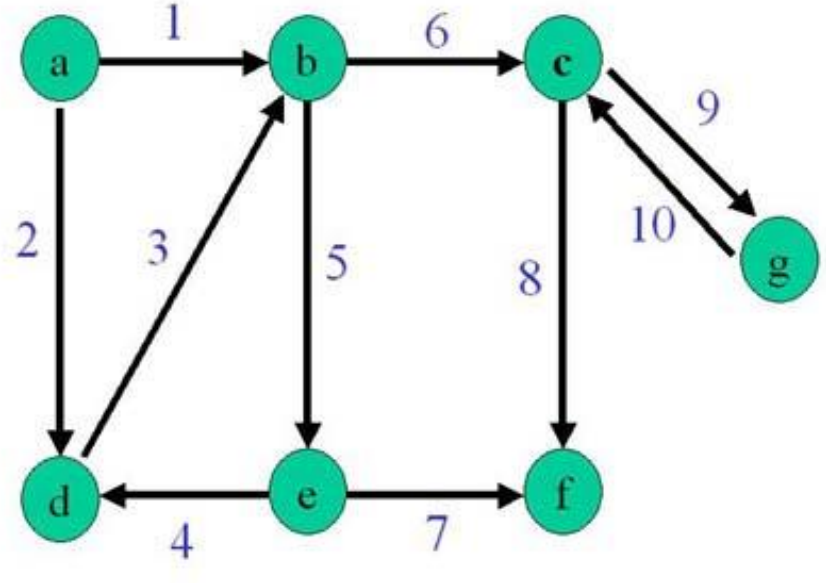
- **Définition :** Un cocycle sera dit **élémentaire** quand il est composé d'arcs reliant deux sous-ensembles de sommets connexes qui partitionnent une composante connexe du graphe (donc tous les sommets en cas de graphe connexe).



- Le cocycle donné dans l'exemple n'est pas élémentaire parce qu'il est composé d'arcs reliant {b, e} à {a, c, d, f}
- {b,e} est connexe,
- ce n'est pas le cas de {a, c, d, f}.
- Le cocycle associé à {a, b, d} dont la notation vectorielle est (0, 1, 0, 1, 1, 0, 0, -1, 0) est élémentaire.

- Le Cardinal (Nombre d'éléments) commun de toutes les bases de cocycles sera appelé **nombre cocyclomatique** et noté λ
- Soit un graphe G d'ordre n (n sommets) avec p composantes connexes: **$\lambda(G) = n - p$.**

- **Exercice:**
- 1) Déterminer tous les cycles élémentaires
- 2) Extrayez-en une base
- 3) Déterminer tous les cocycles élémentaires
- 4) Extrayez-en une base



C hapitre V

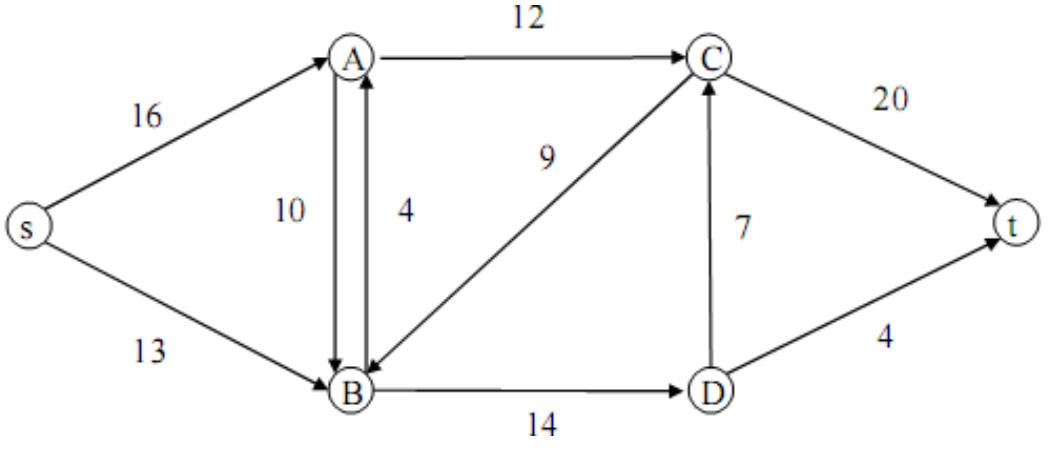
Réseaux de flots
Problème du flot maximum

PROBLEME DE FLOTS

1. Les réseaux de transport
2. Le flot maximum et la coupe minimum
3. L'algorithme de Ford et Fulkerson

Les réseaux de transports

- Réseau de transport : graphe orienté avec pour chaque arc une *capacité*.
- La capacité *c(a)* *est un entier positif ou nul.*
- Il y a aussi une source *s et un puits t*.
- *Aucun arc n'arrive à la source*
- *Et* aucun arc ne quitte le puits.

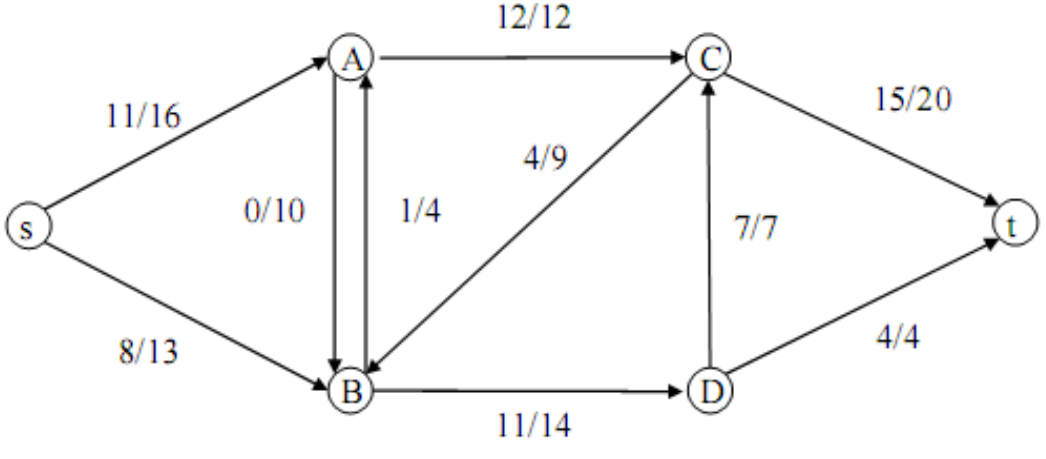


Réseau de transport avec les capacités

- Un flot est une fonction entière positive ou nulle f définie sur les arcs satisfaisant :

- Contrainte de capacité: $f(a) \leq c(a)$;

Pour le graphe si dessous: $11 \leq 16,\ 8 \leq 13, 0 \leq 10,\ 12 \leq 12, \ldots, 4 \leq 4$



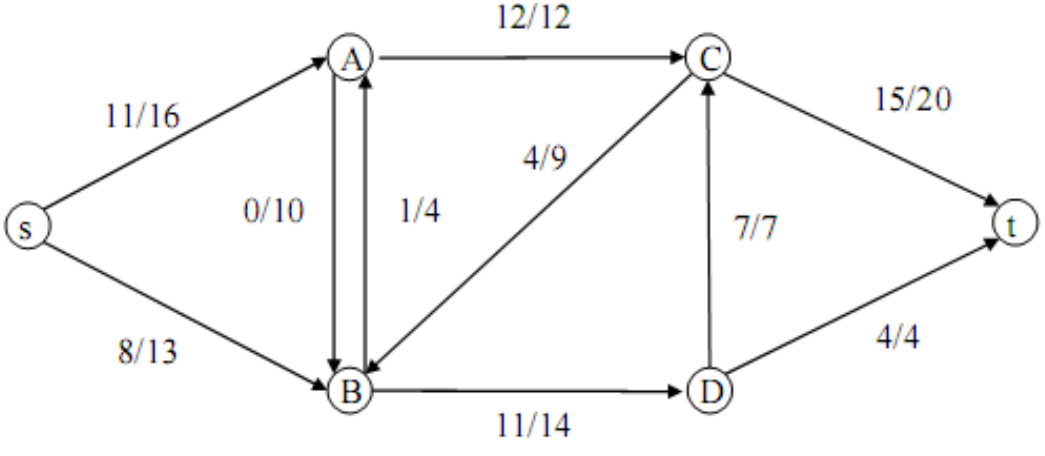
Un flot sur le réseau de transport

Conservation du flot:

pour tout sommet autre que *s* et *t*, la somme des flots sur les arcs entrants et la somme des flots sur les arcs sortants sont égales.

Exemples : circuits électriques ou hydrauliques, réseaux de communication, modélisation de transports

Sommet A : 11+1 = 12 + 0 , sommet C: 12+7=4+15

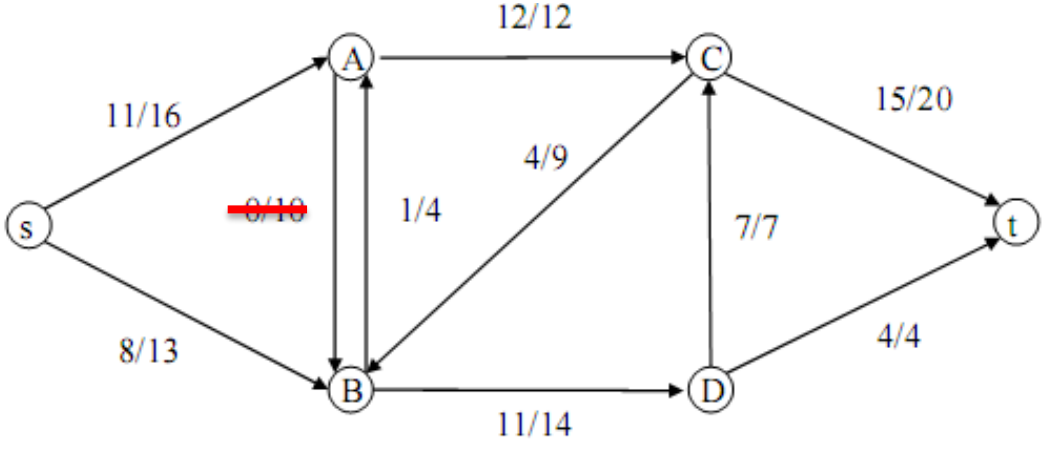


Un flot sur le réseau de transport

Quand deux arcs en sens inverse relient deux sommets, on peut toujours annuler la fonction flot sur l'un des deux.

Propriété :

la somme des flots sur les arcs sortant de source et la somme des flots sur les arcs arrivant au puits sont égales ; cette valeur est la valeur du flot $|f|$;

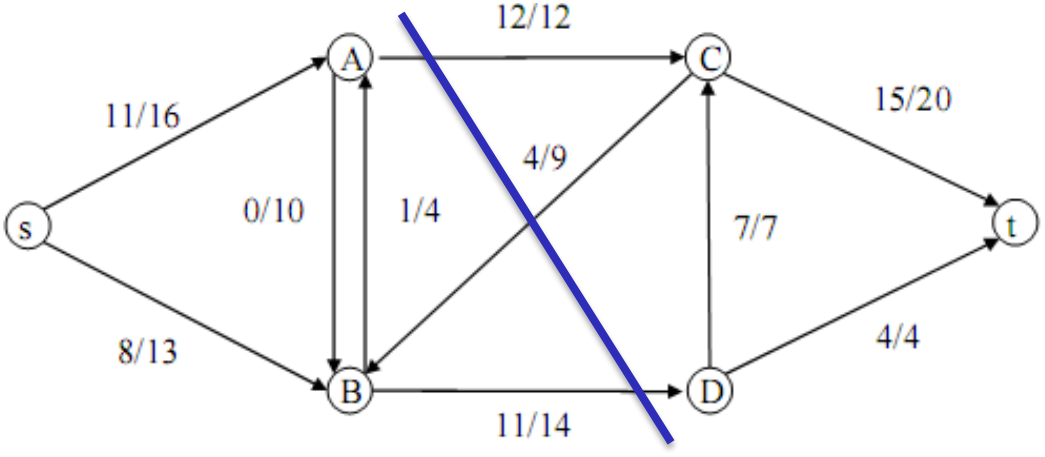


Un flot sur le réseau de transport

Une coupe est une partition de l'ensemble des sommets en 2 parties disjointes, l'une contenant la source et l'autre le puit:

$$E \cup F = A\,, \quad E \cap F = \varnothing\,; \quad s \in E\,,\ t \in F$$

La capacité C(E, F) d'une coupe est la somme des capacités des arcs de E a F.



Un flot sur le réseau de transport

C(E, F) = 26

Propriété :

Le flux de E à F dans un flot *f* est : $f(E,F) = \sum_{u \in ExF} f(u)$

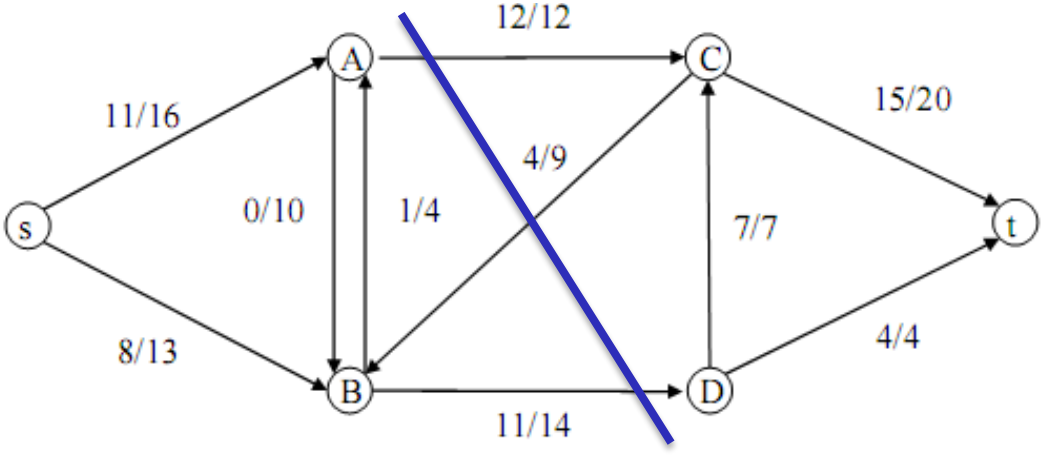
et le flux orienté de la coupe (E , F) ou (flot net traversant la coupe) est:

$$|f| = \Delta(E,F) = f(E,F) - f(F,E)$$

C(E, F) = 12+14 = 26

f(E, F) = 12+11 = 23

Δ(E, F) = $|f|$ = (12+11) - 4 = 19



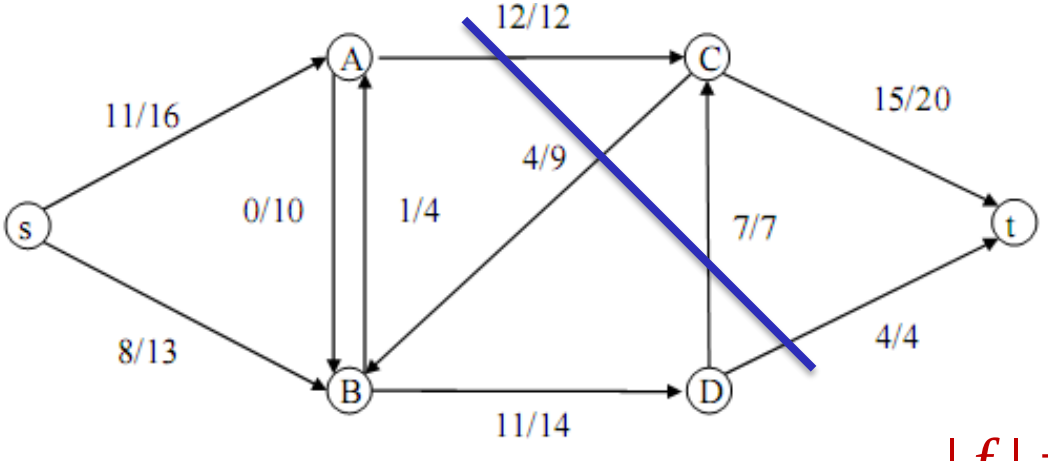
Un flot sur le réseau de transport

La deuxième propriété est donc que le flot net traversant une coupe ne dépend pas de la coupe.

Tout flot a pour valeur $V_f = f(\{s\}, X\backslash\{s\}) = \Delta(\{s\}, X\backslash\{s\})$.

Lemme: Plus généralement $V_f = \Delta(E, F)$ pour toute coupe.

$|f|$ est inférieur à la capacité de n'importe quelle coupe.



Un flot sur le réseau de transport

$|f| = (12+7+4) - 4 = 19$

Le flot maximum et la coupe minimum

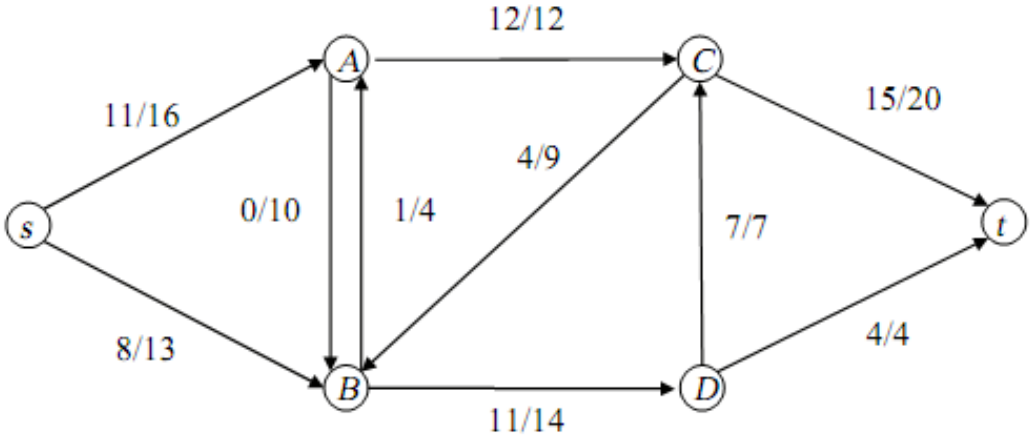
Il existe toujours un flot possible qui est le flot nul.

Problème : comment trouver un flot qui a la valeur maximum ?

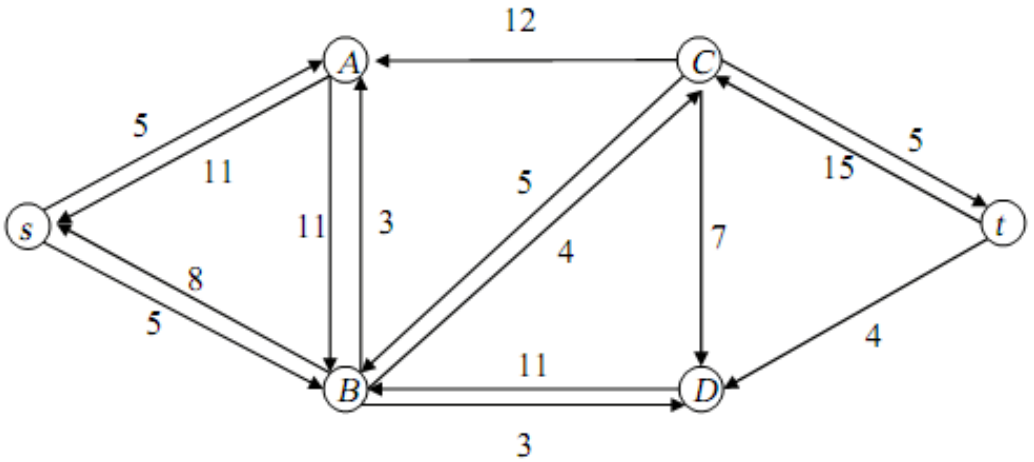
- ✓ Recherche d'un *chemin améliorant.*
- ✓ Déterminer le *réseau résiduel :*

Un flot est saturé si sur tout chemin de s a t il existe un arc a tel que $f(a) = c(a)$.

pour chaque arc $a = uv$, $f(a) \leq c(a)$, *on peut augmenter le flot de* $c(a) - f(a)$, *et on peut le diminuer de* $f(a)$, *donc faire passer un flot* $f(a)$ *sur un arc* $-a = vu$. *Si cet arc existe déjà avec une capacité* $c(-a)$, celle-ci s'ajoute à $f(a)$.



Le flot

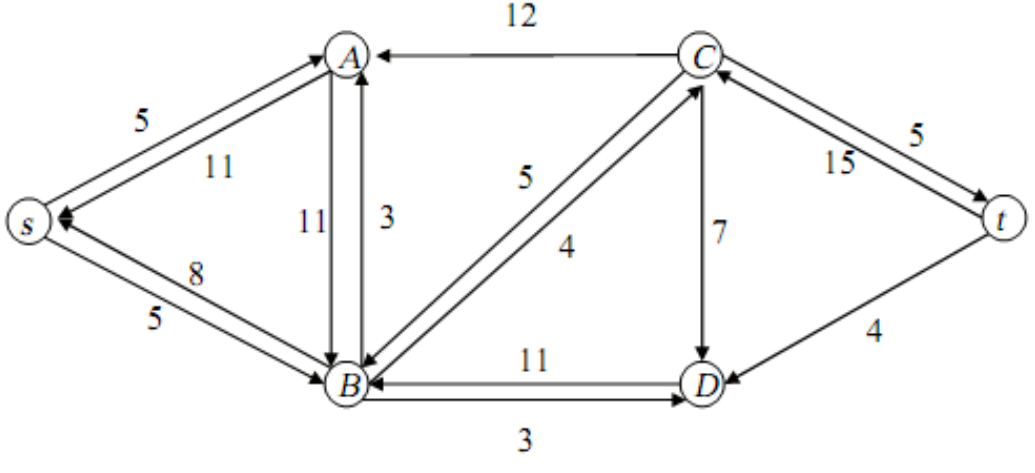


Le réseau résiduel correspondant

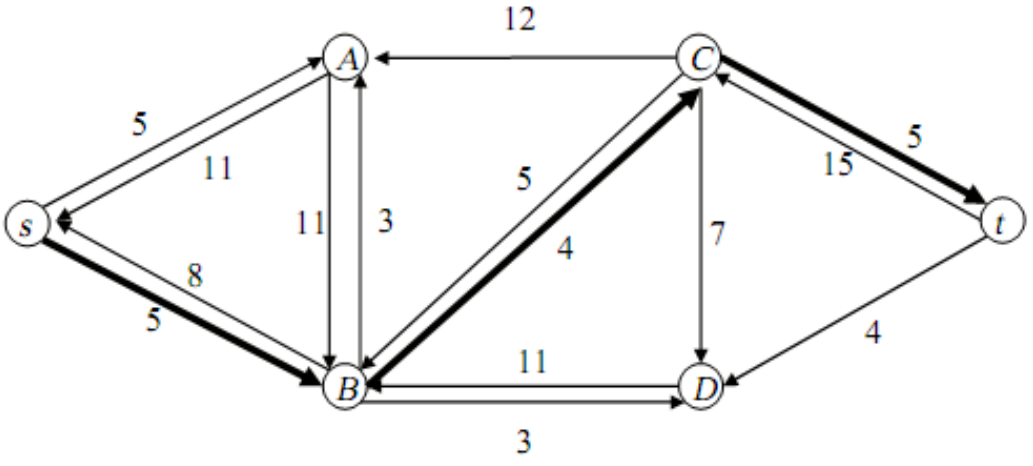
Le graphe orienté avec ces capacités est le *réseau résiduel.*

On cherche un chemin de *s* à *t* dans le réseau résiduel.

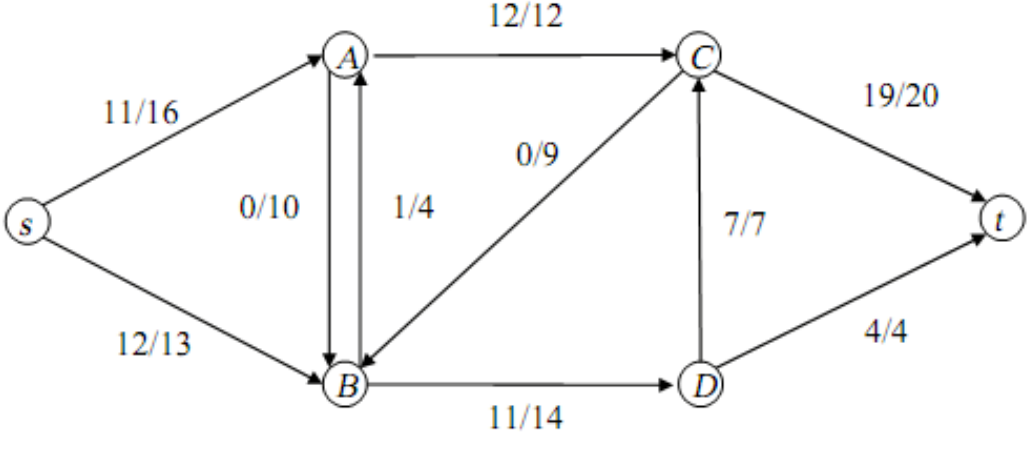
Il correspond à une possibilité d'amélioration du flot en modifiant de la valeur du minimum des capacités résiduelles sur le chemin.



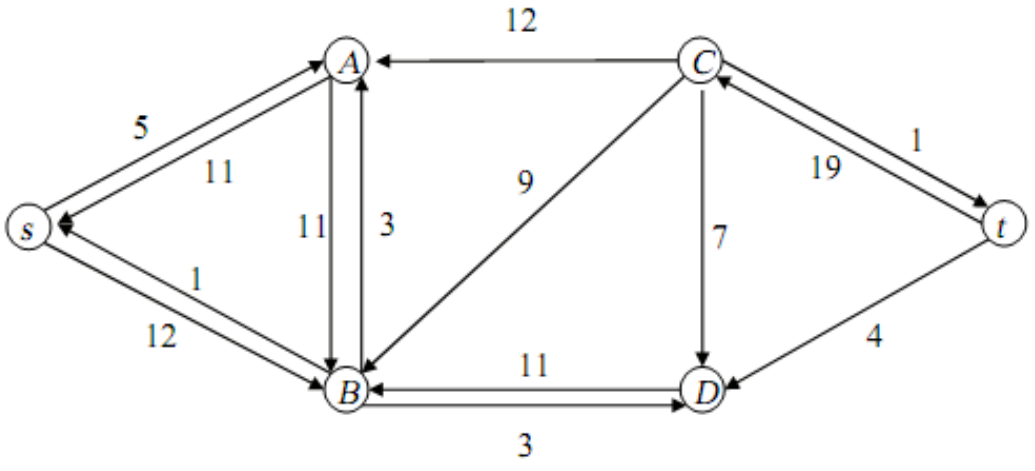
Le réseau résiduel



Un chemin améliorant



Le flot après amélioration



Le nouveau réseau résiduel

Dans ce réseau, il n'y a pas de chemin de *s* à *t*, *donc pas de chemin* améliorant.

Théorème (flot maximum et coupe minimum)

Si *f est un flot dans un réseau de transport, les trois conditions* suivantes sont équivalentes :

1. *f est un flot maximum ;*
2. Le réseau résiduel de *f* ne contient aucun chemin améliorant ;
3. Il existe une coupe *E/F dont la capacité vaut* | *f* |.

Remarque :

La condition 3. implique que | *f* | *est la valeur minimum des* capacités des coupes du réseau, puisqu'on sait déjà que | *f* | *est* inférieur à la capacité de n'importe quelle coupe.

Valeur du flot maximal = Capacité de la coupe minimale.

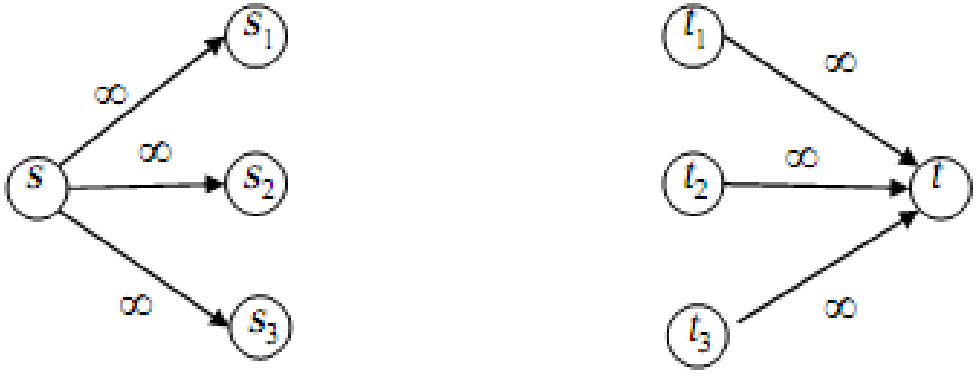
L'algorithme de Ford et Fulkerson

- On part d'un flot quelconque (éventuellement nul) ;
- On fabrique le réseau résiduel ;
- On cherche un chemin améliorant ;
- On itère jusqu'à ce qu'on ne trouve plus de tel chemin.

Variantes et applications :

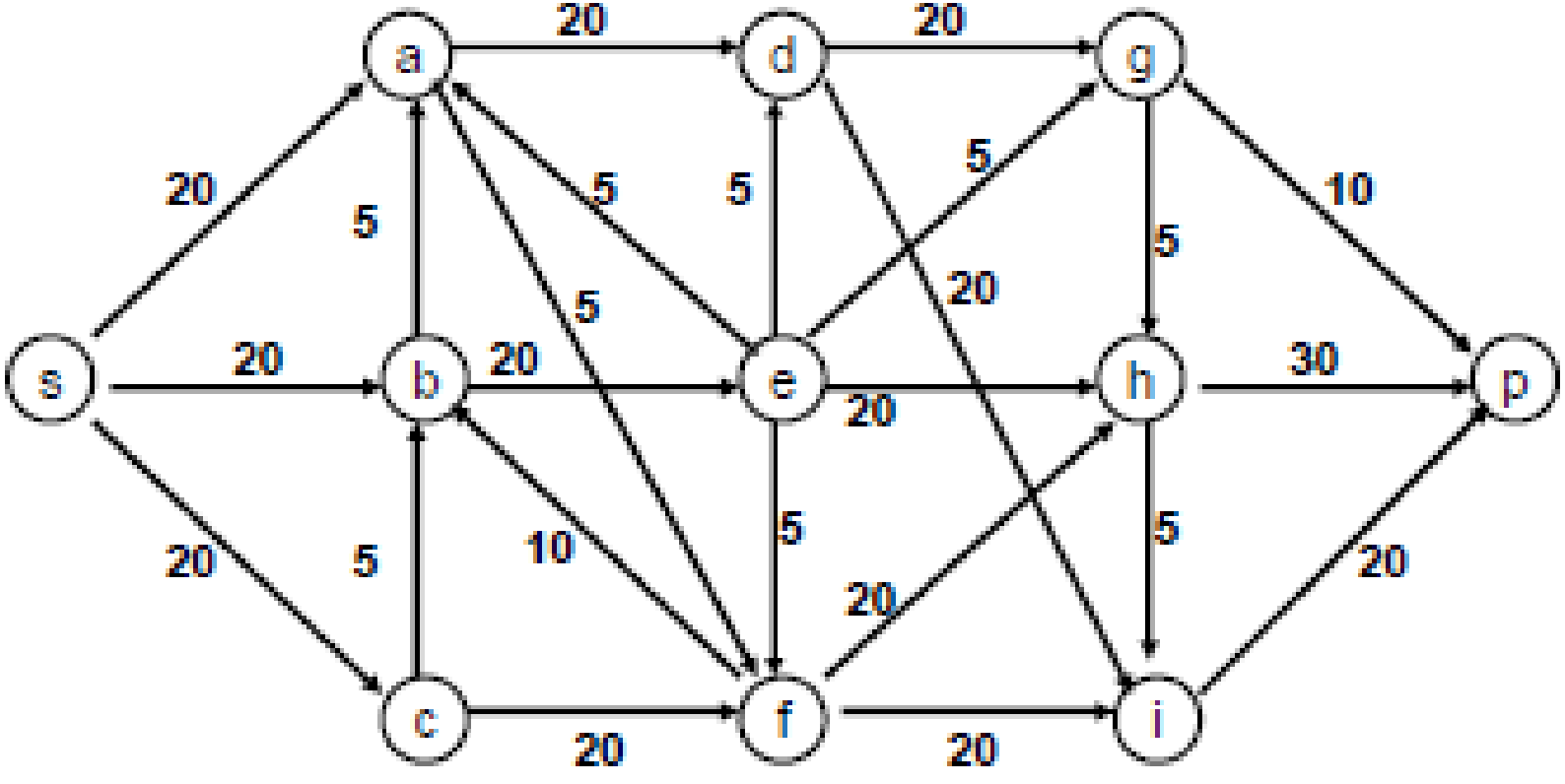
Parfois, il y a plusieurs sources et plusieurs puits.

On peut dans ce cas rajouter une "super-source" et un "super-puits" reliés respectivement aux sources et aux puits par des arcs de capacité infinie.



Adjonction d'une super-source et d'un super-puits

Valeur de ce flot?



Flot maximum?