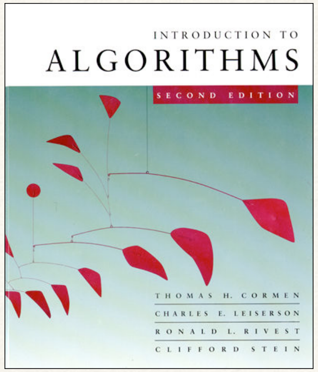
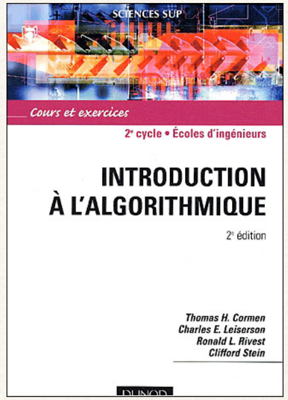



# Plus courts chemin pour tout couple de sommets

## Théorie des graphes

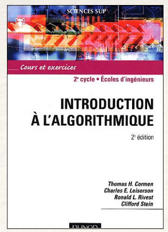
Cours de Lélia Blin

3eme année de Licence

# Poids d'un chemin

- Dans un graphe valué le poids w(p) d'un chemin p est la somme des poids des arêtes le long du chemin.
- Notation mathématique:

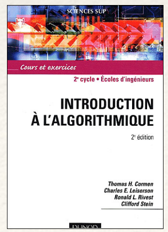
$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$



# Plus court chemin

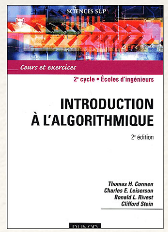
- Le plus court chemin entre deux sommets s et t est défini comme le chemin de plus faible poids reliant s et t.
- Notations mathématique:

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



# Plus court chemin lemme

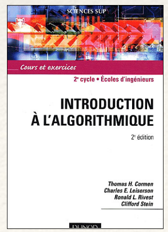
- Tous les sous-chemins d'un plus court chemin sont eux-mêmes des plus courts chemins



# Représentation machine

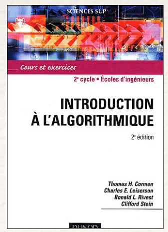
- Matrices d'adjacences.
- Pour des raisons de commodités: on supposons que les sommets sont numérotés 1, 2, . . . , | S | ,
- On utilise une matrice W de type n × n qui représente les poids d'arc d'un graphe orienté G = (S, A) à n sommets.

$$w_{ij} = \begin{cases} 0 & \text{si } i = j \ , \\ \text{le poids de l'arc } (i,j) & \text{si } i \neq j \text{ et } (i,j) \in A \ , \\ \infty & \text{si } i \neq j \text{ et } (i,j) \notin A \ . \end{cases}$$



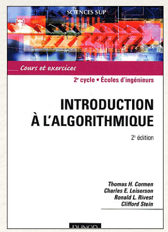
# Remarques

- Les arcs de poids négatif sont autorisés mais on suppose qu'il n'y a aucun circuit de longueur strictement négative.
- La sortie par les algorithmes est une matrice n×n
  - Notation de la matrice: $D = (d_{i,j})$,
  - $d_{i,j}$ est la longueur d'un plus court chemin reliant le sommet i au sommet j.
  - Autrement dit, si l'on appelle $\delta(i, j)$ la longueur de plus court chemin du sommet i au sommet j alors $d_{i,j} = \delta(i, j)$ à la fin de l'exécution.



# Matrice de Liaison

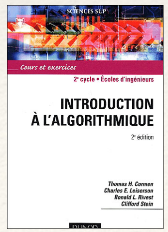
- Pour rechercher les plus courts chemins
  - entre tous les couples de sommets
  - à partir d'une matrice d'adjacences donnée,
- Il faut calculer
  - Les longueurs des plus courts chemins
  - Une matrice de liaison $\Pi = (\pi_{ij})$
    - $\pi_{ij}$ vaut NIL **si** i = j ou s'il n'existe aucun chemin de i vers j ;
    - $\pi_{ij}$ est le prédécesseur de j sur un plus court chemin issu de i, **sinon**



# Algorithme

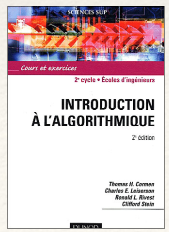
IMPRIMER-PLUS-COURT-CHEMIN-TOUS-COUPLES($\Pi$, $i$, $j$)

**si** $i = j$
  **alors** imprimer $i$
  **sinon** **si** $\pi_{ij} = \text{NIL}$
    **alors** imprimer « il n'existe aucun chemin de » $i$ « à » $j$
    **sinon** IMPRIMER-PLUS-COURT-CHEMIN-TOUS-COUPLES($\Pi$, $i$, $\pi_{ij}$)
      imprimer $j$



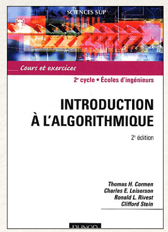
# Multiplication de matrices

- Programmation dynamique
- Graphe orienté G=(S,A)
- Programme similaire à la multiplication de matrices



# Programmation dynamique

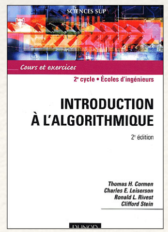
- Elaboration d'un programme dynamique:
  1. Caractérisation de la structure d'une solution optimale.
  2. Définition récursive de la valeur d'une solution optimale.
  3. Calcul de la valeur d'une solution optimale de façon ascendante.



# Structure d'un plus court chemin

- On considère un plus court chemin p du sommet i au sommet j, et on suppose que p contient au plus m arcs.
- En supposant qu'il n'existe aucun circuit de longueur strictement négative, m est fini.
- Si i = j, alors p a une longueur égale à 0 et n'a aucun arc.
- Si les sommets i et j sont distincts, on décompose le chemin p en:
  - $i \overset{p'}{\rightsquigarrow} k \rightarrow j$, où le chemin p′ contient maintenant au plus m − 1 arcs.
- D'après le lemme 1 p′ est un plus court chemin de i vers k, et donc

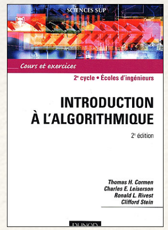
$$\delta(i, j) = \delta(i, k) + w_{kj}.$$



# Solution récursive: cas de base

- Soit $d_{ij}^{(m)}$ la longueur minimale d'un chemin d'au plus m arcs du sommet i au sommet j.
- Pour m = 0, il existe un plus court chemin sans arc de i vers j si et seulement si i = j.
- Formalisation:

$$d_{ij}^{(0)} = \begin{cases} 0 & \text{si } i = j\,, \\ \infty & \text{si } i \neq j\,. \end{cases}$$

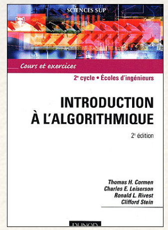


# Solution récursive: cas général

- Pour m ≥ 1, on calcule $d_{ij}^{(m)}$ comme étant le minimum de $d_{ij}^{(m-1)}$
  - $d_{ij}^{(m-1)}$ est le poids du plus court chemin de i à j constitué
    - D'au plus m – 1 arcs
    - De la longueur minimale d'un quelconque chemin d'au plus m arcs de i à j obtenu après examen de tous les prédécesseurs potentiels k de j.
- On définit donc formalise donc récursivement:

$$
\begin{aligned}
d_{ij}^{(m)} &= \min\left(d_{ij}^{(m-1)}, \min_{1\leqslant k\leqslant n}\{d_{ik}^{(m-1)} + w_{kj}\}\right) \\
&= \min_{1\leqslant k\leqslant n}\{d_{ik}^{(m-1)} + w_{kj}\} \ .
\end{aligned}
$$

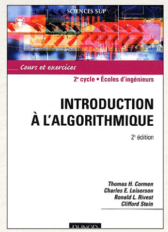
(équation 2)



# Solution récursive: Remarque

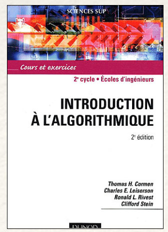
- Quels sont les longueurs réelles de plus court chemin $\delta$(i,j)?
- Si le graphe ne contient aucun circuit de longueur négative:
  - Alors pour toute paire de sommets i et j tels que $\delta(i, j) < \infty$, il existe un plus court chemin de i à j qui est élémentaire et qui contient donc au plus n – 1 arcs.
- Un chemin du sommet i au sommet j de plus de n – 1 arcs ne peut pas avoir une longueur inférieure à celle d'un plus court chemin de i à j.
- Les longueurs de plus court chemin réelles sont donc données par

$$\delta(i,j) = d_{ij}^{(n-1)} = d_{ij}^{(n)} = d_{ij}^{(n+1)} = \cdots \quad \text{(équation 3)}$$



# Calcul ascendant: Principe

- **Entrée**: matrice W = ($w_{i,j}$),
- Calcule d'une série de matrices D(1),D(2),...,D(n−1)
  - pour m = 1,2,...,n − 1, on a D(m) = $\left(d_{ij}^{(m)}\right)$
- La dernière matrice, D(n−1), contient les longueurs de plus court chemin réelles.
- Notons que, comme d(1) = $w_{i,j}$ pour tous les sommets i,j ∈ S,
  - on a D(1) = $W_{i,j}$
- Le principe de l'algorithme est le suivant:
  - étant données les matrices D(m−1) et W,
  - retourne la matrice D(m).
  - Autrement dit, elle étend d'un arc supplémentaire les plus courts chemins calculés jusqu'à présent.

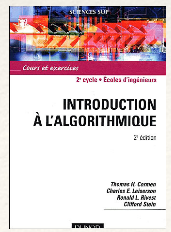


# Algorithme

EXTENSION-PLUS-COURTS-CHEMINS$(D, W)$

$n \leftarrow lignes[D]$
soit $D' = (d'_{ij})$ une matrice $n \times n$
**pour** $i \leftarrow 1$ **à** $n$
  **faire pour** $j \leftarrow 1$ **à** $n$
    **faire** $d'_{ij} \leftarrow \infty$
      **pour** $k \leftarrow 1$ **à** $n$
        **faire** $d'_{ij} \leftarrow \min(d'_{ij}, d_{ik} + w_{kj})$
**retourner** $D'$

Temps d'exécution $\Theta(n^3)$

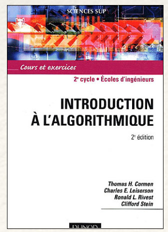


# Algorithme

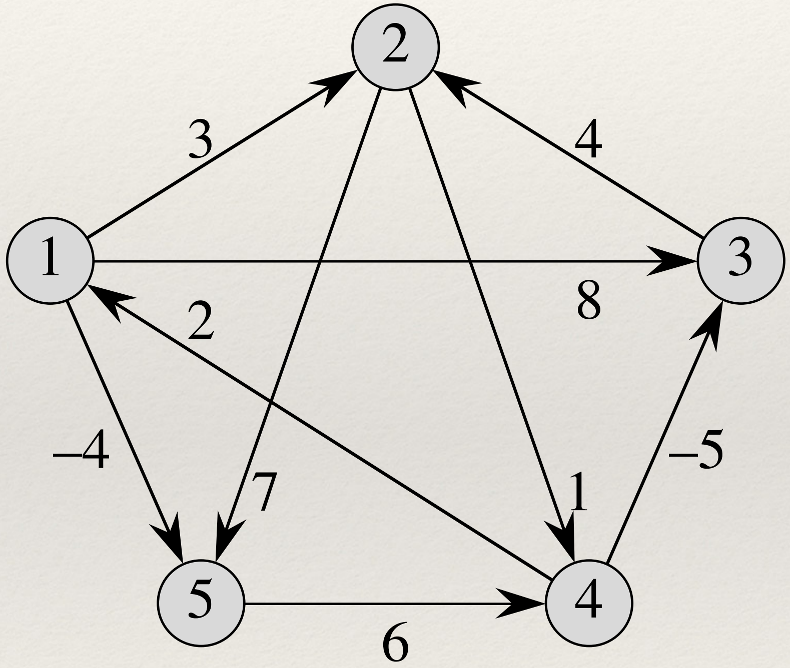
PLUS-COURT-CHEMIN-TOUS-COUPLES-RALENTI($W$)

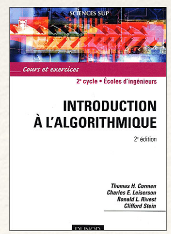
$n \leftarrow lignes[W]$
$D^{(1)} \leftarrow W$
**pour** $m \leftarrow 2$ **à** $n - 1$
  **faire** $D^{(m)} \leftarrow$ EXTENSION-PLUS-COURTS-CHEMINS($D^{(m-1)}, W$)
**retourner** $D^{(n-1)}$

Temps d'exécution $\Theta(n^4)$

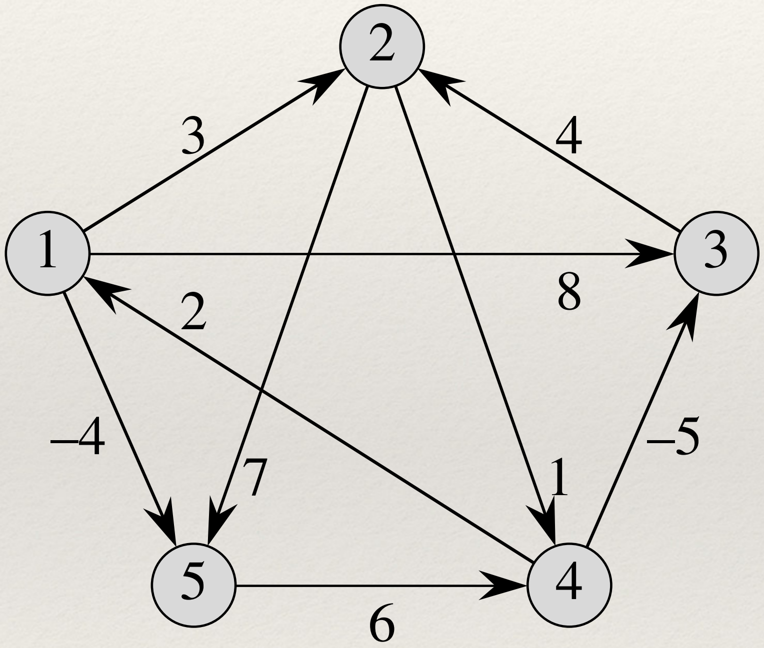


# Exemple

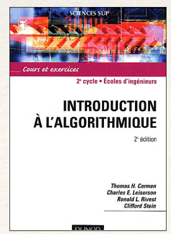


# Exemple




$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

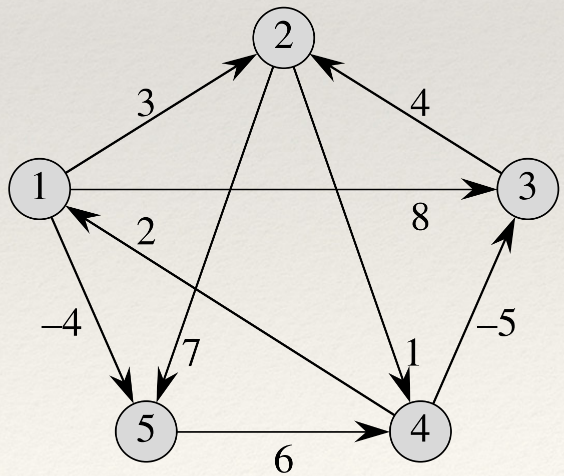


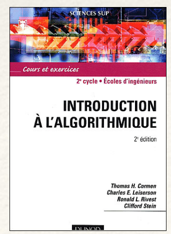
# Exemple

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\min\{d^1_{1,4}; d^1_{1,k} + w_{k,4}\} \rightarrow d^1_{1,4} = \infty$$
$$k = 1 : d^1_{1,1} + w_{1,4} = 0 + \infty = \infty$$
$$k = 2 : d^1_{1,2} + w_{2,4} = 3 + 1 = 4$$
$$k = 3 : d^1_{1,3} + w_{3,4} = 8 + \infty = \infty$$
$$k = 4 : d^1_{1,4} + w_{4,4} = \infty + 0 = \infty$$
$$k = 5 : d^1_{1,5} + w_{5,4} = -4 + 6 = 2$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$



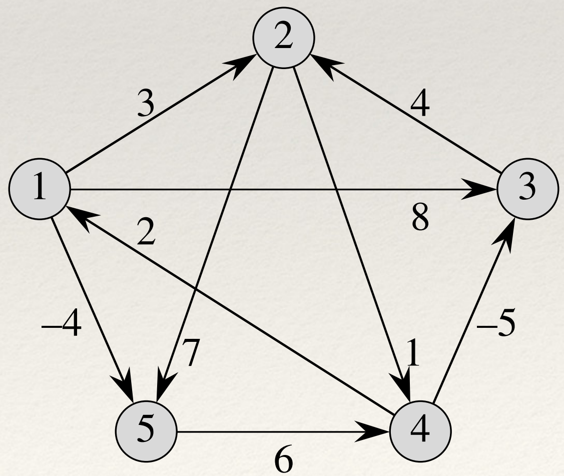



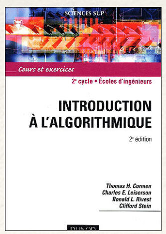
# Exemple

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\min\{d^1_{2,3}; d^1_{2,k} + w_{k,3}\} \rightarrow d^1_{2,3} = \infty$$
$$k = 1 : d^1_{2,1} + w_{1,3} = \infty + 8 = \infty$$
$$k = 2 : d^1_{2,2} + w_{2,3} = 0 + \infty = \infty$$
$$k = 3 : d^1_{2,3} + w_{3,3} = \infty + 0 = \infty$$
$$k = 4 : d^1_{2,4} + w_{4,3} = 1 + (-5) = -4$$
$$k = 5 : d^1_{2,5} + w_{5,3} = 7 + \infty = \infty$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$



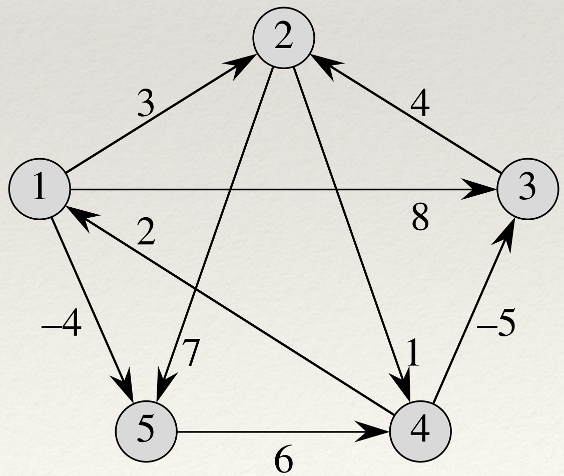
# Exemple

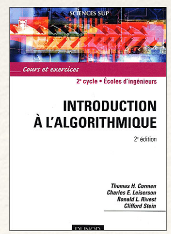
$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$\min\{d^2_{1,3}; d^2_{2,k} + w_{k,3}\} \rightarrow d^2_{1,3} = 8$$
$$k = 1 : d^2_{1,1} + w_{1,3} = 0 + 8 = 8$$
$$k = 2 : d^2_{1,2} + w_{2,3} = 3 + +\infty = \infty$$
$$k = 3 : d^2_{1,3} + w_{3,3} = \infty + 0 = \infty$$
$$k = 4 : d^2_{1,4} + w_{4,3} = 2 + (-5) = -3$$
$$k = 5 : d^2_{1,5} + w_{5,3} = -4 + \infty = \infty$$



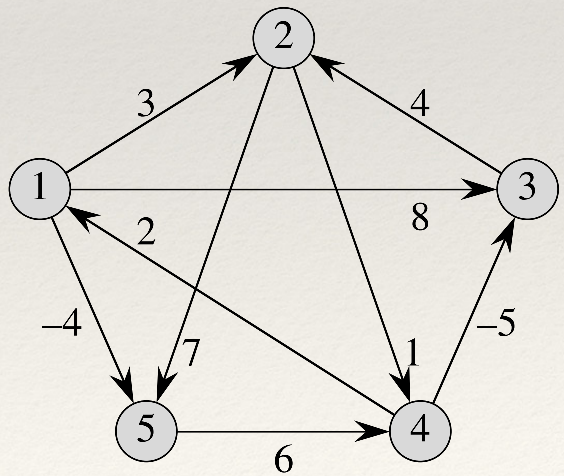



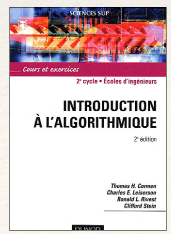
# Exemple

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

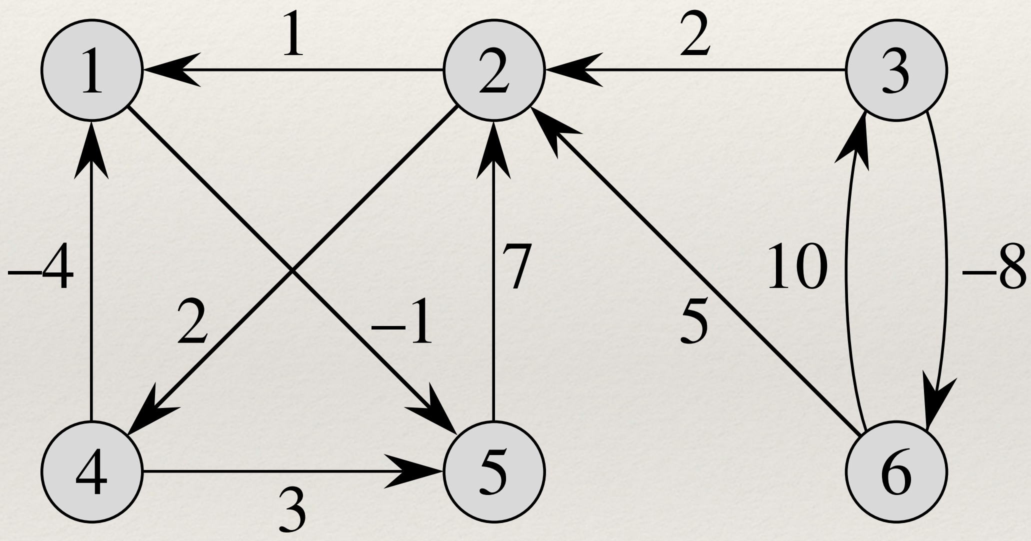
$$D^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

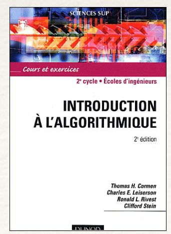
$$D^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$



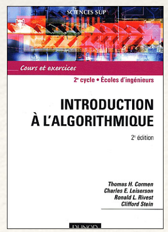
# Autre exemple



# Remarques

- Nous avons calculer toutes les matrices:
  - Avec un arc
  - Puis avec deux arcs
  - ....
- Ce qui nous intéresse ce n'est pas de calculer toutes les matrices
- Mais seulement $D^{(n-1)}$



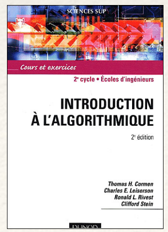
# Amélioration

- Rappelons nous qu'en l'absence de circuits de longueur strictement négative, l'équation (3) implique
- $D^{(m)} = D^{(n-1)}$ pour tout entier $m \geq n - 1$.

- La multiplication matricielle classique est associative
- La multiplication matricielle définie par la procédure EXTENSION-PLUS- COURTS-CHEMINS est associative

$$
\begin{array}{rcccl}
D^{(1)} & = & W & , & \\
D^{(2)} & = & W^2 & = & W \cdot W \,, \\
D^{(4)} & = & W^4 & = & W^2 \cdot W^2 \\
D^{(8)} & = & W^8 & = & W^4 \cdot W^4 \,, \\
 & & \vdots & & \\
D^{(2^{\lceil \lg(n-1) \rceil})} & = & W^{2^{\lceil \lg(n-1) \rceil}} & = & W^{2^{\lceil \lg(n-1) \rceil - 1}} \cdot W^{2^{\lceil \lg(n-1) \rceil - 1}}
\end{array}
$$

- Comme $2^{\lceil \log(n-1) \rceil} \geq n-1$, le produit final est $D^{(2^{\lceil \log(n-1) \rceil})}$ est égal à $D^{(n-1)}$

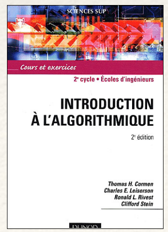


# Algorithme

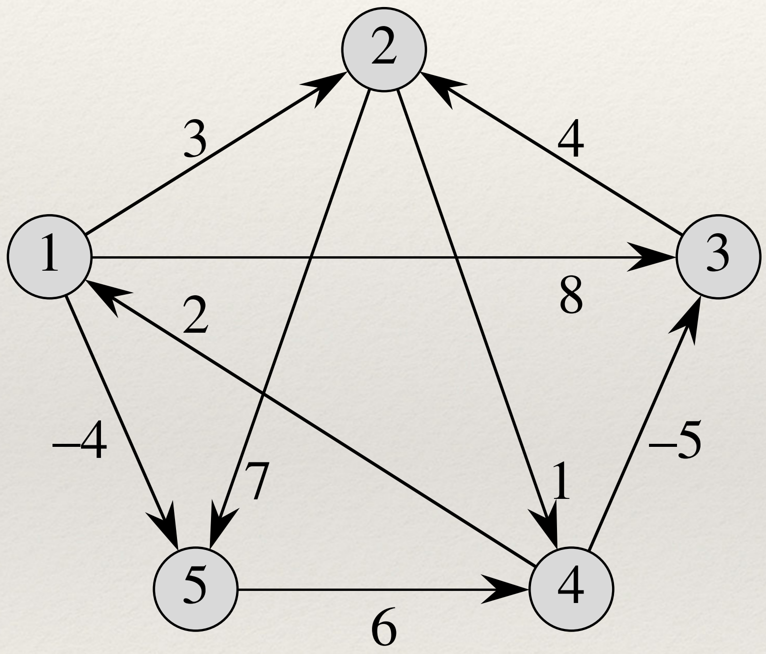
PLUS-COURT-CHEMIN-TOUS-COUPLES-ACCÉLÉRÉ($W$)

$n \leftarrow lignes[W]$
$D^{(1)} \leftarrow W$
$m \leftarrow 1$
**tant que** $m < n - 1$
    **faire** $D^{(2m)} \leftarrow$ EXTENSION-PLUS-COURTS-CHEMINS$(D^{(m)}, D^{(m)})$
        $m \leftarrow 2m$
**retourner** $D^{(m)}$

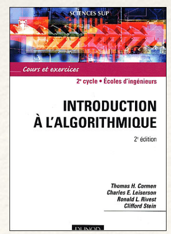
Temps d'exécution $\Theta(n^3 \log n)$

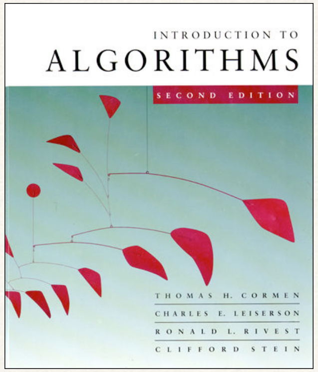


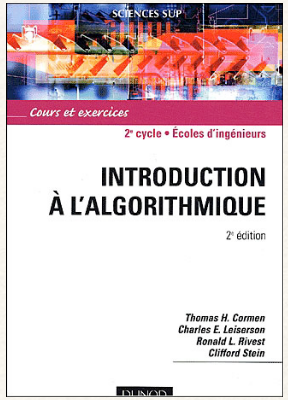
# Exemple




$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$






# L'algorithme de Floyd-Warshall

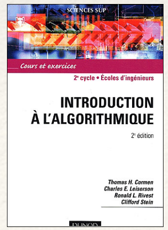
**Théorie des graphes**

Cours de Lélia Blin

3eme année de Licence

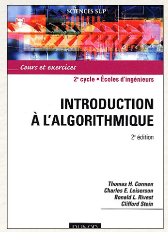
# Algorithme de Floyd-Warshall

- Programmation dynamique
- Graphe orienté G=(S,A)
- Complexité: $\Theta(|V|^3)$
- Il peut y avoir des arcs de poids négatif
  - Mais aucun circuit absorbant



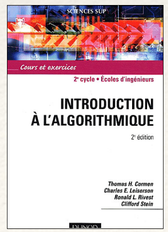
# Programmation dynamique

- Elaboration d'un programme dynamique:
  1. Caractérisation de la structure d'une solution optimale.
  2. Définition récursive de la valeur d'une solution optimale.
  3. Calcul de la valeur d'une solution optimale de façon ascendante.



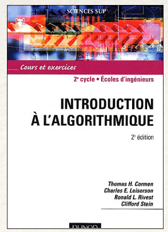
# Structure d'un plus court chemin

- Caractérisation: L'algorithme considère les sommets «intermédiaires» d'un plus court chemin ;
- Un sommet intermédiaire d'un chemin simple
  - $p = \langle v_1, v_2, \ldots, v_l \rangle$ est un sommet de p autre que $v_1$ ou $v_l$,
- Autrement dit un sommet appartenant à l'ensemble $\{v_2, v_3, \ldots, v_{l-1}\}$.



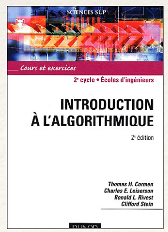
# Caractérisation

- L'algorithme de Floyd-Warshall s'appuie sur l'observation suivante :
  - Si l'on appelle S = {1, 2, . . . , n} les sommets de G,
  - On considère un sous-ensemble {1, 2, . . . , k} de sommets pour un certain k.
  - Pour un couple quelconque de sommets i,j ∈ S, on considère tous les chemins de i à j dont les sommets intermédiaires appartiennent tous à {1,2,...,k}, et on note p un chemin de longueur minimale parmi eux. (Le chemin p est élémentaire.)
  - L'algorithme de Floyd-Warshall exploite une relation entre le chemin p et les plus courts chemins de i vers j dont tous les sommets intermédiaires sont dans {1, 2, . . . , k – 1}.
  - La relation dépend de ce que k est ou n'est pas un sommet intermédiaire de p.



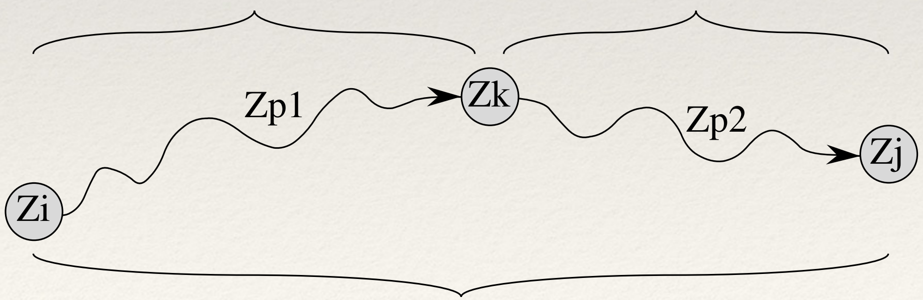
# Sommet intermédiaire

- Si k n'est pas un sommet intermédiaire du chemin p,
- Alors tous les sommets intermédiaires de p se trouvent dans l'ensemble {1, 2, . . . , k – 1}.
- Donc, un plus court chemin du sommet i au sommet j ayant tous les sommets intermédiaires dans l'ensemble {1,2,...,k – 1} est aussi un plus court chemin de i vers j ayant tous les sommets intermédiaires dans l'ensemble {1, 2, . . . , k}.

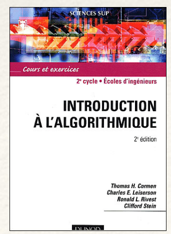


# Sommet intermédiaire

- Si k est un sommet intermédiaire du chemin p,
  - alors on divise p en $i \overset{p_1}{\rightsquigarrow} k \overset{p_2}{\rightsquigarrow} j$ comme illustré à la figure ci-dessous.
- D'après le lemme 1, $p_1$ est un plus court chemin de i vers k, tous les sommets intermédiaires appartenant à l'ensemble {1, 2, . . . , k}.
- Comme le sommet k n'est pas un sommet intermédiaire de $p_1$, on voit que $p_1$ est un plus court chemin de i vers k, tous les sommets intermédiaires étant pris dans l'ensemble {1, 2, . . . , k − 1}.
- De même, $p_2$ est un plus court chemin du sommet k vers le sommet j, tous les sommets intermédiaires étant pris dans {1, 2, . . . , k − 1}.



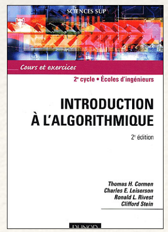

*p* : tous les sommets intermédiaires de {1,2,...,k−1}



# Solution récursive

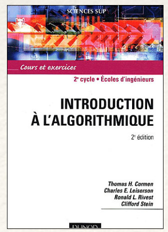
- Soit $d_{ij}^{(k)}$ le poids d'un plus court chemin du sommet i au sommet j dont tous les sommets intermédiaires sont dans l'ensemble {1,2,...,k}.
- Pour k = 0, un chemin de i à j sans sommet intermédiaire de rang supérieur à 0 ne possède en réalité aucun sommet intermédiaire.
  - Il est constitué d'au plus un arc, et on a donc $d_{ij}^{(0)} = w_{ij}$
- Il en résulte la définition récursive que voici

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{si } k = 0\,, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{si } k \geqslant 1\,. \end{cases}$$



# Remarque

La matrice $D^{(n)} = \left(d_{ij}^{(n)}\right)$ donne le résultat final ($d_{ij}^{(n)} = \delta(i, j)$ pour tout $i, j \in S$) ; en effet, quel que soit le chemin, tous les sommets intermédiaires appartiennent à l'ensemble $\{1, 2, \ldots, n\}$.

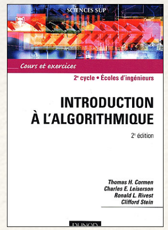


# Algorithme

FLOYD-WARSHALL($W$)

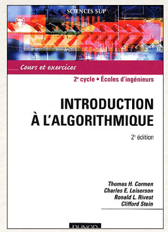
$n \leftarrow lignes[W]$
$D^{(0)} \leftarrow W$
**pour** $k \leftarrow 1$ **à** $n$
  **faire pour** $i \leftarrow 1$ **à** $n$
    **faire pour** $j \leftarrow 1$ **à** $n$
      **faire** $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
**retourner** $D^{(n)}$

Temps d'exécution $\Theta(n^3)$



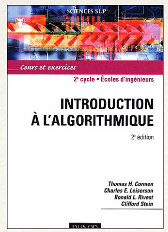
# Construction d'un plus court chemin

- Il existe de nombreuses méthodes différentes pour construire les plus courts chemins avec l'algorithme de Floyd-Warshall.
- Exemple calculer la matrice D des longueurs de pcc, puis construire la matrice de liaison ∏ à partir de la matrice D.
- Connaissant la matrice de liaison ∏ on peut utiliser la procédure
  - IMPRIMER-PLUS-COURT-CHEMIN-TOUS-COUPLES pour imprimer les sommets d'un plus court chemin donné.



# Construction d'un plus court chemin

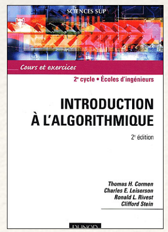
- Il est possible de calculer $\prod$, en même temps que l'algorithme de Floyd-Warshall calcule les matrices $D^{(k)}$.
- Plus précisément, on calcule:
  - Une séquence de matrices $\prod^{(0)}, \prod^{(1)},\ldots, \prod^{(n)}$,
  - où $\prod = \prod^{(n)}$
  - et $\pi_{ij}^{(k)}$ est le prédécesseur du sommet j sur un plus court chemin partant du sommet i et dont tous les sommets intermédiaires sont dans {1, 2, . . . , k}.



# Formule récursive

- Pour k = 0, un plus court chemin de i vers j ne possède aucun sommet intermédiaire. Donc:

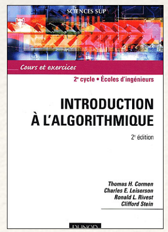
$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{si } i = j \text{ ou } w_{ij} = \infty \ , \\ i & \text{si } i \neq j \text{ et } w_{ij} < \infty \ . \end{cases}$$



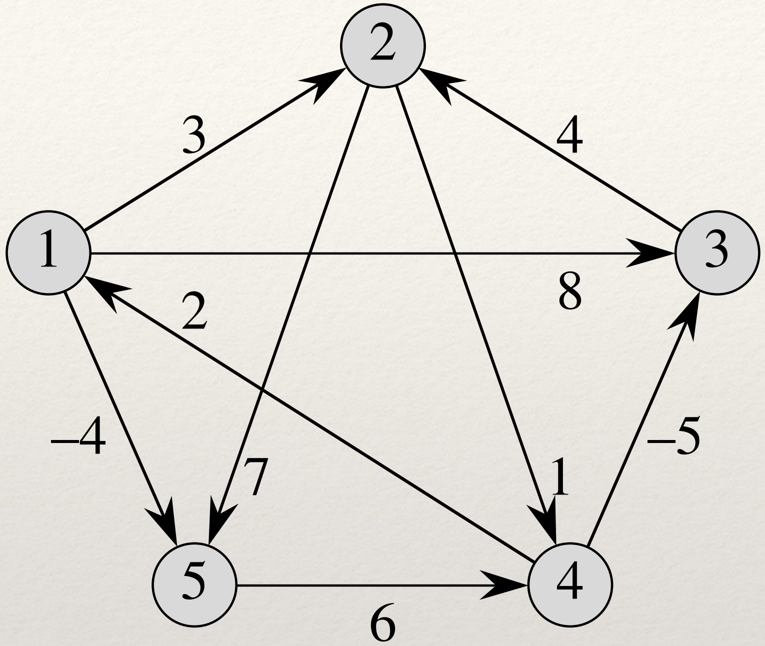
# Formule récursive

- Pour k≥1:
  - **Si** l'on prend le chemin $i \rightsquigarrow k \rightsquigarrow j$ où k≠j, Alors le prédécesseur de j que nous choisissons est le même que celui que nous avions choisi sur un plus court chemin issu de k dont tous les sommets intermédiaires se trouvent dans {1, 2, . . . , k − 1}.
  - **Sinon**, on prend le même prédécesseur de j que celui que nous avions choisi sur un plus court chemin partant de i dont tous les sommets intermédiaires sont dans l'ensemble {1,2,...,k − 1}.
- Formellement, pour k≥1:

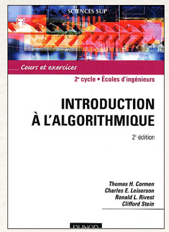
$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \leqslant d_{ik}^{(k-1)} + d_{kj}^{(k-1)} , \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} . \end{cases}$$



# Exemple




$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

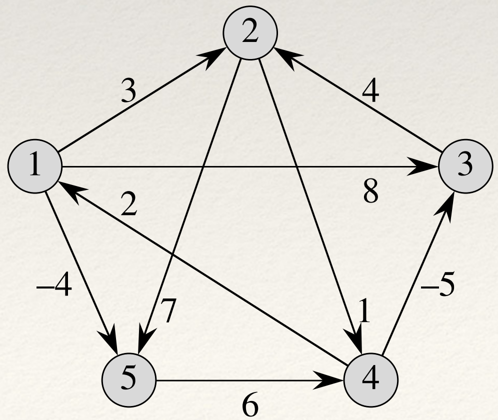


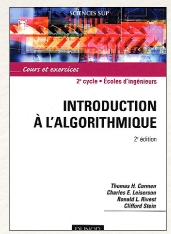
# Exemple

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(0)} = \begin{pmatrix} \mathrm{NIL} & 1 & 1 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 2 & 2 \\ \mathrm{NIL} & 3 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} \\ 4 & \mathrm{NIL} & 4 & \mathrm{NIL} & \mathrm{NIL} \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 5 & \mathrm{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(1)} = \begin{pmatrix} \mathrm{NIL} & 1 & 1 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 2 & 2 \\ \mathrm{NIL} & 3 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} \\ 4 & 1 & 4 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 5 & \mathrm{NIL} \end{pmatrix}$$

$$\min\{d^0_{4,2}; d^0_{4,1} + d^0_{1,2}\}$$
$$d^0_{4,2} = \infty$$
$$d^0_{4,1} + d^0_{1,2} = 2 + 3 = 5$$



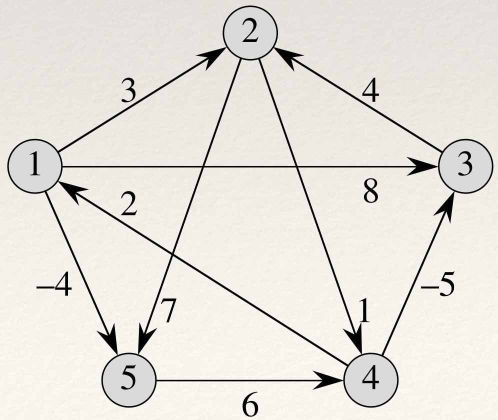



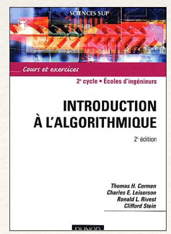
# Exemple

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$d^2_{1,4} = \min\{d^1_{1,4}; d^1_{1,2} + d^1_{2,4}\}$$
$$d^1_{1,4} = \infty$$
$$d^1_{1,2} + d^1_{2,4} = 3 + 1 = 4$$

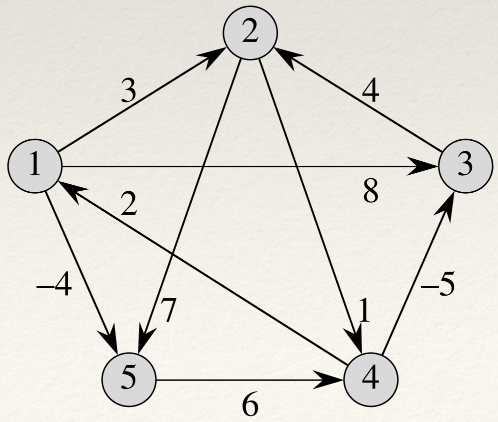


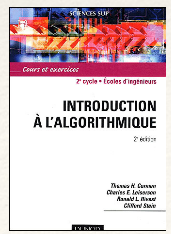



# Exemple

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

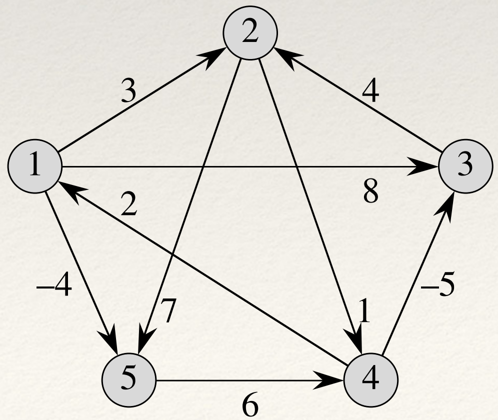


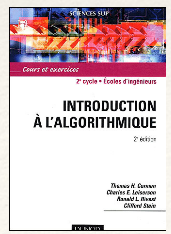



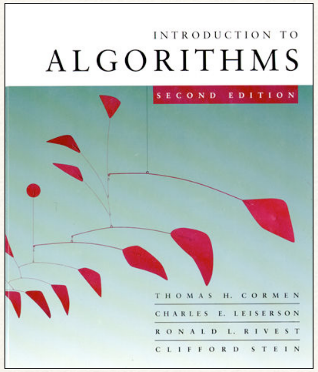
# Exemple

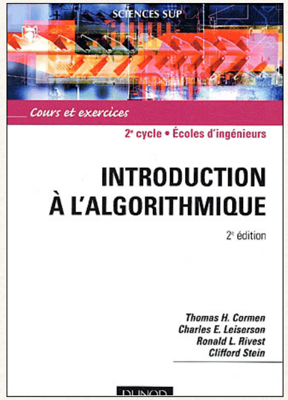
$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$






# *L'algorithme de Johnson*

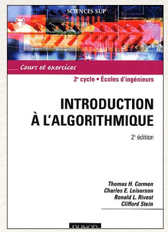
**Théorie des graphes**

Cours de Lélia Blin

3eme année de Licence

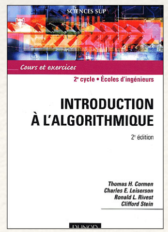
# Algorithme de Johnson

- Il trouve les plus courts chemins entre tous les couples de sommets
  - Complexité $O(S^2 \log S + SA)$
- Il est donc asymptotiquement plus performant que les élévations au carré répétées de matrices ou que l'algorithme de Floyd-Warshall pour les graphes peu denses.



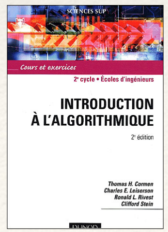
# Algorithme de Johnson

- L'algorithme retourne
  - Soit une matrice de longueurs de plus court chemin pour tous les couples de sommets
  - Soit indique que le graphe contient un circuit de longueur strictement négative.
- L'algorithme de Johnson utilise comme sous-programmes à la fois
  - l'algorithme de Dijkstra
  - l'algorithme de Bellman-Ford.



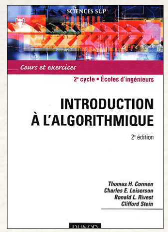
# Re-pondération

- Soit G=(S,A) un graphe
- Tous les poids d'arc w sont positifs ou nuls,
- On peut trouver les pcc entre tous les couples de sommets
  - En exécutant l'algorithme de Dijkstra une fois à partir de chaque sommet ; avec une file de priorité min basée sur un tas de Fibonacci,
  - Le temps d'exécution de cet algorithme toutes-paires est en $O(S^2 \log S + SA)$.



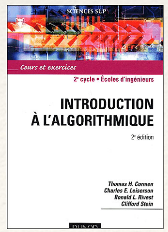
# Re-pondération

- Si G contient des arcs de poids négatif mais pas de circuit de longueur strictement négative,
- On se contente de calculer un nouvel ensemble de poids d'arc positifs qui permettra d'utiliser la même méthode.



# Re-pondération

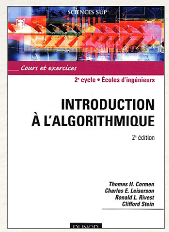
- Le nouvel ensemble de poids d'arc $\hat{w}$ doit vérifier deux propriétés importantes.
  1. Pour tout couple de sommets u, v $\in$ S, un chemin p est un plus court chemin de u à v utilisant la fonction de pondération w si et seulement si p est aussi un plus court chemin de u à v utilisant la fonction de pondération $\hat{w}$ .
  2. Pour tout couple (u, v), le nouveau poids $\hat{w}(u, v)$ est positif.
- Comme nous allons le voir bientôt, le pré traitement de G servant à déterminer la nouvelle fonction de pondération $\hat{w}$ peut être réalisé en O(SA)



# Lemme: Re-pondération

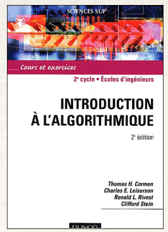
- Etant donné un graphe orienté pondéré G = (S, A) de fonction de pondération w : A → R, soit h : S → R une fonction associant à chaque sommet un nombre réel. Pour chaque arc (u, v) ∈ A, on définit $\widehat{w}(u, v) = w(u, v) + h(u) - h(v)$ . (9)

*Soit $p = \langle v_0, v_1, \ldots, v_k \rangle$ un chemin du sommet $v_0$ au sommet $v_k$. Alors, $p$ est un plus court chemin de $v_0$ à $v_k$ avec la fonction de pondération $w$ si et seulement si c'est un plus court chemin avec la fonction de pondération $\widehat{w}$. En d'autres termes, $w(p) = \delta(v_0, v_k)$ si et seulement si $\widehat{w}(p) = \widehat{\delta}(v_0, v_k)$. En outre, $G$ a un circuit de longueur strictement négative utilisant la fonction de pondération $w$ si et seulement si $G$ a un circuit de longueur strictement négative utilisant la fonction de pondération $\widehat{w}$.*

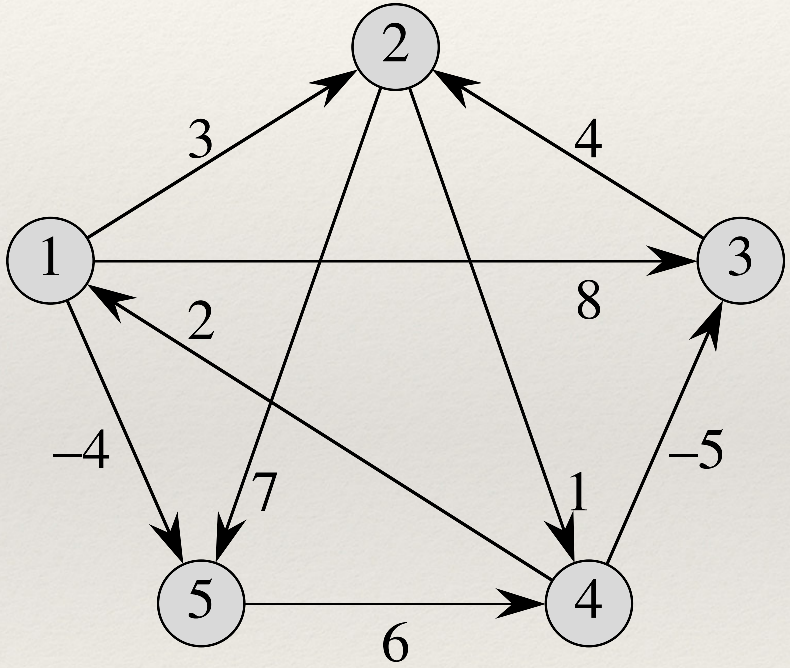


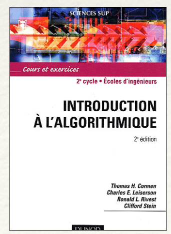
# Poids de re-pondération

- On souhaite que $\widehat{w}(u, v)$ soit positive pour tout arc $(u, v) \in A$.
- Étant donné $G = (S, A)$ de fonction de pondération $w : A \rightarrow R$,
- On construit un nouveau graphe $G' = (S', A')$, où $S' = S \cup \{s\}$ pour un nouveau sommet $s \notin S$ donné et $A' = A \cup \{(s, v) : v \in S\}$.
- La fonction de pondération w est étendue de sorte que $w(s, v) = 0$ pour tout $v \in S$.
- Notez que, comme aucun arc n'entre dans s, aucun plus court chemin de G′, hormis ceux d'origine s, ne contient s.
- Par ailleurs, G′ ne contient aucun circuit de longueur strictement négative si et seulement si G ne contient aucun circuit de longueur strictement négative.

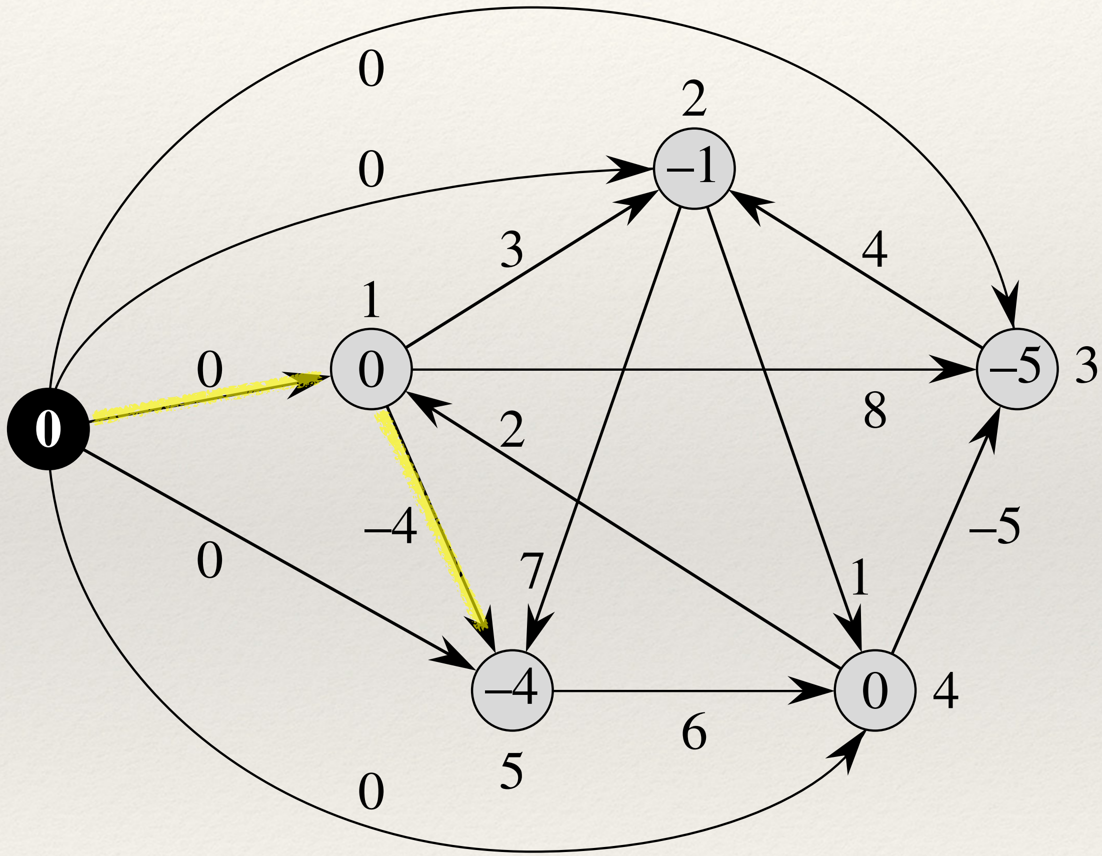


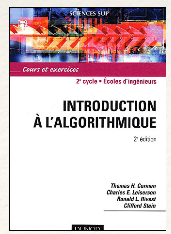
# Exemple



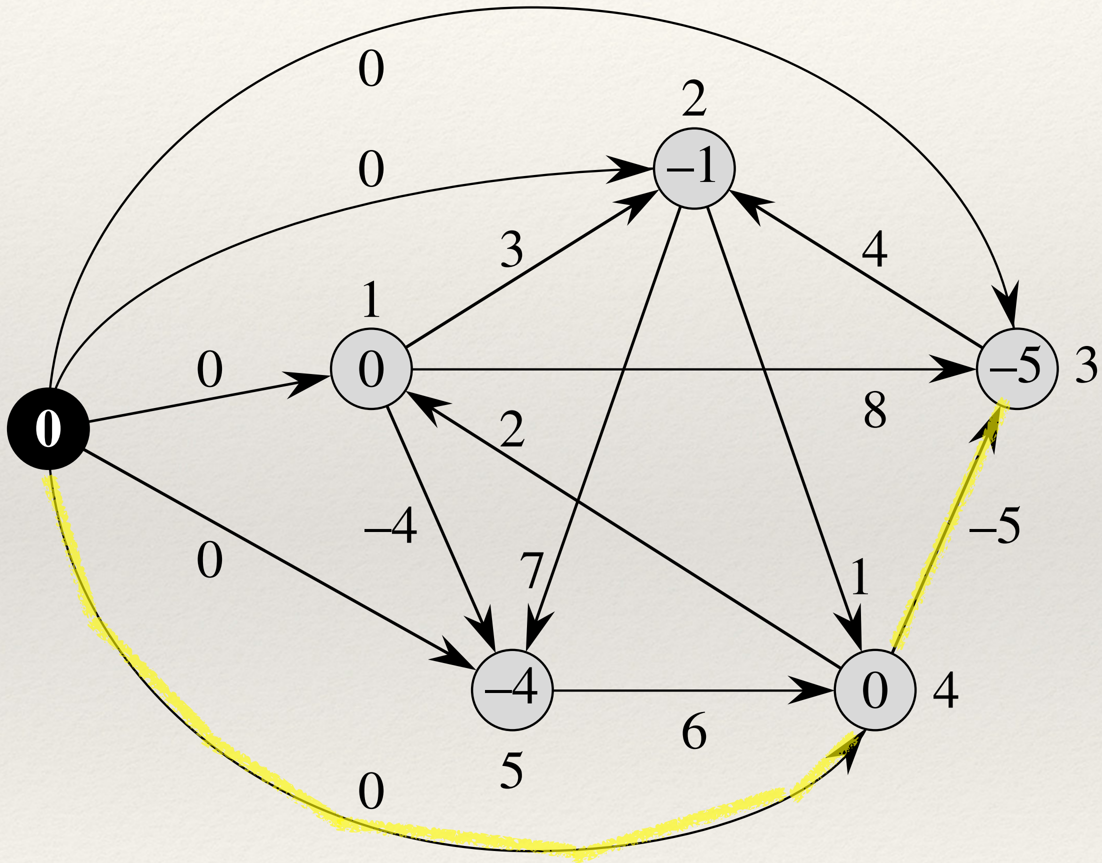



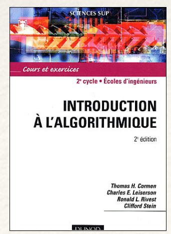
# Exemple: calcul de h(u)



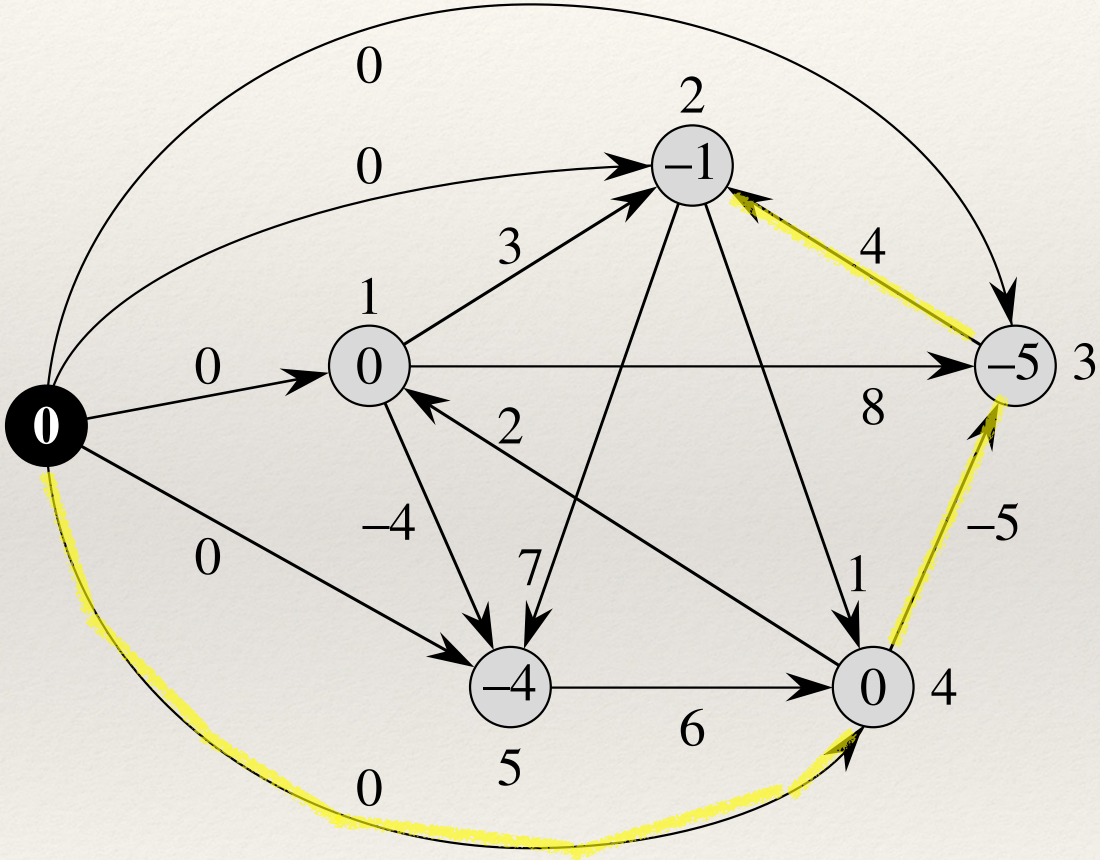



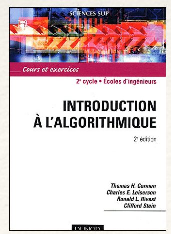
# Exemple: calcul de h(u)



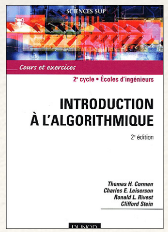
# Exemple: calcul de h(u)



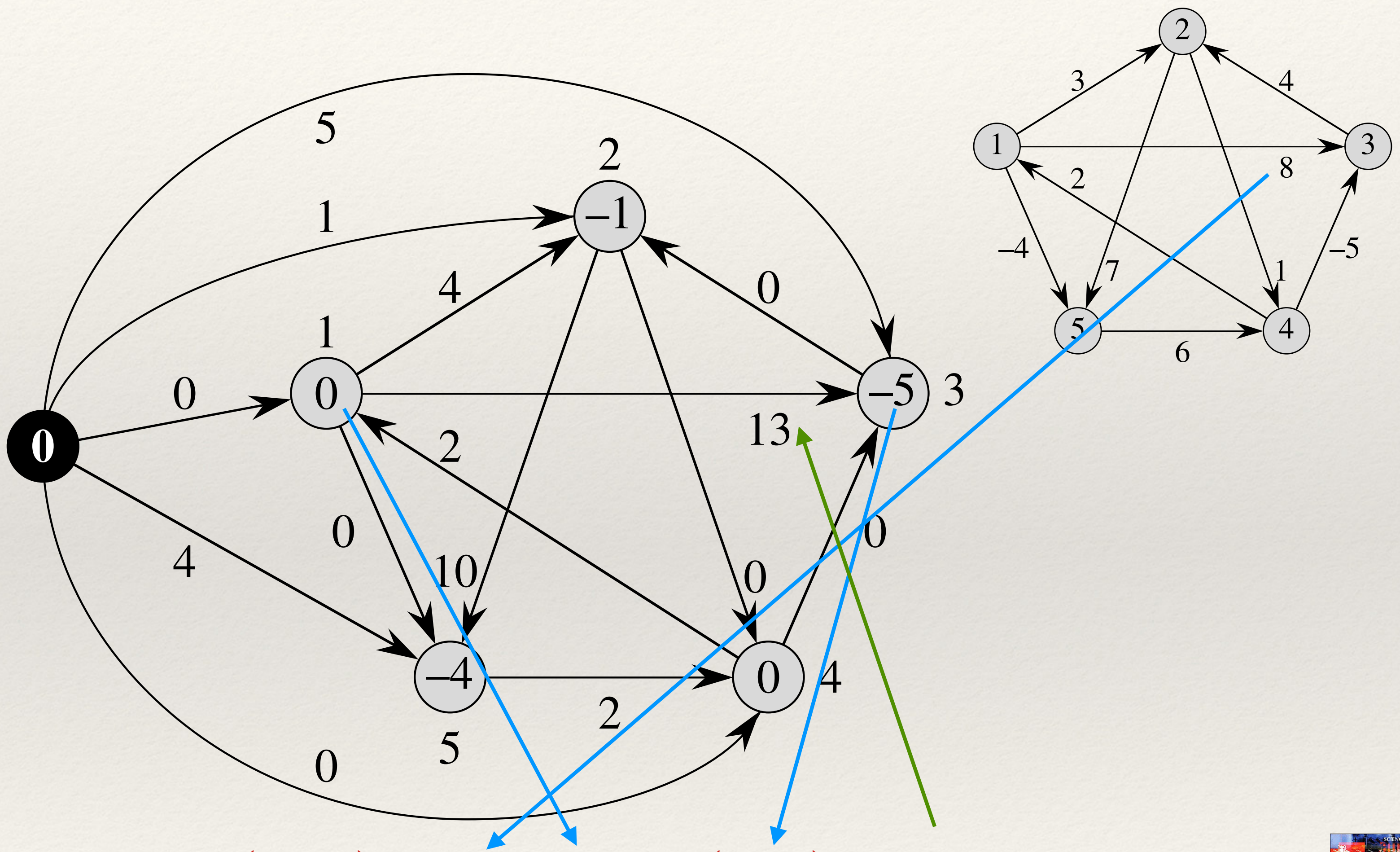



# Inégalité triangulaire

- Si G contient des arcs de poids négatif mais pas de circuit de longueur strictement négative,
- Supposons à présent que G et G′ ne contiennent aucun circuit de longueur strictement négative.
- On définit $h(v) = \delta(s, v)$ pour tout $v \in S'$.
- D'après l'inégalité triangulaire
  - on a $h(v) \leq h(u) + w(u,v)$ pour tout arc $(u, v) \in A'$.
  - Donc, si l'on définit les nouveaux poids $\widehat{w}$
  - d'après l'équation (9),
  - on a $\widehat{w}(u, v) = w(u, v) + h(u) - h(v) \geq 0$,
  - et la seconde propriété est vérifiée.

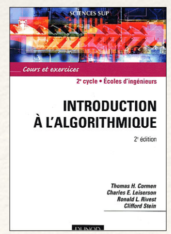


# Exemple: repondération

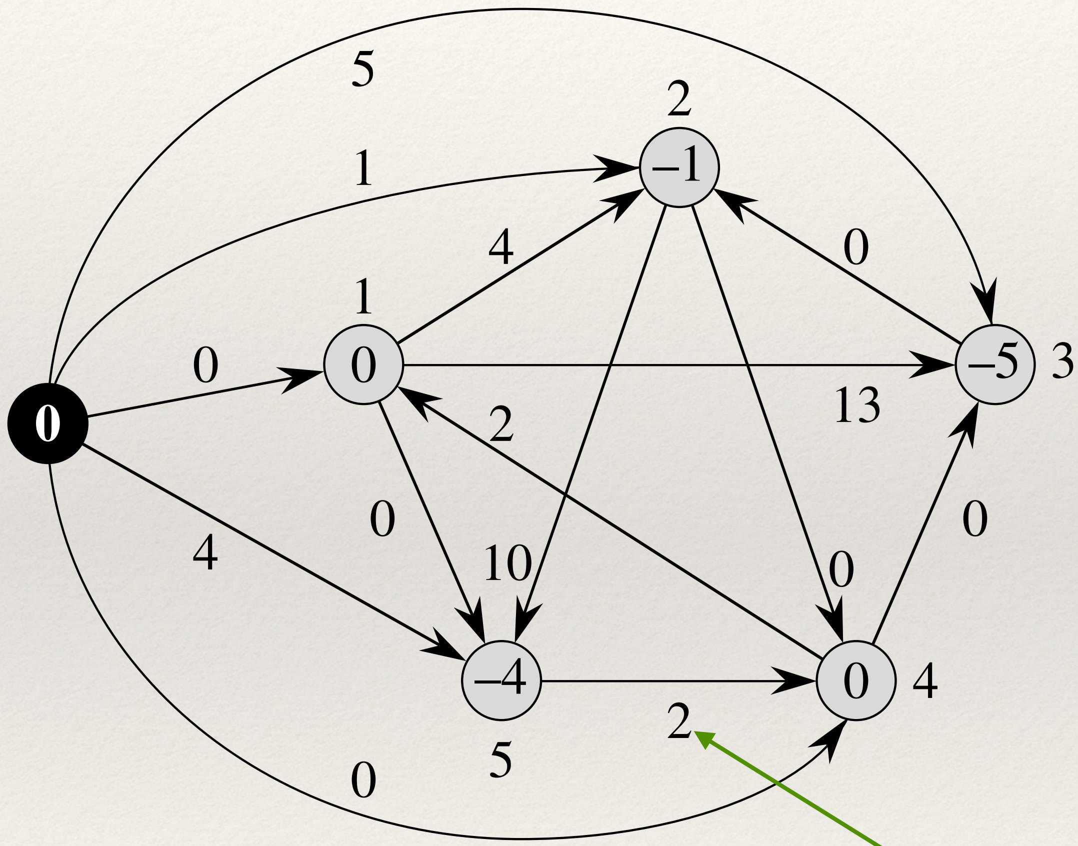


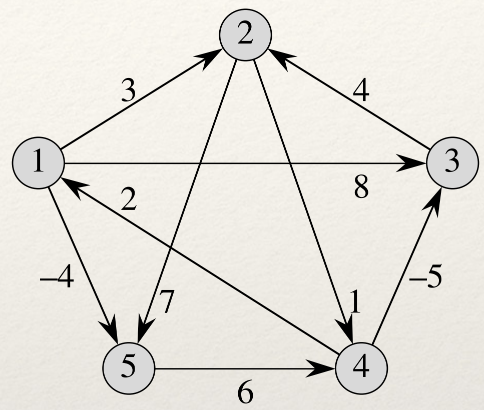

$$\hat{w}(0,3) = 8 + 0 - (-5) = 13$$

$$\hat{w}(u,v) = w(u,w) + h(u) - h(v)$$



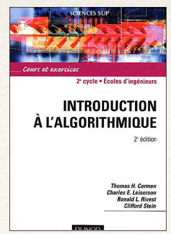
# Exemple: repondération



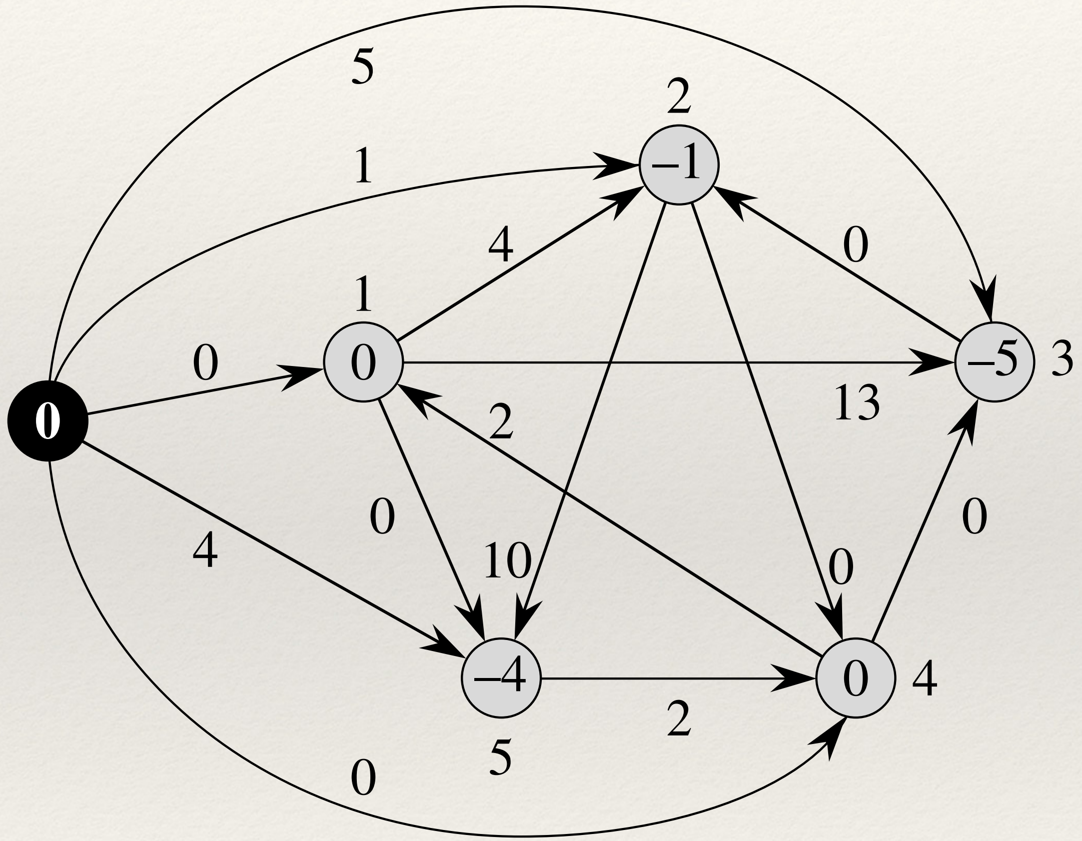



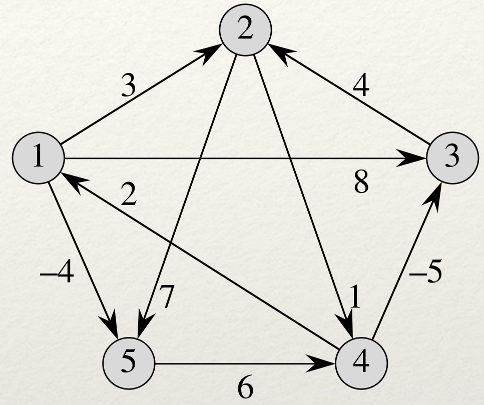

$$\color{red}{\hat{w}(5,4) = 6 + (-4) - 0 = 2}$$

$$\hat{w}(u,v) = w(u,w) + h(u) - h(v)$$



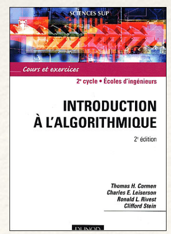
# Exemple: repondération



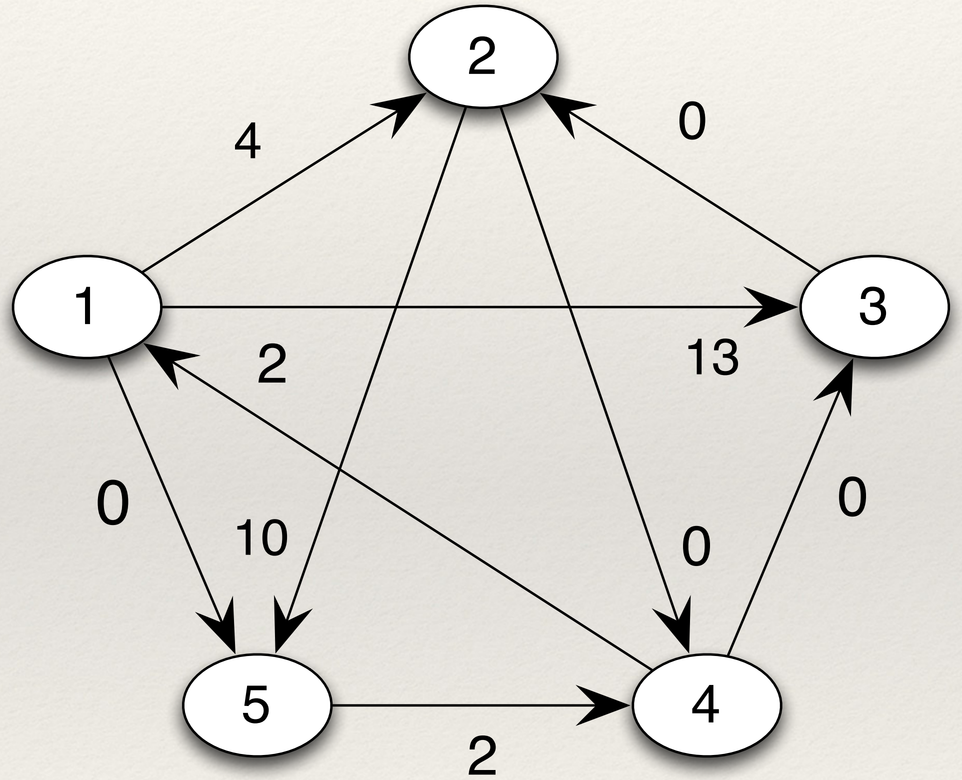



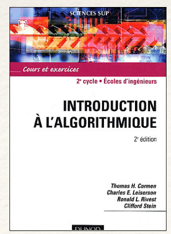

$$\hat{w}(2,5) = 7 + (-1) - (-4) = 10$$

$$\hat{w}(u,v) = w(u,w) + h(u) - h(v)$$



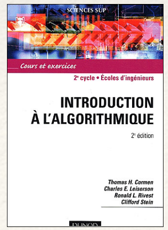
# Exemple: repondération



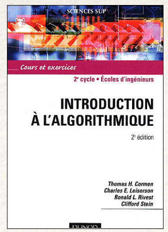
# Calcul de plus court chemin

- L'algorithme de Johnson calculant les plus courts chemins pour tout couple de sommets utilise les algorithmes de Bellman-Ford et de Dijkstra comme sous-programmes.
- Il suppose que les arcs sont représentés par des listes d'adjacences.
- L'algorithme retourne:
  - La matrice $|S| \times |S|$ : $D = d_{ij}$, où $d_{ij} = \delta(i, j)$,
  - Ou bien indique que le graphe contient un circuit de longueur négative.

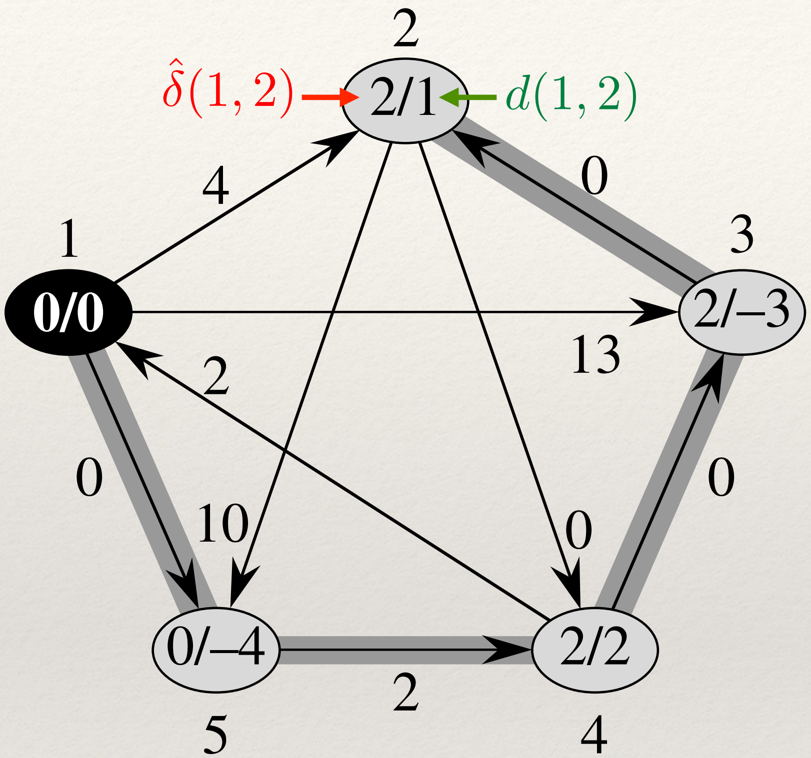


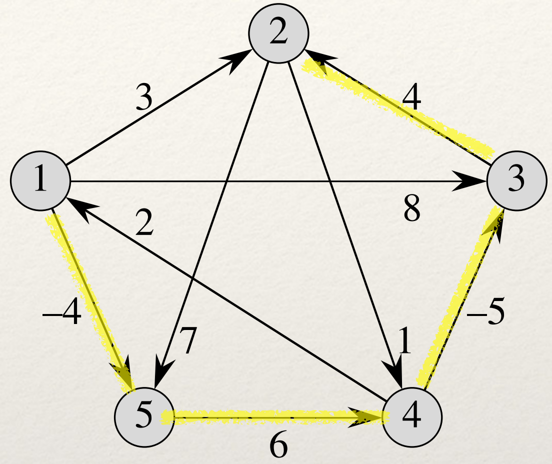
# Algorithme de Johnson

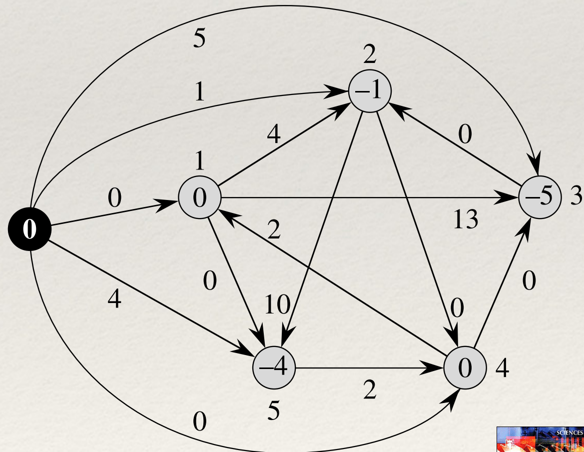
```
JOHNSON(G)
calculer G′, où S[G′] = S[G] ∪ {s},
          A[G′] = A[G] ∪ {(s, v) : v ∈ S[G]}, et
          w(s, v) = 0 pour tout v ∈ S[G]
si BELLMAN-FORD(G′, w, s) = FAUX
   alors imprimer « le graphe contient un circuit de longueur strictement négative »
   sinon pour chaque sommet v ∈ S[G′]
             faire affecter à h(v) la valeur de δ(s, v)
                            calculée par l'algorithme de Bellman-Ford
         pour chaque arc (u, v) ∈ A[G′]
             faire ŵ(u, v) ← w(u, v) + h(u) − h(v)
         pour chaque sommet u ∈ S[G]
             faire exécuter DIJKSTRA(G, ŵ, u) pour calculer δ̂(u, v)
                            pour tout v ∈ S[G]
                   pour chaque sommet v ∈ S[G]
                       faire d_uv ← δ̂(u, v) + h(v) − h(u)
         retourner D
```



# Exemple

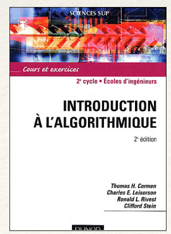


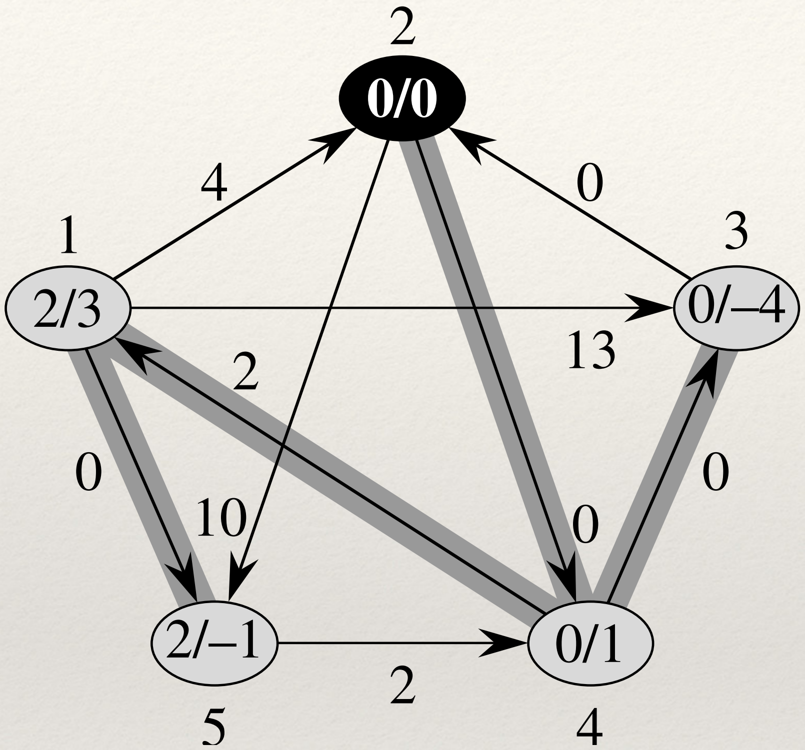


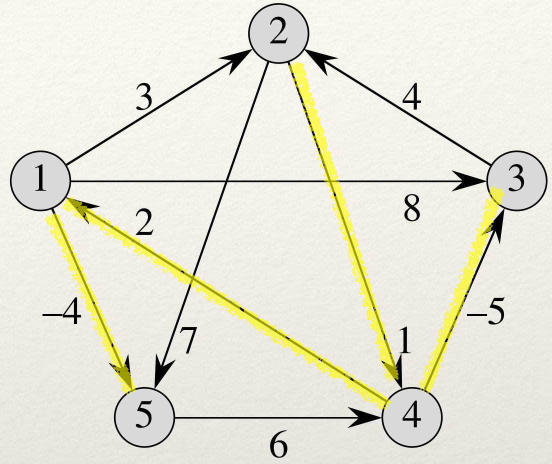
$$d_{u,v} = \hat{\delta}(u,v) + h(v) - h(u)$$
$$d_{1,2} = 2 + (-1) - 0$$
$$d_{1,3} = 2 + (-5) - 0$$
$$d_{1,4} = 2 + (0) - 0$$
$$d_{1,5} = 0 + (-4) - 0$$

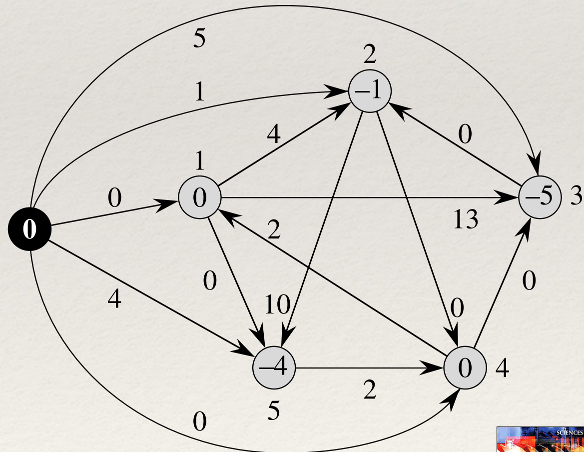
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | 1 | -3 | 2 | -4 |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

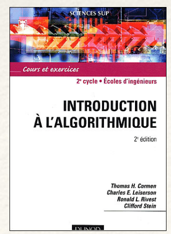


# Exemple



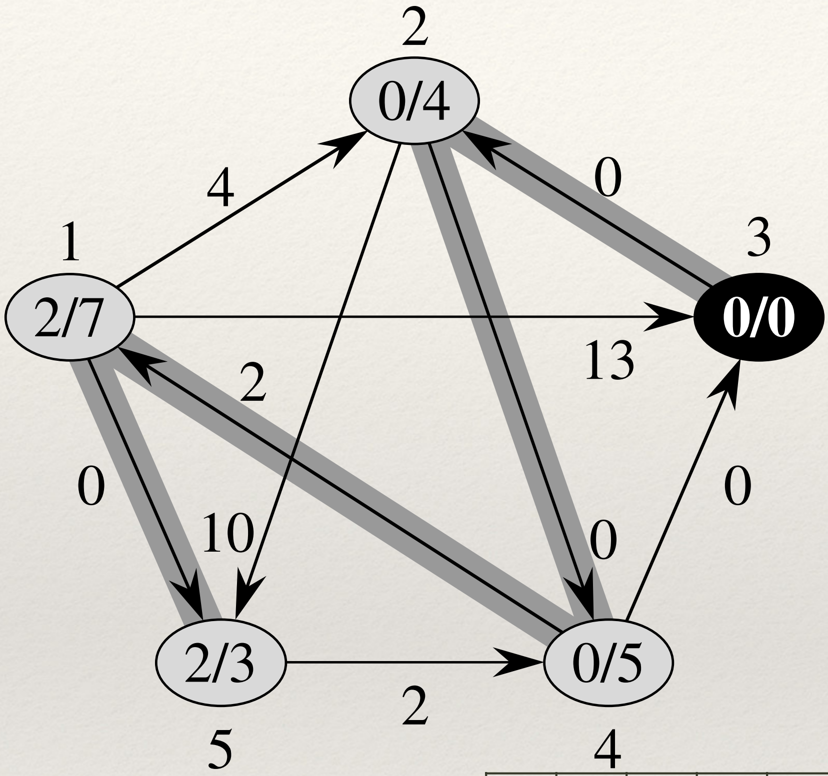



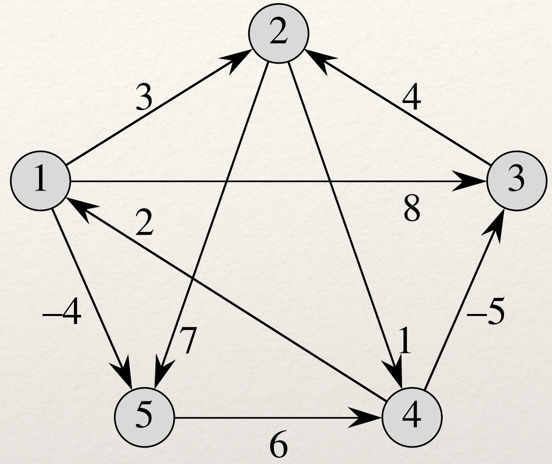

$$d_{u,v} = \hat{\delta}(u,v) + h(v) - h(u)$$
$$d_{2,1} = 2 + 0 - (-1)$$
$$d_{2,3} = 0 + (-5) - (-1)$$
$$d_{2,4} = 0 + 0 - (-1)$$
$$d_{2,5} = 2 + (-4) - (-1)$$

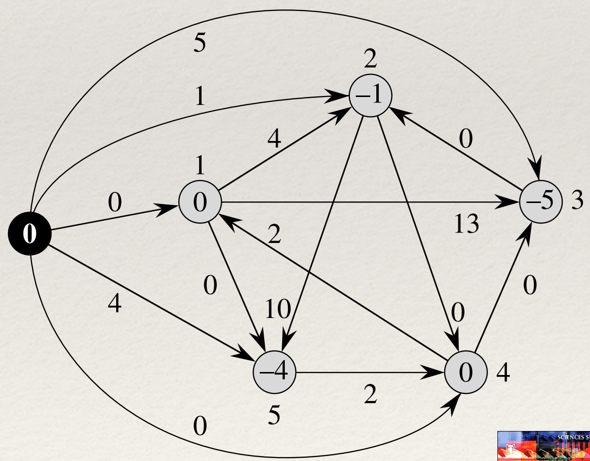
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | 1 | -3 | 2 | -4 |
| 2 | 3 | | -4 | 1 | -1 |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

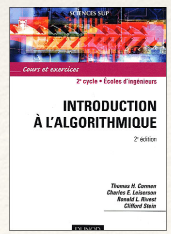
# Exemple



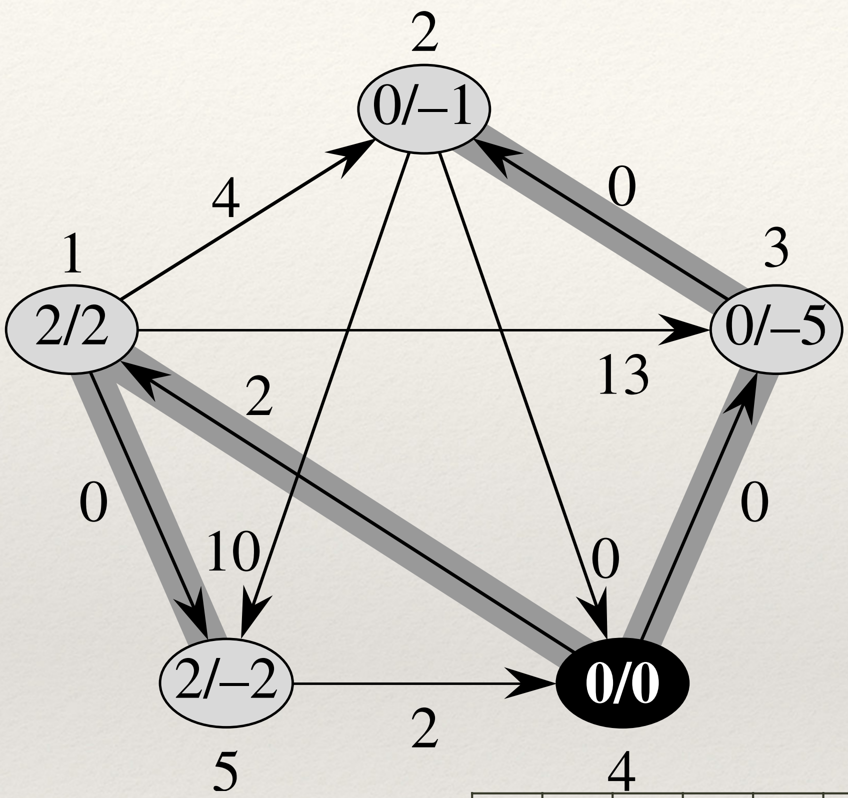

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |



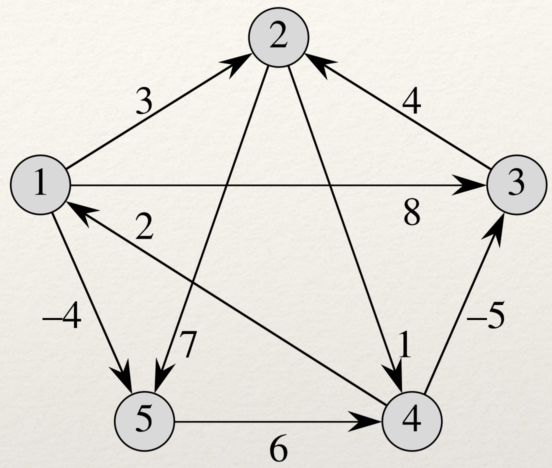



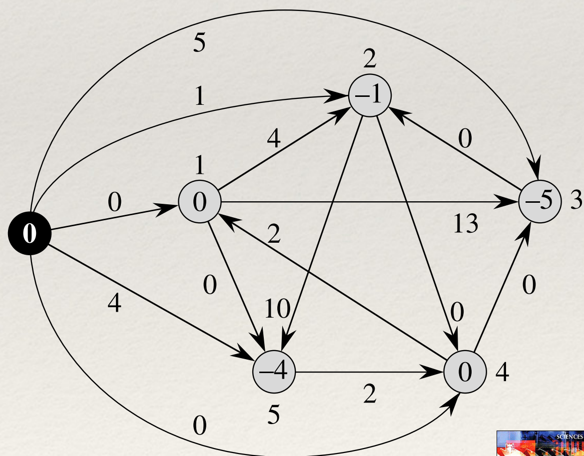



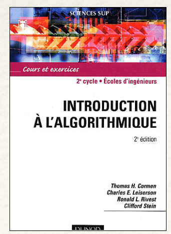
# Exemple



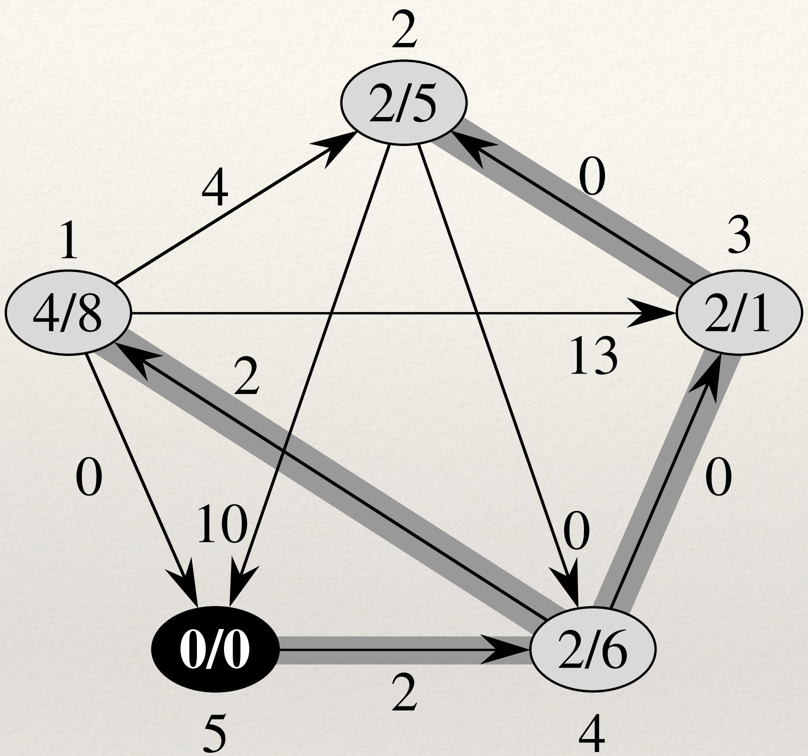

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

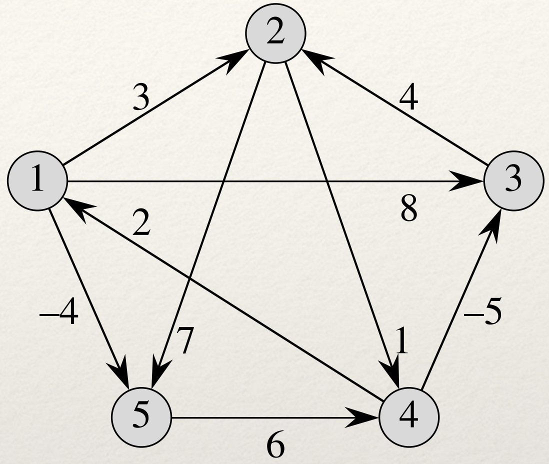


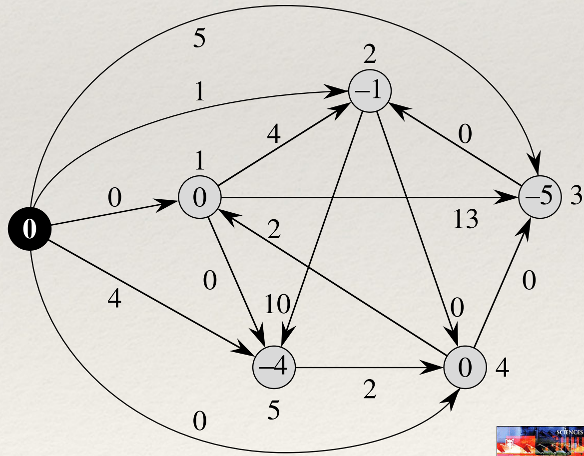



# Exemple




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | 1 | -3 | 2 | -4 |
| 2 | 3 | | -4 | 1 | -1 |
| 3 | 7 | 4 | | 5 | 3 |
| 4 | 2 | -1 | -5 | | -2 |
| 5 | 8 | 5 | 1 | 6 | |