Unité d’Enseignement RCP101 : Recherche Opérationnelle et Aide à la Décision

Cours 2 – Chemins de longueur optimale

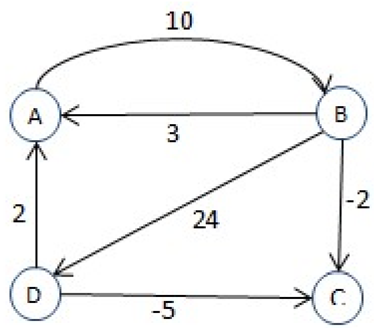
Conservatoire National des Arts et Métiers

UE RCP101 – Recherche Opérationnelle et Aide à la Décision – Plan du cours

- Partie 1- Eléments de Théorie des Graphes
  - *Généralités, fermeture transitive et connexité*
  - ***Chemins de longueur optimale***
- Partie 2 – Ordonnancement
  - *Méthode PERT*
  - *Méthode MPM*
- Partie 3 – Programmation linéaire
  - *Modélisation*
  - *Méthode du simplexe*
  - *Dualité*
- Partie 4 : Processus de Markov et files d'attente
- Partie 5 : Optimisation multicritères

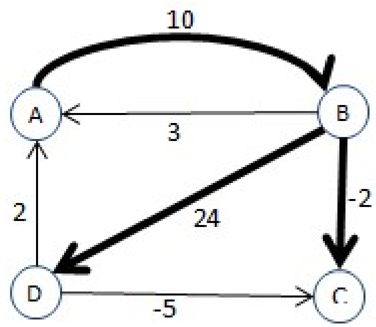
Cheminement optimal

- Soit G=(X,U) un graphe orienté valué
- $\ell : U \to \mathbb{R}$ fonction longueur, ou poids, ou coût, ou profit (<u>Ex</u> : ℓ(A,B)=10)
- On définit la longueur d'un chemin de G comme la somme des longueurs de chacun des arcs qui le composent
- Ainsi étant donné un chemin μ de G :



$$\mu = [u_1, u_2, \ldots, u_p]$$

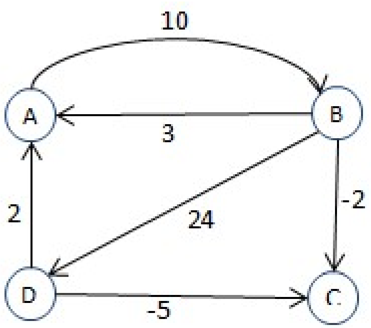
on définit **sa longueur** par :



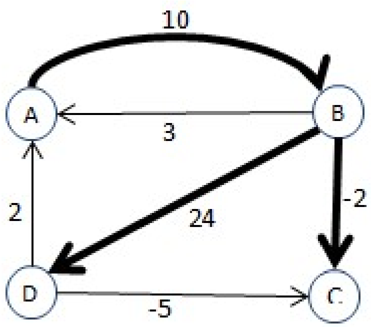
$$L(\mu) = \ell(u_1) + \ell(u_2) + \ldots + \ell(u_p)$$

Cheminement optimal

- **Problème 1** : étant donnés deux sommets x et y, trouver un chemin de longueur minimum de x à y



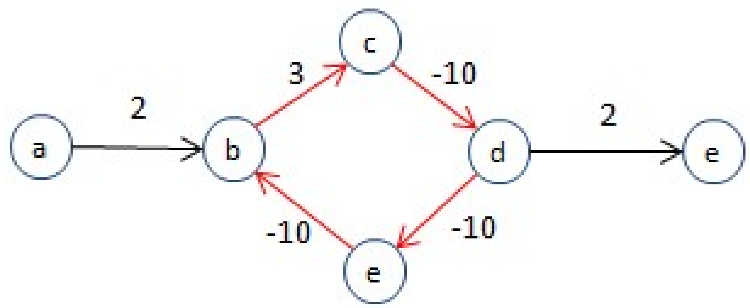
- **Problème 2** : étant donné un sommet x, trouver le plus court chemin de x à tous les sommets du graphe



- **Problème 3** : trouver, pour toute paire de sommets x et y, un plus court chemin de x à y

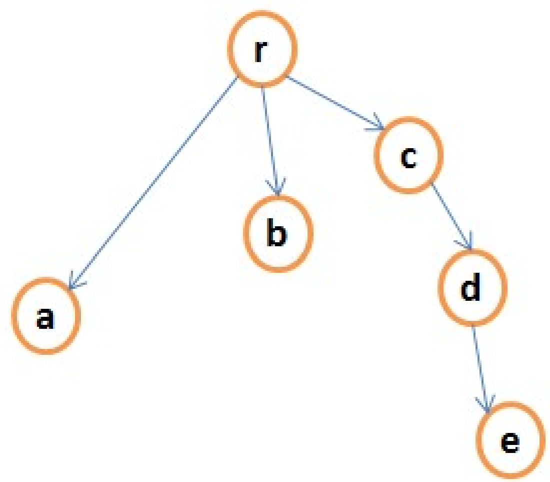
Cheminement optimal – Circuit absorbant

- <u>Définition</u> : on appelle **circuit absorbant** un circuit de longueur négative

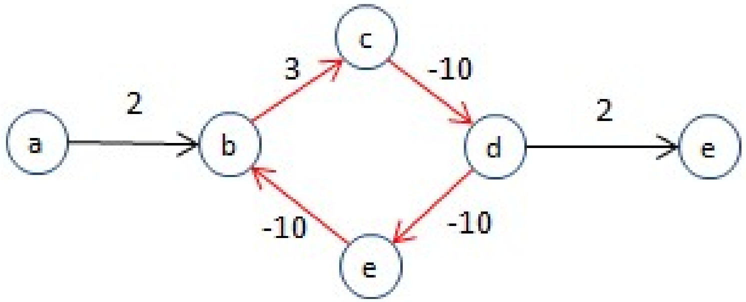


- <u>Définition</u> : on appelle **racine** du graphe orienté G = (X,U) tout sommet de X vérifiant la propriété suivante : pour tout sommet y de X ($\neq$x), il existe dans G un chemin de x à y.

- <u>Définition</u> : un graphe orienté G = (X,U), avec r$\in$X, est une **arborescence de racine r,** si G est un arbre et r une racine de G



Cheminement optimal – Circuit absorbant



- **<u>Théorème</u>** : une CNS pour que le problème 2 ait une solution est que x soit racine du graphe et que le graphe ne contienne pas de circuit absorbant
- Cas (faciles à résoudre) où l'on est sûr de l'absence de circuit absorbant :
  - si toutes les valuations sont positives ou nulles
  - si le graphe est sans circuit

Cheminement optimal – Cas des valuations positives (Dijkstra)

- Cas fréquent en pratique. Ex : routage dans un réseau de télécommunications
- On utilise l'algorithme de Dijkstra. Principe :
  - On étend une arborescence A, initialement réduite à la racine r
  - À chaque itération, on étend l'arborescence par un nouvel arc et un nouveau sommet (son extrémité)
  - À la fin de l'algorithme, l'arborescence construite donne pour chaque sommet x le plus court chemin de r à x

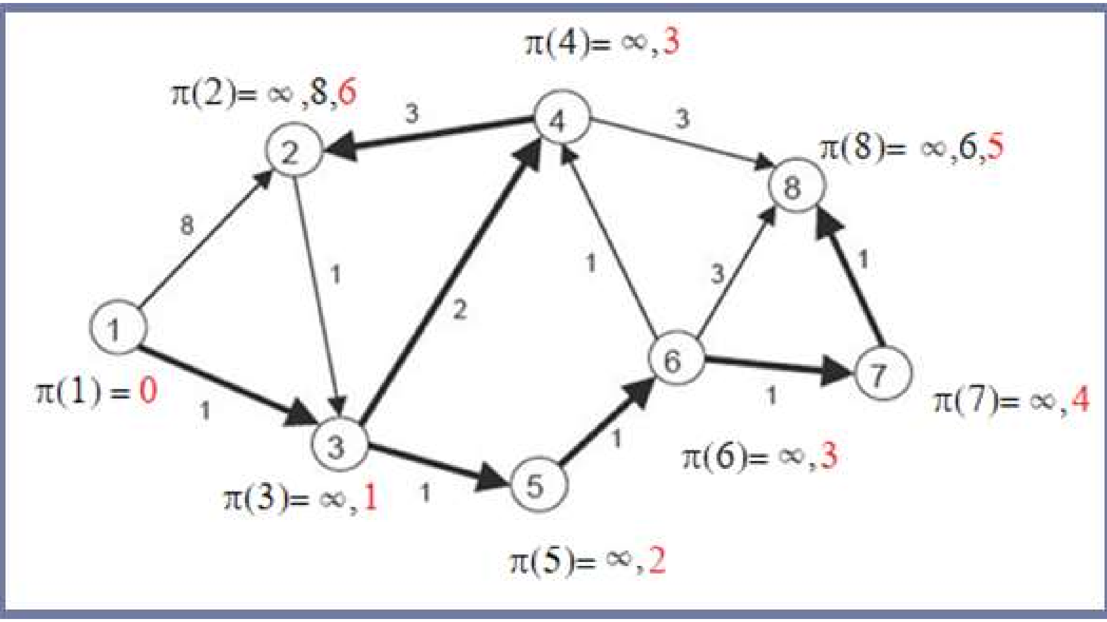
Cheminement optimal – Cas des valuations positives (Dijkstra)

- On appelle distance(r, x) la longueur d'un plus court chemin de r à x
- On définit deux applications : $\pi(x)$ et père(x)
- **<u>À la fin de l'algorithme</u>** : $\pi(x)$=distance(r, x) et père(x) désigne le père de x dans l'arborescence des plus courts chemins
- **<u>À une étape quelconque</u>** : $\pi(x)$ donne la longueur d'un plus court chemin de r à x n'empruntant que des sommets de A (ensemble des sommets déjà dans l'arborescence)
- Dans l'algorithme, $\ell(x \rightarrow y)$ désigne la valuation de l'arc (x, y).

```
algorithme Dijkstra(données: G = (X, U, ℓ); r : sommet; résultat: A : arborescence)
début
    A ← {r}
    pivot ← r
    π(r) ← 0
    pour tout x ≠ r faire π(x) ← ∞ fait
    pour j allant de 1 à n − 1 faire
        pour tout sommet y ∉ A et tel que y ∈ Γ⁺(pivot) faire
            si π(pivot) + ℓ(pivot → y) < π(y) alors
                π(y) ← π(pivot) + ℓ(pivot → y)
                père(y) ← pivot
            fin si
        fait
        // Assertion 1 : ∀y ∉ A, s'il existe au moins un arc dont l'origine est dans A
        // et dont l'extrémité est y, on a : π(y) = min_{x∈A et (x,y)∈U} π(x) + ℓ(x → y)
        Soit y tel que π(y) = min_{z∉A} π(z)
        pivot ← y
        // Assertion 2 : π(pivot) est la longueur d'un plus court chemin de r à pivot
        // et père(pivot) est le prédécesseur de pivot dans ce plus court chemin.
        A ← A ∪ {pivot}
    fait
fin
```

Algorithme de Dijkstra

- Un exemple

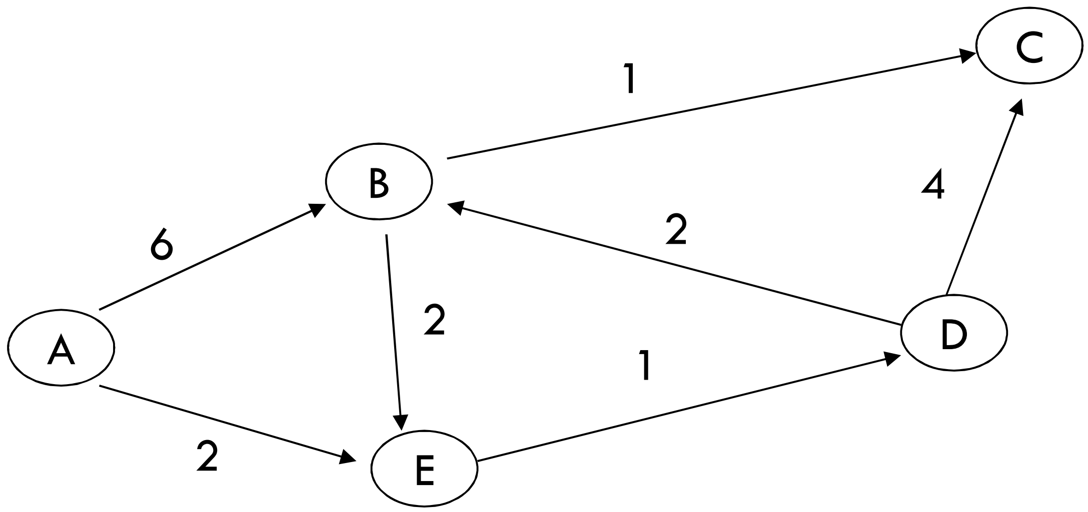


π	1	2	3	4	5	6	7	8
Init.	$\boxed{0}$	∞	∞	∞	∞	∞	∞	∞
$j=1$	0	8	$\boxed{1}$	∞	∞	∞	∞	∞
$j=2$	0	8	1	3	$\boxed{2}$	∞	∞	∞
$j=3$	0	8	1	$\boxed{3}$	2	3	∞	∞
$j=4$	0	6	1	3	2	$\boxed{3}$	∞	6
$j=5$	0	6	1	3	2	3	$\boxed{4}$	6
$j=6$	0	6	1	3	2	3	4	$\boxed{5}$
$j=7$	0	$\boxed{6}$	1	3	2	3	4	5

père	1	2	3	4	5	6	7	8
Init.	-	-	-	-	-	-	-	-
$j=1$	-	1	1	-	-	-	-	-
$j=2$	-	1	1	3	3	-	-	-
$j=3$	-	1	1	3	3	5	-	-
$j=4$	-	4	1	3	3	5	-	4
$j=5$	-	4	1	3	3	5	6	4
$j=6$	-	4	1	3	3	5	6	7
$j=7$	-	4	1	3	3	5	6	7

Exercice

- Appliquer l'algorithme de Dijkstra sur le graphe suivant en prenant comme racine le sommet A



Cheminement optimal – Cas des valuations positives (Dijkstra)

- **Remarque 1:** il peut exister plusieurs plus courts chemin de la racine r à un sommet i donné (qui sont tous de même longueur totale, minimale). L'algorithme se contente de trouver, parmi tous les plus courts chemins de r à i, un seul de ces plus courts chemins.

- **Remarque 2 :** si on applique l'algorithme de Dijkstra à un graphe comportant des valuations négatives, l'algorithme peut ne pas trouver les plus courts chemins (trouver en exercice un contre-exemple)

- **Remarque 3 :** il est possible de modifier l'algorithme afin qu'il trouve les plus longs chemins dans un graphe ne comportant que des valuations négatives ou nulles

Cheminement optimal – Cas des valuations positives (Dijkstra)

- **Preuve de l'algorithme :** on prouve les assertions 1 et 2 par récurrence sur j
- **Complexité** de l'algorithme :
  - À chaque itération, l'actualisation de π nécessite O(d$^+$(pivot) opérations
    - Chaque sommet devient tour à tour pivot
    - Nb d'opérations : $$O\left(\sum_{x \in X} d^+(x)\right) = O(m) \text{ où } m = |U|$$
  - Détermination du pivot :
    - recherche du plus petit élément parmi q, q décroissant de n-1 à 1
    - Nb d'opérations : $$O\left(\sum_{q=1}^{n-1} q\right) = O\left(\frac{n(n-1)}{2}\right) = O(n^2)$$
  - Or $m \leq n^2$. Complexité de l'algorithme : $O(n^2)$

le cnam

Algorithme de Ford : cas général (valuations quelconques, graphe avec ou sans circuit - Problème 2)

- $G=(X,U)$ un graphe valué avec $X=\{x_0,x_1,x_2, \ldots.., x_{n-1}\}$: on choisit un ordre arbitraire $(x_0,x_1,x_2, \ldots.., x_n)$ sur les sommets (x_0 doit toutefois être le sommet de départ)
- On veut déterminer la longueur du plus court chemin de x_0 à tout autre sommet x_i de G
- On associe à tout sommet x_i une pondération λ_i
- **Algorithme :**
  - **Initialisation :** Prendre $\lambda_0 = 0$ et $\lambda_i = +\infty$ pour tout $i \neq 0$.
  - **Répéter :**

    Pour i de 0 à n **faire**

    Étant donné l'arc $x_i \rightarrow x_j$ de G

    si $\lambda_i + \ell(x_i \rightarrow x_j) < \lambda_j$ **alors** $\lambda_j \leftarrow \lambda_i + \ell(x_i \rightarrow x_j)$

    fait

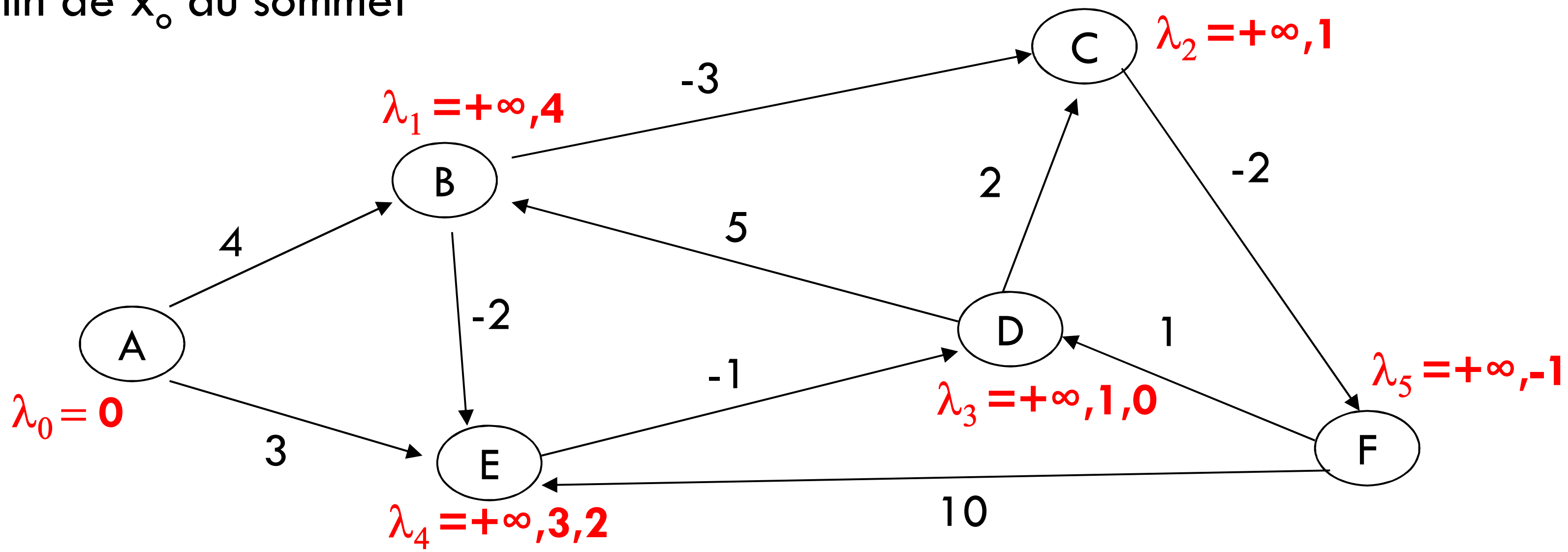
    jusqu'à la stabilisation de toutes les pondérations λ_i

Algorithme de Ford : Un exemple

- Ordre choisi pour les sommets du graphe (ordre alphabétique) :

$$x_0 = A,\ x_1 = B,\ x_2 = C,\ x_3 = D,\ x_4 = E,\ x_5 = F$$

- En noir les valuations
- En rouge l'évolution des λ_i (la dernière valeur donne la longueur du plus court chemin de x_o au sommet



le cnam

Remarques concernant l'algorithme de Ford

- **Remarque 1 :** On démontre qu'à la fin de l'algorithme, la pondération λ_i représente la longueur du plus court chemin du sommet x_0 au sommet x_i. En plus, ce résultat est valable indépendamment des signes des valeurs des arcs.
- **Remarque 2 :** Si on effectue l'initialisation $\lambda_i = -\infty$ (pour $i \neq 0$) et si on remplace la condition de l'itération de base par :

  « **si** $\lambda_i + \ell\,(x_i \rightarrow x_j) > \lambda_j$ **alors** $\lambda_j \leftarrow \lambda_i + \ell(x_i \rightarrow x_j)$ »

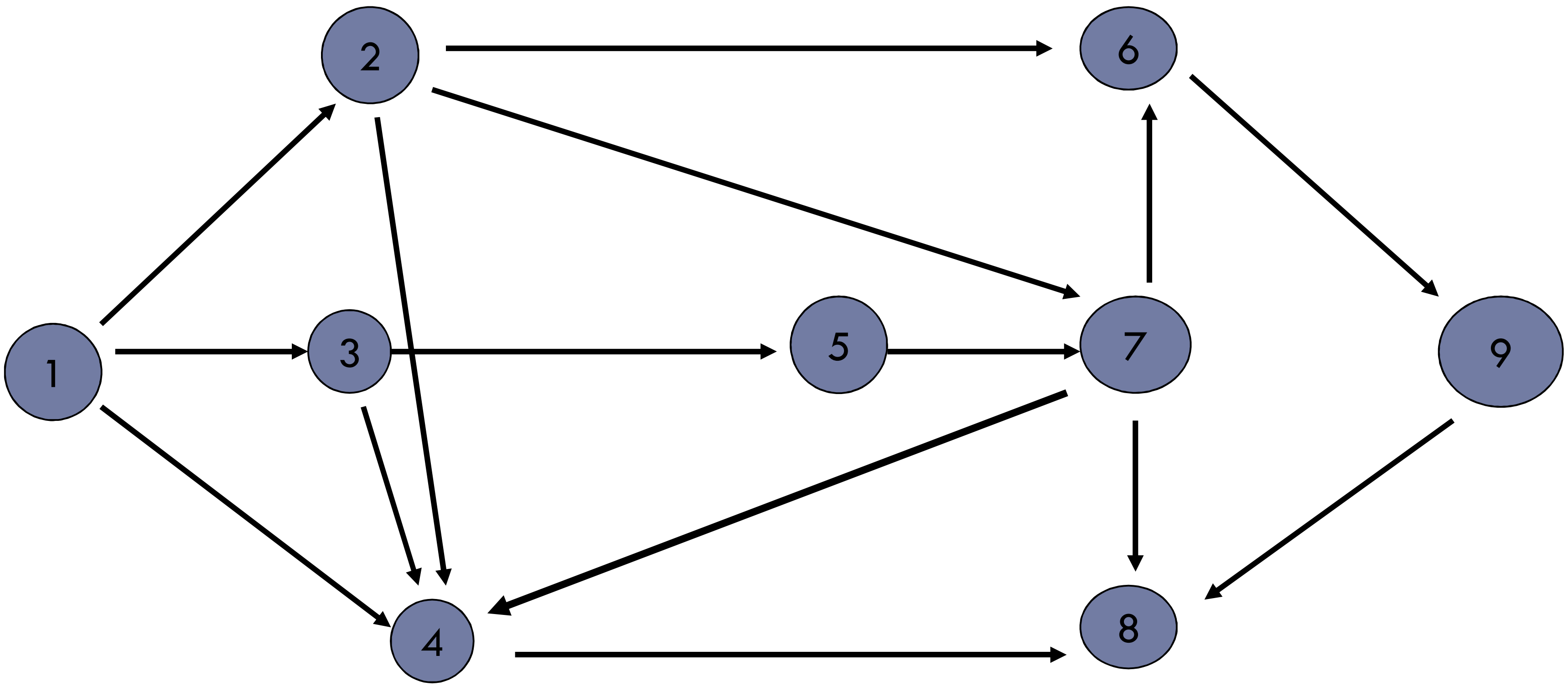
  …alors λ_i représente alors la longueur du plus long chemin et l'algorithme de Ford trouve les plus longs chemins au lieu des plus courts
- **Remarque 3 :** L'algorithme de Ford est général, mais ne précise pas l'ordre du parcours des arcs du graphe. Le parcours des arcs de G peut se réaliser en visitant tous les sommets de G (dans un ordre donné) et puis en parcourant les arcs incidents vers l'extérieur (ou l'intérieur) de chaque sommet visité.
- **L'efficacité de cet algorithme dépend de l'ordre de visite des sommets de G.**

Remarques concernant l'algorithme de Ford

- **Remarque 4 :** Si le graphe auquel on applique l'algorithme possède un circuit absorbant, l'algorithme ne se termine pas (boucle sans fin), les λ_i étant sans cesse mis à jour. Il est toutefois possible de modifier l'algorithme en limitant le nombre d'itérations de la boucle principale (« répéter » … « jusqu'à ») à n et **un changement dans les λ_i à la dernière itération traduit l'existence d'un circuit absorbant.**

- **Remarque 5 :** Tel qu'il est présenté, l'algorithme calcule uniquement les **longueurs** des plus courts chemins du sommet racine à tous les autres. Il est possible de le modifier de sorte à mémoriser les plus courts chemins (et pas seulement leur longueur). Il faut alors, à chaque amélioration, mettre à jour un tableau des « pères » de chaque sommet, de façon similaire à l'algorithme de Dijkstra.

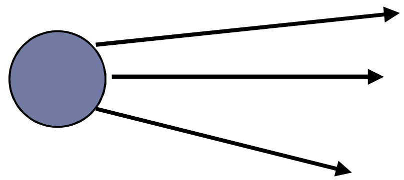
Graphes sans circuit



le cnam

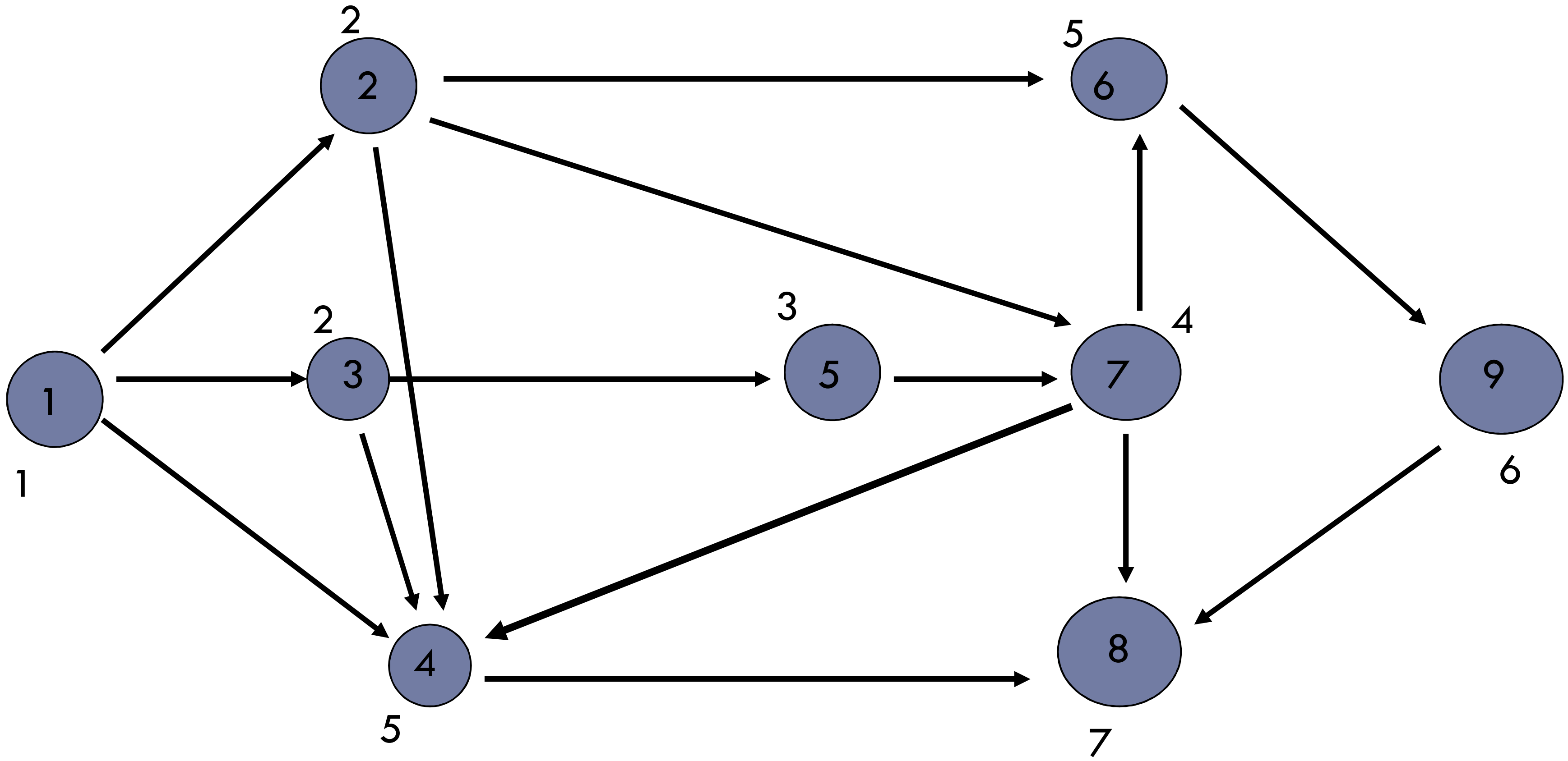
Graphes sans circuit

- **Propriété** : Dans un graphe orienté sans circuit il existe au moins un sommet de degré intérieur nul.

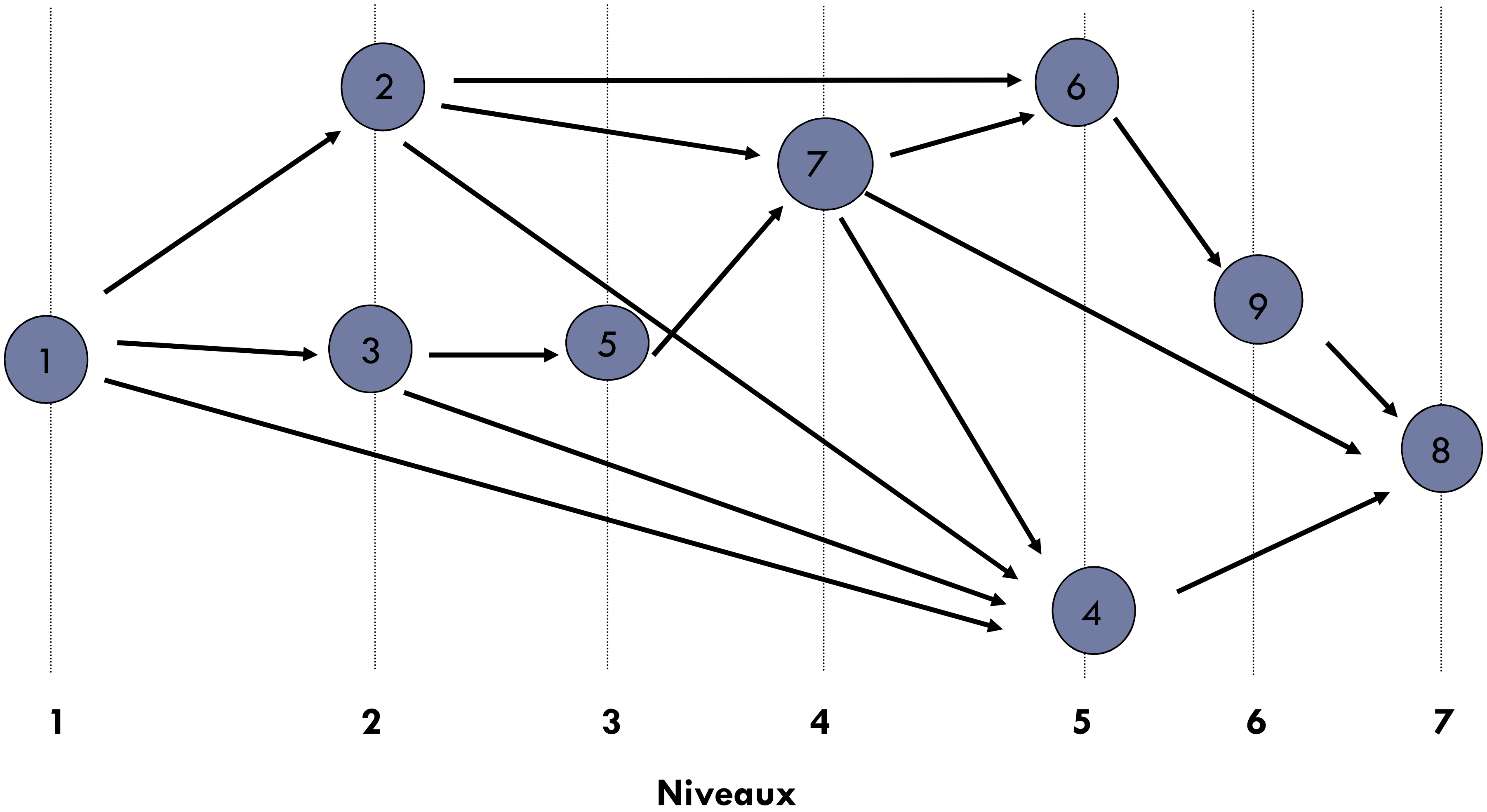


- **Mise en niveaux d'un graphe sans circuit**
  - Poser $k \leftarrow 2$; Étiqueter par « 1 » les sommets ayant un degré intérieur nul.
  - Tant qu'il y a des sommets non étiquetés, répéter les deux étapes suivantes :
    - Étiqueter par k tous les sommets dont tous les prédécesseur sont étiquetés avec une étiquette $< k$;
    - $k \leftarrow k + 1$;

Exemple :
Résultat de la mise en niveaux



Exemple : Résultat de la mise en niveaux



le cnam

Ordre topologique

- **Ordre topologique** : Soit $X=\{x_1,x_2, \ldots, x_n\}$ l'ensemble des sommets du graphe sans circuit G. L'ordre $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ sera dit **ordre topologique** si :

  $$\text{pour tout arc } x_{i_l} \rightarrow x_{i_k} \text{ de G on a } l<k$$

  (autrement dit, tout sommet doit avoir un numéro d'ordre strictement supérieur à chacun de ses prédécesseurs)

- Si on commence par étiqueter les sommets par l'algorithme de mise en niveaux et si on numérote ensuite les sommets des niveaux consécutifs, on obtient un ordre topologique.

- **NB** : dans un même niveau, l'ordre adopté pour les sommets de ce même niveau n'a pas d'importance : il existe en général plusieurs ordres topologiques possibles pour un graphe sans circuit.

Cas d'un graphe sans circuit

- Pour un graphe sans circuit, on peut utilement appliquer **l'algorithme de Ford** en choisissant comme ordre particulier des sommets **un ordre topologique**.
- Si x_0 est une entrée (sommet sans prédécesseur) de G et si l'on parcourt les sommets de G dans un ordre topologique, les pondérations se stabilisent à la fin du premier parcours.
- L'algorithme de FORD devient alors simplement :
  - Poser $\lambda_0 \leftarrow 0$;
  - Parcourir les sommets de G suivant un ordre topologique et pour chaque sommet x_i visité faire :

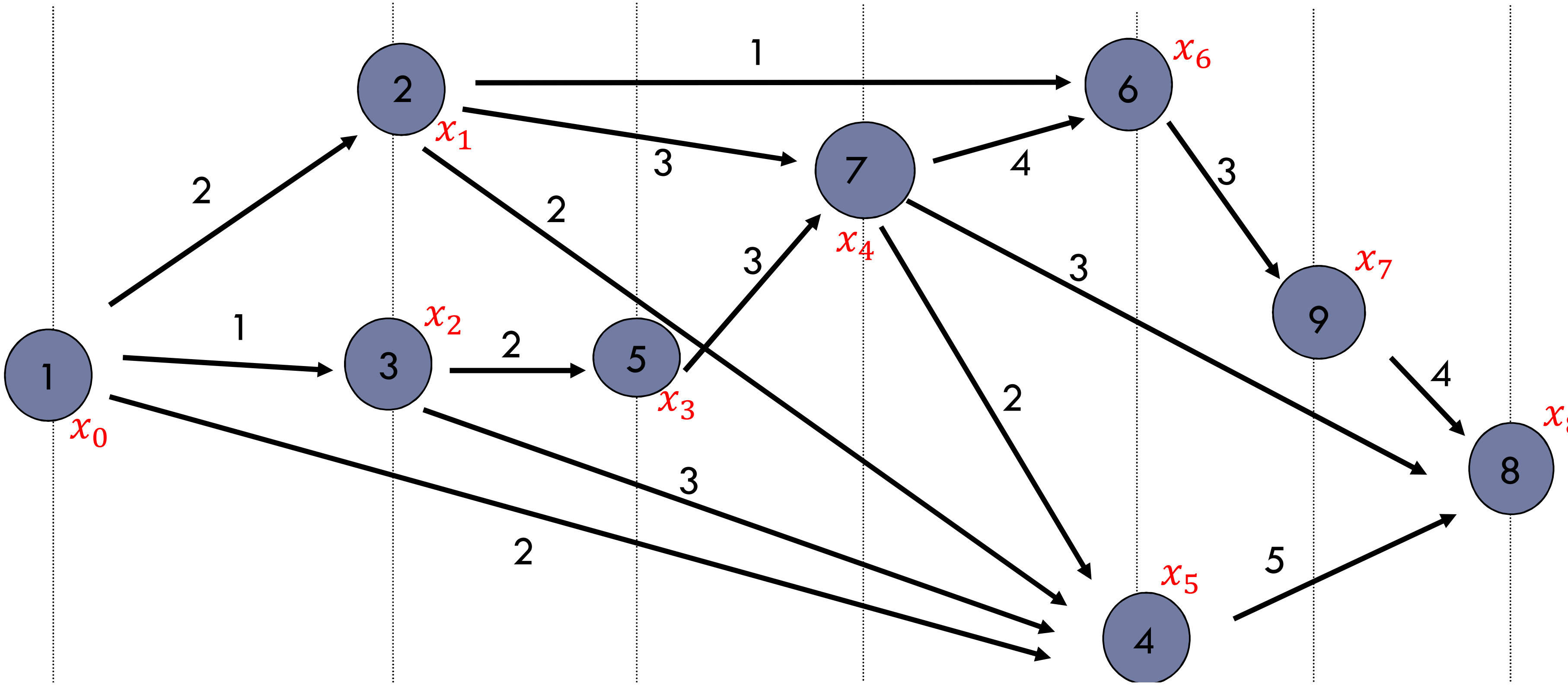
$$\lambda_i = \min_{x_k \in \Gamma^-(x_i)} (\lambda_k + l(x_k \to x_i))$$

Prédécesseurs de x_i

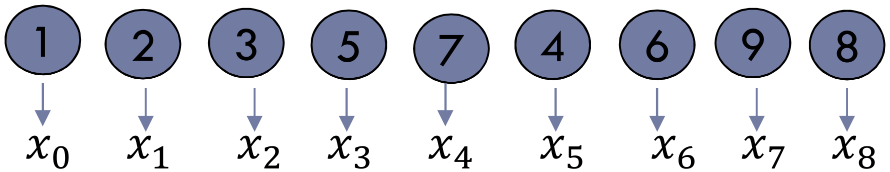
Cas d'un graphe sans circuit

- **Remarque :** L'algorithme de Ford, appliqué sur un ordre topologique des sommets (un tel ordre existe si et seulement si le graphe est sans circuit) est appelé **algorithme de Bellman**.

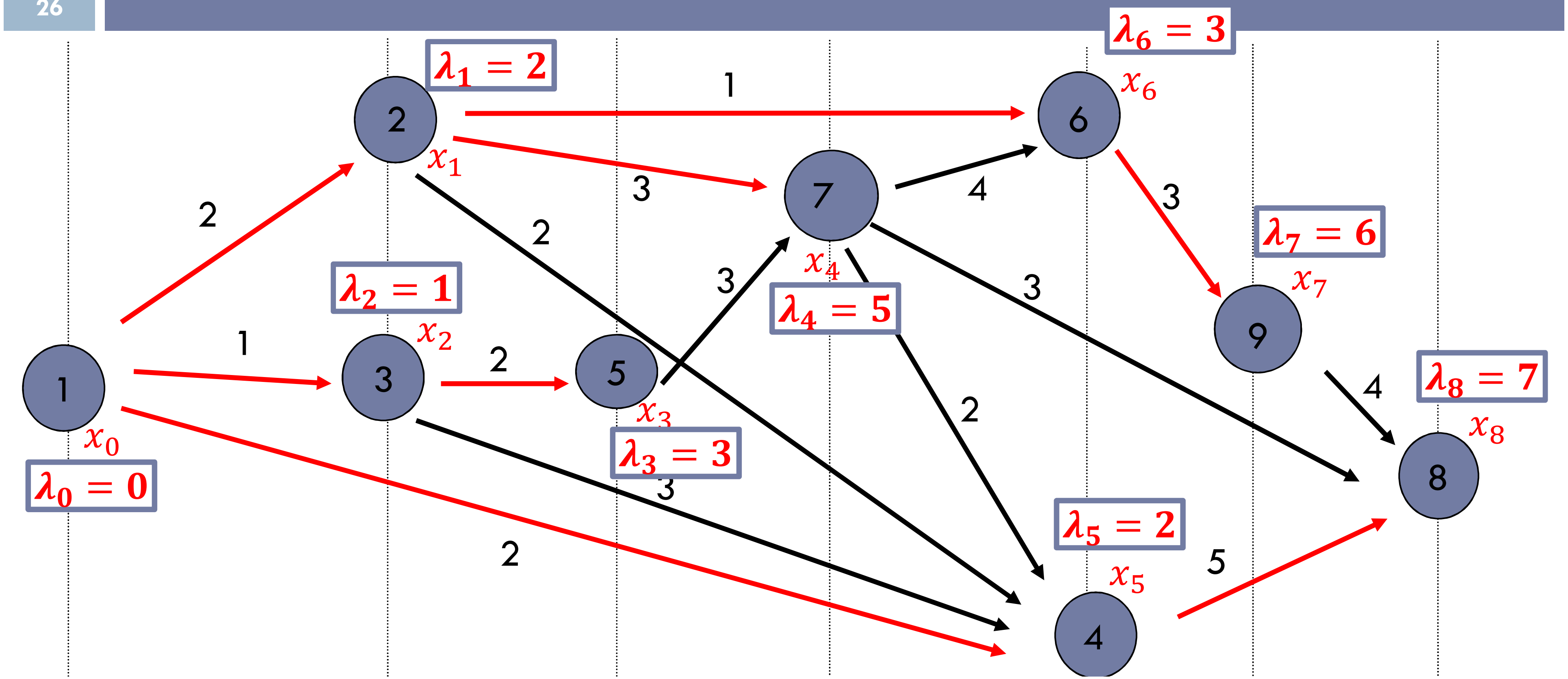
Exemple (suite) : application de l'algorithme sur un graphe valué



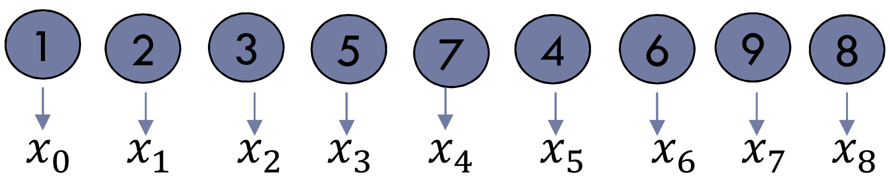
Plusieurs ordres topologiques sont possibles. Nous retenons l'ordre $\{x_0, x_1, x_2, \ldots x_8\}$ suivant :



Exemple (suite) : application de l'algorithme sur un graphe valué



Plusieurs ordres topologiques sont possibles. Nous retenons l'ordre $\{x_0, x_1, x_2, \dots x_8\}$ suivant :



Méthode matricielle – Algorithme de Floyd-Warshall (Problème 3)

- Recherche des plus courts chemins **entre tout couple de sommets de G**, **sans hypothèse sur G** (cas général)
- On construit une suite de matrice (n × n) : $M_0, M_1, \ldots, M_n$
- On note v_{ij}^{k} les éléments de la matrice M_k.
- À la fin de l'algorithme, v_{ij}^{n} représente la longueur du plus court chemin de x_i à x_j dans G.
- On mémorise les chemins à l'aide d'une matrice (n × n) `P` : à la fin de l'algorithme, **`Pij` représente le prédécesseur de `j` dans un chemin optimal de `i` à `j`**
- La présence d'un **circuit absorbant** dans le graphe se détecte par : $v_{ii}^{k} < 0$ (on peut alors arrêter l'algorithme).
- **L'absence de chemin** de i à j dans le graphe se détecte à la fin de l'algorithme par : $v_{ij}^{n} = +\infty$
- **Complexité** de l'algorithme : $O(n^3)$

le cnam

Méthode matricielle – Algorithme de Floyd-Warshall (Problème 3)

Algorithme :

- **Initialisation** : Poser

$$v_{ij}^0 = \begin{cases} l(x_i \to x_j) \text{ si } x_i \to x_j \in U \\ +\infty \text{ sinon} \end{cases} \quad \text{et} \quad P_{ij} = \begin{cases} i \text{ si } x_i \to x_j \in U \\ -(\text{indéterminé}) \text{ sinon} \end{cases}$$

- **Calcul des éléments de M_k** à partir de ceux de M_{k-1} :
  Pour k = 1 à n calculer chaque élément v_{ij}^k de M_k par :

$$\text{si } \left(v_{ik}^{k-1} + v_{kj}^{k-1} < v_{ij}^{k-1}\right) \text{ alors } v_{ij}^k \leftarrow v_{ik}^{k-1} + v_{kj}^{k-1} ; P_{ij} \leftarrow P_{kj} \quad \text{sinon } v_{ij}^k = v_{ij}^{k-1}$$

- On peut montrer par récurrence sur k que v_{ij}^k est la valeur minimale des chemins allant de i à j, n'empruntant que des sommets dont les indices sont inférieurs ou égaux à k, et composés d'au plus k+1 arcs.