INFO0501

ALGORITHMIQUE AVANCÉE

COURS 6

GRAPHES
PLUS COURTS CHEMINS

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Pierre Delisle
Département de Mathématiques, Mécanique et Informatique
Octobre 2020

Plan de la séance

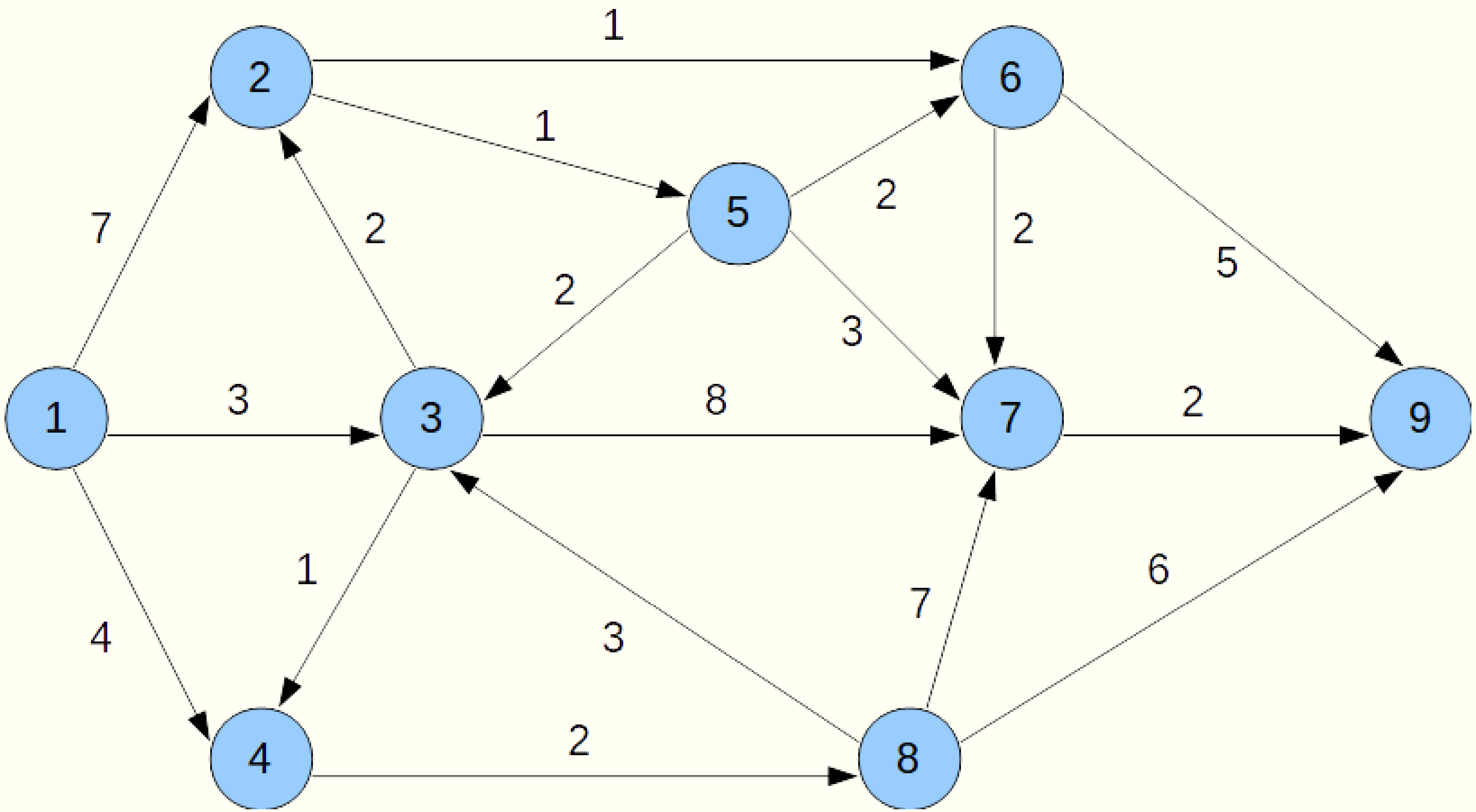
- Le problème des plus courts chemins
- Plus courts chemins à origine unique
  - Algorithme de Bellman-Ford
  - Le cas des graphes acycliques orientés
  - Algorithme de Dijkstra
- Plus courts chemins entre toutes paires de sommets
  - Algorithme de Floyd-Warshall
- Bibliographie
  - T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Algorithmique", 3e édition, Dunod, 2010

PROBLÈME DE PLUS COURTS CHEMINS

Chemin, chemin élémentaire et circuit

- Soit $G = (S, A)$ un graphe orienté et valué
  - On attribue un poids (longueur, coût) $p(u, v)$ à l'arc (u, v)
- Un chemin est une suite
  - $s_1 a_1 s_2 a_2 \ldots\ a_n s_{n+1}$
  - de sommets s_i et d'arcs a_j
  - telle que l'arc a_j ait pour origine s_j et pour extrémité s_{j+1}
- Poids du chemin : somme des poids des arcs
- Chemin élémentaire : n'utilise pas 2 fois le même sommet
- Circuit : chemin dont les extrémités coïncident

Exemple

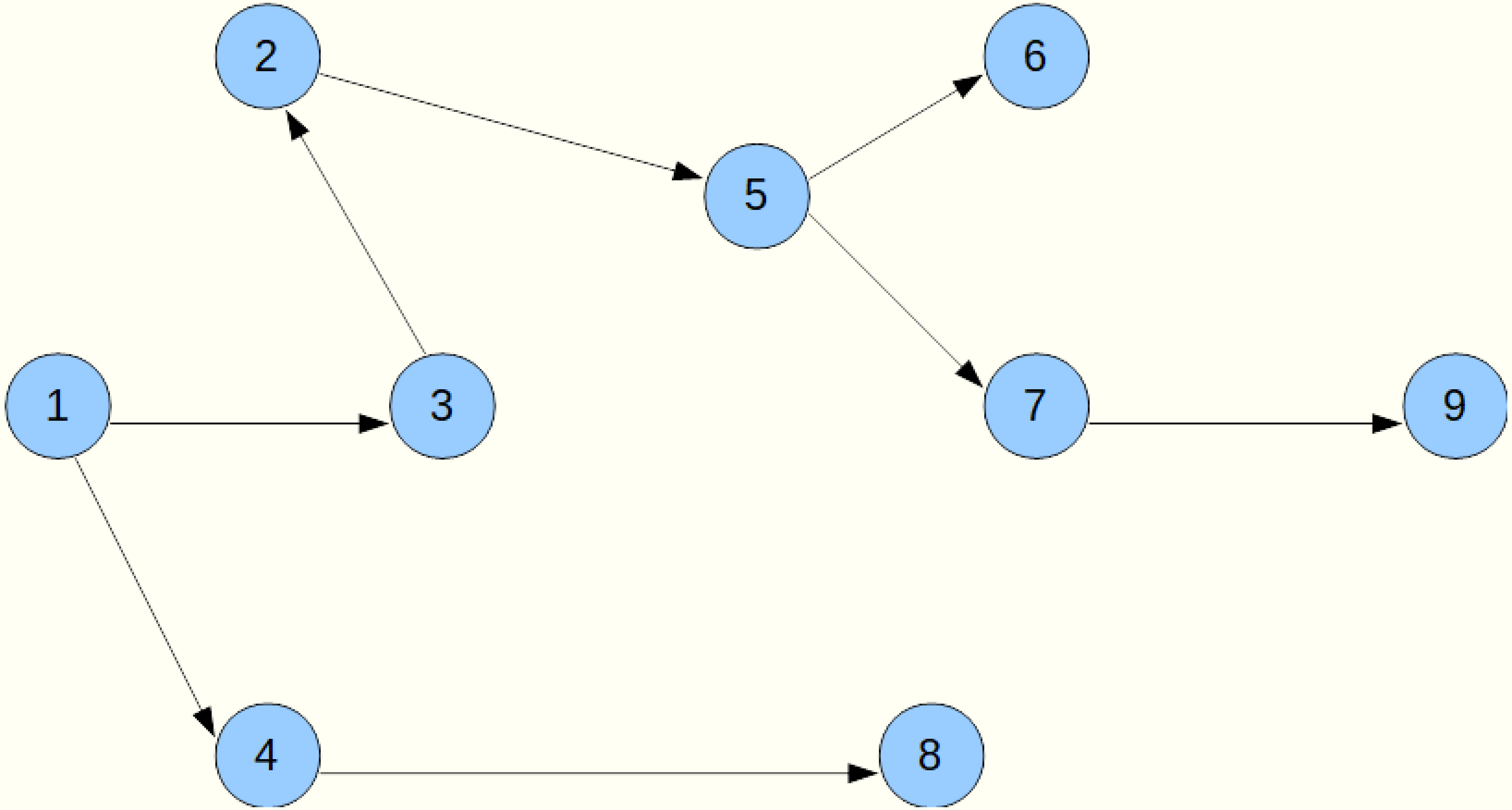


- 1, 3, 7, 9 est un chemin élémentaire de poids 13 entre 1 et 9
- 1, 4, 8, 9 est un chemin élémentaire de poids 12 entre 1 et 9
- 2, 5, 3, 2 est un circuit de poids 5

Arborescence

- La racine d'un graphe G orienté est un sommet s tel qu'il existe un chemin de s à tout sommet de G
- Une arborescence de racine s est un graphe orienté tel que
  - Le graphe non orienté sous-jacent (en oubliant les orientations) est un arbre
  - Pour tout sommet x, l'unique chaîne entre s et x du graphe non orienté sous-jacent correspond à un chemin de s vers x dans l'arborescence

Exemple



- Arborescence dont la racine est le sommet 1
- Le graphe non orienté sous-jacent est un arbre
- L'unique chaîne entre 1 et tout sommet est un chemin

Remarques

- Un graphe orienté A est une arborescence de racine s si et seulement si
  - Le graphe non orienté sous-jacent est un arbre
  - Le degré entrant de s est égal à 0
  - Le degré entrant de tout sommet x différent de s est égal à 1
- On connaît donc une arborescence lorsque l'on connaît, pour chaque sommet, son père dans l'arborescence
- Pour la suite, on suppose qu'il existe toujours
  - un chemin d'un sommet vers un autre
  - un chemin d'un sommet vers tous les autres
- On peut toujours considérer qu'on travaille avec un graphe complet
  - S'il n'existe pas d'arc entre les sommets x et y, on peut en ajouter un de poids infini
  - S'il n'existe pas de chemin entre x et y, le plus court chemin entre x et y sera de poids infini

Définition du problème

- Problème 1 : Problème du plus court chemin pour un couple de sommets donné
  - Étant donnés deux sommets u et v, trouver un chemin de poids minimal (ou plus court chemin) de u à v
- Problème 2 : Problème du plus court chemin à origine unique
  - Étant donné un sommet d'origine s, trouver un plus court chemin de s à tous les sommets du graphe
  - Revient à déterminer une arborescence de racine s couvrant les sommets du graphe et constituée de plus courts chemins de s aux autres sommets
- Problème 3 : Problème du plus court chemin pour tout couple de sommets
  - Trouver, pour toute paire de sommets u et v, un plus court chemin de u à v

- Il existe des algorithmes polynomiaux pour résoudre ces problèmes
- On ne connaît pas, pour résoudre le problème 1, de bien meilleure solution que de résoudre le problème 2
  - Pour trouver le plus court chemin entre u et v, il faut calculer les plus courts chemins entre u et tous les sommets
- Pour résoudre le problème 3, on peut soit
  - Résoudre le problème 2 à partir de tout sommet (S fois pour un graphe d'ordre S)
  - Utiliser une méthode matricielle
- Le problème 2 (plus courts chemins à origine unique) est donc central

PLUS COURTS CHEMINS À ORIGINE UNIQUE

Résolution du problème du plus court chemin à origine unique

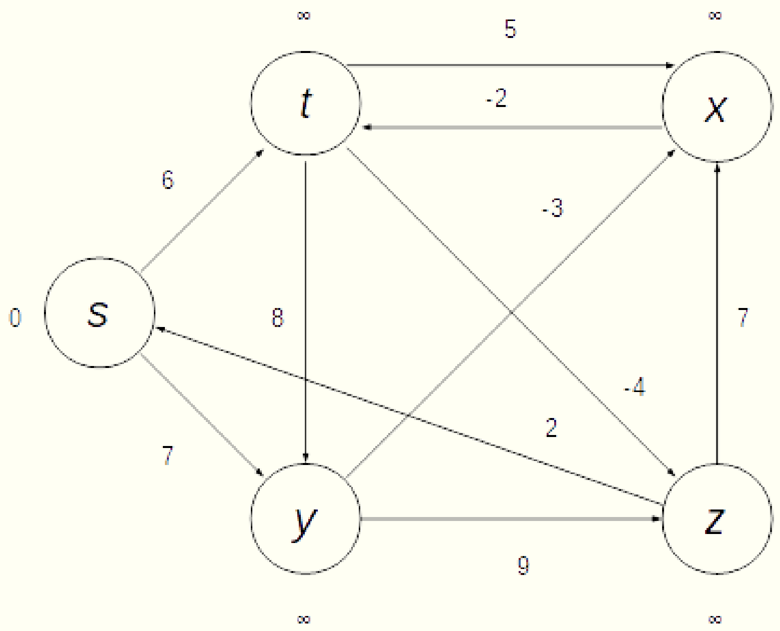
- Cas général : les arcs peuvent avoir des poids négatifs
  - Algorithme de Bellman-Ford
- Cas particulier des graphes orientés acycliques
- Cas particulier : tous les poids sont positifs
  - Algorithme de Dijkstra
- Les 3 algorithmes
  - Gèrent, pour chaque sommet, des valeurs *distance* et *père*
  - Utilisent la technique du **relâchement**

Gestion des sommets dans le calcul des plus courts chemins

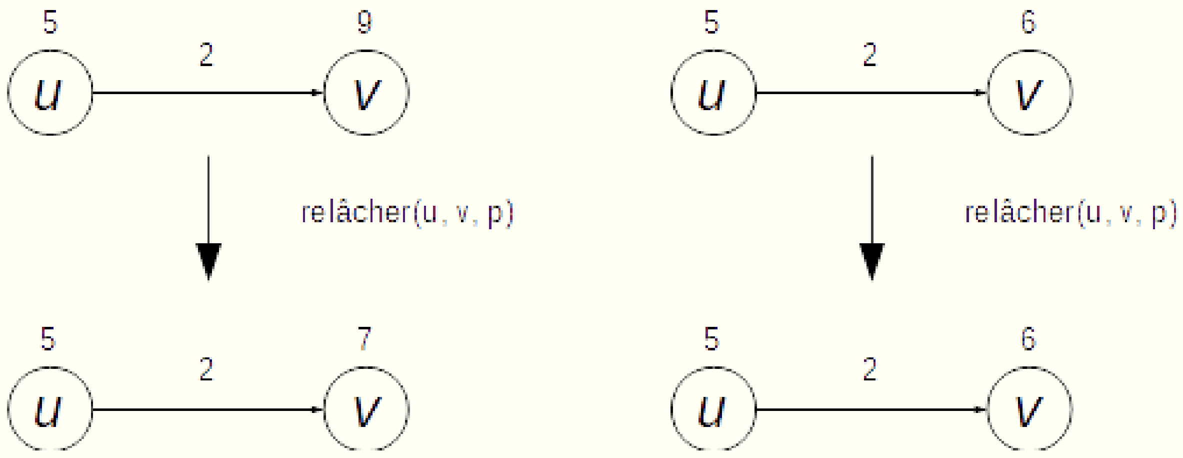
- Pour chaque sommet $v \in S$
  - On gère les attributs *v.d et v.père*
- *v.d*
  - Majorant du poids d'un plus court chemin entre l'origine s et v
  - Durant l'algorithme → estimation du plus court chemin
  - À la fin de l'algorithme → plus court chemin
- *v.père*
  - Père de v dans l'arborescence des plus courts chemins
  - Durant l'algorithme → père dans le plus court chemin estimé
  - À la fin de l'algorithme → père dans le plus court chemin

Gestion des sommets dans le calcul des plus courts chemins

- Initialisation
  - Origine *s*
    - $d \to 0$
    - *père* → NIL
  - Autres sommets
    - $d \to \infty$
    - *père* → NIL



- Relâchement
  - Relâcher un arc (*u*, *v*)
    - Tester si on peut améliorer le plus court chemin vers *v* ...
    - ... trouvé jusqu'ici ...
    - ... en passant par *u* ...
    - ... et, dans l'affirmative, actualiser *v.d* et *v.père*



Algorithme de Bellman-Ford

- Résout le problème des plus courts chemins à origine unique dans le cas général
  - Les poids d'arc peuvent avoir des valeurs négatives
- Indique s'il existe un circuit de poids strictement négatif accessible à partir de l'origine
  - Les plus courts chemins ne sont alors pas bien définis (on pourrait emprunter ce circuit à l'infini)
- S'il n'y a pas de circuit absorbant, donne les plus courts chemins et leurs poids

- Principe
  - Initialisation du sommet d'origine s
    - d $\rightarrow$ 0
    - père $\rightarrow$ NIL
  - Initialisation des autres sommets
    - d $\rightarrow$ ∞
    - père $\rightarrow$ NIL
  - Relâchement de tous les arcs $|S| - 1$ fois

- Exemple 1 : Algorithme de Bellman-Ford

Temps d'exécution

- Initialisation
  - $O(S)$
- On relâche S fois les A arcs
  - $O(SA)$
- À la fin, on parcourt tous les arcs pour vérifier la présence d'un circuit de poids strictement négatif
  - $O(A)$
- Total
  - $O(SA)$

Plus courts chemins à origine unique dans les graphes orientés acycliques

- Principe
  - En relâchant les arcs d'un graphe orienté acyclique dans l'ordre topologique des sommets …
  - … on peut calculer les plus courts chemins à partir d'une origine unique …
  - … en temps $O(S + A)$
  - (Une seule passe dans les arcs est nécessaire (plutôt que S passes pour le cas général))
- Exemple 2 : Algorithme des plus courts chemins dans un graphe orienté acyclique

- Temps d'exécution
- Tri topologique
  - $O(S + A)$
- Initialisation
  - $O(S)$
- On relâche une fois les A arcs
  - $O(A)$
- Total
  - $O(S + A)$

Algorithme de Dijkstra

- Résout le problème des plus courts chemins à origine unique dans le cas où tous les arcs ont des poids positifs ou nuls
- Temps d'exécution inférieur à Bellman-Ford
- Gère un ensemble de sommets ...
  - ... dont les longueurs de plus court chemin à partir de l'origine s ont déjà été calculées ...
  - ... en choisissant successivement un sommet u dont l'estimation de plus court chemin est minimale...
  - ... et en relâchant ensuite les arcs partant de u

- Exemple 3 : Algorithme de Dijkstra

Temps d'exécution

- Dépend de la gestion de la file de priorités
  - Insertion initiale des sommets
  - Recherche du sommet de distance minimale
  - Modification de la valeur des distances

	Insérer (appelé une fois par sommet)	Extraire-min (appelé une fois par sommet)	Diminuer-clé (appelé A fois au total)	Total
Tableau	$O(1)$ par appel $O(S)$ au total	$O(S)$ par appel $O(S^2)$ au total	$O(1)$ par appel $O(A)$ au total	$O(S^2)$ (S^2 toujours > A)
Tas min	$O(1)$ par appel $O(S)$ au total	$O(\lg S)$ par appel $O(S \lg S)$ au total	$O(\lg S)$ par appel $O(A \lg S)$ au total	$O(A \lg S)$ (tous les sommets accessibles de s : A toujours $\geq$ S - 1)

PLUS COURTS CHEMINS ENTRE TOUTES PAIRES DE SOMMETS

Plus courts chemins entre toutes paires de sommets

- Exemple
  - Création d'un atlas routier
- Résultat
  - Matrice de distances
  - L'intersection de la ligne u et de la colonne v contient le poids du plus court chemin entre u et v
- Pour résoudre ce problème
  - On peut appeler S fois un algorithme de plus court chemin à origine unique → une fois par sommet

```
PCC-TOUS-À-TOUS (G, p)
    Pour chaque sommet u ∈ G.S
        Dijkstra(G, p, u)
```

- Temps d'exécution
  - $O(SA \lg S)$
- Si on ne peut utiliser Dijkstra → Bellman-Ford
  - $O(S^2A) \rightarrow O(S^4)$ pour un graphe dense
- On peut faire mieux avec des algorithmes spécialisés
  - Algorithme de Floyd-Warshall
  - Algorithme de Johnson

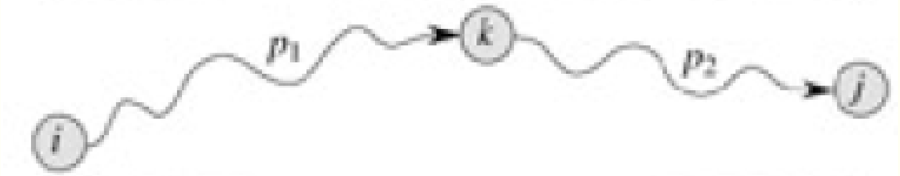
Algorithme de Floyd-Warshall

- Utilise une représentation par matrice d'adjacences
  - Sommets numérotés 1, 2, …, $|S|$
  - Matrice P de taille n x n qui représente les poids d'arc d'un graphe orienté $G = (S, A)$ à n sommets
- $$p_{ij} = \begin{cases} 0 & \text{si } i = j \\ \text{Le poids de l'arc } (i, j) & \text{si } i \neq j \text{ et } (i, j) \in A \\ \infty & \text{si } i \neq j \text{ et } (i, j) \in A \end{cases}$$

- Produit en sortie une matrice D de taille n x n
  - d_{ij}
    - Poids d'un plus court chemin reliant le sommet i au sommet j
- Et une matrice prédécesseur $\prod$
  - π_{ij}
    - Prédécesseur de j sur un plus court chemin issu de i
    - NIL si $i = j$ ou s'il n'existe aucun chemin de i vers j
  - i-ème ligne → arbre de plus courts chemins ayant pour racine i

Algorithme de Floyd-Warshall

- On considère les sommets intermédiaires d'un plus court chemin
  - Sous-ensemble $\{1, 2, \ldots, k\}$ de sommets pour un certain k
- Pour une paire quelconque de sommets (i, j)
  - On considère tous les chemins de i à j dont les sommets intermédiaires appartiennent à $\{1, 2, \ldots, k\}$
  - Et on retient celui dont le poids est minimal
- En fixant successivement k à $\{0, 1, 2, \ldots n\}$, on trouve les plus courts chemins



- 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 $\{1, 2, \ldots, k\}$
- Pour $k = 0$
  - Un chemin de i à j ne possède aucun sommet intermédiaire
  - Il est constitué d'au plus un arc → $d_{ij}^{(0)} = p_{ij}$
- Pour le cas général → définition récursive
  - $d_{ij}^{(k)} = \begin{cases} p_{ij} & \text{si } k = 0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{si } k \geq 1 \end{cases}$
- La matrice $D^{(n)} = (d_{ij}^{(n)})$ donne le résultat final

Algorithme de Floyd-Warshall

- Exemple 4 : Exécution de l'algorithme de Floyd-Warshall
- Exemple 5 : Procédure FLOYD-WARSHALL
  - Temps d'exécution

PROCHAIN COURS

LE PROBLÈME DU FLOT MAXIMUM