Plus courts chemins

option informatique

Généralités

Pondération

Comme nous l'avons vu dans le premier chapitre, étant donné un graphe $G = (V, E)$ (orienté ou non), une **pondération** de G est une application w de $V \times V$ dans $\mathbb{R}$ telle que :

$$\forall (v, v') \in (V \times V) \setminus E, \quad w(v, v') = \begin{cases} 0 & \text{si } v = v' \\ +\infty & \text{sinon} \end{cases}$$

$w(v, v')$ est le **poids de l'arc** reliant v à v'.

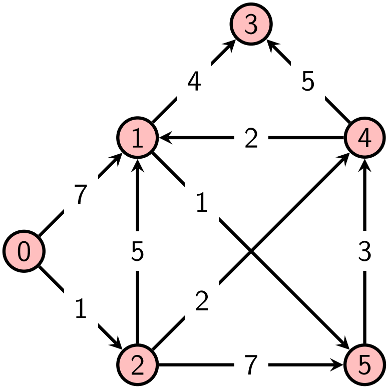
Pondération

- Dans un **graphe fini**, les sommets étant dénombrables, on peut toujours les numéroter de 0 à $n-1$ où n est l'ordre du graphe.
- Cette numérotation permet d'associer une matrice W aux poids d'un graphe.

$$W = \begin{pmatrix} w(v_0, v_0) & w(v_0, v_1) & \dots & w(v_0, v_{n-1}) \\ w(v_1, v_0) & w(v_1, v_1) & \dots & w(v_1, v_{n-1}) \\ \dots & \dots & \dots & \dots \\ w(v_{n-1}, v_0) & w(v_{n-1}, v_1) & \dots & w(v_{n-1}, v_{n-1}) \end{pmatrix}$$

Graphe pondéré

Un **graphe pondéré** est un graphe muni d'une pondération.



Le **poids d'un chemin** est la somme des poids des arêtes qui le composent.

Plus court chemin

Rechercher le **plus court chemin dans un graphe**, c'est trouver, s'il en existe, un chemin de poids minimal allant d'un sommet à un autre dans un graphe.

- Dans la suite, seuls les graphes munis de **poids positifs** sont abordés. La présence de poids négatifs, à l'origine de contraintes supplémentaires, ne sera abordée que dans les exercices.
- Nous parlerons indifféremment de **poids minimal**, de **distance minimale** ou de **longueur minimale**, noté d_{ij}, pour désigner le poids d'un plus court chemin **entre** deux sommets v_i et v_j d'un graphe.

Objectifs

$G = (V, E)$ étant un graphe pondéré, on cherche à déterminer :

- l'ensemble des d_{ij} pour tout couple $(v_i, v_j) \in V^2$;
- l'ensemble des d_{ij} pour un $v_i \in V$ donné, $v_j \in V$;
- un d_{ij} pour un $v_i \in V$ et un $v_j \in V$.

Deux algorithmes

Étant donné un couple $(v_i, v_j) \in V$, il n'existe pas d'algorithme qui calcule directement d_{ij}. Une vision globale du graphe est nécessaire et une visite de tous les sommets est indispensable.

- L'**algorithme de Floyd-Warshall** détermine l'ensemble des d_{ij} pour tout couple $(v_i, v_j) \in V \times V$. Son implémentation exploite la représentation des graphes par matrices d'adjacences.
- L'**algorithme de Dijkstra** détermine l'ensemble des d_{ij} pour un $v_i \in V$ donné, v_j étant n'importe quel sommet de V. Son implémentation exploite la représentation des graphes par listes d'adjacence.

Algorithme de Floyd-Warshall

Principe

- L'**algorithme de Floyd-Warshall** détermine l'ensemble des plus courts chemins entre deux sommets quelconques d'un graphe.
- Pour un graphe d'ordre n, cela représente n^2 plus courts chemins à déterminer.
- Une **matrice** est donc adaptée pour recueillir toutes ces distances.
- Cette matrice est construite de manière itérative en s'appuyant sur le **principe de sous-optimalité**.

Principe de sous-optimalité

Théorème 1

Si $s \overset{c}{\rightsquigarrow} t$ est un plus court chemin qui passe par u, alors $s \overset{c_1}{\rightsquigarrow} u$ et $u \overset{c_2}{\rightsquigarrow} t$ sont aussi des plus courts chemins.

Démonstration

La démonstration se fait par l'absurde. Si $G = (V, E)$ est un graphe pondéré de valuation définie par une fonction w, chaque arc $(v_i, v_j) \in E$ a un poids $w(v_i, v_j)$. Pour tout chemin $c = (x_0, x_1, \dots, x_k)$ dans G, pas abus de notation, le poids du chemin est :

$$w(c) = \sum_{i=0}^{k-1} w(x_i, x_{i+1})$$

Considérons un plus court chemin c du sommet s au sommet t passant par u :

$$s \overset{c_1}{\rightsquigarrow} u \overset{c2_2}{\rightsquigarrow} t$$

Les chemins c_1 et c_2 sont des plus courts-chemins et $w(c) = w(c_1) + w(c_2)$. Supposons qu'il existe un chemin c_1' plus court pour aller de s à u : $w(c_1') < w(c_1)$. Alors il existe un chemin $s \overset{c_1'}{\rightsquigarrow} u \overset{c_2}{\rightsquigarrow} t$ de s à t de poids :

$$w(c_1') + w(c_2) < w(c_1) + w(c_2) = w(c)$$

Ce qui est absurde puisque, par hypothèse, $s \overset{c_1}{\rightsquigarrow} u \overset{c_2}{\rightsquigarrow} t$ est un plus court chemin.
La même analyse vaut pour c_2.

Principe de sous-optimalité

- L'optimalité de la solution du problème du calcul de plus court chemin passe donc par l'optimalité des solutions des sous-problèmes de calcul de plus courts chemins.
- Dit autrement, déterminer un plus court chemin entre deux sommets s et t fournit des plus courts chemins entre s et tous les sommets situés sur le chemin aboutissant en t.
- De tels problèmes peuvent être résolus par des méthodes dites de **programmation dynamique**.

Matrices de Floyd-Warshall

- Soit un graphe pondéré $G = (V, E)$ d'ordre n dont les sommets sont $v_0, v_1, \ldots, v_{n-1}$.
- Pour tout entier $k \in [\![0, n]\!]$, notons $M^{(k)}$ la matrice telle que :

$$M_{ij}^{(k)} = \begin{cases} \textbf{poids du plus court chemin} \\ \textbf{entre deux sommets } v_i \textbf{ et } v_j \\ \textbf{ne passant que par des sommets } v_p \textbf{ où } p \leqslant k-1. \end{cases}$$

Matrices de Floyd-Warshall

- Puisqu'il n'existe pas de sommet d'indice strictement négatif, $M^{(0)}$ est la matrice des poids des plus courts chemins ne passant par aucun sommet. Par construction, c'est la **matrice des poids du graphe G**.

$$\forall (i,j) \in [\![0, n-1]\!]^2 \quad M_{i,j}^{(0)} = w(v_i, v_j)$$

- La matrice $M^{(n)}$ **contient** les poids des plus courts chemins reliant deux sommets quelconques du graphe. C'est la matrice recherchée des poids d_{ij} pour $(i,j) \in [\![0, n-1]\!]^2$.

Matrices de Floyd-Warshall

Le **principe de sous-optimalité** fournit un moyen de construire les matrices $M^{(1)}, M^{(2)}, \ldots, M^{(n)}$.

Pour $k \geqslant 0$, la détermination de $M_{ij}^{(k+1)}$ ne fait intervenir que les sommets $v_0, v_1, \ldots, v_k$ pour calculer le poids du plus court chemin entre v_i et v_j.

- Si le chemin ne passe pas par le sommet v_k, alors :

$$M_{ij}^{(k+1)} = M_{ij}^{(k)}$$

- Si le chemin passe par le sommet v_k, alors :

$$M_{ij}^{(k+1)} = M_{ik}^{(k)} + M_{kj}^{(k)}$$

Matrices de Floyd-Warshall

Pour un graphe $G = (V, E)$ d'ordre n dont les sommets sont $v_0, v_1, \dots, v_{n-1}$, l'**algorithme de Floyd-Warshall** construit la suite des matrices $M^{(k)}$ en exploitant la relation de récurrence suivante.

$$\forall (i, j, k) \in [\![0, n-1]\!]^3 \quad M_{ij}^{(k+1)} = \min\left(M_{i,j}^{(k)}, M_{ik}^{(k)} + M_{kj}^{(k)}\right)$$

sachant :

$$\forall (i, j) \in [\![0, n-1]\!]^2 \quad M_{i,j}^{(0)} = w(v_i, v_j)$$

Preuve

Théorème 2 (algorithme de Floyd-Warshall)

Si G ne contient pas de cycle de poids strictement négatif, alors pour tout entier $k \in [\![0, n]\!]$, $M_{i,j}^{(k)}$ est le poids du plus court chemin reliant v_i à v_j passant par les seuls sommets $v_0, \dots, v_{k-1}$.

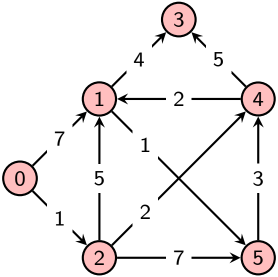
Démonstration

Si $k = 0$, alors $M_{i,j}^{(0)} = w(v_i, v_j)$ est le poids du chemin minimal reliant v_i à v_j sans passer par aucun autre sommet.
Supposons le résultat établi pour un certain rang $k \geqslant 1$ et considérons un chemin de poids minimal $v_i \rightsquigarrow v_j$ ne passant que par les sommets $v_0, \dots, v_k$.

- S'il ne passe pas par v_k, par hypothèse de récurrence, son poids est $M_{i,j}^{(k)}$.
- S'il passe par v_k, d'après le principe de sous-optimalité, les chemins $v_i \rightsquigarrow v_k$ et $v_k \rightsquigarrow v_j$ sont minimaux et ne passent que par $v_0, \dots, v_{k-1}$. Par hypothèse de récurrence, son poids est $M_{ik}^{(k)} + M_{kj}^{(k)}$.

Alors $M_{ij}^{(k+1)} = \min\left(M_{ij}^{(k)}, M_{ik}^{(k)} + M_{kj}^{(k)}\right)$ est le poids minimal d'un plus court chemin reliant v_i à v_j et ne passant que par des sommets de la liste $v_0, \dots, v_k$.

Exemple



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

Implémentation / Complexité

`Code sur machine`

Algorithme de Dijkstra

Principe

- L'**algorithme de Dijsktra** détermine l'ensemble des d_{ij} pour un $v_i \in V$ donné, $v_j \in V$;
- Traditionnellement, si s un sommet particulier d'un graphe G pondéré, pour sommet t de G, on note $\delta(s,t)$ **le plus court chemin de s à t**.
- Si toutes les pondérations sont positives, l'**algorithme de Dijkstra** généralise le parcours en profondeur pour trouver les plus courts chemins.

Avant de présenter cet algorithmes, donnons quelques résultats utiles.

Inégalité triangulaire

Théorème 3

Soit s, t, u trois sommets d'un graphe pondéré. Alors :

$$\delta(s,t) \leqslant \delta(s,u) + w(u,t)$$

Démonstration

S'il existe un chemin de s à u et un arc de u à t, $\delta(s,u)$ et $w(u,t)$ sont des quantités finies. On obtient un chemin de s à t de poids $\delta(s,u) + w(u,t)$ en prenant un plus court chemin de s à u et l'arc (u,t), donc :

$$\delta(s,t) \leqslant \delta(s,u) + w(u,t)$$

Sinon, on a $\delta(s,u) + w(u,t) = +\infty$, et l'inégalité reste valable.

Existence d'un plus court chemin

Théorème 4

Soit t un sommet accessible depuis s. Alors il existe un chemin de poids $\delta(s,t)$ entre s et t composé de sommets tous distincts.

Démonstration

Considérons un chemin c entre s et t, de poids $\delta(s,t)$, et de longueur minimale parmi les chemins de poids $\delta(s,t)$ reliant s à t. S'il existait deux sommets égaux sur le chemin, on obtiendrait un circuit. Comme il n'y a pas de circuit de poids strictement négatif dans le graphe par hypothèse, ce circuit est de poids nul, sinon on pourrait le supprimer pour obtenir un chemin de s à t de poids strictement inférieur à $\delta(s,t)$, ce qui est exclu. Mais le supprimer mène alors à un chemin de même poids mais avec strictement moins d'arcs, ce qui est exclu également. Donc c est composé de sommets distincts.

Relâchement d'arcs

- Soit G un graphe pondéré et s un **sommet fixé** de G.
- Pour tout sommet t de G, désignons par $d_s[t]$ le **tableau des estimations des distances** $\delta(s,t)$, c'est-à-dire les valeurs telles que :

$$\forall t \in G \quad \delta(s,t) \leqslant d_s[t]$$

- **Relâcher l'arc** (u,v), c'est réaliser l'affectation :

$$d_s[v] \leftarrow \min(d_s[v], d_s[u] + w(u,v))$$

Relâchement d'arcs

Théorème 5

Après relâchement de l'arc (u, v)*, on a toujours :*

$$\delta(s, v) \leqslant d_s[v]$$

Démonstration

Par hypothèse, avant relâchement, on a $\delta(s, u) \leqslant d_s[u]$. Ainsi, par inégalité triangulaire :

$$\delta(s, v) \leqslant d_s[u] + w(u, v)$$

Preuve

Théorème 6 (algorithme de Dijkstra)

Soit t un sommet accessible depuis s et $c = (s_0 = s, s_1, \dots, s_k = t)$ un chemin de poids $\delta(s,t)$ entre s et t. Soit un tableau $(d_s[u])_{u \in G}$ vérifiant $d_s[s] = 0$ et pour tout u de G, $\delta(s,u) \leqslant d_s[u]$.
Après relâchements successifs des arcs $(s_0, s_1), \dots, (s_{k-1}, s_k)$ dans cet ordre, le tableau $d_s[t]$ contient $\delta(s,t)$.

Démonstration

Le théorème 1 montre que pour tout entier i de $[\![0,k]\!]$, $(s_0, s_1, \dots, s_i)$ est un plus court chemin de s à s_i. Montrons par récurrence sur i qu'après relâchement de l'arc (s_{i-1}, s_i), $d_s[s_i]$ contient $\delta(s, s_i)$.

- Si $i = 0$, alors $d_s[s]$ vaut 0, qui est bien $\delta(s,s)$.
- Soit $i > 1$ et supposons la propriété démontrée au rang $i-1$. Alors juste avant le relâchement de (s_{i-1}, s_i), $d_s[s_{i-1}]$ contient $\delta(s, s_{i-1})$. Comme $\delta(s, s_i) = \delta(s, s_{i-1}) + w(s_{i-1}, s_i)$, on a $d[s_i] \leqslant \delta(s, s_i)$ après relâchement de l'arc $(s_{i-1}, s_i\})$. Or le théorème 5 prouve que $d_s[s_i]$ était supérieur ou égal à $\delta(s, s_i)$ avant relâchement. Donc la propriété est vraie au rang i.
- Par principe de récurrence, elle est vraie pour tout $i \in [\![0,k]\!]$, et en particulier $d_s[t]$ contient $\delta(s,t)$ à la fin du processus.

Algorithme

L'**algorithme de Dijsktra** construit un tableau d de longueur n dont l'élément $d[i]$ contient, à la fin de l'algorithme, le poids $\delta(s,i)$ du plus court chemin entre s et i.

- `Initialiser` d `avec les poids initiaux :` $\forall v \in V, d[v] \leftarrow w(s,v)$
- `Initialiser la liste des sommets déjà visités` S `avec` $\{s\}$.
- `Définir la liste complémentaire` $\overline{S}$ `des sommets non encore visités. Initialement,` $\overline{S} = V \setminus \{s\}$.
- `Tant que` $\overline{S}$ `n'est pas vide, sélectionner un sommet` u `qui respecte la condition :`

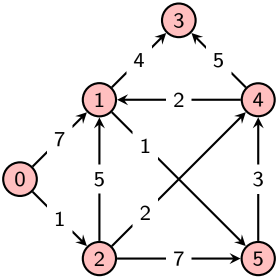
  $$d[u] = \min(d[v] \mid v \in \overline{S})$$

  `Supprimer` u `de` $\overline{S}$.
- `Ajouter` u `à` S `:` $S \leftarrow S \cup \{u\}$.
- `Relâcher les arcs issus de` u.

$$\forall v \in \overline{S} \qquad d[v] \leftarrow \min(d[v], d[u] + w(u,v))$$

Exemple

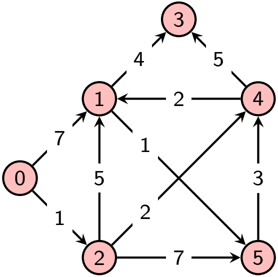
Recherche des plus courts chemins à partir du sommet 0.



S	0	1	2	3	4	5

Exemple

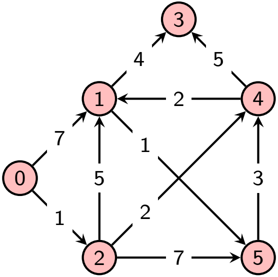
Recherche des plus courts chemins à partir du sommet 0.



S	0	1	2	3	4	5
$\{0\}$	.	7	1	∞	∞	∞

Exemple

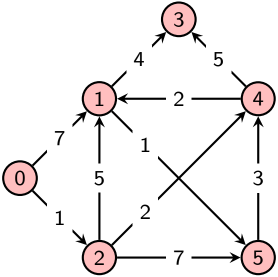
Recherche des plus courts chemins à partir du sommet 0.



S	0	1	2	3	4	5
$\{0\}$	.	7	1	∞	∞	∞
$\{0, 2\}$	.	6	.	∞	3	8

Exemple

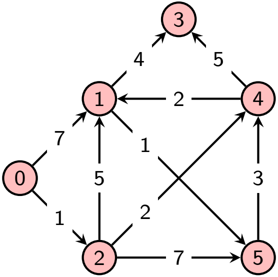
Recherche des plus courts chemins à partir du sommet 0.



S	0	1	2	3	4	5
$\{0\}$	.	7	1	∞	∞	∞
$\{0, 2\}$	.	6	.	∞	3	8
$\{0, 2, 4\}$	.	5	.	8	.	8

Exemple

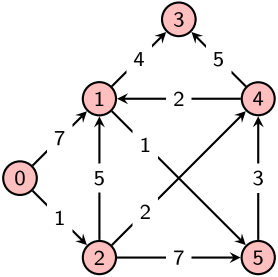
Recherche des plus courts chemins à partir du sommet 0.



S	0	1	2	3	4	5
$\{0\}$	.	7	1	∞	∞	∞
$\{0, 2\}$	.	6	.	∞	3	8
$\{0, 2, 4\}$	.	5	.	8	.	8
$\{0, 2, 4, 1\}$	.	.	.	8	.	6

Exemple

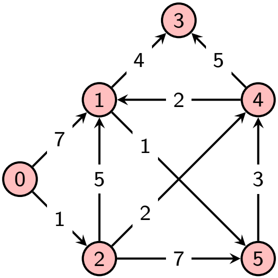
Recherche des plus courts chemins à partir du sommet 0.



S	0	1	2	3	4	5
$\{0\}$	.	7	1	∞	∞	∞
$\{0, 2\}$	.	6	.	∞	3	8
$\{0, 2, 4\}$	.	5	.	8	.	8
$\{0, 2, 4, 1\}$	.	.	.	8	.	6
$\{0, 2, 4, 1, 5\}$	.	.	.	8	.	.

Exemple

Recherche des plus courts chemins à partir du sommet 0.



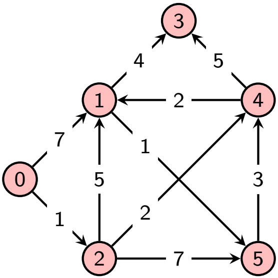
S	0	1	2	3	4	5
$\{0\}$	.	7	1	∞	∞	∞
$\{0, 2\}$	.	6	.	∞	3	8
$\{0, 2, 4\}$	.	5	.	8	.	8
$\{0, 2, 4, 1\}$	.	.	.	8	.	6
$\{0, 2, 4, 1, 5\}$	.	.	.	8	.	.
$\{0, 2, 4, 1, 5, 3\}$	.	.	.	.	.	.

Exemple

Poids de plus courts chemins du sommet 0 vers les autres sommets.

0 vers 1 → 5   0 vers 2 → 1   0 vers 3 → 8

0 vers 4 → 3   0 vers 5 → 6



Implémentation / Complexité

`Code sur machine`