Chemins les plus courts entre toutes les paires de sommets.

On a un graphe simple avec une fonction de poids sur les arcs ou arêtes. On veut pour toute paire de sommets s, s' trouver le chemin (la chaîne) la plus courte (i.e., ayant la somme de poids la plus petite possible) de s à s'. S'il n'y a pas de tel chemin (chaîne), cette longueur est posée $= \infty$.

On va travailler par récurrence sur le nombre d'arcs/arêtes dans un tel chemin/chaîne (sa longueur au sens non pondéré).

Soit $S = \{s_1, \dots, s_n\}$. Pour $i, j \in \{1, \dots, n\}$, on pose

L_{ij} = longueur minimum d'un chemin de s_i à s_j, s'il existe

$$\min \sum_{k=1}^{r} p(a_k), \quad s_i a_1 \dots a_r s_j \text{ chemin de } s_i \text{ à } s_j \quad (a_1, \dots, a_r \in A)$$

$= \infty$ s'il n'y a pas de tel chemin

On place en chaque sommet une boucle de poids 0. Ainsi tout chemin de s_i à s_j

ayant m ~~arêtes~~ peut s'étendre en un chemin à $m+1, m+2, \ldots$ arcs, de même longueur : il suffit d'ajouter à la fin $1, 2, \ldots$ fois la boucle (s_j, s_j) de poids 0.

Pour $i, j \in \{1, \ldots, n\}$ et $m \in \mathbb{N}$ on pose

$L_{ij}^{(m)}$ = longueur minimum d'un chemin de s_i à s_j possédant m arcs, s'il existe

$= \infty$ sinon.

Supposons $p(i,j) \geq 0$ pour $i, j \in \{1, \ldots, n\}$.

On a alors

$L_{ii}^{(0)} = 0$, $L_{ij}^{(0)} = \infty$ pour $i \neq j$

$L_{ij}^{(1)} = p(i,j)$ si $(i,j) \in A$, $= \infty$ si $(i,j) \notin A$

Pour $m \leq m'$, $L_{ij}^{(m)} \geq L_{ij}^{(m')}$ (le chemin le plus court à m arcs peut se prolonger par $m'-m$ boucles en s_j, la longueur reste la même)

Pour $m \geq n-1$, $L_{ij}^{(m)} = L_{ij}$ (le chemin le plus court ne nécessite pas plus de $n-1$ arcs, sinon il y aurait un circuit ou une boucle, qu'on peut enlever).

Un chemin le plus court à ~~m + 1~~ $m+m'$ arcs entre s_i et s_j se décompose en la concaténation d'un chemin le plus court à m arcs de s_i à un sommet s_k $(k=1,\dots,n)$, et d'un chemin le plus court à m' arcs de s_k à s_j. Donc

$$L_{ij}^{(m+m')} = \min_{k=1}^{n} \left(L_{ik}^{(m)} + L_{kj}^{(m')} \right)$$

Comparer avec la formule pour le produit de deux matrices A et B $n \times n$

$$(A \cdot B)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

On définit donc un autre type de produit matriciel pour A $n_1 \times n$ et B $n \times n_2$

$$(A * B)_{ij} = \min_{k=1}^{n} (A_{ik} + B_{kj}) \qquad \begin{array}{l} i=1,\dots,n_1 \\ j=1,\dots,n_2 \end{array}$$

On a ainsi $L^{(m+m')} = L^{(m)} * L^{(m')}$, où $L^{(m)}$ est la matrice des distances $L_{ij}^{(m)}$

Plutôt que de calculer $L^{(1)}, L^{(2)}, L^{(3)}, \dots$, on va calculer $L^{(1)}, L^{(2)}, L^{(4)}, L^{(8)}, \dots$

Initialisation

$L_{ii}^{(1)} = 0$ $\quad i \neq j$: $L_{ij}^{(1)} = p(i,j)$ si $(i,j) \in A$

$= \infty$ si $(i,j) \notin A$

Pour $t = 1, \dots, \lceil \log_2(n-1) \rceil$ faire $L^{(2^t)} = L^{(2^{t-1})} * L^{(2^{t-1})}$.

En détail :

Pour $i := 1, \dots, n$ faire

Pour $j := 1, \dots, n$ faire

$x := \infty$

Pour $k := 1, \dots, n$ faire

$x := \min\left(x, L^{(2^{t-1})}_{ik} + L^{(2^{t-1})}_{kj}\right)$

$L^{(2^t)}_{ij} := x$

Algorithme de Floyd-Warshall

Plutôt que ~~l'induction~~ sur le nombre d'arêtes, on fera l'induction sur le nombre de sommets intermédiaires.

$D^{(k)}_{ij}$ = longueur du plus court chemin de s_i à s_j dont tous les sommets intermédiaires sont parmi $\{s_1, \dots, s_k\}$, s'il existe $(0 \le k \le n)$

$= \infty$ s'il n'existe pas.

$k > 0$: Pour un tel chemin, on a deux possibilités :

a) les sommets intermédiaires sont tous parmi $\{s_1, \dots, s_{k-1}\}$;

b) le chemin passe par s_k, une seule fois, car il n'y a pas de circuit ; il est la concaténation de deux chemins, l'un de s_i à s_k, l'autre de s_k à s_j, tous deux les plus courts possibles avec sommets intermédiaires parmi $\{s_1, \dots, s_{k-1}\}$.

Par conséquent :

$$D_{ij}^{(k)} = \min\left(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\right)$$

Pour $k=0$, un chemin avec sommets intermédiaires dans $\{s_1, \dots, s_k\} = \emptyset$, est un arc. Donc

$$D_{ij}^{(0)} = \begin{cases} 0 & i=j \\ p(i,j) & i \neq j, \ (i,j) \in A \\ \infty & i \neq j, \ (i,j) \notin A \end{cases}$$

On a finalement $D_{ij}^{(n)} = L_{ij}$

Algo

Initialisation

Pour $i := 1$ à n faire Bs.

$D_{ii} := 0$
pour $j \neq i$ faire $D_{ij} = \begin{cases} p(i,j) & \text{si } (i,j) \in A \\ \infty & \text{si } (i,j) \notin A \end{cases}$

Récursion

Pour $k := 1$ à n faire
pour $i := 1$ à n faire
pour $j := 1$ à n faire

$$D_{ij}^{(k)} = \min\left(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)} \right)$$

Ces algorithmes donnent la distance, mais pas le chemin. Pour cela, il faut définir la fonction "prédécesseur"

pred : ~~S×S~~ $\{1, \dots, n\} \times \{1, \dots, n\} \to \{1, \dots, n\}$
pred$(i,j) = k$ si s_k est le prédécesseur de s_j dans le plus court chemin de s_i à s_j

On ~~fait~~ suit le chemin à rebours par l'itération

$k := j$
tant que $k \neq i$ faire
mettre s_k en tête de liste
$k :=$ pred(i, k)
mettre s_i en tête de liste.

Calcul du prédécesseur

Méthode avec $L_{ij}^{(m)}$

$$pred^{(1)}(i,j) = i \quad si\ (i,j) \in A$$
$$= NIL \quad si\ (i,j) \notin A$$

$$pred^{(m+m')}(i,j) = pred^{(m')}(k,j) \text{ pour}$$

le k tel que $L_{ik}^{(m)} + L_{kj}^{(m')}$ est le plus petit

Méthode avec $D_{ij}^{(k)}$

$$pred^{(0)}(i,j) := i \quad si\ (i,j) \in A \quad i \neq j$$
$$:= NIL \quad sinon$$

$$pred^{(k)}(i,j) := pred^{(k-1)}(i,j) \quad si\ D_{ij}^{(k-1)} \leq D_{ik}^{(k-1)} + D_{kj}^{(k-1)}$$
$$:= pred^{(k-1)}(k,j) \quad si\ D_{ij}^{(k-1)} > D_{ik}^{(k-1)} + D_{kj}^{(k-1)}$$

Cas des arêtes à poids négatif

Tester s'il y a un cycle de longueur <0 par l'algo de Bellman-Ford.

si oui : échec

si non : modifier les poids de façon à ramener à des poids ≥ 0.

$h : S \to \mathbb{R}$

$w'(i,j) = w(i,j) + h(i) - h(j)$

Cela modifie les longueurs de chemin comme suit : C chemin de s_i à s_j

$$L'(C) = L(C) + h(i) - h(j)$$

Donc la minimisation reste la même

si on a des poids ≥ 0 et peu d'arêtes, on peut également appliquer l'algorithme de Dijkstra sur chaque s_i, $i = 1, \ldots, n$.

Autre application

Fermeture transitive d'une relation
poids tous $= 1$ $\quad p(i,j) = 1 \quad (i,j) \in A$
Remplacer + par un + booléen ("ou").