

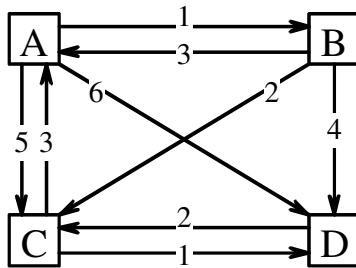
Algorithme du plus court chemin de Floyd-Warshall

Description du problème

On a un graphe pondéré, de N sommets, dont les arêtes sont les couts (distances) qui permettent d'aller d'un sommet S_1 à un autre sommet S_2 . Ce cout peut être noté : $C_{S_1 \rightarrow S_2}$. On peut avoir un graphe non orienté : $C_{S_1 \rightarrow S_2} = C_{S_2 \rightarrow S_1}$ ou non orienté : $C_{S_1 \rightarrow S_2} \neq C_{S_2 \rightarrow S_1}$

Ce graphe peut par exemple être décrit par un dictionnaire de dictionnaires dont les clés (S_1) sont les sommets du graphe et les valeurs, des dictionnaires dont les clés (S_2) sont les sommets voisins du sommet S_1 et les valeurs le cout pour aller du sommet S_1 au sommet S_2 : $C_{S_1 \rightarrow S_2}$.

Exemple on a le graphe ci-dessous :



Ce graphe peut alors est décrit par le dictionnaire :

Graphe = { 'A': { 'B': 1, 'C': 5, 'D': 6 },
 ' B': { 'A': 3, 'C': 2, 'D': 4 },
 ' C': { 'A': 3, 'D': 1 },
 ' D': { 'C': 2 } }

Objectif du problème

L'objectif est de déterminer le plus faible coût (la plus petite distance) permettant de passer d'un sommet de départ S_D à un sommet d'arrivée S_A . Soit en passant directement du sommet S_D au sommet S_A ; Soit en passant par un répertoire R_k de n_k sommets intermédiaires, cette liste pouvant être tous les sommets restant (sans S_D et S_A) ou une partie uniquement des sommets du graphe.

Il s'agit également de déterminer la liste L_i de ces sommets intermédiaires menant du sommet S_D au sommet S_A avec le plus faible cout (la plus petite distance).

En conclusion on peut dire qu'il s'agit de déterminer :

Le plus court chemin, et sa longueur, du sommet S_{Dep} au sommet S_{Arr} .

Remarque :

Si certains couts sont négatifs et que ces couts négatifs forment de cycles, alors il n'y a pas de solutions car en parcourant ces cycles à l'infini on peut arriver à une longueur du chemin égale à $-\infty$.

Pour éviter cela nous interdirons de passer 2 fois par un même sommet. Donc la longueur n_i de la liste L_i des sommets permettant d'aller d'un sommet S_{Dep} à un sommet S_{Arr} est limitée à $N-2$: $n_i \leq N-2$.

Algorithme naïf

Une solution peut-être de calculer le cout de tous les chemins possibles pour aller du sommet S_{Dep} au sommet S_{Arr} . Or le nombre de chemins reliant deux sommets donnés d'un graphe complet à N sommets est de : $\sum_{k=0}^{N-2} \frac{(N-2)!}{(N-2-k)!}$ Ce qui implique une complexité trop importante pour résoudre le problème avec un grand nombre de sommets.

Cette complexité étant liée à la taille des différents répertoires R de sommets intermédiaires qui peuvent être utilisés pour aller de S_{Dep} à S_{Arr} . Et donc à la taille N du graphe puisque que les répertoires intermédiaires R peuvent avoir une taille pouvant aller jusqu'à $N-2$ sommets.

Algorithme de programmation dynamique

L'idée est de se dire que pour aller d'un sommet S_{Dep} à un sommet S_{Arr} en passant par les sommets d'un répertoire R de n_R sommets, on va rechercher les plus courts chemins allant des N sommets S_{Dep} aux N sommets S_i du graphe en passant par les sommets d'un répertoire $(R-S_i)$ de n_{R-1} sommets.

Puis on retiendra à chaque fois la longueur minimale de ces chemins + la longueur minimale permettant d'aller des sommets S_i au sommet S_{Arr} . On commence avec un répertoire R vide puis on ajoute un à un au répertoire R les N sommets intermédiaires S_i du graphe. En calculant à chaque fois toutes les longueurs minimales des chemins entre les différents sommets du graphe en passant par les sommets du répertoire R des sommets intermédiaires.

On a N fois N^2 longueurs minimales à calculer (car on doit faire tous les couples (S_{Dep}, S_{Arr}) du graphe). La complexité d'un tel algorithme est en N^3 au lieu d'être de l'ordre de $N!$.

Donc si on note $(L_{min})_{S_{Dep} \rightarrow S_{Arr}}^R$ la longueur minimale du sommet S_{Dep} au sommet S_{Arr} en passant par les sommets du répertoire R des sommets intermédiaires, alors on a :

$$\forall \text{ les sommets } S_{Dep}, S_{Arr} \text{ et } S_i \text{ du graphe : } (L_{min})_{S_{Dep} \rightarrow S_{Arr}}^{R+S_i} = \min \left((L_{min})_{S_{Dep} \rightarrow S_{Arr}}^R, (L_{min})_{S_{Dep} \rightarrow S_i}^R + (L_{min})_{S_i \rightarrow S_{Arr}}^R \right)$$

Pour obtenir la longueur minimale du sommet S_{Dep} au sommet S_{Arr} il suffit alors de retenir la longueur $(L_{min})_{S_{Dep} \rightarrow S_{Arr}}^{R_T}$ où R_T est un répertoire incluant tous les sommets du graphe.

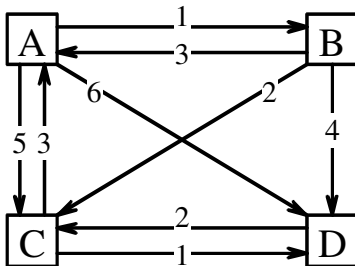
Pour obtenir le chemin il faut en plus de calculer la longueur minimale $(L_{min})_{S_{Dep} \rightarrow S_{Arr}}^{R+S_i}$ retenir le sommet suivant le sommet S_a (noté $(S_{suiv})_{S_{Dep} \rightarrow S_{Arr}}^{R+S_i}$) pour obtenir le chemin de longueur minimale.

$$\text{Si : } (L_{min})_{S_{Dep} \rightarrow S_{Arr}}^R \leq (L_{min})_{S_{Dep} \rightarrow S_i}^R + (L_{min})_{S_i \rightarrow S_{Arr}}^R \text{ alors : } (S_{suiv})_{S_{Dep} \rightarrow S_{Arr}}^{R+S_i} = (S_{suiv})_{S_{Dep} \rightarrow S_{Arr}}^R$$

$$\text{Sinon : } (L_{min})_{S_{Dep} \rightarrow S_i}^R + (L_{min})_{S_i \rightarrow S_{Arr}}^R < (L_{min})_{S_{Dep} \rightarrow S_{Arr}}^R \text{ alors : } (S_{suiv})_{S_{Dep} \rightarrow S_{Arr}}^{R+S_i} = (S_{suiv})_{S_{Dep} \rightarrow S_i}^R$$

Pour obtenir le chemin minimal permettant d'obtenir la longueur minimale du sommet S_{Dep} au sommet S_{Arr} il suffit alors de faire la liste des sommets suivants jusqu'à arriver au sommet S_{Arr} .

Exemple on a le graphe ci-dessous :



$R = \emptyset$	A	B	C	D
A	0 A	1 B	5 C	6 D
B	3 A	0 B	2 C	4 D
C	3 A	∞ B	0 C	1 D
D	∞ A	∞ B	2 C	0 D

(A)	A	B	C	D
A	0 A	1 B	5 C	6 D
B	3 A	0 B	2 C	4 D
C	3 A	4 A	0 C	1 D
D	∞ A	∞ B	2 C	0 D

(A,B)	A	B	C	D
A	0 A	1 B	3 B	5 B
B	3 A	0 B	2 C	4 D
C	3 A	4 A	0 C	1 D
D	∞ A	∞ B	2 C	0 D

(A,B,C)	A	B	C	D
A	0 A	1 B	3 B	4 B
B	3 A	0 B	2 C	3 C
C	3 A	4 A	0 C	1 D
D	5 C	6 C	2 C	0 D

(A,B,C,D)	A	B	C	D
A	0 A	1 B	3 B	4 B
B	3 A	0 B	2 C	3 C
C	3 A	4 A	0 C	1 D
D	5 C	6 C	2 C	0 D

Exemples de résultats :

Longueur minimale et chemin du sommet A au sommet D : **4** [A, B, C, D]

Longueur minimale et chemin du sommet C au sommet B : **4** [C, A, B]