Algorithmique

Les graphes : représentations → implémentations

On montre ici les différentes manières de représenter un graphe, et les implémentations (représentations machines) qui en découlent.

Les sommets doivent obligatoirement être numérotés (ici de 1 à n, l'ordre du graphe, mais ce peut être de 0 à $n-1$). On peut donc ajouter à chacune des représentations données ici une table qui associe un sommet (son nom, ce qu'il représente) à son numéro.

La définition (rappel)

On définit un graphe $G = < S, A >$ en donnant les deux ensembles : S l'ensemble de **sommets** et A l'ensemble de **liaisons**. Si le graphe est **valué** ($G = < S, A, C >$), on ajoute la fonction de coût ($C : A \mapsto \mathbb{R}$).

Soit le graphe non orienté $G_1 = < S_1, A_1 >$ avec
- $S_1 = \{1, 2, 3, 4\}$
- $A_1 = \{\{1,2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

Soit le graphe orienté $G_2 = < S_2, A_2 >$ avec
- $S_2 = \{1, 2, 3, 4\}$
- $A_2 = \{(1,2), (1, 3), (3, 2), (4, 2), (3, 4)\}$

Représentation saggitale

Un graphe peut également être défini graphiquement par une de ses représentations **saggitales** (il y en a plusieurs...) :

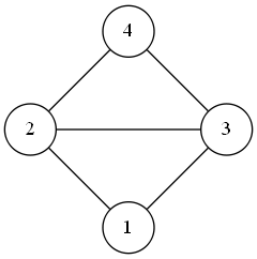


FIGURE 1 – Graphe non orienté G_1

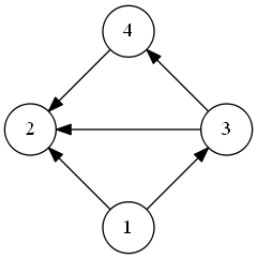


FIGURE 2 – Graphe orienté G_2

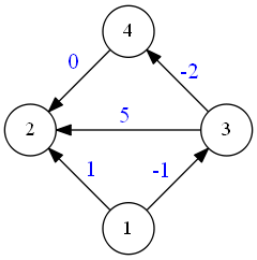


FIGURE 3 – Graphe G'_2 valué

Matrice d'adjacence

Un graphe peut être défini par sa **matrice d'adjacence** : chaque case x, y de la matrice, indique s'il y a une liaison de x vers y (V = vrai, F = faux).

G_1

	1	2	3	4
1	F	V	V	F
2	V	F	V	V
3	V	V	F	V
4	F	V	V	F

G_2

	1	2	3	4
1	F	V	V	F
2	F	F	F	F
3	F	V	F	V
4	F	V	F	F

C'_2

	1	2	3	4
1		1	-1	
2				
3		5		-2
4		0		

Le graphe G_1 est non orienté ⇒ sa matrice d'adjacence est symétrique.

Lorsque le graphe est valué, comme G'_2, on peut lui associer une *matrice de coûts*, ici C'_2.

Si le graphe admet des liaisons multiples, la matrice d'adjacence contiendra des entiers : chaque case x, y de la matrice indiquera le nombre de liaisons de x vers y (0 pour pas de liaison).

⇒ implémentation statique

La première implémentation des graphes consiste tout simplement à représenter la matrice d'adjacence. Ce sera une matrice de booléens, dans le cas d'un graphe simple ou d'un 1-graphe. Une matrice d'entiers dans le cas d'un multigraphe ou d'un p-graphe.

La matrice de coûts (une matrice de réels) peut remplacer la matrice d'adjacence. Dans ce cas il faut trouver une valeur représentant l'absence de liaison (par exemple $+\infty$ lorsque l'on cherche des plus courts chemins).

Utilisation

La matrice permet un accès direct aux liaisons, que l'on peut donc modifier en temps constant.

Pour parcourir la liste des successeurs (ou voisins) d'un sommet x, il suffit de parcourir la ligne x (pour les prédécesseurs, la colonne x). Par contre, il faut parcourir la ligne (ou la colonne) en entier dans tous les cas.

Autre inconvénient : L'implémentation par matrice d'adjacence occupera toujours une place de l'ordre de n^2 quelque soit le nombre de liaisons. Elle n'est donc intéressante que pour des graphes très denses.

Listes d'adjacence

On peut également définir un graphe en donnant pour chaque sommet la liste de ses successeurs [1] (ou voisins dans le cas non orienté) : une **liste d'adjacence**.

$G_1 =$ 1 : {2, 3}
2 : {1, 3, 4} [2]
3 : {1, 2, 4}
4 : {2, 3}

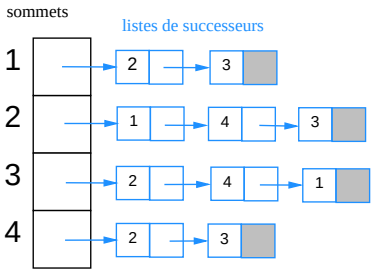
$G_2 =$ 1 : {2, 3}
2 : {}
3 : {2, 4}
4 : {2}

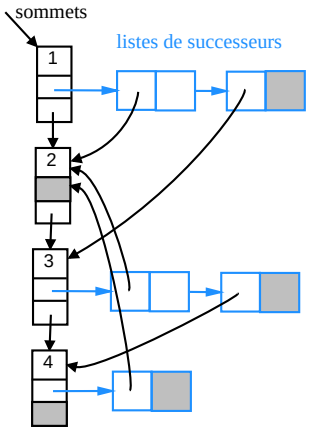
⇒ implémentations par listes d'adjacence

Un graphe est défini par une liste : celle de ses sommets. Pour chaque sommet, une liste représentant les successeurs (ou voisins).

Les listes d'adjacence sont "dynamiques" : chaque liste a une longueur différente (le nombre de successeurs).

Lorsque la liste des sommets est statique [3] (comme ci-dessous la représentation de G_1), les listes d'adjacences contiennent les numéros des sommets.





Lorsque la liste des sommets est dynamique (comme ci-dessus la représentation de G_2), chaque élément de la liste d'adjacence doit "pointer" vers le sommet adjacent.

La place occupée est donc minimale = le nombre de sommets + le nombre de liaisons. Dans le cas d'un graphe non orienté, chaque arête x — y est présente deux fois : y est successeur de x et x est successeur de y.

Si le graphe est valué, chaque élément de la liste d'adjacence contient en plus le coût de la liaison qu'il représente.

Utilisation

Si la liste des sommets est dynamique, il faudra effectuer une recherche linéaire pour atteindre un sommet.

Une fois sur un sommet, le parcours de ses successeurs consiste simplement à parcourir la liste d'adjacence. Par contre, obtenir les prédécesseurs d'un sommet nécessitera un parcours complet du graphe ! La solution est dans ce cas d'ajouter les listes de prédécesseurs en chaque sommet.

Ajouter une liaison n'est pas très couteux. Par contre, pour pouvoir rechercher et éventuellement supprimer un arc (x, y), il faudra parcourir au pire toute la liste des successeurs de x pour y trouver y.

Comparaison des deux implémentations

Voici les ordres de grandeur (au pire) de quelques opérations pour un graphe d'ordre n avec m liaisons :

	matrice d'adjacence	listes d'adjacence
existence arc (x, y)	1	$d^{\circ+}(x)$
calcul $d^{\circ+}(x)$ parcours successeurs de x	n	$d^{\circ+}(x)$
parcours complet du graphe	n^2	$n + m$

1. Dans certains cas, ce sont les listes des prédécesseurs qui sont données (par exemple pour les graphes de dépendances).
2. Dans la représentation par listes d'adjacence d'un graphe non orienté, on peut trouver les arêtes en double (comme ici), ou seulement "dans un sens".
3. Ou permet l'accès direct par indices.