Chapitre
Implémentations des graphes

Nous présentons ici les deux implémentations générales des graphes. La première est dite selon une matrice d'adjacence l'autre selon un tableau de listes.
Ces deux méthodes cohabitent pour une raison : aucune n'est meilleure que l'autre.
Pour être plus précis, leurs compromis espace-temps sont différents :

- la 1ère méthode utilise beaucoup d'espace mais fournit des primitives rapides.
- la 2ème méthode est minimale en espace mais comprend des primitives plus lentes.

section 1 : la quantité d'informations d'un graphe est Θ(n+m)

Nous supposerons que les sommets des graphes ainsi que toutes les étiquettes, couleurs et autres attributs associés aux sommets, arcs ou arêtes des graphes ont une complexité en espace constante.

Conséquence de cette hypothèse et du fait qu'un graphe quelle que soit sa définition est un ensemble de sommets et leurs étiquettes éventuelles augmentée d'un ensemble d'arcs ou d'arêtes et leurs étiquettes éventuelles, il vient immédiatement :

propriété
La *quantité d'informations* d'un graphe G est **Θ(n+m)** où :

- n désigne $|V_G|$, le nombre de sommets.
- m désigne $|E_G|$, le nombre d'arcs ou arêtes.

Hypothèse générale : V_G de la forme [1,n] avec $n \in \mathbb{N}$.
Tous les graphes que nous manipulerons auront un ensemble de sommets égal à un intervalle d'entier de format [1,n]. Nous supposerons en outre que le type entier utilisé pour représenter les sommets est de complexité espace **Θ(1)**.

section 2 : implémentation par matrice d'adjacence

2.1 Implémentations du type graphe simple orienté

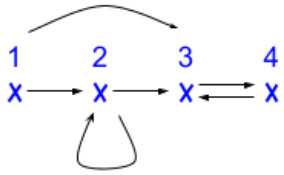
Cette implémentation est extrêmement proche de la définition d'un graphe simple orienté : un graphe est un couple ([1,n],D) avec $D \subseteq [1,n]^2$.

Définition :
Un graphe simple orienté ([1,n],D) est représenté par une structure contenant deux champs:

- un champ `dernier_sommet` de type `entier`
- un champ `estArc` qui est une matrice carrée de taille n*n de booléens où n est le dernier sommet.

Exemple :
Voici un graphe est son implémentation à l'aide d'une structure et d'une matrice :

Un graphe G



Une implémentation de G

dernier_sommet 4

estArc

	1	2	3	4
1	0	1	1	0
2	0	1	1	0
3	0	0	0	1
4	0	0	1	0

Qualité de l'implémentation : primitives en temps **Θ(1)**
Cette implémentation permet d'écrire les primitives courantes en temps constant.

Faiblesse de l'implémentation : espace non linéaire
La complexité espace de cette implémentation est **$\Theta(n^2)$.**

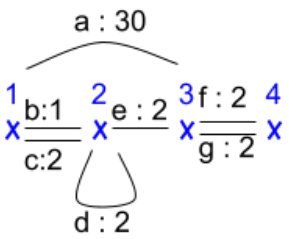
Pour des graphes ayant pas d'arcs (m=0) ou peu d'arcs (n=m) comme les arbres, la complexité espace est quadratique. Elle n'est donc pas linéaire et n'est donc pas optimale.

Cependant, si l'on manipule un graphe aléatoire (avec probabilité 50% pour chaque arc), le nombre d'arcs m est égal à $\Theta(n^2)$. Dans ce cas, l'implémentation est de complexité espace linéaire **$\Theta(n^2)$** en la complexité du graphe **$\Theta(m)$**. Elle est donc minimale.

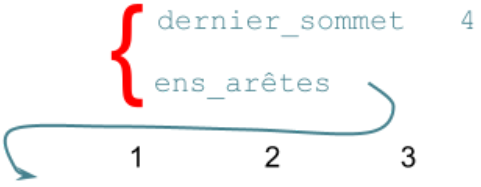
2.2 Implémentations d'autres types de graphes.

Cette implémentation peut naturellement être utilisée pour tout autre type de graphe. L'exemple suivant fournit l'implémentation d'un graphe à arêtes multiples pondérées par des entiers. Il n'est pas précisé ici comme implémenter l'ensemble des arêtes adjacentes à un couple de sommets (i,j) : un choix possible est d'utiliser par exemple une liste chaînée.

Un graphe G



Une implémentation de G



	1	2	3	4
1	()	((b,1),(c,2))	((a,30))	()
2	((b,1),(c,2))	((d,2))	((e,2))	()
3	((a,30))	((e,2))	()	((f,2),(g,2))
4	()	()	((f,2),(g,2))	()

section 2 : implémentation par tableau de listes chaînées

Cette implémentation est adaptée aux graphes ayant peu d'arcs m=Θ(n). Elle consiste à regrouper les arcs (resp. les arêtes) selon le sommet dont ils sortent (resp. dont ils sont sont incidents).

2.1 Implémentations du type graphe orienté à arcs multiples

Définition :

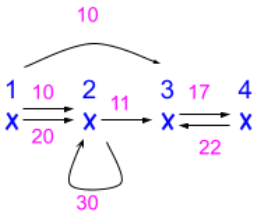
Un graphe orienté à arcs multiples ([1,n],E) est représenté par une structure contenant deux champs:

- un champ `dernier_sommet` de type `entier`
- un champ `listeArcsSortants` qui est un tableau associant à chaque sommet i la liste chaînée des arcs sortants du sommet i.

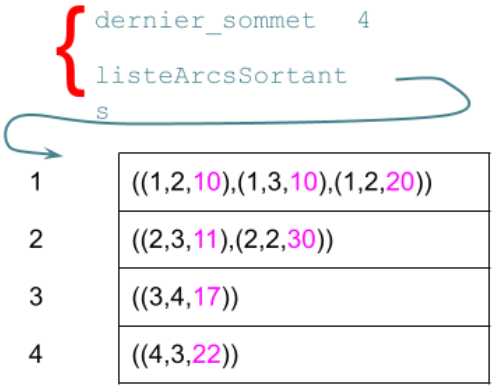
Exemple :

Voici un graphe à arcs pondérés est son implémentation à l'aide d'une structure et d'un tableau de listes chaînées ; l'implémentation chaînée des listes ne figure pas sur le dessin.

Un graphe G



Une implémentation de G



Qualité de l'implémentation : espace linéaire

Cette implémentation est de complexité espace **Θ(n+m)** et est donc linéaire et donc minimale pour tous les graphes (vérifiant n=O(m)).

Défaut de l'implémentation : primitives en temps non constante

Décider si il existe un arc d'un sommet i à un sommet j nécessite de parcourir la liste `listeArcsSortants[i]` et est donc linéaire en sa longueur, par définition égale au degré sortant.

3.2 Implémentations d'autres types de graphes.

Cette approche permet d'implémenter d'autres types de graphes ; citons le cas d'un graphe non orienté G, à chaque sommet i, la tableau `G.listeArêtesIncidentes` associe à chaque sommet i une liste composée de toutes les arêtes incidents à i.

Dans la littérature, le terme employé souvent est "*tableau liste successeurs*" voire "*tableau liste adjacence*" (https://fr.wikipedia.org/wiki/Liste_d%27adjacence#Op%C3%A9rations). La raison ici est que les graphes considérés ici sont simples : les arcs sont de simples couples de sommets ou les arêtes de simples paires de sommets. Aussi, on associe à chaque sommet i, la liste des sommets adjacents.