Leçon 6 : Les graphes

AYIKPA KACOUTCHY JEAN : Enseignant - Chercheur

Table des matières

Introduction

La notion de graphe est une structure qui permet de représenter plusieurs situations réelles, en but de leur apporter des solutions mathématiques et informatique, tel que :

- *Les réseaux de transport (routiers, ferrés, aériens ···),*
- *Les réseaux téléphoniques, électriques, de gaz,··· etc,*
- *Les réseaux d'ordinateurs,*
- *Ordonnancement des tâches,*
- *Circuits électroniques,*
- *etc*

Les arbres et les listes linéaires chaînées ne sont que des cas particuliers des graphes.

1- Généralité sur les graphes

I

1.1- Graphe

Un *graphe* est une *structure de données* composée d'un ensemble de *sommets*, et d'un ensemble de *relations entre ces sommets.*

- Si la relation n'est pas orientée, la relation est supposée exister dans les deux sens, *le graphe est dit non orienté ou symétrique*.
- Dans le cas contraire, si les relations sont orientées, *le graphe est dit orienté.*
- *Une relation est appelée un arc* (quelquefois une arête pour les graphes non orientés).
- *Les sommets sont aussi appelés nœuds ou points.*

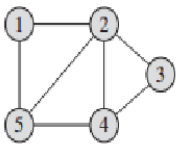
1.2- Un graphe non-orienté

Un graphe non-orienté est un couple G = (X, A) où

- *X* est l'ensemble des sommets et
- *A* est l'ensemble d'arêtes de G.

Un graphe non dirigé est caractérisé par une relation symétrique entre les sommets

- Une arête est un ensemble *{x,y}* de deux sommets
- On la notera tout de même *(x,y) équivalent à (y,x)*



Dans ce schéma ci-dessus nous avons *G = (X,A)* avec :

X = {1, 2, 3, 4, 5} et A= {(1, 2), (1, 5), (2, 4), (2, 5), (2, 3), (3, 4), (4, 5)}

Remarque :

- *Deux nœuds sont adjacents s'ils sont liés par une même arête.*
- *Une arête (x,y) est dite incidente aux nœuds x et y.*
- *Le degré d'un nœud est égal au nombre de ses arêtes incidentes.*
- *Le degré d'un graphe est le nombre maximal d'arêtes incidentes à tout sommet.*
- *Un graphe est connexe s'il existe un chemin de tout sommet à tout autre.*

- *Une composante connexe d'un graphe non oriente est un sous-graphe connexe maximal de ce grapl*
  .

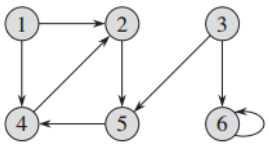
1.3- Graphe orienté

Un graphe orienté est un couple G = (X, A) où

- *X* est l'ensemble des sommets et
- *A* est l'ensemble d'arcs de G.

Un graphe non dirigé est caractérisé par une relation symétrique entre les sommets

- Un arc est un ensemble *{x,y}* de deux sommets
- On la notera tout de même *(x,y) est différent de (y,x)*



Dans ce schéma ci-dessus nous avons *G = (X,A)* avec :

X = {1, 2, 3, 4, 5, 6}, A = {(1, 2), (1, 4), (2, 5), (3, 5), (3, 6), (4, 2), (5, 4), (6, 6)}

Remarque :

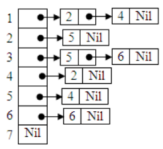
- *Une arête (x,y) possède l'origine x et la destination y.*
- *Cette arête est sortante pour x et entrante pour y.*
- *Le degré entrant (in-degree) et le degré sortant (out-degree) d'un nœud x sont respectivement égaux aux nombres d'arêtes entrantes et d'arêtes sortantes de x*
- *Un graphe est acyclique s'il n'y a aucun cycle, c'est-`a-dire s'il n'est pas possible de suivre les arêtes du graphes à partir d'un sommet x et de revenir à ce même sommet x*

1.3- Représentation des graphes

Les graphes peuvent être représentés en deux manières : en *listes d'adjacence (dynamique)* ou *en matrice d'adjacence (statique).*

- *Listes d'adjacence*

Dans cette représentation, les successeurs d'un nœud sont rangés dans une liste linéaire chaînée. Le graphe est représenté par un tableau T où T[i] contient la tête de la liste des successeurs du sommet numéro i. Le graphe (S, A) présenté au début de cette section peut être représenté comme suit :



Les listes d'adjacence sont triées par numéro de sommet, mais les successeurs peuvent apparaître dans n'importe quel ordre.

Déclaration :

Type TMaillon = Structure

Successeur : entier

Suivant : TMaillon

Fin

Var

Graphe : Tableau[1..n] de ^TMaillon

- *Matrice d'adjacence*

Dans cette représentation le graphe est stocké dans un tableau à deux dimensions de valeurs booléennes ou binaires. Chaque case (x, y) du tableau est égale à vrai (1) s'il existe un arc de x vers y, et faux (0) sinon. Dans la matrice suivante, on représente le graphe précédent (1 pour vrai et 0 pour faux) :

	1	2	3	4	5	6	7
1	0	1	0	1	0	0	0
2	0	0	0	0	1	0	0
3	0	0	0	0	1	1	0
4	0	1	0	0	0	0	0
5	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0

Il est clair que la matrice d'adjacence d'un graphe non orienté est symétrique puisque chaque arête existe dans les deux sens.

Déclaration :

Var

Graphe : Tableau[1..n,1..n] de booléen

Application 1 :

II

Exercice

[solution n°1 p.12*]*

L'adressage direct fonctionne bien :

- lorsque l'univers *U* des clés est raisonnablement grand
- lorsque l'univers *U* des clés est petit
- lorsque l'univers *U* des clés est grand

Exercice

[solution n°2 p.12*]*

Un alvéole est :

- une représentation d'une table
- est l'ensemble dynamique d'un tableau
- est une position de l'adressage direct

2- Parcours et algorithmes de graphes

2.1- Généralité sur les algorithmes de parcours

De même que pour les arbres, il est important de pouvoir parcourir un graphe selon certaines règles, cependant, le parcours des graphe est un peut différent de celui des arbres.

Dans un arbre, si on commence à partir de la racine on peut atteindre tous les nœuds, malheureusement, ce n'est pas le cas pour un graphe où on est obligé de reprendre le parcours tant qu'il y a des sommets non visités. En plus un graphe peut contenir des cycles, ce qui conduit à des boucles infinies de parcours.

Il existe deux types de parcours de graphes : *le parcours en profondeur d'abord (DepthFirst Search) et le parcours en largeur d'abord (Breadth First Search).*

2.2- Parcours en largeur

Dans ce parcours, un sommet *s* est fixé comme origine et l'on visite tous les sommet situés à une distance k de s avant de passer à ceux situés à k + 1. On utilise pour cela une file d'attente.

n est égal au nombre de sommet

Var

Graphe : Tableau[1..n,1..n] de booléen

Visité : Tableau[1..n] de booléen

F : File

Procédure BFS(Sommet : entier)

Var

i,s : entier

Début

Enfiler(F,Sommet)

Tant que (non (File_Vide(F))) faire

Defiler(F,s)

Afficher(s)

Pour i de 1 à n faire

Si (Graphe[s,i] et non Visité[i]) Alors

Enfiler(F,i)

FinSi

FinPour

FinTQ

Fin

2.2- Parcours en Profondeur

Le principe du DFS est de visiter tous les sommets en allant le plus profondément possible dans le graphe.

Var

Graphe : Tableau[1..n,1..n] de booléen

Visité : Tableau[1..n] de booléen

Procédure DFS(Sommet : entier)

var

i : entier ;

Début

Visité[Sommet] ← Vrai

Afficher(sommet)

Pour i de 1 à n faire

Si (Graphe[sommet,i] et non(Visité[i])) Alors

DFS(i)

Fin Si

Fin Pour

Fin

La procédure DFS peut être utilisée pour divers objectifs :

- *Pour vérifier s'il existe une chemin d'un sommet s1 vers un autre s2 en initialisant le tableaux Visite à faux et en appelant DFS(s1) pour ce sommet. Si Visité[s2] est à*

vrai à la fin de l'appel de DFS, alors un chemin existe entre s1 et s2.

- *Pour vérifier si un circuit contentent deux sommets s1 et s2 existe, en appelant DFS(s1) pour un tableaux Visité1 et DFS(s2) pour un tableaux Visité2, si Visité1[s2]*

et Visité2[s1] sont à Vrai après l'appel alors un tel circuit existe.

- *De la même manière pour vérifier si un graphe est cyclique (contient des circuits) ou non.*
- *Pour trouver les chemins minimums d'un sommet s vers tous les autres.*

Application 2 :

Exercice

[solution n°3 p.12]

Une table de hachage

- est une structure de données permettant d'implémenter des valeurs à indice numérique
- est une structure de données qui permet l'implémentation d'une table des symboles lorsque les clés sont des chaînes de caractères.
- est une structure de données permettant d'utiliser les listes

Exercice

[solution n°4 p.12]

Une fonction de hachage est :

- établit une correspondance entre un association clé,valeur et un indice du tableau
- établit une correspondance entre un couple(clé,valeur) et une chaîne
- établit une correspondance entre un couple(clé,valeur) et une liste chaînée

Solutions des exercices

> **Solution** n°1

Exercice p. 7

L'adressage direct fonctionne bien :

- ○ lorsque l'univers *U* des clés est raisonnablement grand
- ◉ lorsque l'univers *U* des clés est petit
- ○ lorsque l'univers *U* des clés est grand

> **Solution** n°2

Exercice p. 7

Un alvéole est :

- ○ une représentation d'une table
- ○ est l'ensemble dynamique d'un tableau
- ◉ est une position de l'adressage direct

> **Solution** n°3

Exercice p. 11

Une table de hachage

- ○ est une structure de données permettant d'implémenter des valeurs à indice numérique
- ◉

est une structure de données qui permet l'implémentation d'une table des symboles lorsque les clés sont des chaînes de caractères.

- ○ est une structure de données permettant d'utiliser les listes

> **Solution** n°4

Exercice p. 11

Une fonction de hachage est :

- ◉ établit une correspondance entre un association clé,valeur et un indice du tableau
- ○ établit une correspondance entre un couple(clé,valeur) et une chaîne
- ○ établit une correspondance entre un couple(clé,valeur) et une liste chaînée