TP6 MPSI: Graphes

Alexis Saurin
Alexis.Saurin@ens.fr
http://www.eleves.ens.fr/~saurin/tp_caml

10 mai 2003

1 Introduction

Cette séance va être consacrée à l'étude d'une structure informatique très classique et très centrale : les graphes. On va d'abord donner les définitions principales concernant les graphes. On s'intéressera ensuite aux manières de représenter des graphes avec des structures de données informatiques et on s'intéressera ensuite à quelques algorithmes de base sur les graphes, en fait à un algorithme de parcours de graphe et à de applications de ce parcours.

Vous trouverez dans l'ouvrage *Introduction à l'algorithmique* de Th. Cormen, Ch. Leiserson et R. Rivest de nombreux algorithmes sur les graphes (arbres couvrants minimaux, plus courts chemins, flot maximal) qui peuvent sans doute servir de base à un TIPE sur les graphes. Les définitions de théorie des graphes de cette feuille de TP sont reprises de ce livre. Ce livre regorge de problèmes algorithmiques très variés.

2 Définitions

Définition 1 (Graphe orienté) *Un graphe orienté G est la donnée d'un couple (S, A), où S est un ensemble fini de sommets et A une relation binaire sur S (c'est-à-dire un ensemble de couples d'éléments de S), l'ensemble des arêtes. Pour représenter graphiquement un graphe, on dessinera les sommets comme des cercles et les arêtes comme des flèches allant d'un sommet à un autre sommet.*

Définition 2 (Graphe non orienté) *Un graphe non orienté est un graphe pour lequel la relation binaire A est symétrique et anti-réflexive (symétrique pour qu'il n'y est pas d'orientation et anti-réflexive car on veut empêcher les boucles d'un sommet vers lui-même dans le cas non-orienté). On peut voir aussi l'ensemble A comme un ensemble de paires et non de couples.*

Remarques et notations

- Beaucoup de définitions sont semblables pour les graphes orientés ou non orienté, mais dans certains cas, les termes varient légèrement. Il faut donc toujours bien savoir quel type de graphe on considère.
- Dans le cas d'un graphe orienté, on dit que l'arête (u, v) part de u et arrive en v. Pour un graphe non orienté, on dit que l'arête (u, v) est incidente aux sommets u et v.
- La relation binaire A est aussi appelée relation d'adjacence : v est adjacent à u si $(u, v) \in A$. Dans un graphe orienté, on note parfois v adjacent à u

par $u \to v$.

Définition 3 (Degré d'un sommet) *Le degré d'un sommet est d'un graphe non orienté est le nombre d'arêtes qui lui sont incidentes. Dans le cas d'un graphe orienté, on distingue en outre le degré rentrant (nombre d'arcs qui arrivent à ce sommet) et le degré sortant (nombre d'arcs qui en partent).*

Définition 4 (Chemin dans un graphe) *Un chemin de longueur k d'un sommet u vers un sommet v dans un graphe orienté G=(S, A) est une suite finie de sommets* $\langle u_0, \ldots, u_i, \ldots, u_n \rangle$ *avec* $u_0 = u$ *et* $u_n = v$ *et pour tout* $i \in [0, n-1]$, $(u_i, u_{i+1}) \in A$.

On dit que v est accessible à partir de v s'il existe un chemin de u à v.

Un chemin est élémentaire si tous les sommets du chemin sont distincts.

Définition 5 (Circuits et cycles) *Un circuit dans un graphe orienté est un chemin* $\langle u_0, \ldots, u_i, \ldots, u_n \rangle$ *tel que* $u_0 = u_n$, *avec* $n \geq 1$. *On parle aussi de circuit élémentaire si* $\langle u_1, \ldots, u_i, \ldots, u_n \rangle$ *est en plus un chemin élémentaire.*

Un cycle (élémentaire) dans un graphe non orienté est un chemin $\langle u_0, \ldots, u_i, \ldots, u_n \rangle$ *tel que* $u_0 = u_n$, *avec* $n \geq 3$ *tel que* $u_1, \ldots, u_n$ *sont tous distincts.*

Un graphe non orienté sans cycle est dit acyclique.

Définition 6 (Composante connexes) *Un graphe non oritenté est connexe si chaque paire de sommet est reliée par une chaîne. Les composantes connexes sont constituées des classes d'équivalence pour la relation être "accessible à partir de".*

Définition 7 (CFC) *Un graphe orienté est fortement connexe si chaque sommet est accessible à partir de n'importe quel autre. Les composantes fortement connexes du graphe sont les classes d'équivalence de la relation "être mutuellement accessibles".*

Définition 8 (Isomorphisme de graphes) *Deux graphes G=(S, A) et G'=(S', A') sont isomorphes s'il existe une bijection* $f : S \mapsto S'$ *telle que* $(u, v) \in A$ *si, et seulement si,* $(f(u), f(v)) \in A'$

Définition 9 (Sous-graphe) *G'=(S', A') est un sous-graphe de G=(S, A) si* $S' \subset S$ *et* $A' \subset A$.

Définition 10 (Graphe complet) *Un graphe complet est un graphe non orienté dans lequel A est la relation binaire anti-refléxive pleine (ie tout couple de sommets distincts est dans A).*

Définition 11 (Graphe biparti) *Un graphe biparti est un graphe non orienté G=(S, A) dans lequel S peut-être partitionné est* S_1 *et* S_2 *tels que* $(u, v) \in A$ *implique* $u \in S_1$ *et* $v \in S_2$ *ou* $u \in S_2$ *et* $v \in S_1$ *(ie toutes les arêtes passent d'un ensemble de sommets à l'autre ensemble).*

3 Exercices

Exercice 1: *Arbres*

Donner les conditions nécessaires et suffisantes sur un graphe pour que ce soit un arbre.

Exercice 2: *Graphes chromatiques*

Une k-coloration d'un graphe non orienté G=(S, A) est une fonction $c : S \mapsto \{0, 1, \ldots, k-1\}$ telle que pour chaque arête $(u, v) \in A$, $c(u) \neq c(v)$. Les nombres représentent k couleurs et les sommets adjacents doivent avoir des couleurs différentes.

1. Montrer que tout arbre peut être colorié avec uniquement deux couleurs
2. Montrer que les énoncés suivants sont équivalents :
   (a) G est biparti ;
   (b) G est 2-chromatique ;

(c) G n'a aucun cycle de longueur impaire.

3. Soit d le degré maximal d'un sommet du graphe G. Démontrer que G peut être colorié avec d+1 couleurs.
4. Montrer que si G a O(|S|) arêtes, alors G peut-être colorié avec O($\sqrt{|S|}$) couleurs.

Exercice 3: *Graphes d'amitié*

Réécrire chacun des énoncés suivants comme un théorème sur les graphes non orientés, puis en faire la démonstration. On suppose que l'amitié est symétrique mais n'est pas réflexive.

1. Dans un groupe quelconque de $n \geq 2$ personnes, il existe deux personnes ayant le même nombre d'amis dans le groupe.
2. Chaque groupe de six personnes contient soit trois amis, soit trois personnes étrangères les unes aux autres.
3. Un groupe de personnes quelconque peut être divisé en deux sous-groupes tels qu'au moins la moitié de amis de chaque personne appartiennent au sous-groupe dont cette personne n'est pas membre.
4. Si chacun dans un groupe est l'ami d'au moins la moitié des personnes du groupe, alors le groupe peut être réparti autour d'une table de façon que chacun soit assis entre deux amis.

4 Représentation informatique

Il existe plusieurs manière de représenter les graphes dans un ordinateur. Ces manières n'auront pas les mêmes propriétés, et en particulier elles ne donneront pas nécessairement lieu à des algorithmes de la même complexité pour résoudre un problème donné. En outre, elles ne prendront pas toutes le même espace mémoire, certaines seront plus facilement modifiables que d'autres...

Question 1: *Représentations informatiques*

Proposez quelques manières de représenter les graphes en Caml Light. Définir à chaque fois un type pour les graphes correspondant à cette représentation.

Question 2: *Différence des représentations*

Définition 12 (transposée d'un graphe) *La transposée d'un graphe orieenté G=(S, A) est le graphe G'=(S, A') où* $A' = \{(u,v) \in S^2 / (v,u) \in A\}$.

Programmez la transposition de graphe pour chacune des représentations.

Définition 13 (carré d'un graphe) *Le carré d'un graphe orienté G=(S, A) est le graphe* G^2*=(S,* A^2*) tel que* $(u,v) \in A^2 ssi \exists w \in S tq (u,w) \in A, (w,v) \in A$. *C'est-à-dire qu'il y a un arc dans le carré si, et seulement si, il y a un chemin de longueur exactement deux dans le graphe de départ.*

Programmez le carré d'un graphe pour les diverses représentations.

Question 3: *Degré entrant, degré sortant*

Étant donnée une représentation par listes d'adjacence d'un graphe orienté, combien de temps faut-il pour calculer le degré sortant de tous les sommets ? et pour le degré entrant ?

Question 4: *Fermeture transitive d'un graphe*

Étant donné la représentation par matrice d'adjacence d'un graphe orienté, programmez une fonction qui calcule la fermeture transitive d'un graphe G donné, c'est-à-dire un nouveau graphe ayant même sommet que le précédent mais dont les arcs sont les couples de sommets (u, v) tels que v est accessible à partir de u dans G (on s'inspirera de l'exercice sur le carré d'un graphe).

5 Parcours d'un graphe orienté

On représente maintenant un graphe par sa matrice d'adjacence.

On va s'intéresser maintenant à des manières de parcourir un graphe. On va voir le parcours en profondeur d'un graphe (on pourrait aussi étudier le parcours en largeur), et un certain nombre d'application de ce parcours. Il n'est pas surprenant que beaucoup d'alogrithme sur les graphes soient fondés sur des algorithmes de parcours de graphe dans la mesure où pour faire un trvail quelconque sur un graphe il faudra la plupart du temps avoir vu tous ses sommets, et que les algorithmes de bases de parcours de graphes permettent de conserver beaucoup d'information sur le graphe.

Question 5: *Parcours en profondeur*

Le parcours en profondeur d'un graphe consiste à parcourir tous les sommets du graphe de la manière suivante :

```
PP(G):
pour chaque sommet u de G
  faire couleur(u)<- Blanc
         pere(u)<- NIL
temps<- 0
pour chaque sommet u de G
  faire si couleur(u)=Blanc
            alors Visiter-PP(u, G)

Visiter-PP(u):
couleur(u)<- Gris
d(u)<-temps
temps<- temps + 1
pour chaque sommet v adjacent
                    à u dans G
  faire si couleur(v)=Blanc
            alors pere(v)<- u
                  Visiter-PP(v)
couleur(u)<- Noir
f(u)<- temps
temps<- temps + 1
```

Réfléchissez, avant de vous lancer dans la programmation de l'agorithme, à la manière dont vous représenterez le compteur de temps, les attributs de couleur, de temps de début de traitement et de fin de parcours, ainsi que de parenté dans le parcours.

Qu'avez-vous construit une fois que vous avez parcouru le graphe ?

Réfléchissez à l'utilité de conserver les données pere(u), d(u) et f(u).

Question 6: *Détermination de l'acyclicité d'un graphe*

Utilisez le principe du parcours en profondeur pour déterminer si un graphe a des cycles ou non.

Vous commencerez par écrire l'algorithme, puis vous le programmerez.

Question 7: *Tri topologique d'un graphe orienté*

Trier topologiquement un graphe orienté revient à mettre sur l'ensemble de ses sommets un ordre qui soit compatible avec la relation d'adjacence, à savoir si $u \to v$ alors $u \leq v$.

Montrer que si un graphe a un cycle on ne peut pas le trier topologiquement

On suppose maintenant que notre graphe est acyclique. Programmer un algorithme qui permette de faire un tri topologique grâce à un parcours en profondeur du graphe (pensez à utiliser les dates de fin de parcours d'un sommet, ie les f(u)).