Option Informatique en Spé MP

Devoir surveillé du mardi 18 mars 1997

Résumé

Nous étudions la notion d'arbre recouvrant minimal (en anglais : *minimal spanning tree*) d'un graphe valué. Nous analysons deux algorithmes de construction d'un tel arbre : le premier a été publié par Kruskal en 1956, le second par Prim en 1957 (mais était déjà connu de Jarník en 1930). Enfin, nous donnons un cadre algébrique général pour l'étude d'une famille d'algorithmes *gloutons*, dont font partie les algorithmes de Kruskal et de Prim.

Veuillez rédiger chaque partie sur une copie séparée.

Table des matières

1 Graphes valués : définitions et notations

▶ Un *graphe valué* est un triplet (S, A, π) où S est un ensemble fini de *sommets*, A un ensemble d'*arêtes* (une arête est une paire $\{x, y\}$ de sommets distincts) et π une application de A dans $\mathbb{R}$: $\pi\big(\{x, y\}\big)$ est le *poids* de l'arête $\{x, y\}$. La figure 1 représente un graphe valué comportant huit sommets et onze arêtes.

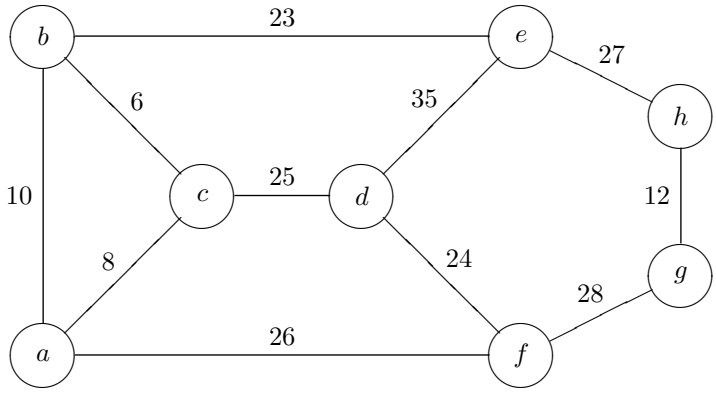

Figure 1: un graphe valué

▶ Les notions de *voisin*, de *chemin*, de *cycle* et de *connexité* sont connues. Un chemin est *élémentaire* s'il ne repasse pas deux fois par un même sommet. Un graphe est *acyclique* s'il ne comporte aucun cycle. Tous les graphes considérés dans la suite sont connexes et non orientés ; on note $|S|$ le nombre de sommets et $|A|$ le nombre d'arêtes.

▶ $G' = (S', A')$ est un *sous-graphe* de $G = (S, A)$ si $S' \subset S$ et $A' \subset A$; c'est un *graphe partiel* si de plus $S' = S$. Soit S' une partie de S, et A' l'ensemble des arêtes de G reliant deux sommets appartenant à S' ; nous dirons que (S', A') est le sous-graphe de G *induit* par S'.

▶ Rappel : soit $G = (S, A)$ un graphe à n sommets ; les assertions suivantes sont deux à deux équivalentes, et définissent un *arbre* :

1. G est connexe et acyclique
2. G est connexe et comporte $n - 1$ arêtes
3. G est acyclique et comporte $n - 1$ arêtes
4. G est acyclique, mais ne le reste pas si on lui ajoute une arête
5. G est connexe, mais ne le reste pas si on lui enlève une arête
6. deux sommets quelconques de G sont reliés par un et un seul chemin élémentaire

▶ Un *arbre recouvrant* de G est un graphe partiel de G connexe et acyclique : ce qui justifie sa dénomination. L'arbre est minimal si la somme des poids de ses arêtes est minimale.

▶ Une *forêt* est une famille d'arbres deux à deux disjoints ; la définition d'une *forêt recouvrante* est évidente.

2 Quelques questions d'ordre général

Question 1 • Justifiez l'existence d'un arbre recouvrant minimal. Donnez un exemple *très simple* où l'on n'a pas l'unicité.

Question 2 • Montrez que si les poids des arêtes sont deux à deux distincts, il n'existe qu'un seul arbre recouvrant minimal.

Question 3 • Montrez que l'on peut supposer, sans perte de généralité, que les poids des arêtes sont tous strictement positifs.

Question 4 • L'étudiant Maurice ALANISETTE se propose d'appliquer une méthode *diviser pour régner* à la recherche d'un arbre recouvrant minimal d'un graphe valué $G = (S, A, \pi)$ donné. Il procède comme suit : si le graphe n'a qu'un sommet, le problème est résolu ; sinon, soient $S = S_1 \cup S_2$ une partition de l'ensemble des sommets, T_1 un arbre recouvrant minimal du sous-graphe de G induit par S_1, et T_2 défini de façon analogue. Parmi les arêtes reliant un sommet de S_1 et un sommet de S_2, choisissons-en une a de poids minimal ; en réunissant T_1, T_2 et cette arête, on obtient un arbre recouvrant minimal du graphe G. Que pensez-vous de cette méthode ?

3 L'algorithme de Kruskal

▶ Nous décrivons un premier algorithme de construction d'un arbre recouvrant minimal.

Question 5 • Soit $\{x, y\}$ une arête de poids minimal du graphe valué $G = (S, A, \pi)$. Montrez que cette arête fait partie d'un arbre recouvrant minimal.

Question 6 • Soit $G_1 = (S_1, A_1)$ un sous-graphe d'un arbre recouvrant minimal T de G. On suppose S_1 contenu *strictement* dans S. Montrez qu'il existe au moins une arête $a = \{x, y\} \in G \setminus A_1$ telle que, notant $S_2 = S_1 \cup \{x, y\}$ et $A_2 = A_1 \cup \{a\}$, le graphe $G_2 = (S_2, A_2)$ soit acyclique.

Question 7 • Montrez que l'on peut choisir a de façon que $G_2 = (S_2, A_2)$ soit un sous-graphe d'un arbre recouvrant minimal de G.

Question 8 • En déduire un algorithme de construction d'un arbre recouvrant minimal. Note : on ne demande pas, ici, d'écrire un programme en Caml ; en revanche, la preuve du bon fonctionnement de l'algorithme devra apparaître sur la copie.

Question 9 • Appliquez cet algorithme au graphe de la figure 1.

Question 10 • Comment modifier l'algorithme précédent pour obtenir un arbre recouvrant de poids *maximal* ?

Question 11 • Que pensez-vous de l'algorithme suivant, proposé par l'étudiant Marcel-Paul TORTUEUX : on examine les arêtes par poids *décroissants* ; si l'arête examinée participe à un cycle, on la supprime, sinon on la garde et on passe à la suivante.

4 Mise en œuvre naïve de l'algorithme de Kruskal

▶ Fondamentalement, l'algorithme de KRUSKAL consiste à faire évoluer une partition de l'ensemble S des sommets de G ; initialement, la partition est formée de $n = |S|$ singletons. À chaque fois que l'on ajoute une arête reliant deux sommets x et y, on fusionne les classes de x et y. L'algorithme se termine lorsque la partition ne compte plus qu'une classe.

▶ Une partition de l'intervalle discret $[\![0, n-1]\!]$ peut être commodément représentée par un vecteur d'entiers `repres` : `repres.(i)` est le représentant privilégié de la classe de `i`. Ainsi, on pourra représenter la partition $\big\{\{1, 2, 6, 7\}; \{5\}; \{0, 4, 3\}\big\}$ de $[\![0, 7]\!]$ par le vecteur `[|4;6;6;4;4;5;6;6|]`.

Question 12 • Comment doit être initialisé `repres` ?

Question 13 • Écrire une fonction qui fusionne les classes de x et y. Calculez son coût, en fonction de n ; l'unité de coût est la lecture ou la modification d'une composante de `repres`.

Question 14 • Supposons que les arêtes de G sont fournies sous forme d'une liste triée par poids croissants. Écrire une fonction qui calcule un arbre recouvrant minimal de G.

Question 15 • Estimez le coût de ce calcul.

5 Mise en œuvre de l'algorithme de Kruskal avec la méthode *union-find*

▶ La méthode que nous venons de proposer est coûteuse parce que la fusion de deux classes nécessite le parcours complet du vecteur `repres`. Pour éviter ceci, on décide que, lors d'une fusion, le représentant de la nouvelle classe sera le représentant de la classe de plus grand cardinal.

▶ Nous allons donc gérer deux vecteurs : l'un, `card`, de type `int vect`, donne le cardinal de la classe de chaque sommet. L'autre vecteur, `classe`, de type `int list vect`, donne pour chaque représentant la liste des sommets de sa classe ; pour un sommet qui n'est plus représentant, la liste associé est quelconque.

Question 16 • Comment doivent être initialisés ces vecteurs ?

Question 17 • Écrivez une fonction qui fusionne les classes de x et y. Quel est le coût de cette fusion ?

Question 18 • Montrez que le coût d'une suite de fusions est dominé par $n \lg n$.

6 L'algorithme de Prim

▶ Nous décrivons un deuxième algorithme de construction d'un arbre recouvrant minimal.

Question 19 • Soit x un sommet du graphe valué $G = (S, A, \pi)$, et $\{x, y\}$ une arête de poids minimal parmi celles qui relient x à un autre sommet. Montrez que $\{x, y\}$ fait partie d'un arbre recouvrant minimal.

Question 20 • Soit A_1 une partie de A strictement contenue dans l'ensemble des arêtes d'un arbre recouvrant minimal T de G. Soit S_1 l'ensemble des sommets reliés par les arêtes de A_1 ; on suppose S_1 contenu *strictement* dans S. Justifiez l'existence d'une arête $a = \{u, v\}$ telle que $u \in S_1$, $v \notin S_1$, et de poids minimal pour cette propriété.

Question 21 • Notant $S_2 = S_1 \cup \{v\}$ et $A_2 = A_1 \cup \{a\}$, montrez que (S_2, A_2) est un sous-graphe d'un arbre recouvrant minimal.

Question 22 • En déduire un algorithme de construction d'un arbre recouvrant minimal. Note : on ne demande pas, ici, d'écrire un programme en Caml ; en revanche, la preuve du bon fonctionnement de l'algorithme devra apparaître sur la copie.

Question 23 • Appliquez cet algorithme au graphe de la figure 1, en prenant comme sommet initial a.

Question 24 • Que se passe-t-il lorsque l'on applique l'algorithme de Prim à un graphe valué dont toutes les arêtes ont le même poids ?

7 Matroïdes

▶ Un *matroïde* est une structure algébrique formée d'un ensemble fini E, et d'une famille $\mathcal{I}$ non vide de parties de E dites *libres*, vérifiant les propriétés suivantes :

1. tout sous-ensemble X d'une partie libre Y est à son tour une partie libre
2. si X et Y sont deux parties libres vérifiant $|X| < |Y|$, alors il existe $x \in Y \setminus X$ tel que $X \cup \{x\}$ soit encore une partie libre

Notons que la partie vide est libre.

Question 25 • Soit M une matrice ; exhiber un matroïde naturellement associé à M (ce qui expliquera le choix de ce terme par WHITNEY en 1935).

Question 26 • Soit $G = (S, A)$ un graphe non orienté (mais non nécessairement connexe). Justifier : la famille des graphes partiels acycliques de G définit un matroïde.

Question 27 • Montrer que deux parties libres X et Y d'un même matroïde, maximales pour l'inclusion, ont même cardinal.

▶ Un matroïde $(E, \mathcal{I})$ est *pondéré* s'il est muni d'une application $\pi : E \to \mathbb{R}$. Le *poids* d'une partie X de E est la somme des poids des éléments de X :

$$\pi(X) = \sum_{x \in X} \pi(x)$$

▶ Une partie X de E est *optimale* si elle est libre maximale, et si elle est de poids minimal parmi les parties libres maximales. On se propose de décrire une méthode générique pour construire une partie optimale de E.

Question 28 • On suppose qu'il existe au moins une partie non vide de E libre. Soit x un élément de E tel que $\{x\}$ soit libre, et de poids minimal vis-à-vis de cette propriété. Montrer que x appartient à une partie optimale de E.

▶ On considère la fonction Caml suivante, où `est_libre` est une fonction qui indique si une partie donnée de E est libre.

```
let rec glouton courants = function
 | []   -> courants
 | t::q -> if est_libre (t::courants)
           then glouton (t::courants) q
           else glouton courants q;;
```

Question 29 • Montrer que `glouton []`, appliquée à une liste dans laquelle les éléments de E sont rangés par poids croissants, rend une liste énumérant une partie optimale de E.

Question 30 • Montrer que l'algorithme de KRUSKAL est un cas particulier d'application du schéma général qui vient d'être étudié.

FIN