Option Informatique
Arbres couvrants minimaux

Sujet

30 novembre 2006

1 Petit rappel sur les graphes

▶ Un graphe non-orienté est un couple (V, E) ou V est l'ensemble des sommets (*vertices*) et $E \subseteq \mathcal{P}_2(V)$ est l'ensemble des arêtes (*edges*). E est un ensemble de paires de sommets.

Deux sommets sont dits *voisins* s'ils sont reliés par une arête. Une arête est dite *adjacente* à un sommet e si ce sommet est l'une de ses deux extrémités. On peut pour certains problèmes associer un poids (ou un coût) à chaque arête.

▶ La représentation habituelle des graphes se fait par *listes d'adjacence* : un graphe est représenté par un tableau de longueur $|V|$ dont la case i contient la liste des voisins du sommet i. Une autre représentation classique mais plus coûteuse en mémoire est la *matrice d'adjacence* : un graphe est alors représenté par une matrice $M \in \{0,1\}^{|E|\times|E|}$ telle que $m_{i,j} = 1$ si et seulement s'il y a une arête entre i et j. Nous utiliserons ici la représentation *forward star* qui est proche de la représentation par listes d'adjacence. La seule différence est que toutes les listes d'adjacence sont concaténées dans une liste. Pour chaque sommet on stocke la position dans ce tableau du début de sa liste d'adjacence. On a donc :

```
(* Une arête est un triplet contenant
    les deux extrémités et le poids. *)
type arete = Edge of int*int*int ; ;
type graphe =  Sommets : int list ;
               Aretes : arete list ; ;
```

▶ On considère ici des graphes non orientés valués : $G = (E, A, v)$ où E est un ensemble fini (ensemble des sommets), $A \subset \mathcal{P}_2(E)$ est l'ensemble des arêtes et $v : A \to \mathbb{R}$ est la fonction de poids. Le poids de G est $\Sigma_{\alpha\in A} v(\alpha)$.

Vous trouverez un exemple de graphe pondéré en FIG. 1, et sa représentation en FIG. 2.

▶ Un *sous-graphe* de G est un graphe de la forme $G' = (E', A', v_{|A'})$ où $E' \subset E, A' \subset \mathcal{P}_2(E') \cap A$. Si $E = E'$, G' est couvrant.

Un *chemin* de a à b, $a, b \in E$ est une suite finie de sommets $(x_1, \ldots, x_n)$ avec $a = x_1$, $b = x_n$. On dit qu'un chemin est *simple* si la suite est injective. On dit que G est *connexe* s'il existe toujours un chemin entre deux sommets quelconques (il y a alors un chemin simple). Si, de plus, il y a unicité du chemin simple, alors on dit que G est un arbre.

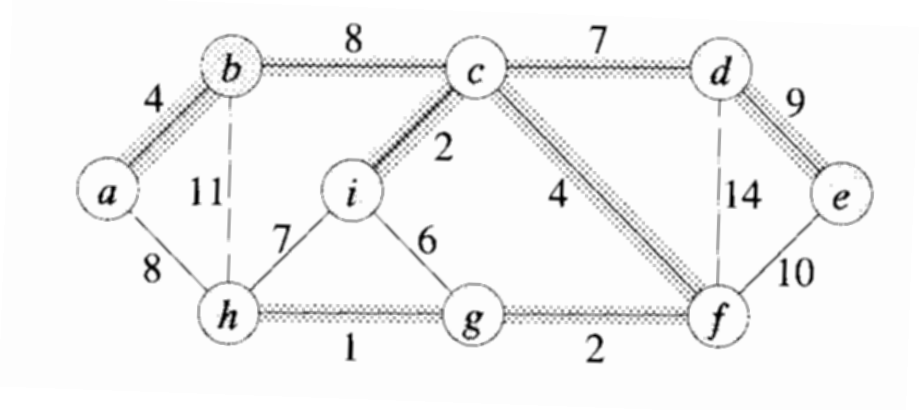

FIG. 1 – Graphe pondéré

Listes d'adjacence	`[|[("b",4);("h",8)]; [("a",4);("c",8);("h",11)]; [("b",8);("d",7);("f",4);("i",2)]; ...|]`
Matrice d'adjacence	$\begin{bmatrix} 0 & 4 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ 4 & 0 & 8 & 0 & 0 & 0 & 0 & 11 & 0 \\ 0 & 8 & 0 & 7 & 0 & 4 & 0 & 0 & 2 \\ & ... & & 0 & & & & & \\ & & & & 0 & & & & \\ & & & & & 0 & & & \\ & & & & & & 0 & & \\ & & & & & & & 0 & \\ & & & & & & & & 0 \end{bmatrix}$
Forward star	`{ Sommets = ["a"; "b"; "c"; "d"; "e"; "f"; "g"; "h"; "i"]; Aretes = [("a",4,"b"); ("a",8,"h"); ("b",11,"h"); ("b",8,"c"); ("c",7,"d",); ("c",4,"f"); ("c",2,"i"); ...] }`

FIG. 2 – Quelques représentations du graphe de la FIG. 1

Question 1 *Soit $G = (V, E)$ un graphe non orienté à n sommets. Justifier (rapidement) l'équivalence des assertions suivantes :*

1. *G est connexe et acyclique.*
2. *G est connexe et comporte $n - 1$ arêtes reliant des sommets distincts.*
3. *G est acyclique et comporte $n - 1$ arêtes reliant des sommets distincts.*
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 ont reliés par un et un seul chemin élémentaire.*

2 Problème de l'arbre couvrant

▶ On se pose le problème suivant : supposant G connexe, trouver un sous-graphe couvrant G' qui soit un arbre, et de poids minimal. Evidemment, cela revient à chercher un sous-ensemble de A. On identifie $F \subset A$ au sous-graphe (E', F) contenant les sommets extremités d'arêtes de F.

Question 2 *On commence avec l'idée de partir d'un graphe sans arêtes, auquel on rajoute des arêtes de notre graphe initial, jusqu'à atteindre la condition 4 de la question 1. Soit un graphe $G = (V, E)$ acyclique, et $e \notin E$. Montrer que $G' = (V, E \cup \{e\})$ est acyclique si et seulement si e relie deux composantes connexes de G.*

3 Algorithme de Kruskal

▶ On peut donc tirer de la question précédente un algorithme de construction d'un arbre couvrant d'un graphe $G = (V, E)$: on part d'un graphe $G_0 = (V, \emptyset)$ sans arêtes, et on réunit au fur et à mesure les composantes connexes du graphe en ajoutant à ce graphe des arêtes prises dans E. On oublie pour l'instant la minimalité de l'arbre.

Question 3 *Écrire l'explosion d'un graphe en un ensemble de ses sous-graphes élémentaires. Évaluer sa complexité.*
(`explose : graphe → graphe list`)

Question 4 *Écrire une fonction `fusion` réalisant la fusion par l'arête a de deux graphes g_1 et g_2 d'une forêt (liste de graphes), de type `graphe list → graphe → graphe → arete → graphe list`. Évaluer sa complexité.*

Question 5 *Écrire `cherche_arbre`, de type `int → graphe list → graphe`, qui à tout élément `x` et à toute forêt `f` associe le graphe de `f` auquel appartient `x`. Évaluer sa complexité.*

Question 6 *Écrire la recherche de **la première** arête d'une liste reliant deux arbres d'une forêt*
(`cherche_areteK : graphe list → arete list → graphe * graphe * arete`).

Question 7 *En déduire une implémentation de la recherche d'arbre couvrant, et évaluer sa complexité.*
(`spanningtree : graphe → graphe`).

▶ On cherche maintenant à modifier le programme ci-dessus pour qu'il cherche un graphe couvrant **de poids minimal**. On se propose de considérer, dans l'algorithme précédent, les arêtes *triées par ordre de poids croissant* : cela donne l'algorithme de Kruskal.

Question 8 *Écrire une fonction* `tri_aretes` *de type* `arete list → arete list` *effectuant le tri d'une liste d'arêtes par ordre de poids croissant.*

Question 9 *Éxécuter (sur un bout de papier) l'algorithme de Kruskal sur le graphe de la* FIG. *1 pour trouver un arbre couvrant de poids minimal.*

Question 10 *A-t-on unicité de l'arbre couvrant de poids minimal ? Donner un arbre de même poids que celui de la question précédente, mais différant d'une arête de celui que vous avez trouvé.*

Question 11 *Prouver le lemme suivant :*

> ***Lemme 1*** *Si F_1 et F_2 sont deux sous-arbres couvrants et si $\alpha \in F_1, \alpha \notin F_2$, alors il existe une arête $\beta \in F_2$ telle que $F_1 \backslash \{\alpha\} \cup \{\beta\}$ soit un sous-arbre couvrant.*

Question 12 *En déduire une preuve de la validité de l'algorithme de Kruskal.*

Question 13 *Modifier le programme de la question 7 pour qu'il donne un arbre couvrant de poids minimal. Comment évolue la complexité ?*
(`spanningtree_Kruskal : graphe → graphe`*).*

4 Minimalité

Question 14 *Montrer que si les poids des arêtes sont distincts deux à deux, il n'existe qu'un seul abre recouvrant minimal.*

Question 15 *Soit $H = (F, W)$ un arbre recouvrant minimal de $G = (E, V)$. Montrer qu'il existe $a \in W$ et $b \in E \backslash W$ telles que $(F, V \cup \{b\} \backslash \{a\})$ Soit un arbre recouvrant de G de poids minimal parmi les arbres recouvrants qui ne sont pas de poids minimal.*
On supposera dans la fin de cette section que les arêtes sont de poids deux à deux distincts.

Question 16 *Déduire de ce qui précède un algorithme construisant un arbre G de poids minimal parmi les non minimaux. Complexité de cet algorithme ?*

Question 17 *A-t-on unicité de l'arbre de poids minimal parmi les non-minimaux ?*

5 Algorithme de Prim

▶ Dans cette partie, on va plutôt tenter de partir d'un graphe vide, auquel on ajoute progressivement des arêtes du graphe initial en conservant la connexité.

▶ On construit ici un sous-graphe connexe que l'on fait croître.

Algorithme 1 (Prim)

1. *$F \leftarrow \emptyset$ (ensemble d'arêtes, qui définit un sous-arbre)*
2. *Tant que F n'est pas couvrant, lui rajouter une des arêtes de poids minimal qui le relie à un sommet extérieur.*

Question 18 *Exécuter (à la main) l'algorithme de Prim sur le graphe de la* FIG. *1.*

Question 19 *Démontrer que l'algorithme de Prim renvoie bien un arbre couvrant minimal.*

Question 20 *Implémenter l'algorithme de Prim. On écrira tout d'abord une fonction* `poids` *qui renvoie le poids d'un graphe passé en paramètre, puis une fonction* `cherche_areteP` *qui prend en arguments une liste de sommets et une liste d'arêtes, et qui renvoie la première arête de la liste qui ne relie pas deux sommets cette première liste.*