Examen d'Algorithmique - Maitrise d'Informatique

01 Fevrier 2002

Durée: 3 heures.
Documents: Seuls sont autorisés les notes manuscrites de cours et TD.
NB: Toutes les réponses doivent être clairement justifiées.

Exercice 1

On considère des graphes pondérés orientés où les poids sont des entiers (dans $\mathbb{Z}$). On veut résoudre les problèmes d'accessibilité suivants:

> Etant donné un graphe G, deux sommets s et s' dans G, et un intervalle I (dont les bornes sont dans $\mathbb{Z} \cup \{-\infty, +\infty\}$),

(Π_1) : décider s'il existe un chemin dans G allant de s à s', dont le poids est un entier dans I.

(Π_2) : décider s'il existe un chemin dans G allant de s à s', dont le poids de *chaque préfixe* est dans I.

Question 1 Résoudre (Π_1) et (Π_2) pour les couples de sommets (s_1, s_2) et (s_1, s_3) dans le graphe de la figure 1, et les intervalles $[0, +\infty]$, $[-\infty, 1]$, $[5, +\infty]$, $[0, 2]$.

Question 2 Donner un algorithme pour résoudre le problème (Π_1) dans le cas où l'intervalle I est de la forme $]-\infty, n]$ où $n \in \mathbb{Z}$. Considérer ensuite le même problème dans le cas où I est de la forme $[n, +\infty[$.

Question 3 Même question que la précédente en considérant le problème (Π_2).

Question 4 Donner un algorithme pour résoudre le problème (Π_2) pour un intervalle I quelconque. Indication: mémoriser le poids du chemin menant de la source au sommet courant (un sommet peut être atteint par plusieurs, voire une infinité de chemins).

On considère maintenant des *graphes simples* qui sont de la forme représentée dans la figure 2. Dans ces graphes les sommets s et s' sont respectivement la seule source (sommet sans arcs entrants) et le seul puit (sommet sans arcs sortants). Un graphe simple est constitué d'un chemin $s, s_1, \cdots, s_m, s'$ et de circuits simples attachés à chacun des sommets s_i. Pour chaque sommet s_i, il existe n_i circuits simples partageant le sommet s_i (et uniquement ce sommet). Le nombre n_i est un entier quelconque pouvant être nul. Ce nombre peut varier d'un sommet s_i à un autre.

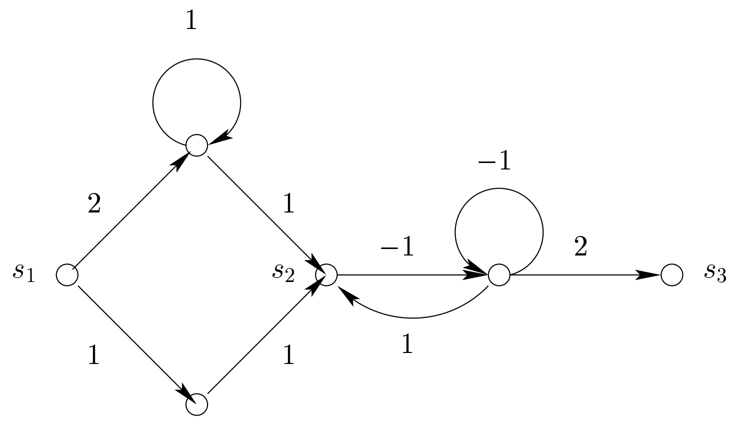

Figure 1: Exemple

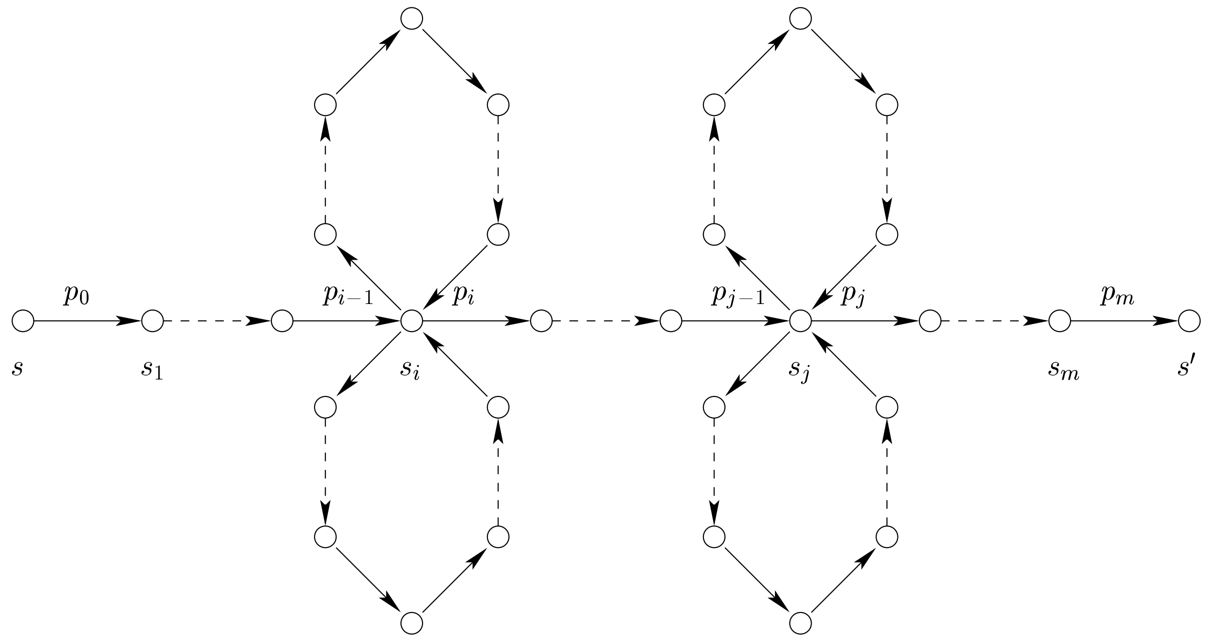

Figure 2: Graphe Simple

Soit $X = \{x_1, \dots, x_n\}$ un ensemble de variables à valeurs dans $I\!N$. Une contrainte arithmétique linéaire sur X est une formule de la forme

$$b_i \leq \sum_{i=1}^{n} a_i x_i \leq c_i$$

où et les a_i sont des constantes entières (dans $\mathbb{Z}$), les b_i sont dans $\mathbb{Z} \cup \{-\infty\}$, et les c_i sont dans $\mathbb{Z} \cup \{+\infty\}$. On admet qu'il existe un algorithme permettant de décider si une contrainte linéaire est satisfaisable (il existe une valuation des variables x_i pour laquelle la contrainte est vraie).

Question 5 Proposer un algorithme qui résoud le problème (Π_1) (avec I quelconque) dans le cas des graphes simples.

Exercice 2

On étudie dans cet exercice un algorithme parallèle permettant de calculer les composantes fortement connexes d'un graphe *non orienté* $G = (S, A)$, où S est un ensemble de sommets et $A \subseteq S \times S$ est un ensemble d'arcs. On suppose en fait que S est un ensemble d'entiers naturels, et que A correspond à une relation symétrique entre les sommets, c'est-à-dire que $(u, v) \in A$ si et seulement si $(v, u) \in A$.

L'algorithme que l'on considère utilise le modèle CRCW *arbitraire* (et non pas le modèle common-CRCW vu en cours). Dans ce modèle, si plusieurs écritures concurrentes ont lieu, on suppose que l'une quelconque parmi elles réussit.

L'algorithme considéré utilise $|S+A|$ processeurs. Les composantes connexes sont représentées par une fôret d'arbres où chaque sommet s pointe vers son père $p[s]$. On suppose de plus que chaque racine pointe vers elle même.

Initialement, on pose $p[s] = s$ pour tout sommet s (on a une fôret où chaque arbre consiste en un seul sommet qui pointe vers lui même). Le but de l'algorithme est de transformer (itérativement) cette fôret de telle sorte qu'à la fin de son exécution, on ait pour toute paire de sommets $s_1, s_2 \in S$, $p[s_1] = p[s_2]$ si et seulement si s_1 et s_2 appartiennent à la même composante connexe de G (il existe un chemin dans G allant de s_1 à s_2).

On appelle *étoile* un arbre dans la fôret tel que, pour tous sommets s_1 et s_2 dans cet arbre, on a $p[s_1] = p[s_2]$. Autrement dit, une étoile est un arbre où tous les sommets (y compris la racine) pointent directement vers la racine.

Question 1 Ecrire un algorithme $\text{E}\textsc{toile}(G)$ de type common-CRCW, qui construit en $O(1)$ un tableau *étoile* tel que, pour tout sommet s, *étoile*$[s] = \mathit{vrai}$ si et seulement si s appartient à une étoile.

L'algorithme des composantes connexes utilise deux procédures Accrocher et Sauter dont le pseudo-code est donné ci-dessous.

Accrocher(G):

1. Etoile(G)
2. pour tout arc $(u, v) \in A$ faire en parallèle
3. si *étoile*$[u]$ et $p[u] > p[v]$ alors $p[p[u]] := p[v]$

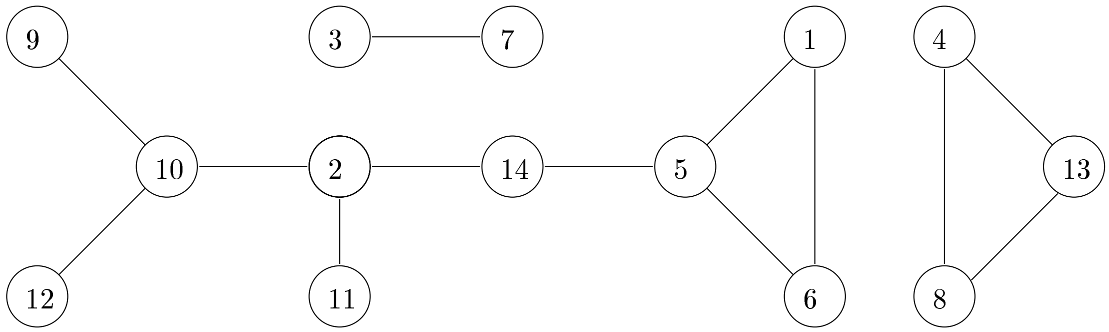

Figure 3: Exemple de graphe non orienté

4. Etoile(G)
5. pour tout arc $(u, v) \in A$ faire en parallèle
6. si *étoile*$[u]$ et $p[u] \neq p[v]$ alors $p[p[u]] := p[v]$

Sauter(G):

1. pour tout sommet $s \in S$ faire en parallèle $p[s] := p[p[s]]$

L'algorithme consiste alors à faire un premier appel à Accrocher, puis ensuite, d'appeler itérativement Accrocher suivie de Sauter, jusqu'à ce qu'aucun pointeur ne soit modifié par Sauter.

Question 2 Appliquer l'algorithme à l'exemple de la figure 3. Pourquoi cet algorithme est CRCW *arbitraire* ?

Question 3 Démontrer qu'après chaque itération (appel de Accrocher suivie de Sauter), le tableau p définit bien une fôret telle que tous les sommets d'un même arbre appartiennent à la même composante fortement connexe de G.

Question 4 Démontrer qu'après chaque itération, si r est la racine d'une étoile, alors pour tout sommet s de la composante connexe de r dans G, on a $p[s] = r$.

Question 5 Démontrer que l'algorithme est correct: il termine, et quand il termine, on a $p[s_1] = p[s_2]$ si et seulement si s_1 et s_2 appartiennent à la même composante connexe.