Algorithmique

Magistère STIC

Examen du 16 novembre 2006

durée 3 heures

Les notes de cours et de TD sont autorisées, mais pas les livres.

Les exercices sont indépendants.

Toutes les réponses devront être correctement justifiées. La rigueur des raisonnements, la clarté des explications, et la qualité de la présentation influeront sensiblement sur la note.

1 Arbres AVL (Adelson-Velskii et Landis, 1962)

Un AVL est un arbre binaire de recherche (ABR) tel que pour chaque nœud de l'arbre, la différence de hauteur entre le sous-arbre gauche et le sous-arbre droit est d'au plus 1.

a) Soit n le nombre de nœuds et h la hauteur d'un AVL (la hauteur d'un arbre réduit à sa racine est 1). Montrer que $F_{h+2} - 1 \leq n \leq 2^h - 1$ où F_k est le k-ième nombre de Fibonacci avec $F_0 = 0$, $F_1 = 1$ et $F_{k+2} = F_{k+1} + F_k$ pour $k \geq 0$. Montrer que ces bornes sont atteintes.

Montrer que pour $k \geq 0$ on a $F_{k+2} \geq \phi^k$ où $\phi = (1 + \sqrt{5})/2$.
En déduire que $\log_2(n+1) \leq h \leq \log_\phi(n+1)$ et que la recherche dans un AVL se fait au pire en temps $O(\log(n))$.

Pour les algorithmes, un nœud x d'un AVL sera représenté par une structure comportant

- une clé, notée $x \cdot \text{cle}$ d'un type totalement ordonné (par exemple de type entier);
- un sous-arbre gauche, noté $x \cdot g$ et un sous-arbre droit, noté $x \cdot d$;
- la hauteur de l'arbre, notée $x \cdot h$.

L'arbre vide est noté `NULL`. On notera $h(x)$ la hauteur d'un arbre x avec $h(\texttt{NULL}) = 0$. On notera $n(x)$ le nombre de nœuds d'un arbre x.

b) Le but de cette question est d'écrire une procédure `équilibrer`(x) pour transformer en AVL en temps constant un ABR de racine x en supposant que ses deux sous-arbres sont des AVL et que la différence de hauteur entre les deux sous-arbres est d'au plus 2.

Proposer un rééquilibrage dans le cas où $h(a \cdot g) - h(a \cdot d) = 2$. On pourra distinguer les cas $h(a \cdot g \cdot g) \geq h(a \cdot g \cdot d)$ et $h(a \cdot g \cdot g) < h(a \cdot g \cdot d)$. Illustrer les transformations sur des dessins. Écrire la procédure `équilibrer`(x).

c) Écrire un algorithme pour insérer une clé c dans un AVL x en temps au pire $O(1+h(x))$. Si x' est l'arbre obtenu, comparer $h(x')$ et $h(x)$. Justifier la correction de l'algorithme (on ne demande pas une preuve de programme). Montrer que cet algorithme engendre au plus un rééquilibrage.

d) Écrire un algorithme pour supprimer une clé c dans un AVL x en temps au pire $O(1 + h(x))$. Justifier la correction de l'algorithme (on ne demande pas une preuve de programme). Combien de rééquilibrages peuvent être nécessaires ?

e) Écrire un algorithme qui réalise la fusion de deux AVL x et y et d'une clé c en supposant que toutes les clés de x sont strictement inférieures à c et que toutes les clés de y sont strictement supérieures à c. Cet algorithme devra fonctionner en temps $O(1+|h(x)-h(y)|)$. Justifier.

f) Écrire un algorithme qui réalise la scission d'un AVL x en deux AVL y et z contenant respectivement les clés de x inférieures ou égales à c pour y et strictement supérieures à c pour z. Cet algorithme devra fonctionner en temps $O(1 + |h(x)|)$. Justifier.

2 Recherche de motif et automates

Dans cet exercice, l'alphabet A est fixé. Le mot vide est noté ε.

On note $t = t[1] \cdots t[n]$ le texte de longueur n et $x = x[1] \cdots x[m]$ le motif de longueur m. On notera $t[i \cdot\cdot j] = t[i] \cdots t[j]$ le facteur de t commençant à la position i et se terminant à la position j. Par convention, ce facteur est vide si $j < 1$ ou $i > n$ ou $j < i$.

a) On suppose donné un automate déterministe complet $\mathcal{A} = (Q,A,\delta,q_0,F)$ qui reconnaît le langage $A^* \cdot \{x\}$ des mots qui se terminent par x. Écrire un algorithme, basé sur l'automate $\mathcal{A}$, qui affiche toutes les positions des occurrences de x dans t en temps $O(|t|)$.

Le but des questions **(b)** à **(g)** est de calculer l'automate minimal du langage $A^* \cdot \{x\}$ en temps $O(|A| \cdot |x|)$. On rappelle que la fonction de transition δ s'étend à A^* en posant pour tout $q \in Q$, $\delta(q,\varepsilon) = q$ et $\delta(q,va) = \delta(\delta(q,v),a)$ pour $v \in A^*$ et $a \in A$. On rappelle aussi que l'automate $\mathcal{A}$ est minimal si pour tous $p,q \in Q$ avec $p \neq q$ il existe un mot $v \in A^*$ tel que $\delta(p,v) \in F$ et $\delta(q,v) \notin F$ ou le contraire.

On utilisera les notions de *bord* et de *bord disjoint* définies en cours. On note $\mathrm{Bord}(v)$ le plus long bord de v et si v est un préfixe de x on note $\mathrm{BD}_x(v)$ le plus long bord de v disjoint dans x (on pose $\mathrm{BD}_x(v) = \perp$ si v n'a pas de bord disjoint dans x).

Le motif x étant fixé, pour $1 \leq j \leq m$, on note $b(j) = |\mathrm{Bord}(x[1 \cdot\cdot j])|$ et $bd(j) = |\mathrm{BD}_x(x[1 \cdot\cdot j])|$ avec la convention $bd(j) = -1$ si $x[1 \cdot\cdot j]$ n'a pas de bord disjoint dans x. On rappelle qu'on peut calculer en temps $O(m)$ les fonctions (tableaux) b et bd.

Si $v \in A^*$, on note $\mathrm{Pref}(v)$ l'ensemble des préfixes de v. On note aussi $\mathrm{Suff}(v)$ l'ensemble des suffixes de v. Noter que $|\mathrm{Pref}(v)| = |\mathrm{Suff}(v)| = |v| + 1$. Le motif x étant toujours fixé, on définit la fonction $f : A^* \to \mathrm{Pref}(x)$ par $f(v) = \max(\mathrm{Pref}(x) \cap \mathrm{Suff}(v))$.

b) Soit $v \in A^*$ et $a \in A$. Montrer que $f(va) = f(f(v)a)$.

c) Soit $v \in \mathrm{Pref}(x)$ et $a \in A$. Montrer que $f(va) = \begin{cases} va & \text{si } va \in \mathrm{Pref}(x) \\ \mathrm{Bord}(va) & \text{sinon.} \end{cases}$

d) Soit $v \in \mathrm{Pref}(x)$ et $a \in A$. Montrer que $f(va) = \begin{cases} va & \text{si } va \in \mathrm{Pref}(x) \\ f(\mathrm{Bord}(v)a) & \text{sinon.} \end{cases}$

e) Soit $v \in \mathrm{Pref}(x)$ et $a \in A$. Montrer que $f(va) = \begin{cases} va & \text{si } va \in \mathrm{Pref}(x) \\ f(\mathrm{BD}_x(v)a) & \text{si } \mathrm{BD}_x(v) \neq \perp \\ \varepsilon & \text{sinon.} \end{cases}$

On définit l'automate des occurrences $\mathcal{A}_x = (\mathrm{Pref}(x),A,\delta,\varepsilon,\{x\})$ où l'ensemble des états est $\mathrm{Pref}(x)$, l'état initial est le mot vide ε, l'état final est le mot x et la fonction de transition est définie par $\delta(v,a) = f(va)$ pour $v \in \mathrm{Pref}(x)$.

f) Montrer que $\mathcal{A}_x$ est un automate déterministe complet qui reconnaît $A^* \cdot \{x\}$. Montrer que A_x est minimal.

g) Écrire un algorithme qui calcule la fonction $\tilde{\delta} : \{0,\ldots,m\} \times A \to \{0,\ldots,m\}$ définie par $\tilde{\delta}(i,a) = |\delta(x[1 \cdot\cdot i],a)|$ en temps $O(|A| \cdot |x|)$.
Remarque : la fonction $\tilde{\delta}$ code la fonction δ.

Dans la suite, on veut éviter le facteur induit par l'alphabet dans le codage et le calcul de l'automate $\mathcal{A}_x$. On cherche donc un codage de δ que l'on puisse calculer en temps $O(|x|)$ (et qui utilise donc un espace $O(|x|)$).

Une transition $p \xrightarrow{a} q$ de $\mathcal{A}_x$ est dite *passive* si $q = \varepsilon$ et active sinon. Une transition active est une *flèche avant* si $q = pa$ et c'est une *flèche arrière* sinon, i.e., si $\varepsilon \neq q \neq pa$. Clairement, il y a donc m flèches avant dans $\mathcal{A}_x$. On va montrer qu'il y a au plus m flèches arrière.

On dit qu'un entier $p > 0$ est période d'un mot w si $w[i] = w[i+p]$ pour tout $1 \leq i \leq |w|-p$. On remarque que si $w \neq \varepsilon$ alors $|w|$ est période de w. On note $\mathrm{per}(w)$ la plus petite période de w.

h) Soit $w \neq \varepsilon$. Montrer que $|w| = \mathrm{per}(w) + |\mathrm{Bord}(w)|$.

i) Soient $u \xrightarrow{a} f(ua)$ et $v \xrightarrow{b} f(vb)$ deux flèches arrière. Montrer que $ua \neq vb$ implique $\mathrm{per}(ua) \neq \mathrm{per}(vb)$. En déduire que l'automate $\mathcal{A}_x$ comporte au maximum m flèches arrière.

Pour chaque entier i avec $0 \leq i \leq m$, soit $\bar{\delta}(i)$ la liste des flèches actives issues de $x[1 \cdot\cdot i]$ ordonnée par longueurs décroissantes. Chaque flèche est représentée par un couple (a,j) où j est la longueur de l'état d'arrivée : $x[1 \cdot\cdot i] \xrightarrow{a} x[1 \cdot\cdot j]$. Noter que si (a,j) est dans la liste $\bar{\delta}(i)$ alors $a = x[j]$ puisque la flèche représentée est active. Il n'est donc pas utile en fait de mémoriser les lettres. Si $i < m$, le premier élément de la liste $\bar{\delta}(i)$ est $(x[i+1],i+1)$.

j) Expliquer pourquoi $\bar{\delta}$ est un codage de $\tilde{\delta}$ (et donc de δ). Soit $i \in \{1,\ldots,m\}$ tel que $bd(i) \neq -1$. Comparer $\bar{\delta}(i)$ et $\bar{\delta}(bd(i))$. En déduire un algorithme qui calcule $\bar{\delta}$ en temps $O(|x|)$. Justifier cette complexité.

k) Écrire l'algorithme de recherche de motif qui s'en déduit. Cet algorithme est appelé algorithme de Simon.

l) Montrer que si $v \xrightarrow{a} ua$ est une flèche arrière alors il existe k tel que $u = \mathrm{BD}_x^k(v)$. En déduire que l'algorithme de Simon ne fait pas plus de comparaisons de caractères que l'algorithme de Knuth, Morris et Pratt. Quelle est la complexité de l'algorithme de Simon ?