IFT2015 automne 2011 — Devoir 6

Miklós Csűrös

28 novembre 2011

À remettre avant 20 :15 le 5 décembre par email (à csuros@iro...). Remettez un seul fichier PDF. Travaillez seul.

6.1 Déploiement (15 points)

On splaye à la main (5 points). Dessiner l'ABR complet de hauteur 3 avec clés $\{1, 2, \dots, 15\}$. Montrer le résultat du déploiement (*splaying*) après chaque opération dans la séquence

search(12), search(1), delete(8).

Séquence alternante (10 points). Considérez la séquence d'opérations

insert(1), insert(n), insert(2), insert($n - 1$), insert(3), $\dots$, insert($\lceil (n + 1)/2 \rceil$),

exécutées dans un arbre *splay* initialement vide. Démontrez que (1) le coût amorti des opérations est $o(\log n)$ dans cette séquence, et que (2) la hauteur de l'arbre est $\Theta(i)$ après le i -ème insertion. **Indice.** Observez comment la structure de l'arbre change avec les opérations, énoncez un lemme, et démontrez-le par induction.

Morale : le coût amorti $O(\log n)$ n'implique ni que le coût amorti est $\Theta(\log n)$, ni que la hauteur moyenne de l'arbre est $O(\log n)$.

6.2 Arbres rouge, rouge et noir (15+5 points)

Rudolf Bayer a inventé une structure de données sous le nom «symmetric binary B tree» qui est équivalente à l'arbre rouge et noir. Il mentionne une généralisation qu'on considère ici. Un *arbre rouge-rouge-noir* (RRN) est un arbre binaire de recherche. Chaque nœud de l'arbre a un *rrang*. (Ne pas confondre avec le rang des arbres RN.) Les rrangs des nœuds satisfont les propriétés suivantes.

1. Pour chaque nœud x excepté la racine,

$$\text{rrang}(x) \leq \text{rrang}(x.\text{parent}) \leq \text{rrang}(x) + 1.$$

2. Pour chaque nœud x avec un arrière-grand-parent $y = x.\text{parent}.\text{parent}.\text{parent}$,

$$\text{rrang}(x) < \text{rrang}(y).$$

3. Pour chaque nœud externe x on a $\text{rrang}(x) = 0$ et $\text{rrang}(x.\text{parent}) = 1$.

On colorie les nœuds soit rouge soit noir : le nœud x est rouge si et seulement s'il n'est pas la racine et $\text{rrang}(x) = \text{rrang}(x.\text{parent})$. Tous les autres nœuds sont noirs.

Coloriage et rrang (7 points) Complétez l'énoncé du théorème 6.2.1 et démontrez qu'il est vrai en utilisant les propriétés de rrang. (Mettez «rouge» ou «noir» à tout)

Théorème 6.2.1. *Le coloriage d'un arbre rouge-rouge-noir est tel que*

- (i) chaque nœud externe est colorié par .
- (ii) si le parent d'un nœud est , alors son grand-parent (s'il existe) est .
- (iii) chaque chemin reliant un nœud à un nœud externe dans son sous-arbre contient le même nombre de nœuds .

La hauteur d'un arbre rouge-rouge-noir (8 points) On veut démontrer que la hauteur d'un arbre rouge-rouge-noir avec n nœuds internes est $O(\log n)$. Il n'est pas trop difficile de démontrer le théorème suivant.

Théorème 6.2.2. *Pour tout nœud x d'un arbre rouge-rouge-noir, le nombre de nœuds internes dans le sous-arbre enraciné à x est au moins $2^{\text{rrang}(x)} - 1$.*

Vous pouvez utiliser le théorème 6.2.2 sans preuve pour démontrer **b** ici.

a (4 points) Démontrez que

$$\text{hauteur}(x) \leq 3 \cdot \text{rrang}(x) \quad (\star)$$

pour chaque nœud x . (Vous avez besoin du théorème 6.2.1 dans la preuve.)

b (4 points) Utilisez le théorème 6.2.2 et l'équation $(\star)$ pour démontrer que la hauteur d'un arbre RRN avec n nœuds internes est borné par $c \lg(n + 1)$ — remplacez c par un nombre aussi petit que possible.

Comparez ce résultat avec la borne sur la hauteur des arbres RN.

c (5 points boni) Donnez une preuve formelle au théorème 6.2.2.

6.3♥ TAD Partition (10 points boni)

Proposez une structure efficace pour implanter un type abstrait qui représente la partition des nombres non-négatifs en intervalles¹. Une partition est l'ensemble $\{[0, x_1), [x_1, x_2), \dots, [x_m, \infty)\}$ avec $x_0 = 0 < x_1 < x_2 < \dots < x_m$ et un $m = 0, 1 \dots$, quelconque. L'interface du type comprend les opérations suivantes avec un paramètre $x > 0$:

recherche(x)	retourne l'intervalle $[x_{i-1}, x_i)$ qui contient x
fusion(x)	recherche $x \in [x_{k-1}, x_k)$, et remplace les intervalles $[x_{k-1}, x_k)$ et $[x_k, x_{k+1})$ par $[x_{k-1}, x_{k+1})$
tranche(x)	recherche $x \in [x_{k-1}, x_k)$, et remplace l'intervalle $[x_{k-1}, x_k)$ par $[x_{k-1}, x)$ et $[x, x_k)$

Votre structure doit supporter les trois opérations en temps $O(\log m)$.

¹L'intervalle $[a, b)$ est l'ensemble de valeurs réelles $\{x : a \leq x < b\}$ où $a \leq b$. Il est permis d'avoir $b = \infty$: c'est interprété comme $[a, \infty) = \{x : a \leq x\}$. Notez que $[a, a)$ est l'ensemble vide.