UPMC PARISUNIVERSITAS

Algorithmique Avancée
Examen réparti du 9 novembre 2009

Version du 10 novembre 2009

Examen de 2 heures, seuls sont autorisés le polycopié de cours, ainsi qu'une feuille-double rédigée personnellement ("anti-sèche"). Le barême est donné à titre indicatif.

Exercice 1 – Questions rapides [4 points]

1. **Files de priorités.**
   (a) On considère une file binomiale $\mathcal{B}$ composée d'arbres binomiaux B_6, B_4, B_3, B_1, B_0 et une file binomiale $\mathcal{B}'$ composée d'arbres binomiaux B_5, B_2, B_1, B_0. Donner la composition de la file binomiale $\mathcal{B}''$ obtenue par fusion des files $\mathcal{B}$ et $\mathcal{B}'$.
   (b) On considère une file binomiale relâchée $\mathcal{R}$ composée d'arbres binomiaux B_3, B_3, B_3, B_2, B_1, B_1, B_1, B_0. Donner la composition de la file binomiale $\mathcal{B}$ obtenue par consolidation de la file $\mathcal{R}$.
   (c) On considère une file de Fibonacci $\mathcal{F}$ dont les racines sont de degrés 1, 2 et 4. Quelle est la taille maximale de $\mathcal{F}$? sa taille minimale ?
2. **Arbres 2-3-4.** On considère A, un arbre 2-3-4 ayant 4 niveaux de nœuds, et on insère une nouvelle valeur v dans A, en faisant les éclatements à la descente. Soit x_1, x_2, x_3, x_4 les nœuds du chemin suivi pour insérer v (x_1 est la racine, x_4 est une feuille). Calculer la variation du nombre de 2-nœuds, du nombre de 3-nœuds et du nombre de 4-nœuds dans chacun des cas suivants :
   (a) x_1, x_2, x_3, x_4 sont des 4-nœuds ;
   (b) x_1, x_3 sont des 2-nœuds et x_2, x_4 sont des 4-nœuds ;
   (c) x_1, x_3 sont des 3-nœuds et x_2, x_4 sont des 4-nœuds.

Exercice 2 – Coût amorti de l'empilement dans une *multipile* [5 points]

Une *multipile* est une séquence infinie de piles P_0, P_1, P_2, ..., où la i-ième pile P_i peut contenir jusqu'à 3^i éléments.

Lorsqu'on tente d'empiler un élément sur une pile P_i dont la capacité a été atteinte, on transfère d'abord tous les éléments de la pile P_i vers la pile P_{i+1} pour faire de la place dans P_i. Si P_{i+1} n'a pas suffisamment de place pour contenir les éléments de P_i, alors on vide d'abord P_{i+1}, etc.

On considère que déplacer un élément d'une pile vers une autre se fait en temps $O(1)$.

1. Montrer que, dans le pire des cas, la complexité de l'empilement d'un élément sur une multipile contenant n éléments (c'est-à-dire l'empilement de cet élément sur la pile P_0 de la multipile) est en $O(n)$.
2. Montrer que le coût amorti de l'opération d'empilement sur la multipile contenant n éléments est en $O(\log n)$.
   (a) avec la méthode par agrégat ;
   (b) avec la méthode du potentiel (beaucoup plus difficile).

Exercice 3 – Hachage universel [6 points]

On suppose que les clefs sont des suites de k bits, $\mathcal{C} = \{0,1\}^k$, et que la taille de table de hachage est une puissance de 2, $m = 2^p$. On définit une fonction de hachage par la donnée d'une matrice M de dimension $p \times k$, dont les entrées sont binaires (0 ou 1), et on associe à toute clef $x \in \{0,1\}^k$, l'adresse dont la représentation binaire est le vecteur résultant du produit de la matrice M par le vecteur x, calculé modulo 2 : $h_M(x) = M \cdot x \,[\mathrm{mod}\, 2]$.

Par exemple, si

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{et} \quad x = (1,0,0,1)^T$$

alors $h_M(x) = (0,1)^T$, et l'adresse de x dans la table est 1.
Remarque : on note v^T le vecteur ligne transposé du vecteur colonne v.

1. On donne
$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$
Calculer les valeurs de hachage de $x = (1,0,1,1)^T$, de $y = (0,0,1,0)^T$ et de $z = (1,0,1,0)^T$, pour M.
2. Donner un algorithme qui calcule $h_M(x)$ en ajoutant les colonnes de M qui correspondent aux bits à 1 de x (noter que l'addition modulo 2 correspond à l'opérateur booléen OU-exclusif).
3. On considère l'ensemble $\mathcal{H} = \{h_M : \mathcal{C} \to \{0,...,m-1\}$, avec $M \in \{0,1\}^{p \times k}\}$. Quel est le cardinal de $\mathcal{H}$?
4. Montrer que pour tout couple de clefs différentes x et y (on suppose que le j-ème bit de x est différent du j-ème bit de y), quelles que soient les valeurs des colonnes M_i, pour $i \neq j$, de la matrice M, il existe une unique manière de déterminer la colonne M_j, de telle sorte que $h_M(x) = h_M(y)$.
5. En déduire que l'ensemble $\mathcal{H}$ est universel.

Exercice 4 – Arbres doubles [5 points]

Un *treap* est une structure d'arbre binaire, où chaque nœud x possède une *clef*, notée $\mathrm{clef}(x)$, à laquelle est associée une *priorité* fixe, notée $\mathrm{prio}(x)$. Si x est un nœud qui a deux fils y et z, respectivement gauche et droit (l'un, l'autre ou les deux peuvent éventuellement ne pas exister), alors le *treap* vérifie :

- la propriété d'arbre binaire de recherche, $\mathrm{clef}(x) > \mathrm{clef}(y)$ et $\mathrm{clef}(x) < \mathrm{clef}(z)$;
- la propriété de tas-min, $\mathrm{prio}(x) < \mathrm{prio}(y)$ et $\mathrm{prio}(x) < \mathrm{prio}(z)$.

1. Montrer par récurrence forte que, lorsque les priorités sont toutes distinctes, il existe un *unique* arbre binaire qui vérifie les deux propriétés d'arbre binaire de recherche et de tas-min.
2. En utilisant les primitives sur les arbres binaires de recherche, auxquelles on ajoute des primitives de rotation, donner l'algorithme INSERTTREAP(c_x, p_x, T) qui insère la clef c_x associée à la priorité p_x dans le treap T.
Rappel : l'arbre résultant doit respecter la propriété d'arbre binaire de recherche, et celle de tas-min.

La figure ci-dessous représente les différentes étapes (insertion de la clef aux feuilles d'un ABR puis rotations pour tenir compte de la priorité) pour insérer la clef K de priorité 2, dans l'arbre initial T.