Algorithmique Avancée — Examen Réparti 2

Les seuls documents autorisés sont les polys de cours, ainsi que la copie personnelle. Le barème donné est indicatif.

1 Structures arborescentes. Hachage

Exercice 1 *: Questions rapides [2 points]*

Question 1 Pour un arbre 2-3-4 contenant n clefs, la complexité (en nombre de nœuds traversés) du calcul du plus petit élément est au pire en :

A. $\Theta(n)$ **B.** $\Theta(\log n)$ **C.** $\Theta(1)$

Décrire succinctement un algorithme réalisant ce calcul.

Question 2 Dans un AVL, dire quel parcours donne les clefs dans l'ordre croissant (**justifier la réponse**) :

A. parcours infixe **B.** parcours préfixe **C.** parcours suffixe

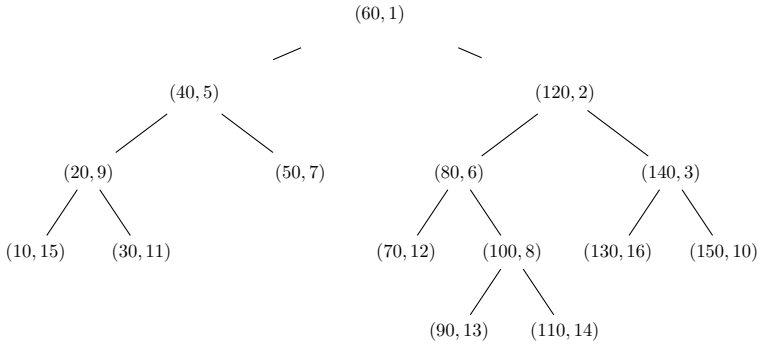
Exercice 2 *: Arbres de recherche et files de priorité [6 points]*

Dans un arbre binaire de recherche, les nœuds ont pour valeur des clés et le parcours infixe de l'arbre donne la liste des clés en ordre croissant. Dans cet exercice, on considère des arbres binaires de recherche dans lesquels chaque nœud contient, en plus de sa clé, une priorité : la priorité d'un nœud est toujours inférieure à celle de ses fils. De tels arbres sont appelés ici *abr-prior*. Dans la suite on supposera toujours que dans un abr-prior, toutes les clés sont deux à deux distinctes et toutes les priorités sont deux à deux distinctes.

Question 1 Étant donné un ensemble de couples (clé, priorité), montrer, par récurrence sur le nombre de couples, qu'il existe un unique abr-prior pour cet ensemble de couples.

Par exemple, l'arbre AP ci-dessous est l'abr-prior associé à l'ensemble :

{(40,5), (20,9), (10,15), (30,11), (50,7), (120,2), (80,6), (130,16), (70,12), (100,8), (60,1), (90,13), (140,3), (110,14), (150,10)}



Question 2 La recherche d'une clé dans un abr-prior se fait exactement comme dans un arbre binaire de recherche. L'insertion d'un couple (c, p) suit l'algorithme d'insertion aux feuilles dans un arbre binaire de recherche, selon la valeur de la clé. Lorsque le couple est ainsi inséré, on le fait remonter, par rotations, vers la racine de l'arbre, jusqu'à ce que l'ordre des priorités soit respecté.

1. Montrer l'insertion du couple $(25, 4)$ dans l'arbre ci-dessus.
2. Donner le pseudo-code de l'algorithme d'insertion (utilisez les primitives sur les arbres binaires vues en cours, les rotations gauche RG et droite RD et les fonctions *cle* et *prio* permettant de récupérer la clé et la priorité d'un couple). Expliquer pourquoi cet algorithme est correct.
3. Analyser la complexité de cet algorithme.

Question 3

1. Décrire un algorithme de fusion de deux arbres abr-prior A_1 et A_2 tels que toutes les clés de A_1 sont strictement inférieures à toutes les clés de A_2.

2. En déduire un algorithme de suppression de la racine dans un abr-prior et l'appliquer à l'arbre AP.

Exercice 3 *: Familles de fonctions de hachage [6 points]*

Soit $\mathcal{H}$ une famille de fonctions de hachage dans laquelle chaque fonction $h \in \mathcal{H}$ envoie l'univers de clefs U dans l'intervalle d'entiers $[0, 1, \ldots, m-1]$. On rappelle que :

- $\mathcal{H}$ est **universelle** ssi, pour toutes clefs y, z telles que $y \neq z$: $|\{h \in \mathcal{H}; h(y) = h(z)\}| = \frac{|\mathcal{H}|}{m}$;
- $\mathcal{H}$ est **2-universelle** ssi, pour toutes clefs y, z telles que $y \neq z$ et pour toutes valeurs u, v dans $[0, 1, \ldots, m-1]$: $|\{h \in \mathcal{H}; h(y) = u, h(z) = v\}| = \frac{|\mathcal{H}|}{m^2}$.
- $\mathcal{H}$ est **1-universelle** ssi, pour toute clef y et pour toute valeur u dans $[0, 1, \ldots, m-1]$: $|\{h \in \mathcal{H}; h(y) = u\}| = \frac{|\mathcal{H}|}{m}$.

Question 1 On munit $\mathcal{H}$ de la probabilité uniforme, exprimer les définitions précédentes (famille universelle, 2-universelle, 1-universelle) en termes de probabilités.

Question 2 Montrer que, si $\mathcal{H}$ est une famille 2-universelle, alors $\mathcal{H}$ est une famille 1-universelle.

Dans la suite de l'exercice, l'univers U est l'ensemble des n-uplets de valeurs de $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, où p est premier ; autrement dit $U = (\mathbb{Z}_p)^n$. Soit $x = \langle x_0, \ldots, x_{n-1} \rangle \in U$. Pour tout n-uplet $a = \langle a_0, \ldots, a_{n-1} \rangle \in U$, définissons la fonction de hachage $h_a : U \to \mathbb{Z}_p$ par :

$$h_a(x) = \left(\sum_{j=0}^{n-1} a_j x_j \right) \bmod p.$$

Soit $\mathcal{H} = \{h_a \ ; \ a \in U\}$.

Question 3 Pour $p = 2$, $n = 3$ et $a = \langle 1, 0, 1 \rangle$, déterminer l'ensemble des triplets x tels que $h_a(x) = 0$.

Question 4 Montrer que $(a \neq b) \Rightarrow (h_a \neq h_b)$. En déduire que $|\mathcal{H}| = p^n$. La famille $\mathcal{H}$ est-elle égale à l'ensemble de toutes les fonctions de U dans $\mathbb{Z}_p$?

Question 5 Déterminer une clef x_0 pour laquelle toutes les fonctions de $\mathcal{H}$ produisent la même valeur.

Question 6 Montrer que $\mathcal{H}$ n'est pas 2-universelle.

Question 7 Montrer que $\mathcal{H}$ est universelle. On rappelle que, si p est premier, alors $\mathbb{Z}_p$ est un corps.

2 Géométrie algorithmique [6 points]

IMPORTANT : traiter les exercices de cette section sur une copie à part.

Sauf mention explicite du contraire, toute réponse concernant une analyse de complexité doit être appuyée par une preuve formelle la justifiant (par analyse directe, par analyse amortie, par induction, etc). Toute réponse sans preuve aura valeur nulle.

Exercice 4 *: Question de cours*

- Rappeler les I/O (arguments en entrée et valeur de retour) du filtrage Alk-Toussaint, le pseudo-code de ses étapes principales, ainsi que sa complexité.
- Rappeler les I/O du filtrage "tri par pixel", le pseudo-code de ses étapes principales, ainsi que sa complexité.

Exercice 5 *: Complexité de l'enveloppe convexe*

Nous voulons maintenant prouver que la complexité du calcul du contour positif de l'enveloppe convexe d'un ensemble de n points est en $\Theta(n \log n)$.

- Afin de prouver que la complexité mentionnée est en $O(n \log n)$, nous pouvons procéder par dichotomie. Dans un premier temps, donner le pseudo-code d'une fonction qui prend en entrée deux contours positifs convexes P_1, P_2 comprenant au total n points, et qui retourne en temps $O(n)$ le contour positif de l'enveloppe convexe de $P_1 \cup P_2$. Ensuite, utiliser la procédure précédente afin de donner le pseudo-code d'une fonction récursive qui prend en entrée une liste quelconque de n points et qui retourne en temps $O(n \log n)$ le contour positif de l'enveloppe convexe de ces points.
- Afin de prouver que la complexité mentionnée est en $\Omega(n \log n)$, supposer que l'on ait un algorithme en $o(n \log n)$ calculant le contour positif de l'enveloppe convexe, et prouver qu'il serait alors possible de trier une liste de n entiers en temps $o(n \log n)$, par une utilisation judicieuse du contour positif de certains points en 2D à définir.