**Algorithmique avancée – Examen de rattrapage**
**UPMC — Master d'Informatique —**
**Mai 2013 – durée 2h**

*Les seuls documents autorisés sont les polys de cours, ainsi que la copie double personnelle.*

# 1 Files de priorité [8 points]

Dans cet exercice, on considère les files de priorité représentées par des tas (on rappelle qu'un tas est un arbre binaire croissant dont toutes les feuilles sont situées au plus sur deux niveaux, les feuilles du niveau le plus bas étant calées à gauche).

1. Construire un tas $T$ d'étiquettes 1, 3, 5, 7, 9, 11, 15, 17, 19. En expliquant l'algorithme utilisé, insérer 2 dans le tas précédent, puis supprimer le minimum. Donner un encadrement de la taille d'un tas en fonction de sa hauteur (la racine du tas est à profondeur 0, et la hauteur est la profondeur maximale d'une feuille).
2. Avec cette structure de données, on peut ajouter un élément ou extraire le minimum en $O(\log n)$ comparaisons dans le cas le pire (où $n$ est la taille du tas). Expliquer brièvement cette assertion.
3. Montrer qu'il est impossible de construire une structure de données $S$ représentant une file de priorité, telle que, pour $S$, on ait à la fois le coût de la construction d'une file de $n$ éléments en O(n) et le coût de la suppression du minimum en $O(1)$.
4. Il s'agit maintenant de montrer que l'on peut extraire le minimum en coût amorti $O(1)$ comparaisons, tout en gardant l'ajout en coût amorti $O(\log n)$ comparaisons. On considère la fonction $\Phi$, qui à un tas $T$ associe
$$\Phi(T) = 2 \sum_{x \in T} p(x),$$
où $p(x)$ est la profondeur du nœud $x$ dans l'arbre $T$.
   (a) Montrer que $\Phi$ est une fonction de potentiel.
   (b) Montrer que le coût amorti de l'ajout d'un élément est en $O(\log n)$.
   (c) Montrer que le coût amorti de la suppression du minimum est en $O(1)$.

# 2 Arbres équilibrés [4 points]

On rappelle la définition de la hauteur $h$ d'un arbre binaire $T$ : si $T$ est réduit à 1 nœud alors $h(T) = 0$, et sinon $h(T) = 1 + \max\{h(T_1), h(T_2)\}$, où $T_1$ et $T_2$ sont les sous-arbres à la racine de $T$.
Soit $\mathcal{P}_c$ la propriété suivante définie sur les arbres binaires : un arbre $T$ vérifie la propriété $\mathcal{P}_c$ si et seulement si *il existe une constante c telle que pour tout nœud de T, les hauteurs de ses sous-arbres diffèrent au plus de c.*

1. Caractériser la famille des arbres binaires qui vérifient $\mathcal{P}_0$ et celle des arbres binaires qui vérifient $\mathcal{P}_1$.
   **Sol :** $\mathcal{P}_0$ = arbres parfaits, $\mathcal{P}_1$ = arbres H-equilibrés
   Donner un exemple d'arbre vérifiant $\mathcal{P}_2$ et un exemple d'arbre ne vérifiant pas $\mathcal{P}_2$.
2. Montrer (ou admettre) que pour tout $c \in \mathbb{N}^*$, il existe $\alpha \in ]0, 1]$, tel que $\alpha(1 + \alpha)^c \leq 1$.
   **Sol :** la fonction vaut 0 en 0 et $2^c$ en 1 et elle est continue, donc il existe $\alpha \in ]0, 1]$, tel que $\alpha(1 + \alpha)^c \leq 1$.

3. Montrer, par récurrence sur la hauteur, que pour tout arbre binaire de taille $n$ et de hauteur $h$, qui vérifie $\mathcal{P}_c$, on a $n \geq (1+\alpha)^h - 1$, pour un certain $\alpha$ tel que $0 < \alpha \leq 1$.
   **Sol :** vrai pour $n = 1$ et $h = 0$
   Supposons vrai pour tout $m < n$ ; soit $T$ arbre de taille $n$. T est formé de 2 ss arbres $T_1$ et $T_2$ de tailles $n_1$ et $n_2$ avec $n = n_1 + n_2 + 1$ et de hauteurs $h_1$ et $h_2$ avec $h_1 \geq h_2 - c$, et $h = h_2 + 1$. On choisit $\alpha \in ]0, 1]$, tel que $\alpha(1+\alpha)^c \leq 1$. Par hyp de récurrence on a $n_1 \geq (1+\alpha)^{h_1} - 1$ et $n_2 \geq (1+\alpha)^{h_2} - 1$. Donc $n \geq (1+\alpha)^{h_1} + (1+\alpha)^{h_2} - 1 \geq (1+\alpha)^{h_2-c} + (1+\alpha)^{h_2} - 1 = (1+\alpha)^{h_2}(1 + (1+\alpha)^{-c}) - 1 \geq (1+\alpha)^{h_2}(1+\alpha) - 1 = (1+\alpha)^h - 1$
4. En déduire que tout arbre qui vérifie la propriété $\mathcal{P}_c$ a une hauteur en $O(\log n)$.
   **Sol :** $n \geq (1+\alpha)^h - 1$ donc $h \log(1+\alpha) \leq \log(1+n)$.

# 3 Compression Huffmann Dynamique [4 points]

Décrire la suite des arbres de Huffman dynamiques obtenus pour la compression du texte `paradigite`, et donner le texte compressé.

# 4 Circuit polaire [4 points]

Soit $\mathcal{S}$ un ensemble de $n$ points du plan, et $O \notin \mathcal{S}$. Il s'agit ici de décrire le circuit polaire de $\mathcal{S}$ par rapport à $O$.

1. Donner un algorithme, de complexité $O(n \log n)$, qui calcule le circuit polaire de $\mathcal{S}$ par rapport à $O$, en utilisant l'ordre polaire.
2. Donner un algorithme, de complexité $O(n \log n)$, qui calcule le circuit polaire de $\mathcal{S}$ par rapport à $O$ sans utiliser l'ordre polaire.
3. Peut-on trouver un algorithme de calcul du circuit polaire d'un ensemble de $n$ points, dont la complexité en nombre de comparaisons est $O(n)$ ? Argumentez votre réponse.