Algorithmique
TD3. Semaine du 22/10/01

Exercice 0.1

Montrer que si $f(x) = \Theta(g(x))$, alors $f(x) \times h(x) = \Theta(g(x) \times h(x))$. En déduire que si $h(x) = \Theta(k(x))$, alors $f(x) \times h(x) = \Theta(g(x) \times k(x))$

Exercice 0.2

Démontrer que $\lfloor \log_2(x) \rfloor = \Theta(\log_2(x))$.

Exercice 0.3

Soient f, g, deux fonctions tendant vers $+\infty$ quand leur argument tend vers $+\infty$. Montrer que si $f(x) = \Theta(g(x))$, alors $\ln(f(x)) = \Theta(\ln(g(x)))$.

Exercice 1

1. Montrer qu'un arbre binaire de profondeur p a au plus $2^{p+1}-1$ nœuds.
2. En déduire qu'un arbre binaire à n noeuds a une profondeur d'au moins $\lceil \log_2(n+1) \rceil - 1 = \lfloor \log_2(n) \rfloor$.
3. Un arbre binaire de profondeur p sera dit *équilibré* si tout noeud ayant moins de deux descendants est à la profondeur p ou $p-1$.

   Montrer qu'un arbre équilibré de profondeur p a entre 2^p et $2^{p+1}-1$ noeuds.

   En déduire qu'un arbre équilibré à n nœuds est de profondeur exactement égale à $\lfloor \log_2(n) \rfloor$.
4. Appelons *arbre binaire de recherche* un arbre binaire étiqueté par des entiers vérifiant la condition suivante: si un nœud est étiqueté par n,

   - tous ses descendants à gauche sont étiquetés par des entiers strictement plus petits que n,
   - tous ses descendants à droite sont étiquetés par des entiers strictement plus grands que n,

   Donner une méthode pour construire un arbre binaire de recherche équilibré à partir d'un tableau trié d'entiers deux à deux distincts. Prouver que la méthode proposée est correcte, c'est-à-dire qu'elle renvoie bien un arbre équilibré.

5. Combien faut-il de comparaison, dans le pire des cas, pour déterminer si un entier donné est une étiquette d'un arbre binaire équilibré à n noeuds ?

Exercice 3

Un employé de France Télécom dispose d'une pile de n factures de téléphone et d'une pile de p chèques destinés à payer les factures. L'employé a peut-être moins de chèques que de factures (i.e. $p \leq n$), et il veut savoir quelles factures n'ont pas été payées. Il envisage deux algorithmes.

- Pour chaque chèque, parcourir la pile de factures pour chercher la facture qui lui correspond; lorsqu'on la trouve, enlever la facture de la pile.
- Trier les factures, et construire un arbre binaire de recherche équilibré. Pour chaque chèque, rechercher dans l'arbre la facture correspondante.

1. Pour p fixé, donner en fonction de n, pour chaque algorithme, la complexité du nombre de comparaisons dans le pire des cas. On supposera que la création de l'arbre de factures est en $n \ln(n)$.
2. L'employé décide finalement de trier les *deux* piles. Quelle méthode de recherche peut-il utiliser, une fois les deux piles triées ?