Algorithmique avancée – Examen Réparti 1
UPMC — Master d'Informatique —
Novembre 2015 – durée 2h

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

1 QCM [7 points]

Dans ce QCM, pour chaque question vous devez donner 1 seule réponse et expliquer votre choix par une ligne de texte ou une figure. Le barème sera le suivant : **Réponse correcte et correctement argumentée : 1 point. Réponse incorrecte ou correcte mais mal argumentée : 0 point.**

Q1. La récurrence $T(n) = T(n/3) + 0(1)$ a pour solution
A) $T(n) = \Theta(n)$
B) $T(n) = \Theta(n \log n)$
C) $T(n) = \Theta(\log n)$
Donner un exemple d'algorithme qui a cette complexité.

Q2. Pour un arbre bicolore contenant n clés, l'algorithme permettant de calculer le plus petit élément a une complexité au pire (en nombre de noeuds traversés) en
A) $\Theta(n)$
B) $\Theta(\log n)$
C) $\Theta(1)$
Décrire succintement cet algorithme.

Q3. Dans un arbre binômial
A) le parcours infixe donne les clés dans l'ordre croissant
B) le parcours préfixe donne les clés dans l'ordre croissant
C) chacune des branches est étiquetée de façon croissante

Q4. Une opération de rotation sur un arbre A
A) conserve la hauteur de A
B) augmente de 1 a hauteur de A
C) diminue de 1 a hauteur de A

Q5. Quel type de hachage utiliser si l'on veut éviter les collisions
A) hachage uniforme
B) hachage universel
C) hachage parfait

Q6. Étant donnée une fonction de hachage sur b bits (à valeurs dans $[0, \ldots, 2^b - 1]$), on a k clés à hacher. La probabilité qu'au moins 2 clés aient la même valeur de hachage peut être approximée par
A) $\exp(-\frac{k(k-1)}{2^b})$
B) $\exp(-\frac{k(k-1)}{2b})$
C) $\exp(-\frac{k^2}{2})$

Q7. Dans le hachage dynamique, l'index est
A) un trie binaire
B) un arbre binaire de recherche
C) un 2-trie

2 Exercices [13 points]

Exercice 1. Classer les fonctions suivantes selon leur comportement asymptotique suivant une échelle croissante, $f(n) \leq g(n)$ si $f(n) = O(g(n))$.

$$n \log n \quad n^{10} \quad 35n^2 \quad (\log n)^{100} \quad 4^n \quad n! \quad n^3 2^n \quad n^{1/3} \quad 100n \quad n^2/\log n$$

Exercice 2. Donner un algorithme en $O(n \log n)$ comparaisons, qui étant donné une suite S de n nombres entiers positifs différents et un nombre donné x, détermine s'il existe deux nombres p_1 et p_2 dans S tels que $p_1 + 2p_2 = x$.

Exercice 3. On considère l'arbre binômial B_k (formé de 2^k nœuds). On numérote ces nœuds en ordre postfixe et chaque numéro (étiquette) est écrit en binaire. Ainsi chaque étiquette (de 0 à $2^k - 1$) est un mot binaire sur k bits.

1. Dessiner l'arbre binômial B_3 avec ses nœuds étiquetés en binaire en ordre postfixe.
2. Montrer par récurrence que si x est nœud à profondeur i dans B_k , l'étiquette de x possède $k-i$ bits égaux à 1.

Exercice 4. Les algorithmes sur les splay-trees font appel à une rotation "zig-zig" qui fait une double rotation simple. Décrire cette rotation sur une figure, puis donner la spécification et la définition (le code) de cette fonction zig-zig, en utilisant les fonctions primitives sur les arbres binaires.

Exercice 5. On travaille sur un ensemble dynamique $\mathcal{M}$ d'un millier de mots, sur lequel on veut effectuer différents traitements. Selon le traitement voulu, décrire la structure de données la plus appropriée, et l'algorithme de traitement en quelques phrases.

1. traitement 1 : étant donnés deux mots w_1 et w_2 de $\mathcal{M}$ (avec $w_1 < w_2$ en ordre lexicographique), lister en ordre lexicographique croissant, tous les mots de l'ensemble $\mathcal{M}$ qui sont situés entre w_1 et w_2 .
2. traitement 2 : étant donné un mot w de $\mathcal{M}$ lister en ordre lexicographique croissant tous les mots w_i de l'ensemble $\mathcal{M}$ qui ont w comme préfixe ($w_i = w.v_i$).

Exercice 6. Une file F est une structure de donnée linéaire sur laquelle on peut faire deux opérations
— *ajouter(x,F)*, qui ajoute un élément à la file,
— *supprimer(F)*, qui supprime de la file l'élément le plus anciennement présent.
On peut implanter une file F à l'aide de deux piles P_1 et P_2 (et les opérations classiques *empiler(x,P)* et *dépiler(P)*), de la façon suivante :
— *ajouter(x,F)* : *empiler(x,*P_1*)*
— *supprimer(F)* : si P_2 est vide alors dépiler tous les éléments de P_1 et les empiler au fur et à mesure dans P_2. Puis (dans tous les cas) dépiler P_2.

1- Montrer que cette implantation est correcte.

2- Calculer le coût réel d'un ajout et le coût réel d'une suppression (le coût est mesuré en nombre d'opérations *empiler* et *dépiler* faites sur les piles).

3- On va calculer le coût amorti des opérations d'ajout et de suppression sur la file en essayant différentes fonctions de potentiel. Le coût est mesuré en nombre d'opérations *empiler* et *dépiler* faites sur les piles.

a) fonction Φ_1 : le potentiel vaut deux fois le nombre d'éléments de la pile P_1.
Montrer que le coût amorti d'un ajout vaut 3 et celui d'une suppression vaut 1. Le potentiel est-il toujours ≥ 0 ?

b) fonction Φ_2 : potentiel égal au nombre d'éléments de P_1 moins le nombre d'éléments de P_2.
Montrer que le coût amorti d'un ajout vaut 2 et celui d'une suppression vaut 2. Montrer que le potentiel n'est pas toujours ≥ 0. La fonction Φ_2 est-elle une fonction de potentiel acceptable ?