Algorithmique Avancée Examen Réparti 1 (2h)

AVEC IDEES DE CORRECTION

Seuls documents autorisés : les polycopiés de cours et la copie personnelle. Le barème est indicatif.

Exercice 1 : Questions courtes et indépendantes [12 points]

Question 1

a. Décrire la structure d'une file binomiale contenant 285 éléments, nommée A par la suite.

b. Décrire la structure d'une file binomiale contenant 301 éléments, nommée B par la suite.

c. Décrire en quelques lignes les étapes importantes de la fusion de A et B.

d. Quel est le nombre de comparaisons nécessaires pour fusionner A et B ?

Question 2

a. Combien y-a-t-il de clés dans un arbre 2-3-4 de hauteur h ne contenant que des nœuds de type 2 ?

b. Combien y-a-t-il de clés dans un arbre 2-3-4 de hauteur h ne contenant que des nœuds de type 4 ?

c. Étant donné l'ensemble des arbres 2-3-4 contenant n clés et dont la racine est un nœud de type 4, montrer qu'il existe des arbres dont le sous-arbre de la racine le plus à droite peut ne contenir que $\Theta(\sqrt{n})$ clés.

d. Qu'en est-il si la racine est un nœud de type 2 ?

Question 3

a. On considère un texte formé de 5 lettres a, b, c, d, e, de fréquences respectives $1, 1, 1, 2, 3$. Donner 2 arbres de Huffman construits sur ce texte : l'un de hauteur 3 et l'autre de hauteur 4.

b. Étant donnés ces deux arbres de Huffman, quelles sont les tailles des textes compressés qui en résultent ?

Question 4 On considère la suite (g_k) définie par $g_0 = g_1 = g_2 = 1$ et pour $k \geq 3, g_k = g_{k-1} + g_{k-2}$. Pour une telle suite (g_k), on rappelle :

$$\sum_{i=0}^{k} g_i = g_{k+2}.$$

Dans la suite, le texte T_k contient $k + 1$ lettres qui ont pour fréquences $g_0, \dots, g_k$.

a. Écrire un algorithme qui construit un arbre de Huffman de hauteur k pour T_k.

b. Montrer par récurrence que la taille du texte compressé résultant d'un arbre de Huffman pour T_k est $g_{k+4} - 3$.

Solution

1.

$$285 = 2^0 + 2^2 + 2^3 + 2^4 + 2^8.$$

$$301 = 2^0 + 2^2 + 2^3 + 2^5 + 2^8.$$

6 comparaisons : 0-0 -> 2-2 -> 3-3 -> 4-4 -> 5-5 -> 8-8

2. a. prendre des 4-nœuds partout sauf sur le sous-arbre de droite où l'on prend uniquement des 2-nœuds nombre de clés : $n = 3(racine) + 2^h - 1(sousarbredroite) + 3(4^h - 1)(les3autres) => h = \log_4 n$ donc dans le sous arbre droit le nombre de clés est de l'ordre de $2^{\log_4 n} = n^{\log 2/\log 4} = \sqrt(n)$

b. même résultat

3. a. (((a. b) . (c.d)). e) et ((((a. b) .c).d).e)

b. 18

c. c'est la taille du texte compressé

4. a. $A :=$ arbrebin(g_0, g_1) ; Pour i = 2 à k $A :=$ arbrebin(A, g_i) ; return A
b. vrai pour k= 2 : poids = 5 = 8-3
On travaille sur l'arbre de Huffman H_k de hauteur k On suppose vrai pour k et on montre vrai pour $k+1$

la taille du texte compressé de H_k est $g_{k-1} + 2g_{k-2} + 3g_{k-3} + \cdots + kg_0$ (se démontre facilement par récurrence) On a $H_{k+1} =$ arbrebin(H_k, g_{k+1}) donc le poids de H_{k+1} est égal à g_{k+1} plus le poids de H_k dans lequel toutes les profondeurs sont augmentées de $1 = g_{k+1} + poids(H_k) + g_0 + \cdots + g_k$. Par hypothèse de récurrence on a donc $poids(H_{k+1}) = g_{k+1} + g_{k+4} - 3 + g_0 + \cdots + g_k = g_{k+4} - 3 + g_{k+3} = g_{k+5} - 3$

Exercice 2 *: (Problème) Tournoi auto-adaptatif [10 points]*

Un tournoi auto-adaptatif une une structure de données qui permet d'effectuer la fusion de deux tournois efficacement. Pour cela on va s'autoriser à déséquilibrer la structure.

Un tournoi auto-adaptatif est un arbre binaire étiqueté, tel que la suite des clés de la racine à n'importe quelle feuille est (strictement) croissante. La *taille* d'un arbre est le nombre de clés qu'il contient. Voilà les notations que nous utiliserons :

— $< >$ est le tournoi vide
— $< a >$ est le tournoi réduit à une feuille dont la clé est a
— $< T1\ a\ T2 >$ est le tournoi dont la racine contient la clé a, et les tournois $T1$ et $T2$ sont les enfants gauche et droit (qui peuvent être éventuellement vides).

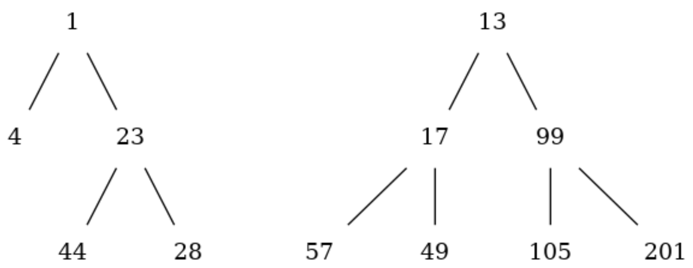
Opérations de base sur les tournois A et B (*ne contenant aucune clé identique*) :

```
Fusion(A, < >) = A          Fusion(< >, A) = A
Fusion(A, B) = Jonction(A, B)   # si la clé de la racine de A est plus petite que celle de B
Fusion(A, B) = Jonction(B, A)   # sinon

Jonction(< A1 a A2 >, B) = < Fusion(A2, B) a A1 >   # A1 ou A2 sont éventuellement vides
```

*Pour **fusionner deux tournois** A **et** B, on appelle* `Fusion(A, B)`.

Question 1 Soit A (à gauche) et B (à droite) les deux tournois suivants :



Donner l'arbre résultant de `Fusion(A, B)` (en exhibant les étapes intermédiaires)

Question 2 À l'aide des fonctions ci-dessus, proposer une fonction `Insertion` prenant une clé x et un tournoi A et renvoyant un tournoi contenant les clés de A ainsi que x.

Question 3 À partir de l'arbre vide, insérer successivement les clés `1,2,3,4`.
À partir de l'arbre vide, insérer successivement les clés `4,3,2,1`.

Question 4 Prouver que la `Fusion` de 2 tournois est un tournoi.
En déduire la correction de `Insertion`.

Question 5 Donner un algorithme `ExtractionMin` pour la fonction d'extraction de la clé minimale d'un tournoi A et renvoyant un couple formé de la clé minimale et d'un tournoi contenant les autre clés de A.
Prouver la correction de `ExtractionMin`.

Le **poids** d'un nœud est la taille du sous-arbre enraciné en ce nœud. Un nœud (autre que la racine) est appelé *gauche* (respectivement *droit*) s'il est l'enfant gauche de son parent (resp. l'enfant droit).

On dit qu'un nœud (autre que la racine) est **lourd**, si son poids est strictement supérieur à la moitié du poids de son parent. Sinon il est **léger**. La racine n'est ni lourde ni légère.

On définit le potentiel $\phi(A)$ pour un arbre A comme étant le nombre de nœuds droits et légers qu'il contient. On pose $\phi(<>) = 0$.

Question 6 Étant donné $A =< A1\ a\ A2 >$, donner la récurrence permettant de calculer $\phi(A)$.

Question 7 Soit A un tournoi de taille n. Étant donné un nœud de A, montrer que ses deux enfants ne peuvent pas être simultanément lourds.

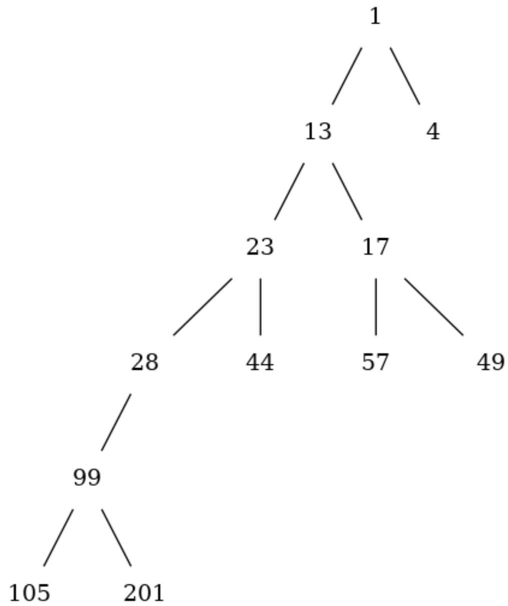
Question 8 Soit A un tournoi de taille n. Étant donné un chemin de la racine à une feuille de A, montrer que ce chemin contient au plus $\lfloor \log_2 n \rfloor$ nœuds légers.

La mesure de complexité correspond au nombre de comparaisons de clés ; on note $|A|$ la taille de l'arbre A et $\hat{c}(op)$ le coût amorti pour effectuer l'opération *op*.

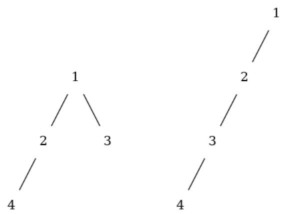
Question 9 Soit A et B deux tournois de tailles respectives n_1 et n_2, avec $n_1 + n_2 = n$. Montrer que la complexité amortie de `Fusion(A,B)` est $O(\log n)$.

Question 10 Conclure sur la complexité amortie des opérations d'insertion et d'extraction du minimum dans les tournois auto-adaptatifs.

Solution



1.
2. `Insertion(a, B) = Fusion(< a >, B)   #a est une clé`
   ou alors
   `Insertion(a, B) = Fusion(B, < a >)`
3. Peu importe l'algo d'insertion on tombe sur les mêmes résultats.



4. Il suffit de montrer que le résultat est un tournoi. Induction sur `Fusion`.
5. `ExtractionMin(<A1 a A2>) = (a, Fusion(A1, A2))   #renvoie le couple dont le premier`
   `# élément est la clé min et le deuxième, le tournoi privé de a`
   La correction résulte de la correction de `Fusion`.
6.
$$\phi(A) = \phi(A1) + \phi(A2) + 1_{\{si|A2| \leq |A1|\}}.$$
7. C'est trivial (par l'absurde).
8. récurrence : taille n nb noeuds légérs <= log n
   A arbre de taille n+1. A = <A1 x A2> A1 et A2 on au plus log2 (n-1) noeuds légers : attention à leur racine -> pb
   supposons racine A1 devient léger => |A1| <= n/2 donc A1 possède au plus log2 n/2 = log2n - 1 noeuds légér + racine -> donne le résultat (symétrie pour A2)
9. Soit l_1 et l_2 le nombre de nœuds lourds de la branche la plus à droite de A et de B. Le nombre de nœuds légers de ces branches est majoré par $\lfloor \log_2 n_1 \rfloor$ et $\lfloor \log_2 n_2 \rfloor$. Par ailleurs les racines de A et B ne sont ni lourde ni légère, donc le nombre total de nœuds sur les branches les plus à droites de A et B est majoré par
$$\leq 2 + l_1 + l_2 + 2\log_2 n.$$
   Dans le processus de `Mix`, les $l_1 + l_2$ nœuds qui étaient des nœuds droits lourds perdent ce statut (mais ils restent lourds !). Mais les échanges d'enfants va engendrer de nouveaux nœuds droits lourds. Notons l_3 le nombre de ces nouveaux nœuds droits lourds. Remarquons que ces nœuds ont tous un frère léger sur la branche la plus à gauche de l'arbre résultat. Or il y a au plus $\lfloor \log_2 n \rfloor$ nœuds légers sur la branche la plus à gauche. Au final, la complexité amortie d'une fusion est majorée par
$$\leq 2 + l_1 + l_2 + 2\log_2 n + \log_2 n - l_1 - l_2 \leq 2 + 3\log n.$$
10. L'insertion et l'extraction du min se font en complexité amortie $O(\log n)$.