Master 1 : STL Année 2022/2023

Algorithmique Avancée Examen Réparti 1

AVEC CORRECTION

Les seuls documents autorisés sont les polys de cours, ainsi que la copie personnelle. Le barème donné, sur 22, est indicatif. La note finale sera la valeur du $\min\{\textit{addition des points}; 20\}$*.)*

Exercice 1 *: QCM [2 points]*

Dans ce QCM, pour chaque question vous devez donner 1 seule réponse et expliquer votre choix par une à deux lignes 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.** *Il n'y a pas de points négatifs.*

Question 1 Dans un tournoi binomial,

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.

Question 2 Toute file binomiale

A. a une taille qui est une puissance de 2.

B. est une file binomiale relâchée.

C. contient au moins deux tournois binomiaux.

Solution

1. C.

2. B.

Exercice 2 *: ROBDD [3 points]*

Étant donné le ROBDD d'une fonction booléenne, contenant n nœuds internes (étiquetés par des variables) et 2 nœuds externes (étiquetés par $True$ et $False$).

Question 1 Donner un algorithme linéaire en n permettant de calculer pour combien d'affectations des variables la fonction s'évalue à $True$? On peut éventuellement enrichir le ROBDD afin de satisfaire la complexité (mais cet enrichissement doit être pris en compte dans la complexité linéaire).

Question 2 Justifier la complexité de votre algorithme.

Solution

1. Il faut parcourir le ROBDD, en parcours suffixe : en chaque noeud on calcule l'addition des $True$ dans son enfant gauche avec celui de son enfant droit. Si jamais un nœud a déjà été visité (la valeur existe déjà), on ne redescend pas dans les enfants du nœud.

2. On ne passe qu'une fois par chaque nœud.

Exercice 3 *: Algorithme de Huffman statique [7 points]*

Question 1 On utilise l'algorithme de Huffman statique pour coder un texte sur trois lettres $\{a, b, c\}$ de fréquences f_a, f_b, f_c. Dans chacun des 3 cas suivants, vous devez : ou bien donner un exemple de fréquences f_a, f_b, f_c à partir desquelles on obtient le code proposé et expliquer la construction de l'arbre, ou bien expliquer pourquoi il est impossible d'obtenir ce code.

- Code1 : $\{0, 10, 11\}$
- Code2 : $\{0, 1, 00\}$
- Code3 : $\{10, 01, 00\}$

Question 2 Montrer que la propriété suivante est vraie pour tout code de Huffmann statique :
Si toutes les lettres du texte à coder ont une fréquence inférieure à $1/3$*, alors aucune lettre n'a un codage de longueur* 1.

Question 3 Quelle est la plus grande longueur possible pour le codage d'une lettre, si l'on utilise un codage de Huffman statique pour un texte de n lettres de fréquences $f_1, f_2, \ldots, f_n$? Donner un exemple.

Question 4 Pour le texte $T = abracadabra$ de 5 lettres, quel est le nombre moyen de bits par lettre dans un codage de Huffman statique ? Quelle est la formule numérique de l'entropie de T ? En évaluant cette formule on obtient environ $2,04\ldots$ Comparer les 2 valeurs précédentes ; y a-t-il contradiction avec l'optimalité du codage de Huffman statique ?

Solution

1. — Code1 : $\{0, 10, 11\}$: $f_a = 1/2, f_b = 1/4, f_c = 1/4$.
— Code2 : $\{0, 1, 00\}$ impossible pas préfixe.
— Code3 : $\{10, 01, 00\}$ pas complet donc pas minimal.

2. Comme chaque lettre a une fréquence $< 1/3$, il y a au moins 4 lettres. Si "à la fin", il reste une lettre a de poids p et deux arbres T1 et T2 de poids $p1 \geq p2$ (T1 est de hauteur au moins 1), on a $p < 1/3$ donc $p1 + p2 > 2/3$ et $p1 > 1/3$. On rassemble donc a et T2 en un arbre T puis T et T1 (et c'est fini).

3. C'est $n - 1$. Exemple : prendre $f_i = 2i - n$, pour $i = 0..n - 2$, et $f_{n-1} = 1/2 - 1/2n$; on a $f_0 + ... + f_i = (2i + 1 - 1)/2n < f_{i+1}$, pour $i = 0..n - 1$.

4. $|T_C| = 23$, donc le nombre moyen de bits par lettre du texte est $23/11 = 2.09$. L'entropie de T est $\sum_i pi \log(1/pi) = 5/11 \log(11/5) + 2 * 2/11 \log(11/2) + 2 * 1/11 \log(11) = 2.04...$ Cette valeur est inférieure à la précédente, mais pas contradiction avec l'optimalité du codage de Huffman statique, qui est optimal pour codage par "lettre".

Exercice 4 *: Dichotomie dynamique [10 points]*

La recherche dichotomique dans un tableau trié consomme un temps logarithmique (en nombre de comparaisons), et le temps d'insertion d'un nouvel élément est linéaire (en nombre d'écritures) par rapport à la taille du tableau, mais est en temps logarithmique (en nombre de comparaisons).

Dans tout l'exercice, la mesure de complexité correspond au nombre de comparaisons d'éléments. On peut améliorer le temps d'insertion en conservant séparément plusieurs tableaux triés. Plus précisément, on suppose que l'on souhaite implanter les opérations de recherche et d'insertion sur un ensemble E ayant n éléments. Supposons que n s'écrive $b_{k-1} \ldots b_1 b_0$ en base 2. On a k tableaux triés $A_0, A_1, \ldots, A_{k-1}$, de tailles respectives $2^0, 2^1, \ldots, 2^{k-1}$. Chaque tableau A_i est soit plein, soit vide selon que $b_i = 1$ ou $b_i = 0$. Le nombre total d'éléments contenus dans les k tableaux est donc $b_0 \cdot 2^0 + b_1 \cdot 2^1 + \cdots + b_{k-1} \cdot 2^{k-1} = n$. Bien que chaque tableau soit trié, il n'existe pas de relation particulière entre les éléments des différents tableaux.

Voilà un exemple :

Si $E = \{1, 2, \ldots, 13\}$ alors n s'écrit 1101 en base 2. Le tableau A_1 est vide et les tableaux A_0, A_2 et A_3

sont pleins, par exemple : $A_0 = [13]$, $A_2 = [1, 6, 8, 11]$ et $A_3 = [2, 3, 4, 5, 7, 9, 10, 12]$ (mais ce n'est pas la seule manière de remplir ces trois tableaux).

Recherche d'un élément :

Question 1 Décrire une méthode de recherche opérant sur la structure de données présentée dans l'introduction.

Question 2 Calculer la complexité de l'opération de recherche dans le pire cas.

Insertion d'un élément :

Question 3 On suppose que la complexité (en nombre de comparaisons) de l'interclassement de deux tableaux triés de tailles t_1 et t_2 est égale à $t_1 + t_2$. On considère que les tableaux $A_0, A_1, \ldots, A_{i-1}$, de tailles $2^0, 2^1, \ldots, 2^{i-1}$, sont pleins et triés. Montrer que l'on peut réaliser leur interclassement en au plus 2^{i+1} comparaisons (lorsque $i \geq 1$). Comment faut-il procéder ? Justifier la complexité.

Question 4

1. Décrire une méthode d'insertion d'un nouvel élément (distinct de tous éléments déjà présents) opérant sur la structure de données présentée dans l'introduction.
2. On suppose que A_0, A_1, A_2 et A_3 sont vides et que $A_4 = [1, 2, \ldots, 16]$. Insérer 20.
3. On suppose que A_3 est vide et que $A_0 = [20], A_1 = [3, 13], A_2 = [1, 6, 8, 11]$ et $A_4 = [21, 22, \ldots, 36]$. Insérer 12.

Question 5 Calculer la complexité de l'insertion (en nombre de comparaisons) dans le pire cas :

1. en fonction de i, où i est tel que $A_0, A_1, \ldots, A_{i-1}$ sont pleins et A_i est vide
2. en fonction de n.

Coût amorti d'une insertion :

Question 6 Soit $n = 2^k - 1$. On effectue une suite de n insertions dans notre structure de données, initialement vide. Pour un entier $1 \leq \ell \leq n$ s'écrivant $c_{k-1} \ldots c_1 c_0$ en binaire, on note $\alpha(\ell)$ le plus petit indice i tel que $c_i = 0$ (dans le cas où $\ell = n$, on définit $\alpha(\ell) = k$).

Montrer que la complexité amortie d'une insertion est $O(1)$.

Solution

1. On recherche par dichotomie dans chacun des tableaux.

2. La complexité vaut

$$\sum_{i=0}^{k-1} \log_2 2^i = \sum_{i=0}^{k-1} i = \frac{k(k-1)}{2}.$$

Donc la recherche, au pire cas, se fait en $\Theta(\log^2 n)$. On peut faire plus simple en disant qu'on cherche dans $\log n$ tableaux de taille au plus $\log n$, mais on n'obtient que $O(\log^2 n)$.

3. On trie la suite de tableaux 2 à 2 en prenant à chaque fois les 2 plus petits. La complexité vaut $(2^0 + 2^1) + (2^0 + 2^1 + 2^2) + \cdots + (2^0 + \ldots 2^i - 1)$. 2^0 est sommé $i - 1$ fois et sinon ($j \geq 1$), 2^j est sommé $(i - j)$ fois. Le nombre exact de comparaisons vaut $2^{i+1} - i + 1$.

4. Soit i le plus petit indice tel que le tableau A_i est vide. On interclasse tous les A_j pour $j < i$ et on insère le nouvel élément, on obtient un tableau A_i, et on vide les A_j.
On met 20 dans A_0
On suit l'algo pour remplir A_3 et vider les A_0, A_1, A_2.

5. $2^{i+1} + \log 2^i = \Theta(2^{i+1})$
$i \leq \log_2 n$ donc la complexité au pire vaut $\Theta(n)$.

6. la complexité amortie vaut le double par rapport à celle de l'incrémentation du compteur (exo 2.4)