Algorithmique Avancée — Examen Réparti 1

Les seuls documents autorisés sont les polys de cours, ainsi que la copie personnelle. Le barème donné est indicatif.

Exercice 1 *: Comportement asymptotique [2 points]*

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

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

Exercice 2 *: Arbres 2-3-4 [4 points]*

Soit A un arbre 2-3-4 ayant 4 niveaux de nœuds ; on insère une nouvelle valeur v dans A, en faisant les éclatements à la descente. Soit x_1, x_2, x_3, x_4 les nœuds du chemin suivi pour insérer v (x_1 est la racine, x_4 est une feuille).

Question 1 Calculer la variation du nombre de 2-nœuds, du nombre de 3-nœuds, du nombre de 4-nœuds après insertion de v dans A dans chacun des cas suivants :

1. x_1 est un 4-nœud, x_2, x_3, x_4 sont des 3-nœuds ;
2. x_1, x_2, x_3, x_4 sont des 4-nœuds ;
3. x_1, x_3 sont des 2-nœuds, x_2, x_4 sont des 4-nœuds ;
4. x_1, x_3 sont des 3-nœuds, x_2, x_4 sont des 4-nœuds.

Exercice 3 *: Coût amorti [6 points]*

Soit T un tableau dynamique de taille initiale $m = 1$ dont on note $s(T)$ le nombre de cases au total et $n(T)$ le nombre d'éléments y figurant. Ce tableau va au fur et à mesure du temps grandir et diminuer en fonction du rapport $r = n(T)/s(T)$.

Ainsi dès que l'on ajoute un élément dans T et que $r = 1$ avant l'ajout alors on double la taille de T pour pouvoir effectivement faire cet ajout. A contrario, si l'on retire un élément de T et que $r = 1/4$ avant la suppression alors le tableau est diminué de la moitié de ses cases (on enlève celles ne contenant pas d'éléments bien sûr).

Lorsqu'on ajoute un élément dans T, c'est en position $n(T)$ et lorsqu'on supprime un élément de T, c'est l'élément en position $n(T) - 1$ que l'on supprime (les positions dans le tableau sont numérotées à partir de 0).

Lorsque la taille du tableau est modifiée (doublée ou diminuée de moitié), on crée un nouveau tableau et on recopie les valeurs de l'ancien tableau dans le nouveau tableau.

Question 1 Illustrer par un schéma l'évolution de cette structure de données au cours des opérations ci-dessous :
– ajouts successifs des valeurs x_1, x_2, x_3, x_4, x_5
– puis suppressions successives de x_5, x_4, x_3, x_2, x_1.

Question 2 Soit la fonction P_C définie comme suit :

$$P_C = \begin{cases} 2n(T) - s(T) & \text{si } r \geq C \\ s(T)/2 - n(T) & \text{si } 0 < r < C \\ 0 & \text{si } r = 0 \end{cases}$$

1. Déterminer la valeur de C pour laquelle la fonction P_C est une fonction de potentiel.
2. Étudier le coût amorti des opérations d'ajout et de suppression dans les cas suivants :

(a) $1/2 < r < 1$ (b) $r = 1$
(c) $r = 1/4$ (d) $1/4 < r < 1/2$

Exercice 4 *: Files XH-équilibrées [10 points]*

Dans cet exercice, on utilisera les nombres de Fibonacci, dont on rappelle la définition et une propriété :

— $F_0 = 0$, $F_1 = 1$ et $F_k = F_{k-1} + F_{k-2}$ pour $k \geq 2$

— $F_{k+2} \geq \Phi^k$ avec $\Phi = \dfrac{1+\sqrt{5}}{2}$ (on a $\Phi \sim 1,61803$)

On dit qu'un arbre binaire est *H-équilibré* si en tout nœud de l'arbre, les hauteurs des sous-arbres gauche et droit diffèrent d'au plus 1. La hauteur d'un arbre A est notée $h(A)$; par convention $h(\emptyset) = -1$.

On définit récursivement les arbres *XH-équilibrés* de la façon suivante :

— $A_0 = \emptyset$, $A_1 = \bullet$

— pour $k \geq 2$, A_k est l'arbre binaire dont le sous-arbre gauche est A_{k-1} et le sous-arbre droit est A_{k-2}.

On dit que l'arbre XH-équilibré A_k est de *rang* k.

Question 1 Construire A_4.

Question 2 Montrer que $h(A_k) = k - 1$ et que tout arbre XH-équilibré est H-équilibré.

Question 3 On définit récursivement les nombres G_k de la façon suivante :

$G_0 = 0$, $G_1 = 1$ et $G_k = G_{k-1} + G_{k-2} + 1$ pour $k \geq 2$

Montrer que la taille n de A_k est égale à G_k et que $G_k = F_{k+2} - 1$. En déduire que $h(A_k) < \log_\Phi(n+1)$.

Files XH-équilibrées Une *file XH-équilibrée* est une suite d'arbres XH-équilibrés étiquetés, de tailles strictement décroissantes et dont les étiquettes croissent de la racine vers les feuilles (autrement dit, chaque arbre est un tournoi). À toute file XH-équilibrée $\mathcal{A}$, on peut associer une suite $(b_k, \ldots, b_2, b_1)$ telle que k est le rang du plus grand tournoi XH-équilibré de la file $\mathcal{A}$, b_i vaut 1 s'il y a un tournoi XH-équilibré de rang i dans la file $\mathcal{A}$ et b_i vaut 0 sinon. Par exemple, à la file (A_8, A_4, A_2, A_1) on associe la suite $(1, 0, 0, 0, 1, 0, 1, 1)$. On remarque que la taille de cette file est égale à $G_8 + G_4 + G_2 + G_1 = 54 + 7 + 2 + 1 = 64$.

Question 4 Donner un exemple de file XH-équilibrée de taille 43.

Question 5 Soit $\mathcal{A}$ une file XH-équilibrée de taille n et $(b_k, \ldots, b_2, b_1)$ la suite associée à $\mathcal{A}$. Exprimer la taille n de la file $\mathcal{A}$ en fonction de $G_k, \ldots, G_2, G_1$. En déduire que $k \leq \log_\Phi(n+1)$.

Une *bonne* file XH-équilibrée est une file XH-équilibrée dont la suite associée $(b_k, \ldots, b_2, b_1)$ ne contient jamais trois 1 consécutifs. Par exemple (A_6, A_1) et $(A_5, 4_4, A_2)$ sont deux bonnes files XH-équilibrées (toutes deux de taille 21), mais $(A_7, A_5, A_4, A_3, A_1)$ n'en est pas une.

Insertion On veut insérer un élément x dans une bonne file XH-équilibrée $\mathcal{A}$ de taille n, de façon à obtenir une bonne file XH-équilibrée.

Idée : trouver, s'il existe, le plus grand i tel que $b_i = 1$ et $b_{i-1} = 1$, puis faire avec A_i, A_{i-1} et x un nouveau tournoi XH-équilibré (de rang $i+1$) ; traiter les cas où un tel i n'existe pas.

Question 6

1. Décrire un algorithme permettant de déterminer le plus grand i tel que $b_i = 1$ et $b_{i-1} = 1$, s'il existe.
2. Traiter les cas où un tel i n'existe pas.
3. Décrire un algorithme d'insertion d'un élément x dans une bonne file XH-équilibrée $\mathcal{A}$. Montrer que le résultat obtenu est encore une bonne file.

Question 7 Quelle est la complexité de cet algorithme (en nombre de comparaisons entre éléments, et de tests sur les b_i) ?

Suppression du minimum On veut supprimer le minimum d'une bonne file XH-équilibrée $\mathcal{A}$ de taille n, de façon à obtenir une bonne file XH-équilibrée.

Idée : soit A_k l'arbre dont la racine est égale au minimum ; trouver le plus petit i tel que $b_i = 1$, et $b_j = 0$ si $j < i$, "échanger" la racine de A_i et la racine de A_k puis supprimer le minimum.

Question 8

1. Traiter les cas où $i < 3$ et illustrer la méthode avec l'exemple $\mathcal{A} = (A_4, A_2)$ dans lequel $A_4 =< 2, < 3, < 6, 11, \emptyset >, 5 >, < 8, 9, \emptyset >>$ et $A_2 =< 7, 12, \emptyset >$.
2. Traiter les cas où $i \geq 3$.
3. Décrire un algorithme de suppression du minimum dans une bonne file XH-équilibrée $\mathcal{A}$. Montrer que le résultat obtenu est encore une bonne file.

Question 9 Quelle est la complexité de cet algorithme (en nombre de comparaisons entre éléments, et de tests sur les b_i) ?