Eléments d'Algorithmique

Examen 2ème session du 24 juin 2009 - durée : 2h00

Il va sans dire que la rigueur des raisonnements, la clarté des explications, mais aussi la qualité de la présentation influera sensiblement sur la note.

Le barème n'est donné qu'à titre indicatif et pourra donc être modifié.

Exercice 1 - Questions de cours. (1,5 points)

Expliquez en quelques phrases très claires ce que l'on appelle :

- une borne de complexité optimale
- une majoration de la complexité
- la complexité en moyenne.

Exercice 2 - Tableau simple. (1,5 points)

On dira qu'un tableau T de taille n est un cycle s'il contient tous les éléments $1..n$, et que la suite $P[1], P[P[1]], P[P[P[1]]]$ etc énumère exactement toutes les valeurs de 1 à n . Donnez un algorithme linéaire qui dise si un tableau d'entiers quelconque est un cycle.

Exercice 3 - Tri par Tas (7 points).

Comme on l'a vu en cours, un tas est un arbre binaire satisfaisant deux propriétés. La première (dite ici *Propriété 1*) concerne la disposition des nœuds dans l'arbre (la "forme" de l'arbre), la deuxième concerne l'ordre relatif des éléments (ou plutôt de leurs clés) stockés dans les nœuds de l'arbre. La Propriété 1, en particulier, permet de représenter l'arbre sous forme de tableau.

1. Donner la représentation sous forme d'arbre du tableau $T = [15; 27; 31; 19; 53; 73; 14]$ en sachant qu'il s'agit d'un arbre binaire satisfaisant la Propriété 1.
2. En supposant que vous disposez d'une méthode `entasser(int [] X, int i)` qui entasse l'indice i dans un tas X (plus précisément : si les arbres ayant comme racine les fils du nœud i sont des tas max, alors `entasser(int [] X, int i)` transforme le sous-arbre de racine i en un tas max), rappelez l'algorithme qui permet de transformer en tas max un tableau quelconque représentant un arbre satisfaisant la Propriété 1.
3. Donner les représentations des arbres intermédiaires obtenus en appliquant cette construction au tableau T .
4. On considère à présent le tas max TM suivant : $[100; 50; 20; 17; 35; 5; 19]$. Donnez les 7 arbres intermédiaires obtenus pendant la procédure de tri par tas appliquée à TM .

Exercice 4 - Plus longue sous-suite croissante (10 points).

Soit T un tableau de n entiers distincts. On appelle sous-suite de T toute suite de la forme $T[i_1], T[i_2], \dots, T[i_k]$, où $1 \leq i_1 < i_2 < \dots < i_k \leq n$ (en d'autres termes, on choisit dans l'ordre certains éléments de T). L'entier k est la taille de la sous-suite. On cherche une plus longue sous-suite croissante de T , c'est-à-dire une sous-suite de taille maximale telle que $T[i_1] < T[i_2] < \dots < T[i_k]$. Par exemple, si $T = [5, 1, 7, 3, 2, 4, 8]$, alors il y a deux plus longues sous-suites croissantes : $[1, 3, 4, 8]$ et $[1, 2, 4, 8]$ (toutes deux de taille 4). On se propose de décrire un algorithme efficace pour trouver la taille d'une plus longue sous-suite croissante.

$M[j] < T[1..i]$
 $M = [1, 3]$

L'algorithme parcourt chaque élément du tableau T dans l'ordre. À l'itération i , en fonction de la valeur $T[i]$, il met à jour un tableau M défini comme suit : $M[j]$ est le plus petit élément de $T[1..i]$ qui termine une sous-suite croissante de taille j . Par exemple, pour T comme ci-dessus et $i = 4$, on a $M = [1, 3]$: en effet 1 est le plus petit élément de $T[1..4]$ terminant une sous-suite croissante de taille 1, alors que 3 est le plus petit élément de $T[1..4]$ terminant une sous-suite croissante de taille 2, et il n'y a pas de sous-suite croissante de $T[1..4]$ de taille supérieure.

1. Montrer qu'à tout moment, les entiers dans le tableau M sont en ordre croissant.
2. Quelles sont les opérations à faire lors de la première itération de la boucle pour initialiser le tableau M ? (en d'autres termes, quelle(s) valeur(s) doit contenir M après l'itération $i = 1$?)
3. Supposons que l'on soit dans le cas particulier où $T[i]$ est le plus grand élément de $T[1..i]$. Quelles opérations doit-on faire pour mettre à jour M au cours de l'itération i ?
4. Supposons maintenant que l'on soit dans le cas général (donc $T[i]$ n'est plus nécessairement le plus grand élément de $T[1..i]$). Comment mettre à jour M au cours de l'itération i ? On pourra regarder comment les éléments $M[j]$ doivent être modifiés dans chacun des cas suivants :
 - j est tel que $M[j] < T[i]$,
 - j est tel que $M[j - 1] < T[i] < M[j]$, et
 - j est tel que $M[j - 1] > T[i]$.

Montrer que votre méthode de mise à jour préserve l'invariant du tableau M , c'est-à-dire : " $M[j]$ est le plus petit élément de $T[1..i]$ qui termine une sous-suite croissante de taille j ".

5. Au vu de vos réponses aux questions 3. et 4., donner une procédure maj, de complexité logarithmique, pour mettre à jour M en fonction de la valeur $T[i]$ (on pourra penser à la recherche dichotomique).
6. En utilisant la procédure maj de la question précédente, écrire l'algorithme calculant la taille d'une plus longue sous-suite croissante pour un tableau T .
7. Analyser la complexité de votre algorithme.