Institut Galilée — Algorithmique et arbres
Année 2009-2010 — L2

Examen du 28 juin 2010

Aucun document autorisé. Le barême est uniquement indicatif.

Vous pouvez écrire les algorithmes en C ou en pseudo-code. Si cela vous pose trop de difficultés n'hésitez pas à répondre en décrivant un algorithme par des phrases ! Vous pouvez faire appel à des fonctions auxilliaires vues en cours (comparaison, tris, sous-tableau, etc.).

Première partie — 10 points

Exercice 1 (Notation asymptotique)**.**
Rappeler les définitions utilisées et justifier (démontrer) vos réponses à partir de ces définitions.

1. Est-ce que $n = \Theta(n^2)$? — 1 pt, 9 min
2. Est-ce que $\sum_{k=1}^{n} k = \Omega(n^2)$? — 1 pt, 9 min
3. Est-ce que $\log(n!) = O(n \log n)$? — 1 pt, 9 min

Exercice 2 (Piles)**.**
On forme une nouvelle pile en empilant successivement 3, 2, 1 puis on dépile deux éléments. Quel élément reste-t-il dans la pile ? (Facile) — 0,5 pt, 4 min

Exercice 3 (Tas puis ABR)**.**
Soit la liste d'entiers : $1, 6, 3, 9, 5, 7, 2, 0, 4, 8$. Vous pouvez répondre à chacune des questions en représentant seulement l'arbre obtenu et, lorsqu'il y a deux réponses correctes possibles, ne donner qu'une seule réponse. — tot: 4.5 pt

1. À partir de la liste, former un **tas** par insertions successives (dans l'ordre de la liste). — 1 pt, 9 min
2. Supprimer l'élément maximum du tas obtenu. — 0.5 pt, 4 min
3. À partir de la liste, former un tas par l'autre méthode (qui utilise MaintienBas). — 1 pt, 9 min
4. À partir de la liste, former un **arbre binaire de recherche** par insertions successives. — 1 pt, 9 min
5. À partir de l'arbre de recherche obtenu supprimer l'élément 7. — 0.5 pt, 4 min
6. En repartant de l'arbre obtenu à la question 4, supprimer l'élément 6. — 0.5 pt, 4 min

Exercice 4.
Classer les fonctions de complexité $n \log n, 2^n, \log n, n^2, n$ par ordre croissant et pour chacune d'elle donner l'exemple d'un algorithme (du cours ou des TD) qui a asymptotiquement cette complexité en pire cas, en temps. Répondre dans un tableau en donnant le nom de chaque algorithme. — 2 pt, 18 min

Seconde partie : problèmes — 10 points

Exercice 5 (Min et max (partiel 2007))**.**
On se donne un tableau d'entiers T, non trié, de taille N. On cherche dans T le plus petit entier (le minimum) ainsi que le plus grand (le maximum). *Si vous écrivez en C : ne vous souciez pas de la manière de rendre les deux entiers :* `return(a, b)` *où a est le minimum et b le maximum sera considéré comme correct.* — tot: 3 pt

1. Écrire un algorithme Minimum(T) (C ou pseudo-code) qui prend en entrée le tableau T et rend le plus petit de ses entiers. — 1 pt, 9 min

Pour trouver le maximum, on peut d'écrire l'algorithme Maximum(T) équivalent (en inversant simplement la relation d'ordre dans Minimum(T)).

On peut alors écrire une fonction MinEtMax(T) qui renvoie *(Minimum(T), Maximum(T))*.

2. Combien la fonction MinEtMax fera t-elle de comparaisons, exactement, sur un tableau de taille N ? — 1 pt, 9 min

On propose la méthode suivante pour la recherche du minimum et du maximum. Supposons pour simplifer que N est pair et non nul.

On considère les éléments du tableau par paires successives : $(T[0], T[1])$ puis $(T[2], T[3])$, $(T[4], T[5])$, $\ldots$ $(T[N-2], T[N-1])$.

- On copie la plus petite valeur entre $T[0]$ et $T[1]$ dans une variable Min et la plus grande dans une variable Max.
- Pour chacune des paires suivantes, $(T[2i], T[2i+1])$:
  - on trouve le minimum entre $T[2i]$ et $T[2i+1]$ et on le range dans MinLocal de même le maximum entre $T[2i]$ et $T[2i+1]$ est rangé dans MaxLocal ;
  - On trouve le minimum entre Min et MinLocal et on le range dans Min de même le maximum entre MaxLocal et Max est rangé dans Max.
- On rend Min et Max.

3. On réalise cet algorithme avec un minimum de comparaisons pour chaque étape : expliquer quelles seront les comparaisons (mais inutile d'écrire tout l'algorithme). Combien fait-on de comparaisons au total ?

1 pt
9 min

Exercice 6.

Rappel : un arbre binaire est dit complet lorsque chaque nœud a soit deux fils soit aucun fils.

1. Écrire une fonction récursive NombreGaucheDroite(x) prenant en entrée un arbre binaire de recherche et donnant le nombre de nœuds de l'arbre qui ont à la fois un fils gauche et un fils droite.

1.5 pt
13 min

2. Pour tout nombre N de nœuds est-il possible de construire un arbre binaire de recherche **complet** ayant une hauteur h en $\Omega(N)$?

1 pt
9 min

3. Écrire une fonction EstComplet(x) qui renvoie vrai si x est complet, faux sinon.

1.5 pt
13 min

4. Donner une relation entre le nombre F de feuilles (les nœuds sans fils) et le nombre I de nœuds internes (les nœuds ayant deux fils), dans un arbre binaire complet quelconque. Démontrer votre résultat.

1.5 pt
13 min

Exercice 7 (À la fois ABR et tas ?)**.**

Un tas est nécessairement un arbre binaire quasi-parfait. Est-il toujours possible d'organiser un ensemble de n clés (n quelconque) en tas max de manière à ce que cet arbre binaire soit aussi un arbre binaire de recherche ? (Justifier par un raisonnement ou un contre-exemple).

1,5 pt
13 min