IFT2015 A10 — Examen Intra

Miklós Csűrös

22 octobre 2010

Aucune documentation n'est permise. L'examen vaut 100 points.
► **Répondez à toutes les questions dans les cahiers d'examen.**

0 Votre nom (1 point)

► Écrivez votre nom et code permanent sur tous les cahiers soumis.

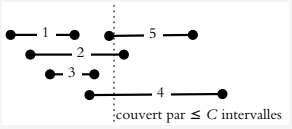
1 Tout est linéaire (10 points)

On a la récurrence $T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/3 \rfloor) + O(n)$ pour tout $n > 1$.
► Démontrez que $T(n) = O(n)$.

2 Dèque (20 points)

Le type abstrait de *dèque* est la combinaison d'une pile et d'une file FIFO avec les opérations push, pop, enqueue, dequeue. ► Montrez comment implanter un dèque à l'aide d'un tableau. Toutes les opérations doivent s'exécuter en $O(1)$. (Ignorez la question d'expansion/réduction dynamique du tableau dans l'implantation — mais expliquez [dans quelques mots] comment le faire dans votre structure.)

3 Tri d'intervalles (25 points)



On a une séquence d'intervalles $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ (stocké dans un tableau $T[1..n]$) avec $x_i < y_i$, triées par leur côté gauche : $x_1 < x_2 < x_3 < \cdots < x_n$. Les intervalles peuvent être de longueurs différentes. On sait qu'à chaque $z \in \mathbb{R}$ il y a tout au plus C intervalles chevauchantes. Dans d'autres mots, il existe tout au plus C intervalles (x_i, y_i) avec $x_i < z < y_i$.

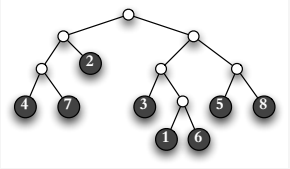
► Donnez un algorithme pour trier les intervalles par leur côté droit. L'algorithme doit trier $T[]$ en utilisant un espace de travail de $O(C)$ au plus, et en un temps $O(n \log C)$.

4 Taux de croissance (20 points)

► Remplissez le tableau suivant : chaque réponse vaut 2 point. Pour chaque paire f, g, écrivez "=" si $\Theta(f) = \Theta(g)$, "$\ll$" si $f = o(g)$, "$\gg$" si $g = o(f)$, et "???" si aucun des trois cas n'applique. Il n'est pas nécessaire de justifier vos réponses. $\lg n$ dénote le logarithme binaire de n.

	$f(n)$	$g(n)$
a	$f(n) = \sqrt{n}$	$g(n) = n^{2/3}$
b	$f(n) = 2n^2 - \log_4 n$	$g(n) = 4^{\lg n}$
c	$f(n) = 2^{2^n}$	$g(n) = 4^n$
d	$f(n) = n!$	$g(n) = n^n$
e	$f(n) = 1.12^n$	$g(n) = 1.13^n$
f	$f(n) = \lg^* n$	$g(n) = \lg \lg n$
g	$f(n) = n \lg n$	$g(n) = \lg(n!)$
h	$f(n) = \lg n$	$g(n) = \begin{cases} 1 & \{n < 3\} \\ f(\lceil n/3 \rceil) + O(1) & \{n \geq 3\} \end{cases}$
i	$f(n) = \log_{\lg n} n$	$g(n) = \frac{\lg n}{\lg \lg n}$
j	$f(n) = \lg n$	$g(n) = \begin{cases} 1 & \{n = 1\} \\ g(n-1) + 1/n & \{n > 1\} \end{cases}$

5 Arbre étiqueté (24 points)



On a un arbre binaire où les feuilles sont «étiquetées» par les valeurs $1, 2, \ldots, n$ en un ordre quelconque : il existe une bijection entre les feuilles et les étiquettes $\{1, \ldots, n\}$. (Les autres nœuds ne sont pas étiquetés.)

Implantation : pour chaque nœud x, x.left donne l'enfant gauche, x.right donne l'enfant droit, et x.etiquette donne son étiquette (quand x est une feuille).

a. (12 points) ► Donnez un algorithme MINETIQUETTE(x) qui retourne l'étiquette minimale dans le sous-arbre enraciné au nœud x fourni comme argument.

b. (12 points) On veut ordonner les enfants de chaque nœud de telle sorte que MINETIQUETTE est plus petit dans le sous-arbre gauche que dans le sous-arbre droit. Pour cela, il faut parcourir l'arbre et échanger les deux enfants quand nécessaire. ► Donnez un algorithme qui met les enfants dans cet ordre à chaque nœud en temps total $O(n)$.

BONNE CHANCE !

English translation

No documentation is allowed. The examen is worth 100 points.
Answer each question in the exam booklet.

0 Your name (1 point)

► Write your name and *code permanent* on each booklet that you submit.

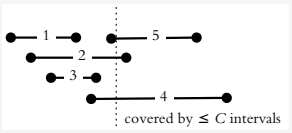
1 Everything is linear (10 points)

Consider the recurrence $T(n) \leq T(\lceil n/2 \rceil) + T(\lfloor n/3 \rfloor) + O(n)$, holding for all $n > 1$. ► Show that $T(n) = O(n)$.

2 Deque (20 points)

The abstract data type *deque* (double-ended queue) combines the stack and the queue, with operations push, pop, enqueue, dequeue. ► Show how to implement a deque by using a single table. Every operation must be done in $O(1)$. (You can ignore the dynamic table expansion/reduction in your implementation, but explain in a few words how your structure would accommodate it.)

3 Sorting intervals (25 points)



Consider a sequence of intervals $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ (stored in a table $T[1..n]$) with $x_i < y_i$, sorted by their left-hand sides :$x_1 < x_2 < x_3 < \cdots < x_n$. The intervals may have different lengths. We know that at every point $z \in \mathbb{R}$, there are at most C overlapping intervals. In other words, there are at most C intervals (x_i, y_i) with $x_i < z < y_i$.

► Give an algorithm that sorts the intervals by their right-hand side. The algorithm has to sort $T[]$ by using at most $O(C)$ additional work space, in $O(n \log C)$ time.

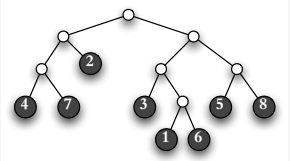
4 Growth rates (20 points)

► Fill out the following table : every answer is worth 2 points. For each pair f, g, write "=" if $\Theta(f) = \Theta(g)$, "$\ll$" if $f = o(g)$, "$\gg$" if $g = o(f)$, and "???"

if neither of the three applies. You do not need to justify the answers. $\lg n$ denotes the binary logarithm of n.

	$f(n)$	$g(n)$
a	$f(n) = \sqrt{n}$	$g(n) = n^{2/3}$
b	$f(n) = 2n^2 - \log_4 n$	$g(n) = 4^{\lg n}$
c	$f(n) = 2^{2^n}$	$g(n) = 4^n$
d	$f(n) = n!$	$g(n) = n^n$
e	$f(n) = 1.12^n$	$g(n) = 1.13^n$
f	$f(n) = \lg^* n$	$g(n) = \lg \lg n$
g	$f(n) = n \lg n$	$g(n) = \lg(n!)$
h	$f(n) = \lg n$	$g(n) = \begin{cases} 1 & \{n < 3\} \\ f(\lceil n/3 \rceil) + O(1) & \{n \geq 3\} \end{cases}$
i	$f(n) = \log_{\lg n} n$	$g(n) = \frac{\lg n}{\lg \lg n}$
j	$f(n) = \lg n$	$g(n) = \begin{cases} 1 & \{n = 1\} \\ g(n-1) + 1/n & \{n > 1\} \end{cases}$

5 Labeled tree (24 points)



You are given a binary tree in which the leaves are bijectively labeled by $\{1, 2, \ldots, n\}$. (The other nodes have no labels.)

Tree implementation : for each node x, x.left is the left child, x.right is the right child, and x.etiquette gives the label of x [when it is a leaf].

a. (12 points) ▶ Give an algorithm MINETIQUETTE(x) that returns the minimal label in the subtree of node x given as argument.

b. (12 points) We prefer to have the children of each node ordered in the sense that MINETIQUETTE is less for the left child than for the right child. For this, one needs to traverse the tree and switch the left and right children whenever necessary. ▶ Give an algorithm that orders the children at every node in $O(n)$ overall time.

GODSPEED !