Partiel d'Eléménts d'Algorithmique (EA4)

L2 d'informatique — durée 2h

Mardi 24 mars 2009

La rédaction sera prise en compte dans la notation. Justifiez toutes vos réponses. Le barême est donné seulement à titre indicatif.

Exercice 1 (4 points)
Pour les fonctions f et g suivantes, dire si $f \in O(g)$ et si $g \in O(f)$. On ne fera recours à la méthode qui utilise les limites et la règle de l'Hopital que si vous êtes incapables de fournir un autre argument. La fonction ln désigne le logarithme naturel (en base e).

1. $f(n) = \ln n$ et $g(n) = \sqrt{n}$;
2. $f(n) = e^n$ et $g(n) = n^n$;
3. $f(n) = \frac{n^2}{\ln n}$ et $g(n) = n^{3/2}$;

Exercice 2 (4 points)
Simuler, en détaillant les étapes, l'algorithme de tri Shell sur le tableau :

$$[1, 5, 3, 6, 10, 55, 2, 87, 12, 34, 75, 33, 47]$$

en utilisant la suite de sauts définie par $h_n = 3h_{n-1} + 1$ avec $h_1 = 1$.

Exercice 3 (7 points)
Soit A un tableau d'entiers unimodal d'éléments tous distincts, (on rappelle qu'un tableaux est unimodal s'il est constitué *exactement* de deux suites monotones). Le tableau A possède donc soit un unique pic soit un unique creux. Vous devez donner un algorithme aussi rapide que possible et idéalement en $O(\log n)$ pour calculer la position du creux ou du pic.

Simulez votre algorithme sur les tableaux suivants et vérifiez qu'il donne la bonne réponse :

- $[2, 9, 8, 7, 6, 5, 4, 3, 1]$
- $[1, 3, 4, 7, 8, 9, 6, 5, 2]$
- $[4, 7, 8, 9, 6, 5, 3, 2, 1]$
- $[2, 9, 8, 7, 6, 5, 4, 3, 1]$
- $[2, 3, 5, 8, 9, 7, 6, 3, 1]$

Prouvez la correction de votre algorithme (on donnera un invariant si l'algorithme est itératif, ou une preuve par récurrence s'il est récursif). Justifier sa complexité.

Si le tableau a un pic, ce pic est le maximum du tableau. Peut-on ajouter une instruction en $O(1)$ à votre algorithme pour qu'il retourne aussi la position du

minimum ? Et symétriquement, dans le cas où il y a un creux, peut-on calculer la position du maximum en $O(1)$?

Exercice 4 (5 points)
Ecrire un algorithme `booléen doublon(t : tableau [1..n] d'entiers)` permettant de déterminer si le tableau `t` contient un doublon, c'est-à-dire si il y a un entier qui apparait au moins 2 fois dans le tableau. Quelle est sa complexité ? Donner ensuite un deuxième algorithme qui fait la même chose et dont la fonction de complexité est asymptotiquement meilleure que celle du premier algorithme (dans le sens qu'elle appartient à une classe O strictement plus petite).