Partiel d'algorithmique

L2 d'informatique — durée 2h

Vendredi 27 mars 2009

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

Exercice 1 (4 points)
Comparer les fonctions f et g suivantes pour les relations o, O et θ. La fonction $\ln$ désigne le logarithme naturel (en base e).

1. $f(n) = n \ln n$ et $g(n) = n^2$;
2. $f(n) = 2^{n^2}$ et $g(n) = 2^{2n^2}$;
3. $f(n) = n \ln(4n^5)$ et $g(n) = n \ln n$;
4. $$f(n) = n \quad \text{et} \quad g(n) = \begin{cases} 2^n & \text{si } n^3 < 15n^2 \\ 3n+2 & \text{sinon} \end{cases}$$

Exercice 2 (3 points)
Quelle est la valeur renvoyée par l'algorithme suivant, en fonction de n ? Quelle est la complexité de l'algorithme ?

```
Enigme (n : entier)
  x := 0
  Pour i de 1 à n faire
    x := x+2
    Pour j de 1 à n faire
      Pour k de 1 à j faire
        x := x+1
  Renvoyer x
```

Exercice 3 (5 points)
Soit A un tableau d'entiers dont les éléments sont d'abord strictement croissants, puis strictement décroissants (c'est-à-dire qu'il existe $i \in \{1, \ldots, n\}$ tel que $A[1..i]$ est strictement croissant et $A[i..n]$ est strictement décroissant). Par exemple, $[1, 2, 4, 8, 7, 2]$ est un tel tableau (on notera que les éléments de la partie croissante sont distincts, ainsi que ceux de la partie décroissante, mais il peut y avoir des valeurs communes entre la partie croissante et la partie décroissante).

Proposer un algorithme logarithmique pour trouver le **maximum** d'un tel tableau. 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.

Proposer un algorithme pour trouver le **minimum** d'un tel tableau. Justifier brièvement pourquoi il est correct et évaluer sa complexité.

Exercice 4 (3 points)
Considérons l'algorithme suivant de tri d'un tableau $T[i..j]$.

```
Tri(T,i,j)
  Si i >= j alors
    sortir
  k := i
  p := T[i]
  Pour l de i+1 à j faire
    Si T[l] <= p alors
      T[k] := T[l]
      T[l] := T[k+1]
      T[k+1] := p
      k++
  Tri(T,i,k-1)
  Tri(T,k+1,j)
```

Exécuter cet algorithme sur le tableau $T = [5, 9, 1, 6, 3, 2, 4]$ avec $i = 1$ et $j = 7$ en indiquant l'état du tableau avant chaque appel récursif. Quel est le nom de cet algorithme ? Quel est sa complexité dans le pire cas ? Décrire les pires cas. Comment peut-on utiliser l'aléatoire pour éviter ces pires cas et que devient la complexité moyenne de l'algorithme (pas de justification demandée) ?

Exercice 5 (5 points)
Donner un algorithme en $O(n \log n)$ pour décider s'il existe trois valeurs entières consécutives dans un tableau d'entiers distincts. Par exemple, l'algorithme répondra Vrai sur le tableau $[2, 6, 1, 5, 7]$ (car il contient les trois entiers consécutifs 5, 6 et 7) et Faux sur le tableau $[3, 1, 5, 6]$. Justifier la complexité de l'algorithme et prouver sa correction en utilisant un invariant.