Structures de données et algorithmes

Répétition 2: Outils d'analyse

Jean-Michel Begon – http://www.montefiore.ulg.ac.be/~jmbegon

26 février 2016

Exercice 1

L'algorithme suivant est-il correct ? Partiellement ? Totalement ?

```
// {C ≥ 0}
x = A
y = C
product = 0
while y > 0
      product = product + x
      y = y − 1
// {product = A × C}
```

Exercice 2

Montrer que la fonction récursive $g(n)$ retourne $3^n - 2^n$ pour tout $n \geq 0$.

```
g(n)
if n ≤ 1
      return n
else
      return 5g(n − 1) − 6g(n − 2)
```

Exercice 3

(a) L'algorithme A nécessite $10n^3$ opérations pour résoudre un problème. L'algorithme B résoud le même problème en $1000n^2$ opérations. Quel est l'algorithme le plus rapide ?

(b) L'algorithme A nécessite $32n \log_2 n$ opérations pour résoudre un problème. L'algorithme B résoud le même problème en $3n^2$ opérations. Quel est l'algorithme le plus rapide ?

Exercice 4

Classer ces fonctions par ordre croissant de complexité (selon l'opérateur $O(.)$).

$n \log_2 n$	$\frac{4}{n}$	$\sqrt{n}$	2^{2^n}
$\log_2 \log_2 n$	$8n^3$	$8^{\ln n}$	$\frac{n}{2+n}$
$\log_2 n^7$	$5^{\ln \log_2 n}$	$(\log_2 n)^3$	$\frac{n}{\log_2(2+n)}$

Exercice 5

(a) Montrer que $2n + 100$ est $\Theta(n)$.

(b) Montrer que $5n^2 + 500n + 5000$ est $\Theta(n^2)$.

(c) Montrer que 2^{n+1} est $\Theta(2^n)$.

(d) Expliquer pourquoi la phrase *"Le temps d'exécution d'un algorithme A est au moins $O(n^2)$"* n'a aucun sens.

(e) Montrer que le temps d'exécution d'un algorithme est $\Theta(g(n))$ si et seulement si le temps d'exécution du pire cas est $O(g(n))$ et le temps d'exécution du meilleur cas est $\Omega(g(n))$.

Exercice 6

Quelle serait la complexité minimum d'un algorithme qui imprime à l'écran :

(a) Toutes les sous-chaînes d'une chaîne de caractères.

(b) Tous les sous-ensembles de caractères d'une chaîne.

Exercice 7

Pour chacun des pseudo-codes suivants, déterminer ce que fait l'algorithme, puis la complexité asymptotique (en termes de $\Theta(.)$ et de n).

```
Code1(n)
limit = n * n
sum = 0
for i = 1 to limit
    sum = sum + 1
return sum
```

```
Code2(n)
i = 1
limit = n * n * n
sum = 0
while i < limit
    sum = sum + 1
    i = i * 2
return sum
```

```
Code3(n)
limit = n * n
sum = 0
for i = 1 to limit
    for j = 1 to i
        sum = sum + 1
return sum
```

```
Code4(n)
if n ≤ 1
    return n
else
    return Code4(n - 1) + Code4(n - 1)
```

```
Code5(a, b, c, n)
for i = 1 to n
    for j = 1 to n
        a[i][j] = 0
        for k = 1 to n
            a[i][j] = a[i][j] + b[i][k] * c[k][j]
```

```
Code6(n)
if n == 0
    return ""
else
    tmp = Code6(n/2)
    if n%2 == 0
        return tmp + tmp
    else
        return tmp + tmp + "x"
```

```
Code7(n)
s = ""
for i = 1 to n
    s = s+"x"
return s
```

Exercice 8

Soit un tableau de N entiers où chaque entier de l'intervalle $1..N$ apparaît exactement une fois, à l'exception d'un entier apparaîssant 2 fois et d'un entier manquant. Proposer un algorithme linéaire pour trouver l'entier manquant, en utilisant au plus $\Theta(1)$ d'espace mémoire supplémentaire.

Exercice 9

On se propose de coder une fonction polynomiale, de la forme :

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

(a) Quelle est la complexité d'un algorithme implémentant $p(x)$ sous la forme présentée ci-dessus ? (En utilisant uniquement des additions et des multiplications.)

(b) Proposer un algorithme linéaire.