Structures de données et algorithmes

Répétition 2: Outils d'analyse

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

21 février 2020 — 28 février 2020

Exercice 1

1. 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 ?
2. 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 2

Soit $f(n) = n$. Pour autant que ça soit possible, trouvez une fonction $g(n)$ telle que
— $f(n) \in O(g(n))$ et $f(n) \notin \Omega(g(n))$
— $f(n) \notin O(g(n))$ et $f(n) \in \Omega(g(n))$
— $f(n) \in O(g(n))$ et $f(n) \in \Omega(g(n))$
— $f(n) \notin O(g(n))$ et $f(n) \notin \Omega(g(n))$

Exercice 3

1. Montrez 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 $\Omega(g(n))$.
2. Montrez que $5n^2 - 3n + 4$ est $\Theta(n^2)$.
3. Montrez que 2^{n+1} est $\Theta(2^n)$.
4. Expliquez pourquoi la phrase *"Le temps d'exécution d'un algorithme A est au moins $O(n^2)$"* n'a aucun sens.

Exercice 4

Soit un algorithme dont le temps d'exécution pour $N = 1000, 2000, 3000$ et 4000 est respectivement de $5s$, $20s$, $45s$ et $80s$. Estimez le temps d'exécution pour $N = 5000$ et donnez sa complexité en temps.

Exercice 5

Classez les fonctions suivantes 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 6

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

```
CODE1(n)
1  limit = n * n
2  sum = 0
3  for i = 1 to limit
4        sum = sum + 1
5  return sum
```

```
CODE2(n)
1  limit = n * n
2  sum = 0
3  for i = 1 to limit
4        for j = 1 to i
5              sum = sum + 1
6  return sum
```

```
CODE3(n)
1  i = 1
2  limit = n * n * n
3  sum = 0
4  while i < limit
5        sum = sum + 1
6        i = i * 2
7  return sum
```

```
CODE4(a, b, c, n)
1  for i = 1 to n
2        for j = 1 to n
3              a[i][j] = 0
4              for k = 1 to n
5                    a[i][j] = a[i][j] + b[i][k] * c[k][j]
```

Exercice 7

Soit l'algorithme suivant permettant de calculer la somme des éléments d'un tableau A entre les positions p et q incluses :

```
SUM(A, p, q)
1  if q < p
2        return 0
3  n = q - p + 1
4  d = ⌊n/3⌋,   r = p + d,   s = q - d
5  sum = 0
6  for i = r to s
7        sum = sum + A[i]
8  return SUM(A, p, r - 1) + sum + SUM(A, s + 1, q)
```

1. Fournissez une borne inférieure pour ce problème.
2. Formulez récursivement la complexité en temps de cet algorithme.
3. Sur base de cette formulation, quelle serait l'ordre de complexité d'après le *Master theorem* ?
4. Vérifiez cette complexité par la méthode de l'arbre, ainsi qu'en développant l'expression récursive.

Exercice 8

Solutionner par la méthode de l'arbre et par plug-and-chuck la récurrence suivante :

$$T(n) = \begin{cases} 0, & \text{si } n = 1 \\ 3T(n/4) + n^2, & \text{sinon} \end{cases} \tag{1}$$

Exercice 9

Pour chacun des pseudo-codes suivants, déterminez ce que fait l'algorithme, puis la complexité asymptotique (en termes de $\Theta(.)$ et de n). Commencez par établir l'équation de récurrence correspondant à la complexité.

```
CODE5(n)
1  if n ≤ 1
2        return n
3  else
4        return CODE5(n − 1) + CODE5(n − 1)
```

```
CODE6(n)
1  if n ≤ 1
2        return n
3  else
4        return CODE6(n − 1) ∗ 2
```

```
CODE7(x, n)
1  if n == 0
2        return 1
3  if n is even
4        y = CODE7(x, n/2)
5        return y ∗ y
6  else
7        return x ∗ CODE7(x, n − 1)
```

Exercice 10

Soient deux algorithmes récursifs dont les complexités sont données respectivement par :
— $T_1(n) = T_1(n/2) + 1$,
— $T_2(n) = T_2(n/3) + 1$.
Est-il vrai qu'on a $T_1(n) \in O(T_2(n))$?

Exercice 11

Soit un tableau de N entiers où chaque entier de l'intervalle $1..N$ apparaît exactement une fois, à l'exception d'un entier apparaissant deux fois et d'un entier manquant.

1. Proposez un algorithme au plus quadratique en temps pour trouver l'entier manquant, en utilisant au plus $\Theta(1)$ d'espace mémoire supplémentaire.
2. Proposez un algorithme linéaire en temps et en espace pour trouver l'entier manquant.
3. Proposez un algorithme linéaire en temps pour trouver l'entier manquant, en utilisant au plus $\Theta(1)$ d'espace mémoire supplémentaire.