# Structures de données et algorithmes

Répétition 2: Outils d'analyse

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

20 février 2015

## Exercice 1

L'algorithme suivant échange les valeurs $x$ et $y$. Montrer que cet algorithme est correct.

Swap$(x, y)$
1 $x = x + y$
2 $y = x - y$
3 $x = x - y$

## Exercice 2

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

1 // $\{C \geq 0\}$
2 $x = A$
3 $y = C$
4 $product = 0$
5 **while** $y > 0$
6 $\quad product = product + x$
7 $\quad y = y - 1$
8 // $\{product = A \times C\}$

## Exercice 3

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

$g(n)$
1 **if** $n \leq 1$
2 $\quad$ **return** $n$
3 **else**
4 $\quad$ **return** $5g(n-1) - 6g(n-2)$

## Exercice 4

(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 5

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 $N = 5000$.

## Exercice 6

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 7

(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 8

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 9

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

```
CODE2(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
```

```
CODE3(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
```

```
CODE4(n)
1  if n ≤ 1
2       return n
3  else
4       return CODE4(n - 1) + CODE4(n - 1)
```

```
CODE5(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]
```

```
CODE6(n)
1  if n == 0
2       return ""
3  else
4       tmp = CODE6(n/2)
5       if n%2 == 0
6            return tmp + tmp
7       else
8            return tmp + tmp + "x"
```

```
CODE7(n)
1  s = ""
2  for i = 1 to n
3       s = s+"x"
4  return s
```

## Exercice 10

Soit $A$, un tableau de $n$ valeurs classées par ordre croissant. On se propose de rechercher si une valeur $b$ est présente dans ce tableau.

(a) Ecrire le pseudo-code d'un algorithme brutal pour rechercher la valeur $b$. Analyser sa complexité dans le meilleur cas et dans le pire cas.

(b) Proposer un algorithme dichotomique pour trouver la valeur $b$. Analyser sa complexité dans le meilleur cas et dans le pire cas.

## 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 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 12

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.

## Exercice 13

Une attaque par *déni de service* (DoS) est une attaque visant à saturer une application de sorte qu'elle ne puisse plus répondre à de nouvelles requêtes. Typiquement, un pirate essaiera de saturer l'application par une multiplication de requêtes lourdes en temps de calcul. Un exemple classique est celui de Apache qui, en 1997, intégrait la fonction suivante :

```
void no2slash(char *name) {
    register int x,y;

    for(x=0; name[x];)
        if(x && (name[x-1] == '/') && (name[x] == '/'))
            for(y=x+1;name[y-1];y++)
                name[y-1] = name[y];
        else x++;
}
```

(a) Que fait cette fonction ?

(b) Quelle est sa complexité dans le pire des cas ?

(c) Comment pouvait-on donc faire pour saturer un serveur web Apache ?

(d) Le patch proposé pour corriger cette faille de sécurité implémentait `no2slash` de la façon ci-après. Quelle est la complexité de cette seconde fonction ? La faille est-elle corrigée ?

```
void no2slash(char *name) {
    char *d, *s;

    s = d = name;
    while (*s) {
        if ((*d++ = *s) == '/') {
            do {
                ++s;
            } while (*s == '/');
        }
        else {
            ++s;
        }
    }
    *d = '\0';
}
```