CORRECTION PARTIEL : Algorithme et Programmation Université Paris-Dauphine MIDO L2

Exercice 1 ((2 points)) *Combien vaut x ?*

$$\left(\sqrt{\frac{3}{5}}\right)^{7x+1} = \frac{125}{27}$$

[2 points] $x = -\frac{1}{21}$.

Exercice 2 ((2 points)) *Soient a, b des constantes entières ≥ 2 et n une variable entière. Rappelez la définition de $\log_b(n)$ et démontrez que cette définition implique*

1. $\log_b n = \log_b a \times \log_a n$.
2. $a^{\log_b n} = n^{\log_b a}$.

Def. $u = \log_b n \Leftrightarrow b^u = n$.
[1 point] Soient u, v, w tels que $b^u = n$, $b^v = a$ et $a^w = n$, alors $n = (b^v)^w = b^{vw}$, d'où $u = vw$ puisque $n = b^u$.
[1 point] De plus, $a^u = a^{vw} = (a^w)^v = n^v$.

Exercice 3 ((3 points)) *Démontrez que* $\Theta\Big(f(n)\Theta\big(g(n)\big)\Big) = \Theta\Big(f(n)g(n)\Big)$.

(Les fonctions étudiées sont positives et croissantes.)

[2 points] En général, pour deux fonctions f, g on a $f(n) = \Theta(g(n))$ si et seulement si $g(n) = \Theta(f(n))$, en effet, $\alpha g(n) \leq f(n) \leq \beta g(n)$ pour $\alpha, \beta > 0$ est équivalent à $\beta^{-1}f(n) \leq g(n) \leq \alpha^{-1}f(n)$ pour $\alpha^{-1}, \beta^{-1} > 0$. Il suffit donc de montrer que $f(n)\Theta(g(n)) = \Theta(f(n)g(n))$.

[1 point] Soit $h(n) = \Theta(g(n))$ avec $\alpha, \beta > 0$ tels que : $\alpha g(n) \leq h(n) \leq \beta g(n)$ pour $n > n_0$. Alors $\alpha f(n)g(n) \leq f(n)h(n) \leq \beta f(n)g(n)$ pour $n > n_0$. D'où $f(n)\Theta(g(n)) = \Theta(f(n)g(n))$.

Exercice 4 ((3 points)) *Soit $f : \mathbb{R}^+ \to \mathbb{R}^+$ une fonction croissante et des constantes $a, c \geq 0$ et $b > 1$. Démontrez que*

1. *si $f(b^p) = \Theta(b^{pc})$ pour tout $p \in \mathbb{N}$, alors $f(x) = \Theta(x^c)$ pour tout $x \in \mathbb{R}^+$.*
2. *si $f(b^p) = \Theta(p)$ pour tout $p \in \mathbb{N}$, alors $f(x) = \Theta(\log x)$ pour tout $x \in \mathbb{R}^+$.*

(Les fonctions étudiées sont positives et croissantes.)

1. [1 point] Soient α, β telles que $\alpha b^{cp} \leq f(b^p) \leq \beta b^{cp}$. Pour tout $x > 0$, il existe un entier p tel que $b^p \leq x < b^{p+1}$. D'où

$$\alpha b^{cp} \leq f(b^p) \leq f(x) \leq f(b^{p+1}) \leq \beta b^{cp+c} = \beta' b^{cp}$$

avec $\beta' = \beta b^c$.

2. [1 point] Soient α, β telles que $\alpha p \leq f(b^p) \leq \beta p$. On a $p \leq \log x < p + 1$, d'où

$$\alpha p \leq f(p) \leq f(x) \leq f(b^{p+1}) \leq \beta(p+1) \leq \beta' p$$

avec $\beta' = 2\beta$ [1 point] pour $p \geq 1$ c'est-à-dire pour $x \geq b$.

Exercice 5 ((5 points)) *Donnez trois implémentations différentes en python d'un algorithme qui retourne x^n où x et n sont des entiers, de complexité*

1. `powerlin(x,n)` *en* $\Theta(n)$,
2. `powerlog(x,n)` *en* $\Theta(\log n)$,
3. `powerexp(x,n)` *exponentielle.*

1. [1 point] On peut répondre avec un algorithme récursif ou itératif,

```
def powerlin(x,n):
        if n==0:
                return 1
        else:
                y=powerlin(x,n-1)
                return y*x
```

ou

```
def powerlin(x,n):
        y=1
        for i in range(n):
                y = x * y
        return y
```

2. [2 points] Il faut utiliser un algorithme récursif du paradigme $T(n) = aT(n/b) + \Theta(n^c)$ avec $a = 1$, $b \geq 2$ et $c = 0$. Le plus naturel est $b = 2$ avec:

```
def powerlog(x,n):
        if n==0:
                return 1
        else:
                y=powerlog(x,n//2)
                if n%2==0:
                        return y*y
                else:
                        return y*y*x
```

2. [2 points] Il faut utiliser un algorithme récursif du paradigme $T(n) = aT(n-b) + \Theta(n^c)$ avec $a \geq 2$. Par exemple:

```
def powerexp(x,n):
        if n==0:
                return 1
        else:
                y=powerexp(x,n-1)
                y=powerexp(x,n-1)
                return y*x
```

Exercice 6 ((5 points)) *Soient des entiers $a, b, c \geq 1$. Démontrez que si $T(n)$ satisfait:*

$$T(n) = \begin{cases} \Theta(1) & si\ n < b \\ aT(n-b) + \Theta(n^c) & si\ n \geq b \end{cases}$$

alors $T(n) = \Theta\Big(a^q + \sum_{i=1}^{q} a^{q-i}(r+ib)^c\Big)$ où $n = qb + r$ avec $0 \leq r < b$ et q, b entiers.

On note $f(n) = a^q + \sum_{i=1}^{q} a^{q-i}(r+ib)^c$, et α', β', α'', β'' les constantes des fonctions $\Theta(n^c)$ et $\Theta(1)$.

[3 points] On définit $\alpha = \min\{1, \alpha', \alpha''\}$ et $\beta = \max\{1, \beta', \beta''\}$, ainsi puisque $f(0.b + r) = a^0 = 1$ et $T(r) = \Theta(1)$, on a

$$\alpha f(0.b + r) \leq T(0.b + r) \leq \beta f(0.b + r)$$

Par ailleurs si

$$\alpha f(qb + r) \leq T(qb + r) \leq \beta f(qb + r)$$

alors

$$\alpha f((q + 1)b + r) \leq T((q + 1)b + r) \leq \beta(f((q + 1)b + r)$$

puisque

[2 points]

$$
\begin{array}{rcccl}
aT(qb+r)+\alpha n^c & \leq & T(n) & \leq & aT(qb+r)+\beta n^c \\
a\alpha\Big(a^q+\sum_{i=1}^{q} a^{q-i}(r+ib)^c\Big)+\alpha n^c & \leq & T(n) & \leq & a\beta\Big(a^q+\sum_{i=1}^{q} a^{q-i}(r+ib)^c\Big)+\beta n^c \\
\alpha\Big(a^{q+1}+\sum_{i=1}^{q} a^{q+1-i}(r+ib)^c+\big((q+1)b+r\big)^c\Big) & \leq & T(n) & \leq & \beta\Big(a^{q+1}+\sum_{i=1}^{q} a^{q+1-i}(r+ib)^c+\big((q+1)b+r\big)^c\Big) \\
\alpha\Big(a^{q+1}+\sum_{i=1}^{q+1} a^{q+1-i}(r+ib)^c\Big) & \leq & T(n) & \leq & \beta\Big(a^{q+1}+\sum_{i=1}^{q+1} a^{q+1-i}(r+ib)^c\Big)
\end{array}
$$

où l'on a noté $n=(q+1)b+r$.