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

Exercice 1 *Combien valent* $\log_3(9)$, $\log_2(1024)$, $\log_3(27)$, $\log_2(101372)$, $\log_{10}(1000000000000)$ *?*

$3^2 = 9$, $2^{10} = 1024$, $3^3 = 27$ et $10^{12} = 1000000000000$ donc $\log_3(9) = 2$, $\log_2(1024) = 10$, $\log_3(27) = 3$, et $\log_{10}(1000000000000) = 12$. (1 point)
Puisque $2^6 = 64$, on a $2^{16} < 1024 \times 70 < 80000 < 101372 < 2 \times 60000 < 2^{17}$. Donc $16 < \log_2(101372) < 17$. (1 point)

Exercice 2 *Montrez que* $a^{\log_b n} = n^{\log_b a}$.

Soient $b^x = n$ et $b^y = a$, alors $n^{\log_b a} = n^y = b^{xy} = b^{yx} = a^{\log_b n}$ (1 point).

Exercice 3 *Montrez que* $\log_b a = \frac{1}{\log_a b}$.

Soient $b^x = a$ et $a^y = b$, alors $b^x = a^{xy} = a$ d'où, puisque a est quelconque, $xy = 1$ et donc $\log_b a \times \log_a b = 1$ (1 point).

Exercice 4 *Montrez que* `fibo rec` *est* $\Omega(1.41^n)$.

```
def fibo_rec(n):
    if n <= 1:
        return 1
    else:
        return fibo_rec(n-1) + fibo_rec(n-2)
```

Soit $T(n)$ le temps d'exécution de `fibo rec`, alors $T(n) \geq T(n-1) + T(n-2) \geq 2T(n-2)$. Avec $2k \leq n \leq 2k+1$, on a $T(n) \geq 2^k T(0)$. Puisque $T(0) = \Theta(1)$, alors $T(n) \geq \Theta(2^k) = \Theta(2^{\frac{n}{2}}) = \Theta({2^{\frac{1}{2}}}^n) = \Theta(\sqrt{2}^n) = \Omega(1.41^n)$ (2 points).

Exercice 5 *Donnez, très précisément, l'affichage écran à l'appel de* `P(6)`.

```
def P(n):
    B=[]
    C=[]
    for i in range(n+1):
        B.append(1)
        C.append(1)
    for i in range(2,n+1):
        for j in range(1,i):
            C[j] = B[j-1] + B[j]
        print(*C)
        for k in range(n+1):
            B[k]=C[k]
```

1 2 1 1 1 1 1
1 3 3 1 1 1 1
1 4 6 4 1 1 1
1 5 10 10 5 1 1
1 6 15 20 15 6 1
(1 point)

Exercice 6 *Donnez un programme python* `binom(n,p)` *d'au-plus 12 lignes retournant le coefficient binomial* $\frac{n!}{(n-p)!p!}$ *sans effectuer ni multiplication ni division.*

```
def binom(n,p):
    B=[]
    C=[]
    for i in range(n+1):
        B.append(1)
        C.append(1)
    for i in range(2,n+1):
```

```
        for j in range(1,i):
            C[j] = B[j-1] + B[j]
        for k in range(n+1):
            B[k]=C[k]
    return B[p]
```

Solution aussi tolérée (mais c'est une mauvaise solution car la complexité est désastreuse):

```
def C(n,p):
    if p==n or p==0:
        return 1
    return C(n-1,p-1)+C(n-1,p)
```

(1 point)

Exercice 7 *Donnez les deux lignes manquantes de* `tri insertion`*.*

```
def tri_insertion(tab):
    for j in range(1,len(tab)):
        // solution =                                      x=tab[j]
        i=j-1
        while i>=0 and tab[i]>x:
            tab[i+1]=tab[i]
            // solution =                         i=i-1   ou   i-=1
        tab[i+1]=x
```

(1 point)

Exercice 8 *Donnez les deux lignes manquantes de* `tri fusion`*.*

```
def tri_fusion(tab):
     n=len(tab)
     if n >1:
        mi = n//2
        L = tab[:mi]
        R = tab[mi:]
        tri_fusion(L)
        // solution =                               tri_fusion(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                tab[k] = L[i]
                i+= 1
            else:
                tab[k] = R[j]
                j+= 1
            k+= 1
        while i < len(L):
            tab[k] = L[i]
            i+= 1
            k+= 1
        while j < len(R):
            tab[k] = R[j]
            // solution =                         j+= 1   ou   j=j+1
            k+= 1
```

(1 point)

Exercice 9 *Soit $T(n) = aT(n/b) + n^c$ avec $T(1) = 1$ le temps d'exécution d'un algorithme divide-and-conquer en fonction de la taille $n = b^t$ des données en entrée.*

1. *Donnez l'expression exacte de $T(b^3)$ en fonction de a, b, c.*
2. *Donnez $f(a, b, c, t)$ telle que $T(n) = f(a, b, c, t)$.*
3. *Donnez la raison $r(a, b, c)$ de la suite géométrique dont $f(a, b, c, t)/n^c$ est la somme des $t + 1$ premiers termes.*
4. *Montrez que si $r(a, b, c) > 1$, alors $T(n) > n^{\log_b a}$.*

1. $T(b^3) = aT(b^2) + b^{3c} = a(aT(b) + b^{2c}) + b^{3c} = a(a(a(T(1) + b^c) + b^{2c}) + b^{3c} = a^3 + a^2b^c + ab^{2c} + b^{3c}$ (0.5 point)
2. $f(a, b, c, t) = \sum_{i=0}^{i=t} a^{t-i}b^{ic} = \sum_{i=0}^{i=t} a^i b^{(t-i)c}$ (0.5 point)
3. $f(a, b, c, t)/n^c = \frac{\sum_{i=0}^{i=t} a^i b^{(t-i)c}}{b^{tc}} = \sum_{i=0}^{i=t} (\frac{a}{b^c})^i$ donc $f(a, b, c, t)/n^c$ est la somme des termes d'une suite géométrique de raison $r(a, b, c) = \frac{a}{b^c}$ (1 point)
4. Puisque $\frac{a}{b^c} > 0$, alors $T(n) = f(a, b, c, t) = n^c \times \sum_{i=0}^{i=t} (\frac{a}{b^c})^i > n^c(\frac{a}{b^c})^t = b^{tc}\frac{a^t}{b^{tc}} = a^t = a^{\log_b n} = n^{\log_b a}$ (1 point)

Exercice 10 *On élabore un algorithme divide-and-conquer pour trouver le produit $C = AB$ de deux matrices carrée AB de taille 2^t à partir de produits de sous-matrices de tailles 2^{t-1}. Pour améliorer la complexité, au lieu d'effectuer les 8 appels récursifs de l'expression*

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

on détermine la matrice C à partir de seulement 7 produits $P_1, \ldots, P_7$ satisfaisant

$$\begin{aligned} C_{12} &= P_5 + P_3 \\ C_{21} &= P_2 + P_4 \\ C_{11} &= P_1 + P_4 - P_5 + P_7 \\ C_{22} &= P_1 + P_3 - P_2 + P_6 \end{aligned}$$

Déterminer l'expression des 7 produits à l'aide du schéma ci-dessous.

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & + & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ - & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} \cdot & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & + & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & - \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} + & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & - \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} - \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & + & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & - \\ \cdot & + & \cdot & - \end{pmatrix}$$

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \end{pmatrix} = \begin{pmatrix} + & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ - & \cdot & \cdot & \cdot \end{pmatrix} - \begin{pmatrix} \cdot & \cdot & + & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} - & \cdot & + & \cdot \\ - & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ P_2 &= (A_{21} + A_{22})(B_{11}) \\ P_3 &= (A_{11})(B_{12} - B_{22}) \\ P_4 &= (A_{22})(B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{12})(B_{22}) \\ P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

(1 point)

Exercice 11 *Donnez un algorithme en $O(n^{1.59})$ pour calculer X^n où X est la matrice*

$$X = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

Il faut effectuer récursivement le calcul de $X^n = X^{2k} = (X^k)^2$ ou $= X^{2k+1} = X(X^k)^2$ en effectuant les multiplications avec l'algorithme de Karatsuba. Le temps d'exécution $T(n)$ de cet algorithme satisfait $T(n) = T(n/2) + \Theta(n^{\log_2 3}) = \Theta(n^{\log_2 3}) = O(n^{1.59})$ (2 points)

Exercice 12 *Représentez toutes les comparaisons possibles effectuées, et toutes les permutations déterminées par l'algorithme de tri insertion sur un tableau de trois entiers distincts a, b, c.*

$a < b$

$b < c$ $a < c$

abc $a < c$ bac $b < c$

acb cab bca cba

(1 point)

Exercice 13 *Représentez toutes les comparaisons possibles effectuées, et toutes les permutations déterminées par l'algorithme de tri fusion sur un tableau de trois entiers distincts a, b, c.*

$b < c$

$a < b$ $a < c$

abc $a < c$ acb $a < b$

bac bca cab cba

Solution également acceptée venant d'une séparation $ab|c$ au lieu de $a|bc$

$a < b$

$a < c$ $b < c$

$b < c$ cab $a < c$ cba

abc acb bac bca

(1 point)

Exercice 14 *Quelle est la complexité minimum d'un algorithme de tri comparatif ?*

Le temps minimum $T(n)$ d'un algorithme de tri comparatif est égal à la hauteur $h(n)$ minimum d'un arbre binaire ayant $n!$ feuilles. Puisque $n! \leq \Theta(2^{h(n)})$, alors $h(n) \geq \Theta(\log n!) = \Theta(n \log n)$. Par-ailleurs, le tri-fusion est un algorithme de tri comparatif en $\Theta(n \log n)$, donc $T(n) = \Theta(n \log n)$. (2 points)