CORRECTION PARTIEL : Algorithme et Programmation 3 Université Paris-Dauphine MIDO L2

Exercice 1 *Combien d'opérations élémentaires effectuent ces programmes en python ?*

```
def A():
     a=5
     while a > 0:
          a = a - 1
def B():
    for i in range(7):
         A()
def C(n):
    a = 1
    for i in range(1,n+1):
        a = i * a
    return a
```

A(): $a = 5, \ldots, 0$ donc 6 affectations, 6 tests et 5 soustractions, font 17. B(): $i = 0, \ldots, 7$ donc 8 affectations, 8 tests et 7 additions, donc 23, plus $7 \times 17 = 119$, font 142. C(n): $a = 1, 1a, 2a, \ldots, na$ et $i = 1, \ldots, n+1$ donc $2(n+1)$ affectations, n multiplications, n additions, et $n+1$ tests, font $5n+3$.

Exercice 2 *Soit un entier n. Comparez n, $2^{\lfloor \log_2 n \rfloor}$ et $2^{\lfloor \log_2 n \rfloor + 1}$.*

$2^{\lfloor \log_2 n \rfloor} \leq n < 2^{\lfloor \log_2 n \rfloor + 1}$

Exercice 3 *Montrez que $\log_b n = \log_b a \times \log_a n$.*

Soient les réels x, y, z tels que $b^x = n$, $b^y = a$ et $a^z = n$ ainsi $b^x = a^z = b^{yz} = b^{yz}$ et $x = yz$ découle de la bijectivité de la fonction puissance; ce qui implique l'égalité demandée (par définition de log).

Exercice 4 *Démontrez que $\log_b n = O(\log_2 n)$ pour toute base $b > 1$.*

Il suffit de montrer qu'il existe une constante $c > 0$ telle que $\log_b n \leq c \log_2 n$, ce qui découle de l'exercice précédent puisque, avec $a = 2$, il implique: $\log_b n = \log_b 2 \times \log_2 n$.

Exercice 5 *Codez 1023 en base 2.*

$1023 = 1024 - 1 = 2^{10} - 1 = \sum_{i=0}^{9} 2^i$, donc 1111111111 en base 2.

Exercice 6 *On considère le programme python suivant:*

```
def D(n,b):
    tab=[]
    while n>0:
        tab=[n%b]+tab
        n=n//b
    return tab
```

1. *Donnez l'équation reliant les composantes, notées $a_0, a_1, \ldots, a_{k-1}$, du tableau renvoyé par* `D(n,b)` *et les entiers n, b.*
2. *Donnez l'équation reliant les entiers n, b, k.*

1: $n = \sum_{i=0}^{k-1} a_{k-1-i} b^i$. 2: $k = \lfloor \log_b n \rfloor + 1$.

Exercice 7 *Montrez que, pour $c > 0$ et $f(n) = 1 + c + c^2 + \ldots + c^n$ alors*

1. *$f(n) = \Theta(1)$ si $c < 1$*
2. *$f(n) = \Theta(n)$ si $c = 1$*

3. $f(n) = \Theta(c^n)$ *si* $c > 1$

1: Il suffit de montrer qu'il existe une constante $b > 0$ telle que $f(n) \leq b$; ce qui découle de $f(n) = \frac{1-c^{n+1}}{1-c} < \frac{1}{1-c}$. 2: Il suffit de montrer qu'il existe deux constantes $a, b > 0$ telle que $a\,n \leq f(n) \leq b\,n$; ce qui découle de $f(n) = n + 1$ (prendre par exemple $a = 1$ et $b = 2$). 3: Il suffit de montrer qu'il existe deux constantes $a, b > 0$ telle que $a\,c^n \leq f(n) \leq b\,c^n$; ce qui découle de $f(n) = \frac{c^{n+1}-1}{c-1}$ puisque $\frac{c^{n+1}-1}{c-1} \geq \frac{c^{n+1}}{c} = c_n$ et $\frac{c^{n+1}-1}{c-1} = c^n \times \frac{c-\frac{1}{c^n}}{c-1} \leq c^n \times \frac{c}{c-1}$ (prendre par exemple $a = 1$ et $b = \frac{c}{c-1}$).

Exercice 8 *Montrez que* $\log(n!) = \Theta(n \log n)$.

Puisque $\frac{n}{2}^{\frac{n}{2}} \leq n! \leq n^n$ et $\log(n^n) = n \log n$, il suffit de montrer qu'il existe une constante $a > 0$ telle que $a\,n \log n \leq \log(\frac{n}{2}^{\frac{n}{2}}) = \frac{n}{2}(\log n - \log 2)$. Pour $n \geq 2^2$, on a $\log n \geq 2 \log 2$, d'où $\frac{n}{2}(\log n - \log 2) \geq \frac{n}{2}\,\frac{1}{2} \log n$ donc on peut prendre $a = \frac{1}{4}$.

Exercice 9 *Montrez que* $\sum_{i=1}^{i=n} \frac{1}{i} = \Theta(\log n)$.

Pour tout entier i, il existe un unique entier $p(i) = \lfloor \log_2 i \rfloor$ tel que $2^{p(i)} \leq i < 2^{p(i)+1}$. On obtient une borne inférieure (respectivement, supérieure) de la somme $\sum_{i=1}^{i=n} \frac{1}{i}$ en remplacant chaque terme $\frac{1}{i}$ par $\frac{1}{2^{p(i)+1}}$ (respectivement, $\frac{1}{2^{p(i)}}$); ce qui implique $1 + \frac{p}{2} \leq \frac{1}{2^{p+1}} + \ldots + \frac{1}{8} + \frac{1}{4} + \frac{1}{4} + \frac{1}{2} + 1 \leq \sum_{i=1}^{i=n} \frac{1}{i} \leq 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^p} \leq p + 1$, en notant $p = p(n) = \Theta(\log n)$.

Exercice 10 *Montrez que* `E(n,p)` *est exponentiel.*

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

Le temps $T(n,p)$ d'exécution de l'algorithme satisfait $T(n,0), T(n,n) \geq 1$ et $T(n,p) \geq T(n-1,p-1) + T(n-1,p)$, donc $T(n,p) \geq \frac{n!}{(n-p)!p!} = \frac{n}{p}\,\frac{n-1}{p-1} \ldots \frac{n-p+1}{1} \geq \left(\frac{n}{p}\right)^p$.

Exercice 11 *Donnez les 5 lignes de code python constituant un algorithme récursif* `Euclide(a,b)` *renvoyant le pgcd de deux entiers* $a \geq b$*, et montrer qu'il est* $O(\log b)$.

```
def Euclide(a,b):
    if b==0:
        return a
    else:
        return Euclide(b,a%b)
```

On peut supposer $b \geq 1$, car sinon l'algorithme est $\Theta(1)$. Soit (b_i) la suite telle que b_0 est la valeur retournée par l'algorithme, b_{n+1}, b_n sont les valeurs initiales `a,b`, et les valeurs intermédiaires b_{i-1} sont les restes de la division entière de b_{i+1} par b_i. Ainsi $b_{i+2} = q_i b_{i+1} + b_i$ avec $1 \leq b_i < b_{i+1}$ (pour $i = 0 \ldots n-1$). Puisque $q_i \geq 1$, on a $b_{i+2} \geq b_{i+1} + b_i$, et de plus, $b_0, b_1 \geq 1$. Donc b_i est supérieur au $i$ème terme de la suite de Fibonnacci, et donc $b_n \geq \Theta(\phi^n)$ où ϕ est le nombre d'or (la solution positive de $x^2 = x+1$). Il existe donc une constante $c > 0$ telle que $b \geq c\,\phi^n$ d'où $n \leq \log_\phi b - \log_\phi c = O(\log b)$.

Exercice 12 *Remplissez les lignes manquantes:*

```
def tri_den(A):
    n=len(A)
    m=max(A)
    B=[]
    for j in range(n):
        B.append(0)
    C=[]
    for i in range(m+1):
        C.append(0)
    for j in range(n):
```

```
        #  solution => C[A[j]]=C[A[j]]+1
    for i in range(1,m+1):
        #  solution => C[i]=C[i-1]+C[i]
    for j in range(n):
        B[C[A[j]]-1]=A[j]
    A=B
```

Exercice 13 *Remplissez les lignes manquantes:*

```
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:
           # solution => tab[i+1]=tab[i]
            i=i-1
        tab[i+1]=x
```

Exercice 14 *Donnez la sortie écran à l'appel de la fonction* `tri(tab)` *avec l'entrée* `tab=[1,8,7,54,78,53]`.

```
def tri(tab):
    print(*tab)
    if len(tab) >1:
        mi = len(tab)//2
        L = tab[:mi]  # le sous-tableau tab[0]..tab[mi-1]
        R = tab[mi:]  # le sous-tableau tab[mi]..tab[len-1]
        tri(L)
        tri(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]
            j+= 1
            k+= 1
```

sortie:

1 8 7 54 78 53
1 8 7
1
8 7
8
7
54 78 53
54
78 53

78
53