UNIVERSITÉ DE PARIS-DAUPHINE

Le candidat inscrit ici très lisiblement ses

Nom : Hernoune

Prénoms : Sami

N° GROUPE : 4

Numéro de convocation : 22103785

CYCLE

ANNÉE : 2022-2023 SESSION : ~~2022-2023~~ Partiel

MATIÈRE : Algo-prog. 3

UV = ----------------

Il est interdit aussi bien de signer à la fin de la composition que d'indiquer son nom ou son numéro sur les feuilles intercalaires.

N° de groupe :

Nombre d'intercalaires :

	Note	Signature	Note finale
1er correcteur			
2e correcteur			

APPRÉCIATIONS EXPLIQUANT LA NOTE

20/20

Ne pas écrire dans cette marge

Sujet : Exercice 1

0,5

1) ~~$\log_b(x)$ est le réel~~

Soit $x>0$, $\log_b(x)$ est le réel tel que $b^{\log_b(x)} = x$.

Il existe et il est unique. C'est aussi ~~le réel~~ la fonction tel que $\log_b(b^x) = x$

0,5

2) On a $1024 \leq 1025 < 2048$

$\Leftrightarrow 2^{10} \leq 1025 < 2^{11}$

$\Leftrightarrow 10 \leq \log_2(1025) < 11$ par stricte croissance du log

$\Leftrightarrow \lfloor \log_2(1025) \rfloor = 10$ par définition de la partie entière

Avis important : Tout candidat convaincu de fraude sera immédiatement expulsé et perdra les droits versés sans préjudice des peines prévues par le régime disciplinaire des Facultés, décret du 21 juillet 1897 (art. 33, 34 et 41) et par la loi du 23 décembre 1961.

PARTIE A RABATTRE

On a $27 \leq 72 < 81$

$\Leftrightarrow 3^3 \leq 72 < 3^4$

$\Leftrightarrow 3 \leq \log_3(72) < 4$ par stricte croissance du log.

$\Leftrightarrow \lfloor \log_3(72) \rfloor = 3$ par définition de la partie entière.

0,5

3) a) $\log_2(8) = \log_2(2^3) = 3 \neq 4 \longrightarrow$ FAUX

0,5

b) $\log_2(8) = \log_2(2^3) = 3 \longrightarrow$ VRAI

0,5

c) Posons $b=2$, $a=16$ et $n=4$.

On a $\log_b(n) = \log_2(2^2) = 2$ et

$\log_b(a) + \log_a(n) = \log_2(2^4) + \log_{16}(16^{1/2})$ 0,5

$= 4 + \frac{1}{2} = \frac{9}{2} \neq \log_b(n) \Longrightarrow$ FAUX

d) en posant $a = b^u$, $n = a^v$, on a :

$\log_b(a) \times \log_a(n) = uv$ et aussi : 0,5

Vrai

$\log_b(n) = \log_b(a^v) = \log_b((b^u)^v) = \log_b(b^{uv}) = uv$

e) en posant $a = b^u$ (si $u=0$, $a=1$ et $\log_b(a) = 0 \Rightarrow \frac{1}{\log_b(a)}$ indéterminé)

$\Rightarrow a^{1/u} = (b^u)^{1/u} = b$, on a :

0,5

$\log_a(b) = \frac{1}{u} = \frac{1}{\log_b(b^u)} = \frac{1}{\log_b(a)} \longrightarrow$ VRAI

f) en posant $a = b^u$, $n = b^v$, on a :

$a^{\log_b(n)} = a^v = (b^u)^v = b^{uv}$ et aussi

0,5

$n^{\log_b(a)} = n^u = (b^v)^u = b^{uv} \longrightarrow$ VRAI.

g) En posant $a = 16$, $b = 2$ et $n = 4$, on a

0,5

$a^{\log_n(b)} = 16^{\log_4(2)} = 16^{\log_4(4^{1/2})} = 16^{1/2} = \sqrt{16} = 4$

et $n^{\log_b(a)} = 4^{\log_2(16)} = 4^{\log_2(2^4)} = 4^4 = 256 \neq 4 \longrightarrow$ FAUX

h) $(b^q)^{\log_b(a)} = b^{q\log_b(a)}$

0,5

$= (b^{\log_b(a)})^q$

$= a^q$

Exercice 2

1

```
def f(n):
    y=0
    i=0
    while i<n:
        if oracle(n)==1:
            y=y+1
        i+=1
    return y.
```

Exercice 3

1/

```
def f(x,y):
    i=0
    while y*i <= x:
        i+=1
    return i-1
```

Exercice 4

1/

```
def f(x,y):
    return x-(x//y)*y
```

Exercice 5

1/

y=0

f(28) ⟹ 3×f(9) car ~~28~~ > 0 ⟹ on appelle 3 fois f(28//3) = f(9)

⟹ 3×(3f(3)) car 9>0 ⟹ 3f(9//3) = 3f(3)

⟹ 9f(3)

⟹ 9(3f(1)) car 3>0 et 3//3=1

⟹ 27f(1)

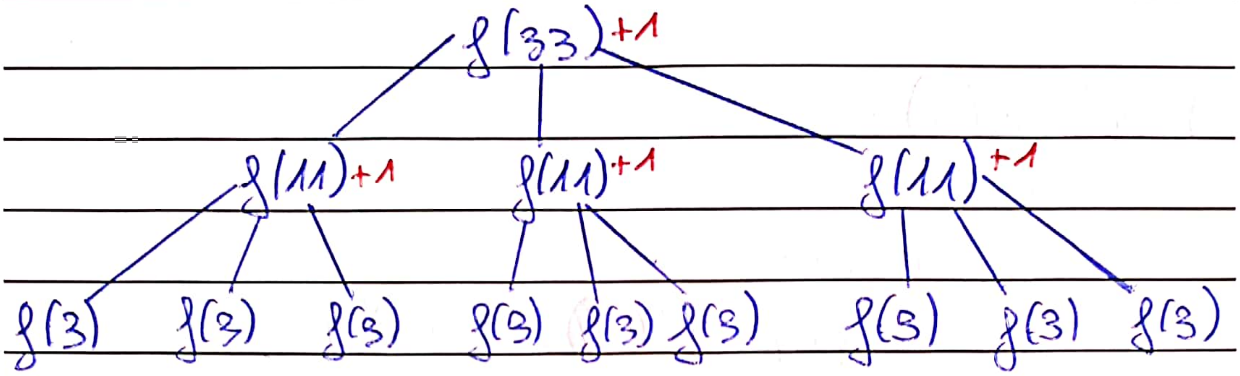
⟹ 27(3f(0)) car 1>0 et 1//3=0

⟹ 81f(0)

Or, f(0) incrémente y de 1. En appelant f(28), on incrémente ainsi 81 fois y, soit y=0+81 $\boxed{=81}$

Exercice 6 1/

On peut représenter cet appel comme dans l'exercice 5, mais c'est plus simple de faire un arbre ici :



On appelle ici f(3) 9 fois.
Chaque appel de f(3) appelle 3 fois f(1) et incrémente y :

$f(3) \Rightarrow 3f(1)$ $+1$ → on incrémente y

$\Rightarrow 3(\underbrace{3f(0)}_{\text{rien}} +1)$ $+1$ → chaque appel de f(1) incrémente y de 1

$\Rightarrow 3+1=4$

On a alors $f(3) \Rightarrow y = y+4$.

On incrémente y de 4 : 9 fois, donc on incrémente de 36. En rajoutant les incrémentations des f(11) et f(33), on a alors $y = y+40 \Rightarrow y = 40$

Exercice 7

1) def f(x): 1

```
def f(x):
    i=0
    while b**i <= x:
        i+=1
    return i-1
```

2) def g(x): 1

```
def g(x):
    res = []
    while x>0:
        res.append(x//(b**f(x)))
        y = x//(b**f(x))        # y=x_k
        res.append(y)
        x -= y(b**f(x))
    return res
```

3) 2

Par récurrence forte

<u>Initialisation</u>: Pour $x=1$, $x=b^0$ et il est unique.
La propriété est vraie pour $x=1$

<u>Hérédité</u>: Supposons que $x = \sum_{i=0}^{k} x_i b_i$ sa décomposition unique dans la base b. Montrons donc que $x+1$ est aussi décomposable comme x.

$$x+1 = \sum_{i=0}^{k} x_i b^i + 1 \quad \text{par hypothèse}$$

$$= x_k b^k + \dots + x_1 b + (x_0+1).$$

Si $x_0+1 < b$, on a fini.

Sinon : $x+1 = x_k b^k + \dots + (x_1+1)b + 0$

Si $x_1+1 < b$, on a fini. Sinon, on réitère ce processus.

Scanné avec CamScanner

On arrive alors à la fin si $\forall i \in [\![0;k-1]\!], x_i+1 = b$; ②

$x+1 = (x_k+1)b^k + 0b^{k-1} + \dots + 0$

Si $x_k+1 < b$ on a fini, sinon : $x_k+1 = b$ et

$x+1 = b^{k+1} = 1 \times b^{k+1} + 0b^k + 0b^{k-1} + \dots + 0b + 0$

Dans tt les cas, $x+1$ admet une décomposition unique.

D'où x_i défini de manière unique $\forall x \in \mathbb{N}$.

Exercice 8 2

On a : $\frac{1}{2^{p+1}} + \dots + \frac{1}{8} + \underbrace{\frac{1}{4} + \frac{1}{4}}_{2^i \frac{1}{2^{i+1}}} + \frac{1}{2} \leqslant \sum_{i=1}^{n} \frac{1}{i} \leqslant 1 + \underbrace{\frac{1}{2} + \frac{1}{2}}_{2^i \frac{1}{2^i}} + \frac{1}{4} + \dots + \frac{1}{2^p}$

où $p = \lfloor \log_2(n) \rfloor$ tel que $2^p \leqslant n < 2^{p+1}$

On obtient $\sum_{i=0}^{p} 2^i \frac{1}{2^{i+1}} \leqslant \sum_{i=1}^{n} \frac{1}{i} \leqslant \sum_{i=0}^{p} 2^i \frac{1}{2^i}$

$\frac{1}{2} \sum_{i=0}^{p} 1 \leqslant \qquad \leqslant \sum_{i=0}^{p} 1$

$\frac{1}{2}(p+1) \leqslant \qquad \leqslant p+1$

$\frac{1}{2} p \leqslant \qquad \leqslant 2p$ pour $n \geqslant 2$

$\frac{1}{2} \lfloor \log_2(n) \rfloor \leqslant \qquad \leqslant 2\log_2(n)$

$\frac{1}{2}(\log_2(n) - 1) \leqslant \qquad \leqslant 2\log_2(n)$

$\frac{1}{4} \log_2(n) \leqslant \qquad \leqslant 2\log_2(n)$ pour $n \geqslant 2$.

D'où $\exists c_1, c_2 \in \mathbb{R}^+, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0 \quad \frac{1}{4}\log_2(n) \leqslant \sum_{i=1}^{n} \frac{1}{i} \leqslant c_2 \log_2(n)$, avec $c_1 = \frac{1}{4}$, $c_2 = 2$, $n_0 = 2$

D'où $\sum_{i=1}^{n} = \Theta(\log n)$

Exercice 9 2/

On a $n! \leq n^n$

$\Rightarrow \log(n!) \leq \log(n^n) = n\log n$ par croissance du log

D'où $\log(n!) = O(n\log n)$.

De plus, $n! = n(n-1)\times(n-2)\times\dots\times 2\times 1$

$\geq n(n-1)\times\dots\times\frac{n}{2}!$ (ou $\frac{n+1}{2}$ si n impair)

$\geq n(n-1)\times\dots\times\frac{n}{2}$

$\geq (n/2)^{n/2}$

$\Rightarrow \log(n!) \geq \log((n/2)^{n/2})$

$\geq \frac{n}{2}\log(n/2)$

$\geq \frac{1}{2}n(\log n - \log\frac{1}{2})$

$\geq \frac{1}{2}n(\log n - \underbrace{\log 1}_{=0})$

$\geq \frac{1}{2}(n\log n)$

D'où $\log(n!) = \Omega(n\log n)$

On a donc $\log(n!) = O(n\log n)$ et $\Omega(n\log n) \Rightarrow \log(n!) = \Theta(n\log n)$

Exercice 10 2/

Supposons que l'on veuille additionner x et y avec

$\begin{cases} x = x_0x_1\dots x_k \\ y = y_0\,y_1\dots y_k \end{cases}$ $(k = \max(\lfloor\log_{10}(x)\rfloor, \lfloor\log_{10}(y)\rfloor))$

Scanné avec CamScanner

③

Pour additionner des chiffres, ~~on~~ 0, ..., 9, on prend 1 seule opération élémentaire avec les tables d'addition. Quand on a plusieurs chiffres comme x et y, on doit poser :

~~$x_0 x_1 x_2 x_3 \ldots x_{k-3} x_{k-2} x_{k-1} x_k$~~

$$\begin{array}{r} x_k\, x_{k-1} \ldots x_2\, x_1 x_0 \\ +\; y_k\, y_{k-1} \ldots y_2\, y_1\, y_0 \\ \hline \end{array}$$

$$= \sum_{i=0}^{k} x_i 10^i + \sum_{i=0}^{k} y_i 10^i$$

$$= \sum_{i=0}^{k} \underbrace{(x_i+y_i)}_{1\,\text{opéra}^{\circ}} 10^i$$

On fait ici k opérations élémentaires avec potentiellement des retenues qui viennent ajouter 1 opérations.

On a $\underbrace{k+1}_{\text{le cas où on a aucune retenue}} \leq T(n) \leq \underbrace{2(k+1)}_{\text{le cas où on a des retenues tout le temps}}$

D'où. $T(n) = \Theta(\log_{10}(x)) = \Theta(\log x) = \mathcal{O}(k)$

(ou y si $y > x$).

* Lorsque l'on fait une multiplica° xy :

$$\left(\sum_{i=0}^{k} x_i 10^i\right)\left(\sum_{i=0}^{k} y_i 10^i\right) = \sum_{i=0}^{k} \sum_{j=0}^{k} \underbrace{x_i y_j}_{1\,\text{op}} 10^{i+j} \rightarrow 1\,\text{op}$$

On a ici $\Theta(k^2)$ additions et multiplications élémentaires,

D'où $T(n) = \Theta(\log^2 x)$

Avec un algorithme récursif naïf, on a

$$\left(\sum_{i=0}^{k} x_i 10^i\right)\left(\sum_{i=0}^{k} y_i 10^i\right)$$

$$= \left(\underbrace{\sum_{i=0}^{k/2} x_i 10^i}_{S_1} + \underbrace{\sum_{i=k/2+1}^{k} x_i 10^i}_{S_2}\right)\left(\underbrace{\sum_{i=0}^{k/2} y_i 10^i}_{S_3} + \underbrace{\sum_{i=k/2+1}^{k} y_i 10^i}_{S_4}\right)$$

$$\cancel{\sum_{j=0}^{k/2}\sum_{i=0}^{k/2} y_i x_j 10^{i+j} + \sum}$$

$$= \overbrace{S_1S_3}^{T(n/2)} + \overbrace{S_1S_4}^{T(n/2)} + \overbrace{S_2S_3}^{T(n/2)} + \overbrace{S_2S_4}^{T(n/2)}$$

~~On a alors~~ $\cancel{T(n) = 4T(n/2)}$

$= \cancel{4}$

$\cancel{= 4 \times 4\, T(n/4)}$

$= \dots$

$\cancel{= 4^k\, T(1)}$

$\cancel{= 4^{\log_2(x)}\, \Theta(1)}$

On a alors $T(n) = 4\frac{k}{2}\, T(n/2)$ car pour additionner $S_1S_3\dots$ il faut $\frac{k}{2}$ temps $\times 4$.

$= 2k\; k\; T(n/4)\dots$

$\cancel{k\, \frac{k}{2}\, \frac{k}{4} \cdots \frac{k}{\dots} T(1)}$

$\cancel{\sum_{i=0}}$

$$= 2k \times k \times \frac{k}{2} \times \dots \times \frac{k}{2^k}\, T(1)$$ pas améliorée.

$$= \boxed{k\,k}\left(\sum_{i=0}^{k}\frac{1}{2^i}\right)\underbrace{T(1)}_{\Theta(1)} = \Theta(k^2)$$

Scanné avec CamScanner