UNIVERSITÉ DE PARIS-DAUPHINE

CYCLE

ANNÉE : ..... SESSION : .....

MATIÈRE : Algo 3

UV = 22000242

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 : 5

Nombre d'intercalaires : 1

	Note	Signature	Note finale	APPRÉCIATIONS EXPLIQUANT LA NOTE
1er correcteur			12	
2e correcteur				

Ne pas écrire dans cette marge

Sujet : Ex 1) 1) ① $Ht_1 = tf(t_1)$

② $Ht_2 = tf(t_2)$

2

Ex 2 : ① $x = A[j]$

② $A[i+1] = A[i]$

2

2 Ex 3) ① $C[A[i]] - 1$

Ex 5 : ① ① 5

② 5

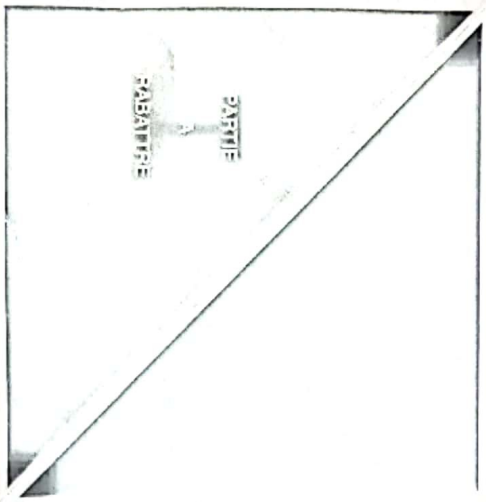
③ $A[i] = j$

2 ④ $(i == (m-1))$

⑤ count (0)

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.

Scanné avec CamScanner



2 Exs 1) ① $[y, x-(a//b)*y, d]$

2) Ex 6 :

1) Ce code effectue le

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

1

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

Il décompose les matrice en 4 sous matrice.

2) ~~Ce programme décrit le programme~~

Ce programme recursif appel 8 fois se rappelle 8 fois en problème de taille $\frac{n}{2}$, Il satisfait l'equation

$$T(n) = \begin{cases} 8\,T\left(\frac{n}{2}\right) + \Theta(n^2) \\ \Theta(1) \text{ si } n=0 \end{cases}$$

on $\Theta(n^2)$ et le temps pour ~~résoudre~~ recombiner les sous problème, il faut n^2 multiplication

d'après le théorème

Scanné avec CamScanner

D'après le cours on sait que $T(b^p) = \Theta\left(\sum_{i=0}^{p} a^i b^{(p-i)c}\right)$

Vérifions le P.M.I pour $n = 2^p$ par notre equation

$$T(n) = T(2^p) = 8\,T\left(\frac{2^p}{2}\right) + \Theta(2^{p^2})$$

~~C'est vrai pour n=0~~

C'est vrai pour $p = 0$ $\quad (T(2^0) = 1)$

~~Vérifions~~ Supposons que $T(2^p) = \Theta\left(\sum_{i=0}^{p} 8^i\, 2^{(p-i)2}\right)$:

on a $T(2^{p+1}) = 8\,T(2^p) + \Theta(2^{(p+1)^2})$

$$= \Theta\left(8 \sum_{i=0}^{p} 8^i\, 2^{(p-i)2} + 2^{(p+1)2}\right)$$

$$= \Theta\left(\sum_{i=0}^{p} 8^{i+1}\, 2^{(p-i)2} + 2^{(p+1)2}\right)$$

$$= \Theta\left(\sum_{i=0}^{p+1} 8^i\, 2^{(p-i)2}\right)$$

on a donc ~~Reprenons~~ $T(2^p) = \Theta\left(\sum_{i=0}^{p} 8^i\, 2^{(p-i)2}\right)$

Vérifions maintenant pour tout $n \in \mathbb{N}$.

Soit $2^p \leq n \leq 2^{p+1}$

On a d'ailleurs $T(2^p) = \Theta\left(\sum_{i=0}^{p} 8^i\, 2^{(p-i)2}\right)$

$$= \Theta\left(\sum_{i=0}^{p} 2^{2p} \sum_{i=0}^{p} \left(8\left(\frac{1}{2}\right)^i\right)\right)$$

($\left(\frac{1}{2}\right)^i \leq 1$)

$$= \Theta\left(2^{2p}\right) = \Theta\left((2^p)^2\right) = \Theta(n^2)$$

$$\Theta\left(2^{2p} \sum_{i=0}^{p} \left(\frac{1}{2}\right)^p\right) = \Theta\left(2^{2p}\, 2\left(1 - \left(\frac{1}{2}\right)^{p+1}\right)\right)$$

$$= \Theta(2^{2p+1} \cdot 2^p)$$

$$= \Theta(2^{2p})$$

~~$p = \log_2 b$~~

Scanné avec CamScanner

Vérifier pour tout n

$$2^p < n < 2^{p+1}$$

$$2^{2p} < n^2 < 2^{2(p+1)}$$

on a donc

$$c\,2^{2p} < T(2^p) < T(n) < T(2^{p+1}) < d\,2^{2(p+1)}$$

$$\leq 4d\,2^{2p} < 4d\,n^2$$

$$\frac{e}{4} n^2 \leq \frac{e}{4} e^{2p+2} \leq$$

on a donc si on prend les constante $\alpha = \frac{e}{4}$ et $\beta = 4d$

On a bien $\text{mult} = O(n^2)$ ✗

3) mult (A)

P11 = ~~mult(A11, A12) + mul~~

P11 =

4) On peut implémenter l'algo qui donne

$Re(u \times v) = (x+y)(b+a) - xa - by$ → pour la partie réel

et la $Im(u \times v) = xa - by$ pour la partie imaginaire (déjà calculé)

où $u = x + iy$ et $v = a + bi$

```
def mul([x,y], [a,b]):
    c = x * a
    d = b * y
    return [
    e = (x+y)(b+a)
    return [e - c - d, c - d]
```

$uv = (ax - by) + i(xb + ay)$

$= xb + xa + yb + ya$

Re(1) Im(1)

✗

Scanné avec CamScanner

EX6) 3) ~~On peut~~ Pour cette Algorithme on peut avoir les 7 produits

$P_1 = (A_{11} + A_{22})(B_{11} + B_{22})$

$P_2 = (A_{21} + A_{22}) B_{11}$

$P_3 = (A_{11})(B_{12} - B_{22})$

$P_4 = (A_{22})(B_{21} - B_{11})$

$P_5 = (A_{11} + A_{12})(B_{22})$

$P_6 = (A_{21} - A_{11})(B_{11} + B_{12})$

$P_7 = (A_{12} - A_{22})(B_{21} + B_{22})$

On peut ainsi avoir C avec

$C_{11} = P_1 + P_4 - P_5 + P_7$, $C_{12} = P_3 + P_4$, $C_{21} = P_2 + P_4$ ~~et C_{22}~~ =

et $C_{22} = P_1 - P_2 + P_3 + P_6$

On peut ensuite l'algorithme :

```
def ~~mult_Aux~~ mult1(A, B):
    if n<=1 :
        return C = A*B
    else :
        P1 = mult1((A11 + A22), (B11 + B22))
        P2 = mult1((A21 + A22), B11)
        P3 = mult1(A11, (B21 - B11))
        P4 = mult1(A22, (B21 - B11))
        P5 = mult1((A11 + A12), B22)
        P6 = ~~(A21 - A11)~~ mult((A21 - A11), (B11 + B12))
        P7 = mult((A12 - A22), (B21 + B22))
        C11 = P1 + P4 - P5 + P7
```

On ~~va~~ utilise mult

Scanné avec CamScanner

$C12 = P3 + P4$

$C21 = P2 + P4$

$C22 = P1 - P2 + P3 + P6$

, return C

On voit ici il y a 7 multiplication sous problème de taille $\frac{n}{2}$ qui mettent le même temps de mult pour se refaire : on a bien

$$T(n) = 7\,T\left(\frac{n}{2}\right) + \Theta(n^2)$$

Scanné avec CamScanner