0.1

(1pt) Appliquer le Master Theorem sur les cas suivants :

— **(0.5pt)** $T(n) = 3T(n/4) + nlogn$

— **(0.5pt)** $T(n) = T(2n/3) + 1$

(1pt) Montrez que :

— **(0.5pt)** $nlogn = O(n^2)$

— **(0.5pt)** $25n^4 - 10n^3 + 22n^2 = O(n^4)$

0.2

Quelle est la complexité de ce programme.

```
for (int i=0 ;i<n ;i++) do
    for (j=0 ;j<i ;j++) do
        printf("hello") ;
    end
end
```

Algorithme 1 : *DFor*

0.3

Quelle est la sortie de l'algorithme *Algo(T,x)* et quelle est sa complexité (utiliser la notation $\mathcal{O}$) en fonction de la taille n de tableau T.

```
a ← 1
b ← n
while b ≥ a do
    j ← (a+b)/2
    if x = T[j] then
        Retourner oui
    else
        if T[j] < x then
            a ← j + 1
        else
            b ← j − 1
        end
    end
    Retourner non
end
```

Algorithme 2 : Algo(T,x)

0.4

Soit la fonction suivante, *Atrouver(A,s,f,e)*.

```
if s = f then
    if A[s] = e then
        retourner 1
    else
        retourner 0
    end
end
m=(s+f)/2
gauche=Atrouver(A,s,m,e)
droite=Atrouver(A,m+1,f,e)
retourner gauche + droite
```

Algorithme 3 : entier : Atrouver(A,s,f,e)

Quelle est la valeur retournée par cette fonction.

0.5

L'algorithme suivant ayant comme entrée un tableau T de nombres entiers, de taille n, et comme sortie une variable binaire r (oui/non).

```
for i allant de 2 à n do
    if T[i − 1] > T[i] then
        r ← non
    end
end
Retourner r
```

Algorithme 4 : *Afind(T,n)*

Que fait cet algorithme.