SOLUTIONS SÉRIE 3 (Chapitre 2b)

Tous les numéros dans cette série sont pertinents. Il est recommandé de tous les faire.

***Question # 1**

Soit l'algorithme suivant.

Algorithm *Secret*($A[0...n-1]$) :
  //Entré : Un tableau $A[0...n-1]$ de n nombres réels
  $minval \leftarrow A[0]$
  $maxval \leftarrow A[0]$
  FOR $i \leftarrow 1$ TO $n-1$
    IF $A[i] < minval$
      $minval \leftarrow A[i]$
    IF $A[i] > maxval$
      $maxval \leftarrow A[i]$
  RETURN $maxval - minval$

A) *Que fait cet algorithme ?*
   Solution :
   Il calcul l'écart maximal entre deux éléments du tableau. C'est donc l'écart entre le plus grand élément et le plus petit.

B) *Quelle est son opération de base ?*
   Solution :
   La comparaison d'un élément.

C) *Combien de fois l'opération de base est-elle exécutée ?*
   Solution :
   $C(n) = \sum_{i=1}^{n-1} 2 = 2(n-1)$

D) *À quelle classe d'efficacité appartient cet algorithme ?*
   Solution :
   Linéaire : $\Theta(n)$

***Question # 2**

Soit l'algorithme suivant.

Algorithm *inefficace*($A[0...n-1]$) :
//Entrée : Un tableau $A[0...n-1]$ de n nombres entiers
FOR $i \leftarrow 0$ TO $n-1$
Effacer($A, 0$)
RETURN A

*Supposons que la fonction **Effacer** est l'équivalent de la méthode **erase** de la classe **vector** en C++. Pour répondre à cette question, il est conseillé de consulter la documentation de cette méthode.*

A) *Que fait cet algorithme ?*
Solution :
Il efface tous les éléments du vecteur A, à partir du début. Chaque fois qu'un élément est effacé, tous les éléments qui le suivent sont recopiés.

B) *Quel est le temps d'exécution d'un appel à **Effacer** ?*
Solution :
*Soit $C_{\text{effacer}}(m)$, le temps d'exécution de la fonction **Effacer**, où m est le nombre d'éléments dans le vecteur lors de l'appel à la fonction. Selon la documentation de la STL, cette fonction est linéaire en fonction du nombre d'éléments effacés plus le nombre d'éléments déplacés. L'algorithme supprime un seul élément à la fois et c'est toujours le premier élément du vecteur. Il y a donc une supression et $m-1$ éléments déplacés. On a donc $C_{\text{effcer}}(m) = 1 + m - 1 = m$.*

C) *Quel est le temps d'exécution de l'algorithme inefficace ?*
Solution :
Soit $C_{\text{inefficace}}(n)$, le temps d'exécution de l'algorithme inefficace. On a

$C_{\text{inefficace}}(n)$

= $\langle$ *À chaque itération, le vecteur contient un élément de moins* $\rangle$

$\sum_{i=0}^{n-1} C_{\text{effacer}}(n-i)$

= $\langle\ C_{\text{effacer}}(m) = m\ \rangle$

$\sum_{i=0}^{n-1} (n-i)$

= $\langle \sum_{i=0}^{n-1}(n-i) = (n-0) + (n-1) + ... + (n-n+1) = 1 + 2 + ... + n = \sum_{i=1}^{n} i \rangle$

$\sum_{i=1}^{n} i$

= $\langle$ *Aide-mémoire* $\rangle$

$\frac{n(n+1)}{2} = \frac{n^2+n}{2}$

D) À quelle classe d'efficacité appartient cet algorithme ?

Solution :

$C_{\text{inefficace}}(n) = \frac{n^2+n}{2} \in \Theta(n^2)$ *par la règle du maximum. En effet, puisque* $\frac{n^2}{2} \in \Theta(n^2)$ *et* $\frac{n}{2} \in \Theta(n)$, *nous avons* $\frac{n^2+n}{2} \in \Theta(\max(n^2, n)) = \Theta(n^2)$. *L'algorithme appartient donc à la classe d'efficacité quadratique.*

***Question # 3**

Soit l'algorithme suivant.

Algorithm *Enigma*($A[0...n-1, 0...n-1]$) :
    //Entré : Une matrice $A[0...n-1, 0..n-1]$ de nombres réels
    FOR $i \leftarrow 0$ TO $n-2$
        FOR $j \leftarrow i+1$ TO $n-1$
            IF $A[i,j] \neq A[j,i]$
                RETURN false
    RETURN true

A) *Que fait cet algorithme ?*
   Solution :
   Il retourne true si la matrice d'entrée est symmétrique, et false sinon.

B) *Quelle est son opération de base ?*
   Solution :
   La comparaison de deux éléments de la matrice.

C) *Combien de fois l'opération de base est-elle exécutée en pire et en meilleur cas ?*
   Solution :
   $C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}((n-1)-(i+1)+1) = \sum_{i=0}^{n-2}(n-i-1) = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$
   $C_{best}(n) = 1$

D) *À quelle classe d'efficacité appartient cet algorithme ?*
   Solution :
   Quadratique en pire cas : $C_{worst}(n) \in \Theta(n^2)$ *(ou* $C(n) \in O(n^2)$*)*
   Constant en meilleur cas : $C_{best}(n) \in \Theta(1)$

***Question # 4**

Analysez la complexité de l'algorithme suivant :

```
Algorithme Mystérieux(A[0...n − 1]) :
    /* Input : A[0...n − 1], un vecteur d'entiers positifs. */
    TriFusion(A)        /* S'exécute en Θ(n log n) dans tous les cas.*/
    val ← ∞
    i ← 0
    WHILE i ≤ n − 2
        IF A[i] = A[i + 1]
            RETURN 0
        IF |A[i] − A[i + 1]| < val
            val ← |A[i] − A[i + 1]|
        i ← i + 1
    RETURN val
```

Effectuez toutes les étapes pour l'analyse :

- *Choix d'une opération de base.*
- *Calcul du nombre de fois où l'opération de base est exécutée.*
- *Poser la classe de complexité.*

Utilisez la notation Θ. Si le temps d'exécution peut varier entre deux instances de même taille alors il faut procéder à l'analyse en meilleur cas et en pire cas. ***Solution :***

L'algorithme *Mystérieux* :

- La complexité l'algorithme *TriFusion* est de $\Theta(n \log n)$ dans tous les cas.
- L'opération de base est IF $|A[i] - A[i+1]| < val$.
- Pire cas :
  En pire cas, il n'y a aucuns doublons et on doit donc parcourir le tableau au complet. La sommation à poser pour calculer le nombre $C_{worst}(n)$ de fois où celle-ci est exécutée est

$$C_{worst}(n) = \sum_{i=0}^{n-2} 1 \tag{1}$$

  Après simplification de la sommation nous avons une complexité de $\Theta(n \log n + n)$ (où $n \log n$ provient de l'appel à *TriFusion*). Par l'utilisation de la règle du maximum, nous obtenons que *Mystérieux* est en $\Theta(n \log n)$ en pire cas.
- Meilleur cas :
  Dans le meilleur cas l'opération de base n'est exécutée qu'une seule fois lorsque les deux premiers nombres du tableau sont égaux. Nous avons donc une complexité de $\Theta(n \log n + 1)$ (où $n \log n$ provient de l'appel à *TriFusion*). Par la règle du maximum, *Mystérieux* est en $\Theta(n \log n)$ en meilleur cas.

***Question # 5**

En utilisant l'approximation par intégrale, déterminer l'ordre exacte de croissance pour les fonctions suivante :

A) $\sum_{i=0}^{n-1}(i^2+1)^2$

Solution :

$$S = \sum_{i=0}^{n-1}(i^2+1)^2 = \sum_{i=0}^{n-1}(i^4+2i^2+1)$$

Approximation par intégrale

$$\int_{-1}^{n-1}(x^4+2x^2+1)\,dx \leq S \leq \int_0^n (x^4+2x^2+1)\,dx$$
$$\left[\frac{x^5}{5}+\frac{2x^3}{3}+x\right|_{-1}^{n-1} \leq S \leq \left[\frac{x^5}{5}+\frac{2x^3}{3}+x\right|_0^n$$
$$\frac{(n-1)^5}{5}+\frac{2(n-1)^3}{3}+n-1-\left(-\frac{1}{5}-\frac{2}{3}-1\right) \leq S \leq \frac{n^5}{5}+\frac{2n^3}{3}+n$$

Lorsque $n \geq 2$, *nous avons* $\frac{2(n-1)^3}{3} \geq 0$, $(n-1) \geq 0$ *et* $-\left(-\frac{1}{5}-\frac{2}{3}-1\right) \geq 0$. *De plus, nous avons* $n-1 \geq n-\frac{n}{2}$.

$$\frac{(n-\frac{n}{2})^5}{5}+0+0+0 \leq S \leq \frac{n^5}{5}+\frac{2n^5}{3}+n^5$$
$$\frac{1}{2^5\cdot 5}n^5 \leq S \leq \frac{28}{15}n^5$$
$$S \in \Theta(n^5)$$

B) $\sum_{i=2}^{n-1}\log_2 i^2$

Solution :

$\sum_{i=2}^{n-1}\log_2 i^2 = 2\sum_{i=2}^{n-1}\log_2 i$

$\Rightarrow$ $\langle$ *Approximation par intégrale* $\rangle$

$2\int_1^{n-1}\log_2 x\,dx \leq 2\sum_{i=2}^{n-1}\log_2 i \leq 2\int_2^n \log_2 x\,dx$

$\Rightarrow$

$2(\frac{x\ln(x)}{\ln(2)}-\frac{x}{\ln(2)})|_1^{n-1} \leq 2\sum_{i=2}^{n-1}\log_2 i \leq 2(\frac{x\ln(x)}{\ln(2)}-\frac{x}{\ln(2)})|_2^n$

$\Rightarrow$

$2(\frac{\ln(n-1)n-\ln(n-1)-n+2}{\ln(2)}) \leq 2\sum_{i=2}^{n-1}\log_2 i \leq 2(\frac{n\ln(n)-n+2-2\ln(2)}{\ln(2)})$

$\Rightarrow$

$2(\frac{\ln(\frac{n}{2})n-\frac{1}{4}n\ln(n)-\frac{1}{4}n\ln(n)}{\ln(2)}) \leq 2\sum_{i=2}^{n-1}\log_2 i \leq \frac{2}{\ln(2)}n\ln(n)$ *pour* $n \geq 4$

$\Rightarrow$ $\langle$ *Puisque* $\ln(\frac{n}{2}) = \ln(n)-\ln(2)$ $\rangle$

$2(\frac{(\ln(n)-\ln(2))n-\frac{1}{4}n\ln(n)-\frac{1}{4}n\ln(n)}{\ln(2)}) \leq 2\sum_{i=2}^{n-1}\log_2 i \leq \frac{2}{\ln(2)}n\ln(n)$

$\Rightarrow$

$\frac{1}{\ln(2)}n\ln(n)-2n \leq 2\sum_{i=2}^{n-1}\log_2 i \leq \frac{2}{\ln(2)}n\ln(n)$

$\Rightarrow$ $\langle$ *En multipliant* $2n$ *par* $\frac{n}{4\ln(2)}$ *pour obtenir deux termes en* $n\ln n$ $\rangle$

$\frac{1}{\ln(2)} n \ln(n) - \frac{1}{2\ln(2)} n \ln(n) \leq 2 \sum_{i=2}^{n-1} \log_2 i \leq \frac{2}{\ln(2)} n \ln(n)$ *pour* $n \geq 4\ln(2)$

$\Rightarrow$

$\frac{1}{2\ln(2)} n \ln(n) \leq 2 \sum_{i=2}^{n-1} \log_2 i \leq \frac{2}{\ln(2)} n \ln(n)$

$\Rightarrow$

$\sum_{i=2}^{n-1} \log_2 i^2 \in \Theta(n \ln(n))$

C) $\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} (i+j)$

Solution :

Posons :

$$S = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} (i+j)$$

$$\int_{-1}^{n-1} \int_{-1}^{x-1} (x+y) dy\, dx \leq S \leq \int_{0}^{n} \int_{0}^{x} (x+y) dy\, dx$$

$$\int_{-1}^{n-1} \left[xy + \frac{1}{2} y^2 \right|_{-1}^{x-1} dx \leq S \leq \int_{0}^{n} \left[xy + \frac{1}{2} y^2 \right|_{0}^{x} dx$$

$$\int_{-1}^{n-1} \left(x(x-1) + \frac{1}{2}(x-1)^2 + x - \frac{1}{2} \right) dx \leq S \leq \int_{0}^{n} (x^2 + \frac{1}{2} x^2 - 0 - 0)\, dx$$

$$\int_{-1}^{n-1} \left(\frac{3}{2} x^2 - x \right) dx \leq S \leq \int_{0}^{n} \frac{3}{2} x^2\, dx$$

$$\left[\frac{1}{2} x^3 - \frac{1}{2} x^2 \right|_{-1}^{n-1} \leq S \leq \left[\frac{1}{2} x^3 \right|_{0}^{n}$$

$$\frac{1}{2} \left[(n-1)^3 - (n-1)^2 + 1 + 1 \right] \leq S \leq \frac{1}{2} n^3$$

$$\frac{1}{2} n^3 - 2n^2 + \frac{5}{2} n \leq S \leq \frac{1}{2} n^3$$

$$\frac{1}{2} n^3 - 2n^2 \cdot \frac{1}{8} n + 0 \leq S \leq \frac{1}{2} n^3 \qquad \textit{Pour } n \geq 8$$

$$\frac{1}{4} n^3 \leq S \leq \frac{1}{2} n^3$$

$$S \in \Theta(n^3)$$

***Question # 6**

Analysez la complexité de l'algorithme suivant en fonction de n *:*

Agorithme Complexe(n)
pour $i = 2..n$
$c = 0$
while $c < n$
$c = c + i$

Effectuez toutes les étapes pour l'analyse :

- *Choix d'une opération de base.*
- *Calcul du nombre de fois où l'opération de base est exécutée.*
- *Poser la classe de complexité.*

Utilisez la notation Θ. Si le temps d'exécution peut varier entre deux instances de même taille alors il faut procéder à l'analyse en meilleur cas et en pire cas. ***Solution :***
L'opération de base est $c = c + i$. Il faut incrémenter $\lceil \frac{n}{i} \rceil$ fois c de i avant de sortir du while. *Ainsi, dans tous les cas cette opération de base est exécutée*

$$C(n) = \sum_{i=2}^{n} \left\lceil \frac{n}{i} \right\rceil \tag{2}$$

fois. Pour résoudre la sommation (2), il faut utiliser l'approximation de la somme par des intégrales définies (voir l'aide-mémoire). Notons tout d'abord que $C(n)$ est non croissante. Ainsi, nous avons la borde inférieure suivante :

$$\begin{align}
\sum_{i=2}^{n} \left\lceil \frac{n}{i} \right\rceil &\geq \sum_{i=2}^{n} \frac{n}{i} \tag{3}\\
&\geq \int_{2}^{n+1} \frac{n}{x}\, dx \tag{4}\\
&= \Big[n \ln x \Big|_{2}^{n+1} \tag{5}\\
&= n \ln(n+1) - n \ln 2 \tag{6}\\
&\geq n \ln n - n \ln 2 \tag{7}\\
&\geq n \ln n - \frac{1}{2} n \ln n \qquad (\forall n \geq 4) \tag{8}\\
&= \frac{1}{2} n \ln n \tag{9}\\
&\in \Omega(n \ln n) \tag{10}
\end{align}$$

Posons maintenant la borne supérieure :

$$\begin{align}
\sum_{i=2}^{n} \left\lceil \frac{n}{i} \right\rceil &\leq \sum_{i=2}^{n} \left(\frac{n}{i} + 1 \right) \tag{11}\\
&\leq \int_{1}^{n} \left(\frac{n}{x} + 1 \right) dx \tag{12}\\
&= \Big[n \ln x + x \Big|_{1}^{n} \tag{13}\\
&= n \ln n + n - 1 \tag{14}\\
&\leq n \ln n + n \tag{15}\\
&\in O(n \ln n) \qquad \textit{// Loi du maximum} \tag{16}
\end{align}$$

D'où $C(n) \in \Theta(n \ln n)$.

***Question # 7**
Résolvez les relations de récurrences suivantes :

A) $x(n) = x(n-1) + 5, x(1) = 0$
Solution :

$$\begin{aligned}
x(n) &= x(n-1) + 5 \\
&= [x(n-2) + 5] + 5 \\
&= [x(n-3) + 5] + 5 \times 2 \\
&= \ldots \\
&= x(n-i) + 5 \times i \\
&= \ldots \\
&= x(1) + 5 \times (n-1) \\
&= 5(n-1)
\end{aligned}$$

B) $x(n) = 3x(n-1), x(1) = 4$
Solution :

$$\begin{aligned}
x(n) &= 3x(n-1) \\
&= 3[3x(n-2)] \\
&= 3^2[3x(n-3)] \\
&= \ldots \\
&= 3^i x(n-i) \\
&= \ldots \\
&= 3^{n-1}x(1) \\
&= 4 \times 3^{n-1}
\end{aligned}$$

C) $x(n) = x(n-1) + n, x(0) = 0$
Solution :

$$\begin{aligned}
x(n) &= x(n-1) + n \\
&= [x(n-2) + (n-1)] + n \\
&= [x(n-3) + (n-2)] + (n-1+n) \\
&= \ldots \\
&= x(n-i) + (n-i+1) + (n-i+2) + \ldots + n \\
&= \ldots \\
&= x(0) + 1 + 2 + \ldots + n \\
&= \sum_{i=1}^{n} i \\
&= \frac{n(n+1)}{2}
\end{aligned}$$

D) $x(n) = x(n/2) + n$, $x(1) = 1$ *(on suppose $n = 2^k$)*
Solution :

$$\begin{aligned}
x(2^k) &= x(2^{k-1}) + 2^k \\
&= [x(2^{k-2}) + 2^{k-1}] + 2^k \\
&= [x(2^{k-3}) + 2^{k-2}] + 2^{k-1} + 2^k \\
&= \ldots \\
&= x(2^{k-i}) + 2^{k-i+1} + 2^{k-i+2} + \ldots + 2^k \\
&= \ldots \\
&= x(2^{k-k}) + 2^1 + 2^2 + \ldots 2^k \\
&= 1 + 2^1 + 2^2 + \ldots 2^k \\
&= \sum_{i=0}^{k} 2^i \\
&= 2^{k+1} - 1 \\
&= 2 \cdot 2^k - 1 \\
&= 2n - 1
\end{aligned}$$

E) $x(n) = x(n/3) + 1$, $x(1) = 1$ *(on suppose $n = 3^k$)*
Solution :

$$\begin{aligned}
x(3^k) &= x(3^{k-1}) + 1 \\
&= [x(3^{k-2}) + 1] + 1 \\
&= [x(3^{k-3}) + 1] + 1 + 1 \\
&= \ldots \\
&= x(3^{k-i}) + i \\
&= \ldots \\
&= x(3^{k-k}) + k \\
&= x(1) + k \\
&= 1 + \log_3 n
\end{aligned}$$

***Question # 8**

Soit l'algorithme suivant.

```
Algorithm Min1(A[0...n − 1]) :
    //Entrée : Un tableau A[0...n − 1] de nombres réels
    IF n = 1
        RETURN A[0]
    ELSE
        temp ← Min1(A[0...n − 2])
        IF temp ≤ A[n − 1]
            RETURN temp
        ELSE
            RETURN A[n − 1]
```

A) *Que fait cet algorithme ?*
   Solution :
   Il trouve la valeur du plus petit élément du tableau.

B) *Écrivez la relation de récurrence qui exprime le nombre de fois où l'opération de base est exécutée et résolvez-la.*
   Solution :
   La récurrence pour le nombre de comparaison entre deux élément est $C(n) = C(n-1) + 1$, $C(1) = 0$. *En résolvant cette récurrence, on obtient* $C(n) = n - 1$.

***Question # 9**

Soit l'algorithme suivant qui résout le même problème que l'algorithme de la question 8.

```
Algorithm Min2(A[l...r]) :
    IF l = r
        RETURN A[l]
    ELSE
        temp1 ← Min2(A[l...⌊(l + r)/2⌋])
        temp2 ← Min2(A[⌊(l + r)/2⌋ + 1...r])
        IF temp1 ≤ temp2
            RETURN temp1
        ELSE
            RETURN temp2
```

A) *Écrivez la relation de récurrence qui exprime le nombre de fois où l'opération de base est exécutée et résolvez-la.*
Solution :
La récurrence pour le nombre de comparaison entre deux élément est $C(n) = C(\lfloor n/2 \rfloor) + C(\lceil n/2 \rceil) + 1$, $C(1) = 0$. *En résolvant cette récurrence avec* $n = 2^k$, *on obtient* $C(n) = n - 1$.

B) *Lequel des algorithmes Min1 ou Min2 est le plus rapide ? Pouvez-vous contruire un nouvel algorithme qui serait plus efficace que Min1 et Min2 tout en résolvant le même problème ?*
Solution :
Les deux algorithmes Min1 et Min2 ont la même efficacité. De plus, il est clair que tout algorithme voulant trouver l'élément minimal d'un tableau quelconque doit faire au moins n *comparaisons, donc* $\Omega(n)$ *comparaisons. Cependant, un algorithme séquentiel n'aurait pas le "overhead" des appels récursifs.*

***Question # 10**

Soit A la matrice $n \times n$ suivante.

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}$$

On dénote par $detA$ le déterminant de la matrice A. Pour $n = 1$, $detA = a_{11}$ et pour $n > 1$, $detA = \sum_{j=1}^{n} s_j a_{1j} detA_j$ où s_j est +1 lorsque j est impair et -1 lorsque j est pair, a_{1j} est l'élément de la matrice en ligne 1 et colonne j, et A_j est la matrice $(n-1) \times (n-1)$ obtenue de la matrice A en enlevant la ligne 1 et la colonne j de cette dernière.

A) *Écrivez la relation de récurrence décrivant le nombre de multiplications faite par l'algorithme implementant cette définition récursive pour le calcul du déterminant.*
   Solution :
   Soit $M(n)$, le nombre de multiplications effectuées par l'algorithme en se basant sur la formule $detA = \sum_{j=1}^{n} s_j a_{1j} detA_j$. Si on ne tient pas compte des multiplications par s_j (c'est juste un changement de signe), alors :

$$M(n) = \sum_{j=1}^{n} (M(n-1) + 1) = n(M(n-1) + 1)$$

B) *Sans résoudre la récurrence, que pouvez-vous dire au sujet de l'ordre de croissance de sa valeur par rapport à $n!$?*
   Solution :
   Puisque $M(n) = nM(n-1) + n$, la fonction $M(n)$ grossit au moins aussi rapidement que la fonction factorielle qui est définie par $n! = n \times (n-1)!$, $1! = 1$.