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 ?

B) Quelle est son opération de base ?

C) Combien de fois l'opération de base est-elle exécutée ?

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

***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 ?

B) Quel est le temps d'exécution d'un appel à ***Effacer*** *?*

C) Quel est le temps d'exécution de l'algorithme inefficace ?

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

***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 ← 0 TO n − 2
        FOR j ← i + 1 TO n − 1
            IF A[i, j] ≠ A[j, i]
                RETURN false
    RETURN true
```

A) Que fait cet algorithme ?

B) Quelle est son opération de base ?

C) Combien de fois l'opération de base est-elle exécutée en pire et en meilleur cas ?

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

***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* $\Theta(n \log n)$ *dans tous les cas.*/
  $val \leftarrow \infty$
  $i \leftarrow 0$
  WHILE $i \leq n-2$
    IF $A[i] = A[i+1]$
      RETURN 0
    IF $|A[i] - A[i+1]| < val$
      $val \leftarrow |A[i] - A[i+1]|$
    $i \leftarrow i+1$
  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.*

***Question # 5**

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

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

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

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

***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.

***Question # 7**

Résolvez les relations de récurrences suivantes :

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

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

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

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

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

***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 ?

B) Écrivez la relation de récurrence qui exprime le nombre de fois où l'opération de base est exécutée et résolvez-la.

***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
    $temp1 \leftarrow$ *Min2*$(A[l...\lfloor (l+r)/2 \rfloor])$
    $temp2 \leftarrow$ *Min2*$(A[\lfloor (l+r)/2 \rfloor + 1...r])$
    IF $temp1 \leq temp2$
      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.*

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 ?*

***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.

B) Sans résoudre la récurrence, que pouvez-vous dire au sujet de l'ordre de croissance de sa valeur par rapport à $n!$?