

Questions : Chapitre 2

*Question # 1

Utilisez la notation la plus appropriée parmi O , Ω et Θ pour indiquer la classe d'efficacité (par rapport au temps) de la recherche séquentielle ci-dessous dans les différents cas.

Algorithme 1 : RechercheSequentielle($A[0..n - 1], d$)

```
1 pour i = 0..n - 1 faire
2   |   si A[i] = d alors
3   |   |   retourner i
4 retourner -1
```

A) Pire cas

B) Meilleur cas

C) Cas moyen

*Question # 2

En vous appuyant uniquement sur notre analyse de l'algorithme d'Euclide effectuée au chap 1, exprimez l'efficacité de l'algorithme d'Euclide en pire cas à l'aide de la notation asymptotique.

Question # 3

Démontrez la règle du maximum pour Ω .

Question # 4

Démontrez la règle du maximum pour Θ .

Question # 5

Démontrez que $f(n) \in \Theta(g(n)) \Leftrightarrow \Theta(f(n)) = \Theta(g(n)) \Leftrightarrow O(f(n)) = O(g(n))$

Question # 6

Démontrez que :

$$\begin{aligned} f(n) \in O(g(n)) \wedge f(n) \notin \Theta(g(n)) &\Leftrightarrow f(n) \in O(g(n)) \wedge f(n) \notin \Omega(g(n)) \\ &\Leftrightarrow f(n) \in O(g(n)) \wedge g(n) \notin O(f(n)) \\ &\Leftrightarrow O(f(n)) \subset O(g(n)) \end{aligned}$$

Question # 7

Soit deux fonctions positives asymptotiquement $f(n)$ et $g(n)$. Les relations $=$, \neq , \subset et $\not\subset$ sont quatre relations intéressantes pour comparer les ensembles $O(f(n))$ et $O(g(n))$.

A) Pourquoi n'est-il pas intéressant d'utiliser seulement $=$ ou bien \subset pour comparer les ensembles $\Theta(f(n))$ et $\Theta(g(n))$?

B) Que devrait-on utiliser pour comparer $\Theta(f(n))$ et $\Theta(g(n))$?

*Question # 8

Pour chacune des fonctions suivantes, indiquez la classe $\Theta(g(n))$ à laquelle la fonction appartient (Utilisez la forme la plus simple possible pour $g(n)$). Démontrez vos affirmations.

A) $(n^2 + 1)^{10}$

B) $\sqrt{10n^2 + 7n + 3}$

C) $2n \lg(n+2)^2 + (n+2)^2 \lg(\frac{n}{2})$

D) $2^{n+1} + 3^{n-1}$

E) $\lfloor \lg n \rfloor$

*Question # 9

Placez les fonctions suivantes en ordre croissant de leur ordre de croissance.

$$(n-2)!, 5 \lg(n+100)^{10}, 2^{2n}, 0.001n^4 + 3n^3 + 1, (\ln n)^2, \sqrt[3]{n}, 3^n$$

Question # 10

Les valeurs contenus dans le tableau 2.1 (voir A. Levitin) suggère que les fonctions suivantes sont placées en ordre croissant de leur ordre de croissance :

$$\lg n, n, n \lg n, n^2, n^3, 2^n, n!$$

A) Est-ce que les valeurs du tableau forme une preuve que les fonctions sont placées en ordre croissant de leur ordre de croissance ?

B) Démontrez que les fonctions ci-haut sont listées en ordre croissant de leur ordre de croissance.

Question # 11

Démontrez que tout polynôme $p(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_0$ tel que $a_k > 0$ appartient à $\Theta(n^k)$.

Question # 12

Démontrez que deux fonctions exponentielle a^n et b^n ayant des bases différentes ($a \neq b$) ont des ordre de croissance différente.

Question # 13

Démontrez que, pour tout $a \geq 2$ et $b \geq 2$, on a : $\log_b(n) \in \Theta(\log_a(n))$ (et alors $\log_a(n) \in \Theta(\log_b(n))$).

Question # 14

- Elvis, Jimmy et Janis analysent un même algorithme A et obtiennent les résultats $T_E(n)$, $T_i(n)$ et $T_a(n)$ respectivement. Elvis obtient $T_E(n) \in O(n^2)$ en pire cas. Jimmy obtient $T_i(n) \in \Omega(n^2)$ en meilleur cas. Janis affirme que $T_a(n) \in \theta(n^2)$ pour tous les cas. Si Elvis et Jimmy ont raison, le résultat de Janis est-il correct ? Justifiez.

2. Soient A et B des algorithmes prenant des temps, en meilleur cas, dans $\Omega(n^2)$ et $\Omega(n \log n)$ respectivement. Est-il possible qu'une implantation de l'algorithme A soit plus efficace qu'une implantation de l'algorithme B sur tous les exemplaires ? Justifiez.
3. Alice et Bob analysent un même algorithme A et obtiennent les résultats $T_A(n)$ et $T_B(n)$ respectivement. Alice obtient que $T_A(n) \in \theta(n^3)$ en pire cas. Bob obtient que $T_B(n) \in \Omega(n^2 \log n^2)$ pour tous les cas. Le résultat d'Alice implique-t-il celui de Bob ? Justifiez.
4. Énoncez le principe d'invariance en algorithmique.

*Question # 15

Soient A, B, C, D et E cinq algorithmes qui résolvent le même problème P et pour lesquels nous avons obtenu les résultats d'analyse suivants respectivement :

1. $T_A(n) \in O(n^2)$ en pire cas ;
 2. $T_B(n) \in \Omega(n^2)$ dans tous les cas ;
 3. $T_C(n) \in \Theta(n)$ en meilleur cas ;
 4. $T_D(n) \in \Theta(n^2 \lg n)$ dans tous les cas ;
 5. $T_E(n) \in \Theta(n^2)$ en pire cas.
- a) Vous êtes à concevoir un algorithme Z qui résout un problème P' ayant des instances de taille n . Votre algorithme Z ne doit jamais prendre plus de $n^4 \lg n$ unités de temps (à une constante près et pour des entrées de taille assez grande). Dans Z , il y a $n^2 \lg n$ appels à Résoudre le problème P (tous sur des instances de taille n). Donnez, parmi les cinq algorithmes ci-haut, tous ceux qui peuvent être utilisés dans votre algorithme Z , sans dépasser la limite de temps.
 - b) Vous êtes à élaborer un système qui doit fonctionner en temps réel (le facteur temps est primordial). Pour ce faire, vous devez choisir un (et un seul) algorithme parmi les 5 ci-haut. Quel est, selon vous, le choix le plus prometteur ? Justifiez.
 - c) Pour chacun des 5 algorithmes ci-haut, dites, avec justifications, si on est assuré que son temps d'exécution **en pire cas** appartient à l'ensemble suivant :
- $$S = (O(n^3) \cap \Omega(n^2)) \cup \Theta(n).$$
- d) En supposant que les analyses 2 et 3 soient exactes, est-il possible que les algorithmes B et C soient en fait un seul et même algorithme ? Justifiez.

*Question # 16

Déterminez si les énoncés suivants sont vrais ou faux. Justifiez en mots (ou en montrant les étapes de votre démarche si nécessaire).

- A) $376n + 98 \in O(n)$
- B) $2n^2 + n \in \Omega(n)$
- C) $\ln(n^n) + \log_2 n \in \Theta(\ln n)$

*Question # 17

Soit l'algorithme suivant.

```
Algorithm Secret(A[0...n - 1]) :  
    //Entré : Un tableau A[0...n - 1] de n nombres réels  
    minval ← A[0]  
    maxval ← A[0]  
    FOR i ← 1 TO n - 1  
        IF A[i] < minval  
            minval ← A[i]  
        IF A[i] > maxval  
            maxval ← 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 # 18

Soit l'algorithme suivant.

```
Algorithm inefficace(A[0...n - 1]) :  
    //Entrée : Un tableau A[0...n - 1] de n nombres entiers  
    FOR i ← 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 # 19

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 ?

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 # 20

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 # 21

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 # 22

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

```
Algorithm Complex(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 # 23

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 # 24**

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 # 25**

Soit l'algorithme suivant qui résout le même problème que l'algorithme de la question 24.

```
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.

Puisque $C(2^k) = n - 1$ est éventuellement non-décroissante, que $f(n) = n$ est harmonieuse, et que $C(n) \in \Theta(n)$ pour $n = 2^k$ avec $k \in \mathbb{N}$, alors $C(n) \in \Theta(n)$ pour tout $n \in \mathbb{N}$ par la règle de l'harmonie.

B) Lequel des algorithmes Min1 ou Min2 est le plus rapide ? Pouvez-vous construire un nouvel algorithme qui serait plus efficace que Min1 et Min2 tout en résolvant le même problème ?

***Question # 26**

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 $\det A$ le déterminant de la matrice A . Pour $n = 1$, $\det A = a_{11}$ et pour $n > 1$, $\det A = \sum_{j=1}^n s_j a_{1j} \det A_j$ 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 $n!$?