



**POLYTECHNIQUE
MONTREAL**

UNIVERSITÉ
D'INGÉNIERIE

LOG2810
STRUCTURES DISCRÈTES

TD 6 : ALGORITHMES ET ANALYSE DE COMPLEXITÉ H2022

SOLUTIONNAIRE

Directives pour la remise :

- La remise est individuelle, mais le travail en équipe est encouragé.
- La remise est individuelle se fait à la fin de la séance de TD.
- Répondez directement sur ce document Word (docx). Dans l'intérêt de l'équité pour tous les étudiants, vous devez modifier le fichier Word. Modifiez le fichier **EXCLUT** le fait d'intégrer des scans de rédaction manuscrite ou d'y écrire avec un styler.
- Lorsque vous avez terminé, générez un PDF avec le nom sous le format :
Matricule-TDNuméro.pdf (exemple : 1 234567-TD1.pdf).
- Téléversez votre fichier PDF dans la boîte de remise située dans la Zone TDs de la page Moodle du cours.
- Choisissez la boîte de remise qui correspond à votre section de TD.
- **Aucun retard et aucune remise par courriel ne seront acceptés.**
- **Le non-respect des consignes entraînera automatiquement la note 0 pour ce TD.**

Exercice 1. En vous basant sur les définitions des notations de Landau que sont : O , Ω , θ , dites si les affirmations suivantes sont vraies ou fausses. Il n'est pas nécessaire de justifier votre réponse ici.

a) $n(n+1)(n+2)/6 \in O(n^3)$

Réponse : VRAI

En tant que fonction polynôme, elle est O du monôme de plus haut degré, soit n^3 .

b) $n(n+1)(n+2)/6 \in O(n^4)$

Réponse : VRAI

En tant que fonction polynôme, elle est O du monôme de plus haut degré, soit n^3 . Or $n^3 \leq n^4$.

Donc $n(n+1)(n+2)/6 \in O(n^4)$

c) $n(n+1)(n+2)/6 \in \theta(n^4)$

Réponse : FAUX

$n(n+1)(n+2)/6 \in O(n^4)$ mais n^4 n'est pas $O(n(n+1)(n+2))$.

d) $n(n+1)(n+2)/6 \in \Omega(n)$

Réponse : VRAI

n est bien une borne inférieure de $n(n+1)(n+2)/6$

Exercice 2. Déterminez l'ordre du temps de calcul pour les algorithmes suivants. Justifiez vos réponses tout en montrant toutes les étapes qui ont conduit à votre résultat. Si vous avez fait des hypothèses, énoncez-les.

a) **Algorithme 1**

```
1. I = 0
2. Tant que I < N
3.     I = I + 1
```

Réponse :

On fait l'hypothèse qu'une opération élémentaire s'exécute en une unité de temps. Comptons le nombre d'opérations.

- Méthode 1

Ligne 1 : 1 opération

Ligne 2 : $(N + 1)$ comparaisons, donc $(N + 1)$ opérations.

Ligne 3 : 1 opération réalisée N fois, donc N opérations. On considère l'incrémentaire comme une opération élémentaire.

Au total $(1 + (N + 1) + N)$ opérations, soit $(2N + 2)$ opérations.

L'algorithme 1 est $O(N)$.

- Méthode 2

Ligne 1 : 1 opération

Ligne 2 : 1 comparaison, donc 1 opération

Ligne 3 : 1 opération. On considère l'incrémentation comme une opération élémentaire.

La boucle **Tant que** s'exécute N fois et une dernière comparaison qui détermine la sortie de la boucle. Le nombre total d'opérations dans la boucle est donc $2N + 1$.

Au total $(1 + (2N + 1))$ opérations, soit $(2N + 2)$ opérations.

L'algorithme 1 est $O(N)$.

b) **Algorithme 2**

```
1. I = N
2. Tant que I < N
3.     I = I - 2
```

Réponse :

On fait l'hypothèse qu'une opération élémentaire s'exécute en une unité de temps. Comptons le nombre d'opérations.

Ligne 1 : 1 opération

Ligne 2 : 1 comparaison, donc 1 opération

Ligne 3 : 0 opération. La ligne ne s'exécutera pas car on ne rentrera pas dans la boucle.

Au total $(1 + 1)$ opérations, soit 2 opérations.

L'algorithme 2 est $\Theta(1)$.

c) **Algorithme 3.** On suppose que N est connu et est plus grand que 1.

```
1. I = 1
2. P = 0
3. Tant que I <= N
4.     J = 1
5.     Tant que J <= N
6.         P = P + J
7.         J = J + 1
8.     I = I + 1
```

Réponse :

On fait l'hypothèse qu'une opération élémentaire s'exécute en une unité de temps. Comptons le nombre d'opérations.

Ligne 1 : 1 opération

Ligne 2 : 1 opération

Ligne 3 : $(N + 1)$ comparaisons, donc $(N + 1)$ opérations.

Ligne 4 : 1 opération réalisée N fois, donc N opérations.

Ligne 5 : 1 comparaison réalisée N fois, donc N opérations.

Ligne 6 : 1 opération réalisée N fois, donc N opérations.

Ligne 7 : 1 opération réalisée N fois, donc N opérations.

Ligne 8 : 1 opération réalisée N fois, donc N opérations.

La boucle de la ligne 5 à 7 s'exécute N fois en tant que boucle imbriquée. Cette boucle fait donc $N(N+N+N)$ soit $3N^2$.

Au total $(1 + 1 + (N + 1) + N + 3N^2 + N)$ opérations, soit $(3N^2 + 3N + 3)$ opérations.

L'algorithme 3 est $O(N^2)$.

d) Algorithme 4

```
Algorithme Enigme(A[0..n-1, 0..n-1])
// Entrée : Matrice A[0..n-1, 0..n-1] de nombre réels
1. for i=0 to n - 2
2.     for j=i + 1 to n - 1
3.         if A[i, j] = A[j, i]
4.             return false
5. return true
```

Réponse :

On fait l'hypothèse qu'une opération élémentaire s'exécute en une unité de temps. Comptons le nombre d'opérations.

Ligne 1 : 1 opération d'initialisation + $(n - 1)$ incrémentations + $(n - 1)$ comparaisons, soit **$(2n - 1)$ opérations.**

Ligne 2 :

- La boucle for va s'exécuter $(n - 1)$ fois pour les valeurs de i.
- À chaque exécution de cette ligne on dénombre 1 opération d'initialisation + $(n - 1 + 1 - i - 1 = n - 1 - i)$ incrémentations + $(n - 1 + 1 - i - 1 = n - 1 - i)$ comparaisons, soit $(2n - 2i - 1)$ opérations.
- On a donc Somme de $(2n - 2i - 1)$ pour i allant de 0 à $n - 2$ pour la ligne 2. Ce qui donne $(2n(n - 1) - (2(n - 1)(n - 2)/2) - (n - 1))$ opérations, soit **$(n - 1)(n + 1)$ opérations.**

Ligne 3 :

- 2 accès + 1 comparaison, soit 3 opérations à chaque exécution.
- La ligne 3 va s'exécuter à chaque passage dans la boucle for de la ligne 2. Donc pour chaque valeur de i et pour chaque valeur de j. Lorsque i est fixé, la boucle s'exécute $(n - 1 - (i + 1) + 1)$ fois, soit $(n - i - 1)$ fois. Or i change de valeur allant de 0 à $(n - 2)$. Il faut donc somme de $(n - i - 1)$ pour i allant de 0 à $(n - 2)$ passages. On a $(n(n - 1)/2)$ passages.
- Le nombre total d'opération à la ligne 3 est **$(3n(n - 1)/2)$ opérations**

Ligne 4 : 1 opération réalisée à chaque passage dans le if. Dans le **pire scénario**, il y a $(n(n - 1)/2)$ passages, donc **$(n(n - 1)/2)$ opérations.**

Ligne 5 : 1 opération.

Au total $((2n - 1) + (n - 1)(n + 1) + (3n(n - 1)/2) + (n(n - 1)/2) + 1)$ opérations, soit $(3N^2 - 1)$ opérations.

L'algorithme 4 est $O(N^2)$.

Exercice 3. Soit k un nombre entier positif. Montrez que $1^k + 2^k + 3^k + \dots + n^k \in O(n^{(k+1)})$

Réponse :

Pour $n \geq 1$, on a :

$$1^k \leq n^k$$

$$2^k \leq n^k$$

$$3^k \leq n^k$$

...

...

...

$$n^k \leq n^k$$

En sommant membre à membre les n inégalités, on obtient :

$$1^k + 2^k + 3^k + \dots + n^k \leq n^k + n^k + n^k + \dots + n^k$$

n^k étant sommés n fois dans la partie droite de l'inégalité, on peut réécrire l'inégalité comme suit :

$$1^k + 2^k + 3^k + \dots + n^k \leq n \cdot n^k$$

$$1^k + 2^k + 3^k + \dots + n^k \leq n^{(k+1)}$$

En posant $c = 1$, on a :

$$1^k + 2^k + 3^k + \dots + n^k \leq c \cdot n^{(k+1)}$$

Rappelons que cette inégalité est établie pour $n \geq 1$.

En considérant donc $k = 1$ et $c = 1$ on a bien :

$$\text{Pour } n \geq k \text{ et } c = 1, 1^k + 2^k + 3^k + \dots + n^k \leq c \cdot n^{(k+1)}$$

D'où $1^k + 2^k + 3^k + \dots + n^k \in O(n^{(k+1)})$

Exercice 4. Démontrez que $(n^3 + 2n)/(2n + 1)$ est $O(n^2)$.

Réponse :

Par définition, si $(n^3 + 2n)/(2n + 1)$ est $O(n^2)$ alors, il existe c et k tel $n \geq k$ et $(n^3 + 2n)/(2n + 1) \leq c \cdot n^2$

Supposons que l'inégalité est vérifiée et trouvons les valeurs c et k .

$$(n^3 + 2n)/(2n + 1) \leq c \cdot n^2$$

On sait que pour $n \geq 1$, $2n + 1 > 0$. On peut donc écrire successivement :

$$(n^3 + 2n) \leq c \cdot n^2 \cdot (2n + 1)$$

$$(n^3 + 2n) \leq c \cdot (2n^3 + n^2)$$

$$0 \leq (2c - 1) \cdot n^3 + c \cdot n^2 - 2n$$

En posant $c = 1$. On a :

$$0 \leq (2 - 1) \cdot n^3 + n^2 - 2n$$

$$0 \leq n^3 + n^2 - 2n$$

$$0 \leq n(n^2 + n - 2)$$

$$0 \leq n(n + 2)(n - 1)$$

- Lorsque $n \geq 1$, on peut déduire que $n \geq 0$.
- Lorsque $n \geq 1$, on a $(n + 2) \geq 3$ et on déduit que $(n + 2) \geq 0$.
- Lorsque $n \geq 1$, on a $(n - 1) \geq 0$

Ainsi, lorsque $n \geq 1$, $0 \leq n(n + 2)(n - 1)$ est vérifiée.

On peut donc considérer que lorsque $k = 1$ et $c = 1$ on a : pour $n \geq k$, $(n^3 + 2n) / (2n + 1) \leq c.n^2$
D'où $(n^3 + 2n)/(2n + 1)$ est $O(n^2)$

Exercice 5. Donnez une évaluation du comportement asymptotique de chacune des fonctions suivantes, en utilisant le O.

a) $3n^5 + (\log n)^4$

Réponse :

Évaluons d'abord le comportement asymptotique de $(\log n)^4$

On a $\log n \leq n$ donc $(\log n)^4 \leq n^4$

Ainsi, $3n^5 + (\log n)^4 \leq 3n^5 + n^4$

Or $(3n^5 + n^4)$ est $O(n^5)$

Donc $(3n^5 + (\log n)^4)$ est $O(n^5)$

b) $(2^n + n^2)(n^3 + 3^n)$

Réponse :

$(2^n + n^2)$ est $O(2^n)$

$(n^3 + 3^n)$ est $O(3^n)$

$(2^n + n^2)(n^3 + 3^n)$ est $O(2^n \cdot 3^n)$

$(2^n + n^2)(n^3 + 3^n)$ est $O(6^n)$

Exercice 6 (facultatif). Ordonnez les fonctions suivantes de sorte que chaque fonction soit un grand-O de la suivante.

$$1.5^n, n^{100}, (\log n)^3, \sqrt{n} \log n, 10^n, (n!)^2 \text{ et } n^{99} + n^{98}$$

Exemple: $n, n^3, 2^n$

Réponse :

L'ordre est simple lorsque nous nous souvenons que les fonctions factorielles croissent plus vite que les fonctions exponentielles, que les fonctions exponentielles croissent plus vite que les fonctions polynomiales, et que les fonctions logarithmiques croissent très lentement. L'ordre est :

$$(\log n)^3, \sqrt{n} \log n, n^{99} + n^{98}, n^{100}, 1.5^n, 10^n, (n!)^2$$