POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

LOG2810
STRUCTURES DISCRÈTES

TD 6 : **ALGORITHMES ET ANALYSE DE COMPLEXITÉ**

É2022

SOLUTIONNAIRE

PARTIE 1

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) $(5n^2 +13n+4) \in \mathbf{O}(n^2)$
Réponse : VRAI
En tant que fonction polynôme, $(5n^2 +13n+4)$ en O du monôme de plus haut degré, soit $\mathbf{O}(n^2)$.

b) $(5n^2 +13n+4) \in \mathbf{O}(n^3)$
Réponse : VRAI
En tant que fonction polynôme, $(5n^2 +13n+4)$ est en O du monôme de plus haut degré, soit $\mathbf{O}(n^2)$.
Puisque $n^2 \leq n^3$, alors $n^2 \in \mathbf{O}(n^3)$. Par conséquent $(5n^2 +13n+4) \in \mathbf{O}(n^3)$.

c) $n^3 \in \Omega(n^2)$
Réponse : VRAI
Puisque $n^2 \leq n^3$, alors $n^3 \in \mathbf{\Omega}\ (n^2)$.

d) $n^3 \in \theta(n^2)$
Réponse : FAUX
$n^2 \in \mathbf{O}(n^3)$, mais $n^3 \notin \mathbf{O}(n^2)$, puisque $n^3 \in \mathbf{\Omega}\ (n^2)$ alors $n^3 \notin \mathbf{\theta}(n^2)$.

e) $(3n^3 + 2\ n^2) \in \Omega(n^3)$
Réponse : VRAI
$n^3 \leq 3n^3 + 2\ n^2$, alors $(3n^3 + 2\ n^2) \in \Omega(n^3)$ **.**

Exercice 2. Soit deux modules ***A*** et ***B*** de complexités respectives ***A*** ∈ **O**(f(n)) et ***B*** ∈ **O**(g(n)) tel que :

$$f(n) = \begin{cases} n^4, & si\ n\ est\ pair \\ n^2, & si\ n\ est\ impair \end{cases} \qquad g(n) = \begin{cases} n^2, & si\ n\ est\ pair \\ n^3, & si\ n\ est\ impair \end{cases}$$

Déterminez la complexité lorsque les deux modules sont exécutés séquentiellement (A suivi de B ou B suivi de A).

Réponse :
Lorsque les deux modules sont exécutés séquentiellement, la complexité recherchée est celle de A+B. En appliquant la règle de la somme, elle vaut :

$$\mathrm{O}(\max(\mathrm{f(n)}, \mathrm{g(n)}) = \begin{cases} max(n^4, n^2), & si\ n\ est\ pair \\ max(n^2, n^3), & si\ n\ est\ impair \end{cases}$$

Soit :

$$O(\max(f(n), g(n)) = \begin{cases} n^4, & si\ n\ est\ pair \\ n^3, & si\ n\ est\ impair \end{cases}$$

Exercice 3. Démontrez que :

$$log_2(n^2 + 1) \in \theta(log_2(n))$$

Réponse :

Il faut trouver C_1, C_2 et k tel que : $C_1 log_2(n) \leq log_2(n^2 + 1) \leq C_2 log_2(n)$ pour $n > k$.

- On a : $log_2(n) \leq log_2(n^2 + 1)$. Nous pouvons donc choisir $C_1 = 1$ pour tout $n > 0$.

- Ensuite, $log_2(n^2 + 1) \leq log_2(2n^2)$ lorsque $n > 2$.
  Ainsi, $log_2(n^2 + 1) \leq 1 + 2log_2(n)$ lorsque $n > 2$.
  Or, $log_2(n^2 + 1) \leq 1 + 2log_2(n) \leq 3log_2(n)$ lorsque $n > 2$.
  Donc, $log_2(n^2 + 1) \leq 3log_2(n)$ lorsque $n > 2$. Nous pouvons choisir $C_2 = 3$ et $k = 2$.

Conclusion : En choisissant par exemple $C_1 = 1$, $C_2 = 3$ et $k = 2$, on a :

$$log_2(n^2 + 1) \in \boldsymbol{\theta}(log_2(n))$$

PARTIE 2 : EXERCICES SUPPLÉMENTAIRES

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 $(n^3 + 2n)/(2n + 1) \leq c.n^2$
Supposons que l'inégalité est vérifiée et trouvons les valeurs c et k.

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

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

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

En posant c = 1. On a :

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

- 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. 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'incrémentation 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 **θ**(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 6. 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 $O(2^n.3^n)$

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

Exercice 7. Ordonnez les fonctions suivantes de sorte que chaque fonction soit un grand-O de la suivante.

{1.5}^n, n^{100}, (logn)^3, \$sqrt n$ logn, {10}^n, (n!)^2 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 :

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