

TD 1 : GRANDEUR DES FONCTIONS ET COMPLEXITE DES ALGORITHMES ITÉRATIFS

Exercice 1 -

[E. Hebrard, 2018] Soit un algorithme en $O(n^2)$. Un ordinateur X permet de traiter en 1mn des données de taille n_0 . Quelle est la taille des problèmes que l'on pourra traiter en 1mn avec un ordinateur 100 fois plus rapide ? Même question pour $O(2^n)$.

Exercice 2 -

[E. Hebrard, 2018] Lesquelles des assertions suivantes sont vraies. Justifiez votre réponse.

1. $1000 \in O(1)$	6. $n^3 + 3n^2 + n + 2019 \in O(n^3)$
2. $n^2 \in \Omega(n^3)$	7. $n^2 * n^3 \in O(n^3)$
3. $n^3 \in \Omega(n^2)$	8. $2^{2^n} \in O(2^n)$
4. $2^{n+1} \in O(2^n)$	9. $\frac{1}{2}n^2 - 3n \in \theta(n^2)$
5. $(n+1)^2 \in \theta(n^2)$	10. $\frac{1}{2}n^{2*} \Omega(n) \in \Omega(n^3)$

Exercice 3 -

[E. Hebrard, 2018] Est-ce que les propositions suivantes sont vraies pour tout f, g ? Pour chacune, prouvez-le ou trouvez un contre-exemple.

1. $f(n) \in \Omega(g(n))$ alors $g(n) \in O(f(n))$
2. $f(n) \in O(g(n))$ ou $g(n) \in O(f(n))$

Exercice 4 - Ordre de grandeur

Quel est l'ordre de grandeur de ces fonctions suivantes ?

1. $f(n) = 5n^4 + 3n^3 + 2n^2 + 4n + 1$
2. $f(n) = 5n^2 + 3n \cdot \log n + 2n + 5$
3. $f(n) = 2^{n+2}$

Exercice 5 -

a. Ecrivez une relation de récurrence pour chacune des expressions :

- $y = (x_1 + x_2 + \dots + x_n)$
- $y = (x_1 x_2 \dots x_n)$
- $y = 1 - x + x^2/2 - x^3/6 + x^4/24 + \dots + (-1)^n x^n / n!$

b. Soit la définition de relation de récurrence suivante :

$$H(n) = \begin{cases} 1 & \text{si } n = 1 \\ H(n-1) + 1/n & \text{si } n > 1 \end{cases}$$

1. Décrivez le calcul que réalise la définition, pour $n \in \mathbb{N}$.
2. La fonction est-elle récursive terminale ? Si oui, pourquoi ? Sinon, précisez pourquoi elle ne l'est pas et proposez une version récursive terminale.

Exercice 6 -

1. [E. Hebrard, 2018] Que retournent les fonctions ci-dessous ? Evaluer l'ordre de grandeur de leur complexité.

<p>Algorithme: $f_1(n)$ Données: un entier $n \geq 0$ Résultat: ? r, i, j: entier; début $r \leftarrow 0$; pour i allant de 1 à n faire pour j allant de $i+1$ à n faire $r \leftarrow r+1$; retourner(r); fin.</p>	<p>Algorithme: $f_2(n)$ Données: un entier $n \geq 0$ Résultat: ? r, i: entier; début $i \leftarrow n$; $r \leftarrow 0$; tant que $i > 1$ faire pour j allant de $i+1$ à n faire $r \leftarrow r+1$; $i \leftarrow i/2$; retourner(r); fin.</p>
---	---

2. Prouver que la complexité temporelle de l'algorithme de tri classique (tri par sélection) et l'algorithme de tri avancé (tri par fusion) est $O(n^2)$ et $O(n \log n)$ respectivement ?

Exercice 7 -

[E. Hebrard, 2018] Quelle est la complexité de la boucle suivante ?

```

début
1.  $r \leftarrow 0$ ;
2. pour  $i$  allant de 1 à  $n-1$  faire
3.   pour  $j$  allant de  $i+1$  à  $n$  faire
4.     pour  $k$  allant de 1 à  $j$  faire
5.       Instruction;
6.   retourner( $r$ );

```

fin.

Exercice 8 -

Déterminer la complexité des algorithmes **itératifs** suivants, **en fonction de deux paramètres**, où m et n sont deux entiers positifs ? (justifier)

1. Par rapport au nombre d'itérations effectuées pour les algms A, B, C et D.
2. Par rapport au nombre d'opérations arithmétiques pour les algms E et F.

<p><u>Algorithme A</u> Début $i \leftarrow 1$; $j \leftarrow 1$; Tant que $(i \leq m)$ et $(j \leq n)$ faire $i \leftarrow i+1$; $j \leftarrow j+1$; fin tant que; fin.</p>	<p><u>Algorithme B</u> Début $i \leftarrow 1$; $j \leftarrow 1$; tant que $(i \leq m)$ ou $(j \leq n)$ faire $i \leftarrow i+1$; $j \leftarrow j+1$; fin tant que; fin.</p>	<p><u>Algorithme C</u> Début $i \leftarrow 1$; $j \leftarrow 1$; tant que $j \leq n$ faire si $i \leq m$ alors $i \leftarrow i+1$; sinon $j \leftarrow j+1$; fin si; fin tant que; fin.</p>
<p><u>Algorithme D</u> Début $i \leftarrow 1$; $j \leftarrow 1$; tant que $j \leq n$ faire si $i \leq m$ alors $i \leftarrow i+1$; sinon $j \leftarrow j+1$; $i \leftarrow 1$; fin si; fin tant que; fin.</p>	<p><u>Algorithme E</u> Début 1. $s \leftarrow 0$; 2. $i \leftarrow 1$; 3. tant que $i \leq n$ faire 4. pour $j \leftarrow n^2$ à 5 faire 5. $s \leftarrow s+1$; fin pour; 6. $i \leftarrow i+2$; fin tant que; 7. return s Fin.</p>	<p><u>Algorithme F</u> Début 1. Pour i de 1 à $n-1$ faire 2. $tmp \leftarrow i$; 3. pour j de $i+1$ à n faire 4. Si $T[j] < T[tmp]$ alors 5. $tmp \leftarrow j$; fin pour; 6. $T[i] \leftrightarrow T[tmp]$; fin pour; Fin.</p>

Bon courage