

## TD 1 (algorithmique) - Complexité.

### Remarques :

1. Il existe plusieurs manières de répondre aux questions où l'on vous demande d'écrire du code. Trouvez la manière de procéder qui donne la meilleure complexité possible.
2. Certains exercices font appel à la boucle `while`, dont nous reparlerons. Pour l'heure, sachez simplement que :

```
1 while <condition>:
2     # instructions
```

répète le bloc d'instruction du `while` tant que la condition donnée est vraie. N'utilisez pas cette boucle pour répondre aux questions.

### Exercice 1

Donnez la complexité des fonctions suivantes :

<pre>1 def f1(n): 2     s = 0 3     for i in range(n): 4         s = s + 1 - 2 * i 5     return s</pre>	<pre>1 def f2(n): 2     s = 0 3     for i in range(n): 4         s = s * s 5         for j in range(n): 6             s = s + 1 7     return s</pre>	<pre>1 def f3(n): 2     s = 0 3     for i in range(n): 4         for j in range(n * n): 5             s = s + 1 6     return s</pre>
<pre>1 def f4(n): 2     s = 0 3     for i in range(n): 4         j = 0 5         while j &lt;= i: 6             s = s + 1 7     return s</pre>	<pre>1 def f5(n): 2     s = 0 3     for i in range(n): 4         for j in range(i * i): 5             for k in range(j): 6                 s = s + 1 7     return s</pre>	

Que devient la complexité de la fonction `f4` si l'on rajoute `j = j + 1` dans le corps de la seconde boucle ?

### Exercice 2

Calculez la complexité de la fonction suivante (on suppose que l'appel à `print()` est en  $O(1)$ ) :

```
1 def g(n):
2     i = 1
3     p = n
4     while i <= n:
5         for j in range(1, p+1):
6             print(j)
7             i = i + 1
8         i = i + 1
9     return p
```

### Exercice 3

Calculez la complexité du code suivant :

```
1 def h(x, p, n):
2     # x, p, n sont des variables entières
3     r = 0
4     for i in range(1, n+1):
5         for j in range(1, p+1):
6             r = 2 * (i + j) + r
7
8         for k in range(1, x+1):
9             r = r * k
10    print(r)
```

**Exercice 4**

Le  $n^{\text{ème}}$  nombre harmonique  $H_n$  est défini par

$$H_n = \sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}.$$

Écrivez une fonction renvoyant  $H_n$  ( $n$  est supposé naturel) et donnez sa complexité.

**Exercice 5**

La série de Leibniz permet de calculer  $\pi$  comme suit :

$$\pi = 4 \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = 4 \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \right)$$

Écrivez une fonction utilisant cette série pour calculer  $\pi$ , qui utilisera au lieu de  $\infty$  un paramètre naturel  $n$  donnant le nombre d'itérations à effectuer, et donnez sa complexité.

**Exercice 6**

La constante  $e$  peut être calculée par la série suivante, qui s'appuie sur la factorielle  $n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$  (avec  $0! = 1$ ) :

$$e = \sum_{i=0}^{\infty} \frac{1}{i!} = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \dots$$

Écrivez une fonction utilisant cette série pour calculer  $e$ , qui utilisera au lieu de  $\infty$  un paramètre naturel  $n$  donnant le nombre d'itérations à effectuer, et donnez sa complexité.

**Indices**

Pour obtenir une solution efficace, il faut éviter de recalculer chaque factorielle à partir de 1. Utilisez la définition pour trouver une récurrence utile.

**Exercice 7**

Classez les fonctions suivantes par ordre croissant selon la notation  $O(\cdot)$ . Précisez si certaines d'entre elles sont équivalentes, et expliquez pourquoi. Vous pouvez vous aider du graphique comparant les fonctions dans les notes de cours.

- |                     |                  |               |  |
|---------------------|------------------|---------------|--|
| 1) $n$              | 2) $2^n$         | 3) $n \log n$ | 4) $\ln n$                                       |
| 5) $n - n^3 + 7n^5$ | 6) $\log n$      | 7) $\sqrt{n}$ | 8) $e^n$   |
| 9) $n^2 + \log n$   | 10) $n^2$        | 11) $2^{n-1}$ | 12) $\log \log n$                                |
| 13) $n^3$           | 14) $(\log n)^2$ | 15) $n!$      | 16) $n^{1+\varepsilon}$ où $0 < \varepsilon < 1$ |