EA4 – Éléments d'algorithmique
TD n° 2 : notations de Landau et complexité

Les questions facultatives, pour aller plus loin, sont marquées **(*)**.

Exercice 1 : notations O et Θ

1. Montrer que $3n^2 + 5n + 12 \in O(n^2)$.

2. Montrer que $5n^3 - n^2 \in O(n^4)$.

3. Comparer $n!$ à n^n.

4. Montrer que $n^2 + n \log n \in \Theta(n^2)$.

5. Montrer que $8\sqrt{n} \notin \Theta(n)$.

6. **(*)** Montrer que $\log(n!) \in \Theta(n \log n)$.

7. **(*)** Montrer que pour tout $a, b > 1$, $\log_a(n) \in \Theta(\log_b(n))$.

Exercice 2 : propriétés de Θ

Dans la suite, f et g désignent des fonctions à valeurs dans $\mathbb{N}^\star$.

1. Montrer que $\forall f, g,\ f \in \Theta(g) \iff g \in \Theta(f)$.

2. Montrer que $\forall f, g,\ \max(f, g) \in \Theta(f + g)$.

3. Qu'en est-il de $\min(f, g)$?

Exercice 3 : complexité des boucles

Exprimer le plus simplement possible l'ordre de grandeur du nombre d'additions effectuées par les algorithmes suivants :

1.
```
for i in range(n):
    for j in range(n):
      for k in range(n):
        val1 = val1 + 1
```

2.
```
for i in range(n):
    for j in range(i):
      val1 = val1 + 1
```

On considère maintenant des algorithmes prenant un paramètre entier `n`, et deux fonctions de `n` notées f et g.

3. On suppose que l'algorithme `A(n)` se décompose en deux parties consécutives, de complexité (en temps) respective $\Theta(f(n))$ et $\Theta(g(n))$. Quel est l'ordre de grandeur de sa complexité ?

4. On suppose que l'algorithme `A(n)` fait $\Theta(f(n))$ tours de boucle, chacun ayant une complexité $\Theta(g(n))$. Quel est l'ordre de grandeur de sa complexité ?

Exercice 4 : un algorithme récursif

On considère l'algorithme suivant :

```
def F(n) :
  if n < 3 : return 1
  else : return 2 * F(n - 1) + F(n - 3)
```

Soit $A(n)$ le nombre d'additions effectuées lors de l'exécution de `F(n)`.

1. Donner une définition de $A(n)$ par récurrence. En déduire que A est croissante.

2. Montrer que $A(n) \in \Omega(2^{n/3})$ où `/` est la division euclidienne.

3. Proposer un algorithme qui fait un nombre linéaire d'additions pour calculer les mêmes valeurs que `F(n)`.

4. Quel est la complexité de cet algorithme ?

5. Existe-t-il un algorithme de complexité inférieure ?

Exercice 5 : classement

Classer les fonctions suivantes en fonction de leur ordre de grandeur dans les classes Θ_1 à Θ_7 en respectant les conditions suivantes :

– deux fonctions appartiennent à la même classe Θ_i si et seulement si elles sont du même ordre de grandeur :

$$\forall f, g, \quad f \in \Theta(g) \iff \exists i,\ f, g \in \Theta_i$$

– les classes Θ_i sont rangées en ordre de grandeur croissant :

$$\forall i,\ \forall f, g, \quad f \in \Theta_i \text{ et } g \in \Theta_{i+1} \implies f \in O(g)$$

Liste des fonctions à traiter :

n^2-5n, $n+2n^3 \log n$, $n^2+n \log n$, $n^2\sqrt{n}$, $\sqrt{n}$, 2^n, e^n, 2^{n+4}, $\log n$, $\log(n^4)$

Θ_1	Θ_2	Θ_3	Θ_4	Θ_5	Θ_6	Θ_7