TD 01 – Ordres de grandeur, matrices, graphes

Exercice 1. *Ordres de grandeur*

Soit $f(n) = 3n^2 - 100n + 6$.

1. Indiquer (sans justifier) lesquels des énoncés suivants sont vrais ou faux.

a. $f(n) \in \mathcal{O}(n)$	**b.** $f(n) \in \mathcal{O}(n^2)$	**c.** $f(n) \in \mathcal{O}(n^3)$
d. $f(n) \in \Omega(n)$	**e.** $f(n) \in \Omega(n^2)$	**f.** $f(n) \in \Omega(n^3)$
g. $f(n) \in \Theta(n)$	**h.** $f(n) \in \Theta(n^2)$	**i.** $f(n) \in \Theta(n^3)$

2. Classer (sans justifier) les fonctions suivantes selon leur comportement asymptotique, c'est-à-dire $f(n) \preceq g(n)$ ssi $f(n) \in \mathcal{O}(g(n))$.

$$\begin{array}{llll} f_1(n) = n^2 + 10 & f_2(n) = \log(n) & f_3(n) = 2^n & f_4(n) = \frac{n^2}{4} \\ f_5(n) = n! & f_6(n) = n\ \log(n) & f_7(n) = n^n & f_8(n) = \sqrt{\log(n)} \end{array}$$

3. Monter que la relation $\preceq$ est réflexive et transitive, donc c'est un préordre.

Soient $f(n) \in \mathcal{O}(n^2)$ et $g(n) \in \mathcal{O}(n^3)$.

4. Montrer que $f(n) + g(n) \in \mathcal{O}(n^3)$.
5. Montrer que $f(n) - |g(n)| \in \mathcal{O}(n^2)$.
6. Montrer que pour toute fonction $h(n)$ et toute constante $d > 0$, on a $d \cdot h(n) \in \Theta(h(n))$.
7. Montrer que $f(n) \cdot g(n) \in \Theta(n^5)$.
8. Donner deux fonctions $f(n)$ et $g(n)$, qui ne soient pas des polynomes, et telles que $f(n) \in \mathcal{O}(g(n))$ et $f(n) \notin \Omega(g(n))$.

Exercice 2. *Matrices*

Pour une matrice carrée A de taille $n \times n$, on note $A[i][j]$ sa valeur en ligne i et colonne j. On vous rappelle que le produit de deux matrices carrées de taille $n \times n$ est une matrice carrée de taille $n \times n$, donnée par $A \cdot B = P$ avec $P[i][j] = \sum_{k=1}^{n} A[i][k] \cdot B[k][j]$ pour tous $1 \leq i, j \leq n$.

1. Donner un algorithme calculant le produit de deux matrices carrées.
2. Évaluer la complexité de votre algorithme, comme une fonction de n.
3. Votre algorithme est-il de complexité linéaire, quadratique, cubique ?

Exercice 3. *Analyse de complexité*

Soit l'algorithme suivant, prenant en entrée un entier naturel n.

```
res = 0
pour i de 1 à n faire:
    pour j de 1 à i faire:
        res = res + 1
retourner res
```

1. Évaluer la complexité de cet algorithme, comme une fonction de n.

2. Cet algorithme est-il de complexité linéaire, quadratique, cubique ?

Soit l'algorithme suivant, prenant en entrée un entier naturel n.

```
res = 0
pour i de 1 à n faire:
    pour j de 1 à log2(i) faire:
        pour k de j à j+5 faire:
            res = res + 1
retourner res
```

3. Évaluer la complexité de cet algorithme, comme une fonction de n.
   Indication : de la formule de Stirling $n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$ on déduit $\log_2(n!) \in \Theta(n \log_2 n)$.

4. Cet algorithme est-il de complexité linéaire, quadratique, cubique ?

Exercice 4. *Graphes*

Pour un graphe orienté $G = (V, A)$, voici deux algorithmes de calcul du degré sortant $deg^+(v)$ d'un sommet donné $v \in V$.

Matrice d'adjacence `G.A` avec `G.n` $= |V|$:

```
deg = 0
pour i de 1 à G.n faire:
    si G.A[v][i] == 1 alors:
        deg = deg + 1
retourner deg
```

Listes d'adjacence `G.A` avec `G.n` $= |V|$:

```
deg = 0
e = G.A[v]
tant que e != nil faire:
    deg = deg + 1
    e = e.suivant
retourner deg
```

1. Analyser la complexité des deux algorithmes ci-dessus.

2. Donner un algorithme de calcul du degré entrant $deg^-(v)$ pour chacune des deux représentations de graphe (matrice d'adjacence, listes d'adjacence). Analyser leur complexité.

3. Donner un algorithme qui ajoute un arc entre deux sommets u et v donnés en entrée, sans créer un multigraphe, pour les listes d'adjacence. Analyser sa complexité.

4. Donner un algorithme qui calcule le graphe réciproque de $G = (V, A)$, ayant un arc (v, u) pour tout arc $(u, v) \in A$, pour les listes d'adjacence. Analyser sa complexité.

5. Donner un algorithme qui calcule le graphe complémentaire de $G = (V, A)$, ayant l'arc (u, v) si et seulement si $(u, v) \notin A$, pour la matrice d'adjacence. Analyser sa complexité.