

Chapitre 6
Complexité.

Table des matières

1 Définitions.	2
2 Ordre de grandeur	2
3 Type de complexité :	3
4 Estimation du temps de calcul en fonction de la complexité :	3
5 Complément sur les suites récurrentes :	4
5.1 Récurrence linéaire d'ordre 1	4
5.2 Diviser pour régner	5

1 Définitions.

Soit \mathcal{A} un algorithme et d une donnée d'entrée de \mathcal{A} . On appelle complexité de \mathcal{A} pour d (notée $C_{\mathcal{A}}(d)$) le nombre de fois que l'opération élémentaire est exécutée lors de l'exécution de \mathcal{A} sur d .

Soit \mathcal{A} un algorithme, et $D(n)$ l'ensemble de toutes les données d'entrée possibles de \mathcal{A} de tailles n .

Définition 1 :

Complexité dans le pire des cas :

On appelle complexité de \mathcal{A} dans le pire des cas (notée $C_{\mathcal{A},\text{pire}}(n)$ pour une entrée de taille n) la complexité de cet algorithme dans le cas le plus défavorable :

$$C_{\mathcal{A},\text{pire}}(n) = \max_{d \in D(n)} (C_{\mathcal{A}}(d))$$

Définition 2 :

Complexité dans le meilleur des cas :

On appelle complexité de \mathcal{A} dans le meilleur des cas (notée $C_{\mathcal{A},\text{meilleur}}(n)$ pour une entrée de taille n) la complexité de cet algorithme dans le cas le plus favorable :

$$C_{\mathcal{A},\text{meilleur}}(n) = \min_{d \in D(n)} (C_{\mathcal{A}}(d))$$

Définition 3 :

Complexité en moyenne :

On appelle complexité de \mathcal{A} en moyenne (notée $C_{\mathcal{A},\text{moyen}}(n)$ pour une entrée de taille n) la complexité de cet algorithme dans le cas moyen :

$$C_{\mathcal{A},\text{moyen}}(n) = \frac{\sum_{d \in D(n)} (C_{\mathcal{A}}(d))}{\text{card}(D(n))}$$

2 Ordre de grandeur

1. Domination asymptotique : Le grand O :

Définition 4 :

Soit deux fonctions $f : \mathbb{N}^* \rightarrow \mathbb{R}$ et $g : \mathbb{N}^* \rightarrow \mathbb{R}$.

On dira que f est dominée asymptotiquement par g si :

$$\exists c \in \mathbb{R}_+^*, \exists n_0 \in \mathbb{N}^* \text{ tels que } \forall n \in \mathbb{N}^*, n > n_0 \Rightarrow f(n) \leq c.g(n)$$

On note $f = O(g)$.

2. Ordre de grandeur : Le grand thêta :

Définition 5 :

Soit deux fonctions $f : \mathbb{N}^* \rightarrow \mathbb{R}$ et $g : \mathbb{N}^* \rightarrow \mathbb{R}$.

On dira que f et g ont même ordre de grandeur asymptotique si :

$\exists c \in \mathbb{R}_+^*$, $\exists d \in \mathbb{R}_+^*$, $\exists n_0 \in \mathbb{N}^*$ tels que $\forall n \in \mathbb{N}^*$, $n > n_0 \Rightarrow d.g(n) \leq f(n) \leq c.g(n)$

On note $f = \Theta(g)$.

3. Propriétés :

Propriété 1 :

Pour tous $q > 1$, $\alpha > 0$ et $\beta > 0$:

$$\sum_{k=0}^n q^k = \Theta(q^{n+1}) \quad , \quad \sum_{k=0}^n k^\alpha = \Theta(n^{\alpha+1}) \quad , \quad \sum_{k=0}^n k^\alpha (\ln(k))^\beta = \Theta(n^{\alpha+1} (\ln n)^\beta)$$

3 Type de complexité :

$O(1)$	complexité constante (indépendante de la taille de la donnée)
$O(\log(n))$	complexité logarithmique
$O(n)$	complexité linéaire
$O(n \log(n))$	complexité quasi-linéaire
$O(n^2)$	complexité quadratique
$O(n^p)$	complexité polynomiale
$O(2^n)$	complexité exponentielle
$(n!)$	complexité factorielle

4 Estimation du temps de calcul en fonction de la complexité :

Taille n / $C(n)$	$\log(n)$	n	$n \log(n)$	n^2	n^3	2^n
10^2	6,64 ns	0,1 μ s	0,66 μ s	10 μ s	1 ms	4.10 ¹³ années
10^3	9,96 ns	1 μ s	9,96 μ s	1 s	16,6 min	3,4.10 ²⁸⁴ années

5 Complément sur les suites récurrentes :

5.1 Récurrence linéaire d'ordre 1

Soit une suite (u_n) vérifiant la relation :

$$\begin{cases} u_0 \\ u_n \end{cases} = b(n).u_{n-1} + c(n)$$

avec b et c des fonctions et u_0 une constante.

Méthode des facteurs sommant :

On pose : $\begin{cases} f(0) = 1 \\ f(n) = \frac{1}{\prod_{i=1}^n b(i)} \end{cases}$, on obtient $\forall n > 0, f(n)u_n = f(n-1)u_{n-1} + f(n)c(n)$.

On obtient :

$$\forall n > 0, u_n = \frac{1}{f(n)} \left(u_0 + \sum_{i=1}^n f(i)c(i) \right)$$

Exemple 1 :

suite arithmético-géométrique :

Soit une suite (u_n) vérifiant la relation :

$$\begin{cases} u_0 \\ u_n \end{cases} = au_{n-1} + b$$

avec a , b et u_0 trois constantes réelles.

Pour $a > 1$, on trouve $u_n = O(a^n)$

Exemple 2 :

Soit une suite (u_n) vérifiant la relation :

$$\begin{cases} u_0 \\ u_n \end{cases} = u_{n-1} + an + b$$

avec a , b et u_0 trois constantes réelles.

On trouve $u_n = \Theta(n^2)$

Exemple 3 :

Soit une suite (u_n) vérifiant la relation :

$$\begin{cases} u_0 \\ u_n \end{cases} = u_{n-1} + f(n)$$

avec $f(n) = O(n^\alpha)$. On trouve $u_n = O(n^{\alpha+1})$

Propriété 2 :

Si $f = O(g)$ alors $\sum_{k=1}^n f(k) = O\left(\sum_{k=1}^n g(k)\right)$.

5.2 Diviser pour régner

Soit une suite (u_n) vérifiant la relation :

$$\begin{cases} u_0 \\ u_n \end{cases} = au_{\lfloor \frac{n}{2} \rfloor} + bu_{\lfloor \frac{n}{2} \rfloor} + c(n)$$

avec u_0, a, b constantes entières non nulles , c une fonction croissante.

Propriété 3 :

On pose $\alpha = \log(a + b)$:

- Si $c(n) = \Theta(n^\beta)$, avec $\beta < \alpha$, alors $u_n = \Theta(n^\alpha)$.
- Si $c(n) = \Theta(n^\alpha)$, alors $u_n = \Theta(n^\alpha \log(n))$.
- Si $c(n) = \Theta(n^\beta)$, avec $\beta > \alpha$, alors $u_n = \Theta(n^\beta)$.