Complexité pire cas

1 Introduction

L'efficacité d'un programme ne se mesure pas à la concision, ni à la clarté de son code source, mais à la consommation en ressources (temps, espace mémoire...) de son exécution.

Dans ce chapitre, on se concentrera sur la complexité temporelle des fonctions, c'est-à-dire la consommation en temps de leur appel en fonction des valeurs des arguments ou de leurs tailles.

Pour évaluer cette consommation en temps, on dénombre les opérations élémentaires qui seront effectuées à l'appel de la fonction (additions, multiplications, comparaisons...). Ce nombre dépend évidemment des valeurs des arguments à l'appel.

Exemple : Étudions les complexités de quatre fonctions différentes qui, toutes, prennent en argument $(a, b, c, d, n) \in \mathbb{Z}^5$ et testent si $n \in [a..b] + [c..d]$ (*cf.* annexe de ce chapitre).

Si on note $\begin{cases} l_1 = \text{Card}([a..b]) = \max(b - a + 1, 0) \\ l_2 = \text{Card}([c..d]) = \max(d - c + 1, 0) \end{cases}$, alors :

- `est_somme_naif` réalise de l'ordre de $l_1 \times l_2$ opérations
- `est_somme_naif_bis` en réalise aussi de l'ordre de $l_1 \times l_2$ (dans le pire cas)
- `est_somme_bis` en réalise de l'ordre de $l_1 + l_2$
- `est_somme_ter` en réalise de l'ordre de 1.

Remarque : On peut obtenir le temps d'exécution d'un exécutable, chronométré par bash, en tapant la commande `time ./executable` dans le terminal (*cf.* annexe pour les résultats sur l'exemple précédent).

Pour évaluer la complexité d'une fonction, on s'intéresse ici au pire cas, c'est-à-dire que pour une taille donnée, on compte le nombre d'opérations élémentaires des entrées qui maximisent ce nombre.

Pour t une taille d'entrées, on regarde donc :

$$f(t) = \max_{e \text{ entrée de taille } t} g(e)$$

où $g(e)$ donne le nombre d'opérations élémentaires de la fonction sur l'entrée e.

Exemple : Dans l'exemple précédent, `est_somme_naif` et `est_somme_naif_bis` ont donc la même complexité pire cas, même si on peut s'attendre à ce que la dernière fasse en général moins d'opérations à l'appel que la première.

Pour exprimer l'ordre de grandeur de la complexité d'une fonction, on utilise les notations de Landau, que l'on définit dans la partie qui suit.

2 Notations de Landau

Les notations de Landau permettent de comparer les comportements asymptotiques de deux suites, c'est-à-dire leur comportement à l'infini, et ce en omettant les constantes multiplicatives.

Définitions (*notations de Landau*) :

Soient $u, v \in \mathbb{N}^{\mathbb{N}}$ deux suites. On a :

- $u \in \mathrm{O}(v) \iff \exists\, C \in \mathbb{R}^{+*},\ \exists\, n_0 \in \mathbb{N}$ tels que $\forall\, n \in \mathbb{N},\ n \geq n_0 \Longrightarrow u_n \leq C v_n$
- $u \in \Omega(v) \iff \exists\, C \in \mathbb{R}^{+*},\ \exists\, n_0 \in \mathbb{N}$ tels que $\forall\, n \in \mathbb{N},\ n \geq n_0 \Longrightarrow u_n \geq C v_n$
- $u \in \Theta(v) \iff \exists\, (C_1, C_2) \in (\mathbb{R}^{+*})^2,\ \exists\, n_0 \in \mathbb{N}$ tels que $\forall\, n \in \mathbb{N},\ n \geq n_0 \Longrightarrow C_1 v_n \leq u_n \leq C_2 v_n$

Si $u \in \mathrm{O}(v)$, on dit que u est dominée par v.
Si $u \in \Theta(v)$, on dit que u et v sont équivalentes (à ne pas confondre avec "$u \sim v$" en maths).

Remarque : $u \in \mathrm{O}(v) \iff v \in \Omega(u)$
$u \in \Theta(v) \iff (u \in \mathrm{O}(v) \text{ et } u \in \Omega(v)) \iff v \in \Theta(u)$.

Afin d'alléger les notations, on se permettra désormais, pour $((u_n)_{n\in\mathbb{N}}, (v_n)_{n\in\mathbb{N}}) \in (\mathbb{N}^{\mathbb{N}})^2$, d'écrire que $u_n \in \mathrm{O}(v_n)$, que $u_n \in \Omega(v_n)$ ou que $u_n \in \Theta(v_n)$.

Exemple : $(3n+2)_{n\in\mathbb{N}} \in \mathrm{O}(n)$, $\mathrm{O}(n^2)$, $\mathrm{O}(n^3)$ car $\forall\, n \geq 2$, $3n+2 \leq 4n \leq 4n^2 \leq 4n^3$.

Par contre $3n+2 \notin \Theta(n^2)$. En effet, si c'était le cas, on aurait en particulier $3n+2 \in \Omega(n^2)$ donc il existerait $n_0 \in \mathbb{N}^*$ et $C \in \mathbb{R}^{+*}$ tels que

$$\forall\, n \geq n_0,\ 3n+2 \geq Cn^2 \text{ soit } \frac{3n+2}{n^2} \geq C$$

ce qui est impossible car $\dfrac{3n+2}{n^2} = \dfrac{3}{n} + \dfrac{2}{n^2} \xrightarrow[n\to+\infty]{} 0$.

Propriété :

Les relations binaires $\mathcal{R}$ et $\mathcal{R}'$ définies pour $(u, v) \in (\mathbb{N}^{\mathbb{N}})^2$ par

$$u\,\mathcal{R}\,v \iff u \in \mathrm{O}(v) \qquad\qquad u\,\mathcal{R}'\,v \iff u \in \Omega(v)$$

sont transitives.

Preuve :
On utilise le fait que

$$\left.\begin{array}{l} \forall\, n \geq n_0,\ u_n \leq C v_n \\ \forall\, n \geq n_0',\ v_n \leq C' w_n \end{array}\right\} \Longrightarrow \forall\, n \geq \max(n_0, n_0'),\ u_n \leq CC' w_n$$

pour la première équivalence, et on obtient la deuxième par le même principe (appliqué dans l'autre sens).

Corollaire : La relation $\widehat{\mathcal{R}}$ définie pour $(u, v) \in (\mathbb{N}^{\mathbb{N}})^2$ par $u\,\widehat{\mathcal{R}}\,v \iff u \in \Theta(v)$ est transitive. En fait, on vérifie aisément que c'est même une relation d'équivalence.

2.1 Conditions suffisantes

Propriétés :

Soient $u, v \in \mathbb{N}^{\mathbb{N}}$. On suppose que v ne s'annule pas. Alors,

i. Si $\dfrac{u_n}{v_n} \xrightarrow[n\to+\infty]{} 0$, alors $\begin{cases} u \in \mathrm{O}(v) \\ u \notin \Theta(v) \end{cases}$

ii. S'il existe $\ell \in \mathbb{R}^{+*}$ tel que $\dfrac{u_n}{v_n} \xrightarrow[n\to+\infty]{} \ell$, alors $u \in \Theta(v)$

iii. Si $\dfrac{u_n}{v_n} \xrightarrow[n\to+\infty]{} +\infty$, alors $u \notin \mathrm{O}(v)$.

Remarque : Dans le cas ***iii.***, on a aussi $u \in \Omega(v)$, mais cette caractérisation nous intéresse moins pour l'étude de complexités puisque l'on cherche à écrire des algorithmes ou des fonctions de complexité la plus faible possible.

Remarque : Attention, la réciproque de la propriété ***i.*** est fausse, les suites u et v définies ci-dessous en constituent un contre-exemple :

$$\forall\, n \in \mathbb{N},\ u_n = 1 + (-1)^n \times \frac{1}{2} = \begin{cases} 3/2 & \text{si } n \text{ pair} \\ 1/2 & \text{si } n \text{ impair} \end{cases}$$

$$v_n = 1 + (-1)^{n+1} \times \frac{1}{2} = \begin{cases} 1/2 & \text{si } n \text{ pair} \\ 3/2 & \text{sinon} \end{cases}$$

En effet, u_n/v_n n'a pas de limite (ne converge pas et ne diverge pas non plus vers $+\infty$), et pourtant $u \in \Theta(v)$ puisque $u, v \in \Theta(1)$ et que $\widehat{\mathcal{R}}$ est transitive et symétrique.

3 Sommes utiles

Voici les comportements asymptotiques de quelques sommes usuelles :

- $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \in \Theta(n^2)$
- $\displaystyle\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \in \Theta(n^3)$
- $\displaystyle\sum_{i=1}^{n} \frac{1}{i} \in \Theta(\ln n)$
- $\displaystyle\sum_{i=1}^{n} 2^i = 2^{n+1} - 1 \in \Theta(2^n)$
- $\displaystyle\sum_{i=1}^{n} \frac{1}{2^i} = 1 - \frac{1}{2^n} \in \Theta(1)$
- $\displaystyle\sum_{i=1}^{n} \log i \in \Theta(n \log n)$ $(*)$

Preuve de $(*)$:
Pour tout $n \in \mathbb{N}^*$, on a :

$$\begin{aligned} S_n = \textstyle\sum_{i=1}^{n} \log i &= \textstyle\sum_{i=1}^{\lfloor n/2 \rfloor} \log i + \sum_{i=\lfloor n/2 \rfloor + 1}^{n} \log i \\ &\geq \textstyle\sum_{i=\lfloor n/2 \rfloor + 1}^{n} \underbrace{\log(i)}_{\geq \log(n/2)} \\ &\geq \Big(\underbrace{\lfloor n/2 \rfloor - 1}_{\geq n/4} \Big) (\log(n) - \log(2)) \\ &\geq \frac{n}{4}(\log(n) - \log(2)) \text{ pour } n \text{ assez grand} \\ &= \underbrace{\frac{1}{4} n \log(n) - \frac{n \log 2}{4}}_{:= u_n} \end{aligned}$$

Ainsi, $S_n \in \Omega(u_n)$. Or $u_n \in \Theta(n \log n)$, donc en particulier $u_n \in \Omega(n \log n)$, puisque

$$\frac{u_n}{n \log n} = \frac{\frac{n}{4}(\log(n) - \log(2))}{n \log n} = \frac{1}{4}\Big(1 - \frac{\log 2}{\log n}\Big) \xrightarrow[n \to +\infty]{} \frac{1}{4},$$

Par transitivité de la relation $\mathcal{R}'$ définie plus haut, on a alors $S_n \in \Omega(n \log n)$.

D'autre part, $\sum_{i=1}^{n} \log(i) \leq n \log(n)$ donc $S_n \in \mathrm{O}(n \log n)$ ce qui achève la démonstration.