Université Assane SECK de Ziguinchor

Unité de Formation et de Recherche des Sciences et Technologies

Département d'Informatique

CROISSANCE DES FONCTIONS

Licence 1 Math - Physique – Informatique

Janvier 2021

©Youssou DIENG

ydieng@univ-zig.sn

1 - INTRODUCTION

L'ordre de grandeur du temps d'exécution d'un algorithme, défini au chapitre 2, donne une caractérisation simple de l'efficacité de l'algorithme et permet également de comparer les performances relatives d'algorithmes servant à faire le même travail. Quand la taille de l'entrée n devient suffisamment grande, le tri par fusion, avec son temps d'exécution en Θ (n lg n) pour le cas le plus défavorable, l'emporte sur le tri par insertion, dont le temps d'exécution dans le cas le plus défavorable est en Θ (n^2). Bien qu'il soit parfois possible de déterminer le temps d'exécution exact d'un algorithme, comme nous l'avons fait pour le tri par insertion au chapitre 2, cette précision

supplémentaire ne vaut généralement pas la peine d'être calculée. Pour des entrées suffisamment grandes, les effets des constantes multiplicatives et des termes d'ordre inférieur d'un temps d'exécution exact sont négligeables par rapport aux effets de la taille de l'entrée.

Quand on prend des entrées suffisamment grandes pour que seul compte réellement l'ordre de grandeur du temps d'exécution, on étudie les performances asymptotiques des algorithmes. En clair, on étudie la façon dont augmente à la limite le temps d'exécution d'un algorithme quand la taille de l'entrée augmente indéfiniment. En général, un algorithme plus efficace asymptotiquement qu'un autre constitue le meilleur choix, sauf pour les entrées très petites.

Ce chapitre va présenter plusieurs méthodes classiques pour la simplification de l'analyse asymptotique des algorithmes. La première section définira plusieurs types de « notation asymptotique », dont nous avons déjà vu un exemple avec la notation Θ. Seront ensuite présentées diverses conventions de notation utilisées tout au long de ce livre. Le chapitre se terminera par une révision du comportement de fonctions qui reviennent fréquemment dans l'analyse des algorithmes.

2 - NOTATION ASYMPTOTIQUE

Les notations que nous utiliserons pour décrire le temps d'exécution asymptotique d'un algorithme sont définies en termes de fonctions dont le domaine de définition est l'ensemble des entiers naturels N = {0, 1, 2, . . .}. De telles notations sont pratiques pour décrire la fonction T(n) donnant le temps d'exécution du cas le plus défavorable, qui n'est généralement définie que sur des entrées de taille entière. Cependant, il est parfois avantageux d'étendre abusivement

la notation asymptotique de diverses manières. Par exemple, la notation peut être facilement étendue au domaine des nombres

réels, ou, inversement, réduite à un sous-ensemble des entiers naturels. Il est donc important de comprendre la signification précise de la notation, de sorte que, même si l'on en abuse, on n'en mésuse pas. Cette section va définir les notations asymptotiques fondamentales et présenter certains abus fréquents.

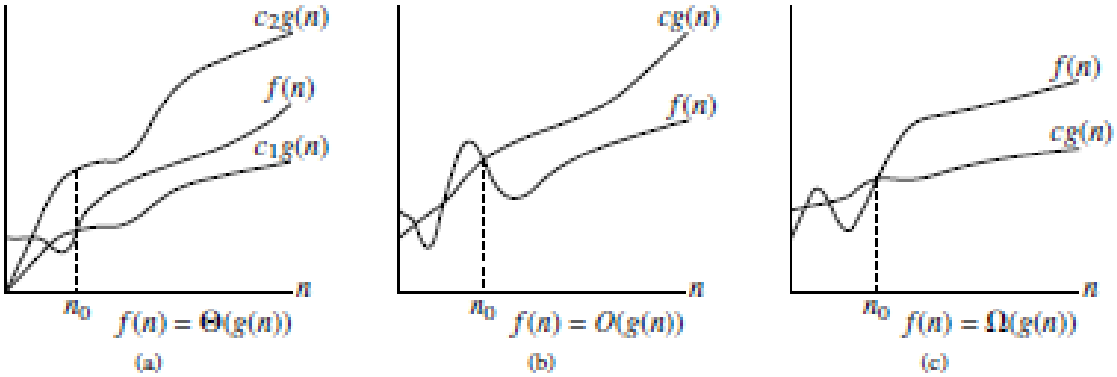
2.1 - Notation Θ

Au chapitre 2, nous avons trouvé que le temps d'exécution, dans le cas le plus défavorable, du tri par insertion est $T(n) = \mathbf{\Theta}(n^2)$. Définissons le sens de cette notation. Pour une fonction donnée g(n), on note $\mathbf{\Theta}(g(n))$ l'ensemble de fonctions suivant :

$\mathbf{\Theta}(g(n)) = \{f(n)$: il existe des constantes positives c1, c2 et n0 telles que $0 \leq c1g(n) \leq f(n) \leq c2g(n)$ pour tout $n \geq n0\}$.

Une fonction f (n) appartient à l'ensemble **Θ** (g(n)) s'il existe des constantes positives c1 et c2 telles que f (n) puisse être « prise en sandwich » entre c1g(n) et c2g(n), pour n assez grand. Comme **Θ** (g(n)) est un ensemble, on pourrait écrire « f (n) ∈ **Θ** (g(n)) » pour indiquer que f (n) est un membre de **Θ** (g(n)). En fait, on préfère écrire « f (n) = **Θ** (g(n)) ». Cet emploi abusif de l'égalité pour représenter l'appartenance à un ensemble peut dérouter au début, mais nous verrons plus loin dans cette section que cela offre des avantages.

La définition de **Θ** (g(n)) impose que chaque membre f (n) ∈ **Θ** (g(n)) soit asymptotiquement positif, c'est-à-dire que f (n) soit toujours positive pour n suffisamment grand. (Une fonction asymptotiquement strictement positive est une fonction qui est positive pour n suffisamment grand.) En conséquence, la fonction g(n) elle-même doit être asymptotiquement positive ; sinon, l'ensemble **Θ** (g(n)) serait vide. On supposera donc que toute fonction utilisée dans la notation **Θ** est asymptotiquement positive. Cette hypothèse reste valable pour toutes les autres notations asymptotiques définies dans ce chapitre.



Dans chaque partie, la valeur de n0 est la valeur minimale possible ; n'importe quelle valeur supérieure ferait aussi l'affaire. (a) La notation Θ borne une fonction entre des facteurs constants. On écrit f(n) = Θ (g(n)) s'il existe des constantes positives n0, c1 et c2 telles que, à droite de n0, la valeur de f(n) soit toujours comprise entre c1g(n) et c2g(n) inclus. (b) La notation O donne une borne supérieure pour une fonction à un facteur constant près. On écrit f(n) = O(g(n)) s'il existe des constantes positives n0 et c telles que, à droite de n0, la valeur de f(n) soit toujours inférieure ou égale à cg(n). (c) La notation V donne une borne inférieure pour une fonction à un facteur constant près. On écrit f(n) = V(g(n)) s'il existe des constantes positives n0 et c telles que, à droite de n0, la valeur de f(n) soit toujours supérieure ou égale à cg(n).

Au chapitre 2, nous avons défini de manière informelle suivante la notation **Θ** : on élimine les termes d'ordre inférieur et on ignore le coefficient du terme d'ordre supérieur. Justifions brièvement cette définition intuitive en utilisant la définition formelle pour montrer que

$$\frac{1}{2}n^2 - 3n = \Theta(n^2)$$

Pour ce faire, on doit déterminer des constantes positives c1, c2 et n0 telles que

$$c_1 n^2 \leqslant \frac{1}{2}n^2 - 3n \leqslant c_2 n^2$$

Pour $n \geq n_0$.

En divisant par n^2 on a :

$$c_1 \leqslant \frac{1}{2} - \frac{3}{n} \leqslant c_2 .$$

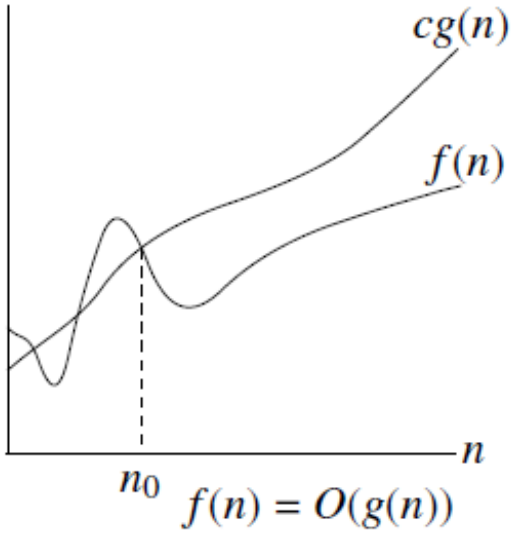
On peut s'arranger pour que le membre droit de l'inégalité soit valide pour toute valeur de $n \geq 1$, et ce en choisissant $c2 \geq 1/2$. De même, on peut faire en sorte que le membre gauche de l'inégalité soit valide pour toute valeur de $n \geq 7$, en choisissant $c1 \leq 1/14$. Donc, en prenant c1 = 1/14, c2 = 1/2 et n0 = 7, on peut vérifier que $1/2\ n^2 - 3n = \Theta(n^2)$. D'autres choix sont bien possibles pour les constantes ; l'important est qu'il existe au moins une possibilité. Notez que ces constantes dépendent de la fonction $\frac{1}{2}\ n^2 - 3n$ une autre fonction appartenant à $\Theta\ (n^2)$ exigerait normalement des constantes différentes.

2.2 - Notation O

La notation O borne asymptotiquement une fonction à la fois par excès et par défaut. La notation O est utilisée donner une borne supérieure asymptotique. Pour une fonction g(n) donnée, on note O(g(n)) (prononcer « grand O de g de n » ou « O de g de n ») l'ensemble de fonctions suivant :

$$O(g(n)) = \{f(n) : \text{ il existe des constantes positives } c \text{ et } n_0 \text{ telles que } 0 \leqslant f(n) \leqslant cg(n) \text{ pour tout } n \geqslant n_0\} .$$

La notation O sert à majorer une fonction, à un facteur constant près. Pour indiquer qu'une fonction f(n) est membre de l'ensemble O(g(n)), on écrit f (n) = O(g(n)).



On remarquera que f (n) = Θ(g(n)) implique f (n) = O(g(n)), puisque la notation Θ est une notion plus forte que la notation O. En termes de théorie des ensembles, on a Θ(g(n)) ⊆ O(g(n)). Donc, notre preuve que toute fonction quadratique $an^2 + bn + c$, avec a > 0, est dans $\Theta(n^2)$ prouve également que toute fonction quadratique de ce genre est dans $O(n^2)$. f (n) = O(g(n)), veut dire simplement qu'un certain multiple constant de g(n) est une borne supérieure de f (n), (sans indiquer quoi que ce soit sur le degré d'approximation de la borne supérieure).

La notation O permet souvent de décrire le temps d'exécution d'un algorithme rien qu'en étudiant la structure globale de l'algorithme. Par exemple, la structure de boucles imbriquées de l'algorithme du tri par insertion (chapitre 2) donne immédiatement une borne supérieure en $O(n^2)$ pour le temps d'exécution du cas le plus défavorable : le coût de chaque itération de la boucle intérieure est borné supérieurement par O(1) (constante), les indices i et

j valent tous deux au plus n, et la boucle intérieure est exécutée au plus une fois pour chacune des n^2 paires de valeurs de i et j.

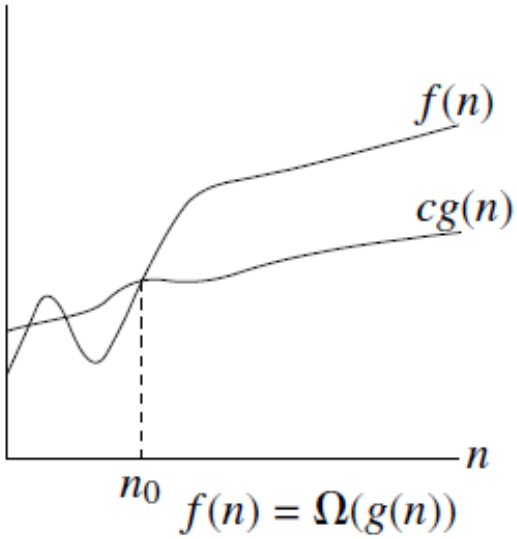
2.3 - Notation O & Notation Θ

Puisque la notation O décrit une borne supérieure, quand on l'utilise pour borner le temps d'exécution du cas le plus défavorable d'un algorithme, on borne donc aussi le temps d'exécution de cet algorithme pour des entrées quelconques. Ainsi, la borne $O(n^2)$ concernant le cas le plus défavorable du tri par insertion s'applique à toutes les entrées possibles. En revanche, la borne $\Theta(n^2)$ concernant le temps d'exécution du tri par insertion dans le cas le plus défavorable n'implique pas que, pour chaque entrée, le tri par insertion ait un temps d'exécution borné par $\Theta(n^2)$. Par exemple, nous avons vu au chapitre 2 que, si l'entrée était déjà triée, le tri par insertion s'exécutait avec un temps en $\Theta(n)$.

2.4 - Notation Ω

De même que la notation O fournit une borne supérieure asymptotique pour une fonction, la notation Ω fournit une borne inférieure asymptotique. Pour une fonction g(n) donnée, on note $\Omega(g(n))$ (prononcer « grand oméga de g de n » ou « oméga de g de n ») l'ensemble des fonctions suivant :

$$\Omega(g(n)) = \{f(n) : \text{ il existe des constantes positives } c \text{ et } n_0 \text{ telles que } 0 \leqslant cg(n) \leqslant f(n) \text{ pour tout } n \geqslant n_0\} .$$



Théorème : *Pour deux fonctions quelconques f (n) et g(n), on a f(n) = Θ(g(n)) si et seulement si f (n) = O(g(n)) et f (n) = Ω (g(n)).*

Comme exemple d'application de ce théorème, notre démonstration que $an^2 + bn + c = \Theta(n^2)$ pour toutes constantes a, b et c, avec a > 0, implique immédiatement que $an^2 + bn + c = \Omega(n^2)$ et que $an^2 + bn + c = O(n^2)$.

Puisque la notation Ω décrit une borne inférieure, alors: l'utiliser pour borner le temps d'exécution d'un algorithme dans le cas optimal, c'est borner également le temps d'exécution de l'algorithme pour des entrées arbitraires. Par exemple, le temps d'exécution du tri par insertion dans cas optimal est Ω(n), ce qui implique que le temps d'exécution du tri par insertion est Ω(n). Le temps d'exécution du tri par insertion est donc compris entre Ω(n) et $O(n^2)$, puisqu'il se situe quelque part entre une fonction linéaire de n et une fonction quadratique de n.

2.5 - Notation o

La borne supérieure asymptotique fournie par la notation O peut être ou non asymptotiquement approchée. La borne $2n^2 = O(n^2)$ est asymptotiquement approchée, mais la borne $2n = O(n^2)$ ne l'est pas. On utilise la notation o pour noter une borne supérieure qui n'est pas asymptotiquement approchée. On définit formellement o(g(n)) (« petit o de g de n ») comme étant l'ensemble

$$o(g(n)) = \{f(n) : \text{ pour toute constante } c > 0, \text{ il existe une constante } n_0 > 0 \text{ telle que } 0 \leqslant f(n) < cg(n) \text{ pour tout } n \geqslant n_0\}$$

Par exemple, $2n = o(n^2)$ mais $2n^2 \neq o(n^2)$.

Les définitions des notations O et o se ressemblent. La différence principale est que dans f (n) = O(g(n)), la borne 0 ≤ f (n)≤ cg(n) est valable pour une certaine constante c > 0, alors que dans f (n) = o(g(n)), la borne 0 ≤ f (n) < cg(n) est valable pour toutes les constantes c > 0. De manière intuitive, avec la notation o, la fonction f (n) devient négligeable par rapport à g(n) quand n tend vers l'infini ; en d'autres termes :

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

2.6 - Notation ω

Par analogie, la notation ω est à la notation Ω ce que la notation o est à la notation O. On utilise la notation ω pour indiquer une borne inférieure qui n'est pas asymptotiquement approchée. Une façon de la définir est :

$$f(n) \in \omega(g(n)) \text{ si et seulement si } g(n) \in o(f(n))$$

Formellement, on définit $\omega(g(n))$ (« petit oméga de g de n ») comme l'ensemble:

$$\omega(g(n)) = \{f(n) : \text{ pour toute constante } c > 0, \text{ il existe une constante } n_0 > 0 \text{ telle que } 0 \leqslant cg(n) < f(n) \text{ pour tout } n \geqslant n_0\}$$

Exemple: $n^2/2 = \omega(n)$ mais $n^2/2 \neq \omega(n^2)$.

La relation $f(n) = \omega(g(n))$ implique que

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

si la limite existe.

2.7 - Comparaison de fonctions

On suppose que $f(n)$ et $g(n)$ sont asymptotiquement positives.

- **Transitivité :**

$$\begin{aligned}
f(n) &= \Theta(g(n)) \text{ et } g(n) = \Theta(h(n)) \text{ implique } f(n) = \Theta(h(n)), \\
f(n) &= O(g(n)) \text{ et } g(n) = O(h(n)) \text{ implique } f(n) = O(h(n)), \\
f(n) &= \Omega(g(n)) \text{ et } g(n) = \Omega(h(n)) \text{ implique } f(n) = \Omega(h(n)), \\
f(n) &= o(g(n)) \text{ et } g(n) = o(h(n)) \text{ implique } f(n) = o(h(n)), \\
f(n) &= \omega(g(n)) \text{ et } g(n) = \omega(h(n)) \text{ implique } f(n) = \omega(h(n)).
\end{aligned}$$

- **Réflexivité :**

$$\begin{aligned}
f(n) &= \Theta(f(n)), \\
f(n) &= O(f(n)), \\
f(n) &= \Omega(f(n)).
\end{aligned}$$

- **Symétrie :**

$$f(n) = \Theta(g(n)) \text{ si et seulement si } g(n) = \Theta(f(n)).$$

- **Symétrie transposée :**

$$f(n) = O(g(n)) \quad \text{si et seulement si} \quad g(n) = \Omega(f(n)) ,$$
$$f(n) = o(g(n)) \quad \text{si et seulement si} \quad g(n) = \omega(f(n)) .$$

Comme ces propriétés sont vraies pour des notations asymptotiques, on peut dégager une analogie entre la comparaison asymptotique de deux fonctions *f* et *g* et la comparaison de deux réels *a* et *b*:

$$\begin{aligned} f(n) = O(g(n)) &\approx a \leqslant b , \\ f(n) = \Omega(g(n)) &\approx a \geqslant b , \\ f(n) = \Theta(g(n)) &\approx a = b , \\ f(n) = o(g(n)) &\approx a < b , \\ f(n) = \omega(g(n)) &\approx a > b . \end{aligned}$$