

Algorithmique III. L2 Informatique I41.

TD 3. Complexité et notations asymptotiques¹

Dans tous les exercices, \mathcal{F} désigne l'ensemble des fonctions de \mathbb{R}_+^* dans \mathbb{R}_+^* dont la variable est dénotée n .

EXERCICE 1. (1) En utilisant les notations asymptotiques, évaluez la fonction de coût en temps $C(n, m)$ de chacun des 4 algorithmes ci-dessous où n et m sont les paramètres de ces algorithmes.

00 ALGORITHME A(n,m)	00 ALGORITHME B(n,m)
01 DONNEES	01 DONNEES
02 · n,m:entiers	02 · n,m:entiers
03 VARIABLES	03 VARIABLES
04 · i,j:entiers	04 · i,j:entiers
05 DEBUT	05 DEBUT
06 · i ← 0	06 · i ← 0
07 · j ← 0	07 · j ← 0
08 · TQ (i < n) ET (j < m) FAIRE	08 · TQ (i < n) OU (j < m) FAIRE
09 · · i ← i + 1	09 · · i ← i + 1
10 · · j ← j + 1	10 · · j ← j + 1
11 · FTQ	11 · FTQ
12 FIN	12 FIN
00 ALGORITHME C(n,m)	00 ALGORITHME D(n,m)
01 DONNEES	01 DONNEES
02 · n,m:entiers	02 · n,m:entiers
03 VARIABLES	03 VARIABLES
04 · i,j:entiers	04 · i,j:entiers
05 DEBUT	05 DEBUT
06 · i ← 0	06 · i ← 0
07 · j ← 0	07 · j ← 0
08 · TQ (j < m) FAIRE	08 · TQ (j < m) FAIRE
09 · · SI (i < n) ALORS	09 · · SI (i < n) ALORS
10 · · · i ← i + 1	10 · · · i ← i + 1
11 · · SINON ·	11 · · SINON ·
12 · · · j ← j + 1	12 · · · j ← j + 1
13 · · FSI	13 · · · i ← 0
14 · FTQ	14 · · FSI
15 FIN	15 · FTQ
	16 FIN

(2) Pourquoi les fonctions de coût de ces algorithmes ainsi calculées ne sont pas leurs fonctions de complexité? Quels paramètres faudrait-il utiliser pour le faire?

Solution. (1) Dans ces 4 algorithmes, le passage des paramètres et l'initialisation des variables i et j a un coût constant de $\Theta(1)$.

Algorithme A : les variables i et j sont incrémentées à chaque passage dans la boucle TQ (#08) et la condition d'arrêt est

$$\neg((i < n) \wedge (j < m)) \equiv (i \geq n) \vee (j \geq m).$$

c'est donc la variable qui atteindra sa borne la première qui fera sortir de la boucle. La condition de boucle sera donc évaluée $\min\{n, m\} + 1$ fois et les incréments le sont $\min\{n, m\}$ fois, avec un coût unitaire de $\Theta(1)$. On a donc

$$\begin{aligned} T(n, m) &= \Theta(1) + (1 + \min\{n, m\})\Theta(1) + (\min\{n, m\})\Theta(1) \\ &= \Theta(\min\{n, m\}). \end{aligned}$$

Algorithme B : même analyse mais cette fois on ne sort de la boucle qu'après que les deux variables ont atteint leurs bornes :

$$\begin{aligned} T(n, m) &= \Theta(1) + (1 + \max\{n, m\})\Theta(1) + (\max\{n, m\})\Theta(1) \\ &= \Theta(\max\{n, m\}). \end{aligned}$$

Algorithme C : compte tenu de la conditionnelle SI dans la boucle TQ, les n premiers passages dans la boucle ne font qu'incrémenter i atteignant ainsi la borne n , les m passages suivants ne font qu'incrémenter la variable j . La condition d'entrée dans la boucle est donc évaluée $n + m + 1$ fois et la conditionnelle SI $n + m$ fois, les instructions #10 et #12 étant exécutées n fois et m fois respectivement avec un coût unitaire constant de $\Theta(1)$. On a donc

$$\begin{aligned} T(n, m) &= \Theta(1) + (n + m + 1)\Theta(1) + (n + m)\Theta(1) + n\Theta(1) + m\Theta(1) \\ &= \Theta(n + m). \end{aligned}$$

Algorithme D : la variable j est incrémentée à chaque fois que la variable i atteint la borne n , ce qui nécessite n incréments de i à chaque fois puisque i est réinitialisée à cette occasion (#13). La condition de boucle TQ est donc évaluée $(n + 1)m + 1$ fois, la conditionnelle SI $(n + 1)m$ fois,

1. Version du 3 mars 2023, 10 : 06

l'incrémentation de i est réalisée nm fois et celle de j est réalisée m fois. On obtient

$$\begin{aligned} T(n, m) &= \Theta(1) + (n + 1)m\Theta(1) + nm\Theta(1) + m\Theta(1) \\ &= \Theta(nm). \end{aligned}$$

(2) Par définition, les fonctions de complexité dépendent de la *taille* des données à traiter par l'algorithme. Cette taille est définie sans ambiguïté dans le contexte de la machine RAM puisqu'il s'agit du nombre de cellules utilisées pour coder les données mais dans le cadre plus souple de notre pseudo-langage algorithmique, cela demande réflexion. L'erreur à ne pas commettre est de considérer que la taille des données est égale au nombre de paramètres de l'algorithme, ici 2, il y aurait alors confusion entre un entier naturel et sa représentation. Il est clair que la taille mémoire nécessaire à stocker l'entier 13 n'est pas la même que pour l'entier 13^{2022} . En anticipant sur la [planche de TD suivante](#), le nombre de chiffres nécessaires pour coder un entier n en base b est égal à $\lfloor \log_b(n) \rfloor + 1$. Autrement dit, les fonctions de complexité auraient dû dépendre de deux variables k_n et k_m désignant respectivement le nombre de chiffres de la représentation des entiers n et m dans une base arbitraire b .

EXERCICE 2. Donnez des exemples de fonctions qui sont en $\Omega(n^2)$ mais pas en $\Theta(n^2)$.

Solution. On a

$$\begin{aligned} f = \Theta(g) &\Leftrightarrow (f = O(g)) \wedge (f = \Omega(g)) \\ \text{donc } f \neq \Theta(g) &\Leftrightarrow (f \neq O(g)) \vee (f \neq \Omega(g)) \end{aligned}$$

Par conséquent, si $f = \Omega(n^2)$ et $f \neq \Theta(n^2)$, nécessairement $f \neq O(n^2)$. On cherche donc des fonctions qui sont minorées mais pas majorées par des fonctions quadratiques. On n'a que l'embarras du choix : $f(n) := n^3$, $f(n) := n^2 \log(n)$, etc.

EXERCICE 3. Soit f et g deux fonctions de \mathcal{F} . Démontrez que

$$f = \Theta(g) \Leftrightarrow (f = O(g)) \wedge (f = \Omega(g)). \quad (1)$$

Solution. On rappelle que :

$$\Theta(g) = \{f \in \mathcal{F} \mid \exists(a, b, N) \in (\mathbb{R}_+^*)^2 \times \mathbb{N} \ \forall n \geq N \ ag(n) \leq f(n) \leq bg(n)\}.$$

$$\Omega(g) = \{f \in \mathcal{F} \mid \exists(c, N) \in \mathbb{R}_+^* \times \mathbb{N} \ \forall n \geq N \ cg(n) \leq f(n)\}.$$

$$O(g) = \{f \in \mathcal{F} \mid \exists(c, N) \in \mathbb{R}_+^* \times \mathbb{N} \ \forall n \geq N \ f(n) \leq cg(n)\}.$$

Dans le sens direct, si $f = \Theta(g)$, alors on dispose d'un triplet (a, b, N) tel qu'à partir du rang N la fonction f est minorée par la fonction ag et majorée par la fonction bg .

Pour la réciproque, si $f \in \Omega(g)$ alors on dispose d'un couple (c_Ω, N_Ω) tel qu'à partir du rang N_Ω , f est minorée par la fonction $c_\Omega g$; symétriquement si $f \in O(g)$ alors on dispose d'un couple (c_O, N_O) tel qu'à partir du rang N_O , f est majorée par la fonction $c_O g$. On pose alors $N := \max\{N_\Omega, N_O\}$, $a := c_\Omega$ et $b := c_O$, ce qui nous fournit le triplet (a, b, N) tel qu'à partir du rang N la fonction f est bornée par les deux fonctions ag et bg .

EXERCICE 4. Soit f et g deux fonctions de \mathcal{F} . Démontrez que

$$f = O(g) \Leftrightarrow g = \Omega(f). \quad (2)$$

Solution. Si $f = O(g)$, on sait qu'on dispose d'un couple (c, N) tel qu'à partir du rang N , la fonction f est majorée par la fonction cg , i.e

$$\forall n \geq N \ f(n) \leq cg(n)$$

et en posant $c' := \frac{1}{c}$ (on rappelle que $c > 0$), on en déduit que

$$\forall n \geq N \ c'f(n) \leq g(n).$$

Autrement dit il existe un couple (c', N) tel qu'à partir du rang N la fonction g est minorée par la fonction $c'f$ soit $g \in \Omega(f)$. La preuve de la réciproque est symétrique.

EXERCICE 5. Soit $f \in \mathcal{F}$. Montrez que si f est définie par $f(n) := 3n^2 + 7$, alors $f = O(n^2)$ et que si f est définie par $f(n) := 3n^3 - 2n + 1$, alors $f = \Omega(n^3)$.

Solution. La constante $c := 4$ et le rang $N := 3$, entre autres, nous donnent le résultat attendu :

$$\forall n \geq 4 \quad 3n^2 + 7 \leq 4n^2.$$

En effet, l'inégalité est équivalente à $n^2 - 7 \geq 0$ et toutes les valeurs de n supérieures à $\sqrt{7}$ conviennent.

Pour la fonction $3n^3 - 2n + 1$, la constante $c := 2$ et le rang $N := 1$ conviennent :

$$\forall n \geq 1 \quad 2n^3 \leq 3n^3 - 2n + 1. \quad (3)$$

En effet, l'inégalité est équivalente à $n^3 - 2n + 1 \geq 0$, or la fonction $l(x) := n^3 - 2n + 1$ a pour dérivée $3n^2 - 2$ qui est positive pour $n \geq 1$ ce qui valide la proposition (3) puisque $l(1) = 0$ et l est croissante.

EXERCICE 6. Montrez que si f est une fonction polynomiale de degré d à coefficient dominant positif (i.e. le monôme de plus haut degré a un coefficient positif), alors

$$f = \Theta(n^d). \quad (4)$$

Solution. D'après (1) il faut montrer que $f = O(n^d)$ et que $f = \Omega(n^d)$. Notons $f(x) = a_0 + a_1x + \dots + a_dx^d$ avec $a_d > 0$ par hypothèse. Il est clair que si on définit $a := \max\{a_i \mid 0 \leq i \leq d\}$, on a

$$\forall x \geq 1 \quad \forall i \in [0, d] \quad a_i x^i \leq ax^d.$$

On en déduit que

$$\forall n \geq 1 \quad f(n) \leq (d+1)an^d.$$

Ce qui nous fournit un rang $N := 1$ et une constante $c := (d+1)a$ pour prouver que $f = O(n^d)$ (notons que la majoration est grossière).

Montrons à présent que $f = \Omega(n^d)$. En choisissant la constante $c := \frac{a_d}{2}$, nous allons montrer qu'il existe un rang N à partir duquel $f(x)$ est minorée par $\frac{a_d}{2}x^d$, autrement dit que la fonction $g(x) := f(x) - \frac{a_d}{2}x^d$ est positive à partir de ce rang N . On a :

$$g(x) = x^d \underbrace{\left(\frac{a_d}{2} + \frac{a_{d-1}}{x} + \dots + \frac{a_1}{x^{d-1}} + \frac{a_0}{x^d} \right)}_S.$$

Comme on ne s'intéresse qu'aux valeurs x positives, il nous reste à trouver un rang N à partir duquel $S \geq 0$. Notons a le plus grand coefficient du polynôme en valeur absolue, i.e. $a := \max\{|a_i| \mid 0 \leq i \leq d\}$. En posant

$$N := \left\lceil \frac{2da}{a_d} \right\rceil,$$

on assure que si $x \geq N$, chacun des d termes $\frac{|a_i|}{x^{d-i}}$ pour $i \in [0, d-1]$ est bien inférieur à $\frac{a_d}{2d}$ et par conséquent que $S \geq 0$.

EXERCICE 7. Soit $g \in \mathcal{F}$. Montrez que si $k \in \mathbb{R}$ est une constante strictement positive, alors

$$k \times O(g) = O(g)$$

mais qu'en revanche

$$n \times O(g) = O(ng).$$

Solution. Pour toute fonction f dans la classe $O(g)$ il existe un couple $(c, N) \in \mathbb{R}_+^* \times \mathbb{N}$ tel que

$$\forall n \in \mathbb{N} \quad n \geq N \quad g(n) \leq cg(n)$$

et la fonction kf est encore dans la classe $O(g)$ avec le couple (kc, N) puisque k est une constante. En revanche la fonction nf est bien dans la classe $O(ng)$ puisque

$$\forall n \in \mathbb{N} \quad n \geq N \quad nf(n) \leq cng(n).$$

EXERCICE 8. Soit $n \in \mathbb{N}$. Calculez et/ou simplifiez :

- | | |
|-----------------------------|-----------------------------|
| (1) $\Theta(n) + \Theta(n)$ | (5) $O(n)O(n)$ |
| (2) $O(n) + \Theta(n)$ | (6) $n\Theta(1)$ |
| (3) $\Theta(n) + \Theta(1)$ | (7) $O(n) + O(n^2)$ |
| (4) $O(n) + \Omega(n)$ | (8) $\Theta(\log_2(n) + n)$ |

Solution. Pour effectuer ces calculs, il faut se donner une fonction de chaque classe intervenant dans l'expression, réaliser l'opération entre ces fonctions et déduire de nouvelles majorations ou minorations en partant de celles fournies par l'appartenance à ces classes.

- | | |
|---|---|
| (1) $\Theta(n) + \Theta(n) = \Theta(n)$ | (5) $O(n)O(n) = O(n^2)$ |
| (2) $O(n) + \Theta(n) = \Theta(n)$ | (6) $n\Theta(1) = \Theta(n)$ |
| (3) $\Theta(n) + \Theta(1) = \Theta(n)$ | (7) $O(n) + O(n^2) = O(n^2)$ |
| (4) $O(n) + \Omega(n) = \Omega(n)$ | (8) $\Theta(\log_2(n) + n) = \Theta(n)$ |

Attention, malgré l'apparente symétrie entre les notations O et Ω , il faut prendre garde au fait que les fonctions sont supposées *positives*, donc minorées par 0, alors qu'elles ne sont pas majorées par défaut. Ainsi pour le calcul de $O(n) + \Omega(n)$, si on prend deux fonctions f et g représentatives de ces classes respectivement, on dispose de deux constantes b et a et d'un rang N (le max des deux rangs fournis par $f = O(n)$ et $g = \Omega(n)$) à partir duquel

$$\begin{aligned} 0 &\leq f(n) \leq bn \\ an &\leq g(n) \end{aligned}$$

On en déduit la minoration $an \leq f(n) + g(n)$, mais ne disposant pas d'une majoration de $g(n)$, on ne peut en déduire une majoration de $f(n) + g(n)$.

EXERCICE 9. Démontrez que la fonction définie par $n \mapsto n^2$ est en $o(n^3)$.

Solution. On rappelle que $f = o(g)$ si et seulement si

$$\forall c \in \mathbb{R}_+^* \quad \exists N \in \mathbb{N} \quad \forall n \geq N \quad f(n) \leq cg(n).$$

Soit $c \in \mathbb{R}_+^*$, il est facile de vérifier que si $n \geq \frac{1}{c}$ alors $n^2 \leq cn^3$, on peut donc choisir le rang $N := \lceil \frac{1}{c} \rceil$ où $\lceil x \rceil$ désigne le plus petit entier supérieur ou égal au réel x .

EXERCICE 10. (1) Démontrez que la somme de la série de terme général 2^{-n} est égale à 2.

(2) Démontrez que la série harmonique de terme général $\frac{1}{n}$ n'est pas convergente en utilisant une [minoration par une intégrale](#).

(3) En remarquant que

$$1 + \frac{1}{2} + \underbrace{\frac{1}{3} + \frac{1}{4}}_{> \frac{1}{4} + \frac{1}{4}} + \underbrace{\frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8}}_{> \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}} + \dots$$

démontrez d'une autre façon que cette série est divergente.

Solution. (1) Soit $q \in \mathbb{R}$ avec $q \neq 1$ et $n \in \mathbb{N}$. On rappelle que

$$S_q(n) := \sum_{i=0}^n q^i = \frac{q^{n+1} - 1}{q - 1}.$$

On a donc

$$\begin{aligned} S_{2^{-1}}(n) &= \frac{2^{-(n+1)} - 1}{2^{-1} - 1} \\ &= 2 - \frac{1}{2^n} \end{aligned}$$

Cette suite est croissante et majorée (par sa borne supérieure 2), on en déduit que la somme de la série existe et

$$\lim_{n \rightarrow +\infty} S_{2^{-1}}(n) = 2.$$

(2) La suite de terme général $\frac{1}{n}$ est monotone *décroissante*, on peut donc appliquer la minoration par une intégrale pour la fonction réelle définie par $f(x) := \frac{1}{x}$:

$$\begin{aligned} \sum_{i=1}^n \frac{1}{i} &\geq \int_1^{n+1} f(x) dx = \int_1^{n+1} \frac{1}{x} dx. \\ &= [\ln x]_1^{n+1} \\ &= \ln(n+1). \end{aligned}$$

et la suite de terme général $\ln(n+1)$ est divergente, c'est-à-dire :

$$\forall M \in \mathbb{R}_+ \quad \exists N \in \mathbb{N} \quad \forall n \geq N \quad \ln(n+1) > M,$$

proposition que l'on résume par :

$$\lim_{n \rightarrow +\infty} \ln(n+1) = +\infty.$$

(3) Les ensembles $I_0 := \{1\}$, $I_1 := \{2\}$ et $I_k := [2^{k-1} + 1, 2^k]$ pour $k \geq 2$ forment une partition de \mathbb{N}^* et pour $n \in \mathbb{N}^*$, on définit $k := \lceil \log_2 n \rceil$ et les sommes

$$P_k := \sum_{i \in I_k} \frac{1}{i}.$$

Ce que l'on illustre ici :

$$\underbrace{1}_{P_0} + \underbrace{\frac{1}{2}}_{P_1} + \underbrace{\frac{1}{3} + \frac{1}{4}}_{P_2} + \underbrace{\frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8}}_{P_3} + \dots$$

Pour tout $k \in \mathbb{N}$, chaque terme de la somme P_k est supérieur à 2^{-k} et il y en a 2^{k-1} , on en déduit immédiatement que

$$\forall k \in \mathbb{N} \quad P_k \geq \frac{1}{2}.$$

On a donc :

$$\sum_{i=1}^n \frac{1}{i} \geq \sum_{j=0}^{k-1} P_j \geq \frac{k}{2} = \frac{\lceil \log_2 n \rceil}{2}.$$

Ce qui prouve que la série de terme général $\frac{1}{n}$ diverge puisque la fonction logarithme est strictement croissante et divergente.

EXERCICE 11. Deux algorithmes A et B sur la machine RAM résolvent le même problème et ont respectivement pour complexité moyenne les fonctions $\bar{T}_A(n) := 1\,220n + 654$ et $\bar{T}_B(n) := \frac{1}{4}n^2 + 4n - 63$ où n désigne la taille des données à traiter. Quelle est la plus petite valeur de n pour laquelle l'algorithme A est plus efficace que l'algorithme B ?

Solution. Il suffit d'étudier l'inégalité

$$1\,220n + 654 < \frac{1}{4}n^2 + 4n - 63$$

qui se traduit par une étude de signe :

$$n^2 - 4\,864n - 2\,868 > 0$$

et il est aisé de vérifier que la plus petite valeur est $n = 4865$. On rappelle qu'une fonction polynomiale réelle de degré 2, de coefficient dominant a et de discriminant Δ a le signe :

- (1) de a pour tout $x \in \mathbb{R}$ si $\Delta < 0$.
- (2) de a pour tout $x \in \mathbb{R} \setminus \{x_0\}$ si $\Delta = 0$ et x_0 est la racine double.
- (3) de a pour tout $x \in \mathbb{R} \setminus [x_1, x_2]$ si $\Delta > 0$ et x_1 et x_2 sont les deux racines, avec $x_1 < x_2$, et de $-a$ pour tout $x \in]x_1, x_2[$.