L'algorithme d'Euclide

Marc Lorenzi

10 décembre 2023

Résumé

L'étude d'un algorithme est une opération en trois étapes : sa *description*, sa *correction* et sa *complexité*. Nous allons dans cet article étudier l'algorithme d'Euclide qui permet de calculer le plus grand commun diviseur de deux entiers.

1 Description

1.1 L'algorithme

Dans tout l'article, on se donne $a, b \in \mathbb{N}$. L'algorithme que nous allons étudier fonctionne très bien si $a, b \in \mathbb{Z}$ mais prenons a et b positifs pour simplifier l'exposé.

On définit une suite $(r_n)_{n\geq 0}$ d'éléments de $\mathbb{N}$ par récurrence à deux termes. Tout d'abord, $r_0 = a$ et $r_1 = b$. Puis, pour tout $n \in \mathbb{N}$,

- Si $r_{n+1} \neq 0$ alors r_{n+2} est le reste de la division euclidienne de r_n par r_{n+1}.
- Si $r_{n+1} = 0$ alors $r_{n+2} = 0$.

Remarque. Si $a < b$ alors le reste de la division euclidienne de a par b est a. On a donc $r_1 = b$ et $r_2 = a$.

Remarque. Si $b = 0$ alors pour tout $n \geq 1$, $r_n = 0$. Dans ce cas, les propriétés que nous allons énoncer dans la suite sont immédiates.

Voici les premières valeurs de la suite $(r_n)_{n\geq 0}$ pour $a = 654$ et $b = 456$.

n	0	1	2	3	4	5	6	7	8
r_n	654	456	198	60	18	6	0	0	0

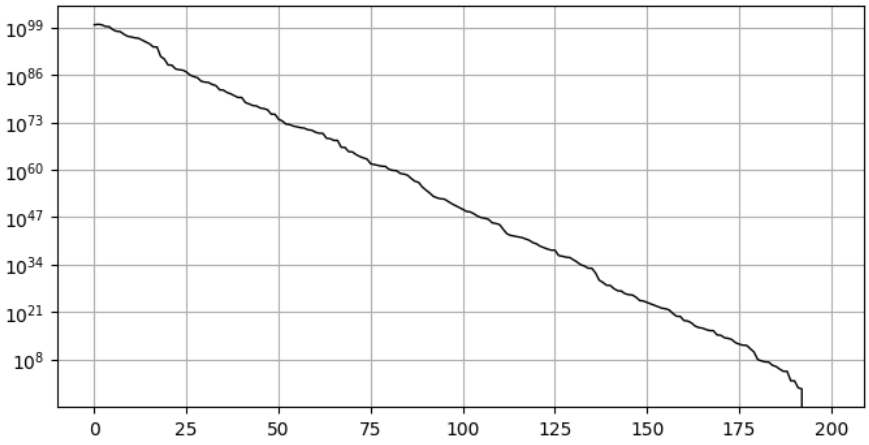
On vérifie facilement que pour tout $n \geq 1$,

- Si $r_n = 0$ alors pour tout $k \in [\![n, +\infty[\![$, $r_k = 0$.
- Si $r_n \neq 0$ alors pour tout $k \in [\![1, n]\!]$, $r_k \neq 0$.

Nous sommes donc en présence de deux possibilités.

- Première possibilité : il existe $n \geq 1$ tel que $r_n = 0$.
- Deuxième possibilité : pour tout $n \geq 1$, $r_n \neq 0$.

Pour prendre un exemple un peu plus compliqué, voici le graphe de la suite associée à deux entiers aléatoires a et b de 100 chiffres en base 10. Le graphique est en coordonnées semi-logarithmiques. Dans cet exemple, $r_n = 0$ dès que n dépasse environ 190.



1.2 Un exemple assez particulier

Introduisons la *suite de Fibonacci* $(F_n)_{n\geq 0}$ définie par $F_0 = 0$, $F_1 = 1$ et, pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$.

Soit $N \in \mathbb{N}$. Posons $a = F_{N+1}$ et $b = F_N$. Quelle est la suite $(r_n)_{n\geq 0}$ associée à a et b ?

Proposition 1. *Pour tout $k \in [\![0, N+1]\!]$, $r_k = F_{N-k+1}$.*

Démonstration. Prouvons le résultat par récurrence à deux termes sur k. C'est immédiat pour $k = 0$. Soit $k \in [\![0, N-1]\!]$. Supposons $r_k = F_{N-k+1}$ et $r_{k+1} = F_{N-k}$. Comme $k+1 \leq N$, on a $N - k \geq 1$ et donc $F_{N-k} \neq 0$. L'entier r_{k+2} est donc le reste de la division euclidienne de r_k par r_{k+1}. Écrivons

$$r_k = F_{N-k+1} = F_{N-k} + F_{n-k-1} = r_{k+1} + F_{n-k-1}$$

Remarquons que $F_{n-k-1} < F_{N-k}$ pour en déduire que le reste cherché est F_{n-k-1}. Ainsi, $r_{k+2} = F_{n-k-1}$. □

Par la proposition 1, $r_N = F_1 \neq 0$ et $r_{N+1} = F_0 = 0$. Pour tout $k \geq N+1$, on a donc $r_k = 0$. La suite associée à $a = F_{N+1}$ et $b = F_N$ est donc

$$(F_{N+1}, F_N, F_{N-1}, \ldots, F_1, F_0 = 0, 0, 0, \ldots)$$

1.3 Un autre exemple

Pour tout $n \geq 1$, le n^{e} nombre de Mersenne est $M_n = 2^n - 1$. Soient $A = M_a$ et $B = M_b$. Quelle est la suite des restes $(R_n)_{n\geq 0}$ associée à A et B ?

Proposition 2. *Pour tout $n \in \mathbb{N}$, $R_n = M_{r_n}$.*

Démonstration. Montrons le résultat par récurrence à deux termes sur n. C'est clair pour $n = 0$ et $n = 1$. Soit $n \in \mathbb{N}$. Supposons que $R_n = M_{r_n}$ et $R_{n+1} = M_{r_{n+1}}$. Si $r_{n+1} = 0$ alors $R_{n+1} = M_0 = 0$, donc $R_{n+2} = 0 = M_0 = M_{r_{n+2}}$. Sinon, R_{n+2} est le reste de la division de R_n par R_{n+1}. Écrivons $r_n = qr_{n+1} + r_{n+2}$, où $q \in \mathbb{Z}$. On a

$$\begin{aligned} R_n &= 2^{qr_{n+1}+r_{n+2}} - 1 \\ &= (2^{r_{n+1}})^q 2^{r_{n+2}} - 1 \\ &= (R_{n+1}+1)^q 2^{r_{n+2}} - 1 \\ &= 2^{r_{n+2}} \sum_{k=0}^{q} \binom{q}{k} R_{n+1}^k - 1 \\ &= R_{n+1} 2^{r_{n+2}} \sum_{k=1}^{q} \binom{q}{k} R_{n+1}^{k-1} + 2^{r_{n+2}} - 1 \\ &= QR_{n+1} + M_{r_{n+2}} \end{aligned}$$

où

$$Q = 2^{r_{n+2}} \sum_{k=1}^{q} \binom{q}{k} R_{n+1}^{k-1}$$

Comme $2^{r_{n+2}} - 1 < 2^{r_{n+1}} - 1$, on en déduit que le quotient et le reste de la division euclidienne de R_n par R_{n+1} sont Q et $R_{n+2} = M_{r_{n+2}}$. □

1.4 Et maintenant ?

Décrire un algorithme est un bon début. C'est ce que nous venons de faire, en rajoutant quelques exemple pour bien comprendre son fonctionnement. Cela dit, encore faut-il que l'algorithme que nous venons de décrire fasse ce que l'on attend de lui (et qu'en attend-on précisément ?). Nous allons maintenant prouver deux choses.

1. L'algorithme termine. Rappelons que nous sommes en présence de deux possibilités.
   - Première possibilité : il existe $n \geq 1$ tel que $r_n = 0$.
   - Deuxième possibilité : pour tout $n \geq 1$, $r_n \neq 0$.

   Nous allons montrer que la deuxième possibilité est exclue.
2. Il est correct, c'est à dire que l'un des restes r_n est un nombre qui nous intéresse. Remarquons que, même si le lecteur s'en doute, nous n'avons pas encore précisé ce que l'on entend par « intéressant ».

2 Terminaison et correction

2.1 Terminaison

Proposition 3. *Il existe $n \geq 1$ tel que $r_n = 0$.*

Démonstration. Supposons le contraire. Pour tout $n \in \mathbb{N}$, r_{n+2} est le reste de la division euclidienne de r_n par r_{n+1} et donc $0 \leq r_{n+2} < r_{n+1}$. La suite $(r_n)_{n\geq 1}$ est alors une suite strictement décroissante d'entiers naturels. Or, une telle suite n'existe pas (pourquoi ?), contradiction. □

Notation. Dorénavant, n_0 désigne le plus petit entier $n \geq 1$ tel que $r_n = 0$.

On a donc

- $r_{n_0} = 0$.
- Pour tout $k \in [\![1, n_0 - 1]\!]$, $r_k \neq 0$. Et donc, pour tout $k \in [\![0, n_0 - 2]\!]$, r_{k+2} est le reste de la division euclidienne de r_k par r_{k+1}.
- Pour tout $k \geq n_0$, $r_k = 0$.

2.2 Correction

Nous allons voir que le nombre « intéressant » dont nous avons parlé un peu plus haut est r_{n_0-1}.

Lemme 4. *Soit $d \in \mathbb{Z}$. Pour tout $k \in [\![0, n_0 - 2]\!]$, d divise r_k et r_{k+1} si et seulement si d divise r_{k+1} et r_{k+2}.*

Démonstration. Soit $k \in [\![0, n_0 - 2]\!]$. Supposons que d divise r_k et r_{k+1}. Écrivons $r_k = qr_{k+1} + r_{k+2}$ où $q \in \mathbb{Z}$ est le quotient de la division euclidienne de r_k par r_{k+1}. Comme d divise r_k et r_{k+1}, il divise aussi $r_{k+2} = r_k - qr_{k+1}$. Donc, d divise r_{k+1} et r_{k+2}. La réciproque se montre de la même façon. □

lemme 5. *Soit $d \in \mathbb{Z}$. d divise a et b si et seulement si d divise r_{n_0-1}.*

Démonstration. Par le lemme 4, une récurrence immédiate sur k montre que pour tout $k \in [\![0, n_0 - 1]\!]$, d divise a et b si et seulement si d divise r_k et r_{k+1}. En prenant $k = n_0 - 1$, d divise a et b si et seulement si d divise r_{n_0-1} et $r_{n_0} = 0$, c'est à dire si et seulement si d divise r_{n_0-1}. □

Proposition 6. *$r_{n_0-1} = pgcd(a, b)$.*

Démonstration. C'est immédiat à partir du lemme 5, par la définition même du pgcd de deux entiers. □

2.3 Et maintenant ?

Savoir qu'un algorithme termine et qu'il est correct, c'est bien. Mais cet algorithme est-il efficace ? Et qu'entend-on par « efficace » ? C'est ce que nous allons maintenant examiner.

3 Complexité

Notons $N(a, b)$ le nombre de divisions euclidiennes nécessaires à l'algorithme d'Euclide pour calculer le pgcd de a et b. On a clairement $N(a, b) = n_0 - 1$.

Que peut-on au mieux attendre concernant la valeur de $N(a, b)$? De façon purement heuristique, pour calculer le pgcd de a et b il faut au minimum regarder b. Et pour regarder b il faut regarder chacun des chiffres de b (en base 2, par exemple). Le nombre de chiffres de l'écriture de b en base 2 est

$$\lambda(b) = \lfloor \lg b \rfloor + 1 = O(\lg b)$$

Nous allons voir que l'algorithme d'Euclide est très efficace puisque l'on a effectivement $N(a,b) = O(\lg b)$.

Lemme 7. *Pour tout $k \in [\![1, n_0 - 2]\!]$, $r_k \geq r_{k+1} + r_{k+2}$.*

Démonstration. Soit $k \in [\![1, n_0 - 2]\!]$. Comme r_{k+1} est le reste de la division euclidienne de r_{k-1} par r_k, on a $r_{k+1} < r_k$. De là, le quotient q de la division euclidienne de r_k par r_{k+1} n'est pas nul et donc

$$r_k = qr_{k+1} + r_{k+2} \geq r_{k+1} + r_{k+2}$$

□

Lemme 8. *Pour tout $k \in [\![0, n_0 - 1]\!]$, $r_{n_0-k} \geq F_k$.*

Démonstration. Montrons le résultat par récurrence à deux termes sur k. Tout d'abord, $r_{n_0} = 0 = F_0$ et $r_{n_0-1} \geq 1 = F_1$. Soit $k \in [\![0, n_0 - 3]\!]$. Supposons que $r_{n_0-k} \geq F_k$ et $r_{n_0-k-1} \geq F_{k+1}$. Par le lemme 7 appliqué à $n_0 - k - 2$,

$$r_{n_0-k-2} \geq r_{n_0-k-1} + r_{n_0-k} \geq F_{k+1} + F_k = F_{k+2}$$

□

Proposition 9. *$F_{N(a,b)} \leq b$.*

Démonstration. On applique le lemme 8 à $k = n_0 - 1$ pour obtenir

$$b = r_1 \geq F_{n_0-1} = F_{N(a,b)}$$

□

L'équation $x^2 - x - 1 = 0$ d'inconnue $x \in \mathbb{R}$ possède deux racines réelles qui sont $\frac{1}{2}(1 \pm \sqrt{5})$. Posons $\varphi = \frac{1}{2}(1 + \sqrt{5})$ et $\overline{\varphi} = \frac{1}{2}(1 - \sqrt{5})$. Remarquons que $\varphi > 1$ et $-1 < \overline{\varphi} < 0$.

Proposition 10. *$N(a,b) \leq \log_\varphi(b\sqrt{5} + 1)$.*

Démonstration. Pour tout $n \in \mathbb{N}$,

$$F_n = \frac{\varphi^n - \overline{\varphi}^n}{\sqrt{5}}$$

De là,

$$\varphi^{N(a,b)} \leq b\sqrt{5} + \overline{\varphi}^{N(a,b)} \leq b\sqrt{5} + 1$$

En passant aux logarithmes en base φ on obtient le résultat. □

Corollaire 11. *$N(a,b) = O(\lg b)$.*

Démonstration. Pour tout réel $x \geq 1$, on a

$$\log_\varphi(x+1) - \log_\varphi x = \log_\varphi\left(1 + \frac{1}{x}\right) \leq \log_\varphi 2$$

De là, si $b \neq 0$,

$$\begin{aligned} N(a,b) &\leq \log_\varphi(b\sqrt{5}) + \log_\varphi 2 \\ &= \log_\varphi b + \log_\varphi(2\sqrt{5}) \end{aligned}$$

Ainsi, pour tout $b \geq 2$,

$$0 \leq \frac{N(a,b)}{\lg b} \leq \frac{1}{\lg \varphi} + \frac{\log_\varphi(2\sqrt{5})}{\lg 2}$$

La suite $\left(\frac{N(a,b)}{\lg b}\right)_{b\geq 2}$ est donc bornée. □

Remarquons que $1/\lg\varphi \leq 1.4405$ et $\log_\varphi(2\sqrt{5}) \leq 3.1127$. On a donc la majoration plus explicite

$$N(a,b) \leq 1.4405 \lg b + 3.1127$$

Remarque. L'étude de complexité que nous venons de faire ne tient pas compte du coût réel d'une division euclidienne. Évidemment, le coût d'une division est d'autant plus important que les nombres mis en jeu sont grands. Il s'avère que la division de a par b peut être faite en temps $O(\lg a \lg b)$. Le coût de l'algorithme d'Euclide en nombre d'opérations sur des bits est aussi, de façon assez surprenante, $O(\lg a \lg b)$.

Bibliographie.

[1] Joachim von zur Gathen, Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.

[2] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2008.