UFR Sciences
M2 Calcul Formel Martin WEIMANN

Algorithme d'Euclide et applications

Soit A un anneau principal, et soient a et b deux éléments de A tels que $(a,b) \neq (0,0)$. Il existe alors d dans A tel que:

- $d|a$ et $d|b$
- Si $e|a$ et $e|b$, alors $e|d$.

Cet élément d, unique à multiplication près par un élément inversible de A, est appelé le plus grand diviseur commun ou pgcd de a et b, et noté $\mathrm{pgcd}(a,b)$ (ou parfois (a,b) si le contexte ne permet pas d'ambiguïté avec l'élément (a,b) de A^2).

On a alors $aA + bA = dA$. En particulier, il existe u et v dans A tels que $au + bv = d$ (*relation de Bézout*). Ceci donne une définition légèrement plus générale : on appelle pgcd de (a,b) tout générateur de l'idéal $aA + bA$ de A engendré par a et b ; cette définition est équivalente à la précédente quand $(a,b) \neq 0$ et permet de définir $\mathrm{pgcd}(0,0) = 0$.

On dit que a et b sont copremiers (ou premiers entre eux) si on peut choisir $\mathrm{pgcd}(a,b) = 1$. De manière équivalente, a et b sont copremiers s'il existe une relation de Bezout $au + bv = 1$.

Tout anneau euclidien étant principal, on peut y définir la notion de pgcd. Une succession de divisions euclidiennes permet alors de calculer ce pgcd: c'est l'algorithme d'Euclide.

1 Algorithme d'Euclide

1.1 Calcul du pgcd de deux entiers

Soient a et b deux entiers. On définit une suite (a_i) en posant $a_0 = a$, $a_1 = b$ puis en définissant a_{i+1} à partir de la division euclidienne de a_{i-1} par a_i :

$$a_{i-1} = q_i a_i + a_{i+1}, \quad 0 \leqslant a_{i+1} < |a_i|.$$

Cette suite est bien définie tant que $a_i \neq 0$. En notant $imax$ l'indice du dernier reste non nul, on a alors $\mathrm{pgcd}(a,b) = a_{imax}$.

Remarque 1 Le pgcd de deux entiers n'est défini qu'au signe près.

Remarque 2 Il y a beaucoup de généralisations de cet algorithme. Par exemple, on peut choisir a_{i+1} dans tout intervalle de longueur $|a_i|$ contenant 0, e.g. l'intervalle centré $-\frac{|a_i|}{2} < a_{i+1} \leqslant \frac{|a_i|}{2}$.

Exercice 1 Prouver que l'algorithme d'Euclide est correct.

Exercice 2 Calculer le pgcd de 10 et 13, de 12 et 15, de 10^{1000} et $10^{1000}+3$, de $10^{1000}+2$ et $10^{1000}+5$, de n et $n+3$ (on pourra discuter en fonction de la valeur de $n \in \mathbb{Z}$).

Exercice 3 Soit $n \in \mathbb{Z}$. Calculer le pgcd de n^2+2n+3 et n^2+n+9 en discutant en fonction des valeurs de n. Même question avec n^2 et n^2+n+29.

Exercice 4 Soient a et b deux entiers naturels. On note g leur pgcd. Montrer que pour tout entier n, le pgcd de n^a-1 et n^b-1 vaut n^g-1.

Exercice 5 Suppose $a \geq b \geq 0$. Montrer que l'algorithme d'Euclide sur a, b réalise au plus $O(\log_2(a))$ divisions.

Exercice 6 (Théorème de Lamé) On cherche une borne plus fine sur le nombre d'itérations que celle de l'exercice précédent. On note F_k le k-ème terme de la suite de Fibonacci, définie par $F_0 = 0$, $F_1 = 1$ et $F_{k+2} = F_{k+1} + F_k$ (récurrence linéaire d'ordre 2 à coefficients constants).

1. Montrer que pour tout $k \geq 1$, si $a > b \geq 1$, et $b < F_{k+1}$, alors l'algorithme d'Euclide sur a et b réalise moins de k itérations (e.g. montrer par récurrence que si l'algo utilise plus de k divisions, alors $a \geq F_{k+2}$ et $b \geq F_{k+1}$).
2. En utilisant une formule explicite pour F_k, en déduire que l'algorithme d'Euclide effectue au plus $\log_\phi(1+\sqrt{5}b) - 1$ divisions euclidiennes, où $\phi = (1+\sqrt{5})/2$ est le nombre d'or.

1.2 Relation de Bézout – algorithme d'Euclide étendu

Etant donnés $a, b \in \mathbb{Z}$, il existe $u, v \in \mathbb{Z}$ tels que $au + bv = \text{pgcd}(a, b)$. On appelle une telle égalité une *relation de Bezout*. On dit que (u, v) sont des *coefficients de Bezout* du couple (a, b). Les coefficients de Bezout sont *réduits* si

$$|u| < \frac{|b|}{|\text{pgcd}(a,b)|} \quad \text{et} \quad |v| < \frac{|a|}{|\text{pgcd}(a,b)|}.$$

Lors de l'algorithme d'Euclide, on peut garder l'information donnée par les quotients successifs q_i pour calculer des coefficients de Bezout de a et b. Il s'agit de l'algorithme d'Euclide étendu :

L'algorithme d'Euclide étendu. On pose $u_0 = 0$, $v_0 = 1$, $u_1 = 1$, $v_1 = 0$. On définit alors u_{i+1} et v_{i+1} par

$$\begin{cases} u_{i+1} = u_{i-1} - q_i u_i \\ v_{i+1} = v_{i-1} - q_i v_i \end{cases}$$

facile à retenir étant donnée l'égalité $a_{i+1} = a_{i-1} - q_i a_i$. Une récurrence (attention à initialiser pour $i = 0$ et $i = 1$) permet d'établir la relation $u_i a + v_i b = a_i$ d'où il suit que $(u, v) := (u_{imax}, v_{imax})$ sont des coefficients de Bezout de (a, b).

Exercice 7 Quel est le pgcd de 7 et 5 ? Écrire l'algorithme d'Euclide correspondant et l'identité de Bézout obtenue.

Exercice 8 Soient $a, b \in \mathbb{Z}$ de coefficients de Bezout (u, v). Décrire l'ensemble de tous les coeficients de Bezout de (a, b). Montrer qu'au signe près, les coefficients de Bezout réduits sont uniques.

Remarque 3 Est-ce que l'algorithme d'Euclide étendu calcule de fait cette unique (au signe près) paire u, v ? A vérifier...

Exercice 9 Soit $N \in \mathbb{N}^\times$. On représente en général sur machine un élément de $\mathbb{Z}/N\mathbb{Z}$ par un entier dans l'intervalle $[[0, N-1]]$. Comment calculer l'inverse d'un élément inversible de $\mathbb{Z}/N\mathbb{Z}$?

1.3 Algorithme d'Euclide (étendu) sur $K[X]$

Dès que l'on dispose d'une division euclidienne, on peut écrire un algorithme d'Euclide. En particulier, on obtient ainsi un algorithme d'Euclide pour les polynômes similaire à celui pour les entiers, avec maintenant la condition $\deg(a_{i+1}) < \deg(a_i)$. Le calcul des coefficients de Bezout est identique au cas des entiers.

Remarque 4 Le pgcd de deux polynômes n'est défini qu'à un inversible de $K[X]$ près, c'est-à-dire ici à multiplication par une constante non nulle près.

Exercice 10 En analogie avec les entiers, énoncer ce qu'est une relation de Bezout réduite dans $K[X]$. Montrer qu'il existe une unique relation de Bezout réduite à la multiplication par un scalaire près. Expliquer comment calculer cette relation de Bezout réduite à partir d'une relation de Bezout quelconque.

2 Complexité

2.1 Euclide

Théorème 1 L'algorithme d'Euclide coûte :

1. $O(mn)$ opérations dans K pour deux polynômes $a, b \in K[X]$ de degrés respectifs n et m.
2. $O(mn)$ opérations binaires pour deux entiers $a, b \in \mathbb{Z}$ de tailles respectives n et m.

Preuve (cas des polynômes). Soit $n = \deg a$ et $m = \deg b$. On peut toujours supposer $m \leq n$. On note $\ell = imax + 1$ (donc $q_\ell = 0$). Pour tout $i < \ell$, on note $n_i = \deg a_i$. D'après le théorème 6 du CM1, on peut diviser a_{i-1} par a_i avec au plus $(2n_i + 1)(n_{i-1} - n_i + 1)$ opérations $+, -, \times$ dans K et une division. Donc le coût total est au plus

$$T = \sum_{i=1}^{\ell-1} (2n_i + 1)(n_{i-1} - n_i + 1) \leq (2m+1) \sum_{i=1}^{\ell-1} (n_{i-1} - n_i + 1) \leq (2m+1)(n - n_\ell + \ell - 1)$$

opérations dans K (et $\ell - 1$ divisions). Comme $n_1 = m$ et la suite des degrés n_i est strictement décroissante, on a $\ell \leq m + 1$. Ainsi, l'algorithme utilise au plus $(2m+1)(n+m-1)$ opérations dans K, soit une complexité $T = O(mn)$. □

2.2 Euclide étendu

Pour estimer la complexité de l'algorithme d'Euclide étendu, il faut établir une borne sur la taille des u_i, v_i. On se restreint au cas des polynômes.

Proposition 1 Soit $a, b \in K[X]$. On note $n_i = \deg a_i \leq \ell$ pour tout i. On a :

- $\deg v_i = n_0 - n_{i-1}, \quad 1 \leq i \leq \ell$
- $\deg u_i = n_1 - n_{i-1}, \quad 2 \leq i \leq \ell$

Preuve. Par récurrence sur i. On a $\deg v_1 = 0$ comme requis. Soit $1 < i < \ell$. On suppose que pour tout $j \in \{1, \ldots, i\}$, on a $\deg v_j = n_0 - n_{j-1}$. La suite (n_j) étant strictement décroissante, on a en particulier $\deg v_{j-1} < \deg v_j$ ce qui implique $\deg v_{j-1} < \deg(v_j q_j)$ pour $j \leq i$ puisque q_j est non nul. On en déduit que

$$\deg v_{i+1} = \deg(v_{i-1} - q_i v_i) = \deg(q_i v_i) = \deg q_i + \deg v_i = \deg q_i + n_0 - n_{i-1}$$

la dernière égalité par récurrence. Comme $a_{i-1} = q_i a_i + a_{i+1}$ (division euclidienne), on a $n_{i-1} = \deg q_i + n_i$ d'où il suit $\deg(v_{i+1}) = n_0 - n_i$, comme requis. L'égalité pour le degré de u_i se prouve de la même façon. □

Théorème 2 Soient a et b deux polynômes de $K[X]$ de degrés respectifs m, n. L'algorithme d'Euclide étendu décrit ci-dessus permet de calculer le pgcd et les coefficients de Bezout réduits de a et b avec $O(mn)$ opérations sur K (complexité arithmétique quadratique).

Preuve. Le calcul de $v_{i+1} = v_{i-1} - q_i v_i$ demande au plus $2 \deg q_i \deg v_i + \deg v_i + \deg q_i + 1 =$ opérations dans K pour le produit (cf. exercice 6 du CM 1) et au plus $\deg v_{i+1}$ opérations pour la soustraction, soit en tout $2((n_{i-1} - n_i)((n_0 - n_{i-1}) + n_0 - n_i + 1)$, sauf pour $i = 1$: le calcul de $v_2 = -q_1$ demande $n - m + 1$ changements de signe. On obtient que le nombre d'opérations dans K pour calculer les v_i est majoré par

$$\begin{aligned} C &= n - m + 1 + 2\sum_{i=2}^{\ell-1}((n_{i-1} - n_i)(n_0 - n_{i-1}) + n_0 - n_i + 1) \\ &\leq n - m + 1 + 2n\sum_{i=2}^{\ell-1}((n_{i-1} - n_i) + 2(\ell - 2)(n + 1) \\ &\leq n - m + 1 + 2n(n_1 - n_{\ell-1}) + 2(\ell - 2)(n + 1) \\ &\leq n - m + 1 + 2nm + 2m(n + 1) = O(mn). \end{aligned}$$

(le nombre d'itérations est $\ell - 1 = m + 1$). On montrerait de même que le nombre d'opérations dans K nécessaires au calcul des u_i est en $O(mn)$. Comme le calcul du pgcd demande aussi $O(mn)$ opérations sur K (cf théorème 1), le théorème est prouvé. □

Théorème 3 (Cas des entiers) L'algorithme d'Euclide étendu décrit ci-dessus permet de calculer le pgcd et les coefficients de Bezout réduits de deux entiers $a, b \in \mathbb{Z}$ de tailles $n = \log(a)$ et $m = \log(b)$ avec une complexité binaire en $O(mn)$.

2.3 Algorithmes rapides.

Comme pour la multiplication et la division, il existe des algorithmes rapides de calcul du pgcd et des coefficients de Bezout :

Théorème 4 On peut calculer le pgcd et les coefficients de Bezout de deux polynômes ou deux entiers de tailles au plus n avec une complexité $O(\mathrm{M}(n)\log(n))$.

Ces algorithmes sont donc quasi-linéaire $\tilde{O}(n)$ en la taille de l'entrée en utilisant la multiplication rapide. Ils sont basés sur *l'algorithme du demi-pgcd* (technique "pas de géant, pas de bébé"). Voir par exemple ([BCGLLSS], Section 6.3). Ils sont très importants en calcul formel, en particulier du fait de leur utilisation pour l'inversion modulaire.

3 Quelques applications de l'algorithme d'Euclide

3.1 Inversion modulaire

L'algorithme d'Euclide étendu permet de calculer l'inverse d'un élément inversible dans un anneau quotient $A/(a)$ d'un anneau euclidien. En effet, $x \mod a \in A/(a)$ est inversible si et seulement si x est premier avec a (exo). Il existe dans ce cas une relation de Bezout $ux + va = 1$ dans A et l'inverse de $x \mod a$ est simplement $u \mod a$. D'où le résultat de complexité suivant :

Théorème 5 (Inversion modulaire des polynômes) Soit $P \in K[X]$ de degré n.

- L'algorithme d'Euclide étendu permet de calculer l'inverse d'un élément (inversible !) de $K[X]/(P)$ avec $O(n^2)$ opérations arithmétiques dans K.
- L'algorithme du "demi-pgcd" permet de calculer l'inverse d'un élément (inversible !) de $K[X]/(P)$ avec $O(\mathrm{M}(n)\log(n))$ opérations arithmétiques dans K, soit $\tilde{O}(n)$ avec la multiplication rapide.

Remarque 5 De manière analogue, l'algorithme d'Euclide étendu permet de calculer l'inverse d'un inversible de $\mathbb{Z}/N\mathbb{Z}$ en $O(\log(N)^2)$ opérations binaires, et il existe des variantes rapides de complexité $O(\log\log(N)\,\mathrm{M}(\log(N)))$, soit $\tilde{O}(\log(N))$ avec la multiplication rapide.

Exercice 11 Prouver le théorème.

Exercice 12 Soit ϕ le nombre d'or. Simplifier la fraction $R(\phi) = \frac{\phi^4-\phi+1}{\phi^7-1}$ en détaillant les calculs.

Exercice 13 Soit p un nombre premier et soit a un entier non multiple de p. Donner un algorithme pour le calcul de l'inverse de $a \mod p$ utilisant le petit théorème de Fermat et l'exponentiation binaire. Analyser la complexité binaire de cet algorithme et comparer avec l'approche par Euclide étendu.

3.2 Restes chinois, évaluation, interpolation

Soit A un anneau principal et soient $a_1, \ldots, a_k$ et $b_1, \ldots, b_k$ des éléments de A. Le problème des restes chinois (initialement posé pour $A = \mathbb{Z}$) consiste à résoudre un système d'équations modulaires du type :

$$\begin{aligned} x &\equiv b_1 \mod a_1 \\ x &\equiv b_2 \mod a_2 \\ &\vdots \\ x &\equiv b_k \mod a_k \end{aligned}$$

C'est donc une généralisation de l'inversion modulaire qui consiste à résoudre $x \equiv 1 \mod a$.

Théorème 6 (restes chinois) Soit A un anneau principal et $a_1, \ldots, a_k \in A$ *deux à deux premiers entre eux*. Soit $a = a_1 \cdots a_k$. Alors le morphisme d'anneaux

$$\begin{array}{rccc} f: & A/aA & \longrightarrow & A/a_1A \times \cdots \times A/a_kA \\ & x \bmod a & \longmapsto & (x \bmod a_1, \cdots, x \bmod a_k) \end{array}$$

est un isomorphisme.

Preuve. Il est clair que f est un morphisme d'anneaux. Montrons qu'il est surjectif. Pour chaque i, les éléments a_i et $\hat{a}_i = \frac{a}{a_i}$ sont par hypothèse premiers entre eux. Il existe donc un inverse v_i de $\hat{a}_i$ modulo a_i, que l'on obtient via un pgcd étendu $u_ia_i + v_i\hat{a}_i = 1$ (cf paragraphe précédent). Si on pose $e_i = v_i\hat{a}_i$, on obtient

$$e_i \equiv 1 \mod a_i \quad \text{et} \quad e_i \equiv 0 \mod a_j,\ j \neq i.$$

Une solution particulière du système de congruences $x \equiv b_i \mod a_i$ pour tout i est par conséquent $x = \sum_{i=1}^{k} b_ie_i$, d'où la surjectivité. Si $f(x \mod a_i) = 0$ pour tout i, alors a_i divise x pour tout i. Comme les a_i sont premiers deux à deux, il suit du lemme de Gauss que a divise x, soit $x \equiv 0 \mod a$ ce qui prouve l'injectivité. □

Corollaire 1 Sous l'hypothèse que les a_i sont deux à deux premiers entre eux, alors il existe une solution $x \in A$ aux congruences $x \equiv b_i \mod a_i$, et cette solution est unique modulo $a = a_1 \cdots a_k$.

- Le calcul des $x \mod a_i$ pour tout i s'appelle le problème des *modules multiples*.
- Le problème inverse de calculer $x \in A$ tel que $x \mod a_i \equiv b_i \mod a_i$ pour tout i s'appelle le problème des *restes chinois*.

On a le résultat de complexité suivant:

Théorème 7 Soient $a_1, \ldots, a_k \in \mathbb{Z}$ des entiers positifs *deux à deux premiers entre eux*. On note n la taille binaire de $a = a_1 \cdots a_k$.

- Si x est de taille au plus n, le problème des *modules multiples* pour x peut se résoudre en $O(n^2)$ opérations binaires.
- Si $b_1, \ldots, b_k \in \mathbb{Z}$ vérifient $|b_i| < a_i$, le problème des *restes chinois* peut se résoudre en $O(n^2)$ opérations binaires.

Remarque 6 Bien entendu, ce résultat de complexité admet son analogue dans le cas des polynômes $A = K[X]$.

Remarque 7 Il existe ici encore des algorithmes rapides de complexité quasi-linéaire $\tilde{O}(n)$ en la taille de l'entrée, basés sur le principe "diviser pour régner". Avez-vous une idée de comment procéder ? On verra cela en TP.

Exercice 14 Prouver le théorème 7.

Exercice 15 Résoudre le système de congruences

$$\begin{aligned} x &\equiv 3 \mod 7 \\ x &\equiv 5 \mod 4 \end{aligned}$$

Exercice 16 Je possède un sachet de billes. Si je les dispose en tas de 2 billes, il en reste une toute seule, en tas de 3 billes, il n'en reste pas, en tas de 5 billes, il en reste deux. Combien ai-je de billes?

Exercice 17 (Evaluation et interpolation polynomiale) Soit K un corps et soient $x_1, \ldots, x_n \in K$ *deux à deux distincts*. Le problème d'évaluation multipoints consiste à calculer les valeurs $P(x_1), \ldots, P(x_n)$ d'un polynôme donné P de degré $< n$. Le problème d'interpolation consiste à calculer P de degré $< n$ tel que $P(x_1) = y_1, \ldots, P(x_n) = y_n$.

1. Montrer que ces deux problèmes sont des instances particulières du calcul des modules et des restes chinois.
2. Le théorème de Lagrange assure que le problème d'interpolation admet une unique solution. Redémontrer ce théorème à l'aide du théorème 6 et montrer que la preuve du théorème 6 conduit finalement à la formule d'interpolation de Lagrange.
3. Ecrire un algorithme basé sur la formule d'interpolation de Lagrange qui calcule lui aussi P avec $O(n^2)$ opérations dans K.
4. Déduire de la remarque 7 que l'on peut résoudre les problèmes d'évaluation et d'interpolation avec $O(\log(n)\,\mathrm{M}_K(n))$ opérations dans K. C'est en se basant sur ce résultat fondamental que l'on peut obtenir une multiplication dans $K[X]$ de complexité quasi-linéaire.