Arithmétique Algorithmique

http://www.math.univ-lyon1.fr/~roblot/ens.html

Partie III

Algorithmes classiques

Coût de la multiplication et de la division

Convention. On considère dans cette partie que les algorithmes utilisés pour la division et la multiplication sont les algorithmes classiques et donc la multiplication entre deux entiers a et b ou la division de a avec b comme quotient coûte

$$O(\log a \log b) \subset O((\log N)^2)$$

opérations binaires avec $|a|, |b| \leq N$.

Entiers de taille décroissante. Soient $a_1, \dots a_k$ et $b_1, \dots, b_k$ des entiers (disons positifs) tels que $a_1, \dots, a_k \leq N$ et $\prod_i b_i \leq N$. Alors le coût pour faire toutes les multiplications des a_i par les b_i, ou toutes les divisions des a_i avec les b_i pour quotients, est

$$\begin{aligned} O(\log a_1 \log b_1) + \cdots + O(\log a_k \log b_k) &\subset O((\log N)(\log b_1 + \cdots \log b_k)) \\ &= O((\log N)(\log b_1 \cdots b_k)) \subset O((\log N)^2). \end{aligned}$$

Exponentiation rapide

Problème. Pour G groupe, $g \in G$ et $n \in \mathbb{Z}$, on veut calculer g^n.

Remarques. Méthode naïve donne $O(n)$ opérations dans G.
Quitte à remplacer g par g^{-1}, on peut supposer $n \geq 0$.

Écriture binaire. On écrit $n = \sum_i \epsilon_i 2^i$ avec $\epsilon_i = 0$ ou 1, on a alors

$$g^n = \prod_{\epsilon_i = 1} g^{2^i}.$$

Au plus $O(\log n)$ multiplications dans G et $g^{2^{i+1}} = \left(g^{2^i}\right)^2$.

Algorithme.

(1) Poser $h \leftarrow 1_G$, $t \leftarrow g$

(2) Si $n = 0$ alors retourner h

(3) Si n est impair alors poser $h \leftarrow h \cdot t$

(4) Faire $n \leftarrow \lfloor n/2 \rfloor$, $t \leftarrow t^2$ et retourner en (2)

Exponentiation rapide : algorithme gauche-droite

Bits décroissants. On a la formule

$$g^n = \begin{cases} (g^{n/2})^2 & \text{si } n \text{ est pair,} \\ g \cdot (g^{(n-1)/2})^2 & \text{si } n \text{ est impair.} \end{cases}$$

Algorithme.

(1) Si $n = 0$ alors retourner 1_G

(2) Poser $e \leftarrow \lfloor \log_2(n) \rfloor$, $h \leftarrow g$, $t \leftarrow g$

(3) Tant que $e > 0$, faire
    poser $e \leftarrow e - 1$, $h \leftarrow h^2$
    Si bit$(n, e) = 1$ alors faire $h \leftarrow h \cdot t$

(4) Retourner h

Entiers longs. L'opération $h \leftarrow h \cdot t$ est toujours effectuée avec $t = g$, ce qui coûte beaucoup moins si g est petit.

Exponentiation rapide

Exemple. Calcul de $5^{77} \pmod{37529}$. Note : $77 = (1001101)_2$.

h	t	n	M	e	h	t	M
1	5	77	$*$	6	5	5	
5	25	38		5	25	5	
5	625	19	$*$	4	625	5	
3 125	15 335	9	$*$	3	1 617	5	$*$
34 871	5 511	4		2	13 353	5	$*$
34 871	10 160	2		1	2 330	5	
34 871	20 850	1	$*$	0	11 033	5	$*$
11 033	24 093	0					

(1) Poser $h \leftarrow 1_G$, $t \leftarrow g$
(2) Si $n = 0$ alors retourner h
(3) Si n est impair alors poser $h \leftarrow h \cdot t$
(4) Faire $n \leftarrow \lfloor n/2 \rfloor$, $t \leftarrow t^2$ et retourner en (2)

(1) Si $n = 0$ alors retourner 1_G
(2) Poser $e \leftarrow \lfloor \log_2(n) \rfloor$, $h \leftarrow g$, $t \leftarrow g$
(3) Tant que $e > 0$, faire
    poser $e \leftarrow e - 1$, $h \leftarrow h^2$
    Si bit$(n, e) = 1$ alors faire $h \leftarrow h \cdot t$
(4) Retourner h

Algorithme d'Euclide

Résultat. Soient a, b deux entiers positifs avec $a > b$, on a

$$\text{PGCD}(a, b) = \text{PGCD}(b, a \bmod b).$$

Algorithme.

(1) Si $b = 0$, alors retourner a et terminer.

(2) Faire $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$ et retourner en (1).

Complexité. Si $a, b \leq N$, il y a $O(\log N)$ divisions euclidiennes dans cet algorithme (cf. écran suivant). Donc on a une complexité de $O((\log N)^3)$, et même $O((\log N)^2)$ en prenant en compte le fait que le produit des quotients dans les divisions euclidiennes successives est plus petit que a.

Algorithme d'Euclide

Complexité de l'algorithme d'Euclide

Le nombre de divisions euclidiennes dans l'algorithme d'Euclide est plus petit que $2\log_2 N$.

Preuve. On pose

$$a_i = q_i b_i + r_i \quad i = 0, 1, \ldots, t$$

les différentes divisions de l'algorithme avec $a_0 = a$, $b_0 = b$ et $a_{i+1} = b_i$, $b_{i+1} = r_i$.
Si $r_1 \leq r_0/2$, alors $r_2 < b_2 = r_1 \leq r_0/2$. Sinon $2r_1 > r_0$ et la division de $a_2 (= b_1 = r_0)$ par $b_2 (= r_1)$ s'écrit $a_2 = b_2 + r_2$ et $r_2 = r_0 - r_1 < r_0/2$.
Ainsi, dans tous les cas, on trouve $r_2 < r_0/2$ et, de même, pour tout $i \geq 0$, on a $r_{i+2} < r_i/2$. D'où, on a

$$r_{2k} < r_0/2^k = b/2^k \leq N/2^k$$

et donc dès que $k \geq \log_2 N$, on a $r_{2k} = 0$. □

Algorithme d'Euclide : version binaire

Remarques.

- Après la première division, les nombres sont sensiblement de même taille (et donc les quotients sont relativement petits)
- On a
$$\mathrm{PGCD}(2a, 2b) = 2\,\mathrm{PGCD}(a, b)$$
- Si b est impair, on a
$$\mathrm{PGCD}(2a, b) = \mathrm{PGCD}(a, b)$$
- Si a et b sont impairs avec $a \geq b$,
$$\mathrm{PGCD}(a, b) = \mathrm{PGCD}(a, (a - b)/2).$$
- Diviser par 2 ne coûte rien et une soustraction coûte beaucoup moins qu'une division.

Algorithme d'Euclide : version binaire

Algorithme.

(1) Si $a < b$, alors échanger $a \leftrightarrow b$.

(2) Si $b = 0$ alors retourner a et terminer.
Sinon, faire $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.
Si $b = 0$ alors retourner a et terminer.

(3) Poser $k \leftarrow 0$ et tant que a et b sont tous deux pairs, faire
$k \leftarrow k + 1$, $a \leftarrow a/2$, $b \leftarrow b/2$

(4) Si a est pair, alors tant que a est pair, faire $a \leftarrow a/2$.
Sinon tant que b est pair, faire $b \leftarrow b/2$,

(5) Faire $t \leftarrow (a - b)/2$. Si $t = 0$, retourner $2^k a$ et terminer.

(6) Tant que t est pair, faire $t \leftarrow t/2$.
Si $t > 0$, alors faire $a \leftarrow t$, sinon faire $b \leftarrow -t$.
Retourner en (5).

Complexité. $O((\log N)^2)$ opérations binaires.

Algorithme d'Euclide : exemple pour l'algorithme binaire

Données. On cherche le PGCD de $a = 1\,422\,332\,948$ et $b = 72\,182\,136$.
On a $r = 50\,872\,364$, puis pour les étapes (3) et (4)

a	b	k
72 182 136	50 872 364	0
36 091 068	25 436 182	1
18 045 534	12 718 091	2
9 022 767	12 718 091	2

Finalement, pour les étapes (5) et (6)

a	b	$t = (a-b)/2$		$t/2^{v_2(t)}$
9 022 767	12 718 091	−1 847 662	$\longrightarrow$	−923 831
9 022 767	923 831	4 049 468	$\longrightarrow$	1 012 367
1 012367	923 831	44 268	$\longrightarrow$	11 067
11 067	923 831	−456 382	$\longrightarrow$	−228 191
11 067	228 191	−108 562	$\longrightarrow$	−54 281
11 067	54 281	−21 607	$\longrightarrow$	−21 607
11 067	21 607	−5 270	$\longrightarrow$	−2 635
11 067	2 635	4 216	$\longrightarrow$	527
527	2 635	−1 054	$\longrightarrow$	−527
527	527	0		

donc le PGCD est $2^2 \cdot 527$.

Algorithme d'Euclide étendu

Résultat. Posons $d = \mathrm{PGCD}(a, b)$. Alors, il existe deux entiers u et v tels que

$$au + bv = d.$$

Algorithme.

(1) Poser $u \leftarrow 1$, $d \leftarrow a$.
Si $b = 0$, alors poser $v \leftarrow 0$, retourner (u, v, d) et terminer.
Sinon, poser $u_1 \leftarrow 0$ et $d_1 \leftarrow b$.

(2) Si $d_1 = 0$, alors faire $v \leftarrow (d - au)/b$, retourner (u, v, d) et terminer.

(3) Faire $d = qd_1 + d_2$, la division euclidienne de d par d_1.
Faire $u_2 \leftarrow u - qu_1$, $(u, d) \leftarrow (u_1, d_1)$, $(u_1, d_1) \leftarrow (u_2, d_2)$ et retourner en (2).

Validité. Au début de l'étape (2), on a toujours

$$au_2 + bv_2 = d_2, \quad au + bv = d, \quad au_1 + bv_1 = d_1$$

pour certains entiers v_1 et v_2 (avec $u_2 = d_2 = 0$ la première fois).

Algorithme d'Euclide étendu : exemple

Données. $a = 9\,768\,932$ et $b = 2\,163\,072$

d_2	q	u_2	d	u	u_1	d_1
$d \bmod d_1$	$\lfloor d/d_1 \rfloor$	$u - qu_1$	d_1	u_1	u_2	d_2
$-$	$-$	$-$	9 768 932	1	0	2 163 072
1 116 644	4	1	2 163 072	0	1	1 116 644
1 046 428	1	-1	1 116 644	1	-1	1 046 428
70 216	1	2	1 046 428	-1	2	70 216
63 404	14	-29	70 216	2	-29	63 404
6 812	1	31	63 404	-29	31	6 812
2 096	9	-308	6 812	31	-308	2 096
524	3	955	2 096	-308	955	524
0	4	$-4\,128$	524	955	$-4\,128$	0

Donc le PGCD est 524 et $v = \dfrac{d - au}{b} = \dfrac{524 - 9\,768\,932 \cdot 955}{2\,163\,072} = -4\,313$

Algorithme d'Euclide étendu : remarques

Complexité. A priori $O((\log N)^3$ opérations binaires, mais en fait $O((\log N)^2)$ opérations par un raisonnement similaire à celui fait pour l'algorithme d'Euclide classique.

Version binaire. Une version binaire est possible en modifiant l'algorithme donné ci-dessus.

Inversion modulaire. L'algorithme d'Euclide étendu est très souvent utilisé pour calculer $a^{-1} \mod N$ avec a et N premiers entre eux. Une autre possibilité, quand la factorisation de N est connue, est d'utiliser le théorème d'Euler

$$a^{\varphi(N)} \equiv 1 \pmod{N} \Rightarrow a^{-1} \equiv a^{\varphi(N)-1} \pmod{N}$$

mais le calcul de l'exponentiation se fait en $O((\log N)^3)$ opérations binaires.

Reconstruction d'un nombre rationnel

Problème. Soit $q \in \mathbb{Q}$ donné par une approximation dans $\mathbb{R}$, retrouver q.
Ou encore, soit $r \in \mathbb{R}$, donné à une certaine précision, trouver $a, b \in \mathbb{Z}$ tels que $|r - a/b|$ est petit.

Solution triviale. Tout nombre réel donné à une certaine précision est un nombre rationnel.
Par exemple, pour π à la précision 10^{-10}, on a

$$3,1415926536 = \frac{31\,415\,926\,536}{10\,000\,000\,000} = \frac{3\,926\,990\,817}{1\,250\,000\,000}.$$

Reformulation du problème. Soit $r \in \mathbb{R}$, donné à une certaine précision, trouver $a, b \in \mathbb{Z}$ tels que $|r - a/b|$ est significativement plus petit que $1/b$.
Par exemple, on a

$$\left|\pi - \frac{312\,689}{99\,532}\right| < 10^{-10}.$$

Reconstruction d'un nombre rationnel

Développement périodique. Soit $q \in \mathbb{Q}$, le développement décimal de q est ultimement périodique. Donc si $r \in \mathbb{R}$ a un (début de) développement (ultimement) périodique, on peut s'en servir pour trouver a/b proche de r.
Par exemple, pour $r = 2,3\,4587\,4587\,4587$. On pose

$$x = 0,4587\,4587\,4587\,4587\,4587\ldots$$

d'où on trouve

$$10\,000 \cdot x = 4587,4587\,4587\,4587\,4587\cdots = 4\,587 + x$$

donc

$$x = \frac{4\,587}{10\,000 - 1} = \frac{139}{303}.$$

Ainsi, finalement

$$r \approx \frac{23}{10} + \frac{x}{10} = \frac{3\,554}{1\,515} \quad (\text{à } 10^{-14} \text{ près}).$$

Inconvénient. Même pour b petit, la période de a/b peut être très grande (par exemple, 6 pour $b = 13$).

Reconstruction d'un nombre rationnel : fractions continues

Notation. Soient $a_0, a_1, a_2, \ldots$ une suite finie ou infinie de nombres entiers ≥ 1, on note

$$[a_0, a_1, a_2, \ldots] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots}}$$

Application de Gauss. Pour $x \in \mathbb{R}_{>0}$, on pose

$$T(x) = \left\{ \frac{1}{x} \right\}.$$

On pose $a_0 = \lfloor x \rfloor$, et pour $n \geq 0$ et tant que $T^n(x) \neq 0$, on définit

$$a_{n+1} = \lfloor T^n(x - a_0)^{-1} \rfloor.$$

Alors, la suite (a_n) est finie si et seulement si x est rationnel et, dans tous les cas, on a

$$[a_0, a_1, a_2, \ldots] = x.$$

Reconstruction d'un nombre rationnel : fractions continues

Réduites. Soit $x \in \mathbb{R}_{>0}$ de développement en fraction continue $[a_0, a_1, a_2, \dots]$ de longueur $l \in \mathbb{N} \cup \{\infty\}$. Pour $n \in \mathbb{N}$ avec $0 \leq n \leq l$, on note p_n, q_n les deux nombres entiers positifs et premiers entre eux tels que

$$\frac{p_n}{q_n} = [a_0, a_1, \dots, a_n] \quad (n\text{-iéme réduite}).$$

Meilleures approximations

Pour tout $n \geq 0$, on a $\quad |x - p_n/q_n| \leq 1/q_n^2$.

Exemple. On a $\pi = [3, 7, 15, 1, 292, 1, 1, 2, \dots]$ et les réduites successives

$$3\ (\epsilon \approx 0,1),\ \frac{22}{7}\ (\epsilon \approx 0,001),\ \frac{333}{106}\ (\epsilon \approx 8 \cdot 10^{-5}),\ \frac{355}{113}\ (\epsilon \approx 2 \cdot 10^{-7}),\ \frac{103\,993}{33\,102}\ (\epsilon \approx 5 \cdot 10^{-10}),$$
$$\frac{104\,348}{33\,215}\ (\epsilon \approx 3 \cdot 10^{-10}),\ \frac{208\,341}{66\,317}\ (\epsilon \approx 1 \cdot 10^{-12}),\ \frac{521\,030}{165\,849}\ (\epsilon \approx 3 \cdot 10^{-13}) \dots$$

Reconstruction d'un nombre rationnel : exemple d'utilisation des fractions continues

On prend $x = 0,126141123456902839532097536827$

On a $a_0 = 0$, puis on a

n	$T^{n-1}(x)$	a_n	p_n/q_n	ϵ
1	$0,126141123456902839532097536827$	7	$1/7$	$0,0167$
2	$0,927628774779846395110378382725$	1	$1/8$	$0,0011$
3	$0,078017443170981221715534132089$	12	$13/103$	$7 \cdot 10^{-5}$
4	$0,817646405156128458717635292810$	1	$14/111$	10^{-6}
5	$0,223022560478390867083446588747$	4	$69/547$	10^{-7}
6	$0,483851310176721511273613650213$	2	$152/1\,205$	$4 \cdot 10^{-8}$
7	$0,066750629722921914357682619647$	14	$2\,197/17\,417$	$8 \cdot 10^{-9}$
8	$0,981132075471698113207547169811$	1	$2\,349/18\,622$	$5 \cdot 10^{-11}$
9	$0,019230769230769230769230769230$	52	$124\,345/985\,761$	$< 10^{-30}$

et donc $x = 124\,345/985\,761$.

Carrés dans $\mathbb{F}_p$: symbole de Legendre-Jacobi

Rappels. Pour p premier, l'anneau $\mathbb{Z}/p\mathbb{Z}$ est un corps (fini) dénoté $\mathbb{F}_p$.

Le groupe multiplicatif $\mathbb{F}_p^\times$ est cyclique d'ordre $p-1$ et donc a exactement $(p-1)/2$ carrés (p impair) : si g est un générateur alors g^i ($0 \leq i < p-1$) est un carré si et seulement si i est pair.

Pour tout $a \in \mathbb{F}_p^\times$, on a $a^{p-1} = 1$ par le petit théorème de Fermat, et donc a est un carré si et seulement si $a^{(p-1)/2} = 1$ (sinon c'est -1).

Définition. Pour $a \in \mathbb{Z}$, on définit le symbole de Jacobi en p par

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{si } p \mid a, \\ 1 & \text{si } p \nmid a \text{ et } a \text{ est un carré modulo } p, \\ -1 & \text{si } p \nmid a \text{ et } a \text{ n'est pas un carré modulo } p. \end{cases}$$

Note. Par la remarque précédente, on a $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p$ et donc le symbole $\left(\frac{\cdot}{p}\right)$ est multiplicatif.

Carrés dans $\mathbb{F}_p$: symbole de Legendre-Jacobi

Cas particuliers. Pour p premier impair, on a

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = \begin{cases} 1 & \text{si } p \equiv 1 \pmod 4, \\ -1 & \text{si } p \equiv 3 \pmod 4. \end{cases}$$

$$\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8} = \begin{cases} 1 & \text{si } p \equiv 1, 7 \pmod 8, \\ -1 & \text{si } p \equiv 3, 5 \pmod 8. \end{cases}$$

Loi de réciprocité quadratique

Soient p et q deux nombres premiers impairs, on a

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}} \left(\frac{q}{p}\right).$$

Carrés dans $\mathbb{F}_p$: symbole de Legendre-Jacobi

Premier calcul. On peut calculer

$$\left(\frac{a}{b}\right) = a^{(p-1)/2} \pmod{p}$$

en $O((\log p)^3)$ opérations (multiplication classique).

Itérations. On peut aussi utiliser la loi de réciprocité quadratique

$$\left(\frac{113}{347}\right) = \left(\frac{347}{113}\right) = \left(\frac{8}{113}\right) = \left(\frac{2}{113}\right) = 1$$

Mais, pour calculer $\left(\frac{a}{p}\right)$, il faut d'abord factoriser a ce qui est très coûteux en général.

Carrés dans $\mathbb{F}_p$: symbole de Legendre-Jacobi

Généralisation. Soient $a, b \in \mathbb{Z}$ avec $b > 0$ et impair. On écrit la factorisation $b = \prod_i p_i^{e_i}$ et on pose

$$\left(\frac{a}{b}\right) = \prod_i \left(\frac{a}{p_i}\right)^{e_i}$$

Attention. Symbole égal à 1 n'implique pas que a est un carré modulo b.

Propriétés. Le symbole est périodique en a de période b.
Il est égal au symbole de Legendre quand b est premier (impair).
Il vérifie les mêmes règles pour le calcul en $a = -1$ et $a = 2$, et la généralisation de la loi de réciprocité quadratique

$$\left(\frac{a}{b}\right) = (-1)^{\frac{a-1}{2}\frac{b-1}{2}} \left(\frac{b}{a}\right) \quad \text{pour } a,\ b \text{ impairs positifs.}$$

Remarques. On peut généraliser avec le symbole de Kronecker défini pour tout $b \in \mathbb{Z}$ sans restriction.

Carrés dans $\mathbb{F}_p$: symbole de Legendre-Jacobi

Données. $a, b \in \mathbb{Z}$ avec b impair et positif.

Algorithme.

(1) Faire $a \leftarrow a \bmod b$.

(2) Si $a = 0$, alors retourner 0 et terminer.

(3) Si a est pair, alors faire
Faire $r \leftarrow b \bmod 8$
Si $r = 3$ ou $r = 5$, alors renvoyer $-\text{jacobi}(a/2, b)$ et terminer
Sinon, renvoyer $\text{jacobi}(a/2, b)$ et terminer

(4) Si $a = 1$, alors retourner 1 et terminer

(5) Retourner $(-1)^{(a-1)(b-1)/4}\text{jacobi}(b, a)$ et terminer.

Remarque. Le signe $(-1)^{(a-1)(b-1)/4}$ est calculé par des congruences !

Complexité. Similaire à l'algorithme d'Euclide et donc $O((\log N)^2)$ opérations binaires où $|a|, b \leq N$.

Carrés dans $\mathbb{F}_p$: symbole de Legendre-Jacobi, exemple

Données. Calcul du symbole de Legendre de 20 406 967 en 76 545 467.

$$\left(\frac{20\,406\,967}{76\,545\,467}\right) = \left(\frac{76\,545\,467}{20\,406\,967}\right) = \left(\frac{15\,324\,566}{20\,406\,967}\right) = \left(\frac{2}{20\,406\,967}\right)\left(\frac{7\,662\,283}{20\,406\,967}\right) = -\left(\frac{20\,406\,967}{7\,662\,283}\right)$$

$$= -\left(\frac{5\,082\,401}{7\,662\,283}\right) = -\left(\frac{7\,662\,283}{5\,082\,401}\right) = -\left(\frac{2\,579\,882}{5\,082\,401}\right) = -\left(\frac{2}{5\,082\,401}\right)\left(\frac{1\,289\,941}{5\,082\,401}\right)$$

$$= -\left(\frac{5\,082\,401}{1\,289\,941}\right) = -\left(\frac{1\,212\,578}{1\,289\,941}\right) = -\left(\frac{2}{1\,289\,941}\right)\left(\frac{1\,289\,941}{606\,289}\right) = -\left(\frac{77\,363}{606\,289}\right) = -\left(\frac{606\,289}{77\,363}\right)$$

$$= -\left(\frac{64\,748}{77\,363}\right) = -\left(\frac{4}{77\,363}\right)\left(\frac{16\,187}{77\,363}\right) = \left(\frac{77\,363}{16\,187}\right) = \left(\frac{12\,615}{16\,187}\right) = -\left(\frac{16\,187}{12\,615}\right) = -\left(\frac{3\,572}{12\,615}\right)$$

$$= -\left(\frac{4}{12\,615}\right)\left(\frac{893}{12\,615}\right) = -\left(\frac{12\,615}{893}\right) = -\left(\frac{113}{893}\right) = -\left(\frac{893}{113}\right) = -\left(\frac{102}{113}\right)$$

$$= -\left(\frac{2}{113}\right)\left(\frac{51}{113}\right) = -\left(\frac{113}{51}\right) = -\left(\frac{11}{51}\right) = \left(\frac{51}{11}\right) = \left(\frac{7}{11}\right) = -\left(\frac{11}{7}\right) = -\left(\frac{4}{11}\right) = -1$$

Conclusion. 20 406 967 n'est pas un carré modulo 76 545 467.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées

Problème. Soient a entier et p premier impair avec $\left(\frac{a}{p}\right) = 1$, on veut trouver r tel que $r^2 \equiv a \pmod{p}$.

Premier cas facile. Si $p \equiv 3 \pmod{4}$, on prend $r = a^{(p+1)}/4 \mod p$. En effet

$$r^2 \equiv a^{(p+1)/2} \equiv a \cdot a^{(p-1)/2} \equiv a \pmod{p}$$

puisque $a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \equiv 1 \pmod{p}$.

Deuxième cas (un peu moins) facile. Si $p \equiv 5 \pmod{8}$, on considère

$$a^{(p-1)/4} \equiv \pm 1 \pmod{p}.$$

- Si c'est $+1$, on prend $r = a^{(p+3)/8} \mod p$,
- Sinon, on prend $r = (2a)(4a)^{(p-5)/8} \mod p$ en utilisant le fait que $2^{(p-1)/2} \equiv -1 \pmod{p}$ puisque 2 n'est pas un carré modulo p.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Tonelli-Shanks

Méthode. Pour le cas général, on considère

$$(\mathbb{Z}/p\mathbb{Z})^\times \simeq \underbrace{(\mathbb{Z}/2^e\mathbb{Z})}_{G} \times \underbrace{(\mathbb{Z}/q\mathbb{Z})}_{H}$$

avec $p-1 = 2^e \cdot q$ et q impair. Si a est un carré modulo p, on a

$$a^{(p-1)/2} \equiv (a^q)^{2^{e-1}} \equiv 1 \pmod{p}$$

et $b = a^q$ est un carré dans G.
Notons g un générateur de G, il existe k, pair, tel que $bg^k = 1$ et si on pose

$$r = a^{(q+1)/2} g^{k/2}$$

alors r est une racine carrée de a modulo p.

Deux problèmes. Trouver g et calculer k.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Tonelli-Shanks

Trouver g. Pour tout entier n premier avec p, on a $n^q \in G$. De plus, G a $\varphi(2^e) = 2^{e-1}$ générateurs. Donc on peut prendre n au hasard et on a une chance sur deux que n^q est un générateur. En fait c'est le cas si et seulement si n n'est pas un carré modulo p. Cette partie de l'algorithme est de complexité **probabiliste**.
La probabilité de n'avoir toujours pas de générateurs après 20 tests est < 0.000001.

Calculer k. On procède de manière indirecte. On considère les couples (e_1, e_2) tels que

$$a^{e_1}\, g^{e_2} \equiv 1 \pmod{p}.$$

On peut commencer avec $e_1 = (p-1)/2$ et $e_2 = 0$. On fait des transformations (cf. prochain écran) jusqu'à obtenir la relation avec e_1 impair. Alors

$$r = a^{(e_1+1)/2} g^{e_2/2}$$

est une racine carrée de a modulo p.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Tonelli-Shanks

Transformations sur (e_1, e_2). On considère les couples (e_1, e_2) tels que

$$a^{e_1} g^{e_2} \equiv 1 \pmod{p}$$

en commençant avec $(e_1, e_2) \leftarrow ((p-1)/2, 0)$.

Puisque g n'est pas un carré modulo p, alors e_2 est toujours pair.

On suppose donc e_1 pair et on considère

$$a^{e_1/2} g^{e_2/2} \equiv \pm 1 \pmod{p}.$$

- Si c'est $+1$, on peut faire $(e_1, e_2) \leftarrow (e_1/2, e_2/2)$.
- Si c'est -1, on utilise le fait que $g^{(p-1)/2} \equiv -1 \pmod{p}$ et on fait $(e_1, e_2) \leftarrow (e_1/2, (e_2 + p - 1)/2)$.

En recommençant au plus $\log_2(p-1)$ fois, on obtient e_1 impair et donc une racine carrée de a modulo p.

Remarque. En fait, on a juste besoin de g non carré modulo p.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Tonelli-Shanks

Algorithme.

(1) Prendre g au hasard jusqu'à ce que $\left(\frac{g}{p}\right) = -1$.

(2) Poser $(e_1, e_2) \leftarrow ((p-1)/2, 0)$.

(3) Tant que e_1 est pair, faire

$(e_1, e_2) \leftarrow (e_1/2, e_2/2)$

Si $a^{e_1} g^{e_2} \equiv -1 \pmod{p}$, alors faire $e_2 \leftarrow e_2 + (p-1)/2$

(4) Retourner $a^{(e_1+1)/2} g^{e_2/2}$ et terminer.

Complexité. Le coût de (1) est (probablement) $O((\log p)^2)$ et la boucle (3) est effectuée $O(\log p)$ fois, ce qui donne une complexité probable de $O((\log p)^4)$ opérations binaires. En fait, la boucle (3) est effectuée en général beaucoup moins que $(\log p)$ fois car $p-1$ est rarement divisible par une grande puissance de 2, et donc la complexité (probable) en moyenne est $O((\log p)^3)$.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Tonelli-Shanks, un exemple

Données. On cherche une racine carrée de $a = 9\,964$ modulo $p = 57\,649$. On choisit au hasard : $g = 41\,570$ et $\left(\frac{g}{p}\right) = 1$, $g = 27\,565$ et $\left(\frac{g}{p}\right) = -1$. Donc $g = 27\,565$. On pose $(e_1, e_2) \leftarrow (28\,824, 0)$.

On calcule $(e_1, e_2) \leftarrow (e_1/2, e_2/2) = (14\,412, 0)$ puis

$$a^{e_1}\, g^{e_2} \equiv 1 \pmod{p}.$$

Puis, on pose $(e_1, e_2) \leftarrow (e_1/2, e_2/2) = (7\,206, 0)$ puis

$$a^{e_1}\, g^{e_2} \equiv 1 \pmod{p}.$$

Puis, on pose $(e_1, e_2) \leftarrow (e_1/2, e_2/2) = (3\,603, 0)$ puis

$$a^{e_1}\, g^{e_2} \equiv -1 \pmod{p},$$

d'où on pose $e_2 \leftarrow e_2 + (p-1)/2 = 28\,824$.

Conclusion.

$$a^{(3\,603+1)/2}\, g^{28\,824/2} = 6\,745.$$

est une racine carrée de a modulo p.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Cipolla

Arithmétique de $\mathbb{F}_{p^2}$. Soit n un entier tel que $n^2 - a$ n'est pas un carré modulo p. Alors, le polynôme

$$X^2 - 2nX + a$$

est irréductible sur $\mathbb{F}_p[X]$. On note ω une racine de ce polynôme dans $\bar{\mathbb{F}}_p$ alors $\mathbb{F}_p(\omega) = \mathbb{F}_{p^2}$ et

$$N_{\mathbb{F}_{p^2}/\mathbb{F}_p}(\omega) = \omega \cdot \omega^p = a.$$

Et donc $\omega^{(p+1)/2} \in \mathbb{F}_p$ est une racine carrée de a modulo p.

Remarques. La probabilité de trouver un n convenable au hasard est toujours de $1/2$. Il faut travailler dans

$$\mathbb{F}_{p^2} = \mathbb{F}_p[X]/(X^2 - 2nX + a).$$

La complexité est aussi de $O((\log p)^3)$ opérations, mais toute la complexité de travailler dans $\mathbb{F}_{p^2}$ est cachée dans la constante du O.

Carrés dans $\mathbb{F}_p$: calcul des racines carrées, algorithme de Cipolla, un exemple

Données. On cherche une racine carrée de $a = 9\,964$ modulo $p = 57\,649$.

On prend au hasard $n = 20\,030$ et $n^2 - a$ n'est pas un carré modulo p.

On travaille dans $\mathbb{F}_p[X]/(X^2 + 17\,589X + 9\,964)$ et on calcule

$$X^{(57\,649+1)/2} \mod (57\,649, X^2 + 17\,589X + 9\,964)$$

par l'exponentiation rapide.

On trouve (dernières étapes de l'exponentiation rapide gauche-droite)

$$\begin{aligned}
X^{14\,412} &\mod (57\,649, X^2 + 17\,589X + 9\,964) = 47\,770X + 41\,615 \\
X^{28\,824} &\mod (57\,649, X^2 + 17\,589X + 9\,964) = 13\,752X + 46\,373 \\
X^{28\,825} &\mod (57\,649, X^2 + 17\,589X + 9\,964) = 6\,745
\end{aligned}$$

Donc 28 825 est une racine carrée de 9 964 modulo 57 649.

L'algorithme LLL : réseaux de $\mathbb{R}^n$

Définitions. Un réseau $L \subset \mathbb{R}^n$ de dimension k est une sous-$\mathbb{Z}$-module tel qu'il existe une famille libre de vecteurs $(\vec{v}_1, \ldots, \vec{v}_k)$ de $\mathbb{R}^n$ avec

$$L = \mathbb{Z}\,\vec{v}_1 + \cdots \mathbb{Z}\,\vec{v}_k.$$

La famille $(\vec{v}_1, \ldots, \vec{v}_k)$ est une base de L.

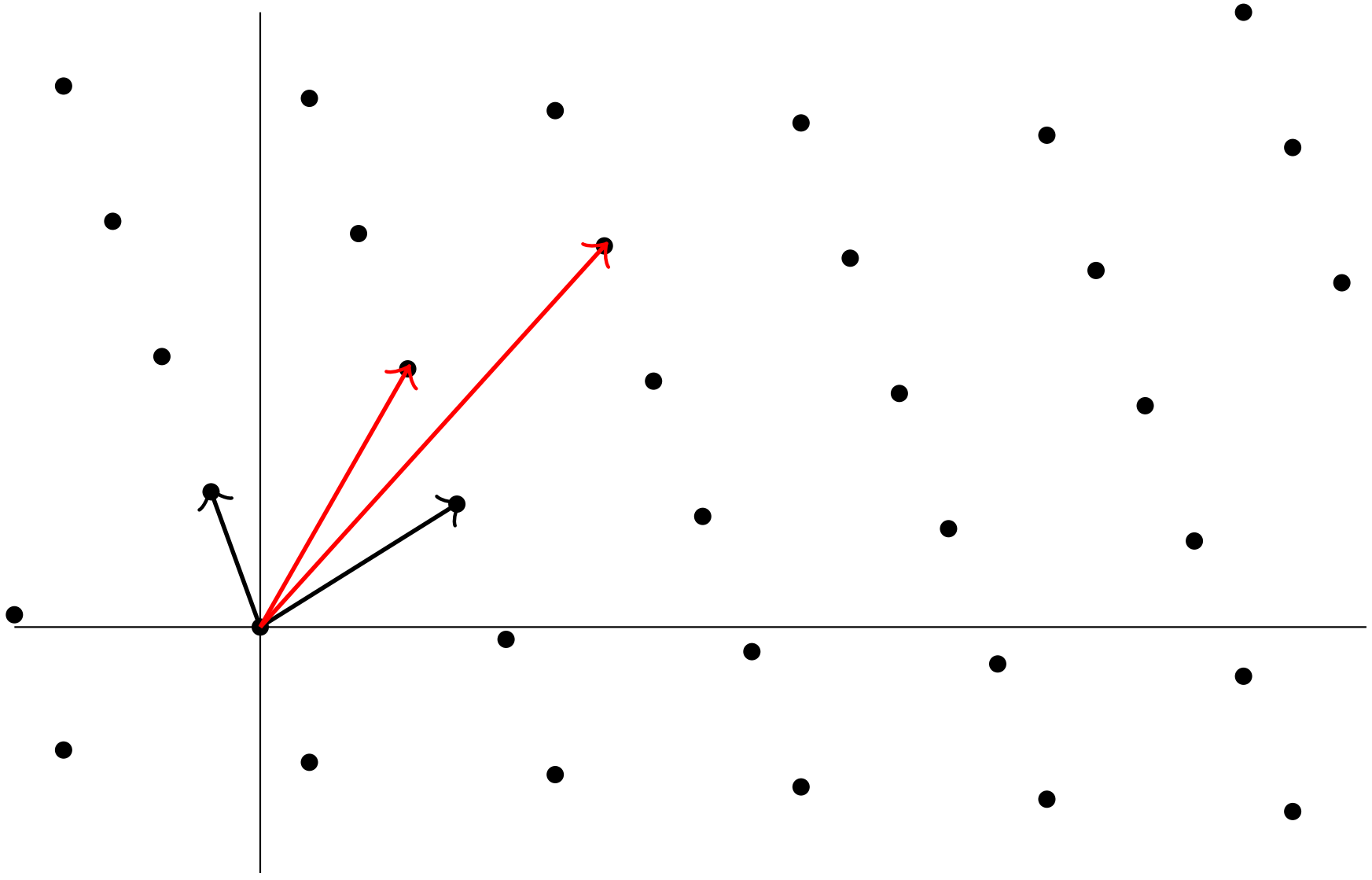
Un réseau L est un sous-ensemble discret, en particulier il n'y a qu'un nombre fini de points de L dans toute boule de $\mathbb{R}^n$.

Convention. On suppose à présent par défaut que les réseaux considérés sont de dimension maximale, c'est-à-dire $k = n$. Donc on passe d'une base à une autre par multiplication par une matrice de $\mathrm{GL}_n(\mathbb{Z})$.

Problèmes à résoudre.

1. Trouver $\vec{v} \in L$, non nul, tel que $||\vec{v}||$ est minimal.
2. Pour $\vec{x} \in \mathbb{R}^n$, trouver $\vec{v} \in L$ tel que $||\vec{x} - \vec{v}||$ est minimal.
3. Trouver une base de L avec des vecteurs de longueur minimale ou la plus orthogonale possible.

L'algorithme LLL : un exemple de réseau de $\mathbb{R}^2$



Réseau de base (en rouge) $\vec{v}_1 = {}^t(12, 21)$ et $\vec{v}_2 = {}^t(28, 31)$

L'algorithme LLL : formalisme des réseaux

Représentation. Un réseau L de $\mathbb{R}^n$ est donné par la matrice $M_{\mathcal{B}}$ de dimension $n \times n$ exprimant les vecteurs d'une base $\mathcal{B}$ de L en colonnes.

Déterminant. Le déterminant de L est

$$\det(L) = |\det(M_{\mathcal{B}})| \neq 0$$

pour une base $\mathcal{B}$ de L. Il ne dépend pas du choix de la base.

Inégalités de Hadamard. Soit $(\vec{v}_1, \ldots, \vec{v}_n)$ une base de L, alors

$$\det(L) \leq \prod_{i=1}^{n} ||\vec{v}_i|| \leq n^{n/2} B$$

où B est une borne sur les coefficients des $\vec{v}_i$.

Autre formulation. Une autre manière de voir les réseaux est en terme de formes quadratiques définies positives.

L'algorithme LLL : réseaux en dimension 2

Algorithme de Gauss. Soit L un réseau de $\mathbb{R}^2$ de base $(\vec{v}_1, \vec{v}_2)$. L'algorithme suivant trouve $(a, b) \in \mathbb{Z}^2 - \{(0,0)\}$ tel que

$$||a\,\vec{v}_1 + b\,\vec{v}_2|| \quad \text{est minimal.}$$

Algorithme.

(1) Poser $V_1 \leftarrow ||\vec{v}_1||^2$, $V_2 \leftarrow ||\vec{v}_2||^2$.

Si $V_1 < V_2$, alors faire $\vec{v}_1 \leftrightarrow \vec{v}_2$, $V_1 \leftrightarrow V_2$.

(2) Faire $t \leftarrow \vec{v}_1 \cdot \vec{v}_2$, $r \leftarrow \lfloor n/V_2 \rceil$, $T \leftarrow V_1 - 2rt + r^2 V_2$.

(3) Si $T \geq V_2$, alors retourner $\vec{v}_2$ et terminer.

Sinon, faire $(\vec{v}_1, \vec{v}_2) \leftarrow (\vec{v}_2, \vec{v}_1 - r\vec{v}_2)$, et $(V_1, V_2) \leftarrow (V_2, T)$ et retourner en (2).

Remarques. On a $||\vec{v}_1 - r\vec{v}_2||^2 = V_1 - 2rt + r^2 V_2$.

Cet algorithme est une forme adaptée de l'algorithme d'Euclide.

L'algorithme LLL : un exemple de l'algorithme de Gauss

Données. On prend $\vec{v}_1 = {}^t(12, 21)$ et $\vec{v}_2 = {}^t(28, 31)$

On a $V_1 = 585$ et $V_2 = 1745$, donc on échange $\vec{v}_1$ et $\vec{v}_2$, et V_1 et V_2.

On calcule $t \leftarrow \vec{v}_1 \cdot \vec{v}_2 = 987$, puis $r \leftarrow \lfloor t/V_2 \rceil = 2$ et finalement $T \leftarrow V_1 - 2rt + r^2 V_2 = 137$.
On fait

$$(\vec{v}_1, \vec{v}_2) \leftarrow (\vec{v}_2, \vec{v}_1 - r\vec{v}_2) = ({}^t(12, 21), {}^t(4, -11),$$

et $(V_1, V_2) \leftarrow (V_2, T) = (585, 137)$.

On calcule $t \leftarrow \vec{v}_1 \cdot \vec{v}_2 = -183$, puis $r \leftarrow \lfloor t/V_2 \rceil = -1$ et finalement $T \leftarrow V_1 - 2rt + r^2 V_2 = 356$.

On a $T \geq V_2$ donc l'algorithme est terminé et un vecteur de longueur minimal est

$${}^t(4, -11) \text{ de norme } \sqrt{137}.$$

L'algorithme LLL : orthogonalisation de Gram-Schmidt

Orthogonalisation. Soit $(\vec{v}_1, \ldots, \vec{v}_n)$ une base d'un réseau L, on définit de manière récursive

$$\vec{v}_i^* = \vec{v}_i - \sum_{j=1}^{i-1} \mu_{i,j} \vec{v}_j^* \quad \text{où } \mu_{i,j} = \frac{\vec{v}_i \cdot \vec{v}_j^*}{||\vec{v}_j^*||^2} \quad \text{pour } 1 \leq j < i.$$

Propriétés. Le vecteur $\vec{v}_i^*$ est la projection de $\vec{v}_i$ sur le complémentaire orthogonal du sous-espace engendré par $(\vec{v}_1, \ldots, \vec{v}_{i-1})$ donc on a $\vec{v}_i \cdot \vec{v}_i^* = ||\vec{v}_i^*||^2$ pour tout i, et la base $(\vec{v}_1^*, \ldots, \vec{v}_n^*)$ est orthogonale.

On a, pour tout i, $\vec{v}_i^* \in \mathcal{K}^n$ où $\mathcal{K} \subset \mathbb{R}$ (corps) avec $\vec{v}_i \in \mathcal{K}^n$ pour tout i.

Le procédé d'orthogonalisation prend $O(n^3)$ opérations dans le corps $\mathcal{K}$.

La matrice des coordonnées des $(\vec{v}_i)_i$ dans la base $(\vec{v}_i^*)_i$ est une matrice triangulaire supérieure avec des 1 sur la diagonale. En particulier, on a

$$\det(L) = \prod_{i=1}^{n} ||\vec{v}_i^*||.$$

Attention. En général, les $\vec{v}_i^*$ ne sont pas dans L.

L'algorithme LLL : base LLL-réduite

Définition. Soit $\mathcal{B} = (\vec{v}_i)_i$ une base d'un réseau L. On note $\vec{v}_i^*$ et $\mu_{i,j}$ comme dans l'écran précédent. La base $\mathcal{B}$ est LLL-réduite si on a

$$|\mu_{i,j}| \leq 1/2 \quad \text{pour } 1 \leq j < i \leq n$$

$$\text{et} \quad ||\vec{v}_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right) ||\vec{v}_{i-1}^*||^2 \quad \text{pour } 1 < i \leq n.$$

Propriétés des bases LLL-réduites. Supposons que $(\vec{v}_i)_i$ est une base LLL-réduite d'un réseau L.

- $\det(L) \leq \prod_i ||\vec{v}_i|| \leq 2^{n(n-1)/4} \det(L)$.
- Pour tout $\vec{x} \in L$, non nul, on a $||\vec{v}_1|| \leq 2^{(n-1)/2} ||\vec{x}||$.
- Plus généralement pour $(\vec{x}_1, \ldots, \vec{x}_k)$ une famille libre de vecteurs de L, on a

$$||\vec{v}_j|| \leq 2^{(n-1)/2} \max_{1 \leq i \leq k} \{||\vec{x}_i||\} \quad \text{pour tout } 1 \leq j \leq k.$$

L'algorithme LLL : procédure de réduction de taille

Etape de réduction. Supposons que $(\vec{v}_1, \ldots, \vec{v}_{k-1})$ vérifient les conditions de LLL-réduction. Pour le prochain vecteur $\vec{v}_k$, on doit d'abord assurer que $|\mu_{k,j}| \leq 1/2$ pour $1 \leq j < k$.

Pour cela, on pose $q = \lfloor \mu_{k,k-1} \rceil$, et on remplace

$$\vec{v}_k \leftarrow \vec{v}_k - q\,\vec{v}_{k-1}.$$

Il suit que

$$\mu_{k,k-1} \leftarrow \frac{\vec{v}_k \cdot \vec{v}^*_{k-1}}{||v^*_{k-1}||^2} - q\frac{\vec{v}_{k-1} \cdot \vec{v}^*_{k-1}}{||v^*_{k-1}||^2} = \mu_{k,k-1} - q$$

et on a bien $|\mu_{k,k-1}| \leq 1/2$.

Puis, on pose $q = \lfloor \mu_{k,k-2} \rceil$, et on remplace de même

$$\vec{v}_k \leftarrow \vec{v}_k - q\,\vec{v}_{k-2}.$$

Par le raisonnement ci-dessus, on obtient bien $|\mu_{k,k-2}| \leq 1/2$, et comme $\vec{v}_{k-2}$ est orthogonal à $\vec{v}^*_{k-1}$, la valeur de $\mu_{k,k-1}$ n'est pas modifiée.

On procède de même pour $k-3, k-4, \ldots, 1$.

L'algorithme LLL : exemple de réduction de taille

Données. On considère le réseau L de base $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$ avec

$$\vec{v}_1 = {}^t(19, -10, 18), \quad \vec{v}_2 = {}^t(-11, 14, -12), \quad \vec{v}_3 = {}^t(10, 13, 15).$$

On a $\vec{v}_1^* = \vec{v}_1$ et on calcule $\mu_{2,1} = -113/157$. On a $\lfloor \mu_{2,1} \rceil = -1$ et donc on remplace

$$\vec{v}_2 \leftarrow \vec{v}_2 + \vec{v}_1 = {}^t(8, 4, 6) \quad \text{et} \quad \mu_{2,1} \leftarrow \mu_{2,1} + 1 = \frac{44}{157}.$$

On calcule $\mu_{3,2} = 3\,389/1\,422$ et $\lfloor \mu_{3,2} \rceil = 2$. Donc on remplace

$$\vec{v}_3 \leftarrow \vec{v}_3 - 2\vec{v}_2 = {}^t(-6, 5, 3) \quad \text{et} \quad \mu_{3,2} \leftarrow \mu_{3,2} - 2 = \frac{545}{1\,422}.$$

Puis, on calcule $\mu_{3,1} = -22/157$ et $|\mu_{3,1}| \leq 1/2$ donc la réduction est terminée.
On obtient finalement

$$\vec{v}_1 = {}^t(19, -10, 18), \quad \vec{v}_2 = {}^t(-6, 5, 3), \quad \vec{v}_3 = {}^t(8, 4, 6).$$

Algorithme de A.K. Lenstra, H.W. Lenstra et L. Lovász

Principe. On suppose donc que $(\vec{v}_1, \ldots, \vec{v}_{k-1})$ vérifient les conditions de LLL-réduction. On a vu comment modifier $\vec{v}_k$ de telle sorte que $|\mu_{k,j}| \leq 1/2$ pour $1 \leq j < k$.

Maintenant la condition

$$||\vec{v}_k^*||^2 \geq \left(\frac{3}{4} - \mu_{k,k-1}^2\right) ||\vec{v}_{k-1}^*||^2$$

n'est pas nécessairement vérifiée. Si c'est le cas, on remplace $k \leftarrow k+1$.

Si ce n'est pas le cas, on échange $\vec{v}_k$ et $\vec{v}_{k-1}$ et la condition est vérifiée, mais alors on sait juste que $(\vec{v}_1, \ldots, \vec{v}_{k-2})$ vérifie les conditions de LLL-réduction donc on doit faire $k \leftarrow k-1$.

Théorème

La procédure donnée ci-dessus termine et, si les $\vec{v}_i$ ont des coordonnées entières bornées par B, elle calcule une base LLL-réduite en $O(n^6 \log^3 B)$ opérations binaires.

L'algorithme LLL : exemple

Données. On considère le réseau L de base $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$ avec

$$\vec{v}_1 = {}^t(19, -10, 18), \quad \vec{v}_2 = {}^t(-11, 14, -12), \quad \vec{v}_3 = {}^t(10, 13, 15).$$

$k = 1$. On a $\vec{v}_1^* = \vec{v}_1$ et on calcule $\mu_{2,1} = -113/157$. On a $\lfloor \mu_{2,1} \rceil = -1$ et donc on remplace

$$\vec{v}_2 \leftarrow \vec{v}_2 + \vec{v}_1 = {}^t(8, 4, 6) \quad \text{et} \quad \mu_{2,1} \leftarrow \mu_{2,1} + 1 = \frac{44}{157}.$$

On calcule

$$8\,532/157 = ||\vec{v}_2^*||^2 < \left(\frac{3}{4} - \mu_{2,1}^2\right) ||\vec{v}_1^*||^2 = 331\,015/628$$

et donc on échange $\vec{v}_1$ et $\vec{v}_2$

$$\vec{v}_1 = {}^t(8, 4, 6) \quad \text{et} \quad \vec{v}_2 = {}^t(19, -10, 18).$$

On a $\vec{v}_1^* = \vec{v}_1$ et on calcule $\mu_{2,1} = 55/29$. On a $\lfloor \mu_{2,1} \rceil = 2$ et donc on remplace

$$\vec{v}_2 \leftarrow \vec{v}_2 - 2\vec{v}_1 = {}^t(3, -18, 6) \quad \text{et} \quad \mu_{2,1} \leftarrow \mu_{2,1} - 2 = \frac{-3}{29}.$$

On calcule

$$10\,665/29 = ||\vec{v}_2^*||^2 \geq \left(\frac{3}{4} - \mu_{2,1}^2\right) ||\vec{v}_1^*||^2 = 2\,487/29$$

et donc on passe à $\vec{v}_3$.

L'algorithme LLL : exemple

$k = 2$. On a

$$\vec{v}_1 = {}^t(8, 4, 6), \quad \vec{v}_2 = {}^t(3, -18, 6), \quad \vec{v}_3 = {}^t(10, 13, 15).$$

On calcule $\mu_{3,2} = -176/711$ donc pas de réduction. On calcule $\mu_{3,1} = 111/58$ et $\lfloor \mu_{3,1} \rceil = 2$ et donc on remplace

$$\vec{v}_3 \leftarrow \vec{v}_3 - 2\vec{v}_1 = {}^t(-6, 5, 3) \quad \text{et} \quad \mu_{3,1} \leftarrow \mu_{3,1} - 2 = \frac{-5}{58}.$$

On calcule

$$11\,045/237 = ||\vec{v}_3^*||^2 < \left(\frac{3}{4} - \mu_{3,2}^2\right) ||\vec{v}_2^*||^2 = 6\,963\,295/27\,492$$

et donc on échange $\vec{v}_2$ et $\vec{v}_3$

$$\vec{v}_2 = {}^t(-6, 5, 3) \quad \text{et} \quad \vec{v}_3 = {}^t(3, -18, 6).$$

$k = 1$. On calcule $\mu_{2,1} = -5/28$, donc pas de réduction, puis on calcule

$$2\,005/29 = ||\vec{v}_2^*||^2 < \left(\frac{3}{4} - \mu_{2,1}^2\right) ||\vec{v}_1^*||^2 = 2\,498/29$$

et donc on échange $\vec{v}_1$ et $\vec{v}_2$

$$\vec{v}_1 = {}^t(-6, 5, 3) \quad \text{et} \quad \vec{v}_2 = {}^t(8, 4, 6).$$

L'algorithme LLL : exemple

Suite $k = 1$. On a

$$\vec{v}_1 = {}^t(-6, 5, 3), \quad \vec{v}_2 = {}^t(8, 4, 6), \quad \vec{v}_3 = {}^t(3, -18, 6).$$

On a $\vec{v}_1^* = \vec{v}_1$. On calcule $\mu_{2,1} = -1/7$ donc pas de réduction.

$k = 2$. On calcule $\mu_{3,2} = -87/401$, donc pas de réduction, puis $\mu_{3,1} = -9/7$. Ainsi, $\lfloor \mu_{3,1} \rceil = -1$ et donc on remplace

$$\vec{v}_3 \leftarrow \vec{v}_3 + \vec{v}_1 = {}^t(-3, -13, 9) \quad \text{et} \quad \mu_{3,1} \leftarrow \mu_{3,1} - 2 = \frac{-2}{7}.$$

On calcule

$$99\,405/401 = ||\vec{v}_3^*||^2 \leq \left(\frac{3}{4} - \mu_{3,2}^2\right) ||\vec{v}_2^*||^2 = 452\,127/5\,614$$

et donc la réduction est terminée.

Conclusion. On obtient la base LLL-réduite

$$\vec{v}_1 = {}^t(-6, 5, 3), \quad \vec{v}_2 = {}^t(8, 4, 6), \quad \vec{v}_3 = {}^t(-3, -13, 9),$$
$$||\vec{v}_1||^2 = 70, \quad ||\vec{v}_2||^2 = 116, \quad ||\vec{v}_3||^2 = 259.$$

à comparer avec la base de départ

$$\vec{v}_1 = {}^t(19, -10, 18), \quad \vec{v}_2 = {}^t(-11, 14, -12), \quad \vec{v}_3 = {}^t(10, 13, 15),$$
$$||\vec{v}_1||^2 = 785, \quad ||\vec{v}_2||^2 = 461, \quad ||\vec{v}_3||^2 = 494.$$

L'algorithme LLL : résolutions partielles des trois problèmes

Base LLL-réduite. Soit $(\vec{v}_1, \ldots, \vec{v}_n)$ une base LLL-réduite d'un réseau L.

Vecteur minimal. l'algorithme LLL donne un vecteur $\vec{v}_1$ presque minimal. On peut alors énumérer les vecteurs de L de norme plus petite pour voir s'il y en a un peu plus petit (complexité exponentielle en n).

Vecteur le plus près. Pour $\vec{x} \in \mathbb{R}^n$, on peut trouver facilement $(r_1, \ldots, r_n) \in \mathbb{R}^n$ avec $r_1\,\vec{v}_1 + \cdots + r_n\,\vec{v}_n = \vec{x}$. On peut prendre

$$\lfloor r_1 \rceil\,\vec{v}_1 + \cdots + \lfloor r_n \rceil\,\vec{v}_n \in L$$

comme vecteur de L proche de $\vec{x}$.
Le vecteur le plus proche est obtenu en prenant la bonne combinaison entre $\lfloor r_i \rfloor$ ou $\lceil r_i \rceil$ (complexité en 2^n).

Bases optimales. Une base LLL-réduite est presque optimale et souvent suffisante dans la plupart des applications.

Exemple de recherche du vecteur minimal

Données. On considère le réseau L de base LLL-réduite

$$\vec{v}_1 = {}^t(-6,5,3), \quad \vec{v}_2 = {}^t(8,4,6), \quad \vec{v}_3 = {}^t(-3,-13,9),$$

Vecteur minimal. Soit la forme quadratique définie positive

$$\begin{aligned} Q(n_1,n_2,n_3) &= ||n_1\vec{v}_1 + n_2\vec{v}_2 + n_3\vec{v}_3||^2 \\ &= 70n_1^2 + 116n_2^2 + 259n_3^2 - 20n_1n_2 - 40n_1n_3 - 44n_2n_3 \\ &= 70(n_1 - \tfrac{1}{7}n_2 - \tfrac{2}{7}n_3)^2 + \tfrac{802}{7}(n_2 - \tfrac{87}{401}n_3)^2 + \tfrac{99\,405}{401}n_3^2 \end{aligned}$$

On cherche $(n_1,n_2,n_3) \in \mathbb{Z}^3$ tel que $Q(n_1,n_2,n_3) < ||\vec{v}_1||^2 = 70$. Donc

$$|n_3| < \sqrt{\tfrac{401}{99\,405} \times 70} \approx 0,53 \quad \text{d 'où} \quad n_3 = 0.$$

Puis, on doit avoir

$$|n_2| < \sqrt{\tfrac{7}{802} \times 70} \approx 0,78 \quad \text{d 'où} \quad n_2 = 0.$$

Et finalement n_1 doit vérifier

$$|n_1| < \sqrt{\tfrac{1}{70} \times 70} = 1 \quad \text{d 'où} \quad n_1 = 0.$$

Donc $\vec{v}_1$ est un vecteur minimal de L.

Exemple de recherche du vecteur le plus proche

Données. On considère le réseau L de base LLL-réduite

$$\vec{v}_1 = {}^t(-6, 5, 3), \quad \vec{v}_2 = {}^t(8, 4, 6), \quad \vec{v}_3 = {}^t(-3, -13, 9),$$

Vecteur le plus proche. On prend $\vec{x} = {}^t(158, 370, 444) \in \mathbb{R}^3$.
On a

$$\vec{x} = \underbrace{\frac{9\,356}{235}}_{r_1}\vec{v}_1 + \underbrace{\frac{11\,869}{235}}_{r_2}\vec{v}_2 + \underbrace{\frac{562}{235}}_{r_3}\vec{v}_3.$$

On calcule

$$\vec{v} = \lfloor r_1 \rceil \vec{v}_1 + \lfloor r_2 \rceil \vec{v}_2 + \lfloor r_3 \rceil \vec{v}_3 = {}^t(162, 378, 444) \in L,$$

et $||\vec{x} - \vec{v}||^2 = 80$.

Si on teste les 7 autres combinaisons pour arrondir (r_1, r_2, r_3), on trouve

$$\vec{v}' = \lceil r_1 \rceil \vec{v}_1 + \lfloor r_2 \rfloor \vec{v}_2 + \lfloor r_3 \rfloor \vec{v}_3 = {}^t(154, 374, 438) \in L$$

avec $||\vec{x} - \vec{v}'||^2 = 68$ qui est le vecteur de L le plus proche de $\vec{x}$.

Quelques applications de LLL : image et noyau d'une matrice

Modification. On peut modifier l'algorithme LLL pour travailler avec une famille de vecteurs liés. Quand une réduction fait apparaître un vecteur nul, on le fait passer devant et on l'ignore par la suite.

On obtient finalement une famille composée d'un certain nombre de vecteurs nuls, puis d'une famille libre de vecteurs.

Si on applique cet algorithme aux colonnes d'une matrice à coefficients entiers A de dimension $n \times m$, les r premiers vecteurs nuls correspondent au noyau de A (donc de dimension r) et les $m - r$ vecteurs non nuls engendrent l'image de A (donc de dimension $m - r$).
En gardant en mémoire les transformations utilisées dans l'algorithme, on a donc les relations qui ont menées à l'obtention des r vecteurs nuls, c'est-à-dire à un système de relations engendrant le noyau de A.

Avantage. Cette méthode a l'avantage par rapport à d'autres méthodes, comme la réduction d'Hermite, de construire des générateurs simples pour l'image et le noyau de A.

Quelques applications de LLL : polynôme minimal d'un nombre algébrique

Problème. Soit $\alpha \in \bar{\mathbb{Q}} \cap \mathbb{R}$ un nombre algébrique (donnée par une approximation réelle). Supposons que le degré de α est au plus n. Alors, il existe

$$A(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_0 \in \mathbb{Z}[X]$$

tel que $A(\alpha) = 0$.

Réseau. Trouver A revient à travailler une relation entière entre les nombres $1, \alpha, \alpha^2, \ldots, \alpha^n$. C'est-à-dire une relation entre les $n+1$ vecteurs

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ S\alpha^n \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ S\alpha^{n-1} \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ S\alpha \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ S \end{pmatrix}$$

avec S paramètre (grand entier) qui dépend de la précision à laquelle α est connu, ainsi que de la taille maximale des coefficients a_i.

Remarque. Il est possible de modifier la méthode pour α complexe.
La forme quadratique correspondante est

$$Q(a_0, \ldots a_n) = a_0^2 + \cdots + a_n^2 + S^2 \left(a_n + a_1\alpha + \cdots + a_n\alpha^n\right)^2 .$$

Quelques applications de LLL : exemple du calcul du polynôme minimal d'un nombre algébrique

Données. On considère $\alpha = \sqrt[3]{2} + \sqrt{2} + \sqrt{3} \approx 4.406185419836845507$. Alors α est un entier algébrique de degré au plus $3 \times 2 \times 2 = 12$.

On suppose qu'on connaît α avec 100 chiffres de précision et on prend $S = 10^{60}$. On arrondit les entrées de la matrice pour pouvoir travailler avec des nombres entiers plutôt que des nombres réels (problème de stabilité).

On trouve la relation

$$^t(-1, 0, 30, 8, -303, 0, 1\,036, 1\,104, -663, -3\,488, -1\,290, -696, 3\,863)$$

qui donne le polynôme

$$X^{12} - 30X^{10} - 8X^9 + 303X^8 - 1\,036X^6 - 1\,104X^5 + 663X^4 + 3\,488X^3 + 1\,290X^2 + 696X - 3\,863.$$