Hoofdstuk 3

Algoritmen voor roosters

3.1 Vinden van korte vectoren in een rooster

Om voor twee roosters te kunnen beslissen of ze equivalent zijn, moeten we testen of de automorfisme groepen geconjugeerd zijn onder een matrix $T \in GL_n(\mathbb{Z})$.

Maar om zo'n test te kunnen toepassen, moeten in eerste instantie de automorfisme groepen bepalen. Voor speciale roosters laat zich de automorfisme groep makkelijk aangeven, maar in het algemeen is dit een serieus probleem. Een aanpak om dit op te lossen is, de automorfisme groep als groep van permutaties van korte vectoren in het rooster te construeren. Dit vereist nu een methode, om in een gegeven rooster de vectoren tot een zekere lengte te bepalen.

Gegeven zijn een rooster L met Gram matrix F en een constante $C > 0$. Gezocht zijn alle roostervectoren $0 \neq v \in L$ met $\|v\|^2 \leq C$.

We bekijken eerst het voorbeeld van een kubisch rooster in $\mathbb{R}^3$ met Gram matrix $\begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix}$. Een vector $v = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ heeft norm $a(x^2 + y^2 + z^2)$ en uit $a(x^2 + y^2 + z^2) \leq C$ volgt in het bijzonder $x^2, y^2, z^2 \leq \frac{C}{a}$. De coördinaten van v zijn dus begrensde.

Een iets algemener rooster heeft nog steeds een basis van vectoren die loodrecht op elkaar staan maar niet meer noodzakelijk dezelfde lengte hebben. Zo'n rooster heeft Gram matrix $\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$ en uit $ax^2 + by^2 + cz^2 \leq C$ volgt $x^2 \leq \frac{C}{a}$, $y^2 \leq \frac{C}{b}$ en $z^2 \leq \frac{C}{c}$.

We kunnen voor dit rooster de vectoren van norm $\leq C$ als volgt bepalen:

- kies de laatste coördinaat z zo dat $z^2 \leq \frac{C}{c}$;
- kies vervolgens y zo dat $y^2 \leq \frac{C - cz^2}{b}$;
- kies tenslotte x zo dat $x^2 \leq \frac{C - by^2 cz^2}{a}$.

Door in iedere stap alle mogelijkheden te doorlopen, vinden we de volledige lijst van vectoren met norm $\leq C$. Het zal duidelijk zijn dat deze methode precies hetzelfde werkt voor n-dimensionale roosters met diagonale Gram matrix.

Maar hoe zit het met roosters die geen diagonale Gram matrix hebben? We bekijken als voorbeeld het hexagonale rooster met Gram matrix $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$. Voor een vector $v = \begin{pmatrix} x \\ y \end{pmatrix}$ geldt

$$\|v\|^2 = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 2x^2 - 2xy + 2y^2.$$

Hoe vinden we nu bijvoorbeeld de vectoren met $2x^2 - 2xy + 2y^2 \leq 6$?

Het cruciale idee is, de kwadratische vorm met behulp van de methode van *kwadratische aanvulling* (completing the square) te herschrijven als som van kwadraten. Er geldt

$$2x^2 - 2xy + y^2 = 2(x - \frac{1}{2}y)^2 + \frac{3}{2}y^2.$$

Met betrekking tot de nieuwe coördinaten $x' = x - \frac{1}{2}y$ en $y' = y$ zijn we nu terug naar het geval van diagonale Gram matrices.

Om de vectoren tot en met norm 6 te vinden, moeten we nu de mogelijke waarden van y' doorlopen en voor iedere van deze waarden de mogelijkheden voor x' bepalen.

Uit $\frac{3}{2}y^2 \leq 6$ volgt $y^2 \leq 4$, dus $|y| \leq 2$. We beginnen met de laagste waarde voor y en gaan door tot 0:

- $y = -2$: In dit geval moet $2(x - \frac{1}{2}y)^2 \leq 6 - \frac{3}{2}2^2 = 0$ zijn, dus $x = \frac{1}{2}y$ en dus $x = -1$.
- $y = -1$: In dit geval moet $2(x - \frac{1}{2}y)^2 \leq 6 - \frac{3}{2}$ zijn, dus $(x - \frac{1}{2}y)^2 \leq \frac{9}{4}$ en dus $x \in [-\frac{1}{2}y - \frac{3}{2}, -\frac{1}{2}y + \frac{3}{2}] = [-2, 1]$. De mogelijke waarden van x zijn dus $-2, -1, 0, 1$.
- $y = 0$: In dit geval moet $2(x - \frac{1}{2}y)^2 \leq 6$ zijn, dus $(x - \frac{1}{2}y)^2 \leq 3$ en dus $x \in [-\frac{1}{2}y - \sqrt{3}, -\frac{1}{2}y + \sqrt{3}] = [-\sqrt{3}, \sqrt{3}]$. De mogelijke waarden van x zijn dus $-1, 0, 1$.

In principe zouden we nog met $y = 1$ en $y = 2$ door moeten gaan, maar omdat $\|v\| = \| - v\|$ vinden we deze vectoren als negatieven van vectoren die al bepaald zijn. In feite moeten we in het proces in de laatste component die niet nul is alleen maar tot 0 lopen. Als deze component de eerste component is, krijgen we als laatste vector de nulvector en weten op deze manier dat het algoritme beeindigd is.

In het voorbeeld boven zouden we dus in de laatste stap alleen maar de waarde $x = -1$ nemen, bij $x = 0$ hebben we de nulvector gevonden en het algoritme eindigt. De vectoren die we gevonden hebben zijn: $\begin{pmatrix} -1 \\ -2 \end{pmatrix}$, $\begin{pmatrix} -2 \\ -1 \end{pmatrix}$, $\begin{pmatrix} -1 \\ -1 \end{pmatrix}$, $\begin{pmatrix} 0 \\ -1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$, $\begin{pmatrix} -1 \\ 0 \end{pmatrix}$ en hun negatieven.

Het algemene algoritme, dat bekend staat als *Fincke-Pohst* algoritme werkt als volgt: We diagonaliseren de Gram matrix F met elementaire operaties over $\mathbb{Q}$ die we simultaan op de rijen en kolommen van F toepassen. De transformatiematrix krijgen we cadeau als we de operaties op de kolommen van een identiteitsmatrix toepassen. Op deze manier vinden we een boven driehoeksmatrix T met $T^{tr}FT = D$ voor een diagonaalmatrix D. Omdat T een driehoeksmatrix is, vinden we makkelijk de inverse T^{-1} en er geldt $F = T^{-tr}DT^{-1}$. In het bijzonder is de kwadratische vorm $q(v) = v^{tr}Fv$ op de coördinaten van v gegeven door $q(v) = (T^{-1}v)^{tr}D(T^{-1}v)$, d.w.z. in de componenten van $T^{-1}v$ hebben we een som van kwadraten.

We noteren de elementen van T^{-1} met q_{ij}, d.w.z. de nieuwe i-de component is $x_i' = \sum_{j=i}^{n} q_{ij}x_j$ (merk op dat $q_{ij} = 0$ voor $j < i$).

Voorbeeld: We diagonaliseren de matrix $\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$ en passen de transformaties tegelijkertijd op de eenheidsmatrix toe.

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \quad \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{3}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{3}{2} \end{pmatrix} \quad \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{3} \\ 0 & 1 & -\frac{1}{3} \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & \frac{4}{3} \end{pmatrix}$$

We hebben dus $T = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{3} \\ 0 & 1 & -\frac{1}{3} \\ 0 & 0 & 1 \end{pmatrix}$ en dus $(q_{ij}) = T^{-1} = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & \frac{1}{3} \\ 0 & 0 & 1 \end{pmatrix}$, dus zijn de nieuwe coördinaten

$$x' = x + \frac{1}{2}y + \frac{1}{2}z, \quad y' = y + \frac{1}{3}z, \quad z' = z.$$

De kwadratische vorm $q(v) := v^{tr}Fv$ voor $v = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ is dus gegeven door $q(v) = 2x'^2 + \frac{3}{2}y'^2 + \frac{4}{3}z'^2 = 2(x + \frac{1}{2}y + \frac{1}{2}z)^2 + \frac{3}{2}(y + \frac{1}{3}z)^2 + \frac{4}{3}z^2$.

In het algemeen kunnen we dus van de volgende situatie uitgaan: De kwadratische vorm $q(v)$ voor $v = (x_1, \ldots, x_n)^{tr}$ is gegeven door $q(v) = \sum_{i=1}^{n} a_i x_i'^2 = \sum_{i=1}^{n} a_i(x_i + \sum_{j=i+1}^{n} q_{ij}x_j)^2$.

Stel nu we hebben de coördinaten $x_n, x_{n-1}, \ldots, x_{k+1}$ al gekozen. Dan is $N_i := \sum_{i=k+1}^{n} a_i x_i'^2 = \sum_{i=k+1}^{n} a_i(x_i + \sum_{j=i+1}^{n} q_{ij}x_j)^2$ de norm die van deze componenten al opgeleverd wordt, en we moeten dus hebben dat $a_i x_k'^2 \leq C - N_i$, dus $|x_k + \sum_{j=k+1}^{n} q_{kj}x_j| \leq \sqrt{\frac{C-N_i}{a_i}}$. Hieruit volgt dat x_k in het interval

$$[-\sqrt{\frac{C-N_i}{a_i}} - \sum_{j=k+1}^{n} q_{kj}x_j, \sqrt{\frac{C-N_i}{a_i}} - \sum_{j=k+1}^{n} q_{kj}x_j]$$

moet liggen.

Opdracht 14 Implementeer (bij voorkeur in MAGMA) het tijdens het college behandelde algoritme dat de vectoren in een rooster tot een zekere lengte bepaald. (Het algoritme wordt soms het *Fincke-Pohst algoritme* genoemd.)

`Input`: Gram matrix van het rooster, grens M voor de normen van de roostervectoren.

`Output`: Lijst van paren (tuples) met als eerste component een vector en als tweede component de norm van de vector (handig om later vectoren van een bepaalde lengte uit de lijst te vissen).

Opmerkingen:

- Maak de functie robuust tegen onverwachte input, bijvoorbeeld tegen matrices die niet symmetrisch of niet positief definiet zijn.
- De nulvector hoort niet in de lijst.
- Je mag zelf beslissen of je van elk paar $(v, -v)$ slechts één vector of beide vectoren teruggeeft (maar natuurlijk wel consistent).

Bepaal voor de roosters A_n en D_n met $2 \leq n \leq 8$ de aantallen van vectoren van lengte 2, 4 en 6. Controleer je resultaten door deze aantallen met combinatorische argumenten ook theoretisch te berekenen. •

3.2 Berekenen van de automorfisme groep van een rooster

Met behulp van de korte kunnen we nu ook de automorfisme groep van een rooster expliciet bepalen. Stel dat het rooster L de roosterbasis $(b_1, \ldots, b_n)$ en Gram matrix F (met betrekking tot deze basis) heeft. Een automorfisme van L heeft de eigenschap dat $g^{tr}Fg = F$ geldt, dus dat normen van en hoeken tussen de basisvectoren onder g invariant zijn. Maar dit betekent in het bijzonder dat $\|g(b_i)\|^2 = \|b_i\|^2$ voor alle i en als we $m := \max(\|b_i\|^2 \mid 1 \leq i \leq n)$ definiëren volgt hieruit dat de mogelijke beelden van b_i in de eindige verzameling $S(L, m)$ van vectoren met norm $\leq m$ liggen.

We kunnen nu een automorfisme van L stapsgewijs construeren, door een bij een de beelden van de basisvectoren uit $S(L, m)$ te kiezen. Dit betekent dat we de matrix g kolomsgewijs bepalen:

- Voor de eerste kolom $v_1 = g(b_1)$ moet alleen maar gelden dat $\|v_1\|^2 = \|b_1\|^2$.
- Als de eerste kolom v_1 gekozen is, moet de tweede kolom $v_2 = g(b_2)$ voldoen aan $\|v_2\|^2 = \|b_2\|^2$ en $v_2 \cdot v_1 = b_2 \cdot b_1$.
- Als de eerste $i-1$ kolommen gekozen zijn, moet de i-de kolom $v_i = g(b_i)$ voldoen aan $\|v_i\|^2 = \|b_i\|^2$ en $v_i \cdot v_j = b_i \cdot b_j$ voor $j < i$.

Deze stappen worden nu in een *backtrack algoritme* uitgevoerd. Als de eerste $i-1$ kolommen gekozen zijn, wordt een lijst C_i van candidaten aangemaakt, die aan de voorwaarden $\|v_i\|^2 = \|b_i\|^2$ en $v_i \cdot v_j = b_i \cdot b_j$ voor $j < i$ voldoen. Vervolgens wordt de eerste vector uit C_i als i-de kolom v_i van g gekozen en i tot $i+1$ verhoogd.

Als op gegeven moment geen candidaten voor de i-de kolom gevonden worden, dus de lijst C_i leeg is, laten zich de eerste $i-1$ kolommen niet tot een automorfisme voortzetten. In dit geval moeten we de gekozen vector v_{i-1} uit de lijst C_{i-1} schrappen en de volgende vector uit deze lijst als v_{i-1} kiezen. Als we tenslotte v_n succesvol hebben kunnen kiezen, hebben we een automorfisme $g \in Aut(L)$ gevonden.

Als het algoritme zo als beschreven uitgevoerd wordt, zouden alle elementen van $Aut(L)$ bepaald worden. Dit is natuurlijk voor grotere groepen ondoenlijk. In plaats hiervan worden voortbrengers voor een keten van stabilisatoren bepaald, namelijk voortbrengers voor de groepen $G_0 := Aut(L)$, $G_i := Stab_{G_{i-1}}(b_i) = \{g \in G_{i-1} \mid g(b_i) = b_i$. Het idee hierbij is als volgt: We veronderstellen dat een ondergroep $H \leq Aut(L)$ al gevonden is. Na het vinden van het eerste automorfisme g van L is H de cyklische groep voortgebracht door g en voor elk verder geconstrueerd automorfisme wordt de groep H groter. We bekijken nu de baan B_1 van b_1 onder de groep H. Voor de elementen $v \in B_1$ weten we al dat er een element $h \in Aut(L)$ bestaat met $h(b_1) = v$. We kunnen daarom de elementen van B_1 uit de lijst C_1 van candidaten schrappen en moeten alleen maar voor de overige vectoren proberen een automorfisme te construeren. Als we omgekeerd voor een vector $v_1 \in C_1$ hebben gezien, dat er geen voortzetting van deze vector tot een automorfisme bestaat, kan ook voor de andere vectoren in de baan van v_1 onder H geen automorfisme bestaan en we mogen dus met v_1 de volledige baan van v_1 onder H uit C_1 schrappen.

Hetzelfde argument kunnen we ook op de andere levels van vectoren b_i toepassen, waarbij we op zo'n level met de stabilisator G_{i-1} werken, die de eerste $i-1$ basisvectoren vast laat.

We merken nog op dat we tijdens het berekenen van de baan van de vector b_i op level i ook voortbrengers voor de stabilisator G_i (bijna) cadeau krijgen, daarom moet in veel gevallen het backtrack algoritme niet eens zo vaak doorlopen worden om een nieuw automorfisme te bepalen.

Opdracht 15 De schaling van het standaardrooster $\mathbb{Z}^n$ met $\sqrt{2}$ heet het wortelrooster B_n van type `B` en dimensie n. De meest geschikte roosterbasis hiervoor bestaat uit de geschaalde vectoren $b_i := \sqrt{2}\, e_i$ van de standaardbasis van $\mathbb{R}^n$.

(i) Bepaal de minimale vectoren van B_n.

(ii) De symmetrische groep S_n werkt op de vectoren van $\mathbb{R}^n$ door permutatie van de componenten. Laat zien dat deze werking automorfismen van B_n induceert en concludeer dat $Aut(B_n)$ een ondergroep isomorf met S_n heeft.

(iii) Ga na dat de minimale vectoren onder deze werking van S_n in twee banen liggen, dus dat S_n *niet transitief* op de minimale vectoren werkt.

(iv) Vind een element $g \in Aut(B_n)$ dat b_1 op $-b_1$ afbeeldt.

(v) Laat zien dat $Aut(B_n)$ een normaaldeler heeft, die een elementair abelse groep van orde 2^n is, dus van de vorm $C_2^n = \underbrace{C_2 \times C_2 \times \ldots \times C_2}_{n}$.

(vi) Bewijs dat $Aut(B_n)$ isomorf met het semidirecte product $C_2^n \rtimes S_n$ is.

•

Opdracht 16 Zij $A_n := \{v \in \mathbb{Z}^{n+1} \mid \sum_{i=1}^{n+1} v_i = 0\}$ het wortelrooster van type `A` en dimensie n.

(i) Bepaal de minimale vectoren van A_n.

(ii) De symmetrische groep S_{n+1} werkt op de vectoren van $\mathbb{R}^{n+1}$ door permutatie van de componenten. Laat zien dat deze werking automorfismen van A_n induceert en concludeer dat $Aut(A_n)$ een ondergroep isomorf met S_{n+1} heeft.

(iii) Ga na dat de minimale vectoren onder deze werking van S_{n+1} in een baan liggen, dus dat S_{n+1} *transitief* op de minimale vectoren werkt.

(iv) Bewijs dat $Aut(A_n) \cong C_2 \times S_{n+1}$ voor $n \geq 2$, waarbij de C_2 door de *centrale inversie* $-\mathbb{I}$ voortgebracht is.

Hint: Uit (ii) volgt dat $Aut(A_n)$ een ondergroep isomorf met $C_2 \times S_{n+1}$ heeft. Er moet dus aangetoond worden dat dit al de volledige automorfisme groep is. Uit (i) en (iii) volgt de lengte van de baan van de eerste basisvector, bijvoorbeeld $e_1 - e_2$. Pas nu de baanstelling toe die zegt dat $|G| = |x^G| \cdot |G_x|$, waarbij x^G de baan van x onder G is en G_x de stabilisator van x in G. Bepaal nu de lengte van de baan van de tweede basisvector (bijvoorbeeld $e_1 - e_3$) onder de stabilisator van $e_1 - e_2$ en itereer dit proces.

•

3.3 LLL-reductie

Door A.K. Lenstra, H.W. Lenstra en L. Lovász is in 1982 in een articel over factorisatie van veeltermen een nieuwe definitie van gereduceerde basis voorgesteld, die aan de ene kant goede eigenschappen heeft en aan de andere kant ook efficiënt berekenbaar is. Volgens de initialen van de drie mensen heet deze eigenschap van een basis nu *LLL-gereduceerd* of *L^3-gereduceerd*.

Voor dat we aan de definitie van een LLL-gereduceerde basis toe komen, herhalen we eerst de *Gram-Schmidt orthogonalisatie* die uit een willekeurige basis een basis van loodrecht op elkaar staande vectoren maakt.

3.3.1 Gram-Schmidt orthogonalisatie

Gegeven een basis $(b_1, \ldots, b_n)$ van een vectorruimte V, bepalen we een nieuwe basis $(b_1^*, \ldots, b_n^*)$ van V die uit orthogonale vectoren bestaat, d.w.z. waarvoor $b_i^* \cdot b_j^* = 0$ voor $i \neq j$. Vaak wordt de nieuwe basis achteraf nog zo aangepast dat de lengte van de vectoren 1 is, om een *orthonormale* basis te krijgen, maar dat hebben we hier niet nodig.

Het idee bij de orthogonalisering is heel simpel: Als $b_1^*, \ldots, b_{i-1}^*$ al gevonden zijn, trekken we van b_i zijn projectie in de deelruimte opgespannen door $b_1^*, \ldots, b_{i-1}^*$ af, de resterende vector staat dan loodrecht op deze deelruimte. Hieruit krijgen we de volgende formule voor b_i^*:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \quad \text{met} \quad \mu_{ij} = \frac{b_i \cdot b_j^*}{\|b_j^*\|^2}.$$

> Herinnering: Voor de projectie $v_{||}$ van een vector v in de richting van een vector w geldt (volgens Pythagoras): $v_{||} = \|v\| \cos(\varphi) \frac{w}{\|w\|}$, waarbij φ de hoek tussen de twee vectoren is. Maar $\cos(\varphi)$ wordt met behulp van het inproduct uitgedrukt door $\cos(\varphi) = \frac{v \cdot w}{\|v\| \, \|w\|}$, dus is $v_{||} = \frac{v \cdot w}{\|w\|^2} w$.

Het is duidelijk dat bij de Gram-Schmidt orthogonalisering geldt dat $b_1^* = b_1$. Verder is b_i^* de orthogonale projectie van b_i op $\langle b_1^*, \ldots, b_{i-1}^* \rangle^\perp = \langle b_1, \ldots, b_{i-1} \rangle^\perp$.

3.1 Gevolg *(Ongelijkheid van Hadamard)*
Voor een $n \times n$-matrix $A = (a_{ij})$ geldt:

$$\det(A)^2 \leq \prod_{i=1}^{n} (\sum_{j=1}^{n} a_{ij}^2).$$

BEWIJS: Zij $B = (b_1, \ldots, b_i)$ de basis met b_i de i-de kolom van A. Verder zij $B^* = (b_1^*, \ldots, b_n^*)$ de Gram-Schmidt orthogonalisatie van B. Voor de Gram matrix F^* van B^* geldt dat $\det(F^*) = \det(A^{tr} A)$, want in het orthogonalisatie proces worden alleen maar transformaties met determinant 1 toegepast. Omdat de basis B^* orthogonaal is, geldt $\det(F^*) = \prod_{i=1}^{n} \|b_i^*\|^2$.

Maar de b_i^* zijn orthogonale projecties van de b_i, daarom geldt $\|b_i^*\|^2 \leq \|b_i\|^2$. Hieruit volgt

$$\det(A)^2 = \det(A^{tr}A) = \det(F^*) = \prod_{i=1}^{n} \|b_i^*\|^2 \leq \prod_{i=1}^{n} \|b_i\|^2 = \prod_{i=1}^{n} (\sum_{j=1}^{n} a_{ij}^2).$$

□

3.3.2 LLL-gereduceerde bases

We behouden de notaties van de Gram-Schmidt orthogonalisering.

3.2 Definitie Een roosterbasis $B = (b_1, \ldots, b_n)$ van een rooster L heet *LLL-gereduceerd* als geldt:

(i) $|\mu_{ij}| \leq \frac{1}{2}$ voor alle $1 \leq j < i \leq n$;

(ii) $\|b_i^*\|^2 + \mu_{i,i-1}^2 \|b_{i-1}^*\|^2 = \|b_i^* + \mu_{i,i-1} b_{i-1}^*\|^2 \geq \frac{3}{4}\|b_{i-1}^*\|^2$.

Conditie (ii), soms ook de *Lovász-conditie* geheten, zegt dat de projectie van b_i op $\langle b_1, \ldots, b_{i-2}\rangle^\perp$ niet veel korter is dan de projectie van b_{i-1}.

3.3 Opmerking De constante $\frac{3}{4}$ in (ii) kan worden vervangen door een willekeurige constante α in het open interval $(\frac{1}{4}, 1)$. In principe is ook de waarde $\alpha = 1$ mogelijk, maar dan laat zich niet meer aantonen dat de algoritme voor het berekenen van een LLL-gereduceerde basis in polynomiale tijd stopt.

3.4 Stelling *Voor een LLL-gereduceerde basis $B = (b_1, \ldots, b_n)$ van L met Gram matrix F geldt:*

(i) $\|b_j\|^2 \leq 2^{i-1}\|b_i^*\|^2$ *voor* $1 \leq j \leq i \leq n$;

(ii) $\det(F) \leq \prod_{i=1}^{n} \|b_i\|^2 \leq 2^{\frac{n(n-1)}{2}} \det(F)$;

(iii) $\|b_1\|^2 \leq 2^{\frac{n-1}{2}} \det(F)^{\frac{1}{n}}$;

(iv) $\|b_1\|^2 \leq 2^{n-1}\|x\|^2$ *voor alle* $0 \neq x \in L$.

Is B LLL-gereduceerd met betrekking tot een factor $\alpha \in (\frac{1}{4}, 1)$ in plaats van $\frac{3}{4}$, moet men in (i)-(iv) de factor 2 door $\frac{4}{4-\alpha} = \frac{1}{\alpha - \frac{1}{4}}$ vervangen.

Bewijs: (i): Omdat $|\mu_{ij}| \leq \frac{1}{2}$, geldt $\frac{3}{4}\|b_{i-1}^*\|^2 \leq \|b_i^*\|^2 + \frac{1}{4}\|b_{i-1}^*\|^2$ en hieruit volgt $\|b_{i-1}^*\|^2 \leq 2\|b_i^*\|^2$. Per inductie volgt hieruit $\|b_j^*\|^2 \leq 2^{i-j}\|b_i^*\|^2$.

Nu hebben we $\|b_i\|^2 = \|b_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 \|b_j^*\|^2 \leq \|b_i^*\|^2(1 + \frac{1}{4}\sum_{j=1}^{i-1} 2^{i-j}) = \|b_i^*\|^2(1 + \frac{1}{4}(2^i - 1)) \leq \|b_i^*\|^2 2^{i-1}$. Dit toegepast op b_j en gecombineerd met $\|b_j^*\|^2 \leq 2^{i-j}\|b_i^*\|^2$ geeft $\|b_j\|^2 \leq 2^{j-1}\|b_j^*\|^2 \leq 2^{j-1}\,2^{i-j}\|b_i^*\|^2 = 2^{i-1}\|b_i^*\|^2$.
(ii): De linkerkant is juist de ongelijkheid van Hadamard. Voor de rechterkant volgt met behulp van (i): $\det(F) = \prod_{i=1}^{n} \|b_i^*\|^2 \geq \prod_{i=1}^{n} 2^{-(i-1)}\|b_i\|^2 = 2^{-\sum_{i=1}^{n}(i-1)} \prod_{i=1}^{n} \|b_i\|^2 = 2^{-\frac{n(n-1)}{2}} \prod_{i=1}^{n} \|b_i\|^2$.

(iii): Dit volgt meteen uit (i): $\|b_1\|^{2n} \leq \prod_{i=1}^{n} 2^{i-1}\|b_i^*\|^2 = 2^{\frac{n(n-1)}{2}} \prod_{i=1}^{n} \|b_i^*\|^2 = 2^{\frac{n(n-1)}{2}} \det(F)$.
(iv): Zij $x = \sum_{i=1}^{n} c_i b_i \in L$ met $c_i \in \mathbb{Z}$, dan is $x = \sum_{i=1}^{n} a_i b_i^*$ en $\|x\|^2 = \sum_{i=1}^{n} a_i^2 \|b_i^*\|^2$. Voor de hoogste index i met $c_i \neq 0$ geldt volgens de constructie van de b_i^* dat $a_i = c_i$, dus is $\|x\|^2 \geq c_i^2\|b_i^*\|^2 \geq \|b_i^*\|^2 \geq 2^{-(i-1)}\|b_1\|^2 \geq \frac{1}{2^{n-1}}\|b_1\|^2$. □

Punt (iv) van deze stelling zegt dat de kortste vector van een LLL-gereduceerde basis niet willekeurig veel langer dan een minimale vector kan zijn. In de praktijk zit dit meestal veel beter, in het algemeen zijn de vectoren in een LLL-gereduceerde basis niet veel langer dan de minimale vectoren. Er is echter een andere reden waarom de afschatting uit (iv) nuttig is: Als in een rooster een eenduidige minimale vector (tot op ± na) bestaat en de op deze na korste vector minstens een factor 2^{n-1} langer is, dan ligt de minimale vector altijd in een LLL-gereduceerde basis. Roosters met deze eigenschap spelen in veel toepassingen een belangrijke rol.

Opdracht 17 Twee lineair onafhankelijke vectoren $v, w \in \mathbb{R}^2$ heten *paargewijs gereduceerd* als $\|v\|^2 \leq \|w\|^2$ en $|v \cdot w| \leq \frac{1}{2}\|v\|^2$.

(i) Laat zien dat iteratie van de volgende twee stappen een willekeurige roosterbasis (v, w) van een 2-dimensionaal rooster L in een paarsgewijs gereduceerde basis transformeert:

   (a) Als $\|v\| > \|w\|$, verruil v en w.

   (b) Vervang w door $w - \lfloor \frac{v \cdot w}{\|v\|^2} \rceil v$.
   (Met $\lfloor x \rceil$ noteren we het gehele getal dat het dichtst bij x ligt, voor $x \in \frac{1}{2} + \mathbb{Z}$ wordt meestal in de richting van 0 afgerondt.)

(ii) Beschrijf de meetkundige betekenis van de reductie stap (b).

(iii) Laat zien dat paargewijs gereduceerde vectoren LLL-gereduceerd zijn.

(iv) Bewijs dat de kortere vector v van een paarsgewijs gereduceerde basis noodzakelijk een minimale vector van L is.

•

3.4 Het LLL-algoritme

Tot nu toe hebben we alleen maar eigenschappen van een LLL-gereduceerde basis genoemd, zonder te weten of zo'n basis überhaupt bestaat. Het vordeel tegenover andere definities van gereduceerde bases is echter, dat er een algoritme bestaat waarmee een LLL-gereduceerde basis gevonden wordt en dat deze algoritme zelfs relatief snel is (in polynomiale tijd in de dimensie van het rooster).

Het idee van het algoritme dat een LLL-gereduceerde basis oplevert is eigenlijk heel simpel. Men gaat ervan uit dat de vectoren $(b_1, \ldots, b_{k-1})$ al een

LLL-gereduceerde basis van het door deze vectoren voortgebrachte deelrooster zijn. Vervolgens probeert men deze basis voort te zetten met de basisvector b_k zo dat ook $(b_1, \ldots, b_k)$ weer LLL-gereduceerd zijn. Hierbij zijn er twee hoofdstappen:

(i) Zorg ervoor dat $|\mu_{kj}| \leq \frac{1}{2}$ voor $j < k$ door b_k door $b_k - qb_j$ voor een geschikte q te vervangen.

(ii) Als aan de Lovász conditie voldaan is, dus als $\|b_k^*\|^2 + \mu_{k,k-1}^2 \|b_{k-1}^*\|^2 \geq \frac{3}{4}\|b_{k-1}^*\|^2$, is $(b_1, \ldots, b_k)$ LLL-gereduceerd en we kunnen met k een stap verder gaan, dus $k \to k+1$ zetten.

Als dit niet het geval is, dus als $\|b_k^*\|^2 + \mu_{k,k-1}^2 \|b_{k-1}^*\|^2 < \frac{3}{4}\|b_{k-1}^*\|^2$, verruilen we b_k en b_{k-1} en gaan met k een stap terug, d.w.z. we zetten $k \to k-1$, want nu is alleen maar bekend dat $(b_1, \ldots, b_{k-2})$ LLL-gereduceerd is.

We bekijken nu de details van het algoritme. Hiervoor gebruiken we als nieuwe notatie $B_i := \|b_i^*\|^2$, dan geldt $b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*$ met $\mu_{ij} = \frac{b_i \cdot b_j^*}{B_j}$. Omdat b_j^* de orthogonale projectie van b_j is, geldt in het bijzonder $b_j \cdot b_j^* = B_j$.

Stap 1:
Stel dat $(b_1, \ldots, b_{k-1})$ LLL-gereduceerd is. We moeten testen of $|\mu_{kj}| \leq \frac{1}{2}$ is. Dit doen we van achter naar voren, d.w.z. we beginnen met $\mu_{k,k-1}$ en kijken dan naar $\mu_{k,k-2}$ enzovoorts. De reden hiervoor is dat de μ_{kj} met $j > l$ niet veranderen als we b_k door $b_k - qb_l$ vervangen, want $b_l \cdot b_j^* = 0$ voor $j > l$.

Zij dus l de grootste index waarvoor $|\mu_{kl}| > \frac{1}{2}$. We definiëren $q := \lfloor \mu_{kl} \rceil$ en vervangen b_k door $b_k' := b_k - qb_l$. De nieuwe μ_{kl}' vinden we door

$$\mu_{kl}' = \frac{(b_k - qb_l) \cdot b_l^*}{B_l} = \mu_{kl} - q\frac{b_l \cdot b_l^*}{B_l} = \mu_{kl} - q.$$

Verder moeten we ook de μ_{kj} voor $j < l$ aanpassen, hiervoor krijgen we:

$$\mu_{kj}' = \frac{(b_k - qb_l) \cdot b_j^*}{B_j} = \mu_{kj} - q\frac{b_l \cdot b_j^*}{B_j} = \mu_{kl} - q\mu_{kj}.$$

Merk op dat bij deze stap de vector b_k^* onveranderd blijft, omdat de verandering van b_k in de deelruimte opgespannen door $(b_1, \ldots, b_{k-1})$ ligt.

Stap 2:
Stel nu dat $(b_1, \ldots, b_{k-1})$ LLL-gereduceerd zijn, dat $|\mu_{kj}| \leq \frac{1}{2}$, maar dat $B_k + \mu_{k,k-1}^2 B_{k-1} \geq \frac{3}{4} B_{k-1}$. In deze situatie verruilen we b_k en b_{k-1} en moeten hiervoor nu de nieuwe $b_k^{*\prime}$, ${b_{k-1}^*}'$ en μ_{ij}' bepalen. Om de notatie te vereenvoudigen, schrijven we kort μ voor $\mu_{k,k-1}$.

Er geldt

$$ {b_{k-1}^*}' = b_k - \sum_{i=1}^{k-2} \mu_{ki} b_i^* = b_k^* + \mu b_{k-1}^*$$

en dit geeft $B_{k-1}' = B_k + \mu^2 B_{k-1}$, omdat $b_k^* \perp b_{k-1}^*$. Omdat we dit nog vaker nodig zullen hebben, noteren we kort $B := B_k + \mu^2 B_{k-1}$.

Verder zien we rechtstreeks in dat

$$\mu'_{k-1,i} = \mu_{ki} \text{ voor } i \leq k-2.$$

Voor $b_k^{*\prime}$ krijgen we

$$b_k^{*\prime} = b_{k-1}^* - \frac{b_{k-1} \cdot b_{k-1}^{*\prime}}{B'_{k-1}} b_{k-1}^{*\prime} = b_{k-1} - \sum_{i=1}^{k-2} \mu_{k-1,i} b_i^* - \frac{b_{k-1} \cdot b_{k-1}^{*\prime}}{B'_{k-1}} b_{k-1}^{*\prime}.$$

Dit geeft

$$\mu'_{ki} = \mu_{k-1,i} \text{ voor } i \leq k-2$$

en

$$\mu'_{k,k-1} = \frac{b_{k-1} \cdot (b_k^* + \mu b_{k-1}^*)}{B} = \frac{\mu B_{k-1}}{B}$$

want $b_{k-1} \cdot b_k^* = 0$ en $b_{k-1} \cdot b_{k-1}^* = b_{k-1}^* \cdot b_{k-1}^* = B_{k-1}$.

Voor $b_k^{*\prime}$ volgt hieruit dat

$$b_k^{*\prime} = b_{k-1}^* - \frac{\mu B_{k-1}}{B}(b_k^* + \mu b_{k-1}^*) = (1 - \mu^2 \frac{B_{k-1}}{B_k + \mu^2 B_{k-1}}) b_{k-1}^* - \frac{\mu B_{k-1}}{B} b_k^*$$
$$= \frac{B_k}{B} b_{k-1}^* - \frac{\mu B_{k-1}}{B} b_k^*.$$

Voor de norm $B'_k = \|b_k^{*\prime}\|^2$ krijgen we

$$B'_k = \frac{B_k^2 B_{k-1} + \mu^2 B_{k-1}^2 B_k}{B^2} = B_{k-1} B_k \frac{B_k + \mu^2 B_{k-1}}{B^2} = \frac{B_{k-1} B_k}{B}.$$

Tenslotte moeten we nog de nieuwe $\mu'_{i,k-1}$ en μ'_{ik} voor $i > k$ bepalen, want deze zijn veranderd omdat we b_{k-1}^* en b_k^* hebben gewijzigd. Er geldt

$$\mu'_{i,k-1} = \frac{b_i \cdot b_{k-1}^{*\prime}}{B'_{k-1}} = \frac{b_i \cdot b_k^*}{B_k} \frac{B_k}{B} + \mu \frac{b_i \cdot b_{k-1}^*}{B_{k-1}} \frac{B_{k-1}}{B} = \mu_{ik} \frac{B_k}{B} + \mu_{i,k-1} \mu'_{k,k-1},$$

want $\mu'_{k,k-1} = \frac{\mu B_{k-1}}{B}$. Wegens $\frac{B_k}{B} = \frac{B - \mu^2 B_{k-1}}{B} = 1 - \mu^2 \frac{B_{k-1}}{B} = 1 - \mu \mu'_{k,k-1}$ laat zich dit ook schrijven als

$$\mu'_{i,k-1} = \mu_{i,k-1} \mu'_{k,k-1} + \mu_{ik}(1 - \mu \mu'_{k,k-1}).$$

Voor $\mu'_{i,k}$ krijgen we

$$\mu'_{i,k} = \frac{b_i \cdot b_k^{*\prime}}{B'_k} = \frac{B_k}{B} \frac{b_i \cdot b_{k-1}^*}{B_{k-1}} \frac{B_{k-1}}{B'_k} - \frac{\mu B_{k-1}}{B} \frac{b_i \cdot b_k^*}{B_k} \frac{B_k}{B'_k} = \mu_{ik} - \mu \mu_{i,k-1},$$

want $B'_k = \frac{B_{k-1} B_k}{B}$.

We kunnen stap 2 als volgt samenvatten: Zij $B_i := \|b_i^*\|^2$ voor $1 \leq i \leq n$, $\mu := \mu_{k,k-1}$, $B := B_k + \mu^2 B_{k-1}$, dan wordt:

$$\begin{pmatrix} b'_{k-1} \\ b'_k \end{pmatrix} = \begin{pmatrix} b_k \\ b_{k-1} \end{pmatrix}$$

$$\begin{pmatrix} b_{k-1}^{*\prime} \\ b_k^{*\prime} \end{pmatrix} = \begin{pmatrix} \mu & 1 \\ \frac{B_k}{B} & -\mu\frac{B_{k-1}}{B} \end{pmatrix} \cdot \begin{pmatrix} b_{k-1}^* \\ b_k^* \end{pmatrix}$$

$$B'_{k-1} = B \quad \text{en} \quad B'_k = \frac{B_{k-1}B_k}{B}$$

$$\begin{pmatrix} \mu'_{k-1,i} \\ \mu'_{ki} \end{pmatrix} = \begin{pmatrix} \mu_{ki} \\ \mu_{k-1,i} \end{pmatrix} \text{ voor } i \leq k-2$$

$$\mu'_{k,k-1} = \mu\frac{B_{k-1}}{B}$$

$$\begin{pmatrix} \mu'_{i,k-1} \\ \mu'_{ik} \end{pmatrix} = \begin{pmatrix} 1 & \mu'_{k,k-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu \end{pmatrix} \cdot \begin{pmatrix} \mu_{i,k-1} \\ \mu_{ik} \end{pmatrix}$$

Nu dat we het LLL-algoritme in detail hebben bekeken, moeten we nog een laatste stap doen, namelijk aantonen dat het algoritme naar eindig veel stappen stopt. In feite laat zich aantonen dat het aantal stappen van het algoritme begrensd is door een polynoom in de dimensie n van het rooster, maar dit zullen we hier niet verder verdiepen.

3.5 Propositie *Het LLL-algoritme stopt naar eindig veel stappen.*

BEWIJS: Zij F de Gram matrix van L ten opzichte van de roosterbasis $(b_1, \ldots, b_n)$. We definiëren $d_k := \det(F_{ij\,1\leq i,j\leq k}$ als determinant van de links-boven $d \times d$ deelmatrix van F. Volgens het Gram-Schmidt orthogonalisatie proces geldt dan $d_k = \prod_{i=1}^{k} \|b_i^*\|^2$. Volgens de Hermite ongelijkheid is d_k naar beneden begrensd door $d_k \geq \frac{3}{4}^{\frac{k(k-1)}{2}} \min(L)^k$. We bekijken nu de grootheid $D := \prod_{k=1} n - 1 d_k$, dan is ook D naar beneden begrensd.

De waarde van D verandert alleen maar in de stappen waar we b_k en b_{k-1} verruilen. In zo'n geval blijven $d_1, \ldots, d_{k-2}$ onverandert, hetzelfde geldt voor $d_k, \ldots, d_{n-1}$. Het laatste geldt omdat (met de eerder gebruikte notaties) $B'_{k-1} = B$ en $B'_k = \frac{B_{k-1}B_k}{B}$ en dus $B'_{k-1}B'_k = B_{k-1}B_k$. Maar d_{k-1} verandert wel, en omdat niet aan de Lovász conditie voldaan is, geldt $B'_{k-1} = B_k + \mu^2_{k,k-1}B_{k-1} < \frac{3}{4}B_{k-1}$. Dit betekent dat d_k met een factor die kleiner is dan $\frac{3}{4}$ vermenigvuldigd wordt en dus wordt in iedere verruil-stap D met een factor $< \frac{3}{4}$ vermenigvuldigd. Omdat D naar beneden is, kan dit slechts eindig vaak gebeuren. □

Zonder bewijs geven we het originele resultaat van de complexiteits-analyse aan:

3.6 Propositie *(Lenstra, Lenstra, Lovász)*
Zij L een rooster met roosterbasis $(b_1, \ldots, b_n)$ en zij $C \geq 2$ met $\|b_i\|^2 \leq C$ voor $1 \leq i \leq n$. Dan is het aantal bit-operaties dat benodigd wordt om een LLL-gereduceerde basis van L te bepalen $O(n^6 \log(C)^3)$.

We merken nog op dat het LLL-algoritme in de praktijk meestal veel sneller blijkt te zijn dan de complexiteit $O(n^6)$ laat vermoeden.

3.4.1 Het MLLL-algoritme

Een grondige analyse van het LLL-algoritme door M. Pohst heeft tot een iets algemenere versie van het LLL-algoritme geleid, die niet meer veronderstelt dat de gegeven vectoren lineair onafhankelijk zijn, maar ook werkt als de vectoren een afhankelijk stelsel van voortbrengers zijn. Deze versie van het algoritme staat bekend onder de naam MLLL-algoritme (met 'M' voor *modified*).

Zij dus $(b_1, \ldots, b_m)$ een stelsel vectoren dat een rooster L voortbrengt. Als op gegeven moment in het algoritme de vector b_k lineair afhankelijk van de vectoren $b_1, \ldots, b_{k-1}$ is, dan is de orthogonale projectie b_k^* van b_k op $\langle b_1, \ldots, b_{k-1}\rangle^\perp$ de nulvector. In dit geval is dus $b_k^* = 0$ en dus ook $B_k = 0$.

Voor de verdere projecties is b_k^* natuurlijk overbodig, daarom zet men $\mu_{kl} = 0$ voor $l = k+1, \ldots, m$. Verder is $\|b_k^* + \mu_{k,k-1}^2 b_{k-1}^*\|^2 = \mu_{k,k-1}^2 B_{k-1} \leq \frac{1}{4} B_{k-1}$, daarom wordt b_k met b_{k-1} verruild.

Het gemodificeerde algoritme eindigt met een stelsel vectoren $b_1', \ldots, b_m'$ met de volgende eigenschappen:

- De laatste n vectoren $(b_{m-n+1}', \ldots, b_m')$ zijn lineair onafhankelijk en vormen een roosterbasis van L.
- Voor $i \leq m - n$ is $b_i' = 0$.
- De eerste $m - n$ kolommen van de transformatiematrix van het stelsel $(b_1, \ldots, b_m)$ naar $(b_1', \ldots, b_m')$ zijn een $\mathbb{Z}$-basis voor de lineaire afhankelijkheden van de vectoren b_i.

Van wege de laatste eigenschap is het MLLL-algoritme niet alleen maar geschikt om een roosterbasis van een rooster te vinden, maar ook om de $\mathbb{Z}$-basis van de kern van een matrix $A \in \mathbb{Z}^{m\times m}$ te vinden (waarbij de b_i de kolommen van A zijn).

3.5 Toepassingen van LLL-reductie

De oorspronkelijke motivatie voor de LLL-reductie was een algoritme die veeltermen over $\mathbb{Q}$ in polynomiale tijd factoriseert. Maar inmiddels is het LLL-algoritme een van de meestgebruikte algoritmes in de computeralgebra, die alle soorten van toepassingen heeft.

Vaak wordt de reductie bijvoorbeeld toegepast om explosie van getallen tegen te werken. Een voorbeeld hiervoor is de Smith normaal vorm. Al bij matrices van grote 50×50 worden de getallen bij het berekenen van de Smith normaal vorm vaak zo groot (duisenden van cijfers) dat men problemen met het geheugen en in ieder geval met de rekentijd krijgt. Met behulp van LLL-reductie kan men proberen, de getallen na een aantal elementaire transformaties weer kleiner te maken.

Voor dat we naar de factorisatie van veeltermen kijken, zullen we nog een paar andere toepassingen van de LLL-reductie bekijken.

3.5.1 Algebraïsche reconstructie

Een vraagstelling die in de algebraïsche getaltheorie vaak een rol speelt is als volgt: Gegeven een benadering a van een algebraïsch getal α, vind de minimum veelterm van α. (Herinnering: Een algebraïsch getal is een getal α dat de nulpunt van een veelterm met geheeltallige coëfficiënten is, dus dat voldoet aan een vergelijking $\sum_{i=0}^{n} c_i\alpha^i = 0$ met $c_i \in \mathbb{Z}$, $c_n \neq 0$.)

De benadering a is hierbij typisch een element van $\mathbb{R}$, $\mathbb{C}$ of het lichaam der p-adische getallen $\mathbb{Q}_p$.

> De p-adische getallen $\mathbb{Q}_p$ zijn naast $\mathbb{R}$ alternatieve mogelijkheden om de rationale getallen $\mathbb{Q}$ in een volledig lichaam in te bedden, d.w.z. in een lichaam waarin iedere convergente rij een limiet heeft. Bij de reële getallen gebeurt dit formeel door de Cauchy-rijen over $\mathbb{Q}$ modulo de nulrijen te bekijken. Hetzelfde idee wordt ook voor de p-adische getallen toegepast, het verschil is dat de nulrijen met een andere definitie van absolute waarde gedefinieerd worden.
>
> Voor een rationaal getal $a = p^k \frac{m}{n}$ met $p \nmid mn$ heet $\nu_p(a) := k$ de *p-adische valuatie* van a. Men gaat na dat $|a|_p := p^{-\nu_p(a)}$ een absolute waarde op $\mathbb{Q}$ geeft, die aan de gewone eisen voldoet, waarbij we de extra definitie $|0|_p = 0$ (en dus formeel $\nu_p(0) = \infty$) nodig hebben.
>
> We merken op dat de p-adische absolute waarde aan een sterkere driehoeksongelijkheid voldoet, namelijk $|a + b|_p \leq \max(|a|_p, |b|_p)$ met $|a + b|_p = \max(|a|_p, |b|_p)$ als $|a|_p \neq |b|_p$.
>
> Een alternatieve mogelijkheid, om $\mathbb{Q}_p$ te definiëren, is eerst de ring $\mathbb{Z}_p$ van de gehele p-adische getallen te construeren, $\mathbb{Q}_p$ is dan het breukenlichaam van $\mathbb{Z}_p$.
>
> We kunnen een getal $a \in \mathbb{Z}$ in het p-tallig stelsel schrijven als $a = a_0 + a_1 p + \ldots + a_n p^n$ met $a_i \in \{0, \ldots, p-1\}$. We kunnen het getal a dus representeren door de rij $(a_0, a_1, \ldots, a_n, 0, \ldots)$ die slechts eindig veel componenten ongelijk aan 0 heeft. De p-adische gehele getallen $\mathbb{Z}_p$ is nu de verzamelin van alle rijen $(a_0, \ldots, a_n, \ldots)$ met $a_i \in \{0, \ldots, p-1\}$, waarbij ook oneindig veel componenten $\neq 0$ mogen zijn. Het optellen is in principe componentsgewijs, maar als we hiermee buiten de getallen $\{0, \ldots, p-1\}$ vallen, moeten we een *carry* naar de volgende component meenemen, net zo als bij het schrijftelijke optellen. Het vermenigvuldigen werkt net zo als bij machtsreeksen, dus $(a_0, \ldots, a_n, \ldots) \cdot (b_0, \ldots, b_n, \ldots) = (c_0, \ldots, c_n, \ldots)$ met $c_n = \sum_{i=1}^{n} a_i b_{n-1}$, waarbij ook hier achteraf de componenten weer op de waarden $\{0, \ldots, p-1\}$ genormaliseerd moeten worden (door doorschuiven naar hogere componenten, niet door modulo rekenen).
>
> Een getal $(a_0, \ldots, a_n, \ldots)$ heeft valuatie k als $a_0 = \ldots = a_{k-1} = 0$ en $a_k \neq 0$. Hoe meer nullen in het begin, hoe dichter ligt het getal dus bij 0.

De vraag bij de algebraïsche reconstructie is uit de benadering a van α de coëfficiënten c_i te bepalen zo dat $\sum_{i=1}^{n} c_i\alpha^i = 0$. Omdat a een benadering van α is, is dan ook $\sum_{i=1}^{n} c_i a^i \approx 0$.

Een iets algemenere vraag is, voor algebraïsche getallen $\alpha_0, \ldots, \alpha_n$ coëfficiënten c_i te vinden zo dat $\sum_{i=1}^{n} c_i\alpha_i = 0$, dat wil zeggen een lineaire relatie tussen de α_i te vinden. De algebraïsche reconstructie is dan het speciaal geval $\alpha_i = \alpha^i$. We zullen ons hier tot dit speciaal geval beperken, het algemenere geval van de lineaire relaties werkt analoog.

$\mathbb{Q}_p$: Zij α een algebraïsch getal en zij $a \in \mathbb{Q}_p$ een voldoende nauwkeurige benadering van α (dit betekent dat de benadering meer relevante cijfers heeft dan we in het algoritme nodig hebben). We maken nu het rooster

$$L_m := \{v = (c_0, \ldots, c_n) \in \mathbb{Z}^{n+1} \mid \sum_{i=0}^{n} c_i a^i \equiv 0 \bmod p^m\}.$$

Het rooster L_m heeft index p^m in $\mathbb{Z}^{n+1}$, want L_m is de kern van het homomorfisme $(c_0, \ldots, c_n) \mapsto \sum_{i=0}^{n} c_i a^i \bmod p^m$. De vectoren $v_0 = \begin{pmatrix} p^m \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, $v_1 = \begin{pmatrix} -a \\ 1 \\ 0 \\ \vdots \end{pmatrix}$, $v_2 = \begin{pmatrix} -a^2 \\ 0 \\ 1 \\ \vdots \end{pmatrix}$, …, $v_n = \begin{pmatrix} -a^n \\ 0 \\ \vdots \\ 1 \end{pmatrix}$ liggen in L_m en uit de driehoeksvorm van deze vectoren volgt rechtstreeks dat het rooster opgespannen door $(v_0, \ldots, v_n)$ index p^m in $\mathbb{Z}^{n+1}$ heeft, dus is $(v_0, \ldots, v_n)$ een roosterbasis van L_m.

Het idee om LLL-reductie op dit rooster toe te passen is nu als volgt: De vector $v = (c_0, \ldots, c_n)$ met de coëfficiënten c_i van de minimum veelterm van α ligt in L_m voor iedere m. Als n de juiste graad van α over $\mathbb{Q}$ is (dus de graad van de minimum veelterm) zijn de vectoren die in L_m liggen en lineair onafhankelijk van v zijn slechts virtuele afhankelijkheden van de a^i en naarmate m groeit, worden de normen van deze virtuele afhankelijkheden steeds groter. Voor voldoende grote m zijn dus $\pm v$ de vectoren van minimale lengte in L_m en alle andere vectoren in L_m hebben normen die veel groter zijn dan die van v.

Uit de eigenschappen van een LLL-gereduceerde basis volgt, dat de vector v vanaf een zekere m gegarandeerd door het LLL-algoritme gevonden wordt, maar in de praktijk gebeurt dit al veel vroeger dan theoretisch bewijsbaar.

Als n kleiner dan de graad van α is, wordt nooit een vector gevonden, die veel korter is dan de andere vectoren in een LLL-gereduceerde basis, en in zo'n geval zal men een grotere n proberen.

Als n groter is dan de graad d van α, krijgt men in plaats van een eenduidige minimale vector $n - d + 1$ korte vectoren, want de vectoren $(c_0, \ldots, c_d, 0, \ldots, 0)$, …, $(0, \ldots, 0, c_0, \ldots, c_d)$ zijn alle kort.

Als de graad van α niet bekend is, zal men dus eerste kleine graden n proberen en deze verhogen als geen korte vectoren gevonden worden.

$\mathbb{R}$: In principe werkt de algebraïsche reconstructie over $\mathbb{R}$ bijna hetzelfde als over $\mathbb{Q}_p$. We vermenigvuldigen de a^i met een hoge macht $N = 10^s$, ronden Na^i vervolgens op gehele getallen af en bekijken het rooster met basis $v_0 = \begin{pmatrix} N \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, $v_1 = \begin{pmatrix} -Na \\ 1 \\ 0 \\ \vdots \end{pmatrix}$, $v_2 = \begin{pmatrix} -Na^2 \\ 0 \\ 1 \\ \vdots \end{pmatrix}$, $\ldots$, $v_n = \begin{pmatrix} -Na^n \\ 0 \\ \vdots \\ 1 \end{pmatrix}$. Een korte vector in dit rooster moet noodzakelijk 0 in de eerste component hebben en geeft dus een lineaire relatie tussen de a^i.

3.5.2 Simpele factorisatie van veeltermen over $\mathbb{Q}$ ($\mathbb{Z}$)

Zij $f \in \mathbb{Q}[X]$ een rationale veelterm, dan proberen we f in irreducibele factoren te ontbinden. Het is natuurlijk voldoende, als we een reducibele f in twee factoren $f = gh$ kunnen opsplitsen, iteratie geeft dan uiteindelijk irreducibele factoren.

We bepalen nu een (numerieke) benadering a van een nulpunt α van f over $\mathbb{R}$, $\mathbb{C}$ of $\mathbb{Q}_p$. Als f reducibel is, is $f = gh$ met $\deg(g) \geq 1$ en $\deg(h) \geq 1$. Uit $f(\alpha) = 0$ volgt $g(\alpha) = 0$ of $h(\alpha) = 0$. In dit geval is de minimum veelterm van α een deler van g of h en dus in het bijzonder een niet-triviale deler van f.

Met de methode van de algebraïsche reconstructie laat zich de minimum veelterm van α uit de benadering a bepalen en we vinden zo een factor van f.

3.5.3 Factorisatie van veeltermen over $\mathbb{Q}$ in polynomiale tijd

De algemene strategie voor de factorisatie van veeltermen over $\mathbb{Q}$ bevat drie hoofstappen:

(1) Bepaal een factorisatie van f modulo p, d.w.z. behandel f als een veelterm over het eindige lichaam $\mathbb{F}_p$. Voor veeltermen over eindige lichamen zijn er efficiënte methoden, vooral de algorithmen van Berlekamp en van Cantor-Zassenhaus.

(2) Verbeter de in (1) gevonden factorisatie tot een factorisatie modulo p^m voor een voldoende grote m. Dit gebeurt met de methode van *Hensel lift*.

(3) Probeer producten van de in (2) gevonden factoren te vinden, die echte factoren van f zijn. Het cruciale punt is dat de coëfficiënten van factoren van f afhankelijk van de coëfficiënten van f begrensd zijn door de *Mignotte grens* en voor p^m duidelijk groter dan deze grens moet het product dus tot relatief kleine coëfficiënten leiden.

Het probleem bij deze aanpak is vooral stap (3), deze kan namelijk tot een combinatorische explosie leiden. Als een veelterm van graad 100 bijvoorbeeld twee irreducibele factoren van graad 50 heeft, maar modulo p^m in 50 factoren van graad 2 opsplitst moeten zelfs in het geval dat de graad van de irreducibele

factoren bekend is $\binom{50}{25} \approx 1.3 \cdot 10^{14}$ mogelijke producten geprobeerd worden. Dit probleem van combinatorische explosie was de reden dat er voor de toepassing van de LLL-reducite op dit probleem geen algoritme bekend was, die veeltermen over $\mathbb{Q}$ in polynomiale tijd factoriseert.

We zullen nu de drie stappen van de factorisatie nader toelichten:

Stap 1:
We kiezen een priemgetal p die de *discriminant* $d(f)$ van f niet deelt. De discriminant van f laat zich uit de coëfficiënten van f berekenen, en de priemdelers van $d(f)$ geven (onder meer) aan, waar bij het liften van irreducibele factoren modulo p tot factoren modulo p^m problemen kunnen onstaan.

Voor het gemak noteren we de reductie van f modulo p weer met f. We mogen ervan uitgaan dat f kwadraatvrij is, d.w.z. dat voor $g \mid f$ geldt dat $g^2 \nmid f$. Dit is geen beperking, want meervoudige factoren van f zijn ook factoren van de (formele) afgeleide f' en dus ook van $\text{ggd}(f, f')$. Door f door $\text{ggd}(f, f')$ te delen wordt f dus kwadraatvrij gemaakt.

Voor het eindig lichaam $\mathbb{F}_{p^k}$ met p^k elementen geldt dat $\mathbb{F}_{p^k}$ precies de nulpunten van de veelterm $X^{p^k} - X$ bevat. Maar de wortels van een irreducibele veelterm g van graad k over $\mathbb{F}_p$ liggen in $\mathbb{F}_{p^k}$, dus is g een deler van $X^{p^k} - X$. Hieruit volgt dat $\text{ggd}(f, X^{p^k} - X)$ het product van alle irreducibele delers van f is die graad een deler van k hebben.

Door $\text{ggd}(f, X^{p^k} - X)$ voor opstijgende $k = 1, 2, \ldots$ te berekenen, wordt f opgesplitst in producten van irreducibele factoren van dezelfde (bekende) graad.

We gaan nu ervan uit dat f een product van m irreducibele factoren van graad k is, waarbij mk de graad van f is. Dit betekent dat $\mathbb{F}_p[X]/(f) \cong \underbrace{\mathbb{F}_{p^k} \oplus \ldots \oplus \mathbb{F}_{p^k}}_{m}$. We beschrijven nu de methode van Cantor-Zassenhaus die behalve voor kleine priemgetallen p efficiënter is dan de methode van Berlekamp.

We kiezen een willekeurige veelterm $t \in \mathbb{F}_p[X]$, dan heeft t een projectie in elke van de m componenten van de directe som, en de projectie van t^{p^k-1} in iedere componente is 1. Omdat de multiplicatieve groep van $\mathbb{F}_{p^k}$ cyklisch is, zijn de projecties van $t^{\frac{p^k-1}{2}}$ in de componenten ± 1 met kans $\frac{1}{2}$ voor de twee waarden. We kunnen dus $t^{\frac{p^k-1}{2}}$ zien als een element van de vorm $(\pm 1, \ldots, \pm 1)$ waarbij in iedere componente de waarde -1 met kans $\frac{1}{2}$ aangenomen wordt. Als we nu $\text{ggd}(f, t^{\frac{p^k-1}{2}} - 1)$ berekenen, levert dit f op, als alle componenten $+1$ waren, en 1 als alle componenten -1 waren. In alle andere gevallen is de *ggd* een echte deler van f, en dit gebeurt met kans $1 - (\frac{1}{2})^{m-1}$, voor twee factoren dus nog steeds met kans $\frac{1}{2}$. Door een paar toevallig gekozen veeltermen t te proberen, wordt dus snel een echte factor gevonden.

Stap 2:
Stel we hebben in stap 1 een factorisatie $f = g_1 h_1$ modulo p gevonden. We proberen deze factorisatie nu tot een factorisatie $f = g_m h_m$ modulo p^m te verbeteren. We mogen weer ervan uitgaan dat f geen meervoudige factoren heeft, daarom kunnen we ook veronderstellen dat $\text{ggd}(g_1, h_1) = 1$ is over $\mathbb{F}_p$.

Met behulp van het uitgebreide algoritme van Euclides vinden we cofactoren u en v met $ug_1 + vh_1 \equiv 1 \bmod p$.

We bekijken nu hoe we een factorisatie $f \equiv g_k h_k \bmod p^k$ tot een factorisatie modulo p^{k+1} kunnen verbeteren, waarbij we veronderstellen dat $g_k \equiv g_1 \bmod p$ en $h_k \equiv h_1 \bmod p$ en dus ook $ug_k + vh_k \equiv 1 \bmod p$. We definiëren de foutterm r_k door $p^k r_k := f - g_k h_k$, dan is $g_k h_k = f - p^k r_k$. Het idee is nu g_{k+1} en h_{k+1} te definiëren door $g_{k+1} := g_k + p^k x$ en $h_{k+1} := h_k + p^k y$ en x en y zo te kiezen dat de $f \equiv g_{k+1} h_{k+1} \bmod p^{k+1}$. Er geldt $(g_k + p^k x)(h_k + p^k y) \equiv g_k h_k + p^k(yg_k + xh_k) \equiv f - p^k r_k + p^k(yg_k + xh_k) \bmod p^{2k}$. We moeten dus x en y zo kiezen dat $yg_k + xh_k = r_k \bmod p$. Hiervoor vermenigvuldigen we $ug_k + vh_k \equiv 1 \bmod p$ met r_k, hieruit krijgen we $y = r_k u$ en $x = r_k v$. Als de graad $\deg(x) \geq \deg(g_k)$, moeten we x nog met rest door g_k delen, dit geeft $x = r_k v + w g_k$ en $y = r_k u - w h_k$ (het tweede moet noodzakelijk zo zijn).

Merk op dat we in feite kwadratische convergentie kunnen bereiken, door in ieder stap de cofactoren u en v zo te verbeteren dat $ug_k + vh_k \equiv 1 \bmod p^k$, dan klopt de nieuwe factorisatie inderdaad modulo p^{2k}.

Stap 3:
In deze stap is het optreden van de LLL-reductie. Het idee is, voor een factor g_0 van f modulo p^m niet alle andere factoren modulo p^m te proberen om zo een product $g = g_0 h_0$ te vinden dat een echte factor van f is, maar om g rechtstreeks uit g_0 te construeren.

We veronderstellen de volgende situatie: Zij $f \in \mathbb{Z}[X]$ een veelterm van graad n en zij $g_0 \in \mathbb{Z}[X]$ een veelterm met de volgende eigenschappen:

(i) g_0 heeft kopcoëfficiënt 1;

(ii) $(g_0 \bmod p^m)$ is een deler van $(f \bmod p^m)$, d.w.z. er bestaat een $h_0 \in \mathbb{Z}[X]$ met $f \equiv g_0 h_0 \bmod p^m$;

(iii) $(g_0 \bmod p)$ is irreducibel in $\mathbb{F}_p[X]$;

(iv) $(g_0 \bmod p)^2 \nmid (f \bmod p)$ in $\mathbb{F}_p[X]$.

Dan geldt de volgende stelling (die we hier niet gaan bewijzen):

3.7 Propositie *De veelterm f heeft een (tot op ± 1 na) eenduidige irreducibele factor $g \in \mathbb{Z}[X]$ waarvoor geldt dat $(g_0 \bmod p) \mid (g \bmod p)$. Verder zijn voor een factor $h \in \mathbb{Z}[X]$ van f de volgende uitspraken equivalent:*

(i) $(g_0 \bmod p) \mid (h \bmod p)$ *in* $\mathbb{F}_p[X]$*;*

(ii) $(g_0 \bmod p^m) \mid (h \bmod p^m)$ *in* $(\mathbb{Z}/p^m\mathbb{Z})[X]$*;*

(iii) $g \mid h$ *in* $\mathbb{Z}[X]$.

De vraag is nu, voor een gegeven g_0 die f modulo p^m deelt, een veelterm g te vinden zo dat g_0 modulo p^m een deler van g is en g tegelijkertijd een deler van f in $\mathbb{Z}[X]$. Maar de delers van f hebben begrensde coëfficiënten, terwijl de veelvouden van g_0 die geen delers van f zijn voor groeiende m steeds grotere coëfficiënten hebben.

We veronderstellen nu dat we de graad van g kennen en definiëren $l := \deg(g)$. In de praktijk is dit natuurlijk meestal niet het geval, maar we kunnen altijd met een lage graad l beginnen en deze verhogen als we geen deler vinden. Verder zij $l_0 := \deg(g_0)$. Dan maken we het volgende rooster aan, waarbij we veeltermen $\sum_{i=0}^{k} c_i X^i$ met vectoren $(c_0, \ldots, c_k)$ identificeren:

$$L := \{u \in \mathbb{Z}[X] \mid \deg(u) \leq l, (g_0 \bmod p^m) \mid (u \bmod p^m)\}.$$

Een roosterbasis voor L is $\{p^m X^j \mid 0 \leq j \leq l_0\} \cup \{g_0 X^j \mid 0 \leq j \leq l - l_0\}$.

Merk op dat de elementen van de roosterbasis grote lengtes hebben, waarbij we de lengte van een veelterm $f = \sum_{i=0}^{k} c_i X^i$ met $|f| := \sqrt{\sum_{i=0}^{k} c_i^2}$ definiëren. Voor voldoende grote m vinden we de factor g van f met $(g_0 \pmod{p^m}) \mid (g \pmod{p^m})$ met behulp van LLL-reductie als een vector van minimale lengte in L. Preciezer geldt (met de notaties van boven):

3.8 Propositie *Als $b \in L$ met $|f|^l \cdot |b|^n < p^{ml_0}$, dan geldt $g \mid b$ in $\mathbb{Z}[X]$, in het bijzonder is* $\gcd(f, b) \neq 1$.

Voor $l < n$ wordt met deze methode dus een niet-triviale factor van f gevonden. Omdat de LLL-reductie in polynomiale tijd loopt, levert deze aanpak een algoritme die ook stap 3 van de boven beschreven methode in polynomiale tijd uitvoert. Stappen 1 en 2 zijn sowieso polynomiaal.

> Alhoewel de factorisatie methode middels LLL-reductie qua complexiteit beter is dan het proberen of een product van factoren modulo p^m een echte factor is, is deze simplere methode in de praktijk meestal sneller. Hierbij wordt echter gebruik gemaakt van een iets slimmere aanpak, namelijk er wordt naar factorisaties modulo verschillende priemgetallen gekeken. De graad van een factor over $\mathbb{Z}$ moet namelijk de som van graden van factoren modulo p voor elk priemgetal zijn, en hiermee laten zich veel combinaties rechtstreeks uitsluiten.