Einführung in die Computeralgebra
Universität Saarbrücken
WS 91/92

Johannes Buchmann
Volker Müller

28. Januar 1994

Inhaltsverzeichnis

Kapitel 1

Einleitung

Während es in der **Numerik** oder der **Praktischen Mathematik** darum geht, lineare Gleichungssysteme, Differentialgleichungen, Optimierungsprobleme usw. **näherungsweise** zu lösen, geht es in der Computeralgebra darum, algorithmische Probleme, vorzugsweise aus der Algebra, aber auch aus der algebraischen Theorie der Differentialgleichungen, **exakt** zu lösen. Man kann beispielsweise die Frage stellen, welche Lösungen die quadratische Gleichung

$$x^2 - 2x - 1 = 0$$

besitzt. Nun gilt $x^2 - 2x - 1 = (x-1)^2 - 2$ und damit sind die Lösungen dieser Gleichung $x_0 = 1 + \sqrt{2}$ und $x_1 = 1 - \sqrt{2}$. Der Numeriker würde die Lösungen typischerweise mit $x_0 = 2.41442135623\ldots$ und $x_1 = -0.41421356237\ldots$ angeben, also eine angenäherte Lösung mit einer bestimmten Präzision bestimmen. In der Computeralgebra dagegen gibt man sich mit den Lösungen $x_0 = \sqrt{2} + 1$ und $x_1 = -\sqrt{2} + 1$ zufrieden. Man muß also eine **symbolische** Darstellung der Lösungen erlauben, in unserem Fall die Verwendung des Symbols $\sqrt{2}$.

In der Vorlesung wird oft der Begriff des **Algorithmus** verwendet. Leider existiert keine zufriedenstellende und mathematisch exakte Fassung dieses Begriffes, der für die Computeralgebra adäquat wäre. Wir begnügen uns daher mit einer informellen Beschreibung des Begriffes:

Ein **Algorithmus** ist eine endliche Folge von Anweisungen, die bei Eingabe einer Folge ganzer Zahlen, der **Eingabe** des Algorithmus, eine Folge ganzer Zahlen ausgibt. Diese Folge nennen wir die **Ausgabe** des Algorithmus. **Rechenzeit** und **Platzbedarf** eines Algorithmus werden in Abhängigkeit von der **Eingabelänge** angegeben. Die (binäre) Länge einer ganzen Zahl $a \neq 0$ ist $\lfloor \log_2 |a| \rfloor + 1$. Die Länge einer Eingabefolge ist die Summe der Längen ihrer Elemente.

Kapitel 2

Algorithmen für ganze Zahlen

2.1 Elementare Operationen für ganze Zahlen

In diesem Abschnitt behandeln wir die Addition, Subtraktion, Multiplikation und Division von ganzen Zahlen. Dazu müssen wir zuerst eine Darstellung von ganzen Zahlen bestimmen. Im folgenden stellen wir ganze Zahlen immer binär dar. Ein String

$$(u_1 u_2 \dots u_n) \in \{0,1\}^n$$

stellt dabei die Zahl

$$\sum_{i=1}^{n} u_i \cdot 2^{n-i}$$

dar. Wir wollen im folgenden ganze Zahlen mit ihrer Darstellung identifizieren. Dann addiert der folgende Algorithmus zwei ganze Zahlen u und v.

2.1. Algorithmus (Addition ganzer Zahlen)

Eingabe: $u = (u_1 u_2 \dots u_n),\ v = (v_1 \dots v_n) \in \{0,1\}^n$.

Ausgabe: $w = (w_0 \dots w_n)$ mit $w = u + v$.

$carry = 0;$

FOR $j = n;\ j \geq 1;\ j = j - 1$

  DO

  $w_j = (u_j + v_j + carry) \bmod 2;$

  $carry = \lfloor (u_j + v_j + carry)/2 \rfloor;$

  OD

$w_0 = carry;$

2.2. Satz *Die Laufzeit von Algorithmus 2.1 ist $O(n)$.*

Analog kann man auch leicht einen Algorithmus zur Subtraktion zweier ganzer Zahlen angeben. Daher wenden wir uns nun der Multiplikation zweier ganzer Zahlen zu.

2.3. Algorithmus (Multiplikation ganzer Zahlen)

Eingabe: $u = (u_1 \dots u_n),\ v = (v_1 \dots v_m)$.

Ausgabe: $w = (w_1 \dots w_{n+m})$ mit $w = u \cdot v$.

```
FOR j = 1; j ≤ n + m; j = j + 1 DO    w_j = 0;    OD
FOR i = m; i ≥ 1; i = i − 1
   DO
   carry = 0;
   FOR j = n; j ≥ 1; j = j − 1
        DO
        t = v_i · u_j + w_{i+j} + carry;
        w_{i+j} = t mod 2;
        carry = ⌊t/2⌋;
        OD
   OD
```

Die Idee des Algorithmus ist die folgende:

Für $i = m, \dots, 1$ addiert der Algorithmus die Zahl

$$2^{m-i} \cdot v_i \cdot \sum_{j=1}^{n} u_j \cdot 2^{n-j}$$

zu dem bisher schon berechneten Ergebnis $\sum_{k=1}^{m+n} w_k \cdot 2^{m+n-k}$ und speichert das Ergebnis als neue Zahl $(w_1 \dots w_{n+m})$ ab.

2.4. Satz *Die Laufzeit von Algorithmus 2.3 ist $O(n \cdot m)$.*

Abschließend behandeln wir nun noch die Division mit Rest. Angenommen, wir wollen $u = (u_1 \dots u_{n+m}) \in \mathbb{N}$ durch $v = (v_1 \dots v_n) \in \mathbb{N}$ mit Rest teilen. Wir wollen also ganze Zahlen q und r bestimmen, so daß

$$u = q \cdot v + r \qquad \text{und} \qquad 0 \leq r < v$$

gilt. Dazu können wir den folgenden Algorithmus benutzen:

2.5. Algorithmus (Division ganzer Zahlen)

Eingabe: $u = (u_1 \dots u_{n+m}), v = (v_1 \dots v_n) \in \mathbb{N}$.

Ausgabe: $q = (q_1 \dots q_m)$ und $r = (r_1 \dots r_n)$, so daß $u = q \cdot v + r$.

```
u_0 = 0;
FOR j = 0; j ≤ m; j = j + 1
    DO
    IF (u_j ... u_{j+n}) < (v_1 ... v_n) THEN q_j = 0;
    ELSE
        q_j = 1;
        (u_j ... u_{j+n}) = (u_j ... u_{j+n}) − (v_1 ... v_n);
        FI
    OD
(r_1 ... r_n) = (u_{m+1} ... u_{m+n});
```

2.6. Satz *Die Laufzeit von Algorithmus 2.5 ist* $O(n \cdot m)$.

Es sei angemerkt, daß es noch eine Reihe von Tricks gibt, mit denen man die obigen Algorithmen schneller machen kann (siehe dazu [KNUTH2]).

2.2 Euklidischer Algorithmus

Um den größten gemeinsamen Teiler zweier natürlicher Zahlen a_1 und a_2 zu berechnen, bildet man die Rekursion

$$a_j = q_j \cdot a_{j+1} + a_{j+2} \tag{2.1}$$

mit ganzen Zahlen $q_j \geq 0$ und $0 \leq a_{j+2} < a_{j+1}$. Durch diese Bedingungen hängen a_{j+2} und q_j in eindeutiger Weise von a_j und a_{j+1} ab.

2.7. Satz *Es gibt ein* $j_0 \in \mathbb{N}$ *derart, daß* $a_{j_0} \neq 0$ *und* $q_j = 0$ *für alle* $j \geq j_0$. *Die Zahl* a_{j_0} *ist der größte gemeinsame Teiler von* a_1 *und* a_2.

Beweis: Die Folge $(a_j)_{j \in \mathbb{N}_{>2}}$ der Divisionsreste ist solange streng monoton fallend, bis sie Null erreicht. Also muß es einen letzten von Null verschiedenen Rest a_{j_0} geben. Wir wollen beweisen, daß a_{j_0} der gesuchte ggT ist. Gemäß (2.1) gilt

$$a_{j_0-1} = q_{j_0-1} \cdot a_{j_0}.$$

Also ist a_{j_0} ein Teiler von a_{j_0-1}. Wegen der gleichen Bedingung gilt aber auch

$$a_{j_0-2} = q_{j_0-2} \cdot a_{j_0-1} + a_{j_0}.$$

Also ist a_{j_0} auch ein Teiler von a_{j_0-2} und durch Induktion sieht man, daß a_{j_0} ein Teiler aller a_j für $j \leq j_0$ ist.

Umgekehrt sieht man leicht, daß jeder Teiler von a_1 und a_2 auch ein Teiler von a_{j_0} ist. Hieraus folgt, daß a_{j_0} der gesuchte größte gemeinsame Teiler von a_1 und a_2 ist. ∎

2.8. Beispiel Wir wollen mit diesem Verfahren den größten gemeinsamen Teiler von 143 und 91 bestimmen. Dann erhalten wir

$$\begin{array}{rcl} 143 & = & 1 \cdot 91 + 52 \\ 91 & = & 1 \cdot 52 + 39 \\ 52 & = & 1 \cdot 39 + 13 \\ 39 & = & 3 \cdot 13 + 0 \end{array}$$

und damit ist 13 der gesuchte größte gemeinsame Teiler.

Es sei angemerkt, daß der euklidische Algorithmus in einem sehr viel allgemeineren Zusammenhang funktioniert. Es sei R ein Integritätsbereich. R heißt **euklidischer Ring**, wenn es eine Höhenfunktion

$$h : R - \{0\} \to \mathbb{N}$$

gibt, derart, daß für alle $a, b \in R$ Ringelemente $q, r \in R$ mit

$$a = b \cdot q + r$$

existieren, wobei entweder $h(r) < h(q)$ oder $r = 0$ ist.

Beispiele für euklidische Ringe sind Polynomringe über Körpern. Dabei wird die Höhenfunktion durch die Gradfunktion für Polynome gegeben.

Bei der Berechnung des größten gemeinsamen Teilers zweier ganzer Zahlen kann man auch eine obere Abschätzung für die Länge der Folge $a_1, a_2, \ldots, a_{j_0}$ angeben.

2.9. Satz *Es sei a_1 eine n_1-Bit-Zahl und a_2 eine n_2-Bit-Zahl. Dann gilt in Satz 2.7*

$$j_0 \leq 2 \cdot \min\{n_1, n_2\} + 2.$$

Beweis: Trivialerweise gilt $a_3 \leq \min\{a_1, a_2\}$ und $a_j < a_{j+1}$ für alle $j \geq 2$. Wir zeigen, daß für $j \geq 2$ die Anzahl der Bits in a_{j+2} wenigstens eins kleiner ist als die Anzahl der Bits in a_j. Daraus folgt dann sofort die Behauptung des Satzes.

Wenn wir a_{j+2} mit a_j vergleichen, so gibt es zwei Möglichkeiten: entweder ist schon $a_{j+1} \leq \frac{1}{2}a_j$ und damit gilt die Behauptung sicherlich auch für a_{j+2} oder $a_{j+1} > \frac{1}{2}a_j$. Dann ist aber

$$a_{j+2} = a_j - q_j \cdot a_{j+1} \leq a_j - a_{j+1} \leq \frac{1}{2}a_j.$$

∎

Daraus ergibt sich direkt der folgende Algorithmus:

2.10. Algorithmus (**ggT-Berechnung**)

Eingabe: $a_1, a_2 \in \mathbb{N}$.

Ausgabe: $d = \mathrm{ggT}(a_1, a_2)$.

REPEAT

$\quad rest = a_1 \bmod a_2;$

$\quad a_1 = a_2;$

$\quad a_2 = rest;$

$\quad$ UNTIL $rest = 0$;

$d = a_1;$

2.11. Satz *Sind in Algorithmus 2.10 die Zahlen a_1 und a_2 beide n-Bit-Zahlen, dann ist die Laufzeit des Algorithmus $O(n^2)$.*

Beweis: Analysiert man die Laufzeit einer Division mit Rest genauer, so kann man zeigen, daß man die Division mit Rest $a_j = q_j \cdot a_{j+1} + a_{j+2}$ in Zeit $O(\log a_{j+1} \cdot \log q_j)$ durchführen kann (siehe [KNUTH2]). Da die a_j eine monoton fallende Folge bilden, ist die Gesamtzeit des euklidischen Algorithmus damit von der Größenordnung

$$O\left(n \cdot \sum_{j=1}^{j_0} \log q_j\right) \;=\; O\left(n \cdot \log \prod_{j=1}^{j_0} q_j\right).$$

Nun gilt aber für $1 \le j \le j_0$, daß $a_j \ge q_j \cdot a_{j+1}$ ist, woraus durch Induktion folgt, daß $a_1 \ge \prod_{j=1}^{j_0} q_j$ gilt. Dies beweist die Behauptung des Satzes. ■

Es gibt einige Varianten des euklidischen Algorithmus, die sich unserer praktischen Erfahrung nach für kleine Zahlen nicht bewährt haben. Eine wichtige Eigenschaft des euklidischen Algorithmus besteht darin, daß man mit seiner Hilfe eine Darstellung des ggT der Form

$$\mathrm{ggT}(a_1, a_2) \;=\; x_1 \cdot a_1 + x_2 \cdot a_2$$

mit ganzen Zahlen $x_1, x_2 \in \mathbb{Z}$ berechnen kann. Es gilt

$$\mathrm{ggT}(a_1, a_2) = a_{j_0} \;=\; -a_{j_0-1} \cdot q_{j_0-2} + a_{j_0-2},$$

woraus man durch weiteres Rückwärtseinsetzen die gewünschte Darstellung berechnen kann.

Um dies praktisch durchzuführen, berechnet man sich während des Algorithmus iterativ die Darstellung der beiden Zahlen a_j und a_{j+1}. Sicherlich gilt

$$a_1 = 1 \cdot a_1 + 0 \cdot a_2 \quad \text{und} \quad a_2 = 0 \cdot a_1 + 1 \cdot a_2,$$

d.h. die Initialisierung ist einfach. Sei nun

$$a_j = x_1 \cdot a_1 + x_2 \cdot a_2 \quad \text{und} \quad a_{j+1} = y_1 \cdot a_1 + y_2 \cdot a_2$$

vorgegeben. Wenn man jetzt $a_{j+2} = a_j - q_j \cdot a_{j+1}$ berechnet hat, dann erhält man daraus

$$\begin{aligned} a_{j+2} &= x_1 \cdot a_1 + x_2 \cdot a_2 - q_j \cdot (y_1 \cdot a_1 + y_1 \cdot a_2) \\ &= a_1 \cdot (x_1 - q_1 \cdot y_1) + a_2 \cdot (x_2 - q_j \cdot y_2). \end{aligned}$$

Damit kann man iterativ die Darstellung des Zahl a_{j+2} berechnen. Hieraus ergibt sich direkt der folgende Algorithmus:

2.12. Algorithmus (ggT **mit Darstellung**)

Eingabe: $a_1, a_2 \in \mathbb{N}$.

Ausgabe: $\mathrm{ggT}(a_1, a_2)$ und $x_1, x_2 \in \mathbb{Z}$ mit $\mathrm{ggT}(a_1, a_2) = x_1 \cdot a_1 + x_2 \cdot a_2$.

$x_1 = 1,\ x_2 = 0,\ y_1 = 0,\ y_2 = 1,\ rest = a_1 \bmod a_2\,;$

WHILE $rest \neq 0$ DO

$q = \lfloor a_1/a_2 \rfloor$

$rest = a_1 \bmod a_2;$

$\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix}$

$a_1 = a_2,\ a_2 = rest;$

OD

$\mathrm{ggT}(a_1, a_2) = a_1;$

2.3 Der chinesische Restsatz

Zuerst betrachte man zur Motivation folgendes Beispiel: Angenommen, wir wollen $9 \cdot 4$ berechnen. Dann rechnen wir aus

$$\begin{aligned} 9 \cdot 4 &\equiv 2 \cdot 4 \equiv 1 \bmod 7 \\ 9 \cdot 4 &\equiv -2 \cdot 4 \equiv -8 \equiv 3 \bmod 11. \end{aligned}$$

Also muß $9 \cdot 4$ eine Zahl sein, die bei der Division durch 7 den Rest 1 und bei der Division durch 11 den Rest 3 läßt. Wie konstruiert man eine solche Zahl?

Allgemeiner seien $m_1, \ldots, m_k \in \mathbb{N}$ und $x_1, \ldots, x_k \in \mathbb{Z}$. Wie findet man ein $x \in \mathbb{Z}$ mit $x \equiv x_i \bmod m_i$ für $1 \leq i \leq k$. Zuerst muß man beachten, daß i.a. eine solche Zahl nur dann existiert, wenn die m_i paarweise teilerfremd sind. Als Beispiel betrachte man $m_1 = 3$, $m_2 = 9$ und $x_1 = 2$, $x_2 = 1$. Zahlen, die bei der Division durch 9 den Rest 1 lassen, können bei der Division durch 3 nicht den Rest 2 lassen. Also kann es das gesuchte x in diesem Falle nicht geben.

Nehmen wir daher zuerst einmal an, daß die m_i paarweise teilerfremd sind. In diesem Fall kann die gesuchte Zahl x wie folgt konstruiert werden:

Wir setzen $M = \prod_{i=1}^{k} m_i$ und $M_i = M/m_i$ für $1 \leq i \leq k$. Dann berechnen wir Zahlen $G_i \in \mathbb{Z}$ mit

$$G_i \cdot M_i \equiv 1 \bmod m_i.$$

Dies kann man mit Hilfe des erweiterten euklidischen Algorithmus machen. Da der ggT von M_i und m_i nach Konstruktion 1 ist, kann man Zahlen G_i und g_i mit

$$G_i \cdot M_i + g_i \cdot m_i = 1$$

berechnen. Betrachtet man dies modulo m_i, so gilt $G_i \cdot M_i \equiv 1 \bmod m_i$.

Setzt man jetzt $e_i \equiv G_i \cdot M_i \bmod M$ und

$$x \equiv \sum_{i=1}^{k} x_i \cdot e_i \bmod M,$$

so folgt offensichtlich die gewünschte Bedingung $x \equiv x_i \bmod m_i$ für alle $1 \leq i \leq k$.

Wie kann man das benutzen? Beispielsweise kann man damit das Produkt zweier Zahlen $a, b \in \mathbb{Z}$ berechnen. Man berechnet zuerst

$$x_i \equiv a \cdot b \bmod m_i \quad \text{für } 1 \leq i \leq k$$

und dann mit obigem Verfahren eine Zahl x mit $x \equiv x_i \bmod m_i$ für $1 \leq i \leq k$. Dann folgt natürlich $x \equiv a \cdot b \bmod M$. Jetzt gilt es aber immer noch viele Möglichkeiten für x. Die Zahl x ist irgendein Divisionsrest von $a \cdot b \bmod M$. Wählen wir aber x als den absolut kleinsten Rest mod$M$, d.h. $-M/2 \leq x < M/2$, dann ist x eindeutig bestimmt als der absolut kleinste Rest von $a \cdot b \bmod M$. Gilt nun auch $-M/2 \leq a \cdot b < M/2$, so muß offensichtlich $x = a \cdot b$ sein. Damit können wir für ein gegebenes System paarweise teilerfremder Moduln m_i alle Zahlen multiplizieren, die im Intervall $\left[-\sqrt{M/2}, \sqrt{M/2} \right]$ liegen.

Ich analysiere diese Art zu multiplizieren: Setze dazu

$$M(x) = \prod_{p \leq x} p.$$

Dabei läuft das Produkt über alle Primzahlen $\leq x$. Außerdem sei $\delta(x) = \log M(x)$. Dann gilt nach einem Satz von Rosser und Schoenfeld (Approximate formulas for some functions of prime numbers, Ill J. Math. (1961), 64-94)

$$\delta(x) = \Omega(x). \tag{2.2}$$

Sei außerdem $\pi(x)$ die Anzahl der Primzahlen $\leq x$. Es gilt nach derselben Arbeit

$$\pi(x) = O(x/\log x). \tag{2.3}$$

Wir wollen die Moduln m_i als Primzahlen wählen. Ist $|a \cdot b| \leq M$, so genügt es, für ein x mit $M(x) \geq 2 \cdot M$ alle Primzahlen $m_i \leq x$ zu verwenden. Damit muß offensichtlich auch die Bedingung $\delta(x) \geq \log(2 \cdot M)$ gelten. Gemäß (2.2) gibt es ein solches x mit $x = O(\log M)$.

Die Anzahl der Primzahlen $\leq x$ ist gemäß (2.3) $O(\log M/\log\log M)$. Analysiert man mit diesen Abschätzungen die einzelnen Operationen, so erhält man folgende Zeiten:

Für die Berechnung von $x_i \equiv a \cdot b \bmod m_i$ brauchen wir 2 Divisionen mit Rest durch m_i und eine Multiplikation $\bmod m_i$, also Zeit

$$O\Big(\underbrace{\frac{\log M}{\log\log M}}_{\text{Anz. Op.}} \cdot \underbrace{\log M}_{\text{Länge } a,b} \cdot \underbrace{\log\log M}_{\text{Länge } m_i}\Big) \quad = \quad O((\log M)^2).$$

Für die Berechnung der Summe $\sum_{i=1}^{k} e_i \cdot x_i \bmod M$ braucht man Zeit

$$O\Big(\underbrace{\frac{\log M}{\log\log M}}_{\text{Anz. Summ.}} \cdot \underbrace{\underbrace{\log M}_{\text{Länge } e_i} \cdot \underbrace{\log\log M}_{\text{Länge } x_i}}_{\text{Produkt}}\Big) \quad = \quad O((\log M)^2).$$

Damit ist bewiesen

2.13. Satz *Berechnet man sich eine ausreichende Anzahl von Moduln m_i und die entsprechenden Zahlen G_i vor und speichert diese Werte in einer Tabelle, so können mittels des chinesischen Restsatzes zwei n-Bit-Zahlen in Zeit $O(n^2)$ multipliziert werden.*

Die Methode der Multiplikation mittels des chinesischen Restsatzes hat den Vorteil, daß sie sich leicht verteilen bzw. parallelisieren läßt. Außerdem kann man die Moduln m_i so geschickt wählen, daß Rechnen modulo m_i sehr leicht ist. Betrachte z.B. die Moduln

$$m = 2^e - 1.$$

Wenn wir modulo m rechnen, erlauben wir, daß für das Ergebnis x

$$0 \leq x < 2^e$$

gilt. Dabei erlauben wir als Darstellung für Ergebnisse, die kongruent zu $0 \bmod m$ sind, sowohl 0 als auch $2^e - 1$. Seien x_1 und x_2 zwei so reduzierte Zahlen, dann kann man eine Zahl y mit $0 \leq y < 2^e$ und $y \equiv x_1 + x_2 \bmod m$ so bestimmen:

$$y = x_1 \oplus x_2 = \begin{cases} x_1 + x_2 & \text{falls } x_1 + x_2 < 2^e \\ ((x_1 + x_2) \bmod 2^e) + 1 & \text{falls } x_1 + x_2 \geq 2^e \end{cases}$$

Das bedeutet, daß man diese Operation durch einen Shift und eine Addition von 1 realisieren kann. Auf ähnlich leichte Weise kann man eine Zahl y mit $0 \leq y < 2^e$ und $y \equiv x_1 \cdot x_2 \bmod m$ finden:

$$y = x_1 \otimes x_2 = (x_1 \cdot x_2 \bmod 2^e) \oplus \lfloor (x_1 \cdot x_2)/2^e \rfloor.$$

Letzteres ist leicht einsichtig, denn es gilt:

$$x_1 \cdot x_2 = z + k \cdot 2^e = z + k \cdot (2^e - 1) + k$$

mit $z =$ kleinster positiver Rest von $x_1 \cdot x_2 \bmod 2^e$ und $k = \lfloor (x_1 \cdot x_2)/2^e \rfloor$.

Um geeignete Moduln zu finden, muß man nur beachten, daß

$$\mathrm{ggT}(2^e - 1, 2^f - 1) = 2^{\mathrm{ggT}(e,f)} - 1$$

ist. Dies bedeutet, daß $2^e - 1$ und $2^f - 1$ genau dann teilerfremd sind, wenn e und f teilerfremd sind.

Jetzt behandle ich noch eine Variante des chinesischen Restsatzes, nämlich die Aufgabe, die simultane Kongruenz $x \equiv x_i \bmod m_i$ $(1 \leq i \leq k)$ zu lösen, wenn die Moduln nicht paarweise teilerfremd sind. Dazu müssen wir zuerst das Problem lösen, wann eine solche Kongruenz $u \cdot x \equiv v \bmod m$ mit ganzen Zahlen $u, v, m \in \mathbb{Z}$ für m nicht notwendig Primzahl überhaupt nach x lösbar ist. Dies werden wir dann im anschließenden "Verallgemeinerten Chinesischen Restsatz" verwenden.

2.14. Satz *Sei $m \in \mathbb{N}$, $v, u \in \mathbb{Z}$, $d = \mathrm{ggT}(u, m)$ und $w \in \mathbb{Z}$ Lösung von $u \cdot w \equiv d \bmod m$. Dann ist $u \cdot x \equiv v \bmod m$ genau dann nach x lösbar, wenn d ein Teiler von v ist. Es gilt in diesem Fall für die Lösung x, daß $x \equiv v' \bmod m'$ mit*

$$\begin{aligned} m' &= \frac{m}{d} \\ v' &= \frac{v \cdot w}{d}. \end{aligned}$$

Beweis: Sei zuerst d ein Teiler von v. Dann gilt

$$v = \frac{v}{d} \cdot d \equiv \frac{v}{d} \cdot (u \cdot w) \equiv u \cdot \frac{w \cdot v}{d} \bmod m$$

und damit ist $x \equiv \frac{w \cdot v}{d} \bmod m$ eine Lösung.

Sei andererseits x eine Lösung von $u \cdot x \equiv v \bmod m$. Da $d = \mathrm{ggT}(u, m)$ ist, kann man dies auch schreiben als

$$(u' \cdot d) \cdot x \equiv v \bmod (m' \cdot d),$$

was gleichbedeutend ist mit $v = u' \cdot d \cdot x - k \cdot m' \cdot d$ für eine Zahl $k \in \mathbb{Z}$. Daraus folgt, daß d ein Teiler von v ist.

Nehmen wir nun an, die Gleichung ist lösbar. Teile die Gleichung durch d. Dann gilt

$$u' \cdot x \equiv v' \bmod m' \quad \text{mit } u' = \frac{u}{d},\ v' = \frac{v}{d},\ m' = \frac{m}{d}.$$

Jetzt ist aber $\mathrm{ggT}(u', m') = 1$ und damit kann u' modulo m' invertiert werden. Also ist $x \equiv v' \cdot u'^{-1} \bmod m'$. Führt man die gleiche Vorgehensweise für die Gleichung $u \cdot w \equiv d \bmod m$ aus, so erkennt man, daß $u'^{-1} \equiv w \bmod m'$, woraus die zweite Behauptung folgt. ■

Damit haben wir eine Voraussetzung geschaffen, um den Verallgemeinerten Chinesischen Restsatz anzugehen und einen Algorithmus dafür zu entwerfen.

2.15. Satz (Verallgemeinerter Chinesischer Restsatz)

Es seien $m_1, \ldots, m_k \in \mathbb{N}$ und $u_1, \ldots, u_k \in \mathbb{Z}$. Dann ist das System

$$x \equiv u_i \bmod m_i \quad \textit{für } 1 \le i \le k$$

genau dann lösbar, wenn $u_i \equiv u_j \bmod \mathrm{ggT}(m_i, m_j)$ für alle $1 \le i < j \le k$ gilt und die Lösung ist eindeutig bestimmt modulo $n = \mathrm{kgV}(m_1, \ldots, m_k)$.

Wir wollen nun einen Algorithmus entwerfen, der die Lösung einer solchen simultanen Kongruenz berechnet, falls dies möglich ist und andernfalls die Meldung "unlösbar" ausgibt. Die Idee dieses Algorithmus ist die folgende:

Bestimme iterativ für $i = 1, \ldots, k$ Zahlen m_i, $M_i \in \mathbb{N}$ und $v_i \in [0, \ldots, M_i]$, so daß jedes $v \in \mathbb{Z}$ mit

$$v \equiv v_i \bmod M_i$$

die ersten i Kongruenzen löst.

Offensichtlich startet die Iteration mit $v_1 = u_1$ und $M_1 = m_1$. Nehmen wir nun an, wir haben v_{i-1} und M_{i-1} bestimmt. Dann muß v_i insbesondere die ersten $(i-1)$ Gleichungen erfüllen, d.h.

$$v_i \equiv v_{i-1} \bmod M_{i-1}.$$

Also ist v_i von der Form $v_i = v_{i-1} + k \cdot M_{i-1}$ mit $k \in \mathbb{Z}$. Außerdem muß natürlich auch die i-te Gleichung erfüllt sein und damit gelten

$$v_i = v_{i-1} + k \cdot M_{i-1} \equiv u_i \bmod m_i.$$

Nach Satz 2.14 ist diese Gleichung genau dann nach k lösbar, wenn $d = \mathrm{ggT}(m_i, M_{i-1})$ ein Teiler von $u_i - v_{i-1}$ ist. Dann kann man k aus dem Bereich $0 \le k \le \frac{m_i}{d} - 1$ wählen. Also gilt

$$0 \le v_i < M_{i-1} + \left(\frac{m_i}{d} - 1\right) \cdot M_{i-1} = \frac{m_i}{d} \cdot M_{i-1} =: M_i.$$

Dann kann man mit Hilfe von Satz 2.14 aber auch leicht den Wert von k modulo M_i ausrechnen (vgl. im Algorithmus).

Damit erhalten wir den folgenden Algorithmus:

2.16. Algorithmus (Verallgemeinerter Chinesischer Restsatz)

Eingabe: $m_1, \ldots, m_k \in \mathbb{N}, \quad u_1, \ldots, u_k \in \mathbb{Z}$

Ausgabe: $x \in [0, \ldots, kgV(m_1, \ldots, m_k) - 1]$, das simultane Kongruenzen löst bzw. "unlösbar".

Setze $M_1 = m_1, v_1 \equiv u_1 \bmod M_1$.

FOR $i = 2$ TO k

DO

setze $c = u_i - v_{i-1}$;

berechne $d = \mathrm{ggT}(M_{i-1}, m_i) = c' \cdot M_{i-1} + c'' \cdot m_i$;

IF $d \not| c$ THEN terminiere mit "unlösbar"

setze $M_i = \frac{m_i}{d} \cdot M_{i-1}$;

setze $v_i \equiv (v_{i-1} + \frac{c \cdot c'}{d} \cdot M_{i-1}) \bmod M_i$;

OD

Return $x \equiv v_k \bmod M_k$;

2.17. Beispiel Löse

$$\begin{aligned} x &\equiv 3 \bmod 4 \\ x &\equiv 3 \bmod 6 \\ x &\equiv 12 \bmod 15. \end{aligned}$$

Nach der Initialisierung gilt $M_1 = 4$ und $v_1 = 3$. In der zweiten Iteration ist $c = 0$ und dann $d = \mathrm{ggT}(4,6) = 2 = -1 \cdot 4 + 1 \cdot 6$. Daher erhalten wir $M_2 = \frac{6}{2} \cdot 4 = 12$ und $v_2 = 3 + \frac{0 \cdot -1}{2} \cdot 4 \equiv 3 \bmod 12$.

In der dritten Iteration ist $c = 12 - 3 = 9$ und $d = \mathrm{ggT}(12,15) = 3 = -1 \cdot 12 + 1 \cdot 15$. Somit gilt $M_3 = \frac{15}{3} \cdot 12 = 60 = kgV(4,6,15)$. Das Ergebnis v_3 berechnet sich als $v_3 = 3 + \frac{9 \cdot -1}{3} \cdot 12 = 3 - 36 \equiv 27 \bmod 60$.

2.4 Schnelle Exponentiation

Bei praktischen Anwendungen der Computeralgebra tritt häufig das Problem auf, daß man zu gegebenen Zahlen $a, d, m \in \mathbb{N}$ den Wert

$$a^d \bmod m$$

ausrechnen muß. Dies kann man durch d Multiplikationen mit anschließender Reduktion modulo m berechnen. Häufig ist allerdings d sehr groß z.B. hat man bei Pseudoprimzahltest häufig $d \approx 10^{50}$. In diesem Fall ist d-fache Multiplikation unmöglich. Es gibt jedoch ein schnelleres Verfahren, die sogenannte schnelle Exponentiation.

2.18. Beispiel Berechne $2^9 \bmod 10$. Zerlegen wir 9 im Binärsystem, so gilt $9 = (1001)_2$. Damit erhalten wir

$$2^9 = 2^{1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1} = 2^8 \cdot 2^0 \cdot 2^0 \cdot 2^1 \equiv 2^8 \cdot 2 \equiv 6 \cdot 2 \equiv 2 \bmod 10.$$

Allgemein führt diese Idee zu folgender Vorgehensweise: sei $d = \sum_{i=0}^{s} \beta_i \cdot 2^i$ mit $\beta_i \in \{0,1\}$. Dann gilt:

$$a^d = a^{\sum_{i=0}^{s} \beta_i 2^i} = \prod_{\beta_i = 1} a^{2^i}.$$

Nun sollte man beachten, daß man die benötigten Teilergebnisse a^{2^i} aus den vorher berechneten Ergebnissen a^{2^j} für $j < i$ berechnen kann. Damit erhält man den folgenden Algorithmus:

2.19. Algorithmus (Schnelle Exponentiation)

Eingabe: $a, d, m \in \mathbb{N}$.

Ausgabe: $a^d \bmod m$.

Initialisierung: $b \equiv 1 \bmod m$, $c \equiv a \bmod m$;

WHILE $d > 0$

DO

IF d ungerade THEN setze $b = b \cdot c \bmod m$, $d = d - 1$; FI

Setze $d = \frac{d}{2}$, $c = c \cdot c \bmod m$;

OD

Return $b \bmod m$;

Nun zur Komplexität dieses Algorithmus: sicherlich benötigt ein Durchlauf der WHILE-Schleife Zeit $O((\log m)^2)$, denn alle Zwischenergebnisse sind durch m beschränkt. Da weiterhin der Exponent in jeder Iteration mindestens halbiert wird, gibt es maximal $O(\log d)$ viele Iterationen. Damit haben wir bewiesen:

2.20. Satz *Man kann mit schneller Exponentiation den Wert $a^d \bmod m$ in Zeit $O(\log d \cdot (\log m)^2)$ berechnen.*

Abschließend noch einige Bemerkungen: schnelle Exponentiation funktioniert in jeder abelschen Halbgruppe. Außerdem gibt es noch einige Varianten zu dem hier vorgestellten Algorithmus. Die erste ist die, die Binärstellen des Exponenten von links nach rechts abzuarbeiten. Hierzu benutzt man die Formel

$$g^n = \begin{cases} (g^{n/2})^2 & \text{für } n \equiv 0 \bmod 2 \\ g \cdot (g^{(n-1)/2})^2 & \text{für } n \not\equiv 0 \bmod 2 \end{cases}.$$

2.21. Übung Man schreibe einen Algorithmus für die left to right binary exponentiation.

Die tatsächliche Rechenzeit des Exponentiationsalgorithmus hängt stark von der Anzahl der Nullen in der Binärdarstellung des Exponenten ab. Um diese Anzahl zu erhöhen, kann man auch Darstellungen mit Koeffizienten zulassen. Ich nenne das die erweiterte Binärdarstellung. Zum Beispiel ist

$$(1111)_2 = 8 + 4 + 2 + 1 = 15, \quad \text{aber auch} \quad (1000)_2 - (1)_2 = 16 - 1 = 15.$$

Damit kann man a^{15} ausrechnen als $a^8 \cdot a^4 \cdot a^2 \cdot a^1$ und als $a^{16} \cdot a^{-1}$. Kann die Invertierung eines Gruppenelementes in der gleichen Zeit wie eine Multiplikation erfolgen, so ist die zweite Methode häufig schneller.

2.22. Übung Modifiziere Algorithmus 2.19 für die Verwendung der erweiterten Binärdarstellung.

2.23. Übung Man schreibe einen Algorithmus, der eine Zahl in Binärdarstellung in eine Zahl in erweiterter Binärdarstellung verwandelt.

2.5 Rechnen in der primen Restklassengruppe modulo m

Es sei $m \in \mathbb{N}, m \geq 2$. In diesem Abschnitt bespreche ich einige grundlegende Algorithmen für das Rechnen in $(\mathbb{Z}/m\mathbb{Z})^*$, der primen Restklassengruppe modulo m.

Wie ist diese Gruppe definiert? Eine **Restklasse** $\bmod m$ ist eine Menge

$$\overline{e} = \{l + k \cdot m : k \in \mathbb{Z}\}.$$

Jedes Element in $\overline{e}$ heißt **Vertreter** von $\overline{e}$. Offensichtlich folgt aus dem Umstand, daß ein Vertreter von $\overline{e}$ teilerfremd zu m ist, daß alle Vertreter von $\overline{e}$ teilerfremd zu m sind. Solche Restklassen heißen **prime Restklassen modulo** m.

2.24. Satz *Die Menge der primen Restklassen modulo m ist eine multiplikative Gruppe.*

Beweis: Übung ■

Es gibt verschiedene Möglichkeiten, die Elemente von $(\mathbb{Z}/m\mathbb{Z})^*$ darzustellen, z.B. durch ihren absolut kleinsten Vertreter oder durch ihren kleinsten nichtnegativen Vertreter. Sofern nichts anderes gesagt wird, gehe ich davon aus, daß prime Restklassen durch ihre kleinsten nichtnegativen Vertreter dargestellt werden.

Die elementaren Operationen in $(\mathbb{Z}/m\mathbb{Z})^*$ können mit Hilfe der elementaren Operationen in $\mathbb{Z}$ und mit Hilfe des euklidischen Algorithmus realisiert werden: Die Multiplikation zweier Elemente in $(\mathbb{Z}/m\mathbb{Z})^*$ benutzt die Multiplikation in $\mathbb{Z}$ und eine Division mit Rest. Inverse berechnet man mit Hilfe des erweiterten euklidischen Algorithmus.

2.25. Satz *Die Multiplikation und Inversenbildung von Elementen aus $(\mathbb{Z}/m\mathbb{Z})^*$ kann in Zeit $O((\log m)^2)$ ausgeführt werden.*

Ich benötige jetzt einige Begriffe aus der Gruppentheorie. Die Elementanzahl einer Gruppe G heißt **Ordnung der Gruppe** oder auch **Gruppenordnung**. Wir schreiben dafür $|G|$. Für $g \in G$ nennt man

$$\langle g \rangle \quad = \quad \{g^e : e \in \mathbb{N} \cup \{0\}\}$$

die von g erzeugte Untergruppe von G. Die **Ordnung von** g ist definiert als $|\langle g \rangle|$. Man erkennt leicht, daß die Ordnung eines Elementes g die kleinste positive Zahl x mit $g^x = 1$ ist. Ist $G = \langle g \rangle$ für ein Gruppenelement $g \in G$, so heißt G **zyklisch**.

2.26. Satz *Ist G eine endliche Gruppe, so gilt* *(i)* $g^{|G|} = 1$ *für alle* $g \in G$.

(ii) $|\langle g\rangle| \,\Big|\, |G|$ *für alle* $g \in G$.

Beweis: Sei $g \in G$. Die Abbildung

$$G \to G, \quad h \longmapsto g \cdot h$$

ist, wie man leicht beweisen kann, ein Isomorphismus. Also muß

$$\prod_{h\in G} h = \prod_{h\in G} h \cdot g = g^{|G|} \cdot \prod_{h\in G} h$$

sein. Daraus folgt $g^{|G|} = 1$ und (i) ist gezeigt.

Die Zahl $|\langle g\rangle|$ ist der kleinste Exponent $e > 0$ mit $g^e = 1$. Schreibe dann

$$|G| = e \cdot q + r \quad \text{mit } 0 \le r < e.$$

Dann gilt $g^r = g^{|G|} \cdot (g^e)^{-q} = 1 \cdot 1 = 1$. Da $r < e$ ist, muß $r = 0$ sein und (ii) ist bewiesen. ■

Jetzt bespreche ich die Struktur der primen Restklassengruppe modulo m. Die Ordnung von $(\mathbb{Z}/m\mathbb{Z})^*$ wird mit $\varphi(m)$ bezeichnet. Diese Funktion nennt man die **Eulersche Phi-Funktion**.

2.27. Satz *Sei* p *eine Primzahl. Dann ist*

$$\varphi(p^e) \;=\; (p-1) \cdot p^{e-1}.$$

Beweis: Sei $k \in \mathbb{Z}$, $0 \le k < p^e$. Schreibe $k = q \cdot p + r$ mit $0 \le r < p$ und $0 \le q < p^{e-1}$. Dann ist k genau dann teilerfremd zu p, wenn r teilerfremd zu p ist. Es gibt aber $p-1$ zu p teilerfremde Zahlen im Intervall $[0, p-1]$, woraus die Behauptung folgt. ■

2.28. Satz *Es seien* $m_1, \ldots, m_k$ *paarweise teilerfremd. Dann ist die Abbildung*

$$\begin{array}{rcl} \mathbb{Z}/m & \to & \mathbb{Z}/m_1 \times \ldots \times \mathbb{Z}/m_k \\ x \bmod m & \longmapsto & (x \bmod m_1, \ldots, x \bmod m_k) \end{array}$$

ein Ringisomorphismus.

Beweis: Übung, vgl. Chinesischer Restsatz ■

2.29. Korollar *Seien* $m_1, \ldots, m_k$ *paarweise teilerfremd und* $m = m_1 \cdot \ldots \cdot m_k$. *Dann gilt:*

$$\varphi(m) \;=\; \varphi(m_1) \cdot \ldots \cdot \varphi(m_k).$$

Beweis: Eine Zahl x ist genau dann teilerfremd zu m, wenn sie teilerfremd zu allen m_i ist. ■

2.30. Korollar *Für $m \in \mathbb{N}$ gilt:*

$$\varphi(m) \; = \; m \cdot \prod_{p|m} \left(1 - \frac{1}{p}\right).$$

Beweis: Übung. ■

2.31. Satz *Sei p eine Primzahl. Dann ist $(\mathbb{Z}/p\mathbb{Z})^*$ zyklisch von der Ordnung $p-1$.*

Beweis: Offensichtlich hat $(\mathbb{Z}/p\mathbb{Z})^*$ die Ordnung $p-1$. Sei nun d ein Teiler von $p-1$. Angenommen, es gibt ein Element $\overline{x} \in (\mathbb{Z}/p\mathbb{Z})^*$ der Ordnung d. Dann gilt für alle Exponenten $e \in \mathbb{Z}$ mit $\mathrm{ggT}(e,d) = 1$, daß $\overline{x}^e$ ebenfalls die Ordnung d besitzt. Also gibt es mindestens $\varphi(d)$ Elemente der Ordnung d in $(\mathbb{Z}/p\mathbb{Z})^*$. Berücksichtigt man, daß alle Elemente mit Exponenten d (d.h. alle Elemente $a \in G$ mit $a^d = 1$) Nullstelle des Polynoms $x^d - 1$ sind und daß die Gruppenelemente $\overline{x}^j$ für $j = 1, \ldots, d$ Exponenten d besitzen, so wird deutlich, daß es genau $\varphi(d)$ viele Elemente der Ordnung d in der Gruppe G gibt.

Nun gilt aber

$$\begin{aligned} p-1 &= |(\mathbb{Z}/p\mathbb{Z})^*| \\ &= \sum_{d|p-1} \text{Anzahl der Elemente mit Ordnung } d \\ &= \sum_{d|p-1} \varphi(d). \end{aligned}$$

Nun gilt aber auch $\sum_{d|n} \varphi(d) \; = \; n$ (Beweis Übung) und damit muß $d = p-1$ tatsächlich vorkommen, d.h. es muß ein Element der Ordnung $p-1$ geben. Daher ist $(\mathbb{Z}/p\mathbb{Z})^*$ zyklisch. ■

Ein wichtiges Problem besteht darin, ein erzeugendes Element für $(\mathbb{Z}/p\mathbb{Z})^*$ zu finden. Eine Zahl x, deren Restklasse mod$p$ diese Gruppe erzeugt, heißt **Primitivwurzel** mod$p$. Es zeigt sich, daß man zwar eine Primitivwurzel leicht raten kann, daß aber die Verifikation, daß x wirklich eine Primitivwurzel ist, schwer ist.

Ein Algorithmus für die Identifikation einer Primitivwurzel beruht auf folgendem Satz.

2.32. Satz *Sei G eine Gruppe und $g \in G$ sowie $n \in \mathbb{N}$. Dann ist genau dann $n = |\langle g \rangle|$, wenn*

$$g^n = 1 \quad \textit{und} \quad g^{n/q} \neq 1 \textit{ für alle Primteiler von } n.$$

Beweis: Die Ordnung e von g ist nach Definition der kleinste Exponent $e > 0$ mit $g^e = 1$. Gilt $g^n = 1$ mit einer natürlichen Zahl n, so muß e also ein Teiler von n sein. Gilt $g^{n/p_i} \neq 1$ für eine Primzahl $p_i | n$, so kann e kein Teiler von n/p_i sein. Ist also für alle Primteiler p_i von n $g^{n/p_i} \neq 1$, so muß $e = n$ sein. ■

Also muß man zum Beweis, ob ein Element g eine Primitivwurzel modulo p ist, die Primfaktoren von $p-1$ bestimmen und prüfen, ob $g^{(p-1)/q} = 1$ ist für einen Primfaktor q von $p-1$. Die Bestimmung der Primfaktoren, d.h. das Problem der Faktorisierung, ist i.a. ein nur schwer lösbares Problem, das wir später behandeln werden. Hat man diese Faktorisierung bestimmt, so kann man durch Probieren von "wenigen" zufällig gewählten Elementen $g \in (\mathbb{Z}/p\mathbb{Z})^*$ in der Praxis schnell eine Primitivwurzel "berechnen". Wenden wir uns daher nun dem Problem zu, eine Primitivwurzel modulo p^e zu finden. Dies ist sehr viel einfacher, wie der folgende Satz zeigt:

2.33. Satz *Sei p eine ungerade Primzahl und $m = p^e$. Dann ist $(\mathbb{Z}/m\mathbb{Z})^*$ zyklisch. Außerdem gilt für jede Primitivwurzel $x \bmod p$, daß entweder $\overline{x}$ oder $\overline{x+p}$ ein erzeugendes Element für $(\mathbb{Z}/m\mathbb{Z})^*$ ist.*

Beweis: Wir wissen schon, daß die Ordnung von $(\mathbb{Z}/m\mathbb{Z})^*$ genau $p^{e-1} \cdot (p-1)$ ist. Da $\overline{x}$ eine Primitivwurzel modulo p ist, müssen wir nach Satz 2.32 nur noch beweisen, daß entweder

$$x^{p^{e-2}\cdot(p-1)} \not\equiv 1 \bmod p^e \quad \text{oder} \quad (x+p)^{p^{e-2}\cdot(p-1)} \not\equiv 1 \bmod p^e$$

ist. Durch Induktion zeigt man leicht, daß aus $y^p \equiv 1 \bmod p^j$ die Beziehung $y \equiv 1 \bmod p^{j-1}$ folgt. Damit folgt aus $y^{p^{e-2}\cdot(p-1)} \equiv 1 \bmod p^e$ leicht die Gleichung $y^{p-1} \equiv 1 \bmod p^2$. Diese Gleichung kann aber nicht für x und $x+p$ gelten, was man mit Hilfe der ersten binomischen Formel leicht zeigen kann. Damit gilt für eines der beiden Elemente $\overline{x}$ bzw. $\overline{x+p}$, daß es eine Primitivwurzel modulo p^e ist. Insbesondere ist $(\mathbb{Z}/m\mathbb{Z})^*$ damit zyklisch. ■

2.34. Satz *Unter Annahme der ERH gibt es eine Primitivwurzel modulo p kleiner gleich $r^4 \cdot (\log r + 1)^4 \cdot (\log p)^2$, wobei r die Anzahl der Primteiler von $p-1$ ist.*

Beweis: V. Shoup, STOC 1990, 546–554. ■

Kapitel 3

Algorithmen für Polynome

3.1 Polynomarithmetik

Es sei R ein kommutativer Ring mit Einselement 1. Wir studieren elementare Operationen in dem Polynomring $R[x]$. Kandidaten für R, die wir bisher kennengelernt haben, sind $\mathbb{Z}$ und $\mathbb{Z}/m\mathbb{Z}$. Es könnte aber auch ein endlicher Körper oder ein Polynomring als Koeffizientenbereich vorkommen.

Die Komplexität der Algorithmen wird in Abhängigkeit von der Komplexität der elementaren Operationen in R bestimmt. Die Bitkomplexität ergibt sich dann durch Einsetzen der Bitkomplexität für die elementaren Ringoperationen. Elementare Ringoperationen sind Addition, Multiplikation, Negation, Test auf Einheit und Bestimmung des Inversen einer Einheit.

Ein Polynom

$$p(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0$$

wird durch den Vektor $(a_0, a_1, \ldots, a_n) \in R^{n+1}$ seiner Koeffizienten dargestellt. Ist $a_n \neq 0$, so heißt n der **Grad** von $p(x)$, wir schreiben $n = \deg p(x)$. a_n heißt der **Leitkoeffizient** von $p(x)$, wir schreiben $a_n = l(p(x))$.

Addition und skalare Multiplikation von Polynomen realisiert man durch die entsprechenden Operationen auf den Koeffizientenvektoren.

Multiplikation benutzt die Formel

$$\left(\sum_{i=0}^{n} a_i \cdot x^i\right) \cdot \left(\sum_{j=0}^{m} b_j \cdot x^j\right) = \sum_{k=0}^{n+m} \left(\sum_{i=0}^{k} a_i \cdot b_{k-i}\right) \cdot x^k.$$

Für sehr große Polynomgrade benutzt man auch FFT.

3.1. Satz *Addition oder skalare Multiplikation zweier Polynome vom Grad n benötigt $O(n)$ Ringoperationen. Multiplikation eines Polynoms vom Grad m und eines Polynoms vom Grade n benötigt $O(m \cdot n)$ Ringoperationen.*

Als nächstes besprechen wir die Division mit Rest. Nehmen wir dazu zuerst an, daß $K = R$ ein Körper ist. Dann lassen sich alle Elemente invertieren und die Division bereitet keine Probleme.

3.2. Algorithmus (Division von Polynomen über Körpern)

Eingabe: $a(x), b(x) \in K[x],\ b(x) \neq 0.$

Ausgabe: $q(x), r(x) \in K[x]$ mit $a(x) = q(x) \cdot b(x) + r(x)$ und $\deg r(x) < \deg b(x)$.

Setze $r(x) = a(x),\ q(x) = 0;$

WHILE $\deg r(x) \geq \deg b(x)$

DO

$s(x) = \frac{l(r(x))}{l(b(x))} \cdot x^{\deg r(x) - \deg b(x)};$

$q(x) = q(x) + s(x);$

$r(x) = r(x) - s(x) \cdot b(x);$

OD

Ist R kein Körper, so lassen sich evtl. nicht alle Leitkoeffizienten des Polynoms $b(x)$ invertieren. Deshalb muß man in diesem Fall die Division etwas anders definieren und erhält so die sogenannte Pseudodivision. Vergleiche dazu den folgenden Algorithmus:

3.3. Algorithmus (Pseudodivision)

Eingabe: $a(x), b(x) \in R[x],\ b(x) \neq 0,\ \deg a(x) \geq \deg b(x).$

Ausgabe: $q(x), r(x) \in R[x]$ mit $l(b(x))^{\deg a(x) - \deg b(x) + 1} \cdot a(x) = b(x) \cdot q(x) + r(x)$ und $\deg r(x) < \deg b(x)$.

Setze $r(x) = a(x),\ q(x) = 0;$

Setze $e = \deg a(x) - \deg b(x) + 1,\ d = l(b(x));$

WHILE $\deg r(x) \geq \deg b(x)$

DO

$s(x) = l(r(x)) \cdot x^{\deg r(x) - \deg b(x)};$

$q(x) = d \cdot q(x) + s(x);$

$r(x) = d \cdot r(x) - s(x) \cdot b(x);$

$e = e - 1;$

OD

Return $q(x) = d^e \cdot q(x),\ r(x) = d^e \cdot r(x).$

3.2 ggT in Polynomringen

Ist $R = K$ ein Körper, so ist $R[x]$ ein euklidischer Ring und der ggT zweier Polynome kann wie bei ganzen Zahlen mit Hilfe des euklidischen Algorithmus berechnet werden.

Jetzt verallgemeinern wir die Situation. Wir nehmen an, daß R ein ZPE-Ring ist. Um diesen Begriff zu erklären, führen wir einige Begriffe ein, die im Zusammenhang mit $\mathbb{Z}$ schon bekannt sind.

Invertierbare Elemente in R heißen **Einheiten**. Die Menge der Einheiten in R bezeichnet man mit R^*. Für $a, b \in R$ mit $b \neq 0$ sagen wir "b **teilt** a" (in Zeichen $b|a$), falls es ein $q \in R$ gibt mit $a = b \cdot q$. Ist $a = 0$ und $q \neq 0$, so heißt b ein **Nullteiler**. Nullteilerfreie Ringe heißen **Integritätsbereiche**. Wir nehmen im weiteren immer an, daß R ein Integritätsbereich ist. Ein Element $p \in R - (R^* \cup \{0\})$ heißt **Primelement**, falls für $a, b \in R$ aus $p|a \cdot b$ entweder $p|a$ oder $p|b$ folgt. Ein Element $p \in R$ heißt **irreduzibel**, falls für $q \in R$ aus $q|p$ folgt, daß entweder q oder p/q eine Einheit ist. Der Ring R heißt **ZPE-Ring**, falls jedes Element von R als Produkt von Primelementen geschrieben werden kann, wobei die Faktoren bis auf die Reihenfolge und bis auf Multiplikation mit Einheiten eindeutig bestimmt sind. Ein gutes Beispiel für einen ZPE-Ring ist der Ring $\mathbb{Z}$ der ganzen Zahlen. In diesem Zusammenhang ist besonders interessant:

3.4. Satz *Es sei R ein ZPE-Ring und $p \in R$. Genau dann ist eine Nichteinheit $p \neq 0$ ein Primelement, wenn p irreduzibel ist.*

3.5. Satz *Ist R ein ZPE-Ring, so ist auch $R[x]$ ein ZPE-Ring.*

Es sei ab jetzt R ein ZPE-Ring und $S \subseteq R$. Ein Element $d \in R$, das alle Elemente von S teilt, heißt **gemeinsamer Teiler** von S. Ein gemeinsamer Teiler von S, der von jedem anderen gemeinsamen Teiler geteilt wird, heißt ein **größter gemeinsamer Teiler** ggT von S.

3.6. Satz *Sei R ein ZPE-Ring, $S \subseteq R$. Dann existiert ein* ggT *von S und dieser ist bis auf Multiplikation mit Einheiten eindeutig bestimmt.*

Beweis: Sei P ein vollständiges Vertretersystem aller Primelemente des Rings R. Da R ZPE-Ring ist, kann man jedes $s \in S$ schreiben als

$$s = u \cdot \prod_{p \in P} p^{e_p(s)}$$

mit einer Einheit $u \in R^*$ und nur endlich vielen von Null verschiedenen Exponenten $e_p(s)$. Man kann leicht zeigen, daß dann

$$\prod_{p \in P} p^{\min_{s \in S}(e_p(s))}$$

ein ggT von S ist. Beachte dabei, daß S auch unendlich viele Elemente enthalten darf, da nur endlich viele Exponenten ungleich Null sind. ■

Ist $S \subseteq R$ und ist jeder ggT von S eine Einheit, so heißen die Elemente von S **paarweise teilerfremd**.

Wir besprechen nun die Berechnung eines ggT in $R[x]$. Sei $p(x)$ ein Polynom aus $R[x]$. Ein ggT der Koeffizienten von $p(x)$ heißt **Inhalt** von $p(x)$, abgekürzt $\operatorname{cont}(p(x))$. Das Polynom $p(x)$ heißt **primitiv**, falls seine Koeffizienten teilerfremd sind. Das Polynom $p(x)/\operatorname{cont}(p(x))$ ist primitiv und heißt ein primitiver Teil von $p(x)$ (Abkürzung $\operatorname{pp}(p(x))$). Wenn es im folgenden heißt, bestimme $\operatorname{cont}(p(x))$ oder bestimme $\operatorname{pp}(p(x))$, dann meinen wir, daß man *einen* Inhalt bzw. *einen* primitiven Teil bestimmen soll. Man sollte jedoch die Uneindeutigkeit, die durch Multiplikation mit Einheiten entsteht, bedenken.

3.7. Satz *Seien $a(x), b(x) \in R[x]$. Weiterhin sei $\operatorname{ggT}(a(x), b(x))$ und die folgenden Inhalte $\operatorname{cont}(a(x))$, $\operatorname{cont}(b(x))$, $\operatorname{cont}(a(x) \cdot b(x))$ und $\operatorname{cont}(\operatorname{ggT}(a(x), b(x)))$ gewählt. Seien die entsprechenden primitiven Teile durch Division durch diese Inhalte hervorgegangen. Außerdem sei $\operatorname{ggT}(\operatorname{cont}(a(x)), \operatorname{cont}(b(x))$ und $\operatorname{ggT}(\operatorname{pp}(a(x)), \operatorname{pp}(b(x))$ gewählt. Dann gibt es Einheiten $u, v, w \in R$, so daß*

$$\begin{aligned}
\operatorname{cont}(a(x) \cdot b(x)) &= u \cdot \operatorname{cont}(a(x)) \cdot \operatorname{cont}(b(x)) \\
\operatorname{pp}(a(x) \cdot b(x)) &= u^{-1} \cdot \operatorname{pp}(a(x)) \cdot \operatorname{pp}(b(x)) \\
\operatorname{cont}(\operatorname{ggT}(a(x), b(x)) &= v \cdot \operatorname{ggT}(\operatorname{cont}(a(x)), \operatorname{cont}(b(x))) \\
\operatorname{pp}(\operatorname{ggT}(a(x), b(x))) &= w \cdot \operatorname{ggT}(\operatorname{pp}(a(x)), \operatorname{pp}(b(x))).
\end{aligned}$$

Damit erhalten wir den folgenden Algorithmus zur Berechnung des größten gemeinsamen Teilers über dem Polynomring $R[x]$.

3.8. Algorithmus (**ggT in** $R[x]$)

Eingabe: $a(x)$, $b(x) \in R[x]$, R ZPE-Ring.

Ausgabe: ein $\operatorname{ggT}(a(x), b(x))$.

Setze $d = \operatorname{ggT}(\operatorname{cont}(a(x)), \operatorname{cont}(b(x)))$;

Setze $a(x) = a(x)/\operatorname{cont}(a(x))$, $b(x) = b(x)/\operatorname{cont}(b(x))$;

Setze $r(x) = b(x)$, $b(x) = a(x)$;

REPEAT

  $a(x) = b(x)$;

  $b(x) = \operatorname{pp}(r(x))$

  berechne $q(x), r(x)$ mit $l(b(x))^{\deg a(x) - \deg b(x) + 1} \cdot a(x) = b(x) \cdot q(x) + r(x)$;

  UNTIL $r(x) = 0$;

Return $d \cdot b(x)$;

Das Problem in diesem Algorithmus besteht darin, daß in der REPEAT-Schleife zu oft der ggT der Koeffizienten von $r(x)$ berechnet werden muß. Dies wird später repariert.

Nun wollen wir noch die Korrektheit des ggT-Algorithmus beweisen. Sei $(a_j(x))_{j\in\mathbb{N}}$ die Folge der im Algorithmus berechneten Polynome. Sei $S_k = \deg a_{k+1}(x) - deg a_k(x)$. Im letzten Schritt gilt

$$l(a_k(x))^{S_{k-1}} \cdot a_{k-1}(x) = q_k(x) \cdot a_k(x).$$

Hierbei ist $a_k(x)$ ein primitives Polynom. Weil R ein ZPE-Ring ist, muß $a_k(x)$ ein Teiler von $a_{k-1}(x)$ sein. Im allgemeinen hat man

$$l(a_j(x))^{S_{j-1}} \cdot a_{j-1}(x) = q_j(x) \cdot a_j(x) + h_j \cdot a_{j+1}(x)$$

mit $h_j = \mathrm{cont}(r(x))$. Hierbei ist $a_{j+1}(x)$ primitiv und ein Teiler von $a_j(x)$, also auch ein Teiler von $a_{j-1}(x)$. Umgekehrt ist jeder gemeinsame Teiler von $a_j(x)$ und $a_2(x)$ primitiv und damit ein Teiler von $a_k(x)$.

3.3 Diskriminante und Resultante

Es sei R ein Integritätsbereich. Daraus kann man den **Quotientenkörper** K konstruieren, so wie man aus $\mathbb{Z}$ den Körper $\mathbb{Q}$ der rationalen Zahlen erzeugen kann. Man bildet einfach die Paarmenge $\{(a,b) \,:\, a \in R,\, b \in R - \{0\}\}$. Ein solches Paar steht für einen Bruch. Also muß man nur noch die richtigen Rechenregeln einführen und zwei Paare (a_1, b_1), (a_2, b_2) für äquivalent erklären, wenn $b_2 \cdot a_1 = a_2 \cdot b_1$ ist. Dann zeigt man, daß die Menge der Äquivalenzklassen einen Körper bildet.

Zu dem Quotientenkörper kann man den **algebraischen Abschluß** $\overline{K}$ konstruieren. Das ist der kleinste Oberkörper, in dem jedes Polynom aus K in Linearfakoren zerfällt. Beispielsweise ist der algebraische Abschluß von $\mathbb{R}$ der Körper $\mathbb{C}$ der komplexen Zahlen.

Es sei $p(x) \in R[x]$ und $p(x) = a \cdot \prod_{i=1}^{n}(x - \alpha_i)$ die Zerlegung von $p(x)$ in Linearfaktoren (in $\overline{K}$). Dann heißt

$$\operatorname{disc} p(x) \;=\; a^{2n-2} \cdot \prod_{1 \le i < j \le n} (\alpha_i - \alpha_j)^2$$

die **Diskriminante** von $p(x)$. Wozu kann man die Diskriminante gebrauchen?

Wir werden zeigen, daß sie eine Zahl in R ist und daß man sie mit Hilfe eines Algorithmus, der nur Zahlen in R benutzt, berechnen kann. Gleichzeitig gibt die Diskriminante aber Auskunft über die Nullstellen des Polynoms. Es gelten nämlich die folgenden Sätze:

3.9. Satz *Sei $p(x) \in R[x]$. Genau dann ist $\operatorname{disc} p(x) = 0$, wenn $p(x)$ in $\overline{K}$ mehrfache Nullstellen besitzt.*

3.10. Satz *Sei $p(x) \in \mathbb{Z}[x]$ ein quadratisches oder kubisches Polynom. Genau dann hat $p(x)$ nur einfache reelle Nullstellen, wenn $\operatorname{disc} p(x) > 0$ ist.*

Beweis: Übung: man beachte, daß für eine Nullstelle $\alpha \in \mathbb{C}$ von $p(x)$ auch die komplex konjugierte Zahl $\overline{\alpha}$ eine Nullstelle von $p(x)$ ist. ■

3.11. Satz *Sei $p(x) \in \mathbb{Z}[x]$. Falls $p(x)$ nur einfache, reelle Nullstellen besizt, ist die Diskriminante* $\operatorname{disc} p(x) > 0$.

Eine Verallgemeinerung der Diskriminante ist die **Resultante** zweier Polynome. Seien

$$p(x) = a \cdot \prod_{i=1}^{m} (x - \alpha_i) \quad \text{und} \quad q(x) = b \cdot \prod_{i=1}^{n} (x - \beta_i)$$

Polynome aus $R[x]$. Dann ist die Resultante von $p(x)$ und $q(x)$ definiert als

$$\operatorname{res}(p(x), q(x)) \;=\; a^n \cdot b^m \cdot \prod_{\substack{1 \le i \le m \\ 1 \le j \le n}} (\alpha_i - \beta_j).$$

Offensichtlich gilt die Gleichheit

$$\operatorname{res}(p(x), q(x)) \;=\; a^n \cdot q(\alpha_1) \cdot \ldots \cdot q(\alpha_m) \;=\; (-1)^{m \cdot n} \cdot b^m \cdot p(\beta_1) \cdot \ldots \cdot p(\beta_n).$$

Auch die Resultante zweier Polynome macht eine Aussage über die Nullstellen dieser beiden Polynome im algebraischen Abschluß von R. Es gilt nämlich der folgende Satz.

3.12. Satz *Seien $p(x)$, $q(x) \in R[x]$. Genau dann ist* $\operatorname{res}(p(x), q(x)) = 0$, *wenn $p(x)$ und $q(x)$ wenigstens eine gemeinsame Nullstelle in $\overline{K}$ haben.*

Beweis: Trivial. ■

Die **formale Ableitung** $p'(x)$ eines Polynoms $p(x) \in R[x]$ wird definiert wie aus der Analysis bekannt: Ist $p(x) = \sum_{i=0}^{n} a_i \cdot x^i$, so wird die (formale) Ableitung auf $p'(x) = \sum_{i=1}^{n} a_i \cdot i \cdot x^{i-1}$ gesetzt. Damit wird die Beziehung zwischen Resultante und Diskriminante klar:

3.13. Satz *Sei $p(x) \in R[x]$. Dann ist* $\operatorname{res}(p(x), p'(x)) \;=\; (-1)^{\deg p(x) \cdot (\deg p(x) - 1)/2} \cdot l(p(x)) \cdot \operatorname{disc} p(x)$.

Beweis: Sei $p(x) = a \cdot \prod_{j=1}^{n} (x - \alpha_j)$. Dann erhalten wir nach der Produktregel (die auch für formale Ableitungen gilt)

$$p'(x) \;=\; a \cdot \sum_{i=1}^{n} \prod_{j \ne i} (x - \alpha_j).$$

Damit ist $p'(\alpha_i) \;=\; a \cdot \prod_{j \ne i} (\alpha_i - \alpha_j)$ und wir erhalten daher

$$\begin{aligned}
\operatorname{res}(p(x), p'(x)) \;&=\; a^{n-1} \cdot p'(\alpha_1) \cdot \ldots \cdot p'(\alpha_n) \\
&=\; a^{n-1} \cdot a \cdot \prod_{j \ne 1} (\alpha_1 - \alpha_j) \cdot \ldots \cdot a \cdot \prod_{j \ne n} (\alpha_n - \alpha_j) \\
&=\; a^{2n-1} \cdot (-1)^{n \cdot (n-1)/2} \cdot \prod_{1 \le i < j \le n} (\alpha_i - \alpha_j)^2 \\
&=\; (-1)^{n \cdot (n-1)/2} \cdot a \cdot \operatorname{disc} p(x).
\end{aligned}$$

■

Wir benötigen damit nur noch einen Algorithmus zur Berechnung der Resultante zweier Polynome, um auch die Diskriminante eines Polynoms zu berechnen. Folgender Algorithmus berechnet sowohl die Resultante als auch den ggT zweier Polynome.

3.14. Algorithmus (Resultante)

Eingabe: $a(x), b(x) \in R[x]$.

Ausgabe: ein ggT von $a(x)$ und $b(x)$ und $\mathrm{res}(a(x), b(x))$.

Setze $u(x) = \mathrm{pp}(a(x)),\ v(x) = \mathrm{pp}(b(x)),\ g = h = s = 1$;
WHILE $\deg v(x) > 0$
DO
$\delta = \deg u(x) - \deg v(x)$;
berechne $r(x)$ mit $l(v(x))^{\delta+1} \cdot u(x) = q(x) \cdot v(x) + r(x)$;
IF $\deg u(x) \cdot \deg v(x)$ ungerade THEN $s = -s$ FI
$u(x) = v(x)$;
$v(x) = r(x)/(g \cdot h^{\delta})$;
$g = l(u(x))$;
$h = h^{1-\delta} \cdot g^{\delta}$;
OD
$\mathrm{ggT}(a(x), b(x)) = \mathrm{ggT}(\mathrm{cont}\, a(x), \mathrm{cont}\, b(x)) \cdot v(x)$;
Setze $\mathrm{res}(a(x), b(x)) \ = \ s \cdot l(a(x))^{\deg b(x)} \cdot l(b(x))^{\deg a(x)} \cdot h$;

Wir beweisen nun, daß der Algorithmus funktioniert. Es sei $u_0(x) = a(x)$, $u_1(x) = b(x)$, $u_3(x), \ldots$ die Folge der Polynome, die in dem Algorithmus entsteht. Weiter sei $\delta_j = n_j - n_{j+1}$, wobei $n_j = \deg u_j(x)$. Es sei außerdem $l_j = l(u_j(x))$ und h_j bzw. g_j die Werte von h bzw. g vor dem j-ten Durchlauf der WHILE-Schleife. Sei t der größte Index mit $\deg u_t(x) > 0$. Sind β_i die Nullstellen von $u_k(x)$, so gilt für $k \geq 1$:

$$\begin{aligned}
\mathrm{res}(u_{k-1}(x), u_k(x)) &= (-1)^{n_{k-1} n_k} \cdot l_k^{n_{k-1}} \cdot \prod_{1 \leq i \leq n_k} u_{k-1}(\beta_i) \\
&= (-1)^{n_{k-1} n_k} \cdot l_k^{n_{k-1}} \cdot \prod_{1 \leq i \leq n_k} \frac{r_{k+1}(\beta_i)}{l_k^{\delta_{k-1}+1}} \\
&= (-1)^{n_{k-1} n_k} \cdot l_k^{n_{k-1} - n_k(\delta_{k-1}+1)} \cdot \prod_{1 \leq i \leq n_k} r_{k+1}(\beta_i) \\
&= (-1)^{n_{k-1} n_k} \cdot l_k^{n_{k-1} - n_k(\delta_{k-1}+1) - n_{k+1}} \cdot \mathrm{res}(u_k(x), g_{k-1} \\
&\quad \cdot h_{k-1}^{\delta_{k-1}} \cdot u_{k+1}(x)).
\end{aligned}$$

Benutzen wir nun die Gleichung $\mathrm{res}(a(x), c \cdot b(x)) = c^{\deg a(x)} \cdot \mathrm{res}(a(x), b(x))$ und die Identitäten $g_k = l_k$ und $h_k = h_{k-1}^{1-\delta_{k-1}} \cdot g_k^{\delta_{k-1}}$ für $k \geq 1$, so erhalten wir daraus

$$\mathrm{res}(u_{k-1}(x), u_k(x)) \ = \ (-1)^{n_{k-1} n_k} \cdot \frac{g_{k-1}^{n_k} \cdot h_{k-1}^{n_{k-1}-1}}{g_k^{n_{k+1}} \cdot h_k^{n_k - 1}} \cdot \mathrm{res}(u_k(x), u_{k+1}(x)).$$

Mit $n_{t+1} = 0$ und so $\delta_t = n_t$ folgt

$$\begin{aligned}
\operatorname{res}(a(x), b(x)) &= (-1)^{\sum_{1\le k\le t} n_{k-1}n_k} \cdot h_t^{1-\delta_t} \cdot \operatorname{res}(u_t(x), u_{t+1}(x)) \\
&= (-1)^{\sum_{1\le k\le t} n_{k-1}n_k} \cdot h_t^{1-n_t} \cdot l_{t+1}^{n_t} \\
&= (-1)^{\sum_{1\le k\le t} n_{k-1}n_k} \cdot h_{t+1},
\end{aligned}$$

woraus die Korrektheit des Algorithmus folgt.

Wir wollen noch zeigen, daß die Resultante immer eine Zahl aus dem Ring R ist, obwohl sie über dem algebraischen Abschluß von R definiert wird. Dazu beachte man den folgenden Satz:

3.15. Satz *Die Resultante von $a(x) = a_n \cdot x^n + \ldots + a_0$ und $b(x) = b_m \cdot x^m + \ldots + b_0$ ist die Determinante der Sylvester-Matrix*

$$\begin{pmatrix}
a_n & a_{n-1} & \ldots & a_0 & 0 & \ldots & 0 \\
0 & a_n & \ldots & a_1 & a_0 & 0 & 0 \\
\vdots & \ddots & \ddots & & \ddots & \ddots & \vdots \\
0 & \ldots & 0 & a_n & \ldots & a_1 & a_0 \\
b_m & b_{m-1} & \ldots & b_0 & 0 & \ldots & 0 \\
0 & b_m & \ldots & b_1 & b_0 & 0 & 0 \\
\vdots & \ddots & \ddots & & \ddots & \ddots & \vdots \\
0 & \ldots & 0 & b_m & \ldots & b_1 & b_0
\end{pmatrix}.$$

Dabei werden die Koeffizienten von $a(x)$ in den ersten m Zeilen verwendet und die Koeffizienten von $b(x)$ in den darauffolgenden n Zeilen.

Beweis: Sei im algebraischen Abschluß $a(x) = a_n \cdot \prod_{i=1}^n (x - \alpha_i)$ und $b(x) = b_m \cdot \prod_{i=1}^m (x - \beta_i)$. Angenommen, die Nullstellen sind paarweise verschieden. Betrachte die Vandermonde-Matrix $V = (v_{ij})_{1\le i,j\le n+m}$, die folgendermaßen definiert ist:

$$v_{ij} = \begin{cases} \beta_j^{n+m-i} & \text{für } 1 \le j \le m. \\ \alpha_{j-m}^{n+m-i} & \text{für } m+1 \le j \le m+n. \end{cases}$$

Beachtet man die Berechnungsregel für die Vandermonde-Determinante

$$\det(x_i^j)_{1\le i\le n, 0\le j\le n-1} = \prod_{1\le i<j\le n} (x_j - x_i)$$

für die Vandermonde-Determinante und die spezielle Gestalt der Matrix V, so erkennt man, daß

$$\det V = \prod_{1\le i<j\le m} (\beta_i - \beta_j) \cdot \prod_{1\le i<j\le n} (\alpha_i - \alpha_j) \cdot \prod_{\substack{1\le i\le n \\ 1\le j\le m}} (\beta_j - \alpha_i).$$

Ist M die Sylvestermatrix aus dem Satz, so gilt andererseits

$$M \cdot V \;=\; \begin{pmatrix} a(\beta_1)\cdot\beta_1^{m-1} & \dots & a(\beta_m)\cdot\beta_m^{m-1} & 0 & \dots & 0 \\ a(\beta_1)\cdot\beta_1^{m-2} & \dots & a(\beta_m)\cdot\beta_m^{m-2} & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ a(\beta_1)\cdot\beta_1^{0} & \dots & a(\beta_m)\cdot\beta_m^{0} & 0 & \dots & 0 \\ 0 & \dots & 0 & b(\alpha_1)\cdot\alpha_1^{n-1} & \dots & b(\alpha_n)\cdot\alpha_n^{n-1} \\ 0 & \dots & 0 & b(\alpha_1)\cdot\alpha_1^{n-2} & \dots & b(\alpha_n)\cdot\alpha_n^{n-2} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & b(\alpha_1)\cdot\alpha_1^{0} & \dots & b(\alpha_n)\cdot\alpha_n^{0} \end{pmatrix}.$$

Also gilt nach dem Determinantensatz für Blockmatrizen

$$\det(M\cdot V) \;=\; a(\beta_1)\cdot\ldots\cdot a(\beta_m)\cdot b(\alpha_1)\cdot\ldots\cdot b(\alpha_n)\cdot \prod_{1\le i<j\le m}(\beta_i-\beta_j)\cdot \prod_{1\le i<j\le m}(\alpha_i-\alpha_j).$$

Nach dem Determinantenmultiplikationssatz können wir folgern, daß

$$\det M \cdot \prod_{\substack{1\le i\le n\\ 1\le j\le m}} (\alpha_i-\beta_j) \;=\; a(\beta_1)\cdot\ldots\cdot a(\beta_m)\cdot b(\alpha_1)\cdot\ldots\cdot b(\alpha_n)$$

ist. Nun gilt aber $\prod_{\substack{1\le i\le n\\ 1\le j\le m}}(\beta_j-\alpha_i) = \frac{1}{a_n^m}\cdot\prod_{1\le j\le m} a(\beta_j)$ und damit erhalten wir

$$\det M \;=\; a_n^m \cdot b(\alpha_1)\cdot\ldots\cdot b(\alpha_n) = \operatorname{res}(a(x),b(x)).$$

■

Da die einzigen Einträge ungleich Null in der ersten Spalte der Matrix M nur a_n und b_m sind, erkennen wir leicht, daß die Resultante $\operatorname{res}(a(x),b(x))$ in R ist und zusätzlich noch von $\operatorname{ggT}(a_n,b_m)$ geteilt wird. Im Falle der Diskriminante können wir daher folgern, daß a_n ein Teiler von $\operatorname{res}(a(x),a'(x))$ ist und damit folgt mit Satz 3.13, daß auch die Diskriminante ein Element des Grundrings R ist.

Kapitel 4

Lineare Algebra

4.1 Reduktionen auf Grundprobleme

Es sei R ein kommutativer Ring mit Eins. Wir wollen lineare Algebra über R treiben und setzen dazu voraus, daß wir die Operationen Addition, Negation, Multiplikation und Vergleich in R ausführen können.

Sei $n \in \mathbb{N}$. Dann ist $R^n = \{(v_1, \ldots, v_n) \ : \ v_i \in R, \ 1 \leq i \leq n\}$. Wir bezeichnen ein Element $(v_1, \ldots, v_n) \in R^n$ im folgenden auch mit $\underline{v}$. Mittels koordinatenweiser Addition wird R^n zu einer additiven Gruppe. Außerdem kann man in natürlicher Weise eine skalare Multiplikation erklären. Diese ist distributiv, d.h. $(r_1+r_2)\cdot\underline{v} = r_1\cdot\underline{v}+r_2\cdot\underline{v}$ und $r\cdot(\underline{v}_1+\underline{v}_2) = r\cdot\underline{v}_1 + r\cdot\underline{v}_2$. Sie ist weiterhin assoziativ, d.h. $r_1\cdot(r_2\cdot\underline{v}) = (r_1\cdot r_2)\cdot\underline{v}$. Und sie ist verträglich mit der Eins ($1\cdot\underline{v} = \underline{v}$). Damit wird R^n zum R-Modul. Weil wir noch eine Basis $\underline{e}_1, \ldots, \underline{e}_n$, nämlich die Standardbasis, kennen, ist R^n sogar ein freier R-Modul vom Rang n.

4.1. Satz *Ein System* $\{\underline{v}_1, \ldots, \underline{v}_n\} \in R^n$ *ist genau dann eine Basis des* R^n*, wenn die Determinante* $\det(\underline{v}_1, \ldots, \underline{v}_n)$ *eine Einheit in* R *ist.*

Beweis: Sei $V = (\underline{v}_1, \ldots, \underline{v}_n)$ die Matrix, die die Vektoren $\underline{v}_1, \ldots, \underline{v}_n$ als Spalten hat. Das System $\{\underline{v}_1 \ldots, \underline{v}_n\}$ ist genau dann eine Basis, wenn sich durch eine Linearkombination der Spalten jedes Element der Standardbasis erzeugen läßt, d.h. (in Matrixschreibweise) wenn eine Matrix $T \in R^{n\times n}$ existiert mit $V \cdot T = E_n$. Hieraus folgt $\det V \cdot \det T = 1$, also insbesondere muß $\det V$ eine Einheit sein. ∎

Sei $\{\underline{b}_1, \ldots, \underline{b}_n\}$ Basis von R^n und $\{\underline{c}_1, \ldots, \underline{c}_m\}$ eine Basis von R^m. Sei eine Abbildung

$$\begin{aligned} \varphi : R^n &\longrightarrow R^m \\ \underline{b}_j &\longmapsto \sum_{i=1}^{m} t_{ij} \cdot \underline{c}_i \end{aligned}$$

gegeben. Dann heißt $T = (t_{ij}) \in R^{m\times n}$ **die Matrix von** φ bezüglich der gewählten Basen.

Wir werden Moduln im folgenden immer durch Basen darstellen. Auch freie Untermoduln können durch eine Basis dargestellt werden. Folgende algorithmische Fragen kann man dann stellen:

1. **(Kern)** Bestimme wenn möglich
$$\mathrm{Kern}\,\varphi \;=\; \{\underline{v} \in R^n : \varphi(\underline{v}) = \underline{0}\}.$$
2. **(Bild)** Bestimme wenn möglich
$$\mathrm{Im}\,\varphi \;=\; \varphi(R^n).$$
3. **(Urbild)** Für $\underline{w} \in R^m$ bestimme wenn möglich ein Urbild in R^n, d.h. ein Element $\underline{v} \in R^n$ mit $\varphi(\underline{v}) = \underline{w}$.

Diese Probleme kann man auch mit Hilfe von Matrizen formulieren:

1'. Finde für eine Matrix $T \in R^{m\times n}$ wenn möglich die Lösungsmenge von $T \cdot \underline{x} = \underline{0}$.

3'. Löse für eine Matrix $T \in R^{m\times n}$ wenn möglich $T \cdot \underline{x} = \underline{w}$.

Der Vorteil der ersten drei Formulierungen ist der, daß man basisunabhängige Objekte sucht. Man nennt das auch die invariante Formulierung der Probleme. Natürlich werden wir zur konkreten Lösung auf die Darstellung der Homomorphismen durch Matrizen zurückgreifen müßen. Aber wir müssen bei der gewöhnlichen Lösung von Gleichungssystemen ja auch auf die interne Zahldarstellung zurückgreifen.

Ist $n = m$, so nennt man φ einen **Endomorphismus**. Sei im folgenden $T \in R^{n\times n}$ die Matrix des Endomorphismus φ, wenn wir im Bild- und im Urbildbereich die gleiche Basis benutzen. Folgende weitere Fragen kann man dann stellen:

4. **(Determinante)** Berechne die Determinante von φ bzw. $\det T$.
5. **(charakteristisches Polynom)** Berechne das charakteristische Polynom von φ, d.h. $\det(T - x \cdot I_n)$.
6. **(Minimalpolynom)** Berechne ein Minimalpolynom von φ.

Weitere mögliche Fragen beziehen sich auf freie Untermoduln M, N von R^n. Wir nehmen dabei an, daß wir Basen von M und N kennen.

7. **(Durchschnitt)** Finde wenn möglich eine Basis von $M \cap N$.
8. **(Summe)** Finde wenn möglich eine Basis von $M + N = \{m + n \,:\, m \in M,\, n \in N\}$.
9. **(Unterraum)** Für $\{\underline{b}_1, \ldots, \underline{b}_k\} \subseteq R^n$ bestimme wenn möglich eine Basis des zugehörigen erzeugten Unterraums $\underline{b}_1 \cdot R + \ldots + \underline{b}_k \cdot R$.

10. **(Basisergänzung)** Für $\{\underline{b}_1, \ldots, \underline{b}_k\} \subseteq R^n$ bestimme wenn möglich eine Basisergänzung, d.h. Elemente $\underline{b}_{k+1}, \ldots, \underline{b}_n$, so daß $\{\underline{b}_1, \ldots, \underline{b}_n\}$ eine Basis von R^n ist.

11. **(Faktormodul)** Bestimme wenn möglich eine Basis für R/N.

Bevor wir auf konkrete Lösungsstrategien eingehen, erläutern wir einige Reduktionen, die möglich sind. Dabei gilt die Berechnung des Kerns, des Bildes, der Determinante und die Basisergänzung als Grundaufgaben, für die wir später Lösungen angeben.

- Lösung von 3.(Urbild) durch Berechnung des Kerns:

  Betrachte dazu für eine Basis $\{\underline{b}_1, \ldots, \underline{b}_n\}$ von R^n die Abbildung

$$\begin{array}{rcl} \psi : R^{n+1} & \longrightarrow & R^m \\ (x_1, \ldots, x_{n+1}) & \longmapsto & x_1 \cdot \varphi(\underline{b}_1) + \ldots + x_n \cdot \varphi(\underline{b}_n) + x_{n+1} \cdot \underline{w}. \end{array}$$

  Finde im Kern von ψ ein Element $(x_1, \ldots, x_{n+1})$, in dem x_{n+1} eine Einheit ist. Dann ist $-\frac{x_1}{x_{n+1}} \cdot \underline{b}_1 - \ldots - \frac{x_n}{x_{n+1}} \cdot \underline{b}_n \in R^n$ ein Urbild von $\underline{w}$, wie man leicht zeigen kann.

- Lösung von 5.(charakteristisches Polynom) kann trivialerweise durch Berechnung einer Determinante erfolgen, obwohl das nicht notwendig die beste Strategie ist.

- Lösung von 6.(Minimalpolynom) durch Berechnung eines Kerns:

  Ein Minimalpolynom von φ ist ein normiertes Polynom $p(x) \in R[x]$ von minimalem Grad mit $p(\varphi) = 0$. Wir wissen, daß das charakteristische Polynom c_φ von φ normiert und $c_\varphi(\varphi) = 0$ ist. Weiterhin besitzt c_φ maximal den Grad n. Betrachte also die Abbildung

$$\begin{array}{rcl} R^{n+1} & \longrightarrow & R^{n\times n} \\ (r_0, \ldots, r_n) & \longmapsto & \sum_{i=0}^{n} r_i \cdot T^i. \end{array}$$

  Ein Minimalpolynom entspricht einem Element im Kern, das maximal viele Nullen "am Ende" hat.

- Lösung von 7.(Durchschnitt) durch Berechnung eines Kerns und eines Bildes:

  Sei $\{\underline{b}_1, \ldots, \underline{b}_k\}$ eine Basis von M und $\{\underline{c}_1, \ldots, \underline{c}_l\}$ eine Basis von N. Betrachte die Abbildung

$$\begin{array}{rcl} R^{k+l} & \longrightarrow & R^n \\ (x_1, \ldots, x_k, y_1, \ldots, y_l) & \longmapsto & \sum_{i=1}^{k} x_i \cdot \underline{b}_i - \sum_{j=1}^{l} y_j \cdot c_j. \end{array}$$

  Wenn wir den Kern dieser Abbildung bestimmt haben, dann kennen wir ein Erzeugendensystem für $M \cap N$. Sei nämlich $\{\underline{d}_1, \ldots, \underline{d}_s\}$ eine Basis für den Kern und $d_j = (d_{1j}, \ldots, d_{k+l,j})$ für $1 \leq j \leq s$. Dann wird $M \cap N$ erzeugt von

$$\underline{d}_j = \left\{ \sum_{i=1}^{k} d_{ij} \cdot \underline{b}_i \right\} \quad \text{für } 1 \leq j \leq s.$$

Um daraus eine Basis zu berechnen, sollte man das Bild der Abbildung

$$\begin{array}{ccc} R^s & \longrightarrow & R^n \\ (x_1, \ldots, x_s) & \longmapsto & \sum_{i=1}^{s} x_i \cdot \underline{d}_i \end{array}$$

berechnen.

- Lösung von 8.(Summe) durch Berechnung eines Bildes:

  Seien die Voraussetzungen wie bei der Lösung von 7. Betrachte die Abbildung

$$\begin{array}{ccc} R^{k+l} & \longrightarrow & R^n \\ (x_1, \ldots, x_k, y_1, \ldots, y_l) & \longmapsto & \sum_{i=1}^{k} x_i \cdot \underline{b}_i + \sum_{j=1}^{l} y_j \cdot \underline{c}_j. \end{array}$$

  Das Bild dieser Abbildung ist $M + N$.

- Lösung von 11.(Faktormodul) durch Basisergänzung:

  Sei $\{\underline{c}_1, \ldots, \underline{c}_l\}$ eine Basis von N. Ergänze dies zu einer Basis von R^n, etwa zu $\{\underline{c}_1, \ldots, \underline{c}_l, \underline{v}_1, \ldots, \underline{v}_{n-l}\}$. Dann bildet $\{\underline{v}_1 + N, \ldots, \underline{v}_{n-l} + N\}$ eine Basis von R^n/N. Sicherlich bilden diese Vektoren ein Erzeugendensystem. Die lineare Unabhängigkeit läßt sich dann leicht aus der linearen Unabhängigkeit der Basis $\{\underline{c}_1, \ldots, \underline{c}_l, \underline{v}_1, \ldots, \underline{v}_{n-l}\}$ von R^n folgern.

4.2 Berechnung des Kerns und des Bildes

Wenden wir uns daher nun der Lösung der Basisprobleme zu. Wir interessieren uns dabei im wesentlichen für zwei Typen von Grundringen. Der erste Typ ist der des **Hauptidealrings**. Sei R ein Integritätsbereich. Eine Teilmenge A von R heißt ein **Ideal** von R, falls A eine additive Untergruppe von R ist und falls $R \cdot A \subseteq A$ gilt. Ein Hauptidealring ist ein Ring, in dem jedes Ideal von einem Element erzeugt wird. Man kann zeigen, daß jeder euklidische Ring auch ein Hauptidealring ist.

Im zweiten Fall sind wir schon zufrieden, wenn wir einen Nullteiler im Ring gefunden haben. Wir nehmen an, daß wir ein von Null verschiedenes Element invertieren können. Ist dies nicht der Fall, so soll man bei der Invertierung einen Nullteiler finden. Als Beispiel für einen solchen Ring betrachte man $\mathbb{Z}/m\mathbb{Z}$.

Sei nun R ein *HIR*. Sei $\varphi \in \mathrm{Hom}_R(R^n, R^m)$ und sei $T \in R^{m \times n}$ die Matrix von φ bezgl. den Standardbasen von R^n und R^m. Um den Kern und das Bild von φ zu bestimmen, transformieren wir T in folgende **Spaltenstufen-Form**:

$$T' = \begin{pmatrix} 0 & \ldots & 0 & * & \ldots & * \\ 0 & \ldots & 0 & 0 & * & * \\ 0 & \ldots & 0 & 0 & \ldots & * \\ 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \ldots & 0 & 0 & \ldots & 0 \end{pmatrix}.$$

Diese Spaltenstufen-Form wird charakterisiert durch die folgende Regel: sei $j_0 = \min\{j;\ \exists 1 \leq i \leq m \text{ mit } t_{ij} \neq 0\}$. Falls $j \geq j_0$ ist, gibt es in Spalte j ein von 0 verschiedenes Element. Sei weiterhin $i_j = \max\{i : t_{ij} \neq 0\}$. Dann gilt $i_j < i_{j+1}$ für alle $j_0 \leq j < n$.

Die Umwandlung der Matrix T in Spaltenstufen-Form findet durch elementare Spaltenoperationen statt, d.h. es gibt eine Matrix $V \in GL(n, R)$ mit $T' = T \cdot V$.

4.2. Satz *Die von Null verschiedenen Spalten der Matrix T' (etwa die letzten $n - t$ Spalten) bilden eine Basis des Bildes von φ und die ersten t Spalten von V bilden eine Basis des Kerns des Homomorphismus φ.*

Beweis: Wie man durch einfaches Ausrechnen leicht feststellen kann, ist $T \cdot V$ die Abbildungsmatrix bzgl. der transformierten Basis $(\underline{e}_1, \ldots, \underline{e}_n) \cdot V = V$. Die letzten $n-t$ Spalten dieser transformierten Matrix erzeugen das Bild von φ und sind linear unabhängig. Also bilden sie eine Basis des Bildes. Damit hat der Kern die Dimension t. Aber die ersten t Vektoren der neuen Basis $(\underline{e}_1, \ldots, \underline{e}_n) \cdot V$, d.h. die ersten t Spalten von V, gehören zum Kern und sind linear unabhängig. ■

Wie soll man diese Transformation jetzt praktisch durchführen? Dazu führen wir elementare Spaltenumformungen durch, also solche, die die Determinante eins haben. Man muß aber beachten, daß Division dabei i.a. verboten ist.

Seien also die letzten $n - i$ Zeilen bereits in der richtigen Form. Angenommen, es werden in diesen letzten $n - i$ Zeilen in allen Spalten jenseits der j-ten von Null verschiedene Elemente gefunden. Jetzt suchen wir also in der j-ten Spalte ein solches Element. Hierzu prüfen wir, ob in der i-ten Zeile in Spalte 1 bis j vielleicht ein von Null verschiedenes Element ist; wenn nicht, sind wir mit der i-ten Zeile fertig. Andernfalls sorgen wir durch Spaltenvertauschung dafür, daß Eintrag (i, j) von Null verschieden wird. Diesen Eintrag benutzen wir jetzt, um die anderen Einträge zu eliminieren. Dies kann man in einem HIR folgendermaßen durchführen:

Angenommen $t_{ij'} \neq 0$ mit $1 \leq j' < j$. Dann erzeugen t_{ij} und $t_{ij'}$ ein Hauptideal $d \cdot R$. Hierbei gilt $t_{ij}, t_{ij'} \in d \cdot R$, also $d \,|\, t_{ij}$ und $d \,|\, t_{ij'}$. Außerdem kann man Elemente $x, y \in R$ finden mit

$$x \cdot t_{ij} + y \cdot t_{ij'} = d \quad \text{bzw.} \quad x \cdot \frac{t_{ij}}{d} + y \cdot \frac{t_{ij'}}{d} = 1.$$

Dann setzen wir $v = \frac{t_{ij}}{d}$ und $u = -\frac{t_{ij'}}{d}$. Somit ist die Determinante der Transformation

$$\begin{pmatrix} x & u \\ y & v \end{pmatrix}$$

eins. Also haben wir es mit einer unimodularen Transformation zu tun. Seien $\underline{t}_j$ und $\underline{t}_{j'}$ die entsprechenden Spalten von T. Dann setzen wir

$$(\underline{t}_j, \underline{t}_{j'}) \ = \ (\underline{t}_j, \underline{t}_{j'}) \cdot \begin{pmatrix} x & u \\ y & v \end{pmatrix}.$$

Somit gilt

$$\begin{aligned} t_{ij}^{(neu)} &= x \cdot t_{ij} + y \cdot t_{ij'} = d \\ t_{ij'}^{(neu)} &= u \cdot t_{ij} + v \cdot t_{ij'} = 0 \end{aligned}$$

und wir haben ein weiteres Element der i-ten Zeile links der j-ten Spalte reduziert. Dieses Verfahren führt zu dem folgenden Algorithmus:

4.3. Algorithmus (Spaltenstufen-Form)

Eingabe: $T \in R^{m \times n}$.

Ausgabe: $T' \in R^{m \times n}$ in Spaltenstufen-Form.

Setze $j_0 = n$;

FOR $(i = m,\, i \geq 1,\, i = i-1)$

  DO

  $j = j_0$;

  WHILE $j \geq 1$ and $t_{ij} = 0$ DO $j = j - 1$ OD

  IF $j \neq 0$ THEN

    vertausche Spalte j und Spalte j_0;

    FOR $(j = j_0 - 1,\ j \geq 1,\ j = j-1)$ DO eliminiere t_{ij} OD

    $j_0 = j_0 - 1$;

    FI

  OD

Falls R ein HIR ist, führt man die Elimination genau nach dem oben beschriebenen Verfahren durch. Ist R aber ein Körper bzw. ein Ring des zweiten Typus, so geht der Eliminationsschritt einfacher. Berechne einfach

$$(\underline{t}_j, \underline{t}_{j_0}) \;=\; (\underline{t}_j, \underline{t}_{j_0}) \cdot \begin{pmatrix} 1 & 0 \\ -t_{ij}/t_{ij_0} & 1 \end{pmatrix}.$$

Gleichzeitig kann man in diesem Algorithmus direkt noch die Matrix V mitberechnen, wenn man sich alle Substitutionen (auch Spaltenvertauschungen), die man während des Algorithmus macht, merkt, indem man sie auf eine weitere Matrix anwendet. Diese Matrix muß natürlich mit der Einheitsmatrix initialisiert werden.

Ein Problem ist es, daß die Zwischenergebnisse in den ersten Zeilen sehr groß werden können. In euklidischen Ringen gibt es eine Variante. Man setzt dort t_{ij_0} immer auf das größte Element der Zeile (bezüglich der Höhenfunktion) und reduziert dies dann mit dem zweitgrößten.

4.3 Basisergänzung

Da die Berechnung der Determinante einer Matrix einfach durch eine Gauss-Elimination durchgeführt werden kann, was bekannt sein sollte, wenden wir uns nun noch der Basisergänzung zu. Seien die Vektoren $\underline{v}_1, \ldots, \underline{v}_k \in R^n$ linear unabhängig. Wir wollen prüfen, ob sich diese Vektoren zu einer Basis von R^n ergänzen lassen und wir wollen diese gegebenenfalls finden. Dabei unterscheiden wir wieder die zwei Typen von Grundringen.

1. Fall: $R - \{0\}$ hat nur Nullteiler oder Einheiten und wir sind zufrieden damit, einen Nullteiler zu finden. In diesem Fall kann man das Standardverfahren benutzen:

4.4. Algorithmus (Basisergänzung)

Eingabe: $\underline{v}_1, \ldots, \underline{v}_k \in R^n$, linear unabhängig und $\neq \underline{0}$.

Ausgabe: Ein Nullteiler von R oder eine Basis $\{\underline{v}_1, \ldots, \underline{v}_n\}$ von R^n.

Setze $\underline{w}_j = \underline{e}_j^{(n)}$ für $1 \leq j \leq n$.
FOR $(j = 1, j \leq k, j = j + 1)$
  DO
  sei $\underline{v}_j = x_1 \cdot \underline{w}_1 + \ldots + x_n \cdot \underline{w}_n$
  suche ein $i > j$ mit $x_i \neq 0$;
  IF x_i existiert nicht THEN $i = j$ FI
  IF x_i Nullteiler THEN return(x_i) FI
  ersetze $\underline{w}_i$ durch $\underline{v}_j$;
  vertausche $\underline{w}_i$ und $\underline{w}_j$;
  OD
Return $\{\underline{w}_1, \ldots, \underline{w}_n\}$;

2. Fall: $R - \{0\}$ ist ein HIR. In diesem Fall ist es nicht mehr klar, daß linear unabhängige Vektoren sich zu einer Basis ergänzen lassen. Beispielsweise ist 2 linear unabhängig in $\mathbb{Z}$, aber 2 ist keine Basis von $\mathbb{Z}$ und läßt sich auch nicht zu einer solchen ergänzen. Wir gehen nun folgendermaßen vor. Es sei bereits eine Basis $\underline{w}_1, \ldots, \underline{w}_n$ gefunden mit $\underline{w}_{j'} = \underline{v}_{j'}$ für $j' < j$. Jetzt soll $\underline{v}_j$ in die Basis eingefügt werden. Hierzu betrachte ich die Darstellung

$$\underline{v}_j = \sum_{i=1}^{n} v_{ij} \cdot \underline{w}_i.$$

4.5. Lemma *Genau dann kann $\underline{v}_j$ in die Basis $\{\underline{w}_1, \ldots, \underline{w}_n\}$ eingefügt werden, wenn* $\mathrm{ggT}(v_{jj}, \ldots, v_{nj}) \in \mathbb{R}^*$ *ist.*

Beweis: Nehmen wir zuerst an, daß $\underline{v}_1, \ldots, \underline{v}_j, \underline{w}'_{j+1}, \ldots, \underline{w}'_n$ eine Basis ist. Dann gilt

$$(\underline{v}_1, \ldots, \underline{v}_j, \underline{w}'_{j+1}, \ldots, \underline{w}'_n) \;=\; (\underline{v}_1, \ldots, \underline{v}_{j-1}, \underline{w}_j, \ldots, \underline{w}_n) \cdot T,$$

wobei

$$T = \begin{pmatrix} 1 & \dots & 0 & v_{1j} & * & \dots & * \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & v_{j-1,j} & * & \dots & * \\ 0 & \dots & 0 & v_{j,j} & * & \dots & * \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & v_{nj} & * & \dots & * \end{pmatrix}.$$

Sicherlich ist $\det T \in R^*$ und nach den Determinantenregeln für Blockmatrizen ist auch die Determinante der rechten unteren Blockmatrix eine Einheit. Daher ist auch $\mathrm{ggT}(v_{jj}, \dots, v_{nj}) \in R^*$, denn sonst könnte man diesen Faktor herausziehen und die "untere" Determinante könnte dann keine Einheit sein.

Sei nun umgekehrt $\mathrm{ggT}(v_{jj}, \dots, v_{nj}) \in R^*$. Ich beschreibe eine Prozedur, die den Vektor $\underline{v}_j$ in die Basis einfügt. Zuerst betrachten wir die Vektoren $\underline{w}_1, \dots, \underline{w}_n, \underline{v}_j$. Für diese Vektoren haben wir die Relation

$$v_{1j} \cdot \underline{w}_1 + \dots + v_{nj} \cdot \underline{w}_n + v_{n+1,j} \cdot \underline{w}_{n+1} = 0.$$

Hierin wurde $v_{n+1,j} = -1$ und $\underline{w}_{n+1} = \underline{v}_j$ gesetzt. Der Koeffizientenvektor $(v_{1j}, \dots, v_{n+1,j})$ heißt **Relation** für das Tupel $(\underline{w}_1, \dots, \underline{w}_{n+1})$. Sei $d_k = \mathrm{ggT}(v_{jj}, \dots, v_{j+k,j})$. Wir ersetzen im k-ten Schritt die Vektoren $\underline{w}_{j+k-1}$ und $\underline{w}_{j+k}$ durch neue Vektoren, so daß die neue Relation

$$(v_{1j}, \dots, v_{j-1,j}, 0, \dots, 0, d_k, v_{k+j+1,j}, \dots, v_{n+1,j})$$

ist. Dabei steht d_k an der Stelle $k+j$. Sobald d_k eine Einheit ist, hat man dann

$$\underline{w}_k = -d_k^{-1} \cdot \Big(v_{1j} \cdot \underline{w}_1 + \dots + v_{j-1,j} \cdot \underline{w}_{j-1} + v_{k+j+1,j} \cdot \underline{w}_{j+k+1} + \dots + v_{n+1,j} \cdot \underline{w}_{n+1} \Big)$$

und damit kann man $\underline{w}_k$ weglassen, ohne die Eigenschaft, ein Erzeugendensystem zu haben, zu ändern. Damit hat man also eine neue Basis $\{\underline{w}_1, \dots, \underline{w}_{k-1}, \underline{w}_{k+1}, \dots, \underline{w}_{n+1}\}$ konstruiert, der $\underline{v}_j = \underline{w}_{n+1}$ angehört.

Wie funktioniert der Ersetzungsschritt? Wir benötigen eine unimodulare Transformation, so daß

$$d_{k-1} \cdot \underline{w}_{k+j-1}^{(alt)} + v_{j+k,j} \cdot \underline{w}_{k+j}^{(alt)} \;=\; 0 \cdot \underline{w}_{k+j-1}^{(neu)} + d_k \cdot \underline{w}_{k+j}^{(neu)}$$

gilt. Um eine solche unimodulare Transformation zu erhalten, bestimmen wir die Darstellung von $\mathrm{ggT}(d_{k-1}, v_{k+j,j})$, etwa $d_k = x \cdot d_{k-1} + y \cdot v_{k+j,j}$. Dann setzen wir

$$(\underline{w}_{k+j-1}, \underline{w}_{k+j}) \;=\; (\underline{w}_{k+j-1}, \underline{w}_{k+j}) \cdot \begin{pmatrix} y & \frac{d_{k-1}}{d_k} \\ -x & \frac{v_{k+j,j}}{d_k} \end{pmatrix}.$$

Damit sind alle Bedingungen erfüllt, wie man durch Nachrechnen beweisen kann. Man beachte noch, daß nach Voraussetzung die Bedingung $d_k \in R^*$ nach endlich vielen Schritten erfüllt ist. ∎

4.6. Beispiel Ergänze $\underline{v} = \begin{pmatrix} 6 \\ 10 \\ 15 \end{pmatrix}$ zu einer Basis von $\mathbb{Q}^3$ bzw. $\mathbb{Z}^3$:

Setze $\underline{w}_1 = \underline{e}_1$, $\underline{w}_2 = \underline{e}_2$, $\underline{w}_3 = \underline{e}_3$. Stelle $\underline{v}$ mit der Basis $\{\underline{w}_1, \underline{w}_2, \underline{w}_3\}$ dar:

$$\underline{v} \;=\; 6 \cdot \underline{w}_1 + 10 \cdot \underline{w}_2 + 15 \cdot \underline{w}_3.$$

Da 10 eine Einheit in $\mathbb{Q}$ ist, läßt sich $\underline{w}_2$ darstellen durch $\underline{w}_1, \underline{w}_3$ und $\underline{v}$, nämlich $\underline{w}_2 = \frac{1}{10} \cdot (\underline{v} - 6 \cdot \underline{w}_1 - 15 \cdot \underline{w}_3)$. Damit ist $\{\underline{v}, \underline{w}_1, \underline{w}_3\}$ eine Basis von $\mathbb{Q}^3$.

Versuchen wir nun, $\underline{v}$ zu einer Basis von $\mathbb{Z}^3$ zu ergänzen. Dann ist $d_1 = \mathrm{ggT}(6, 10) = 2 = 2 \cdot 6 - 1 \cdot 10$. Substituiere

$$(\underline{w}_1, \underline{w}_2) \;=\; (\underline{w}_1, \underline{w}_2) \cdot \begin{pmatrix} -1 & 3 \\ -2 & 5 \end{pmatrix} \;=\; \begin{pmatrix} -1 & 3 \\ -2 & 5 \\ 0 & 0 \end{pmatrix}.$$

Da $d_1 = 2$ keine Einheit in $\mathbb{Z}$ ist, müssen wir das Verfahren fortsetzen. Dann erhalten wir $d_2 = \mathrm{ggT}(2, 15) = 1 = 8 \cdot 2 - 1 \cdot 15$. Damit substitieren wir

$$(\underline{w}_2, \underline{w}_3) \;=\; (\underline{w}_2, \underline{w}_3) \cdot \begin{pmatrix} -1 & 2 \\ -8 & 15 \end{pmatrix} \;=\; \begin{pmatrix} -3 & 6 \\ -5 & 10 \\ -8 & 15 \end{pmatrix}.$$

d_2 ist eine Einheit von $\mathbb{Z}$ und damit sind wir fertig. Eine gewünschte Basis von $\mathbb{Z}^3$ ist

$$\begin{pmatrix} 6 \\ 10 \\ 15 \end{pmatrix}, \begin{pmatrix} -1 \\ -2 \\ 0 \end{pmatrix}, \begin{pmatrix} -3 \\ -5 \\ -8 \end{pmatrix}.$$

Kapitel 5

Gitter

5.1 Gitter und quadratische Formen

Es sei $n \in \mathbb{N}$. Ein **Gitter** im $\mathbb{R}^n$ ist eine additive Untergruppe des $\mathbb{R}^n$, die als Punktmenge diskret ist, d.h. jede Kugel im $\mathbb{R}^n$ enthält nur endlich viele Punkte des Gitters.

5.1. Satz *Sei $L \subseteq \mathbb{R}^n$ ein Gitter. Dann ist L von der Form $L = \mathbb{Z} \cdot \underline{b}_1 \oplus \ldots \oplus \mathbb{Z} \cdot \underline{b}_k$, wobei $\underline{b}_1, \ldots, \underline{b}_k$ über $\mathbb{R}$ linear unabhängig sind.*

Das Tupel $(\underline{b}_1, \ldots, \underline{b}_k)$ in Satz 5.1 heißt **Gitterbasis**. Es ist bis auf unimodulare Transformation von rechts eindeutig bestimmt.

Eine k**-dimensionale quadratische Form** ist ein Polynom

$$q(x_1, \ldots, x_k) \;=\; \sum_{i,j=1}^{k} q_{ij} \cdot x_i \cdot x_j,$$

welches homogen vom Grad 2 ist und für welches $q_{ij} = q_{ji}$ für alle $1 \leq i, j \leq k$ gilt. Wir betrachten solche Formen mit Koeffizienten in $\mathbb{R}$. Die Form heißt **positiv definit**, wenn $q(\underline{y}) \geq 0$ für alle $\underline{y} \in \mathbb{Z}^k$ und $q(\underline{y}) = 0$ genau dann, wenn $\underline{y} = \underline{0}$. Alle quadratischen Formen, die wir im folgenden benutzen, seien positiv definit.

Einem Gitter $L = \mathbb{Z} \cdot \underline{b}_1 \oplus \ldots \oplus \mathbb{Z} \cdot \underline{b}_k$ kann man die quadratische Form

$$q(\underline{x}) \;=\; \underline{x}^t \cdot B^t \cdot B \cdot \underline{x} \;=\; \underline{x}^t \cdot Q \cdot \underline{x}$$

zuordnen, wobei $\underline{x} = (x_1, \ldots, x_k) \in \mathbb{Z}^k$ und $B = (\underline{b}_1, \ldots, \underline{b}_k) \in \mathbb{R}^{n \times k}$ ist. Man beachte, daß dann

$$B^t \cdot B \;=\; \left(\langle \underline{b}_i, \underline{b}_j \rangle\right)_{1 \leq i,j \leq k}$$

mit dem Vektorprodukt $\langle\,.\,\rangle$ ist. Umgekehrt kann man einer quadratischen Form $q(\underline{x}) = \sum_{i,j=1}^{n} q_{ij}\cdot x_i\cdot x_j$ auch ein Gitter zuordnen: Durch quadratische Ergänzung ermittelt man Koeffizienten $(a_{ij})_{1\le i\le j\le k}$ derart, daß

$$q(\underline{x}) \;=\; \sum_{i=1}^{n} a_{ii}\cdot\left(x_i + \sum_{j=i+1}^{n} a_{ij}\cdot x_j\right)^2 \tag{5.1}$$

gilt. Schreiben wir dies in Matrixschreibweise, so erhalten wir

$$q(\underline{x}) = \left(\begin{pmatrix} 1 & a_{12} & \dots & a_{1k}\\ 0 & 1 & \dots & a_{2k}\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & 1\end{pmatrix}\cdot\underline{x}\right)^t\cdot\begin{pmatrix} a_{11} & 0 & \dots & 0\\ 0 & a_{22} & 0 & 0\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & a_{kk}\end{pmatrix}\cdot\begin{pmatrix} 1 & a_{12} & \dots & a_{1k}\\ 0 & 1 & \dots & a_{2k}\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & 1\end{pmatrix}\cdot\underline{x}.$$

Teilen wir also die Diagonalmatrix auf, so erkennen wir, daß $q(\underline{x}) = \underline{x}^t\cdot B^t\cdot B\cdot\underline{x}$ mit der Matrix

$$B \;=\; \begin{pmatrix} 1 & a_{12} & \dots & a_{1k}\\ 0 & 1 & \dots & a_{2k}\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & 1\end{pmatrix}\cdot\begin{pmatrix} a_{11}^{1/2} & 0 & \dots & 0\\ 0 & a_{22}^{1/2} & 0 & 0\\ \vdots & \ddots & \ddots & \vdots\\ 0 & \dots & 0 & a_{kk}^{1/2}\end{pmatrix}$$

ist. Damit ist B die Basis eines Gitters mit zugehöriger quadratischer Form $q(\underline{x})$.

5.2. Beispiel Wir wollen die quadratische Form

$$q(x_1,x_2,x_3) \;=\; 2\cdot x_1^2 + 2\cdot x_1\cdot x_2 + 2\cdot x_1\cdot x_3 + 2\cdot x_2^2 + 2\cdot x_2\cdot x_3 + 2\cdot x_3^2$$

in die gewünschte Form bringen. Dazu klammern wir zuerst den Koeffizienten von x_1^2, führen dann quadratische Ergänzung aus und korrigieren den gemachten Fehler. Somit erhalten wir

$$q(\underline{x}) \;=\; 2\cdot\left(x_1 + \frac{1}{2}\cdot x_2 + \frac{1}{2}\cdot x_3\right)^2 + \frac{3}{2}\cdot x_2^2 + x_2\cdot x_3 + \frac{3}{2}\cdot x_3^2.$$

Wiederholen wir dieses Verfahren, so bekommen wir schon die gewünschte Form als

$$q(\underline{x}) \;=\; 2\cdot\left(x_1 + \frac{1}{2}\cdot x_2 + \frac{1}{2}\cdot x_3\right)^2 + \frac{3}{2}\cdot\left(x_2 + \frac{1}{3}\cdot x_3\right)^2 + \frac{4}{3}\cdot x_3^2.$$

Wir wollen dies als Algorithmus formulieren (man beachte im Algorithmus, daß $q_{ij} = q_{ji}$ gilt):

5.3. Algorithmus

Eingabe: Koeffizienten $(q_{ij})_{1\le i,j\le k}$ einer k-dimensionalen quadratischen Form.

Ausgabe: Koeffizienten $(a_{ij})_{1\le i\le j\le k}$ gemäß (5.1).

vspace0.25cm FOR $i = 1, i \leq k, i = i + 1$

DO

$a_{ii} = q_{ii}$;

FOR $j = i + 1, j \leq k, j = j + 1$ DO $a_{ij} = \frac{q_{ij}}{a_{ii}}$; OD

FOR $j = i + 1, j \leq k, j = j + 1$

DO

$q_{jj} = q_{jj} - a_{ij}^2 \cdot a_{ii}$;

FOR $l = j + 1, l \leq k, l = l + 1$ DO $q_{jl} = q_{jl} - \frac{a_{ii} \cdot a_{ij} \cdot a_{il}}{2}$ OD

OD

OD

Return (x_{ij});

Seien L und L' zwei k-dimensionale Gitter. Dann heißen L und L' **isometrisch**, falls es einen längenerhaltenden $\mathbb{Z}$-Modul-Isomorphismus $\varphi : L \to L'$ gibt, d.h. es muß für alle $\underline{x} \in L$ gelten $|| \varphi(\underline{x}) || = || \underline{x} ||$. Hieraus folgt insbesondere, daß $\langle \varphi(\underline{x}), \varphi(\underline{y}) \rangle = \langle \underline{x}, \underline{y} \rangle$ für alle $\underline{x}, \underline{y} \in L$ gilt (zum Beweis beachte man die Gleichung $|| \underline{x} + \underline{y} ||^2 = || \underline{x} ||^2 + || \underline{y} ||^2 + 2 \cdot \langle \underline{x}, \underline{y} \rangle$). Isometrie ist eine Äquivalenzrelation. Die Äquivalenzklassen heissen Isometrieklassen.

Seien $q(\underline{x}) = \underline{x}^t \cdot Q \cdot \underline{x}$ und $q'(\underline{x}) = \underline{x}^t \cdot Q' \cdot \underline{x}$ zwei quadratische Formen. Dann heißen $q(\underline{x})$ und $q'(\underline{x})$ **äquivalent**, wenn sie durch unimodulare Transformation der Variablen ineinander übergehen, d.h. wenn es eine Matrix $T \in \mathrm{Gl}(k, \mathbb{Z})$ gibt, so daß

$$Q' = T^t \cdot Q \cdot T$$

ist. Dies ist ebenfalls eine Äquivalenzrelation.

5.4. Satz *Die Isometrieklassen der k-dimensionalen Gitter entsprechen eineindeutig den Äquivalenzklassen von positiv definiten k-dimensionalen quadratischen Formen.*

Beweis: Sei $\mathcal{L}$ eine Isometrieklasse, $L \in \mathcal{L}$ und $B = (\underline{b}_1, \ldots, \underline{b}_k)$ eine Basis von L. Dann ordnen wir $\mathcal{L}$ die Äquivalenzklasse der quadratischen Form $q(\underline{x}) = \underline{x}^t \cdot B^t \cdot B \cdot \underline{x}$ zu.

Wir müssen zuerst zeigen, daß diese Abbildung wohldefiniert ist. Sei also $L' \in \mathcal{L}$ und $B' = (\underline{b}'_1, \ldots, \underline{b}'_k)$ eine Basis von L'. Dann gibt es eine Isometrie $\varphi : L \to L'$ und $B^\varphi = (\varphi(\underline{b}_1), \ldots, \varphi(\underline{b}_k))$ ist eine Basis von L'. Also gibt es eine Matrix $T \in \mathrm{Gl}(k, \mathbb{Z})$, so daß $B' = B^\varphi \cdot T$ ist. Damit gilt:

$$\underline{x}^t \cdot B'^t \cdot B' \cdot \underline{x} = \underline{x}^t \cdot T^t \cdot (B^\varphi)^t \cdot B^\varphi \cdot T \cdot \underline{x} = \underline{x}^t \cdot T^t \cdot B^t \cdot B \cdot T \cdot \underline{x},$$

denn wegen der Längenerhaltung gilt

$$(B^\varphi)^t \cdot B^\varphi = \Big(\langle \varphi(\underline{b}_i), \varphi(\underline{b}_j) \rangle \Big)_{1 \leq i,j \leq k} = \Big(\langle \underline{b}_i, \underline{b}_j \rangle \Big)_{1 \leq i,j \leq k} = B^t \cdot B.$$

Damit sind die zugehörigen quadratischen Formen $\underline{x}^t \cdot B'^t \cdot B' \cdot \underline{x}$ und $\underline{x}^t \cdot B^t \cdot B \cdot \underline{x}$ äquivalent und die Abbildung ist wohldefiniert.

Die Surjektivität folgt direkt, wenn man mit dem oben beschriebenen Verfahren zu einem Vertreter aus einer Äquivalenzklasse äquivalenter quadratischer Formen ein zugehöriges Gitter bestimmt und die entsprechende Isometrieklasse betrachtet.

Nun zur Injektivität: seien $\underline{x}^t \cdot B^t \cdot B \cdot \underline{x} = \underline{x}^t \cdot U^t \cdot B'^t \cdot B' \cdot U \cdot \underline{x}$ mit $U \in \mathrm{Gl}(k, \mathbb{Z})$ zwei äquivalente quadratische Formen. Seien weiterhin $B = (\underline{b}_1, \ldots, \underline{b}_k)$ und $B' = (\underline{b}'_1, \ldots, \underline{b}'_k)$ Basen der zugehörigen Gitter L bzw. L'. Es ist zu zeigen, daß L und L' isometrisch sind. Definiere dazu die Abbildung φ als homomorphe Fortsetzung von

$$\begin{array}{rccll} \varphi: & L & \longrightarrow & L' & \\ & \underline{b}_i & \longmapsto & \underline{b}'_i \cdot U & \text{für alle } 1 \leq i \leq k. \end{array}$$

Sicherlich ist φ dann ein Isomorphismus, denn er bildet eine Basis auf eine andere ab. Weiterhin ist φ auch längenerhaltend:

$$\begin{aligned} \left|\left| \varphi\left(\sum_{i=1}^{k} x_i \cdot \underline{b}_i \right) \right|\right|^2 &= \left|\left| \sum_{i=1}^{k} x_i \cdot \underline{b}'_i \cdot U \right|\right|^2 \\ &= ||\, B' \cdot U \cdot \underline{x} \,||^2 \\ &= \underline{x}^t \cdot U^t \cdot B'^t \cdot B' \cdot U \cdot \underline{x} \\ &= \underline{x}^t \cdot B^t \cdot B \cdot \underline{x} \\ &= \left|\left| \sum_{i=1}^{k} x_i \cdot \underline{b}_i \right|\right|^2 \end{aligned}$$

■

Wir sind an den folgenden Invarianten für die Isometrieklassen von Gittern bzw. für die Äquivalenzklassen quadratischer Formen interessiert. Sei dabei immer

$$L = \mathbb{Z} \cdot \underline{b}_1 \oplus \ldots \oplus \mathbb{Z} \cdot \underline{b}_k \quad \text{ein Gitter und} \quad q(\underline{x}) = \underline{x}^t \cdot Q \cdot \underline{x}$$

die zugehörige positiv definite quadratische Form.

1. **Formdiskriminante:** Sie ist definiert als $\operatorname{disc} q(\underline{x}) = \det Q$.

2. **Gitterdeterminante:** $\det L = (\det(\langle \underline{b}_i, \underline{b}_j \rangle)_{1 \leq i,j \leq k})^{1/2} = \det(B^t \cdot B)^{1/2} = \det B$. Geometrisch kann man die Gitterdeterminante als das Volumen des von $\underline{b}_1, \ldots, \underline{b}_k$ aufgespannten k-dimensionalen Parallelotops verstehen. Man beachte, daß damit $\operatorname{disc} q(\underline{x}) = (\det L)^2$ gilt.

3. **Sukzessive Minima:** Das j-te sukzessive Minimum ist definiert als $\lambda_j(L) = \min\{r :$ Es gibt eine Kugel vom Radius r, die j linear unabhängige Vektoren des Gitters enthält $\}$. Insbesondere ist $\lambda_1(L)$ die Länge des kürzesten von Null verschiedenen Vektors in L.

Zwischen diesen Größen gibt es die folgenden Beziehungen:

5.5. Satz

(i) $\det L \;\le\; \lambda_1(L)\cdot\ldots\cdot\lambda_k(L) \;\le\; c_k^k \cdot \det L$ *für eine Konstante* c_k.

(ii) $\lambda_1(L) \;\le\; \gamma_k \cdot (\det L)^{1/k}$

Beweis: (Idee) Die erste Abschätzung erhält man aus der Anwendung der Hadamarschen Ungleichung, die zweite Ungleichung ist eine Folge des 2. Gitterpunktsatzes von Minkowski. Teil (ii) ist eine direkte Folge von (i). ■

5.2 Kurze Gittervektoren

Um die Berechnung kurzer Gittervektoren zu motivieren, betrachte man die folgenden Anwendungen:

- **Dichteste Kugelpackungen:**

  Wir wollen den $\mathbb{R}^k$ mit Kugeln überdecken und zwar so, daß das Innere zweier verschiedener Kugeln disjunkt ist und daß die Kugelmittelpunkte ein k-dimensionales Gitter bilden. Außerdem soll damit möglichst viel Raum überdeckt werden.

  Sei L das Gitter der Kugelmittelpunkte. Offensichtlich ist $r = \lambda_1(L)/2$ der optimale Kugelradius, den wir wählen können. Denn zwei Kreise von solchem Radius können sich nicht schneiden, weil sonst zwei Gitterpunkte existieren, deren Abstand kleiner wäre als $\lambda_1(L)$. Wählt man andererseits den Nullpunkt und einen kürzesten Gittervektor als Mittelpunkte, so berühren sich die entsprechenden Kreise gerade.

  Damit ist für die Güte der Überdeckung folgendes Verhältnis wichtig:

  $$\frac{\text{Kugelvolumen}}{\text{Gitterdeterminante}} \sim \frac{(\lambda_1(L))^k}{\det L} = \gamma_k^k.$$

  Die Tatsache, daß dies wirklich ein Maß für die Güte der Überdeckung ist, wird klar, wenn man sich die Situation im zwei-dimensionalen Raum ansieht. Betrachte das Parallelogramm, das von den zwei Basisvektoren aufgespannt wird. Addiert man alle Kreisausschnitte, die innerhalb des Parallelogramms liegen, so erhält man genau die Fläche eines ganzen Kreises. Dies läßt sich auch auf den mehrdimensionalen Raum übertragen.

  Findet man also das maximal mögliche γ_k und ein zugehöriges Gitter, so ist das k-dimensionale Problem dichtester Kugelpackungen gelöst.

- **Algebraische Abhängigkeit:**

  Es seien zwei algebraische Zahlen $\alpha, \beta \in \mathbb{C}$ als Nullstellen ihrer Minimalpolynome gegeben. Gesucht ist das Minimalpolynom von $\alpha + \beta$. Wir können natürlich $\alpha + \beta$ beliebig genau approximieren, wollen nun aber den exakten Wert bestimmen.

  Wenn das Minimalpolynom von α vom Grad n und das von β vom Grad m ist, dann kann das Minimalpolynom von $\alpha + \beta$ maximal den Grad $k \le n \cdot m$ besitzen. Dies wird offensichtlich aus

5.6. Übung Seien $A(x), B(x)$ die Minimalpolynome von α bzw. β. Zeige, daß dann der primitive Anteil der Resultante von $A(x-y)$ und $B(y)$ (als Polynome der Variablen y) ein Vielfaches des Minimalpolynoms von $\alpha+\beta$ ist.

Setze $\gamma = \alpha + \beta$ und betrachte das Gitter L, das durch die Spalten der Matrix

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \\ g_0 & g_1 & \dots & g_k \end{pmatrix}$$

erzeugt wird. Dabei sei $g_i = \lfloor 2^q \cdot \gamma^i \rceil$ mit einer approximativen Lösung γ, $\lfloor\ \rceil$ die nächste ganze Zahl und q damit die Anzahl der berücksichtigten Bits der approximativen Lösung für γ. Dann entspricht einer Polynomgleichung $\sum_{i=0}^k a_i \cdot \gamma^i = 0$ mit Koeffizienten $a_i \in \mathbb{Z}$ (also insbesondere auch einer solchen Gleichung mit dem Minimalpolynom) der Gittervektor

$$\underline{v} \;=\; \left(a_0, \dots, a_k, \sum_{i=0}^k a_i \cdot g_i\right).$$

Hierin gilt

$$\sum_{i=0}^k a_i \cdot g_i \;=\; \sum_{i=0}^k a_i \cdot (g_i - 2^q \cdot \gamma_i) \;\le\; \frac{1}{2} \cdot \sum_{i=0}^k |a_i|.$$

Also kann man die Länge dieses Vektors $\underline{v}$ durch $||\,\underline{v}\,||^2 \le 2 \cdot (k+1)^2 \cdot \max\{a_i : 1 \le i \le k\}^2$ abschätzen. Diese obere Abschätzung ist unabhängig von q. Andererseits läßt sich zeigen, daß man für eine Gleichung $\sum_{i=0}^k b_i \cdot \gamma^i \neq 0$ mit $b_i \in \mathbb{Z}$ die Länge von $(b_0, \dots, b_k, \sum_{i=0}^k b_i \cdot g_i)$ nach unten in Abhängigkeit von q abschätzen kann:

$$\begin{aligned}
\left|\left| \left(b_0, \dots, b_k, \sum_{i=0}^k b_i \cdot g_i\right) \right|\right|^2 \;&=\; \sum_{i=0}^k b_i^2 + \left(2^q \cdot \sum_{i=0}^k b_i \cdot \gamma^i + \sum_{i=0}^k b_i \cdot \left(g_i - 2^q \cdot \gamma^i\right)\right)^2 \\
&\ge\; \sum_{i=0}^k b_i^2 + 2^{q+1} \cdot \underbrace{\left(\sum_{i=0}^k b_i \cdot \gamma^i\right)^2}_{\ge c_{\underline{b}}} - \frac{1}{4} \cdot \left(\sum_{i=0}^k b_i\right)^2 \\
&\ge\; \sum_{i=0}^k \frac{b_i^2}{2} + 2^{q+1} \cdot c_{\underline{b}} \\
&\ge\; \frac{1}{2}\,||\,\underline{b}\,||^2 + 2^{q+1} \cdot c_{\underline{b}}
\end{aligned}$$

mit einer Konstante $c_{\underline{b}}$. Macht man nun q hinreichend groß, so entsprechen Polynomgleichungen, die von γ erfüllt werden, kurzen Vektoren; alle anderen aber längeren Vektoren. Kann man also sehr kurze Gittervektoren berechnen, so läßt sich daraus ein guter Kandidat für die algebraische Gleichung bestimmen.

Damit ist das Problem der Berechnung kurzer und kürzester Vektoren motiviert. Jetzt diskutieren wir zuerst die Berechnung kürzester Vektoren. Dies ist algorithmisch sehr schwierig und es sind nur Exponentialzeitalgorithmen bekannt. Wir stellen den Algorithmus von Fincke und Pohst vor:

Sei $L = \mathbb{Z} \cdot \underline{b}_1 \oplus \ldots \oplus \mathbb{Z} \cdot \underline{b}_k$ ein Gitter. Berechne die zugehörige quadratische Form

$$q(\underline{x}) \;=\; \sum_{i=1}^{k} a_{ii} \cdot \left(x_i + \sum_{j=i+1}^{k} a_{ij} \cdot x_j \right)^2 .$$

Um den kürzesten Vektor des Gitters zu berechnen, wählen wir eine obere Schranke C für das erste sukzessive Minimum, z.B. $C = \frac{4}{3} \cdot (\det L)^{1/k}$. Gesucht sind dann Vektoren $\underline{x}$ mit $q(\underline{x}) \leq C^2$. Ein solcher Vektor $\underline{x}$ führt dann zu einem gewünschten Gitterelement, denn $q(\underline{x}) = \underline{x}^t \cdot B^t \cdot B \cdot \underline{x} = ||\, B \cdot \underline{x}\, ||^2 \leq C^2$ und damit ist $B \cdot \underline{x}$ ein Gittervektor mit Länge $\leq C$.

Die Idee ist folgende: Angenommen, wir haben schon die Komponenten $x_{l+1}, \ldots, x_k$ gewählt. Dann muß x_l der Bedingung

$$a_{ll} \cdot \left(x_l + \sum_{j=l+1}^{k} a_{lj} \cdot x_j \right)^2 \;\leq\; C_l$$

genügen, wobei

$$C_l \;=\; C - \sum_{i=l+1}^{k} a_{ii} \cdot \left(x_i + \sum_{j=i+1}^{k} a_{ij} \cdot x_j \right)^2$$

ist. Ist $C_l < 0$, so gibt es keine Lösung $\underline{x}$ mit dem Suffix $x_{l+1}, \ldots, x_k$, denn wegen der positiven Definitheit muß $a_{ll} > 0$ sein. Andernfalls müssen wir alle x_l ausprobieren, die der Bedingung

$$-\left(\frac{C_l}{a_{ll}}\right)^{1/2} - \sum_{j=l+1}^{k} a_{lj} \cdot x_j \;\leq\; x_l \;\leq\; \left(\frac{C_l}{a_{ll}}\right)^{1/2} - \sum_{j=l+1}^{k} a_{lj} \cdot x_j$$

genügen. Hieraus ergibt sich der folgende Algorithmus:

5.7. Algorithmus (Vektoren kurzer Länge)

Eingabe: Koeffizienten a_{ij} einer quadratischen Form $q(\underline{x})$ gemäß (5.1) und Schranke $C > 0$.

Ausgabe: Alle $\underline{x}$ mit $q(\underline{x}) \leq C$.

Setze $l = k,\ C_k = C$;

Berechne $x_k = \left\lceil -\left(\frac{C}{a_{kk}}\right)^{1/2} \right\rceil - 1$;

Berechne $UB_k = \left\lfloor \left(\frac{C}{a_{kk}}\right)^{1/2} \right\rfloor$;

WHILE $l \leq k$

  DO

  $x_l = x_l + 1$;

  IF $x_l > UB_l$ THEN $l = l + 1$ und gehe zu Schritt 4. FI

  IF $l = 1$ THEN Ausgabe $(x_1, \ldots, x_k)$ und gehe zurück zu Schritt 4. FI

$C_{l-1} = C_l - a_{ll} \cdot \left(x_l + \sum_{j=l+1}^{k} a_{lj} \cdot x_j\right)^2$;

IF $C_{l-1} \geq 0$ THEN

$x_{l-1} = \left\lceil -\left(\frac{C_{l-1}}{a_{l-1,l-1}}\right)^{1/2} - \sum_{j=l}^{k} a_{l-1,j} \cdot x_j \right\rceil - 1;$

$UB_{l-1} = \left\lfloor \left(\frac{C_{l-1}}{a_{l-1,l-1}}\right)^{1/2} - \sum_{j=l}^{k} a_{l-1,j} \cdot x_j \right\rfloor;$

$l = l - 1;$

FI

OD

Diesen Algorithmus kann man so modifizieren, daß er alle sukzessiven Minima berechnet. Das Problem des Algorithmus ist natürlich seine exponentielle Komplexität.

Daher stellen wir nun einen polynomiellen Algorithmus vor, der eine approximative Lösung liefert. In der Praxis sind die Ergebnisse dieses Algorithmus häufig sehr nah an dem Optimum, aber trotzdem können sie theoretisch exponentiell weit davon entfernt sein. Bevor wir diesen Algorithmus kennenlernen, machen wir uns das Prinzip am zweidimensionalen Fall deutlich.

Sei also L ein zweidimensionales Gitter. Angenommen, wir starten mit einer Basis $\{\underline{b}_1, \underline{b}_2\}$. Wir wollen durch unimodulare Transformation eine möglichst kurze Basis finden. Dazu erlauben wir die folgenden zwei Operationen:

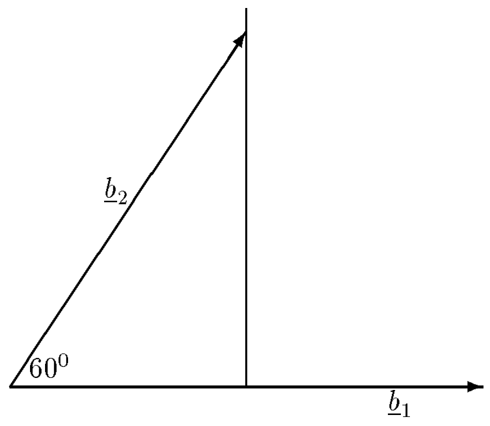
a) Addition eines Vielfachen des ersten Basisvektors zum zweiten Basisvektor: $\underline{b}_2 = \underline{b}_2 + k \cdot \underline{b}_1$ für $k \in \mathbb{Z}$.

b) Vertauschung der Basiselemente: $(\underline{b}_1, \underline{b}_2) = (\underline{b}_2, \underline{b}_1)$.

Durch Operation a) verkürzt man nur die Projektion von $\underline{b}_2$ auf $\underline{b}_1$. Setze nun $\underline{b}_1^* = \underline{b}_1$ und $\underline{b}_2^*$ auf die Projektion von $\underline{b}_2$ auf das orthogonale Komplement von $\underline{b}_1$. Dann erhält man die Gleichung

$$\underline{b}_2 = \underline{b}_2^* + \mu_{21} \cdot \underline{b}_1^*,$$

wobei μ_{21} gerade der Anteil der Länge der Projektion von $\underline{b}_2$ auf $\underline{b}_1$ an der Länge von $\underline{b}_1^*$ ist. Wir werden später angeben, wie man μ_{21} wirklich berechnen kann. Da $\underline{b}_1^* = \underline{b}_1$ ist, kann man durch die Wahl von $k = \lfloor \mu_{21} \rceil$ und Operation a) erreichen, daß es Vektoren $\underline{b}_1^*, \underline{b}_2^*$ und $\underline{b}_2$ mit $|\mu_{21}| \leq \frac{1}{2}$ gibt. Das ist in der Tat das beste, was man durch Anwendung der Regel a) erreichen kann. Betrachte nämlich die folgende geometrische Situation:



In diesem Fall ist $\mu_{21} = \frac{1}{2}$. Eine weitere Anwendung der Regel a) verkleinert den Wert von μ_{21} betragsmäßig nicht mehr.

Durch eine Reduktion mit Regel a) und $k = \lfloor \mu_{21} \rceil$ befindet sich $\underline{b}_2$ also links von der Mittelsenkrechten von $\underline{b}_1$ und rechts von der Mittelsenkrechten von $-\underline{b}_1$. Anschliessend kann man die beiden Basisvektoren nach Regel b) vertauschen. Wie lange ist dies sinnvoll? Gilt vor der Vertauschung $||\,\underline{b}_2\,|| > ||\,\underline{b}_1\,||$, so hat eine Reduktion nach der Vertauschung keine Wirkung mehr, da immer längere Vektoren mit Hilfe von kürzeren Vektoren reduziert werden. In diesem Fall kann man also das Verfahren schon vor dem Vertauschen abbrechen. Ansonsten führen wir eine neue Reduktion durch und iterieren diese Vorgehensweise. Also erhält man folgenden einfachen Reduktionsalgorithmus:

5.8. Algorithmus (Reduktion zweier Vektoren)

Eingabe: Gitterbasis $\underline{b}_1, \underline{b}_2$ mit $||\,\underline{b}_2\,|| \leq ||\,\underline{b}_1\,||$.

Ausgabe: reduzierte Basis $\underline{b}_1, \underline{b}_2$.

WHILE $||\,\underline{b}_2\,|| \leq ||\,\underline{b}_1\,||$

DO

Reduziere $\underline{b}_2$ mit $\underline{b}_1$, so daß $|\mu_{21}| \leq \frac{1}{2}$;

Vertausche $\underline{b}_1$ und $\underline{b}_2$;

OD

Dies wird im LLL-Algorithmus auf beliebige Dimensionen verallgemeinert. Sei $\underline{b}_1, \ldots, \underline{b}_k$ eine Basis eines Gitters L. Zuerst berechnen wir die Gram-Schmidt Orthogonalisierung $\underline{b}_1^*, \ldots, \underline{b}_k^*$ dieser Basis. Setze dazu

$$\underline{b}_1^* = \underline{b}_1 \quad \text{und} \quad \underline{b}_i^* = \underline{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \cdot \underline{b}_j^*, \text{ wobei } \mu_{ij} = \frac{\langle \underline{b}_i, \underline{b}_j^* \rangle}{\langle \underline{b}_j^*, \underline{b}_j^* \rangle} \tag{5.2}$$

ist. Wir verlangen weiterhin, daß $|\mu_{ij}| \leq \frac{1}{2}$ für $1 \leq j < i \leq k$ gilt. Dies führt zu kürzeren Vektoren, denn da die $\underline{b}_i^*$ orthogonal zueinander sind, gilt

$$|| \, \underline{b}_j \, ||^2 \; = \; || \, \underline{b}_j^* \, ||^2 + \sum_{i=1}^{j-1} (\mu_{ij})^2 \cdot || \, \underline{b}_i^* \, ||^2 \tag{5.3}$$

und damit führen "kleine" μ-Werte zu "kurzen" Vektoren. Diese Bedingung an die μ-Werte erreicht man, indem man ein geeignetes Vielfaches von $\underline{b}_j^*$ von $\underline{b}_i$ abzieht:

5.9. Algorithmus (REDUCE)

Eingabe: Basis $\underline{b}_1, \ldots \underline{b}_k$ und entsprechende Darstellung (5.2).

Ausgabe: modifizierte Basis $\underline{b}_1, \ldots \underline{b}_k$, so daß $|\mu_{ij}| \leq \frac{1}{2}$ in Darstellung (5.2).

FOR $j = 2, j \leq k, j = j + 1$

    DO

    FOR $i = j - 1, i \geq 1, i = i - 1$

        DO

        $\underline{b}_j \; = \; \underline{b}_j - \lfloor \mu_{ji} \rceil \cdot \underline{b}_i;$

        FOR $k = 1, k \leq i - 1, k = k + 1$ DO $\mu_{jk} = \mu_{jk} \cdot \lfloor \mu_{ji} \rceil \cdot \mu_{ik}$ OD

        OD

    OD

Anschliessend imitieren wir das zweidimensionale Reduktionsverfahren für die Basisvektoren $\underline{b}_i$ und $\underline{b}_{i-1}$. Für diese Basisvektoren kennen wir nun Darstellungen

$$\begin{aligned} \underline{b}_i \quad &= \quad \underline{b}_i^* + \mu_{i,i-1} \cdot \underline{b}_{i-1}^* + \mu_{i,i-2} \cdot \underline{b}_{i-2}^* + \ldots + \mu_{i,1} \cdot \underline{b}_1^* \\ \underline{b}_{i-1} \quad &= \quad \underline{b}_{i-1}^* + \mu_{i-1,i-2} \cdot \underline{b}_{i-2}^* + \ldots + \mu_{i-1,1} \cdot \underline{b}_1^* \end{aligned}$$

mit $|\mu_{i,i-1}| \leq \frac{1}{2}$. Wir projizieren die Basisvektoren $\underline{b}_{i-1}$ und $\underline{b}_i$ auf das orthogonale Komplement des von $\{\underline{b}_{i-2}, \ldots, \underline{b}_1\}$ erzeugten Unterraums und vergleichen ihre Längen. Diese Projektionen erhalten wir als $\underline{b}_{i-1}^*$ und $\underline{b}_i^* + \mu_{i,i-1} \cdot \underline{b}_{i-1}^*$. Falls

$$|| \, \underline{b}_i^* + \mu_{i,i-1} \cdot \underline{b}_{i-1}^* \, ||^2 \; < \; \frac{3}{4} \cdot || \, \underline{b}_{i-1}^* \, ||^2$$

ist, sollte man $\underline{b}_{i-1}$ und $\underline{b}_i$ vertauschen, denn dann bringt eine anschliessende Reduktion eine weitere Verbesserung. Der Faktor $\frac{3}{4}$ kann dabei ersetzt werden durch 1, aber dann funktioniert der Algorithmus nicht mehr in Polynomzeit. So erhalten wir folgenden Algorithmus:

5.10. Algorithmus (LLL)

Eingabe: Gitterbasis $\underline{b}_1, \ldots, \underline{b}_k$.

Ausgabe: LLL-reduzierte Basis $\underline{b}_1, \ldots, \underline{b}_k$.

Berechne Gram-Schmidt-Orthogonalbasis $\underline{b}_1^*, \ldots, \underline{b}_k^*$ und die Darstellung (5.2).

Reduziere die Zahlen μ_{ij} mit Algorithmus 5.9.

WHILE es gibt $i \geq 2$ mit $||\, \underline{b}_i^* + \mu_{i,i-1} \cdot b_{i-1}^* \,||^2 < \frac{3}{4} \cdot ||\, \underline{b}_{i-1}^* \,||^2$

DO

Vertausche $\underline{b}_i$ und $\underline{b}_{i-1}$.

Berechne neues $\underline{b}_i^*, \underline{b}_{i-1}^*$, ändere die μ's entsprechend.

Reduziere μ_{ij} mit Algorithmus 5.9.

OD

Man sollte beachten, daß man in der WHILE-Schleife die neuen Vektoren $\underline{b}_{i-1}^*$ und $\underline{b}_i^*$ sowie die neuen Zahlen μ_{ij} aus den alten Werten berechnen kann und nicht einen neuen Gram-Schmitt Orthogonalisierungsschritt machen muß. Die Formeln für diese Änderungen lassen wir hier weg, man findet sie in dem Originalartikel.

Eine Basis, die man durch diesen Algorithmus erhält, heißt LLL-reduziert. Die Laufzeit dieses Algorithmus ist polynomiell, wie man aus folgendem Satz ersieht:

5.11. Satz *Algorithmus 5.10 terminiert nach $O(k)$ Iterationen. Für die ausgegebene LLL-reduzierte Basis gilt*

$$|\mu_{ij}| \leq \frac{1}{2} \qquad \text{für} \quad 1 \leq i \leq k,\ 1 \leq j < i$$

und

$$\left|\left|\, \underline{b}_j^* + \mu_{j,j-1} \cdot \underline{b}_{j-1}^* \,\right|\right|^2 \;\geq\; \frac{3}{4} \cdot \left|\left|\, \underline{b}_{j-1}^* \,\right|\right|^2 \qquad \text{für} \quad 1 < j \leq k.$$

Beweis: Die Basiseigenschaften sind trivial. Zur Laufzeitaussage geben wir eine Beweisskizze:

Sei $D = \prod_{l=1}^{k} ||\, \underline{b}_l^* \,||^{k+1-l}$. Bei jedem Vertauschungsschritt wird dann D um den Faktor $\left(\frac{3}{4}\right)^{1/2}$ kleiner, denn

$$\left|\left|\, \underline{b}_j^* \,\right|\right|^2 \;<\; \left(\frac{3}{4} - \mu_{j,j-1}^2\right) \cdot \left|\left|\, \underline{b}_{j-1}^* \,\right|\right|^2 \;\leq\; \frac{3}{4} \cdot \left|\left|\, \underline{b}_{j-1}^* \,\right|\right|^2$$

gilt, bevor $\underline{b}_j$ und $\underline{b}_{j-1}$ vertauscht wurden. Nach Konstruktion von D erkennt man die Teilbehauptung. Die Laufzeit folgt nun, indem man eine obere Schranke für D beweist. ■

Wir listen abschliessend noch einige Eigenschaften von LLL-reduzierten Basen auf:

5.12. Lemma *Für jeden von $\underline{0}$ verschiedenen Vektor $\underline{b}$ in L gilt:*

$$||\, \underline{b} \,|| \;\geq\; \min\{||\, \underline{b}_j^* \,|| \;:\; 1 \leq j \leq n\}.$$

Beweis: Betrachte Gleichung (5.3). ■

5.13. Satz *Sei $\underline{b}_1, \ldots, \underline{b}_n$ LLL-reduziert. Dann gilt:*

a) $||\,\underline{b}_1\,|| \;\leq\; 2^{(k-1)/4} \cdot (\det L)^{1/k}.$

b) $||\,\underline{b}_j\,|| \;\leq\; 2^{(k-1)/2} \cdot \lambda_j(L)$ *für* $1 \leq j \leq k$.

c) $||\,\underline{b}_1\,|| \cdot \ldots \cdot ||\,\underline{b}_k\,|| \;\leq\; 2^{k \cdot (k-1)/2} \cdot \det L.$

Dabei zeigt die Ungleichung b), daß man möglicherweise exponentiell weit vom gesuchten Ergebnis entfernt ist.

Beweis: Wir zeigen hier nur den Fall b) für $j = 1$. Nach Lemma 5.12 gilt

$$\lambda_1(L) \;=\; \min\{||\,\underline{b}\,|| \,:\, \underline{b} \in L - \{\underline{0}\}\} \;\geq\; \min\{||\,\underline{b}_j^*\,|| \,:\, 1 \leq j \leq k\}.$$

Sei also $\underline{b} = \sum_{i=1}^{k} x_i \cdot \underline{b}_i$ ein Vektor in $L - \{\underline{0}\}$, d.h. $x_i \in \mathbb{Z}$ für $1 \leq i \leq k$. Dann gilt nach (5.2), daß es eine Darstellung

$$\underline{b} \;=\; \sum_{i=1}^{l} x_i^* \cdot \underline{b}_i^*$$

gibt. Dabei sei l der größte Index mit $x_l \neq 0$. Insbesondere gilt dann $x_l^* = x_l$. Daher erhalten wir

$$||\,\underline{b}\,||^2 \;=\; \sum_{i=1}^{l} (x_i^*)^2 \cdot ||\,\underline{b}_i^*\,||^2 \;\geq\; ||\,\underline{b}_l^*\,||^2.$$

Da die Basis $\underline{b}_1, \ldots, \underline{b}_k$ LLL-reduziert ist, gilt nun aber auch

$$||\,\underline{b}_j^*\,||^2 \;\geq\; \left(\frac{3}{4} - \mu_{j,j-1}^2\right) \cdot ||\,\underline{b}_{j-1}^*\,||^2 \;\geq\; \frac{1}{2} \cdot ||\,\underline{b}_{j-1}^*\,||^2.$$

Hieraus folgt

$$||\,\underline{b}_1\,||^2 \;=\; ||\,\underline{b}_1^*\,||^2 \;\leq\; 2 \cdot ||\,\underline{b}_2^*\,||^2 \;\leq\; 2^{k-1} \cdot \min\{||\,\underline{b}_j^*\,||^2 \,:\, 1 \leq j \leq k\} \;\leq\; 2^{k-1} \cdot \lambda_1(L).$$

■

Kapitel 6

Algorithmen für endliche abelsche Gruppen

6.1 Der Babystep-Giantstep Algorithmus

Es sei G eine endliche abelsche Gruppe. Wir nehmen an, daß wir in G multiplizieren und Elemente invertieren können. Außerdem können wir Elemente vergleichen. Wir besprechen nun als erstes das Problem, die Ordnung eines beliebigen Gruppenelementes $g \in G$ zu bestimmen. Der primitivste Algorithmus zur Bestimmung der Ordnung des Elementes g ist natürlich der folgende:

6.1. Algorithmus (triviale Bestimmung der Elementordnung)

Eingabe: $g \in G$.

Ausgabe: $\operatorname{ord} g$.

Setze $\operatorname{ord} g = 1$;
WHILE $g^{\operatorname{ord} g} \neq 1$ DO $\operatorname{ord} g = \operatorname{ord} g + 1$ OD
Return $\operatorname{ord} g$;

Dieser Algorithmus ist offensichtlich ein sehr schlechter Algorithmus, denn er probiert nur systematisch alle Möglichkeiten für die Elementordnung aus. Etwas cleverer ist der Algorithmus von Shanks. Er beruht auf folgender einfachen Idee:

6.2. Lemma *Sei $k \leq K$. Dann gibt es eine Zerlegung von k der Form*

$$k = q \cdot \left\lceil \sqrt{K} \right\rceil + r$$

mit $0 \leq q,\, r \leq \left\lfloor \sqrt{K} \right\rfloor$.

Beweis: Wir müssen nur die obere Schranke für q beweisen. Führen wir eine $\left\lceil\sqrt{K}\right\rceil$-adische Entwicklung von k durch, so erhalten wir eine Zerlegung

$$k \;=\; q \cdot \left\lceil\sqrt{K}\right\rceil + r$$

mit $0 \le r < \left\lceil\sqrt{K}\right\rceil$. Da wir die obere Abschätzung $k \le K$ kennen, erhalten wir damit $q \le \frac{K}{\lceil\sqrt{K}\rceil} \le \lfloor\sqrt{K}\rfloor$. ∎

Man kann somit also folgendermaßen vorgehen: Wir raten eine vermeintliche obere Schranke K für die Elementordnung $\operatorname{ord} g$. Dann prüfen wir, ob Lemma 6.2 für $k = \operatorname{ord} g$ mit Zahlen q, r und $0 \le q, r \le \lfloor\sqrt{K}\rfloor$ erfüllt ist. Dazu beachte man, daß dann

$$1 \;=\; g^{\operatorname{ord} g} \;=\; g^{q\cdot\lceil\sqrt{K}\rceil + r}$$

gilt. Durch eine einfache Umformung erhalten wir hieraus

$$g^{-r} \;=\; \left(g^{\lceil\sqrt{K}\rceil}\right)^{q}.$$

Wir berechnen also die Werte g^{-r} für alle $0 \le r \le \left\lfloor\sqrt{K}\right\rfloor$ vor und speichern sie in einer Tabelle ab. Anschliessend berechnen wir die Werte $\left(g^{\lceil\sqrt{K}\rceil}\right)^{q}$ für alle $0 \le q \le \left\lfloor\sqrt{K}\right\rfloor$ und vergleichen jeweils, ob dieses Element schon in der Tabelle vorkommt. Falls ja, so ist $q \cdot \lceil\sqrt{K}\rceil + r$ ein Vielfaches der Elementordnung. Der kleinste solche Wert ist dann die Elementordnung. Andernfalls verdoppeln wir K und führen das gleiche Verfahren erneut durch. Dies liefert folgenden Algorithmus:

6.3. Algorithmus (Shanks' Babystep-Giantstep Algorithmus)

Eingabe: $g \in G$.

Ausgabe: $\operatorname{ord} g$.

Setze $K = 2$, $\operatorname{ord} g = \infty$;

WHILE ($\operatorname{ord} g = \infty$)

  DO

  $K = 2 \cdot K$;

  $h = g^{\lceil\sqrt{K}\rceil}$;

  Speichere für $0 \le q \le \lfloor\sqrt{K}\rfloor$ das Tupel (h^q, q) in einer Tabelle;

  FOR ($r = 0$, $r \le \lfloor\sqrt{K}\rfloor$, $r = r + 1$)

    DO

    IF $g^{-r} = h^q$ für ein Tabellenelement (h^q, q)

      THEN $\operatorname{ord} g = \min\{\operatorname{ord} g,\, q \cdot \lceil\sqrt{K}\rceil + r\}$ FI

    OD

  OD

Wir analysieren die Laufzeit von Algorithmus 6.3 und machen dazu folgende Zusatzannahme: Die Elemente von G sind durch ganze Zahlen z mit $\log(|z|+1) = O(\log|G|)$ eindeutig darstellbar. Dann kann man die Potenzen $(g^{\lceil K \rceil})^q$ sortiert in einer Tabelle ablegen. Dies geht in Zeit $O(\log|G|)$ pro Element (Beweis als Übung). Jeder Vergleich in der FOR-Schleife kostet dann mit binärer Suche ebenfalls Zeit $O(\log|G|)$. Insgesamt müssen wir zur Berechnung der Tabelle und in der anschliessenden FOR-Schleife jeweils $O(\sqrt{\operatorname{ord} g})$ viele Multiplikationen durchführen, denn wir benötigen zur Berechnung eines neuen Tabellenelementes nur eine Multiplikation mit dem alten Tabellenelementes. Außerdem ist die Anzahl der Iterationen in der WHILE-Schleife $O(\log \operatorname{ord} g)$, denn bei Abbruch gilt $K \leq 2 \cdot \operatorname{ord} g$. Bezeichnet man mit $T(G)$ die Zeit für eine Multiplikation oder Inversenbildung in G, so erhält man somit den folgenden Satz:

6.4. Satz *Algorithmus 6.3 benötigt zur Bestimmung der Ordnung des Elementes $g \in G$ Zeit*

$$O\left(\sqrt{\operatorname{ord} g} \cdot \log \operatorname{ord} g \cdot (T(G) + \log|G|)\right).$$

6.5. Übung Das **diskrete Logarithmusproblem** in G ist das folgende Problem: Gegeben zwei Elemente $g \in G$ und $h \in \langle g \rangle \subseteq G$. Man bestimme einen minimalen nichtnegativen Exponenten $x \in \mathbb{N}_0$ mit

$$g^x = h.$$

Verändere Algorithmus 6.3 so, daß man damit auch das diskrete Logarithmusproblem in einer endlichen abelschen Gruppe G lösen kann. Wie ist die Laufzeit dieses neuen Algorithmus?

Man beachte, daß Algorithmus 6.3 im allgemeinen exponentielle Zeit benötigt. Eine triviale obere Schranke für die Ordnung eines Elementes ist die Gruppenordnung, so daß damit die Laufzeit exponentiell in der Gruppenordnung ist. Für Elemente kleiner Ordnung ist der Algorithmus natürlich schnell fertig. Kennt man jedoch die Gruppenordnung $|G|$, so kann man die Ordnung eines beliebigen Elementes leichter feststellen:

6.6. Algorithmus (Elementordnung bei bekannter Gruppenordnung)

Eingabe: $|G|$ und $g \in G$.

Ausgabe: $\operatorname{ord} g$.

Faktorisiere $|G| = \prod_{i=1}^{k} p_i^{e_i}$;

FOR $(i = 1, i \leq k, i = i + 1)$

  DO

  Setze $P_i = \frac{|G|}{p_i^{e_i}}$;

  Berechne $h = g^{P_i}$;

  Berechne $f_i = \min\{f : h^{{p_i}^f} = 1\}$;

  OD

Return $\operatorname{ord} g = \prod_{i=1}^{k} p_i^{f_i}$;

Ist die Faktorisierung von $|G|$ bekannt, so kann man damit jede Elementordnung in Polynomzeit finden. Außerdem läßt sich $|G|$ in subexponentieller Zeit faktorisieren und damit kann man die Ordnung von g mit Algorithmus 6.6 auch in subexponentieller Zeit berechnen. Dies ist gegenüber Algorithmus 6.3 eine deutliche Verbesserung.

Ein deutlich schwieriges Problem ist es, die Ordnung der Gruppe G zu bestimmen. Wir nehmen dabei an, daß wir ein Erzeugendensystem für G kennen. Kennt man dies nicht, so kann man eine probabilistische Strategie anwenden und beliebige Elemente raten. Dann nimmt man an, daß die geratenen Elemente diese Eigenschaft erfüllen. Ist dies nicht der Fall, so muß man neue Elemente raten. Ansonsten muß man nach der Berechnung der Gruppenordnung natürlich noch verifizieren, daß die geratenen Elemente wirklich ein Erzeugendensystem für die ganze Gruppe bilden. Seien aber im folgenden immer Gruppenelemente $g_1, \ldots, g_k$ mit

$$G = \langle g_1, \ldots, g_k \rangle_{\mathbb{Z}}$$

bekannt. Betrachte dann die folgende Abbildung:

$$\begin{array}{rrcl} \varphi: & \mathbb{Z}^k & \longrightarrow & G \\ & (v_1, \ldots, v_k) & \longmapsto & \prod_{i=1}^{k} g_i^{v_i}. \end{array}$$

6.7. Satz *Der Kern der Abbildung φ ist ein k-dimensionales Gitter L im $\mathbb{R}^k$ und es gilt für die Gruppenordnung* $\det L = |G|$.

Beweis: $L = \operatorname{Kern} \varphi$ ist sicherlich ein Gitter, denn es ist eine Untergruppe des $\mathbb{Z}^k$. Weiterhin ist die Abbildung φ offensichtlich ein Homomorphismus. Da $\{g_1, \ldots, g_k\}$ ein Erzeugendensystem für G ist, ist φ auch surjektiv. Also gilt nach dem Isomorphiesatz $\mathbb{Z}^k/L \cong G$. Wir müssen damit nur noch zeigen, daß $|\mathbb{Z}^k/L| = \det L$ ist. Zuerst sieht man leicht, daß $\dim L = k$ ist. Wäre nämlich $\dim L < k$, so bräuchte man nämlich nur einen Vektor $\underline{v} \in \mathbb{Z}^k$ zu wählen, der nicht in dem von L aufgespannten Teilraum von $\mathbb{Z}^k$ liegt. Dieser existiert in diesem Fall und die Vielfachen von $\underline{v}$ sind dann paarweise inkongruent modulo L. Dann ist damit aber die Ordnung der Gruppe unendlich. Weil G aber endlich ist, muß also $\dim L = k$ sein.

Nun wählt man eine Basis $\underline{b}_1, \ldots, \underline{b}_k$ von L, die in "Dreiecksform" ist, d.h. es sei

$$\underline{b}_j = (b_{1j}, \ldots, b_{jj}, 0, \ldots, 0).$$

Wir fordern zusätzlich, daß $b_{jj} > 0$ und $0 \leq b_{ij} < b_{ii}$ für $1 \leq i < j$ gilt. Dann ist die Basis sogar eindeutig bestimmt und heißt **Hermite-Normalform-Basis** von L. Die Matrix mit den Spalten $\underline{b}_1, \ldots, \underline{b}_k$ ist dann ebenfalls in Hermite-Normalform. Wir werden im folgenden Abschnitt einen Algorithmus vorstellen, der aus einer gegebenen Basis die Hermite-Normalform-Basis bestimmt.

Kennen wir die Hermite-Normalform-Basis $\underline{b}_1, \ldots, \underline{b}_k$ von L, so können wir ein vollständiges, minimales Vertretersystem für $\mathbb{Z}^k/L$ hinschreiben:

$$V = \{\underline{v} = (v_1, \ldots, v_k) : 0 \leq v_i < b_{ii} \text{ für } 1 \leq i \leq k\}.$$

Offensichtlich ist dieses ein vollständiges Vertretersystem. Um die Minimalität zu zeigen, nehmen wir an, daß es zwei verschiedene Vektoren $\underline{v} = (v_1, \ldots, v_k) \in V$ und $\underline{v}' = (v_1', \ldots, v_k') \in V$ gibt, die modulo L gleich sind. Dann ist $\underline{v} - \underline{v}' \in L$ und es gibt einen maximalen Index l, so daß $v_l - v_l' \neq 0$ ist. Nun ist aber $0 < |v_l - v_l'| < b_{ll}$, was im Widerspruch dazu steht, daß $v_l - v_l'$ ein Vielfaches von b_{ll} ist. Bei der besonderen Gestalt unserer Basis muß dies jedoch der Fall sein.

Damit können wir zeigen, daß $\det L = |\mathbb{Z}^k/L|$ ist. Die Determinante des Gitters ist nach Definition die Determinante der Matrix, die die Basisvektoren als Spalten besitzt. Da diese in Dreiecksform ist, erhalten wir

$$\det L = \prod_{i=1}^{k} b_{ii}.$$

Andererseits gilt nach einem einfachen Abzählargument aber auch

$$|\mathbb{Z}^k/L| = |V| = \prod_{i=1}^{k} b_{ii},$$

womit der Satz bewiesen ist. ■

Die Gruppenordnung der Gruppe G wird nun bestimmt, indem wir die im vorigen Satz eingeführte Hermite-Normalform-Basis bestimmen. Hierzu beweisen wir

6.8. Lemma *Der Zeilenvektor $(b_{1j}, \ldots, b_{jj}, 0, \ldots, 0)$ in der Hermite-Normalform-Basis läßt sich wie folgt charakterisieren: b_{jj} ist der kleinste Exponent mit der Eigenschaft, daß $g_j^{b_{jj}}$ in der von $g_1, \ldots, g_{j-1}$ erzeugten Untergruppe liegt. Die b_{ij} sind die kleinsten Exponenten mit*

$$g_1^{b_{1j}} \cdot \ldots \cdot g_j^{b_{jj}} = 1.$$

Beweis: Klar. ■

Damit kommen wir zum Algorithmus: Angenommen, $\underline{b}_1, \ldots, \underline{b}_{j-1}$ sind schon bestimmt. Wir wollen ein Verfahren vorstellen, um dann $\underline{b}_j$ zu finden. Wir wissen, daß $0 \leq b_{ij} < b_{ii}$ für $1 \leq i < j$ gilt. Damit können wir b_{ij} als

$$b_{ij} = q_i \cdot \left\lceil \sqrt{b_{ii}} \right\rceil + r_i$$

mit $0 \leq q_i, r_i \leq \lfloor \sqrt{b_{ii}} \rfloor$ schreiben. In diesen Fällen ist die Anzahl der Baby- und Giantsteps also klar. Da wir aber leider keine obere Schranke für b_{jj} kennen, müssen wir zur Bestimmung von b_{jj} dieselbe Strategie wie bei der Bestimmung der Elementordnung in Algorithmus 6.3 anwenden.

6.9. Algorithmus (Bestimmung der Gruppenordnung)

Eingabe: Gruppe $G = \langle g_1, \ldots, g_k \rangle_{\mathbb{Z}}$.

Ausgabe: Gruppenordnung $|G|$.

FOR $(j = 1, j \leq k, j = j + 1)$

DO

$UB_j = 2,\ b_{jj} = \infty$;

WHILE ($b_{jj} = \infty$)

DO

$UB_j = 2 \cdot UB_j$;

FORALL $(0 \leq r_i \leq \lfloor\sqrt{b_{ii}}\rfloor$ für $1 \leq i < j$ AND $0 \leq r_j \leq UB_j)$

DO

Berechne $h = \prod_{i=1}^{j} g_i^{-r_i}$;

Speichere $(h, r_1, \ldots, r_j)$ in einer Tabelle;

OD

$G_j = g_j^{UB_j+1}$;

FORALL $(0 \leq q_i \leq \lfloor\sqrt{b_{ii}}\rfloor$ für $1 \leq i < j$ AND $0 \leq q_j \leq UB_j)$

DO

Berechne $H = \prod_{i=1}^{j} G_i^{q_i}$;

IF $H = h$ für ein Tabellenelement $(h, r_1, \ldots, r_j)$ AND $q_j \cdot (UB_j + 1) + r_j < b_{jj}$

THEN

Setze $b_{ij} = q_i \cdot \lceil\sqrt{b_{ii}}\rceil + r_i$ für $1 \leq i < j$ und $b_{ij} = 0$ für $j + 1 \leq i \leq k$;

Setze $b_{jj} = q_j \cdot (UB_j + 1) + r_j$;

FI

OD

OD

Setze $G_j = g^{\lceil\sqrt{b_{jj}}\rceil}$;

OD

Return $|G| = \prod_{i=1}^{k} b_{ii}$;

6.10. Satz *Der Algorithmus 6.9 benötigt Zeit*

$$O\left(k \cdot \sqrt{|G|} \cdot \log|G| \cdot (\log|G| + T(G))\right),$$

um aus einem Erzeugendensystem für G aus k Elementen die Gruppenordnung zu berechnen.

Beweis: Offensichtlich ist die Anzahl der äußeren Schleifen genau k. Weiterhin gilt $\prod_{i=1}^{j} b_{ii} \leq |G|$ für alle $1 \leq j \leq k$, also insbesondere $b_{ii} \leq |G|$. Beachtet man, daß man alle Babysteps durch eine Multiplikation in der Gruppe ausführen kann (wenn man eine geeignete Reihenfolge wählt), so erkennt man, daß ein Baby- und Giantstep-Schritt jeweils

$O(\sqrt{|G|})$ viele Multiplikationen in der Gruppe braucht. Zusätzlich ist $b_{ii} \leq |G|$ und damit gibt es maximal $O(\log|G|)$ viele Iterationen in der WHILE-Schleife. Setzt man dies zusammen, so erhält man die Laufzeit des Satzes. ∎

6.2 Berechnung der Hermite-Normalform

Wir gehen davon aus, daß wir eine Matrix $B \in Z^{n\times m}$ mit $m \geq n$ kennen, deren Spalten die Erzeuger eines Gitters sind. Die Hermite-Normalform H dieser Matrix ist folgendermassen definiert:

- die ersten $m-n$ Spalten von H sind Null.
- die $n \times n$-Matrix W, die aus den letzten n Spalten von H besteht, ist in Hermite-Normalform, d.h. $w_{ij} = 0$ für $i > j$, $w_{ii} > 0$ und $0 \leq w_{ij} < w_{ii}$ für $1 \leq i < j$.

Wir wollen durch unimodulare Spaltenoperationen die Matrix B so umformen, daß wir die Matrix H in Hermite-Normalform erhalten. Welche unimodularen Operationen können wir benutzen? Sicherlich ist das Vertauschen zweier Spalten eine geeignete Operation. Außerdem können wir das Vielfache einer Spalte zu einer anderen addieren.

Wir gehen dann so vor: Angenommen wir wollen die i-te Zeile in die gewünschte Form bringen, nachdem die Zeilen $i+1,\ldots,n$ schon die gewünschte Gestalt besitzen. Wir bringen zuerst das betragsmäßig kleinste Element b_{ij} für $i \geq j$ an die Stelle (i,i), indem wir die Spalten i und j vertauschen. Anschliessend reduzieren wir mit der i-ten Spalte alle Einträge links von b_{ii}, indem wir ein "geeignetes" Vielfaches der i-ten Spalte subtrahieren. Dann wiederholen wir das Verfahren solange, bis alle Elemente links von b_{ii} Null sind. Durch eventuelle Multiplikation der i-ten Spalte mit -1 kann man nun dafür sorgen, daß $b_{ii} > 0$ ist. Anschliessend reduziert man auf dieselbe Art und Weise auch die Einträge rechts von b_{ii}. Da die Zeilen $i+1,\ldots,n$ schon die gewünschte Gestalt besitzen, erkennt man leicht, daß man sich darin nichts zerstört.

Alle diese Operationen kann man sich merken, wenn man die entsprechende Transformation auch auf die $m \times m$-Einheitsmatrix anwendet. Dies ergibt den folgenden Algorithmus. Dabei sei H_i immer die jeweilige i-te Spalte der Matrix H.

6.11. Algorithmus (Hermite-Normalform)

Eingabe: Matrix $B \in \mathbb{Z}^{n\times m}$ mit $m \geq n$.

Ausgabe: Hermite-Normalform $H \in \mathbb{Z}^{n\times m}$.

Setze $H = B$ und $k = m$;

FOR $(i = n,\ i \geq 1,\ i = i-1)$

DO

WHILE (es gibt $1 \leq j < k$ mit $h_{ij} \neq 0$)

```
        DO
        Berechne 1 ≤ l ≤ k mit h_il = min{|h_ij| > 0 : 1 ≤ j ≤ k};
        IF l < k THEN tausche Spalte H_k und H_l FI
        IF h_ik < 0 THEN H_k = −1 · H_k FI
        FOR (j = 1, j < k, j = j + 1)
            DO
            Setze H_j = H_j − ⌊h_ij / h_ik⌉ · H_k;
            OD
        OD
    IF h_ik < 0 THEN H_k = −1 · H_k FI
    FOR (j = k + 1, j ≤ m, j = j + 1)
        DO
        Setze H_j = H_j − ⌊h_ij / h_ik⌋ · H_k;
        OD
    k = k − 1;
    OD
Return H;
```

6.3 Berechnung der Struktur einer Gruppe

Wir wollen in diesem Abschnitt einen Algorithmus vorstellen, der die Struktur einer Gruppe bestimmt. Der Begriff der Struktur einer Gruppe wird im folgenden Satz definiert.

6.12. Satz *Eine endliche abelsche Gruppe G kann eindeutig dargestellt werden in der Form*

$$G \;=\; G_1 \times G_2 \times \ldots \times G_l,$$

wobei die G_i zyklische Untergruppen von G mit $|G_{i+1}| \,\Big|\, |G_i|$ für $1 \le i < l$ sind.

Die in Satz 6.12 angegebenen Darstellung nennt man auch die **Elementarteiler-Normalform** von G. Um die Struktur von G zu bestimmen, transformieren wir die bei der Bestimmung der Gruppenordnung berechnete Matrix

$$B \;=\; (\underline{b}_1, \ldots, \underline{b}_k)$$

auf Smith-Normalform. Diese ist wie folgt definiert:

6.13. Satz *Es sei $B \in \mathbb{Z}^{k \times k}$ eine nichtsinguläre Matrix. Dann gibt es Transformationsmatrizen $U, V \in Gl(k, \mathbb{Z})$ derart, daß $S = U^{-1} \cdot B \cdot V$ in* **Smith-Normalform (Elementarteiler-Normalform)** *ist, d.h. S ist eine Diagonalmatrix mit Diagonaleinträgen $s_1, \ldots, s_k$, wobei $s_i > 0$ und $s_{i+1} | s_i$ für $1 \le i < k$.*

Der Beweis wird gleich konstruktiv geführt. Zuerst zeigen wir aber noch den Zusammmenhang mit der Elementarteiler-Normalform von G auf:

Sei B die Hermite-Normalform-Basis des Relationengitters von $g_1, \ldots, g_k$. Wir führen folgende Notation ein: Für eine Matrix $A = (a_{ij}) \in \mathbb{Z}^{l \times k}$ definieren wir

$$(g_1, \ldots, g_k)^A = \left(\prod_{i=1}^{k} g_i^{a_{1i}}, \ldots, \prod_{i=1}^{k} g_i^{a_{li}} \right).$$

Dann gilt

$$\begin{aligned}(1, \ldots, 1) &= (g_1, \ldots, g_k)^B \\ &= \left((g_1, \ldots, g_k)^B\right)^{U^{-1}} \\ &= (g_1, \ldots, g_k)^{U^{-1} \cdot B}.\end{aligned}$$

Die letzte Gleichheit kann man leicht durch eine kleine Rechnung verifizieren, wir lassen sie als Übung. Dann kann man fortfahren mit

$$\begin{aligned}(g_1, \ldots, g_k)^{U^{-1} \cdot B} &= (g_1, \ldots, g_k)^{U^{-1} \cdot B \cdot V \cdot V^{-1}} \\ &= (g_1, \ldots, g_k)^{S \cdot V^{-1}} \\ &= \left((g_1, \ldots, g_k)^{V^{-1}}\right)^S \\ &= (g_1', \ldots, g_k')^S.\end{aligned}$$

Hieraus erkennt man , daß S die Hermite-Normalform-Basis für das Relationengitter zu dem Erzeugendensystem $(g_1', \ldots, g_k') = (g_1, \ldots, g_k)^{V^{-1}}$ ist. Ist nun l der letzte Index, für den $s_l \neq 1$ ist, so folgt

$$G \cong \langle g_1' \rangle \times \ldots \times \langle g_l' \rangle$$

ist die Elementarteiler-Normalform von G.

Wir müssen damit nur noch die Smith-Normalform der Matrix B berechnen können, um die Struktur von G zu ermitteln. Wir gehen dabei genauso vor wie bei Berechnung der Hermite-Normalform, nur wenden wir jetzt zusätzlich noch unimodulare Zeilentransformationen an. Die verwendeten unimodularen Transformationen und die Vorgehensweise zur Reduktion von Zeilen- bzw. Spaltenelemente sind analog zur Berechnung der Hermite-Normalform. Dann erhalten wir den folgenden Algorithmus:

6.14. Algorithmus (Smith-Normalform)

Eingabe: Matrix $B \in \mathbb{Z}^{k \times k}$.

Ausgabe: Smith-Normalform S von B.

Setze $S = B$;

FOR $(i = k,\ i \geq 1,\ i = i - 1)$

DO

WHILE (es gibt $1 \leq j < i$ mit $s_{ij} \neq 0$ oder $s_{ji} \neq 0$)

DO

wende unimodulare Zeilentransformationen an, so daß $s_{ji} = 0$ für alle $1 \leq j < i$;

wende unimodulare Spaltentransformationen an, so daß $s_{ij} = 0$ für alle $1 \leq j < i$;

IF $s_{ii} \not| s_{lt}$ für $1 \leq l, t < i$ THEN addiere Zeile l zu Zeile i FI

OD

OD

Warum funktioniert der Algorithmus? Offensichtlich produziert er eine Matrix in Diagonalform. Außerdem ist das i-te Diagonalelement nach der Elimination der größte gemeinsame Teiler aller Elemente der i-ten Zeile und der i-ten Spalte. Die Elimination wird sooft fortgesetzt, bis das neue Diagonalelement alle Elemente in der darüberliegenden Teilmatrix teilt. Diese Eigenschaft ist invariant gegenüber den nun folgenden unimodularen Operationen.

6.4 Wurzelziehen in zyklischen Gruppen

Wir behandeln zuerst das folgende Problem: Für eine Primzahl p und eine Zahl $a \in \mathbb{Z}$ entscheide man, ob a ein Quadrat modulo p ist und wenn ja, ziehe man eine Wurzel modulo p.

6.15. Beispiel Wähle $p = 11$ und $a = 2$. Dann berechnen wir sukzessive durch Probieren alle Quadrate modulo 11 als $1, 4, 9, 5, 3$. Damit erkennen wir direkt, daß 2 kein Quadrat modulo 11 ist.

Dieses Verfahren ist für große Primzahlen p sicherlich nicht praktikabel, denn wir müssen $O(p)$ Versuche machen. Wir können aber folgendes Lemma benutzen:

6.16. Lemma (Euler-Kriterium)
Eine Zahl $a \in \mathbb{Z}$ ist genau dann ein Quadrat modulo p, wenn $a^{(p-1)/2} \equiv 1 \bmod p$ gilt.

Beweis: Wir benutzen zum Beweis den Satz, daß $(\mathbb{Z}/p\mathbb{Z})^*$ zyklisch von der Ordnung $p-1$ ist. Sei etwa z ein Erzeuger dieser Gruppe.

Falls a ein Quadrat modulo p ist, d.h. $a \equiv b^2 \bmod p$ für ein $b \equiv z^{k'} \bmod p$, dann ist $a \equiv (z^{k'})^2 \equiv z^{2 \cdot k'} \bmod p$ und damit ist $a^{(p-1)/2} \equiv (z^{p-1})^{k'} \equiv 1 \bmod p$. Leicht erkennen wir durch dieselbe Vorgehensweise, daß $a \equiv z^k \bmod p$ genau dann ein Quadrat modulo p ist, wenn k gerade ist. Damit folgt die Umkehrung direkt. ∎

6.17. Übung Entwerfe ein ähnliches Kriterium zur Entscheidung, ob $a \in \mathbb{Z}$ eine q-te Potenz modulo p ist.

Damit können wir entscheiden, ob eine ganze Zahl ein Quadrat in der primen Restklassengruppe modulo p ist. Nun wollen wir einen Algorithmus vorstellen, der eine Quadratwurzel berechnen kann. Dabei verallgemeinern wir die Situation etwas: wir nehmen an, daß wir in einer beliebigen zyklischen Gruppe G arbeiten. Weiterhin wollen wir für eine Primzahl q eine q-te Wurzel aus einem beliebigen Gruppenelement $a \in G$ ziehen, wenn eine solche existiert bzw. andernfalls eine Fehlermeldung ausgeben. Wir nehmen weiterhin an, daß wir die Ordnung der Gruppe kennen, etwa $|G| = h$. Nun gibt es zwei mögliche Fälle:

1. $q \not| h$:

   Dann lösen wir mit Hilfe des erweiterten euklidischen Algorithmus die Kongruenz $h \cdot h' \equiv -1 \bmod q$ und berechnen

   $$b = a^{(h \cdot h'+1)/q}.$$

   Damit gilt $b^q = a^{h \cdot h'+1} = a^{h \cdot h'} \cdot a = (a^h)^{h'} \cdot a = a$ und b ist so eine q-te Wurzel von $a \in G$.

2. $q \mid h$:

   Sei etwa $h = q^k \cdot h_0$ mit $q \not| h_0$. Sei G_0 die Untergruppe von G, die alle Elemente von q-Potenzordnung enthält. Dann gilt $|G_0| = q^k$ und G_0 ist zyklisch, da G schon zyklisch war. Sei z ein Erzeuger von G_0.

   Wir berechnen nun h_0' mit $h_0 \cdot h_0' \equiv -1 \bmod q$. Damit können wir das Problem des q-te Wurzelziehens in G darauf reduzieren, in der Untergruppe G_0 eine q-te Wurzel zu ziehen, denn $a^{h_0 \cdot h_0'} = (a^{h_0})^{h_0'} \in G_0$. Hierzu müssen wir eine ganze Zahl $m \in \mathbb{N}$ mit $a^{h_0 \cdot h_0'} = z^m$ bestimmen. Kennen wir die Zahl m, so läßt sich leicht entscheiden, ob a eine q-te Potenz ist, und gegebenenfalls eine q-te Wurzel bestimmen:

   Nehmen wir an, q wäre kein Teiler von m, aber $a = b^q$ eine q-te Potenz. Dann erhalten wir

   $$z^m = a^{h_0 \cdot h_0'} = (b^{h_0})^{h_0' \cdot q}$$

   und da $b^{h_0} \in G_0$ ist und es damit ein $l \in \mathbb{N}$ mit $b^{h_0} = z^l$ gibt, folgt

   $$z^m = z^{l \cdot h_0' \cdot q}.$$

   Da die Ordnung von G_0 gerade q^k ist, gilt damit $m \equiv l \cdot h_0' \cdot q \bmod q^k$. Dies ist aber ein Widerspruch zu unserer Voraussetzung. Falls also q kein Teiler von m ist, kann a keine q-te Potenz sein und wir geben eine Fehlermeldung aus.

   Ansonsten gilt dann $(z^{m/q})^q = a^{h_0 \cdot h_0'}$ und damit können wir ein Element $b \in G$ berechnen als

   $$b = a^{(h_0 \cdot h_0'+1)/q} \cdot z^{-m/q} = a^{(h_0 \cdot h_0'+1)/q} \cdot z^{(q^k-m)/q}.$$

   Dann gilt

   $$b^q = a^{h_0 \cdot h_0'+1} \cdot z^{q^k-m} = a \cdot a^{h_0 \cdot h_0'} \cdot z^{-m} = a$$

   und damit ist b eine gesuchte q-te Wurzel von a.

   Nun bleiben noch zwei Probleme: Das erste besteht darin, einen Erzeuger z für die Gruppe G_0 zu finden. Dazu wählen wir ein Element $z_0 \in G$ zufällig. Dann bestimmen wir $z = z_0^{h_0}$. z ist dann genau dann ein Erzeuger von G_0, wenn z_0 keine q-te Potenz ist. Die Wahrscheinlichkeit dafür ist $\frac{q-1}{q}$, also fast 1. Dies kann man leicht überprüfen, indem man den kleinsten Exponenten $j \geq 1$ mit $z^{q^j} = 1$ bestimmt. Ist $j = k$, so ist z der gewünschte Erzeuger; ansonsten wählen wir ein anderes Element z_0 und beginnen von vorne. Eine zweite Möglichkeit zur Entscheidung dieser Frage liefert die folgende Beobachtung: z ist genau dann ein Erzeuger der Gruppe G_0, wenn $z^{q^{k-1}} \neq 1$ ist.

Die zweite Frage ist: Wie finden wir m? Sei dazu $c \in G_0$ mit $c = z^m$ für eine ganze Zahl $0 \leq m < q^k$ vorgegeben. Wir entwickeln m nach Potenzen von q:

$$m = m_0 + m_1 \cdot q + m_2 \cdot q^2 + \ldots + m_{k-1} \cdot q^{k-1}$$

mit $0 \leq m_i < q$. Sei dann j_i der Index des i-ten von Null verschiedenen Koeffizienten m_j. Es gilt:

$$j_1 = \min_j \{c^{q^{k-j-1}} \neq 1\}.$$

Die Zahl j_1 läßt sich also leicht durch sukzessives Berechnen der q-ten Potenz bestimmen. Seien nun $m_{j_1}, \ldots, m_{j_{l-1}}$ bekannt. Dann ist auch

$$\tilde{m}_{l-1} = m_0 + m_1 \cdot q + \ldots + m_{j_{l-1}} \cdot q^{j_{l-1}}$$

bekannt. Setze $c_l = c \cdot z^{(q^k - \tilde{m}_{l-1})}$. Dann ist $j_l = \min_j \{c_l^{q^{k-j-1}} \neq 1\}$. Damit kann man auf selbe Art wie oben den Index des nächsten von 0 verschiedenen Koeffizienten m_l bestimmen.

Nun bleibt noch die Frage, wie man den genauen Wert des Koeffizienten m_l bestimmt. Für $q = 2$ ist das einfach, denn falls $m_l \neq 0$ ist, muß $m_l = 1$ gelten. Wie können wir jedoch den Wert von m_l für $q \geq 3$ bestimmen?

Angenommen, wir kennen die Zahlen $m_0, \ldots, m_{i-1}$ und damit auch $M_i = m_0 + m_1 \cdot q + \ldots + m_{i-1} \cdot q^{i-1}$. Weiterhin sei bekannt, daß $m_i \neq 0$ ist. Setze $c_i = c \cdot z^{-M_i}$ und bestimme

$$c_i^{q^{k-i-1}} = z^{m_i \cdot q^{k-1}} = (z^{q^{k-1}})^{m_i}.$$

Dies läßt sich auf das Problem des diskreten Logarithmus reduzieren: Um eine Zahl $0 \leq n < q$ mit $y^n = d$ zu bestimmen, schreibe $n = u \cdot \lceil\sqrt{q}\rceil - v$ für $1 \leq u \leq \lceil\sqrt{q}\rceil$ und $0 \leq v \leq \lfloor\sqrt{q}\rfloor$. Dann kann mit der Babystep Giantstep Strategie überprüft werden, ob $d = y^{u \cdot \lceil\sqrt{q}\rceil - v}$ bzw. $d \cdot y^v = \left(y^{\lceil\sqrt{q}\rceil}\right)^u$ gilt. Dazu gehen wir folgendermaßen vor:

- tabelliere und sortiere die Werte $y^v \cdot d$ für $0 \leq v \leq \lfloor\sqrt{q}\rfloor$;
- setze $y_0 = \left(y^{\lceil\sqrt{q}\rceil}\right)$;
- bestimme ein u mit y_0^u kommt schon in der Tabelle vor;

Für diesen Babystep Giantstep Schritt benötigen wir offensichtlich Zeit $O(\sqrt{q} \cdot \log q)$.

Damit können wir eine Gesamtanalyse des Algorithmus durchführen. Wir beschreiben die Analyse für unendliche Menge $\mathcal{G}$ von endlichen zyklischen Gruppen. Annahme:

- Für $G \in \mathcal{G}$ haben wir eine Darstellung, d.h. eine injektive Abbildung in die Menge $\{1, \ldots, n_G\}$, wobei $\log n_G = O(\log |G|)$.

- Wir können in G multiplizieren und invertieren.

Bei der Zeitanalyse nehmen wir an, daß die Gruppenoperationen in Zeit $O(1)$ erfolgen können. Dann erhalten wir für den deterministischen Teil folgende Zeiten:

Bestimmung der Zerlegung $h = h_0 \cdot q^k$	$O(\log h)$
Berechnung $h_0 \cdot h_0' \equiv -1 \bmod q$	$O((\log q)^2 + \log(h) \cdot \log(q))$
Berechnung $c = a^{h_0 \cdot h_0'}$	$O(\log q + \log h)$
Berechnung $c = z^m$	$O((\log h)^2 \cdot \log q)$
diskreter Logarithmus	$O(\sqrt{q} \cdot \log q)$
Berechnung $m = \sum_{i=0}^{k-1} m_i \cdot q^i$	$O((\log h)^2 \cdot \log q)$
Berechnung $b = a^{(h_0 \cdot h_0' + 1)/q} \cdot z^{-m/q}$	$O(\log h)$

Dabei sollte man bei der Berechnung von $c = z^m$ beachten, daß es $k = O(\log h)$ viele Koeffizienten m_i gibt. Weiterhin dauert die Berechnung von $c_i = c_{i-1} \cdot z^{-m_{i-1} \cdot q^{i-1}}$ Zeit $O(\log q)$, die Berechnung $c_{i-1} \cdot (z^{q^{i-1}})^{-m_i}$ benötigt Zeit $O(\log h)$.

Damit erhalten wir die deterministische Gesamtzeit

$$O(\sqrt{q} \cdot \log q + \log(h)^2 \cdot \log q) \;=\; O((\sqrt{q} + \log(h)^2) \cdot \log q).$$

6.18. Satz *Für festes q läßt sich in einer zyklischen endlichen Gruppe G mit bekannter Ordnung die q-te Wurzel mit einem probabilistischen Algorithmus in Zeit $O((\log|G|)^2)$ Gruppenoperationen ziehen.*

Beschäftigen wir uns nun noch mit dem Quadratwurzelziehen modulo zusammengesetzter Zahlen n. Es ist bekannt, daß die Bestimmung aller Quadratwurzeln modulo zusammengesetzter Zahlen polynomiell äquivalent ist zum Faktorisieren:

Sei etwa $n = p \cdot q$. Angenommen, wir hätten einen Algorithmus zum Bestimmen von Quadratwurzeln modulo n, der ohne die Faktorisierung von n arbeitet. Dann wählen wir eine ganze Zahl $\sqrt{n} < x < n$ und berechnen $z \equiv x^2 \bmod n$. Nun ziehen wir alle Quadratwurzeln modulo n aus z. Davon gibt es (hier) 4 Stück. Bestimmen wir unter der Menge dieser Quadratwurzeln y mit $y \not\equiv \pm x \bmod n$. Dann ist n ein Teiler von $x^2 - y^2 = (x-y) \cdot (x+y)$, aber kein Teiler von $x \pm y$. Daher ist $\mathrm{ggT}(x \pm y, n)$ ein echter Teiler von n.

6.19. Übung Entwerfe eine Methode, wie man Quadratwurzeln modulo Primzahlpotenzen ziehen kann (wenn sie existieren). Dabei versuche man, aus einer gegeben Quadratwurzel modulo p^i eine Qudartwurzel modulo p^{i+1} zu bestimmen.

Übungen

1. Berechne mit Hilfe des Euklidischen Algorithmus den ggt der zwei Zahlen
$$a = 20785 \qquad \text{und} \qquad b = 44350$$
sowie die Darstellung des ggt's.

2. Entwickele ein Programm, mit dem man den ggt einer Menge von n ganzen Zahlen berechnen kann. Dabei sei n und eine Liste der Zahlen $(a_1, \ldots, a_n)$ die Eingabe des Programms.

3. Seien $n, m \in \mathbb{N}$ und $a \in \mathbb{Z}_{>1}$. Zeige:
$$ggt(a^m - 1, a^n - 1) \quad = \quad a^{ggt\ (m,n)} - 1.$$

4. Untersuche, wann die **lineare Diophantische Gleichung**
$$a \cdot x + b \cdot y \quad = \quad c$$
zu drei Zahlen $a, b, c \in \mathbb{Z}$ Lösungen $x, y \in \mathbb{Z}$ besitzt. Falls Lösungen existieren, so bestimme sie.

5. Seien $m_1, m_2 \in \mathbb{N}_{>1}$ und $a_1, a_2 \in \mathbb{Z}$. Zeige, daß das System
$$\begin{aligned} x &\equiv a_1 \mod m_1 \\ x &\equiv a_2 \mod m_2 \end{aligned}$$
genau dann eine Lösung x besitzt, wenn $\mathrm{ggT}(m_1, m_2) \mid (a_1 - a_2)$. Hieraus kann man mit vollständiger Induktion die Korrektheit des Verallgemeinerten Chinesischen Restsatzes folgern.

6. Entwickele mit den Hinweisen aus der Vorlesung einen left-to-right Algorithmus zur schnellen Exponentiation.

7. Formuliere eine Verbesserung des Algorithmus zur schnellen Exponentiation durch alternative Behandlung von Blöcken von Einsen in der Binärdarstellung des Exponenten wie in der Vorlesung angegeben.

8. Sei $m \in \mathbb{N}_{>2}$ und $a_1, \ldots, a_{\varphi(m)}$ ein vollständiges Restsystem von $(\mathbb{Z}/m\mathbb{Z})^*$ und $b \in \mathbb{Z}$ eine zu m teilerfremde Zahl, für die auch $b - 1$ zu m teilerfremd ist. Zeige, daß dann die Gleichungen
$$a_1 + a_2 + \ldots + a_{\varphi(m)} \quad \equiv \quad 0 \mod m$$
und
$$1 + b^1 + \ldots + b^{\varphi(m)-1} \quad \equiv \quad 0 \mod m$$
gelten.

9. Sei p eine Primzahl. Zeige, daß, falls es ein Element der Ordnung d in $(\mathbb{Z}/p\mathbb{Z})^*$ gibt, es genau $\varphi(d)$ viele Elemente der Ordnung d gibt.

   Hinweis: Sei $a \in (\mathbb{Z}/p\mathbb{Z})^*$ ein Element der Ordnung d. Zeige, daß alle Elemente mit Exponenten d dann die Form a^j für geeignetes j besitzen. Damit kann der Beweis von Satz 6.6 der Vorlesung berichtigt werden.

10. Zeige, daß für alle $a \in \mathbb{Z}$ und $j \in \mathbb{N}_{>1}$ die folgende Beziehung gilt:

$$a^p \equiv 1 \bmod p^j \qquad \Longrightarrow \qquad a \equiv 1 \bmod p^{j-1}.$$

11. Sei g eine Primitivwurzel modulo p. Zeige, daß dann entweder

$$g^{p-1} \not\equiv 1 \bmod p^2 \qquad \text{oder} \qquad (g+p)^{p-1} \not\equiv 1 \bmod p^2$$

ist.

12. Wie in der Vorlesung angegeben, ist für $i = \sqrt{-1}$ der Ring $\mathbb{Z}(i)$ ein ZPE-Ring. Zeige, daß

$$\mathbb{Z}(i)^* \quad = \quad \{\pm 1, \pm i\}$$

die Menge der Einheiten von $\mathbb{Z}(i)$ ist.

13. Sei $f(x) \in \mathbb{Z}[x]$ ein Polynom vom Grad 3. Zeige, daß die Diskriminante des Polynoms $f(x)$ genau dann positiv ist, wenn die Nullstellen von $f(x)$ reell und paarweise verschieden sind.

14. In dieser Aufgabe soll die Diskriminante eines Polynoms vom Grad zwei explizit berechnet werden. Zeige:

$$\operatorname{disc}\left(a_2x^2 + a_1x + a_0\right) \quad = \quad a_1^2 - 4a_2a_0 \qquad (a_2 \neq 0).$$

Bemerkung: Mit größerem Arbeitsaufwand kann man auf die selbe Art und Weise auch die Diskriminante eines Polynoms vom Grad drei bestimmen. Wer dazu Lust verspürt, kann beweisen, daß

$$\operatorname{disc}\left(a_3x^3 + a_2x^2 + a_1x + a_0\right) = a_2^2a_1^2 - 4a_2^3a_0 - 4a_3a_1^3 - 27a_3^2a_0^2 + 18a_3a_2a_1a_0.$$

15. (a) Zeige, daß alle Basen eines freien R-Moduls (R kommutativer Ring mit 1) die gleiche Mächtigkeit besitzen.
    (b) Sei M ein Untermodul von R^n. Zeige, daß dann $R^n\big/M$ ebenfalls ein R-Modul ist.

16. Sei $R = \mathbb{Q}$ und $\varphi \in \operatorname{End}_{\mathbb{Q}}(\mathbb{Q}^5)$. Bestimme das Bild und den Kern von φ, wenn φ (bzgl. der Standardbasis) die folgende Darstellungsmatrix besitzt:

$$\begin{pmatrix} 1 & 3 & -1 & 2 & 5 \\ 3 & -1 & 2 & -2 & 4 \\ 7 & -9 & 8 & -10 & 2 \\ -2 & 1 & 1 & 0 & 8 \\ 1 & -6 & 11 & -10 & 26 \end{pmatrix}.$$

17. Ergänze die Vektoren

$$\begin{pmatrix} 2 \\ 3 \\ 7 \\ 9 \end{pmatrix} \qquad \text{und} \qquad \begin{pmatrix} 1 \\ 6 \\ 2 \\ 8 \end{pmatrix}$$

zu einer Basis von $\mathbb{Q}^4$ bzw. $\mathbb{Z}^4$.

18. Zeige, daß die Menge

$$M = \{x + y\sqrt{2};\ x, y \in \mathbb{Z}\}$$

als Punktmenge nicht diskret ist. Damit ist M nach Satz 11.1 der Vorlesung kein Gitter.

19. Bestimme mit Hilfe von Algorithmus 11.1 zu der durch die Matrix

$$Q = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 7 & 12 & 14 \\ 3 & 12 & 27 & 36 \\ 4 & 14 & 36 & 60 \end{pmatrix}$$

gegebenen quadratischen Form ein "zugehöriges" Gitter. Überprüfe damit die Korrektheit des Algorithmus 11.1.

20. Beweise oder widerlege folgende Aussage:

Es gibt eine Konstante a_k, die nur von der Dimension k des Gitters L abhängt, so daß für das erste sukzessive Minimum des Gitters L gilt:

$$\lambda_1(L) \quad \geq \quad a_k \cdot \det(L)^{2/k}.$$

21. Versuche eine Lösung des Kugelpackungsproblems im $\mathbb{R}^2$ mit den Hinweisen aus der Vorlesung zu finden.

Hinweis: Die Ungleichungen aus Satz 11.4 der Vorlesung sind leider falsch. Die korrekten Ungleichungen lauten

$$\det(L)^2 \ \leq\ \lambda_1(L) \cdot \ldots \cdot \lambda_k(L) \ \leq\ c_k \cdot \det(L)^2.$$

Im Falle $k = 2$ ist die bestmögliche Konstante $c_2 = \frac{4}{3}$.

22. Verändere Algorithmus 11.2 so, daß er

(a) möglichst effizient einen kürzesten Gittervektor

(b) alle Gittervektoren, deren Länge zwischen zwei Schranken C und C' liegt,

berechnet.

23. Zeige: Die Menge $G = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix};\ a, b, c, d \in \mathbb{Z},\ a \cdot d - b \cdot c = \pm 1 \right\}$ wird von den Matrizen

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \text{und} \qquad C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

erzeugt.

Hinweis: Zeige zuerst, daß A und B die Menge aller Matrizen aus G mit Determinate $+1$ erzeugen. Dazu berechne $D = B^{-1}AB$ und betrachte für eine Matrix $L^{(0)} \in G$ mit Determinante 1 die Folge

$$L^{(2i+1)} = L^{(2i)} \cdot A^{\lambda_{2i+1}} \qquad \text{und} \qquad L^{(2i+2)} = L^{(2i+1)} \cdot D^{\lambda_{2i+2}}$$

für geeignete ganze Zahlen λ_i. Zeige, daß man durch eine solche Iteration Matrizen in Dreiecksform erhalten kann und wie man diese durch A und B erzeugen kann. Folgere schließlich daraus die Aussage der Aufgabe.

24. Formuliere den LLL-Algorithmus 11.4 in einer herkömmlichen Programmiersprache. Bestimme dabei insbesondere die noch fehlenden Neuberechnungen der Werte μ_{ji} bei der Vertauschung zweier Basisvektoren.

25. Beschreibe einen Algorithmus, mit dem man zum sortierten Abspeichern eines Tabellenelements im Shanks-Algorithmus maximal $O(\log(|G|)$ viele Bitoperationen benötigt.

26. Benutze die Babystep-Giantstep Idee von Shanks zur Lösung des diskreten Logarithmus-Problems in einer endlichen abelschen Gruppe G, d.h. bestimme für zwei Elemente $g_1, g_2 \in G$ mit $g_2 \in \langle g_1 \rangle$ den kleinsten Exponenten $j \in \mathbb{N}_0$, so daß $g_1^j = g_2$ ist.

27. Schreibe einen Algorithmus, der die Hermite-Normalform einer Matrix $A \in \mathbb{Z}^{n\times n}$ berechnet.

28. Berechne die Smith-Normalform der Matrix

$$\begin{pmatrix} 2 & 6 & 8 \\ 4 & 7 & 1 \\ 6 & 3 & 4 \end{pmatrix}.$$

Übungen zu Faktorisierung (Theorie siehe Skript zu Faktorisierung)

1. Sei $n = 1 + 2^t \cdot n_0$ mit ungeradem n_0. Zeige, daß für $a \in \mathbb{Z}$ die Gleichheit

$$a^{n-1} - 1 = (a^{n_0} - 1) \cdot (a^{n_0} + 1) \cdot (a^{2n_0} + 1) \cdot \ldots \cdot (a^{2^{t-1}n_0} + 1)$$

gilt. Wie kann man diese Gleichheit verwenden, um einen Pseudoprimzahltest zu entwerfen? Dieser Test wird in der Literatur der Test von Miller genannt.

2. Eine Faktorisierungsmethode, die von Fermat vorgeschlagen wurde, ist die folgende:

Sei n ungerade, zusammengesetzt und kein Quadrat. Man versucht, eine Darstellung der Form

$$n = x^2 - y^2$$

zu finden. Zeige, daß es so eine Darstellung immer gibt und daß $x > \sqrt{n}$ sein muß. Starte mit $x_0 = \lfloor\sqrt{n}\rfloor + 1$, teste, ob obige Gleichung für ein $y \in \mathbb{Z}$ gilt und fahre gegebenenfalls mit $x_{i+1} = x_i + 1$ fort. Formuliere diesen Algorithmus und wende ihn an, um die Zahl 117581 zu faktorisieren. Wann findet dieser Algorithmus schnell eine Lösung?

3. Zeige, daß es immer mindestens eine Lösung von

$$\begin{aligned} x^2 &\equiv y^2 \bmod n \\ x &\not\equiv \pm y \bmod n \end{aligned}$$

gibt, wenn n zwei verschiedene Primfaktoren enthält und ungerade ist. Wieviele Lösungspaare (x, y) gibt es mindestens?

Hinweis: Sei $n = n_1 n_2$ mit n_1 und n_2 teilerfremd und $y \in \mathbb{Z}$, $\operatorname{ggT}(y, n) = 1$. Zeige dann, daß eine Zahl x mit $x \equiv y \bmod n_1$ und $x \equiv -y \bmod n_2$ eine "gute" Wahl ist.

4. Den Test, ob eine Zahl $a \in \mathbb{Z}$ ein Quadrat modulo einer Primzahl p ist, kann man durch Berechnung des Legendre-Symbols $\left(\frac{a}{p}\right)$ durchführen. Schreibe einen Algorithmus, der das Legendre-Symbol $\left(\frac{a}{p}\right)$ berechnet. Benutze dabei einige grundlegende Eigenschaften des Legendre-Symbols, die in jedem Buch über Zahlentheorie zu finden sind.

5. Bestimme eine einfache Formel zur Bestimmung der Nullstellen von
$$z(x) \equiv (x + m)^2 - n \bmod p,$$
wenn $p \equiv 3 \bmod 4$ und n ein Quadrat modulo der Primzahl p ist.

   Hinweis: Betrachte dazu $n^{(p+1)/4} \bmod p$.

6. Gib ein Kriterium analog zum Euler-Kriterium an, wann eine Zahl $a \in \mathbb{Z}$ eine q-te Potenz modulo einer Primzahl p ist (dabei sei q ebenfalls eine Primzahl).

7. Formuliere auf Basis des in der Vorlesung vorgestellten Algorithmus zum Ziehen einer q-ten Wurzel in einer zyklischen Gruppe einen Algorithmus zur Berechnung der Quadratwurzeln modulo einer Primzahl p.

8. Sei die Funktion L_n definiert als
$$L_n(\alpha, \beta) \;=\; \exp\left(\log(n)^\alpha \cdot \log\log(n)^{1-\alpha}\right)^{\beta + o(1)}.$$
Berechne für $\alpha, \beta, \gamma, \delta \in \mathbb{R}$, $0 \le \alpha, \gamma \le 1$ und $k \in \mathbb{N}$ die Werte

   - $L_n(\alpha, \beta) + L_n(\gamma, \delta)$.
   - $L_n(\alpha, \beta) \cdot L_n(\gamma, \delta)$.
   - $L_n(\alpha, \beta) \cdot \log(n)^k$.

9. Analysiere die Laufzeit der "Random Square Methode" von Dixon.

   Hinweis: Gehe analog zur Analyse des quadratischen Siebs vor. Verwende zur Bestimmung der Größe der Faktorbasis die Funktion $L_n[\alpha] := L_n(\frac{1}{2}, \alpha)$.

10. Benutze den in der Vorlesung vorgestellten Algorithmus von Berlekamp-Massey, um das charakteristische Polynom der Folge
$$s_0 = 2,\ s_1 = -1,\ s_2 = 1,\ s_3 = 1,\ s_4 = 5$$
zu bestimmen.

11. Formuliere den Algorithmus von Berlekamp-Massey möglichst effizient in einer Pseudoprogrammiersprache.

12. Sei der Zahlkörper $K = \mathbb{Q}[x] \big/ f(x)\mathbb{Q}[x]$ mit $f(x) = x^3 + 2$ gegeben. Bestimme zu den zwei Elementen

$$a = x^2 + 5x + 2 \qquad \text{und} \qquad b = x^2 - 1$$

aus K die Summe, die Differenz, das Produkt und den Quotienten. Überlege Dir dabei, wie man den Quotienten auch durch Lösen eines linearen Gleichungssystems bestimmen kann.

13. Sei K ein algebraischer Zahlkörper und $\mathcal{O}_K$ der Ring der ganzen Zahlen von K. Zeige, daß für ein Ideal $\mathcal{A}$ von $\mathcal{O}_K$ gilt:

$$N(\mathcal{A}) \;=\; |\,\mathcal{O}_K/\mathcal{A}\,|\,.$$

Hinweis: Zeige, daß es eine Basis $\{\omega_1, \ldots, \omega_d\}$ von $\mathcal{O}_K$ und eine Basis von $\mathcal{A}$ gibt, die die Gestalt

$$\{a_1 \cdot \omega_1,\, a_2 \cdot \omega_2,\, \ldots,\, a_d \cdot \omega_d\}$$

besitzt, wobei die Zahlen $a_i \in \mathbb{Z}$ sind. Folgere daraus das gewünschte Resultat.

14. Sei der Zahlkörper $K = \mathbb{Q}[\sqrt[3]{2}\,]$ gegeben. Bestimme alle Primideale von $\mathcal{O}_K$, deren Norm eine Potenz von 5 ist.