Effiziente Algorithmen für Probleme der linearen Algebra über $\mathbb{Z}$

Diplomarbeit

von

Achim Müller

nach einem Thema von Prof. Dr. Buchmann,
am Fachbereich 14 - Informatik -
der Universität des Saarlandes

Saarbrücken, den 22. April 1994

Eidesstattliche Erklärung

"Hiermit versichere ich an Eides statt, daß ich diese Arbeit selbständig, und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe."

Saarbrücken, den 22. April 1994

Achim Müller

An dieser Stelle möchte ich mich bei all denen bedanken, die zum Gelingen dieser Arbeit beigetragen haben, insbesondere Patrick Theobald, Sandro Amendola, Volker Müller, H.W.Lenstra, Stefan Neis, Christine Hollinger und Thomas Denny.

Mein besonderer Dank gilt Herrn Prof. Dr. Buchmann, der mich in Diskussionen mit Ideen und Anregungen unterstützt hat. Seine Kommentare und Rückfragen haben mich immer wieder dazu gebracht, einzelne Teile neu zu überdenken und besser zu verstehen.

Last but not least möchte ich mich auch bei meinem Vater bedanken, der mir den nötigen Freiraum verschaffte, den eine solche Arbeit erfordert.

Inhaltsverzeichnis

Einleitung

Wir beschäftigen uns in der vorliegenden Arbeit mit linearer Algebra über dem Ring der ganzen Zahlen. Für einen $\mathbb{Z}$-Modulhomomorphismus $\varphi : \mathbb{Z}^n \to \mathbb{Z}^m$ untersuchen wir die Zeitkomplexität typischer Fragestellungen, wie zum Beispiel die Berechnung des Ranges. Als Kostenmaß verwenden wir die Bitkomplexität arithmetischer Operationen in $\mathbb{Z}$.

Die Darstellung gliedert sich im Einzelnen wie folgt :

Teil I : Einführung

Als Einstieg wiederholen wir einige Grundbegriffe der linearen Algebra über Moduln. Dann formulieren wir das Thema dieser Arbeit, also welche Probleme wir untersuchen, und geben einen Überblick über die wichtigsten Ergebnisse. Wir stellen ferner einfache zahlentheoretische Algorithmen vor, die wir Basisalgorithmen nennen, da sie der gemeinsame Ausgangspunkt für spätere Kapitel sind. Insbesondere erläutern wir ein Konzept, in dessen Mittelpunkt der Chinesische Restsatz steht, und welches oft als modulare Arithmetik bezeichnet wird. Wir verwenden es, um ein unkontrolliertes Anwachsen von Zwischenergebnissen zu vermeiden.

Teil II : Lineare Algebra über Körpern

Eine zentrale Aussage der modularen Arithmetik ist, daß wir unter gewissen Vorraussetzungen Berechnungen in $\mathbb{Z}^{m\times n}$ durch analoge Operationen in geeigneten Primkörpern ersetzen können. Aus diesem Grund lösen wir die von uns betrachteten Probleme zuerst über Körpern, und leiten Aussagen der Art

“ *Um den Rang von $A \in \mathbb{K}^{m\times n}$ zu berechnen,*
benötigen wir $O(m^2 n)$ Körperoperationen in $\mathbb{K}$ “

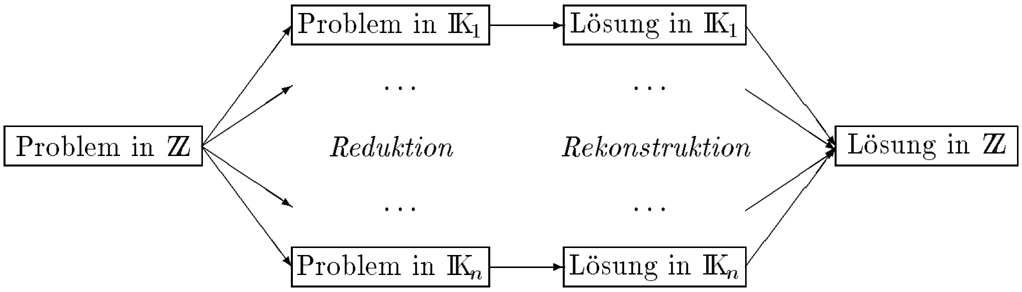
her. Wir führen ferner einige hilfreiche Bezeichnungen ein.

Teil III : Elementare Probleme der linearen Algebra über $\mathbb{Z}$

Für jene Fragestellungen, die sich durch modulare Arithmetik bewältigen lassen, geben wir sowohl Algorithmen als auch Laufzeitanalysen an. Dies sind unter anderem

- Determinante und charakteristisches Polynom
- Adjungierter Homomorphismus und Rang

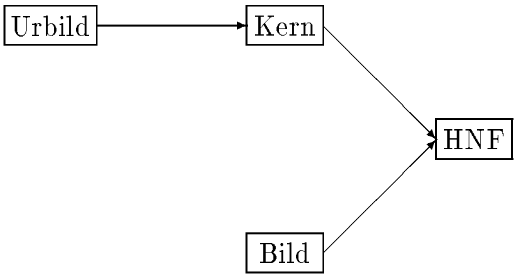
Wir betonen, daß allen Verfahren ein einheitlicher Lösungsansatz gemeinsam ist, welcher sich in eine Reduktions- und eine Rekonstruktionsphase zerlegen läßt :



Auf der Grundlage der über Körpern formulierten Algorithmen berechnen wir für jedes der Teilprobleme eine Lösung, wobei $\mathbb{K}_1, \ldots, \mathbb{K}_n$ geeignete Primkörper sind. Die Teillösungen setzen wir anschließend mit dem Chinesischen Restsatz zur Gesamtlösung in $\mathbb{Z}$ zusammen.

Teil IV : Kern, Bild und Urbild über $\mathbb{Z}$

Obwohl die modulare Arithmetik für viele Probleme das Mittel der Wahl ist, gibt es dennoch solche, die sich ihrer Anwendung entziehen. Beispiel sind Kern, Bild und Urbild. Als Ersatz verwenden wir eine Reduktion auf die Hermite-Normalform :



Auf der Grundlage des Gitterbegriffes erläutern wir ausführlich deren Berechnung, und konstruieren darauf aufbauend dann Algorithmen für Kern und Bild.

Die beigefügten Verzeichnisse erhöhen, so hoffen wir, die Lesbarkeit.

Teil I

Einführung

Lineare Algebra über $\mathbb{Z}$

1.1 Freie Moduln über Ringen

Eine mathematische Struktur, die in der linearen Algebra betrachtet wird, sind Vektorräume über Körpern. Wir erlauben als Grundlage für die äußere Verknüpfung , also die Multiplikation, auch Ringe und erhalten eine naheliegende Verallgemeinerung.

1.1.1 Begriff des Moduls

Definition 1.1 (R-Modul)
Es sei R ein Ring mit Einselement. Ein **Modul** *über R ist ein Tripel $(M, +, \cdot)$, wobei M eine nichtleere Menge und*

$$\begin{array}{ccccc} + & : & M \times M & \rightarrow & M \\ \cdot & : & R \times M & \rightarrow & M \end{array}$$

zwei Abbildungen sind, welche die folgenden Axiome erfüllen :

1. *$(M, +)$ ist eine abelsche Gruppe*
2. *Distributivgesetze :*
   Für alle $\alpha, \beta \in R$ und alle $a, b \in M$ ist
   $$(\alpha + \beta) \cdot a = \alpha \cdot a + \beta \cdot a$$
   $$\alpha \cdot (a + b) = \alpha \cdot a + \alpha \cdot b$$
3. *Assotiativgesetz für die Multiplikation :*
   Für alle $\alpha, \beta \in R$ und beliebige $a \in M$ gilt
   $$\alpha \cdot (\beta \cdot a) = (\alpha \cdot \beta) \cdot a$$
4. *Für das Einselement $1 \in R$ und jedes $a \in M$ ist*
   $$1 \cdot a = a$$

Additive Untergruppen von M, die selbst Moduln sind, nennen wir Untermoduln.

Definition 1.2 (R-Untermodul)
Ein R-Modul $(U, \oplus, \odot)$ heißt R-Untermodul von $(M, +, \cdot)$, wenn gilt :

1. *$U \subseteq M$*
2. *Für beliebige $a, b \in U$ und $\alpha \in R$ ist*
   $$a \oplus b = a + b$$
   $$\alpha \odot a = \alpha \cdot a$$

Wir notieren diese Beziehung durch $U \leq_R M$.

1.1.2 Eigenschaften von Moduln

Zunächst stellen wir fest, daß eine ganze Reihe elementarer Begriffe, die wir für Vektorräume kennen, ganz analog auch für Moduln erklärt sind.

Definition 1.3 *Es seien $(M, +, \cdot)$ ein R-Modul und $m_1, \ldots, m_k \in M$.*

1. *Das Tupel $(m_1, \ldots, m_k)$ heißt* **linear unabhängig** *oder* **frei** *über R, wenn gilt :*

$$\sum_{i=1}^{k} \alpha_i \cdot m_i \;=\; 0_M \quad \text{mit} \quad \alpha_i \in R \quad \Rightarrow \quad \alpha_1 = \ldots = \alpha_k = 0_R$$

2. *Gibt es eine nicht-triviale Darstellung der Null*

$$\sum_{i=1}^{k} \beta_i \cdot m_i = 0_M \quad \text{und} \quad \beta_j \neq 0_R \quad \text{für ein } j \in \{1, \ldots, k\}$$

   nennen wir $(m_1, \ldots, m_k)$ **linear abhängig** *über R.*

3. *Die* **lineare Hülle** *von $(m_1, \ldots, m_k)$ über R ist definiert als*

$$\langle m_1, \ldots, m_k \rangle_R \;:=\; \left\{ m \in M \;:\; m = \sum_{i=1}^{k} \gamma_i \cdot m_i \quad \text{mit} \quad \gamma_i \in R \right\}$$

   und bildet einen R-Untermodul von M :

$$\langle m_1, \ldots, m_k \rangle_R \;\leq_R\; M$$

4. *Das Tupel $(m_1, \ldots, m_k)$ heißt* **Erzeugendensystem** *von M, wenn gilt :*

$$\langle m_1, \ldots, m_k \rangle_R \;=\; M$$

Daß die Analogie zu Vektorräumen bei unvorsichtiger Verwendung rasch zusammenbricht, belegt das folgende Lemma, welches die nachfolgende Definition motiviert.

Lemma 1.1 *Nicht jeder Modul besitzt ein freies Erzeugendensystem.*

Beweis: Es sei $(G, +)$ eine endliche abelsche Gruppe mit mindestens zwei Elementen und $|G|$ die Ordnung von G. Definiere eine Multiplikation $\odot$ über $\mathbb{Z}$ durch

$$\left.\begin{array}{rcll} n \odot a & := & a^n & \forall n \in \mathbb{N} \\ 0 \odot a & := & 0_G & \text{(neutrales Element)} \\ (-1) \odot a & := & a^{-1} & \end{array}\right\} \text{ für } a \in G$$

Auf diese Weise wird $(G, +, \odot)$ zum $\mathbb{Z}$-Modul. Andererseits gilt aber

$$a^{|G|} \;=\; 0_G \qquad \forall a \in G$$

bzw.

$$|G| \odot a \;=\; 0_G \qquad \forall a \in G$$

Somit existiert für jedes Tupel $(m_1, \ldots, m_k)$ mit $m_i \in G$ eine nicht-triviale Darstellung der Null, d.h. jedes Erzeugendensystem von $(G, +, \odot)$ ist linear abhängig über $\mathbb{Z}$. ■

Definition 1.4 (Freier Modul)
Es seien $(M, +, \cdot)$ ein R-Modul und $(m_1, \ldots, m_k)$ ein Erzeugendensystem von M.

1. *Das Tupel $(m_1, \ldots, m_k)$ heißt* **Basis** *von M, wenn es linear unabhängig über R ist.*
2. *M heißt* **frei** *, wenn es eine Basis für M gibt.*

Die Existenz freier Moduln ist gesichert, da jeder Vektorraum auch ein Modul ist.

Lemma 1.2 (Kanonische Standardbasis von $\mathbb{Z}^n$)
Die Einheitsvektoren $e_1^n, \ldots, e_n^n$ bilden für jedes $n \in \mathbb{N}$ eine Basis des $\mathbb{Z}$-Moduls $\mathbb{Z}^n$.

1.1.3 Modulhomomorphismen

Wir wenden uns dem Gegenstand dieser Arbeit zu.

Definition 1.5 (Modulhomomorphismus)
Es seien $(M, +, \cdot)$ und $(N, +, \cdot)$ Moduln über einem Ring R. Eine Abbildung

$$\varphi : M \to N$$

heißt **Modulhomomorphismus** *genau dann, wenn gilt :*

1. $\varphi(m + n) \;=\; \varphi(m) + \varphi(n) \qquad \forall m, n \in M$
2. $\varphi(r \cdot m) \;=\; r \cdot \varphi(m) \qquad \forall r \in R \quad \forall m \in M$

Ist M und N ein und derselbe Modul, so nennen wir φ einen **Modulendomorphismus**.

Definition 1.6 ($\mathbb{Z}$-Modulhomomorphismus)
Als Notationen vereinbaren wir

$$\mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m) \;:=\; \{\varphi : \mathbb{Z}^n \to \mathbb{Z}^m \;\mid\; \varphi \textit{ ist ein } \mathbb{Z}\textit{-Modulhomomorphismus } \}$$

für die von uns betrachtete Menge der $\mathbb{Z}$-Modulhomomorphismen, sowie

$$\mathrm{End}(\mathbb{Z}^n) \;:=\; \{\varphi : \mathbb{Z}^n \to \mathbb{Z}^n \;\mid\; \varphi \textit{ ist ein } \mathbb{Z}\textit{-Modulendomorphismus } \}$$

für die entsprechenden $\mathbb{Z}$-Modulendomorphismen.

Da Homomorphismen **mathematische Objekte** sind, benötigen wir **Darstellungen**, um ihre Eigenschaften untersuchen zu können. Wir beschränken uns in der nachfolgenden Definition auf den Spezialfall $\varphi : \mathbb{Z}^n \to \mathbb{Z}^m$, da wir nur diesen benötigen.

Definition 1.7 (Darstellung eines Homomorphismus)
Es sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ *. Als Darstellung von* φ *bezeichnen wir die Matrix*

$$\mathrm{Mat}(\varphi) := (\varphi(e_1^n), \ldots, \varphi(e_n^n)) \in \mathbb{Z}^{m\times n}$$

welche die Bedingung

$$\mathrm{Mat}(\varphi) \cdot z = \varphi(z) \qquad \forall z \in \mathbb{Z}^n$$

erfüllt. Umgekehrt ordnen wir einer Matrix $A \in \mathbb{Z}^{m\times n}$ *den Homomorphismus*

$$\varphi_A(x) := A \cdot x \quad , \quad \varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m) \quad , \quad x \in \mathbb{Z}^n$$

zu.

Diese Festlegung ist umkehrbar eindeutig (bezüglich der kanonischen Standardbasen, die wir hier voraussetzen).

Lemma 1.3 (Verknüpfung von Homomorphismen)
Es seien $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ *und* $\psi \in \mathrm{Hom}(\mathbb{Z}^m, \mathbb{Z}^k)$*. Dann hat das Produkt*

$$\psi \circ \varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^k)$$

die Darstellung

$$\mathrm{Mat}(\psi \circ \varphi) = \mathrm{Mat}(\psi) \cdot \mathrm{Mat}(\varphi)$$

1.2 Vorstellung des Themas

1.2.1 Formulierungen in der linearen Algebra

Wir weisen zuerst auf einen einfachen, aber oft nicht beachteten Unterschied in der Formulierung von Problemstellungen der linearen Algebra hin: wir interessieren uns zwar für Homomorphismen, welche **mathematische Objekte** sind, haben aber nur **Darstellungen** zur Verfügung, um ihre Eigenschaften zu untersuchen. Im Rahmen dieser Arbeit verwenden wir hierfür Matrizen. Welche praktischen Konsequenzen hat dies ?
Wir betrachten als Beispiel die Determinante eines Endomorphismus $\varphi \in \mathrm{End}(\mathbb{Z}^n)$:

$$\det(\varphi) = \sum_{\sigma\in\mathcal{S}_n} \left(\mathrm{sign}(\sigma) \cdot \prod_{i=1}^{n} m_{i,\sigma(i)}\right) \quad , \quad \mathrm{Mat}(\varphi) = (m_{i,j})$$

Das Bemerkenswerte an dieser Definition ist, daß die Zuordnung

$$\det : \mathrm{End}(\mathbb{Z}^n) \to \mathbb{Z} \quad , \quad \varphi \to \det(\varphi) \qquad (*)$$

streng genommen nur für Abbildungen erklärt ist. Umgekehrt können wir natürlich auch einer Matrix $A = (a_{i,j}) \in \mathbb{Z}^{n\times n}$, die wir jetzt als ein quadratisches Organisationsschema von n^2 Zahlen auffassen, eine willkürliche Zahl

$$\det(A) = \sum_{\sigma\in\mathcal{S}_n} \left(\mathrm{sign}(\sigma) \cdot \prod_{i=1}^{n} a_{i,\sigma(i)}\right) \qquad (**)$$

zuordnen, die wir dann als Determinante von A bezeichnen. Wodurch unterscheiden sich beide Ansätze ? Wir nennen (*) die **invariante** und (**) die **matrixorientierte** Formulierung der Determinante. Solange wir Homomorphismen durch Matrizen darstellen, mag diese Trennung künstlich erscheinen, aber nicht bei allen Problemen ähneln beide Ansätze einander so stark (vgl. Hermite-Normalform).
In den folgenden Abschnitten stellen wir nun die im Rahmen dieser Arbeit untersuchten Fragen vor. Wir bevorzugen dabei invariante Formulierungen, um zu betonen, daß wir uns in erster Linie für Homomorphismen interessieren.

1.2.2 Kern und Bild

Definition 1.8 *Es sei* $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$*. Als* **Kern** *bzw.* **Bild** *von* φ *bezeichnen wir*

$$\mathrm{Kern}(\varphi) := \{x \in \mathbb{Z}^n \mid \varphi(x) = 0\}$$

und

$$\mathrm{Bild}(\varphi) := \{y \in \mathbb{Z}^m \mid y = \varphi(x) \text{ für ein } x \in \mathbb{Z}^n\}$$

Beide Mengen sind freie Untermoduln von $\mathbb{Z}^n$ *bzw.* $\mathbb{Z}^m$*.*

Zur Darstellung von Kern und Bild eignen sich also Basen.

KERN und BILD	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathbb{Z}^{m\times n}$
Gesucht	Basen von $\mathrm{Kern}(\varphi)$ und $\mathrm{Bild}(\varphi)$

1.2.3 Urbild

Eine interessante Variante des Bildproblemes besteht darin, für einen Vektor $b \in \mathbb{Z}^m$ und einen Homomorphismus $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ zu überprüfen, ob $b \in \mathrm{Bild}(\varphi)$ oder $b \notin \mathrm{Bild}(\varphi)$ gilt, und gegebenenfalls ein $x \in \mathbb{Z}^n$ mit $\varphi(x) = b$ zu bestimmen.

URBILD	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathbb{Z}^{m\times n}$ Vektor $b \in \mathbb{Z}^m$
Gesucht	Information $b \notin \mathrm{Bild}(\varphi)$ oder ein Urbild $x \in \mathbb{Z}^n$ $\varphi(x) = b$ von b unter φ

Für reguläre Endomorphismen[1] betrachten wir noch eine Erweiterung der Urbildaufgabe, die dadurch motiviert ist, daß in Anwendungen manchmal die Frage auftaucht, ob im Fall $b \notin \mathrm{Bild}(\varphi)$ ein geeignetes Vielfaches $\theta \in \mathbb{N}$ mit $\theta \cdot b \in \mathrm{Bild}(\varphi)$ existiert.

REGULÄRES URBILD	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathrm{GL}(n, \mathbb{Z})$ Vektoren $b_1, \ldots, b_k \in \mathbb{Z}^n$
Gesucht	Minimale Vielfache $\theta_1, \ldots, \theta_k \in \mathbb{N}$ sowie Urbilder $x_1, \ldots, x_k \in \mathbb{Z}^n$ mit $\varphi(x_i) = \theta_i \cdot b_i$

1.2.4 Rang und linear unabhängige Zeilen

Definition 1.9 *Sei* $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ *und* $u_1, \ldots, u_r \in \mathbb{Z}^m$ *eine Basis von* $\mathrm{Bild}(\varphi)$*. Dann nennen wir die eindeutig bestimmte Zahl*

$$\mathrm{rang}(\varphi) := r \in \{1, \ldots, \min\{m, n\}\}$$

den **Rang** *von* φ*.*

Eine einfache Methode, um den Rang von φ zu berechnen, ist die Bestimmung einer Basis von $\mathrm{Bild}(\varphi)$. Der Nachteil dabei ist, daß wir zuviel Information berechnen, denn uns interessiert nur die *Länge* einer Basis und nicht die Basis selbst. Geschickter ist eine andere Vorgehensweise, die wir in Kapitel 3.7 vorstellen.

RANG	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathbb{Z}^{m \times n}$
Gesucht	Rang von φ

Da der Rang alleine wenig aussagt, interessieren wir uns außerdem noch dafür, wie diese Zahl aufgeschlüsselt werden kann.

LINEAR UNABHÄNGIGE ZEILEN	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathbb{Z}^{m \times n}$
Gesucht	Indizes $i_1, \ldots, i_r \in \{1, \ldots, m\}$ linear unabhängiger Zeilen aus $\mathrm{Mat}(\varphi)$ mit $r = \mathrm{rang}(\varphi)$

[1] siehe Definition 3.45, Seite 108

1.2.5 Determinante

Definition 1.10 *Es sei* $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ *und* $\mathrm{Mat}(\varphi) = (m_{i,j}) \in \mathbb{Z}^{n\times n}$. *Dann heißt*

$$\det(\varphi) := \sum_{\sigma\in\mathcal{S}_n} \left(\mathrm{sign}(\sigma) \cdot \prod_{i=1}^{n} m_{i,\sigma(i)} \right) \in \mathbb{Z}$$

Determinante *von* φ, *wobei* $\mathcal{S}_n$ *die Menge der Permutationen von* $\{1,\ldots,n\}$ *bezeichnet und* $\mathrm{sign}(\sigma) \in \{+1,-1\}$ *das Vorzeichen von* σ *ist.*

DETERMINANTE	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathbb{Z}^{n\times n}$
Gesucht	Determinante von φ

1.2.6 Adjungierter Homomorphismus

Wir wollen für einen regulären Endomorphismus[2] $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ ein $\psi \in \mathrm{End}(\mathbb{Z}^n)$ mit

$$\psi \circ \varphi = \varphi \circ \psi = \det(\varphi) \cdot \mathrm{id} \;,\; \det(\varphi) \in \mathbb{Z} \setminus \{0\}$$

konstruieren, wobei *id* die identische Abbildung auf $\mathbb{Z}^n$ sei.

Definition 1.11 *Als* **adjungierten Homomorphismus** *von* $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ *bezeichnen wir die eindeutig bestimmte Abbildung* $\mathrm{adj}(\varphi) \in \mathrm{End}(\mathbb{Z}^n)$, *welche*

$$\varphi \circ \mathrm{adj}(\varphi) = \mathrm{adj}(\varphi) \circ \varphi = \det(\varphi) \cdot \mathrm{id}$$

erfüllt.

ADJUNGIERTER HOMOMORPHISMUS	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathrm{GL}(n,\mathbb{Z})$
Gesucht	Adjungierter Homomorphismus von φ

1.2.7 Charakteristisches Polynom

Definition 1.12 *Es sei* X *eine Unbestimmte über* $\mathbb{Z}$ *und* $\mathbb{Z}[X]$ *der Polynomring in* X. *Für ein* $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ *ist*

$$c_\varphi(X) := \det(\varphi - X \cdot id) \in \mathbb{Z}[X]$$

ein Polynom vom Grad n. *Wir nennen* c_φ *das* **charakteristische Polynom** *von* φ.

CHARAKTERISTISCHES POLYNOM	
Gegeben	Matrix $\mathrm{Mat}(\varphi) \in \mathbb{Z}^{n\times n}$
Gesucht	Charakteristisches Polynom von φ

[2] siehe Definition 3.45, Seite 108

1.2.8 Hermite-Normalform

Wir interessieren uns schließlich noch für die Frage, wie wir eine Matrix $A \in \mathbb{Z}^{m\times n}$, die vollen Zeilenrang hat, unimodular auf Hermite-Normalform transformieren können.

HERMITE-NORMALFORM	
Gegeben	Matrix $A \in \mathbb{Z}^{m\times n}$ mit $\mathrm{rang}(A) = m$
Gesucht	Hermite-Normalform $\mathrm{HNF}(A) \in \mathbb{Z}^{m\times n}$ von A, sowie eine Transformationsmatrix $T \in \mathbb{Z}^{n\times n}$ mit $A \cdot T = \mathrm{HNF}(A)$ und $\det(T) \in \{-1, +1\}$

1.3 Ergebnisse

Definition 1.13 *Es sei $A = (a_{i,j}) \in \mathbb{Z}^{n\times n}$. Dann notieren wir durch*

$$|A| := \max\{|a_{i,j}| : 1 \le i \le m\,,\, 1 \le j \le n\}$$

den Betrag des größten Eintrages in A. Entsprechend definieren wir für $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$

$$|\varphi| := |\mathrm{Mat}(\varphi)|$$

Wir stellen im folgenden die Ergebnisse unserer Arbeit in knapper Form zusammen.

1.3.1 Elementare Probleme

Determinante und charakteristisches Polynom Es sei $\varphi \in \mathrm{End}(\mathbb{Z}^n)$. Dann benötigen wir jeweils Zeit

$$O(n^4 \cdot \log^2(n|\varphi|))$$

um die Determinante und das charakteristische Polynom von φ zu berechnen.

Adjungierter Homomorphismus Es sei $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ ein regulärer Endomorphismus. Dann benötigen wir Zeit

$$O(n^4 \cdot \log^2(n|\varphi|))$$

um den adjungierten Homomorphismus von φ zu berechnen.

Rang und linear unabhängige Zeilen Es sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ und $s = \min\{m, n\}$. Dann benötigen wir jeweils Zeit

$$O(s \cdot m^2 n \cdot \log^2(s|\varphi|))$$

um den Rang r von φ sowie Indizes

$$i_1, \ldots, i_r \in \{1, \ldots, m\}$$

linear unabhängiger Zeilen aus $\mathrm{Mat}(\varphi)$ zu berechnen.

Reguläres Urbild Es seien $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ ein regulärer Endomorphismus und

$$b_1, \ldots, b_k \in \mathbb{Z}^n$$

mit

$$|b_i| \leq B \qquad \forall i \in \{1, \ldots, k\}$$

Dann benötigen wir Zeit

$$O(n^3 \cdot (n+k) \cdot (\log(n|\varphi|) + \log(B))^2)$$

um sowohl $x_1, \ldots, x_k \in \mathbb{Z}^n$ als auch minimale Vielfache $\theta_1, \ldots, \theta_k \in \mathbb{N}$ mit

$$\varphi(x_i) = \theta_i \cdot b_i \qquad \forall i \in \{1, \ldots, k\}$$

zu berechnen.

1.3.2 Hermite-Normalform

Hermite-Normalform Es sei $A \in \mathbb{Z}^{m \times n}$ und $\mathrm{rang}(A) = m$. Dann benötigen wir Zeit

$$O(m^2 n \cdot (m^2 + n) \cdot \log^2(m|A|))$$

um die HNF von A zu berechnen. Die Bestimmung der Transformationsmatrix kostet Zeit

$$O(n^3 m^2 \cdot \log^2(m|A|))$$

1.3.3 Kern, Bild und Urbild

1.3.3.1 Allgemeiner Fall

Kern Es sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$. Dann benötigen wir mit

$$s = \min\{m, n\} \quad , \quad r = \mathrm{rang}(\varphi)$$

einen Zeitaufwand der Größenordnung

$$O(n \cdot (sm^2 + n^2 r^2) \cdot \log^2(s|\varphi|))$$

um den Kern von φ zu berechnen. Gilt zusätzlich $m \leq n$, vereinfacht sich dies zu

$$O(n^3 \cdot (m + r^2) \cdot \log^2(m|\varphi|))$$

Bild Es sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$. Dann benötigen wir mit

$$s = \min\{m, n\} \quad , \quad r = \mathrm{rang}(\varphi)$$

einen Zeitaufwand der Größenordnung

$$O(n \cdot (sm^2 + n^2 r^2 + mr \cdot (r + \log(n))) \cdot \log^2(s|\varphi|))$$

um das Bild von φ zu berechnen. Gilt zusätzlich $m \leq n$, vereinfacht sich dies zu

$$O(n^3 \cdot (m + r^2) \cdot \log^2(m|\varphi|))$$

Urbild Es sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ und $b \in \mathbb{Z}^m$. Dann benötigen wir mit

$$s = \min\{m, n\} \quad , \quad \delta = \max\{|\varphi|, |b|\} \quad , \quad r = \mathrm{rang}(\varphi)$$

einen Zeitaufwand der Größenordnung

$$O(n \cdot (sm^2 + n^2 r^2) \cdot \log^2(s\delta))$$

um ein Urbild von b unter φ zu berechnen. Gilt zusätzlich $m \leq n$, vereinfacht sich dies zu

$$O(n^3 \cdot (m + r^2) \cdot \log^2(m\delta))$$

1.3.3.2 Endomorphismen

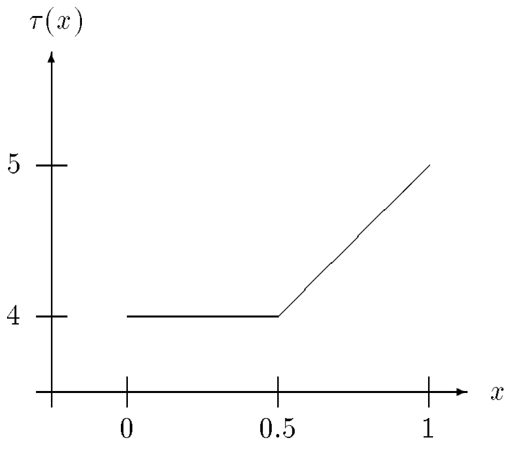
Definition 1.14 (τ-Funktion) *Wir bezeichnen mit τ eine Funktion*

$$\tau : [0, 1] \to [4, 5]$$

die wir wie folgt definieren :

$$\tau(x) := \max\{4, 3 + 2x\}$$

Der Funktionsgraph von τ hat die Form



Auf der Grundlage dieser Notation lassen sich die Laufzeitaussagen für Endomorphismen einfacher formulieren.

Kern Es sei $\varphi \in \mathrm{End}(\mathbb{Z}^n)$. Dann benötigen wir mit

$$\mathrm{rang}(\varphi) = O(n^\epsilon) \quad , \quad \epsilon \in [0,1]$$

einen Zeitaufwand der Größenordnung

$$O(n^{\tau(\epsilon)} \cdot \log^2(n|\varphi|))$$

um den Kern von φ zu berechnen. Gilt sogar

$$\mathrm{rang}(\varphi) = O(\sqrt[2]{n})$$

vereinfacht sich dies zu

$$O(n^4 \cdot \log^2(n|\varphi|))$$

Bild Es sei $\varphi \in \mathrm{End}(\mathbb{Z}^n)$. Dann benötigen wir mit

$$\mathrm{rang}(\varphi) = O(n^\epsilon) \quad , \quad \epsilon \in [0,1]$$

einen Zeitaufwand der Größenordnung

$$O(n^{\tau(\epsilon)} \cdot \log^2(n|\varphi|))$$

um das Bild von φ zu berechnen. Gilt sogar

$$\mathrm{rang}(\varphi) = O(\sqrt[2]{n})$$

vereinfacht sich dies zu

$$O(n^4 \cdot \log^2(n|\varphi|))$$

Urbild Es sei $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ und $b \in \mathbb{Z}^n$. Dann benötigen wir mit

$$\mathrm{rang}(\varphi) = O(n^\epsilon) \quad , \quad \epsilon \in [0,1]$$

sowie $\delta = \max\{|\varphi|, |b|\}$ einen Zeitaufwand der Größenordnung

$$O(n^{\tau(\epsilon)} \cdot \log^2(n\delta))$$

um ein Urbild von b unter φ zu berechnen. Gilt sogar

$$\mathrm{rang}(\varphi) = O(\sqrt[2]{n})$$

vereinfacht sich dies zu

$$O(n^4 \cdot \log^2(n\delta))$$

Basisalgorithmen

Der Entwurf effizienter Algorithmen für verwandte Probleme erfordert - um ein gewisses Maß an Übersichtlichkeit zu gewährleisten - eine in Ebenen gegliederte Betrachtungsweise. Auf der untersten Ebene befinden sich in der vorliegenden Arbeit einfache zahlentheoretische Algorithmen, die wir *Basisalgorithmen* nennen und im folgenden vorstellen. Diese besitzen zwar keine eigenständige Bedeutung, bilden aber das Fundament weiterer Kapitel.

1.4 Grundlagen seminumerischer Algorithmen

Um über Algorithmen zu reden, die mit ganzen Zahlen operieren, benötigt man zunächst einmal zwei Dinge :

- eine Übereinkunft, wie Zahlen zu repräsentieren sind
- einen sinnvollen Laufzeitbegriff

Das Ziel dieses einführenden Kapitels ist es, beides zu präzisieren. Eine ausführliche Beschreibung ist in [Mehlh] , Kap.1 (Seiten 1-10) enthalten.

1.4.1 Darstellung ganzer Zahlen

Eine ganze Zahl $z \in \mathbb{Z}$ wird vorzeichenbehaftet durch

$$\tilde{z} = (z_{L-1} \dots z_0, s) \in \{0,1\}^+ \times \{0,1\}$$

dargestellt, wenn gilt:

1. $z = 0 : \tilde{z} = (0,0)$
2. $z \neq 0$:
   (a) $\tilde{z}$ enthält keine führenden Nullen : $z_{L-1} = 1$
   (b) Wertet man $\tilde{z}$ auf der Basis 2 aus, kommt gerade z heraus :

$$z = (-1)^s \cdot \sum_{i=0}^{L-1} z_i \cdot 2^i$$

1.4.2 Laufzeitaussagen

In einem ersten Schritt ist es erforderlich, sich zu überlegen, welche Gestalt eine Eingabe für einen Algorithmus $\mathcal{A}$ haben kann, also ein Menge

$$\mathrm{E}(\mathcal{A}) := \{e : e \text{ ist erlaubte Eingabe für } \mathcal{A}\}$$

festzulegen. Da die Elemente von $\mathrm{E}(\mathcal{A})$ möglicherweise eine komplexe Struktur besitzen, ordnet man im nächsten Schritt jedem $e \in \mathrm{E}(\mathcal{A})$ eine Größe zu, die als Tupel natürlicher Zahlen gewählt wird :

$$\mathrm{g} : \mathrm{E}(\mathcal{A}) \mapsto \underbrace{\mathbb{N}_0 \times \ldots \times \mathbb{N}_0}_{k} =: \mathbb{N}_0^{1\times k} \quad , \quad k \in \mathbb{N}$$

Die Ausführung eines Programmes für $\mathcal{A}$ in einer Rechenanlage benötigt Ressourcen.
Es bezeichne

$$\mathrm{T}_{\mathcal{A}}^{\mathrm{M}}(e) \in \mathbb{R}_0^+ \cup \{\infty\} \quad , \quad e \in \mathrm{E}(\mathcal{A})$$

den Rechenzeitverbrauch einer Maschine M, die den Algorithmus $\mathcal{A}$ ausführt, und e als Eingabe erhält. Das Verhalten im **schlechtesten Fall** ist dann definiert als

$$\hat{\mathrm{T}}_{\mathcal{A}}^{\mathrm{M}}(n_1,\ldots,n_k) := \sup_{e\in\mathrm{E}(\mathcal{A})} \left\{\mathrm{T}_{\mathcal{A}}^{\mathrm{M}}(e) : \mathrm{g}(e) = (n_1,\ldots,n_k)\right\}$$

wobei $(n_1,\ldots,n_k) \in \mathbb{N}_0^{1\times k}$ stellvertretend für alle erlaubten Eingaben dieser Größe steht. Da die genaue Bestimmung des Supremums schwierig sein kann, ist es oft nur möglich, eine obere Schranke dafür anzugeben.

Definition 1.15 *Es seien Funktionen $\varphi, \psi : \mathbb{N}_0^{1\times k} \mapsto \mathbb{N}_0$ gegeben.*
Das **Landau-Symbol** O *ist erklärt durch*

$$\varphi = O(\psi) \quad \Leftrightarrow \quad \begin{array}{l} \exists c \in \mathbb{R}^+ : \exists (m_1,\ldots,m_k) \in \mathbb{N}_0^{1\times k} \\ \forall (n_1,\ldots,n_k) \in \mathbb{N}_0^{1\times k} \textit{ mit } (n_1,\ldots,n_k) \trianglerighteq (m_1,\ldots,m_k) : \\ \varphi(n_1,\ldots,n_k) \le c\cdot\psi(n_1,\ldots,n_k) \end{array}$$

wobei $\trianglerighteq$ komponentenweise als $\ge$ gelte.

Definition 1.16 (Laufzeitaussagen) *Eine Formulierung*

"Algorithmus $\mathcal{A}$ benötigt für Eingaben aus $\mathrm{E}(\mathcal{A})$ Zeit $O(\varphi(n_1,\ldots,n_k))$"

mit einer Funktion $\varphi : \mathbb{N}_0^{1\times k} \mapsto \mathbb{N}_0$ entspricht der Aussage

$$\hat{\mathrm{T}}_{\mathcal{A}}^{\mathrm{M}}(n_1,\ldots,n_k) = O(\varphi(n_1,\ldots,n_k))$$

Die benutzte Maschine M *sei dabei implizit gegeben.*

1.4.3 Arithmetische Operationen in $\mathbb{Z}$

Die im folgenden vorgestellten Laufzeitannahmen für die

1. Addition und Subtraktion
2. Multiplikation
3. Division mit Rest

ganzer Zahlen beruhen auf den klassischen Verfahren (siehe [Buchm1] , Seiten 3 ff.) für diese Operationen und bilden den Ausgangspunkt für alle Laufzeitanalysen in dieser Arbeit. Als Maß für die Zuordnung von Rechenzeit zu einer Programmzeile verwenden wir die **Bitkomplexität** (siehe [Aho/Hop/Ull] , Seiten 22 ff.) , welche logische Operationen auf einzelnen Bits zählt. Der Bezug zum Zeitbegriff des vorhergehenden Abschnittes wird durch die Forderung sichergestellt, daß jede logische Operation einen konstanten Bedarf an Rechenzeit hat.

Proposition 1.1 (Addition und Subtraktion)
Die Berechnung von $u \pm v$ für $u, v \in \mathbb{Z}$ erfordert Zeit $O(\log(\max\{|u|, |v|\}))$

Proposition 1.2 (Multiplikation)
Um zwei Zahlen $u, v \in \mathbb{Z}$ zu multiplizieren, benötigt man Zeit $O(\log(|u|) \cdot \log(|v|))$

Die Division erfordert eine kleine Vorüberlegung.

Lemma 1.4 (Ganzzahlige Division)
Es seien $u \in \mathbb{Z} \setminus \{0\}$ und $v \in \mathbb{N}$ gegeben. Dann gibt es genau eine Zerlegung

$$u = q \cdot v + r \quad , \quad 0 \leq r < v$$

mit $q \in \mathbb{Z}$ und $r \in \mathbb{N}_0$.

Um darauf aufbauend Funktionen

$$\begin{array}{rcrcl} \text{div} & : & \mathbb{Z} \times \mathbb{N} & \rightarrow & \mathbb{Z} \\ \text{mod} & : & \mathbb{Z} \times \mathbb{N} & \rightarrow & \mathbb{N}_0 \end{array}$$

zu erklären, eignen sich die Fallunterscheidungen

$$u \text{ div } v := \begin{cases} q & , \text{ falls } u \neq 0 \\ 0 & , \text{ falls } u = 0 \end{cases}$$

sowie

$$u \bmod v := \begin{cases} r & , \text{ falls } u \neq 0 \\ 0 & , \text{ falls } u = 0 \end{cases}$$

Proposition 1.3 (Division mit Rest)
Die Berechnung von $(u \text{ div } v) \in \mathbb{Z}$ und $(u \bmod v) \in \mathbb{N}_0$ für $u \in \mathbb{Z}$ und $v \in \mathbb{N}$ kostet Zeit $O(\log(|\frac{u}{v}|) \cdot \log(v))$

1.5 Länge einer Zahl

1.5.1 Problemstellung

Es sei $z \in \mathbb{Z} \setminus \{0\}$ eine beliebige Zahl. Für manche Anwendungen ist es wichtig zu wissen, wieviele Bits eine Darstellung

$$\tilde{z} = (z_{L-1} \ldots z_0, s) \quad , \quad L = \#\mathrm{Bits}(z) = \lfloor \log_2(|z|) \rfloor + 1 \tag{1.1}$$

von z enthält, ohne diese explizit zu kennen. Wir bezeichnen L als die **Länge** von z.

1.5.2 Algorithmus

Zur Bestimmung von L eignet sich die folgende Methode :

$$z = (z_{L-1}, \ldots, z_1, z_0, s) \stackrel{\mathrm{div}\ 2}{\rightarrow} (z_{L-1}, \ldots, z_1, s) \tag{1.2}$$

Eine Division ohne Rest durch 2 entspricht einem Rechtsshift um eine Stelle. Um L zu berechnen wiederholen wir (1.2) einfach solange, bis 0 herauskommt, und zählen dabei mit.

Algorithmus 1.1 (Länge einer Zahl)

Eingabe : $z \in \mathbb{Z} \setminus \{0\}$
Ausgabe : $\#\mathrm{Bits}(z)$

```
(* Zähler i initialisieren *)
i := 0
h := |z|
(* Division durchführen *)
while ( h ≠ 0 )
   do
      h := h div 2
      i := i + 1
   od
(* Die Länge von z steht jetzt in i *)
return (i)
```

1.5.3 Laufzeitanalyse

Lemma 1.5 (Länge einer Zahl)
Die Berechnung von $\#\mathrm{Bits}(z)$ *für ein* $z \in \mathbb{Z} \setminus \{0\}$ *erfordert Zeit* $O(\log^2(|z|))$.

Beweis: Betrachte Algorithmus 1.1
Die While-Schleife wird L mal durchlaufen, wobei ein Durchlauf Zeit (Zeilen (7) und (8))

$$O(\log(h) + \log(i)) \;=\; O(\log(|z|))$$

kostet, also insgesamt

$$O(L \cdot \log(|z|)) \;=\; O(\log^2(|z|)) \qquad \text{(wegen (1.1))}$$

■

Bemerkung 1.1 *Ein kritischer Leser mag einwenden, daß obiges Ergebnis asymptotisch nicht optimal ist, denn eine Implementierung des Datentyps GANZE-ZAHL durch lineare Listen einzelner Bits erlaubt die Bestimmung von #Bits(z) für ein $z \in \mathbb{Z} \setminus \{0\}$ ganz ohne arithmetische Operationen durch ein einfaches Durchlaufen der Liste in Zeit $O(\log(|z|))$. Eine solche Annahme ist nicht unrealistisch, da Programme zur Verarbeitung beliebig großer Zahlen oft ähnliche Listen benutzen. Die Bereitstellung einer Funktion* **length** *bereitet dann keine Schwierigkeiten.*
Daß wir im Rahmen dieser Arbeit dennoch das in Lemma 1.5 formulierte Ergebnis verwenden entspricht der Zielsetzung, unter möglichst allgemeinen Voraussetzungen zu arbeiten, um dadurch weitgehende Unabhängigkeit von speziellen Umgebungen oder Architekturen zu erreichen.

1.6 Schnelle Exponentiation

1.6.1 Problemstellung

Es seien Zahlen $a \in \mathbb{Z} \setminus \{0\}$ und $b \in \mathbb{N}$ gegeben.
Um die b-te Potenz von a zu berechnen, eignen sich die folgenden Umformungen :

$$a^b = \operatorname{sgn}(a)^b \cdot |a|^b \quad \text{mit} \quad \operatorname{sgn}(a) \in \{+1, -1\}$$

und

$$|a|^b = |a|^{(b_{L-1} \cdot 2^{L-1} + \ldots + b_0 \cdot 2^0)} = \prod_{b_i = 1} |a|^{2^i}$$

Zur Bestimmung von a^b reicht es also aus, die Folge

$$|a| \,,\, |a|^2 \,,\, \ldots \,,\, |a|^{2^{L-1}}$$

der fortgesetzten Quadrate von $|a|$ zu bestimmen.

1.6.2 Algorithmus

Algorithmus 1.2 (Schnelle Exponentiation)

Eingabe : $a \in \mathbb{Z} \setminus \{0\}$, $b \in \mathbb{N}$
Ausgabe : $a^b \in \mathbb{Z} \setminus \{0\}$

```
(* Initialisierung *)
erg := 1
exp := b
bas := |a|
(* Vorzeichen bestimmen *)
if ( a > 0 )
   then sg := +1
   else
      if ( ( b mod 2 ) == 1 )
         then sg := - 1
         else  sg := +1
      fi
fi
(* Berechne |a|^b *)
while ( exp > 0 )
   do
      if ( (exp mod 2) == 1 )
         then erg := erg * bas
      fi
      exp := exp div 2
      bas := bas * bas
   od
(* Ergebnis zurückgeben *)
return (sg * erg)
```

1.6.3 Laufzeitanalyse

Lemma 1.6 (Schnelle Exponentiation)
Die Berechnung von a^b für $a \in \mathbb{Z} \setminus \{0\}$ und $b \in \mathbb{N}$ erfordert Zeit $O(b^2 \cdot \log^2(|a|))$.

Beweis: Wir beobachten zunächst für Algorithmus 1.2 die Schranken

$$\mathrm{erg} \leq \mathrm{bas} \qquad , \qquad \mathrm{bas} = |a|^{2^{i-1}} \qquad \text{(vor dem } i\text{-ten Schleifendurchlauf)}$$

und folgern daraus als Laufzeit

$$O\left(\sum_{i=1}^{\log(b)} \log^2(|a|^{2^{i-1}})\right) \;=\; O\left(b^2 \cdot \log^2(|a|)\right)$$

denn die Anzahl der Schleifendurchläufe ist $O(\log(b))$. ∎

1.7 Euklidischer Algorithmus

1.7.1 Problemstellung

Es seien zwei Zahlen $a_1, a_2 \in \mathbb{N}$ mit $a_1 \geq a_2$ gegeben.
Wir wollen den größten gemeinsamen Teiler von a_1 und a_2

$$\mathrm{ggT}(a_1, a_2) \;=\; \max\{n \in \mathbb{N} : n|a_1 \text{ und } n|a_2\}$$

berechnen und verwenden dafür die Iteration

$$a_j \;=\; q_j \cdot a_{j+1} + a_{j+2} \quad , \quad j \in \mathbb{N} \tag{1.3}$$

mit

$$0 \;\leq\; a_{j+2} < a_{j+1} \quad \text{ und } \quad a_{j+1}, a_{j+2} \in \mathbb{N} \quad , \quad q_j \in \mathbb{N}$$

Dies liefert eine monoton fallende Folge $(a_j)_{j \in \mathbb{N}}$, die nach endlich vielen Schritten abbricht. Für den Index $j_0 \in \mathbb{N}$ des letzten von 0 verschiedenen Elementes gilt :

$$a_{j_0} \;=\; \mathrm{ggT}(a_1, a_2)$$

Die Anzahl der Bits in a_1 bzw. a_2 werde mit n_1 bzw. n_2 bezeichnet. Die Iterationstiefe j_0 läßt sich dann (siehe [Buchm1] , Satz 2.9 , Seite 6) abschätzen durch

$$j_0 \;\leq\; 2 \cdot n_2 + 2$$

Um eine Darstellung des $\mathrm{ggT}(a_1, a_2)$ der Art

$$\mathrm{ggT}(a_1, a_2) \;=\; x \cdot a_1 + y \cdot a_2 \quad \text{mit} \quad x, y \in \mathbb{Z}$$

zu erhalten, bietet sich das folgende Schema an : die Ausdrücke

$$\begin{array}{lclcccccl} a_j & = & x_j & \cdot & a_1 & + & y_j & \cdot & a_2 \quad , \quad x_j, y_j \in \mathbb{Z} \\ a_{j+1} & = & x_{j+1} & \cdot & a_1 & + & y_{j+1} & \cdot & a_2 \quad , \quad x_{j+1}, y_{j+1} \in \mathbb{Z} \end{array}$$

seien bereits bekannt. Wir setzen diese in die Iterationsvorschrift (1.3) ein und erhalten

$$a_{j+2} \;=\; a_j - q_j \cdot a_{j+1} \;=\; (x_j - q_j \cdot x_{j+1}) \cdot a_1 + (y_j - q_j \cdot y_{j+1}) \cdot a_2$$

woraus

$$x_{j+2} \;=\; x_j - q_j \cdot x_{j+1} \quad , \quad y_{j+2} \;=\; y_j - q_j \cdot y_{j+1} \tag{1.4}$$

folgt. Als Startwerte sind

$$x_1 := 1 \quad , \quad y_1 := 0$$

und

$$x_2 := 0 \quad , \quad y_2 := 1$$

vorgegeben.

Beispiel 1.1 *Wir wollen mit diesem Verfahren den* ggT *von 143 und 91 bestimmen und erhalten als Divisionsfolge*

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

$$\Rightarrow \mathrm{ggT}(143, 91) = 13$$

Für die Darstellungen der Divisionsreste ergibt sich

$$\begin{aligned} 52 &= 1 \cdot 143 + (-1) \cdot 91 \\ 39 &= (-1) \cdot 143 + 2 \cdot 91 \\ 13 &= 2 \cdot 143 + (-3) \cdot 91 \\ 0 &= (-7) \cdot 143 + 11 \cdot 91 \end{aligned}$$

1.7.2 Koeffizientenabschätzung

Eine für die Laufzeitanalyse wichtige Frage ist, wie groß die Koeffizienten $x_j, y_j \in \mathbb{Z}$ in den Darstellungen der a_j werden können.

Lemma 1.7 (Koeff-EA)
Die Zahlen $x, y \in \mathbb{Z}$ in der vom euklidischen Algorithmus gelieferten Darstellung

$$\mathrm{ggT}\,(a_1, a_2) \;=\; x \cdot a_1 + y \cdot a_2 \quad , \quad a_1, a_2 \in \mathbb{N} \quad \textit{mit} \quad a_1 \geq a_2$$

sowie die Koeffizienten $x_j, y_j \in \mathbb{Z}$ in den Darstellungen der Divisionsreste

$$a_j \;=\; x_j \cdot a_1 + y_j \cdot a_2 \quad , \quad j \in \{1, \ldots, j_0\}$$

genügen den Abschätzungen

$$|x| \; , \; |x_j| \quad \leq \quad \frac{a_2}{2 \cdot \mathrm{ggT}(a_1, a_2)}$$

und

$$|y| \; , \; |y_j| \quad \leq \quad \frac{a_1}{2 \cdot \mathrm{ggT}(a_1, a_2)}$$

Beweis: Die Vorgehensweise gliedert sich in drei Teile :

1. Setze die Division über a_{j_0} hinaus bis a_{j_0+1} fort. Dies liefert eine Darstellung

   $$0 = x_{j_0+1} \cdot a_1 + y_{j_0+1} \cdot a_2$$

   Zeige dann, daß immer

   $$\mathrm{ggT}(x_j, y_j) = 1 \qquad \forall j \in \{1, \ldots, j_0 + 1\}$$

   gilt und folgere daraus

   $$|x_{j_0+1}| = \frac{a_2}{\mathrm{ggT}(a_1, a_2)} \quad , \quad |y_{j_0+1}| = \frac{a_1}{\mathrm{ggT}(a_1, a_2)}$$

2. Betrachte die Beträge der Koeffizienten x_j, y_j und beweise

$$\begin{aligned} |x_j| &\leq |x_{j+1}| \quad \forall j \in \{3, \ldots, j_0\} \\ |y_j| &\leq |y_{j+1}| \quad \forall j \in \{2, \ldots, j_0\} \end{aligned}$$

3. Zeige schließlich

$$|x_{j_0}| \leq \frac{1}{2} \cdot |x_{j_0+1}| \qquad , \qquad |y_{j_0+1}| \leq \frac{1}{2} \cdot |y_{j_0+1}|$$

und folgere die Behauptung des Lemmas aus

$$|x_j| \leq |x_{j_0}| \leq \frac{1}{2} \cdot |x_{j_0+1}| = \frac{a_2}{2 \cdot \mathrm{ggT}(a_1, a_2)} \qquad \forall j \in \{3, \ldots, j_0\}$$

$$|y_j| \leq |y_{j_0}| \leq \frac{1}{2} \cdot |y_{j_0+1}| = \frac{a_1}{2 \cdot \mathrm{ggT}(a_1, a_2)} \qquad \forall j \in \{3, \ldots, j_0\}$$

Nun zu den Einzelheiten.

Ad 1. :

Hilfsbehauptung 1 :
Es gilt

$$x_j \cdot y_{j+1} - y_j \cdot x_{j+1} \in \{+1, -1\} \qquad \forall j \in \{1, \ldots, j_0\}$$

Beweis : Durch Induktion nach dem Index j.
(<u>Induktionsanfang</u> : $j = 1$)
Offenbar erfüllt
(<u>Induktionsschluß</u> : $j \to j + 1$)
Die Behauptung sei für ein $j \in \{1, \ldots, j_0 - 1\}$ wahr :

$$x_j \cdot y_{j+1} - y_j \cdot x_{j+1} \in \{+1, -1\}$$

Dann ist auch

$$x_{j+1} \cdot y_{j+2} - y_{j+1} \cdot x_{j+2} \in \{+1, -1\}$$

(<u>Beweis des Induktionsschrittes</u>)
Nach Konstruktion des Algorithmus gilt (siehe (1.4))

$$\begin{aligned} x_{j+2} &= x_j - q_j \cdot x_{j+1} \\ y_{j+2} &= y_j - q_j \cdot y_{j+1} \end{aligned}$$

woraus durch Einsetzen

$$\begin{aligned} & x_{j+1} \cdot y_{j+2} - y_{j+1} \cdot x_{j+2} \\ = \; & x_{j+1} \cdot y_j - q_j \cdot x_{j+1} \cdot y_{j+1} - y_{j+1} \cdot x_j + q_j \cdot y_{j+1} \cdot x_{j+1} \\ = \; & -(x_j \cdot y_{j+1} - x_{j+1} \cdot y_j) \in \{+1, -1\} \end{aligned}$$

folgt.
□

Die obige Hilfsbehauptung liefert unmittelbar

$$\text{ggT}(x_j, y_j) = 1 \qquad \forall j \in \{1, \dots, j_0 + 1\}$$

denn für beliebige $a, b \in \mathbb{N}$ sind die durch

$$\alpha \cdot a + \beta \cdot b \quad , \quad \alpha, \beta \in \mathbb{Z}$$

dargestellten Zahlen genau die Vielfachen des $\text{ggT}(a, b)$.

Hilfsbehauptung 2 :
Die Beträge der Koeffizienten x_{j_0+1}, y_{j_0+1} erfüllen

$$|x_{j_0+1}| = \frac{a_2}{\text{ggT}(a_1, a_2)} \quad , \quad |y_{j_0+1}| = \frac{a_1}{\text{ggT}(a_1, a_2)}$$

Beweis : Zunächst ist zu bemerken, daß

$$0 = \pm \frac{a_2}{\text{ggT}(a_1, a_2)} \cdot a_1 \mp \frac{a_1}{\text{ggT}(a_1, a_2)} \cdot a_2$$

nur *eine* mögliche Darstellung der 0 durch a_1 und a_2 ist.
Durch Addition geeigneter Vielfacher von $a_1 a_2$ erhalten wir *alle* Kandidaten :

$$0 = \left((1 + \gamma) \cdot \frac{a_2}{\text{ggT}(a_1, a_2)}\right) \cdot a_1 - \left((1 + \gamma) \cdot \frac{a_1}{\text{ggT}(a_1, a_2)}\right) \cdot a_2 \quad , \quad \gamma \in \mathbb{Z}$$

Die Teilerfremdheit von x_{j_0+1} und y_{j_0+1} in der Darstellung

$$0 = x_{j_0+1} \cdot a_1 + y_{j_0+1} \cdot a_2 \quad , \quad x_{j_0+1}, y_{j_0+1} \neq 0$$

läßt für γ dann nur die Wahl $\gamma \in \{0, -2\}$, denn für $\gamma \notin \{0, -2\}$ ist

$$1 + \gamma \neq \pm 1$$

ein nicht-trivialer Teiler von x_{j_0+1} bzw. y_{j_0+1} .
□

Ad 2.

Hilfsbehauptung 3 :
Die Koeffizienten x_j, y_j alternieren im Vorzeichen :

$$\text{sgn}(x_j) = (-1)^{j+1} \qquad \forall j \in \{3, \dots, j_0 + 1\}$$

$$\text{sgn}(y_j) = (-1)^{j} \qquad \forall j \in \{2, \dots, j_0 + 1\}$$

Beweis : Durch Induktion nach dem Index j.
(<u>Induktionsanfang</u> : $j = 2, 3$)
Die Anfangsstücke sind gegeben durch

$$x_1 = 1 \quad , \quad x_2 = 0 \quad , \quad x_3 = 1$$

und

$$y_1 = 0 \quad , \quad y_2 = 1 \quad , \quad y_3 = -q_1$$

Also ist die Behauptung erfüllt.
(<u>Induktionsschluß</u> : $j+1 \to j+2$)
Die Behauptung sei für zwei Indizes j und $j+1$ mit $j \in \{2, \ldots, j_0 - 1\}$ wahr :

$$\operatorname{sgn}(x_j) = (-1)^{j+1} \quad , \quad \operatorname{sgn}(y_j) = (-1)^j$$

$$\operatorname{sgn}(x_{j+1}) = (-1)^{j+2} \quad , \quad \operatorname{sgn}(y_{j+1}) = (-1)^{j+1}$$

Dann gilt auch

$$\operatorname{sgn}(x_{j+2}) = (-1)^{j+3} \quad , \quad \operatorname{sgn}(y_{j+2}) = (-1)^{j+2}$$

(<u>Beweis des Induktionsschrittes</u>)
Wir betrachten nur x_j - die Vorgehensweise für y_j ist analog. Es sind zwei Fälle möglich :
1. Fall : $x_j > 0$ und $x_{j+1} < 0$

$$\Rightarrow x_{j+2} = \underbrace{x_j}_{>0} - \underbrace{q_j}_{>0} \cdot \underbrace{x_{j+1}}_{<0} > 0$$

2. Fall : $x_j < 0$ und $x_{j+1} > 0$

$$\Rightarrow x_{j+2} = \underbrace{x_j}_{<0} - \underbrace{q_j}_{>0} \cdot \underbrace{x_{j+1}}_{>0} < 0$$

Somit ist die Behauptung auch für x_{j+2} wahr.
$\square$

Wir zeigen nun die Monotonie der Beträge.

Hilfsbehauptung 4 :
Für die Beträge der Koeffizienten x_j, y_j gilt

$$\begin{aligned} |x_j| &\leq |x_{j+1}| \quad \forall j \in \{3, \ldots, j_0\} \\ |y_j| &\leq |y_{j+1}| \quad \forall j \in \{2, \ldots, j_0\} \end{aligned}$$

Beweis :
Durch Induktion nach dem Index j - die Iterationstiefe sei o.B.d.A. mindestens 2.
(<u>Induktionsanfang</u> : $j = 2, 3$)
Es ist

$$x_3 = 1 \quad , \quad x_4 = -q_2 \quad \text{und} \quad q_2 \in \mathbb{N}$$

woraus die erste Ungleichung folgt. Mit

$$y_2 = 1 \quad , \quad y_3 = -q_1 \quad \text{und} \quad q_1 \in \mathbb{N}$$

ergibt sich entsprechend die zweite Ungleichung.
(<u>Induktionsschluß</u> : $j \to j+1$)
Die Behauptung sei für ein $j \in \{2, \ldots, j_0 - 1\}$ erfüllt :

$$|x_j| \leq |x_{j+1}| \quad , \quad |y_j| \leq |y_{j+1}|$$

Dann gilt auch

$$|x_{j+1}| \leq |x_{j+2}| \quad , \quad |y_{j+1}| \leq |y_{j+2}|$$

(Beweis des Induktionsschrittes)
Im folgenden betrachten wir nur x_j - die Vorgehensweise für y_j ist analog.
Es sind wieder zwei Fälle zu unterscheiden :
1. Fall : $x_j > 0$, $x_{j+1} < 0$, $x_{j+2} > 0$
Dann ist

$$\begin{aligned} |x_{j+2}| &= x_j - q_j \cdot x_{j+1} \\ |x_{j+1}| &= -x_{j+1} \end{aligned}$$

und mit

$$\begin{aligned} x_j - q_j \cdot x_{j+1} &\geq -x_{j+1} \\ \Leftrightarrow \qquad \underbrace{x_j}_{>0} &\geq \underbrace{(q_j - 1)}_{\geq 0} \cdot \underbrace{x_{j+1}}_{<0} \end{aligned}$$

erhalten wir die Behauptung.
2. Fall : $x_j < 0$, $x_{j+1} > 0$, $x_{j+2} < 0$
Dann ist

$$\begin{aligned} |x_{j+2}| &= -(x_j - q_j \cdot x_{j+1}) \\ |x_{j+1}| &= x_{j+1} \end{aligned}$$

und mit

$$\begin{aligned} -(x_j - q_j \cdot x_{j+1}) &\geq x_{j+1} \\ \Leftrightarrow \qquad x_j - q_j \cdot x_{j+1} &\leq -x_{j+1} \\ \Leftrightarrow \qquad \underbrace{x_j}_{<0} &\leq \underbrace{(q_j - 1)}_{\geq 0} \cdot \underbrace{x_{j+1}}_{>0} \end{aligned}$$

ergibt sich entsprechend die Behauptung.
□

Ad 3.

Hilfsbehauptung 5:
Für die Beträge der Koeffizienten x_{j_0}, x_{j_0+1} und y_{j_0}, y_{j_0+1} gilt

$$|x_{j_0}| \leq \frac{1}{2} \cdot |x_{j_0+1}| \qquad , \qquad |y_{j_0}| \leq \frac{1}{2} \cdot |y_{j_0+1}|$$

Beweis : wir haben zunächst

$$x_{j_0+1} = x_{j_0-1} - q_{j_0-1} \cdot x_{j_0} \qquad \Rightarrow \qquad x_{j_0} = \frac{1}{q_{j_0-1}} \cdot (x_{j_0-1} - x_{j_0+1})$$

sowie

$$y_{j_0+1} = y_{j_0-1} - q_{j_0-1} \cdot y_{j_0} \qquad \Rightarrow \qquad y_{j_0} = \frac{1}{q_{j_0-1}} \cdot (y_{j_0-1} - y_{j_0+1})$$

Die Hilfsbehauptung 3 liefert uns ferner

$$\operatorname{sgn}(x_{j_0-1}) = \operatorname{sgn}(x_{j_0+1}) \qquad , \qquad \operatorname{sgn}(y_{j_0-1}) = \operatorname{sgn}(y_{j_0+1})$$

Daraus leiten wir

$$|x_{j_0}| = \frac{1}{q_{j_0-1}} \cdot |x_{j_0-1} - x_{j_0+1}| \leq \frac{|x_{j_0+1}|}{q_{j_0-1}}$$

sowie

$$|y_{j_0}| = \frac{1}{q_{j_0-1}} \cdot |y_{j_0-1} - y_{j_0+1}| \leq \frac{|y_{j_0+1}|}{q_{j_0-1}}$$

ab, woraus sich wegen $q_{j_0-1} \geq 2$ die Behauptung ergibt.

□

Also folgt insgesamt die Ausage des Lemmas. ■

1.7.3 Algorithmus

Algorithmus 1.3 (Euklidischer Algorithmus)

Eingabe : $a_1, a_2 \in \mathbb{N}$ mit $a_1 \geq a_2$
Ausgabe : $\mathrm{ggT}(a_1, a_2) = x \cdot a_1 + y \cdot a_2 \in \mathbb{N}$ mit $x, y \in \mathbb{Z}$

(* Initialisierung *)
a1 := a_1
a2 := a_2

$$\begin{pmatrix} \mathrm{x}_1 & \mathrm{x}_2 \\ \mathrm{y}_1 & \mathrm{y}_2 \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

rest := a1 mod a2
(* Iteration durchführen *)
while (rest $\neq$ 0)
do

$$\left.\begin{array}{lll} \mathrm{q} & := & \mathrm{a1\ div\ a2} \\ \mathrm{rest} & := & \mathrm{a1\ mod\ a2} \end{array}\right\} \text{(I)}$$

$$\left.\begin{pmatrix} \mathrm{x}_1 & \mathrm{x}_2 \\ \mathrm{y}_1 & \mathrm{y}_2 \end{pmatrix} := \begin{pmatrix} \mathrm{x}_1 & \mathrm{x}_2 \\ \mathrm{y}_1 & \mathrm{y}_2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & -\mathrm{q} \end{pmatrix}\right\} \text{(II)}$$

a1 := a2
a2 := rest
od
(* Ergebnis bestimmen *)
if ((a_1 mod a_2) == 0)
then
ggT := a_2
x := 0
y := 1
else
ggT := a1
x := x_1
y := y_1
fi
return (ggT , x , y)

1.7.4 Laufzeitanalyse

Lemma 1.8 (Euklidischer Algorithmus)
Die Berechnung des größten gemeinsamen Teilers zweier Zahlen $a_1, a_2 \in \mathbb{N}$ mit $a_1 \geq a_2$ durch den euklidischen Algorithmus, sowie einer Darstellung

$$\mathrm{ggT}(a_1, a_2) = x \cdot a_1 + y \cdot a_2$$

mit Zahlen $x, y \in \mathbb{Z}$, erfordert einen Zeitaufwand der Größenordnung $O(\log^2(a_1))$.

Beweis: Betrachte Algorithmus 1.3

1. Die Anzahl der Schleifendurchläufe ist $j_0 - 1$.

2. Kosten für einen Schleifendurchlauf :
Für (I) : Die beiden Divisionen erfordern Zeit

$$O(\log(\frac{a_j}{a_{j+1}}) \cdot \log(a_{j+1}))$$

denn in (I) wird

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

realisiert. Mit

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

ergibt dies einen Zeitaufwand von

$$O(\log(q_j) \cdot \log(a_{j+1}))$$

Für (II) : Aus (Koeff-EA) folgt

$$|x_2|\,,\,|y_2| \ \leq \ a_1$$

d.h. die Multiplikationen kosten jeweils Zeit

$$O(\log(q_j) \cdot \log(a_1))$$

3. Insgesamt :

$$\begin{aligned} & O(\sum_{j=1}^{j_0-1}(\log(q_j) \cdot (\log(a_{j+1}) + \log(a_1)))) \\ = \ & O(\log(a_1) \cdot \log(\prod_{j=1}^{j_0-1} q_j)) \qquad\qquad (\text{wegen } a_{j+1} \leq a_1 \ ) \\ = \ & O(\log(a_1) \cdot \log(\xi)) \end{aligned}$$

für ein $\xi \in \mathbb{N}$ mit

$$\prod_{j=1}^{j_0-1} q_j \ \leq \ \xi$$

Andererseits ist

$$a_1 \geq q_1 \cdot a_2 \geq (q_1 \cdot q_2) \cdot a_3 \geq \ldots \geq (\prod_{j=1}^{j_0-1} q_j) \cdot a_{j_0}$$

woraus

$$\prod_{j=1}^{j_0-1} q_j \leq \frac{a_1}{a_{j_0}} \leq a_1$$

folgt. Die Gesamtlaufzeit ist also $O(\log^2(a_1))$. ■

1.7.5 Übertragung auf ganze Zahlen

Definition 1.17 (lggT , rggT)
Für die Koeffizienten $x, y \in \mathbb{Z}$ in der Darstellung

$$\mathrm{ggT}(a_1, a_2) = x \cdot a_1 + y \cdot a_2$$

seien die Bezeichnungen

$$\mathrm{lggT}(a_1, a_2) := x$$
$$\mathrm{rggT}(a_1, a_2) := y$$

eingeführt.

Um zu Funktionen

$$\begin{array}{lcccc} \mathrm{ggT} & : & \mathbb{Z} \times \mathbb{Z} & \to & \mathbb{N}_0 \\ \mathrm{lggT}\ ,\ \mathrm{rggT} & : & \mathbb{Z} \times \mathbb{Z} & \to & \mathbb{Z} \end{array}$$

zu gelangen, benötigen wir noch die **Regeln**

a) $\mathrm{ggT}(u, v) = \mathrm{ggT}(v, u)$

b) $\mathrm{ggT}(u, v) = \mathrm{ggT}(-u, v)$

c) $\mathrm{ggT}(u, 0) = u$

d) $\mathrm{ggT}(0, 0) = 0$

sowie

e) $\mathrm{lggT}(-u, v) = -\mathrm{lggT}(u, v)$

f) $\mathrm{lggT}(u, -v) = \mathrm{lggT}(u, v)$

g) $\mathrm{rggT}(u, -v) = -\mathrm{rggT}(u, v)$

h) $\mathrm{rggT}(-u, v) = \mathrm{rggT}(u, v)$

wobei $u, v \in \mathbb{Z}$ ganze Zahlen seien.

Bemerkung 1.2 *Die Regel* $\mathrm{lggT}(a_1, a_2) = \mathrm{rggT}(a_2, a_1)$ *gilt nur für $a_1 < a_2$, da sonst die Bedingung $a_1 \geq a_2$ verletzt ist.*

1.8 Mehrfacher ggT

1.8.1 Problemstellung

Wir wollen auf der Grundlage des euklidischen Algorithmus für Zahlen

$$a_1, \ldots, a_n \in \mathbb{Z} \quad , \quad n \geq 3$$

sowohl deren größten gemeinsamen Teiler

$$g \;=\; \mathrm{ggT}(a_1, \ldots, a_n)$$

als auch eine Darstellung

$$g \;=\; x_1 \cdot a_1 + \ldots + x_n \cdot a_n$$

mit möglichst kleinen Koeffizienten

$$x_1, \ldots, x_n \in \mathbb{Z}$$

berechnen. Eine naheliegende Lösung wäre, nacheinander

$$\begin{array}{lclcl} g_1 & = & \mathrm{ggT}(a_1, a_2) & = & y_1 \cdot a_1 + y_2 \cdot a_2 \\ g_2 & = & \mathrm{ggT}(g_1, a_3) & = & \tilde{y}_2 \cdot g_1 + y_3 \cdot a_3 \\ & & \vdots & & \\ g_{n-2} & = & \mathrm{ggT}(g_{n-3}, a_{n-1}) & = & \tilde{y}_{n-2} \cdot g_{n-3} + y_{n-1} \cdot a_{n-1} \\ g_{n-1} & = & \mathrm{ggT}(g_{n-2}, a_n) & = & \tilde{y}_{n-1} \cdot g_{n-2} + y_n \cdot a_n \end{array}$$

zu ermitteln und

$$g \;=\; g_{n-1}$$

bzw.

$$\begin{array}{lcll} x_1 & = & \tilde{y}_{n-1} \cdot \ldots \cdot \tilde{y}_2 \cdot y_1 & \\ x_j & = & \tilde{y}_{n-1} \cdot \ldots \cdot \tilde{y}_j \cdot y_j & \forall j \in \{2, \ldots, n-1\} \\ x_n & = & y_n & \end{array}$$

zu setzen. Der Nachteil dieser Methode ist jedoch, daß die so bestimmten x_j relativ groß werden können : ohne zusätzliche Annahmen über die a_i haben wir mit

$$|a| \;=\; \max\{|a_1|, \ldots, |a_n|\}$$

nur die Schranken

$$\begin{array}{lcll} |x_1| & \leq & (|a|/2)^n & \\ |x_j| & \leq & (|a|/2)^{n-j+1} & \forall j \in \{2, \ldots, n-1\} \\ |x_n| & \leq & (|a|/2) & \end{array}$$

zur Verfügung, welche für unsere Zwecke zu grob sind. Dies liefert die Motivation, nach einem anderen Verfahren zu suchen, welches uns kleinere Koeffizienten garantieren kann.

1.8.2 Berechnungsverfahren

Wir stellen im folgenden einen Algorithmus vor, der bereits 1970 von [Bradley] veröffentlicht wurde, und der geschickt gewisse Eigenschaften Diophantischer Gleichungen ausnutzt.

1.8.2.1 Lineare Diophantische Gleichungen

Definition 1.18 (LDG in zwei Variablen)
Es seien $a, b, c \in \mathbb{Z}$ und $a \cdot b \neq 0$. Dann nennen wir

$$a \cdot x + b \cdot y \; = \; c \tag{1.5}$$

eine lineare Diophantische Gleichung oder kurz LDG in zwei Variablen und notieren durch

$$\mathcal{LD}(a, b, c) := \{(x, y) \in \mathbb{Z}^2 \mid a \cdot x + b \cdot y = c\}$$

die Menge der Lösungen. Wir bezeichnen (1.5) als **konsistent**, *falls es eine Lösung gibt.*

Welche Struktur hat nun $\mathcal{LD}(a, b, c)$?

Lemma 1.9 (Lösungsschar einer LDG in zwei Variablen)
Es sei $a \cdot x + b \cdot y = c$ eine LDG in zwei Variablen. Dann ist

$$\mathcal{LD}(a, b, c) \neq \emptyset \quad \Leftrightarrow \quad \mathrm{ggT}(a, b) \mid c$$

und für jede ganzzahlige Lösung (x_0, y_0) gilt (vgl. [Hav/Maj])

$$\mathcal{LD}(a, b, c) = \left\{ (x, y) \in \mathbb{Z}^2 \;\middle|\; x = x_0 + \frac{b \cdot k}{\mathrm{ggT}(a, b)} \;\; \textit{und} \;\; y = y_0 - \frac{a \cdot k}{\mathrm{ggT}(a, b)} \;\; \textit{mit} \;\; k \in \mathbb{Z} \right\}$$

Wir interessieren uns für Lösungen, deren y-Komponente einen kleinen Absolutbetrag hat, und erläutern in der nächsten Definition, wie wir solche Lösungen berechnen werden.

Definition 1.19 (Minimierungsabbildung ϕ einer LDG in zwei Variablen)
Es sei $a \cdot x + b \cdot y = c$ eine konsistente LDG. Wir notieren durch ϕ_b^a eine Abbildung

$$\phi_b^a \; : \; \mathcal{LD}(a, b, c) \; \rightarrow \; \mathcal{LD}(a, b, c)$$

die wir wie folgt definieren :

$$\phi_b^a(x, y) \; := \; \left(x + \lambda \cdot \frac{b}{\mathrm{ggT}(a, b)} \; , \; y - \lambda \cdot \frac{a}{\mathrm{ggT}(a, b)} \right)$$

mit

$$\lambda \; := \; \left\| \frac{y \cdot \mathrm{ggT}(a, b)}{a} \right\|$$

Dabei bezeichne $\|q\|$ für ein $q \in \mathbb{Q}$ die nächstgelegene ganze Zahl.

1.8.2.2 Balanciertes Rückwärtseinsetzen

In einer ersten Phase bestimmen wir mit dem euklidischen Algorithmus

$$g_i \;:=\; \mathrm{ggT}(a_1, \ldots, a_i) \;=\; \mathrm{ggT}(g_{i-1}, a_i) \qquad \forall i \in \{2, \ldots, n\}$$

sowie jeweils eine Darstellung

$$g_i \;=\; y_i \cdot g_{i-1} + z_i \cdot a_i$$

und versuchen nun, möglichst kleine $x_i \in \mathbb{Z}$ mit

$$g_n \;=\; x_1 \cdot a_1 + \ldots + x_n \cdot a_n$$

zu konstruieren. Zunächst beobachten wir

$$g_n \;=\; y_n \cdot g_{n-1} + z_n \cdot a_n \tag{1.6}$$

weshalb wir einfach $x_n = z_n$ wählen. Dann ersetzen wir g_{n-1} in (1.6) durch

$$g_{n-1} \;=\; y_{n-1} \cdot g_{n-2} + z_{n-1} \cdot a_{n-1}$$

und erhalten

$$y_n y_{n-1} \cdot g_{n-2} + y_n z_{n-1} \cdot a_{n-1} \;=\; g_n - x_n a_n \tag{1.7}$$

Dies ist gerade eine LDG im Sinne des letzten Abschnittes, wobei wir die Koeffizienten von $g_{n-2} \neq 0$ und $a_{n-1} \neq 0$ als variabel ansehen. Wir wenden also auf (1.7) unsere Minimierungsabbildung ϕ an

$$(\hat{y}_{n-1}, x_{n-1}) \;:=\; \phi_{a_{n-1}}^{g_{n-2}}(y_n y_{n-1}, y_n z_{n-1})$$

und haben als neue Gleichung

$$\hat{y}_{n-1} y_{n-2} \cdot g_{n-3} + \hat{y}_{n-1} z_{n-2} \cdot a_{n-2} \;=\; g_n - \sum_{i=n-1}^{n} x_i a_i$$

Der Iterationsschritt hat für ein $j \in \{1, \ldots, n-2\}$ somit allgemein die Form

$$\hat{y}_{n-j+1} y_{n-j} \cdot g_{n-j-1} + \hat{y}_{n-j+1} z_{n-j} \cdot a_{n-j} \;=\; g_n - \sum_{i=n-j+1}^{n} x_i a_i$$

$$\Downarrow \quad \phi_{a_{n-j}}^{g_{n-j-1}} \text{ anwenden}$$

$$\hat{y}_{n-j} \cdot g_{n-j-1} + x_{n-j} \cdot a_{n-j} \;=\; g_n - \sum_{i=n-j+1}^{n} x_i a_i$$

$$\Downarrow \quad \text{expandieren von } g_{n-j-1}$$

$$\hat{y}_{n-j} y_{n-j-1} \cdot g_{n-j-2} + \hat{y}_{n-j} z_{n-j-1} \cdot a_{n-j-1} \;=\; g_n - \sum_{i=n-j}^{n} x_i a_i$$

woraus sich als **Berechnungsvorschrift**

$$(\hat{y}_{n-j}, x_{n-j}) \;:=\; \phi_{a_{n-j}}^{g_{n-j-1}}(\hat{y}_{n-j+1} y_{n-j}, \hat{y}_{n-j+1} z_{n-j})$$

bzw.

$$\begin{aligned}\hat{y}_{n-j} &:= \hat{y}_{n-j+1}y_{n-j} + \left\| \frac{\hat{y}_{n-j+1}z_{n-j}}{g_{n-j-1}/g_{n-j}} \right\| \cdot \frac{a_{n-j}}{g_{n-j}} \\ x_{n-j} &:= \hat{y}_{n-j+1}z_{n-j} - \left\| \frac{\hat{y}_{n-j+1}z_{n-j}}{g_{n-j-1}/g_{n-j}} \right\| \cdot \frac{g_{n-j-1}}{g_{n-j}}\end{aligned}$$

sowie

$$x_1 := \hat{y}_2 \qquad , \qquad x_n := z_n \qquad , \qquad \hat{y}_n := y_n$$

ergibt.

1.8.2.3 Koeffizientenabschätzung

Lemma 1.10 (Koeffizientenabschätzung für balanciertes Rückwärtseinsetzen) *Es seien $a_1, \ldots, a_n \in \mathbb{N}$ und $x_1, \ldots, x_n \in \mathbb{Z}$ Koeffizienten mit*

$$\mathrm{ggT}(a_1, \ldots, a_n) = x_1 a_1 + \ldots + x_n a_n$$

die wir durch balanciertes Rückwärtseinsetzen berechnet haben. Dann gilt

$$|x_1| \le \frac{1}{a_1}\left(g_n + \sum_{i=2}^{n} \frac{a_i g_{i-1}}{2g_i}\right)$$

und

$$|x_i| \le \frac{g_{i-1}}{2g_i} \qquad \forall i \in \{2, \ldots, n\}$$

Beweis: wir haben zunächst

$$\left| \alpha - \left\| \frac{\alpha}{\beta} \right\| \cdot \beta \right| \le \frac{|\beta|}{2} \qquad \forall \alpha, \beta \in \mathbb{Z}$$

woraus mit

$$\alpha = \hat{y}_{i+1}z_i \qquad , \qquad \beta = \frac{g_{i-1}}{g_i}$$

die Ungleichung für $|x_i|$ folgt. Ferner gilt

$$\hat{y}_2 \cdot g_1 + x_2 \cdot a_2 = g_n - \sum_{i=3}^{n} x_i a_i$$

bzw.

$$\hat{y}_2 = \frac{1}{a_1}\left(g_n - \sum_{i=2}^{n} x_i a_i\right)$$

und durch Einsetzen der Schranke für $|x_i|$ erhalten wir

$$|x_1| = |\hat{y}_2| \le \frac{1}{a_1}\left(g_n + \sum_{i=2}^{n} |x_i| a_i\right) \le \frac{1}{a_1}\left(g_n + \sum_{i=2}^{n} \frac{a_i g_{i-1}}{2g_i}\right)$$

■

Bemerkung 1.3 *Die im vorhergehenden Lemma 1.10 konstruierte Schranke*

$$S \;=\; \frac{1}{a_1}\left(g_n + \sum_{i=2}^{n}\frac{a_i g_{i-1}}{2g_i}\right) \;\le\; \max a_i \;+\; \frac{n}{2}\,(\max a_i)^2$$

für $|x_1|$ *mag auf den ersten Blick wenig vielversprechend aussehen. Beachten wir aber*

$$a_1 \;=\; g_1 \;\ge\; g_2 \;\ge\; \ldots \;\ge\; g_n \;\ge\; 1$$

und nehmen einmal an, daß $\mathrm{ggT}(a_1, a_2) \;=\; 1$ *gilt, erhalten wir die bessere Abschätzung*

$$S \;\le\; \frac{1}{a_1}\left(1 + \frac{1}{2}a_1 a_2 + \frac{1}{2}\sum_{i=3}^{n} a_i\right) \;\le\; 1 + \frac{1}{2}\sum_{i=2}^{n} a_i \tag{1.8}$$

Ein Pessimist wird einwenden, daß (1.8) nur selten greift, aber bereits 1849 konnte Dirichlet beweisen, daß zwei zufällig ausgewählte natürliche Zahlen mit einer Wahrscheinlichkeit von $6/\pi^2$ *teilerfremd sind (siehe [Knuth] , Seite 324), so daß wir die zweite Ungleichung sogar als aussagekräftiger betrachten dürfen.*

1.8.3 Algorithmus

Algorithmus 1.4 (Mehrfacher ggT)

Eingabe : $a_1, \ldots, a_n \in \mathbb{N}$
Ausgabe : $\mathrm{ggT}(a_1, \ldots, a_n) \;=\; x_1 a_1 + \ldots + x_n a_n$ mit $x_i \in \mathbb{Z}$

Vorgehen :

```
HAUPTPROGRAMM :
(* Vorberechnungsphase *)
g[1] := a_1
for ( i := 2 ; i ≤ n ; i := i + 1 )
   do
      g[i] :=  ggT ( g[i-1] , a_i )
      y[i] := lggT ( g[i-1] , a_i )
      z[i] := rggT ( g[i-1] , a_i )
   od
(* Balanciertes Rückwärtseinsetzen *)
x[n] := z[n]
ŷ[n] := y[n]
for ( i := n - 1 ; i ≥ 2 ; i := i - 1 )
   do
      G[i]:= g[i-1] div g[i]
      S[i] := nearest ( ŷ[i+1]*z[i] , G[i] )
      ŷ[i] := ŷ[i+1]*y[i] + S[i]*(a_i div g[i])
      x[i] := ŷ[i+1]*z[i] - S[i]*G[i]
   od
x[1] := ŷ[2]
(* Ergebnis zurückgeben *)
return ( g[n] , x[1] , . . . , x[n] )
```

BESTIMME DIE NÄCHSTGELEGENE GANZE ZAHL FÜR $\frac{u}{v} \in \mathbb{Q}$:

```
procedure nearest
    (integer : u ; positive integer : v )
rest := |u| mod v
if ( 2*rest > v )
    then z := ( |u| div v ) + 1
    else  z :=   |u| div v
fi
if ( u < 0 ) then z := -z fi
return (z)
```

1.8.4 Laufzeitanalyse

Lemma 1.11 (Mehrfacher ggT mit Darstellung)
Um für $a_1, \ldots, a_n \in \mathbb{N}$ durch balanciertes Rückwärtseinsetzen eine Darstellung

$$\mathrm{ggT}(a_1, \ldots, a_n) \;=\; x_1 a_1 + \ldots + x_n a_n \qquad , \qquad x_i \in \mathbb{Z}$$

zu berechnen, benötigen wir mit

$$|a| \;=\; \max\{a_1, \ldots, a_n\}$$

einen Zeitaufwand der Größenordnung $O(n \cdot \log(n|a|) \cdot \log(|a|))$.

Beweis: betrachte Algorithmus 1.4 : wir folgern aus

$$\hat{y}_i \cdot g_{i-1} + x_i \cdot a_i \;=\; g_n - \sum_{k=i+1}^{n} x_i a_i \qquad \forall i \in \{2, \ldots, n-1\}$$

bzw.

$$\hat{y}_i \;=\; \frac{1}{g_{i-1}} \left(g_n - \sum_{k=i}^{n} x_k a_k \right) \qquad \forall i \in \{2, \ldots, n-1\}$$

die obere Schranke

$$|\hat{y}_i| \;\leq\; \frac{1}{g_{i-1}} \left(g_n + \sum_{k=i}^{n} |x_k| a_k \right) \;\leq\; \frac{1}{g_{i-1}} \left(g_n + \sum_{k=i}^{n} \frac{a_k g_{k-1}}{2 g_k} \right) \qquad \forall i \in \{2, \ldots, n-1\}$$

welche wir weiter durch

$$|\hat{y}_i| \;\leq\; |a| \;+\; (n - i + 1) \cdot \frac{|a|^2}{2} \tag{1.9}$$

abschätzen (vgl. Bemerkung 1.3), so daß wir einen Zeitaufwand von

$$O(\log(n|a|^2 \cdot |a|) \cdot \log(|a|))$$

für *einen* Aufruf von nearest erhalten. ■

Wir geben noch eine Aussage für den Fall an, daß wir nur den ggT berechnen wollen.

Korollar 1.1 (Mehrfacher ggT ohne Darstellung)
Um für $a_1, \ldots, a_n \in \mathbb{N}$ den $\mathrm{ggT}(a_1, \ldots, a_n)$ zu bestimmen, benötigen wir mit

$$|a| \;=\; \max\{a_1, \ldots, a_n\}$$

einen Zeitaufwand der Größenordnung $O(n \cdot \log^2(|a|))$.

1.9 Chinesischer Restsatz

1.9.1 Problemstellung

Gegeben seien paarweise teilerfremde Zahlen $m_1, \ldots, m_k \in \mathbb{N}$, sowie $x_1, \ldots, x_k \in \mathbb{Z}$.

Definition 1.20 (Kongruenzbegriff in $\mathbb{Z}$)
Eine Zahl $x \in \mathbb{Z}$ heißt **kongruent** *zu $y \in \mathbb{Z}$ modulo $u \in \mathbb{N}$, wenn $u \,|\, x - y$ gilt. Wir schreiben dann*

$$x \equiv y \mod u$$

Welche Struktur hat die Menge

$$X := \{x \in \mathbb{Z} \,:\, x \equiv x_i \mod m_i \quad \forall i \in \{1, \ldots, k\}\}$$

und wie kann effizient ein geeignetes Element x_0 aus dieser Menge berechnet werden ?

1.9.2 Berechnungsverfahren

Aufbau der Menge X :
Es seien $x, y \in X$ und $M := m_1 \cdot \ldots \cdot m_k$. Dann gilt :

$$\begin{aligned}
& x \equiv x_i \mod m_i \;\wedge\; y \equiv x_i \mod m_i && \forall i \in \{1, \ldots, k\} \\
\Rightarrow \quad & x \equiv y \mod m_i && \forall i \in \{1, \ldots, k\} \\
\Rightarrow \quad & m_i \,|\, (x - y) && \forall i \in \{1, \ldots, k\} \\
\Rightarrow \quad & M = m_1 \cdot \ldots \cdot m_k \,|\, (x - y) \quad \text{(da die } m_i \text{ paarweise teilerfremd sind)}
\end{aligned}$$

Die Differenz zweier Elemente aus X ist also immer ein ganzzahliges Vielfaches von M. Sei nun umgekehrt $k \in \mathbb{Z}$ beliebig und $x \in X$. Dann ist

$$\begin{aligned}
& x + k \cdot M \equiv x_i \mod m_i \quad \forall i \in \{1, \ldots, k\} \\
\Rightarrow \quad & x + k \cdot M \in X
\end{aligned}$$

Somit folgt : die Menge X läßt sich darstellen als

$$X = x_0 + M \cdot \mathbb{Z}$$

wobei $x_0 \in X$ ein bereits bekanntes Element ist.

Bestimmung einer speziellen Lösung $x_0 \in X$:
Es sei $M_i := M/m_i$ für $i \in \{1, \ldots, k\}$. Da die m_i paarweise teilerfremd sind, folgt :

$$\mathrm{ggT}(M_i, m_i) = 1 \quad \forall i \in \{1, \ldots, k\}$$

Es gibt also Zahlen $G_i, g_i \in \mathbb{Z}$, so daß

$$1 = G_i \cdot M_i + g_i \cdot m_i \quad \forall i \in \{1, \ldots, k\} \tag{1.10}$$

gilt. Somit ist

$$G_i \cdot M_i \equiv \begin{cases} 0 \mod m_j & , \quad j \neq i \\ 1 \mod m_i \end{cases}$$

Definiere nun

$$e_i := G_i \cdot M_i \quad \Rightarrow \quad |e_i| \leq M \quad \forall i \in \{1, \ldots, k\}$$

denn aus (Koeff-EA) folgt

$$|G_i| \leq m_i \quad \Rightarrow \quad |G_i \cdot M_i| \leq m_i \cdot M_i = M$$

Ferner sei

$$x_0 := \left(\sum_{i=1}^{k} x_i \cdot e_i \right) \mod M$$

Nach Konstruktion ist $e_i = G_i \cdot M_i$ und somit

$$e_i \equiv \delta_{i,j} \mod m_j \quad \forall j \in \{1, \ldots, k\}$$

wobei δ das Kronecker-Symbol bezeichne. Dies bedeutet

$$x_0 \equiv x_i \mod m_i \quad \forall i \in \{1, \ldots, k\}$$

denn gemäß Definition von x_0 gilt (mit $p \in \mathbb{Z}$)

$$x_0 = \left(\sum_{j=1}^{k} x_j \cdot e_j \right) + p \cdot M \equiv \left(\sum_{j=1}^{k} x_j \cdot e_j \right) \mod m_i \quad \forall i \in \{1, \ldots, k\}$$

bzw.

$$x_0 \equiv \left(\sum_{j=1}^{k} x_j \cdot \delta_{j,i} \right) \mod m_i \quad \forall i \in \{1, \ldots, k\}$$

Also folgt : $x_0 \in X$.

Berechnung der Koeffizienten G_i :
Ein naheliegender Ansatz zur Konstruktion einer Darstellung (vgl. (1.10))

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

besteht darin, unmittelbar den euklidischen Algorithmus auf M_i und m_i anzuwenden. Falls M_i wesentlich größer ist als m_i , ergibt dies einen Zeitbedarf von $O(\log^2(M_i))$.

Wir verbessern diese Laufzeit, indem wir etwas anders vorgehen :

1. Zuerst dividieren wir M_i ganzzahlig durch m_i und erhalten

$$M_i = q \cdot m_i + r \quad , \quad q \in \mathbb{Z} \quad \text{und} \quad 0 \leq r < m_i \tag{1.11}$$

2. Mit dem euklidischen Algorithmus ermitteln wir

$$1 = a \cdot m_i + b \cdot r \quad , \quad a, b \in \mathbb{Z}$$

und setzen (1.11) in diese Darstellung ein :

$$1 = b \cdot M_i + (a - b \cdot q) \cdot m_i$$

Also ist

$$G_i = b = \text{rggT}(m_i, r)$$

1.9.3 Algorithmus für die einfache Anwendung

Es sei ein symmetrisches Intervall $I \subseteq \mathbb{Z}$

$$I = [-\lambda\,;+\lambda] \quad , \quad \lambda \in \mathbb{N}$$

gegeben, das maximal M Elemente hat :

$$\lambda \;\leq\; \left\lfloor \frac{M-1}{2} \right\rfloor$$

Dann gibt es höchstens ein $x \in X \cap I$ (vgl. 1.9.2). Wir verwenden die zuvor beschriebene Konstruktion einer speziellen Lösung $x_0 \in X$ und beantworten die Frage, ob ein $x \in X \cap I$ existiert und gegebenenfalls welchen Wert es hat, wegen

$$x_0 \in [0\,, M-1]$$

durch die folgende Fallunterscheidung :

1. $x_0 \in [-\lambda\,;+\lambda] : x = x_0$
2. $x_0 \notin [-\lambda\,;+\lambda] :$

   2.1 $x_0 - M \in [-\lambda\,;+\lambda] : x = x_0 - M$

   2.2 $x_0 - M \notin [-\lambda\,;+\lambda] : X \cap I = \emptyset$

Beispiel 1.2 *Angenommen, wir wollen* $9 \cdot 4$ *berechnen und wissen, daß*

$$9 \cdot 4 \in [-100\,;+100]$$

ist. Als Moduln seien

$$m_1 \;=\; 5 \quad , \quad m_2 \;=\; 7 \quad , \quad m_3 \;=\; 11$$

vorgegeben. Die Intervallgrenze $\lambda = 100$ *erfüllt die Bedingung*

$$\lambda \;\leq\; \left\lfloor \frac{5 \cdot 7 \cdot 11}{2} \right\rfloor$$

und garantiert somit die Korrektheit des Ergebnisses. Im ersten Schritt rechnen wir aus

$$\begin{array}{lllllll} 9 \cdot 4 & \equiv & 4 \cdot 4 & \equiv & 1 & \bmod & 5 \\ 9 \cdot 4 & \equiv & 2 \cdot 4 & \equiv & 1 & \bmod & 7 \\ 9 \cdot 4 & \equiv & 3 & & & \bmod & 11 \end{array}$$

und setzen

$$x_1 \;:=\; 1 \quad , \quad x_2 \;:=\; 1 \quad , \quad x_3 \;:=\; 3$$

sowie

$$M_1 \;:=\; 77 \quad , \quad M_2 \;:=\; 55 \quad , \quad M_3 \;:=\; 35 \quad , \quad M \;:=\; 385$$

Für die Darstellungen der ggT's erhalten wir

$$\begin{array}{lllllll} 1 & = & (-2) \cdot 77 & + & 31 \cdot 5 & \Rightarrow & G_1 = -2 \\ 1 & = & (-1) \cdot 55 & + & 8 \cdot 7 & \Rightarrow & G_2 = -1 \\ 1 & = & (-5) \cdot 35 & + & 16 \cdot 11 & \Rightarrow & G_3 = -5 \end{array}$$

woraus sich

$$e_1 = 231 \quad , \quad e_2 = 330 \quad , \quad e_3 = 210$$

ergibt. Die spezielle Lösung x_0 *lautet also*

$$x_0 = (1 \cdot 231 + 1 \cdot 330 + 3 \cdot 210) \bmod 385 = 36$$

Algorithmus 1.5 (Einfache Anwendung des chinesischen Restsatzes)

Eingabe :

1. Natürliche Zahlen $m_1, \ldots, m_k \in \mathbb{N}$, die paarweise teilerfremd sind : $\mathrm{ggT}(m_i, m_j) = 1 \quad \forall i, j \in \{1, \ldots, k\}$, $i \neq j$
2. Ganze Zahlen $x_1, \ldots, x_k \in \mathbb{Z}$ mit $|x_i| \leq m_i \quad \forall i \in \{1, \ldots, k\}$
3. Intervallgrenze $\lambda \in \mathbb{N}$ mit $\lambda \leq \left\lfloor \frac{1}{2}(M-1) \right\rfloor$, wobei $M := m_1 \cdot \ldots \cdot m_k$

Ausgabe :

1. $x \in [-\lambda\,; +\lambda]$ mit $x \equiv x_i \mod m_i \quad \forall i \in \{1, \ldots, k\}$
2. *No solution* , falls kein solches x existiert

```
(* Berechne M *)
M := 1
for ( i := 1 ; i ≤ k ; i := i + 1 )
   do
      M := M * m[i]
   od
(* Berechne die e[i] *)
for ( i := 1 ; i ≤ k ; i := i + 1 )
   do
      M[i] := M div m[i]
      r    := M[i] mod m[i]
      G[i] := rggT( m[i] , r )
      e[i] := G[i] ∗M[i]
   od
(* Spezielle Lösung bestimmen *)
X := 0
for ( i := 1 ; i ≤ k ; i := i + 1 )
   do
      X := X + e[i]∗x[i]
   od
X := X mod M
(* Ergebnis ermitteln *)
if ( −λ ≤ X ≤ +λ )
   then return (X)
   else
      if ( −λ ≤ X-M ≤ +λ )
```

```
(27)        then return (X-M)
(28)        else return (No solution)
(29)     fi
(30) fi
```

1.9.4 Laufzeit der einfachen Anwendung

Lemma 1.12 (Einfache Anwendung des chinesischen Restsatzes)
Es seien die Voraussetzungen des Abschnittes 1.9.3 gegeben. Dann erfordert die Berechnung einer Lösung $x \in X \cap I$ bzw. das Erkennen von $X \cap I = \emptyset$ unter den Voraussetzungen

1. $|x_i| \;\leq\; m_i \;\leq\; \alpha \quad \forall i \in \{1, \ldots, k\} \quad \text{und} \quad \alpha \in \mathbb{N}$
2. $\lambda \;\leq\; \left\lfloor \frac{1}{2}(M-1) \right\rfloor \quad \text{mit} \quad \lambda \in \mathbb{N} \quad \text{und} \quad M := m_1 \cdot \ldots \cdot m_k$

einen Zeitaufwand der Größenordnung $O(k^2 \cdot \log^2(\alpha))$.

Beweis: betrachte Algorithmus 1.5

1. Bestimmung von M : Zeilen (1)-(6)
   Wegen $M \;\leq\; \alpha^k$ kostet dies Zeit
   $$O(k \cdot \log(\alpha^k) \cdot \log(\alpha)) \;=\; O(k^2 \cdot \log^2(\alpha))$$
2. Berechnung der $e[i]$: Zeilen (8)-(14)
   Aus (Koeff-EA) folgt
   $$|G[i]| \;\leq\; m[i] \;\leq\; \alpha$$
   weshalb ein Durchlauf der for-Schleife in den Zeilen (8)-(14) Zeit
   $$O(\log(M) \cdot \log(m[i]) + \log^2(m[i])) \;=\; O(k \cdot \log^2(\alpha))$$
   erfordert.
3. Bestimmung des Ergebnisses : Zeilen (15)-(30)
   Es gelten die Abschätzungen
   $$|e[i] * x[i]| \;\leq\; \alpha^{k+1}$$
   und
   $$|X| \;\leq\; \sum_{i=1}^{k} |e[i] \cdot x[i]| \;\leq\; k \cdot \alpha^{k+1}$$
   d.h. die Bestimmung von X kostet Zeit
   $$\begin{aligned} & O(k \cdot (\log(\alpha^k) \cdot \log(\alpha) + \log(k \cdot \alpha^{k+1}))) \\ = \;\; & O(k^2 \cdot \log^2(\alpha)) \end{aligned}$$
   Die anschließende Division X mod M erfordert dann noch Zeit
   $$O(\log(k \cdot \alpha^{k+1}) \cdot \log(\alpha^k)) = O(k^2 \cdot \log^2(\alpha))$$ ■

1.9.5 Algorithmus für die wiederholte Anwendung

In manchen Anwendungen tritt der Fall auf, daß für ein festes System von Moduln

$$m_1, \ldots, m_k \in \mathbb{N}$$

und $\beta \in \mathbb{N}$ Listen von ganzen Zahlen

$$\left(x_1^{(1)}, \ldots, x_k^{(1)}\right) \quad \text{bis} \quad \left(x_1^{(\beta)}, \ldots, x_k^{(\beta)}\right)$$

verschiedene Repräsentanten

$$x^{(r)} \in [-\lambda \,;\, +\lambda] \quad , \quad r \in \{1, \ldots, \beta\}$$

bzw. *No solution* berechnet werden müssen. Dabei soll wie zuvor

$$x^{(r)} \equiv x_i^{(r)} \bmod m_i \quad \forall i \in \{1, \ldots, k\} \quad \text{und} \quad r \in \{1, \ldots, \beta\}$$

gelten.

Algorithmus 1.6 (Wiederholte Anwendung des Chinesischen Restsatzes)

Eingabe :

1. Paarweise teilerfremde Zahlen $m_1, \ldots, m_k \in \mathbb{N}$
2. Listen $\left(x_1^{(1)}, \ldots, x_k^{(1)}\right)$ bis $\left(x_1^{(\beta)}, \ldots, x_k^{(\beta)}\right)$ ganzer Zahlen mit $\beta \in \mathbb{N}$ und

$$|x_i^{(j)}| \leq m_i \quad \forall i \in \{1, \ldots, k\} \quad \forall j \in \{1, \ldots, \beta\}$$

3. Intervallgrenze $\lambda \in \mathbb{N}$ mit $\lambda \leq \left\lfloor \frac{1}{2}(M-1) \right\rfloor$ wobei $M := m_1 \cdot \ldots \cdot m_k$

Ausgabe :

1. Lösungen $x^{(1)}, \ldots, x^{(\beta)} \in [-\lambda \,;\, +\lambda]$ mit

$$x^{(j)} \equiv x_i^{(j)} \bmod m_i \quad \forall i \in \{1, \ldots, k\} \quad \forall j \in \{1, \ldots, \beta\}$$

2. Jeweils *No solution* , falls ein $x^{(j)}$ nicht existiert

Vorgehen :

1. Berechne das System $e_1, \ldots, e_k \in \mathbb{Z}$ von Idempotenten :
   führe die Zeilen (1)-(14) von Algorithmus 1.5 aus
2. Ermittele mit den $e_1, \ldots, e_k$ die gesuchten Zahlen $x^{(1)}, \ldots, x^{(\beta)}$:

```
for ( j := 1 ; j ≤ β ; j := j + 1 )
  do
    for ( i := 1 ; i ≤ k ; i := i + 1 )
      do
        x[i] := x_i^(j)
      od
    Durchlaufe die Zeilen (15)-(30) von Algorithmus 1.5
  od
```

1.9.6 Laufzeit der wiederholten Anwendung

Lemma 1.13 (Wiederholte Anwendung des Chinesischen Restsaztes)
Es seien paarweise teilerfremde Zahlen $m_1, \ldots, m_k \in \mathbb{N}$ *mit* $m_i \leq \alpha \in \mathbb{N}$ *, sowie eine Intervallgrenze* $\lambda \in \mathbb{N}$

$$\lambda \;\leq\; \left\lfloor \frac{1}{2}(M-1) \right\rfloor$$

gegeben, wobei $M := m_1 \cdot \ldots \cdot m_k$ *gelte. Ferner seien* $\beta \in \mathbb{N}$ *Listen von Zahlen*

$$x_1^{(r)}, \ldots, x_k^{(r)} \in \mathbb{Z} \quad , \quad r \in \{1, \ldots, \beta\}$$

mit

$$|x_i^{(r)}| \;\leq\; m_i \quad \forall i \in \{1, \ldots, k\} \quad \forall r \in \{1, \ldots, \beta\}$$

bereits bekannt. Dann kostet die Berechnung von Zahlen

$$x^{(r)} \in [-\lambda\,; +\lambda] \quad , \quad r \in \{1, \ldots, \beta\}$$

mit

$$x^{(r)} \equiv x_i^{(r)} \bmod m_i \quad \forall i \in \{1, \ldots, k\} \quad \forall r \in \{1, \ldots, \beta\}$$

bzw. von No solution, falls ein $x^{(r)}$ *nicht existiert, einen Zeitaufwand der Ordnung*

$$O(\beta \cdot k^2 \cdot \log^2(\alpha))$$

Beweis: betrachte Algorithmus 1.6

1. Die Bestimmung der e_i verursacht einmalig einen Aufwand von $O(k^2 \cdot \log^2(\alpha))$
2. Jedes $x^{(r)}$ kostet Zeit $O(k^2 \cdot \log^2(\alpha))$, also insgesamt $O(\beta \cdot k^2 \cdot \log^2(\alpha))$ ■

1.10 Kompakte Primzahllisten

1.10.1 Problemstellung

Gegeben seien $C, m \in \mathbb{N}$ mit $C \geq 2$. Wir suchen eine Liste $p_1, \ldots, p_k \in \mathbb{N}$ von Primzahlen, welche die Bedingungen

$$\prod_{i=1}^{k} p_i \;>\; C \qquad \text{und} \qquad p_i \nmid m \quad \forall i \in \{1, \ldots, k\}$$

erfüllen. Um die p_i möglichst klein zu halten, betrachten wir mit $2, 3, \ldots$ beginnend nacheinander alle Primzahlen bis zu einer geeigneten Schranke und verwerfen nur Primteiler von m. Wieviele Primzahlen benötigen wir und wie groß können diese maximal werden ?

1.10.2 Aussagen über Primzahlen

Für ein $n \in \mathbb{N}$ bezeichne $\vartheta(n)$ die Summe

$$\vartheta(n) := \sum_{\substack{p \in \mathbb{P} \\ p \leq n}} \log_2(p)$$

Nach [Rosser/Schoe] (Korrollar 3.16 , Seite 70) erfüllt ϑ die Abschätzung

$$\vartheta(n) \quad > \quad \frac{n}{1 - \frac{1}{\log_2(n)}} \quad \geq \quad \frac{n}{2} \qquad \forall n \geq 43 \tag{1.12}$$

Definiere ferner für ein $n \in \mathbb{N}$:

$$\pi(n) := \#\{p \in \mathbb{P} : p \leq n\} \quad \text{(Mächtigkeit)}$$

Dann ist (siehe [Cor/Leis/Riv] , Seite 837)

$$\lim_{n \to \infty} \left(\frac{\pi(n) \cdot \ln(n)}{n} \right) \;=\; 1$$

und somit

$$\pi(n) \;=\; O\left(\frac{n}{\log(n)} \right) \tag{1.13}$$

1.10.3 Konstruktion einer oberen Schranke

Wie lassen sich diese Aussagen nun kombinieren, um eine obere Schranke für die gesuchten Primzahlen zu bestimmen ?

Lemma 1.14 (Primzahlschranke)
Für $m, C \in \mathbb{N}$ mit $C \geq 2$ sei $B \in \mathbb{N}$ definiert als

$$B := \max\{43 \,, \lceil 2 \cdot \log_2(m \cdot C) \rceil\}$$

Dann gilt die Ungleichung

$$\prod_{\substack{p \in \mathbb{P},\, p \leq B \\ p \nmid m}} p \quad > \quad C$$

Beweis: Die untere Schranke (1.12) liefert

$$\vartheta(B) \quad > \quad \frac{\max\{43 \,, \lceil 2 \cdot \log_2(m \cdot C) \rceil\}}{2} \quad \geq \quad \max\{\frac{43}{2} \,, \log_2(m \cdot C)\}$$

bzw.

$$2^{\vartheta(B)} \quad > \quad \max\{\sqrt{2} \cdot 2^{21}, m \cdot C\}$$

Nach Definition von ϑ ist

$$2^{\vartheta(B)} \quad = \quad \prod_{\substack{p \in \mathbb{P} \\ p \leq B}} p \tag{1.14}$$

Weiter sei

$$\mathbb{P}_m \;:=\; \{p \in \mathbb{P} : p|m\}$$

die Menge aller Primzahlen, die m teilen. Betrachte die Primfaktorzerlegung von m :

$$m \;=\; p_1^{e_1} \cdot \ldots \cdot p_r^{e_r} \quad \text{mit} \quad r, e_1, \ldots, e_r \in \mathbb{N}$$

Dann ist doch

$$\mathbb{P}_m \;=\; \{p_1, \ldots, p_r\}$$

und somit

$$\prod_{p \in \mathbb{P}_m} p \;\leq\; p_1^{e_1} \cdot \ldots \cdot p_r^{e_r} \;=\; m$$

Insbesondere folgt

$$\prod_{\substack{p \in \mathbb{P}_m \\ p \leq B}} p \;\leq\; m \quad \Leftrightarrow \quad \Big(\prod_{\substack{p \in \mathbb{P}_m \\ p \leq B}} p\Big)^{-1} \;\geq\; \frac{1}{m} \tag{1.15}$$

Wir setzen (1.14) sowie (1.15) in die Behauptung ein und erhalten

$$\prod_{\substack{p \in \mathbb{P},\, p \leq B \\ p \nmid m}} p \;=\; \Big(\prod_{\substack{p \in \mathbb{P} \\ p \leq B}} p\Big) \cdot \Big(\prod_{\substack{p \in \mathbb{P}_m \\ p \leq B}} p\Big)^{-1} \;\geq\; 2^{\vartheta(B)} \cdot \frac{1}{m} \;>\; C$$

■

1.10.4 Berechnungsverfahren und Algorithmus

Um das gestellte Problem zu lösen, eignet sich also die folgende Methode :

a) "Berechne" $B = \max\{43\,,\lceil 2 \cdot \log_2(m \cdot C)\rceil\}$

b) Ermittele mit dem Sieb von Erathostenes alle Primzahlen kleiner oder gleich B

c) Streiche aus dieser Liste alle Primzahlen, die m teilen

Als **Ersatz** für B kann jede Zahl B^* , die für ein festes $\epsilon \in \mathbb{R}^+$

$$B \;\leq\; B^* \;\leq\; B + \epsilon \quad \forall m \in \mathbb{N} \quad \forall C \in \mathbb{N}_{\geq 2}$$

erfüllt, verwendet werden.
Einen möglichen Kandidaten liefert die folgende Abschätzung :

$$\begin{aligned}
B &= \max\{\lceil 2 \cdot \log_2(m \cdot C)\rceil\,, 43\} \\
&\leq \max\{\lceil 2 \cdot \log_2(m)\rceil + \lceil 2 \cdot \log_2(C)\rceil\,, 43\} \\
&\leq \max\{2 \cdot \lceil \log_2(m)\rceil + 2 \cdot \lceil \log_2(C)\rceil\,, 43\} \\
&\leq \max\{2 \cdot (\lfloor \log_2(m)\rfloor + 1) + 2 \cdot (\lfloor \log_2(C)\rfloor + 1)\,, 43\} \\
&= \max\{2 \cdot \#\mathrm{Bits}(m) + 2 \cdot \#\mathrm{Bits}(C)\,, 43\} \\
&=: B^*
\end{aligned}$$

wobei für den Fehler

$$B \leq B^* \leq B + 8$$

gilt.

Bemerkung 1.4 (Verwendung von Tabellen für kleine Werte von m und C) *Für vergleichsweise kleine Werte von m und C ist es sinnvoll, ergänzende Tabellen zur Bestimmung der erforderlichen Primzahlen zu verwenden. Wir veranschaulichen diese Vorgehensweise an einem Beispiel : es sei $m = 385$ und $C = 500$.*

Primzahl p	*Produkt aller Primzahlen $\leq p$*
2	2
3	6
$\vdots$	$\vdots$
13	30030
17	510510
$\vdots$	$\vdots$
43	$1.308 \cdot 10^{16}$

Dann gilt

$$30030 < 385 \cdot 500 < 510510$$

d.h. als Kandidaten kommen

$$2\,,3\,,5\,,7\,,11\,,13\,,17$$

in Frage. Wegen

$$5|385\ ,\ 7|385\ ,\ 11|385$$

verbleiben noch die Primzahlen

$$2\,,3\,,13\,,17 \qquad \textit{mit} \qquad 2 \cdot 3 \cdot 13 \cdot 17 = 1326 > 500$$

welche die gestellte Aufgabe auch lösen.

Bemerkung 1.5 *In mehreren Anwendungen tritt der Fall auf, daß wir nur*

$$p_1 \cdot \ldots \cdot p_k > C$$

anstelle von

$$p_1 \cdot \ldots \cdot p_k > C \qquad , \qquad p_i \nmid m$$

zur Verfügung haben. Wir wählen dann $m = 1$ und ersparen uns das Streichen ungeeigneter Primzahlen im letzten Abschnitt des Algorithmus.

Algorithmus 1.7 (Primzahlliste)

Eingabe : Zahlen $C, m \in \mathbb{N}$ mit $C \geq 2$
Ausgabe : Liste $p_1, \ldots, p_k \in \mathbb{P}$ von Primzahlen mit
a) $p_1 \cdot \ldots \cdot p_k > C$
b) $p_i \nmid m \ \forall i \in \{1, \ldots, k\}$

```
(* Berechne die obere Schranke B* *)
B* := #Bits(m) + #Bits(C)
B* := max{B*+ B*, 43}
(* Markiere zunächst alle Zahlen als prim *)
for ( i := 1 ; i ≤ B* ; i := i + 1 )
  do
    A[i] := 1
  od
(* 1 streichen, da keine Primzahl *)
A[1] := 0
(* Anzahl der gefundenen Primzahlen ist 0 *)
i := 0
(* Bestimme alle Primzahlen ≤ B* *)
j := 2
finish := false
while ( j ≤ B* and finish ≠ true )
  do
    (* Suche nächste Primzahl *)
    while ( j ≤ B* and A[j] == 0 )
      do
        j := j + 1
      od
    if ( j ≤ B* and A[j] ≠ 0 )
      then
        (* Neue Primzahl gefunden *)
        i := i + 1
        q[i] := j
        (* Streiche ab j^2 Vielfache von j *)
        s := j∗j
        while ( s ≤ B* )
          do
            A[s] := 0
            s := s + 2∗j
          od
        (* Alle Primzahlen gefunden ? *)
        if ( j∗j > B* )
          then finish := true
        fi
    fi
    j := j + 1
  od
(* Streiche alle Primzahlen , die m teilen *)
k := 0
j := 0
while ( j ≤ i and m ≠ 1 )
  do
    j := j + 1
    if ( ( m mod q[j] ) ≠ 0 )
      then
```

```
(50)            k := k + 1
(51)            p[k] := q[j]
(52)        fi
(53)    od
(54)  (* Gib Ergebnis zurück *)
(55)  return (p[1] , ... , p[k])
```

Bemerkung 1.6 *Der Startwert j^2 für s in Zeile 29 , sowie die Wahl von $2j$ als Schrittweite in Zeile 33 verhindern, daß eine Nicht-Primzahl doppelt gestrichen wird.*

1.10.5 Laufzeitanalyse

Lemma 1.15 (Primzahlliste)
Die Berechnung von Primzahlen $p_1, \dots, p_k \in \mathbb{P}$ mit

$$p_1 \cdot \ldots \cdot p_k \; > \; C \qquad \textit{und} \qquad p_i \nmid m \quad \forall i \in \{1, \dots, k\}$$

für Zahlen $m \in \mathbb{N}$ und $C \in \mathbb{N}_{\geq 2}$ erfordert Zeit $O(\log^2(m \cdot C))$.

Beweis: betrachte Algorithmus 1.7

1. Bestimmung von B^* : Zeilen (1)-(3)
   Da $B^* = O(\log(m \cdot C))$ ist, kostet dies nach Lemma 1.5 Zeit

   $$O(\log^2(m) + \log^2(C) + \log\log(m \cdot C)) \;=\; O(\log^2(m \cdot C))$$

2. Initialisierung von A[*], i und j : Zeilen (4)-(15)
   Der teuerste Schritt ist das Erhöhen des Schleifenzählers i - dies erfordert jeweils Zeit $O(\log(B^*))$. Zusammen also

   $$O(B^* \cdot \log(B^*)) \;=\; O(\log(m \cdot C) \cdot \log\log(m \cdot C))$$

3. Berechnung aller Primzahlen $\leq B^*$: Zeilen (16)-(41)

   1. Da j in Zeile (21) nur vergrößert, aber nie verkleinert wird, verursacht dies einen Aufwand von

      $$O(B^* \cdot \log(B^*)) \;=\; O(\log(m \cdot C) \cdot \log\log(m \cdot C))$$

   2. Die Zeilen (25)-(29) sowie (36) werden nur für neu ermittelte Primzahlen ausgeführt - d.h. höchstens $\pi(B^*)$ mal. Der Zeitaufwand beträgt somit

      $$O(\pi(B^*) \cdot \log^2(B^*)) \;=\; O(\log(m \cdot C) \cdot \log\log(m \cdot C))$$

   3. Das Streichen von Nicht-Primzahlen in den Zeilen (30)-(34) verursacht noch Kosten

      $$O(B^* \cdot \log(B^*)) \;=\; O(\log(m \cdot C) \cdot \log\log(m \cdot C))$$

4. Entfernen aller Primzahlen, die m teilen : Zeilen (42)-(52)
   Die While-Schleife wird $\pi(B^*)$ mal durchlaufen, wobei ein einzelner Durchlauf Zeit $O(\log(m) \cdot \log(B^*))$ kostet :

$$O(\pi(B^*) \cdot \log(m) \cdot \log(B^*)) \;=\; O(\log(m) \cdot \log(m \cdot C)) \qquad \blacksquare$$

Bemerkung 1.7 *Das Erstaunliche an der obigen Aussage ist, daß die Laufzeit des Siebes von Erathostenes geringer ist als die Kosten für die Berechnung von B^* und das Entfernen unbrauchbarer Primzahlen - dies erklärt das Gesamtergebnis.*

1.11 Maximale Teilvektoren

1.11.1 Problemstellung

Wir beschließen die Vorstellung der Basisalgorithmen mit einer Sortieraufgabe, die wir im Kapitel 3.2 über die Hadamard-Schranke benötigen werden.

Definition 1.21 (Maximaler Teilvektor)
Es sei $v = (v_1, \ldots, v_n)^T \in \mathbb{Z}^n$ und $v^{sort} \in \mathbb{Z}^n$ ein Vektor

$$v^{sort} \;=\; (v_{\sigma(1)}, \ldots, v_{\sigma(n)})^T \quad , \quad \sigma \in \mathcal{S}_n$$

der aus v durch Sortieren entsteht :

$$|v_{\sigma(1)}| \;\geq\; |v_{\sigma(2)}| \;\geq\; \cdots \;\geq\; |v_{\sigma(n)}|$$

Wir nennen $u \in \mathbb{Z}^k$ einen maximalen Teilvektor von v der Länge k, wenn u die ersten k Einträge von v^{sort} in einer beliebigen Reihenfolge enthält.

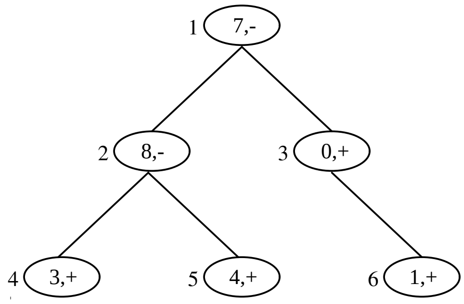
1.11.2 Berechnungsverfahren

Als Grundlage verwenden wir eine Variante des Heapsort-Algorithmus, den wir als bekannt vorraussetzen. Eine ausführliche Darstellung dieses Sortierverfahrens ist in [Cor/Leis/Riv] enthalten. Zur Illustration jener Teile des Algorithmus, die wir verändern, führen wir ein kurzes Beispiel vor.

Beispiel 1.3 *Es sei $v = (-7, -8, 0, 3, 4, 1)^T$ und $k = 3$. Wir schreiben v zuerst als*

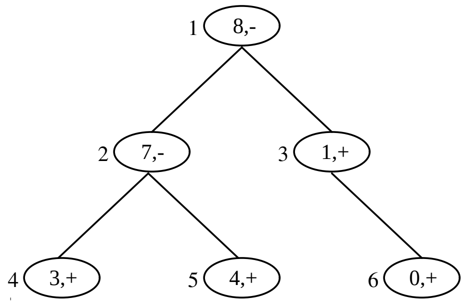
$$v^* = ((7, -), (8, -), (0, +), (3, +), (4, +), (1, +))$$

und betrachten in der Folge nur die erste Komponente jedes Eintrages. Da Heapsort für Bäume besonders einfach zu erklären ist, stellen wir v^* als binären Baum dar, dessen Knoten wir mit zunehmender Tiefe von links nach rechts besetzen :



B

Die erste Phase des Sortiervorganges, also die Umwandlung von B in einen binären Heap, übernehmen wir unverändert : als Ergebnis erhalten wir

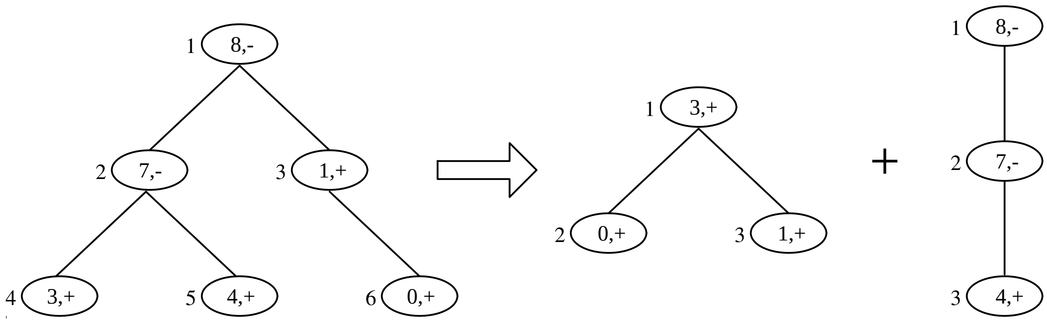


H

Die übliche Vorgehensweise bestünde nun darin, genau n mal die folgenden Schritte in H auszuführen :

- entferne die Wurzel
- füge das Blatt mit der höchsten Nummer als neue Wurzel ein
- lasse die Wurzel absinken

Wiederholen wir diese Anweisungen aber nur k mal, also gerade so oft, wie uns die Länge des gesuchten Teilvektors vorschreibt, kommen wir mit einem Minimum an Austauschschritten aus. Wir wenden dies auf unser Beispiel an :



Der nachfolgend vorgestellte Algorithmus realisiert dieses Schema in einem Array.

1.11.3 Algorithmus

Algorithmus 1.8 (Maximaler Teilvektor)

Eingabe : $v \in \mathbb{Z}^n$ und $k \in \{1, \ldots, n\}$
Ausgabe : Maximaler Teilvektor $u \in \mathbb{Z}^k$ von v der Länge k

Vorgehen :

HAUPTPROGRAMM :

```
(* Korrigiere die Vorzeichen in v *)
for ( i := 1 ; i ≤ n ; i := i + 1 )
   do
      if ( v[i] < 0 )
         then
            v[i] := -v[i]
            sign[i] := -1
         else
            sign[i] := +1
      fi
   od
(* Wandle v in einen binären Heap um *)
heap_size := n
build_heap ( v , n , heap_size )
(* Bestimme die k größten Einträge in v *)
for ( i := n ; i ≥ n-k+1 ; i := i-1 )
   do
      v[1] ↔ v[i]
      sign[1] ↔ sign[i]
      heap_size := heap_size - 1
      (* Wurzel absinken lassen *)
      sink_down ( v , 1 , heap_size )
   od
(* Ergebnis bestimmen *)
for ( i := n ; i ≥ n-k+1 ; i := i-1 )
   do
      u[n-i+1] := sign[i]*v[i]
   od
return (u)
```

WANDLE EINEN VEKTOR V IN EINEN BINÄREN HEAP UM :

```
procedure build-heap
   (array of integer : v ; integer : n , heap_size )
for ( i := n div 2 ; i ≥ 1 ; i := i-1 )
   do
      (* Knoten i absinken lassen *)
      sink_down ( v , i , heap_size )
   od
return
```

Lasse den Knoten i in v absinken :

```
procedure sink_down
   ( array of integer : v ; integer : i , heap_size )
finish := false
while ( finish ≠ true )
   do
      L := 2*i
      R := 2*i+1
      largest := i
      if ( L ≤ heap_size and v[L] > v[i] )
         then largest := L
      fi
      if ( R ≤ heap_size and v[R] > v[largest] )
         then largest := R
      fi
      if ( largest ≠ i )
         then
            v[i] ↔ v[largest]
            sign[i] ↔ sign[largest]
            i := largest
         else
            finish := true
      fi
   od
return
```

1.11.4 Laufzeitanalyse

Lemma 1.16 (Maximaler Teilvektor)
Um für ein $v \in \mathbb{Z}^n$ einen maximalen Teilvektor der Länge k zu berechnen, benötigen wir einen Zeitaufwand der Größenordnung $O((n + k \cdot \log(n)) \cdot \log(|v|))$.

Beweis: wir beziehen uns auf [Cor/Leis/Riv] , Kapitel 7.1-7.4

1. Die Umwandlung von v in einen binären Heap kostet uns $O(n)$ Austauschschritte
2. Das k-fache Absinken der Wurzel erfordert dann noch $O(k \cdot \log(n))$ Vertauschungen

■

Modulare Arithmetik

Als Abschluß der Einführung stellen wir noch ein Konzept vor, welches oft als modulare Arithmetik bezeichnet wird, und uns erlaubt, ein unkontrolliertes Anwachsen von Zwischenergebnissen zu vermeiden. Wir verwenden es als Grundlage für den Teil III dieser Arbeit.
Im Mittelpunkt steht der Chinesische Restsatz und die Frage, wie wir diesen effizient anwenden können.

1.12 Verwendung von Primkörpern

Als erstes benötigen wir einen geeigneten mathematischen Rahmen

- um die für den Chinesischen Restsatz erforderlichen Kongruenzbedingungen aufzustellen
- der uns gestattet, gute obere Schranken für alle auftretenden Zwischenergebnisse anzugeben
- der einfach zu handhaben ist

Als Mittel der Wahl erweist sich hier der Primkörper $\mathbb{F}_p$.

Definition 1.22 (Körperoperation)
Es sei $\mathbb{K}$ ein beliebiger Körper. Als **Körperoperation** *in $\mathbb{K}$ bezeichnen wir das Addieren, Multiplizieren und Invertieren von Elementen aus $\mathbb{K}$.*

Proposition 1.4 (Primkörper $\mathbb{F}_p$) *Die Menge*

$$\mathbb{F}_p := \{0, \ldots, p-1\} \subseteq \mathbb{N}_0 \quad , \quad p \in \mathbb{P}$$

bildet zusammen mit den Verknüpfungen

$$\left.\begin{array}{ccccccccc} a & \oplus & b & := & (a & +_{\mathbb{N}_0} & b) & \bmod & p \\ a & \odot & b & := & (a & \cdot_{\mathbb{N}_0} & b) & \bmod & p \end{array}\right\} \quad \forall a, b \in \mathbb{F}_p$$

einen endlichen Körper.

Wie teuer sind Körperoperationen in $\mathbb{F}_p$?

Lemma 1.17 (Rechnen in $\mathbb{F}_p$)
Eine Körperoperation in $\mathbb{F}_p$ kostet Zeit $O(\log^2(p))$.

Beweis: Für Addition und Multiplikation ist dies klar, bei der Division ist zu beachten, daß wir a^{-1} für ein $a \in \mathbb{F}_p$ mit dem euklidischen Algorithmus berechnen : es sei

$$1 = \alpha \cdot a + \beta \cdot p \quad , \quad \alpha, \beta \in \mathbb{Z}$$

bekannt. Dann ist $a^{-1} = \alpha$. ■

1.13 Direkter Ansatz

1.13.1 Anwendung in $\mathbb{Z}$

Wir erläutern die Methode der modularen Arithmetik in einer einfachen Situation: es bezeichne $x \in \mathbb{Z}$ eine Zahl, die wir berechnen wollen, und über die wir bereits einige Informationen besitzen, so zum Beispiel eine Schranke $S(x) \in \mathbb{N}$ mit

$$-S(x) \leq x \leq +S(x)$$

Wir nehmen weiter an, daß uns ein elementares Verfahren bekannt ist, um x zu bestimmen, welches jedoch den Nachteil hat, daß es zuviel Laufzeit in Anspruch nimmt.

Beispiel 1.4 *Um $x = 94 \cdot 96$ durch handschriftliches Multiplizieren der Binärdarstellungen zu berechnen, benötigen wir $\#\mathrm{Bits}(94) \cdot \#\mathrm{Bits}(96) = 49$ Bitoperationen.*

Wie läßt sich ein solches x mit dem Chinesischen Restsatz berechnen ?

1.Schritt Berechne eine Liste $p_1, \ldots, p_k \in \mathbb{P}$ kleiner Primzahlen mit

$$p_1 \cdot \ldots \cdot p_k > 2 \cdot S(x)$$

und reduziere x modulo dieser Primzahlen, d.h. bestimme

$$x_i \in \mathbb{F}_{p_i} \qquad \forall i \in \{1, \ldots, k\}$$

mit

$$x \equiv x_i \mod p_i$$

Beispiel 1.5 *(Fortsetzung von Beispiel 1.4) Wir haben zunächst*

$$-10^{2+2} \leq 94 \cdot 96 \leq +10^{2+2}$$

sowie

$$2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 > 2 \cdot 10^{2+2}$$

Ferner wissen wir, daß für beliebige $a, b \in \mathbb{Z}$ und $p \in \mathbb{P}$ immer

$$a \cdot b \equiv (a \mod p) \cdot (b \mod p) \mod p$$

gilt. Dies liefert uns die Kongruenzen

$$\begin{array}{lllllll}
94 \cdot 96 & \equiv & 0 \cdot 0 & \equiv & 0 & \mathrm{mod} & 2 \\
94 \cdot 96 & \equiv & 1 \cdot 0 & \equiv & 0 & \mathrm{mod} & 3 \\
94 \cdot 96 & \equiv & 4 \cdot 1 & \equiv & 4 & \mathrm{mod} & 5 \\
94 \cdot 96 & \equiv & 3 \cdot 5 & \equiv & 1 & \mathrm{mod} & 7 \\
94 \cdot 96 & \equiv & 6 \cdot 8 & \equiv & 4 & \mathrm{mod} & 11 \\
94 \cdot 96 & \equiv & 3 \cdot 5 & \equiv & 2 & \mathrm{mod} & 13
\end{array}$$

wobei wir die reduzierten Produkte in den Körpern $\mathbb{F}_{p_i}$ berechnen. Hierfür benötigen wir

$$3 \cdot 1 + 2 \cdot 3 + 3 \cdot 4 + 2 \cdot 3 = 27$$

elementare Bitoperationen.

2.Schritt Wende den Chinesischen Restsatz auf die im 1.Schritt ermittelten Kongruenzen an, d.h. bestimme ein $x^* \in \mathbb{Z}$ mit

$$x^* \equiv x_i \mod p_i \qquad \forall i \in \{1, \ldots, k\}$$

und setze

$$x := x^*$$

Beispiel 1.6 *(Fortsetzung von Beispiel 1.5) Wir suchen ein $x^* \in \mathbb{Z}$ mit*

$$\begin{array}{lclcl} x^* & \equiv & 0 & \mod & 2 \\ x^* & \equiv & 0 & \mod & 3 \\ x^* & \equiv & 4 & \mod & 5 \\ x^* & \equiv & 1 & \mod & 7 \\ x^* & \equiv & 4 & \mod & 11 \\ x^* & \equiv & 2 & \mod & 13 \end{array}$$

und konstruieren zusammen mit der Bedingung

$$-10^{2+2} \leq x^* \leq +10^{2+2}$$

die dann eindeutige Lösung $x^* = 9024$.

Resümee Auch wenn wir im Beispiel 1.5 die Kosten für das Reduzieren, also die Division mit Rest, vernachlässigt haben, ist die nur exemplarisch vorgestellte Methode der *Chinese remainder muliplication* auch im allgemeinen Fall ebenso schnell wie das handschriftliche Multiplizieren (vgl. [Buchm1], Kap. 2.3). Da wir in diesem einführenden Abschnitt ohnehin nur die Methode als solche vorstellen wollen, verweisen wir noch skeptische Leser auf die weniger trivialen Anwendungen der Teile III und IV.

Bemerkung 1.8 (Determinante) *Nach dem gleichen Schema berechnen wir die Determinante eines Endomorphismus $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ - siehe Kapitel 3.3*

1.13.2 Anwendung in $\mathbb{Z}^n$ und $\mathbb{Z}^{m\times n}$

Soll keine einzelne Zahl $x \in \mathbb{Z}$, sondern ein Vektor

$$u = (u_1, \ldots, u_n) \in \mathbb{Z}^n$$

oder eine Matrix

$$M = (m_{i,j}) \in \mathbb{Z}^{m\times n}$$

berechnet werden, wenden wir die zuvor beschriebene Methode komponentenweise an. Wir konstruieren zuerst geeignete Schranken

$$\begin{array}{llll} S(u) \in \mathbb{N} & \text{mit} \quad |u_i| \leq S(u) & \forall i \in \{1, \ldots, n\} & \\ S(M) \in \mathbb{N} & \text{mit} \quad |m_{i,j}| \leq S(M) & \forall i \in \{1, \ldots, m\} & \forall j \in \{1, \ldots, n\} \end{array}$$

und ermitteln kleine Primzahlen $p_1, \ldots, p_k \in \mathbb{P}$, so daß

$$p_1 \cdot \ldots \cdot p_k \;>\; 2 \cdot S(u) \quad \text{bzw.} \quad p_1 \cdot \ldots \cdot p_k \;>\; 2 \cdot S(M)$$

gilt. Entsprechend berechnen wir im nächsten Schritt in $\mathbb{F}_{p_i}^n$ bzw. $\mathbb{F}_{p_i}^{m \times n}$ Vektoren

$$u^{(1)} \in \mathbb{F}_{p_1}^n \,, \ldots , \, u^{(k)} \in \mathbb{F}_{p_k}^n$$

oder Matrizen

$$M^{(1)} \in \mathbb{F}_{p_1}^{m \times n} \,, \ldots , \, M^{(k)} \in \mathbb{F}_{p_k}^{m \times n}$$

mit

$$\begin{aligned} u^{(i)} &\equiv u \quad \bmod \; p_i \quad \forall i \in \{1, \ldots, k\} \\ M^{(i)} &\equiv M \quad \bmod \; p_i \quad \forall i \in \{1, \ldots, k\} \end{aligned}$$

wobei die Kongruenz komponentenweise zu verstehen ist. Als Intervallgrenze wählen wir

$$S(u) \quad \text{bzw.} \quad S(M)$$

und rekonstruiere dann u bzw. M durch wiederholte (komponentenweise) Anwendung des Chinesischen Restsatzes.

Bemerkung 1.9 (Char. Polynom , Adj. Homomorphismus , Reg. Urbild) *Die Berechnung des charakteristischen Polynomes und des adjungierten Homomorphismus verläuft entsprechend obigem Schema - siehe Kapitel 3.6 sowie 3.4. Auch das Urbild von* $b \in \mathbb{Z}^n$ *unter einem regulären* $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ *ließe sich so bestimmen - wir reduzieren dieses Problem jedoch auf (DET) und (ADJ) - vgl. Kapitel 3.5.*

1.14 Verallgemeinerter Ansatz

Eine weitere Variante erhalten wir, wenn für eine *gegebene* Matrix $M \in \mathbb{Z}^{m \times n}$ eine Menge $\mathcal{I}$ von Informationen über M berechnet werden soll. Wie zuvor ermitteln wir in einem ersten Schritt eine Schranke $U \in \mathbb{N}$, deren Bedeutung zunächst offen bleibt, sowie eine Liste kleiner Primzahlen $p_1, \ldots, p_k \in \mathbb{P}$ mit

$$p_1 \cdot \ldots \cdot p_k \;>\; 2 \cdot U$$

Als nächstes bestimmen wir für jede der Matrizen

$$(M \bmod p_i) \;\in\; \mathbb{F}_{p_i}^{m \times n}$$

die entsprechende Information $\mathcal{I}_{p_i}$, wobei wir das jeweilige Problem wie zuvor in $\mathbb{F}_{p_i}^{m \times n}$ betrachten. Im Unterschied zu den beiden vorhergehenden Ansätzen rekonstruieren wir die Elemente von $\mathcal{I}$ aber nicht *unmittelbar* mit dem Chinesischen Restsatz aus den Elementen der Mengen $\mathcal{I}_{p_i}$, sondern mittels einer spezifischen Rekonstruktionsvorschrift $\mathcal{R}$:

$$\mathcal{I} \;:=\; \mathcal{R}\,(\,\mathcal{I}_{p_1}, \ldots, \mathcal{I}_{p_k}\,)$$

Die Korrektheit von $\mathcal{R}$ hängt in der Regel davon ab, ob die Schranke U hinreichend groß gewählt wurde.

Bemerkung 1.10 (Rang und linear unabhängige Zeilen) *Sowohl die Berechnung des Ranges, als auch die Bestimmung von Indizes linear unabhängiger Zeilen sind Beispiele für den verallgemeinerten Ansatz - siehe Kapitel 3.7.*

1.15 Grenzen der modularen Arithmetik

Ein aufmerksamer Leser wird festgestellt haben, daß wir bis auf (KERN) und (BILD) sowie (URBILD) und (HNF) alle betrachteten Probleme durch Varianten der modularen Arithmetik lösen. Wir setzen diese Vorgehensweise aus mehreren Gründen nicht fort :

1. Zum einen sind Basen freier Moduln, also auch von Kern und Bild nicht eindeutig bestimmt : ein unmittelbarer Ansatz über Kongruenzen scheidet also aus.
2. Zum anderen schließt bei der Hermite-Normalform die Natur des Problemes eine Anwendung des Chinesischen Restsatzes aus - die Transformation ist auch hier nicht eindeutig.
3. Das allgemeine Urbildproblem reduzieren wir auf die Berechnung eines Kernes.

Wir verschieben die Diskussion, welche Strategie sich anstelle der modularen Arithmetik hierfür eignet und weisen darauf hin, daß wir dies an geeigneter Stelle nachholen.

Teil II

Lineare Algebra über Körpern

2.1 Einordnung von Teil II

Wir wiederholen zunächst den Begriff des Vektorraumhomomorphismus und erläutern dann die Bedeutung des Teiles II innerhalb dieser Arbeit.

2.1.1 Vektorraumhomomorphismen

Definition 2.23 ($\mathbb{K}$-Vektorraumhomomorphismus)
Es seien $(V, +, \cdot)$ und $(W, +, \cdot)$ Vektorräume über einem Körper $\mathbb{K}$. Eine Abbildung

$$\varphi : V \to W$$

heißt $\mathbb{K}$-Vektorraumhomomorphismus genau dann, wenn

1. $\varphi(v + u) \;=\; \varphi(v) + \varphi(u) \qquad \forall v, u \in V$
2. $\varphi(\alpha \cdot v) \;=\; \alpha \cdot \varphi(v) \qquad \forall \alpha \in \mathbb{K} \quad \forall v \in V$

gilt. Wir vereinbaren die Notationen

$$\mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m) \;:=\; \{\varphi : \mathbb{K}^n \to \mathbb{K}^m \mid \varphi \textit{ ist ein } \mathbb{K}\textit{-Vektorraumhomomorphismus } \}$$

sowie

$$\mathrm{End}(\mathbb{K}^n) \;:=\; \{\varphi : \mathbb{K}^n \to \mathbb{K}^n \mid \varphi \textit{ ist ein } \mathbb{K}\textit{-Vektorraumendomorphismus } \}$$

Als Darstellungen von Homomorphismen verwenden wir Matrizen mit Einträgen aus $\mathbb{K}$.

Definition 2.24 (Darstellung eines Homomorphismus)
Es sei $\varphi \in \mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m)$. Als Darstellung von φ bezeichnen wir die Matrix

$$\mathrm{Mat}(\varphi) \;:=\; (\varphi(e_1^n), \ldots, \varphi(e_n^n)) \in \mathbb{K}^{m \times n}$$

wobei

$$e_1^n, \ldots, e_n^n \in \mathbb{K}^n$$

die aus Einheitsvektoren bestehende Standardbasis von $\mathbb{K}^n$ sei.

Lemma 2.18 (Verknüpfung von Homomorphismen)
Es seien $\varphi \in \mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m)$ und $\psi \in \mathrm{Hom}(\mathbb{K}^m, \mathbb{K}^l)$. Dann gilt

$$\mathrm{Mat}(\psi \circ \varphi) \;=\; \mathrm{Mat}(\psi) \cdot \mathrm{Mat}(\varphi)$$

Die so erklärte Darstellung von $\varphi \in \mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m)$ erfüllt

$$\varphi(v) \;=\; \mathrm{Mat}(\varphi) \cdot v$$

wodurch eine umkehrbar eindeutige Zuordnung zwischen Homomorphismen und Matrizen möglich wird.

Definition 2.25 *Wir notieren den einer Matrix $A \in \mathbb{K}^{m \times n}$ zugeordneten Homomorphismus durch φ_A. Für diesen gilt*

$$\varphi_A(x) \;=\; A \cdot x \quad , \quad \varphi_A \in \mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m) \quad , \quad x \in \mathbb{K}^n$$

2.1.2 Untersuchte Problemstellungen

Als Vorbereitung und Grundlage für Teil III betrachten wir einige der Probleme, die wir in Teil I vorgestellt haben, auch für Vektorraumhomomorphismen, verwenden jedoch der Einfachheit halber nur matrixorientierte Formulierungen. Dies sind für quadratische Matrizen

- Determinante, inverse Matrix und charakteristisches Polynom

sowie für beliebige Matrizen

- Rang und linear unabhängige Zeilen, sowie Kern und Bild

Wir untersuchen ferner als Pendant zum allgemeinen Urbild, wie wir die Bestimmung der Lösungsmenge eines linearen Gleichungssystemes auf die Berechnung von Kernen zurückführen können. Weshalb interessieren wir uns für diese Fragen, wo wir eigentlich Modulhomomorphismen untersuchen wollen ?
Ein Anliegen der Einführung war die Vorstellung der modularen Arithmetik : wir lösen ein Problem, indem wir es in reduzierter Form in Primkörpern betrachten und die so erhaltenen Informationen mit dem Chinesischen Restsatz zur Gesamtlösung zusammensetzen - direkt oder indirekt. Wir wenden uns im Teil II dem **ersten** Schritt dieser Vorgehensweise zu : der Untersuchung von Problemen in Primkörpern - oder besser in beliebigen Körpern. Unser Ziel ist es, Aussagen der Art

“*Zur Lösung des Problemes $\mathcal{P}$ über dem Körper $\mathbb{K}$ benötigen wir $O(f)$ Körperoperationen in $\mathbb{K}$*“

herzuleiten, wobei f eine geeignete Funktion ist. Zuerst stellen wir jedoch ein zentrales Hilfsmittel vor, welches für mehrere Kapitel relevant ist : der Begriff der Spaltenstufenform für Matrizen.

2.2 Matrizen in Spaltenstufenform

2.2.1 Problemstellung

Definition 2.26 (Spaltenstufenform über Körpern)
Sei $A \in \mathbb{K}^{m\times n}$ mit den Spaltenvektoren $a_1, \ldots, a_n \in \mathbb{K}^m$. Wir definieren Zahlen

$$l^A(j) := \begin{cases} 0 & , \text{ falls } \quad a_j = 0_{\mathbb{K}^m} \\ \text{Zeilenindex des letzten von } 0_{\mathbb{K}} \text{ verschiedenen Eintrages in } a_j & , \text{ falls } \quad a_j \neq 0_{\mathbb{K}^m} \end{cases}$$

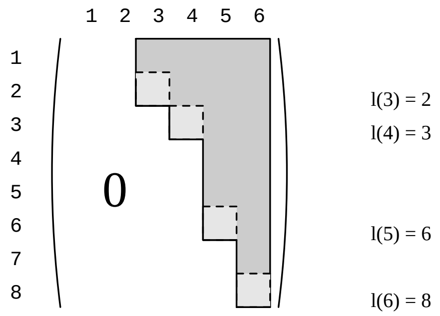
für $j \in \{1, \ldots, n\}$. A ist in Spaltenstufenform, wenn gilt :

$$l^A(j+1) \;\geq\; l^A(j) \qquad \forall j \in \{1, \ldots, n-1\}$$

und

$$l^A(j) > 0 \quad \Rightarrow \quad l^A(j+1) > l^A(j) \qquad \forall j \in \{1, \dots, n-1\}$$

Die Matrix A hat dann etwa folgende Gestalt :



Wir benötigen noch den Begriff der unimodularen Transformation.

Lemma 2.19 (Unimodulare Matrix)
Eine Matrix $T \in \mathbb{K}^{n \times n}$ *heißt unimodular, wenn* $\det(T) \in \{+1_{\mathbb{K}}, -1_{\mathbb{K}}\}$ *ist.*

Definition 2.27 (Unimodulare Transformation)
Als unimodulare Transformation bezeichnen wir das Heranmultiplizieren einer unimodularen Matrix von rechts an eine gegebene Ausgangsmatrix.

Aufgabenstellung :
Gegeben sei $A \in \mathbb{K}^{m \times n}$. *Transformiere* A *unimodular auf Spaltenstufenform : berechne*

$$A \cdot T \;=\; \mathrm{STF}(A)$$

mit einem unimodularem $T \in \mathbb{K}^{n \times n}$ *und* $\mathrm{STF}(A) \in \mathbb{K}^{m \times n}$ *in Spaltenstufenform.*

2.2.2 Berechnungsverfahren

Eine naheliegende Vorgehensweise besteht darin, die Ausgangsmatrix A durch **Gaußsche Eliminationsschritte** in Spaltenstufenform zu überführen, wobei wir alle Umformungen auch in $T \in \mathbb{K}^{n \times n}$ ausführen und zu Beginn $T := I_n$ setzen. Die Zeilen in A bearbeiten wir von unten nach oben, und die Spalten in einer Zeile von rechts nach links.

2.2.2.1 Eliminationsschritt

Um $u \in \mathbb{K} \setminus \{0\}$ mittels $v \in \mathbb{K} \setminus \{0\}$, zu eliminieren, eignet sich die Transformation

$$(u, v) \cdot \begin{pmatrix} 1 & 0 \\ -\frac{u}{v} & 1 \end{pmatrix} \;=\; (0, v)$$

Befindet sich u an der Stelle (i, j) und v an Stelle (i, j_0) , wobei j_0 der Spaltenindex des aktuellen Pivotelementes sei, entspricht dies der **Spaltenumformung**

$$S_j \;\mapsto\; S_j - \left(\frac{u}{v}\right) \cdot S_{j_0}$$

2.2.2.2 Pivotsuche

Es bezeichne A^* eine im Verlaufe der Berechnung entstandene Matrix. Ein **Sonderfall** tritt auf, wenn in der aktuell betrachteten Zeile i in der Pivotspalte j_0 eine 0 erscheint, sich aber links davon noch ein von 0 verschiedenes Element befindet :

$$A^*[i, j] \neq 0_{\mathbb{K}} \quad , \quad A^*[i, j_0] = 0_{\mathbb{K}} \quad \text{und} \quad j < j_0$$

Um den Eliminationsschritt anwenden zu können, vertauschen wir die Spalten j und j_0 in A^* und registrieren dies in einer Variablen

$$\text{exchange} \in \{+1_{\mathbb{K}}, -1_{\mathbb{K}}\}$$

welche zu Beginn den Wert $+1_{\mathbb{K}}$ hat. Am Ende der Berechnung gilt dann

$$\det(T) \;=\; \text{exchange}$$

2.2.3 Algorithmus

Algorithmus 2.9 (Spaltenstufenform über Körpern)

Eingabe :
Matrix $A \in \mathbb{K}^{m \times n}$

Ausgabe :
1. Unimodulares $T \in \mathbb{K}^{n \times n}$ sowie $\mathrm{STF}(A) \in \mathbb{K}^{m \times n}$ in Spaltenstufenform mit

$$A \cdot T \;=\; \mathrm{STF}(A)$$

2. Determinante von T

```
(* Initialisierung *)
exchange := +1_K
j0 := n
T := I_n
(* Berechne die Spaltenstufenform *)
for ( i := m ; i ≥ 1 ; i := i - 1 )
  do
    j := j0
    (* Suche Pivotelement *)
    while ( j ≥ 1 and A[i,j] == 0_K )
      do
        j := j - 1
```

```
         od
      (* Vertausche gegebenenfalls die Spalten j und j0 in A und T *)
      if ( j ≠ 0 ) then
         if ( j ≠ j0 ) then
            exchange := -exchange
            for ( z := 1 ; z ≤ m ; z := z + 1 )
               do
                  A[z,j]↔A[z,j0]
               od
            for ( z := 1 ; z ≤ n ; z := z + 1 )
               do
                  T[z,j]↔T[z,j0]
               od
         fi
         (* Eliminiere in A links vom Pivotelement *)
         for ( j := j0 - 1 ; j ≥ 1 ; j := j - 1 )
            do
               (* Eliminationsschritt *)
               q := A[i,j] / A[i,j0]
               for ( z := 1 ; z ≤ m ; z := z + 1 )
                  do
                     A[z,j] := A[z,j] - q · A[z,j0]
                  od
               for ( z := 1 ; z ≤ n ; z := z + 1 )
                  do
                     T[z,j] := T[z,j] - q · T[z,j0]
                  od
            od
         (* Aktualisiere j0 *)
         j0 := j0 - 1
      fi
   od
(* Ergebnis bestimmen *)
STF(A) := A
det(T) := exchange
return (T , STF(A) , det(T))
```

2.2.4 Laufzeitanalyse

Lemma 2.20 (Spaltenstufenform über Körpern)
Es sei $A \in \mathbb{K}^{m\times n}$. Dann benötigen wir $O(m^2 n)$ Körperoperationen in $\mathbb{K}$, um eine Matrix $\mathrm{STF}(A) \in \mathbb{K}^{m\times n}$ zu berechnen, die Spaltenstufenform hat und aus A durch eine unimodulare Transformation hervorgeht. Die Bestimmung der Transformationsmatrix $T \in \mathbb{K}^{n\times n}$ kostet uns $O(mn \cdot (m+n)))$ Körperoperationen in $\mathbb{K}$.

2.3 Determinante

2.3.1 Problemstellung

Wir präzisieren im folgenden den bereits mehrfach verwendeten Determinantenbegriff.

Definition 2.28 (Permutation)
Eine Permutation σ von n Elementen ist eine bijektive Abbildung

$$\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$$

der Menge $\mathbb{N}_n$ in sich selbst. Wir verwenden die Notation

$$\sigma \;=\; \begin{pmatrix} 1 & 2 & \ldots & n \\ \sigma_1 & \sigma_2 & \ldots & \sigma_n \end{pmatrix} \quad , \quad \sigma_i := \sigma(i)$$

Faßt man für ein festes $n \in \mathbb{N}$ alle Permutationen zu einer Menge $\mathcal{S}_n$ zusammen, und erklärt durch

$$\circ \;:\; \mathcal{S}_n \times \mathcal{S}_n \to \mathcal{S}_n$$
$$(\sigma \circ \tau)(i) \;:=\; \sigma(\tau(i))$$

eine Verknüpfung, erhält man eine Gruppe $(\mathcal{S}_n, \circ)$, die als **Permutationsgruppe** *der Ordnung n bezeichnet wird.*

Definition 2.29 *Sei $\sigma \in \mathcal{S}_n$ eine Permutation.*

1. *Ein Paar $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ heißt* **Fehlstand** *von σ, wenn*

$$i < j \quad \text{und} \quad \sigma(i) > \sigma(j)$$

   gilt.

2. *Zählt man alle Fehlstände von σ zusammen, ergibt dies die* **Inversionszahl** *:*

$$\mathrm{inv}(\sigma) \;:=\; \sum_{\substack{i<j \\ \sigma(j)>\sigma(j)}} 1$$

3. *Das* **Vorzeichen** *von σ über $\mathbb{K}$ ist definiert als*

$$\mathrm{sign}_{\mathbb{K}}(\sigma) \;:=\; (-1_{\mathbb{K}})^{\mathrm{inv}(\sigma)}$$

Die nachfolgende Definition wird in manchen Lehrbüchern als Lemma formuliert - wir bevorzugen sie jedoch gegenüber der Einführung von Determinanten durch Multilinearformen, da sie für unsere Zwecke besser geeignet ist.

Definition 2.30 (Determinante über Körpern)

1. *Als Determinante einer Matrix* $A = (a_{i,j}) \in \mathbb{K}^{n\times n}$ *bezeichnen wir die Summe*

$$\det(A) := \sum_{\sigma\in\mathcal{S}_n} \left(\operatorname{sign}_{\mathbb{K}}(\sigma) \cdot \prod_{i=1}^{n} a_{i,\sigma(i)} \right)$$

2. *Für einen Endomorphismus* $\varphi \in \operatorname{End}(\mathbb{K}^n)$ *definieren wir die Determinante von* φ *durch*

$$\det(\varphi) := \det(\operatorname{Mat}(\varphi))$$

Aufgabenstellung :
Berechne die Determinante von $A \in \mathbb{K}^{n\times n}$.

2.3.2 Berechnungsverfahren

2.3.2.1 Rechenregeln für Determinanten

1. Determinantenmultiplikationssatz :

$$\det(A \cdot B) = \det(A) \cdot \det(B) \quad \forall A, B \in \mathbb{K}^{n\times n}$$

2. Dreiecksmatrizen :
Ist $A = (a_{i,j}) \in \mathbb{K}^{n\times n}$ *eine obere Dreiecksmatrix*

$$a_{i,j} = 0_{\mathbb{K}} \quad \textit{für} \quad 1 \le j < i \le n$$

entspricht die Determinante von A dem Produkt der Diagonalelemente :

$$\det(A) = \prod_{i=1}^{n} a_{i,i}$$

3. Spaltenumformungen :
Es seien $A, B \in \mathbb{K}^{n\times n}$.
   1. *Geht B aus A durch Vertauschen der Spalten i und j hervor*

   $$A \overset{S_i \leftrightarrow S_j}{\mapsto} B$$

   gilt

   $$\det(B) = -\det(A)$$

   2. *Ist* $\lambda \in \mathbb{K}$ *und entsteht B aus A durch Addition des* λ*-fachen der* j*-ten Spalte zur* i*-ten Spalte*

   $$A \overset{S_i \mapsto S_i + \lambda\cdot S_j}{\mapsto} B$$

   so ist

   $$\det(B) = \det(A)$$

   3. *Durch Multiplikation der Spalte j in A mit* $\lambda \in \mathbb{K}$

   $$A \overset{S_j \mapsto \lambda\cdot S_j}{\mapsto} B$$

   verändert sich die Determinante von A linear in λ *:*

   $$\det(B) = \lambda \cdot \det(A)$$

2.3.2.2 Verwendung der Spaltenstufenform

Wir transformieren $A \in \mathbb{K}^{n \times n}$ unimodular auf Spaltenstufenform

$$A \cdot T = \mathrm{STF}(A) = (s_{i,j}) \in \mathbb{K}^{n \times n}$$

und wenden den Determinantenmultiplikationssatz an :

$$\det(A) = \det(T) \cdot \det(\mathrm{STF}(A))$$

Da $\mathrm{STF}(A)$ eine obere Dreiecksmatrix ist, erhalten wir

$$\det(A) = \det(T) \cdot \prod_{i=1}^{n} s_{i,i}$$

2.3.3 Algorithmus

Algorithmus 2.10 (Determinante über Körpern)

Eingabe : $A \in \mathbb{K}^{n \times n}$
Ausgabe : $\det(A) \in \mathbb{K}$

Vorgehen :

1. Transformiere A unimodular auf Spaltenstufenform : berechne

$$A \cdot T = \mathrm{STF}(A)$$

sowie

$$\det(T) \in \{+1_{\mathbb{K}}, -1_{\mathbb{K}}\}$$

2. Bestimme das Produkt der Hauptdiagonalelemente von $\mathrm{STF}(A) = (s_{i,j})$:

$$D := \prod_{i=1}^{n} s_{i,i}$$

3. Ermittele als Ergebnis $\det(A) := \det(T) \cdot D$

2.3.4 Laufzeitanalyse

Lemma 2.21 (Determinante über Körpern)
Die Berechnung der Determinante von $A \in \mathbb{K}^{n \times n}$ kostet $O(n^3)$ Körperoperationen in $\mathbb{K}$.

2.4 Inverse Matrix

2.4.1 Problemstellung

Definition 2.31 *Wir nennen eine Matrix* $A \in \mathbb{K}^{n\times n}$ **regulär**, *wenn* $\det(A) \neq 0_{\mathbb{K}}$ *ist, und vereinbaren die Notation*

$$\mathrm{GL}(n,\mathbb{K}) := \{A \in \mathbb{K}^{n\times n} \mid \det(A) \neq 0_{\mathbb{K}}\}$$

Falls $\det(A) = 0_{\mathbb{K}}$ *gilt, heißt* A **singulär**.

Eine wichtige Eigenschaft regulärer Matrizen ist, daß die über $\mathbb{K}$ invertierbar sind.

Lemma 2.22 (Inverse Matrix über Körpern)
Für jedes $A \in \mathrm{GL}(n,\mathbb{K})$ *gibt es eine eindeutig bestimmte Matrix* $A^{-1} \in \mathrm{GL}(n,\mathbb{K})$ *mit*

$$A \cdot A^{-1} = A^{-1} \cdot A = I_n$$

Wir bezeichnen A^{-1} *als inverse Matrix von* A.

Aufgabenstellung :
Berechne A^{-1} *für ein* $A \in \mathrm{GL}(n,\mathbb{K})$.

2.4.2 Berechnungsverfahren

Wir verwenden die übliche Vorgehensweise zum Invertieren von $A \in \mathrm{GL}(n,\mathbb{K})$:

1. Forme A durch elementare Spaltenumformungen (Gaußsche Eliminationsschritte) zur Einheitsmatrix I_n um.
2. Führe gleichzeitig alle Spaltenoperationen in einer Matrix $B \in \mathbb{K}^{n\times n}$ durch, die mit I_n initialisiert wird.

Am Ende der Berechnung gilt dann $B = A^{-1}$.

2.4.3 Algorithmus

Algorithmus 2.11 (Inverse Matrix über Körpern)

Eingabe : $A \in \mathrm{GL}(n,\mathbb{K})$
Ausgabe : $A^{-1} \in \mathrm{GL}(n,\mathbb{K})$

```
(* Initialisierung *)
B := I_n
(* Forme A zur Einheitsmatrix um *)
for ( i := n ; i ≥ 1 ; i := i - 1 )
  do
    (* Überprüfe Pivotelement *)
    j := i
    while ( j ≥ 1 and A[i,j] == 0_𝕂 )
      do
        j := j - 1
      od
    (* Vertausche falls nötig die Spalten i und j *)
    if ( j ≠ i ) then
      for ( z := 1 ; z ≤ n ; z := z+1 )
        do
          A[z,i]↔A[z,j]
          B[z,i]↔B[z,j]
        od
    fi
    (* Eliminiere außerhalb der Diagonalen *)
    for ( j := 1 ; j ≤ n ; j := j + 1 )
      do
        if ( j ≠ i ) then
          q := A[i,j] / A[i,i]
          for ( z := 1 ; z ≤ n ; z := z+1 )
            do
              A[z,j] := A[z,j] - q · A[z,i]
              B[z,j] := B[z,j] - q · B[z,i]
            od
        fi
      od
  od
(* Korrigiere die Diagonale *)
for ( j := 1 ; j ≤ n ; j := j + 1 )
  do
    for ( z := 1 ; z ≤ n ; z := z + 1 )
      do
        B[z,j] := B[z,j] / A[j,j]
      od
  od
(* Bestimme Ergebnis *)
A^-1 := B
return (A^-1)
```

2.4.4 Laufzeitanalyse

Lemma 2.23 (Inverse Matrix über Körpern)
Die Berechnung der Inversen von $A \in \mathrm{GL}(n, \mathbb{K})$ kostet $O(n^3)$ Körperoperationen in $\mathbb{K}$.

2.5 Charakteristisches Polynom

2.5.1 Problemstellung

Definition 2.32 (Charakteristisches Polynom über Körpern)
Es sei X eine Unbestimmte über $\mathbb{K}$ und $\mathbb{K}[X]$ der Polynomring in X.

1. *Wir ordnen einer Matrix $A \in \mathbb{K}^{n \times n}$ durch die Festlegung*

$$c_A(X) := \det(A - X \cdot I_n) \in \mathbb{K}[X]$$

ein Polynom vom Grad n zu und nennen c_A das charakteristische Polynom von A.

2. *Entsprechend definieren wir*

$$c_\varphi(X) := \det(\varphi - X \cdot \mathrm{id}) \in \mathbb{K}[X]$$

für ein $\varphi \in \mathrm{End}(\mathbb{K}^n)$, wobei id die identische Abbildung auf $\mathbb{K}^n$ bezeichne.

Eine wichtige Anwendung solcher Polynome ist die Berechnung von Eigenwerten.

Aufgabenstellung :
Bestimme den Koeffizientenvektor

$$c = (c_n, \ldots, c_0)^T \in \mathbb{K}^{n+1}$$

des charakteristischen Polynomes

$$c_A(X) = c_n \cdot X^n + \ldots + c_1 \cdot X + c_0$$

einer Matrix $A \in \mathbb{K}^{n \times n}$.

2.5.2 Eigenschaften des charakteristischen Polynomes

2.5.2.1 Ähnliche Matrizen

Definition 2.33 (Ähnliche Matrizen über Körpern)
Eine Matrix $A \in \mathbb{K}^{n \times n}$ heißt ähnlich zu $B \in \mathbb{K}^{n \times n}$, wenn es ein $M \in \mathrm{GL}(n, \mathbb{K})$ mit

$$B = M^{-1} \cdot A \cdot M$$

gibt.

Die so erklärte Relation

$$A \sim B \quad \Leftrightarrow \quad A \text{ ist ähnlich zu } B$$

ist reflexiv, transitiv und symmetrisch : die Menge $\mathbb{K}^{n\times n}$ wird dadurch in Äquivalenzklassen zerteilt, deren Elemente das selbe charakteristische Polynom haben - wir zeigen dies im folgenden Lemma.

Lemma 2.24 *Es seien $A, B \in \mathbb{K}^{n\times n}$ und A ähnlich zu B. Dann gilt*

$$c_A(X) \;=\; c_B(X)$$

Beweis: Nach Voraussetzung gibt es ein $M \in \mathrm{GL}(n, \mathbb{K})$ mit

$$B \;=\; M^{-1} \cdot A \cdot M$$

woraus

$$\begin{aligned} c_B(X) &= \det(B - X \cdot I_n) \\ &= \det(M^{-1} \cdot (A - X \cdot I_n) \cdot M) \\ &= \det(M)^{-1} \cdot \det(A - X \cdot I_n) \cdot \det(M) \\ &= c_A(X) \end{aligned}$$

folgt. ■

Es reicht also aus, einen geeigneten **Vertreter** einer Äquivalenzklasse zu betrachten.

2.5.2.2 Determinante und Normierung

Eine weitere interessante Eigenschaft des charakteristischen Polynomes ist, daß es die Determinante der betrachteten Matrix enthält.

Lemma 2.25 *Es sei $A = (a_{i,j}) \in \mathbb{K}^{n\times n}$ und $c_A(X) = c_n \cdot X^n + \ldots + c_1 \cdot X + c_0$.*
1. Das Absolutglied von c_A stimmt mit der Determinante überein :

$$c_0 \;=\; \det(A)$$

2. Der Betrag des höchsten Koeffizienten ist gleich 1 :

$$c_n \;=\; (-1_{\mathbb{K}})^n$$

Beweis:
1. Ergibt sich aus $c_0 = c_A(0) = \det(A)$.
2. Der einzige Summand, der in

$$\det(A - X \cdot I_n) \;=\; \sum_{\sigma\in\mathcal{S}_n} \left(\mathrm{sign}_{\mathbb{K}} \cdot \prod_{i=1}^{n} (a_{i,\sigma(i)} - \delta_{i,\sigma(i)} \cdot X) \right)$$

genau n Faktoren mit X enthält, gehört zu $\sigma = \mathrm{id}$:

$$\Rightarrow\; c_n = (-1_{\mathbb{K}})^n$$ ■

2.5.3 Berechnungsverfahren

Wie bereits erläutert, wollen wir aus der Menge aller äquivalenten Matrizen einen geeigneten Vertreter herausgreifen. Wir erläutern nun eine mögliche Vorgehensweise.

Definition 2.34 (Matrix in Hessenberg-Form)
Eine Matrix $H = (h_{i,j}) \in \mathbb{K}^{n\times n}$ ist in (oberer) Hessenberg-Form, wenn in ihr nur das obere Dreieck und die erste Nebendiagonale besetzt ist :

$$h_{i,j} \;=\; 0_{\mathbb{K}} \qquad \forall i,j \in \{1,\ldots,n\} \text{ mit } i > j+1$$

Weshalb interessieren wir uns für diesen speziellen Matrixtyp ?

1. In nächsten Abschnitt werden wir zeigen, daß jede quadratische Matrix A in eine zu A ähnliche Matrix H in Hessenberg-Form transformiert werden kann.
2. Das charakteristische Polynom der so bestimmten Matrix H berechnen wir dann mit einer passenden Rekursionsformel.

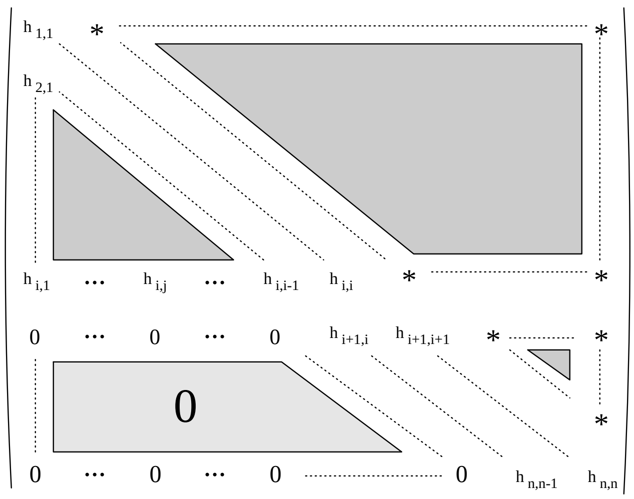
2.5.4 Transformation auf Hessenberg-Form

2.5.4.1 Vorgehensweise

Lemma 2.26 (Hessenberg-Form als Vertreter der Ähnlichkeitsrelation)
Jede Matrix $A \in \mathbb{K}^{n\times n}$ ist ähnlich zu einem $H \in \mathbb{K}^{n\times n}$ in Hessenberg-Form.

Beweis: durch Konstruktion.
Wir verwenden Gaußsche Eliminationsschritte, um H aus A zu berechnen, führen jedoch auch komplementäre Zeilenoperationen durch. Als Pivotelement wählen wir immer den Schnittpunkt der unteren Nebendiagonalen mit der aktuellen Zeile.
Um die Vorgehensweise zu veranschaulichen, bezeichne $H^* = (h_{i,j}) \in \mathbb{K}^{n\times n}$ eine Matrix, in der die letzten $n-i$ Zeilen bereits die richtige Form besitzen :

$H^* =$



Zunächst suchen wir unter den Einträgen

$$h_{i,1}, \ldots, h_{i,i-1}$$

einen von $0_{\mathbb{K}}$ verschiedenen : gibt es den nicht, ist die Zeile i in Ordnung. Andernfalls machen wir den Eintrag $h_{i,i-1}$ durch Vertauschen zweier Spalten von Null verschieden. Damit dies eine Ähnlichkeitstransformation bleibt, müssen wir auch noch die entsprechenden Zeilen vertauschen - die Form wird dadurch nicht verändert.
Als nächstes benutzen wir das Pivotelement $h_{i,i-1}$, um die restlichen Einträge

$$h_{i,i-2}, \ldots, h_{i,1}$$

zu eliminieren, wobei wir die komplementären Zeilenumformungen gleich mit ausführen :

1. Setze $q := h_{i,j} \,/\, h_{i,i-1}$
2. $S_j \mapsto S_j - q \cdot S_{i-1}$
3. $Z_{i-1} \mapsto Z_{i-1} + q \cdot Z_j$

für alle Spaltenindizes $j \in \{1, \ldots, i-2\}$.
Beide Transformationen haben keinen Einfluß auf die Form von H^*.
Bearbeiten wir die Zeilen in der Ausgangsmatrix A von unten nach oben gemäß diesem Schema, erhalten wir die gesuchte Matrix H. ■

2.5.4.2 Algorithmus

Algorithmus 2.12 (Transformation auf Hessenberg-Form)

Eingabe : $A \in \mathbb{K}^{n \times n}$
Ausgabe : $H \in \mathbb{K}^{n \times n}$ in Hessenberg-Form, das ähnlich zu A ist

```
(* Initialisierung *)
H := A
(* Transformiere H auf Hessenberg-Form *)
for ( i := n ; i ≥ 2 ; i := i - 1 )
  do
    j := i - 1
    while ( j > 0 and H[i,j] == 0𝕂 )
      do
        j := j - 1
      od
    if ( j ≠ 0 ) then
      (* Korrigiere falls nötig das Pivotelement *)
      if ( j ≠ i - 1 ) then
        (* Vertausche die Spalten j und i-1 *)
        for ( z := 1 ; z ≤ n ; z := z + 1 )
          do
            H[z,j]↔H[z,i-1]
          od
```

```
            (* Vertausche die Zeilen j und i-1 *)
            for ( s := 1 ; s ≤ n ; s := s + 1 )
              do
                H[j,s]↔H[i-1,s]
              do
          fi
          (* Eliminiere links vom Pivotelement *)
          for ( j := i - 2 ; j ≥ 1 ; j := j - 1 )
            do
              q := H[i,j] / H[i,i-1]
              for ( z := 1 ; z ≤ n ; z := z + 1 )
                do
                  H[z,j] := H[z,j] - q · H[z,i-1]
                od
              for ( s := 1 ; s ≤ n ; s := s + 1 )
                do
                  H[i-1,s] := H[i-1,s] + q · H[j,s]
                od
            od
      fi
  od
(* Gib Ergebnis zurück *)
return (H)
```

2.5.4.3 Laufzeitanalyse

Lemma 2.27 (Transformation auf Hessenberg-Form)
Sei $A \in \mathbb{K}^{n\times n}$. Wir benötigen $O(n^3)$ Körperoperationen in $\mathbb{K}$, um eine Matrix $H \in \mathbb{K}^{n\times n}$ zu berechnen, die in Hessenberg-Form ist und aus A durch eine Ähnlichkeitstransformation hervorgeht.

2.5.5 Charakteristisches Polynom einer Hessenberg-Matrix

2.5.5.1 Rekursionsformel

Lemma 2.28 (Charakteristisches Polynom einer Matrix in Hessenberg-Form)
Sei $H = (h_{i,j}) \in \mathbb{K}^{n\times n}$ in Hessenberg-Form und $p_r(X)$ das charakteristische Polynom des r-ten Hauptminors H_r von H, also

$$H_r \;=\; (h_{i,j})_{1\le i,j\le r} \in \mathbb{K}^{r\times r}$$

und

$$p_r(X) \;=\; \det(H_r - X\cdot I_r)$$

Dann gilt

$$\begin{aligned} p_r(X) &= (h_{r,r} - X) \cdot p_{r-1}(X) \; + \\ & \sum_{j=1}^{r-1} \left((-1_{\mathbb{K}})^{r-j} \cdot p_{j-1}(X) \cdot h_{j,r} \cdot \prod_{k=j+1}^{r} h_{k,k-1} \right) \end{aligned}$$

Beweis: Durch Entwickeln der Determinante von $(H_r - X \cdot I_r)$ nach der letzten Spalte, wobei deren Einträge von unten nach oben durchlaufen werden. ■

2.5.5.2 Auswerten der Rekursionsformel

Wir wollen für eine Matrix $H = (h_{i,j}) \in \mathbb{K}^{n \times n}$ in Hessenberg-Form

$$c_H(X) \;=\; \sum_{j=0}^{n} c_j \cdot X^j$$

bestimmen, und berechnen gemäß Lemma 2.28 nacheinander die Polynome

$$p_0(X) := 1 \qquad \text{sowie} \qquad p_1(X), \ldots, p_n(X)$$

Dann ist $c_H(X) = p_n(X)$. Die mehrfach auftretenden Produkte

$$K_r(j) \;:=\; \prod_{k=j+1}^{r} h_{k,k-1} \qquad \forall r \in \{1, \ldots, n\} \;\; \forall j \in \{1, \ldots, r-1\}$$

werten wir nicht jedesmal neu aus, sondern benutzen die Rekursion

$$\begin{aligned} K_r(j) &= K_{r-1}(j) \cdot h_{r,r-1} & \forall r \in \{2, \ldots, n\} \;\; \forall j \in \{1, \ldots, r-1\} \\ K_1(j) &= 1 & \forall j \in \{1, \ldots, n\} \end{aligned}$$

Ferner verwenden wir zur Abspeicherung der Koeffizienten von

$$p_r(X) \;=\; c_r^{(r)} \cdot X^r + \ldots + c_1^{(r)} \cdot X + c_0^{(r)}$$

eine Matrix

$$P \;=\; (p_{i,j}) \in \mathbb{K}^{(n+1)\times(n+1)}$$

deren Zeilen bzw. Spalten wir mit 0 beginnend numerieren, und die wir wie folgt belegen :

$$p_{r,j} \;=\; c_r^{(j)} \qquad \forall j \in \{0, \ldots, n\} \;\; \forall r \in \{0, \ldots, j\}$$

Am Ende der Berechnung gilt dann

$$c_H(X) \;=\; p_{n,n}(X) \cdot X^n + \ldots + p_{1,n} \cdot X + p_{0,n}$$

2.5.5.3 Algorithmus

Algorithmus 2.13 (Char. Polynom einer Hessenberg-Matrix)

Eingabe : $H \in \mathbb{K}^{n\times n}$ in Hessenberg-Form
Ausgabe : Koeffizientenvektor

$$(c_n, \ldots, c_0)^T \in \mathbb{K}^{n+1}$$

des charakteristischen Polynomes von H

```
(* Initialisierung *)
for ( j := 1 ; j ≤ n ; j := j + 1 )
  do
    K[j] := 1
  od
(* Belege die Koeffizientenmatrix *)
P := 0
P[0,0] := 1
(* Berechne nacheinander die Polynome p_r *)
for ( r := 1 ; r ≤ n ; r := r + 1 )
  do
    (* Aktualisiere K *)
    for ( j := 1 ; j ≤ r - 1 ; j := j + 1 )
      do
        K[j] := K[j] · H[r,r-1]
      od
    (* Addiere ersten Summanden *)
    P[r,r] := -P[r-1,r-1]
    for ( i := 1 ; i ≤ r - 1 ; i := i + 1 )
      do
        P[i,r] := H[r,r] · P[i,r-1] - P[i-1,r-1]
      od
    P[0,r] := H[r,r] · P[0,r-1]
    (* Addiere den Summenterm *)
    sign := 1_K
    for ( j := r - 1 ; j ≥ 1 ; j := j - 1 )
      do
        sign := -sign
        for ( i := 0 ; i ≤ j - 1 ; i := i + 1 )
          do
            P[i,r] := P[i,r] + sign · P[i,j-1] · H[j,r] · K[j]
          od
      od
  od
(* Bestimme den Koeffizientenvektor von c_H *)
for ( i := 0 ; i ≤ n ; i := i + 1 )
  do
```

```
    c[j] := P[i,n]
  od
return (c[n] , ... , c[0])
```

2.5.5.4 Laufzeitanalyse

Lemma 2.29 (Charakteristisches Polynom einer Hessenberg-Matrix)
Es sei $H \in \mathbb{K}^{n \times n}$ eine Matrix in Hessenberg-Form. Dann benötigen wir $O(n^3)$ Körperoperationen in $\mathbb{K}$, um das charakteristische Polynom von H zu berechnen.

2.5.6 Algorithmus

Wir fügen nun die Einzelteile zu **einem** Algorithmus zusammen.

Algorithmus 2.14 (Charakteristisches Polynom über Körpern)

Eingabe : $A \in \mathbb{K}^{n \times n}$
Ausgabe : Koeffizientenvektor $(c_n, \dots, c_0)^T \in \mathbb{K}^{n+1}$
des charakteristischen Polynomes von A :

$$c_A(X) \;=\; c_n \cdot X^n + \ldots + c_1 \cdot X + c_0$$

Vorgehen :

1. Bestimme eine Matrix $H \in \mathbb{K}^{n \times n}$ in Hessenberg-Form,
   die aus A durch eine Ähnlichkeitstransformation hervorgeht
2. Berechne das charakteristische Polynom $c_H(X)$ von H
3. Setze $c_A(X) := c_H(X)$

2.5.7 Laufzeitanalyse

Lemma 2.30 (Charakteristisches Polynom über Körpern)
Es sei $A \in \mathbb{K}^{n \times n}$. Dann benötigen wir $O(n^3)$ Körperoperationen in $\mathbb{K}$, um das charakteristische Polynom von A zu berechnen.

2.6 Rang und linear unabhängige Zeilen

2.6.1 Problemstellung

Wir wiederholen zunächst den Begriff des Ranges.

Definition 2.35 *Es seien* $A \in \mathbb{K}^{m\times n}$ *und* $\varphi_A \in \mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m)$.

1. *Als* **Rang** *von* φ_A *über* $\mathbb{K}$ *bezeichnen wir die Dimension des Bildes von* φ_A

$$\mathrm{Bild}(\varphi_A) \;=\; \{y \in \mathbb{K}^m \mid y = \varphi_A(x) \text{ für ein } x \in \mathbb{K}^n\}$$

   welches ein $\mathbb{K}$*-Untervektorraum von* $\mathbb{K}^m$ *ist :*

$$\mathrm{rang}_{\mathbb{K}}(\varphi_A) \;:=\; \dim_{\mathbb{K}} \mathrm{Bild}(\varphi_A)$$

2. *Der* **Spaltenrang** *von* A *über* $\mathbb{K}$ *ist erklärt als*

$$\mathrm{Spaltenrang}_{\mathbb{K}}(A) \;:=\; \mathrm{rang}_{\mathbb{K}}(\varphi_A)$$

   und entsprechend definieren wir den **Zeilenrang** *über* $\mathbb{K}$ *durch*

$$\mathrm{Zeilenrang}_{\mathbb{K}}(A) \;:=\; \mathrm{rang}_{\mathbb{K}}(\varphi_{A^T})$$

   wobei $A^T \in \mathbb{K}^{n\times m}$ *die transponierte Matrix von* A *sei.*

Durch das folgende Lemma ordnen wir auch einer Matrix einen einheitlichen Rang zu.

Lemma 2.31 *Für jedes* $A \in \mathbb{K}^{m\times n}$ *gilt*

$$\mathrm{Spaltenrang}_{\mathbb{K}}(A) \;=\; \mathrm{Zeilenrang}_{\mathbb{K}}(A)$$

Definition 2.36 (Rang einer Matrix über einem Körper)
Sei $A \in \mathbb{K}^{m\times n}$*. Dann nennen wir*

$$\mathrm{rang}_{\mathbb{K}}(A) \;:=\; \mathrm{Spaltenrang}_{\mathbb{K}}(A)$$

den Rang von A *über* $\mathbb{K}$*.*

Aufgabenstellung :
Berechne den Rang r *einer Matrix* $A \in \mathbb{K}^{m\times n}$ *über* $\mathbb{K}$*, sowie Indizes*

$$i_1, \ldots, i_r \in \{1, \ldots, m\}$$

linear unabhängiger Zeilen aus A*.*

2.6.2 Berechnungsverfahren

2.6.2.1 Zeilenrangfunktion einer Matrix

Definition 2.37 *Sei* $A \in \mathbb{K}^{m\times n}$ *mit den Zeilenvektoren* $c_1, \ldots, c_m \in \mathbb{K}^n$*.*
Als **Zeilenrangfunktion** *von* A **über** $\mathbb{K}$ *bezeichnen wir die Abbildung*

$$\mathrm{zr}_A^{\mathbb{K}} : \{1, \ldots, m\} \to \mathbb{N}_0$$

mit

$$\mathrm{zr}_A^{\mathbb{K}}(i) := \mathrm{rang}_{\mathbb{K}}(c_i, c_{i+1}, \ldots, c_m)$$

Dann ist

$$\mathrm{zr}_A^{\mathbb{K}}(1) = \mathrm{rang}_{\mathbb{K}}(A)$$

sowie

$$\mathrm{zr}_A^{\mathbb{K}}(i+1) \leq \mathrm{zr}_A^{\mathbb{K}}(i) \qquad \forall i \in \{1, \ldots, m-1\}$$

Von besonderem Interesse sind die Sprungstellen von $\mathrm{zr}_A^{\mathbb{K}}$, also Indizes i mit

$$\mathrm{zr}_A^{\mathbb{K}}(i) > \mathrm{zr}_A^{\mathbb{K}}(i+1)$$

Lemma 2.32 (Auswahl linear unabhängiger Zeilen über Körpern)
Sei $A \in \mathbb{K}^{m \times n}$ und $r = \mathrm{rang}_{\mathbb{K}}(A)$. Die Zeilenrangfunktion $\mathrm{zr}_A^{\mathbb{K}}$ hat genau r Sprungstellen

$$i_1, \ldots, i_r \in \{1, \ldots, m\}$$

mit

$$\mathrm{zr}_A^{\mathbb{K}}(i_j) > \mathrm{zr}_A^{\mathbb{K}}(i_j+1) \qquad \forall j \in \{1, \ldots, r\}$$

Insbesondere sind die Zeilen mit den Indizes $i_1, \ldots, i_r$ linear unabhängig über $\mathbb{K}$.

Wie berechnet man nun die Zeilenrangfunktion einer Matrix ?

2.6.2.2 Verwendung der Spaltenstufenform

Lemma 2.33 *Die Zeilenrangfunktion $\mathrm{zr}_A^{\mathbb{K}}$ von $A \in \mathbb{K}^{m \times n}$ wird durch Heranmultiplizieren einer regulären Transformationsmatrix von rechts nicht verändert.*

Beweis: Es sei $H \in \mathrm{GL}(n, \mathbb{K})$ eine reguläre Transformationsmatrix und $A_h \in \mathbb{K}^{h \times n}$ jene Teilmatrix von A, die aus den letzten h Zeilen bestehe. Dann ist

$$\mathrm{rang}_{\mathbb{K}}(A_h \cdot H) = \mathrm{rang}_{\mathbb{K}}(A_h) \quad \text{(siehe [Fischer] , Satz 2.8.4 , Seite 100)}$$

und somit

$$\mathrm{zr}_{A \cdot H}^{\mathbb{K}}(h) = \mathrm{zr}_A^{\mathbb{K}}(h) \qquad \forall h \in \{1, \ldots, m\}$$

■

Wir verwenden das nächste Lemma als Grundlage für einen Algorithmus.

Lemma 2.34 (Berechnung der Zeilenrangfunktion über Körpern)
Sei $S \in \mathbb{K}^{m \times n}$ in Spaltenstufenform mit den Spaltenvektoren $s_1, \ldots, s_n$ und $l^S(j)$ gemäß Definition 2.26 der Zeilenindex des letzten von $0_{\mathbb{K}}$ verschiedenen Eintrages in s_j. Ferner seien $i_1, \ldots, i_r$ entsprechend Lemma 2.32 die Sprungstellen der Zeilenrangfunktion $\mathrm{zr}_S^{\mathbb{K}}$. Sind die letzten r Spalten in S von $0_{\mathbb{K}^m}$ verschieden, also

$$l^S(1) = \ldots = l^S(n-r) = 0 \quad , \quad l^S(n-r+1) \neq 0$$

gilt

$$
\begin{aligned}
i_r &= l^S(n-r+1) \\
&\vdots \\
i_j &= l^S(n-j+1) \\
&\vdots \\
i_1 &= l^S(n-1+1)
\end{aligned}
$$

Beweis: Folgt aus der Definition der Spaltenstufenform und

$$\text{Spaltenrang}_{\mathbb{K}}(S_h) = \text{Zeilenrang}_{\mathbb{K}}(S_h)$$

wobei S_h die letzten h Zeilen von S enthalte. ∎

2.6.3 Algorithmus

Algorithmus 2.15 (Rang und linear unabhängige Zeilen über Körpern)

Eingabe :
Matrix $A \in \mathbb{K}^{m \times n}$

Ausgabe :
1. Rang r von A über $\mathbb{K}$
2. Indizes $i_1, \ldots, i_r \in \{1, \ldots, m\}$ linear unabhängiger Zeilen aus A

Vorgehen :

1. Berechne eine Matrix $\text{STF}(A) \in \mathbb{K}^{m \times n}$ in Spaltenstufenform, die aus A durch eine unimodulare Transformation hervorgeht
2. Lies aus $\text{STF}(A)$ die Indizes $i_1, \ldots, i_r$ ab :

```
(* Initialisierung *)
S := STF(A)
(* Bestimme die Zahlen l^S(j) *)
for ( j := 1 ; j ≤ n ; j := j + 1 )
   do
      i := m
      while ( i ≥ 1 and S[i,j] == 0_K )
         do
            i := i - 1
         od
      l[j] := i
   od
(* Ergebnis ermitteln *)
j0 := 1
while ( j0 ≤ n and l[j0] == 0 )
   do
      j0 := j0 + 1
```

```
   od
r := n - j0 + 1
for ( j := j0 ; j ≤ n ; j := j + 1 )
   do
      i[n-j+1] := l[j]
   od
return (r , i[1] , ... , i[r])
```

2.6.4 Laufzeitanalyse

Lemma 2.35 (Rang und linear unabhängige Zeilen über Körpern)
Um den Rang r einer Matrix $A \in \mathbb{K}^{m \times n}$ sowie Indizes $i_1, \ldots, i_r$ linear unabhängige Zeilen aus A zu berechnen, benötigen wir $O(m^2 n)$ Körperoperationen in $\mathbb{K}$.

2.7 Linearkombinationen

2.7.1 Problemstellung

Definition 2.38 (Linearkombinationen über Körpern)
Es seien $v_1, \ldots, v_k \in \mathbb{K}^n$. Wir nennen jedes $v \in \mathbb{K}^n$ mit

$$v \;=\; \beta_1 \cdot v_1 + \ldots + \beta_k \cdot v_k \qquad , \qquad \beta_1, \ldots, \beta_k \in \mathbb{K}$$

eine Linearkombination von $v_1, \ldots, v_k$ über $\mathbb{K}$.

Wir weisen bereits an dieser Stelle darauf hin, daß wir Linearkombinationen als Hilfsmittel im Kapitel 4.2 über die Urbildberechnung in $\mathbb{Z}$ benötigen werden.

Aufgabenstellung :
Bestimme $v = \beta_1 \cdot v_1 + \ldots + \beta_k \cdot v_k$ für $v_1, \ldots, v_k \in \mathbb{K}^n$ und $\beta_1, \ldots, \beta_k \in \mathbb{K}$.

2.7.2 Berechnungsverfahren

Wir werten die auftretenden Summen einfach komponentenweise aus.

2.7.3 Algorithmus

Algorithmus 2.16 (Linearkombinationen über Körpern)

Eingabe : 1. Vektoren $v_1, \ldots, v_k \in \mathbb{K}^n$
2. Koeffizienten $\beta_1, \ldots, \beta_k \in \mathbb{K}$

Ausgabe : Summe $v = \beta_1 \cdot v_1 + \ldots + \beta_k \cdot v_k$

```
(* Initialisierung *)
v := 0
(* Berechne die Summe *)
for ( j := 1 ; j ≤ k ; j := j + 1 )
  do
    for ( i := 1 ; i ≤ n ; i := i + 1 )
      do
        v[i] := v[i] + β_j · v_j[i]
      od
  od
(* Ergebnis zurückgeben *)
return (v)
```

2.7.4 Laufzeitanalyse

Lemma 2.36 (Linearkombinationen über Körpern)
Es seien $v_1, \ldots, v_k \in \mathbb{K}^n$ und $\beta_1, \ldots, \beta_k \in \mathbb{K}$. Dann müssen wir $O(kn)$ Körperoperationen in $\mathbb{K}$ ausführen, um die Linearkombination $v = \beta_1 \cdot v_1 + \ldots + \beta_k \cdot v_k$ zu berechnen.

2.8 Kern und Bild

2.8.1 Problemstellung

Definition 2.39 (Kern und Bild über Körpern)
Für ein $A \in \mathbb{K}^{m \times n}$ definieren wir

$$\begin{aligned} \mathrm{Kern}(A) &:= \{x \in \mathbb{K}^n \mid A \cdot x = 0_{\mathbb{K}^m}\} \\ \mathrm{Bild}(A) &:= \{y \in \mathbb{K}^m \mid y = A \cdot x \text{ für ein } x \in \mathbb{K}^n\} \end{aligned}$$

Beide Mengen sind $\mathbb{K}$-Untervektorräume von $\mathbb{K}^n$ bzw. $\mathbb{K}^m$.

Als Darstellungen von Kern und Bild verwenden wir wie üblich Basen.

Lemma 2.37 *Sei $A \in \mathbb{K}^{m \times n}$ und r der Rang von A über $\mathbb{K}$. Dann gilt*

$$n = \dim_{\mathbb{K}} \mathrm{Kern}(A) + \dim_{\mathbb{K}} \mathrm{Bild}(A)$$

bzw.

$$\begin{aligned} \dim_{\mathbb{K}} \mathrm{Kern}(A) &= n - r \\ \dim_{\mathbb{K}} \mathrm{Bild}(A) &= r \end{aligned}$$

Der Einfachheit halber betrachten wir im folgenden die Berechnung von Kern und Bild als Einheit, obwohl dies natürlich nicht zwingend ist. Eine Aufspaltung des Algorithmus in zwei getrennte Darstellungen bereitet keine Schwierigkeiten.

Aufgabenstellung :
Bestimme für ein $A \in \mathbb{K}^{m \times n}$ *Basen von Kern(A) und Bild(A) .*

2.8.2 Berechnungsverfahren

Wir benutzen das folgende Lemma als Grundlage für einen Algorithmus.

Lemma 2.38 *Sei* $A \in \mathbb{K}^{m \times n}$ *und* $T \in \mathbb{K}^{n \times n}$ *eine unimodulare Matrix, die A auf Spaltenstufenform transformiert*

$$A \cdot T \;=\; \mathrm{STF}(A) \;=\; (0, \ldots, 0, s_1, \ldots, s_r)$$

wobei r die Anzahl der von Null verschiedenen Spalten in $\mathrm{STF}(A)$ *bezeichne. Dann bilden die ersten* $n - r$ *Spalten in T eine Basis von Kern(A) und die letzten r Spalten in* $\mathrm{STF}(A)$ *eine Basis von Bild(A).*

Beweis: Aus der Definition der Spaltenstufenform folgt, daß die Vektoren

$$s_1, \ldots, s_r \in \mathbb{K}^m$$

linear unabhängig sind. Zusammen mit

$$\mathrm{Bild}(A) \;=\; \mathrm{Bild}(\mathrm{STF}(A))$$

und

$$\dim_{\mathbb{K}} \mathrm{Bild}(A) \;=\; r$$

liefert dies den ersten Teil der Behauptung. Ferner sind die ersten $n-r$ Spalten in T wegen $T \in \mathrm{GL}(n, \mathbb{K})$ ebenfalls linear unabhängig und im Kern von A enthalten. Aus

$$\dim_{\mathbb{K}} \mathrm{Kern}(A) \;=\; n - r$$

folgt dann der zweite Teil des Lemmas. ■

2.8.3 Algorithmus

Algorithmus 2.17 (Kern und Bild über Körpern)

Eingabe : $A \in \mathbb{K}^{m \times n}$
Ausgabe : 1. Basis $(u_1, \ldots, u_{n-r})$ von $\mathrm{Kern}(A)$
2. Basis $(w_1, \ldots, w_r)$ von $\mathrm{Bild}(A)$

Vorgehen :

1. Transformiere A unimodular auf Spaltenstufenform : berechne
$$A \cdot T = \mathrm{STF}(A)$$
Dabei sei
$$T = (t_1, \ldots, t_n) \in \mathbb{K}^{n \times n}$$
und
$$\mathrm{STF}(A) = (s_1, \ldots, s_n) \in \mathbb{K}^{m \times n}$$
2. Zähle die Anzahl α der führenden Nullspalten in $\mathrm{STF}(A)$
3. Übernimm die ersten α Spalten von T als Basis des Kernes
$$u_i := t_i \qquad \forall i \in \{1, \ldots, \alpha\}$$
und die letzten $n - \alpha$ Spalten von $\mathrm{STF}(A)$ als Basis des Bildes :
$$w_i := s_i \qquad \forall i \in \{\alpha + 1, \ldots, n\}$$

2.8.4 Laufzeitanalyse

Lemma 2.39 (Kern und Bild über Körpern)
Es sei $A \in \mathbb{K}^{m \times n}$. Um eine Basis des Bildes von A zu berechnen, benötigen wir $O(m^2 n)$ Körperoperationen in $\mathbb{K}$. Die Bestimmung einer Basis von $\mathrm{Kern}(A)$ *kostet uns hingegen $O(mn \cdot (m+n))$ Körperoperationen in $\mathbb{K}$.*

2.9 Lineare Gleichungssysteme

2.9.1 Problemstellung

Definition 2.40 (Spezielle Lösung und Lösungsmenge eines LGLS)
Es sei ein lineares Gleichungssystem mit einer Koeffizientenmatrix $A \in \mathbb{K}^{m \times n}$ und einer rechten Seite $b \in \mathbb{K}^m$ gegeben. Wir nennen jedes $x \in \mathbb{K}^n$, das $A \cdot x = b$ erfüllt, eine spezielle Lösung dieses Systemes. Ferner notieren wir durch
$$\mathcal{LG}(A, b) := \{x \in \mathbb{K}^n \mid A \cdot x = b\}$$
die Menge aller Lösungen.

Wir interessieren uns für solche Gleichungssysteme aus dem einfachen Grund, daß sich für ein $\varphi \in \mathrm{Hom}(\mathbb{K}^n, \mathbb{K}^m)$ das Urbildproblem $\varphi(x) = b$ als $\mathrm{Mat}(\varphi) \cdot x = b$ formulieren läßt.

Aufgabenstellung :
Es seien $A \in \mathbb{K}^{m \times n}$ und $b \in \mathbb{K}^m$ gegeben. Berechne ein $x^* \in \mathcal{LG}(A, b)$.

2.9.2 Berechnungsverfahren

2.9.2.1 Berechnung einer speziellen Lösung

Wir schreiben das Gleichungssystem

$$A \cdot x \;=\; b \tag{2.16}$$

für ein A mit den Spalten

$$a_1, \ldots, a_n \in \mathbb{K}^m$$

und

$$x = (x_1, \ldots, x_n)^T \in \mathbb{K}^n$$

als

$$x_1 \cdot a_1 + \ldots + x_n \cdot a_n - 1 \cdot b \;=\; 0$$

Daraus folgt, daß (2.16) genau dann lösbar ist, wenn der Kern von (A, b) einen Vektor

$$x^* = (x_1^*, \ldots, x_n^*, -1)^T \in \mathbb{K}^{n+1}$$

enthält. Um dies herauszufinden, überprüfen wir zuerst

$$\dim_{\mathbb{K}} \operatorname{Kern}(A, b) > 0$$

und berechnen dann eine Basis

$$U = (u_1, \ldots, u_{n+1-r}) \in \mathbb{K}^{(n+1)\times(n+1-r)}$$

des Kernes von (A, b), wobei

$$r = \operatorname{rang}_{\mathbb{K}}(A, b)$$

gelte. Enthält U in der letzten Zeile nur Nullen, gibt es kein solches x^* und somit auch keine spezielle Lösung von (2.16). Andernfalls wählen wir eine Spalte u_j , deren $n+1$.te Komponente $u_{n+1,j}$ von $0_{\mathbb{K}}$ verschieden ist, und setzen

$$\hat{x} \;:=\; -\frac{1}{u_{n+1,j}} \cdot u_j$$

Die gesuchte Lösung x^* von (2.16) steht dann in den ersten n Komponenten von $\hat{x}$.

2.9.2.2 Berechnung der Lösungsmenge

Um auf der Grundlage einer speziellen Lösung x^* von $A \cdot x = b$ die Menge $\mathcal{LG}(A, b)$ zu bestimmen, reicht es aus zu beobachten, daß für beliebige $y, z \in \mathcal{LG}(A, b)$ immer

$$y - z \in \operatorname{Kern}(A)$$

gilt. Wir haben also die Darstellung

$$\mathcal{LG}(A, b) \;=\; x^* + \operatorname{Kern}(A)$$

2.9.3 Algorithmus

Algorithmus 2.18 (Lineare Gleichungssysteme über Körpern)

Eingabe :

1. Koeffizientenmatrix $A \in \mathbb{K}^{m \times n}$
2. Rechte Seite $b \in \mathbb{K}^m$

Ausgabe :

Spezielle Lösung $x^* \in \mathcal{LG}(A, b)$ oder $\mathcal{LG}(A, b) = \emptyset$

Vorgehen :

1. Transformiere (A, b) unimodular auf Spaltenstufenform :

$$(A, b) \cdot T \ = \ \mathrm{STF}(A, b)$$

   Dabei sei

$$T \ = \ (t_1, \ldots, t_{n+1}) \in \mathbb{K}^{(n+1) \times (n+1)}$$

2. Zähle die Anzahl α der führenden Nullspalten in $\mathrm{STF}(A, b)$ und setze

$$U := \ (t_1, \ldots, t_\alpha) \in \mathbb{K}^{(n+1) \times \alpha}$$

3. Ist $\alpha = 0$ oder enthält U in der letzten Zeile nur den Wert $0_{\mathbb{K}}$, so gib

$$\mathcal{LG}(A, b) \ = \ \emptyset$$

   aus und beende die Bearbeitung.

4. Andernfalls wähle eine Spalte

$$\tilde{u} = (\tilde{u}_1, \ldots, \tilde{u}_{n+1})^T \in \mathbb{K}^{n+1}$$

   aus U mit $\tilde{u}_{n+1} \neq 0_{\mathbb{K}}$ und setze

$$\hat{x} := -\frac{1}{\tilde{u}_{n+1}} \cdot \tilde{u}$$

5. Die gesuchte Lösung x^* steht jetzt in den ersten n Komponenten von $\hat{x}$

2.9.4 Laufzeitanalyse

Lemma 2.40 (Lineare Gleichungssysteme über Körpern)
Um eine spezielle Lösung $x^* \in \mathcal{LG}(A, b)$ eines linearen Gleichungssystemes

$$A \cdot x \ = \ b$$

mit einer Koeffizientenmatrix $A \in \mathbb{K}^{m \times n}$ und einer rechten Seite $b \in \mathbb{K}^m$ zu berechnen, benötigen wir $O(mn \cdot (m + n))$ Körperoperationen in $\mathbb{K}$.

Teil III

Elementare Probleme der linearen Algebra über $\mathbb{Z}$

3.1 Einordnung von TeilIII und TeilIV

Wir beginnen nun mit der Untersuchung jener Problemstellungen, die wir am Anfang von Teil I als Thema dieser Arbeit vorgestellt haben. Unser Ziel ist es, einige typische Fragen zu beantworten, die sich bei der Anwendung von $\mathbb{Z}$-Modulhomomorphismen stellen, wie zum Beispiel die Berechnung des Ranges. Wir interessieren uns nicht nur dafür, wie wir diese Probleme prinzipiell lösen können, sondern vor allem dafür, wieviel Zeit uns dies kostet, also für Aussagen der Art

Um den Rang von $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ zu berechnen, benötigen wir Zeit $O(f(\varphi, n, m))$

mit einer geeigneten Funktion f. Dabei legen wir den in Kapitel 1.4 eingeführten Laufzeitbegriff zugrunde, also die Bitkomplexität arithmetischer Operationen in $\mathbb{Z}$. Als Kriterium dafür, ob wir ein Problem als elementar ansehen und entsprechend im TeilIII abhandeln, verwenden wir die Frage, ob modulare Arithmetik zur Lösung ausreicht :

- im TeilIII erläutern wir auf der Grundlage der im TeilII über Körpern formulierten Algorithmen für die elementaren Probleme die Anwendungsmöglichkeiten des Chinesischen Restsatzes. Dabei erlauben wir sowohl den direkten als auch den verallgemeinerten Ansatz.
- im TeilIV stellen wir Algorithmen zur Berechnung von Kern und Bild sowie der Hermite-Normalform und der entsprechenden Transformationsmatrix vor, welche nicht auf dem Chinesischen Restsatz aufbauen. Wir reduzieren ferner das allgemeine Urbildproblem auf die Berechnung von Kernen.

Wie bereits erläutert müssen wir, um einen Homomorphismus φ zu untersuchen, immer eine Darstellung von φ, also $\mathrm{Mat}(\varphi)$ verwenden. Daher geben wir zu Beginn eines jeden Abschnittes eine **matrixorientierte Formulierung** des jeweiligen Problemes an. Für das Beispiel der Rangberechnung lautet die Aufgabenstellung also

Bestimme den Rang von $A \in \mathbb{Z}^{m \times n}$

wobei wir A als Stellvertreter für $\mathrm{Mat}(\varphi)$ ansehen. Die **Ergebnisse** formulieren wir dann sowohl für Matrizen als auch für Homomorphismen.

3.2 Hadamard-Schranke

Wir werden öfters obere Schranken für die Absolutbeträge von Determinanten quadratischer Matrizen oder Untermatrizen benötigen. Aus diesem Grund führen wir im folgenden die Hadamard-Schranke ein. Dabei betrachten wir zuerst den quadratischen Fall und übertragen dann die dortige Vorgehensweise in geeigneter Form auf nichtquadratische Matrizen.

3.2.1 Quadratische Matrizen

3.2.1.1 Definition und Eigenschaften

Definition 3.41 (Hadamard-Schranke) *Es sei $A \in \mathbb{Z}^{n\times n}$ mit den Spalten $a_1, \ldots, a_n$ und den Zeilen $\hat{a}_1, \ldots, \hat{a}_n$. Wir definieren*

$$\mathrm{H}_{\mathcal{S}}(A) := \prod_{j=1}^{n} \|a_j\|_2 \qquad , \qquad \mathrm{H}_{\mathcal{Z}}(A) := \prod_{i=1}^{n} \|\hat{a}_i\|_2$$

sowie

$$\mathrm{H}(A) := \min\{\mathrm{H}_{\mathcal{S}}(A), \mathrm{H}_{\mathcal{Z}}(A)\}$$

und nennen $\mathrm{H}(A)$ Hadamard-Schranke von A.

Das nächste Lemma rechtfertigt die Bezeichnung Schranke.

Lemma 3.41 *Für eine Matrix $A \in \mathbb{Z}^{n\times n}$ gelten mit $\mathcal{R} \in \{\mathcal{S}, \mathcal{Z}\}$ die Ungleichungen*

$$|\det(A)| \leq \mathrm{H}_{\mathcal{R}}(A)$$

und

$$\mathrm{H}_{\mathcal{R}}(A) \leq (\sqrt[2]{n} \cdot |A|)^n$$

Beweis: Aus der analytischen Geometrie ist bekannt, daß wir das Volumen eines Parallelotops im $\mathbb{R}^n$ bestimmen können, indem wir die Determinante der Seitenvektoren ausrechnen. Wir betrachten jenes Parallelotop $\mathcal{P}$, welches von den Spalten bzw. Zeilen der Matrix A aufgespannt wird und sehen, das $\mathrm{vol}(\mathcal{P})$ genau dann maximal wir, wenn die Seitenvektoren paarweise aufeinander senkrecht stehen. ∎

3.2.1.2 Approximation der Hadamard-Schranke

Definition 3.42 (Genaue obere Schranke)
Es seien $x, y \in \mathbb{N}$ und $x \leq y$. Wir nennen y eine genaue obere Schranke für x, wenn

$$\#\mathrm{Bits}(y) - \#\mathrm{Bits}(x) \leq 1$$

gilt.

Wir berechnen die Hadamard-Schranke nicht explizit, da die Bestimmung von Wurzeln teuer ist. Stattdessen verwenden wir eine gute Näherung, die uns weniger Zeit kostet.

Lemma 3.42 (Approximation von Wurzeln) *Sei $n \in \mathbb{N}$. Dann ist*

$$\langle n \rangle^{\frac{1}{2}} := 2^{\lfloor \frac{1}{2}(\#\mathrm{Bits}(n)-1)\rfloor + 2} - 1$$

eine genaue obere Schranke für $\lceil \sqrt[2]{n} \rceil$.

Beweis: Zunächst wissen wir

$$\begin{aligned}
\#\mathrm{Bits}(\lfloor\sqrt[2]{n}\rfloor) &= \lfloor\log_2(\lfloor\sqrt[2]{n}\rfloor)\rfloor + 1 \\
&= \lfloor\log_2(\sqrt[2]{n})\rfloor + 1 && \text{(siehe [Grah/Knuth/Pat] , Seite 73 , 3.10)} \\
&= \lfloor\tfrac{1}{2}\cdot\log_2(n)\rfloor + 1 \\
&= \lfloor\tfrac{1}{2}\cdot\lfloor\log_2(n)\rfloor\rfloor + 1 && \text{(siehe [Grah/Knuth/Pat] , Seite 73 , 3.10)} \\
&= \lfloor\tfrac{1}{2}\cdot(\#\mathrm{Bits}(n)-1)\rfloor + 1
\end{aligned}$$

sowie

$$\#\mathrm{Bits}(\langle n\rangle^{\frac{1}{2}}) \;=\; \lfloor\tfrac{1}{2}\cdot(\#\mathrm{Bits}(n)-1)\rfloor + 2$$

woraus wegen

$$\#\mathrm{Bits}(\lceil\sqrt[2]{n}\rceil) - \#\mathrm{Bits}(\lfloor\sqrt[2]{n}\rfloor) \;\le\; 1$$

die Behauptung folgt. ■

Wir übertragen diese Vorgehensweise auf die Hadamard-Schranke.

Korollar 3.2 (Approximation der Hadamard-Schranke)
Für eine quadratische Matrix $A \in \mathbb{Z}^{n\times n}$ ist

$$\mathrm{H}^*(A) \;:=\; \langle\mathrm{H}(A)^2\rangle^{\frac{1}{2}}$$

mit

$$\mathrm{H}(A)^2 \;=\; \min\{\mathrm{H}_{\mathcal{S}}(A)^2, \mathrm{H}_{\mathcal{Z}}(A)^2\}$$

eine genaue obere Schranke für $\lceil\mathrm{H}(A)\rceil$.

3.2.2 Nichtquadratische Matrizen

3.2.2.1 Definition und Eigenschaften

Ein naheliegender Ansatz, um auch im allgemeinen Fall eine zur Hadamard-Schranke analoge Konstruktion zu erhalten, besteht darin, alle quadratischen Untermatrizen maximaler Größe zu berücksichtigen.

Definition 3.43 (Verallgemeinerte Hadamard-Schranke)
Wir notieren für ein $A \in \mathbb{Z}^{m\times n}$ mit $s = \min\{m,n\}$ durch

$$\mathcal{M}(A) \;:=\; \{M \in \mathbb{Z}^{s\times s} \mid M \textit{ ist Untermatrix von } A\}$$

die Menge der maximalen quadratischen Untermatrizen von A und nennen

$$\mathrm{H}^{\Box}(A) \;:=\; \min\{\max\{\mathrm{H}_{\mathcal{S}}(M) \mid M \in \mathcal{M}(A)\}\,,\, \max\{\mathrm{H}_{\mathcal{Z}}(M) \mid M \in \mathcal{M}(A)\}\}$$

verallgemeinerte Hadamard-Schranke von A.

Die alte Definition ist in der neuen enthalten, denn für ein $A \in \mathbb{Z}^{n\times n}$ gilt $\mathrm{H}^{\Box}(A) = \mathrm{H}(A)$.

Lemma 3.43 (Unterdeterminanten beliebiger Matrizen)
Es sei $A \in \mathbb{Z}^{m\times n}$ und $M \in \mathbb{Z}^{k\times k}$ eine Untermatrix von A. Dann ist

$$|\det(M)| \le \mathrm{H}^{\Box}(A)$$

Beweis: Wir wählen ein $\hat{M} \in \mathcal{M}(A)$, welches M vollständig enthält, und folgern

$$\mathrm{H}_{\mathcal{S}}(M) \le \mathrm{H}_{\mathcal{S}}(\hat{M}) \le \max\{\mathrm{H}_{\mathcal{S}}(B) \,|\, B \in \mathcal{M}(A)\}$$

bzw.

$$\mathrm{H}_{\mathcal{Z}}(M) \le \mathrm{H}_{\mathcal{Z}}(\hat{M}) \le \max\{\mathrm{H}_{\mathcal{Z}}(B) \,|\, B \in \mathcal{M}(A)\}$$

Andererseits gilt aber

$$|\det(A)| \le \mathrm{H}_{\mathcal{R}}(M) \quad \text{für } \mathcal{R} \in \{\mathcal{S}, \mathcal{Z}\}$$

woraus wir

$$|\det(M)| \le \mathrm{H}^{\Box}(A)$$

ableiten können. ∎

3.2.2.2 Ansatz für eine Näherung

Wir formulieren zunächst eine zum Korollar 3.2 analoge Aussage.

Korollar 3.3 (Theoretische Approximation der Hadamard-Schranke)
Sei $A \in \mathbb{Z}^{m\times n}$. Dann ist

$$\mathrm{H}^{\Diamond}(A) := \langle \mathrm{H}^{\Box}(A)^2 \rangle^{\frac{1}{2}}$$

mit

$$\mathrm{H}^{\Box}(A)^2 = \min\{\max\{\mathrm{H}_{\mathcal{S}}(M)^2 \,|\, M \in \mathcal{M}(A)\}, \max\{\mathrm{H}_{\mathcal{Z}}(M)^2 \,|\, M \in \mathcal{M}(A)\}\}$$

eine genaue obere Schranke für $\lceil \mathrm{H}^{\Box}(A) \rceil$.

Der Nachteil der neuen Approximation $\mathrm{H}^{\Diamond}(A)$ gegenüber der bisherigen Näherung $\mathrm{H}^*(A)$ ist nun, daß wir die Menge $\mathcal{M}(A)$ nicht kennen, und da diese sehr viele Elemente enthält, es auch nicht sinnvoll ist, diese vorab zu berechnen. Wir werden daher eine für praktische Zwecke ausreichende Näherung für $\mathrm{H}^{\Diamond}(A)$ einführen, die wir dann der Einfachheit halber wieder mit $\mathrm{H}^*(A)$ bezeichnen. Dabei gehen wir so vor, daß wir im Spezialfall einer quadratischen Matrix gerade den Wert aus Korollar 3.1 erhalten, und insofern kein Konflikt entsteht.

3.2.2.3 Approximation der verallgemeinerten Hadamard-Schranke

Voraussetzung :
Es sei $A \in \mathbb{Z}^{m\times n}$ und $m \le n$. Andernfalls betrachten wir A^T.

Vorgehensweise :

1. Spaltenweise Analyse :
   Die Matrix A habe die Spalten $a_1, \ldots, a_n \in \mathbb{Z}^m$. Zuerst bestimmen wir

$$h \;=\; \left(\|a_1\|_2^2, \ldots, \|a_n\|_2^2\right)^T \in \mathbb{N}_0$$

   und suchen dann einen maximalen Teilvektor

$$h_{teil} \;=\; (h_1^*, \ldots, h_m^*)^T \in \mathbb{N}_0^m$$

   der Länge m von h. Aus der Definition von $\mathrm{H}_{\mathcal{S}}$ folgern wir

$$\max\{\mathrm{H}_{\mathcal{S}}(M)^2 \mid M \in \mathcal{M}(A)\} \;=\; \prod_{j=1}^{m} h_j^*$$

   und setzen

$$columns \;:=\; \prod_{j=1}^{m} h_j^*$$

2. Zeilenweise Analyse :
   Wir betrachten nun die Zeilen $\hat{a}_1, \ldots, \hat{a}_m \in \mathbb{Z}^n$ von A und berechnen für jedes $\hat{a}_i$ einen maximalen Teilvektor $\hat{a}_{i,teil}$ der Länge m. Dann ist

$$\max\{\mathrm{H}_{\mathcal{Z}}(M)^2 \mid M \in \mathcal{M}(A)\} \;\leq\; \prod_{i=1}^{m} \|\hat{a}_{i,teil}\|_2^2$$

   weswegen wir

$$rows \;:=\; \prod_{i=1}^{m} \|\hat{a}_{i,teil}\|_2^2$$

   wählen.

3. Bestimmung von $\mathrm{H}^*(A)$:
   Als Approximation für die verallgemeinerte Hadamard-Schranke nehmen wir

$$\mathrm{H}^*(A) \;:=\; \langle \min\{columns, rows\}\rangle^{\frac{1}{2}}$$

Wir geben als Abschluß noch eine für fast alle weiteren Laufzeitanalysen grundlegende Aussage über die Größenordnung der Näherung $\mathrm{H}^*(A)$ an.

Korollar 3.4 *Es sei $A \in \mathbb{Z}^{m\times n}$. Dann gilt mit $s = \min\{m, n\}$*

$$\mathrm{H}^{\square}(A) \;\leq\; \mathrm{H}^*(A) \;=\; O((s|A|)^s)$$

bzw.

$$\log_2 \mathrm{H}^{\square}(A) \;\leq\; \log_2 \mathrm{H}^*(A) \;=\; O(s \cdot \log(s|A|))$$

3.2.3 Algorithmus

Algorithmus 3.19 (Approximation der Hadamard-Schranke)

Eingabe : $A \in \mathbb{Z}^{m\times n}$
Ausgabe : $\mathrm{H}^*(A) \in \mathbb{N}_0$

```
(* Überprüfe die Form von A *)
if ( m > n ) then A := A^T fi
(* Spaltenweise Analyse *)
for ( j := 1 ; j ≤ n ; j := j + 1 )
  do
    h[j] := 0
    for ( i := 1 ; i ≤ m ; i := i + 1 )
      do
        h[j] := h[j] + A[i,j]*A[i,j]
      od
  od
if ( m ≠ n )
  then hteil := maximaler Teilvektor von h der Länge m
  else  hteil := h
fi
columns := 1
for ( j := 1 ; j ≤ m ; j := j + 1 )
  do
    columns := columns*hteil[j]
  od
(* Zeilenweise Analyse *)
for ( i := 1 ; i ≤ m ; i := i + 1 )
  do
    a_i := A[i,*]
    if ( m ≠ n )
      then â_i := maximaler Teilvektor von a_i der Länge m
      else  â_i := a_i
    fi
    h[i] := 0
    for ( j := 1 ; j ≤ m ; j := j + 1 )
      do
        h[i] := h[i] + â_i[j]*â_i[j]
      od
  od
rows := 1
for ( i := 1 ; i ≤ m ; i := i + 1 )
  do
    rows := rows*h[i]
  od
(* Bestimmung von H*(A) *)
M := min { columns , rows }
```

```
(42) B := #Bits(M) - 1
(43) E := ( B div 2 ) + 2
(44) H := 2^E - 1
(45) return (H)
```

3.2.4 Laufzeitanalyse

Lemma 3.44 (Approximation der Hadamard-Schranke)
Um für ein $A \in \mathbb{Z}^{m\times n}$ die Approximation $\mathrm{H}^*(A)$ zu berechnen, benötigen wir mit

$$s = \min\{m,n\} \qquad , \qquad t = \max\{m,n\}$$

einen Zeitaufwand der Größenordnung $O(s^2 t \cdot \log^2(s|A|))$.

Beweis: betrachte Algorithmus 3.19

1. Spaltenweise Analyse : (3)-(20)
   Die Bestimmung von h erfordert Zeit

   $$O(ts \cdot (\log^2(|A|) + \log(s|A|^2)))$$

   und für die Auswahl von *hteil* entsteht ein Aufwand von

   $$O((t + s \cdot \log(t)) \cdot \log(s|A|^2))$$

   Die Berechnung von *columns* verursacht noch Kosten

   $$O(s \cdot \log((s|A|^2)^s) \cdot \log(s|A|^2))$$

2. Zeilenweise Analyse : (21)-(39)
   Um die Teilvektoren $\hat{a}_i$ zu suchen, benötigen wir

   $$O(s \cdot (t + s \cdot \log(t)) \cdot \log(|A|))$$

   Zeit, und für die Bestimmung von h entstehen Kosten

   $$O(s^2 \cdot (\log^2(|A|) + \log(s|A|^2)))$$

   Die Berechnung von *rows* erfordert dann noch Zeit

   $$O(s \cdot \log((s|A|^2)^s) \cdot \log(s|A|^2))$$

3. Bestimmung von $\mathrm{H}^*(A)$: (40)-(44)
   Mit schneller Exponentiation verursacht dies Kosten $O(\log^2((s|A|^2)^s))$.

$\blacksquare$

3.3 Determinante

3.3.1 Problemstellung

Analog zur Definition über Körpern erklären wir nun die Determinante über $\mathbb{Z}$.

Definition 3.44 (Determinante über $\mathbb{Z}$)
1. Als Determinante einer Matrix $A = (a_{i,j}) \in \mathbb{Z}^{n\times n}$ bezeichnen wir die Summe

$$\det(A) := \sum_{\sigma \in \mathcal{S}_n} \left(\operatorname{sign}_{\mathbb{Q}}(\sigma) \cdot \prod_{i=1}^{n} a_{i,\sigma(i)} \right)$$

2. Für einen Endomorphismus $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ definieren wir die Determinante von φ durch

$$\det(\varphi) := \det(\mathrm{Mat}(\varphi))$$

Aufgabenstellung :
Berechne die Determinante von $A \in \mathbb{Z}^{n\times n}$.

3.3.2 Berechnungsverfahren

Im Kapitel 1.13.1 über modulare Arithmetik in $\mathbb{Z}$ hatten wir als Beispiel die Berechnung der Determinante angeführt. Wir präzisieren dies im folgenden, wobei wir das Aufstellen der Kongruenzbedingungen und die Angabe einer oberen Schranke trennen.

3.3.2.1 Aufstellen der Kongruenzbedingungen

Zuerst formulieren wir ein grundlegendes Lemma.

Lemma 3.45 *Es seien $p \in \mathbb{P}$ und $A, B \in \mathbb{Z}^{n\times n}$ zwei Matrizen mit*

$$A \equiv B \mod p \quad \textit{(komponentenweise)}$$

Dann gilt auch

$$\det(A) \equiv \det(B) \mod p$$

Beweis: Folgt aus den Rechenregeln für Kongruenzen. ■

Um den Chinesischen Restsatz zur Berechnung der Determinante einer Matrix A zu verwenden, benötigen wir für gewisse Primzahlen p_i Kongruenzbedingungen der Form

$$\det(A) \equiv D(p_i) \mod p_i$$

Wie finden wir diese ? Zunächst wissen wir aus dem vorhergehenden Lemma 3.45, daß

$$\det(A) \equiv \det(A \mod p_i) \mod p_i$$

gilt. Dabei entstehe die Matrix

$$(A \mod p_i) \in \mathbb{F}_{p_i}^{n \times n}$$

aus A durch komponentenweises Reduzieren modulo p_i. Also rechnen wir

$$\det(A \mod p_i) \in \mathbb{F}_{p_i}$$

wie in Teil II beschrieben aus und setzen

$$D(p_i) := \det(A \mod p_i) \in \mathbb{F}_{p_i}$$

Dies tun wir für jedes p_i und erhalten so die gesuchten Bedingungen.

3.3.2.2 Größe der Zahlen

Eine weitere Voraussetzung ist, eine geeignete Abschätzung der Form

$$-\mathrm{S}(A) \leq \det(A) \leq +\mathrm{S}(A)$$

zu finden. Im vorangegangenen Kapitel haben wir dieses Problem sowohl prinzipiell durch Einführung der Hadamard-Schranke als auch algorithmisch durch eine Approximation gelöst. Die gesuchte Bedingung lautet also

$$-\mathrm{H}^*(A) \leq \det(A) \leq +\mathrm{H}^*(A)$$

3.3.3 Algorithmus

Algorithmus 3.20 (Determinante über $\mathbb{Z}$)

Eingabe : $A \in \mathbb{Z}^{n \times n}$
Ausgabe : $\det(A) \in \mathbb{Z}$

Vorgehen :

1. Berechne die Approximation $\mathrm{H}^*(A)$ der Hadamard-Schranke von A
2. Bestimme eine Liste $p_1, \ldots, p_k \in \mathbb{P}$ von Primzahlen mit

$$p_1 \cdot \ldots \cdot p_k > 2 \cdot \mathrm{H}^*(A)$$

3. Berechne für jede dieser Primzahlen

$$D(p_i) := \det(A \mod p_i) \in \mathbb{F}_{p_i}$$

4. Ermittele mit dem Chinesischen Restsatz eine Zahl $X^* \in \mathbb{Z}$ mit

$$-\mathrm{H}^*(A) \;\le\; X^* \;\le\; +\mathrm{H}^*(A)$$

und

$$X^* \;\equiv\; D(p_i) \mod p_i \qquad \forall i \in \{1, \ldots, k\}$$

5. Setze $\det(A) := X^*$

3.3.4 Laufzeitanalyse

3.3.4.1 Ergebnis für Matrizen

Satz 3.1 (Determinante einer Matrix über $\mathbb{Z}$)
Die Berechnung der Determinante von $A \in \mathbb{Z}^{n\times n}$ erfordert Zeit $O(n^4 \cdot \log^2(n|A|))$.

Beweis: betrachte Algorithmus 3.20

1. Die ersten beiden Schritte kosten zusammen Zeit $O(n^3 \cdot \log^2(n|A|))$.
2. Um den Aufwand für Schritt 3 abzuschätzen, benötigen wir die folgenden Aussagen :
   1. Die Primzahlen p_i sind beschränkt durch

   $$p_i \;\le\; \lceil 2 \cdot \log_2(2 \cdot \mathrm{H}^*(A)) \rceil \;=\; O(\log \mathrm{H}^*(A))$$

   2. Die Anzahl k der Primzahlen ist

   $$k \;=\; O\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)}\right)$$

   3. Der gesamte Aufwand ist also

   $$O\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)} \cdot n^3 \cdot \log^2 \log \mathrm{H}^*(A)\right)$$

   und mit

   $$\log \mathrm{H}^*(A) \;=\; O(n \cdot \log(n|A|))$$

   erhalten wir

   $$O(n^4 \cdot \log(n|A|) \cdot \log(n \cdot \log(n|A|)))$$

3. Die Rekonstruktion von X^* erfordert noch Zeit

$$O\left(\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)}\right)^2 \cdot \log^2 \log \mathrm{H}^*(A)\right) \;=\; O(n^2 \cdot \log^2(n|A|))$$

■

3.3.4.2 Ergebnis für Homomorphismen

Satz 3.2 (Determinante eines Endomorphismus über $\mathbb{Z}$)
Die Bestimmung der Determinante von $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ kostet Zeit $O(n^4 \cdot \log^2(n|\varphi|))$.

3.4 Adjungierter Homomorphismus

3.4.1 Problemstellung

Definition 3.45 (Regulärer Endomorphismus über $\mathbb{Z}$)
1. Wir notieren die Menge der regulären Matrizen über $\mathbb{Z}$ durch

$$\mathrm{GL}(n, \mathbb{Z}) \;:=\; \{A \in \mathbb{Z}^{n\times n} \mid \det(A) \neq 0\}$$

2. Ein Endomorphismus $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ heißt regulär, wenn $\mathrm{Mat}(\varphi) \in \mathrm{GL}(n, \mathbb{Z})$ ist.

Für solche Endomorphismen wollen wir eine Art *Umkehrabbildung* konstruieren.

Lemma 3.46 (Adjungierter Homomorphismus über $\mathbb{Z}$)
Sei $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ ein regulärer Endomorphismus. Dann gibt es genau einen ebenfalls regulären Endomorphismus $\mathrm{adj}(\varphi) \in \mathrm{End}(\mathbb{Z}^n)$, welcher

$$\varphi \circ \mathrm{adj}(\varphi) \;=\; \mathrm{adj}(\varphi) \circ \varphi \;=\; \det(\varphi) \cdot \mathrm{id}_{\mathbb{Z}^n}$$

erfüllt. Wir nennen $\mathrm{adj}(\varphi)$ den adjungierten Homomorphismus von φ.

Wenden wir auf das obige Lemma die Darstellungsabbildung Mat an, erhalten wir

$$\mathrm{Mat}(\varphi) \cdot \mathrm{Mat}(\mathrm{adj}(\varphi)) \;=\; \mathrm{Mat}(\mathrm{adj}(\varphi)) \cdot \mathrm{Mat}(\varphi) \;=\; \det(\varphi) \cdot I_n$$

Dies motiviert die folgende Definition.

Definition 3.46 (Adjungierte Matrix über $\mathbb{Z}$)
Als adjungierte Matrix von $A \in \mathbb{Z}^{n\times n}$ bezeichnen wir die Darstellung des adjungierten Homomorphismus von φ_A :

$$\mathrm{adj}(A) \;:=\; \mathrm{Mat}(\mathrm{adj}(\varphi_A)) \in \mathbb{Z}^{n\times n}$$

Dann ist

$$A \cdot \mathrm{adj}(A) \;=\; \mathrm{adj}(A) \cdot A \;=\; \det(A) \cdot I_n$$

Aufgabenstellung :
Berechne die adjungierte Matrix von $A \in \mathrm{GL}(n, \mathbb{Z})$.

3.4.2 Berechnungsverfahren

Wir erinnern uns an die in Kapitel 1.13.2 beschriebene modulare Arithmetik in $\mathbb{Z}^{n \times n}$: eine unbekannte Matrix wird durch komponentenweise Anwendung des Chinesischen Restsatzes bestimmt. Um mit dieser Methode die adjungierte Matrix zu berechnen, benötigen wir wieder Kongruenzbedingungen, deren Herleitung uns in den nächsten beiden Teilabschnitten beschäftigen wird. Wir gehen dabei wie folgt vor :

- zuerst erläutern wir die Bestimmung der adjungierten Matrix über einem Körper
- dann kombinieren wir diese Vorgehensweise mit modularer Arithmetik

Ferner geben wir eine geeignete Schranke für die Einträge in der Adjungierten an.

3.4.2.1 Betrachtung über Körpern

Lemma 3.47 (Adjungierte Matrix über Körpern)
Es sei $\mathbb{K}$ *ein Körper und* $A \in \mathrm{GL}(n, \mathbb{K})$. *Dann gibt es genau ein* $\mathrm{adj}(A) \in \mathrm{GL}(n, \mathbb{K})$ *mit*

$$\mathrm{adj}(A) \cdot A \;=\; A \cdot \mathrm{adj}(A) \;=\; \det(A) \cdot I_n$$

Als Grundlage für die Berechnung der Adjungierten über einem Körper $\mathbb{K}$ verwenden wir das folgende Korollar.

Korollar 3.5 *Für ein* $A \in \mathrm{GL}(n, \mathbb{K})$ *ist* $\mathrm{adj}(A) = \det(A) \cdot A^{-1}$

3.4.2.2 Aufstellen der Kongruenzbedingungen

Lemma 3.48 (Darstellung der Adjungierten über $\mathbb{Z}$)
Es sei $A \in \mathrm{GL}(n, \mathbb{Z})$ *und* $A_{i,j} \in \mathbb{Z}$ *das Komplement von* A *bezüglich* (i, j)

$$A_{i,j} \;:=\; (-1)^{i+j} \cdot \det S_{i,j}(A)$$

wobei

$$S_{i,j}(A) \in \mathbb{Z}^{(n-1)\times(n-1)}$$

aus A *durch Streichen der* i*-ten Zeile und der* j*-ten Spalte entsteht. Dann gilt*

$$\mathrm{adj}(A) \;=\; (A_{i,j})^T_{1 \le i,j \le n}$$

Beweis: siehe [Fischer] , Seite 152 , Satz 4.3.6 ■

Wir können mit Korollar 3.47 adjungierte Matrizen über Körpern ausrechnen, also z.B.

$$\operatorname{adj}(A \bmod p) \in \mathbb{F}_p^{n\times n}$$

für eine Primzahl p und ein $A \in \mathrm{GL}(n,\mathbb{Z})$. Um diesen Ansatz mit modularer Arithmetik in $\mathbb{Z}^{n\times n}$ zu kombinieren, benötigen wir noch Bedingungen der Form

$$\operatorname{adj}(A) \equiv A(p) \mod p$$

mit einer Matrix $A(p) \in \mathbb{Z}^{n\times n}$.

Lemma 3.49 *Es sei $A \in \mathrm{GL}(n,\mathbb{Z})$ und $p \in \mathbb{P}$ eine Primzahl mit $p \nmid \det(A)$. Dann ist*

$$\operatorname{adj}(A) \equiv \operatorname{adj}(A \bmod p) \mod p$$

Beweis: die Bedingung $p \nmid \det(A)$ garantiert uns

$$(A \bmod p) \in \mathrm{GL}(n,\mathbb{F}_p)$$

und mit Lemma 3.48 erhalten wir die Behauptung. ∎

3.4.2.3 Größe der Zahlen

Lemma 3.50 *Sei $A \in \mathrm{GL}(n,\mathbb{Z})$. Dann gilt*

$$|\operatorname{adj}(A)| \leq \mathrm{H}^*(A)$$

Beweis: Folgt aus Lemma 3.48 und Lemma 3.43 ∎

3.4.3 Algorithmus

Algorithmus 3.21 (Adjungierte Matrix über $\mathbb{Z}$)

Eingabe : $A \in \mathrm{GL}(n,\mathbb{Z})$
Ausgabe : $\operatorname{adj}(A) \in \mathrm{GL}(n,\mathbb{Z})$

Vorgehen :

1. Berechne die Approximation $\mathrm{H}^*(A)$ der Hadamard-Schranke von A
2. Bestimme die Determinante von A
3. Berechne eine Liste $p_1,\ldots,p_k \in \mathbb{P}$ von Primzahlen mit

$$p_1 \cdot \ldots \cdot p_k > 2 \cdot \mathrm{H}^*(A)$$

und

$$p_i \nmid \det(A) \qquad \forall i \in \{1,\ldots,k\}$$

4. Ermittele für jede dieser Primzahlen
   - die inverse Matrix von $(A \mod p_i)$ in $\mathbb{F}_{p_i}^{n\times n}$
   - die Determinante von $(A \mod p_i)$ in $\mathbb{F}_{p_i}$

   und setze
   $$A(p_i) := \det(A \mod p_i) \cdot (A \mod p_i)^{-1} \in \mathbb{F}_{p_i}^{n\times n}$$

5. Ermittele durch den Chinesischen Restsatz eine Matrix $A^* = (a^*_{i,j}) \in \mathbb{Z}^{n\times n}$ mit
   $$-\mathrm{H}^*(A) \leq a^*_{i,j} \leq +\mathrm{H}^*(A) \qquad \forall i,j \in \{1,\ldots,n\}$$
   und
   $$A^* \equiv A(p_i) \mod p_i \qquad \forall i \in \{1,\ldots,k\}$$

6. Setze $\mathrm{adj}(A) := A^*$

3.4.4 Laufzeitanalyse

3.4.4.1 Ergebnis für Matrizen

Satz 3.3 (Adjungierte Matrix über $\mathbb{Z}$)
Die Berechnung der adjungierten Matrix von $A \in \mathrm{GL}(n,\mathbb{Z})$ kostet Zeit $O(n^4 \cdot \log^2(n|A|))$

Beweis: betrachte Algorithmus 3.21

1. Die Schritte 1 und 3 erfordern zusammen Zeit $O(n^3 \cdot \log^2(n|A|))$
2. Für die Berechnung der Determinante von A benötigen wir Zeit $O(n^4 \cdot \log^2(n|A|))$
3. Den Aufwand für Schritt 4 schätzen wir wie folgt ab :
   1. Die Primzahlen p_i sind jeweils beschränkt durch
      $$p_i \leq \lceil 2 \cdot \log_2(2 \cdot \mathrm{H}^*(A)) \rceil = O(\log \mathrm{H}^*(A))$$
   2. Die Anzahl k der Primzahlen ist
      $$k = O\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)}\right)$$
   3. Der gesamte Aufwand ist also
      $$O\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)} \cdot n^3 \cdot \log^2 \log \mathrm{H}^*(A)\right)$$
      und mit
      $$\log \mathrm{H}^*(A) = O(n \cdot \log(n|A|))$$
      erhalten wir
      $$O(n^4 \cdot \log(n|A|) \cdot \log(n \cdot \log(n|A|))$$
4. Die Rekonstruktion von A^* erfordert noch Zeit
   $$O\left(n^2 \cdot \left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)}\right)^2 \cdot \log^2 \log \mathrm{H}^*(A)\right) = O(n^4 \cdot \log^2(n|A|))$$

■

3.4.4.2 Ergebnis für Homomorphismen

Satz 3.4 (Adjungierter Homomorphismus über $\mathbb{Z}$)
Um für ein reguläres $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ den adjungierten Homomorphismus zu bestimmen, benötigen wir einen Zeitaufwand der Größenordnung $O(n^4 \cdot \log^2(n|\varphi|))$.

3.5 Reguläres Urbild

3.5.1 Problemstellung

Definition 3.47 (Urbild eines Vektors über $\mathbb{Z}$)
Sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ und $b \in \mathbb{Z}^m$. Wir nennen einen Vektor $x \in \mathbb{Z}^n$ ein Urbild von b unter φ, wenn $\varphi(x) = b$ gilt. Dann ist $\mathrm{Mat}(\varphi) \cdot x = b$.

Die Berechnung von Urbildern entspricht also dem Lösen *linearer Gleichungssysteme* über dem Ring $\mathbb{Z}$. Eine interessante Variante besteht darin, im Fall $b \notin \mathrm{Bild}(\varphi)$ eine möglichst kleine Zahl $\theta \in \mathbb{N}$ und ein geeignetes $x \in \mathbb{Z}^n$ mit

$$\theta \cdot b \in \mathrm{Bild}(\varphi) \qquad , \qquad \varphi(x) = \theta \cdot b$$

zu suchen. Da dies im allgemeinen nicht immer möglich ist, betrachten wir zuerst nur den Fall regulärer Endomorphismen - dann ist die Existenz eines solchen θ gesichert.

Aufgabenstellung :
Gegeben seien eine Koeffizientenmatrix $A \in \mathrm{GL}(n, \mathbb{Z})$ und k rechte Seiten $b_1, \ldots, b_k \in \mathbb{Z}^n$. Bestimme Vektoren $x_1, \ldots, x_k \in \mathbb{Z}^n$ sowie minimale Vielfache $\theta_1, \ldots, \theta_k \in \mathbb{N}$ mit

$$A \cdot x_i \;=\; \theta_i \cdot b_i \qquad \forall i \in \{1, \ldots, k\}$$

3.5.2 Berechnungsverfahren

Wir betrachten für ein variables $\theta \in \mathbb{N}$ das Gleichungssystem

$$A \cdot x \;=\; \theta \cdot b \tag{3.17}$$

mit $A \in \mathrm{GL}(n, \mathbb{Z})$ und $b \in \mathbb{Z}^n$. Im Gegensatz zur Vorgehensweise über Körpern, die wir erst bei der Untersuchung des allgemeinen Urbildproblemes auf $\mathbb{Z}$ übertragen werden, lösen wir (3.17) einfach nach x auf :

$$\begin{array}{rrcl} & A \cdot x & = & \theta \cdot b \\ \Leftrightarrow & \mathrm{adj}(A) \cdot (A \cdot x) & = & \mathrm{adj}(A) \cdot (\theta \cdot b) \\ \Leftrightarrow & \det(A) \cdot x & = & \theta \cdot (\mathrm{adj}(A) \cdot b) \\ \Leftrightarrow & x & = & (\theta / \det(A)) \cdot (\mathrm{adj}(A) \cdot b) \end{array}$$

Dann berechnen wir für den Vektor

$$u = (u_1, \ldots, u_n)^T := \operatorname{adj}(A) \cdot b$$

den größten gemeinsamen Teiler seiner Einträge

$$g := \operatorname{ggT}(u_1, \ldots, u_n)$$

und bestimmen θ als

$$\theta := \left| \frac{\det(A)}{\operatorname{ggT}(g, \det(A))} \right|$$

Dies garantiert uns die Minimalität von θ sowie die Abschätzung

$$1 \leq \theta \leq |\det(A)|$$

3.5.3 Algorithmus

Algorithmus 3.22 (Reguläres Urbild über $\mathbb{Z}$)

Eingabe :

1. Koeffizientenmatrix $A \in \mathrm{GL}(n, \mathbb{Z})$
2. Rechte Seiten $b_1, \ldots, b_k \in \mathbb{Z}^n$

Ausgabe :

1. Vektoren $x_1, \ldots, x_k \in \mathbb{Z}^n$
2. Minimale Vielfache $\theta_1, \ldots, \theta_k \in \mathbb{N}$ mit $A \cdot x_i = \theta_i \cdot b_i$

Vorgehen :

1. Berechne die Determinante und die adjungierte Matrix von A
2. Bestimme die Vektoren

$$u_i := \operatorname{adj}(A) \cdot b_i \qquad \forall i \in \{1, \ldots, k\}$$

mit

$$u_i = (u_{1,i}, \ldots, u_{n,i})^T$$

3. Berechne die größten gemeinsamen Teiler

$$g_i := \operatorname{ggT}(u_{1,i}, \ldots, u_{n,i}) \qquad \forall i \in \{1, \ldots, k\}$$

und setze

$$\theta_i := \left| \frac{\det(A)}{\operatorname{ggT}(g_i, \det(A))} \right| \qquad \forall i \in \{1, \ldots, k\}$$

4. Ermittele dann als Lösungen

$$x_i := \frac{\theta_i}{\det(A)} \cdot u_i \qquad \forall i \in \{1, \ldots, k\}$$

3.5.4 Laufzeitanalyse

3.5.4.1 Ergebnis für Matrizen

Satz 3.5 (Reguläre lineare Gleichungssysteme über $\mathbb{Z}$)
Es seien eine Koeffizientenmatrix $A \in \mathrm{GL}(n, \mathbb{Z})$ sowie rechte Seiten $b_1, \ldots, b_k \in \mathbb{Z}^n$ mit

$$|b_i| \leq B \qquad \forall i \in \{1, \ldots, k\}$$

gegeben. Dann benötigen wir Zeit

$$O(n^3 \cdot (n+k) \cdot (\log(n|A|) + \log(B))^2)$$

um sowohl Vektoren $x_1, \ldots, x_k \in \mathbb{Z}^n$ als auch minimale Vielfache $\theta_1, \ldots, \theta_k \in \mathbb{N}$ mit

$$A \cdot x_i = \theta_i \cdot b_i \qquad \forall i \in \{1, \ldots, k\}$$

zu berechnen.

Beweis: betrachte Algorithmus 3.22

1. Der erste Schritt erfordert Zeit $O(n^4 \cdot \log^2(n|A|))$
2. Die Berechnung der Vektoren u_i kostet Zeit
$$\begin{aligned} & O(k \cdot (n^2 \cdot (\log((n|A|)^n) \cdot \log(B) + \log(n \cdot B \cdot (n|A|)^n)))) \\ = \; & O(n^3 \cdot k \cdot \log(B) \cdot \log(n|A|)) \end{aligned}$$
3. Für die Bestimmung aller θ_i entsteht ein Aufwand von
$$O(nk \cdot \log^2(B) + n^2 k \cdot \log^2(n|A|))$$
4. Um u_i eventuell durch $|\det(A)|$ zu dividieren, brauchen wir Zeit
$$O(n \cdot \log(n \cdot B \cdot (n|A|)^n) \cdot \log((n|A|)^n))$$
also insgesamt
$$O(n^2 \cdot k \cdot \log(n|A|) \cdot (\log(B) + n \cdot \log(n|A|)))$$ ■

3.5.4.2 Ergebnis für Homomorphismen

Satz 3.6 (Reguläres Urbild über $\mathbb{Z}$)
Es seien $\varphi \in \mathrm{End}(\mathbb{Z}^n)$ ein regulärer Endomorphismus und $b_1, \ldots, b_k \in \mathbb{Z}^n$ mit

$$|b_i| \leq B \qquad \forall i \in \{1, \ldots, k\}$$

Dann benötigen wir Zeit

$$O(n^3 \cdot (n+k) \cdot (\log(n|\varphi|) + \log(B))^2)$$

um Vektoren $x_1, \ldots, x_k \in \mathbb{Z}^n$ als auch minimale Vielfache $\theta_1, \ldots, \theta_k \in \mathbb{N}$ mit

$$\varphi(x_i) = \theta_i \cdot b_i \qquad \forall i \in \{1, \ldots, k\}$$

zu berechnen.

3.6 Charakteristisches Polynom

3.6.1 Problemstellung

Definition 3.48 (Charakteristisches Polynom über $\mathbb{Z}$)
Es sei X eine Unbestimmte über $\mathbb{Z}$ und $\mathbb{Z}[X]$ der Polynomring in X.

1. *Wir ordnen einer Matrix $A \in \mathbb{Z}^{n\times n}$ durch die Festlegung*

$$c_A(X) := \det(A - X \cdot I_n) \in \mathbb{Z}[X]$$

ein Polynom vom Grad n zu und nennen c_A das charakteristische Polynom von A.

2. *Entsprechend definieren wir*

$$c_\varphi(X) := \det(\varphi - X \cdot \mathrm{id}_{\mathbb{Z}^n}) \in \mathbb{Z}[X]$$

für einen Endomorphismus $\varphi \in \mathrm{End}(\mathbb{Z}^n)$.

Die Ähnlichkeit der obigen Definition mit derjenigen über Körpern ist insofern trügerisch, als daß weder das im Teil II vorgestellte Berechnungsverfahren in $\mathbb{Z}$ anwendbar ist, noch die Eigenschaften char. Polynom über Körpern alle übertragbar sind. Insbesondere ist der Ähnlichkeitsbegriff in $\mathbb{Z}$ anders definiert.

Aufgabenstellung :
Bestimme den Koeffizientenvektor

$$c = (c_n, \ldots, c_0)^T \in \mathbb{Z}^{n+1}$$

des charakteristischen Polynomes

$$c_A(X) = c_n \cdot X^n + \ldots + c_1 \cdot X + c_0$$

einer Matrix $A \in \mathbb{Z}^{n\times n}$.

3.6.2 Berechnungsverfahren

Wir wollen wie zuvor modulare Arithmetik zur Bestimmung des gesuchten Koeffizientenvektors c verwenden, diesmal allerdings in $\mathbb{Z}^n$. Hierfür benötigen wir wieder Kongruenzbedingungen der Form

$$c \equiv c(p_i) \mod p_i$$

für geeignete p_i, sowie eine obere Schranke für $|c|$.

3.6.2.1 Aufstellen der Kongruenzbedingungen

Mit dem im Teil II vorgestellten Berechnungsverfahren für das charakteristische Polynom über Körpern können wir zwar nicht direkt die Aufgabenstellung dieses Abschnittes lösen, aber wir können zum Beispiel das charakteristische Polynom von $(A \bmod p)$ über dem Körper $\mathbb{F}_p$ ausrechnen, wobei p wie gehabt eine passende Primzahl ist. Tun wir dies gleich für mehrere p_i, erhalten wir entsprechend mehrere Koeffizientenvektoren, aus denen wir mit dem Chinesischen Restsatz einen neuen Vektor rekonstruieren. Daß dieser neue Vektor gerade die Koeffizienten des eigentlich gesuchten Polynomes $c_A(X)$ enthält, sofern wir die Primzahlen p_i richtig wählen, zeigen wir im folgenden.

Lemma 3.51 *Es seien $A \in \mathbb{Z}^{n \times n}$ und $p \in \mathbb{P}$. Für die Koeffizientenvektoren*

$$c = (c_n, \ldots, c_0)^T \in \mathbb{Z}^{n+1}$$

und

$$c(p) = (c_n^p, \ldots, c_0^p)^T \in \mathbb{F}_p^{n+1}$$

der charakteristischen Polynome

$$c_A(X) = c_n \cdot X^n + \ldots + c_1 \cdot X + c_0 \in \mathbb{Z}[X]$$

sowie

$$c_{A \bmod p}(X) = c_n^p \cdot X^n + \ldots + c_1^p \cdot X + c_0^p \in \mathbb{F}_p[X]$$

gilt die Kongruenzbedingung

$$c \equiv c(p) \mod p$$

Beweis: Aus der Definition des char. Polynomes folgt

$$c_A(z) \equiv c_{A \bmod p}(z \bmod p) \mod p \qquad \forall z \in \mathbb{Z} \tag{3.18}$$

Wir nehmen an, es gibt einen Koeffizienten c_k in $c_A(X)$ mit

$$c_k \not\equiv c_k^p \mod p$$

Dann ist

$$c_A(X) - c_{A \bmod p}(X) \neq 0_{\mathbb{F}_p[X]} \qquad \text{(Differenzpolynom)}$$

Somit existiert ein $y \in \mathbb{F}_p$ mit

$$c_A(y) - c_{A \bmod p}(y) \neq 0_{\mathbb{F}_p}$$

bzw.

$$c_A(y) \not\equiv c_{A \bmod p}(y \bmod p) \mod p$$

was der Bedingung (3.18) widerspricht. ∎

3.6.2.2 Größe der Zahlen

Lemma 3.52 (Satz von Gerschgorin)
Sei $A = (a_{i,j}) \in \mathbb{C}^{n\times n}$ und $\lambda \in \mathbb{C}$ ein Eigenwert von A. Mit K_i bezeichnen wir den Kreis

$$K_i := \{z \in \mathbb{C} \mid |z - a_{i,i}| \le r_i\} \qquad , \qquad i \in \{1,\dots,n\}$$

mit Radius

$$r_i := \sum_{\substack{j=1\\ j\ne i}}^{n} |a_{i,j}|$$

in $\mathbb{C}$. Dann gilt

$$\lambda \in K_1 \cup \ldots \cup K_n$$

Wir geben nun eine Abschätzung für die Größe der Koeffizienten von c_A an.

Lemma 3.53 (Größe der Koeffizienten des charakteristischen Polynomes)
Es sei $A \in \mathbb{Z}^{n\times n}$ und

$$c_A(X) \;=\; c_n \cdot X^n + \ldots + c_1 \cdot X + c_0$$

Dann gilt

$$|c_i| \;\le\; (n|A|)^n \qquad \forall i \in \{0,\dots,n\}$$

Beweis: Nach dem Satz von Gerschgorin erfüllt jeder komplexex Eigenwert α von A die Abschätzung

$$|\alpha| \;\le\; n \cdot |A|$$

Seien nun $\alpha_1,\dots,\alpha_n$ die komplexen Eigenwerte von A, also die Nullstellen des charakteristischen Polynomes von A in $\mathbb{C}$. Dann gilt die Formel (vgl. [Buchm2])

$$c_i \;=\; (-1)^i \cdot \sum_{1\le k_1<\ldots<k_i\le n} \alpha_{k_1} \cdot \ldots \cdot \alpha_{k_i}$$

Hieraus folgt

$$|c_i| \;\le\; \binom{n}{i} \cdot (n|A|)^i \;\le\; n^{n-i} \cdot (n|A|)^i \;\le\; (n|A|)^n$$

■

3.6.3 Algorithmus

Algorithmus 3.23 (Charakteristisches Polynom über $\mathbb{Z}$)

Eingabe :
Matrix $A \in \mathbb{Z}^{n\times n}$

Ausgabe :
Koeffizientenvektor

$$c = (c_n,\dots,c_0)^T \in \mathbb{Z}^{n+1}$$

des charakteristischen Polynomes von A :

$$c_A(X) = c_n \cdot X^n + \ldots + c_1 \cdot X + c_0$$

Vorgehen :

1. Berechne $H = (n|A|)^n$ mit schneller Exponentiation
2. Bestimme eine Liste $p_1, \dots, p_k \in \mathbb{P}$ von Primzahlen mit
$$p_1 \cdot \ldots \cdot p_k > 2 \cdot H$$
3. Berechne für jede dieser Primzahlen den Koeffizientenvektor
$$c(p_i) = (c_n^{p_i}, \dots, c_0^{p_i})^T \in \mathbb{F}_{p_i}^{n+1}$$
des charakteristischen Polynomes von $(A \mod p_i)$:
$$c_{A \bmod p_i}(X) = c_n^{p_i} \cdot X^n + \ldots + c_0^{p_i} \in \mathbb{F}_{p_i}[X]$$
4. Ermittele mit dem Chinesischen Restsatz einen Vektor
$$c^* = (c_0^*, \dots, c_n^*)^T \in \mathbb{Z}^{n+1}$$
mit
$$-H \le c_i^* \le +H \qquad \forall i \in \{0, \dots, n\}$$
und
$$c^* \equiv c(p_i) \mod p_i \qquad \forall i \in \{1, \dots, k\}$$
5. Setze $c := c^*$

3.6.4 Laufzeitanalyse

3.6.4.1 Ergebnis für Matrizen

Satz 3.7 (Charakteristisches Polynom einer Matrix über $\mathbb{Z}$)
Die Berechnung des charakteristischen Polynomes einer Matrix $A \in \mathbb{Z}^{n \times n}$ erfordert einen Zeitaufwand der Größenordnung $O(n^4 \cdot \log^2(n|A|))$.

Beweis: betrachte Algorithmus 3.23

1. Die ersten beiden Schritte kosten zusammen Zeit $O(n^2 \cdot \log^2(n|A|))$
2. Den Zeitaufwand für Schritt 3 schätzen wir wie folgt ab :
   1. Die Primzahlen sind jeweils beschränkt durch
   $$p_i \le \lceil 2 \cdot \log_2(2 \cdot H) \rceil = O(\log H)$$
   2. Die Anzahl k der Primzahlen ist
   $$k = \left(\frac{\log H}{\log \log H} \right)$$

3. Der gesamte Aufwand ist also

$$O\left(\frac{\log H}{\log\log H}\cdot n^3\cdot\log^2\log H\right)$$

und mit

$$\log H \;=\; O(n\cdot\log(n|A|))$$

erhalten wir

$$O(n^4\cdot\log(n|A|)\cdot\log(n\cdot\log(n|A|)))$$

3. Die Rekonstruktion von c^* erfordert dann noch Zeit

$$O\left(n\cdot\left(\frac{\log H}{\log\log H}\right)^2\cdot\log^2\log H\right) \;=\; O(n^3\cdot\log^2(n|A|))$$

■

3.6.4.2 Ergebnis für Homomorphismen

Satz 3.8 (Charakteristisches Polynom eines Endomorphismus über $\mathbb{Z}$)
Um das charakteristische Polynom von $\varphi\in\mathrm{End}(\mathbb{Z}^n)$ zu berechnen, benötigen wir einen Zeitaufwand der Größenordnung $O(n^4\cdot\log^2(n|\varphi|))$.

3.7 Rang und linear unabhängige Zeilen

3.7.1 Problemstellung

Wir fassen zwei Probleme zusammen, die wir zunächst getrennt formuliert hatten :

- die Berechnung des Ranges von $\varphi\in\mathrm{Hom}(\mathbb{Z}^n,\mathbb{Z}^m)$
- die Bestimmung von Indizes $i_1,\ldots,i_r$ linear unabhängiger Zeilen aus $\mathrm{Mat}(\varphi)$, wobei $r=\mathrm{rang}(\varphi)$ sei

Zuerst präzisieren wir jedoch die verwendeten Begriffe.

Lemma 3.54 (Lineare Unabhängigkeit über $\mathbb{Z}$ und $\mathbb{Q}$)
Es seien $u_1,\ldots,u_l\in\mathbb{Z}^n$. Dann gilt :

$$u_1,\ldots,u_l \textit{ linear unabhängig über } \mathbb{Z} \quad\Leftrightarrow\quad u_1,\ldots,u_l \textit{ linear unabhängig über } \mathbb{Q}$$

Beweis: wir gehen von der Voraussetzung

$$\sum_{i=1}^{l}\alpha_i\cdot u_i = 0_{\mathbb{Z}^n}\ \text{ mit }\ \alpha_i\in\mathbb{Z}\quad\Rightarrow\quad\alpha_i=0_{\mathbb{Z}}\qquad\forall i\in\{1,\ldots,l\}$$

aus und zeigen

$$\sum_{i=1}^{l} \left(\frac{p_i}{q_i}\right) \cdot u_i = 0_{\mathbb{Q}^n} \text{ mit } p_i, q_i \in \mathbb{Z} \qquad \Rightarrow \qquad p_i = 0_{\mathbb{Z}} \qquad \forall i \in \{1, \ldots, l\}$$

Als erstes multiplizieren wir obigen Ausdruck mit

$$Q := q_1 \cdot \ldots \cdot q_l \neq 0_{\mathbb{Z}}$$

und erhalten

$$\sum_{i=1}^{l} \left(\frac{p_i \cdot Q}{q_i}\right) \cdot u_i = 0_{\mathbb{Z}^n} \qquad , \qquad \frac{p_i \cdot Q}{q_i} \in \mathbb{Z} \qquad \forall i \in \{1, \ldots, l\}$$

Daraus folgt dann

$$\frac{p_i \cdot Q}{q_i} = 0_{\mathbb{Z}} \qquad \Rightarrow \qquad p_i = 0_{\mathbb{Z}} \qquad \forall i \in \{1, \ldots, l\}$$

wegen $Q \neq 0_{\mathbb{Z}}$. ■

Dies erlaubt uns, eine Familie von Vektoren einfach als linear unabhängig zu bezeichnen.

Definition 3.49 (Rang einer Matrix über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m \times n}$. Wir definieren den Rang von A über $\mathbb{Z}$ als

$$\operatorname{rang}_{\mathbb{Z}}(A) := \operatorname{rang}(\varphi_A)$$

wobei $\varphi_A \in \operatorname{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ der A zugeordnete $\mathbb{Z}$-Modulhomomorphismus sei.

Der so erklärte Rangbegriff unterscheidet sich, sofern wir nur $\mathbb{Z}$ und $\mathbb{Q}$ betrachten, nicht von demjenigen über Körpern.

Lemma 3.55 *Für eine Matrix $A \in \mathbb{Z}^{m \times n}$ gilt*

$$\operatorname{rang}_{\mathbb{Q}}(A) = \operatorname{rang}_{\mathbb{Z}}(A)$$

Beweis: wir transformieren A mit einem $T \in \mathbb{Z}^{n \times n}$ unimodular auf Spaltenstufenform und zählen die Anzahl der von Null verschiedenen Spalten. ■

Da der Rang eines Homomorphismus φ gleich dem Rang seiner darstellenden Matrix ist, betrachten wir im folgenden nur noch Matrizen.

Aufgabenstellung :
Berechne für $A \in \mathbb{Z}^{m \times n}$ den Rang r sowie Indizes $i_1, \ldots, i_r$ linear unabhängiger Zeilen.

3.7.2 Berechnungsverfahren

Ebenso wie in den vorherigen Kapiteln ist auch hier das im Teil II vorgestellte Verfahren zur Berechnung linear unabhängiger Zeilen in $\mathbb{Z}$ nicht durchführbar. Wir wenden also aufs Neue modulare Arithmetik an - diesmal allerdings den **verallgemeinerten Ansatz**. Der Unterschied zur bisherigen Vorgehensweise besteht nun darin, daß wir die gesuchten Zeilenindizes als Information

$$\mathcal{I} = \{i_1, \ldots, i_r\}$$

über die Matrix A auffassen, die wir mit einer noch nicht näher beschriebenen Vorschrift $\mathcal{R}$ aus den entsprechenden Informationen

$$\mathcal{I}(p_i) = \{i_1(p_i), \ldots, i_r(p_i)\}$$

über die Matrizen (A mod p_i) rekonstruieren, wobei wir passende Primzahlen $p_1, \ldots, p_k$ verwenden :

$$\mathcal{I} := \mathcal{R}\left(\mathcal{I}(p_1), \ldots, \mathcal{I}(p_k)\right)$$

Die jeweiligen Indizes $i_1(p_i), \ldots, i_r(p_i)$ berechnen wir - wie in Teil II beschrieben - über den Körpern $\mathbb{F}_{p_i}$. Welche Gestalt $\mathcal{R}$ hat, und welche Primzahlen wir benötigen, wird uns der Chinesische Restsatz sagen.

3.7.2.1 Zeilenrangfunktion einer Matrix über $\mathbb{Z}$

Definition 3.50 *Sei $A \in \mathbb{Z}^{m\times n}$ mit den Zeilenvektoren $c_1, \ldots, c_m \in \mathbb{Z}^n$. Als* **Zeilenrangfunktion** *von A* **über $\mathbb{Z}$** *bezeichnen wir*

$$\mathrm{zr}_A^{\mathbb{Z}} : \{1, \ldots, m\} \to \mathbb{N}_0$$

mit

$$\mathrm{zr}_A^{\mathbb{Z}}(i) := \mathrm{rang}_{\mathbb{Z}}(c_i, c_{i+1}, \ldots, c_m)$$

Dann ist

$$\mathrm{zr}_A^{\mathbb{Z}}(1) = \mathrm{rang}_{\mathbb{Z}}(A)$$

sowie

$$\mathrm{zr}_A^{\mathbb{Z}}(i+1) \leq \mathrm{zr}_A^{\mathbb{Z}}(i) \qquad \forall i \in \{1, \ldots, m-1\}$$

Wir interessieren uns wieder für die Sprungstellen von $\mathrm{zr}_A^{\mathbb{Z}}$, also für Indizes i mit

$$\mathrm{zr}_A^{\mathbb{Z}}(i) > \mathrm{zr}_A^{\mathbb{Z}}(i+1)$$

Lemma 3.56 (Auswahl linear unabhängiger Zeilen über $\mathbb{Z}$)
Sei $A \in \mathbb{Z}^{m\times n}$ und $r = \mathrm{rang}_{\mathbb{Z}}(A)$. Die Zeilenrangfunktion $\mathrm{zr}_A^{\mathbb{Z}}$ hat genau r Sprungstellen

$$i_1, \ldots, i_r \in \{1, \ldots, m\}$$

mit

$$\mathrm{zr}_A^{\mathbb{Z}}(i_j) > \mathrm{zr}_A^{\mathbb{Z}}(i_j+1) \qquad \forall j \in \{1, \ldots, r\}$$

Insbesondere sind die Zeilen mit den Indizes $i_1, \ldots, i_r$ linear unabhängig.

Wir lösen die Aufgabenstellung also genauso, wie wir dies über Körpern getan haben.

3.7.2.2 Vergleich der Zeilenrangfunktionen über $\mathbb{Z}$ und $\mathbb{F}_p$

Wir können einer Matrix $A \in \mathbb{Z}^{m \times n}$ auch noch auf eine andere Art eine Zeilenrangfunktion zuordnen, indem wir $(A \mod p)$ über $\mathbb{F}_p$ für eine Primzahl p betrachten. Welcher Zusammenhang besteht nun zwischen dem Ansatz über $\mathbb{Z}$ und demjenigen über $\mathbb{F}_p$?

Lemma 3.57 *Es seien $A \in \mathbb{Z}^{m \times n}$ und $p \in \mathbb{P}$. Dann ist*

$$\mathrm{zr}^{\mathbb{F}_p}_{A \bmod p}(i) \leq \mathrm{zr}^{\mathbb{Z}}_A(i) \qquad \forall i \in \{1, \ldots, m\}$$

Beweis: Wir transformieren A mit $T \in \mathbb{Z}^{n \times n}$ unimodular auf Spaltenstufenform und betrachten jene Teilmatrix $A_i \in \mathbb{Z}^{i \times n}$ von A, welche aus den letzten i Zeilen besteht :

$$A_i \cdot T = \mathrm{STF}(A_i) \qquad (\text{in } \mathbb{Z})$$

bzw.

$$(A_i \mod p) \cdot (T \mod p) = (\mathrm{STF}(A_i) \mod p) \qquad (\text{in } \mathbb{F}_p)$$

Dabei beachten wir, daß über $\mathbb{F}_p$ nur noch $T \in \mathrm{GL}(n, \mathbb{F}_p)$ gilt. Durch die Reduktion modulo p wird nun die Anzahl der führenden Nullspalten in $\mathrm{STF}(A_i)$ nicht verringert, woraus

$$\mathrm{rang}_{\mathbb{F}_p}(\mathrm{STF}(A_i) \mod p) \leq \mathrm{rang}_{\mathbb{Z}}(\mathrm{STF}(A_i))$$

folgt. Wegen

$$T \in \mathrm{GL}(n, \mathbb{Z}) \qquad \text{bzw.} \qquad (T \mod p) \in \mathrm{GL}(n, \mathbb{F}_p)$$

gilt ferner

$$\mathrm{rang}_{\mathbb{F}_p}(A_i \mod p) = \mathrm{rang}_{\mathbb{F}_p}(\mathrm{STF}(A_i) \mod p)$$

sowie

$$\mathrm{rang}_{\mathbb{Z}}(A_i) = \mathrm{rang}_{\mathbb{Z}}(\mathrm{STF}(A_i))$$

und mit

$$\mathrm{zr}^{\mathbb{F}_p}_{A \bmod p}(i) = \mathrm{rang}_{\mathbb{F}_p}(A_i \mod p)$$

erhalten wir die Behauptung. ■

3.7.2.3 Idee für einen Algorithmus

Um die Anwendbarkeit des Lemmas 3.58 zu veranschaulichen, nehmen wir einmal an, wir hätten ein *Orakel* und wollen für eine Matrix $A \in \mathbb{Z}^{m \times n}$ die Zeilenrangfunktion über $\mathbb{Z}$ berechnen. Wir versuchen dies wie folgt :

- zuerst raten wir einige Primzahlen $p_1, \ldots, p_k$
- dann bestimmen wir durch Transformation auf Spaltenstufenform die Zeilenrangfunktionen der Matrizen $(A \mod p_i)$ über $\mathbb{F}_{p_i}$ und setzen

$$\mathrm{rate}(i) := \max\left\{\mathrm{zr}^{\mathbb{F}_{p_1}}_{A \bmod p_1}(i), \ldots, \mathrm{zr}^{\mathbb{F}_{p_k}}_{A \bmod p_k}(i)\right\}$$

Das vorhergehende Lemma 3.58 sagt uns nun, daß wir mit dieser Methode Glück haben können, also unsere Ratefunktion mit $\mathrm{zr}_A^{\mathbb{Z}}$ übereinstimmen kann. Sollten wir aber wirklich nur würfeln, wird dies wohl kaum der Fall sein. Wir geben daher im nächsten Abschnitt an, wie wir die Primzahlen wählen müssen, damit wir immer Glück haben, und bauen unser Rateverfahren zu einem Algorithmus aus.

3.7.2.4 Konstruktion der Zeilenrangfunktion über $\mathbb{Z}$

Lemma 3.58 *Es sei $A \in \mathbb{Z}^{m\times n}$ und $p_1, \ldots, p_k \in \mathbb{P}$ eine Liste von Primzahlen mit*

$$p_1 \cdot \ldots \cdot p_k > 2 \cdot \mathrm{H}^*(A)$$

Dann gibt es für jeden Zeilenindex $i \in \{1, \ldots, m\}$ mindestens eine Primzahl

$$p(i) \in \{p_1, \ldots, p_k\}$$

für die

$$\mathrm{zr}_A^{\mathbb{Z}}(i) \;=\; \mathrm{zr}_{A \bmod p(i)}^{\mathbb{F}_{p(i)}}(i)$$

gilt. Daraus folgt

$$\mathrm{zr}_A^{\mathbb{Z}}(i) \;=\; \max\left\{\mathrm{zr}_{A \bmod p_1}^{\mathbb{F}_{p_1}}(i), \ldots, \mathrm{zr}_{A \bmod p_k}^{\mathbb{F}_{p_k}}(i)\right\}$$

Beweis: wir bezeichnen wieder für ein $i \in \{1, \ldots, m\}$ mit $A_i \in \mathbb{Z}^{i\times n}$ jene Teilmatrix von A, welche aus den letzten i Zeilen besteht : nach Definition der Zeilenrangfunktion ist

$$\mathrm{rang}_{\mathbb{Z}}(A_i) \;=\; \mathrm{zr}_A^{\mathbb{Z}}(i) \;=:\; r(i)$$

und gemäß [Fischer] , Bemerkung 4.3.8 , Seite 154 gibt es dann eine reguläre Teilmatrix

$$M(i) \in \mathrm{GL}(r(i), \mathbb{Z})$$

von A_i. Die Bedingung

$$p_1 \cdot \ldots \cdot p_k > 2 \cdot \mathrm{H}^*(A)$$

garantiert uns zunächst

$$\det(M(i)) \not\equiv 0 \mod p(i)$$

für mindestens ein $p(i) \in \{p_1, \ldots, p_k\}$, denn wegen $|\det(M(i))| \leq \mathrm{H}^*(A)$ müßte sonst $\det(M(i)) = 0$ sein (Chinesischer Restsatz). Also gilt

$$(M(i) \bmod p(i)) \in \mathrm{GL}(r(i), \mathbb{F}_{p(i)})$$

woraus umgekehrt

$$\mathrm{rang}_{\mathbb{F}_{p(i)}}(A_i \bmod p(i)) \;=\; r(i)$$

folgt. Dies heißt aber gerade

$$\mathrm{zr}_{A \bmod p(i)}^{\mathbb{F}_{p(i)}}(i) \;=\; r(i)$$

■

Auch wenn uns das Lemma 3.58 zumindest prinzipiell ein Verfahren liefert, um für eine Matrix A die Zeilenrangfunktion zu ermitteln, bleibt noch ein Detail zu klären : der im Teil II angegebene Algorithmus 2.15 zur Bestimmung linear unabhängiger Zeilen über Körpern berechnet nicht die vollständige Wertetabelle von

$$\mathrm{zr}^{\mathbb{F}_p}_{A \bmod p} : \{1, \ldots, m\} \to \mathbb{N}_0$$

sondern nur Sprungstellen

$$i_1(p), \ldots, i_r(p) \in \{1, \ldots, m\}$$

mit

$$\mathrm{zr}^{\mathbb{F}_p}_{A \bmod p}(i_j(p)) > \mathrm{zr}^{\mathbb{F}_p}_{A \bmod p}(i_j(p)+1)$$

Also müssen wir uns noch überlegen, wie wir die gesuchten Indizes

$$i_1, \ldots, i_r \in \{1, \ldots, m\}$$

mit

$$\mathrm{zr}^{\mathbb{Z}}_{A}(i_j) > \mathrm{zr}^{\mathbb{Z}}_{A}(i_j+1)$$

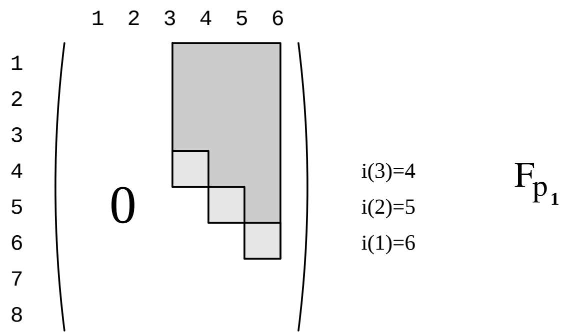
durch eine geeignete Rekonstruktionsvorschrift $\mathcal{R}$ aus den Werten

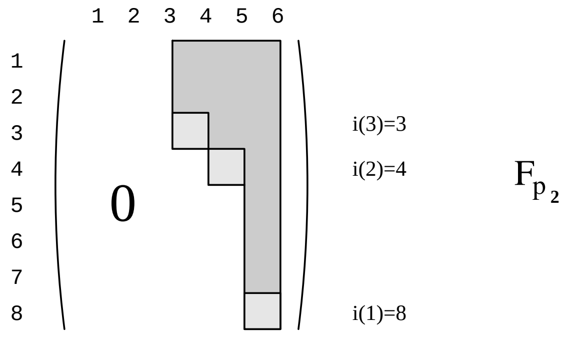
$$i_1(p_l), \ldots, i_r(p_l) \qquad , \qquad l \in \{1, \ldots, k\}$$

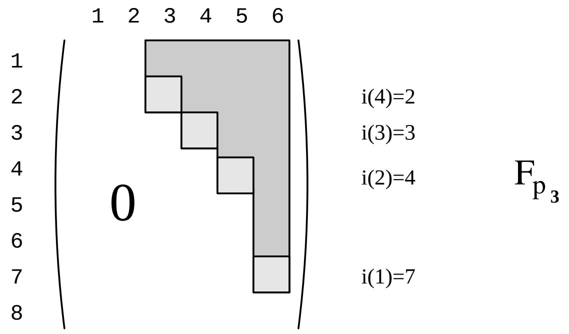
zusammenzusetzen. Dabei seien $p_1, \ldots, p_k$ die gemäß Lemma 3.58 gewählten Primzahlen.

3.7.2.5 Grafische Darstellung der Rekonstruktionsvorschrift

Wir betrachten ein einfaches Beispiel. Es sei $A \in \mathbb{Z}^{8\times 6}$ eine Matrix, für die wir über $\mathbb{F}_{p_1}, \mathbb{F}_{p_2}$ und $\mathbb{F}_{p_3}$ etwa folgende Spaltenstufenformen erhalten haben :







Das Lemma 3.58 besagt nun **anschaulich**, daß wir die Zeilenrangfunktion von A über $\mathbb{Z}$ und somit den *Treppenverlauf* in einer Spaltenstufenform von A erhalten, indem wir die obigen drei Grafiken folienartig übereinanderlegen. Wie sieht dies rechnerisch aus ? Ein erstes Problem, das wir bewältigen müssen, ist die unterschiedliche Anzahl von Indizes in den drei Fällen : dies entspricht der Tatsache, daß

$$\operatorname{rang}_{\mathbb{F}_p}(A \mod p) < \operatorname{rang}_{\mathbb{Z}}(A)$$

gelten kann. Wir gleichen dies durch Ergänzen von Nullen aus, und schreiben die Werte in eine Tabelle :

l	$i_4(p_l)$	$i_3(p_l)$	$i_2(p_l)$	$i_1(p_l)$	$r(p_l)$
p_1	0	4	5	6	3
p_2	0	3	4	8	3
p_3	2	3	4	7	4
max	2	4	5	8	4

In der letzten Zeile stehen die Spaltenmaxima, welche den gesuchten Indizes sowie dem Rang von A entsprechen. Allgemein sieht die **Rekonstruktionsvorschrift** $\mathcal{R}$ also wie folgt aus :

1. Für $r(p_i) = \operatorname{rang}_{\mathbb{F}_{p_i}}(A \mod p_i)$ ist der Rang von A über $\mathbb{Z}$ gleich

$$r = \max\{r(p_1), \ldots, r(p_k)\}$$

2. Wir bezeichnen mit

$$i_1(p_l), \ldots, i_r(p_l) \qquad , \qquad l \in \{1, \ldots, k\}$$

die Sprungstellen von

$$\operatorname{zr}^{\mathbb{F}_{p_l}}_{A \bmod p_l} : \{1, \ldots, m\} \to \mathbb{N}_0$$

wobei wir

$$i_j(p_l) := 0 \quad \text{für} \quad j \in \{r(p_l)+1, \ldots, r\}$$

setzen. Dann gilt

$$i_j = \max\{i_j(p_1), \ldots, i_j(p_k)\} \qquad \forall j \in \{1, \ldots, r\}$$

3.7.3 Algorithmus

Algorithmus 3.24 (Rang und linear unabhängige Zeilen über $\mathbb{Z}$)

Eingabe :
Matrix $A \in \mathbb{Z}^{m \times n}$

Ausgabe :
Rang r von A sowie Indizes $i_1, \ldots, i_r$ linear unabhängiger Zeilen

Vorgehen :

1. Berechne die Approximation $\mathrm{H}^*(A)$ der Hadamard-Schranke von A
2. Ermittele eine Liste $p_1, \ldots, p_k \in \mathbb{P}$ von Primzahlen mit
$$p_1 \cdot \ldots \cdot p_k > 2 \cdot \mathrm{H}^*(A)$$
3. Berechne für jede dieser Primzahlen p_i den Rang
$$r(p_i) := \mathrm{rang}_{\mathbb{F}_{p_i}}(A \mod p_i)$$
sowie Indizes
$$i_1(p_i), \ldots, i_{r(p_i)}(p_i) \in \{1, \ldots, m\}$$
linear unabhängiger Zeilen aus $(A \mod p_i)$
4. Bestimme den Rang r von A als
$$r := \max\{r(p_1), \ldots, r(p_k)\}$$
5. Falls $r(p_i) < r$ für ein p_i gilt, setze
$$i_j(p_i) := 0 \qquad \text{für} \qquad j \in \{r(p_i)+1, \ldots, r\}$$
6. Berechne die Indizes $i_1, \ldots, i_r$ als
$$i_j := \max\{i_j(p_1), \ldots, i_j(p_k)\} \qquad \forall j \in \{1, \ldots, r\}$$

3.7.4 Laufzeitanalyse

3.7.4.1 Ergebnis für Matrizen

Satz 3.9 (Rang und linear unabhängige Zeilen einer Matrix über $\mathbb{Z}$)
Um den Rang r von $A \in \mathbb{Z}^{m \times n}$, sowie Indizes $i_1, \ldots, i_r$ linear unabhängiger Zeilen aus A zu berechnen, benötigen wir mit $s = \min\{m, n\}$ Zeit $O(s \cdot m^2 n \cdot \log^2(s|A|))$.

Beweis: betrachte Algorithmus 3.24

1. Die ersten beiden Schritte erfordern Zeit $O(s^2 t \cdot \log^2(s|A|))$, wobei $t = \max\{m, n\}$ sei.

2. Den Zeitaufwand für Schritt 3 schätzen wir wie folgt ab :

   1. Die Primzahlen p_i sind jeweils beschränkt durch

   $$p_i \leq \lceil 2 \cdot \log_2(2 \cdot \mathrm{H}^*(A)) \rceil \;=\; O(\log \mathrm{H}^*(A))$$

   2. Die Anzahl k der Primzahlen ist

   $$k = O\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)}\right)$$

   3. Der gesamte Aufwand ist also

   $$O\left(\frac{\log \mathrm{H}^*(A)}{\log\log \mathrm{H}^*(A)} \cdot m^2 n \cdot \log^2 \log \mathrm{H}^*(A)\right)$$

   und mit

   $$\log \mathrm{H}^*(A) = O(s \cdot \log(s|A|))$$

   erhalten wir

   $$O(s \cdot m^2 n \cdot \log(s|A|) \cdot \log(s \cdot \log(s|A|)))$$

3. Die Berechnung von r sowie $i_1, \ldots, i_r$ in den Schritten 4-6 erfordert noch Zeit

   $$O(kn \cdot \log(m)) \;=\; O(s \cdot m^2 n \cdot \log^2(s|A|))$$

   wobei k wie zuvor die Anzahl der Primzahlen sei. ■

3.7.4.2 Ergebnis für Homomorphismen

Satz 3.10 (Rang eines Homomorphismus über $\mathbb{Z}$)
Um den Rang von $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ zu berechnen, benötigen wir mit $s = \min\{m, n\}$ einen Zeitaufwand der Größenordnung $O(s \cdot m^2 n \cdot \log^2(s|\varphi|))$.

3.8 Linearkombinationen

3.8.1 Problemstellung

Als Abschluß von Teil III wenden wir uns noch der Frage zu, wie teuer die Berechnung von Linearkombinationen über $\mathbb{Z}$ ist. Wir benötigen dies als Hilfsmittel in Kapitel 4.2.

Definition 3.51 (Linearkombinationen über $\mathbb{Z}$)
Es seien $v_1, \ldots, v_k \in \mathbb{Z}^n$. Wir nennen jedes $v \in \mathbb{Z}^n$ mit

$$v \;=\; \gamma_1 \cdot v_1 + \ldots + \gamma_k \cdot v_k \qquad , \qquad \gamma_1, \ldots, \gamma_k \in \mathbb{Z}$$

eine Linearkombination von $v_1, \ldots, v_k$ über $\mathbb{Z}$.

Eine naheliegende Vorgehensweise, um solche Terme auszuwerten, ist natürlich elementares Rechnen in $\mathbb{Z}^n$. Dies kostet uns Zeit

$$O(kn \cdot \log(\max |\gamma_i|) \cdot \log(\max(|v_i|))$$

Verwenden wir jedoch wieder den Chinesischen Restsatz, erhalten wir für den Fall, daß k im Vergleich zu $\max |\gamma_i|$ und $\max |v_i|$ klein ist, ein besseres Resultat.

Aufgabenstellung :
Bestimme $v = \gamma_1 \cdot v_1 + \ldots + \gamma_k \cdot v_k$ für $v_1, \ldots, v_k \in \mathbb{Z}^n$ und $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}$.

3.8.2 Berechnungsverfahren

Wir beachten, daß für eine Primzahl $p \in \mathbb{P}$

$$v \;=\; \sum_{j=1}^{k} \gamma_j \cdot v_j \;\equiv\; \sum_{j=1}^{k} (\gamma_j \mod p) \cdot (v_j \mod p) \mod p$$

gilt, und beobachten für

$$|v_i| \leq \delta \qquad \forall i \in \{1, \ldots, k\} \qquad , \qquad |\gamma_i| \leq \eta \qquad \forall i \in \{1, \ldots, k\}$$

die Abschätzung

$$|v| \;\leq\; k \cdot \delta \cdot \eta$$

Auf dieser Grundlage wenden wir dann wie bisher den Chinesischen Restsatz an.

3.8.3 Algorithmus

Algorithmus 3.25 (Linearkombinationen über $\mathbb{Z}$)

Eingabe :

1. Vektoren $v_1, \ldots, v_k \in \mathbb{Z}^n$ mit $|v_i| \leq \delta \quad \forall i \in \{1, \ldots, k\}$
2. Koeffizienten $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}$ mit $|\gamma_i| \leq \eta \quad \forall i \in \{1, \ldots, k\}$

Ausgabe :
Summe $v = \gamma_1 \cdot v_1 + \ldots + \gamma_k \cdot v_k$

Vorgehen :

1. Berechne die Schranke $H = k \cdot \delta \cdot \eta$
2. Bestimme eine Liste $p_1, \ldots, p_t \in \mathbb{P}$ von Primzahlen mit

$$p_1 \cdot \ldots \cdot p_t \;>\; 2 \cdot H$$

3. Berechne für jede dieser Primzahlen die Linearkombination

$$v(p_i) := \sum_{j=1}^{k} (\gamma_j \mod p_i) \cdot (v_j \mod p_i) \in \mathbb{F}_{p_i}^n$$

4. Ermittele mit dem Chinesischen Restsatz einen Vektor

$$v^* = (v_1^*, \ldots, v_n^*)^T \in \mathbb{Z}^n$$

mit

$$-H \leq v_i^* \leq +H \qquad \forall i \in \{1, \ldots, n\}$$

und

$$v^* \equiv v(p_i) \mod p_i \qquad \forall i \in \{1, \ldots, t\}$$

5. Setze $v := v^*$

3.8.4 Laufzeitanalyse

Lemma 3.59 (Linearkombinationen über $\mathbb{Z}$)
Es seien $v_1, \ldots, v_k \in \mathbb{Z}^n$ und $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}$ mit

$$|v_i| \leq \delta \qquad \forall i \in \{1, \ldots, k\} \qquad , \qquad |\gamma_i| \leq \eta \qquad \forall i \in \{1, \ldots, k\}$$

Dann benötigen wir, um die Linearkombination

$$v = \gamma_1 \cdot v_1 + \ldots + \gamma_k \cdot v_k$$

zu berechnen, einen Zeitaufwand der Größenordnung

$$O(\log(k \cdot \delta \cdot \eta) \cdot (kn \cdot \log\log(k \cdot \delta \cdot \eta) + \log(k \cdot \delta \cdot \eta)))$$

Beweis: betrachte Algorithmus 3.25

1. Die ersten beiden Schritte kosten Zeit $O(\log^2(k \cdot \delta \cdot \eta))$
2. Den Zeitaufwand für Schritt 3 schätzen wir wie folgt ab :
   1. Die Primzahlen p_i sind jeweils beschränkt durch

      $$p_i \leq \lceil 2 \cdot \log_2(2 \cdot k \cdot \delta \cdot \eta) \rceil = O(\log(k \cdot \delta \cdot \eta))$$

   2. Die Anzahl t der Primzahlen ist

      $$t = O\left(\frac{\log(k \cdot \delta \cdot \eta)}{\log\log(k \cdot \delta \cdot \eta)}\right)$$

   3. Die Reduktion der Vektoren v_i bzw. Koeffizienten γ_i kostet Zeit

      $$O(kn \cdot \log(\delta \cdot \eta) \cdot \log\log(k \cdot \delta \cdot \eta))$$

   4. Der gesamte Aufwand ist also

      $$O\left(\left(\frac{\log(k \cdot \delta \cdot \eta)}{\log\log(k \cdot \delta \cdot \eta)}\right) \cdot kn \cdot \log^2\log(k \cdot \delta \cdot \eta) + kn \cdot \log(\delta \cdot \eta) \cdot \log\log(k \cdot \delta \cdot \eta)\right)$$

3. Die Rekonstruktion von v^* erfordert noch Zeit

$$O\left(n \cdot \left(\frac{\log(k \cdot \delta \cdot \eta)}{\log\log(k \cdot \delta \cdot \eta)}\right) \cdot \log^2\log(k \cdot \delta \cdot \eta)\right)$$

■

Teil IV

Kern, Bild und Urbild über $\mathbb{Z}$

4.1 Kern und Bild

4.1.1 Problemstellung

Definition 4.52 (Kern und Bild über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m\times n}$. Dann bezeichnen wir

$$\begin{aligned}\mathrm{Kern}(A) &:= \{x \in \mathbb{Z}^n \mid A \cdot x = 0\}\\ \mathrm{Bild}(A) &:= \{y \in \mathbb{Z}^m \mid y = A \cdot x \textit{ für ein } x \in \mathbb{Z}^n\}\end{aligned}$$

als Kern bzw. Bild von A. Entsprechend definieren wir für $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$

$$\begin{aligned}\mathrm{Kern}(\varphi) &:= \mathrm{Kern}(\mathrm{Mat}(\varphi))\\ \mathrm{Bild}(\varphi) &:= \mathrm{Bild}(\mathrm{Mat}(\varphi))\end{aligned}$$

Beide Mengen sind freie Untermoduln von $\mathbb{Z}^n$ bzw. $\mathbb{Z}^m$.

Aufgabenstellung :
Berechne für $A \in \mathbb{Z}^{m\times n}$ Basen von $\mathrm{Kern}(A)$ *und* $\mathrm{Bild}(A)$.

4.1.2 Lösungsansatz

Bei der Betrachtung über Körpern hatten wir eine gegebene Matrix A unimodular auf Spaltenstufenform transformiert, um sowohl den Kern als auch das Bild von A zu berechnen. Im folgenden Lemma zeigen wir, daß diese Vorgehensweise zumindest prinzipiell auf $\mathbb{Z}$ übertragen werden kann.

Lemma 4.60 (Berechnung von Kern und Bild über $\mathbb{Z}$)
Sei $A \in \mathbb{Z}^{m\times n}$ und $T \in \mathbb{Z}^{n\times n}$ eine unimodulare Matrix, die A auf Spaltenstufenform transformiert :

$$T = (t_1, \ldots, t_n)$$

und

$$A \cdot T = \mathrm{STF}(A) = (0, \ldots, 0, s_1, \ldots, s_r)$$

mit

$$r = \mathrm{rang}(A)$$

Dann bilden die letzten r Spalten von $\mathrm{STF}(A)$ *eine Basis des Bildes und die ersten $n - r$ Spalten von T eine Basis des Kernes von A.*

Beweis: wir gliedern den Beweis in zwei Teile :

1. Es sei $y \in \text{Bild}(A \cdot T)$. Dann gilt :
$$y = (A \cdot T) \cdot x \text{ mit } x \in \mathbb{Z}^n$$
$$\Rightarrow y = (A \cdot T \cdot T^{-1}) \cdot (T \cdot x) \in \text{Bild}(A)$$
Umgekehrt läßt sich ein $\hat{y} \in \text{Bild}(A)$ schreiben als
$$\hat{y} = A \cdot \hat{x} \text{ mit } \hat{x} \in \mathbb{Z}^n$$
$$\Rightarrow \hat{y} = (A \cdot T) \cdot (T^{-1} \cdot x) \in \text{Bild}(A \cdot T)$$
denn T ist nach Voraussetzung unimodular.

2. Sei nun $A \cdot x = 0$ für ein $x \in \mathbb{Z}^n$. Wir zeigen, daß der Vektor x sich als Linearkombination der ersten $n - r$ Spalten von T mit ganzzahligen Koeffizienten darstellen läßt :
$$A \cdot x = 0$$
$$\Rightarrow (A \cdot T) \cdot (T^{-1} \cdot x) = 0$$
$$\Rightarrow \text{STF}(A) \cdot y = 0 \text{ mit } y = T^{-1} \cdot x \Leftrightarrow x = T \cdot y$$
Da die letzten r Spalten von $\text{STF}(A)$ linear unabhängig sind, müssen die entsprechenden Einträge in y jeweils 0 sein. Also folgt
$$x = \sum_{j=1}^{n-r} y_j t_j \quad \text{und} \quad y_{n-r+1} = \ldots = y_n = 0$$

■

Dies legt nahe, zur Berechnung einer solchen Transformation fast den selben Algorithmus zu verwenden, den wir für Körper formuliert haben. Dabei tritt jedoch das Problem auf, daß wir mit enorm großen Zwischenergebnissen rechnen müssen, denn jeder Eliminationsschritt kann im ungünstigsten Fall den größten Eintrag in der betrachteten Matrix quadrieren. Aus diesem Grund wählen wir eine andere Vorgehensweise, welche auf der Hermite-Normalform sowie den im TeilIII vorgestellten Algorithmen aufbaut, und in deren Mittelpunkt die Aussage des nächsten Lemmas steht.

Lemma 4.61 (Transformation auf Spaltenstufenform über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m \times n}$ und r der Rang von A. Wir bezeichnen mit $i_1, \ldots, i_r$ die Sprungstellen der Zeilenrangfunktion
$$\text{zr}_A^{\mathbb{Z}}(i_j) > \text{zr}_A^{\mathbb{Z}}(i_j + 1) \qquad \forall j \in \{1, \ldots, r\}$$
und fassen die Zeilen aus A mit den Indizes i_j zu einer Matrix $\mathcal{ZB}(A)$ zusammen :
$$\mathcal{ZB}(A) = \left(\hat{a}_{i_r}^T, \ldots, \hat{a}_{i_1}^T\right)^T$$
Dabei sei $\hat{a}_i$ der i-te Zeilenvektor von A. Kennen wir eine unimodulare Matrix $T \in \mathbb{Z}^{n \times n}$, die $\mathcal{ZB}(A)$ auf Spaltenstufenform transformiert
$$\mathcal{ZB}(A) \cdot T \ = \ \text{STF}(\mathcal{ZB}(A))$$
so können wir auch die Ausgangsmatrix A mit T auf Spaltenstufenform transformieren :
$$A \cdot T \ = \ \text{STF}(A)$$

Beweis: es bezeichne $i \notin \{i_1, \ldots, i_r\}$ den Index einer Zeile aus A, die nicht in $\mathcal{ZB}(A)$ enthalten ist. Wir unterscheiden zwei Fälle :

1. $i_{j+1} < i < i_j$ für ein $j \in \{1, \ldots, r-1\}$: dann ist $\hat{a}_i$ von den letzten j Zeilen der Matrix $\mathcal{ZB}(A)$ linear abhängig

$$\hat{a}_i = \sum_{k=1}^{j} \alpha_k \cdot \hat{a}_{i_k} \quad \text{mit} \quad \alpha_1, \ldots, \alpha_j \in \mathbb{Q}$$

   und es gilt

$$\hat{a}_i \cdot T \ = \ (\underbrace{0, \ldots, 0}_{n-j}, \gamma_j, \ldots, \gamma_1) \in \mathbb{Z}^{1 \times n}$$

   Die Anzahl der führenden Nullen bleibt also gleich.

2. $1 \leq i < i_r$: wir haben entsprechend

$$\hat{a}_i = \sum_{k=1}^{r} \alpha_k \cdot \hat{a}_{i_k} \quad \text{mit} \quad \alpha_1, \ldots, \alpha_r \in \mathbb{Q}$$

   bzw.

$$\hat{a}_i \cdot T \ = \ (\underbrace{0, \ldots, 0}_{n-r}, \gamma_r, \ldots, \gamma_1) \in \mathbb{Z}^{1 \times n}$$

   so daß auch hier die Anzahl der führenden Nullen konstant bleibt.

Die in der Matrix $\mathcal{ZB}(A) \cdot T$ gegebene Spaltenstufenform wird also nur entzerrt, aber nicht zerstört. ■

Wir sind der Lösung des Problemes, eine Matrix A unimodular auf Spaltenstufenform zu transformieren, insofern einen Schritt näher gekommen, als daß wir uns in der weiteren Untersuchung auf Matrizen beschränken können, die vollen Zeilenrang haben. In den folgenden Abschnitten stellen wir für diesen Spezialfall ein efiizientes Verfahren vor. Auf dieser Grundlage konstruieren wir dann anschließend Algorithmen zur Berechnung von Kern und Bild beliebiger Matrizen.

4.1.3 Hermite-Normalform

Bevor wir die Definition der Hermite-Normalform formulieren und einen Algorithmus beschreiben, um diese zu berechnen, führen wir den Begriff des Gitters ein. Dies mag umständlich erscheinen, hat aber den einfachen Grund, daß die HNF traditionell eher einen Bezug zu Gittern als zu Moduln hat. Wir weisen an dieser Stelle ferner darauf hin, daß die HNF nur eine von vielen möglichen Spaltenstufenformen einer Matrix ist, welche aber den Vorteil der Eindeutigkeit hat.

4.1.3.1 Gitter

Definition 4.53 (Gitter im $\mathbb{R}^n$)
Ein Gitter im $\mathbb{R}^n$ ist eine additive Untergruppe des $\mathbb{R}^n$, die als Punktemenge diskret ist, d.h. jede Kugel im $\mathbb{R}^n$ enthält nur endlich viele Punkte des Gitters.

Einen anschaulichen Zugang zu Gittern vermittelt das folgende Lemma.

Lemma 4.62 *Sei $\mathcal{L} \subseteq \mathbb{R}^n$ ein Gitter. Dann ist $\mathcal{L}$ von der Form*

$$\mathcal{L} = \mathbb{Z} \cdot b_1 \oplus \ldots \oplus \mathbb{Z} \cdot b_m$$

wobei die Vektoren $b_1, \ldots, b_m \in \mathbb{R}^n$ über $\mathbb{R}$ linear unabhängig sind, und die Addition

$$M_1 \oplus \ldots \oplus M_m := \left\{ x \in \mathbb{R}^n \;:\; x = \sum_{i=1}^{m} x_i \text{ mit } x_i \in M_i \right\}$$

für $M_1, \ldots, M_m \subseteq \mathbb{R}^n$ punkteweise erklärt ist. Das Tupel $(b_1, \ldots, b_m)$ heißt **Gitterbasis** *und ist bis auf unimodulare Transformationen von rechts eindeutig bestimmt.*

Wir ordnen Gittern ferner eine Determinante zu, wobei wir voraussetzen, daß wir mindestens eine Basis bereits kennen.

Lemma 4.63 (Gitterdeterminante) *Es sei $\mathcal{L} \subseteq \mathbb{R}^n$ ein Gitter und*

$$B = (b_1, \ldots, b_m) \in \mathbb{R}^{n \times m}$$

eine Basis von $\mathcal{L}$. Dann ist der Wert

$$\det(\mathcal{L}) := \sqrt[2]{|\det(B^T B)|}$$

von der gewählten Basis unabhängig. Wir nennen $\det(\mathcal{L})$ Gitterdeterminante von $\mathcal{L}$.

Nach dieser etwas allgemeineren Einführung spezialisieren wir den Gitterbegriff auf $\mathbb{Z}^n$.

Lemma 4.64 *Es sei $A \in \mathbb{Z}^{m \times n}$. Dann ist die Menge*

$$\mathcal{L}(A) := \mathrm{Bild}(A) \subseteq \mathbb{Z}^m$$

ein Gitter im $\mathbb{R}^m$, so daß jede Basis des Bildes auch eine Gitterbasis ist und umgekehrt.

Dies erlaubt uns, für *ganzzahlige* Matrizen die Berechnung des Bildes auf die Bestimmung einer Gitterbasis zu reduzieren.

4.1.3.2 Problemstellung

Definition 4.54 (Hermite-Normalform über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m\times n}$ eine Matrix mit $\mathrm{rang}(A) = m$. A hat Hermite-Normalform, wenn gilt :

1. *A ist in Spaltenstufenform*
2. *Die Einträge hinter der Diagonalen sind reduziert :*

$$l^A(j) > 0 \text{ für ein } j \in \{1,\dots,n\}$$
$$\Rightarrow 0 \le A[l^A(j), q] < A[l^A(j), j] \qquad \forall q \in \{j+1,\dots,n\}$$

Das Bemerkenswerte an der Hermite-Normalform ist, daß sie nicht nur immer existiert, sondern sogar eindeutig ist.

Lemma 4.65 (Existenz und Eindeutigkeit der HNF)
Sei $A \in \mathbb{Z}^{m\times n}$ und $\mathrm{rang}(A) = m$. Dann gilt :

1. *Es gibt genau eine Matrix $\mathrm{HNF}(A) \in \mathbb{Z}^{m\times n}$, die Hermite-Normalform hat, und in die A durch eine uniodulare Transformation überführt werden kann :*

   $$A \cdot T \;=\; \mathrm{HNF}(A)$$

   Die Transformationsmatrix T ist nicht eindeutig bestimmt.

2. *Geht $B \in \mathbb{Z}^{m\times n}$ aus A durch eine unimodulare Transformation hervor, hat B die selbe Hermite-Normalsform wie A.*

Beweis: Siehe [Gro/Lo/Sch], Seite 43-45 sowie 155-156 ■

Ein naheliegender Ansatz, um die Hermite-Normalform zu berechnen, ist natürlich, eine gegebene Matrix durch unimodulare Umformungen in Spaltenstufenform zu überführen, und dann die Nicht-Diagonaleinträge modulo der Diagonaleinträge zu reduzieren. Der Nachteil dabei ist, daß die auftretenden Zwischenergebnisse enorm groß werden können.

Beispiel 4.7 *Wir betrachten die Matrix*

$$A = \begin{pmatrix} 1 & 8 & 1 \\ 7 & -5 & 3 \\ 9 & 4 & 5 \end{pmatrix}$$

und transformieren diese elementar auf Spaltenstufenform :

$$A = \begin{pmatrix} 1 & 8 & 1 \\ 7 & -5 & 3 \\ 9 & \mathbf{4} & \mathbf{5} \end{pmatrix} \longmapsto \begin{pmatrix} 1 & 36 & -7 \\ 7 & -37 & 8 \\ \mathbf{9} & 0 & \mathbf{1} \end{pmatrix} \longmapsto \begin{pmatrix} 64 & 36 & -7 \\ \mathbf{-65} & \mathbf{-37} & 8 \\ 0 & 0 & 1 \end{pmatrix} \longmapsto$$

$$\begin{pmatrix} -\mathbf{28} & -4 & -7 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{pmatrix} \longmapsto \begin{pmatrix} 28 & -4 & -7 \\ 0 & 1 & 8 \\ 0 & 0 & 1 \end{pmatrix}$$

Anschließend reduzieren wir die Nicht-Diagonaleinträge :

$$\begin{pmatrix} 28 & -4 & -7 \\ 0 & 1 & \mathbf{8} \\ 0 & 0 & 1 \end{pmatrix} \longmapsto \begin{pmatrix} 28 & -\mathbf{4} & 25 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \longmapsto \begin{pmatrix} 28 & 24 & 25 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathrm{HNF}(A)$$

Das Bemerkenswerte dabei ist, daß alle auftretenden Zahlen erstaunlich *klein* sind :

- Hadamard-Schranke von A : $\mathrm{H}(A) \approx 693$
- Größtes Zwischenergebnis : 65

Einem Leser, dem nun die Notwendigkeit guter Schranken nicht plausibel ist, empfehlen wir als Kontrast hierzu [Hafn/Curl]: die Autoren zitieren eine Matrix $A \in \mathbb{Z}^{18\times 18}$ mit $\mathrm{H}(A) \approx 10^{33}$, bei der die Zwischenergebnisse im Verlaufe der HNF-Berechnung in der Größenordnung 10^{5011} liegen. Dies belegt die Notwendigkeit, nach einem anderen Verfahren zu suchen. Bevor wir ein solches vorstellen, erläutern wir noch einige Eigenschaften der Hermite-Normalform.

4.1.3.3 Eigenschaften der HNF

Definition 4.55 (Wesentlicher Teil der HNF)
Es sei $A \in \mathbb{Z}^{m\times n}$ und $\mathrm{rang}(A) = m$. Als wesentlichen Teil der $\mathrm{HNF}(A)$ bezeichnen wir die quadratische Matrix $\mathrm{EHNF}(A) \in \mathrm{GL}(m, \mathbb{Z})$, welche die von 0 verschiedenen Spalten der $\mathrm{HNF}(A)$ enthält :

$$\mathrm{HNF}(A) \;=\; (0, \mathrm{EHNF}(A))$$

Die $\mathrm{EHNF}(A)$ hat immer obere Dreiecksgestalt.

Eine wichtige Eigenschaft der Hermite-Normalform ist, daß wir für deren Einträge gute Betragschranken zur Verfügung haben.

Lemma 4.66 (Größe der Zahlen in der HNF)
Sei $A \in \mathbb{Z}^{m\times n}$ und $\mathrm{rang}(A) = m$. Dann gilt

$$\det(\mathcal{L}(A)) \;=\; \det(\mathrm{EHNF}(A))$$

Ferner sind die Einträge in der $\mathrm{HNF}(A)$ beschränkt durch

$$|\mathrm{HNF}(A)| \;\leq\; \det(\mathcal{L}(A)) \;\leq\; H^{\square}(A)$$

Beweis: die erste Aussage ergibt sich unmittelbar aus der Definition der HNF. Wir beachten, daß die Determinante einer Teilmatrix $R \in \mathrm{GL}(m, \mathbb{Z})$ von A ein Vielfaches von $\det(\mathcal{L}(A))$ ist, denn für R existiert eine Darstellung

$$R = \mathrm{EHNF}(A) \cdot U \qquad , \qquad U \in \mathrm{GL}(m, \mathbb{Z})$$

Daraus folgern wir

$$\det(\mathcal{L}(A)) \leq |\det(R)| \leq H^{\square}(A)$$

und mit

$$\det(\mathcal{L}(A)) = h_{1,1} \cdot \ldots \cdot h_{m,m}$$

erhalten wir die Behauptung, wobei $h_{i,i}$ das i-te Diagonalelement in der EHNF(A) bezeichne. ∎

Wir stellen noch eine Strukturaussage für Gitter vor, die wir zur Formulierung des Algorithmus benötigen werden.

Definition 4.56 (HNF-Teiler einer Matrix)
Sei $A \in \mathbb{Z}^{m \times n}$ mit $\mathrm{rang}(A) = m$ und $\mathrm{EHNF}(A) = (h_{i,j})$. Dann sind für ein Vielfaches $h \in \mathbb{N}$ der Gitterdeterminanten $\det(\mathcal{L}(A))$ die Zahlen

$$\delta_0(h)\,,\ \delta_1(h)\,,\ \ldots\,,\ \delta_{m-1}(h) \in \mathbb{N}$$

mit

$$\delta_0(h) := h \qquad , \qquad \delta_{i+1}(h) := \frac{\delta_i(h)}{h_{m-i,m-i}} \qquad \forall i \in \{1, \ldots, m-1\}$$

eindeutig bestimmt. Wir nennen $\delta_i(h)$ den i-ten HNF-Teiler von A bezüglich h.

Auf der Grundlage dieser Definition können wir zeigen, daß gewisse Vektoren, die wir im folgenden angeben, immer im Gitter einer Matrix enthalten sind, und wir diese als zusätzliche Spalten hinzufügen können, ohne das erzeugte Gitter zu verändern.

Lemma 4.67 *Für die HNF-Teiler einer Matrix $A \in \mathbb{Z}^{m \times n}$ mit $\mathrm{rang}(A) = m$ gilt*

$$\delta_i(h) \cdot e_j^m \in \mathcal{L}(A) \qquad \forall i \in \{0, \ldots, m-1\} \qquad \forall j \in \{1, \ldots, m-i\}$$

wobei e_j^m der j-te Einheitsvektor in $\mathbb{Z}^m$ sei.

Beweis: es bezeichne $\mathrm{EHNF}_r(A)$ jene Teilmatrix der $\mathrm{EHNF}(A) = (h_{i,j})$, die durch Streichen der letzten $m - r$ Spalten und Zeilen entsteht. Dann gilt wegen

$$\delta_i(h) = \frac{\delta_0(h)}{\prod\limits_{k=m-i+1}^{m} h_{k,k}} \qquad \forall i \in \{1, \ldots, m-1\}$$

und

$$\det(\mathrm{EHNF}(A)) = \det(\mathcal{L}(A)) \,|\, h = \delta_0(h)$$

zunächst

$$\det(\mathrm{EHNF}_r(A)) \;=\; \frac{\det(\mathrm{EHNF}(A))}{\prod\limits_{k=r+1}^{m} h_{k,k}} \;\Big|\; \frac{\delta_0(h)}{\prod\limits_{k=r+1}^{m} h_{k,k}} \;=\; \delta_{m-r}(h)$$

Mit der Darstellung

$$\mathrm{EHNF}_r(A)^{-1} \;=\; \frac{1}{\det(\mathrm{EHNF}_r(A))} \cdot \mathrm{adj}(\mathrm{EHNF}_r(A))$$

haben wir

$$\delta_{m-r}(h) \cdot I_r \;=\; \mathrm{EHNF}_r(A) \cdot \left(\frac{\delta_{m-r}(h)}{\det(\mathrm{EHNF}_r(A))} \cdot \mathrm{adj}(\mathrm{EHNF}_r(A)) \right)$$

woraus sich die Behauptung ergibt. ■

4.1.3.4 Berechnungsverfahren

Wir erläutern nun ein Verfahren, um die Hermite-Normalform zu berechnen, welches 1987 von P.D.Domich, R.Kannan und L.E.Trotter veröffentlicht wurde (siehe [Dom/Kan/Tro]). Im folgenden bezeichne $A \in \mathbb{Z}^{m\times n}$ immer eine Matrix mit vollem Zeilenrang.

Eliminationschritt Um eine Zahl $u \in \mathbb{Z}\backslash\{0\}$ durch eine unimodulare Transformation zu eliminieren, eignet sich

$$(u, v) \cdot \begin{pmatrix} \frac{v}{\mathrm{ggT}(u,v)} & \mathrm{lggT}(u, v) \\ -\frac{u}{\mathrm{ggT}(u,v)} & \mathrm{rggT}(u, v) \end{pmatrix} \;=\; (0, \mathrm{ggT}(u, v))$$

Dies entspricht den *simultan* auszuführenden Spaltenumformungen

$$S_j \mapsto \frac{v}{\mathrm{ggT}(u, v)} \cdot S_j - \frac{u}{\mathrm{ggT}(u, v)} \cdot S_{j_0}$$

$$S_{j_0} \mapsto \mathrm{lggT}(u, v) \cdot S_j + \mathrm{rggT}(u, v) \cdot S_{j_0}$$

wobei sich u in Spalte j und v in Spalte j_0 befinde.

Bestimmung eines Vielfachen der Gitterdeterminanten Wir werden öfters ein Vielfaches $h \in \mathbb{N}$ der Gitterdeterminanten von $\mathcal{L}(A)$ benötigen. Aus diesem Grund beschreiben wir vorab, wie wir ein solches h berechnen können.

1. Wir wissen, daß jede Teilmatrix $R \in \mathrm{GL}(m, \mathbb{Z})$ von A

   $$\det(R) = \det(\mathcal{L}(A)) \cdot \alpha \quad , \qquad \alpha \in \mathbb{Z}$$

   erfüllt.

2. Wenden wir den Algorithmus zur Bestimmung linear unabhängiger Zeilen auf A^T an, können wir ein solches R einfach ermitteln.

Also kombinieren wir beide Aussagen und wählen $h := |\det(R)|$.

Transformation auf Spaltenstufenform Sei $h \in \mathbb{N}$ ein Vielfaches von $\det(\mathcal{L}(A)$. Um zu garantieren, daß wir arithmetische Operationen nur auf Zahlen der Größenordnung $O(h^\alpha)$ mit einer Konstanten $\alpha \in \mathbb{N}$ ausführen müssen, verwenden wir die im Lemma 4.67 formulierte Eigenschaft der HNF-Teiler von A bezüglich h.

Korollar 4.6 (Hinzunahme einer Diagonalmatrix)
Für ein $A \in \mathbb{Z}^{m\times n}$ *mit* $\operatorname{rang}(A) = m$ *gilt*

$$\mathrm{HNF}(A) \;=\; \mathrm{HNF}(A, h\cdot I_m) \;=\; \mathrm{HNF}(A \mod h, h\cdot I_m)$$

wobei $h \in \mathbb{N}$ *ein Vielfaches von* $\det(\mathcal{L}(A))$ *sei.*

Die Vorgehensweise ist also wie folgt :

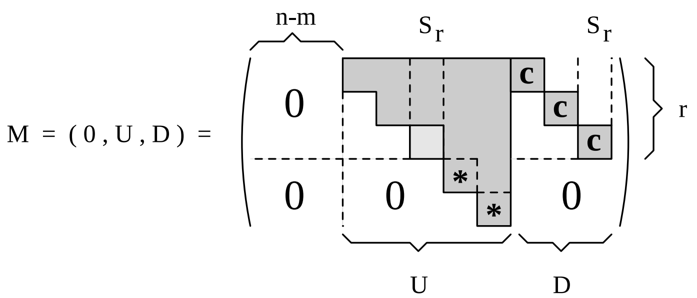
1. Forme $(A \mod h)$ unimodular auf Spaltenstufenform um, und reduziere nach jedem Eliminationsschritt die veränderten Spalten modulo h.
2. Tritt der Fall auf, daß das Pivotelement 0 wird, ersetze es wieder durch h.

Als Ergebnis erhalten wir eine Matrix

$$A^{(1)} \;=\; (0, U^{(1)}, h\cdot I_m) \in \mathbb{Z}^{m\times(n+m)}$$

mit $U^{(1)} \in \mathrm{GL}(m, \mathbb{Z})$ in oberer Dreiecksform.

Entfernen der hinzugenommenen Diagonalmatrix Wir sind unserem eigentlichen Ziel, die Hermite-Normalform von A zu berechnen, insofern einen Schritt näher gerückt, als daß wir uns nur noch überlegen müssen, wie wir die Diagonalmatrix $h\cdot I_m$ in $A^{(1)}$ eliminieren können, ohne die Dreiecksform in $U^{(1)}$ zu zerstören. Um die Vorgehensweise zu veranschaulichen, bezeichne M eine Matrix, die für $r \in \{1,\ldots,m\}$ den folgenden Aufbau habe :



Die Diagonalelemente von $U = (u_{i,j})$ in den Zeilen $r+1\ldots m$ seien mit den entsprechenden Einträgen in der $\mathrm{EHNF}(A) = (h_{i,j})$ identisch :

$$u_{i,i} \;=\; h_{i,i} \qquad \forall i \in \{r+1,\ldots,m\}$$

Der Wert c in D stimme mit dem $m-r$.ten HNF-Teiler von M bezüglich h überein :

$$c \;=\; \delta_{m-r}(h)$$

Wir eliminieren zuerst den Eintrag in der letzten Spalte von D durch die Umformung

$$(u_{r,r}, c) \cdot \begin{pmatrix} \mathrm{lggT}(u_{r,r}, c) & -\frac{c}{\mathrm{ggT}(u_{r,r},c)} \\ \mathrm{rggT}(u_{r,r}, c) & \frac{u_{r,r}}{\mathrm{ggT}(u_{r,r},c)} \end{pmatrix} = (\mathrm{ggT}(u_{r,r}, c), 0)$$

bzw.

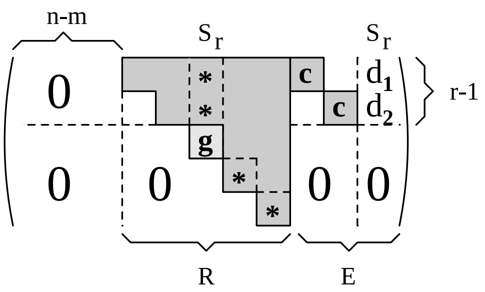
$$\begin{aligned} S_r(U) &\mapsto \mathrm{lggT}(u_{r,r}, c) \cdot S_r(U) + \mathrm{rggT}(u_{r,r}, c) \cdot S_r(D) \\ S_r(D) &\mapsto \left(-\frac{c}{\mathrm{ggT}(u_{r,r}, c)}\right) \cdot S_r(U) + \left(\frac{u_{r,r}}{\mathrm{ggT}(u_{r,r}, c)}\right) \cdot S_r(D) \end{aligned}$$

und reduzieren anschließend die r-te Spalte von U modulo c :

$$S_r(U) \mapsto S_r(U) \mod c$$

Beide Operationen sind unimodular. Die so bestimmte Matrix N sieht mit $g = \mathrm{ggT}(u_{r,r}, c)$ wie folgt aus :

N = (0 , R , E) =



Dabei unterscheidet sich R von U nur in der r-ten Spalte, und der Vektor

$$d = (d_1, \ldots, d_{r-1})^T$$

in der letzten Spalte von E erfüllt wegen

$$d_i = \left(-\frac{c}{g}\right) \cdot u_{i,r} \qquad \forall i \in \{1, \ldots, r-1\}$$

die Teilbarkeitsbedingung

$$\frac{c}{g} \mid d_i \qquad \forall i \in \{1, \ldots, r-1\}$$

Wir beobachten, daß g mit dem korrespondierenden Eintrag in der HNF(A) übereinstimmt

$$g = h_{r,r}$$

und folgern daraus

$$\delta_{m-r+1} = \frac{\delta_{m-r}(h)}{h_{r,r}} = \frac{c}{g}$$

sowie

$$\begin{aligned} &\delta_{m-r+1}(h) \mid c \\ &\delta_{m-r+1}(h) \mid d_i \qquad \forall i \in \{1, \ldots, r-1\} \end{aligned}$$

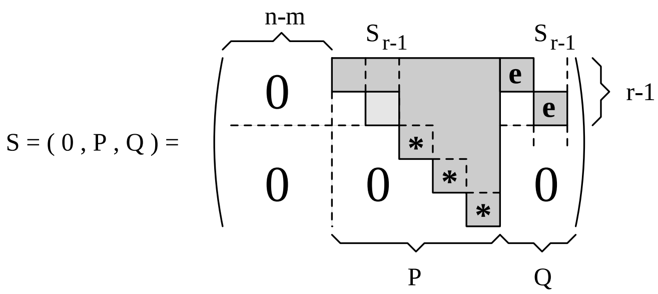
Dies erlaubt uns, die Matrix E durch

$$F \;=\; (\delta_{m-r+1}(h) \cdot I_{r-1}, 0)^T \in \mathbb{Z}^{m \times (n+r-1)}$$

zu ersetzen, ohne dadurch das Gitter $\mathcal{L}(N)$ zu verändern, denn Lemma 4.67 garantiert uns

$$\delta_{m-r+1}(h) \cdot e_j^m \in \mathcal{L}(N) \qquad \forall j \in \{1, \ldots, r-1\}$$

Wir erhalten schließlich eine Matrix S, welche prinzipiell den selben Aufbau hat wie unsere Ausgangsmatrix M, jedoch um eine Spalte reduziert ist :



Für $P = (p_{i,j})$ gilt wieder

$$p_{i,i} \;=\; h_{i,i} \qquad \forall i \in \{r, \ldots, m\}$$

und e ist gleich

$$e \;=\; \delta_{m-r+1}(h)$$

Beginnen wir also mit

$$M \;:=\; A^{(1)} \;=\; (0, U^{(1)}, h \cdot I_m)$$

läßt sich diese Methode solange fortsetzen, bis wir die in $A^{(1)}$ enthaltene Diagonalmatrix entfernt haben, und in der verbleibenden Dreiecksmatrix auf der Hauptdiagonalen die entsprechenden Einträge der EHNF(A) stehen.

Reduktion der Nicht-Diagonaleinträge Wir betrachten $C = (0, W) \in \mathbb{Z}^{m \times n}$ und nehmen an, daß $W \in \mathrm{GL}(m, \mathbb{Z})$ eine obere Dreiecksmatrix mit der Eigenschaft ist, daß auf der Diagonalen bereits die Einträge der EHNF(C) stehen. In einem ersten Schritt ergänzen wir C durch $h \cdot I_m$, wobei h ein Vielfaches von $\det(\mathcal{L}(C))$ sei :

$$C^{(1)} \;=\; (0, W, h \cdot I_m)$$

Während der anschließenden elementaren Reduktion der Nicht-Diagonaleinträge in W sind dann alle Zwischenergebnisse von der Größenordnung $O(h^\alpha)$ mit einer Konstanten $\alpha \in \mathbb{N}$. Als Ergebnis erhalten wir

$$C^{(2)} \;=\; (0, \mathrm{EHNF}(C), h \cdot I_m)$$

Nach Vorraussetzung war $\det(\mathcal{L}(C))$ ein Teiler von h, so daß wir $h \cdot I_m$ wieder entfernen dürfen, wonach die Hermite-Normalform von C vorliegt.

4.1.3.5 Algorithmus

Algorithmus 4.26 (Hermite-Normalform über $\mathbb{Z}$)

Eingabe :

1. Matrix $A \in \mathbb{Z}^{m \times n}$ mit $\mathrm{rang}(A) = m$
2. Vielfaches $h \in \mathbb{N}$ von $\det(\mathcal{L}(A))$

Ausgabe :
Matrix $H \in \mathbb{Z}^{m \times n}$ in Hermite-Normalform mit $\mathcal{L}(H) = \mathcal{L}(A)$

```
(* Initialisierung *)
H := A mod h
(* Transformation auf Spaltenstufenform *)
for ( i := m ; i ≥ 1 ; i := i - 1 )
  do
    (* Eliminiere links vom Pivotelement *)
    for ( j := n-m+i-1 ; j ≥ 1 ; j := j - 1 )
      do
        (* Überprüfe Pivotelement *)
        if ( H[i,n-m+i] == 0 )
          then H[i,n-m+i] := h
        fi
        (* Transformationsmatrix berechnen *)
        g := ggT ( H[i,j] , H[i,n-m+i] )
        a := lggT ( H[i,j] , H[i,n-m+i] )
        b := rggT ( H[i,j] , H[i,n-m+i] )
        x := H[i,n-m+i] div g
        y := - ( H[i,j] div g )
        (* Spalten j und n-m+i aktualisieren *)
        for ( z := 1 ; z ≤ m ; z := z + 1 )
          do
            E[z] := x*H[z,j] + y*H[z,n-m+i]
            F[z] := a*H[z,j] + b*H[z,n-m+i]
            H[z,j] := E[z] mod h
            H[z,n-m+i] := F[z] mod h
          od
      od
  od
(* Entferne die Diagonalmatrix h · I_m *)
D := h
for ( i := m ; i ≥ 1 ; i := i - 1 )
  do
    (* Überprüfe Pivotelement *)
    if ( H [i,n-m+i] == 0)
      then H[i,n-m+i] := D
    fi
    (* Berechne ggT-Darstellung *)
```

```
        g := ggT ( H[i,n-m+i] , D )
        a := lggT ( H[i,n-m+i] , D )
        (* Spalte n-m+i aktualisieren *)
        for ( z := 1 ; z ≤ m ; z := z + 1 )
            do
                E[z] := a*H[z,n-m+i]
                H[z,n-m+i] := ( E[z] mod D )
            od
        H[i,n-m+i] := g
        (* HNF-Teiler aktualisieren *)
        D := D div g
    od
(* Reduktion der Nicht-Diagonaleinträge *)
for ( i := m ; i ≥ 1 ; i := i - 1 )
    do
        (* Überprüfe Diagonalelement *)
        if ( H[i,n-m+i] == 0 )
            then H[i,n-m+i] := h
        fi
        for ( j := n-m+i+1 ; j ≤ n ; j := j + 1 )
            do
                q := H[i,j] div H[i,n-m+i]
                (* Spalte j aktualisieren *)
                for ( z := 1 ; z ≤ i ; z := z + 1 )
                    do
                        H[z,j] := ( H[z,j] - q*H[z,n-m+i] ) mod h
                    od
            od
    od
(* Ergebnis zurückgeben *)
return (H)
```

4.1.3.6 Laufzeitanalyse

Wir unterscheiden im folgenden die Fälle, ob wir bereits ein Vielfaches der Gitterdeterminanten kennen, oder noch berechnen müssen.

Vielfaches der Gitterdeterminanten bekannt

Satz 4.11 (Hermite-Normalform ohne Vorberechnung)
Es sei $A \in \mathbb{Z}^{m \times n}$ eine Matrix mit $\text{rang}(A) = m$. Ferner bezeichne $h \in \mathbb{N}$ ein Vielfaches von $\det(\mathcal{L}(A))$. Dann benötigen wir Zeit $O(m^2 n \cdot \log^2(h))$, um die HNF von A zu berechnen.

Beweis: betrachte Algorithmus 4.26

1. Die Anzahl der arithmetischen Operationen ist $O(m^2 n)$.
2. Alle Zwischenergebnisse sind durch $2h^2$ beschränkt. ∎

Kein Vielfaches der Gitterdeterminanten bekannt

Satz 4.12 (Hermite-Normalform mit Vorberechnung)
Es sei $A \in \mathbb{Z}^{m\times n}$ eine Matrix mit $\operatorname{rang}(A) = m$. Dann benötigen wir Zeit

$$O(m^2 n \cdot (m^2 + n) \cdot \log^2(m|A|))$$

um die HNF von A zu berechnen.

Beweis: die Vorgehensweise gliedert sich in zwei Teile :

1. Die Bestimmung einer Teilmatrix $R \in \mathrm{GL}(m, \mathbb{Z})$ von A kostet uns Zeit
$$O(m \cdot n^2 m \cdot \log^2(m|A|))$$
2. Wir setzen $h := |\det(R)|$ und wenden Algorithmus 4.26 an.
3. Aus der Abschätzung $h \leq (m|A|)^m$ und Satz 4.11 folgern wir die Behauptung. ∎

4.1.4 Transformationsmatrix für die HNF

4.1.4.1 Problemstellung

Für einige Anwendungen, zum Beispiel die Berechnung von Kernen, reicht die Kenntnis der Hermite-Normalform alleine nicht aus : wir benötigen auch die entsprechende Transformationsmatrix.

Aufgabenstellung :
Es sei $A \in \mathbb{Z}^{m\times n}$ und $\operatorname{rang}(A) = m$. Bestimme eine unimodulare Matrix $T \in \mathbb{Z}^{n\times n}$, die A auf Hermite-Normalform transformiert : $A \cdot T = \mathrm{HNF}(A)$.

4.1.4.2 Berechnungsverfahren

Permutationsmatrizen Wir erläutern zunächst den Begriff der Permutationsmatrix, welcher sich als geeignetes Hilfsmittel erweisen wird, um eine prägnante Darstellung der gesuchten Transformation zu entwickeln.

Definition 4.57 *Sei $\sigma \in \mathcal{S}_n$. Wir notieren mit P_σ eine Matrix, die aus I_n durch Vertauschen der Spaltenvektoren gemäß σ entsteht*

$$P_\sigma = \left(e^n_{\sigma(1)}, \ldots, e^n_{\sigma(n)}\right)$$

und nennen P_σ eine **Permutationsmatrix**.

Unser Interesse an solchen Matrizen erwächst aus der Möglichkeit, durch sie ein Umsortieren von Spalten oder Zeilen einfach hinzuschreiben.

Lemma 4.68 *Es sei $M \in \mathbb{Z}^{m \times n}$ eine Matrix mit den Zeilen*

$$c_1, \ldots, c_m \in \mathbb{Z}^n$$

und den Spalten

$$d_1, \ldots, d_n \in \mathbb{Z}^m$$

Ferner sei $\sigma \in \mathcal{S}_n$ und $\tau \in \mathcal{S}_m$. Dann gilt

$$M \cdot P_\sigma = \left(d_{\sigma(1)}, \ldots, d_{\sigma(n)}\right)$$

sowie

$$P_\tau \cdot M = \left(c^T_{\tau^{-1}(1)}, \ldots, c^T_{\tau^{-1}(m)}\right)^T$$

Reguläre Erweiterung einer Matrix

Definition 4.58 (Reguläre Erweiterung über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m \times n}$ eine Matrix mit $\operatorname{rang}(A) = m$. Ferner bezeichne $R \in \mathrm{GL}(m, \mathbb{Z})$ eine Teilmatrix von A, die aus den Spalten mit Indizes

$$1 \leq j_1 < \ldots < j_m \leq n$$

bestehe. Durch

$$1 \leq k_1 < \ldots < k_{n-m} \leq n$$

notieren wir die restlichen Spaltenindizes. Dann ist die reguläre Erweiterung von A bzgl. R erklärt als

$$\mathcal{E}(A, R) := \left(\frac{C}{A}\right) \in \mathrm{GL}(n, \mathbb{Z})$$

mit

$$C = (c_1, \ldots, c_n) \in \mathbb{Z}^{(n-m) \times n}$$

und

$$c_i := \begin{cases} 0 & , \textit{ falls } i \in \{j_1, \ldots, j_m\} \\ e_\beta^{n-m} & , \textit{ falls } i = k_\beta \end{cases}$$

Zwischen A und $\mathcal{E}(A, R)$ besteht ein einfacher Zusammenhang, den wir als Ausgangspunkt für einen Algorithmus verwenden werden.

Lemma 4.69 (Hermite-Normalform einer regulären Erweiterung)
Es sei $A \in \mathbb{Z}^{m \times n}$ und $\mathcal{E}(A, R)$ die reguläre Erweiterung von A bzgl. einer Teilmatrix $R \in \mathrm{GL}(m, \mathbb{Z})$. Ferner bezeichne $T \in \mathbb{Z}^{n \times n}$ eine unimodulare Matrix, die $\mathcal{E}(A, R)$ auf Hermite-Normalform transformiert :

$$\mathcal{E}(A, R) \cdot T = \mathrm{HNF}(\mathcal{E}(A, R))$$

Dann gilt

$$A \cdot T = \mathrm{HNF}(A)$$

sowie

$$\det(\mathcal{L}(\mathcal{E}(A,R))) \; = \; |\det(R)| \; \leq \; H^{\Box}(A)$$

Beweis: die erste Aussage folgt aus der oberen Dreiecksform der HNF, und für die Gitterdeterminante von $\mathcal{E}(A,R)$ haben wir

$$\det(\mathcal{L}(\mathcal{E}(A,R))) \; = \; \det(\mathrm{HNF}(\mathcal{E}(A,R)))$$

sowie

$$\det(\mathrm{HNF}(\mathcal{E}(A,R))) \; = \; |\det(\mathcal{E}(A,R))|$$

Zusammen mit

$$|\det(\mathcal{E}(A,R))| \; = \; |\det(R)|$$

und

$$|\det(R) \; \leq \; H^{\Box}(A)$$

erhalten wir die Behauptung. ■

Darstellung der Transformation Wir erinnern zunächst an Lemma 4.65, welches uns sagt, daß zwar die Hermite-Normalform immer eindeutig bestimmt ist, es aber dennoch mehrere Transformationsmatrizen dafür geben kann. Das nächste Lemma liefert uns also nur einen Hinweis, wie wir *einen* möglichen Kandidaten berechnen können.

Lemma 4.70 (Transformationsmatrix für die HNF)
Es sei $A \in \mathbb{Z}^{m\times n}$ und $\mathcal{E}(A,R)$ die reguläre Erweiterung von A bzgl. einer Teilmatrix $R \in \mathrm{GL}(m,\mathbb{Z})$. Ferner bezeichne P_σ eine Permutationsmatrix mit

$$A \cdot P_\sigma \; = \; (R, F)$$

bzw.

$$\mathcal{E}(A,R) \cdot P_\sigma \; = \; \left(\begin{array}{c|c} 0 & I_{n-m} \\ \hline R & F \end{array}\right)$$

Wir unterteilen die Hermite-Normalform von $\mathcal{E}(A,R)$ in

$$\mathrm{HNF}(\mathcal{E}(A,R)) \; = \; \left(\begin{array}{c|c} U & V \\ \hline 0 & \mathrm{EHNF}(A) \end{array}\right)$$

und definieren eine Matrix $T \in \mathbb{Z}^{n\times n}$ durch

$$T \; := \; \frac{1}{\det(R)} \cdot P_\sigma \cdot \left(\begin{array}{c|c} -\mathrm{adj}(R)\cdot F & \mathrm{adj}(R) \\ \hline \det(R)\cdot I_{n-m} & 0 \end{array}\right) \cdot \left(\begin{array}{c|c} U & V \\ \hline 0 & \mathrm{EHNF}(A) \end{array}\right)$$

Dann gilt

$$A \cdot T \; = \; \mathrm{HNF}(A)$$

sowie

$$\det(T) \in \{+1, -1\}$$

Für die Einträge in T haben wir die Abschätzung

$$|T| \leq (n-m+1) \cdot m|A| \cdot H^{\square}(A)$$

Beweis: wir multiplizieren $A \cdot T$ aus und erhalten

$$\begin{aligned}
A \cdot T &= \frac{1}{\det(R)} \cdot (R, F) \cdot \left(\begin{array}{c|c} -\mathrm{adj}(R) \cdot F & \mathrm{adj}(R) \\ \hline \det(R) \cdot I_{n-m} & 0 \end{array}\right) \cdot \left(\begin{array}{c|c} U & V \\ \hline 0 & \mathrm{EHNF}(A) \end{array}\right) \\
&= \frac{1}{\det(R)} \cdot (0, \det(R) \cdot I_m) \cdot \left(\begin{array}{c|c} U & V \\ \hline 0 & \mathrm{EHNF}(A) \end{array}\right) \\
&= (0, I_m) \cdot \left(\begin{array}{c|c} U & V \\ \hline 0 & \mathrm{EHNF}(A) \end{array}\right) \\
&= (0, \mathrm{EHNF}(A))
\end{aligned}$$

Ferner berechnen wir die Determinante von T als

$$\begin{aligned}
\det(T) &= \det(P_\sigma) \cdot \frac{\det(\mathcal{L}(\mathcal{E}(A,R))) \cdot \det(R)^{n-m} \cdot \det(R)^{m-1}}{\det(R)^n} \\
&= \frac{\mathrm{sign}(\sigma) \cdot |\det(R)|}{\det(R)} \in \{+1, -1\}
\end{aligned}$$

Die Abschätzung für $|T|$ folgern wir aus

$$|T| \leq \frac{(n-m) \cdot m|A| \cdot H^{\square}(A) \cdot \max\{|U|, |V|\} + m \cdot H^{\square}(A) \cdot |\mathrm{EHNF}(A)|}{|\det(R)|}$$

und

$$|U|, |V|, |\mathrm{EHNF}(A)| \leq \det(\mathcal{L}(\mathcal{E}(A,R))) = |\det(R)|$$

Schließlich beachten wir noch, daß für reguläre Matrizen, also insbesondere $\mathcal{E}(A,R)$, die Transformationsmatrix eindeutig bestimmt ist, und wir für T die Darstellung

$$\mathcal{E}(A,R) \cdot T = \mathrm{HNF}(\mathcal{E}(A,R)) \quad \Leftrightarrow \quad T = \mathcal{E}(A,R)^{-1} \cdot \mathrm{HNF}(\mathcal{E}(A,R))$$

haben, was uns $T \in \mathbb{Z}^{n \times n}$ garantiert. ∎

4.1.4.3 Algorithmus

Algorithmus 4.27 (Transformationsmatrix für die HNF)

Eingabe :
Matrix $A \in \mathbb{Z}^{m \times n}$ mit $\mathrm{rang}(A) = m$

Ausgabe :
Unimodulares $T \in \mathbb{Z}^{n \times n}$ mit $A \cdot T = \mathrm{HNF}(A)$

Vorgehen :

1. Bestimme eine Teilmatrix $R \in \mathrm{GL}(m, \mathbb{Z})$ von A
2. Fasse die nicht in R enthaltenen Spalten von A zu einer Matrix F zusammen
3. Konstruiere $\mathcal{E}(A, R)$ sowie eine Permutationsmatrix P_σ mit
$$\mathcal{E}(A,R) \cdot P_\sigma = \left(\begin{array}{c|c} 0 & I_{n-m} \\ \hline R & F \end{array}\right)$$
4. Berechne die Determinante und die Adjungierte von R
5. Bestimme die Hermite-Normalform von $\mathcal{E}(A, R)$ und teile diese auf in
$$\mathrm{HNF}(\mathcal{E}(A,R)) = \left(\begin{array}{c|c} U & V \\ \hline 0 & \mathrm{EHNF}(A) \end{array}\right)$$
6. Berechne die Matrix
$$M := \left(\begin{array}{c|c} -\mathrm{adj}(R) \cdot (F \cdot U) & -\mathrm{adj}(R) \cdot (F \cdot V) + \mathrm{adj}(R) \cdot \mathrm{EHNF}(A) \\ \hline \det(R) \cdot U & \det(R) \cdot V \end{array}\right)$$
7. Ermittele T als
$$T := \frac{1}{\det(R)} \cdot P_\sigma \cdot M$$

4.1.4.4 Laufzeitanalyse

Satz 4.13 (Transformationsmatrix für die HNF)
Um für $A \in \mathbb{Z}^{m\times n}$ mit $\mathrm{rang}(A) = m$ eine unimodulare Matrix $T \in \mathbb{Z}^{n\times n}$ zu berechnen, die A auf Hermite-Normalform transformiert, benötigen wir Zeit $O(n^3m^2 \cdot \log^2(m|A|))$.

Beweis: betrachte Algorithmus 4.27

1. Schritte 1-4 : erfordern Zeit $O(m^2n^2 \cdot \log^2(m|A|))$.
2. Schritt 5 : wir wählen $h = |\det(R)|$ und folgern aus
$$h \leq (m|A|)^m$$
einen Aufwand von
$$O(n^3m^2 \cdot \log^2(m|A|))$$

3. Schritt 6 : für die Komponenten von M erhalten wir mit $|U|, |V|, |\mathrm{EHNF}(A)| \leq H^{\square}(A)$

Teilmatrix	Zeit
$F \cdot U$	$O(m(n-m)^2 \cdot (m + \log(n-m)) \cdot \log^2(m\|A\|))$
$F \cdot V$	$O(m^2(n-m) \cdot (m + \log(n-m)) \cdot \log^2(m\|A\|))$
$-\mathrm{adj}(R) \cdot (F \cdot U)$	$O(m^3(n-m) \cdot (m + \log(n-m)) \cdot \log^2(m\|A\|))$
$-\mathrm{adj}(R) \cdot (F \cdot V)$	$O(m^4 \cdot (m + \log(n-m)) \cdot \log^2(m\|A\|))$
$\mathrm{adj}(R) \cdot \mathrm{EHNF}(A)$	$O(m^5 \cdot \log^2(m\|A\|))$
$-\mathrm{adj}(R) \cdot (F \cdot V)$ $+\mathrm{adj}(R) \cdot \mathrm{EHNF}(A)$	(Addition) $O(m^2 \cdot (m + \log(n-m)) \cdot \log(m\|A\|))$
$\det(R) \cdot U$	$O(m^2 \cdot (n-m)^2 \cdot \log^2(m\|A\|))$
$\det(R) \cdot V$	$O(m^3 \cdot (n-m) \cdot \log^2(m\|A\|))$
M	$O(n^3 \cdot m^2 \cdot \log^2(m\|A\|))$

4. Schritt 7 : die Division von M durch $\det(R)$ kostet uns schließlich noch Zeit

$$O(n^2 m \cdot (m+n) \cdot \log^2(m|A|))$$

■

4.1.5 Berechnung des Kernes

4.1.5.1 Algorithmus

Algorithmus 4.28 (Kern über $\mathbb{Z}$)

Eingabe :
Matrix $A \in \mathbb{Z}^{m \times n}$ mit den Zeilen $a_1, \ldots, a_m \in \mathbb{Z}^n$

Ausgabe :
Basis $(u_1, \ldots, u_{n-r}) \in \mathbb{Z}^{n \times (n-r)}$ von Kern(A)

Vorgehen :

1. Berechne den Rang r von A sowie Indizes $i_1, \ldots, i_r$ linear unabhängiger Zeilen
2. Übernimm die Zeilen $i_r, \ldots, i_1$ in eine Matrix $\mathcal{ZB}(A)$:
$$\mathcal{ZB}(A) := \left(a_{i_r}^T, \ldots, a_{i_1}^T\right)^T \in \mathbb{Z}^{r \times n}$$
3. Bestimme ein unimodulares T, das $\mathcal{ZB}(A)$ auf Hermite-Normalform transformiert
4. Lies die Basis in den ersten $n-r$ Spalten von T ab :
$$u_i := t_i \qquad \forall i \in \{1, \ldots, n-r\}$$

4.1.5.2 Laufzeitanalyse

4.1.5.2.1 Ergebnis für Matrizen

Satz 4.14 (Kern einer Matrix über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m \times n}$ und $s = \min\{m, n\}$. Dann benötigen wir Zeit

$$O(n \cdot (sm^2 + n^2 r^2) \cdot \log^2(s|A|))$$

um den Kern von A zu berechnen.

Beweis: sei $r = \text{rang}(A)$. Betrachte Algorithmus 4.28 :

1. Die Bestimmung von $\mathcal{ZB}(A)$ kostet Zeit $O(s \cdot m^2 n \cdot \log^2(s|A|))$.
2. Aus Satz 4.13 folgern wir einen Aufwand von

   $$O(n^3 r^2 \cdot \log^2(r|A|))$$

   für die Berechnung von T. ■

4.1.5.2.2 Ergebnis für Homomorphismen

Satz 4.15 (Kern eines Homomorphismus über $\mathbb{Z}$)
Es sei $\varphi \in \text{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ und $s = \min\{m, n\}$. Dann benötigen wir Zeit

$$O(n \cdot (sm^2 + n^2 r^2) \cdot \log^2(s|\varphi|))$$

um den Kern von φ zu berechnen.

4.1.6 Berechnung des Bildes

4.1.6.1 Algorithmus

Algorithmus 4.29 (Bild über $\mathbb{Z}$)

Eingabe :
Matrix $A \in \mathbb{Z}^{m \times n}$ mit den Zeilen $a_1, \ldots, a_m \in \mathbb{Z}^n$

Ausgabe :
Basis $(w_1, \ldots, w_r) \in \mathbb{Z}^{m \times r}$ von Bild(A)

Vorgehen :

1. Berechne den Rang von r von A sowie Indizes $i_1, \ldots, i_r$ linear unabhängiger Zeilen

2. Teile A in zwei Matrizen X und Y auf :

$$A = (X, Y)$$

mit

$$X \in \mathbb{Z}^{m \times r} \qquad , \qquad Y \in \mathbb{Z}^{m \times (n-r)}$$

3. Übernimm die Zeilen $i_r, \ldots, i_1$ aus A in eine Matrix $\mathcal{ZB}(A)$:

$$\mathcal{ZB}(A) := \left(a_{i_r}^T, \ldots, a_{i_1}^T\right)^T \in \mathbb{Z}^{r \times n}$$

4. Bestimme eine unimodulare Matrix T, die $\mathcal{ZB}(A)$ auf Hermite-Normalform transformiert, und schreibe T als

$$T = \left(\begin{array}{c|c} B & C \\ \hline D & H \end{array}\right)$$

mit

$$B \in \mathbb{Z}^{r \times (n-r)} \qquad , \qquad H \in \mathbb{Z}^{(n-r) \times r}$$

5. Berechne $M := X \cdot C + Y \cdot H$ und lies die Basis in den Spalten von M ab :

$$w_i := m_i \qquad \forall i \in \{1, \ldots, r\}$$

4.1.6.2 Laufzeitanalyse

4.1.6.2.1 Ergebnis für Matrizen

Satz 4.16 (Bild einer Matrix über $\mathbb{Z}$)
Es sei $A \in \mathbb{Z}^{m \times n}$ und $s = \min\{m, n\}$. Dann benötigen wir Zeit

$$O(n \cdot (sm^2 + n^2r^2 + mr \cdot (r + \log(n)) \cdot \log^2(s|A|))$$

um das Bild von A zu berechnen.

Beweis: sei $r = \text{rang}(A)$. Betrachte Algorithmus 4.29 :

1. Die Bestimmung von $\mathcal{ZB}(A)$ kostet Zeit $O(s \cdot m^2 n \cdot \log^2(s|A|))$.

2. Aus Satz 4.13 folgern wir einen Aufwand von

$$O(n^3 r^2 \cdot \log^2(r|A|))$$

für die Berechnung von T.

3. Das Ausrechnen von M kostet uns wegen

$$|C| \leq (n-r+1) \cdot (r|A|)^{r+1} \qquad , \qquad |H| \leq (r|A|)^r$$

dann noch

Teilmatrix	Zeit
$X \cdot C$	$O(mr^2 \cdot (r + \log(n-r)) \cdot \log^2(r\|A\|))$
$Y \cdot H$	$O(mr(n-r) \cdot (r + \log(n-r)) \cdot \log^2(r\|A\|))$
$X \cdot C + Y \cdot H$	$O(mr \cdot (r + \log(n-r)) \cdot \log(r\|A\|))$
M	$O(mnr \cdot (r + \log(n)) \cdot \log^2(r\|A\|))$

■

4.1.6.2.2 Ergebnis für Homomorphismen

Satz 4.17 (Bild eines Homomorphismus über $\mathbb{Z}$)
Es seien $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ und $s = \min\{m, n\}$. Dann benötigen wir Zeit

$$O(n \cdot (sm^2 + n^2r^2 + mr \cdot (r + \log(n))) \cdot \log^2(s|\varphi|))$$

um das Bild von φ zu berechnen.

4.2 Urbild

4.2.1 Problemstellung

Definition 4.59 (Urbild und lineare Gleichungssysteme über $\mathbb{Z}$)
Es sei $\varphi \in \mathrm{Hom}(\mathbb{Z}^n, \mathbb{Z}^m)$ und $b \in \mathbb{Z}^m$. Wir notieren durch

$$\mathcal{UB}(\varphi, b) := \{x \in \mathbb{Z}^n \mid \varphi(x) = b\}$$

die Menge aller Urbilder von b unter φ. Entsprechend bezeichnet

$$\mathcal{LG}_{\mathbb{Z}}(A, b) := \{x \in \mathbb{Z}^n \mid A \cdot x = b\}$$

für $A \in \mathbb{Z}^{m \times n}$ die Lösungsmenge von $A \cdot x = b$.

Wenden wir hierauf die Abbildung Mat an, erhalten wir

$$\mathcal{UB}(\varphi, b) = \mathcal{LG}_{\mathbb{Z}}(\mathrm{Mat}(\varphi), b)$$

Die Berechnung von Urbildern entspricht also dem Lösen linearer Gleichungssysteme über dem Ring der ganzen Zahlen.

Aufgabenstellung :
Es seien $A \in \mathbb{Z}^{m \times n}$ und $b \in \mathbb{Z}^m$ gegeben. Bestimme ein $x^* \in \mathcal{LG}_{\mathbb{Z}}(A, b)$.

4.2.2 Berechnungsverfahren

4.2.2.1 Berechnung eines speziellen Urbildes

Wir schreiben das Gleichungssystem

$$A \cdot x = b \tag{4.19}$$

für ein A mit den Spalten

$$a_1, \ldots, a_n \in \mathbb{Z}^m$$

und

$$x = (x_1, \ldots, x_n)^T \in \mathbb{Z}^n$$

als

$$x_1 \cdot a_1 + \ldots + x_n \cdot a_n - 1 \cdot b \; = \; 0$$

Daraus folgt, daß (4.19) genau dann lösbar ist, wenn der Kern von (A, b) einen Vektor

$$x^* = (x_1^*, \ldots, x_n^*, -1)^T \in \mathbb{Z}^{n+1}$$

enthält. Um dies herauszufinden, überprüfen wir zuerst

$$\dim_{\mathbb{Q}} \mathrm{Kern}(A, b) > 0$$

und berechnen dann eine Basis

$$U = (u_1, \ldots, u_{n+1-r}) \in \mathbb{Z}^{(n+1)\times(n+1-r)}$$

des Kernes von (A, b) , wobei

$$r \; = \; \mathrm{rang}(A, b)$$

gelte. Enthält U in der letzten Zeile nur Nullen, oder ist

$$g = \mathrm{ggT}(u_{n+1,1}, \ldots, u_{n+1,n+1-r}) > 1$$

gibt es kein solches x^*, und somit auch keine spezielle Lösung von (4.19). Andernfalls berechnen wir eine Darstellung

$$g = \sum_{i=1}^{n+1-r} \alpha_i u_{n+1,i} \qquad , \qquad \alpha_i \in \mathbb{Z}$$

mit kleinen Koeffizienten α_i, und setzen

$$\hat{x} \; := \; - \sum_{i=1}^{n+1-r} \alpha_i u_i$$

Die gesucht Lösung x^* von (4.19) steht dann in den ersten n Komponenten von $\hat{x}$.

4.2.2.2 Berechnung der Urbildmenge

Um auf der Grundlage einer speziellen Lösung x^* von $A \cdot x = b$ die Menge $\mathcal{LG}_{\mathbb{Z}}(A, b)$ zu bestimmen, reicht es aus zu beobachten, daß für beliebige $y, z \in \mathcal{LG}_{\mathbb{Z}}(A, b)$ immer

$$y - z \in \mathrm{Kern}(A)$$

gilt. Wir haben also die Darstellung

$$\mathcal{LG}_{\mathbb{Z}}(A, b) \;=\; x^* + \mathrm{Kern}(A)$$

4.2.3 Algorithmus

Algorithmus 4.30 (Urbild über $\mathbb{Z}$)

Eingabe :

1. Koeffizientenmatrix $A \in \mathbb{Z}^{m \times n}$
2. Rechte Seite $b \in \mathbb{Z}^m$

Ausgabe :
Spezielle Lösung $x^* \in \mathcal{LG}_{\mathbb{Z}}(A, b)$ oder $\mathcal{LG}_{\mathbb{Z}}(A, b) = \emptyset$

Vorgehen :

1. Berechne eine Basis
$$U = (u_1, \ldots, u_{n+1-r}) \in \mathbb{Z}^{(n+1)\times(n+1-r)}$$
des Kernes von (A, b)
2. Ist $\dim_{\mathbb{Q}} \mathrm{Kern}(A, b) = 0$ oder enthält U in der letzten Zeile nur Nullen, gib
$$\mathcal{LG}_{\mathbb{Z}}(A, b) = \emptyset$$
aus und beende die Bearbeitung.
3. Andernfalls bestimme
$$g = \mathrm{ggT}(u_{n+1,1}, \ldots, u_{n+1,n+1-r})$$
sowie eine Darstellung
$$g = \sum_{i=1}^{n+1-r} \alpha_i u_{n+1,i} \qquad , \qquad \alpha_i \in \mathbb{Z}$$
4. Ist $g = 1$, berechne
$$\hat{x} = -\sum_{i=1}^{n+1-r} \alpha_i u_i$$
Gilt hingegen $g > 1$, gib
$$\mathcal{LG}_{\mathbb{Z}}(A, b) = \emptyset$$
aus und beende die Bearbeitung.
5. Die gesuchte Lösung x^* steht jetzt in den ersten n Komponenten von $\hat{x}$.

4.2.4 Laufzeitanalyse

4.2.4.1 Ergebnis für Matrizen

Satz 4.18 (Lineare Gleichungssysteme über $\mathbb{Z}$)
Um eine spezielle Lösung $x^* \in \mathcal{LG}_{\mathbb{Z}}(A,b)$ eines linearen GLS

$$A \cdot x = b$$

mit einer Koeffizientenmatrix $A \in \mathbb{Z}^{m \times n}$ und einer rechten Seite $b \in \mathbb{Z}^m$ zu berechnen, benötigen wir mit

$$s = \min\{m,n\} \qquad , \qquad \delta = \max\{|A|,|b|\}$$

einen Zeitaufwand der Größenordnung

$$O(n \cdot (sm^2 + n^2r^2) \cdot \log^2(s\delta))$$

Beweis: es sei $r = \mathrm{rang}(A)$. Betrachte Algorithmus 4.30 :

1. Die Bestimmung von $\mathrm{Kern}(A,b)$ kostet Zeit
$$O(smn \cdot (m + n^2) \cdot \log^2(s\delta))$$
2. Aus Lemma 1.11 folgern wir einen Aufwand von
$$O((n-r) \cdot \log^2((n-r)(r\delta)^r)) \;=\; O(n^3 \cdot \log^2(r\delta))$$
für die Berechnung der ggT-Darstellung, denn die Einträge in U genügen der Abschätzung
$$|U| \le (n-r+2) \cdot (r\delta)^{r+1} \qquad \text{(Lemma 4.70)}$$
3. Um den Zeitbedarf für Schritt 3 abzuschätzen, beachten wir
$$|\alpha_i| \le (n-r+2)^3 \cdot (r\delta)^2 \qquad \forall i \in \{1,\ldots,n-r+1\}$$
sowie
$$\log(N) \cdot \log\log(N) \;=\; O(N)$$
und erhalten
$$\begin{aligned} & O(\log((n-r)(r\delta)^r) \cdot ((n-r)n \cdot \log\log((n-r)(r\delta)^r) + \log((n-r)(r\delta)^r))) \\ = \; & O((n-r)^2 n \cdot r \cdot \log^2(r\delta) + \log^2(n-r) + r^2 \cdot \log^2(r\delta)) \\ = \; & O(n^3 r \cdot \log^2(r\delta)) \end{aligned}$$

■

4.2.4.2 Ergebnis für Homomorphismen

Satz 4.19 (Urbild über $\mathbb{Z}$)
Um ein Urbild $x^* \in \mathcal{UB}(\varphi,b)$ eines Vektors $b \in \mathbb{Z}^m$ unter $\varphi \in \mathrm{Hom}(\mathbb{Z}^n,\mathbb{Z}^m)$ zu berechnen, benötigen wir mit

$$s = \min\{m,n\} \qquad , \qquad \delta = \max\{|\varphi|,|b|\}$$

einen Zeitaufwand der Größenordnung

$$O(n \cdot (sm^2 + n^2r^2) \cdot \log^2(s\delta))$$

Teil V

Beispiele

5.1 Aufgabenstellung

Es sei das lineare Gleichungssystem

$$\begin{array}{rcrcrcrcr} - & a & - & b & - & 3c & - & d & = & 23 \\ - & 5a & + & 3b & - & 3c & - & d & = & 7 \\ & 7a & - & 5b & + & 3c & + & d & = & 1 \end{array} \tag{5.20}$$

in den Variablen $a, b, c, d \in \mathbb{Z}$ gegeben. Wir bestimmen die Lösungsmenge dieses Systemes in drei Schritten :

- zuerst schreiben wir (5.20) als
$$A \cdot x \;=\; b$$
mit
$$A \;=\; \begin{pmatrix} -1 & -1 & -3 & -1 \\ -5 & 3 & -3 & -1 \\ 7 & -5 & 3 & 1 \end{pmatrix} \quad , \quad b \;=\; \begin{pmatrix} 23 \\ 7 \\ 1 \end{pmatrix}$$
- dann berechnen wir eine spezielle Lösung
$$x^* \in \mathcal{LG}_{\mathbb{Z}}(A, b)$$
- anschließend konstruieren wir eine Basis von Kern(A)

Die Menge aller Lösungen von (5.20) hat dann die Form

$$\mathcal{LG}_{\mathbb{Z}}(A, b) \;=\; x^* + \mathrm{Kern}(A)$$

5.2 Bestimmung einer speziellen Lösung x^*

Die Vorgehensweise gliedert sich in zwei Teile :

- zuerst berechnen wir eine Basis von Kern(A, b)
- dann ermitteln wir einen Vektor
$$v^* \;=\; (v_1^*, v_2^*, v_3^*, v_4^*, -1)^T \in \mathrm{Kern}(A, b)$$
sofern es einen solchen gibt, und setzen
$$x^* \;:=\; (v_1^*, v_2^*, v_3^*, v_4^*)^T \in \mathcal{LG}_{\mathbb{Z}}(A, b)$$

5.2.1 Berechnung einer Basis von Kern(A,b)

Wir wenden Algorithmus 4.28 auf (A, b) an.

5.2.1.1 Linear unabhängige Zeilen bestimmen

Wir berechnen den Rang r von (A, b) sowie Indizes $i_1, \ldots, i_r$ linear unabhängiger Zeilen.

Approximation der Hadamard-Schranke Wir erhalten für die Längenquadrate der Spaltenvektoren

$$h = (75, 35, 27, 3, 579)$$

und setzen

$$columns := 579 \cdot 75 \cdot 35$$

Entsprechend verfahren wir mit den Zeilenvektoren :

$$rows := 539 \cdot 83 \cdot 83 > columns$$

Also wählen wir

$$H^*(A) = \langle columns \rangle^{\frac{1}{2}} = 2^{12} - 1 = 4095$$

Bestimmung der Primzahlliste Da wir keine Teilbarkeitsrestriktion beachten müssen, genügen die Primzahlen

$$2, 3, 5, 7, 11, 13$$

der Bedingung

$$\prod p_i > 2 \cdot H^*(A)$$

Rekonstruktion des Verlaufes der Spaltenstufenform Als nächstes transformieren wir die reduzierten Matrizen

$$(A, b) \bmod p_i \in \mathbb{F}_{p_i}^{3\times 5}$$

über den jeweiligen Körpern $\mathbb{F}_{p_i}$ auf Spaltenstufenform, und lesen die Sprungstellen der Zeilenrangfunktionen ab :

$$\mathbb{F}_2 : (A, b) \mapsto^* \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad \Rightarrow \quad r = 1\,,\, i_1 = 3\,,\, i_2 = 0$$

$$\mathbb{F}_3 : (A, b) \mapsto^* \begin{pmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad \Rightarrow \quad r = 2\,,\, i_1 = 3\,,\, i_2 = 2$$

$$\mathbb{F}_5 : (A, b) \mapsto^* \begin{pmatrix} 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & \mathbf{1} & 2 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad \Rightarrow \quad r = 2\,,\, i_1 = 3\,,\, i_2 = 2$$

$$\mathbb{F}_7 : (A, b) \mapsto^* \begin{pmatrix} 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad \Rightarrow \quad r = 2\,,\, i_1 = 3\,,\, i_2 = 2$$

$$\mathbb{F}_{11} : (A,b) \mapsto^* \begin{pmatrix} 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & \mathbf{1} & 7 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad \Rightarrow \quad r = 2\,,\, i_1 = 3\,,\, i_2 = 2$$

$$\mathbb{F}_{13} : (A,b) \mapsto^* \begin{pmatrix} 0 & 0 & 0 & 3 & 10 \\ 0 & 0 & 0 & \mathbf{1} & 7 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad \Rightarrow \quad r = 2\,,\, i_1 = 3\,,\, i_2 = 2$$

Wir tragen diese Informationen in einer Tabelle zusammen :

p	2	3	5	7	11	13	$\mathbb{Z}$
i_2	0	2	2	2	2	2	2
i_1	3	3	3	3	3	3	3
r	1	2	2	2	2	2	2

Die Matrix (A,b) hat also den Rang 2, und die Zeilen mit den Indizes 2 bzw. 3 sind linear unabhängig.

5.2.1.2 Bestimmung der Hermite-Normalform

Im nächsten Schritt bestimmen wir die Hermite-Normalform der regulären Erweiterung von

$$E = \begin{pmatrix} -5 & 3 & -3 & -1 & 7 \\ 7 & -5 & 3 & 1 & 1 \end{pmatrix}$$

bezüglich der Teilmatrix

$$R = \begin{pmatrix} -3 & 7 \\ 3 & 1 \end{pmatrix} \in \mathrm{GL}(2,\mathbb{Z})$$

und wählen als Vielfaches der Gitterdeterminanten $\det(\mathcal{L}(E))$

$$h = |\det(R)| = 24$$

Wir erhalten

$$\mathcal{E}(E,R) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -5 & 3 & -3 & -1 & 7 \\ 7 & -5 & 3 & 1 & 1 \end{pmatrix} \in \mathrm{GL}(5,\mathbb{Z})$$

sowie

$$(\mathcal{E}(E,R), h \cdot I_5) = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 24 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 24 & 0 & 0 \\ -5 & 3 & -3 & -1 & 7 & 0 & 0 & 0 & 24 & 0 \\ 7 & -5 & 3 & 1 & 1 & 0 & 0 & 0 & 0 & 24 \end{array}\right) \mapsto^*$$

$$\left(\begin{array}{ccccc|ccccc} 24 & 7 & 8 & 0 & 0 & 24 & 0 & 0 & 0 & 0 \\ 0 & 15 & 16 & 23 & 0 & 0 & 24 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 24 & 0 & 0 \\ 0 & 0 & 0 & 2 & 7 & 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 24 \end{array}\right) \mapsto^*$$

$$\left(\begin{array}{ccc|cc} 4 & 7 & 8 & 0 & 0 \\ 0 & 3 & 4 & 23 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 2 & 7 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) \mapsto^* \left(\begin{array}{ccc|cc} 4 & 3 & 1 & 2 & 1 \\ 0 & 3 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) = \mathrm{HNF}(\mathcal{E}(E,R))$$

5.2.1.3 Bestimmung der Transformationsmatrix

Wir konstruieren nun eine unimodulare Matrix $T \in \mathbb{Z}^{5\times 5}$, welche $\mathcal{E}(E,R)$ auf die soeben berechnete Hermite-Normalform transformiert, wobei wir die Notationen aus Algorithmus 4.27 verwenden. Zuerst lesen wir aus der $\mathrm{HNF}(\mathcal{E}(E,R))$ die Teilmatrizen

$$U = \begin{pmatrix} 4 & 3 & 1 \\ 0 & 3 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad V = \begin{pmatrix} 2 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$$

sowie

$$\mathrm{EHNF}(E) = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$$

ab. Durch Heranmultiplizieren einer Permutationsmatrix P_σ

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}$$

teilen wir E auf in

$$E \cdot P_\sigma = (R, F)$$

mit

$$R = \begin{pmatrix} -3 & 7 \\ 3 & 1 \end{pmatrix} \quad , \quad F = \begin{pmatrix} -5 & 3 & -1 \\ 7 & -5 & 1 \end{pmatrix}$$

Die Komponenten der Hilfsmatrix M berechnen wir als

$$F \cdot U = \begin{pmatrix} -20 & -6 & -3 \\ 28 & 6 & 3 \end{pmatrix} \quad , \quad F \cdot V = \begin{pmatrix} -7 & -5 \\ 9 & 7 \end{pmatrix}$$

und

$$\mathrm{adj}(R) = \begin{pmatrix} 1 & -7 \\ -3 & -3 \end{pmatrix} \quad , \quad \mathrm{adj}(R) \cdot \mathrm{EHNF}(E) = \begin{pmatrix} 2 & -6 \\ -6 & -6 \end{pmatrix}$$

Wir haben also

$$M = \left(\begin{array}{rrr|rr} 216 & 48 & 24 & 72 & 48 \\ 24 & 0 & 0 & 0 & 0 \\ \hline -96 & -72 & -24 & -48 & -24 \\ 0 & -72 & -24 & -24 & 0 \\ 0 & 0 & -24 & 0 & 0 \end{array}\right)$$

bzw.

$$T = \left(\begin{array}{rrr|rr} 4 & 3 & 1 & 2 & 1 \\ 0 & 3 & 1 & 1 & 0 \\ -9 & -2 & -1 & -3 & -2 \\ 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{array}\right)$$

5.2.1.4 Ablesen der Basis von Kern(A,b)

Die gesuchte Basis des Kernes von (A,b) lesen wir in den ersten drei Spalten von T ab :

$$\mathrm{Kern}(A,b) \;=\; \langle \begin{pmatrix} 4 \\ 0 \\ -9 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \\ -2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \\ 0 \end{pmatrix} \rangle_{\mathbb{Z}}$$

5.2.2 Auswählen einer speziellen Lösung x^* aus Kern(A,b)

Wir beachten, daß der erste Vektor in der obigen Basis bereits die erforderliche Form hat, und erhalten

$$x^* \;=\; (4,0,-9,0)^T \in \mathcal{LG}_{\mathbb{Z}}(A,b)$$

als spezielle Lösung von (5.20).

5.3 Konstruktion einer Basis von Kern(A)

Führen wir mit A die selben Schritte durch wie zuvor mit (A,b), liefert uns dies

$$\mathrm{Kern}(A) \;=\; \langle \begin{pmatrix} 1 \\ 1 \\ 0 \\ -2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ -3 \end{pmatrix} \rangle_{\mathbb{Z}}$$

5.4 Zusammensetzen der vollständigen Lösung

Die Lösungsmenge des Gleichungssystemes (5.20) lautet also

$$\mathcal{LG}_{\mathbb{Z}}(A,b) = \left\{ \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ -9 \\ 0 \end{pmatrix} + \alpha \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ -2 \end{pmatrix} + \beta \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ -3 \end{pmatrix} \quad \text{mit } \alpha,\beta \in \mathbb{Z} \right\}$$

Stichwortverzeichnis

P

R

S

T

U

V

W

Z

Literaturverzeichnis

[Aho/Hop/Ull] Alfred V. Aho,John E. Hopcroft,Jeffrey D. Ullmann , *The design and analysis of computer algorithms* , Addison-Wesley Publishing Company , 1974 , 2-32 , 278-316

[Aigner] Alexander Aigner , *Zahlentheorie* , DeGruyter Lehrbuch , 1975 , 5-70

[Akritas] Alkiviadis Akritas , *Elements of Computer Algebra* , John Wiley and Sons Publications , New York-Chichester-Brisbane , 1989 , 21-100

[Bradley] G.H. Bradley , *Algorithm and bound for the greatest common divisor of n integers*, Communications of the ACM , 1970 , Vol.13 , 433-436

[Bro/Sem] I.N.Bronstein,K.A.Semendjajew,G.Musiol,H.Mühlig ,*Taschenbuch der Mathematik* , Verlag Harri Deutsch , Frankfurt , 1993 , 209-218

[Buchm1] Johannes Buchmann , *Einführung in die Computeralgebra* , Skript zur Vorlesung im WS 91/92 an der Universität Saarbrücken , 2-15 , 29-42

[Buchm2] Johannes Buchmann , *Algorithmen und Komplexität II* , Skript zur Vorlesung im SS 91 an der Universität Saarbrücken , 1-49

[Cassels] J.W.S. Cassels , *An Introduction to the geometry of numbers* , Springer-Verlag , Berlin-Göttingen-Heidelberg , 1959 , 1-26

[Cohen] Henri Cohen , *A Course in computational algebraic number theory* Springer Verlag , Berlin-Heidelberg-New York , 1993 , 17-18

[Cor/Leis/Riv] Thomas H. Cormen,Charles E. Leiserson,Ronald L. Rivest , *Introduction to algorithms* , MIT Press , 1990 , 1-98 , 140-152 , 801-830

[Dom/Kan/Tro] P.D. Domich,R. Kannan,L.E. Trotter Jr. , *Hermite normal form computation using modular determinant arithmetic* , Mathematics of Operations Research , Vol. 12 , No.1 , February 1987 , 50-59

[Edgar] Hugh M.Edgar , *A first Course in Number Theory* , Wadsworth Publishing Company , Belmont California , 1988 , 37-81

[Engel] Arthur Engel , *Elementarmathematik vom algorithmischen Standpunkt* Klett-Studienbücher , 1977 , 49-55

[Fischer] Gerd Fischer , *Lineare Algebra* , Vieweg Studium , Grundkurs Mathematik , 9.Auflage , 1989 , 60-104,130-162

[Grah/Knuth/Pat] Ronald L. Graham,Donald E. Knuth,Oren Patashnik , *Concrete Mathematics* , Addison-Wesley Publishing Company , 1990 , 67-101

[Gro/Lo/Sch] Martin Grötschel,László Lovász,Alexander Schrijver , *Geometric Algorithms and Combinatorial Optimization* , Algorithms and Combinatorics 2 , Springer-Verlag Berlin-Heidelberg-New York , 1988 , 133-156

[Grub/Lek] P.M. Gruber,C.G. Lekkerkerker , *Geometry of numbers* , Mathematical Library , North-Holland Publishing Company , Vol. 37 , 1987 , 17-24

[Hafn/Curl] James L. Hafner,Kevin S. McCurley , *Asymptotically fast triangularization of matrices over rings* , SIAM Journal of Computation , 1991 , Vol.20 , 1068-1083

[Hav/Maj] George Havas,Bodan S. Majewski , *On the greatest common divisor of sets of integers* , Extended Abstract , Key Centre for Software Technology , Department of Computer Science , The University of Queensland Australia , 1994 , to appear

[Irel/Rosen] Kenneth Iredland,Michael Rosen , *A Classical Introduction to Modern Number Theory* ,Springer-Verlag , New York-Berlin-Heidelberg , 1982 Graduate Texts in Mathematics , Volume 84 , 28-38

[Knuth] Donald E. Knuth , *The Art of Computer Programming* , Volume 2 , Addison-Wesley Publishing Company , 1981 , 250-278 , 313-364

[Lampr] Lamprecht , *Einführung in die Algebra* , Birkhäuser Verlag , Basel , 1978 , 7-17 , 247-250

[Mehlh] Kurt Mehlhorn , *Datenstrukturen und effiziente Algorithmen* , Band 1 B.G.Teubner Stuttgart , 2.Auflage , 1988 , 1-10

[Newman] Morris Newman , *Integral Matrices* , Academic Press , New York and London , 1972 , Pure and applied mathematics , Volume 45 , 1-18,88-90

[Rosser/Schoe] J.B. Rosser,L. Schoenfeld , *Approximate formulas for some functions of prime numbers* , Illinois Journal of Mathematics , Vol.6 , 1962 , 64-94