Skriptum Diskrete Mathematik

Sommersemester 2004

Thorsten Theobald

Inhaltsverzeichnis

Stand: 14. Juli 2004

1. Einleitung und Überblick

Gegenstand der diskreten Mathematik sind

- endliche Mengen;
- Strukturen, die „diskret" im Raum liegen (z.B. $\mathbb{Z} \subset \mathbb{R}$).

Aufgrund der engen Beziehung zu algorithmischen Fragen und zu Computeranwendungen hat sich die diskrete Mathematik als ein sehr wichtiges mathematisches Teilgebiet etabliert. In der Vorlesung, die sich an Studierende der Mathematik und Informatik vor dem Vordiplom richtet, wird eine Einführung in dieses Gebiet gegeben.

Überblick:

I: Diskrete algebraische Strukturen

- $\mathbb{Z}$; Euklidischer Algorithmus
- Restklassenring $\mathbb{Z}_m$
- Modulare Arithmetik
- Kryptographie: RSA- Codier- und Unterschriftenschema (Rivest, Shamir, Adleman; Turing Award 2002)
- Endliche Körper
- Fehlerkorrigierende Codes (BCH; Bose, Ray-Chaudhuri, Hocquenghem; Verwendung z.B. bei CDs)

II: Diskrete geometrische Strukturen und kombinatorische Optimierung

- Polytope
- Kombinatorische Optimierung

Literatur:

- Lehrbücher zum Thema Diskrete Mathematik (jeweils neueste Auflagen)
  - M. Aigner: Diskrete Mathematik (Vieweg)
  - T. Ihringer: Diskrete Mathematik (Teubner)
  - D. Lau: Algebra und Diskrete Mathematik 2 (Springer, 2004)
  - N.L. Biggs: Discrete Mathematics (Oxford University Press)
  - B. Korte, J. Vygen: Combinatorial Optimization (Springer, 2000)
  - A. Schrijver: Combinatorial Optimization (Springer, 2003)
  - ...
- Vorlesungsskripten von Prof. Kersting und Prof. Schnorr

Teil 1

Diskrete algebraische Strukturen

2. Der Euklidische Algorithmus

Der Euklidische Algorithmus zur Berechnung größter gemeinsamer Teiler gehört zu den ältesten Rechenverfahren. Er war schon Eudoxus (375 v. Chr.) bekannt und ist im Band 7 der „Elemente“ von Euklid (300 v. Chr.) beschrieben. Er ist von fundamentaler Bedeutung und kommt in vielen Rechenprozeduren zur Anwendung. Wir schreiben $a|b$ für ganze Zahlen a, b, falls a ein Teiler von b ist, d.h., falls es ein $k \in \mathbb{Z}$ mit $b = k \cdot a$ gibt.

Definition 2.1. Eine Zahl $d \in \mathbb{N}$ heißt *größter gemeinsamer Teiler (ggT)* der ganzen Zahlen $a_1, \ldots, a_n \neq 0$, falls

i) $d|a_1, \ldots, d|a_n$,
ii) $z|a_1, \ldots, z|a_n \implies z|d$ für alle $z \in \mathbb{Z}$.

Schreibweise: $d = \mathrm{ggT}(a_1, \ldots, a_n)$. Gilt $\mathrm{ggT}(a_1, \ldots, a_n) = 1$, so werden $a_1, \ldots, a_n$ *relativ prim* oder *teilerfremd* genannt.

Fragen:

(1) Existiert immer ein ggT, und ist er eindeutig bestimmt?
(2) Läßt sich der ggT durch die gegebenen Zahlen ausdrücken?

Eindeutigkeit: Sei d ein ggT von $a_1, \ldots, a_n$. Aus $z|d$ folgt $z \leq d$. In diesem Sinne ist d am größten unter allen gemeinsamen Teilern von $a_1, \ldots, a_n$. Ist d' ein weiterer ggT für $a_1, \ldots, a_n$, so teilen sich d und d' gegenseitig, so dass $d' = d$.

Weniger evident ist, dass immer ein größter gemeinsamer Teiler existiert, etwa für zwei natürliche Zahlen $a, b \in \mathbb{N}$. Eine Möglichkeit besteht darin, a und b in Primfaktoren zu zerlegen. Seien $p_1, \ldots, p_r$ Primzahlen, die in a oder b als Teiler enthalten sind. Dann gibt es Zahlen $e_1, \ldots, e_r, f_1, \ldots, f_r \geq 0$, so dass $a = p_1^{e_1} \cdots p_r^{e_r}$. Die gemeinsamen Teiler von a und b sind dann von der Gestalt $z = \pm p_1^{g_1} \cdots p_r^{g_r}$ mit $g_i \in \{0, \ldots, m_i\}$, $m_i = \min\{e_i, f_i\}$, und der größte gemeinsame Teiler von a und b ist $d = p_1^{m_1} \cdots p_r^{m_r}$. Diese Überlegung macht davon Gebrauch, dass man ganze Zahlen in eindeutiger Weise in Primfaktoren zerlegen kann. Vom algorithmischen Standpunkt ist dieses Vorgehen nicht befriedigend, da die Zerlegung einer Zahl in ihre Primfaktoren sehr rechenaufwendig ist.

Wir gehen hier anders vor und klären die Existenzfrage, indem wir ein Rechenverfahren angeben, das größte gemeinsame Teiler liefert. Es beruht auf einer grundlegenden Eigenschaft ganzer Zahlen, der Division mit Rest. Zu ganzen Zahlen $a, b \neq 0$ gibt es $m, r \in \mathbb{Z}$ mit

$$a = mb + r \quad \text{und } 0 \leq r < |b| .$$

Euklidischer Algorithmus:

Eingabe: $a, b \in \mathbb{Z} \setminus \{0\}$.
Ausgabe: $r_{j-1} = \mathrm{ggT}(a, b)$.

Verfahren: Setze $r_{-1} = a$, $r_0 = b$. Bestimme durch Division mit Rest sukzessive $r_1, \ldots, r_{j-1}$ mit $|b| > r_1 > \cdots > r_{j-1} > r_j = 0$, bis kein Rest mehr bleibt. r_{i+1} sei also der Rest, der bei Division von r_{i-1} durch r_i entsteht:

$$\begin{aligned} r_{-1} &= m_1 r_0 + r_1\,, \\ r_0 &= m_2 r_1 + r_2\,, \\ &\vdots \\ r_{i-1} &= m_{i+1} r_i + r_{i+1}\,, \\ &\vdots \\ r_{j-3} &= m_{j-1} r_{j-2} + r_{j-1}\,, \\ r_{j-2} &= m_j r_{j-1}\,, \end{aligned}$$

mit $m_1, \ldots, m_j \in \mathbb{Z}$.

Termination: Da die Divisionsreste r_i strikt fallen, bricht das Verfahren nach endlich vielen Schritten ab.

Korrektheit: r_{j-1} teilt der Reihe nach $r_{j-2}, r_{j-3}, \ldots, r_0 = b$ und $r_{-1} = a$, wie sich sukzessive aus den Gleichungen $r_{i-1} = m_{i+1} r_i + r_{i+1}$ ergibt. Teilt umgekehrt z sowohl a als auch b, so teilt z der Reihe nach $r_1, \ldots, r_{j-1}$ wie aus den Gleichungen $r_{i+1} = r_{i-1} - m_{i+1} r_i$ folgt.

Beispiel. Für $a = 9876$, $b = 3456$ ergibt sich

$$\begin{aligned} 9876 &= 2 \cdot 3456 + 2964\,, \\ 3456 &= 1 \cdot 2964 + 492\,, \\ 2964 &= 6 \cdot 492 + 12\,, \\ 492 &= 41 \cdot 12\,(+0)\,. \end{aligned}$$

D.h. $\mathrm{ggT}(9876, 3456) = 12$.

Dabei haben wir den *Satz von Bézout* für den Fall $n = 2$ gezeigt.

Satz 2.2. (Bézout) *Zu $a_1, \ldots, a_n \in \mathbb{Z}$ gibt es $\lambda_1, \ldots, \lambda_n \in \mathbb{Z}$ mit* $\mathrm{ggT}(a_1, \ldots, a_n) = \lambda_1 a_1 + \cdots + \lambda_n a_n$.

Beweis. Der Fall $n = 2$ ist bereits geklärt. Den Fall $n \geq 2$ kann man induktiv abhandeln: Sei $d' = \lambda'_1 a_1 + \ldots + \lambda'_{n-1} a_{n-1}$ der ggT von $a_1, \ldots, a_{n-1}$ und $d = \mu_1 d' + \mu_2 a_n$ der ggT von d' und a_n. Dann ist d der ggT von $a_1, \ldots, a_n$ und als ganzzahlige Linearkombination von $a_1, \ldots, a_n$ darstellbar. □

Durch Erweiterung des Euklidischen Algorithmus kann man gleichzeitig mit dem ggT zweier Zahlen a und b auch eine Darstellung nach dem Satz von Bézout gewinnen. Man bestimmt dazu ganze Zahlen

$$\begin{aligned} s_{-1} &= 1\,,\ s_0 = 0\,, & t_{-1} &= 0\,,\ t_0 = 1\,, \\ s_{i-1} &= m_{i+1} s_i + s_{i+1}\,, & t_{i-1} &= m_{i+1} t_i + t_{i+1}\,, \end{aligned}$$

unter Benutzung der vom Euklidischen Algorithmus gewonnenen ganzen Zahlen $m_1, \ldots, m_j$. Dann gilt

$$\mathrm{ggT}(a,b) \;=\; r_{j-1} \;=\; as_{j-1} + bt_{j-1}\,.$$

Beweis. Es gilt sogar $r_i = as_i + bt_i$ für alle $-1 \le i < j$. Für $i = -1, 0$ folgt dies aus der Wahl von s_{-1}, s_0, t_{-1}, t_0, und der Induktionsschritt folgt aus

$$\begin{aligned} r_{i+1} &= r_{i-1} - m_{i+1} r_i \\ &= as_{i-1} + bt_{i-1} - m_{i+1}(as_i + bt_i) \\ &= as_{i+1} + bt_{i+1}\,. \end{aligned}$$

□

Beispiel. $a = 9876$, $b = 3456$.

i	r_{i-1}	r_i	m_{i+1}	s_{i-1}	s_i	t_{i-1}	t_i
0	9876	3456	2	1	0	0	1
				/		/	
1	3456	2964	1	0	1	1	-2
2	2964	492	6	1	-1	-2	3
3	492	$\underline{12}$	41	-1	$\underline{7}$	2	$\underline{-20}$

Also $\mathrm{ggT}(9876, 3456) = 12 = 7 \cdot 9876 - 20 \cdot 3456$.

Laufzeit des Euklidischen Algorithmus. Der Euklidische Algorithmus ist effizient in dem Sinne, dass die Anzahl der benötigten Divisionen logarithmisch in in den Eingabedaten a und b beschränkt ist. Die worst-case Laufzeit des Algorithmus kann abgeschätzt werden, indem Eingaben a und b betrachtet werden, für die besonders viele Divisionen anfallen. Ohne Einschränkung sei $a > b > 0$. (Im Fall $|b| > |a|$ werden a und b im ersten Schritt vertauscht.) Das kleinste Paar (a, b) (formal im Sinne $(a_1, b_1) < (a_2, b_2) \iff (b_1 < b_2$ oder $(b_1 = b_2$ und $a_1 < a_2)))$, für das j Divisionen benötigt werden, ergibt sich, wenn man r_{j-1} und die m_i möglichst klein wählt: $m_1 = \ldots = m_{j-1} = 1$, $m_j = 2$ und $r_{j-1} = 1$ ($m_j = 1$ ist ausgeschlossen, da mit $r_j = 0$ dann $r_{j-1} = r_{j-2}$ folgen würde). Die Gleichungen

$$r_{j-1} = 1\,, \quad r_{j-2} = 2\,, \quad r_{i-1} = r_i + r_{i+1} \text{ für } i = j-2, \ldots, 0$$

bestimmen dann $r_{-1} = a$ und $r_0 = b$ eindeutig. Es ergibt sich ein Zusammenhang zur Fibonacci-Folge $0, 1, 1, 2, 3, 5, 8, 13 \ldots$.

Satz 2.3. *Der Euklidische Algorithmus benötigt bei der Eingabe $a > b > 0$ höchstens $c \ln(b\sqrt{5})$ Divisionen, mit $c = (\ln \frac{1+\sqrt{5}}{2})^{-1} \approx 2.08$.*

Beweis. Siehe Übungen. □

3. Euklidische Ringe

Eine Division mit Rest hat man nicht nur für die ganzen Zahlen, sondern auch für wichtige andere Strukturen (z.B. Polynomringe, siehe Übungen). Allgemein nennt man die Bereiche, in denen eine Division mit Rest möglich ist, *Euklidische Ringe.* Wir rekapitulieren kurz die relevante Terminologie aus der Algebra.

Definition 3.1. Eine nichtleere Menge G mit einer binären Verknüpfung $\circ$ heißt *Gruppe*, wenn folgende Bedingungen erfüllt sind.

i) $\forall a, b, c \in G\,:\,(a \circ b) \circ c = (a \circ b) \circ c$ (Assoziativität) ,
ii) $\forall a \in G\,:\,\exists e \in G\,:\,e \circ a = a \circ e = a$ (neutrales Element) ,
iii) $\forall a \in G \quad \exists b \in G \quad a \circ b = b \circ a = e$ (inverses Element; Schreibweise a^{-1}) .

Gilt darüber hinaus Kommutativität (d.h. $a \circ b = b \circ a\ \forall a, b \in G$), dann heißt G *abelsch.*

Gelten i) und ii), dann heißt $(G, \circ)$ eine Halbgruppe.

Definition 3.2. Eine nichtleere Menge R zusammen mit zwei binären Operationen $+$ und $\cdot$ („Addition" und „Multiplikation") heißt Ring, wenn gilt:

i) $(R, +)$ ist eine abelsche Gruppe, mit neutralem Element 0;
ii) $(R, \cdot)$ ist eine Halbgruppe;
iii) Es gelten die Distributivgesetze $a(b + c) = ab + ac$, $(a + b)c = ab + ac$.

Ein Ring heißt *kommutativ*, wenn die Multiplikation kommutativ ist.

Ein *Einselement* $1 \in R \setminus \{0\}$ in einem Ring ist ein neutrales Element bezüglich der Multiplikation. Sofern nicht ausdrücklich auf das Gegenteil hingewiesen wird, sind alle konkreten Ringe, denen wir begegnen, kommutative Ringe mit Einselement. $R^* := \{a \in \mathbb{R}\,:\,\exists b \in R \text{ mit } ab = 1\}$ heitßt die Einheitengruppe von R. Ist $(R \setminus \{0\}, \cdot)$ eine abelsche Gruppe, dann bildet $(R, +, \cdot)$ einen *Körper.*

Es gibt Ringe, in denen $ab = 0$ mit $a, b \neq 0$ gelten kann. a und b heißen dann *Nullteiler.* In Ringen ohne Nullteiler darf man kürzen, d.h. aus $ab = bc$ folgt $(a - b)c = 0$ und damit $a = b$. Ein kommutativer Ring ohne Nullteiler heißt *Integritätsbereich.*

Wir schreiben $a|b$, wenn es ein $c \in R$ mit $ac = b$ gibt.

Beispiel 3.3. (Der Polynomring $R[x]$).
Sei R ein Ring. Dann is auch die Menge aller (formalen) Polynome $a_n x^n + \ldots + a_1 x + a_0$ mit $a_i \in R$ ein Ring. Zwei Polynome gelten als gleich, wenn sie sich nur um Summanden unterscheiden, deren Koeffizienten 0 sind (die Polynome 0 und $x^2 + x$ in $\mathbb{Z}_2[x]$, die beide überall Null sind, sind beispielsweise sind diesem Sinne verschieden.) Addition und

Multiplikation zweier Polynome $f = \sum_{i=0}^{n} a_i x_i$ und $g = \sum_{j=0}^{n} b_j x^j$ sind definiert durch

$$\begin{aligned} f + g &:= \sum_{i=0}^{n} (a_i + b_i) x^i \quad \text{(o.B.d.A. } m = n) , \\ f \cdot g &:= \sum_{i=0}^{m+n} c_i x^i \quad \text{mit } c_i := \sum_{j+k=i} a_j b_k \, . \end{aligned}$$

R ist in $R[x]$ durch die Polynome vom Grad 0 eingebettet. Ein Einselement in R ist auch ein Einselement in $R[x]$, und für Integritätsbereiche R gilt $R[x]^* = R^*$. Man sagt, $R[x]$ entsteht aus R durch Adjunktion einer Unbestimmten x.

Definition 3.4. Ein Integritätsbereich R heißt *Euklidischer Ring*, falls eine Funktion $g : R \setminus \{0\} \to \mathbb{N}$ mit der folgenden Eigenschaft existiert: Zu beliebigen $a, b \in R \setminus \{0\}$ existieren $m, r \in R$, so dass $a = mb + r$ mit entweder $r = 0$ oder $g(r) < g(b)$.

Satz 3.5. *Sei K ein Körper. Dann ist der Polynomring $K[x]$ ein Euklidischer Ring.*

Beweis. Siehe Übungen. □

Die Überlegungen zu größten gemeinsamen Teilern übertragen sich auf Euklidische Ringe. Man hat erneut den Euklidischen Algorithmus zur Verfügung, daher existieren größte gemeinsame Teiler im folgenden Sinne:

Definition 3.6. Sei R ein Euklidischer Ring. $d \in R$ heißt *ein größter gemeinsamer Teiler von* $a_1, \dots, a_n \in R \setminus \{0\}$, falls

i) $d|a_1, \dots, d|a_n$,
ii) $z|a_1, \dots, z|a_n \implies z|d$ für alle $z \in R$.

Man beachte, dass wir bei der Definition 2.1 für den ggT in $\mathbb{Z}$ die Konvention getroffen haben, dass der ggT positiv ist. Auch in beliebigen Euklidischen Ringen gilt der Satz von Bézout: Für $a, b \neq 0$ und d ein ggT von a und b gibt es $s, t \in R$, so dass $d = sa + tb$.

Außerdem enthält jeder Euklidische Ring nach Definition ein Einselement. Um es zu erhalten, wählen wir ein $b \in R \setminus \{0\}$, für das $g(b)$ minimal ist. Wenn wir $a \in R$ durch b mit Rest teilen, folgt $a = mb$ für ein $m \in R$, denn wegen der Minimalität von $g(b)$ verschwindet bei der Division der Rest. Insbesondere können wir b durch sich selbst teilen: $b = be$ für ein $e \neq 0$. e ist das gesuchte Einselement, denn es folgt $a = mb = mbe = ae$. Man überzeugt sich leicht, dass größte gemeinsame Teiler bis auf Einheiten (d.h. invertierbare Elemente in R) eindeutig bestimmt sind.

4. Der Restklassenring $\mathbb{Z}_m$

Beim Rechnen mit ganzen Zahlen ist es häufig von Vorteil, nicht mit den Zahlen selbst zu operieren, sondern mit den Resten, die beim Teilen der Zahl durch einen fest vorgegebenen *Modul* $m \geq 2$ übrigbleiben.

DEFINITION 4.1. Sei $m \in \mathbb{N}$ und $a, b \in \mathbb{Z}$.

i) a und b heißen *kongruent modulo* m, falls $m|(b-a)$, falls also a und b denselben Rest bei Division durch m haben. Schreibweise: $a \equiv b \pmod{m}$.
ii) Die Menge $\overline{a} := a + m\mathbb{Z} = \{a + mz \,:\, z \in \mathbb{Z}\}$ heißt *Restklasse* von a modulo m. Die Menge aller Restklassen wird mit $\mathbb{Z}_m$ oder $\mathbb{Z}/m\mathbb{Z}$ bezeichnet.

Kongruenz ist eine Äquivalenzrelation. Das nachfolgende Lemma besagt, dass sich das Rechnen mit ganzen Zahlen auf die Restklassen überträgt.

LEMMA 4.2. *Es gilt*

$$a \equiv a',\, b \equiv b' \pmod{m} \implies a + b \equiv a' + b', ab \equiv a'b' \pmod{m},$$

so dass $\overline{a} + \overline{b} := \overline{a+b}$ *und* $\overline{a} \cdot \overline{b} := \overline{a \cdot b}$ *wohldefinierte Verknüpfungen in* $\mathbb{Z}_m$ *sind. Damit wird* $\mathbb{Z}_m$ *zu einem kommutativen Ring mit Einselement.*

BEWEIS. $m|(a-a')$ und $m|(b-b')$ implizieren $m|(a+b-(a'+b'))$, und wegen $ab-a'b' = a(b-b') + (a-a')b'$, auch $m|(ab-a'b')$. Damit ist die erste Behauptung gezeigt. Die Rechenregeln übertragen sich von $\mathbb{Z}$ unmittelbar auf $\mathbb{Z}_m$, z.B.

$$\overline{a} + (\overline{b} + \overline{c}) \;=\; \overline{a} + \overline{b+c} \;=\; \overline{a + (b+c)} \;=\; \overline{(a+b)+c} = (\overline{a} + \overline{b}) + \overline{c}\,.$$

Das Nullelement ist $\overline{0}$, das Einselement ist $\overline{1}$. □

BEISPIEL 4.3. *(Neunerprobe.)* Seien $a = \sum a_i 10^i$ und $b = \sum b_i 10^i$ ganze Zahlen in Dezimaldarstellung und $c = \sum c_i 10^i$ ihr Produkt. Dann gilt wegen $10 \equiv 1 \pmod{9}$

$$\sum a_i \sum b_i \equiv ab \equiv c \equiv \sum c_i \pmod{9}.$$

Die Neunerprobe zur Kontrolle von Multiplikationen (bei der jede Zahl durch die Summe ihrer Ziffern aus ihrer Dezimaldarstellung ersetzt wird) beruht auf dieser Feststellung.

$\mathbb{Z}_m$ wird als *Restklassenring modulo m* bezeichnet. Diese Ringe haben Besonderheiten, wie man sie von $\mathbb{Z}$ nicht kennt.

Nullteiler: Im Gegensatz zu $\mathbb{Z}$ kann $\mathbb{Z}_m$ Nullteiler enthalten, z.B. $\overline{2} \cdot \overline{3} = \overline{0}$ in $\mathbb{Z}_6$. Genauer gilt für $m \geq 2$: $\mathbb{Z}_m$ enthält Nullteiler $\iff$ m ist keine Primzahl.

Einheiten: Sind a und m teilerfremd, dann gibt es ein $\overline{b} \in \mathbb{Z}_m$ mit $a \cdot b \equiv 1 \pmod{m}$. Diese Eigenschaft folgt unmittelbar aus dem Satz von Bézout, der eine Darstellung der Form $sa + tm = 1$ mit $s, t \in \mathbb{Z}$ garantiert, d.h. für $b := s$ gilt $ab \equiv 1 \pmod{m}$.

SATZ 4.4. *Sei* $p \geq 2$. $\mathbb{Z}_p$ *genau dann ein Körper ist, wenn* p *prim ist.*

BEWEIS. Siehe Übungen □

5. Der Chinesische Restsatz

Mit Hilfe des nachfolgend diskutierten Chinesischen Restsatzes ist es möglich, Berechnungsprobleme in kleinere Probleme aufzuteilen. Hierzu wird eine ganze Zahl a durch ein Zahlentupel $(a_1, \ldots, a_k)$ ersetzt, das man dadurch erhält, dass man a modulo k vorgegebener, paarweise teilerfremder Moduln $m_1, \ldots, m_k$ betrachtet. Da sich das Rechnen mit a auf kanonischer Weise auf die a_i überträgt, kann daher mit den Resten modulo m_i gearbeitet werden.

Diese Strategie setzt voraus, dass man am Ende einer Rechnung eine Zahl wieder aus den Resten zurückgewinnen kann. Eine Antwort auf diese Frage der Rekonstruierbarkeit wird durch den Chinesischen Restsatz gegeben, der in einem speziellen Fall bereits Sun Tsu etwa 300 n. Chr. bekannt war.

Satz 5.1. *Seien $m_1, \ldots, m_k \in \mathbb{N}$ paarweise teilerfremd und $a_1, \ldots, a_k \in \mathbb{Z}$. Dann existiert eine Lösung a des Systems von Kongruenzen*

$$x \equiv a_1 \pmod{m_1}, \quad x \equiv a_2 \pmod{m_2}, \quad \ldots, \quad x \equiv a_k \pmod{m_k},$$

und sie ist modulo $m := m_1 \cdot \ldots \cdot m_k$ eindeutig.

Beweis. *Eindeutigkeit:* Sind x und x' Lösungen, d.h. $x \equiv x' \equiv a_i \pmod{m_i}$, so gilt $m_i | (x - x')$ für alle i. Aus der Eindeutigkeit der Primfaktorzerlegung und der paarweisen Teilerfremdheit der m_i folgt $m | (x - x')$, d.h. $x \equiv x' \pmod{m}$.

Existenz: Der nachfolgende Existenzbeweis liefert gleichzeitig ein algorithmisches Verfahren zur Bestimmung von x. Seien e_i, $1 \le i \le k$, Lösungen des Gleichungssystems für den speziellen Fall $a_i = 1$, $a_j = 0$ für $j \neq i$ („Basislösungen“). Setze

$$m_i' := \prod_{j \neq i} m_j = \frac{m}{m_i}.$$

Da die Moduln $m_1, \ldots, m_k$ paarweise teilerfremd sind, gilt $(m_i, m_i') = 1$. Nach dem Satz von Bézout existieren daher ganze Zahlen s_i, t_i, so dass $1 = m_i s_i + m_i' t_i$, und diese Zahlen können mit dem erweiterten Euklidischen Algorithmus berechnet werden. Wir setzen nun

$$e_i := m_i' t_i = 1 - m_i s_i.$$

Dann gilt nach Definition von m_i' wie gewünscht

$$e_i \equiv \begin{cases} 1 \pmod{m_i}, \\ 0 \pmod{m_j} \quad \text{für } j \neq i. \end{cases}$$

Aus den Basislösungen können wir nun eine Lösung

$$x = \sum_{i=1}^{k} a_i e_i$$

für das Ausgangssystem zusammensetzen. □

In Restklassen ausgedrückt bedeutet das:

KOROLLAR 5.2. *Seien* $m_1, \ldots, m_k \in \mathbb{N}$ *paarweise teilerfremd und* $m := m_1 \cdot \ldots \cdot m_k$. *Dann ist die Abbildung*

$$\mathbb{Z}_m \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}, \quad a \bmod m \mapsto (a \bmod m_1, \ldots, a \bmod m_k)$$

bijektiv.

BEWEIS. Die Injektivität folgt aus der Eindeutigkeitssaussage. Die Surjektivität folgt alternativ aus der im Beweis angegebenen Berechnungsmethode oder aus der Tatsache, dass $\mathbb{Z}_m$ und $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}$ gleichmächtig sind (und damit die Injektivität unmittelbar die Surjektivität impliziert). $\square$

BEISPIEL. Gesucht ist eine Lösung des Systems

$$x \equiv 2 \pmod 3, \quad x \equiv 3 \pmod 5, \quad x \equiv 4 \pmod 7.$$

Es gilt $m_1' = 5 \cdot 7 = 35$, $m_2' = 3 \cdot 7 = 21$, $m_3' = 3 \cdot 5 = 15$. Aus

$$\begin{array}{rcccl} 1 &=& (3,35) &=& 12 \cdot 3 - 1 \cdot 35\,, \\ 1 &=& (5,21) &=& -4 \cdot 5 + 1 \cdot 21\,, \\ 1 &=& (7,15) &=& -2 \cdot 7 + 1 \cdot 15 \end{array}$$

erhalten wir die Basislösungen $e_1 = -35$, $e_2 = 21$, $e_3 = 15$. Folglich ist

$$2 \cdot (-35) + 3 \cdot 21 + 4 \cdot 15 = 53$$

Lösung der Kongruenz.

Anwendung: *Ein probabilistischer Gleichheitstest.*

Zwei Personen an den Enden eines Nachrichtenkanals wollen zwei binäre Nachrichten der Länge < 10000 auf Gleichheit hin überprüfen. Dabei sollen möglichst wenige Bits übertragen werden. Das folgende Verfahren erlaubt einen Vergleich der beiden als Zahlen $a, b < 2^{10000}$ interpretierbaren Nachrichten, wobei anstatt der bis zu 10000 Bits für die gesamte Nachricht nur $k \cdot 101$ (bzw. $k \cdot 202$ bei erforderlicher Übertragung der Moduln) Bits gesendet werden. Wir werden sehen, daß das Verfahren schon für $k = 1$ höchste Sicherheit garantiert.

PROBABILISTISCHER GLEICHHEITSTEST:

Ausgabe: „$a \neq b$" oder „mit großer Wahrscheinlichkeit $a = b$"

Verfahren: Wähle zufällig Primzahlen $p_1, \ldots, p_k \in [2^{100}, 2^{101})$. Übertrage a modulo p_i für alle $i = 1, \ldots, k$. Falls $a \not\equiv b \pmod{p_i}$ für ein i, gilt „$a \neq b$". Ansonsten treffe die Entscheidung „mit großer Wahrscheinlichkeit $a = b$"

Das Verfahren setzt voraus, dass man sich die benötigten Primzahlen leicht verschaffen kann. Darauf kommen wir später zurück.

Wir schätzen nun die Wahrscheinlichkeit ab, dass das Verfahren zu einer Fehlentscheidung führt. Nehmen wir dazu zunächst an, dass es 100 verschiedene Primzahlen

$q_1, \ldots, q_{100} \in [2^{100}, 2^{101})$ gibt, für die $a \equiv b \pmod{q_i}$ gilt. Nach dem Chinesischen Restsatz folgt $a \equiv b \pmod m$, mit $m := q_1 \cdot \ldots \cdot q_{100}$. Es gilt $m > (2^{100})^{100} = 2^{10000}$, nach Annahme folgt daher $a = b$. In diesem Fall ist eine Fehlentscheidung also ausgeschlossen.

Eine Fehlentscheidung ist also nur möglich, falls es weniger als 100 Primzahlen $q > 2^{100}$ mit der Eigenschaft $a \equiv b \pmod q$ gibt, und eine Fehlentscheidung tritt nur dann ein, wenn das Verfahren zufälligerweise nur derartige Primzahlen auswählt. Nach dem berühmten Primzahlsatz gilt für die Anzahl $\pi(x)$ der Primzahlen $\leq x$ die asymptotische Formel $\pi(x) \sim x/\ln x$. Zwischen 2^{100} und 2^{101} gibt es daher approximativ

$$\pi(2^{101}) - \pi(2^{100}) \approx \frac{2^{101}}{\ln 2^{101}} - \frac{2^{100}}{\ln 2^{100}} \geq \frac{1}{\ln 2^{100}}(2^{101} - 2^{100}) = \frac{2^{100}}{100 \ln 2} \geq 2^{93}\,.$$

Primzahlen. Bei k-facher unabhängiger Wahl einer Primzahl ist die Fehlerwahrscheinlichkeit also höchstens

$$\left(\frac{99}{2^{93}}\right)^k \leq 2^{-86}\,.$$

Schon für $k = 1$ ist dies ein verschwindend kleiner Wert. Diese Schranke für die Fehlerwahrscheinlichkeit ist unabhängig von den Nachrichten a und b. Wenn wir dagegen an zufälligen Bitpositionen prüfen, erkennen wir die Ungleichheit oft nicht, wenn a und b an fast allen Bitpositionen übereinstimmen.

Eine andere Anwendung des chinesischen Restsatzes ist beispielsweise die exakte Lösung großer ganzzahliger linearer Gleichungssysteme mittels modularer Arithmetik.

6. Die Eulersche φ-Funktion und der kleine Satz von Fermat

Die nachfolgende Definition sortiert die Nullteiler aus $\mathbb{Z}_m$ aus.

Definition 6.1. Sei $a \in \mathbb{Z}$. Die Restklasse $\overline{a}$ heißt *prime Restklasse* modulo m, falls $\text{ggT}(a, m) = 1$. Die Menge der primen Restklassen modulo m wird mit $\mathbb{Z}_m^*$ bezeichnet, ihre Anzahl mit $\varphi(m)$. φ heißt die *Eulersche φ-Funktion.*

Beispiel 6.2.

i) Es gilt $(\mathbb{Z}_m^* = \mathbb{Z} \setminus \{0\}$ und $\varphi(m) = m - 1)$ genau dann, wenn m prim ist.
ii) $\varphi(6) = 2$, $\mathbb{Z}_6^* = \{\overline{1}, \overline{5}\}$.
iii) $\varphi(8) = 4$, $\mathbb{Z}_8^* = \{\overline{1}, \overline{3}, \overline{5}, \overline{7}\}$.
iv) $\varphi(12) = 4$, $\mathbb{Z}_{12}^* = \{\overline{1}, \overline{5}, \overline{7}, \overline{11}\}$.

Satz 6.3. *$\mathbb{Z}_m^*$ ist bezüglich Restklassenmultiplikation eine Gruppe, die* Einheitengruppe *von $\mathbb{Z}_m$.*

Beweis. *Multiplikative Abgeschlossenheit:* Seien $a, b \in \mathbb{Z}$ mit $(a, m) = (b, m) = 1$. Dann ist aufgrund der eindeutigen Primfaktorzerlegung $(ab, m) = 1$.

Inverses Element: Für $a \in \mathbb{Z}$ mit $(a, m) = 1$ existieren nach dem Satz von Bézout $r, s \in \mathbb{Z}$ mit $ra + sm = 1$. Folglich gilt modulo m die Kongruenz $\overline{ra} = \overline{ra} + \overline{0} = \overline{1}$, d.h. $\overline{r} = \overline{a}^{-1}$.

Folglich ist $\mathbb{Z}_m^*$ eine Untergruppe von $(\mathbb{Z}_m, \cdot)$. $\square$

Das Rechnen mit primen Restklassen hat wichtige Anwendungen in der Kryptographie, der Lehre von der geheimen Nachrichtenübertragung. Von fundamentaler Bedeutung ist der Satz von Euler. Wir schreiben wie üblich $\overline{a}^n := \underbrace{\overline{a} \cdot \ldots \cdot \overline{a}}_{n-\text{mal}}$.

Satz 6.4. *(Satz von Euler.) Für alle $\overline{a} \in \mathbb{Z}_m^*$ gilt $\overline{a}^{\varphi(m)} = \overline{1}$.*

Beweis. Sei $\mathbb{Z}_m^* =: \{\overline{a}_1, \ldots, \overline{a}_{\varphi(m)}\}$. Ferner sei $\overline{a} \in \mathbb{Z}_m^*$ beliebig. Aus der Bijektivität der Linkstranslation $L : \mathbb{Z}_m^* \to \mathbb{Z}_m^*, x \mapsto ax$ folgt, dass dann auch $\{\overline{a}\ \overline{a}_1, \ldots, \overline{a}\ \overline{a}_{\varphi(m)}\} = \mathbb{Z}_m^*$. Multiplikation jeweils aller Elemente dieser beiden Vertretersysteme liefert

$$\overline{a}_1 \cdot \ldots \cdot \overline{a}_{\varphi(m)} = \overline{a}\ \overline{a}_1 \cdot \ldots \cdot \overline{a}\ \overline{a}_{\varphi(m)}$$

und nach Kürzen $\overline{a}^{\varphi(m)} = \overline{1}$. $\square$

Korollar 6.5. *(Kleiner Satz von Fermat.) Sei p eine Primzahl. Dann gilt für jedes $a \in \mathbb{Z}$*

$$a^p \equiv a \pmod{p}.$$

Beweis. Für $a \equiv 0 \pmod{p}$ ist die Aussage offensichtlich, und für $a \neq 0$ gilt wegen $\varphi(p) = p - 1$ die Kongruenz $a^{p-1} \equiv 1 \pmod{p}$. $\square$

Falls die Primfaktorzerlegung von m bekannt ist, läßt sich $\varphi(m)$ leicht berechnen.

Satz 6.6. *Seien $m \in \mathbb{N}$ und $p_1, \ldots, p_k$ die unterschiedlichen Primteiler von m. Dann gilt*

$$\varphi(m) \ = \ m \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right).$$

Beweis. Aus Beispiel 6.2 wissen wir bereits $\varphi(p) = p - 1$ für p prim.

Ist $m = p^r$ mit primem p, so sind genau die Zahlen $0, p, 2p, \ldots, (p^{r-1} - 1)p$ Vertreter der Restklassen, die nicht zu $\mathbb{Z}_m^*$ gehören. Also

$$\varphi(p^r) = p^r - p^{r-1} = p^r \left(1 - \frac{1}{p}\right).$$

Sei nun $m = m_1 \cdot \ldots \cdot m_k$ mit paarweise teilerfremden Zahlen $m = m_1, \ldots, m_k$. Nach dem Chinesischen Restsatz ist die Abbildung

$$\mathbb{Z}_m \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}, \quad a \bmod m \ \mapsto \ (a \bmod m_1, \ldots, a \bmod m_k)$$

bijektiv. Da $\operatorname{ggT}(a, m) = 1$ genau dann, wenn $\operatorname{ggT}(a, m_i) = 1$ für alle i, ist auch die auf der Einheitengruppe induzierte Abbildung

$$\mathbb{Z}_m^* \to \mathbb{Z}_{m_1}^* \times \cdots \times \mathbb{Z}_{m_k}^*, \quad a \bmod m \ \mapsto \ (a \bmod m_1, \ldots, a \bmod m_k)$$

bijektiv. Es folgt $\varphi(m) = \varphi(m_1) \cdot \ldots \cdot \varphi(m_k)$. $\square$

7. Das RSA-Codier- und Unterschriftenschema

Die *Kryptographie* ist die Lehre von Chiffriersystemen. Wir betrachten die Situation, dass eine Person A eine geheime Nachricht an einer Person B übermitteln möchte. A codiert sie deshalb mit einer injektiven (hier zur Vereinfachung bijektiven) Codierabbildung

$$E : \mathcal{N} \to \mathcal{K} .$$

Anstelle der Nachricht $a \in \mathcal{N}$ sendet A die chiffrierte Nachricht $E(a)$. Der Empfänger decodiert mittels der inversen Abbildung

$$D = E^{-1} : \mathcal{K}' \to \mathcal{N} ,$$

wobei $\mathcal{K}' := E(\mathcal{N}) \subset \mathcal{K}$. Ein bekanntes Verfahren beruht auf der Addition modulo 2. Wir nehmen an, dass die Nachrichten als binäre Folgen der Länge n vorliegen, d.h. $\mathcal{N} = \mathcal{K} = \{0,1\}^n$. Wir fassen die Folgen als Vektoren der Länge n über dem Körper $\mathbb{Z}_2$ auf. Zum Codieren wird ein binärer String $k = k_1 k_2 \dots k_n$ verwendet. Wir setzen

$$E(a) := a + k ,$$

die beiden binären Folgen a und k werden also komponentenweise modulo 2 addiert. Es gilt $D = E$, denn $E + E$ bildet auf die nur aus Nullen bestehenden Folge. Nachrichten werden daher nach demselben Verfahren decodiert. Dieses klassische Verfahren hat den Namen 'One-time-pad'.

Es gilt $a + E(a) = k$. Gelingt es daher, eine codierte Nachricht $E(a)$ zu entschlüsseln, so kennt man k und damit bereits die vollständige Codier- und Decodiervorschrift. Ist k zufällig aus $\{0,1\}$ gewählt, so gilt dies auch für $E(a)$. Dies bedeutet, dass das Verfahren im folgenden Sinne sicher ist: Ein Unbefugter hat keine Chance, eine Nachricht zu dechiffrieren, wenn er auf k keinen Zugriff hat. Denn die gesendeten Nachrichten sind gleichverteilt auf $\{0,1\}^n$.

Das RSA-Schema. Für die klassischen Chiffrierverfahren besteht ein Sicherheitsproblem darin, dass man die Codiervorschrift- und Decodiervorschrift geheimhalten muss (beim One-time-pad also k) und zuvor vereinbaren muss. Diffie und Hellman haben 1976 den (damals völlig neuartigen) Vorschlag der sogenannten *Public-key-Kryptographie* gemacht. Jeder Teilnehmer des Systems besitzt einen öffentlichen Schüssel k und einen geheimen Schlüssel k'. D und E müssen die Eigenschaft haben, dass E leicht berechnet werden kann, D ohne Kenntnis von k' sehr "schwer" berechenbar ist, mit Kenntnis von k' jedoch leicht (*Trapdoor-Funktion*).

Das bekannteste öffentliche Chiffriersystem ist das 1978 von Rivest, Shamir und Adlemen vorgeschlagene *RSA-Schema*. Es beruht darauf, dass es schwer ist, eine Zahl m in ihre Primfaktoren zu zerlegen.

Erzeugen eines Schlüsselspaares: Seien p, q sehr große Primzahlen (z.B. 1024 Bits), $N := pq$. Sei $e \in \mathbb{Z}^*_{\varphi(N)}$, d.h. $\mathrm{ggT}(e, \varphi(N)) = 1$, $d = e^{-1} \pmod{\varphi(N)}$.

Öffentlicher Schlüssel: (N, e).
Geheimer Schlüssel: d.
Codieren: Aufteilen von Nachrichten in Blöcke der Bitlänge $\leq \log_2 N$ und Betrachtung jedes Blockes als Zahl in $\mathbb{Z}_N$. $E : \mathbb{Z}_N \to \mathbb{Z}_N$, $E(x) = x^e \pmod N$.
Decodieren: $D : \mathbb{Z}_N \to \mathbb{Z}_N$, $D(y) = y^d \pmod N$.

LEMMA 7.1. $(x^e)^d \equiv (x^d)^e \equiv x \pmod N$.

BEWEIS. Nach Konstruktion von e und d gilt

$$e \cdot d = 1 + \nu\varphi(N) \quad \text{mit } \nu \in \mathbb{Z}\,.$$

Falls $x \in \mathbb{Z}_N^*$, dann gilt nach dem Satz von Euler

$$x^{ed} = x^{1+\nu\varphi(N)} \equiv x \pmod N\,.$$

Es verbleibt, die Ausage für $x \notin \mathbb{Z}_N^*$ zu zeigen. Nach dem Chinesischen Restsatz ist die Abbildung

$$\begin{array}{rcl} \mathbb{Z}_N & \to & \mathbb{Z}_p \times \mathbb{Z}_q \\ x & \mapsto & (x \bmod p, x \bmod q) \end{array}$$

bijektiv. Für $x = 0$ gilt die Behauptung des Lemmas offensichtlich. Gilt o.B.d.A. $x \equiv 0 \pmod p$ und $x \not\equiv 0 \pmod q$, dann folgt $x^{ed} \equiv 0 \pmod p$ sowie aus dem kleinen Satz von Fermat $x^{ed} \equiv x \pmod q$, d.h., $x^{ed} \equiv x \pmod N$. □

Analyse: Die Kenntnis von $\varphi(N) = (p-1)(q-1)$ ist praktisch äquivalent zur Kenntnis der Faktorisierung von N, denn wegen (o.B.d.A. $p > q$)

$$\begin{array}{rcl} \varphi(N) & = & (p-1)(q-1) \quad \Longrightarrow p + q = N - \varphi(N) + 1 \\ p - q & = & \sqrt{(p+q)^2 - 4pq} \;=\; \sqrt{(n - \varphi(N) + 1)^2 - 4n} \end{array}$$

ist die Bestimmung von $\varphi(N)$ etwa genau so schwierig wie die Primfaktorzerlegung von N – und Faktorisierungsalgorithmen brauchen sehr viel Zeit. Ist also nur N, nicht aber die Faktorisierung bekannt, so kann man d $(= e^{-1} \pmod{\varphi(N)})$ praktisch nicht bestimmen. Ein offenes Problem ist, ob die Decodierung auch ohne Kenntnis von d möglich ist.

Signaturschema: Das RSA-System kann auch zur Beglaubigung von Nachrichten verwendet werden (*Digitale Unterschrift*). Jeder Teilnehmer A und B besitzt einen öffentlichen Schüssel (N_A, e_A) bzw. (N_B, e_B) sowie einen geheimen d_A bzw. d_B.

Nachricht von A an B: $x \mapsto E_B(x) \;=\; x^{e_B}$
Entschlüsselung durch B: $D_B(E_B(x)) \;=\; (x^{e_B})^{d_B} \;=\; x \pmod{N_B}$
Unterschrift von A: durch Mitteilung von $c := D_A^{-1}(x) = x^{d_A} \pmod{N_A}$
Verifikation der Unterschrift durch B: durch Überprüfung, dass $x = c^{e_A} \pmod{N_A}$.

8. Primalitätstests

Für das RSA-Schema benötigt man große zufällige Primzahlen. Da die Primzahlen $\leq N$ etwa die Dichte $\frac{1}{\log N}$ haben, genügt ein effektiver Primalitätstest. Naive Verfahren (etwa „Dividiere einen Kandidaten N durch alle Primzahlen $p \leq \sqrt{N}$“) sind für die relevanten Größenordnungen nicht praktikabel.)

Viele Primalitätstest beruhen auf Ideen des kleinen Satzes von Fermat. Nach diesem gilt: Sei $N \in \mathbb{N}$. Falls ein $a \in \mathbb{N}$, $0 < a < N$ mit $a^{N-1} \not\equiv 1 \pmod N$ existiert, dann ist N keine Primzahl.

Definition 8.1. Sei N eine zusammengesetzte Zahl. N heißt *pseudoprim* zur Basis a, falls N die Eigenschaft $a^{N-1} \equiv 1 \pmod N$ erfüllt. N heißt *Carmichael-Zahl*, falls N pseudoprim für alle $a \in \mathbb{Z}_N^*$ ist.

Ob eine Zahl N Carmichael-Zahl ist, kann man probabilistisch einfach testen. Die $a \in \mathbb{Z}_N^*$, welche die Fermat-Identität erfüllen, bilden eine Untergruppe von $\mathbb{Z}_N^*$. Die Ordnung dieser Untergruppe ist entweder $\varphi(N)$ oder (nach dem Satz von Lagrange) höchstens $\varphi(N)/2$. Explizit kann man dies auch wie folgt sehen: Seien $a_1, \ldots, a_k \in \mathbb{N}$ alle Elemente aus $\mathbb{Z}_N^*$ mit $a_i^{N-1} \equiv 1 \pmod m$. Gibt es ein $a \in \mathbb{Z}_N^*$ mit $a^{N-1} \not\equiv 1 (\mathrm{mod}\ N)$, dann gilt für $b_i := a \cdot a_i$ („Linkstranslation“):

$$b_i^{N-1} = a^{N-1} a_i^{N-1} \not\equiv 1 \pmod N ,$$

es gibt daher mindestens ebenso viele Elemente in $\mathbb{Z}_N^*$, die die Fermatsche Identität nicht erfüllen. D.h. $k \leq \varphi(N)/2$.

Korollar 8.2. (r-facher Fermat-Test.) *Gibt es zu $N \in \mathbb{N}$ ein $a \in \mathbb{Z}_N^*$ mit $a^{N-1} \not\equiv 1 \pmod N$, dann gilt für zufällige, unabhängige $a_1, \ldots, a_r \in \mathbb{Z}_N^*$*

$$\mathrm{Ws}\left(a_i^{N-1} \equiv 1 \pmod N \text{ für } 1 \leq i \leq r\right) \;\leq\; 2^{-r} .$$

Der r-fache Fermat-Test kann die Zusammengesetztheit einer Nicht-Carmichael-Zahl mit Wahrscheinlichkeit $\geq 1 - 2^r$ erkennen und nachweisen. Die Zusammengesetztheit von Carmichael-Zahlen kann er nicht erkennen. Für viele praktische Anwendungen ist der Fermat-Test jedoch bereits ausreichend, da Carmichael-Zahlen nur selten vorkommen. Die kleinste Carmichael-Zahl ist $561 = 3 \cdot 11 \cdot 17$. Bis Anfang der 90er Jahre war unbekannt, ob es überhaupt unendlich viele Carmichael-Zahlen gibt.

Satz 8.3. (Alford, Granville, Pomerance; Annals of Mathematics, 1994). *Es existieren unendlich viele Carmichael-Zahlen.*

Die Anzahl $C(x)$ der Carmichael-Zahlen kleiner oder gleich x ist für hinreichend große x beschränkt durch $x^{2/7} < C(x) < x^{1-(\ln\ln\ln x)/(\ln\ln x)}$.

Der sogenannte *Primzahltest von Miller-Rabin* erweitert den Fermat-Test auf Carmichael-Zahlen. Mit dem Miller-Rabin-Test kann man die Zusammengesetztheit einer nicht primen, ungeraden Zahl mit Wahrscheinlichkeit mindestens $\frac{3}{4}$ in Polynomialzeit beweisen.

Satz 8.4. *Sei $N \in \mathbb{N}$ zusammengesetzt und ungerade, $N-1 = 2^k t$ mit ungeradem t. Dann gilt für zufälliges $a \in \mathbb{Z}_N^*$:*

$$\mathrm{Ws}\Big(a^t \equiv 1 \pmod N \text{ oder } \exists i \in \{0,\dots,k-1\} \quad a^{2^i t} \equiv -1 \pmod N\Big) \;\le\; \frac{1}{4}\,.$$

Beweis. Siehe D.E. Knuth, The Art of Computer Programming, Vol. 2 (1981), Aufgabe 4.5.4 (22) oder R. Crandall, C. Pomerance, Prime Numbers: A computational perspective (2001), Theorem 3.4.4. □

Definition 8.5. Sei $\mathcal{R}$ die Klasse der Sprachen $L \subset \{0,1\}^*$, für die es einen Polynomialzeit-Algorithmus gibt, der zu $x \in L$ mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ einen Beweis für $x \in L$ findet.

Aus dem Miller-Rabin-Test folgt, dass die Menge der zusammengesetzten Zahlen in $\mathcal{R}$ liegt. Umgekehrt zeigen Adleman, Huang (1992), dass es einen probabilistischen Algorithmus polynomialer Laufzeit gibt, der nach k Durchläufen entweder eine definitive Antwort auf die Frage der Primalität einer gegebenen Zahl liefert (Irrtum ausgeschlossen) oder gar keine Antwort gibt (letzteres mit Wahrscheinlichkeit kleiner 2^k). In der Sprache der Komplexitätstheorie heißt das, das die Menge der Primzahlen in der Klasse $\mathcal{ZPP}$ enthalten ist. Zum Erwürfeln eines Primalitätsbeweises konstruiert man geeignete elliptische Kurven; die Beweisskizze von Adleman, Huang ist 140 Seiten lang. In jüngster Zeit wurde schließlich ein deterministischer Polynomialzeit-Primalitätstest gefunden:

Satz 8.6. (Agrawal, Kayal, Saxena, 2002.) *Die Menge der Primzahlen ist in Polynomialzeit entscheidbar.*

Die derzeitigen praktikabelsten Primzahltests für „allgemeine“ Zahlen beruhen auf elliptischen Kurven. Darüber hinaus gibt es spezialisierte Tests für Zahlen besonderer Bauart, insbesondere für Fermat-Zahlen und Mersenne'sche Zahlen. Ein *Fermat-Zahl* ist eine Zahl der Form $2^{2^n}+1$ mit $n \in \mathbb{N}_0$. $F_0 = 3$, $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$ sind prim. Bis heute ist jedoch keine weitere Fermat-Primzahl bekannt. Eine *Mersenne'sche Zahl* ist eine Zahl der Form $2^p - 1$ mit p prim). Diese Zahlen sind nicht immer prim (z.B. $23 | 2^{11} - 1$), Prime Mersennesche Zahlen waren aber oft „Rekord“-Primzahlen.

9. Ideale

In den nachfolgenden Abschnitten zur Codierungstheorie werden einige Tatsachen aus der Ringtheorie benötigt.

Definition 9.1. Eine Teilmenge I eines Rings $(R,+,\cdot)$ heißt *Ideal*, wenn folgende beiden Bedingungen erfüllt sind:

i) $(I,+)$ ist eine Untergruppe von $(R,+)$.
ii) Für alle $r \in R$ gilt $rI \subset I$ und $Ir \subset I$.

Die Bedeutung von Idealen in der Ringtheorie liegt vor allem darin, dass immer „modulo" einem Ideal gerechnet werden kann und auf diese Weise neue Ringe erzeugt werden können. Ist I ein Ideal, dann wird, wie man leicht nachprüft, in R durch $a \equiv b := a-b \in I$ eine Äquivalenzrelation erklärt. Wir schreiben dann

$$a \equiv b \bmod I$$

und nennen a und b *kongruent modulo* I. Die Relation ist mit der Addition und der Multiplikation verträglich, denn aus $a_1 - b_1, a_2 - b_2 \in I$ folgt $a_1 + a_2 - (b_1 + b_2) \in I$ und $a_1 a_2 - b_1 b_2 = a_1(a_2 - b_2) + (a_1 - b_1)b_2 \in I$. Wir können daher (in Analogie zum Restklassenring $\mathbb{Z}_m$) die Restklassen $\overline{a} = a + I$, $a \in R$ betrachten, und die Operationen $\overline{a} + \overline{b} := \overline{a+b}$, $\overline{a} \cdot \overline{b} := \overline{a \cdot b}$ auf den Restklassen sind wohldefiniert. Dieser Ring wird wird als *Restklassenring* oder *Faktorring* R/I bezeichnet. Speziell erhalten wir bei der Wahl $R = \mathbb{Z}$ und $I = m\mathbb{Z}$ den Restklassenring $\mathbb{Z}_m$.

Definition 9.2. Sei R ein Integritätsbereich. Ein Ideal I heißt *Hauptideal*, wenn es ein $m \in R$ gilt, so dass $I = mR = \{mr : r \in R\}$. Dieses von m erzeugte Ideal wird mit (m) bezeichnet. Ein Integritätsbereich R heißt *Hauptidealring*, wenn jedes Ideal von R ein Hauptideal ist.

Satz 9.3. *Jeder Euklidische Ring ist ein Hauptidealring.*

Beweis. Siehe Übungen. □

Korollar 9.4. *Die folgenden Ringe sind Euklidische Ringe und damit Hauptidealringe:*
a) $\mathbb{Z}$;
b) $K[x]$ *für einen Körper* K;
c) $K[x]/(f)$ *für einen Körper* K *und ein Polynom* $f \in K[x]$.

Beweis. a) und b) sind klar. In Analogie zu b) gibt es auch in $K[x]/(f)$ eine Division mit Rest. □

10. Endliche Körper

Für jede Primzahl p gibt es einen endlichen Körper $\mathbb{Z}_p$ mit p Elementen, wir haben ihn aus den ganzen Zahlen durch Restklassenbildung konstruiert. Diese Körper haben sich in ganz unterschiedlicher Weise als nützlich erwiesen. Es stellt sich daher die Frage nach weiteren endlichen Körpern. Diese werde insbesondere für die anschließend betrachteten Fragen der Codierungstheorie sehr nützlich sein.

Definition 10.1. Der Körper K habe das Element 1 als neutrales Element bezüglich der Multiplikation. Das kleinste $p \in \mathbb{N}$ mit

$$\underbrace{1 + 1 + \ldots + 1}_{p-\text{mal}}$$

heitßt *Charakteristik* von K. Schreibweise: $\operatorname{char}(K) = p$. Existiert kein solches $p \in \mathbb{N}$, wird $char(K) = 0$ gesetzt.

DEFINITION 10.2. Ist $(K, +, \cdot)$ ein Körper und L eine Teilmenge von K, die mit $+$ und $\cdot$ selbst ein Körper ist, dann heißt L ein *Unterkörper* von K (und K ein *Oberkörper* oder *Erweiterungskörper* von L).

LEMMA 10.3. *Sei K ein Körper mit endlicher Charakteristik p. Dann gilt:*

i) *p ist eine Primzahl.*
ii) *Für jedes $a \in K$ gilt $\underbrace{a + a + \ldots + a}_{p-mal} = 0$.*
iii) *Die Elemente $0,\ 1,\ 2 := 1 + 1,\ 3 := 1 + 1 + 1,\ \ldots,\ p - 1 := 1 + 1 + \ldots + 1$ bilden einen Unterkörper P von K. Man nennt P den* Primkörper *von K.*

BEWEIS. i) folgt aus der Nullteilerfreiheit des Körpers.
ii) $a + a + \ldots + a = a \cdot (1 + 1 + \ldots + 1) = a \cdot 0 = 0$.
iii) Der Primkörper P von K ist isomorph zu $\mathbb{Z}_p$. □

Nach Lemma 10.3 kann man den Körper K (und jeden seiner Unterkörper) als Vektorraum über $\mathbb{Z}_p$ auffassen. Per Definition ist $(K, +)$, also die Menge der „Vektoren“, eine abelsche Gruppe. Für alle „Skalare“ $\lambda, \mu \in P$ und alle „Vektoren“ $x, y \in K$ gilt außerdem

$$(\lambda + \mu)x = \lambda x + \mu x\,, \quad \lambda(x + y) = \lambda x + \mu y\,, \quad (\lambda\mu)x = \lambda(\mu x)\,, \quad 1x = x\,,$$

denn dasselbe gilt nach Definition sogar für alle $\lambda, \mu \in K$. Damit sind alle Vektorraumaxiome erfüllt. Jeder endliche Vektorraum hat eine endliche Dimension. Ist also K endlich und von der Dimension m über P, mit $|P| = \operatorname{char}(K) = p$, dann gilt $|K| = p^m$.

SATZ 10.4. *Für jeden endlichen Körper ist die Anzahl der Elemente eine Primzahlpotenz.*

Ein endlicher Körper (engl. Galois field) mit q Elementen wird oft mit $\mathrm{GF}(q)$ oder kürzer mit $\mathbb{F}_q$ bezeichnet. Wir werden bald sehen, dass es für jede Primzahlpotenz $q = p^n$ bis auf Isomorphie genau einen Körper mit q Elementen gibt. Die Primkörper können also mit $\mathbb{F}_p$, p prim, bezeichnet werden; sie sind isomorph zu $\mathbb{Z}_p$.

Der Primkörper von K ist der kleinste Unterkörper von K, der das Einselement enthält. Für ein $\alpha \in K$ sei in Verallgemeinerung hierzu $K(\alpha)$ definiert als der kleinste Unterkörper von K, der α enthält.

SATZ 10.5. *Sei K ein endlicher Körper der Charakteristik p und $\alpha \in K$. Dann ist $1, \alpha, \alpha^2, \ldots, \alpha^{m-1}$ eine Basis von $K(\alpha)$, wobei m die Dimension von $K(\alpha)$ über $\mathbb{Z}_p$ bezeichnet.*

BEWEIS. Sei m die größte natürliche Zahl, so dass $1, \alpha, \ldots, \alpha^{m-1}$ linear unabhängig sind.

Zeige: $K' := \{a_0 + a_1\alpha + \ldots + a_{m-1}\alpha^{m-1} : a_0, \ldots, a_{m-1} \in \mathbb{Z}_p\} = K(\alpha)$.

"$\subset$": klar

"$\supset$": Da $\alpha \in K'$, genügt es zu zeigen, dass K' selbst ein Körper ist. Offenbar ist K' unter Addition abgeschlossen und enthält 0 und 1. Nach Definition von m gibt es $b_0, b_1 \ldots, b_{m-1} \in \mathbb{Z}_p$, so dass

$$\alpha^m = -b_0 - b_1\alpha - \ldots - b_{m-1}\alpha^{m-1}.$$

Bei der Multiplikation zweier Elemente aus K' können wir mit dieser Gleichung Potenzen α^r mit $r \geq m$ durch kleinere Potenzen ersetzen. Dies zeigt, dass K' unter Multiplikation abgeschlossen ist. Um zu zeigen, dass jedes Element in K' ein inverses Element besitzt, benutzen wir (ähnlich wie früher für $\mathbb{Z}_p$) den Satz von Bézout. Wir interpretieren die Koeffizienten $a_0, \ldots, a_{m-1}$ einer Linearkombination in der Definition von K' als Polynom $f(x) := a_0 + a_1x + \ldots + a_{m-1}x^{m-1}$ (d.h., das Element in K' ist $f(\alpha)$). Sei $g(x) := b_0 + b_1x + \ldots, b_{m-1}x^{m-1} + x^m$. g ist normiert und irreduzibel, d.h. nicht als Produkt von Polynomen kleineren Grades darstellbar. Aus einer Zerlegung $g(x) = g_1(x) \cdot g_2(x)$ folgt nämlich wegen $g(\alpha) = 0$ unmittelbar $g_1(\alpha) = 0$ oder $g_2(\alpha) = 0$. Sei o.b.d.A. $g_1(\alpha) = 0$. Nach der Maximalität von m muss daher $g_1(x)$ vom Grad m und $g_2(x)$ vom Grad 0 sein, d.h. g ist irreduzibel.

Da $\mathrm{grad}(f) < \mathrm{grad}(g)$, sind f und g teilerfremd. Daher existieren $s, t \in \mathbb{Z}_p[x]$ mit $sf + tg = 1$. Wegen $g(\alpha) = 0$ folgt $1 = s(\alpha)f(\alpha)$, so dass $s(\alpha)$ das inverse Element zu $f(\alpha)$ ist. Andererseits gehört $s(\alpha)$ zu K', denn Potenzen α^r mit $r \geq m$ können wir erneut durch kleinere Potenzen ersetzen.

Es gilt also $K' = K(\alpha)$. m ist dann offenbar die Dimension von $K(\alpha)$ über $\mathbb{Z}_p$. □

Das in dem Beweis gebildete normierte irreduzible Polynom $g(x) = x^m + b^{m-1}x^{m-1} + \ldots + b_1x + b_0 \in \mathbb{Z}_p[x]$ nennt man Minimalpolynom von α über $\mathbb{Z}_p$. Unsere bisherigen Überlegungen sind von der speziellen Wahl der Unterkörper unabhängig. Allgemeiner kann man jeden Körper K als Vektorraum über einem Unterkörper $K_0 \subset K$ auffassen. Ist K endlich, so kann man für jedes $\alpha \in K$ das Minimalpolynom definieren.

Definition 10.6. Sei K ein endlicher Erweiterungskörper eines Körpers K_0. Ist $\alpha \in K$ und $g \in K_0[x] \setminus \{0\}$ ein normiertes Polynom mit $g(\alpha) = 0$ von kleinstmöglichem Grad, dann heißt g *Minimalpolynom* von α über K_0.

Lemma 10.7. *Sei K ein endlicher Erweiterungskörper eines Körpers K_0. Ist $g \in K_0[x] \setminus \{0\}$ ein Minimalpolynom von $\alpha \in K$ über K_0 und $f \in K_0[x]$ ein beliebiges Polynom mit $f(\alpha) = 0$, dann gilt $g|f$. Insbesondere ist das Minimalpolynom eindeutig bestimmt.*

Beweis. Division von f durch g liefert $f = sg + r$ mit grad $r <$ grad g. Setzt man $x = \alpha$, dann ergibt sich $r(\alpha) = 0$. Aufgrund der Gradminimalität eines Minimalpolynoms folgt $r = 0 \in K_0[x]$. □

Bevor wir Satz 10.4 komplementieren können, indem wir zeigen werden, dass zu jeder Primzahlpotenz ein endlicher Körper existiert, ist es zunächst erforderlich, Einsicht in die Primfaktorzerlegung in Polynomringen zu gewinnen.

11. Faktorielle Ringe

In diesem Abschnitt betrachten wir Ringe mit der Eigenschaft, dass sich ihre Elemente eindeutig in Primfaktoren zerlegen lassen. Wir unterscheiden zwischen zwei Eigenschaften.

Definition 11.1. Sei R ein Integritätsbereich.

i) Ein Element $p \in R \setminus \{0\}$ mit $p \notin R^*$ heißt *irreduzibel*, falls für alle $a, b \in R$ gilt
$$p = ab \quad \Longrightarrow \quad a \in R^* \text{ oder } b \in R^* .$$
ii) Ein Element $p \in R \setminus \{0\}$ mit $p \notin R^*$ heißt *prim*, falls für alle $a, b \in R$ gilt
$$p | ab \quad \Longrightarrow \quad p | a \text{ oder } p | b .$$

R heißt *faktoriell*, falls jedes von Null verschiedene Element, welches keine Einheit ist, Primelement oder Produkt von endlich vielen Primelementen ist.

Beispiel 11.2. Der Integritätsbereich $\mathbb{Z}[\sqrt{-5}] := \{x + y\sqrt{-5} \ : \ x, y \in \mathbb{Z}\}$ ist nicht faktoriell. Es gilt z.B.
$$6 \ = \ 2 \cdot 3 \ = \ (1 + \sqrt{-5})(1 - \sqrt{-5}) .$$
Die Faktoren sind irreduzibel, aber nicht prim. Zum Nachweis dieser Eigenschaften betrachten wir die 'Norm' $N(x + y\sqrt{-5}) := |x + y\sqrt{-5}|^2 = x^2 + 5y^2$ des Elements $x + y\sqrt{5}$. Sie ist multiplikativ, $N(ab) = N(a)N(b)$, und sie nimmt nur Werte in $\mathbb{N}_0$ an. Nun gilt $N(2) = 4$, $N(3) = 9$ und $N(1 + \sqrt{-5}) = N(1 - \sqrt{-5}) = 6$. Daher teilt $1 + \sqrt{-5}$ zwar das Produkt $2 \cdot 3$, weder aber 2 noch 3 (denn $N(1 + \sqrt{-5})$ teilt weder $N(2)$ noch $N(3)$). D.h., keiner der vier angegebenen Teiler von 6 ist prim. Andererseits prüft man leicht nach, dass die Faktoren sich nicht weiter zerlegen lassen (denn Elemente der Norm 2 oder 3 gibt es nicht), die Faktoren sind also irreduzibel.

Im Ring $\mathbb{Z}[\sqrt{-5}]$ gibt es im allgemeinen keine größten gemeinsamen Teiler: 6 und $2 + 2\sqrt{-5}$ haben die gemeinsamen Teiler 2 und $1 + \sqrt{-5}$, die ihrerseits teilerfremd sind.

Im allgemeinen ist es also nicht gleichbedeutend, ob ein Element irreduzibel oder prim ist.

Satz 11.3. *In einem Integritätsbereich R gilt*

i) *Jedes Primelelement ist irreduzibel.*
ii) *In faktoriellen Ringen ist jedes irreduzible Element prim.*
iii) *In faktoriellen Ringen ist eine Zerlegung eines Elements in Primfaktoren ist eindeutig, bis auf Einheiten und die Reihenfolge. Genauer: Hat $a \in R \setminus (\{0\} \cup R^*)$ die Primfaktorzerlegungen $a = p_1 \cdot \ldots \cdot p_r = q_1 \cdot \ldots \cdot q_r$, so folgt $r = s$, und nach einer geeigneten Permutation der q_i gilt $p_i = e_i q_i$ mit Einheiten e_i für $i \in \{1, \ldots, r\}$.*

Beweis. i) Ist $p = ab$ prim, dann gilt $p|a$ oder $p|b$. Im ersten Fall (der zweite ist analog) ergibt sich $a = pc$ und folglich $p = pbc$. Da man in Integritätsbereichen kürzen darf, folgt $1 = bc$ und damit $b \in R^*$.

ii) Sei a irreduzibel und $a = p_1 \cdot \dots \cdot p_r$ eine Primfaktordarstellung von a. Wegen der Irreduzibilität der p_i (nach i)) müssen die p_i bis auf einen einzigen Einheiten sein. Das ist nur im Fall $r = 1$ möglich, da ein Primelement per Definition keine Einheit ist.

iii) Sei $p_1 \cdot \ldots \cdot p_r = q_1 \cdot \ldots \cdot q_s$ mit Primelementen p_i, q_j. Nach Definition eines Primelements folgt, dass p_1 ein q_{j_1} teilt, also $p_1 = g_{j_1} e_1$. Aus der Irreduzibilität von p_1 folgt, dass e_1 eine Einheit ist. Wir können nun q_{j_1} aus der Gleichung kürzen und mit p_2 analog verfahren. Sukzessive ergibt sich $p_i = q_{j_i} e_i$ mit paarweise verschiedenen j_i und Einheiten e_i. Insbesondere folgt $r \leq s$. Aus Symmetriegründen gilt $r = s$. $\square$

In faktoriellen Ringen lässt sich daher auch der größte gemeinsame Teiler bilden. Einen ggT von a und b kann man aus den gemeinsamen Primfaktoren von a und b zusammensetzen. Eine alternative Charakterisierung faktorieller Ringe mittels irreduzibler Elemente liefert der folgende Satz.

Satz 11.4. *Ein Integritätsbereich R mit Einselement ist genau dann faktoriell, falls sich jedes von Null verschiedene Element, das keine Einheit ist, in (bis auf Reihenfolge und Einheiten) eindeutiger Weise als Produkt von endlichen vielen irreduziblen Elementen darstellen lässt.*

Beweis. "$\Longrightarrow$" klar nach Satz 11.3.

"$\Longleftarrow$" Es genügt zu zeigen, dass jedes irreduzible Element p dann prim ist.

Teilt p das Produkt $c := ab$ zweier Elemente $a, b \in R$, dann existiert ein $d \in R$ mit $c = pd$. Nach Voraussetzung lassen sich a, b und d in endlich viele irreduzible Faktoren zerlegen. Ebenfalls nach Voraussetzung sind die zwei auf diese Weise erhaltenen Zerlegungen von c gleich, insbesondere tritt p (bis auf eine Einheit) in der Zerlegung von ab auf. Daher ist p bis auf eine Einheit einer der irreduziblen Teiler von a oder von b. p teilt also a oder b und ist damit Primelement. $\square$

Satz 11.5. *Jeder Hauptidealring ist faktoriell.*

Beweis. *Zeige zunächst:* In einem Hauptidealring ist jedes irreduzible Element prim.

Sei p ein irreduzibles Element, das ab teilt. Es ist zu zeigen, dass p dann a oder b teilt. Definiere das Ideal

$$I := \{rp + sa \,:\, r, s \in R\}.$$

Da I nach Voraussetzung I ein Hauptideal ist, existiert ein $c \in R$ mit

$$I = \{tc \,:\, t \in R\}.$$

c teilt p und a, denn $p, a \in I$. Insbesondere gibt es ein $t \in R$ mit $p = tc$. Da p irreduzibel ist, ist entweder t oder c eine Einheit. Im ersten Fall ist $c = t^{-1}p$, p teilt also c und folglich

a. Im zweiten Fall folgt $1 \in I$, d.h. $1 = rp + sa$ für geeignete $r, s \in R$. In diesem Fall ist p ein Teiler von $b = rpb + s(ab)$.

Zeige nun: R ist faktoriell.

Sei $a \in R \setminus (\{0\} \cup R^*)$. a ist entweder irreduzibel (und damit prim) oder zerfällt in zwei Faktoren, die beide keine Einheiten sind. Wenn diese Faktoren nicht beide prim sind, können wir sie weiter zerlegen, usw. Diese Prozedur bricht nach endlich vielen Schritten mit einer Primfaktorzerlegung von a ab. Andernfalls würde man nämlich eine unendliche Folge $a = a_1, a_2, \ldots$ von Elementen finden, so dass $a_{j+1} \mid a_j$, aber $a_j \nmid a_{j+1}$ für alle j. Dies führt zu einem Widerspruch: Die Hauptideale (a_j) bilden dann eine strikt wachsende Folge (bzgl. Inklusion).

Zeige: $I_\infty := \bigcup_j (a_j)$ ist ein Ideal.

Gilt $b, c \in I_\infty$, so gibt es wegen der Monotonie der (a_j) ein i, so dass $b, c \in (a_i)$, und es folgt $b + c \in (a_i) \subset I_\infty$. Die anderen Forderungen an ein Ideal ergeben sich analog.

Nach Voraussetzung ist I_∞ ein Hauptideal: $I_\infty = (b)$ für ein $b \in R$. Es folgt $b \in (a_j)$ für ein j und damit $I_\infty \subset (a_j) \subset (a_{j+1}) \subset \ldots \subset I_\infty$. Die Folge der Ideale ist also doch nicht strikt wachsend, im Widerspruch zur obigen Aussage. $\square$

KOROLLAR 11.6. *Ist K ein Körper, dann ist $K[x]$ ein faktorieller Ring.*

12. Isomorphie endlicher Körper gleicher Mächtigkeit

Wir zeigen in diesem Abschnitt, dass endliche Körper gleicher Mächtigkeit isomorph sind.

LEMMA 12.1. *In jedem q-elementigen endlichen Körper K gilt*

$$\prod_{a \in K} (x - a) \;=\; x^q - x\,.$$

BEWEIS. Die multiplikative Gruppe $(K \setminus \{0\}, \cdot)$ hat genau $q - 1$ Elemente. Der Satz 6.4 von Euler lässt sich auch für beliebige endliche Gruppen in der Form

“Für jedes Element a in einer endlichen Gruppe G gilt $a^{|G|} = e$.“

formulieren, und für abelsche Gruppen ist der Beweis der gleiche. Deshalb erfüllen alle $a \in K \setminus \{0\}$ die Gleichung $a^{q-1} = 1$. Also ist jedes $a \in K$ (insbesondere auch $a = 0$) eine Nullstelle des Polynoms $x^q - x$. Dieses Polynom wird daher für jedes $a \in K$ vom Polynom $x - a$ geteilt. Da $K[x]$ ein faktorieller Ring ist, sind die irreduziblen Polynome $x - a$ nach Satz 11.3 prim. Aufgrund der eindeutigen Primfaktorzerlegung in $K[x]$ wird $x^q - x$ daher auch von dem Produkt der $x - a$ geteilt,

$$\prod_{a \in K} (x - a) \mid x^q - x\,.$$

Beide Polynome haben den Grad q, und der Koeffizient von x^q ist in beiden Fällen gleich 1. Deshalb sind die Polynome gleich. $\square$

Es stellt sich heraus, dass wir uns bei der Untersuchung von $K(\alpha)$ mit keinen speziellen Unterkörpern befasst haben, stattdessen ist jeder Unterkörper von K von dieser Gestalt. Dies folgt aus dem nächsten Resultat.

SATZ 12.2. *Ist K ein endlicher Körper, so ist K^* bezüglich der Multiplikation eine zyklische Gruppe, d.h. K^* besteht aus den Elementen $1, \alpha, \alpha^2, \dots, \alpha^{|K|-2}$ für ein geeignetes $\alpha \in K^*$. Man nennt ein solches α ein* primitives Element *von K.*

Der Beweis beruht auf dem folgenden gruppentheoretischen Sachverhalt.

LEMMA 12.3. *Sei G eine endliche abelsche Gruppe mit neutralem Element e, und sei m das Maximum der Ordnungen aller Gruppenelemente. Dann gilt $a^m = e$ für alle $a \in G$.*

BEWEIS. Sei $a \in G$, und sei $r := \mathrm{ord}(a)$ seine Ordnung, also die kleinste natürliche Zahl, so dass $a^r = e$. Sei $b \in G$ so, dass $\mathrm{ord}(b) = m$.

Es genügt zu zeigen: $r \mid m$.

Annahme: Es existiert eine Primzahlpotenz p^i mit $p^i \mid m$, $p^{i+1} \nmid m$, aber $p^{i+1} | r$. Sei $a' := a^{r/(p^{i+1})}$, $b' := b^{(p^i)}$. Dann gilt $\mathrm{ord}(a') = p^{i+1}$ und $\mathrm{ord}(b') = m/p^i$. Insbesondere sind $\mathrm{ord}(a')$ und $\mathrm{ord}(b')$ teilerfremd, und in einer endlichen abelschen Gruppe folgt hieraus allgemein $\mathrm{ord}(a'b') = \mathrm{ord}(a')\mathrm{ord}(b')$ (Beweis als Übung). Also gilt $\mathrm{ord}(a'b') = pm$, im Widerspruch zur Maximalität von m. $\square$

BEWEIS VON SATZ 12.2: Sei $q := |K|$ und m die maximale Ordnung m der Elemente aus K^*. Da $m \leq q-1$, genügt es zu zeigen, dass $m \geq q-1$.

Nach dem voranstehenden Lemma gilt $a^m = 1$ für alle $a \in K^*$. Das Polynom $x^m - 1 \in K[x]$ hat daher $q-1$ verschiedene Nullstellen. Folglich ist $m \geq q-1$. $\square$

LEMMA 12.4. *Sei K ein p^m-elementiger Körper, α ein primitives Element von K und g das Minimalpolynom von α über dem Primkörper $\mathbb{Z}_p$. Dann ist g ein über $\mathbb{Z}_p$ irreduzibles Polynom vom Grad m, und es gilt*

$$g \;\mid\; x^{p^m} - x\,.$$

BEWEIS. i) Als Minimalpolynom ist g irreduzibel.
ii) Sei $q := p^m$. Da α eine Nullstelle von $x^q - x$ ist, folgt nach Lemma 10.7, dass $g \mid x^q - x$.
iii) Sei $d := \mathrm{grad}(g)$.

Zu zeigen: $d = m$.

Da K ein Vektorraum über $\mathbb{Z}_p$ der Dimension m ist, gibt es $(c_0, \dots, c_m) \in \mathbb{Z}_p^{m+1} \setminus \{0\}$ mit $c_0 + c_1\alpha + c_2\alpha^2 + \dots + c_m\alpha^m = 0$. Deshalb ist α eine Nullstelle des Polynoms $c_m x^m + \dots + c_1 x + c_0 = 0$. Mit Lemma 10.7 folgt, dass $d := \mathrm{grad}(g) \leq m$. Andererseits ist

$$\{\lambda_{d-1}\alpha^{d-1} + \dots + \lambda_1\alpha + \lambda_0 \;:\; \lambda_0, \lambda_1, \dots, \lambda_{d-1} \in \mathbb{Z}_p\}$$

analog zum Beweis von Satz 10.5 ein p^d-elementiger Unterkörper von K, der das Element α enthält. Da α als primitives Element ganz K erzeugt, gilt $d = m$. $\square$

Eine wichtige Konsequenz der behandelten Strukturaussagen ist, dass sich zwei endliche Körper gleicher Mächtigkeit strukturell nicht unterscheiden.

DEFINITION 12.5. Zwei Körper K und K' heißen *isomorph*, wenn es eine bijektive Abbildung $\phi : K \to K'$ gibt, so dass für alle $x, y \in K$ gilt

$$\phi(x+y) \;=\; \phi(x) + \phi(y)\,, \quad \phi(xy) \;=\; \phi(x)\phi(y)\,.$$

BEMERKUNG 12.6. *Es gilt dann $\phi(0) = 0$, $\phi(1) = 1$ sowie für alle $x \in K^*$ die Eigenschaft $\phi(x)^{-1} = \phi(x^{-1})$.*

SATZ 12.7. *Zwei endliche Körper K und K' gleicher Mächtigkeit sind isomorph.*

BEWEIS. Gilt $|K| = |K'| = q := p^m$, dann gilt $\mathrm{char}(K) = \mathrm{char}(K') = p$. Sei α ein primitives Element von K. Für das Minimalpolynom $g \in \mathbb{Z}_p[x]$ von α über $\mathbb{Z}_p$ gilt nach dem voranstehenden Lemma grad $g = m$ und $g \mid x^q - x$. Da $x^q - x$ über K' vollständig in Linearfaktoren zerfällt, hat g auch in K' eine Nullstelle α' und ist wegen seiner Irreduzibilität das Minimalpolynom von α'. Wir erhalten die Bijektion

$$\begin{array}{rcl} K & \to & K' \\ \lambda_0 + \lambda_1\alpha + \ldots + \lambda_{m-1}\alpha^{m-1} & \mapsto & \lambda_0 + \lambda_1\alpha' + \ldots + \lambda_{m-1}(\alpha')^{m-1}\,, \quad \lambda_1, \ldots, \lambda_m \in \mathbb{Z}_p\,, \end{array}$$

und diese Abbildung ist ein Körperisomorphismus. $\square$

13. Konstruktion endlicher Körper

Wir zeigen nun, wie zu jeder Primzahlpotenz p^m ein endlicher Körper mit p^m Elementen konstruiert werden kann. Da ein endlicher Körper als Vektorraum über seinem Primkörper aufgefasst werden kann, können die Körperelemente als Tupel $(k_0, k_1, \ldots, k_{m-1})$ geschrieben werden, mit den Elementen k_i des Primkörpers. Wie in den vorherigen Abschnitten wird sich als vorteilhaft herausstellen, solche Tupel formal als Polynome $k_{m-1}x^{m-1} + \ldots + k_1x + k_0$ zu interpretieren.

Unter der Voraussetzung, dass ein irreduzibles Polynom f vom Grad m bekannt ist, lässt sich ein endlicher Körper mit p^m Elementen wie folgt konstruieren.

SATZ 13.1. *Sei K ein Körper und $f \in K[x] \setminus \{0\}$. Dann ist $K[x]/(f)$ genau dann ein Körper, wenn f irreduzibel ist. Ist $K = \mathbb{Z}_p$ und f irreduzibel vom Grad m, dann besitzt der Körper $\mathbb{Z}_p[x]/(f)$ genau p^m Elemente.*

BEWEIS. „$\Leftarrow$“ Zu zeigen: a) $(K[x]/(f)) \setminus \{0\}$ ist multiplikativ abgeschlossen, d.h. aus $\overline{g}, \overline{h} \neq 0$ in $K[x]/(f)$ folgt $\overline{g}\overline{h} \neq 0$ in $K[x]/(f)$.
b) Zu jedem $\overline{g} \in (K[x]/(f)) \setminus \{0\}$ existiert ein multiplikatives Inverses, d.h. ein $\overline{h} \neq 0$ mit $\overline{g}\overline{h} = 1$.

Alle anderen Körperaxiome sind offensichtlich erfüllt.

a) Seien $g, h \in K[x]$ mit $\overline{g}, \overline{h} = 0$ in $K[x]/(f)$. Dann existiert ein $a \in K[x]$ mit $gh = af$. Also teilt f das Produkt gh. Hieraus und aus der Irreduzibilität (und daher Primalität) von f folgt, dass f ein Teiler von g oder von h ist. Im ersten der beiden Fälle gilt $g = bf$ für ein $b \in K[x]$, also $\overline{g} = 0$ in $K[x]/(f)$, und analog im zweiten Fall $\overline{h} = 0$ in $K[x]/(f)$.

b) Sei $\overline{g} \in K[x]/(f)$. Aufgrund der Division mit Rest können wir annehmen, dass $\overline{g} = g + (f)$ mit grad $g <$ grad f. Da f irreduzibel ist, folgt $\mathrm{ggT}(f, g) = 1$. Nach dem Satz von Bézout existieren daher $r, s \in K[x]$ mit $rf + sg = 1$, d.h. $\overline{s} := s + (f)$ ist Inverses zu $\overline{g}$ in $K[x]/(f)$.

„$\Rightarrow$“ Sei $f = gh$ mit $g, h \notin K$. Dann sind $\overline{g} := g + (f)$, $\overline{h} := h + (f)$ zwei von Null verschiedene Elemente in $K[x]/(f)$. Es gilt jedoch

$$\overline{g}\overline{h} \;=\; (g + (f))(h + (f)) = gh + (f) = (f)$$

d.h. $\overline{g}$ und $\overline{h}$ sind Nullteiler in $K[x]/(f)$. $K[x]/(f)$ ist also kein Körper .

Die Anzahl der Elemente ergibt sich unmittelbar aus der Tatsache, dass $\mathbb{Z}_p$ der Primkörper von $\mathbb{Z}_p[x]$ ist. $\square$

Beispiel 13.2. Das Polynom $f := x^2 + x + 1 \in \mathbb{Z}_2[x]$ ist irreduzibel. Folglich bilden die Polynome 0, 1, x, $x+1$ eine 4-elementigen Körper, wenn jeweils modulo f gerechnet wird. Die zugehörigen Additions- und Multiplikationstabellen lauten also.

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

$\cdot$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

Die im voranstehenden Satz beschriebene Möglichkeit zur Konstruktion endlicher Körper ist diejenige, welche bei der späteren Betrachtung fehlerkorrigierender Codes verwendet wird. Sie beruht jedoch auf der Existenz irreduzibler Polynome vom Grad m in $\mathbb{Z}_p[x]$. Eine Möglichkeit zur Klärung dieser Frage ist, mittels Erzeugendenfunktionen und Möbius-Inversion die genaue Anzahl der irreduziblen Polynome N_m vom Grad m zu berechnen; eine solches Abzählen zeigt, dass $N_m \geq 1$ (siehe z.B. Cox, Little, O'Shea: Using Algebraic Geometry, §9.1 (algebraic coding theory; finite fields)).

Die nachfolgende strukturelle Kärung der Existenz endlicher Körper mit p^m Elementen liefert weitere Einsicht in den Aufbau von Körpern, die uns im Abschnitt über zyklische Codes hilfreich sein wird. Grundlegende Idee ist es, Körper als Nullstellenmenge eines Polynoms zu konstruieren. Im Falle eines irreduziblen Polynoms erweitern wir den Körper geeignet (analog zu dem über $\mathbb{R}$ irreduziblen Polynom x^2+1, das durch die "Adjunktion" des Elements i in $(x+i)(x-i)$ zerfällt).

LEMMA 13.3. *Ist K_0 ein Körper und $g \in K_0[x]$ irreduzibel, dann gibt es einen Erweiterungskörper K von K_0, in dem g eine Nullstelle besitzt.*

BEWEIS. Sei $g \in K_0[x]$ mit $g(x) = x^n + b_{n-1}x^{n-1} + \ldots + b_0$ ein (o.B.d.A.) normiertes, irreduzibles Polynom vom Grad n. Folglich besitzt g keine Nullstellen in K_0. Wir wählen nun K als Restklassenring von K_0 modulo dem von g erzeugten Hauptideal, $K := K_0/(g)$. Nach Satz 13.1 ist K ein Körper. Ferner ist das Element $x + (g)$ eine Nullstelle von g in K, da $g \equiv 0 \,(\mathrm{mod}\ (g))$. □

LEMMA 13.4. *In einem Körper der Primzahlcharakteristik p gilt für alle Elemente a, b*

$$(ab)^p \;=\; a^p b^p\,, \quad (a \pm b)^p = a^p \pm b^p\,.$$

BEWEIS. Die erste Aussage ist offensichtlich, die zweite ergibt sich aus der Binomialformel

$$(a \pm b)^p \;= \sum_{i=0}^{p} \binom{p}{i} a^i\,(\pm b)^{p-i}\,,$$

denn $\binom{p}{i}$ ist für $i \notin \{0, p\}$ ein Vielfaches von p und die entsprechenden Summanden verschwinden. □

SATZ 13.5. *Sei p prim und $m \in \mathbb{N}$. Dann gibt es einen Körper mit $q = p^m$ Elementen.*

BEWEIS. Wir konstruieren K als Körper, in dem das Polynom $f(x) = x^q - x$ vollständig in Linearfaktoren zerfällt. Dazu zerlegen wir $f \in \mathbb{Z}_p[x]$ in irreduzible Faktoren $g_1, \ldots, g_r \in \mathbb{Z}_p[x]$. Nach Lemma 13.3 gibt es einen Erweiterungskörper K_1, in dem g_1 und damit f eine Nullstelle hat. Daher besitzt f in K_1 einen Linearfaktor, möglicherweise mehrere, die wir von f abspalten. (Ist g_1 bereits linear, so erübrigt sich dieser Schritt.) Den Rest von f zerlegen wir erneut in irreduzible Faktoren und konstruieren einen Körper $K_2 \supset K_1$, in dem f weitere lineare Faktoren abspaltet. Nach maximal q Erweiterungen zerfällt f schließlich in einem Körper K' vollständig in Linearfaktoren.

Wir definieren nun K als die Menge aller Nullstellen von f in K', also derjenigen $a \in K'$ mit $a^q = a$. Wegen $q \cdot 1 = (p^n) \cdot 1 = 0$ in K' gilt für die formale Ableitung $f' = (q \cdot 1) \cdot x^{q-1} - 1 = -1$, daher sind alle Nullstellen voneinander verschieden, so dass K insgesamt q Elemente enthält. K enthält 0 und 1, und für $a, b \in K$ gilt

$$(ab)^q \;=\; a^q b^q \;=\; ab\,,$$

daher folgt $ab \in K$ und wegen $(a^{-1})^q = (a^q)^{-1} = a^{-1}$ auch $a^{-1} \in K$. Weiter folgt aus dem voranstehenden Lemma

$$(a \pm b)^{p^m} = ((a \pm b)^p)^{p^{m-1}} = (a^p \pm b^p)^{p^{m-1}} = \cdots = a^{p^m} \pm b^{p^m} = a \pm b\,,$$

d.h. $a \pm b \in K$. Damit ist K ein Körper mit q Elementen. $\square$

Der konstruierte Körper heißt *Zerfällungskörper* von $x^q - x$. Allgemeiner bezeichnet man als Zerfällungskörper eines nicht-konstanten Polynoms $f \in K_0[x]$ jeden minimalen Körper $K \supset K_0$, so dass $f \in K[x]$ vollständig in Linearfaktoren zerfällt. In der Algebra wird gezeigt, dass Zerfällungskörper immer existieren und bis auf Isomorphie eindeutig sind.

Wir fassen unsere Resultate in dem folgenden Satz zusammen:

SATZ 13.6. *Es gibt genau dann einen endlichen Körper K mit q Elementen, wenn q Potenz einer Primzahl ist. K ist dann bis auf Isomorphie eindeutig bestimmt.*

SATZ 13.7. *Sei p prim und $q = p^m$ für ein $m \in \mathbb{N}$. Dann ist das Polynom $x^q - x \in \mathbb{Z}_p[x]$ gleich dem Produkt aller normierten, irreduziblen Polynome in $\mathbb{Z}_p[x]$, deren Grad m teilt.*

BEWEIS. $x^q - x$ hat keine Quadrate von Polynomen als Faktor, da die formale Ableitung gleich -1 ist (siehe Übungen).

„$\supset$“: Sei g ein irreduzibles Polynom vom Grad n, so dass $n|m$. Nach Satz 13.5 besitzt g eine Nullstelle α in einem Körper mit p^n Elementen. Es folgt $\alpha^{p^n} = \alpha$, und, da m Vielfaches von n ist, $\alpha^{p^m} = (\ldots(\alpha^{p^n})^{p^n}\ldots)^{p^n} = \alpha$. α ist also auch eine Nullstelle von $x^q - x$. Ferner ist g das Minimalpolynom von α, denn die Existenz eines Minimalpolynoms h von α kleineren Grades würde (nach Lemma 10.7) $h \mid g$ und damit die Reduzibilität von g nach sich ziehen. Ebenfalls mit Lemma 10.7 folgt, dass $g \mid x^q - x$.

„$\subset$“: Sei g ein normierter, irreduzibler Teiler von $x^q - x$ vom Grad n.

Zu zeigen: $n \mid m$.

g hat eine Nullstelle $\alpha \in \mathbb{F}_q$. Dann ist g (analog zum ersten Beweisteil) das Minimalpolynom von α. Für den Unterkörper $K(\alpha)$ gilt (wie im Beweis von Satz 10.5)

$$K(\alpha) = \{a_0 + a_1\alpha + \ldots + a_{n-1}\alpha^{n-1} \,:\, a_0, \ldots, a_{n-1} \in \mathbb{Z}_p\},$$

und er besitzt daher p^n Elemente. Wir fassen nun $\mathbb{F}_q$ als Vektorraum über $K(\alpha)$ auf, mit der Basis $\beta_1, \ldots, \beta_d$. Dann hat jedes Element aus $\mathbb{F}_q$ eine eindeutige Darstellung

$$b_1\beta_1 + \ldots + b_d\beta_d \quad \text{mit } b_1, \ldots, b_d \in K(\alpha).$$

$\mathbb{F}_q$ hat also genau $(p^n)^d = p^{nd}$ Elemente, und es folgt $m = nd$. $\square$

BEISPIEL. Sei m prim, $q := p^m$. Dann hat $x^q - x$ nur irreduzible Teiler vom Grad 1 oder m. Nach Lemma 12.1 gilt für das Produkt der normierten Teiler vom Grad 1 in $\mathbb{Z}_p[x]$ die Gleichung $\prod_{a\in\mathbb{Z}_p}(x-a) \;=\; x^p - x$. Bezeichnen $g_1, \ldots, g_r \in \mathbb{Z}_p[x]$ die normierten, irreduziblen Teiler vom Grad m, so gilt also

$$x^{p^m} \;=\; (x^p - x)g_1(x) \cdot \ldots \cdot g_r(x).$$

Es folgt $p^m = p + mr$. Die Anzahl der normierten, irreduziblen Polynome vom Grad m ist für primes p und primes m daher gegeben durch

$$r \;=\; \frac{p^m - p}{m}\,.$$

Für $p = 7, m = 5$ ist $r = 3360$. Für ein allgemeines (nicht notwendig primes) m gibt es in $\mathbb{Z}_p[x]$ genau $\frac{1}{m}\sum_{d|m}\mu(d)p^{m/d}$ normierte, irreduzible Polynome vom Grad m, wobei μ die Möbiussche μ-Funktion $\mu : \mathbb{N} \to \{-1, 0, 1\}$ bezeichnet,

$$\mu(n) = \mu\left(\prod_{i=1}^{r} p_i^{e_i}\right) := \begin{cases} (-1)^r & \text{falls } e_1 = \ldots = e_r = 1\,. \\ 0 & \text{sonst}\,. \end{cases}$$

(Vgl. z.B. L. Childs: A concrete introduction to higher algebra, Springer (1979); W. Heise, P. Quattrocchi: Informations- und Codierungstheorie, Springer (1983)).

14. Fehlerkorrigierende Codes

Die Kommunikation von Informationen erfolgt oft über „gestörte“ Kanäle, welche Fehler in der übertragenen Nachricht bewirken können. Dies ist beispielsweise bei Satellitenübertragungen oder bei der Speicherung von Daten (z.B. auf Magnetbändern oder Compact Discs) der Fall. Daher soll Information möglichst so codiert werden, dass Fehler erkannt und/oder korrigiert werden können.

Ein *Alphabet* A ist eine Menge der Kardinalität mindestens 2. $A^* := \cup_{n\geq 0} A^n$ bezeichnet die Menge der Wörter über dem Alphabet A.

Definition 14.1. Sei A ein Alphabet. Ein *Code* über A ist eine nichtleere Teilmenge von A^*. Falls alle Codewörter die gleiche Länge n haben, spricht man von einem *Blockcode* der Länge n. Ein *binärer Code* ist ein Code über dem Alphabet $A = \{0, 1\}$.

Wir betrachten nur Blockcodes über endlichen Körpern, z.B. $\mathbb{F} = \mathbb{Z}_p$ für p prim oder $\mathbb{F} = \mathbb{Z}_p[x]/(g)$ mit p prim und $g \in \mathbb{Z}_p[x]$ irreduzibel.

Definition 14.2. Der *Hamming-Abstand* (kurz *Abstand*) zweier Vektoren $x, y \in \mathbb{F}^n$ ist die Anzahl der Stellen, in denen sich x und y unterscheiden:

$$d((x_1, \ldots, x_n), (y_1, \ldots, y_n)) := |\{i \,:\, x_i \neq y_i\}|\,.$$

Das *Gewicht* $w(x)$ ist definiert als die Anzahl der von 0 verschiedenen Stellen von x:

$$w((x_1, \ldots, x_n)) := |\{i \,:\, x_i \neq 0\}|\,.$$

Für $\mathbb{F} = \{0, 1\}$ ist $d(x, y) = w(x + y)$.

Definition 14.3. Der *Minimalabstand* eines Codes C ist definiert als

$$d(C) := \min\{d(x, y) \,:\, x, y \in C,\, x \neq y\}\,.$$

Bei der Maximum-Likelihood-Decodierung decodiert man das empfangene Tupel $v = (v_1, \dots, v_n)$ als ein Codewort, dessen Abstand zu v minimal ist.

Definition 14.4. (*t-Fehler-erkennender und -korrigierender Code.*) Ein Code heißt *t-Fehler-korrigierend*, falls Fehler an bis zu t Stellen eines Codeworts immer korrekt decodiert werden können. Ein Code heißt *t-Fehler-erkennend*, falls bei Fehlern an bis zu t Stellen eines Codeworts die *Anzahl* der fehlerhaften Stellen immer korrekt erkannt wird.

Es folgt unmittelbar:

Satz 14.5. *Ein Code mit Minimalabstand d ist $\lfloor \frac{d-1}{2} \rfloor$-Fehler-korrigierend. Ist d gerade, dann ist der Code $\frac{d}{2}$-Fehler-erkennend.*

Wir betrachten hier fast ausschließlich lineare Codes. Diese Codes lassen sich besonders gut untersuchen und anwenden, da hierfür Methoden aus der linearen Algebra benutzt werden können.

Definition 14.6. Ein Code $C \subset \mathbb{F}^n$ heißt *linearer Code*, wenn er ein Unterraum von $\mathbb{F}^n$ ist. Hat C die Dimension k, dann spricht man von einem *$[n,k]$-Code*. Ein *$[n,k,d]$-Code* ist ein $[n,k]$-Code mit Minimalabstand $d := d(C)$.

Lemma 14.7. *Der Minimalabstand d eines linearen Codes C ist*

$$d \;=\; \min\{w(x) \,:\, x \in C,\, x \neq 0\}\,.$$

Beweis. Seien $y \neq z \in C$ mit $d = d(y,z)$. Für $y - z \in C$ folgt

$$w(x) = d(y-z, 0) = d(y,z) = d\,.$$

□

Definition 14.8. Sei C ein linearer $[n,k]$-Code und $g_1, \dots, g_k$ eine Basis des Vektorraums C. Dann heißt die $k \times n$-Matrix $G := (g_{ij})_{1 \le i \le k, 1 \le j \le n}$ eine *Generatormatrix* von C.

Wir verwenden das formale innere Produkt auf $\mathbb{F}^n$, das durch die bilineare Abbildung

$$\begin{array}{rcl} \langle \cdot, \cdot \rangle \,:\, \mathbb{F}^n \times \mathbb{F}^n & \to & \mathbb{F}\,, \\ (v,w) & \mapsto & \sum_{i=1}^n v_i w_i\,. \end{array}$$

gegeben ist. (Im Gegensatz zu einem Skalarprodukt gibt es hier nicht den Begriff der positiven Semidefinitheit.)

Definition 14.9. Sei $C \subset \mathbb{F}^n$ ein $[n,k]$-Code. Der *duale* Code $C^\perp$ zu C ist definiert als

$$C^\perp \;:=\; \{u \in \mathbb{F}^n \,:\, \langle u, c \rangle = 0 \quad \forall c \in C\}\,.$$

$C^\perp$ ist ein $[n, n-k]$-Code, und es gilt $(C^\perp)^\perp = C$.

DEFINITION 14.10. Ein Code $C \subset \mathbb{F}^n$ heißt *systematisch in den Stellen* $i_1, \dots, i_k$, wenn zu jedem Vektor $u = (u_1, \dots, u_k) \in \mathbb{F}^k$ *genau* ein Codewort $c = (c_1, \dots, c_n)$ mit $c_{i_1} = u_1, \dots, c_{i_k} = u_k$ existiert.

Ist ein Code systematisch in den Stellen $i_1, \dots, i_k$, so kann in diesen Stellen eine Ausgangsnachricht der Länge k untergebracht werden. Die verbleibenden Stellen dienen zur Redundanz, mit deren Hilfe Fehler erkannt und korrigiert werden können.

SATZ 14.11. *Sei C ein linearer $[n,k]$-Code, der in den ersten k Stellen systematisch ist. Dann hat C eine* kanonische Generatormatrix*, d.h. eine (sogar eindeutig bestimmte) Generatormatrix der Form $G = (I_k|A)$, wobei I_k die $k \times k$-Einheitsmatrix ist und $A \in \mathbb{F}^{k\times(n-k)}$.*

BEWEIS. Für alle $i \in \{1, \dots, k\}$ existiert nach Voraussetzung genau ein Codewort g_i mit $(g_{i1}, \dots, g_{ik}) = e^{(i)}$ (i-ter Einheitsvektor). Sei G die Generatormatrix mit den Zeilen $g_1, \dots, g_k$. □

BEISPIEL. Sei C der lineare Code $\{(000), (011), (101), (110)\} \subset \{0,1\}^3$. Mögliche Generatormatrizen sind:

$$G_1 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \qquad G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \qquad G_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

G_2 ist die kanonische Generatormatrix. Mit der kanonischen Generatormatrix eines linearen $[n,k]$-Codes $C \subset \mathbb{F}^n$ kann eine k-stellige Nachricht $u \in \mathbb{F}^k$ mittels

$$c \; := \; u \cdot G \; = (u_1, \dots, u_k, c_{k+1}, \dots, c_n)$$

codiert werden.

Bis jetzt haben wir uns mit dem Erzeugen von linearen Codes beschäftigt. Als nächstes untersuchen wir, wie man erkennt, ob ein gegebener Vektor zum Code gehört.

DEFINITION 14.12. Sei $C \subset \mathbb{F}^n$ ein linearer Code. Eine Matrix $H \in \mathbb{F}^{l\times n}$ (mit $l \in \mathbb{N}$) heißt *Kontrollmatrix*, falls

$$C = \{v \in \mathbb{F}^n \; : \; H \cdot v^T = 0\}.$$

SATZ 14.13. *Sei $C \subset \mathbb{F}^n$ ein $[n,k]$-Code mit Generatormatrix G. Eine Matrix $H \in \mathbb{F}^{l\times n}$ ist genau dann eine Kontrollmatrix von C, wenn gilt:*

$$H \cdot G^T = 0 \quad \textit{und} \quad \operatorname{rang}(H) = n - k.$$

BEWEIS. „$\Rightarrow$“: Da H eine Kontrollmatrix ist, ist C der Lösungsraum des linearen Gleichungssystems $H \cdot v^T = 0$. Es folgt $H \cdot G^T = 0$ und $\operatorname{rang}(H) = n - k$. (Insbesondere ist H also eine Generatormatrix von $C^\perp$.)

„$\Leftarrow$“: Sei $H \cdot G^T = 0$ und $\operatorname{rang}(H) = n - k$.

Zeige: $C = \{v \in \mathbb{F}^n \; : \; H \cdot v^T = 0\}$.

„$\subset$“: gilt, da jedes Codewort $c \in C$ eine Linearkombination der Zeilenvektoren von G ist. Da die beiden Unterräume von $\mathbb{F}^n$ die gleiche Dimension k haben, folgt die Gleichheit. □

Mit Hilfe der kanonischen Generatormatrix lässt sich eine Kontrollmatrix besonders leicht finden.

SATZ 14.14. *Der lineare* $[n, k]$*-Code habe die kanonische Generatormatrix* $G = (I_k|A)$. *Dann ist die Matrix* $H = (-A^T|I_{n-k})$ *eine Kontrollmatrix von* C, *die sogenannte* kanonische Kontrollmatrix.

BEWEIS. Die $n - k$ Zeilen von H sind linear unabhängig (wegen I_{n-k}), daher gilt $\mathrm{rang}(H) = n - k$. Die Eigenschaft $H \cdot G^T = 0$ folgt aus

$$\underbrace{(-A^T|I_{n-k})}_{\in \mathbb{F}^{(n-k)\times n}} \cdot \underbrace{(I_k|A)^T}_{\in \mathbb{F}^{n \times k}} \;=\; -A^T + A^T \;=\; 0\,.$$

□

BEISPIEL. Sei $n = 3$, $C = \{(000), (011), (110), (101)\}$ und $k = 2$:

$$G = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Die kanonische Kontrollmatrix ist $H = (-A^T|I_1) = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$.

15. Hamming-Codes

Ziel: Konstruktion fehlerkorrigierender linearer Codes.

SATZ 15.1. *Sei* $C \subset \mathbb{F}^n$ *ein linearer Code mit Kontrollmatrix* H. *Dann sind folgende Aussagen äquivalent:*

(1) $d(C) \geq d$ *;*
(2) *Je* $d - 1$ *der Spalten von* H *sind linear unabhängig.*

BEWEIS. Sei $H = (u_1, \ldots, u_n)$.
„$\Rightarrow$“: *Annahme:* Es existiert ein $d' < d$, $i_1, \ldots, i_{d'} \in \{1, \ldots, n\}$ und $c' = (c'_{i_1}, \ldots, c'_{i'_d}) \neq 0$, so dass $\sum_{j=1}^{d'} c'_{i_j} u_{i_j} = 0$. Der durch Nullen ergänzte Vektor c' liefert ein Codewort c mit Gewicht $\leq d' < d$, im Widerspruch zur Voraussetzung.
„$\Leftarrow$“: *Annahme:* Es existiert ein Codewort $c = (c_1, \ldots, c_n)$ mit $w(c) = d' < d$. Wegen $\sum_{i=1}^{n} u_i c_i = 0$ sind die Vektoren u_i mit $c_i \neq 0$ linear abhängig, im Widerspruch zur Voraussetzung. □

Für die nachfolgend betrachteten Hamming-Codes konzentrieren wir uns auf $\mathbb{F} = \mathbb{F}_2$.

DEFINITION 15.2. Ein linearer Code mit einer Kontrollmatrix H, die jeden der $2^r - 1$ Vektoren aus $\mathbb{F}_2^r \setminus \{0\}$ genau einmal als Spalte enthält (für ein $r \geq 2$) heißt *(binärer)* $[2^r - 1, 2^r - 1 - r]$*-Hamming-Code.*

Beispiel. Für $r = 3$ erhalten wir einen $[7,4]$-Code. H lautet beispielsweise

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Obige Definition wird durch folgenden Satz gerechtfertigt.

Satz 15.3. *Ein $[2^r-1, 2^r-1-r]$-Hamming-Code ist ein linearer $[2^r-1, 2^r-1-r]$-Code, und er ist 1-Fehler-korrigierend.*

Beweis. Der Hamming-Code ist natürlich ein $[n,k]$-linearer Code für ein Paar (n,k). Auf der Definition ergibt sich unmittelbar $n = 2^r - 1$, und da die Kontrollmatrix H (nach eventueller Permutation der Spalten) eine Untermatrix $I_{r,r}$ enthält, hat sie Rang r, d.h. $k = n - r = 2^r - 1 - r$.

Je zwei Spalten von H sind verschieden und damit wegen $\mathbb{F} = \mathbb{F}_2$ linear unabhängig. Mit Satz 15.1 folgt für die Distanz $d(C) = 3$, der Code ist also 1-Fehler-korrigierend. □

Die nachfolgenden Überlegungen zeigen, dass Hamming-Codes optimal sind, wenn $|C|$ bei gegebenem $d(C)$ maximiert werden soll.

Satz 15.4. (Hamming-Schranke.) *Sei $M(n,d,q) := \max\{|C| \,:\, C \subset \mathbb{F}_q^n, d(C) \geq d\}$. Für ungerades $d = 2t+1$ gilt*

$$M(n,d,q) \;\leq\; \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i}.$$

Beweis. Sei $C \subset \mathbb{F}_q^n$ ein Code mit $d(C) \geq 2t+1$, und sei

$$B_t(a) \;:=\; \{b \in \mathbb{F}_q^n \,:\, d(a,b) \leq t\}$$

die „Kugel" um ein Codewort a mit Hamming-Radius t. Wegen $d(C) \geq 2t+1$ sind die Kugeln zu verschiedenen Codewörtern disjunkt, und es folgt $q^n \geq |C| \cdot |B_t(a)|$. Da es $\binom{n}{i}(q-1)^i$ Wörter in $B_t(a)$ mit Hamming-Abstand i zu a gibt, gilt

$$|B_t(a)| \;=\; \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$$

und daher

$$M(n,d,q) \;\leq\; \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i}.$$

□

Definition 15.5. Ein Code $C \subset \mathbb{F}_q^n$ heißt t-perfekt, wenn $d(C) \geq 2t+1$ und

$$|C| = \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i}(q-1)^i}.$$

Satz 15.6. *Der binäre* $[2^r-1, 2^r-1-r]$*-Hamming-Code ist* 1*-perfekt.*

Beweis. Siehe Übungen. □

Benutzt man wie in obigem Beispiel für einen Hamming-Code eine Kontrollmatrix, in der in der i-ten Spalte die Binärdarstellung von i steht, dann lässt sich die Decodierung eines Vektors v besonders einfach ausführen. Zuerst wird das Syndrom $s := Hv^T \in \mathbb{F}_2^r$ berechnet. Ist s der Nullvektor, dann ist v ein Codewort. Anderenfalls suchen wir ein Codewort c, das sich von v nur in einem einzigen Bit unterscheidet. Ist s die Binärdarstellung einer Zahl i, dann wird v als $c := v + e^{(i)}$ decodiert, wobei $e^{(i)}$ der i-te Einheitsvektor von $\mathbb{F}^{2^r-1}$ ist. Denn dann ändern sich beim Übergang von v zu c genau diejenigen Bits im Syndrom, an denen eine 1 in der Binärdarstellung von i steht, d.h. $Hc^T = 0$. Im obigen Beispiel ergibt sich für $v = (1\,1\,0\,0\,0\,1\,0)$ das Syndrom $s^T = (101)$, also $i = 5$ und $c = (1\,1\,0\,0\,\underline{1}\,1\,0)$.

16. Zyklische Codes

Sei $\mathbb{F}_q$ ein endlicher Körper mit q Elementen.

Definition 16.1. Ein Code $C \subset \mathbb{F}_q^n$ heißt *zyklisch*, falls er gegen zyklisches Rotieren abgeschlossen ist, d.h., falls aus $(c_0, \ldots, c_{n-2}, c_{n-1})$ immer $(c_{n-1}, c_0, \ldots, c_{n-2})$ folgt.

Zur Erinnerung: Vektoren der Länge n können mittels des Vektorraum-Isomorphismus

$$\begin{array}{rcl} \psi : \mathbb{F}_q^n & \to & \mathbb{F}_q[x]/(x^n-1) \\ (c_0, c_1, \ldots, c_{n-1}) & \mapsto & c_0 + c_1x + c_2x^2 + \ldots + c_{n-1}x^{n-1} \quad + \quad (x^n-1) \end{array}$$

als Polynome vom Grad $< n$ aufgefasst werden, wobei (x^n-1) das von x^n-1 erzeugte Hauptideal bezeichnet. Das Multiplizieren mit x bewirkt das zyklische Vertauschen der Koeffizienten von c. Von nun an betrachten wir lineare Codes daher als Teilmenge des Restklassenrings $\mathbb{F}_q[x]/(x^n-1)$.

Satz 16.2. *Ein linearer Code* $C \subset \mathbb{F}_q[x]/(x^n-1)$ *ist genau dann zyklisch, wenn* C *ein Ideal von* $\mathbb{F}_q[x]/(x^n-1)$ *ist.*

Beweis. „$\Rightarrow$“: Als Untervektorraum von $\mathbb{F}_q[x]/(x^n-1)$ ist C insbesondere eine Untergruppe (bzgl. $+$). Sei nun $c(x) \in C$, $r(x) = r_0 + r_1x + \ldots + r_{n-1}x^{n-1} \in \mathbb{F}_q[x]/(x^n-1)$. Da C zyklisch ist, gilt $xc(x), x^2c(x), \ldots, x^{n-1}c(x) \in C$, und wegen der Linearität dann

$$r(x)c(x) = r_0c(x) + r_1xc(x) + \ldots + r_{n-1}x^{n-1}c(x) \in C\,.$$

Also ist C ein Ideal.

„$\Leftarrow$“: Als Ideal ist C ein Untervektorraum von $\mathbb{F}_q[x]/(x^n-1)$, d.h. ein linearer Code. Für $r(x) = x$ und $c(x) \in C$ gilt aufgrund der Idealeigenschaft $xc(x) \in C$, d.h. C ist zyklisch. □

Da $\mathbb{F}_q[x]/(x^n-1)$ nach Korollar 9.4 ein Hauptidealring ist, wird jedes Ideal C von einem Polynom $g(x) \in C \setminus \{0\}$ minimalen Grades erzeugt. Ein solches Polynom heißt *Generatorpolynom* von C. Die Elemente von C sind gerade die Vielfachen von $g(x)$. Dies kann noch präziser formuliert werden:

SATZ 16.3. *Sei C ein zyklischer linearer Code und $g(x)$ ein Generatorpolynom von C. Sei $s := \operatorname{grad} g(x)$, $k := n - s$. Dann ist C ein $[n,k]$-Code, und es gilt*

$$C = \{a(x)g(x) \,:\, a(x) \in \mathbb{F}_q[x]/(x^n-1), \operatorname{grad} a(x) < k\}\,.$$

BEWEIS. „$\supset$": klar.

„$\subset$": Sei $c(x) \in C$. Dann gibt es Polynome $a(x), r(x) \in \mathbb{F}_q[x]/(x^n-1)$ mit

$$c(x) \;=\; a(x)g(x) + r(x)\,, \quad \operatorname{grad} r(x) < \operatorname{grad} g(x)\,.$$

Es folgt $r(x) \in C$, und da $g(x) \in C \setminus \{0\}$ minimalen Grad hat, folgt $r(x) = 0$. Ferner gilt $\operatorname{grad} a(x) = \operatorname{grad} c(x) - \operatorname{grad} g(x) < n - s = k$.

Noch zu zeigen: $|C| = |\mathbb{F}|^k$.

Da es genau $|\mathbb{F}|^k$ viele Polynome vom Grad $< k$ gibt, genügt es zu zeigen, dass die Abbildung $\mathbb{F}_q[x]/(x^n-1) \to \mathbb{F}_q[x]/(x^n-1)$, $a(x) \mapsto a(x)g(x)$ injektiv ist.

Annahme: Es existieren $a(x)$ und $a'(x)$ vom Grad $< k$ mit $a(x)g(x) = a'(x)g(x)$.

Für $b(x) := a(x) - a'(x)$ gilt dann $b(x)g(x) = 0$, und zu zeigen ist $b(x) = 0$. Sei $b(x) = \sum_{i=0}^{k-1} b_i x^i$, $g(x) = \sum_{i=0}^{s} g_i x^i$. Dann folgt

$$0 \;=\; b(x)g(x) \;=\; b_0g_0 + (b_0g_1 + b_1g_0)x + (b_0g_2 + b_1g_1 + b_2g_0)x^2 + \ldots + b_{k-1}g_s x^{n-1}\,.$$

Durch Koeffizientenvergleich ergibt sich das lineare Gleichungssystem

$$(b_0, b_1, \ldots, b_{k-1}) \begin{pmatrix} g_0 & g_1 & \cdots & \cdots & g_s & & 0 \\ & g_0 & g_1 & \cdots & \cdots & g_s & \\ & & \ddots & \ddots & & & \ddots \\ 0 & & & g_0 & g_1 & \cdots & \cdots & g_s \end{pmatrix} = (0, \ldots, 0)\,.$$

Die Koeffizientenmatrix hat wegen $g_s \neq 0$ den Rang k. Es folgt $b(x) = 0$. $\square$

Die angegebene Koeffizientenmatrix ist offensichtlich eine Generatormatrix von C.

KOROLLAR 16.4. *Ein zyklischer linearer Code $C \subset \mathbb{F}_q^n/(x^n-1)$ mit Generatorpolynom $g(x) = g_0 + g_1x + \cdots + g_sx^s$ hat die folgende $(n-s) \times n$-Matrix als (nichtkanonische) Generatormatrix:*

$$\begin{pmatrix} g_0 & g_1 & \cdots & \cdots & g_s & & 0 \\ & g_0 & g_1 & \cdots & \cdots & g_s & \\ & & \ddots & \ddots & & & \ddots \\ 0 & & & g_0 & g_1 & \cdots & \cdots & g_s \end{pmatrix}.$$

In Satz 16.3 kann man $a(x)$ als Nachricht betrachten, die zum Codewort $c(x) = a(x)g(x)$ verschlüsselt wird. Das Multiplizieren der Polynome kann mittels sogenannter Schieberegister sehr effizient in Schaltungen implementiert werden.

Für zyklische Codes gibt es nicht nur Generator-, sondern auch Kontrollpolynome. Durch Division mit Rest folgt unmittelbar, dass jedes Generatorpolynom $g(x)$ (welches minimalen Grad in $C \setminus \{0\}$ hat) das Polynom $x^n - 1$ teilt.

SATZ 16.5. *Sei $C \subset \mathbb{F}_q[x]/(x^n - 1)$ ein zyklischer linearer Code mit Generatorpolynom $g(x)$. Für das sog.* Kontrollpolynom *$h(x) := (x^n - 1)/g(x)$ von C und für alle $v(x) \in \mathbb{F}_q^n/(x^n - 1)$ gilt*

$$v(x) \in C \iff h(x)v(x) = 0 \quad (\textit{in } \mathbb{F}_q[x]/(x^n - 1))$$

BEWEIS. „$\Rightarrow$“ Für jedes $c(x) = a(x)g(x) \in C$ gilt

$$h(x)c(x) = h(x)a(x)g(x) = a(x)(x^n - 1) = 0\,.$$

„$\Leftarrow$“ Sei $h(x)v(x) = 0$ in $\mathbb{F}_q[x]/(x^n - 1)$. Dies bedeutet in $\mathbb{F}_q[x]$, dass es ein $a(x)$ gibt mit $h(x)v(x) = a(x)(x^n - 1)$. Wegen $x^n - 1 = g(x)h(x)$ folgt $h(x)v(x) = a(x)g(x)h(x)$. Kürzen liefert $v(x) = a(x)g(x)$ und damit $v(x) \in C$. □

SATZ 16.6. *Ein zyklischer linearer Code $C \subset \mathbb{F}[x]/(x^n - 1)$ mit Kontrollpolynom $h(x) = h_0 + h_1 x + \ldots + h_k x^k$ hat die folgende $(n - k) \times n$-Matrix als Kontrollmatrix:*

$$H = \begin{pmatrix} h_k & \ldots & h_1 & h_0 & & 0 & \\ & h_k & \ldots & h_1 & h_0 & & \\ & & \ddots & & \ddots & \ddots & \\ & 0 & & h_k & \ldots & h_1 & h_0 \end{pmatrix}.$$

BEWEIS. Siehe Übungen. □

Hamming-Codes sind bei geeigneter Anordnung der Koordinaten zyklisch. Die Spaltenvektoren einer Kontrollmatrix H eines Hamming-Codes können als die von 0 verschiedenen Elemente des Körpers $\mathbb{F}_{2^r}$ aufgefasst werden. Ist α ein primitives Element von $\mathbb{F}_{2^r}$, dann ist beispielsweise

$$H = (1\ \alpha\ \alpha^2\ \ldots\ \alpha^{2^r - 2})$$

eine Kontrollmatrix eines $[2^r - 1, 2^r - 1 - r]$-Hamming-Codes. In der i-ten Spalte von H steht α^i, als Spaltenvektor über $\mathbb{F} = \{0, 1\}$ geschrieben.

BEISPIEL. Da das Polynom $f(x) := x^3 + x + 1$ in $\mathbb{F}_2[x]$ irreduzibel ist, lässt sich $\mathbb{F}_8$ als $\mathbb{F}_2[x]/(f)$ konstruieren. x ist ein primitives Element dieses Körpers. Als Kontrollmatrix ergibt sich

$$H = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

wobei die Spalten zu den Elementen 1, x, x^2, …, x^6 ($\equiv x^2+1 \pmod{f}$) korrespondieren.

Allgemein gilt:

Satz 16.7. *Sei $r \in \mathbb{N}$, α ein primitives Element von $\mathbb{F}_{2^r}$ und H die oben beschriebene Kontrollmatrix eines $[2^r-1, 2^r-r-1]$-Hamming-Codes C. Dann gilt:*

(1) *Ein Polynom $v(x) \in \mathbb{F}_2[x]/(x^n-1)$ ist genau dann in C enthalten, wenn $v(\alpha)=0$ in $\mathbb{F}_{2^r}$, d.h.*

$$v(x) \in C \iff v(\alpha) = 0 \text{ in } \mathbb{F}_{2^r}\,.$$

(2) *C ist zyklisch und hat das Minimalpolynom $g(x)$ von α als Generatorpolynom.*

Beweis. (1) Sei $n := 2^r - 1$ und $v(x) = v_0 + v_1 x + \ldots + v_{n-1}x^{n-1}$. Dann gilt:

$$\begin{aligned} & v(x) \in C \\ \iff\quad & Hv^T = 0 \\ \iff\quad & v_0 + v_1\alpha + \ldots + v_{n-1}\alpha^{n-1} = 0 \text{ in } \mathbb{F}_{2^r} \\ \iff\quad & v(\alpha) = 0\,. \end{aligned}$$

(2) Nach Lemma 10.7 gilt $v(\alpha)=0$ genau dann, wenn für das Minimalpolynom $g(x)$ von α gilt, dass $g(x) \mid v(x)$. Mit Aussage (1) sind die Codewörter also gerade die Vielfachen von $g(x)$. Folglich ist C nach Satz 16.2 zyklisch, und $g(x)$ ist ein Generatorpolynom. □

17. BCH-Codes

Wir betrachten zunächst eine weitere Möglichkeit zur Konstruktion einer Kontrollmatrix für einen zyklischen Code C. Sie ist theoretisch wichtig, für das praktische Decodieren jedoch weniger geeignet. Sei $g(x) = g_1(x)g_2(x)\cdots g_k(x)$ das Generatorpolynom von C, zerlegt in seine irreduziblen Teiler $g_1(x), \ldots, g_k(x) \in \mathbb{F}_q[x]$. Wir können dann $g_1(x), \ldots, g_k(x)$ als Minimalpolynome von Elementen $\alpha_1, \ldots, \alpha_l$ in einem Erweiterungskörper von $\mathbb{F}_q$ auffassen. Wir setzen nun voraus, dass $g_1(x), \ldots, g_k(x)$ alle voneinander verschieden sind. Nach den Eigenschaften von Minimalpolynomen ist dann $c(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$ genau dann ein Vielfaches von $g(x)$, falls

$$c(\alpha_1) = c(\alpha_2) = \ldots = c(\alpha_l) = 0$$

gilt. In Matrizenschreibweise lautet diese Bedingung

$$H \cdot c^T$$

mit $c = (c_0, c_1, \dots, c_{n-1})$ und der Matrix

$$H := \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha_l & \alpha_l^2 & \cdots & \alpha_l^{n-1} \end{pmatrix}.$$

H ist also eine Kontrollmatrix des Codes.

Bose, Chaudhuri und Hocquenghem haben vorgeschlagen, solche zyklischen Codes zu betrachten, für die $\alpha_1 = \alpha, \alpha_2 = \alpha^2, \dots, \alpha_l = \alpha^l$ für ein geeignetes α gilt.

Definition 17.1. Ein zyklischer linearer Code $C \subset \mathbb{F}_q[x]/(x^n - 1)$ mit Generatorpolynom $g(x)$ heißt *BCH-Code*, falls ein α in einem Erweiterungskörper von $\mathbb{F}_q$ und eine natürliche Zahl $l < n$ existiert, so dass gilt:

(1) Die *Ordnung* von α ist n, d.h. $\alpha^n = 1$ und $\alpha^j \neq 1$ für $j < n$.
(2) $g(x)$ ist das Produkt der Minimalpolynome von $\alpha, \alpha^2, \dots, \alpha^l$, wobei jedes Minimalpolynom in $g(x)$ als Faktor nur einmal auftritt.

Für einen BCH-Code ist also

$$H := \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^l & \alpha^{2l} & \cdots & \alpha^{l(n-1)} \end{pmatrix}$$

Kontrollmatrix.

Hamming-Codes sind spezielle BCH-Codes. Nach den Überlegungen des Abschnitts über zyklische Codes gilt für einen $[2^r - 1, 2^r - r - 1]$-Hamming-Code mit primitivem Element α: Die Blocklänge n ist $2^r - 1$, und, da für das Minimalpolynom g von α nicht nur $g(\alpha) = 0$, sondern (nach Lemma 13.4) auch $g(\alpha^2) = g(\alpha)^2 = 0$ gilt, ist g das Minimalpolynom von α und α^2, d.h. $l = 2$.

Satz 17.2. *Der Minimalabstand eines BCH-Codes ist mindestens* $l + 1$.

Im Falle der Hamming-Codes ergibt sich also der bereits bekannte Minimalabstand von 3.

Beweis. Sei $H \in \mathbb{F}^{l \times n}$ die Kontrollmatrix des BCH-Codes C. Da die Elemente der ersten Zeile von H alle voneinander verschieden sind, ist jede $l \times l$-Untermatrix von H eine Vandermonde-Matrix und damit regulär. Gilt daher $H \cdot c^T = 0$ für ein $c \neq 0$, so muß c mindestens das Gewicht $l + 1$ haben. Da C ein linearer Code ist, folgt die Behauptung aus Lemma 14.7. □

Beispiel. Wir konstruieren einen binären BCH-Code mit $n = 15$ und $l = 4$.

Das Polynom

$$m(x) := x^4 + x + 1$$

ist in $\mathbb{F}_2[x]$ irreduzibel. (Es hat in $\mathbb{F}_2$ keine Nullstellen und könnte höchstens in zwei Polynome vom Grad 2 zerfallen. Da die formale Ableitung $m'(x) = 1$ ist, müssten die beiden Faktoren verschieden sein. Es gibt aber in $\mathbb{F}_2[x]$ nur ein irreduzibles Polynom vom Grad 2, nämlich x^2+x+1.) Nach Satz 13.7 teilt $m(x)$ das Polynom $x^{16}-x$, daher besitzt $m(x)$ in $\mathbb{F}_{16}$ eine Nullstelle α und ist (mit der Argumentation wie im Beweis von Satz 13.7) das Minimalpolynom von α.

Zeige: α ist ein primitives Element in $\mathbb{F}_{16}$.

Nach dem Satz von Euler gilt $\alpha^{15} = 1$. Es ist jedoch $\alpha^3 \neq 1$ (da sich ansonsten ein Widerspruch zur Minimalpolynomeigenschaft von $m(x)$ ergeben würde), und $\alpha^5 = \alpha^2 + \alpha \neq 1$. Daher ist α primitives Element in $\mathbb{F}_{16}$.

$m(x)$ ist auch das Minimalpolynom von α^2 und α^4, denn es gilt $m(\alpha^2) = m(\alpha)^2 = 0$ und $m(\alpha^4) = m(\alpha^2)^2 = 0$. Wir bestimmen nun das Minimalpolynom von α^3 durch den Ansatz

$$0 \;=\; a + b\alpha^3 + c\alpha^6 + d\alpha^9 + e\alpha^{12}$$

mit $a, \ldots, e \in \{0,1\}$ (da $\alpha^{15} = 1$ ist es höchstens vom Grad 4). Aus $\alpha^4 = \alpha + 1$ ergibt sich

$$\begin{aligned} \alpha^6 &= \alpha^4\alpha^2 = \alpha^3 + \alpha^2\,, \\ \alpha^9 &= (\alpha^4)^2\alpha = (\alpha+1)^2\alpha = \alpha^3 + \alpha\,, \\ \alpha^{12} &= (\alpha^4)^3 = (\alpha+1)^3 = \alpha^3 + \alpha^2 + \alpha + 1\,. \end{aligned}$$

Da α nicht Nullstelle eines Polynoms vom Grad ≤ 3 ist, $a + e = 0$, $d + e = 0$, $c + e = 0$, $b + c + d + e = 0$, d.h. $a = b = c = d = e$. Es gibt also genau ein normiertes Polynom $n(x) \neq 0$ vom Grad höchstens 4, so dass $n(\alpha^3) = 0$, nämlich

$$n(x) \;:=\; x^4 + x^3 + x^2 + x + 1\,,$$

d.h. $n(x)$ ist das Minimalpolynom von α^3. Das Generatorpolynom des Codes lautet deshalb

$$g(x) \;:=\; m(x)n(x) = x^8 + x^7 + x^6 + x^4 + 1\,.$$

$g(x)$ ist das Produkt der Minimalpolynome von $\alpha, \ldots, \alpha^4$, und nach Satz 17.2 ist der Minimalabstand von C folglich mindestens 5. Der Code kann also 2 Fehler korrigieren. Die Ordnung von α ist 15, und aufgrund Satz 16.3 ergibt sich die Dimension von C als $m = n - \operatorname{grad} g = 7$. Es liegt ein (15,7)-Code vor, d.h. der Code sendet Wörter der Länge 15, die 7 Informationsstellen haben. $c \in \mathbb{F}_2^{15}$ ist genau dann ein Codewort, wenn für das zugehörige Polynom $c(x) \in \mathbb{F}_2[x]/(x^{15}-1)$

$$c(\alpha) \;=\; c(\alpha^3) \;=\; 0$$

gilt. Genau dann wird $c(x)$ von den Minimalpolynomen $m(x)$ und $n(x)$ geteilt, und damit von $g(x)$.

Codieren. Für das Codieren von Wörtern der Länge 7 bestehen verschiedene Möglichkeiten. Dies kann mit der in Korollar 16.4 angegebenen Generatormatrix geschehen. Übersichtlich ist die folgende Methode. Dem Wort $w = (w_0, \dots, w_6)$ ordnen wir das Polynom

$$c_1(x) \;=\; w_0x^8 + w_1x^9 + \ldots + w_6x^{14}$$

zu. Wir ergänzen es wie folgt mit einem Polynom $c_2(x)$ vom Grad höchstens 7 zu

$$c(x) \;=\; c_1(x) + c_2(x)\,,$$

so dass $c(x)$ ein Vielfaches von $g(x)$ wird. Für ein Polynom $q(x)$ gilt

$$c_1(x) \;=\; q(x)g(x) - c_2(x) \;=\; q(x)g(x) + c_2(x)\,.$$

$c_2(x)$ ist also eindeutig bestimmt als der Rest, der bei Division von $c_1(x)$ durch $g(x)$ übrig bleibt.

Decodieren. Sei $f = (f_0, \dots, f_{14})$ der Vektor der Fehler, es wird anstelle von c also die Nachricht $\tilde{c} = c + f$ empfangen. Für die zugehörigen Polynome $f(x)$ und $\tilde{c}(x)$ gilt dann

$$f(\alpha) \;=\; \tilde{c}(\alpha)\,,\; f(\alpha^2) \;=\; \tilde{c}(\alpha^2)\,,\; f(\alpha^3) \;=\; \tilde{c}(\alpha^3)\,.$$

Der Empfänger kann nun mit Hilfe des quadratischen Polynoms

$$p(x) \;=\; \tilde{c}(\alpha)x^2 + \tilde{c}(\alpha^2)x + \tilde{c}(\alpha^3) + \tilde{c}(\alpha)\tilde{c}(\alpha^2)$$

Fehler korrigieren. Wir unterscheiden drei Fälle:

Fall 1: Gibt es keine Übertragungsfehler, dann gilt $f(x) = 0$ und $p(x) = 0$.

Fall 2: Bei einem Übertragungsfehler ist $f(x)$ von der Gestalt x^r und damit

$$\begin{aligned} p(x) \;&=\; \alpha^r x^2 + \alpha^{2r}x + \alpha^{3r} + \alpha^r\alpha^{2r} \\ &=\; \alpha^r x(x + \alpha^r)\,. \end{aligned}$$

Fall 3: Bei zwei Übertragungsfehlern gilt $f(x) = x^r + x^s$ mit $r \neq s$. In diesem Fall ergibt sich

$$\begin{aligned} p(x) \;&=\; (\alpha^r + \alpha^s)x^2 + (\alpha^{2r} + \alpha^{2s})x + \alpha^{3r} + \alpha^{3s} + (\alpha^r + \alpha^s)(\alpha^{2r} + \alpha^{2s}) \\ &=\; (\alpha^r + \alpha^s)(x + \alpha^r)(x + \alpha^s)\,. \end{aligned}$$

Der Empfänger kann diese Fälle unterscheiden, indem er die Nullstellen von $p(x)$ bestimmt. Genau dann liegt Fall 2 vor, wenn 0 eine (einfache) Nullstelle ist, und ansonsten Fall 3. Gleichzeitig kann er in diesen beiden Fällen r bzw. r, s aus den Nullstellen bestimmen, da α ein primitives Element ist. Er kann also die falsch übertragenen Bits lokalisieren und korrigieren.

BCH-Codes, für die $n = q - 1$ gilt, heißen *Reed-Solomon-Codes.* Diese Codes werden z.B. in CD-Spielern verwendet. Reed-Solomon-Codes sind besonders gut geeignet, um sogenannte „burst“ Fehler zu korrigieren, d.h. lange Ketten aufeinanderfolgender fehlerhafter

Symbole (z.B. durch einen Kratzer). Zudem existieren effiziente Decodieralgorithmen für Reed-Solomon-Codes.

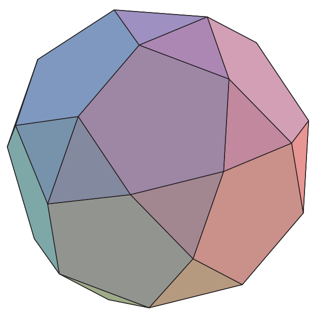
Teil 2

Diskrete geometrische Strukturen

18. Motivation: Algorithmische Geometrie und kombinatorische Optimierung

Im Mittelpunkt des zweiten Teils stehen diskrete geometrische Strukturen, insbesondere Polytope, die als konvexe Hülle endlich vieler Punkte im n-dimensionalen Raum $\mathbb{R}^n$ definiert sind. Polytope besitzen endlich viele Ecken, und wir werden die (anschaulich einsichtige, aber nichttriviale) zentrale Aussage herleiten, dass Polytope auch als Durchschnitt von endlichen vielen Halbräumen dargestellt werden können.

Polytope sind das geometrische Grundmodell der algorithmischen Geometrie und Visualisierung. Eine grundlegende Frage hierbei ist beispielsweise, wie viele Seiten ein Polytop mit k Ecken maximal haben kann.



Aufwendige Visualisierungen (z.B. in der Medizin) werden meist mittels polytopalen Modellen realisiert.

Polytope in beliebigen Dimensionen sind zudem auch das Grundmodell der linearen und kombinatorischen Optimierung. Wir betrachten zwei einfache Beispiele typischer Probleme der kombinatorischen Optimierung.

Job-Zuweisungsproblem (Job Assignment Problem): Eine Menge von n zu erledigenden Aufträgen (Jobs) soll auf Prozessoren (oder menschliche Angestellte) übertragen werden. Die Erledigung jedes Auftrags i erfordert eine festgelegte Zeit t_i, und wir nehmen hierbei an, dass alle Prozessoren, die den Job ausführen können, gleich effizient sind. Mehrere Prozessoren können gleichzeitig zur Erledigung eines Jobs beitragen, und ein Prozessor kann zur Erledigung mehrerer Jobs beitragen (aber nicht zur gleichen Zeit). Ziel ist es, alle Jobs so schnell wie möglich zu erledigen.

In diesem Modell genügt es, für jeden Prozessor j die Zeit x_{ij} zu beschreiben, die er an dem Job i arbeiten soll. Die Reihenfolge, in der der Prozessor die Jobs ausführt, ist nicht von Bedeutung, da der Zeitpunkt, zu dem alle Jobs erledigt sind, offensichtlich nur von der maximalen Gesamtarbeitszeit abhängt, die einem Prozessor zugewiesen wurde. Daher muss folgendes Problem gelöst werden.

Job Assignment Problem:

Gegeben: Eine Menge von Zahlen $t_1, \ldots, t_n > 0$ (die Ausführungszeiten für n Aufträge), eine Zahl $m \in \mathbb{N}$ von Prozessoren, und eine nichtleere Teilmenge $S_i \subset \{1, \ldots, m\}$ von Prozessoren für jeden Auftrag $i \in \{1, \ldots, n\}$.

Aufgabe: Bestimme Zahlen $x_{ij} \geq 0$ für alle $i \in \{1, \ldots, n\}$ und $j \in S_i$, so dass $\sum_{j \in S_i} x_{ij} = t_i$ für $i = 1, \ldots, n$ und $\max_{j \in \{1,\ldots,m\}} \sum_{i: j \in S_i} x_{ij}$ minimal ist

Bin Packing Problem: Gegeben seien n Objekte, jedes mit einer bestimmten Größe, und einige Behälter (Bins) gleicher Kapazität. Wir möchten die (unzerlegbaren) Objekte so auf die Behälter aufteilen, dass so wenige Behälter wie möglich benötigt werden. Natürlich soll hierbei die Gesamtgröße der einem Behälter zugewiesenen Objekte die Kapazität des Behälters nicht übersteigen. Damit kann das Problem wie folgt formuliert werden:

Bin Packing Problem:

Gegeben: Eine Liste von Zahlen $a_1, \ldots, a_n \in \mathbb{N}$, die Kapazität $K \in \mathbb{N}$ der Behälter.

Aufgabe: Finde ein $k \in \mathbb{N}$ und eine Zuweisungsfunktion $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ mit $\sum_{i: f(i) = j} a_i \leq K$ für alle $j \in \{1, \ldots, k\}$, so dass k minimal ist.

Probleme diesen Typs treten beispielsweise auch bei Verschnittproblemen auf. Bei einem einfachen eindimensionalen Verschnittproblem („Cutting Stock Problem“) sind n Stäbe gleicher Länge (etwa 1 Meter) sowie Zahlen $a_1, \ldots, a_n > 0$ gegeben. Wir möchten nun so wenige Stäbe wie möglich zerschneiden, um die Stäbe der vorgebenen Längen zu erhalten.

Das Job Assignment Problem kann als sogenanntes lineares Optimierungsproblem formuliert werden. Bezeichnet T die Variable den Zeitpunkt, zu dem alle Aufträge erledigt sind, dann gilt es, folgendes Problem zu lösen

$$\begin{array}{lrcll} & \min T & & & \\ \text{unter den Nebenbed.} & \sum\limits_{j \in S_i} x_{ij} & = & t_i\,, & i \in \{1, \ldots, n\}\,, \\ & x_{ij} & \geq & 0\,, & i \in \{1, \ldots, n\}\,,\ j \in S_i\,, \\ & \sum\limits_{i: j \in S_i} x_{ij} & \leq & T\,, & j \in \{1, \ldots, m\}\,. \end{array}$$

Die Zahlen t_i und die Mengen S_i ($i \in \{1, \ldots, n\}$) sind gegeben, die Variablen x_{ij} und T sind gesucht. Solch ein Optimierungsproblem mit einer linearen Zielfunktion und linearen Nebenbedingungen heißt *lineares Optimierungsproblem* oder kurz *lineares Programm.* Unter der Voraussetzung, dass der Zulässigkeitsbereich (d.h. die Menge der—nicht notwendig optimalen—Lösungen, die die Nebenbedingung erfüllen) nicht leer ist, ist diese Menge ein Polytop. Ferner werden wir sehen, dass die Menge der Optimallösungen stets eine der endlich vielen Ecken enthält.

Das Bin Packing Problem kann als lineares Optimierungsproblem mit zusätzlichen Ganzzahligkeitsbedingungen formuliert werden. Unter der Voraussetzung, dass für alle $i \in \{1, \ldots, n\}$ die Eigenschaft $a_i \leq K$ gilt, hat das Problem eine Lösung, und die Anzahl der benötigten Behälter ist trivialerweise durch n beschränkt. Bezeichnet die binäre Variable $y_j \in \{0, 1\}$, ob der j-te Behälter leer oder nichtleer bleibt, und $x_{ij} \in \{0, 1\}$, ob Element a_i dem j-ten Behälter zugeordnet wird, dann lässt sich das Problem wie folgt als ganzzahliges Optimierungsproblem formulieren:

$$
\begin{array}{lrcll}
 & \min \sum_{j=1}^{n} y_j & & & \\
\text{unter den Nebenbed.} & \sum_{i=1}^{n} a_i x_{ij} & \leq & K\,, & j \in \{1, \ldots, n\}\,, \\
 & \sum_{i=1}^{n} x_{ij} & \leq & n y_j\,, & i, j \in \{1, \ldots, n\}\,, \\
 & 0 \leq x_{ij}, y_j & \leq & 1\,, & i, j \in \{1, \ldots, n\}\,, \\
 & x_{ij}, y_j & \in & \mathbb{Z}\,, & i, j \in \{1, \ldots, n\}\,.
\end{array}
$$

Hierbei gewährleistet die zweite Nebenbedingung, dass der Behälter als nichtleer gewertet werden muss, sobald mindestens ein Objekt darin abgelegt wird. Die Bestimmung der optimalen Lösung eines Bin Packing Problems erfordert also die Bestimmung optimaler *Gitterpunkte* (des Gitters $\mathbb{Z}^n$) in einem Polytop.

19. Polytope

Definition 19.1. *(Affine Hülle, affine Unabhängigkeit)*
Sei $A \subset \mathbb{R}^n$. Eine *Affinkombination* von Punkten in A ist eine Linearkombination $\sum_{i=1}^{k} \lambda_i a_i$ mit $k \in \mathbb{N}$, $\lambda_i \in \mathbb{R}$, $a_i \in A$, $1 \leq i \leq k$ sowie $\sum_{i=1}^{k} \lambda_i = 1$. Die Menge aller Affinkombinationen von A heißt *affine Hülle* von A (aff A). $a_1, \ldots, a_k \in \mathbb{R}^n$ heißen *affin unabhängig*, wenn sie einen affinen Unterraum der Dimension k erzeugen.

Definition 19.2. *(Konvexe Hülle)*
Sei $A \subset \mathbb{R}^n$. Eine *Konvexkombination* von Punkten $a_1, \ldots, a_k \subset A$ ist eine Linearkombination $\sum_{i=1}^{k} \lambda_i a_i$ mit $k \in \mathbb{N}$, $\lambda_i \geq 0$, $a_i \in A$, $1 \leq i \leq k$ sowie $\sum_{i=1}^{k} \lambda_i = 1$. Die Menge aller Konvexkombinationen von A heißt *konvexe Hülle* von A.

Definition 19.3. *(Polytop)*
Eine Menge $P \subset \mathbb{R}^n$ heißt *Polytop*, wenn sie als konvexe Hülle endlich vieler Punkte dargestellt werden kann. Ein Polytop der (affinen) Dimension k heißt k-Polytop. Die konvexe Hülle $k+1$ affin unabhängiger Punkte heißt *k-Simplex*.

Bemerkung 19.4. Ein Polytop ist eine abgeschlossene, beschränkte Teilmenge des $\mathbb{R}^n$.

Polytope sind die zentralen Objekte der algorithmischen Geometrie sowie der kombinatorischen Optimierung.

Seiten eines Polytops. Jede Hyperebene $H = \{x \in \mathbb{R}^n \ : \ \sum_{i=1}^n a_i x_i = b\}$ zerlegt $\mathbb{R}^n$ in zwei Teilräume

$$\begin{aligned} H^+ &:= \{x \in \mathbb{R}^n \ : \ \sum_{i=1}^n a_i x_i \geq b\} \\ \text{und } H^- &:= \{x \in \mathbb{R}^n \ : \ \sum_{i=1}^n a_i x_i \leq b\}\,. \end{aligned}$$

DEFINITION 19.5. *(Stützende Hyperebene)*
Eine Hyperebene $H \subset \mathbb{R}^n$ *stützt* ein n-Polytop $P \subset \mathbb{R}^n$, falls $H \cap P \neq \emptyset$ und P vollständig in einem der beiden Halbräume H^+ oder H^- enthalten ist.

DEFINITION 19.6. *(Seite eines Polytops)*
Sei $P \subset \mathbb{R}^n$ ein n-Polytop. Der Durchschnitt $P \cap H$ von P mit einer Stützhyperebene H heißt *Seite* von P. Eine Seite der (affinen) Dimension j heißt j-Seite. Eine 0-Seite heißt *Ecke*, eine $(n-1)$-Seite heißt *Facette.*

Im folgenden fügen wir zu diesen *echten* Seiten zwei *unechte* hinzu: die leere Seite (Dimension -1) und P.

Wir haben den Begriff der Stützhyperebene und der Seite für volldimensionale Polytope erklärt. Die Begriffe übertragen sich unmittelbar auf allgemeine k-Polytope P, indem sie jeweils auf die affine Hülle aff P bezogen werden.

SATZ 19.7. *Der Rand eines volldimensionalen Polytops P ist die Vereinigung seiner echten Seiten.*

BEWEIS. Offensichtlich ist die Vereinigung der Seiten von P im Rand von P enthalten. Die umgekehrte Richtung folgt unmittelbar aus der folgenden grundlegenden Aussage der konvexen Analysis:

Sei K eine abgeschlossene, beschränkte, konvexe Teilmenge des $\mathbb{R}^n$. Dann ist jeder Punkt des Randes von K in einer Stützhyperebene enthalten.

Eine Konsequenz dieser Aussage ist, daß durch jeden Punkt p des Randes von P eine stützende Hyperebene gelegt werden kann. Folglich ist p in einer echten Seite enthalten. $\square$

SATZ 19.8. *Die Anzahl der Seiten eines Polytops ist endlich. Seiten von Polytopen sind ebenfalls Polytope.*

BEWEIS. Sei $P = \operatorname{conv} X$ für eine endliche Menge X. Es genügt zu zeigen, dass jede echte Seite von P die konvexe Hülle einer Teilmenge von X ist.

Sei H eine Stützhyperebene von P, und sei $X' := X \cap H$.

Zeige: $H \cap P = \operatorname{conv} X'$.

„$\supset$“: klar.

"$\subset$": Annahme: Sei $p \in P$ mit $p \notin X'$.
Sei $H = \{x \in \mathbb{R}^n : \sum_i a_i x_i = c\}$ und o.B.d.A. $P \in H^+$. Für jedes $x' \in X'$ und damit für jedes $x' \in \operatorname{conv} X'$ gilt $\sum_i a_i x_i = c$. Für jedes $x \in X \setminus X'$ und damit für jedes $x \in \operatorname{conv}(X \setminus X')$ gilt $\sum_i a_i x_i > c$. Da $p \in \operatorname{conv} X$, folgt aus $p \notin X'$ unmittelbar $\sum_i a_i p_i > c$. Folglich ist $p \notin H$. □

Satz 19.9. *Ein Polytop ist die konvexe Hülle seiner Ecken.*

Beweis. Sei $P = \operatorname{conv} X$ für eine endliche Menge X. Nach sukzessivem Entfernen aller Punkte $p \in X$, die als Konvexkombination der anderen ausgedrückt werden können, verbleibt eine minimale Teilmenge $X' \subset X$, so dass $P = \operatorname{conv} X'$. Wir zeigen nun, dass jeder Punkt aus X' eine Ecke von P ist. Sei $p \in X'$. Da X' minimal ist, ist p nicht in der konvexen Hülle der anderen Punkte enthalten. Nach dem Trennungssatz (siehe z.B. Gerd Fischer, Analytische Geometrie) existiert eine Hyperebene H', die p von $\operatorname{conv}(X' \setminus \{p\})$ separiert. Die durch p verlaufende Hyperebene H parallel zu H' stützt P und enthält außer p keinen anderen Punkt aus X'. Folglich ist p eine Ecke von P. □

Die Darstellung eines Polytops als konvexe Hülle endlich vieler Punkte bezeichnet man auch als $\mathcal{V}$-*Darstellung* („vertices"). Die folgenden beiden zentralen Aussagen besagen, daß ein Polytop äquivalent auch als beschränkter Durchschnitt endlich vieler abgeschlossener Halbräume beschrieben werden kann ($\mathcal{H}$-*Darstellung*).

Satz 19.10. *Jedes Polytop ist der Durchschnitt einer endlichen Menge abgeschlossener Halbräume. Speziell: Sei $P \subset \mathbb{R}^n$ ein n-Polytop und $\{F_i : 1 \leq i \leq m\}$ die Menge seiner Facetten, H_i die P stützende Hyperebene entlang F_i, und H_i^- der P enthaltende abgeschlossene Halbraum. Dann gilt*

$$P = \bigcap_{i=1}^{m} H_i^-$$

Beweis. „$\subset$": klar.
„$\supset$" : *Zeige:* Jeder Punkt $p \notin P$ ist nicht im Durchschnitt $\bigcap_{i=1}^m H_i^-$ enthalten.

Sei $p \notin P$ und q ein Punkt im Inneren von P, der nicht in der konvexen Hülle von p und $n-1$ Ecken von P enthalten ist. Ein solcher Punkt existiert, da das Innere eines n-Polytops die Dimension n hat und damit nicht in der Vereinigung einer endlichen Anzahl von Hyperebenen der Dimension $n-1$ enthalten sein kann. Das Segment $\overline{pq}$ schneidet den Rand von p in einem Punkt z, welcher in einer echten Seite von P enthalten ist. Da z nach Definition von q nicht in einer Seite der Dimension $j < n-1$ enthalten ist, existiert ein $i \in \{1, \dots, m\}$ mit $z \in F_i$. Es gilt also $z \in H_i$, $q \in H_i^-$ und damit $p \in H_i^+$, d.h. $p \notin \bigcap_{i=1}^m H_i^-$. □

Satz 19.11. *Ist der Durchschnitt einer endlichen Anzahl abgeschlossener Halbräume beschränkt, dann ist er ein Polytop.*

Beweis. Der Beweis erfolgt durch Induktion über die Dimension n des Raumes. In Dimension 1 ist die Aussage trivial. Sei

$$Q = \bigcap_{i=1}^{m} H_i^-$$

der beschränkte Durchschnitt einer endlichen Anzahl von Halbräumen im $\mathbb{R}^n$. Sei $F_j = H_j \cap Q$, $j \in \{1, \dots, m\}$. F_j ist also ein beschränkter Durchschnitt von Halbr̈aumen in der Hyperebene H_j, der mit einem affinen Raum der Dimension $n-1$ identifiziert werden kann. Nach Induktionsannahme ist F_j ein Polytop in der Hyperebene H_j und daher auch ein Polytop in $\mathbb{R}^n$. Sei V_j die Menge der Ecken von F_j und $V = \cup_{i=1}^{m} V_j$.

Zeige: $Q = \operatorname{conv} V$.
„$\subset$“: Sei $q \in Q$. Falls q Randpunkt von Q: klar, da ein $j \in \{1, \dots, m\}$ mit $q \in F_j$ existiert. Falls q im Inneren von Q enthalten ist: q liegt dann auf einem Segment $\overline{q_0 q_1}$, das der Durchschnitt einer Gerade durch q mit Q ist. Da sowohl q_0 als auch q_1 auf dem Rand von Q liegen, gehören sie zu $\operatorname{conv} V$, folglich auch $q \in \operatorname{conv} V$.

“$\supset$“: folgt unmittelbar, da $V \subset Q$ und Q konvex. □

Beispiel. Um die $\mathcal{H}$-Darstellung des Polytops
$P = \operatorname{conv}\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}\} \subset \mathbb{R}^2$ zu berechnen, schreibe P in der Form

$$P = \{x \in \mathbb{R}^2 \;:\; \exists \lambda_1, \dots, \lambda_4, \quad \sum_{i=1}^{4} \lambda_i = 1,\, x = (\lambda_1 - \lambda_2)e_1 + (\lambda_3 - \lambda_4)e_2\}\,.$$

Eine $\mathcal{H}$-Darstellung des sechsdimensionalen Polytops, dessen zweidimensionale Projektion auf (x_1, x_2) wir suchen, lautet:

$$\begin{array}{rcccl} 0 & \le & x_1 - \lambda_1 + \lambda_2 & \le & 0\,, \\ 0 & \le & x_2 - \lambda_3 + \lambda_4 & \le & 0\,, \\ 1 & \le & \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 & \le & 1\,, \\ 0 & \le & \lambda_1, \lambda_2, \lambda_3, \lambda_4\,. & & \end{array}$$

Die Elimination der Variablen $\lambda_1, \dots, \lambda_4$ kann wie folgt durchgeführt werden („Fourier-Motzkin-Elimination“). Zur Elimination von λ_4 schreiben wir das Ungleichungssystem zunächst in der Form

$$\begin{array}{ccccc} 0 & \le & x_1 - \lambda_1 + \lambda_2 & \le & 0\,, \\ \lambda_3 - x_2 & \le & \lambda_4 & \le & \lambda_3 - x_2\,, \\ 1 - \lambda_1 - \lambda_2 - \lambda_3 & \le & \lambda_4 & \le & 1 - \lambda_1 - \lambda_2 - \lambda_3\,, \\ 0 & \le & \lambda_1, \lambda_2, \lambda_3, \lambda_4\,. & & \end{array}$$

Es existiert genau dann eine Lösung $(x_1, x_2, \lambda_1, \lambda_2, \lambda_3, \lambda_4)$, wenn eine Lösung $(x_1, x_2, \lambda_1, \lambda_2, \lambda_3)$ des folgenden Systems existiert:

$$\begin{array}{ccccc} 0 & \leq & x_1 - \lambda_1 + \lambda_2 & \leq & 0\,, \\ \lambda_3 - x_2 & \leq & 1 - \lambda_1 - \lambda_2 - \lambda_3\,, & & \\ 1 - \lambda_1 - \lambda_2 - \lambda_3 & \leq & \lambda_3 - x_2\,, & & \\ 0 & \leq & \lambda_3 - x_2\,, & & \\ 0 & \leq & 1 - \lambda_1 - \lambda_2 - \lambda_3\,, & & \\ 0 & \leq & \lambda_1, \lambda_2, \lambda_3\,. & & \end{array}$$

Auf gleiche Weise können sukzessive $\lambda_3, \lambda_2, \lambda_1$ elimiert werden, wobei es zweckmäßig ist, (offensichtlich) redundante Gleichungen sofort zu entfernen.

Am Ende dieser sukzessiven Elimination erhält man eine $\mathcal{H}$-Darstellung von P, etwa

$$P = \left\{ x \in \mathbb{R}^2 : \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix} x \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\}.$$

Zum Abschluss dieses Abschnitts seien noch zwei weitere Aussagen ohne Beweis angegeben.

Satz 19.12. *Eine echte Seite eines n-Polytops P ist eine Seite einer $(n-1)$-Seite von P. Umgekehrt ist eine Seite einer Seite von P auch eine Seite von P.*

Satz 19.13. *Für ein n-Polytop P gilt:*

(1) *Der Durchschnitt einer Menge von Seiten von P ist eine Seite von P.*
(2) *Jede $(n-2)$-Seite von P ist der Durchschnitt zweier $(n-1)$-Seiten von P.*
(3) *Für jedes Paar (j, k) mit $0 \leq j \leq k < n$ ist eine j-Seite der Durchschnitt aller sie enthaltenden k-Seiten*

Polyeder. Allgemein bezeichnet man den Durchschnitt endlich vieler Halbräume als *Polyeder.* Ein Polytop ist also ein beschränktes Polyeder.

20. Lineare Optimierung

Grundtechnik vieler Techniken der algorithmischen Geometrie und der kombinatorischen Optimierung ist die lineare Optimierung. Wir betrachten lineare Optimierungsprobleme (kurz: lineare Programme, LP) der folgenden Form:

Eingabe: Eine Matrix $A \in \mathbb{R}^{m \times n}$ und Vektoren $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Aufgabe: Bestimme einen Vektor $x \in \mathbb{R}^n$, so dass $Ax \leq b$ und $c^T x$ maximiert wird, oder entscheide, dass $\{x \in \mathbb{R}^n : Ax \leq b\}$ leer ist, oder entscheide, dass für alle $\alpha \in \mathbb{R}$ ein $x \in \mathbb{R}^n$ mit $Ax \leq b$ und $c^T x > \alpha$ existiert.

Eine *zulässige Lösung* eines linearen Programms ist ein Vektor x mit $Ax \leq b$. Eine zulässige Lösung, die das Maximum annimmt, heißt eine *Optimallösung*.

Wie aus der Problemformulierung hervorgeht, gibt es zwei Möglichkeiten, in denen das LP keine Lösung hat: Das Problem kann *unzulässig* sein (d.h. $P := \{x \in \mathbb{R}^n \,:\, Ax \leq b\} = \emptyset$), oder *unbeschränkt* (d.h. für alle $\alpha \in \mathbb{R}$ existiert ein $x \in P$ so dass $c^T x > \alpha$). Ist ein LP weder unzulässig noch unbeschränkt, dann werden wir im nächsten Abschnitt sehen, dass es eine optimale Lösung hat. Dies rechtfertigt die Notation $\max\{c^T x \,:\, Ax \leq b\}$ anstelle von $\sup\{c^T x \,:\, Ax \leq b\}$.

Der Zulässigkeitsbereich eines linearen Programms ist ein Polyeder. Ist er beschränkt, dann ist er ein Polytop. Das Lösen von linearen Programmen (für alle Zielfunktionen) entspricht geometrisch daher dem Übergang von einer $\mathcal{H}$-Darstellung des Polytops zu einer $\mathcal{V}$-Darstellung.

Dualität. Mit Hilfe der Dualitätstheorie der linearen Programmierung lassen sich insbesondere Optimalitätsbedingungen angeben.

Gegeben sei ein Randpunkt y des Zulässigkeitsbereiches eines linearen Programms $\max\{c^T \,:\, Ax \leq b\}$. Einige der definierenden Bedingungen $Ax \leq b$ sind daher im Punkt y mit Gleichheit erfüllt. Durch Umnummerierung der Zeilen von A und b können wir erreichen, dass

$$\begin{aligned} A_1 y &= b_1\,, \\ A_2 y &< b_2 \end{aligned}$$

mit $A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$. Dann ist y bereits Optimalpunkt des LPs $\{\max c^T x \,:\, A_1 \leq b_1\}$.

Definition 20.1. Eine Teilmenge des $\mathbb{R}^n$ heißt *konvexer Kegel*, wenn K konvex ist und $[0,\infty)K \subset K$ gilt. Zu einer nichtleeren Menge $X \subset \mathbb{R}^n$ heißt der konvexe Kegel

$$\text{pos } X \;:=\; [0,\infty)\text{conv } X$$

die *positive Hülle von K*.

Bezeichnen in obiger Situation $a_1^T, \ldots, a_k^T$ die Zeilen von A_1, dann heißt der konvexe Kegel $N(y) := \text{pos}\{a_1, \ldots, a_k\}$ der *Kegel der äußeren Normalen* in y.

Lemma 20.2. *$N(y) \;=\; \bigcap_{x \in C-y} \{\, c_0 : c_0^T x \leq 0\}$, wobei $C := \{x \in \mathbb{R}^n \,:\, A_1 x \leq b_1\}$.*

Beweis. „$\subset$“: Sei $c_0 \in N(y)$ und $a_1^T \ldots, a_k^T$ die Zeilen von A_1. Dann gibt es $\lambda_1, \ldots, \lambda_k \in [0,\infty)$ mit $c_0 = \sum_{i=1}^k \lambda_j a_j$. Wegen $C - y = \{x \,:\, A_1 x \leq 0\}$ folgt somit für jeden Punkt $x \in C - y$

$$c_0^T x \;=\; \sum_{i=1}^k \lambda_j a_j^T x \;\leq\; 0\,.$$

„$\supset$“: Sei $c_0 \notin N(y)$. Dann gibt es nach dem Trennungssatz einen Vektor w mit $w^T c_0 > 0$, aber $w^T z \leq 0$ für alle $z \in N(y)$. Insbesondere gilt daher $w^T a_j \leq 0$ für alle $j \in \{1, \ldots, k\}$ und somit $A_1 w \leq 0$. Es folgt $w \in C - y$, aber $c_0^T w > 0$, d.h.

$$c_0 \notin \bigcap_{x \in C - y} \{c_1 \,:\, c_1^T x \leq 0\}\,.$$

$\square$

Korollar 20.3. *Seien* $P = \{x \in \mathbb{R}^n \,:\, Ax \leq b\}$, $c \in \mathbb{R}^n$ *und* y *ein Randpunkt von* P. *Der Vektor* y *ist genau dann ein Maximalpunkt des LPs* $\max\{c^T x \,:\, x \in P\}$, *wenn* c *im Kegel* $N(y)$ *der äußeren Normalen an* y *enthalten ist. Dabei wird* $N(y)$ *aufgespannt von den Normalenvektoren* a_i *der* aktiven Nebenbedingungen, *d.h.* a_i^T *durchläuft genau die Zeilenvektoren von* A, *deren zugehörige Nebenbedingungen mit Gleichheit erfüllt sind.*

Beweis. Sei y ein Randpunkt von P. Nach dem voranstehenden Lemma ist y genau dann ein Maximalpunkt, wenn c im Kegel der äußeren Normalen an C $(= \{x \in \mathbb{R}^n \,:\, A_1 x \leq b_1\})$ in y enthalten ist. $\square$

Die Optimalität von y für das LP $\max\{c^T x \,:\, Ax \leq b\}$ wird somit durch die Existenz einer speziellen Lösung des Zulässigkeitsproblems

$$\begin{aligned} A^T u &= c \\ u &\geq 0 \end{aligned}$$

charakterisiert, nämlich einer solchen, für die höchstens solche Komponenten von u von 0 verschieden sind, die zu den aktiven Nebenbedingungen (für y) gehören.

Zu jedem linearen Programm $\max\{c^T x \,:\, Ax \leq b\}$ ist das *duale Programm* als

$$\begin{aligned} &\min b^T y \\ A^T y &= c \\ y &\geq 0 \end{aligned}$$

definiert. Das ursprüngliche Problem $\max\{c^T x \,:\, Ax \leq b\}$ wird auch das *primale Programm* genannt.

Satz 20.4. (Schwacher Dualitätssatz) *Seien* x *und* y *zulässige Lösungen der dualen LPs* $\max\{c^T x \,:\, Ax \leq b\}$ *und* $\min\{b^T y \,:\, A^T y = c\,,\, y \geq 0\}$. *Dann gilt* $c^T x \leq b^T y$.

Beweis. Für zulässige Lösungen x und y gilt

$$c^T x \;=\; (y^T A) x \;=\; y^T (Ax) \;\leq\; y^T b \;=\; b^T y\,.$$

Also $c^T x \leq b^T y$. $\square$

21. Der Simplex-Algorithmus

Der älteste und bekannteste Algorithmus zur linearen Programmierung ist der *Simplex-Algorithmus* (Dantzig, 1951). Wir nehmen zunächst an, dass das Polyeder $P := \{x \in \mathbb{R}^n : Ax \leq b\}$ eine Ecke besitzt, und dass eine Ecke bereits bekannt ist („Startecke"). Anschließend wird gezeigt, wie wir mit einem solchen Algorithmus auch eine Startecke (in einem geeignet transformierten Problem) erhalten.

Grundlegende Idee des Simplex-Algorithmus ist, für die aktuelle Ecke zunächst zu prüfen, ob sie ein Optimalpunkt ist. Dies kann mittels der im vorherigen Abschnitt diskutierten Dualitätstheorie erfolgen. Ist die Ecke kein Optimalpunkt, dann wird entlang der von dieser Ecke ausgehenden Kanten eine benachbarte Ecke ermittelt, die einen größeren Zielfunktionswert hat.

Für eine Menge J von Zeilenindizes bezeichnet A_J die aus den Zeilen in J bestehende Untermatrix von A, und b_J den aus den Komponenten mit Indizes in J bestehenden Untervektor. Wir setzen $a_i := A_{\{i\}}$ und $\beta_i := b_{\{i\}}$.

Simplex-Algorithmus:

Eingabe: Eine Matrix $A \in \mathbb{R}^{m\times n}$ und Spaltenvektoren $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Eine Ecke x von $P := \{x \in \mathbb{R}^n : Ax \leq b\}$.

Ausgabe: Eine Ecke x von P, die $\max\{c^T x : x \in P\}$ annimmt, oder ein Vektor $w \in \mathbb{R}^n$ mit $Aw \leq 0$ und $c^T w > 0$ (d.h. das LP ist unbeschränkt).

1: Wähle eine Menge von n Zeilenindizes J, so dass A_J regulär ist und $A_J x = b_J$ gilt.

2: Berechne $((A_J)^T)^{-1}c$ und füge Nullen hinzu, um einen Vektor y mit $c = A^T y$ zu erhalten, in dem alle Einträge ausserhalb von J Null sind.
Falls $y \geq 0$, dann ENDE mit AUSGABE x und y.

3: Wähle den minimalen Index i mit $y_i < 0$.
Sei w die Spalte von $-(A_J)^{-1}$ mit Index i, so dass $A_{J\setminus\{i\}}w = 0$ und $a_i w = -1$.
Falls $Aw \leq 0$ dann ENDE mit AUSGABE w.

4: Sei $\lambda := \min\left\{\frac{\beta_j - a_j x}{a_j w} : a_j w > 0\right\}$, und sei j der kleinste Zeilenindex, der dieses Minimum annimmt.

5: Sei $J := (J \setminus \{i\}) \cup \{j\}$ und $x := x + \lambda w$. GEHE ZU Schritt 2.

Die Existenz der Menge J in Schritt 1 folgt aus der Tatsache, dass im Falle einer Ecke mindestens n Ungleichungen mit Gleichheit erfüllt sein müssen. Es wird zunächst bestimmt, für welche Indexmenge J' Gleichheit gilt ($|J'| \geq n$), und aus dieser Menge kann mittels Gauß-Elimination eine Menge J mit der benötigten Eigenschaft bestimmt werden.

Die Auswahlregel für i und j in den Schritten 3 und 4 heißt Regel von Bland. Wählt man lediglich ein beliebiges i mit $y_i < 0$ und ein beliebiges j, dann kann es passieren, dass der Algorithmus im Fall überbestimmter Ecken in zyklische Wiederholungen läuft und nicht terminiert.

SATZ 21.1. *Der Simplex-Algorithmus terminiert nach höchstens* $\binom{m}{n}$ *Iterationen. Gibt er* x *und* y *in Schritt 2 aus, dann sind diese Vektoren Optimallösungen der dualen LPs, und es gilt* $c^T x = b^T y$. *Gibt der Algorithmus* w *in Schritt 3 aus, dann gilt* $c^T w > 0$ *und das LP ist unbeschränkt.*

Der Fall der Unzulässigkeit kommt in dem Satz nicht vor, da bei obiger Form des Simplex-Algorithmus die Kenntnis einer Startecke vorausgesetzt wurde.

BEWEIS. Wir zeigen zunächst, dass die folgenden Bedingungen zu jedem Zeitpunkt gelten:

a) $x \in P$;
b) $A_J x = b_J$;
c) A_J ist regulär;
d) $c^T w \geq 0$;
e) $\lambda \geq 0$;

a) und b) sind zu Beginn erfüllt.
Zeige: c), d) und e) sind stets erfüllt.

Schritte 2 und 3 garantieren, dass $c^T w = y^T A w = -y_i > 0$. Wegen Schritt 4 folgt aus $x \in P$ die Eigenschaft $\lambda \geq 0$. c) folgt zu Beginn aus der vor dem Beweis gemachten Bemerkung. Bei der Aktualisierung von J folgt c) aus der Tatsache, dass $A_{J\setminus\{i\}} w = 0$ und $a_j w > 0$, d.h. a_j liegt nicht im Zeilenraum der $(n-1)$-zeiligen Matrix $A_{J\setminus\{i\}}$.
Noch zu zeigen: Schritt 5 erhält a) und b).

Um zu zeigen, dass aus $x \in P$ auch $x + \lambda w \in P$ folgt, unterscheiden wir für einen Zeilenindex k zwei Fälle: Gilt $a_k w \leq 0$, dann ist (wegen $\lambda \geq 0$) $a_k(x + \lambda w) \leq a_k x \leq \beta_k$. Anderenfalls ist $\lambda \leq \frac{\beta_k - a_k x}{a_k w}$ und daher

$$a_k(x + \lambda w) \;\leq\; a_k x + a_k w \frac{\beta_k - a_k x}{a_k w} \;=\; \beta_k\,.$$

(Tatsächlich ist λ in Schritt 5 als größte Zahl mit $x + \lambda w \in P$ gewählt.)

Um b) zu zeigen, betrachten wir zunächt Schritt 4. In diesem gilt $A_J \setminus \{i\} w = 0$ und $\lambda = \frac{\beta_j - a_j x}{a_j w}$. Es folgt

$$A_{J\setminus\{i\}}(x + \lambda w) \;=\; A_{J\setminus\{i\}} x \;=\; b_{J\setminus\{i\}}$$

und

$$a_j(x + \lambda w) \;=\; a_j x + a_j w \frac{\beta_j - a_j x}{a_j w} \;=\; \beta_j\,.$$

Also gilt $A_J x = b_J$ auch nach Schritt 5 weiterhin.
Zeige: Gibt der Algorithmus x und y in Schritt 2 aus, dann sind x und y optimale Lösungen des primalen und des dualen Programms.

x und y sind zulässige Lösungen des primalen und des dualen Programms, und es gilt $c^T x = (y^T A)x = y^T(Ax) = y^T b = b^T y$, da die Komponenten von y außerhalb von J Null sind. Aus dem schwachen Dualitätssatz folgt daher die Optimalität von x und y.

Terminiert der Algorithmus in Schritt 3, dann ist das LP unbeschränkt, da in diesem Fall $x + \mu w \in P$ für alle $\mu \geq 0$, und $c^T w > 0$ (wegen d)).

Zeige schließlich: Der Algorithmus terminiert.

Seien $J^{(k)}$ und $x^{(k)}$ die Menge J bzw. der Vektor x in der k-ten Iteration des Simplexalgorithmus. Falls der Algorithmus nicht nach $\binom{m}{n}$ Iterationen terminiert, dann existieren $k < l$ mit $J^{(k)} = J^{(l)}$. Mit b) und c) folgt $x^{(k)} = x^{(l)}$. Wegen d) und e) wird $c^T x$ im Laufe der Iterationen niemals kleiner, und wird im Fall $\lambda > 0$ echt größer. Folglich gilt in den Iterationen $k, k+1, \ldots, l-1$, dass $\lambda = 0$ und $x^{(k)} = x^{(k+1)} = \ldots = x^{(l)}$.

Sei h der maximale Index, der aus J in einer der Iterationen $k, \ldots, l-1$ entfernt wird, etwa in Iteration p. Der Index h muss wegen $J^{(k)} = J^{(l)}$ auch in einer Iteration $q \in \{k, \ldots, l-1\}$ zu J hinzugefügt worden sein. Sei y' der Vektor y in Iteration p, und sei w' der Vektor w in Iteration q. Es gilt $(y')^T A w' = c^T w' > 0$. Sei daher r ein Index, für den $y'_r a_r w' > 0$ gilt. Da $y'_r \neq 0$ ist, gehört der Index r zu $J^{(p)}$. Im Fall $r > h$ würde der Index r auch zu $J^{(q)}$ und zu $J^{(q+1)}$ gehören, was $a_r w' = 0$ implizieren würde. Also gilt $r \leq h$. Nach Wahl von i in Iteration p gilt jedoch $y'_r < 0$ genau dann wenn $r = h$, und nach Wahl von j in Iteration q gilt $a_r w' > 0$ genau dann wenn $r = h$ (beachte, dass $\lambda = 0$ und $\alpha_r x^{(q)} = \alpha_r x^{(p)} = \beta_r$ wegen $r \in J^{(p)}$). Dies ist ein Widerspruch zu $y'_r a_r w' > 0$. □

Es existieren Beispiele, in denen der Simplex-Algorithmus (mit der Regel von Bland) 2^n Iterationen mit n Variablen und $2n$ linearen Nebenbedingungen benötigt. Dies zeigt, dass der Simplex-Algorithmus mit der Regel von Bland kein Polynomialzeitalgorithmus ist. Es gibt viele andere Möglichkeiten („Pivotregeln"), die Indizes i und j zu wählen, doch es ist nicht bekannt (und die große offene Frage der linearen Optimierung), ob eine Pivotregel existiert, die auf einen Polynomialzeitalgorithmus führt.

Bestimmen einer Startecke. Wir zeigen nun, wie allgemeine lineare Programme mit dem Simplex-Algorithmus gelöst werden können. Hierzu ist noch zu klären, wie eine Startecke gefunden wird. Da es Polyeder gibt, die überhaupt keine Ecke besitzen (z.B. $\{x \in \mathbb{R}^2 : x_1 \leq 0\}$), wird das LP zunächst in eine andere Form gebracht.

Sei $\max\{c^T x : Ax \leq b\}$ ein LP. Wir ersetzen x durch $y - z$ und schreiben das LP in der äquivalenten Form

$$\max\left\{ \begin{pmatrix} c^T & -c^T \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} : \begin{pmatrix} A & -A \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} \leq b\,,\ y, z \geq 0 \right\}.$$

Wir können daher o.B.d.A. annehmen, dass das LP die Form

$$\max\{c^T x : A'x \leq b'\,, A''x \leq b''\,, x \geq 0\} \tag{21.1}$$

mit $b' \geq 0$ und $b'' < 0$ hat. Wir betrachten nun folgendes Hilfsproblem:

$$\min\{(\mathbf{1}^T A'')x + \mathbf{1}^T y \,:\, A'x \leq b'\,,\; A''x + y \geq b''\,,\; x, y \geq 0\}\,, \tag{21.2}$$

wobei $\mathbf{1}$ den aus lauter Einsen bestehenden Vektor bezeichnet.

Da $\begin{pmatrix} x \\ y \end{pmatrix} = 0$ eine Ecke definiert, können wir den Simplexalgorithmus mit dieser Startecke auf das Hilfsproblem anwenden. Das LP ist nicht unbeschränkt, da das Minimum mindestens $\mathbf{1}^T b''$ sein muss. Für jede zulässige Lösung x von (21.1) ist

$$\begin{pmatrix} x \\ b'' - A''x \end{pmatrix}$$

eine optimale Lösung von (21.2). (Beachte jedoch, dass das LP (21.1) unzulässig sein kann.) Ist daher das Minimum von (21.1) größer als $\mathbf{1}^T b''$, dann ist (21.1) unzulässig.

Anderenfalls sei $\begin{pmatrix} x \\ y \end{pmatrix}$ eine optimale Ecke von (21.2).

Zeige: x ist eine Ecke des Zulässigkeitsbereiches von (21.1).

Es gilt $A''x + y = b''$. Bezeichnen n und m die Dimensionen von x und y, dann gibt es eine Menge S von $n + m$ Ungleichungen von (21.2), die mit Gleichheit erfüllt sind, so dass die zu diesen $n + m$ Ungleichungen korrespondierende Untermatrix regulär ist.

Sei S' die Menge der Ungleichungen von $A'x \leq b'$ und von $x \geq 0$, die zu S gehören. Sei S'' die Menge der Ungleichungen von $A''x \leq b''$, für die die korrespondierenden Ungleichungen von $A''x + y \leq b''$ und $y \geq 0$ beide zu S gehören. Wegen $A''x + y = b''$ gilt $|S' \cup S''| \geq |S| - m = n$, und die Ungleichungen von $S' \cup S''$ sind linear unabhängig und an der Stelle x mit Gleichheit erfüllt. Daher erfüllt x mindestens n linear unabhängige Ungleichungen von (21.1) mit Gleichheit; x ist also eine Ecke. Folglich kann x als Startecke für die Anwendung des Simplex-Algorithmus auf das LP (21.1) verwendet werden.

Wie oben erwähnt ist nicht bekannt, ob der Simplex-Algorithmus mit einer geeigneten Pivot-Regel ein Polynomialzeit-Algorithmus ist. Er ist jedoch in der Praxis der beste bekannte Algorithmus.

Es gibt Polynomialzeit-Algorithmen zur Lösung linearer Optimierungsprobleme: den Ellipsoid-Algorithmus (Khachiyan, 1979), der aber nicht praktikabel ist, sowie Innere-Punkt-Verfahren (Karmarkar, 1984).

22. Das Matching-Problem

Eines der klassischen kombinatorischen Optimierungsprobleme, anhand dessen sich sehr viele Grundprobleme und -techniken illustrieren lassen, ist das Matching-Problem.

Definition 22.1. Ein (*ungerichteter*) *Graph* ist ein Paar $G = (V, E)$ mit einer endlichen Menge V, den *Knoten*, und einer endlichen Menge von *Kanten* $E \subset \{\{v, w\} \,:\, v, w \in V\,,\, v \neq w\}$.

Sei $G = (V, E)$ ein ungerichteter Graph. Ein *Matching* (*Zuordnung*) in G ist eine Menge knotendisjunkter Kanten von G. Unter *Kardinalitäts-Matching* versteht man das das folgende kombinatorische Optimierungsproblem, das zahlreiche Anwendungen bei Zuordnungs- und Transportproblemen hat.

KARDINALITÄTS-MATCHING:

Eingabe: Ein Graph $G = (V, E)$.
Ausgabe: Ein Matching in G maximaler Kardinalität.

Erweiternde Wege: Ein zentrales Konzept beim Finden maximaler Matchings sind erweiternde Wege. Ein *Weg* (*der Länge* k) in einem Graphen G ist eine Folge von Knoten $(v_0, v_1, \ldots, v_k)$ mit $e_i := \{v_i, v_{i+1}\} \in G$ (ein *Kantenzug*), bei dem die Knoten paarweise verschieden sind. Ein *Kreis* in G ist ein Kantenzug $(v_0, v_1, \ldots, v_k)$ mit $v_0 = v_k$, bei dem die Knoten $v_0, \ldots, v_{k-1}$ paarweise verschieden sind. Eine Menge M von Kanten *überdeckt* einen Knoten v, wenn v inzident zu einer Kante von M ist.

DEFINITION 22.2. Sei M ein Matching in einem Graphen $G = (V, E)$. Ein Weg P in G heißt *M-erweiternd*, wenn P ungerade Länge hat, beide Endknoten von P nicht von M überdeckt werden und die Kanten von P alternierend in M bzw. nicht in M enthalten sind.

Bezeichnet $E(P)$ die Kantenmenge eines Weges P und $X \Delta Y := (X \setminus Y) \cup (Y \setminus X)$ die symmetrische Differenz zweier Mengen, dann gilt natürlich: Falls P ein M-erweiternder Weg ist, dann ist

$$M' := M \,\Delta\, E(P)$$

wieder ein Matching und erfüllt $|M'| = |M| + 1$. Tatsächlich gilt sogar die folgende Äquivalenz:

SATZ 22.3. *Sei $G = (V, E)$ ein Graph und M ein Matching in G. Dann ist M genau dann ein kardinalitätsmaximales Matching, wenn es keinen M-erweiternden Weg gibt.*

BEWEIS. „$\Longrightarrow$“: Sei M ein kardinalitätsmaximales Matching. Dann kann kein M-erweiternder Weg P existieren, da ansonsten $M \Delta E(P)$ ein Matching mit größerer Kardinalität wäre.

„$\Longleftarrow$“: Sei M nicht maximal. Dann existiert ein Matching M' mit $|M'| > |M|$. Der Graph $G' := (V, M \cup M')$ hat Grad höchstens 2, d.h. jeder Knoten besitzt höchstens zwei ausgehende Kanten. Wir betrachten nun die Zerlegung von G' in Zusammenhangskomponenten, wobei zwei Knoten u und v genau dann in der gleichen Zusammenhangskomponente liegen, wenn es einen Weg von u nach v gibt.

Wegen grad $G' \leq 2$ ist jede Zusammenhangskomponente von G' entweder ein Weg (eventuell mit Länge 0) oder ein Kreis. Da $|M'| > |M|$, enthält mindestens eine dieser

Komponenten mehr Kanten von M' als von M. Eine solche Komponente definiert einen M-erweiternden Weg. □

23. Bipartites Kardinalitäts-Matching

Ein Graph heißt *bipartit*, wenn die Knotenmenge in zwei disjunkte Teilmengen V_1 und V_2 zerlegt werden kann, so dass alle Kanten von G von der Form $\{v_1, v_2\}$ mit $v_1 \in V_1$, $v_2 \in V_2$ sind.

Das folgende klassische Resultat von König (1931) charakterisiert die maximale Größe eines Matchings in einem bipartiten Graphen. Hierbei heißt eine Menge C von Knoten eines Graphen eine *Knotenüberdeckung*, falls jede Kante von G inzident zu mindestens einem Knoten von C ist.

Definition 23.1. Für einen Graphen G heißen

$$\begin{aligned} \nu(G) &:= \text{ die maximale Größe eines Matchings in } G \\ \tau(G) &:= \text{ die minimale Größe einer Knotenüberdeckung von } G \end{aligned}$$

die *Matching-Zahl* bzw. die *Knotenüberdeckungszahl* von G.

Für jeden Graphen G gilt $\nu(G) \leq \tau(G)$, da je zwei Kanten eines Matchings zu verschiedene Knoten einer Knotenüberdeckung inzident sein müssen. Für den Fall eines bipartiten Graphen gilt Gleichheit (und wir werden in Abschnitt 24 von einem höheren Standpunkt hierauf zurückkommen).

Satz 23.2. (König.) *Für jeden bipartiten Graphen G gilt*

$$\nu(G) = \tau(G)\,,$$

d.h., die maximale Größe eines Matchings in einem bipartiten Graphen stimmt mit der minimalen Größe einer Knotenüberdeckung überein.

Beweis. Nach obigen Vorbemerkungen verbleibt nur noch $\nu(G) \geq \tau(G)$ zu zeigen. O.B.d.A. können wir annehmen, dass G mindestens eine Kante besitzt.

Zeige: G besitzt einen Knoten u, der von jedem maximalen Matching überdeckt wird.

Sei $e = \{u, v\}$ eine beliebige Kante von G. Wir nehmen nun an, dass es maximale Matchings M und N gibt, so dass M bzw. N den Knoten u bzw. v nicht überdeckt. Sei P die Zusammenhangskomponente von $(V, M \cup N)$, die u enthält. Dann ist P ein Weg mit Endknoten u. Da P (aufgrund der Maximalität von M) nicht M-erweiternd ist, hat P gerade Länge und kann daher nicht v enthalten (da P ansonsten in v enden würde, im Widerspruch zur Bipartheit von G). Daher würde der um die Kante e erweiterte Weg P einen N-erweiternden Weg bilden, im Widerspruch zur Maximalität von N.

Die gezeigte Hilfsaussage impliziert, dass für den induzierte Untergraphen $G' := G \setminus \{u\}$ die Eigenschaft $\nu(G') = \nu(G) - 1$ gilt. Induktiv folgt darüber hinaus, dass G' eine

Knotenüberdeckung C der Größe $\nu(G')$ besitzt. Dann ist $C \cup \{u\}$ eine Knotenüberdeckung von G der Größe $\nu(G') + 1 = \nu(G)$. □

Der Satz von König liefert also insbesondere ein Optimalitätskriterium. Kennt man ein Matching eines bipartiten Graphen und eine Knotenüberdeckung mit gleicher Kardinalität, dann sind sowohl das Matching als auch die Knotenüberdeckung optimal.

Ein Matching in einem Graphen G heißt *perfekt*, wenn er alle Knoten überdeckt.

KOROLLAR 23.3. (Frobenius.) *Ein bipartiter Graph $G = (V, E)$ besitzt genau dann ein perfektes Matching, wenn jede Knotenüberdeckung aus mindestens $\frac{1}{2}|V|$ Knoten besteht.*

BEWEIS. G besitzt genau dann ein perfektes Matching, wenn $\nu(G) = \frac{1}{2}|V|$. Dies ist nach dem Satz von König genau dann der Fall, wenn jede Knotenüberdeckung mindestens die Größe $\frac{1}{2}|V|$ hat. □

Algorithmische Bestimmung maximaler Matchings in bipartiten Graphen. Wir betrachten nun das algorithmische Problem, ein kardinalitätsmaximales Matching in einem bipartiten Graphen zu finden. In Hinblick auf Satz 22.3 genügt es hierzu, erweiternde Wege zu finden. Im bipartiten Fall kann dies durch die Bestimmung eines gerichteten Pfades in einem gerichteten Hilfsgraphen erfolgen.

MATCHING-VERBESSERUNGSALGORITHMUS FÜR BIPARTITE GRAPHEN

Eingabe: ein bipartiter Graph $G = (V, E)$ und ein Matching M,
Ausgabe: ein Matching M' mit $|M'| > |M|$ (sofern ein solches existiert).

Beschreibung des Algorithmus: Sei V die disjunkte Vereinigung $U \cup W$. Ferner sei D_M der gerichtete Graph mit Knotenmenge V, der durch die nachstehende Orientierung jeder Kante $e = \{u, w\}$ aus G hervorgeht (wobei $u \in U$, $w \in W$):

falls $e \in M$, dann orientiere e von w nach u;

falls $e \notin M$, dann orientiere e von u nach w.

Seien U_M und W_M die Menge der Knoten in U bzw. W, die von M nicht überdeckt werden.

Ein M-erweiternder Weg (sofern existent) kann durch Bestimmung eines gerichteten Pfades in D_M von U_M nach W_M ermittelt werden. Dies liefert ein Matching mit einer Kardinalität größer als $|M|$.

Die Korrektheit des Algorithmus folgt unmittelbar aus Satz 22.3 über erweiternde Wege.

SATZ 23.4. *Sei G ein Graph mit n Knoten und m Kanten. Ein kardinalitätsmaximales Matching kann in $O(nm)$ Schritten und arithmetischen Operationen gefunden werden.*

Beweis. Es werden höchstens n Iterationen benötigt, wobei jede (mittels einer Breitensuche im Graphen) höchstens $O(m)$ viele Schritte benötigt. □

Insbesondere kann das Kardinalitäts-Matching-Problem also in Polynomialzeit gelöst werden. Es gibt Algorithmen mit noch besserer Laufzeit als die in Satz 23.4 beschriebene. Die nachfolgende Tabelle gibt einen Überblick über die asymptotische Komplexität existierender Algorithmen (ein $*$ zeigt einen asymptotisch besten Wert in der Tabelle an).

	$O(nm)$	König (1931), Kuhn (1955); siehe oben
	$O(\sqrt{n}m)$	Hopcroft, Karp (1971,1973); Karzanov (1973)
$*$	$\tilde{O}(n^{\omega})$	Ibarra, Moran (1981)
	$O(n^{3/2}\sqrt{\frac{m}{\log n}})$	Alt, Blum, Mehlhorn, Paul (1991)
$*$	$O(\sqrt{n}m\log_n(n^2/m))$	Feder, Motwani (1991, 1995)

Hierbei ist ω eine reelle Zahl, so dass zwei $n\times n$-Matrizen in Zeit $O(n^{\omega})$ arithmetischen Operationen multipliziert werden können (der aktuelle Rekord ist $\omega = 2.376\ldots$). Ferner unterdrückt $\tilde{O}$ zusätzliche logarithmische Faktoren, d.h. $f = \tilde{O}(g)$, falls $f = O(g\log^k g)$ für ein $k \in \mathbb{N}$.

24. Total unimodulare Matrizen

Mit dem Simplex-Algorithmus steht ein praktisch effizienter Algorithmus zur Lösung linearer Optimierungsprobleme zur Verfügung, und es ist bekannt, dass lineare Programme in Polynomialzeit gelöst werden können. Im Falle ganzzahliger linearer Optimierungsprobleme besteht aber das Problem, dass die optimale(n) Ecke(n) der rationalen Relaxation i.a. fraktionale Koordinaten haben. Im folgenden wird die Frage untersucht, wann alle Ecken des Zulässigkeitsbereiches Polyeder ganzzahlig sind.

Definition 24.1. Eine Matrix heißt *total unimodular*, wenn jede Unterdeterminante von A den Wert 0, 1 oder -1 ist.

Insbesondere muss jeder Eintrag einer total unimodularen Matrix 0, 1 oder -1 sein.

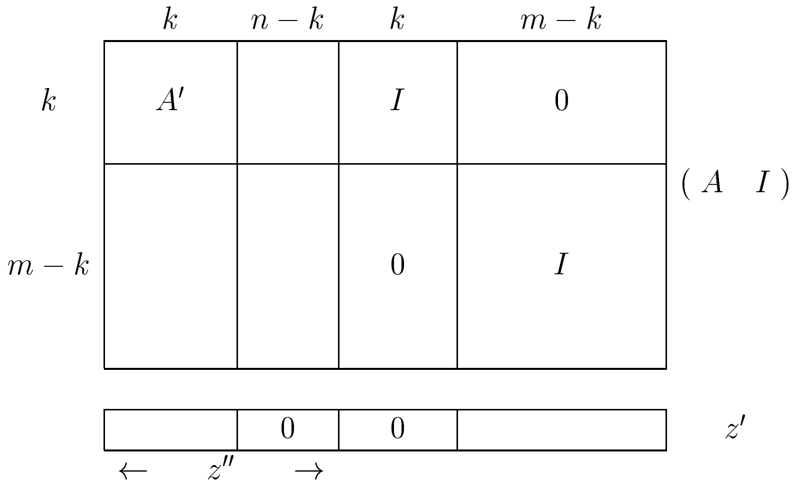
Satz 24.2. *Sei $A \in \mathbb{Z}^{m\times n}$ eine ganzzahlige Matrix mit Rang n. A ist genau dann total unimodular, wenn das Polyeder $\{x \in \mathbb{R}^n : Ax \le b,\ x \ge 0\}$ für jeden ganzzahligen Vektor b ganzzahlig ist (d.h. nur ganzzahlige Ecken hat).*

Beweis. Sei A eine $m \times n$-Matrix und $P := \{x \in \mathbb{R}^n : Ax \le b,\ x \ge 0\}$.
„$\Longleftarrow$": Sei A total unimodular, b ein ganzzahliger Vektor und x eine Ecke von P. x ist Lösung von $A'x = b'$ für ein Teilsystem $A'x \le b'$ von $Ax \le b$, wobei A' eine reguläre $n\times n$-Matrix ist. Da A total unimodular ist, gilt $|\det A'| = 1$, so dass mit der Cramerschen Regel $x = (A^{-1})b'$ ganzzahlig ist.

„$\Longrightarrow$“: Sei $P := \{x \in \mathbb{R}^n \;:\; Ax \leq b, x \geq 0\}$ für jeden Vektor $b \in \mathbb{Z}^m$ ganzzahlig, und sei A' eine reguläre $k \times k$-Untermatrix von A.

Zu zeigen: $|\det A'| = 1$.

O.B.d.A. enthalte A' die Elemente der ersten k Zeilen und Spalten von A. Betrachte die ganzzahlige $m \times m$-Matrix B, die aus den ersten k und den letzten $m-k$ Spalten von $(\, A \quad I \,)$ besteht (siehe Abbildung). Durch Entwicklung nach den letzten $m-k$ Spalten von B ergibt sich unmittelbar $|\det B| = |\det A'|$.



Um $|\det B| = 1$ zu zeigen, zeigen wir, dass B^{-1} ganzzahlig ist. Wegen $\det B \det B^{-1} = 1$ impliziert dies, dass $|\det B| = 1$.

Sei $i \in \{1, \ldots, m\}$. Wir zeigen, dass $B^{-1}e_i$ ganzzahlig ist, wobei e_i der i-te Einheitsvektor ist. Wähle einen ganzzahligen Vektor y, so dass $z := y + B^{-1}e_i \geq 0$. Dann ist $b := Bz = By + e_i$ ganzzahlig. Wir addieren Nullkomponenten zu z, um z' mit

$$(\, A \quad I \,)z' \;=\; Bz = b$$

zu erhalten. Der durch die ersten n Komponenten von z' definierte Vektor z'' gehört zu P. Darüber hinaus sind n linear unabhängige Bedingungen mit Gleichheit erfüllt, nämlich die ersten k und die letzten $n-k$ Ungleichungen von

$$\begin{pmatrix} A \\ -I \end{pmatrix} z'' \;\leq\; \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Also ist z'' eine Ecke von P. Nach Voraussetzung ist z'' ganzzahlig. Dann ist aber auch z' ganzzahlig: seine ersten n Komponenten sind die Komponenten von z', und die letzten m Komponenten sind die Schlupfvariablen $b - Az''$ (und A und b sind ganzzahlig). Also ist auch z ganzzahlig und daher $B^{-1}e_i = z - y$ ganzzahlig. $\square$

Ähnliche Aussagen lassen sich in analoger Weise auch für LP-Probleme in anderer Form herleiten.

Bevor wir zeigen, dass das Bestimmen eines optimalen Kardinalitäts-Matching auf ein lineares Programm mit einer total unimodularen Koeffizientenmatrix zurückgeführt werden kann, benötigen wir einige Hilfsaussagen zur Matrixdarstellung eines Graphen. Die *Inzidenzmatrix* eines Graphen $G = (V, E)$ ist die $V \times E$-Matrix A mit

$$A_{v,e} := \begin{cases} 1 & \text{falls } v \text{ Endkonten von } e, \\ 0 & \text{sonst.} \end{cases}$$

Wir betrachten *Kreise*, d.h. Graphen mit n Knoten und n Kanten, wobei der durch die Kanten definierte Kantenzug einen Kreis der Länge n definiert.

LEMMA 24.3. *Für einen Kreis der Länge n hat die Determinante der Inzidenzmatrix einen Wert*

$$\begin{cases} \pm 2 & \textit{falls } n \textit{ ungerade,} \\ 0 & \textit{falls } n \textit{ gerade.} \end{cases}$$

BEWEIS. Übung. □

SATZ 24.4. *Ein Graph $G = (V, E)$ ist genau dann bipartit, wenn seine Inzidenzmatrix A total unimodular ist.*

BEWEIS. „$\Longleftarrow$“: *Annahme:* A ist total unimodular und G nicht bipartit.

Dann enthält G hat Kreis C ungerader Länge t als (nicht notwendig induzierten) Untergraphen. Die durch die Knoten und Kanten von C induzierte Untermatrix von A ist eine $t \times t$-Matrix mit genau zwei Einsen in jeder Zeile und jeder Spalte. Da t ungerade ist, ist die Determinante dieser Matrix nach der voranstehenden Aussage ± 2, im Widerspruch zur totalen Unimodularität von A.

„$\Longrightarrow$“: Sei G bipartit. Sei B eine $t \times t$-Teilmatrix von A.

Zeige per Induktion nach t: $\det B \in \{-1, 0, 1\}$.

$t = 1$: klar

$t - 1 \to t$: Wir unterscheiden drei Fälle (wobei Fall 1 und Fall 2 nicht disjunkt sind):

Fall 1: B enthält eine Spalte nur mit Nullen. Dann ist $\det B = 0$.

Fall 2: B enthält eine Spalte mit genau einer Eins. In diesem Fall kann B (nach eventueller Umordnung von Zeilen und Spalten) als

$$B = \begin{pmatrix} 1 & b^T \\ \mathbf{0} & B' \end{pmatrix}$$

mit einer Matrix B' und einem Spaltenvektor b geschrieben werden, wobei $\mathbf{0}$ den Vektor aus lauter Nullen in $\mathbb{R}^{t-1}$ bezeichnet. Nach Induktionsannahme gilt $\det B' \in \{0, \pm 1\}$, und es folgt $\det B \in \{0, \pm 1\}$.

Fall 3: Jede Spalte von B enthält genau zwei Einsen. Da G bipartit ist, kann B (nach eventueller Zeilenvertauschung) als

$$B = \begin{pmatrix} B' \\ B'' \end{pmatrix}$$

geschrieben werden, so dass jede Spalte von B' genau eine Eins enthält und jede Spalte von B'' genau eine Eins enthält. Aufaddieren aller Zeilen in B' liefert den Vektor aus lauter Einsen, und ebenso liefert das Aufaddieren aller Zeilen von B' den Vektor aus lauter Einsen. Die Zeilen von B sind daher linear abhängig, und es folgt $\det B = 0$. □

Konsequenzen der totalen Unimodularität. Sei $G = (V, E)$ ein bipartiter Graph mit n Knoten und m Kanten, A seine Inzidenzmatrix. Ferner bezeichne $\mathbf{1}$ den Vektor aus lauter Einsen in $\mathbb{R}^n$. Die Bestimmung eines kardinalitätsmaximalen Matchings in G lässt sich als ganzzahliges lineares Optimierungsproblem

$$\begin{array}{rcl} \max \mathbf{1}^T x & & \\ Ax & \leq & \mathbf{1} \\ x & \geq & 0 \\ x & \in & \mathbb{Z}^m \end{array}$$

formulieren, wobei x der charakteristische Vektor der im Matching enthaltenen Kanten ist. Der Zulässigkeitsbereich P der reellen Relaxation

$$\begin{array}{rcl} \max \mathbf{1}^T x & & \\ Ax & \leq & \mathbf{1} \\ x & \geq & 0 \end{array}$$

dieses Problems ist beschränkt und nach Satz 24.4 ganzzahlig. Folglich kann das kardinalitätsmaximale Matching auch dadurch bestimmt werden, dass dieses lineare Programm (ohne Ganzzahligkeitsbedingungen) gelöst wird. Insbesondere folgt unmittelbar aus dieser Formulierung als lineares Programm, dass das Matching-Problem in Polynomialzeit gelöst werden kann.

In diesem speziellen Fall ist zwar der explizite Algorithmus aus Abschnitt 23 vom praktischen Standpunkt effizienter, die Behandlung als lineares Programm liefert jedoch sehr viele strukturelle Einsichten (siehe Beispiel unten). Ferner ist die Verwendung von Techniken der linearen Programmierung bei aufwendigeren Problemen (z.B. Matching in allgemeinen, d.h. nicht notwendig bipartiten, Graphen) unumgänglich. In den letzten Jahren hat sich die lineare Programmierung darüber hinaus als zentrales Hilfsmittel beim Entwurf und der Analyse von Approximationsalgorithmen entwickelt (siehe z.B. V.V. Vazirani: Approximation Algorithms).

Dualität und der Satz von König. Als exemplarische Anwendung der strukturellen Einsichten der linearen Programmierung leiten wir den Satz von König mittels Dualität erneut her. Sei

$$\begin{array}{rcl} \max \mathbf{1}^T x & & \\ Ax & \leq & \mathbf{1} \\ x & \geq & 0 \end{array}$$

das zu einem Kardinalitäts-Matching in einem bipartiten Graphen zugehörige lineare Programm. Das hierzu duale Problem lautet

$$\begin{array}{rcl} \min \mathbf{1}^T y & & \\ A^T y & \geq & \mathbf{1} \\ y & \geq & 0 \end{array}$$

Da A total unimodular ist, ist auch A^T total unimodular. Ein Optimalpunkt x^* des primalen Programms ist folglich der charakteristische Vektor eines optimalen Matchings, und der Optimalpunkt y^* des dualen Programms ist der charakteristische Vektor einer optimalen Knotenüberdeckung. Nach dem Dualitätssatz stimmen die beiden Optimalwerte x^* und y^* überein, was den Satz von König beweist.

25. Gewichtetes bipartites Matching

Wir betrachten nun Matching-Probleme auf einem bipartiten Graphen, bei dem die Kanten des Graphen mit Gewichten versehen sind. Das gewichtete Matching-Problem (auf beliebigen Graphen) ist wie folgt definiert:

Gewichtetes Matching:

Eingabe: Ein Graph $G = (V, E)$ und Kantengewichte $w_e \in \mathbb{R}$.
Ausgabe: Ein Matching M in G mit maximalem Gewicht $\sum\limits_{e \in M} w_e$.

Der folgende Satz charakterisiert das maximale Gewicht eines Matchings in bipartiten Graphen mittels einer Dualitätsbeziehung.

Satz 25.1. (Egerváry) *Sei $G = (V, E)$ ein bipartiter Graph und $w : E \to \mathbb{R}_+$ eine Gewichtsfunktion auf den Kanten. Dann ist stimmt das maximale Gewicht eines Matchings mit dem minimalen Wert von $\sum_{v \in V} y_v$ überein, wobei $y : V \to \mathbb{R}_+$ eine Funktion mit der Eigenschaft*

$$y_u + y_v \;\geq\; w_e \tag{25.1}$$

für jede Kante $e = \{u, v\}$ ist. Ist w ganzzahlig, dann existiert eine ganzzahlige Funktion $y : V \to \mathbb{Z}$, die den Minimalwert annimmt.

Der Satz kann sowohl explizit bewiesen werden, als auch, wie nachstehend ausgeführt, als Folgerung der Dualitätstheorie hergeleitet werden.

Beweis. Wir betrachten die Verallgemeinerung des in vorhergehenden Abschnitt diskutierten primal-dualen Paars von LPs den Fall des gewichteten bipartiten Matchings. Die zugehörige Dualitätsbeziehung lautet

$$\max\{w^T x \,:\, x \geq 0\,, Ax \leq \mathbf{1}\} \;=\; \min\{\mathbf{1}^T y \,:\, y \geq 0\,, A^T y \geq w\}\,.$$

Aufgrund der totalen Unimodularität der Inzidenzmatrix A wird der Optimalwert des primalen Programms von einem ganzzahligen und damit einem 0-1-Vektor x^* angenommen. $\square$

Zur direkten (d.h. LP-unabhängigen) algorithmischen Bestimmung eines gewichtsmaximalen Matching in einem bipartiten Graphen kann die nachfolgend beschriebene *ungarische Methode* verwendet werden.

Sei $G = (V, E)$ ein bipartiter Graph mit Knotenpartition $V = U \cup W$, und sei $w : E \to \mathbb{R}_+$ eine Gewichtsfunktion. Wir starten mit dem leeren Matching $M = \emptyset$ und geben nun einen Algorithmus an, der zu einem gegebenen Matching M dieses Matching verbessert (sofern möglich).

Sei D_M der aus G hervorgehende gewichtete, gerichtete Graph, bei dem jede Kante $e \in M$ von W nach U orientiert wird, mit Länge $l_e := w_e$, und jede Kante $e \notin M$ von U nach W orientiert wird, mit Länge $l_e := -w_e$. Seien U_M und W_M die Menge der Knoten in U bzw. W, die nicht von M überdeckt werden. Wenn es einen Weg von U_M nach W_M in D_M gibt, bestimme einen kürzesten solchen Weg, genannt P, und definiere M' als das verbesserte Matching $M \Delta E(P)$, wobei Δ die symmetrische Differenz bezeichnet.

Dieser Prozess wird iteriert, bis kein verbessernder Weg von U_M nach W_M in D_M existiert (woraus zunächst folgt, dass M ein *kardinalitäts*maximales Matching ist). Wir zeigen nun, dass das gewichtsmaximale Matching unter all den gefundenen Matchings das größte Gewicht unter allen Matchings hat. Um dies zu sehen, nennen wir ein Matching M *extrem*, wenn es maximales Gewicht unter allen Matchings der Größe $|M|$ hat.

Lemma 25.2. *Jedes gefundene Matching M ist extrem.*

Beweis. Diese Aussage stimmt offensichtlich für $M = \emptyset$. Nehme nun an, dass M extrem ist, und seien P und M' der Weg bzw. das Matching der nächsten Iteration. Sei N ein beliebiges Matching der Größe $|M|+1$. Da $|N| > |M|$, besitzt der Graph$(V, M \cup N)$ eine Zusammenhangskomponente Q, die ein M-erweiternder Weg (im Sinne des Kardinalitäts-Matchings) ist. Weil P ein kürzester M-erweiternder Weg ist, gilt $\sum_{e \in Q} l_e \geq \sum_{e \in P} l_e$. Da $N \Delta Q$ ein Matching der Größe $|M|$ ist und da M extrem ist, folgt $\sum_{e \in N \Delta Q} w_e \leq \sum_{e \in M} w_e$. Daher gilt

$$\sum_{e \in N} w_e \overset{(\text{da } Q \subset N)}{=} \sum_{e \in N \Delta Q} w_e - \sum_{e \in Q} w_e \;\leq\; \sum_{e \in M} w_e - \sum_{e \in P} l_e \;=\; \sum_{e \in M'} w_e\,.$$

$\square$

Es genügt also, in jedem Schritt einen kürzesten Weg von U_M nach W_M zu bestimmen. Zur Bestimmung kürzester Wege in einem Graphen gibt es verschiedene, aus einer Grundvorlesung über theoretische Informatik bekannte Algorithmen (z.B. Dijkstra-Algorithmus, Bellman-Ford-Algorithmus). Voraussetzung für die Anwendung des Dijkstra-Algorithmus ist die Nichtnegativität der Kantengewichte, was hier nicht erfüllt ist. Für die Anwendbarkeit des Bellman-Ford-Algorithmus ist es lediglich erforderlich, dass es keine Kreise C negativer Länge in dem Graphen gibt. Im folgenden zeigen wir, dass diese Voraussetzung erfüllt ist.

Ist M extrem, dann enthält D_M keinen Kreis C negativer Länge (da ansonsten $M\Delta C$ ein Matching der Größe $|M|$ und von größerem Gewicht als M wäre). Mit dem Bellman-Ford-Algorithmus kann daher ein kürzester Weg von U_M nach W_M bestimmt werden (in Laufzeit $O(nm)$), so dass die sukzessive Anwendung des Matching-Verbesserungs-Algorithmus einen $O(n^2m)$-Algorithmus zur Bestimmung eines optimalen gewichteten Matchings liefert.

Gewichtete Matching-Probleme in nichtbipartiten Graphen sowie die Bestimmung gewichtsminimaler perfekter Matchings (d.h. alle Knoten sind am Matching beteiligt) in nichtbipartiten Graphen können ebenfalls in Polynomialzeit gelöst werden (siehe z.B. die angegebene Literatur von Korte/Vygen oder Schrijver). Die zugehörigen Algorithmen beruhen zwar auf den gleichen Ideen wie die hier besprochenen Algorithmen des bipartiten Falls, sind jedoch in ihren technischen Einzelheiten deutlich aufwendiger.

26. Das chinesische Postboten-Problem

Abschließend soll nun ein, mit dem NP-schweren Traveling Salesman Problem, scheinbar eng verwandtes, Problem betrachtet werden, das unter Zuhilfenahme eines polynomialen Algorithmus für das gewichtsminimale perfekte Matching-Problem in Polynomialzeit gelöst werden kann.

Ein Kantenzug $C = (v_0, e_1, v_1, \ldots, e_t, v_t)$ in einem Graphen $G = (V, E)$ heißt eine *chinesische Postboten-Tour*, wenn $v_t = v_0$ und jede Kante von G mindestens einmal in C vorkommt. (Der Name „Chinese Postman Problem“ wurde erstmals von Edmonds (1965) verwendet, nachdem der chinesische Wissenschaftler Guan 1960 das Problem eingeführt hatte).

Gegeben: Ein zusammenhängender Graph $G = (V, E)$ und eine nichtnegative Kantengewichtung l.

Aufgabe: Bestimme eine kürzeste chinesische Postboten-Tour.

Wir betrachten zunächst den Spezialfall der sogenannten Eulerschen Graphen. Hierbei heißt ein Kantenzug P in G *Eulersch*, wenn jede Kante von G genau einmal durchlaufen wird. Ein Graph G heißt *Eulersch*, wenn er einen geschlossenen Eulerschen Kantenzug besitzt. Die folgende, auf Euler zurückgehende Aussage, kann leicht (z.B. durch Induktion über die Anzahl der Knoten) gezeigt werden:

SATZ 26.1. *Ein Graph $G = (V, E)$ ohne isolierte Knoten (d.h. Knoten vom Grad 0) ist genau dann Eulersch, wenn G zusammenhängend ist und jeder Knoten in G geraden Grad hat.*

Hat jeder Knoten von G geraden Grad, dann existiert also eine Euler-Tour, d.h. eine Tour, bei der jede Kante genau einmal durchlaufen wird. Diese ist natürlich eine kürzeste chinesische Postboten-Tour.

Haben nicht alle Knoten von G geraden Grad, dann müssen einige Kanten mehr als einmal durchlaufen werden. Wir werden nun sehen, wie die Knoten T mit ungeradem Grad auf bestmögliche Weise abgehandelt werden.

DEFINITION 26.2. Sei $G = (V, E)$ ein Graph und $T \subset V$. Eine Teilmenge $J \subset E$ heißt *T-Join*, wenn T mit der Menge der Knoten von ungeradem Grad in dem Graphen (V, J) übereinstimmt.

Existiert also ein T-Join, dann ist $|T|$ insbesondere gerade. Ferner ist klar, unter welchen Voraussetzungen überhaupt ein T-Join existiert, nämlich genau dann wenn $|K \cap T|$ für jede Zusammenhangskomponente K von G gerade ist.

T-Joins sind eng mit Matchings verwandt, und ihre Berechnung kann auf die Berechnung eines gewichtsminimalen perfekten Matchings zurückgeführt werden:

SATZ 26.3. *Gegeben sei ein Graph $G = (V, E)$ mit einer nichtnegativen Kantengewichtung und $T \subset V$. Dann kann ein kürzester T-Join in polynomial vielen Schritten und arithmetischen Operationen gefunden werden.*

BEWEIS. Sei K_T der vollständige Graph mit Knotenmenge T. Bestimme für jede Kante $\{s, t\}$ von K_T einen Pfad P_{st} in G minimaler Länge $w(st)$. Bestimme nun ein perfektes Matching $M = \{m_1, \dots, m_r\}$ in K_T, das $\sum_{i=1}^{r} w_i m_i$ minimiert.

Behauptung: Die symmetrische Differenz der Pfade P_{st} für $\{s, t\} \in M$ ist ein kürzester T-Join in G.

Auf elementare Weise sieht man die folgende allgemeiner Darstellbarkeit von T-Joins als Mengen von Pfaden: Jeder T-Join ist die kantendisjunkte Vereinigung von Kreisen sowie von $\frac{1}{2}|T|$ Pfaden, welche disjunkte Knotenpaare in T verbinden. Ferner ist die symmetrische Differenz einer Menge von Kreisen und von $\frac{1}{2}|T|$ Pfaden, welche disjunkte Knotenpaare in T verbinden, ein T-Join. (Hierbei ist die symmetrische Differenz von mehr als zwei Mengen durch $A \Delta B \Delta C := (A \Delta B) \Delta C$ definiert.)

Aus diesen allgemeinen Aussagen folgt, dass die symmetrische Differenz der Pfade P_{st} für $\{s, t\} \in M$ ein *kürzester* T-Join in G ist. $\square$

KOROLLAR 26.4. *Das chinesische Postboten-Problem kann in polynomial vielen Schritten und arithmetischen Operationen gelöst werden.*

Beweis. Sei $T := \{v : \mathrm{grad}_G(v) \text{ ungerade}\}$. Bestimme zunächst einen kürzesten T-Join J. Füge zu jeder Kante e in J eine weitere Kante e' mit den gleichen Endknoten hinzu, so dass der resultierende Graph G' Multikanten besitzen kann. G' ist ein Eulerscher Graph (mit Multikanten).

Behauptung: Jede Euler-Tour in G' liefert eine kürzeste chinesische Postboten-Tour in G (wobei jede neue Kante e' mit der zugehörigen Parallelkante e identifiziert wird).

Jede Euler-Tour liefert eine chinesische Postboten-Tour C der Länge $l(E)+l(J)$, wobei $l(E)$ die Summe der Längen aller Kanten in E bezeichnet. Angenommen es existiert eine kürzere Tour C', dann definiere J' als die Menge der Kanten, die eine gerade Anzahl oft in C' durchlaufen werden. Dann ist J' ein T-Join und daher gilt $l(J') \geq l(J)$. Es folgt

$$l(C') \ \geq \ l(E) + l(J') \ \geq \ l(E) + l(J) \ = \ l(C)\,,$$

im Widerspruch zu $l(C') < l(C)$. □

Exkurs als Anhang: NP-Vollständigkeit (Rupert Hartung)

A.1 Effizienz von Algorithmen.

Beispiel 26.1.

a) Betrachte das folgende algorithmische Problem.

> Integer Programming (Ip). Gegeben ein Ungleichungssystem $A\,x \geq d$ und $B\,x \leq b$ (d.h. $n \in \mathbb{N}$, $A \in \mathbb{Z}^{k\times n}$, $B \in \mathbb{Z}^{m\times n}$, $d \in \mathbb{Z}^k$, $b \in \mathbb{Z}^m$). Gibt es ein $x \in \mathbb{Z}^n$, das alle Ungleichungen erfüllt?

Ausgeschrieben heißen die Systeme einfach

$$\begin{array}{rcl} a_{11}x_1 + \ldots + a_{1n}x_n & \geq & d_1 \\ \vdots & & \vdots \\ a_{k1}x_1 + \ldots + a_{kn}x_n & \geq & d_k \\ b_{11}x_1 + \ldots + a_{1n}x_n & \leq & b_1 \\ \vdots & & \vdots \\ b_{m1}x_1 + \ldots + a_{mn}x_n & \leq & b_m \end{array}$$

Vom analogen Problem mit rationalen Zahlen haben wir bereits effiziente Algorithmen gesehen (oder werden sie noch sehen; z.B. Simplex-Algorithmus). Für Ip ist jedoch kein besserer Algorithmus bekannt als, etwas untertrieben, einigermaßen ‚schlaues' Ausprobieren (in diesen Zusammenhang gehören Verfahren wie *branch and bound, backtracking, dynamische Programmierung*). Ganz naives Durchsuchen würde *ungefähr* $\mathcal{O}\left((\|b\|_\infty)^n\right)$ Operationen erfordern (wenn B kleine Einträge hat).

b) Ein vermeintlich völlig anderes Problem ist es, die Konsistenz eines teilweise aufgedeckten Minesweeper-Feldes zu prüfen.

> MINESWEEPER. Gegeben $m, n, K \in \mathbb{N}$ und eine Matrix $F \in \{-1, 0, \ldots, 8\}^{m \times n}$. Gibt es eine Verteilung von K Minen auf ein $(m \times n)$-Brett, so dass das Feld (i, j) genau an F_{ij} der Minen anstößt (falls $F_{ij} \geq 0$)?

Dabei steht $F_{ij} = -1$ natürlich dafür, dass das Feld (i, j) noch nicht aufgedeckt wurde.- Wir werden später sehen, dass diese beiden Probleme sehr viel gemein haben.

Für beide gerade beschriebenen Probleme sind keine Algorithmen bekannt, die man als ‚effizient' bezeichnen würde. Was aber ist Effizienz? Es bedarf hier einer Terminologie, die uns erlaubt, die anschauliche Unterscheidung zwischen effizienten und ineffizienten Algorithmen zu formalisieren.

DEFINITION 26.2.

a) *Ein Algorithmus $\mathfrak{A}$ läuft in polynomieller Zeit genau dann, wenn es eine Konstante $k \geq 1$ gibt, so dass $\mathfrak{A}$ zu jeder Eingabe x in $\mathcal{O}(|x|^k)$ Schritten terminiert.*

Alternativ: *… wenn es Konstanten $k \geq 1$, $C > 0$ gibt, so dass $\mathfrak{A}$ zu jeder Eingabe x in höchstens $C\,|x|^k$ Schritten terminiert.*

b) *$\mathfrak{A}$ läuft in exponentieller Zeit genau dann, wenn gilt:*

Es gibt Konstanten $k, C, d > 0$ und zulässige Eingaben $(x_i)_i$ beliebiger Länge (d. h. $\lim_{i\to\infty} |x_i| = \infty$), so dass $\mathfrak{A}$ zur Eingabe x_i mindestens $C\,2^{d\,|x|^k}$ Schritte ausführt, bevor er terminiert.

Die Polynomialzeit-Algorithmen *wollen wir als die* effizienten *Algorithmen betrachten.*

Bemerkung: Globale Konstanten (wie hier C) bleiben in solchen Zusammenhängen stets unberücksichtigt, da sie schon durch Übergang zu anderen Zeiteinheiten oder Operationsbegriffen entstehen.

Wie kann man eine solche Festlegung rechtfertigen?

- Zunächst einmal bezieht sie sich auf die Zusatzkosten eines Algorithmus', der bei *längeren Eingaben* (etwa größeren Systemen mit größeren Zahlen bei IP) entsteht. Hier fällt auf, dass bei Polynomen die Laufzeit-Verschlechterung beherrschbar bleibt, im Gegensatz zu den exponentiellen Funktionen. Das bedeutet, Polynomialzeit-Algorithmen sind (i.d.R.) in einem viel größeren Bereich von Instanzen eines Problems praktikabel.
- Andererseits soll unser Begriff nicht von der gegewärtigen Rechnertechnologie abhängen. Fragen wir also nach dem *Kapazitätsgewinn*, die uns schnellere Maschinen für die jeweiligen Algorithmen bescheren, so fällt ein noch deutlicherer Vorteil der Polynomialzeit-Algorithmen auf.

  Beispielrechnung: Eine Operation möge genau 1 μs erfordern (d. h. unser Rechner hat eine Frequenz von 1 MHz). Dann können wir in einer Stunde 3,6 Milliarden Operationen durchführen. Betrachte nun einen Algorithmus mit Laufzeitschranke

  $$C\,n^k$$

mit $C, k > 0$, wobei n die Eingabelänge sei. Dann umfasst die längste Eingabe, die dieser Algorithmus in einer Stunde sicher lösen kann, genau $\lfloor t_0 \rfloor$ Bits (Symbole), wobei

$$t_0 := \sqrt[k]{\frac{1}{C}\, 3,6 \cdot 10^9}\,.$$

Auf einem moderneren Rechner mit 100-facher Geschwindigkeit kann der selbe Algorithmus

$$\left\lfloor \sqrt[k]{\frac{1}{C}\, 3,6 \cdot 10^11} \right\rfloor = \left\lfloor \sqrt[k]{100}\, t_0 \right\rfloor$$

Eingabebits, also etwa das $10^{2/k}$-fache der vorherigen Kapazität, bearbeiten.

Wenden wir uns nun exponentiellen Algorithmen zu. Hätte der Algorithmus eine Laufzeitschranke von L^n Zeiteinheiten (Mikrosekunden), so bewältigte er in einer Stunde immerhin $\lfloor t_1 \rfloor$ Eingabebits mit

$$t_1 := \log_L \left(3,6 \cdot 10^9\right) = (\log L)\Big(\log 3,6 + 9 \,\log 10\Big)\Big(\approx 31,74 \log L\Big)$$

Hoffen wir auf eine wesentliche Verbesserung durch 100-fache Geschwindigkeit, werden wir enttäuscht: Ein solcher Fortschritt verminderte die Laufzeit nur auf

$$\Big\lfloor (\log L)\Big(\log 3,6 + 11 \,\log 10\Big)\Big\rfloor \approx t_1 + 2\,(\log L)\,(\log 10)$$

Man beachte: Im ersten Fall vervielfachte sich die Kapazität, nun hingegen kommt nur ein konstanter Summand dazu!

Bemerkung: Zwischen polynomiellen und exponentiellen Algorithmen gibt es noch solche, deren Laufzeit man als *subexponentiell* bezeichnet. Zu den subexponentiellen Funktionen in n gehören etwa $n^{\log n}$, $2^{\sqrt{n \log n}}$. Bei Problemen, zu denen man keine effizienten Algorithmen mehr zu finden hofft, ist ein subexponentieller Algorithmus eine große Verbesserung, so etwa beim Faktorisieren (*Quadratisches Sieb, Zahlkörpersieb*).

Natürlich nehmen auch die Funktionen

$$1000000000000000000 \cdot n, \qquad n^{1000000}$$

die sicher polynomiell sind, in realistischen Größenordnungen von n zu hohe Werte an. Dennoch hat sich gezeigt, dass in praktischen Problemen solche Fälle kaum auftreten.

Daher kann man sagen, dass sich dieser recht einfache theoretische Begriff von Effizienz in der Praxis bewährt hat.

A.2 Wie schwer ist ein Problem? Ein schlechter Algorithmus zeigt nicht unbedingt, dass das betreffende Problem schwer ist, d. h. dass kein effizienter Algorithmus dafür existieren kann; das sieht man schon bei der Multiplikation von ganzen Zahlen im Vergleich der Schulmethode mit der naiven Iteration der Addition. Wir interessieren uns daher vornehmlich für die *beste* Laufzeit, die ein Algorithmus zum jeweiligen Problem haben kann. Das ist ein sinvolles Maß für die Schwierigkeit eines Problems.

DEFINITION 26.3. *Die worst-case-Laufzeit des schnellsten Algorithmus', der ein gegebenes Poblem löst, heißt die* Komplexität *des Problems.*

Es gibt im wesentlichen zwei Gründe für Überlegungen zur Komplexität von Problemen:

- Für die Kryptographie ist es essentiell, dass ein Angreifer auf ein Kryptosystem sich mit Problemen hoher Komplexität konfrontiert sieht (z.B. FAKTORISIEREN).
- Für alle anderen Anwendungen sucht man möglichst effiziente Algorithmen. Falls aber keine solchen bekannt sind, kann die Komplexität eines Problems Auskunft geben, ob noch Spielraum für Verbesserungen besteht oder die bisherigen Algorithmen eben schlecht, aber optimal laufen.

Bemerkungen:

a) Der Nachteil dieser Definition ist, dass man die Komplexität der meisten Probleme nicht kennt: Ein einziger Algorithmus mit bekannter Laufzeitschranke liefert eine *obere* Schranke für die Komplexität des Problems. Wenn kein schnellerer Algorihmus bekannt ist, muss das nicht bedeuten, dass die Problemkomplexität tatsächlich so groß ist.

b) Andererseits gibt es einige wenige Probleme, zu denen man bewiesene untere Schranken kennt, was die obige Definition tatsächlich mit Inhalt füllt. Etwa beim Sortieren stimmt die theoretische *untere* Komplexitätschranke (nämlich $\Omega(n \log n)$) mit einer *oberen* Laufzeitschranke (nämlich $\mathcal{O}(n \log n)$) für *mergesort* (und *heapsort*) überein. Das heißt, in Fällen wie diesem kennt man die Komplexität eines Problems genau.

c) Wie wir daraus entnehmen können, lässt nicht jedes Problem einen beliebig schnellen Algorithmus zu. Abgesehen von Einzelergebnissen wie für das Sortieren gibt es auch eine allgemeine theoretische Aussage (den *Hierarchiesatz*), die unter anderem sichert, dass es für alle $1 \leq k < k'$ ein algorithmisches Problem gibt, dass man in Zeit $\mathcal{O}(n^{k'})$ lösen kann, aber nicht in Zeit $\mathcal{O}(n^k)$. Ebenso gibt es unendlich viele verschiedene exponentielle und subexponentielle Funktionen, die wirklich als Komplexitäten auftreten. Allerdings erlauben solche Sätze kaum Aussagen über konkrete Probleme.

Da wir also diese Definition selten verwenden können, gehen wir zu einem leichter zu handhabendem Begriff über, der nur obere Schranken verwendet. Dazu benötigen wir aber die folgende Einschränkung. Wir behandeln nun nicht mehr allgemeine algorithmische (Berechnungs-)Probleme, sondern nunmehr nur noch *Entscheidungsprobleme*. Ein Entscheidungsproblem ist ein algorithmisches Problem, bei dem wir als Antwort nur „Ja“ oder „Nein“ erwarten (bzw. eine gewisse Codierung dieser Antworten). Zum Beispiel waren beide Probleme zu Beginn dieser Vorlesung als Enscheidungsprobleme formuliert.

Es ist wichtig zu bemerken, dass diese Einschränkung meist keinen großen Verlust bedeutet. Betrachte als Beispiel das Faktorisieren in den folgenden beiden Gestalten.

> FAKTORISIERUNG (Berechnungsproblem). Gegeben eine Zahl $N \in \mathbb{N}$, berechne die Primfaktorzerlegung von N.

Das zugehörige Entscheidungsproblem lautet:

> FAKTORISIERUNG. Gegeben $a, b, N \in \mathbb{N}$ mit $a < b < N$. Entscheide, ob N einen Primteiler im Intervall $[a, b]$ hat.

Wir wollen uns nun davon überzeugen, dass diese zwei Probleme „gleich schwer“ sind. Was das bedeutet, werden wir gleich sehen.

> Nehmen wir zunächst an, wir könnten das Berechungsproblem mit einem Algorithmus $\mathfrak{A}$ „schnell“ lösen. Sollen wir nun entscheiden, ob N einen Primteiler in $[a, b]$ hat, so faktorisieren wir N zunächst mit dem angenommenen Algorithmus. Dann müssen wir nur dessen Ausgabe, nämlich die Primfaktoren p, durchlaufen und die zwei Ungleichungen $p \geq a$ und $p \leq b$ prüfen. Das bedeutet: Mit einem Aufruf von $\mathfrak{A}$ und linearem Mehraufwand kann das Entscheidungsproblem gelöst werden.
>
> Nimm nun einen Algorithmus $\mathfrak{E}$ an, der das Entscheidungsproblem zum Faktorisieren löst. Sei eine Zahl N gegeben. Frage nun $\mathfrak{E}$ an, ob N einen Faktor $1 \leq p \leq \lceil\sqrt{N}\rceil$ hat. Falls nicht, so ist N schon Primzahl(!). Falls doch, bringe mittels $\mathfrak{E}$ in Erfahrung, ob N einen Faktor $1 \leq p \leq \lceil\frac{1}{2}\sqrt{N}\rceil$ besitzt oder nicht. Falls nicht, so hat N gewiss einen Primfaktor $\lceil\frac{1}{2}\sqrt{N}\rceil + 1 \leq p \leq \lceil\sqrt{N}\rceil$. Fahre induktiv so fort: Wenn N einen Faktor $a \leq p \leq b$ hat, frage als nächstes, ob auch $a \leq p \leq \lfloor\frac{1}{2}b\rfloor$ gilt. Falls N *keinen* Primfaktor $a \leq p \leq b$ aufweist, dann besitzt es jedoch einen mit $b + 1 \leq p \leq 2\,b + a$ (Induktion!), und die nächste Anfrage an $\mathfrak{E}$ lautet auf $b + 1 \leq p \leq b + \lfloor\frac{1}{2}b - a\rfloor$.
>
> Es ist klar, dass sich mit diesem Verfahren die Intervalle, in denen wir einen Primfaktor von N sicher verorten können, in jedem Schritt (bis auf konstante Abweichungen) halbieren. Folglich wird er nur noch eine Zahl p_1 enthalten, und wir können diese Methode rekursiv auf $N_1 := \dfrac{N}{p_1}$ anwenden. Schließlich bilden die N_i eine streng fallende Folge, so dass irgendwann $N_i = 1$ sein wird. An dieser Stelle liegt dann eine vollständige Zerlegung von N vor.
>
> Schätze nun den Aufwand dieses Verfahrens ab. Wenn wir durch Intervallschachtelung (*divide and conquer*) einen Primfaktor von N_i suchen, werden wir einen solchen nach $\mathcal{O}(\log N_i)$ Aufrufen von $\mathfrak{E}$ gefunden haben, da sich das Suchintervall bei jedem Schritte halbiert. Ferner kann N höchstens $\log N$

Primfaktoren haben, und $N_i \leq \frac{N}{2^i}$. Folglich erreichen wir die Lösung des Berechnungsproblems mittels höchstens

$$\begin{aligned}\sum_{i=0}^{\lfloor \log N \rfloor} \mathcal{O}(\log \frac{N}{2^i}) &\leq \mathcal{O}(N \log N) + -\mathcal{O}(\sum_i 2^i) = \mathcal{O}(N \log N) - \mathcal{O}(N) \\ &= \mathcal{O}(N \log N)\end{aligned}$$

Aufrufen von $\mathfrak{E}$ (und höchstens ebensovielen sonstigen Operationen). Beachte, dass $\ldots = \mathcal{O}(\ldots)$ immer eine Abschätzung nach oben beinhaltet.

Die hier verwendete Technik, einen hypothetischen Algorithmus als Subroutine zu verwenden, wird später noch wichtiger werden. Man spricht auch von einem *Orakel* für das betreffende Problem.

Im Sinne der Effizienz geht also bei der Beschränkung auf Entscheidungsprobleme keine Information verloren. Man beachte jedoch, dass die Auswahl eines „äquivalenten" Entscheidungsproblem (wie hier) nicht immer ganz naheliegend sein muss. So ist etwa das folgende Entscheidungsproblem effizient lösbar (und daher das falsche Äquivalent zum Faktorisieren):

PRIMZAHLEN. Gegeben $N \in \mathbb{N}$. Entscheide, ob N eine Primzahl ist.

Dass man hier Entscheidungsprobleme betrachtet, hat zum Zweck, eine gängige Sichtweise der theoretischen Informatik anbringen zu können. Man identifiziert nämlich die (Codierung der) *Ja*-Instanzen eines Entscheidungsproblems als die Wörter einer *Sprache* $L \subseteq \Sigma^*$, wobei Σ eine Menge von Symbolen, das *Alphabet*, ist (z.B. $\Sigma = \{0, 1\}$). Anstatt „Problem entscheiden" sagt man dann „Sprache erkennen" (oder entscheiden). So ist etwa die zum Problem PRIMZAHLEN gehörige Sprache die Menge aller Primzahlen in Binär- (oder Dezimal- oder ...) -schreibweise. Wir werden Sprachen hier jedoch nur als Konkretion des Begriffs ‚Problem' benützen und insbesondere keine Sätze aus der Theorie der formalen Sprachen aufgreifen.

DEFINITION 26.4. *Es sei $\boldsymbol{T}(f(n))$ die Menge aller Sprachen L (ugs.: aller Probleme), für die es einen Algorithmus gibt, der L entscheidet und der zur Eingabe x nach $\mathcal{O}(f(|x|))$ (Bit-)Operationen („in Zeit $\mathcal{O}(f(|x|))$") terminiert.*

Wichtig: Die Elemente von $\mathbf{T}(f(n))$ sind Probleme, nicht Algorithmen!

Natürlich ist $\mathbf{T}(f(n)) \subseteq \mathbf{T}(g(n))$, wann immer $f(n) \leq g(n)$ für große n. Wie wir bereits oben gesehen haben, sind „viele" dieser Inkusionen echt.

Die für uns interessanteste solcher Klassen bilden genau die effizient lösbaren Probleme.

DEFINITION 26.5. *Bezeichne mit $\boldsymbol{P}$ die Menge aller Sprachen, die in polynomieller Zeit entscheidbar sind:*

$$\boldsymbol{P} = \bigcup_{k \in \mathbb{N}} \boldsymbol{T}(n^k)$$

Beobachtungen:

- Irgend ein Polynomialzeit-Algorithmus für L belegt, dass $L \in \mathbf{P}$.
- Wie oben erwähnt: Es gibt (unendlich viele verschiedene Levels von) entscheidbare(n) Problemen außerhalb $\mathbf{P}$.
- Es gibt konkrete, entscheidbare Probleme, die beweisbar nicht in $\mathbf{P}$ liegen z.B.

  Ideal Membership: gegeben $g, f_1, \ldots, f_m \in \mathbb{Q}[x_1, \ldots, x_r]$, entscheide, ob $g \in (f_1, \ldots, f_m)$).

  Ein einleuchtendes Beispiel für die Ursache dafür ist vielleicht die Aufgabe, zu einer Zahl n die Fakultät $n!$ zu berechnen (kein Entscheidungsproblem). Hierfür kann es schlichtweg keinen effizienten Algorithmus geben, da schon die Anzahl der Ausgabeziffern, die auf ein Ausgabemedium geschrieben werden müssen, exponentiell in n wächst (wieso?).

  Obgleich Ideal Membership ein Entscheidungsproblem ist, sind die Gründe ähnlich: Etwas verkürzt gesagt kann man zeigen, dass ein Lösungsalgorithmus exponentiell große Zwischenergebnisse produzieren muss, um das Problem lösen zu können.
- Am wichtigsten ist aber, dass es viele (bedeutende) Probleme gibt, für die man weder Zugehörigkeit zu $\mathbf{P}$ noch zum Komplement $\complement\mathbf{P}$ beweisen kann. Dazu gehört etwa das Faktorisieren. Eine weitere wichtige Klasse solcher Probleme werden wir jetzt betrachten.

A.3 Probleme vergleichen: die Reduktion. Das folgende Problem hat eine gewisse Berühmtheit erlangt.

Satisfiability (SAT): Gegeben eine aussagenlogische *Formel* Φ *in konjunktiver Normalform*, d.h.

$$\Phi = c_1 \wedge \ldots \wedge c_r$$

mit *Klauseln*

$$c_i = (y_{i1} \vee \ldots \vee y_{ik_i})$$

wobei wiederum $y_{ij} \in \{x_1, \ldots, x_s, \bar{x}_1, \ldots, \bar{x}_s\}$. Das heißt, es gibt eine Anzahl von Variablen x_l, die die Werte W :=*wahr*, F :=*falsch* annehmen können; und die *Literale* y_{ij}, die in Φ auftreten, sind solche Variablen oder deren Verneinungen (mit einem Querstrich angedeutet).-

Entscheide nun, ob die Formel Φ durch eine Belegung der Variablen erfüllbar ist, d.h. ob ein Tupel $x \in \{W, F\}^s$ existiert, so dass

$$\Phi(x) = W\,.$$

Dieses Problem ist aufgrund seiner großen Allgemeinheit noch etwas unübersichtlich. Betrachte daher auch die folgende Vereinfachung:

3-Satisfiability (3SAT): Gegeben eine Formel Φ wie oben, nur mit Klauseln der Form

$$c_i = (y_{i1} \vee y_{i2} \vee y_{i3})$$

also der Länge genau drei. Entscheide, ob Φ erfüllbar ist.

In einem gewissen Sinne hat diese Vereinfachung das Problem nicht einfacher gemacht. Was das genau bedeutet, werden wir uns später überlegen.

Zunächst einmal fragen wir uns, ob wir beide Probleme unabhängig voneinander lösen müssen, oder ob ein Algorithmus für den einen für die Lösung des anderen Problems Nutzen bringt. In der einen Richtung ist das Verhältnis klar: 3SAT ist offensichtlich ein Teilproblem von SAT. Das heißt, ein Algorithmus für SAT löst auch 3SAT, ohne dass man Eingabe oder Algorithmus irgendwie modifizieren müsste.

Wie aber ist das umgekehrte Verhältnis gelagert? Können wir eine Lösung von SAT auf eine von 3SAT zurückführen, das heißt, genügt es, 3SAT zu betrachten, um beide in den Griff zu bekommen?

Um zu verstehen, dass die Antwort hierauf „Ja“ ist, führen wir folgende Konstruktion durch: Wir geben an, wie man jede Formel in konjunktiver Normalform in eine eben solche umformen kann, deren Klauseln sämtlich aus genau drei Literalen bestehen, die genau dann erfüllbar ist, wenn die ursprüngliche Formel es war. Die Bedeutung dieses Verfahrens ist die folgende: Angenommen, wir hätten einen Algorithmus zur Lösung von 3SAT, den wir als den besten ansehen, den wir erreichen können. Wenn uns nun eine SAT-Instanz vorgelegt wird, so können wir auf die gerade angekündigte Art und Weise die gegebenene Formel in eine Formel mit 3-Klauseln umwandeln und hierauf den 3SAT-Algorithmus anwenden. Dessen Ausgabe wird dann die Antwort auf die ursprüngliche Frage sein. Ferner hängt der Aufwand dieser Methode im wesentlichen nur von dem benützten Algorithmus für das 3SAT-Problem ab, sofern die Umformung schnell und platzsparend möglich ist (dazu auch s.u.).

Konstruktion: Gegeben die Formel

$$\Phi = c_1 \wedge \ldots \wedge c_r\,.$$

Wir behandeln die Klauseln einzeln (lokal). Dabei ist keine Veränderung nötig, falls eine Klausel schon genau drei Literale enthält. Sei also c eine Klausel mit nicht genau 3 Literalen. Wir unterscheiden drei Fälle:

- 1 Literal: Ersetze $c = (y)$ durch $(y \vee y \vee y)$
- 2 Literale: Ersetze $c = (y_1 \vee y_2)$ durch $(y_1 \vee y_2 \vee y_2)$
- ≥ 3 Literale: Ersetze $c = (y_1 \vee y_2 \vee \ldots \vee y_k)$ durch

  $$(y_1 \vee y_2 \vee z_1) \wedge (\bar{z}_1 \vee y_3 \vee z_2) \wedge (\bar{z}_2 \vee y_4 \vee z_3) \wedge \ldots \wedge (\bar{z}_{k-3} \vee y_{k-1} \vee y_k)\,,$$

  wobei $z_1, \ldots, z_{k-3}$ neu einzuführende Variablen sind.

Man überzeuge sich davon, dass die Erfüllbarkeit der Formel sich durch keine dieser lokalen Modifikationen ändert.

Wie oben erwähnt, ist es nun sinnvoll, den entstandenen Zusatzaufwand abzuschätzen. Dies soll hier nur ganz grob geschehen.

Es ist klar, dass der oben angegebene Prozess algorithmisch sehr effizient umzusetzen ist: Die Fallunterscheidung geschieht durch bloßes Zählen der Literale von Klammer zu Klammer, und der Aufwand zur Änderung der Klauseln besteht im wesentlichen im Ausschreiben des Resultats.

Nun wird aber die Formel durch unseren Eingriff länger. Man bedenke, dass die Laufzeit des 3SAT-Algorithmus', der im Anschluss aufgerufen werden soll, von der Eingabelänge abhängt. Daher müssen wir hier ausschließen, dass die Formel zu sehr anwächst.

Als Maß für die Länge der Formel betrachten wir die Anzahl der auftretenden Literale $|\Phi|$ (man überlege, welche Abschätungen zwischen $|\Phi|$ und der Bitlänge der Formel bestehen!). Im ersten Falle (s.o.) verdreifacht sich diese Zahl, im zweiten ver-$\frac{3}{2}$-fach sie sich. Im dritten Falle steigt sie (mit der obigen Bezeichnung) von k auf $2k-3$, was weniger als Verdopplung bedeutet. Bezeichnet Φ' die neu konstruierte Formel, so wächst deren Länge nur um einen konstanten Faktor:

$$|\Phi'| \quad < \quad 3\,|\Phi|$$

Das bedeutet insbesondere, dass jede obere Komplexitätsschranke für 3SAT bis auf Konstanten auch für SAT gilt.

Wir formalisieren nun das gerade im Konkreten beobachtete Verhältnis von Problemen.

Definition 26.6. *Seien L_1, L_2 Sprachen. Eine* Karp-Reduktion *(oder* many-one-Reduktion*) von L_1 nach L_2 ist ein polynomieller Algorithmus $\mathfrak{A}$ mit $\mathfrak{A}(L_1) = L_2$ und $\mathfrak{A}(\Sigma^* \setminus L_1) = \Sigma^* \setminus L_2$.*

Wenn eine Karp-Reduktion existiert, so schreibe $L_1 \preceq_K L_2$ (in der Literatur auch: $L_1 \propto_K L_2$).

Man beachte, dass die Ausgabe von $\mathfrak{A}$ in diesem Falle polynomiell in seiner Ausgabe beschränkt sein muss, da man in polynomieller Zeit nur polynomiell viele Zeichen auf ein Ausgabemedium schreiben kann. Es ist klar, dass nach dieser Definition das oben angegebene Verfahren eine Karp-Reduktion bildet. Der Begriff ‚Reduktion' leitet sich genau von der oben motivierten Überlegung ab: Wenn $L_1 \preceq_K L_2$, dann kann man L_1 auf L_2 zurückführen, das heißt, es genügt, L_2 zu lösen, wenn man an Lösungen von L_1 interessiert ist.

Anschaulich bedeutet $L_1 \preceq_K L_2$: Bis auf polynomiellen Zeitverlust ist L_1 höchstens so schwer wie L_2. Alternative Formulierung: Bis auf einen polynomiellen Summanden ist die Komplexität von L_1 nicht größer als die von L_2.

Möchte man den Begriff der Sprache vermeiden und lieber über Entscheidungsprobleme sprechen, so formuliert man die beiden Bedingungen an $\mathfrak{A}$ aus der Definition am besten so: „Eine Ja-Instanz von L_1 verarbeitet $\mathfrak{A}$ zu einer Ja-Instanz von L_2, Eine Nein-Instanz von L_1 hingegen zu einer Nein-Instanz von L_2“.

Nun war in unserem Beispiel aber ein gegenseitiges Verhältnis vor: Eine Karp-Reduktion von SAT nach 3SAT wurde konstruiert, während die Umkehrung, $3SAT \preceq_K SAT$, trivial ist (formal: sie wird durch den leeren Algorithmus (der die identische Funktion berechnet) geleistet).

Definition 26.7. *Nenne L_1, L_2* äquivalent *(unter Karp-Reduktionen), in Zeichen $L_1 \sim_K L_2$, falls $L_1 \preceq_K L_2$ und $L_2 \preceq_K L_1$.*

Interessantes Beispiel für eine Karp-Reduktion: SAT $\preceq_K$ Minesweeper. Zum Beweis konstruiert man logische Schaltkreise auf dem Minesweeperfeld, die 1-Bit-Informationen (also eine Variablenbelegungen bei SAT) miteinander verknüpfen. Eine solche Variable wird duch zwei aneinander grenzende Felder dargestellt, von denen genau eines eine Mine verbirgt. Das bewirkt, dass die ursprüngliche SAT-Formel genau dann erfüllbar ist, wenn es eine zulässige Belegung des Feldes mit Minen gibt. Dabei muss (laut Definition der Karp-Reduktion) darauf geachtet werden, dass die Größe des damit erzeugten Feldes polynomiell in der Länge der ursprünglichen Formel ist. Für Details siehe [4].

Erinnern wir uns, welche Idee wir einfangen wollten. Im wesentlichen ging es um die Formalisierung der Frage: Könnten wir L_1 lösen, wenn wir eine Lösung zu L_2 schon gefunden hätten?

Die Existenz einer Karp-Reduktion gibt uns darauf eine bejahende Antwort. Eine solche kann aber auch von ganz anderen Reduktionen herrühren.

Dazu stellen Sie sich die Situation vor, Sie sollte einen Algorithmus zur Lösung von L_1 programmieren. Das Problem L_1 ist Ihnen nicht sehr eingängig, und Sie können trotz intensiver Suche keinen Algorithmus ersinnen, der das Problem mit der benötigten Schnelligkeit löst. Zufällig ist Ihnen aber bekannt, dass bereits ein sehr ausgefeiltes, zeitsparendes Programm (nennen wir es $\mathfrak{S}$) für das Problem L_2 schon in Ihrer Programm- (oder Projekt-) bibliothek existiert. Wie könnte man das nun ausnutzen?

Ein naheliegender Gedanke ist hier sicherlich, ein Programm zu entwerfen, das $\mathfrak{S}$ als Subroutine benützt. Falls es gelingt, solch ein Programm zu schreiben, dass genau dann die richtige Ausgabe für L_1 erzeugt, wenn die $\mathfrak{S}$ das für L_2 tut, dann ist Ihre Aufgabe beendet, ohne dass Sie die Funktionsweise von $\mathfrak{S}$ verstehen müssten (geschweige denn einen Algorithmus angeben könnten, der L_1 ohne vordefinierte Funktionen löst).

Ferner wird dieser zusammengesetzte Algorithmus effizient sein, wenn $\mathfrak{S}$ es ist, wenn $\mathfrak{S}$ nur polynomiell oft aufgerufen wird und wenn auch sonst nur polynomiell viele Einzeloperationen durchgeführt werden.- Diese allgemeinere Möglichkeit der Reduktion fassen wir in den folgenden Begriff.

Definition 26.8. *Seien L_1, L_2 Sprachen. Sei ferner $\mathfrak{A}$ ein Algorithmus, der zusätzlich zu den üblichen Bitoperationen einen (angenommenen) Algorithmus $\mathfrak{S}$, der L_2 entscheidet, benutzen darf. $\mathfrak{A}$ heißt* Cook- *(oder* Turing-*)* Reduktion *von L_1 nach L_2, wenn er ...*

- *$\mathfrak{S}$ nur polynomiell oft aufruft,*
- *nur polynomiell viele Bitoperationen ausführt, und*
- *L_1 entscheidet (vorausgesetzt, dass $\mathfrak{S}$ das Problem L_2 korrekt entscheidet).*

Wir schreiben $L_1 \preceq_T L_2$. Analog zu $\sim_K$ definieren wir die Turing-Äquivalenz *$L_1 \sim_T L_2$ als $L_1 \preceq_T L_2$ und $L_2 \preceq_T L_1$.*

Man beachte, dass $\mathfrak{S}$ tatsächlich rein hypothetischen Charakter hat. Wie oben bereits erwähnt, nennt man $\mathfrak{S}$ daher auch ein *Orakel* für L_2.

Beispiel 26.9. *Erinnerung an die Grundlagen zu RSA: Es ist bekannt, dass die Kenntnis der Faktorisierung eines RSA-Moduls $N = pq$ ‚gleichwertig' ist mit der Kenntnis von $\lambda(N) = (p-1)(q-1)$ (s.). Hinter dem Worte ‚gleichwertig' steckt nichts anderes als eine Turing-Reduktion. Dazu ist das Berechnungsproblem für $\lambda(N)$ in ähnlicher Weise in ein Entscheidungsproblem zu verkleiden, wie wir es für das Faktorisieren getan haben.*

Es ist natürlich klar, dass

$$L_1 \preceq_K L_2 \quad \Longrightarrow \quad L_1 \preceq_T L_2$$

Denn eine Karp-Reduktion entpricht derjenigen Cook-Reduktion, die zuerst die Eingabe in Polynomialzeit umrechnet, um zum Schluss die Hilfsprozedur $\mathfrak{S}$ ein einziges Mal aufzurufen und dessen Ausgabe sogleich an die das Ausgabemedium weiterzuleiten.

Es gibt auch Paare von Problemen (L_1, L_2), für die $L_1 \preceq_T L_2$, aber $L_1 \not\preceq_K L_2$. Für die Problemklasse, die uns hier interessiert (siehe nächster Abschnitt), ist dieser Effekt weder bekannt, noch kann er ausgeschlossen werden.

Proposition 26.10. *Seien L_1, L_2 Sprachen.*

a) *Wenn $L_1 \preceq_K L_2$ und $L_2 \preceq_K L_3$, so $L_1 \preceq_K L_3$.*
b) *Wenn $L_1 \preceq_T L_2$ und $L_2 \preceq_T L_3$, so $L_1 \preceq_T L_3$.*

Beweis. Durch einfaches Aneinanderhängen der Reduktionen. □

Abschließend ist vielleicht noch zu bemerken, dass das Konzept der Turing-Reduktion ohne Probleme auf Berechnungsprobleme erweitert werden kann und dort dann eine ähnliche Bedeutung für die Komplexität von Problemen hat. Eine feste Bezeichnung für solche Reduktionen existiert jedoch nicht.

A.4 NP – eine Grauzone. Wir kennen einige (Entscheidungs-)Probleme, die nach unserem heutigen Wissensstand nicht effizient gelöst werden können und von denen man das auch nicht irgendwann in der Zukunft erwartet, wie z.B. Faktorisieren, IP, SAT, 3SAT, Minesweeper. All diese Probleme haben jedoch eine auffällige Gemeinsamkeit:

Durch eine geeignete Zusatzinformation kann man die Korrektheit einer gegebenen ‚Ja'-Antwort überprüfen.

Konkret heißt das:

- Durch Angabe der Faktoren von N kann man leicht überprüfen, ob N einen Primfaktor im Intervall $[a, b]$ hat (hier wird das nichttriviale Resultat verwendet, dass Primzahltests in Polynomialzeit möglich sind).
- Bei IP genügt die Angabe der Variablenbelegung. Dann kann durch einfaches Rechnen geprüft werden, dass der Schwellwert-Vektor (threshold) d überschritten ist.
- Ähnlich wie bei IP genügt bei SAT und 3SAT die Angabe der Wahrheitswerte der Variablen.
- Beim MINESWEEPER-Problem reicht es, die Position der Minen anzugeben. Geprüft wird dann, ob die Anzahl der Minen mit der geforderten übereinstimmt und ob auf jedem aufgedeckten Felde die Zahl angrenzender Minen mit dem gegebenen Wert übereinstimmt.

Diese Zusatzinformationen nennt man *Zeugen* (*witnesses*).

Wir definieren nun **NP** als die Klasse derjenigen Entscheidungsprobleme, zu denen eine Ja-Antwortdie durch Angabe ‚kurzer' Zeugen ‚schnell' geprüft werden kann.

DEFINITION 26.11. *Sei $L \subseteq \Sigma^*$ eine Sprache. Wir sagen $L \in$ **NP**, wenn eine Funktion $F : \Sigma^* \times \Sigma^* \to \{W, F\}$ existiert, die polynomiell entscheidbar ist, so dass gilt $L = \{x \in \Sigma^* \mid \exists y \in \Sigma^* : \; F(x, y) = W \text{ und } |y| \leq g(|x|)\}$ für ein Polynom g.*

Hier spielt y die Rolle des Zeugen, der $x \in L$ beweist.

Für Nein-Antworten wird die Nachprüfbarkeit bewusst nicht gefordert! Für das Faktorisiersproblem ist das natürlich ohne weiteres möglich (wie?), während die Unerfüllbarkeit einer SAT-Formel anscheinend nur sehr schwer nachzuweisen ist.

Man beachte, dass **NP** auch die ‚leichten' Probleme enthält:

$$\mathbf{P} \subseteq \mathbf{NP}$$

denn für die Probleme aus **P** ist schon das leere Wort als Zeuge ausreichend.

Wie bei den weiter oben definierten Komplexitätsklassen gilt auch hier, dass die Elemente von **NP** Probleme und nicht Algorithmen (oder Prüffunktionen) sind.

Das **P** in dieser Bezeichnung steht nach wie vor für ‚Polynomialzeit', während das **N** auf nichtdeterministische Turing-Maschinen hindeutet, mit denen man diese Klasse ganz formal einführen kann, s. [1, 3].

Ein tiefliegender Satz, dessen Beweis hier nicht nur zu viel Zeit in Anspruch nähme, sondern mit dem bisherigen Maß an Abstraktion gar nicht formuliert werden kann, lautet:

SATZ 26.12. *SAT ist **NP**-*vollständig*, das heißt, SAT $\in$**NP**, und $L \preceq$ SAT für alle $L \in$ **NP**.*

In Fortsetzung der anschaulichen Formulierungen bedeutet das also, dass SAT das schwerste Problem in **NP** ist. Alternativ kann man sagen, dass alle Probleme in **NP** bis auf einen polynomiellen Summanden höchstens die Laufzeit erfordern, die zur Lösung von SAT nötig ist.- Diese Definition macht natürlich nur dann Sinn, wenn man vermutet, dass $\mathbf{P} \neq \mathbf{NP}$.

Diese Vermutung wird auch allgemein als richtig angesehen, ist aber bislang immer noch unbewiesen (auf einen Beweis oder eine Widerlegung ist seit einigen Jahren eine Million Dollar ausgesetzt).

Dass diese Vermutung also immer noch ungeklärt im Raum steht, impliziert, dass man für *kein* **NP**-Problem (wie FAKTORISIEREN, IP) beweisen kann, dass es *nicht* effizient lösbar ist. Mit dem Begriff der **NP**-Vollständigkeit kann man dieser Annahme aber eine gewisse Schärfe und Plausibilität verleihen:

KOROLLAR 26.13. *Genau dann ist SAT in polynomieller Zeit entscheidbar, wenn* alle *Probleme in **NP** dies sind.*

...und das sollte als unrealistisch genug gelten.

BEWEIS.

wenn: klar, da SAT $\in$ **NP**.

genau dann: Wenn $L \in \mathbf{NP}$, so ist laut 26.12 $L \preceq_K$ SAT. Wenn aber SAT effizient lösbar ist, so ist L dies dank der Karp-Reduktion auch. □

Die Entdeckung eines polynomiellen Algorithmus' für SAT hätte also gewaltige Folgen für algorithmische Landschaft. Im Gegensatz dazu gilt es als unwahrscheinlich, dass auch FAKTORISIEREN **NP**-vollständig ist. Daher würde ein effizienter Algorithmus zum Zerlegen ganzer Zahlen sicher großes Aufsehen erregen, und er würde viele kryptographische Anwendungen dieses Problems (wie RSA) bedrohen. Dennoch würde dies nichts über die Komplexität etwa von SAT aussagen.

Man kann sich nun fragen, ob der Begriff ‚**NP**-vollständig' sich inhaltlich ändert, wenn man anstatt Karp- nur noch Cook-Reduktionen fordert. Dazu ist nichts bekannt. Meines Wissens gibt es erstaunlicherweise nicht einmal ein Problem, für das man ein Turing-, aber keine Karp-Reduktion auf SAT bekannt ist.

Zum selbständigen Beweisen sind die folgenden Aussagen wichtig.

PROPOSITION 26.14. *Seien A, B Sprachen. Wenn $A \preceq_K B$, $B \in$ **NP** und A **NP**-vollständig ist, so ist auch B **NP**-vollständig.*

BEWEIS. Nach Voraussetzung ist $A \in \mathbf{NP}$. Sei nun $L \in \mathbf{NP}$. Wegen der **NP**-Vollständigkeit von A gilt dann $L \preceq_K A \preceq_K B$, laut 26.10 also auch $L \preceq_K B$. □

SATZ 26.15. *IP ist **NP**-vollständig.*

BEWEIS. Wie oben erwähnt, ist IP $\in$ **NP**. Wir zeigen nun

$$\text{SAT} \preceq_K \text{01IP} \preceq_K \text{IP}$$

für ein noch zu definierendes Problem 01IP. Im Lichte von 26.14 und 26.10 ist klar, das das genügt.

(0,1)-INTEGER PROGRAMMING (01IP). Gegeben $A \in \mathbb{Z}^{s\times r}$ und $b \in \mathbb{Z}^s$.
Existiert ein $x \in \{0,1\}^r$ so dass $Ax \geq b$?

Die Aussage 01IP $\preceq_K$ IP ist leicht, denn die Forderung $x \in \{0,1\}^r$ ist gleichwertig mit

$$\begin{pmatrix} I \\ -I \end{pmatrix} \cdot x \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ \vdots \\ -1 \end{pmatrix}$$

Danach geht das Problem durch Umkehren der Vorzeichen wie Ungleichheitszeichen in ein Teilproblem von IP über.

Gegeben eine SAT-Formel $\Phi = (c_1 \wedge \ldots c_r)$ in den Variablen $x_1, \ldots, x_s$. ObdA kommt in einer Klausel jede Variable entweder bejaht *oder* verneint vor; denn andernfalls hat die Klausel immer den Wert W und kann daher ausgelassen werden. Definiere eine 01IP-Instanz $(A,\, b)$ durch

$$\begin{aligned} A_{ij} &= \begin{cases} 1 & \text{wenn } x_j \text{ (bejaht) in } c_i \text{ vorkommt} \\ -1 & \text{wenn } \bar{x}_j \text{ in } c_i \text{ vorkommt} \\ 0 & \text{sonst} \end{cases} \\ d_i &= \#\{j \mid x_j \text{ tritt (bejaht) in } c_i \text{ auf}\} - 1 \end{aligned}$$

Man rechnet nun leicht nach, dass die Variablenbelegung $\{W, F\}^r$ von Φ eine Lösung ξ mit

$$\xi_i = \begin{cases} 1 & \text{wenn } x_i = W; \\ 0 & \text{sonst.} \end{cases}$$

für 01IP liefert und dass umgekehrt für jede Lösung ξ der 01IP-Instanz

$$x_i = \begin{cases} W & \text{wenn } \xi_i = 1; \\ F & \text{sonst.} \end{cases}$$

die Formel Φ erfüllt. □

Neben dem Begriff der **NP**-Vollständigkeit gibt es auch die Bezeichnung **NP**-*hart*. Ein Problem H heißt **NP**-hart, wenn $L \preceq H$ für alle $L \in \mathbf{NP}$. Auch hier ist der Reduktionsbegriff variabel. Im Gegensatz zu den **NP**-vollständigen Problemen wird hier nicht gefordert, dass $H \in \mathbf{NP}$.

Zum Beispiel das Problem QUADRATIC PROGRAMMING: Ersetzt man bei der (rationalen) linearen Programmierung die Zielfunktion durch ein quadratisches Polynom, so ist nicht mehr bekannt, ob das betreffende Entscheidungsproblem (vgl. IP) in **NP** liegt. Die

Schwierigkeit, das zu zeigen, liegt darin, dass ein Lösungsvektor (wie so oft der erste Kandidat für einen **NP**-Zeugen) gegenüber den Koeffizienten des Systems sehr große Einträge haben kann, was in Konflikt mit der polynomiellen Länge eines Zeugen steht, die in der Definition gefordert wurde.

Schlusswort. Die Komplexitätstheorie, insbesondere die Theorie der **NP**-Vollständigkeit, kann Hinweise auf die Schwierigkeit eines Problems geben. Das geschieht in den meisten Fällen nicht durch explizite untere Schranken, sondern durch bedingte Folgerungen, wie etwa: *Wenn ein Polynomialzeit-Algorithmus für SAT existiert, dann auch für alle* **NP**-*Probleme.* Das ist also kein Beweis für die Härte des Problems, aber ein sehr plausibler Hinweis.

Mittlerweile sind viele Probleme als **NP**-vollständig erkannt worden (s. [1]). Populäre Vertreter sind etwa TRAVELLING SALESMAN PROBLEM, HAMILTON CIRCUIT, GRAPHEN-3-FÄRBBARKEIT und das RUCKSACK-Problem (das letztere ähnelt sehr dem hier behadelten IP). Es ist typisch, dass viele (vor allem schon früh erkannte) **NP**-vollständige Probleme aus der Graphentheorie stammen. Doch mittlerweile werden immer mehr Probleme aus dem Bereich der diskreten Mathematik in diesen Zusammenhang gestellt.

Bei all dieser Rede von der vermeintlichen Schwierigkeit von Problemen ist aber auch zu bedenken, dass sich Komplexität immer auf die worst-case-Laufzeiten von Algorithmen bezieht. Es ist möglich und tritt bei typischen **NP**-vollständigen Problemen (wie 3SAT) auch tatsächlich ein, dass für die ‚meisten' Instanzen das Problem schnell zu lösen ist und nur ‚wenige' Instanzen eine hohe worst-case-Laufzeit (bei den bekannten Algorithmen) bewirken. Dieser Effekt verstärkt sich noch, wenn man *randomisierte Algorithmen* verwendet. Jedoch ist es sehr unwahrscheinlich, dass man mit Zufalls Hilfe alle **NP**-vollständigen Probleme, mit natürlichen Wahrscheinlichkeitsverteilungen versehen, lösen kann; hierfür gibt es ähnliche Konstruktionen wie jene, die wir hier für den deterministischen Fall durchgeführt haben.

Daneben gibt es noch weitere Bewältigungsstrategien gegen schwere Probleme, etwa die sogenannten *Approximationsalgorithmen.* Im wesentlichen denkt man hier an Optimierungsprobleme, und die Strategie besteht darin, den Schwellwert für die Zielfunktion zu lockern, also sich mit etwas suboptimalen Lösungen zufriedenzugeben.

Ferner sollte man noch beachten, dass die Komplexität eines Problems sich durch kleine Änderungen oder Einschränkungen stark verbessern (bzw. veschlechtern) kann (Beispiel: 2SAT ist in Polynomialzeit lösbar; beachte auch den Unterschied zwischen gewöhnlicher rationaler linearer Programmierung und IP).

Schließlich ist bei konkreten Algorithmen darauf zu achten, wie schmerzlich die Laufzeitschranke wirklich ist. Auch hier können sich unter dem Begriff „exponentiell" sehr unterschiedliche Situationen verbergen.

Alles in allem sollte diese Zusammenstellung davor warnen, bei dem Worte ‚**NP**-vollständig' zu glauben, damit wäre alles zu einem Problem gesagt. Dieser Begriff kann, wie andere Konzepte aus der Komplexitätstheorie, durchaus wertvolle Hinweise auf die Schwierigkeit von Problemen geben. Allein man sollte diese Aussagen im Zusammenhang mit anderen Aspekten des Problems sehen.

Literaturverzeichnis zum Exkurs NP-Vollständigkeit:

[1] Michael R. Garey, David S. Johnson: *Computers and Intractibility – A Guide to the Theory of NP-Completeness*, Freeman, San Francisco 1979.

[2] Kurt Mehlhorn: *Data Structures and Algorithms*, Bd. 2: *Graph Algorithms and NP-Completeness*, Springer, Berlin u.a. 1984.

[3] Christos H. Papadimitriou: *Computational Complexity*, Addison-Wesley, New York u.a. 1994.

[4] `http://www.sed.free.fr/complex/mines.html`