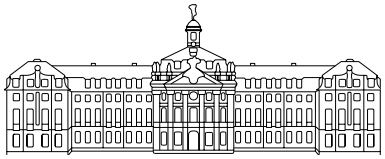


Seminarausarbeitung zum Seminar

Algorithmen der Computeralgebra
(SS 08)

Polynomfaktorisierung über endlichen Körpern

Westfälische Wilhelms-Universität Münster
Fachbereich Mathematik und Informatik
Institut für Informatik

vorgelegt von
Marko Ernsting, Roland Leißa, Steven Keuchel
im Sommersemester 2008

Inhaltsverzeichnis

1 Einleitung

Diese Seminararbeit behandelt Algorithmen zur Faktorisierung von Polynomen über endlichen Körper in einer Variablen. Diese Begrenzung wurde getroffen um das Thema im Rahmen einer Seminararbeit angemessen behandeln zu können. Algorithmen zur Faktorisierung von Polynomen über $\mathbb{Z}$ bzw. $\mathbb{Q}$ oder in mehreren Variablen hätten den Rahmen schnell gesprengt und eine verständliche Einarbeitung in das Thema unmöglich gemacht. Der interessierte Leser wird die hier vermittelten Inhalte nutzen können um einen leichten Einstieg in eine weiterführende Vertiefung ins Themengebiet der Faktorisierung von Polynomen zu erlangen.
Natürlich können wir keine vollständige Einführung in die vielfältigen Algorithmen zur Polynomfaktorisierung geben. Wir beschränken uns hier auf die wesentlichen Algorithmen die in den letzten 20 Jahren eingesetzt wurden. Die weiteren Algorithmen und Anwendungen werden mit der Intention behandelt, die Funktionsweise der eigentlichen Algorithmen zu verdeutlichen.
Die Anwendungen für Faktorisierungsalgorithmen von Polynomen über endlichen Körpern sind vielfältig. Zum einen werden sie auch bei der Faktorisierung über $\mathbb{Z}$ oder $\mathbb{Q}$ benötigt und dadurch z.B. auch bei der symbolischen Integration. Es finden sich auch viele Anwendungen in der Kryptografie, etwa bei dem *Index-Calculus-Algorithmus* für die Berechnung von diskreten Logarithmen. Wir stellen eine der naheliegensten Anwendungen - das Finden von Nullstellen - kurz in einem eigenen Kapitel vor.
Die Faktorisierung von Polynomen über endlichen Körpern läuft gewöhnlich in drei Phasen ab:

(i) Quadratfreie Faktorisierung

(ii) Verschiedengradige Faktorisierung

(iii) Gleichgradige Faktorisierung

In der ersten Phase wird das Polynom in *quadratfreie* Teilpolynome zerlegt, um mehrfach vorkommende Faktoren herauszunehmen. Die zweite Phase kümmert sich dann darum, dass die Faktoren von verschiedenem Grad gefunden werden. Die letzte Phase versucht diese Teilpolynome weiter aufzuteilen. Da hier schon die Faktoren verschiedenen Grades getrennt worden sind, müssen irreduzible Faktoren gleichen Grades gefunden werden.
Wir haben die Vorgehensweise aus [8] übernommen und beginnen mit der zweiten Phase. Diese ist die wichtigste und aus theoretischer Sicht die einfachste der drei Phasen. Sie bietet somit einen schnellen einfacheren Einstieg in die Theorie der Faktorisierung als die algorithmische Chronologie. Insbesondere liefert sie schnell anschauliche Ergebnisse. Eine quadratfreie Zerlegung wird natürlich nachgeliefert, nachdem ein erster Ansatz für einen Faktorisierungsalgorithmus vorgestellt worden sein wird.
Alle Faktorisierungsalgorithmen, insbesondere über endlichen Körpern bauen auf den Ergebnissen auf, die für Algorithmen zur Bestimmung des größten gemeinsamen Teilers und zu Restklassen und deren Arithmetik erzielt wurden. Dementsprechend sind die Theorien endlicher Körper, endlicher Körpererweiterungen, Ringen sowie Restklassenringen in dem Umfang nötig, wie sie in etwa in einer einführenden Veranstaltung zur Algebra oder zur algebraischen Zahlentheorie vorgestellt werden.
Die nötigsten mathematischen Grundlagen und Ergebnisse sowie fundamentale Algorithmen finden deshalb in einem eigenen Kapitel Platz. Das dort vorgestellte Material dient lediglich als Wiederholung und beschränkt sich auf die wesentlichen Aussagen, die für

Korrektheitsbeweise und Nachvollziehbarkeit der zur Polynomfaktorisierung vorgestellten Algorithmen notwendig sind. Aus diesem Grunde werden die Beweise an dieser Stelle ausgelassen. Dem unkundigen Leser sei darum dringend die weiterführende Lektüre etwa in [2] oder [14] nahegelegt.

2 Mathematische und algorithmische Grundlagen

2.1 Ringtheorie

Definition 2.1.1. (Ring). *Ein* Ring *ist eine Menge R mit zwei binären Verknüpfungen $\cdot$, $+ : R \times R \longrightarrow R$, die folgenden Bedingungen genügen*

(i) *R mit $+$ ist abelsche Gruppe mit 0.*

(ii) *$\cdot$ ist assoziativ.*

(iii) *$1 \in R$ ist Einselement für $\cdot$.*

(iv) *Distributivität: $\forall a, b, c \in R$: $a(b+c) = (ab) + (ac)$ und $(b+c)a = (ba) + (ca)$.*

Der Ring heißt kommutativ, *wenn $\cdot$ kommutativ ist.*

Oft wird in Definitionen für Ringe das Einselement fallengelassen. Der Einfachheit halber sei hier im Folgenden mit Ring stets ein kommutativer Ring mit Einselement gemeint.

Definition 2.1.2. (Unterring). *Sei R ein Ring. Ein* Unterring *von R besteht aus einer Teilmenge $S \subset R$, derart dass gilt:*

(i) *S ist bezüglich $+$ eine Untergruppe von R.*

(ii) *S ist multiplikativ abgeschlossen, d.h. $a, b \in S \Rightarrow ab \in S$.*

Definition 2.1.3. (Einheiten). *Besitzt $a \in R$ ein multiplikativ Inverses, d.h. existiert ein $b \in R$ mit $ab = 1$, so nennen wir a eine Einheit in R. Wir nennen die Menge aller Einheiten die Einheitengruppe von R und bezeichnen diese mit $R^\times$.*

Man überlegt sich leicht, dass die Einheitengruppe tatsächlich eine multiplikative Gruppe bildet. Es seien im Folgenden zwei einfache Beispiele für Ringe aufgeführt.

(1) $(\mathbb{Z}, +, \cdot)$, also die ganzen Zahlen zusammen mit der Addition und Multiplikation bilden einen Ring. Es ist $\{1, -1\}$ die Einheitengruppe in $\mathbb{Z}$.

(2) Seien R und S Ringe, dann ist die Menge $R \times S = (r, s) : r \in R, s \in S$ - das *kartesische Produkt* von R und S - wieder ein Ring. Die Ringoperationen sind komponentenweise durch

$$(r_1, s_1) + (r_2, s_2) = (r_1 + r_2, s_1 + s_2) \text{ und } (r_1, s_1) \cdot (r_2, s_2) = (r_1 r_2, s_1 s_2)$$

für alle $r_1, r_2 \in R$ und $s_1, s_2 \in S$ erklärt. Die Einheitengruppe ist das *kartesische Produkt* der Einheitengruppen

$$(R \times S)^\times = R^\times \times S^\times \ .$$

2.1.1 Ringhomomorphismen und Restklassen

Definition 2.1.4. (Ringhomomorphismus). *Ein* Ringhomomorphismus *von einem Ring R in einen Ring S ist eine Abbildung φ, für die gilt:*

(i) *$\varphi(a+b) = \varphi(a) + \varphi(b) \quad \forall a, b \in R$.*

(ii) *$\varphi(a \cdot b)\varphi(a) \cdot \varphi(b) \quad \forall a, b \in R$.*

(iii) $\varphi(1_R) = 1_S$.

Definition 2.1.5. (Ideal). *Sei R ein Ring. Eine Teilmenge $I \subset R$ heißt* Ideal, *falls gilt:*

(i) Es ist I additive Untergruppe von R.

(ii) $a \in I, r \in R \Longrightarrow ar \in I$.

Beispiele 2.1.6. Die Menge $2\mathbb{Z}$, also alle geraden ganzen Zahlen ist ein Ideal im Ring $\mathbb{Z}$. Für $2\mathbb{Z} + 1$ trifft dies jedoch nicht zu.

Definition 2.1.7. (Restklasse). *Sei wieder R ein Ring, I ein Ideal. Für $r \in R$ ist die Menge*

$$r \bmod I = r + I = \{r + a : a \in I\} \subseteq R$$

die Restklasse von r modulo I.

Definition 2.1.8. (Restklassenring). *Die Menge $R/I = \{r \bmod i : r \in R\}$ aller Restklassen mit den Verknüpfungen*

(i) $(r \bmod I) + (s \bmod I) = (r + s) \bmod I$ *und*

(ii) $(r \bmod I) \cdot (s \bmod I) = (rs) \bmod I$

wird als Restklassenring *von R modulo I bezeichnet.*

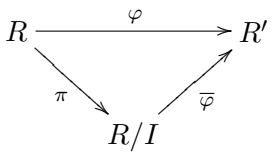
Ein Restklassenring von R erbt dabei die Ringeigenschaft von R. Dazu bildet man zunächst R/I als abelsche Gruppe mit I als additive Gruppe von R. Weiter definiert man mit $x, y \in R$ eine Addition $xI + yI = (x + y)I$ und eine Multiplikation $xI \cdot yI := (xy)I$ bei denen sich schnell einsehen lässt, dass sie wohldefiniert sind.

Definition 2.1.9. (Kanonischer Homomorphismus). *Ein* kanonischer Homomorphismus *oder eine* kanonische Projektion *ist eine Abbildung $\pi : R \longrightarrow R/I$ die jedes $r \in R$ auf seine Restklasse $r \bmod I$ abbildet:*

$$\pi : R \longrightarrow R/I, \quad x \mapsto x \bmod I$$

Insbesondere gilt $\ker \pi = I$.

Satz 2.1.10. (Homomorphiesatz). *Seien R und S Ringe, weiter $\varphi : R \longrightarrow R'$ ein Ringhomomorphismus und $I \subset R$ ein Ideal mit $I \subset \ker \varphi$. Dann existiert eindeutig ein Ringhomomorphismus $\overline{\varphi} : R/I \longrightarrow R'$, so dass das Diagramm*



kommutiert. Es gilt

$$\operatorname{im} \overline{\varphi} = \operatorname{im} \varphi, \ \ker \overline{\varphi} = \pi(\ker \varphi), \qquad \ker \varphi = \pi^{-1}(\ker \overline{\varphi}).$$

Insbesondere ist $\overline{\varphi}$ genau dann injektiv, wenn $I = ker\varphi$.

2.1.2 Polynomringe

Definition 2.1.11. (Polynomring). *Sei R ein Ring. Der* Polynomring $R[x]$ *ist definiert als die Menge S von Folgen $a = (a_0, a_1, \ldots)$ mit Werten aus R, wobei fast alle a_i verschwinden. Die Addition und Multiplikation sind wie folgt definiert:*

$$(a_0, a_1, \ldots) + (b_0, b_1, \ldots) = (a_0 + b_0, a_1 + b_1, \ldots) \text{ und}$$

$$(a_0, a_1, \ldots) \cdot (b_0, b_1, \ldots) = (c_0, c_1, \ldots) \text{ mit } c_n = \textstyle\sum_{i=0}^{n} a_i b_{n-i}.$$

Man schreibt Polynome gewöhnlich in der Form

$$\sum_{i \in \mathbb{N}} a_i x^i \text{ oder } \sum_{i=0}^{n} a_i x^i \ .$$

Man prüft leicht nach, dass $R[x]$ mit diesen Verknüpfungen tatsächlich ein Ring ist. Es ist insbesondere $(0, 0, \ldots)$ das Nullelement und $(1, 0, 0, \ldots)$ das Einselement.

Definition 2.1.12. (Grad von Polynomen). *Der* Grad *von $f = \sum a_i x^i \in R[x]$ ist definiert durch* $\deg f := \max\{i; a_i \neq 0\}$.

2.1.3 Primfaktorzerlegung

Definition 2.1.13. (Nullteiler). *Ein Element $a \in R$ Ring heißt* Nullteiler*, wenn es ein Element $b \in R$, $b \neq 0$ gibt, für das $a \cdot b = 0$ oder $b \cdot a = 0$ ist.*

Definition 2.1.14. (Integritätsring). *Einen kommutativen Ring R mit $1 \neq 0$ nennt man* Integritätsring*, wenn R keine nicht-trivialen Nullteiler besitzt.*

Beispiel 2.1.15. $(\mathbb{Z}, +, \cdot)$ ist Integritätsring.
Der Restklassenring $\mathbb{Z}/n\mathbb{Z}$ ist genau dann ein Integritätsring, wenn n Primzahl ist.

Definition 2.1.16. (Irreduzibilität). *Es sei R ein Integritätsring und $p \in R$ ein Element, welches keine Einheit und von Null verschieden ist.*

(i) p heißt irreduzibel*, wenn aus einer Gleichung $p = ab$ mit $a, b \in R$ stets folgt, dass a oder b eine Einheit in R ist. Andernfalls nennt man p reduzibel.*

(ii) p heißt Primelement, wenn aus $p \mid ab$ mit $a, b \in R$ stets $p \mid a$ oder $p \mid b$ folgt.

Beispiel 2.1.17. Wir betrachten den Polynomring $R[x]$. Wie man sich über den Grad eines Polynoms leicht klar macht, ist jedes Polynom der Gestalt $x - r$ mit $r \in R$ irreduzibel. Das Polynom $x^2 + x$ ist nicht irreduzibel, denn es lässt sich als Produkt der Nichteinheiten $x, x + 1$ schreiben.

Definition 2.1.18. (Faktorieller Ring). *Ein Integritätsring R in dem jede Nichteinheit $a \in R - \{0\}$ in ein bis auf Umordnung und Multiplikation mit Einheiten eindeutiges Produkt von irreduziblen Elementen zerfällt, nennt sich* faktorieller Ring.

In einem faktoriellen Ring ist ein Element genau dann irreduzibel, wenn es ein Primelement ist. Bei der *eindeutigen* Darstellung als Produkt von irreduziblen Elementen bietet es sich an, assoziierte Elemente, d.h. Elemente die sich nur durch Multiplikation mit einer Einheit unterscheiden, zusammenzufassen. Hat man etwa a aus einem faktoriellen Ring R

gegeben, so erhält man eine Darstellung $a = \varepsilon p_1^{\nu_1} \dots p_n^{\nu_n}$ mit einer Einheit $\varepsilon \in R^\times$, paarweise nichtassoziierten Primelementen $p_1, \dots, p_n$ sowie positiven Vielfachheiten $\nu_i > 0$. Im Folgenden nennen wir dies eine Primfaktorzerlegung von a. Die Faktorisierung, insbesondere die Faktorisierung von Polynomen, besteht nun darin, eine Primfaktorzerlegung zu bestimmen.

Definition 2.1.19. (Größter gemeinsamer Teiler). *Sei R ein Ring und a, b, $c \in R$. Dann ist c der* größte gemeinsame Teiler *von a, b, oder kurz $c = \gcd(a, b)$ wenn gilt:*

$c \mid a$ *und* $c \mid b$ *und für alle* $e \in R$ *gilt* $e \mid a$, $e \mid b \Longrightarrow e \mid c$.

Der größte gemeinsame Teiler ist kommutativ, assoziativ und erfüllt das Distributivgesetz. Weitere Eigenschaften liest man etwa in [8, Seite 45] nach.

2.1.4 Der euklidische Algorithmus

Definition 2.1.20. (Euklidischer Ring). *Ein Integritätsring R mit einer Abbildung $d : R - \{0\} \longrightarrow \mathbb{N}$ heißt* euklidischer Ring*, wenn es zu allen Elementen $a, b \in R$, $b \neq 0$ stets Elemente $q, r \in R$ gibt mit*

$$a = qb + r \text{ wobei } d(r) \leq d(b) \text{ oder } r = 0 \ .$$

Wir nennen d die Normabbildung des euklidischen Rings R.

Man kann zeigen, dass jeder euklidische Ring auch ein faktorieller Ring ist. In einem euklidischen Ring gibt es durch die Division mit Rest ein Verfahren zur Bestimmung eines größten gemeinsamen Teilers, nämlich den *euklidischen Algorithmus*. Eine formale Beschreibung liefert [2] auf Seite 51.

Nun besitzt man alle Mittel um den gcd von Polynomen in einer Variablen wieder aufzugreifen. Es fehlte dort lediglich noch eine Normabbildung, die einfach als Grad des Polynoms gewählt wird, also $d(a) = \deg(a)$ - und eine Division mit Rest im Polynomring. Im Allgemeinen existiert diese jedoch in Polynomringen keineswegs. Für Polynome über den ganzen Zahlen ($\mathbb{Z}[x]$) ist es beispielsweise unmöglich x^4 mit Rest durch $3x + 1$ zu dividieren. Allerdings ist es immer möglich durch normierte Polynome mit Rest zu teilen. Oder allgemeiner durch Polynome mit einer Einheit als Leitkoeffizient. Insbesondere existiert die Polynomdivision für Polynomringe über Körpern. Polynomringe über Körpern sind somit euklidische Ringe. Die Polynomdivision sollte bereits aus der Schulmathematik bekannt sein. Die einzige "Schwierigkeit" besteht in der Übertragung dieses Verfahrens in Polynomringe über beliebigen Körpern, insbesondere über endlichen Körpern.

(1) Für die Zahlen 96 und 42 über dem euklidischen Ring $\mathbb{Z}$ verläuft der euklidische Algorithmus etwa folgendermaßen:

$$\begin{aligned} 96 &= 2 \cdot 42 + 12 \\ 42 &= 3 \cdot 12 + 6 \\ 12 &= 2 \cdot 6. \end{aligned}$$

Als Ergebnis erhält man hier 6 als größten gemeinsamen Teiler der beiden Zahlen.

(2) Berechnung von $\gcd(x^2 + 3x + 2,\ x^4 + 3x^3 + x + 4)$ in $F_5[x]$ mit Hilfe des euklidischen Algorithmus:

$$\begin{aligned}(x^4+3x^3+x+4) &= (x^2+3)\cdot(x^2+3x+2)+(2x+3),\\ (x^2+3) &= (3x+1)\cdot(2x+3)+0\ .\end{aligned}$$

Hier ergibt sich also $(2x+3)$ als größter gemeinsamer Teiler.

Im ersten Schritt des Algorithmus ist direkt zu erkennen, dass der gcd offensichtlich nicht eindeutig ist. Er ist allerdings bis auf Multiplikation mit Einheiten eindeutig, so dass wir in einem Polynomring über Körpern stets fordern, dass der größte gemeinsame Teiler normiert ist.

2.1.5 Chinesischer Restsatz

Satz 2.1.21. (Chinesischer Restsatz). *Es sei R ein Ring und $a \in R$. Es sei $a = p_1^{\nu_1}\cdot\ldots\cdot p_n^{\nu_n}$ eine Primfaktorzerlegung von a. Sei weiter $\pi_i : R \to R/\langle p_1^{\nu_1}\rangle$ jeweils der kanonische Homomorphismus. Dann existiert ein Isomorphismus*

$$\begin{aligned}R/\langle a\rangle &\xrightarrow{\sim} R/\langle p_1^{\nu_1}\rangle \times\cdots\times R/\langle p_n^{\nu_n}\rangle,\\ r \bmod a &\mapsto (\pi_1(r),\ldots,\pi_n(r)) = (r \bmod p_1^{\nu_1},\ldots,r \bmod p_n^{\nu_n}).\end{aligned}$$

Dieses Konzept ist grundlegend und wird beispielsweise in Algorithmen zur schnellen Multiplikation, Division oder Berechnung des gcd verwendet. Kommt aber auch in darauf aubauenden Algorithmen z.B. für Polynomfaktorisierung von Polynomen über Körpern in zwei Variablen oder über $\mathbb{Q}[x]$ vor, wie ab Seite 409 in [8] beschrieben. Eine sehr anschauliche Übersicht über Algorithmen, die jeweils verschiedene Varianten dieses Konzeptes verwenden findet sich in [8, Seite 435].

2.2 Körper und Körpererweiterungen

Definition 2.2.1. (Körper). *Ein* Körper *K ist ein Integritätsring in dem jedes Element $a \in K$, $a \neq 0$ eine Einheit ist, also insbesondere ein multiplikatives Inverses besitzt.*

Satz 2.2.2. *Es sei R ein euklidischer Ring und $p \in R$ ein Primelement in R. Dann ist $R/\langle p\rangle$ ein Körper. Dabei bezeichne $\langle p\rangle$ das von p erzeugte Ideal pR.*

Der vorangehende Satz ist grundlegend für die Konstruktion endlicher Körper. Hat man z.B. $R = \mathbb{Z}$ und eine Primzahl p gegeben, so ist $F_p := \mathbb{Z}/p\mathbb{Z}$ ein Körper. Für eine reduzible Zahl $p = qr$ ist allerdings $\mathbb{Z}/(pr)\mathbb{Z}$ kein Körper, denn man hat die Nullteiler q, r. Wir wollen noch eine andere Konstruktion angeben. Es sei K ein Körper. Wir betrachten $R = K[x]$ den Polynomring in einer Variablen über K. Ist nun $f \in K[x]$ prim bzw. irreduzibel so ist $K[x]/\langle f\rangle$ ein Körper. Ein Repräsentantensystem für die Restklassen sind alle Polynome die einen kleineren Grad haben als f.

Beispiel 2.2.3. Wir wollen hier explizit einen endlichen Körper mit 4 Elementen konstruieren. Wir betrachten $\mathbb{F}_2[x]$ sowie das Polynom $f = x^2+x+1$ welches irreduzibel ist, da es keine Nullstellen in $\mathbb{F}_2$ besitzt. Die möglichen Reste sind $0, 1, x, x+1$ und mit $a := x$ und $b := x+1$ erhält man die Verknüpfungen

$+$	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

,

$\cdot$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

.

Ist allgemeiner f ein irreduzibles Polynom vom Grad n über $\mathbb{F}_q$ so kann man die Koeffizienten zu $x^0, \ldots, x^{n-1}$ beliebig aus $\mathbb{F}_q$ wählen. Man hat also q^n verschiedene Reste und damit einen Körper $\mathbb{F}_q[x]/\langle f\rangle$ mit q^n Elementen. Insbesondere kann man zu jeder Primzahlpotenz p^n einen Körper der Ordnung p^n konstruieren. Diese Bedingung ist auch notwendig, d.h. jeder endliche Körper hat eine Primzahlpotenz als Ordnung. Inbesondere sieht man, dass man eine Injektion $\mathbb{F}_q \to \mathbb{F}_q/\langle f\rangle, a \to aX^0$ gegeben ist. Man kann also $\mathbb{F}_p$ als Teilkörper von $\mathbb{F}_{p^n}$ ansehen.

Wie man weiter sieht wird $\mathbb{F}_q[x]/\langle f\rangle$ von der Restklasse von x erzeugt, d.h. es gilt $\mathbb{F}_q[x]/\langle f\rangle = \mathbb{F}_q[x \bmod f]$. Insbesondere ist $(x \bmod f)$ eine Nullstelle von f, denn es gilt $f(x \bmod f) = f(x) \bmod f = 0 \bmod f$.

Definition 2.2.4. (Charakteristik). *Sei K ein Körper und $\varphi : \mathbb{Z} \to K$ der kanonische Ringhomomorphismus. Wenn dann $p \in \mathbb{N}$ ein erzeugendes Element des Hauptideals* $\ker\varphi$ *ist, so heißt $p = \operatorname{char} K$ die* Charakteristik von K.

Etwas anschaulicher ist die Charakteristik eines Körpers nichts anderes als die minimale Anzahl mit der das Einselement zu sich selbst addiert Null ergibt. Für den Fall, dass dies niemals 0 ergibt ist die Charakteristik Null wie z.B. in $\mathbb{Q}, \mathbb{R}, \mathbb{C}$. Hat ein Körper p^n Elemente so ist p die Charakteristik dieses Körpers.

Definition 2.2.5. (Frobenius Homomorphismus). *Ist K ein Körper der Charakteristik p, so nennen wir $\varphi : K \to K,\ a \mapsto a^p$ den Frobenius-Homomorphismus zu K.*

Da $p \nmid \binom{p}{k}$ für $1 < k < p$ gilt, folgt die Homomorphieeigenschaft aus dem binomischen Lehrsatz

$$(a+b)^p = \sum_{k=0}^{p} \binom{p}{k} a^k b^{n-k} = a^p + b^p \ .$$

Von zentraler Bedeutung für die weiteren Betrachtungen ist Fermats Kleines Theorem:

Satz 2.2.6. (Fermats kleines Theorem). *Für jedes $a \in \mathbb{F}_q$, wobei $a \neq 0$, gilt $a^{q-1} = 1$ und für alle $a \in \mathbb{F}_q$ gilt $a^q = a$ und*

$$x^q - x = \prod_{a \in \mathbb{F}_q} (x - a) \in \mathbb{F}_q[x].$$

Beispiele 2.2.7. Für $\mathbb{F}_5$ ergibt sich:

$$\begin{aligned} 1^4 &= 1 \\ 2^4 &= 2 \cdot 2 \cdot 2 \cdot 2 = 1 \cdot 2 \cdot 2 = 1 \cdot 1 = 1 \\ 3^4 &= 3 \cdot 3 \cdot 3 \cdot 3 = -1 \cdot 3 \cdot 3 = -1 \cdot -1 = 1 \quad \text{etc.} \end{aligned}$$

$$\begin{aligned} 1^5 &= 1 \\ 2^5 &= 2^4 \cdot 2 = 1 \cdot 2 = 2 \\ 3^5 &= 3^4 \cdot 3 = 1 \cdot 3 = 3 \quad \text{etc.} \end{aligned}$$

Weiter lässt sich nachrechnen:

$$\begin{aligned}\prod_{a\in\mathbb{F}_5}(x-a) &= x(x-1)(x+1)(x-2)(x+2)\\ &= x(x^2-1)(x^2+1)\\ &= x(x^4-1)\\ &= x^5-x\end{aligned}$$

Dies liegt wiederum in $\mathbb{F}_5$.

Fermats kleines Theorem ist der Spezialfall zu folgendem Theorem

Satz 2.2.8. *Für jedes $d \geq 1$ ist $x^{q^d} - x \in \mathbb{F}_q[x]$ das Produkt von allen normierten irreduziblen Polynomen in $\mathbb{F}_q[x]$, deren Grad d teilt.*

Beispiele 2.2.9. Für $d = 1$ ergibt sich sofort Fermats kleines Theorem.

2.3 Algorithmen und Laufzeiten

2.3.1 Randomisierte Algorithmen

Wir benötigen einige grundlegende Aussagen für die Analyse randomisierter Algorithmen (vgl. [8, Kapitel 25.6]). Wir nennen einen randomisierten Algorithmus einen Las-Vegas Algorithmus, falls dieser mit einer Wahrscheinlichkeit $p < 1$ versagen kann. Versagt der Algorithmus nicht, so muss er ein korrektes Ergebnis zurückliefern.

Lemma 2.3.1. *Ist A ein Las-Vegas Algorithmus mit einer Erfolgswahrscheinlichkeit $0 < p_n < 1$ für eine Eingabe der Größe n, so ist $\frac{1}{p_n}$ die erwartete Anzahl der Iterationen die man benötigt um ein Ergebnis zu bekommen. Liegt weiter die Rechenzeit einer Iteration in $\mathcal{O}(f(n))$, so ist $\mathcal{O}(p_n f(n))$ die erwartete Gesamtrechenzeit.*

Beweis. Die Anzahl der benötigten Iterationen ist eine geometrisch verteilte Zufallsgröße X_n, d.h. es gilt

$$\mathbb{P}(X_n = k) = p_n(1-p_n)^{k-1}, \quad \mathbb{E}[X] = \frac{1}{p_n} \ .$$

□

2.3.2 Multiplikation von Polynomen

Um vom verwendeten Multiplikationsalgorithmus zu abstrahieren, wird folgende Notation verwendet:

Definition 2.3.2. (Multiplikationszeit $\mathbf{M}(n)$). *Sei R ein kommutativer Ring mit Eins. Eine Funktion $\mathbf{M} : \mathbb{N}^+ \longrightarrow \mathbb{R}^+$ heißt* Multiplikationszeit für $R[x]$, *falls für Polynome aus $R[x]$ vom Grad kleiner als n gilt, dass sie sich mit höchstens $\mathbf{M}(n)$ Operationen in R multiplizieren lassen.*

Die Funktion $\mathbf{M}$ bietet also eine Abstraktion vom verwendeten Multiplikationsalgorithmus. Sie wird in den folgenden Kapiteln für die Laufzeitabschätzungen der Faktorisierungsalgorithmen verwendet werden. $\mathbf{M}$ hängt von dem verwendeten Multiplikationsalgorithmus ab. Die Anzahl der Operationen für den klassischen Multiplikationsalgorithmus liegt in $\mathcal{O}(n^2)$. Der Algorithmus von Schönhage und Strassen [13] hat eine

benötigt $\mathcal{O}(n \log n \log\log n)$ Operationen. Der schnellste, aktuell bekannte Algorithmus zur Ganzzahlmultiplikation ist der Algorithmus von Fürer [7]. Er hat eine Laufzeit von $\mathcal{O}(n \log n 2^{\mathcal{O}(\log^* n)})$.

Wie auch in [8] gehen wir davon aus, dass $\mathbf{M}(n)/n \geq \mathbf{M}(m)/m$ für $n \neq m$ und $\mathbf{M}(nm) \leq m^2\mathbf{M}(n)$ gilt. Damit ergibt sich folgendes

Lemma 2.3.3. *Es sei* $\mathbf{M}$ *eine Multiplikationszeit. Es erfüllt* $\mathbf{M}$ *die folgenden Gleichungen*

(i) Superlinearität: $\mathbf{M}(mn) \geq m \cdot \mathbf{M}(n)$,

(ii) Subadditivität: $\mathbf{M}(m+n) \geq \mathbf{M}(n) + \mathbf{M}(m)$.

Lemma 2.3.4. *Es sei* K *ein endlicher Körper und* $f, g \in K[x]$ *zwei Polynome vom Grad kleiner gleich* n.

(i) Die schnelle Division mit Rest zur Berechnung von $q, r \in K[x]$ *mit* $f = qg + r$, *wobei* $\deg r < \deg g$ *oder* $r = 0$ *gilt, benötigt* $\mathcal{O}(\mathbf{M}(n))$ *Körperoperationen.*

(ii) Der schnelle euklidische Algorithmus zur Berechnung eines größten gemeinsamen Teilers benötigt $\mathcal{O}(\mathbf{M}(n) \log n)$ *Körperoperationen.*

(iii) Sind $m_1, \ldots, m_r \in K[x]$ *mit* $\sum_{i=1}^{r} \deg m_i \leq m$, *so können* $\gcd(f, m_1), \ldots, \gcd(f, m_r)$ *in* $\mathcal{O}(\mathbf{M} \log n)$ *Körperoperationen berechnet werden.*

2.3.3 Binäre Exponentiation

Für die schnelle Berechnung der Potenzen von von Zahlen gibt es den Algorithmus der Binären Exponentiation.Man schreibt zunächst den Exponenten in Binärschreibweise. Von links nach rechts gelesen nimmt man nun hiervon jede Ziffer und quadriert die Basis (oder das Ergebnis des vorhergehenden Schrittes) bei einer 0 bzw. quadriert und multipliziert bei einer 1.

Satz 2.3.5. *Sei* R *ein Ring. Für* $a \in R$ *berechnet sich die* $n-te$ *Potenz* a^n *mit Binärer Exponentiation (repeated squaring) in* $\mathcal{O}(\log n)$ *Multiplikationen in* R.

Beispiel 2.3.6. Sei $a = 9 \bmod 11$ und $R = \mathbb{Z}/11\mathbb{Z}$. Wir wollen a^{17} berechnen. Die Binärdarstellung von 17 ist 10001.

$$\begin{aligned} 9^{17} &\equiv (((9^2 \cdot 9)^2)^2)^2)^2 \cdot 9 \equiv (((4 \cdot 9)^2)^2)^2)^2 \\ &\equiv ((3^2)^2)^2)^2 \equiv (4^2)^2 \\ &\equiv 5^2 \equiv 3 \bmod 11 \end{aligned}$$

Beweise zur Laufzeit und Korrektheit findet man beispielsweise im Kapitel 4.3 in [8]. Wie oben im Beispiel bereits gesehen, kann man beim Rechnen in Restklassen das Entstehen großer Zahlen verhindern, indem man von den Zwischenergebnissen die Reste direkt berechnet.

3 Maple

3.1 Allgemeines

Die Entwicklung von Maple began 1980 and der University of Waterloo. Ziel war es ein Algebrasystem zu entwickeln, das auch auf günstigen Rechnern lauffähig sein sollte. 1988 wurde das Projekt in die Firmengründung Waterloo Maple Inc. ausgegliedert, die es seit dem weiterentwickelt. Maple besteht aus einem kleinen in C geschriebenen Kern. Auf dem Kern und weiteren externen Bibliotheken aufsetzend ist der Großteil der Funktionalität von Maple in Maple selbst geschrieben. Für viele Spezialgebiete existieren weitere *Packages* die zur Laufzeit nachgeladen werden können. Ausserdem ist es sowohl möglich Maple-Code nach C, Fortran und Java zu übersetzen als auch Funktionalität aus diesen Sprachen einzubinden. wie Die erste grafische Benutzerschnittstelle entstand 1989 und wurde 2003 von einem in Java geschriebenen abgelöst.

3.1.1 Wertzuweisungen

Für Wertzuweisungen sieht Maple den ":="Operator vor.

```
> u := 5;
```

$$u := 5 \tag{1}$$

```
> w := 6;
```

$$w := 6 \tag{2}$$

```
> u + w;
```

$$11 \tag{3}$$

Symbolische Ausdrücke werden erst auf Anforderung numerisch ausgewertet.

```
> h := sin( 3 * PI / 4 );
```

$$h := \sin\left(\frac{3}{4}\,\Pi\right) \tag{4}$$

```
> evalf(h);
```

$$\sin(0.7500000000\,\Pi) \tag{5}$$

3.1.2 Funktionen

```
> f := x -> 3x + x^2;
```

$$f := x \to 3x + x^2 \tag{6}$$

```
> f(3);
```

$$18 \tag{7}$$

3.1.3 Gleichungen

Gleichungen werden mit Hilfe des einfachen "="Zeichens dargestellt.

```
> a^2 + b^2 = c^2;
```

$$a^2 + b^2 = c^2 \tag{8}$$

```
> gl := a^2 + b^2 = c^2;
```

$$gl := a^2 + b^2 = c^2 \tag{9}$$

Mit dem Befehl *solve* wird versucht eine Gleichung (oder mehrere) zu lösen.

```
> solve(gl);
```

$$\left\{a = \sqrt{-b^2 + c^2},\ b = b,\ c = c\right\},\ \left\{a = -\sqrt{-b^2 + c^2},\ b = b,\ c = c\right\} \tag{10}$$

```
> solve(gl, c);
```

$$\sqrt{a^2 + b^2},\ -\sqrt{a^2 + b^2} \tag{11}$$

```
> solve(gl, [c]);
```

$$\left[\left[c = \sqrt{a^2 + b^2}\right], \left[c = -\sqrt{a^2 + b^2}\right]\right] \tag{12}$$

3.1.4 Datentypen

Maple ist schwach getypt. Um Typen zu überprüfen gibt es die Funktionen *type* und *whattype*.

```
> whattype(gl);
```

$$`=` \tag{13}$$

```
> whattype(1, gl);
```

$$exprseq \tag{14}$$

```
> type(u, integer);
```

$$true \tag{15}$$

```
> type(u, string);
```

$$false \tag{16}$$

Ausdrücke werden intern als Baumstruktur gespeichert. Zugriff auf die einzelnen Bestandteile bietet die Funktion *op(i..j, expr)*. Sie gibt die Operanden i bis j als Sequenz zurück.

```
> op(1..2, gl);
```

$$a^2 + b^2,\ c^2 \tag{17}$$

3.1.5 Definition von Prozeduren

Mit Hilfe von *proc* lassen sich in Maple Prozeduren definieren. Im Rumpf werden mit *local* alle lokalen Variablen deklariert. Mit *global* greift man auf globale Varialben zu.

```
> prozedur := proc(a, b)
    local c;
    c := a + b;
    return c;
  end proc;
```

$$prozedur := \mathbf{proc}(a, b)\ \mathbf{local}\ c;\ c := a + b;\ \mathbf{return}\ c\ \mathbf{end\ proc} \tag{18}$$

```
> prozedur(5, 6);
```

$$11 \tag{19}$$

Maple bietet an sich in erster Linie eine imperative Programmiersprache. Der Großteil von Maple, der in Maple selbst geschrieben ist, ist auch mit Hilfe dieser imperativen Strukturen implementiert. Allerdings unterstützt Maple auch die funktionale Programmierung

```
> map(x→2 x + 1, [1, 2]);
```

$$[3, 5] \quad (1)$$

3.2 Darstellung endlicher Körper in Maple

In Maple gibt es verschiedene Möglichkeiten mit endlichen Körpern zu rechnen. Speziell dafür ausgelegt ist das Paket GF (Galois Field) mit dem man beliebige endliche Körper mit p^k Elementen erzeugen kann, wobei p eine Primzahl ist. Einen Körper mit 3^5 Elementen erzeugt man etwa wie folgt

```
> GF243 := GF(3, 5);
```

$$GF243 := Z_3\,[T]\,\text{`/`}\,\langle T^5 + 2\,T^4 + 2\,T^3 + T^2 + 1\rangle \quad (1)$$

Als Repräsentation des Körpers liefert uns Maple die Darstellung als Restklassenring des Polynomrings über dem Primkörper. Das verwendete irreduzible Polynom vom Grad 5 wird im Konstruktor zufällig erstellt. Dieses kann auch explizit als dritter Parameter übergeben werden. Intern werden die Elemente als tatsächlich als Polynome in einer Variablen vom Grad kleiner k über dem Primkörper dargestellt *(modp1 Repräsentation)*. Man muss eine symbolische Darstellung eines Elements explizit konvertieren

```
> T := GF243:-variable:
> b := GF243:-ConvertIn(T^2 + T + 1);
```

$$b := (T^2 + T + 1) \bmod 3 \quad (1)$$

Leider muss man für die arithmetischen Operationen in diesen Körpern spezielle Funktionen wie etwa **G:-‘+‘** verwenden.

```
> b + b;
> GF243:-‘+‘(b, b);
```

$$(2\,T^2 + 2\,T + 2) \bmod 3 \quad (1)$$

```
> use GF243 in a := b^3; a + b; end use;
```

$$a := (2\,T^4 + T^3 + 2\,T^2 + 2\,T) \bmod 3$$
$$(2\,T^4 + T^3 + 1) \bmod 3 \quad (2)$$

```
> use GF243 in a·x + b; end use;
Error, (in *) modp1: invalid arguments to function Multiply
> a·x + b;
Error, invalid terms in product
```

Wie man oben sieht, ist es nicht möglich die Elemente in symbolischen Ausdrücken zur Repräsentation von Polynomen über diesem Körper zu verwenden.

Eine andere viel allgemeinere Möglichkeit ist es, mit algebraischen Elementen über dem Primkörper zu rechnen. Ein Element heißt dabei algebraisch über dem Körper K, wenn es Nullstelle eines Polynoms über K ist. Hat man etwa ein irreduzibles Polynom f gegeben, so haben wir oben gesehen, dass $x \bmod f \in \mathbb{F}_q[x]/\langle f\rangle$ eine Nullstelle des Polynoms f ist. Insbesondere wird die Körpererweiterung $\mathbb{F}_q \subset \mathbb{F}_q[x]/\langle f\rangle$ von dieser Nullstelle erzeugt, d.h. alle Elemente $\mathbb{F}_q[x]/\langle f\rangle$ lassen sich als polynomialer Ausdruck in $x \bmod f$ darstellen, also $\mathbb{F}_q[x \bmod f] = \mathbb{F}_q[x]/\langle f\rangle$. Man kann zeigen, dass der Erweiterungskörper $\mathbb{F}_q[\alpha]$ der durch *Adjunktion* einer Nullstelle α von f entsteht bis auf Isomorphie eindeutig bestimmt ist.

Wir können also jeden endlichen Erweiterungskörper durch eine beliebige Nullstelle eines irreduziblen Polynoms mit passendem Grad darstellen. In Maple sieht das wie folgt aus

```
> alias(α = RootOf(x^2 + x + 2)):
> alias(β = RootOf(x^2 + 2·x + 2)):
> Normal(α^2 + α + 2) mod 3;
```

$$0 \tag{1}$$

```
> Normal(α^2 + β^3) mod 3;
```

$$2\,\beta + 2\,\alpha + 2 \tag{2}$$

Man kann algebraische Elemente insbesondere dazu verwenden um Polynome im Erweiterungskörper darzustellen.

```
> f := x^2 + (α + 1)·x + α;
```

$$f := x^2 + (\alpha + 1)\,x + \alpha \tag{1}$$

```
> g := f·(x + α);
```

$$g := (x^2 + (\alpha + 1)\,x + \alpha)\,(x + \alpha) \tag{2}$$

```
> Normal(expand(%)) mod 3;
```

$$1 + 2\,\alpha + x^3 + (\alpha + 1)\,x + (1 + 2\,\alpha)\,x^2 \tag{3}$$

3.3 Faktorisierung in Maple

Maple bietet eine Vielzahl an Operationen für das Rechnen mit Polynomen mit algebraischen Koeffizienten, inbesondere für algebraische Koeffizienten über endlichen Körpern. Man hat etwa Operationen für die Berechnung der Polynomdivision sowie eines größten gemeinsamen Teilers.

```
> Quo(f, x + 1, x) mod 3;
```

$$x + \alpha \tag{1}$$

```
> Rem(g, x + 2, x) mod 3;
```

$$1 + 2\,\alpha \tag{2}$$

```
> Gcd(f, g) mod 3;
```

$$x^2 + x\,\alpha + x + \alpha \tag{3}$$

Natürlichen existieren auch Operationen für die Faktorisierung von Polynomen über endlichen Körpern.

```
> Factor(x^4 + 1) mod 3;
```

$$(x^2 + x + 2)\,(x^2 + 2\,x + 2) \tag{1}$$

```
> Factor(x^4 + 1, β) mod 3;
```

$$(x + \beta + 2)\,(x + 2\,\beta + 1)\,(x + 2\,\beta)\,(x + \beta) \tag{2}$$

Man findet auch weitere Funktionen für die Berechnung der quadratfreien sowie der verschiedengradige Zerlegung und auch für das randomisierte aufspalten von Polynomen.

> $Sqrfree(x\cdot(x+1)^3\cdot(x^2+x+2))$ **mod** 3;

$$[1, [[x^3+x^2+2x, 1], [x+1, 3]]] \quad \textbf{(1)}$$

> $DistDeg(x\cdot(x+1)\cdot(x^2+x+2)\cdot(x^3-x+1), x)$ **mod** 3;

$$[[x^2+x, 1], [x^2+x+2, 2], [x^3+2x+1, 3]] \quad \textbf{(2)}$$

> $ProbSplit(x^2+x+2, 1, x, \alpha)$ **mod** 3;

$$\{x+2\alpha, x+\alpha+1\} \quad \textbf{(3)}$$

4 Verschiedengradige Faktorisierung

Wie bereits in der Einleitung erwähnt, ist es sinnvoller, mit der verschiedengradigen Faktorisierung zu beginnen. In diesem Abschnitt werden normierte, quadratfreie Polynome als Eingabe erwartet.

4.1 Funktionsweise des Algorithmus

Definition 4.1.1. *Sei $f \in \mathbb{F}_q[X]$ ein nichtkonstantes Polynom über dem Körper K. Wir nennen $(g_1, \ldots, g_s)$ die verschiedengradige Zerlegung von f, falls gilt*

(i) $f = \prod_{i=1}^{k} g_i$.

(ii) $g_s \neq 1$.

(iii) Alle Primfaktoren von g_i haben Grad i.

Es ist g_i also das Produkt aller normierten, irreduziblen Polynome in $\mathbb{F}_q[x]$ vom Grad i, die f teilen. Es ist s der größte Grad eines irreduziblen Faktors von f. Es kann aber durchaus sein, dass $g_i = 1$ mit $i \neq s$ ist. Für

$$x^5 + 3x^3 + 4x^2 + 3x^4 + 2 = (x+1)(x+2)(x^3 + x + 1) = (x^2 + 3x + 2)(x^3 + x + 1)$$

über $\mathbb{F}_5$ beispielsweise ergibt sich $(g_1, g_2, g_3) = (x^2 + 3x + 2, 1, x^3 + x + 1)$.

Wenn man nun für ein Eingabepolynom $f \in \mathbb{F}_q[x]$ $g = \gcd(x^q - x, f)$ berechnet, ist dies nach Satz 2.2.8 der erste Faktor der verschiedengradigen Faktorisierung, da $x^q - x$ das Produkt aller irreduziblen, normierten Linearfaktoren ist. Dieses gefundene $g \equiv g_1$ kann man nun aus f herausteilen, um schrittweise die weiteren Faktoren zu finden. Dies führt dann zu dem Algorithmus in Abbildung 4.1. Als Eingabepolynom erwartet dieser Maple-Code ein normiertes, quadratfreies Polynom und gibt eine verschiedengradige Faktorisierung aus.

```
> DistinctDegreeFactor := proc( poly :: polynom, x :: symbol, p :: ℕ, α := 1 )
      local f, g, h, i :: ℕ, k :: ℕ, q :: ℕ;
      k := AlgebraicDegree( α ); q := p^k;
      h_0 := x; f_0 := poly;
      for i while f_(i-1) ≠ 1 do
          h_i := Powmod( h_(i-1), q, poly, x ) mod p;
          g_i := Gcdex( h_i - x, f_(i-1), x ) mod p;
          f_i := Quo( f_(i-1), g_i, x ) mod p;
      end do;
      return [ seq( g_j, j = 1 .. i - 1 ) ];
  end proc:
```

Abbildung 4.1: Algorithmus zur verschiedengradigen Faktorisierung

Beispiel 4.1.2. Dieses Beispiel zeigt den Ablauf des Algorithmus mit dem Beispiel-Polynom $f = x^4 + 4x^2 + 3x^3 + x + 4 = (x^2 + 3x + 2)(x^2 + 2) = (x+1)(x+2)(x^2+2)$ im $\mathbb{F}_5$.

Initialisierung $i = 0$
$h_0 = x,\ f_0 = x^4 + 4x^2 + 3x^3 + x + 4$

1. Iteration $i = 1$
$h_1 = h_0^5 \textbf{ rem } f = x^5 \textbf{ rem } x^4 + 4x^2 + 3x^3 + x + 4 = x^2 + 4x + 2$
$g_1 = \gcd(h_1 - x, f_0) = \gcd(x^2 + 3x + 2, x^4 + 4x^2 + 3x^3 + x + 4) = x^2 + 3x + 2$
$f_1 = \frac{f_0}{g_1} = \frac{x^4+4x^2+3x^3+x+4}{x^2+3x+2} = x^2 + 2$

2. Iteration $i = 2$
$h_2 = h_1^5 \textbf{ rem } f = (x^2 + 4x + 2)^5 \textbf{ rem } x^4 + 4x^2 + 3x^3 + x + 4 = x$
$g_2 = \gcd(h_2 - x, f_1) = \gcd(0, x^2 + 2) = x^2 + 2$
$f_2 = \frac{f_1}{g_2} = \frac{x^2+2}{x^2+2} = 1 \rightarrow$ fertig

Wie man sieht wurde $x^2 + 2$ gefunden, $(x + 1)(x + 2)$ wird aber wie erwartet nur zusammenmultipliziert als $x^2 + 3x + 2$ gefunden.

4.2 Korrektheit und Laufzeitanalyse

Satz 4.2.1. *Der Algorithmus braucht $\mathcal{O}(s\mathbf{M}(n)\log(nq))$ Operationen in $\mathbb{F}_q$.*

Beweis. Sei $(G_0, \ldots, G_s)$ die verschiedengradige Zerlegung eines Polynoms $f \in \mathbb{F}_q[x]$. Für die Korrektheit zeigt man mit vollständiger Induktion nach i, dass

$$h_i \equiv x^{q^i} \bmod f, \qquad g_i = G_i \text{ für } i \geq 1 \qquad \text{und} \qquad f_i = G_{i+1} \cdots G_s.$$

Induktionsanfang:

Für $i = 0$ bzw. $i = 1$ ist die Behauptung trivial.

Induktionsannahme:

Sei nun die Behauptung für ein beliebiges aber festes i bewiesen.

Induktionsschritt $i \rightarrow i + 1$:

$$h_{i+1} \equiv h_i^q \equiv x^{q^{iq}} \equiv x^{q^{i+1}} \bmod f$$

Daraus folgt die erste Behauptung. Weiter folgt mit obiger Beobachtung, dass g_{i+1} Teiler von zwei bestimmten Polynomen ist:

$$g_{i+1} = \gcd(h_{i+1} - x, f_i) = \gcd(x^{q^{i+1}} - x, f_i)$$

g_{i+1} teilt also zum einen $f_i = G_{i+1} \cdots G_s$ und zum anderen nach Theorem 2.2.8 das Produkt aller normierten, irreduziblen Polynome in $\mathbb{F}_q[x]$, deren Grad $i + 1$ teilen. Folglich gilt $g_{i+1} = G_{i+1}$, womit die zweite Behauptung impliziert wird.

Für f_{i+1} gilt nun:

$$f_{i+1} = \frac{f_i}{g_{i+1}} = \frac{G_{i+1} \cdots G_s}{G_{i+1}} = G_{i+2} \cdots G_s$$

Damit folgt auch die dritte Behauptung und es ist klar, warum der Algorithmus exakt s Iterationen benötigt.

Für die Laufzeit ergibt sich hier mit Hilfe von Satz 2.3.4 und 2.3.5 für die Berechnung von

$$\begin{aligned} h_i &\rightarrow \mathcal{O}(\mathbf{M}(n)\log q)) \\ g_i &\rightarrow \mathcal{O}(\mathbf{M}(n)\log n)) \\ f_i &\rightarrow \mathcal{O}(\mathbf{M}(n))) \end{aligned}$$

Operationen. Da die Schleife s mal durchlaufen wird, ergibt sich also insgesamt für die Laufzeit $\mathcal{O}(s\mathbf{M}(n)\log(nq))$ Schritte. □

Anzumerken bleibt noch, dass der Algorithmus bereits stoppen kann, wenn der Grad von $f_i < 2(i+1)$ ist, da alle irreduziblen Faktoren von f_i mindestens einen Grad von $i+1$ haben.

5 Gleichgradiges Aufspalten und gleichgradige Faktorisierung

Da nun gezeigt wurde, wie Polynome in gleichgradige Teilpolynome zerlegt werden können, müssen nun noch diese Polynome ggf. weiter faktorisiert werden. Die irreduziblen Faktoren dieses Polynoms können nur den gleichen Grad haben, wobei das Ausgangspolynom normiert ist. Dieser Algorithmus wurde von Cantor und Zassenhaus in [3] vorgestellt.

5.1 Gleichgradiges Aufspalten

Es soll nun ein normiertes quadratfreies Polynom $f \in \mathbb{F}_q[x]$ vom Grad n faktorisiert werden. Dabei habe jeder Primfaktor von f den Grad d und f bestehe aus den $r := \frac{n}{d}$ Primfaktoren $f_1, \ldots, f_r$. Wir nehmen $r \geq 2$ an, denn andernfalls wäre f bereits irreduzibel. Wir wollen in diesem Abschnitt vorstellen, wie man f in 2 nichttriviale Teiler aufspaltet.

Da $\gcd(f_i, f_j) = 1$ für $i \neq j$ ist, liegt ein Ringisomorphismus nach dem chinesischen Restsatz vor

$$\chi : R = \mathbb{F}_q[x]/\langle f \rangle \to \mathbb{F}_q[x]/\langle f_1 \rangle \times \cdots \times \mathbb{F}_q[x]/\langle f_r \rangle =: R_1 \times \cdots \times R_r \ . \tag{5.1}$$

Für jedes $a \in \mathbb{F}_q[x]$ gilt somit

$$\chi(a \bmod f) = (a \bmod f_1, \ldots, a \bmod f_r)) =: (\chi_1(a), \ldots, \chi_r(a)) \ . \tag{5.2}$$

Wenn man nun ein $a \in \mathbb{F}_q[x]$ findet, $\chi_i(a) = 0 \bmod f_i$ für mindestens ein i und $\chi_j(a) \neq 0 \bmod f_j$ für mindestens ein j gilt, also falls $f_i \mid a$ und $f_j \nmid a$ gilt, so ist $\gcd(f, a)$ ein nichttrivialer Teiler von f. Wir nennen ein solches a ein Splittingpolynom.

Für eine ungerade Primzahlpotenz wollen wir ein randomisiertes Verfahren angeben, welches nach einem Splittingpolynom sucht, und dies mit hoher Wahrscheinlichkeit findet. Dazu untersuchen wir zunächst die Menge S aller Quadrate der Einheitengruppe

$$S := \{b^2 : b \in \mathbb{F}_{q'}^\times\}$$

eines endlichen Körpers $\mathbb{F}_{q'}$. Für die weiteren Überlegungen ist folgendes Lemma wichtig.

Lemma 5.1.1. *Sei q' eine ungerade Primzahlpotenz und sei S die Menge der Quadrate in $\mathbb{F}_{q'}^\times$. Dann gilt*

(i) $S \subseteq \mathbb{F}_{q'}^\times$ *ist eine Untergruppe der Ordnung* $(q'-1)/2$,

(ii) $a^{(q'-1)/2} = 1$ *für alle* $a \in S$ *und*

(iii) $a^{(q'-1)/2} = -1$ *für alle* $a \in \mathbb{F}_{q'}^\times \setminus S$.

Beweis. Es ist $\mathbb{F}_{q'}^\times$ zyklisch (vgl. [2, Satz 3.6.14]) wird also von einem Element $\sigma \in \mathbb{F}_{q'}$ der Ordnung $q'-1$ erzeugt. Es ist S als Bild von $\mathbb{F}_{q'}$ unter dem Homomorphismus $a \mapsto a^2$ wieder zyklisch und wird von dem Element σ^2 der Ordnung $(q'-1)/2$ erzeugt, woraus *(i)* folgt. Für $a \in S$ existiert $b \in \mathbb{F}_{q'}^\times$ mit $a = b^2$. Nun gilt $a^{(q'-1)/2} = b^{q'-1} = 1$ nach dem kleinen Fermatschen Satz 2.2.6, woraus *(ii)* folgt. Insbesondere liegt also S im Kern der Abbildung $\varphi : \mathbb{F}_{q'}^\times \to \mathbb{F}_{q'}^\times, a \mapsto a^{(q'-1)/2}$ und stimmt aus Mächtigkeitsgründen schon mit diesem überein. Es ist $\{1, \sigma^{(q'-1)/2}\}$ das Bild von φ, und nach dem kleinen Fermatschen Satz 2.2.6 ist $\sigma^{(q'-1)/2}$ Nullstelle von $x^2 - 1 = (x+1)(x-1)$ woraus $\sigma^{(q'-1)/2} = -1$ folgt *(iii)* folgt. □

Es sei nun eine ungerade Primzahlpotenz q gegeben und man setze $q' := q^d$ sowie $e := \frac{q'-1}{2}$. Für ein $b \in R_i^\times \simeq \mathbb{F}_{q'}^\times$ gilt nach diesem Lemma $b^e \in \{1, -1\}$, je nachdem ob b ein Quadrat ist oder nicht. Dabei sind beide Möglichkeiten gleich häufig anzutreffen. Insbesondere ist $b^e - 1 \in \{0, -2\}$ jeweils mit gleicher Häufigkeit. Wir wollen dies ausnutzen um ein Splittingpolynom zu finden.

Kehren wir wieder zum Ringisomorphismus $\chi : R \to R_1 \times \cdots \times R_r$ zurück. Es induziert χ einen Isomorphismus der Einheitengruppen

$$\chi : R^\times \to R_1^\times \times \cdots \times R_r^\times \quad . \tag{5.3}$$

Hat man nun ein $a \in \mathbb{F}_q[x]$ mit $\gcd(f, a) = 1$ gegeben, so folgt $(a \bmod f_i) \neq (0 \bmod f_i)$, also $(a \bmod f_i) \in R_i^\times$ für alle i und insbesondere $(a \bmod f) \in R^\times$ wegen (5.3). Wählt man also a mit $\deg a < \deg f$ gleichverteilt mit der Bedingung $\gcd(a, f) = 1$ aus, so ist a eine auf $R^\times$ gleichverteilte Zufallsgröße. Weiter sind die Projektionen $\chi_i(A)$ wegen (5.3) gleichverteilt auf $R_i^\times$ und unabhängig. Wegen $R_i \simeq \mathbb{F}_{q'}$ sind nach obigem Lemma 5.1.1 die $\chi_i(A^e - 1) = \chi_i(A)^e - 1$ gleichverteilt auf $\{0, -2\}$ und unabhängig.

Nun ist $a^e - 1$ ein Splittingpolynom, außer für den Fall, dass $\chi_1(a^e - 1) = \cdots = \chi_r(a^e - 1)$ gilt. Denn falls alle $\chi_i(a^e - 1) = 0$ sind, gilt $\gcd(a^e - 1, f) = f$, und falls alle $\chi_i(a^e - 1) = -2$ sind, gilt $\gcd(a^e - 1, f) = 1$. Diese beiden Fälle treten allerdings nur mit einer Wahrscheinlichkeit von $2 \cdot \left(\frac{1}{2}\right)^r = 2^{1-r} \leq \frac{1}{2}$ ein.

Funktionsweise des Algorithmus

Der Maple-Code in Abbildung 5.1 zeigt die Funktionsweise des Algorithmus. Als Eingabe wird auch hier ein quadratfreies normiertes Polynom $f \in \mathbb{F}_q[x]$ erwartet, das zusätzlich vom Grad ungleich null sein muss, und q muss eine Potenz einer ungeraden Primzahl p sein. Außerdem wird ein $d < n$ mit $d \mid n$ erwartet, so dass alle irreduziblen Faktoren von f vom Grad d sind. Schlägt das Aufspalten fehl, so wird eine 1 zurückgegeben im anderen Fall liefert die Prozedur einen nicht trivialen Faktor von f.

```
> EqualDegreeSplit := proc( f :: polynom, d :: ℕ, x :: name, p :: ℕ, α := 1 )
      local a, g, h, t, i :: ℕ, k :: ℕ, n :: ℕ, q :: ℕ;
      n := degree( f, x ); k := AlgebraicDegree( α ); q := p^k;
      a := RandomPolyDegreeLess( n, x, p, α );
      if degree( a, x ) < 1 then
          return 1;
      end if;
      g := Gcdex( f, a, x ) mod p;
      if g ≠ 1 then
          return g;
      end if;
      t := Powmod( a, ( q^d − 1 ) / 2, f, x ) mod p;
      h := Gcdex( t − 1, f, x ) mod p;
      return h;
  end proc:
```

Abbildung 5.1: Algorithmus zum gleichgradigen Zerteilen

Beispiel 5.1.2. Wir betrachten das Polynom

$$f = x^4 + 3x^3 + 3x^2 + 3x + 3 = (x^2 + 2x + 4)(x^2 + x + 2)$$

über $\mathbb{F}_5$. Falls die Wahl für a auf $x + 1$ fällt, so gilt

$$\begin{aligned} g &= \gcd(a,\ f) = \gcd(x + 1,\ x^4 + 3x^3 + 3x^2 + 3x + 3) = 1, \\ t &= a^{12} \textbf{ rem } f = (x + 1)^{12} \textbf{ rem } x^4 + 3x^3 + 3x^2 + 3x + 3 = 4 = -1, \\ h &= \gcd(t - 1, f) = 1 \ . \end{aligned}$$

Das Aufspalten schlägt also für diesen Fall fehl. Wird hingegen das Polynom x für a ausgewählt so gilt

$$\begin{aligned} g &= \gcd(a,\ f) = \gcd(x, x^4 + 3x^3 + 3x^2 + 3x + 3) = 1, \\ t &= a^{12} \textbf{ rem } f = x^{12} \textbf{ rem } x^4 + 3x^3 + 3x^2 + 3x + 3 = 3x^3 + 3x^2 + x + 4, \\ h &= \gcd(t - 1,\ f) = \gcd(3x^3 + 3x^2 + x + 3,\ x^4 + 3x^3 + 3x^2 + 3x + 3) = x^2 + 2x + 4 \ . \end{aligned}$$

Der Algorithmus hat somit die nichttrivialen Faktoren $x^2 + 2x + 4$ und $(x^4 + 3x^3 + 3x^2 + 3x + 3)/(x^2 + 2x + 4) = x^2 + x + 2$. Die folgende Abbildung 5.2 veranschaulicht das Verhalten des Algorithmus für verschiedene Wahlen von a (vgl. [8, Figure 14.5]).

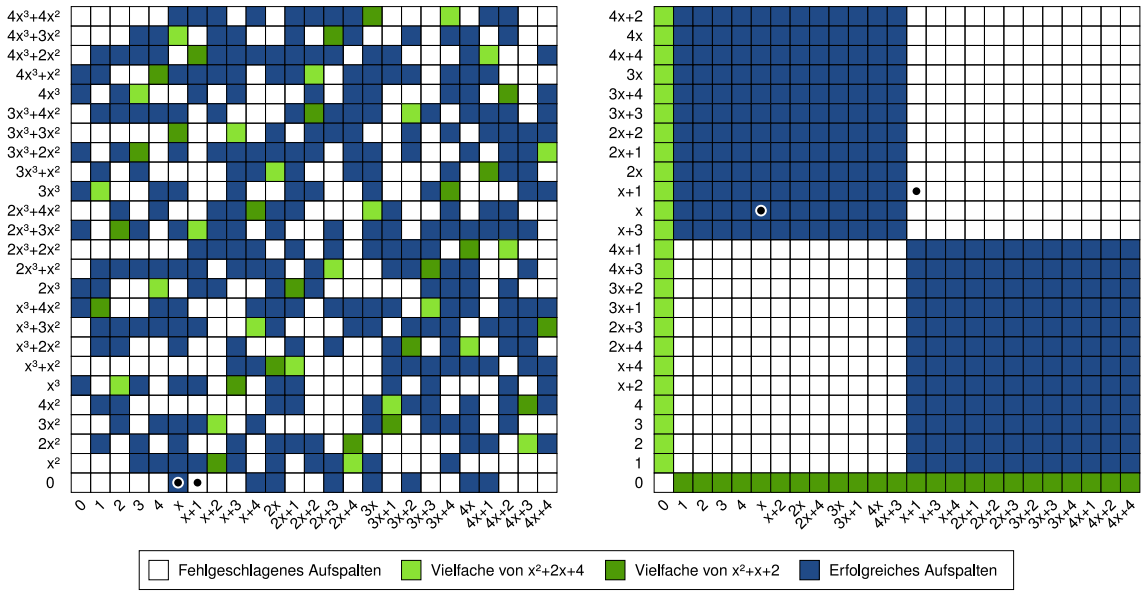


Abbildung 5.2: Verhalten des Algorithmus für $x^4 + 3x^3 + 3x^2 + 3x + 3 \in \mathbb{F}_5[x]$

Auf der linken Seite haben wir $R = \mathbb{F}_5[x]/\langle f \rangle$ bestehend aus den Polynomen $a_3x^3 + a_2x^2 + a_1x + a_0 \bmod f$ mit $a_i \in \mathbb{F}_5$. Die möglichen Werte für $a_1x + a_0$ sind entlang der horizontalen Achse und die möglichen Werte für $a_3x^3 + a_2x^2$ entlang der vertikalen Achse angeordnet. Unsere beiden Wahlen sind markiert. Nach dem chinesischen Restsatz haben wir

$$R = \mathbb{F}_5[x]/\langle f \rangle \simeq \mathbb{F}_5[x]/\langle x^2 + 2x + 4 \rangle \times \mathbb{F}_5[x]/\langle x^2 + x + 2 \rangle = R_1 \times R_2 \simeq \mathbb{F}_{25} \times \mathbb{F}_{25} \ .$$

Auf der rechten Seite sind die 25 Faktoren von R_1 auf der horizontalen Achse und die 25 Faktoren von R_2 auf der vertikalen Achse gegeben. Bei R_1 kommt zunächst die 0, dann

die Quadrate $1, 2, 3, 4, x, x+2, 2x, 2x+4, 3x, 3x+1, 4x, 4x+3$ und dann die Nichtquadrate. Es ist R_2 entsprechend dem Isomorphismus

$$(x \bmod x^2 + 2x + 4) \mapsto (x + 2 \bmod x^2 + x + 2)$$

angeordnet, so dass wirauch hier die zunächst die 0, dann die Quadrate und dann die Nichtquadrate haben.

Auf der rechten Seite sieht man genau was passiert. Ist nur einer der beiden Faktoren 0 so liefert der Algorithmus mit $\gcd(a, f)$ einen nicht trivialen Teiler. Trifft man zwei Quadrate, so landet man im unteren linken Teil und trifft man zwei Nichtquadrate, so landet man im oberen rechten Teil. Beide liefern keinen nicht trivialen Faktor. Landet man hingegen im oberen linken oder im unteren rechten Teil, so erhält man einen trivialen Faktor. Insbesondere sieht man anhand der Abbildung, dass die Wahrscheinlichkeit für einen Erfolg unter der Bedingung $\gcd(a, f)$ bei $\frac{1}{2}$ liegt.

Korrektheit und Laufzeitanalyse

Fehlschlag wird mit einer Wahrscheinlichkeit von weniger als $2^{1-r} \leq \frac{1}{2}$ zurückgegeben, wobei $r = \frac{n}{d} \geq 2$. Es werden $\mathcal{O}((d \log q + \log n)\mathbf{M}(n))$ Schritte durchgeführt.

Beweis. Für die Fehlschlagwahrscheinlichkeit wurde bereits oben gezeigt, dass sie bei $2^{1-r} \leq \frac{1}{2}$ liegt. Da der erste Schritt allerdings schon ein Teilpolynom finden kann liegt die Wahrscheinlichkeit sogar unter diesem Wert.

Für die Laufzeit ergibt sich hier mit Hilfe von Satz 2.3.4 und 2.3.5 für die Berechnung von

$$\begin{aligned} g &\rightarrow \mathcal{O}(\mathbf{M}(n) \log n)) \\ t &\rightarrow \mathcal{O}(\mathbf{M}(n) \log q^d + \mathbf{M}(n)) = \mathcal{O}(d\mathbf{M}(n) \log q) \\ h &\rightarrow \mathcal{O}(\mathbf{M}(n) \log n) \end{aligned}$$

Schritte. Insgesamt resultiert also eine Laufzeit von $\mathcal{O}((d \log q + \log n)\mathbf{M}(n))$.

□

Wenn man den Algorithmus k mal laufen lässt, sinkt die Fehlschlagwahrscheinlichkeit auf $2^{(1-r)k} \leq 2^{-k}$.

5.2 Gleichgradige Faktorisierung

Der vorherige Algorithmus gibt entweder einen *Fehlschlag* oder zwei Faktoren des Ausgangspolynom. Dies lässt sich leicht zu einem Algorithmus erweitern, der alle r Faktoren findet, wie der Maple-Code in Abbildung 5.3 zeigt.

Als Eingabe wird auch hier ein quadratfreies normiertes Polynom $f \in \mathbb{F}_q[x]$ erwartet, das zusätzlich vom Grad ungleich null sein muss und q muss eine Potenz einer ungeraden Primzahl sein. Außerdem wird ein $d < n$ erwartet mit $d \mid n$, so dass alle irreduziblen Faktoren von f vom Grad d sind. Die Ausgabe sind alle irreduziblen Faktoren des Grades d, die f teilen.

Beispiel 5.2.1. In diesem Beispiel soll das normierte quadratfreie Polynom

$$f = x^6 + 1 = (x^2 + 1)(x^2 + 2)(x^2 + 4) \in \mathbb{F}_7[x]$$

```
> EqualDegreeFactor := proc( f :: polynom, d :: ℕ, x :: name, p :: ℕ, α := 1 )
      local g, h;
      if degree( f, x ) = d then
          return { f };
      elif degree( f, x ) < 1 then
          return ∅;
      end if;
      g := 1;
      while degree( g ) < 1 or degree( g ) = degree( f ) do
          g := EqualDegreeSplit( f, d, x, p, α );
      end do;
      h := Quo( f, g, x ) mod p;
      return EqualDegreeFactor( g, d, x, p, α )
          ∪ EqualDegreeFactor( h, d, x, p, α );
  end proc:
```

Abbildung 5.3: Algorithmus zur gleichgradigen Faktorisierung

zerlegt werden. Der Grad $d = 2$ wird als bekannt vorausgesetzt und der Prozedur mit übergeben.

$\deg f = 6$ ist nicht $d = 2$, deshalb wird nicht abgebrochen. Anschließend wird das gleichgradige Zerteilen solange aufgerufen, bis ein geeignetes g gefunden wird. In diesem Fall wird $x^2 + 4$ gefunden und der Algorithmus läuft doppelt rekursiv mit der Eingabe $x^2 + 4$ und $(x^6 + 1)/(x^2 + 4) = x^4 + 3x^2 + 2$ weiter.

Die rekursive Verzweigung mit $x^2 + 4$ bricht sofort ab, da hier $degf = 2$ ist. Auf der anderen Seite wird mit $x^4 + 3x + 2$ weiter gerechnet. Schließlich wird ein $g = x^2 + 1$ gefunden und ein $h = (x^4 + 3x + 2)/(x^2 + 1) = x^2 + 2$ berechnet. Hier greift dann die Abbruchbedingung im nächsten rekursiven Aufruf und der Algorithmus terminiert.

Satz 5.2.2. *Ein quadratfreies Polynom $f \in \mathbb{F}_q[x]$ vom Grad $n = rd$ mit r irreduziblen Faktoren des Grades d lässt sich mit $\mathcal{O}\left(d \log q + \log n\right)\mathbf{M}(n) \log r)$ erwarteten Körperoperationen in $\mathbb{F}_q$ vollständig faktorisieren.*

Beweis. Um den Korrektheitsbeweis durchzuführen ist es hilfreich, sich einen typischen Verlauf des Algorithmus in einer Baumdarstellung anzuschauen, wie es Abbildung 5.4 zeigt. Ein Knoten stellt einen Faktor des Elternknoten dar, wobei alle Kindknoten zusammenmultipliziert den Elternknoten ergeben. Der Wurzelknoten ist das Ausgangspolynom $f = f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9$. Ein Knoten besitzt zwei Kindknoten, wenn das gleichgradige Zerteilen erfolgreich war. Andernfalls hat der Knoten nur ein Kindknoten – sich selbst. Das Zerteilen an einem Knoten mit Grad m benötigt $\mathcal{O}\left((d \log q + \log m)\mathbf{M}(m)\right)$ erwartete Körperoperationen. Bei einer gesamten Ebene des Baumes liegt wegen der Subadditivität 2.3.3 die Laufzeit in $\mathcal{O}((d \log q + \log n)\mathbf{M}(n))$ erwarteter Zeit. Gezeigt wird nun, dass die erwartete Höhe des Baumes in $\mathcal{O}(\log r)$ liegt, womit sich die geforderte Laufzeit ergibt.

Betrachtet man nun auf einer Höhe k des Baumes f_i und f_j mit $1 \leq i < j \leq r$, so gilt nach dem Chinesischen Restsatz, dass die Wahrscheinlichkeit, dass sowohl a mod f_i als a mod f_j Quadrate oder beide keine Quadrate sind, mindestens $\frac{1}{2}$ ist. In anderen Worten heißt das, dass die Wahrscheinlichkeit, dass ein Aufruf in Baumhöhe k f_i und f_j voneinander trennt – falls sie nicht ohnehin schon voneinander getrennt wurden - auch mindestens $\frac{1}{2}$ ist. Die Wahrscheinlichkeit, dass in Baumhöhe k im Aufruf-Baum f_i und

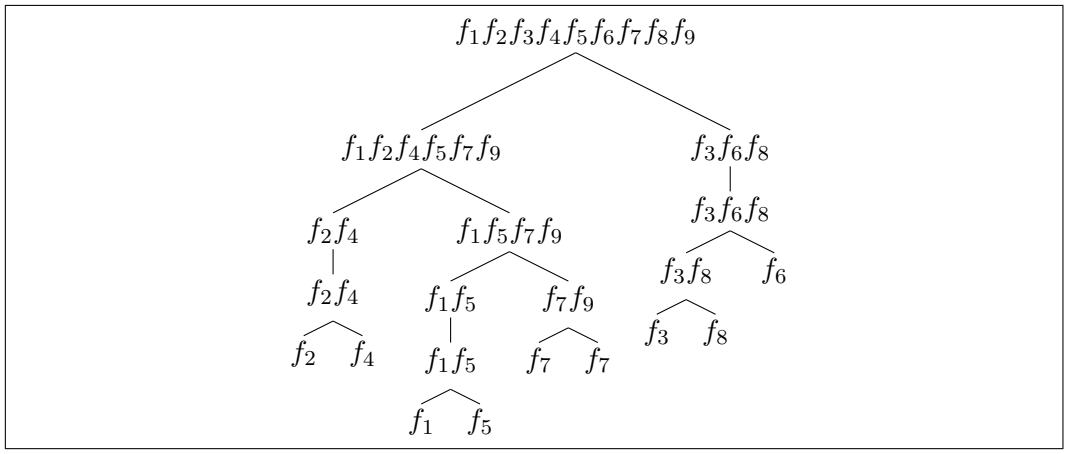


Abbildung 5.4: So könnte ein Aufruf-Baum der gleichgradigen Faktorisierung mit einem Polynom $f = f_1 \cdots f_9$ aussehen.

f_j noch nicht getrennt wurden, ist demnach höchstens $2^{\frac{1}{k}}$. Ferner gibt es nicht mehr als r^2 Paare von verschiedenen irreduziblen Faktoren. Die Wahrscheinlichkeit p_k, dass alle irreduziblen Faktoren auf Höhe k getrennt wurden, beträgt also höchstens $r^2 2^{\frac{1}{k}}$. Dies ist auch die Wahrscheinlichkeit dafür, dass die Baumhöhe größer ist als k. $p_{k-1} - p_k$ ist die Wahrscheinlichkeit, dass die Baumhöhe genau k ist. Für die erwartete Baumhöhe ergibt sich mit $s = 2\lceil \log_2 r \rceil$:

$$\begin{aligned}\sum_{k\geq 1} k(p_{k-1} - p_k) &= \sum_{k\geq 0} p_k = \sum_{0\leq k<s} p_k + \sum_{s\leq k} p_k \leq \sum_{0\leq k<s} 1 + \sum_{s\leq k} r^2 2^{\frac{1}{k}} \\ &= s + r^2 2^{\frac{1}{s}} \sum_{k\geq 0} 2^{\frac{1}{k}} \leq s + 2 \in \mathcal{O}(\log r)\end{aligned}$$

□

Es bleibt zu erwähnen, dass es auch möglich ist, den Algorithmus so anzupassen, dass er auch mit Charakteristik 2 korrekt arbeitet. Eine Modifikation von Algorithmus 5.1 mit Herleitung findet sich beispielsweise in [9, Übung 14.16].

6 Erster Faktorisierungsalgorithmus

An dieser Stelle wurden genug Vorbereitungen getroffen, um einen kompletten Faktorisierungsalgorithmus vorzustellen. Da nicht quadratfreie Eingangspolynome noch nicht behandelt wurden, wird eine Methode vorgestellt, die durch einfaches Ausprobieren die Vielfachheit der Faktoren findet.

6.1 Funktionsweise des Algorithmus

Der Algorithmus geht vor, wie man es vielleicht erwartet: Mit Hilfe der verschiedengradigen Faktorisierung wird das Eingabepolynom $f \in \mathbb{F}_q[x]$ in das quadratfreie Produkt $g = g_1 \cdots g_s$ zerlegt. Jedes g_i mit Grad i wird dann mit der gleichgradigen Faktorisierung in irreduzible Polynome zerlegt. Die Vielfachheit e in f eines Faktors g_j wird versuchsweise durch Divisionen ermittelt.

Der Maple-Code in Abbildung 6.1 zeigt die Funktionsweise. Als Eingabe wird ein nicht konstantes Polynom $f \in \mathbb{F}_q[x]$ erwartet, wobei q eine Potenz einer Primzahl ist. Die Ausgabe sind die normierten, irreduziblen Faktoren von f mit ihrer Vielfachheit.

```
> Factor1 := proc( poly :: polynom, x :: name, p :: ℕ, α :: algebraic := 1 )
      local f, g, h, j, G, T, i :: ℕ, k :: ℕ, m :: ℕ, q :: ℕ, r;
      k := AlgebraicDegree( α ); q := p^k;
      h_0 := x, f_0 := poly / lcoeff( poly );
      T := { };
      for i while f_(i-1) ≠ 1 do
          h_i := Powmod( h_(i-1), q, f_0, x ) mod p;
          g_i := Gcdex( h_i - x, f_(i-1), x ) mod p;
          f_i := Quo( f_(i-1), g_i, x ) mod p;
          if g_1 ≠ 1 then
              G := EqualDegreeFactor( g_i, i, x, p, α );
              for r in G do
                  for m while Divide( f_i, r, 'f_i' ) mod p do end do;
                  T := T ∪ { [r, m] };
              end do
          end if
      end do;
      return T;
  end proc:
```

Abbildung 6.1: Erster Algorithmus zur Faktorisierung von Polynomen über endlichen Körpern

Beispiel 6.1.1. Dieses Beispiel zeigt den Ablauf des Algorithmus mit dem Beispiel-Polynom

$$\begin{aligned} f &= 2x^{17} + 2x^{16} + x^{15} + x^{14} + 4x^{13} + 5x^{12} + x^{11} + 5x^9 + 5x^8 + x^6 + 4x^5 + x^4 \\ &= x^4(x+4)(x+2)(x+1)^3(x^2+2)(x^2+4)^3 \quad \in \mathbb{F}_7[x]. \end{aligned}$$

Initialisierung $i = 0$
$h_0 = x,\ f_0 = f$

1. Iteration $i = 1$
   $h_1 = h_0^7$ **rem** $f_0 = x^7$
   $g_1 = \gcd(h_1 - x, f_0) = \gcd(x^7 - x, f_0) = x^4 + x$
   $f_1 = \frac{f_0}{g_1} = x^{13} + 2x^{12} + x^{11} + 2x^9 + 4x^8 + x^7 + 5x^6 + x^5 + 4x^4 + 2x^3$

   Die gleichgradige Faktorisierung ergibt:
   $G = x, x + 1, x + 2, x + 4$
   Das Durchprobieren der Vielfachheit findet x viermal und $x + 1$ dreimal.

2. Iteration $i = 2$
   $h_2 = h_1^7$ **rem** $f = 6x^{16} + 4x^{15} + 3x^{14} + 2x^{13} + 5x^{12} + x^{11} + 2x^8 + 5x^7 + 6x^6 + 4x^5$
   $g_2 = \gcd(h_2 - x, f_1) = x^4 + 6x^2 + 1$ $f_2 = \frac{f_1}{g_2} = x^4 + x^2 + 2$
   Die gleichgradige Faktorisierung ergibt:
   $G = x^2 + 4, x^2 + 2$
   Das Durchprobieren der Vielfachheit findet $x^2 + 4$ dreimal.

Wie man sieht wurden alle Faktoren und deren Viellfachheit korrekt gefunden.

6.2 Korrektheit und Laufzeitanalyse

Der Algorithmus arbeitet korrekt und braucht erwartungsgemäß $\mathcal{O}(n\mathbf{M}(n)\log(qn))$ Schritte.

Beweis. Zu zeigen ist, dass beim Beginn der Schleife folgende Bedingungen erfüllt sein müssen:

$$h_i \equiv x^{q^i} \bmod f \qquad \text{und} \qquad v_i = lc(f) \prod_{deg f_k > i} f_k^{e_k}$$

$f = lc \prod_{1 \le i \le k} f_k^{e_i}$ sei die Faktorisierung von einem Polynom f mit den irreduziblen Faktoren $f_1, \cdots, f_k \in \mathbb{F}_q[x]$. $e_1, \cdots, e_k$ sind positive ganze Zahlen. Ein e_i stellt also die Vielfachheit eines Faktors f_i da.

Induktionsanfang:

Für $i = 0$ ist die Behauptung trivial.

Induktionsannahme:

Sei nun die Behauptung für ein beliebiges aber festes i bewiesen.

Induktionsschritt $i \to i + 1$:

Auch hier gilt analog wie in Abschnitt 4.2:

$$h_{i+1} \equiv h_i^q \equiv x^{q^{iq}} \equiv x^{q^{i+1}} \bmod f$$

Damit ist die erste Behauptung gezeigt. Weiter folgt mit obiger Beobachtung und nach Theorem 2.2.8:

$$g_{i+1} = \gcd(h_{i+1} - x, v_i) = \gcd(x^{q^{i+1}} - x, v_i) = \prod_{\deg f_k = i}$$

Da am Ende der Schleife alle $g_1 \cdots g_s$ mit der korrekten Vielfachheit entfernt werden gilt:

$$v_{i+1} = \frac{v_i}{f_{i+1}} = \frac{lc(f)\prod_{\deg f_k > i} f_k^{e_k}}{f_{i+1}} = lc(f) \prod_{\deg f_k > i+1} f_k^{e_k}$$

Damit folgt auch die zweite Behauptung.

Um die Laufzeit zur Berechnung der $g_1 \cdots g_s$ zu analysieren, werden einige Abschätzungen benötigt mit $m_i = \deg g$:

$$i \log(\frac{m_i}{i}) = m_i \frac{\log m_i i}{\frac{m_i}{i}} \leq m_i \qquad \text{und} \qquad \sum_i m_i \leq n$$

Und weiter folgt daraus:

$$\begin{aligned} \sum_i (i \log_2 q + \log_2 m_i)\mathbf{M}(m_i) \log_2(\frac{m_i}{i}) &\leq \sum_i (m_i \log_2 q + \log_2^2 m_i)\mathbf{M}(n) \\ &\in \mathcal{O}(n\mathbf{M}(n) \log_2 q). \end{aligned}$$

Für die Laufzeit ergibt sich dann mit Hilfe der Sätze 2.3.4 und 2.3.5, obiger Abschätzung und der erwartungsgemäßen Abschätzung der gleichgradigen Faktorisierung 5.3 für die Berechnung von

$$\begin{aligned} h_i &\rightarrow \mathcal{O}(\mathbf{M}(n) \log q + \mathbf{M}(n)) = \mathcal{O}(\mathbf{M}(n) \log q) \\ g &\rightarrow \mathcal{O}(\mathbf{M}(n) \log n) \\ g_1 \cdots g_s &\rightarrow \mathcal{O}((i \log q + \log m_i)\mathbf{M}(m_i) \log(m_i/i)) = \mathcal{O}(n\mathbf{M}(n) \log q) \end{aligned}$$

Schritte. Die innere Schleife braucht in jeder Iteration $\mathcal{O}(e_j \mathbf{M}(n))$ Schritte insgesamt also $\mathcal{O}(n\mathbf{M}(n))$. Es dominiert demnach die Berechnung der verschiedengradigen Faktoren. Der komplette Algorithmus nimmt schließlich erwartungsgemäß $\mathcal{O}(n\mathbf{M}(n) \log(qn))$ Zeit in Anspruch.

□

7 Anwendung: Finden von Nullstellen

An dieser Stelle soll eine kleine Anwendung eingeschoben werden, um die Faktorisierungsalgorithmen besser zu motivieren. Das Finden von Nullstellen ist eine typische Anwendung für die Algorithmen.

7.1 Finden von Nullstellen über endliche Körper

Das Finden von Nullstellen über endliche Körper ist relativ simpel. Lediglich die linearen Faktoren $g = \gcd(x^q - x, f)$ von einem Polynom $f \in \mathbb{F}_q[x]$ müssen gefunden werden, die dann schließlich mit der gleichgradigen Faktorisierung getrennt werden können. Die Berechnung der kompletten verschiedengradigen Faktorisierung ist also überflüssig.

Der Maple-Code in Abbildung 7.1 zeigt den einfachen Algorithmus. Als Eingabe wird ein nicht konstantes Polynom $f \in \mathbb{F}_q[x]$ erwartet. Die Ausgabe sind die verschiedenen Nullstellen von f.

```
> Polyroots := proc( f :: polynom, x :: name, p :: ℕ, α :: algebraic := 1 )
      local g :: polynom, k :: ℕ, q :: ℕ, t;
      k := AlgebraicDegree( α ); q := p^k;
      g := Powmod( x, q, f, x ) mod p;
      g := Gcdex( g - x, f, x ) mod p;
      if g ≠ 1 then
          return map( h → x - h, EqualDegreeFactor( g, 1, x, p, α ) );
      else
          return ∅;
      end if;
  end proc:
```

Abbildung 7.1: Algorithmus zum Finden von Nullstellen über endlichen Körpern

Beispiel 7.1.1. Als Beispiel sollen die Nullstellen des Polynoms

$$f = x^4 + 4x^2 + 3x^3 + x + 4 = (x^2 + 3x + 2) = (x + 1)(x + 2)(x^2 + 2) \in \mathbb{F}_5$$

gesucht werden:

$g = x^2 + 3x + 2$ findet der Algorithmus schnell, was als Eingabe für die gleichgradige Faktorisierung genutzt wird. Diese findet wiederum $g = (x + 1)(x + 2)$. Die Nullstellen liegen also bei -1 und -2.

7.1.1 Korrektheit und Laufzeitanalyse

Der Algorithmus arbeitet korrekt mit erwartungsgemäß $\mathcal{O}(\mathbf{M}(n) \log n \log(nq))$ Schritten.

Beweis. Die Berechnung von g läuft analog zum ersten Teil der verschiedengradigen Faktorisierung. Die Korrektheit und Laufzeit der gleichgradigen Faktorisierung wurde bereits gezeigt. Damit ist die Korrektheit und Laufzeit von diesem Algorithmus sofort klar. □

8 Quadratfreie Zerlegung

Wie zu Beginn dargestellt, kann die Faktorisierung vereinfacht werden, indem man das Problem der Faktorisierung beliebiger Polynome auf das Problem der Faktorisierung *quadratfreier* Polynome reduziert. Zur Erinnerung: Wir nennen ein Polynom $f \in K[x]$ über einem Körper K quadratfrei, falls $g^2 \nmid f$ für alle $g \in K[x] \setminus K$ gilt.

Definition 8.1.2. *Sei K ein Körper und $f \in K[x]$ ein Polynom.*

(i) Ist $f = \mathrm{lc}(f) \prod_{i=1}^{r} f_i^{e_i}$ eine Primfaktorzerlegung von f mit positiven Vielfachheiten $e_i \in \mathbb{N}$ und normierten, primen Polynomen f_i, so nennen wir $\prod_{i=1}^{r} f_i$ den quadratfreien Anteil von f.

(ii) Hat man eine (nicht notwendigerweise prime) Zerlegung $f = \mathrm{lc}(f) \prod_{i=1}^{k} g_i^i$ mit paarweise teilerfremden, normierten und quadratfreien Polynomen g_i, wobei $g_k \neq 1$ gelte, so nennen wir $(g_1, \ldots, g_k)$ die quadratfreie Zerlegung von f.

Das Bestimmen der quadratfreien Zerlegung entspricht dabei dem Zusammenfassen aller Primfaktoren, die mit Vielfachheit i auftauchen, zu einem Polynom g_i. Die Idee ist nun zunächst die quadratfreie Zerlegung $(g_1, \ldots, g_k)$ zu bestimmen und dann die quadratfreien Polynome g_i zu faktorisieren. Wir betrachten zwei Algorithmen zur quadratfreien Zerlegung (vgl. [16]). Wir beginnen mit einem sehr einfach zu verstehenden Algorithmus, der uns einen leichten Einstieg bietet, und den wir insbesondere dazu verwenden werden Probleme im Falle positiver Charakteristik aufzudecken und um Problemlösungen dafür zu entwickeln.

Zunächst benötigen wir einige technische Hilfsmittel für das vorliegen mehrfacher Primfaktoren.

Definition 8.1.3. *Sei K ein Körper. Es ist*

$$D : K[x] \to K[x], \quad \sum_{i=0}^{n} a_i x^i \mapsto \sum_{i=1}^{n} i \cdot a_i x^{i-1},$$

analog zu der Differentiation in der Analysis definiert. Wir schreiben statt $D(f)$ auch f' und nennen dies die formale Ableitung *von f.*

Diese etwas eigenartige Definition ist notwendig, da man in beliebigen Körpern, insbesondere in endlichen Körper, keine Limiten hat um eine *Ableitung* zu definieren. Wie man leicht nachrechnet erfüllt D die üblichen Rechenregeln für $a, b \in K$, $f, g \in K[x]$

$$D(af + bg) = aD(f) + bD(g), \quad D(fg) = fD(g) + gD(f).$$

Hat man nun ein nicht quadratfreies Polynom $f \in K[x]$, etwa $f = g^2 h$ gegeben, so gilt $f' = g(2g'h + gh')$ und man sieht, dass $g \mid \gcd(f, f')$ gilt. Wir wollen auch untersuchen mit welcher Vielfachheit ein Primfaktor auftritt. Sei dazu $f \in K[x]$ prim. Wir betrachten nun f^e. Es gilt $\gcd(f^e, D(f^e)) = \gcd(f, ef'f^{e-1}) = f^{e-1} \cdot \gcd(f, ef')$. Wir wissen, dass ef' über $\mathbb{Q}, \mathbb{R}$ oder $\mathbb{C}$ nicht verschwindet und der Grad um Eins kleiner ist als der Grad von f. Da f irreduzibel ist, kann es über diesen Körpern keine gemeinsamen Faktoren mit f' haben. In positiver Charakteristik kann dies aber passieren, und zwar genau dann wenn $ef' = 0$ gilt, also genau dann wenn $p \mid e$ oder $f' = 0$ gilt. Wir wollen den zweiten Fall über endlichen Körpern ausschließen.

Lemma 8.1.4. *Es sei $f = \sum_{i=0}^{n} a_i x^i$ mit $f' = 0$ über dem endlichen Körper $\mathbb{F}_{p^m}$. Dann ist f reduzibel und insbesondere existiert eine p-te Wurzel von f in $\mathbb{F}_{p^m}[x]$.*

Beweis. Es gilt $f' = \sum_{i=1}^{n} i \cdot a_i x^{i-1} = 0$ also folgt $a_i = 0$ für alle $p \nmid i$. Damit ist f bereits ein Polynom in x^p, d.h. es existiert $g \in K[x]$ mit $f = g(x^p) = \sum_{i=0}^{k} a_{ip} x^{ip}$. Dann gilt aufgrund der Eigenschaften des Frobenius-Homomorphismus $a \mapsto a^p$ und des kleinen Fermatschen Satzes 2.2.6

$$\left(\sum_{i=0}^{k} a_{ip}^{p^{m-1}} x^i\right)^p = \sum_{i=0}^{k} a_{ip}^{p^m} x^{ip} = \sum_{i=0}^{k} a_{ip} x^{ip} = f \ .$$

Es ist also f nicht irreduzibel und $\sum_{i=0}^{k} a_{ip}^{p^{m-1}} x^i$ eine p-te Wurzel. □

Definition 8.1.5. *Es sei K ein Körper. Wir nennen K* vollkommen, *falls $\gcd(f, f') = 1$ für jedes irreduzible Polynom $f \in K[x]$ gilt.*

Korollar 8.1.6. *Jeder endliche Körper ist vollkommen.*

Im folgenden gehen wir bei jedem Körper davon aus, dass er vollkommen ist. Betrachtet man allgemeiner zu einem normierten $f \in K[x]$ eine Primfaktorzerlegung $f = \prod_{i=1}^{r} f_i^{e_i}$ so gilt

$$f' = \sum_{i=1}^{r} e_i f_i' f_i^{e_i - 1} \prod_{\substack{j=1..r \\ j \neq i}} f_j^{e_j} \ . \tag{8.1}$$

Offenbar teilt $f_i^{e_i-1}$ jedes Produkt in der Summe und folglich gilt $f_i^{e_i-1} \mid f'$ und damit $f_i^{e_i-1} \mid \gcd(f, f')$. Falls $p \nmid e_i$ gilt, wobei $p := \operatorname{char} K \geq 0$ sei, so ist $e_i - 1$ sogar die tatsächliche Vielfachheit in $\gcd(f, f')$ aufgrund des nicht verschwindenden Summanden $e_i f_i^{e_i-1} \prod_{j=1..r, j\neq i} f_j^{e_j}$. Gilt allerdings $p \mid e_i$ so verschwindet dieser Summand, und es gilt $f_i^{e_i} \mid f'$. Es gilt also das folgende

Lemma 8.1.7. *Ist $f \in K[x]$ ein Polynom über dem vollkommenen Körper K der Charakteristik $p = \operatorname{char} K \geq 0$ mit der quadratfreien Zerlegung $(g_1, \ldots, g_k)$, dann gilt*

$$\gcd(f, f') = \prod_{\substack{j=2..k \\ p \nmid j}} g_j^{j-1} \cdot \prod_{\substack{j=1..k \\ p \mid j}} g_j^{j} \ . \tag{8.2}$$

Gilt $\operatorname{char} K = 0$ oder $\operatorname{char} K = p > 0$ mit $k < p$, so gilt

$$\gcd(f, f') = \prod_{j=2..k} g_j^{j-1}, \tag{8.3}$$

$$f / \gcd(f, f') = \prod_{j=1}^{k} g_j \ . \tag{8.4}$$

8.2 Algorithmus von Tobey und Horowitz

Anhand von (8.3), (8.4) sieht man, dass man mittels $\gcd(f, f')$ die Vielfachheiten der Primfaktoren um Eins dekrementieren kann. Dieses Ergebnis ist zentral für alle Algorithmen zur quadratfreien Faktorisierung. Insbesondere gibt uns $f / \gcd(f, f')$ den quadratfreien Anteil. Daraus lässt sich der folgende einfache Algorithmus zur Berechnung der quadratfreien Zerlegung konstruieren.

```
> TobeyHorowitz1 := proc( f :: polynom, x :: name, p :: ℕ, α :: algebraic := 1 )
      local c, d, h, i :: ℕ;
      c_1 := Gcdex( f, ∂/∂x f, x ) mod p;
      d_1 := Quo( f, c_1, x ) mod p;
      for i while d_i ≠ 1 do
          c_{i+1} := Gcdex( c_i, ∂/∂x c_i, x ) mod p;
          d_{i+1} := Quo( c_i, c_{i+1}, x ) mod p;
          h_i := Quo( d_i, d_{i+1}, x ) mod p;
      end do;
      return [ seq( h_j, j = 1 ..i - 1 ) ];
  end proc:
```

Abbildung 8.1: Algorithmus von Tobey und Horowitz zur quadratfreien Zerlegung

Beispiel 8.2.1. Wir schauen uns den Ablauf des Algorithmus 8.1 für das Polynom $f = x(x+1)^3$ über $\mathbb{F}_5$ an. Es gilt $k = 3 < 5 = \operatorname{char}\mathbb{F}_5$.

$$
\begin{aligned}
c_1 &= \gcd\left(x(x+1)^3, (x+1)^3 + 3x(x+1)^2\right) & c_3 &= \gcd(x+1,1) = 1\\
&= (x+1)^2 & d_3 &= (x+1)/1 = (x+1)\\
d_1 &= (x(x+1)^3)/(x(x+1)) & h_2 &= 1\\
c_2 &= \gcd\left((x+1)^2, 2(x+1)\right) = (x+1) & c_4 &= \gcd(1,0) = 1\\
d_2 &= (x+1)^2/(x+1) = (x+1) & d_4 &= 1/1 = 1\\
h_1 &= (x(x+1))/(x+1) = x & h_3 &= (x+1)
\end{aligned}
$$

$$\Rightarrow (h_1, h_2, h_3) = (x, 1, x+1)$$

Beispiel 8.2.2. Was passiert nun im Fall positiver Charakteristik $p > 0$ und $k \geq p$? Dazu schauen wir uns den Ablauf des Algorithmus für das Polynom

$$f = x(x^2+x+2)^3(x+1)^4(x+2)^5(x^2+2x+2)^6 = st^3u^4v^5w^6$$

über $\mathbb{F}_3$ an. Es gilt

$$
\begin{aligned}
f &= st^3u^4v^5w^6\\
c_1 &= \gcd(st^3u^3v^4w^6,\ s't^3u^4v^5w^6 + 4st^3u'u^3v^5w^6 + 5st^3u^4v'v^4w^6)\\
&= t^3u^3v^4w^6\\
d_1 &= (st^3u^4v^5w^6)/(t^3u^3v^4w^6) = suv\\
c_2 &= \gcd(t^3u^3v^4w^6,\ 4t^3u^3v'v^3w^6) = t^3u^3v^3w^6\\
d_2 &= v\\
h_1 &= su\\
c_3 &= \gcd(t^3u^3v^3w^6,\ 0) = t^3u^3v^3w^6\\
d_3 &= 1\\
h_2 &= v
\end{aligned}
$$

$$\Rightarrow (h_1, h_2) = (su, v)$$

Offenbar reduziert der Algorithmus die Vielfachheiten nur bis zum nächstkleineren Vielfachen von p. Inbesondere kann er g_i nicht von g_j trennen falls $i \equiv j \bmod p$ gilt. Seien q_j, r_j für $1 \leq j \leq k$ definiert durch $j = q_j p + r_j$ mit $0 \leq r_j < p$. Sei $\widetilde{g}_j := \prod_{r_i=j} g_i$ für $0 \leq j \leq p-1$, sei $\widetilde{f} := \prod_{j=1}^{p-1} \widetilde{g}_j^j$ und sei $R := \prod_{j=1}^{k} g_j^{pq_j} = f\widetilde{f}$ (vgl. [10]).

Satz 8.2.3. *Sei $f \in K[x]$ ein normiertes Polynom über dem vollkommenen Körper K der Charakteristik $p > 0$. Mit den obigen Notationen gilt: Der Algorithmus berechnet die quadratfreie Zerlegung $(\widetilde{g}_1, \ldots, \widetilde{g}_l)$ von $\widetilde{f}$ in $O(lM(n)\log n)$ Zeit. Insbesondere gilt $c_{l-1} = c_l = R$.*

Beweis. Wir verifizieren den Algorithmus mit den folgenden Invarianten

$$c_i = R \cdot \prod_{j=i+1}^{l} \widetilde{g}_j^{j-i}, \quad d_i = \prod_{j=i}^{k} \widetilde{g}_j, \quad h_i = \widetilde{g}_i.$$

Sei dazu $c_0 := f$ und $\tilde{c}_i := \frac{c_i}{R} = \prod_{j=i+1}^{l} \widetilde{g}_j^{j-i}$ und man beachte, dass $R' = 0$ sowie $l \leq p-1$ gilt. Es gilt für $i \geq 0$

$$\begin{aligned}
c_{i+1} &= \gcd(c_i, c_i') = \gcd\left(R \cdot \tilde{c}_i, R'\tilde{c}_i + R\tilde{c}_i'\right) \\
&= R \cdot \gcd\left(\tilde{c}_i, \tilde{c}_i'\right) \overset{(8.3)}{=} R \cdot \prod_{j=i+2}^{l} \widetilde{g}_j^{j-(i+1)} \\
d_{i+1} &= c_i / c_{i+1} = \left(R \cdot \prod_{j=i+1}^{l} \widetilde{g}_j^{j-i}\right) / \left(R \cdot \prod_{j=i+2}^{l} \widetilde{g}_j^{j-(i+1)}\right) \overset{\text{vgl.}(8.4)}{=} \prod_{j=i+1}^{l} \widetilde{g}_j \\
h_i &= d_i / d_{i+1} = \left(\prod_{j=i}^{l} \widetilde{g}_j\right) / \left(\prod_{j=i+1}^{l} \widetilde{g}_j\right) = \widetilde{g}_i.
\end{aligned}$$

Die Schleife terminiert nach l Iterationen, denn dann gilt $i = l+1$ sowie $d_{l+1} = \prod_{j=l+1}^{l} \widetilde{g}_j = 1$. Jede Iteration der Schleife lässt sich in $O(\mathbf{M}(n)\log n)$ Körperoperationen berechnen, so dass man eine Gesamtlaufzeit von $O(l\mathbf{M}(n)\log n)$ hat (vgl. [8]). □

8.3 Algorithmus von Yun

Wir betrachten nun eine Variation der Idee des obigen Algorithmus. Sei $f \in K[x]$ wieder ein Polynom über K mit quadratfreier Zerlegung $(g_1, \ldots, g_k)$ und es gelte $\operatorname{char} K = 0$ oder $k < \operatorname{char} K$. Mit $h := \gcd(f, f') = \prod_{j=2..k} g_j^{j-1}$ gilt

$$\begin{aligned}
d := \frac{f}{h} &= \prod_{j=1..k} g_j \\
\widetilde{d} := \frac{f'}{h} &= \sum_{i=1..k} i g_i' \prod_{\substack{j=1..k \\ j \neq i}} g_j \\
&= g_1' \prod_{j=2..k} g_j + \sum_{i=2..k} i g_i' \prod_{\substack{j=1..k \\ j \neq i}} g_j \\
&= g_1' \prod_{j=2..k} g_j + g_1 \sum_{i=2..k} i g_i' \prod_{\substack{j=2..k \\ j \neq i}} g_j
\end{aligned}$$

Wie man sieht gilt $g_1 \mid \widetilde{d} - g_1' \prod_{j=2..k} g_j$. Die Idee ist nun, diesen Summanden aus $\widetilde{d}$ zu entfernen und anschließend den größten gemeinsamen Teiler mit d zu bilden, um g_1 zu erhalten. Insbesondere wollen wir versuchen mit $\widetilde{d}$ weiterzurechnen, da dies einen viel kleineren Grad als $f/d = \prod_{j=2}^{k} g_j^{j-1}$. Es gilt weiter

$$d' = \left(\frac{f}{h}\right)' = g_1' \prod_{j=2..k} g_j + \sum_{i=2..k} g_i' \prod_{\substack{j=1..k \\ j\neq i}} g_j,$$

und damit erhält man

$$c := \widetilde{d} - \left(\frac{f}{h}\right)' = \sum_{i=2..k} (i-1) g_i' \prod_{\substack{j=1..k \\ j\neq i}} g_j = g_1 \left(\sum_{i=2..k} (i-1) g_i' \prod_{\substack{j=2..k \\ j\neq i}} g_j \right) .$$

Da die g_i paarweise teilerfremd sind und $g_j \nmid g_j'$ gilt, erhält man $\gcd(c,d) = g_1$. Weiter haben

$$d/g_1 = \prod_{i=2..k} g_i, \quad \text{und} \quad c/g_1 = \sum_{i=2..k} (i-1) g_i' \prod_{\substack{j=2..k \\ j\neq i}} g_j$$

die gleiche Gestalt wie d und $\widetilde{d}$. Wir scheinen dieses Verfahren also rekursiv anwenden zu können.

```
> Yun1 := proc( f :: polynom, x :: name, p :: ℕ, α :: algebraic := 1 )
      local c, d, df, g, h, i :: ℕ;
      df := ∂/∂x f;
      g := Gcdex( f, df, x ) mod p;
      d1 := Quo( f, g, x ) mod p;
      c1 := Quo( df, g, x ) - ∂/∂x d1 mod p;
      for i while di ≠ 1 do
          hi := Gcdex( ci, di, x ) mod p;
          di+1 := Quo( di, hi, x ) mod p;
          ci+1 := Quo( ci, hi, x ) - ∂/∂x di+1 mod p;
      end do;
      [ seq( hj, j = 1..i - 1 ) ];
  end proc:
```

Abbildung 8.2: Einfacher Algorithmus von Yun zur quadratfreien Zerlegung

Beispiel 8.3.1. Wir verfolgen die Ausführung des Algorithmus für das Polynom $f =$

$x(x+1)^3$ über $\mathbb{F}_5$.

$$\begin{aligned} g &= \gcd(x(x+1)^3,\ (x+1)^3+3x(x+1)^2) & d_2 &= x(x+1)/x = (x+1)\\ &= (x+1)^2 & c_2 &= 2x/x-1 = 1\\ d_1 &= (x(x+1)^3)/(x+1)^2 = x(x+1) & h_2 &= \gcd(x+1,\ 1) = 1\\ c_1 &= ((x+1)^3+3x(x+1)^2)/(x+1)^2 - d_1' & d_3 &= (x+1)/1 = x+1\\ &= (x+1)+3x-(x+1)-x & c_3 &= 1/1-1 = 0\\ &= 2x & h_3 &= \gcd(x+1,0) = x+1\\ h_1 &= \gcd(x(x+1),\ 2x) = x \end{aligned}$$

$$\Rightarrow (h_1,h_2,h_3) = (x,1,x+1)$$

Beispiel 8.3.2. Weiter betrachten wir auch diesen Algorithmus für das Polynom $f = x(x^2+x+2)^3(x+1)^4(x+2)^5(x^2+2x+2)^6 = st^3u^4v^5w^6$ über $\mathbb{F}_3$. Da die Vielfachheiten wieder über die formale Ableitung reduziert werden, erwarten wir wieder ein verfälschtes Ergebnis.

$$\begin{aligned} g &= \gcd(f,f') = t^3u^3v^4w^6\\ d_1 &= f/g = suv\\ c_1 &= f'/g - (d_1)'\\ &= (s'uv+4su'v+5suv') - (s'uv+su'v+suv')\\ &= suv'\\ h_1 &= \gcd(suv,suv') = su\\ d_2 &= (suv)/(su) = v\\ c_2 &= v'-v' = 0\\ h_2 &= \gcd(v,0) = v\\ d_3 &= v/v = 1\\ c_3 &= 0/v-0 = 0 \end{aligned}$$

$$\Rightarrow (h_1,h_2) = (su,v)$$

Zu bemerken ist, dass auch dieser Algorithmus die quadratfreie Zerlegung $\widetilde{g}_1,\ldots,\widetilde{g}_l$) von $\widetilde{f}$ berechnet. Weiterhin fällt das Restglied R sofort zu Beginn der Rechnung weg, so dass der Grad der Polynome sofort schlagartig reduziert wird. Nachteilig ist, dass das Restglied mittels $R = \frac{f}{\widetilde{f}} = \frac{f}{\prod_{j=1}^{l}\widetilde{g}_j^j}$ explizit berechnet werden muss.

Satz 8.3.3. *Sei $f \in K[x]$ ein normiertes Polynom über dem vollkommenen Körper K der Charakteristik $p>0$. Es seien $R, \widetilde{f}, (\widetilde{g}_1,\ldots,\widetilde{g}_l)$ wie oben definiert. Der einfache Algorithmus von Yun berechnet die quadratfreie Zerlegung von $\widetilde{f}$ in $O(M(n)\log n)$ Zeit.*

Beweis. Wir verifizieren den Algorithmus mit den folgenden Invarianten

$$c_i = \widetilde{g}_i \sum_{j=i+1}^{l}(j-i)\widetilde{g}_j' \prod_{\substack{k=i..l\\k\neq j}}\widetilde{g}_k, \qquad d_i = \prod_{j=i}^{l}\widetilde{g}_j, \qquad h_i = \widetilde{g}_i.$$

Wendet man obige Überlegungen auf $\widetilde{f}$ an, so erhält man

$$\begin{aligned}
\gcd(f,f') &= \gcd(R\cdot\widetilde{f}, R'\cdot\widetilde{f}+R\cdot\widetilde{f}') = R\cdot\gcd(\widetilde{f},\widetilde{f}')\\
d_1 &= f/\gcd(f,f') = \widetilde{f}/\gcd(\widetilde{f},\widetilde{f}') \overset{(8.4)}{=} \prod_{j=1}^{l}\widetilde{g}_j\\
c_1 &= f'/\gcd(f,f') - d_1' = \widetilde{f}'/\gcd(\widetilde{f},\widetilde{f}') - d_1'\\
&= \sum_{j=1}^{l} j\widetilde{g}_j' \prod_{\substack{k=1..l\\k\neq j}}\widetilde{g}_k - \sum_{j=1}^{l}\widetilde{g}_j' \prod_{\substack{k=1..l\\k\neq j}}\widetilde{g}_k\\
&= \sum_{j=2}^{l}(j-1)\widetilde{g}_j' \prod_{\substack{k=1..l\\k\neq j}}\widetilde{g}_k = \widetilde{g}_1\sum_{j=2}^{l}(j-1)\widetilde{g}_j' \prod_{\substack{k=2..l\\k\neq j}}\widetilde{g}_k\\
h_1 &= \gcd(c_i,d_i) = \widetilde{g}_i\\
d_{i+1} &= d_i/\gcd(c_i,d_i) = \prod_{j=i+1}^{l}\widetilde{g}_j\\
c_{i+1} &= c_i/\gcd(c_i,d_i) - d_{i+1}'\\
&= \sum_{j=i+1}^{l}(j-i)\widetilde{g}_j' \prod_{\substack{k=i+1..l\\k\neq j}}\widetilde{g}_k - \sum_{j=i+1}^{l}\widetilde{g}_j' \prod_{\substack{k=i+1..l\\k\neq j}}\widetilde{g}_k\\
&= \sum_{j=i+2}^{l}(j-(i+1))\widetilde{g}_j' \prod_{\substack{k=i+1..l\\k\neq j}}\widetilde{g}_k\\
&= g_{i+1}\sum_{j=i+2}^{l}(j-(i+1))\widetilde{g}_j' \prod_{\substack{k=i+2..l\\k\neq j}}\widetilde{g}_k
\end{aligned}$$

Der Algorithmus terminiert nach l Schritten, denn dann gilt $i = l+1$ und $d_{l+1} = 1$. Wir analysieren nun die Laufzeitkosten (vgl. [8]) mit Hilfe des Lemmas 2.3.4. Die Berechnung des ersten größten gemeinsamen Teilers benötigt $\mathcal{O}(M(n)\log n)$ Körperoperationen nach 2.3.4. Die beiden Divisionen sind mit $\mathcal{O}(n)$ Operationen realisierbar. Sei $u_i := \deg\widetilde{g}_i$ und sei $v_i := \deg d_i = \sum_{j=i}^{l} u_i$. Die Berechnung des größten gemeinsamen Teilers in der i-ten Iteration der Schleife benötigt $\mathcal{O}(M(v_i)\log v_i) \subset \mathcal{O}(M(v_i)\log n)$ Körperoperationen nach und die Berechnung der beiden Quotienten benötigt $\mathcal{O}(M(v_i))$ Operationen. Mit Hilfe der Subadditivität 2.3.3 erhält man

$$\begin{aligned}
\sum_{i=1}^{l} M(v_i) &\le M\left(\sum_{i=1}^{l} v_i\right) = M\left(\sum_{i=1}^{l}\sum_{j=i}^{l} u_j\right) = M\left(\sum_{j=1}^{l}\sum_{i=1}^{j} u_j\right)\\
&= M\left(\sum_{j=1}^{l} j u_j\right) \le M\left(\deg R + \sum_{j=1}^{l} j u_j\right) = M(n).
\end{aligned}$$

Es ist also $\mathcal{O}(M(n)\log n)$ eine obere Schranke für die Anzahl der Operationen. □

8.4 Quadratfreie Zerlegung in positiver Charakteristik

Wir betrachten im Folgenden einen endlichen Körper $\mathbb{F}_q$ der Charakteristik $p > 0$. Falls nun $R \neq 1$ gilt, ergibt sich die Frage, wie man die restlichen Elemente der quadratfreien Zerlegung aus dem Restglied $R = \prod_{j=1}^{k} g_j^{pq_j}$ berechnet. Wir würden gerne zu einer p-ten Wurzel $S = \prod_{j=1}^{k} g_j^{q_j}$ übergehen und eine quadratfreie Zerlegung von S rekursiv bestimmen [8][9][10]. Beachte, dass R ein Polynom in x^p ist und $R' = 0$ gilt. Falls $R = \sum_{i=0}^{n} b_i x^{ip}$ gilt, so liefert uns Lemma 8.1.4 die p-te Wurzel

$$\sum_{i=0}^{n} b_i^{q/p} x^i \quad .$$

Wie können wir nun aus einer quadratfreien Zerlegung $(s_1, \ldots, s_m)$ von S und der quadratfreien Zerlegung $(h_1, \ldots, h_l)$ von $\widetilde{f}$ wieder die quadratfreie Zerlegung von f bestimmen? Zur einfacheren Notation seien $h_{l+1}, \ldots, h_{p-1} := 1$ und $g_{k+1}, \ldots, g_{(m+1)p} := 1$.

Beachte, dass $s_i = \prod_{j=0}^{p-1} g_{ip+j}$ gilt. Für ein $g_j = g_{q_j p + r_j}$ folgt also $g_j \mid s_{q_j}$. Für $r_j \neq 0$ gilt $g_j \mid h_{r_j}$ und für $i \neq r_j$ gilt $g_j \nmid h_i$. Also gilt $g_j = \gcd(s_{q_j}, h_{r_j})$ für $r_j \neq 0$. Die restlichen g_j ergeben sich durch einfache Divisionen

$$g_{ip} = \frac{s_i}{\prod_{j=ip+1}^{(i+1)p-1} g_j}, \quad \text{für } 1 \leq i \leq l,$$
$$g_i = \frac{h_i}{\prod_{j=1}^{l} g_{jp+i}}, \quad \text{für } 1 \leq i \leq p-1 \quad .$$

Satz 8.4.1. *Sei $f \in \mathbb{F}_q[x]$ ein normiertes Polynom über dem Körper $\mathbb{F}_q$ der Charakteristik $p > 0$. Der erweiterte Algorithmus von Yun berechnet die quadratfreie Zerlegung von f in $\mathcal{O}(\mathbf{M}(n) \log n + n \log(q/p))$ Körperoperationen.*

Beweis. Die Korrektheit ergibt sich aus obigen Überlegungen und Satz 8.3.3. Wir betrachten hier deshalb nur die Laufzeit des Algorithmus (vgl. [9]). Es bezeichne $T(n)$ die Anzahl der Körperoperationen, die der Algorithmus zur Berechnung benötigt. Es kann R in $\mathcal{O}(\mathbf{M}(n)) \subset \mathcal{O}(M(n) \log n)$ Körperoperationen berechnet werden und S in $\mathcal{O}((n/p) \log(n/p))$ Körperoperationen mittels 2.3.5.

Um die gewünschte Laufzeit zu erreichen müssen wir die Rechnung ein wenig modifizieren. Aus der quadratfreien Zerlegung $(s_1, \ldots, s_l)$ von S berechnen wir zunächst $s_1 \cdot \ldots \cdot s_l$ vom Grad kleiner gleich $\lfloor n/p \rfloor$ in $\mathcal{O}(M(n/p) \log(n/p))$ Operationen. Anschließend reduzieren wir jedes h_i modulo s, welches nur im Fall $\deg h_i \geq \deg_s$ gemacht werden muss und dann in $\mathcal{O}(M(\deg h_i))$ Zeit möglich ist. Aus $\sum_{i=1}^{p-1} \deg_h i \leq n$ und der Subadditivität 2.3.3 von $\mathcal{O}(\mathbf{M}(n))$ folgt, dass dies insgesamt in $\mathcal{O}(M(n))$ Operationen möglich ist.

Für ein festes i kann man alle $\gcd(h_i, s_j) = \gcd(h_i \textbf{ rem } s, s_j)$, $1 \leq j \leq l$ nach Lemma 2.3.4*(iii)* in $\mathcal{O}(\mathbf{M}(n/p) \log(n/p)) \subset \mathcal{O}(\mathbf{M}(n/p) \log n)$ berechnen. Für alle i ist dies in $\mathcal{O}(\mathbf{M}(n) \log n)$ aufgrund der Superlinearität 2.3.3 von $\mathbf{M}$ möglich. Eine Division mit Dividend s_j kann ich $\mathcal{O}(M(\deg s_j))$ mit Dividend $\mathcal{O}(M(\deg h_i))$ in $\mathcal{O}(\mathbf{M}(\deg h_i))$ durchgeführt werden. Die Kosten für alle solche Divisionen liegt in $\mathcal{O}(\mathbf{M}(n))$ aufgrund der Subadditivität 2.3.3 von $\mathbf{M}$. Die Gesamtkosten ohne den rekursiven Aufruf liegen damit in $\mathcal{O}(\mathbf{M}(n) \log n + n \log(q/p))$. Es erfüllt $T(n)$ also die folgende Rekurrenz

$$T(n) = T\left(\frac{n}{p}\right) + \mathcal{O}(M(n) \log n + n \log(q/p)) \quad . \tag{8.5}$$

```
> Yun2 := proc( f :: polynom, x :: name, p :: ℕ, α :: algebraic := 1 )
      local g, h, i, j, k, l, s, R, S;
      h := Yun1( f, x, p, α );
      i := nops( h );
      R := Quo( f, ∏(j=1..i) (h_j)^j, x ) mod p;
      if degree( R ) ≤ 0 then
          return h;
      end if;
      S := PolyPthRoot( R, x, p, AlgebraicDegree( α ) );
      s := Yun2( S, x, p, α );
      l := nops( s );
      for i from i + 1 to p - 1 do
          h_i := 1;
      end do;
      for i to l do
          for j to p - 1 do
              g_(i·p+j) := Gcdex( h_j, s_i, x ) mod p;
              s_i := Quo( s_i, g_(i·p+j), x ) mod p;
          end do;
          g_(i·p) := s_i;
      end do;
      for j to p - 1 do
          g_j := Quo( h_j, ∏(ν=1..l) g_(ν·p+j), x ) mod p;
      end do;
      for k from (l + 1)·p - 1 by -1 while g_k = 1 do end do;
      return [ seq( g_j, j = 1..k ) ];
  end proc:
```

Abbildung 8.3: Erweiterter Algorithmus von Yun zur quadratfreien Zerlegung

Es sei c die versteckte Konstante in $\mathcal{O}\left(\mathbf{M}(n)\log n + n\log(q/p)\right)$ und es sei $d := pc/(p-1)$. Weiter sei $U(n) := \mathbf{M}(n)\log n + n\log(q/p)$. Es gilt

$$\begin{aligned} U\left(\frac{n}{p}\right) &\leq \frac{p}{p}\cdot\mathbf{M}\left(\frac{n}{p}\right)\log n + \frac{n}{p}\log(q/p) \\ &\leq \frac{1}{p}\left(\mathbf{M}\left(n\right)\log n + n\log(q/p)\right) = \frac{1}{p}U(n) \ . \end{aligned}$$

Wir zeigen nun $T(n) \leq dU(n)$ und damit $T(n) \in \mathcal{O}(U(n))$. Es gilt induktiv

$$\begin{aligned} T(n) &\leq T\left(\frac{n}{p}\right) + cU(n) \leq dU\left(\frac{n}{p}\right) + cU(n) \\ &\leq \left(\frac{d}{p} + c\right)U(n) = \left(\frac{d}{p} + \frac{d(p-1)}{p}\right)U(n) = dU(n) \ . \end{aligned}$$

□

9 Vollständiger Faktorisierungsalgorithmus

Wir haben nun alle Mittel zusammen um einen vollständigen Faktorisierungsalgorithmus in 3 Phasen zu entwickeln. Dieser ist in Abbildung 9.1 gezeigt.

```
> FinalFactor := proc( poly :: polynom, x :: name, p :: ℕ, α := 1 )
      local F, G, H, T, f, g, h, i, j, k, q;
      F := Yun2( poly/lcoeff( poly), x, p, α );
      T := ∅;
      i := 0;
      for f in F do
          i := i + 1;
          G := DistinctDegreeFactor( f, x, p, α );
          j := 0;
          for g in G do
              j := j + 1;
              if g ≠ 1 then
                  H := EqualDegreeFactor( g, j, x, p, α );
                  T := T ∪ map(h → [h, i], H);
              end if
          end do
      end do;
      return T;
  end proc:
```

Abbildung 9.1: Fertiger Faktorisierungsalgorithmus

Wir fassen die Ergebnisse der letzten Kapitel in einem Lemma zusammen und analysieren dann den obigen Algorithmus.

Lemma 9.0.2. *Es sei $f \in \mathbb{F}_{p^m}[X]$ ein Polynom vom Grad n über dem endlichen Körper $\mathbb{F}_{p^m}$ der Ordnung $q := p^m$. Es gilt*

(i) Es lässt sich f in $\mathcal{O}(M(n)\log n + n\log(q/p))$ Körperoperationen quadratfrei zerlegen.

(ii) Falls f quadratfrei ist, so lässt sich f in $\mathcal{O}(nM(n)\log(nq))$ Körperoperationen verschiedengradig zerlegen.

(iii) Falls f quadratfrei ist und aus $r = \frac{n}{d}$ verschiedenen irreduziblen Poynomen vom Grad d besteht, so lässt sich f in $\mathcal{O}((d\log q + \log n)M(n)\log r)$ Körperoperationen gleichgradig zerlegen.

Satz 9.0.3. *Der Algorithmus berechnet korrekt die Faktorisierung eines Polynoms $f \in K[X]$ über einen vollkommenen Körper K in $\mathcal{O}(nM(n)\log(nq))$ Körperoperationen.*

Beweis. Die Korrektheit ergibt sich mit den Sätzen 8.4.1, 4.2.1, 5.2.2. Hier nur die Laufzeitanalyse. Es sei $(g_1, \ldots, g_k)$ die quadratfreie Zerlegung von f und $(h_{i,1}, \ldots, h_{i,n_i})$ die verschiedengradige Faktorisierung von g_i. Die Iteration der inneren Schleife für $h_{i,j}$ benötigt $\mathcal{O}\left((j\log q + \log(\deg h_{i,j}))M(\deg h_{i,j})\log\left(\frac{\deg h_{i,j}}{j}\right)\right)$ Körperoperationen.

Für $x > 0$ gilt $\log x < x$ und damit

$$j\log\left(\frac{\deg h_{i,j}}{j}\right) = \deg h_{i,j}\frac{\log\left(\frac{\deg h_{i,j}}{j}\right)}{\frac{\deg h_{i,j}}{j}} \leq \deg h_{i,j}, \tag{9.1}$$

und damit erhält man für die Gesamtkosten der inneren Schleife für g_i unter Beachtung von $\sum_{j=1}^{n_i} \deg h_{i,j} \leq \deg g_i$.

$$\begin{aligned}
&\sum_{j=1}^{n_i} \left(j \log q + \log\left(\deg h_{i,j}\right)\right) \mathbf{M}(\deg h_{i,j}) \log\left(\frac{\deg h_{i,j}}{j}\right) \\
\leq &\sum_{j=1}^{n_i} \left(j \log\left(\frac{\deg h_{i,j}}{j}\right) \log q + \log^2(\deg h_{i,j})\right) \mathbf{M}(n) \\
\overset{(9.1)}{\leq} &\deg g_i M(n) \log q + \sum_{j=1}^{n_i} \log^2(\deg h_{i,j}) \mathbf{M}(n) \\
\in &\mathcal{O}(\deg g_i \mathbf{M}(n) \log q)
\end{aligned}$$

Wegen der letzten Abschätzung

$$\sum_{i=1}^{k} \deg g_i \mathbf{M}(n) \log q \leq n \mathbf{M}(n) \log q$$

erhält man die Schranke $\mathcal{O}(n\mathbf{M}(n) \log q)$ für die Anzahl der Körperoperationen für den gesamten Algorithmus. □

10 Irreduziblitätstest und Konstruktion irreduzibler Polynome

Wir wollen hier auf die Konstruktion irreduzibler Polynome über endlichen Körpern eingehen. Insbesondere braucht man irreduzible Polynome für die Konstruktion beliebiger endlicher Körper über den Primkörpern $\mathbb{F}_p$. Die nachfolgenden Sätze werden ohne Beweise angegeben. Man findet diese speziell in [5],[6],[8] oder in allgemeinerer Form in [1],[11].

Wir ziehen zunächst ein Korollar aus der Verallgemeinerung des kleinen Fermatschen Satzes 2.2.8.

Korollar 10.1.4. *Sei $f \in \mathbb{F}_q[X]$ ein Polynom vom Grad $n \geq 1$. Es ist f irreduzibel genau dann, wenn gilt*

(i) $f \mid X^{q^n} - X$,

(ii) $\gcd(X^{n/t} - X, f) = 1$ *für alle Primteiler t von n.*

Beweis. Die Hinrichtung folgt direkt aus Satz 2.2.8. Gilt umgekehrt (i) und (ii) so folgt aus (i) und Satz 2.2.8, dass der Grad jedes Primfaktors n teilt. Sei g so ein Primfaktor und angenommen es gilt $d := \deg g < n$. Sei t ein Primfaktor von n/d. Dann gilt $g \mid X^{n/t} - X$ im Widerspruch zu (ii). □

Daraus lässt sich sofort ein Algorithmus konstruieren, der auf Irreduzibilität testet. Die Laufzeit dieses Algorithmus liegt in $\mathcal{O}\left(\mathbf{M}(n)\log q + n^{(\omega+1)/2+\varepsilon}\right)$ (vgl. [8]) wobei ω eine reelle Zahl ist, so dass man zwei $n \times n$ Matrizen in Zeit $\mathcal{O}(n^\omega)$ multiplizieren kann. Mit $\omega = 2.376$ [4] ergibt sich eine Laufzeit von $\mathcal{O}(\mathbf{M}(n)\log q + n^{1.688+\varepsilon})$.

```
> TestIrreduc := proc(f, x, p, α := 1)
      local g, k, n, q, t;
      k := AlgebraicDegree(α); q := p^k; n := degree(f);
      if Powmod(x, q^n, f, x) mod p ≠ x then
          return false;
      end if;
      for t in numtheory[factorset](n) do
          g := Powmod(x, q^(n/t), f, x) mod p;
          if Gcdex(g − x, f, x) mod p ≠ 1 then
              return false;
          end if
      end do;
      return true;
  end proc:
```

Abbildung 10.1: Irreduzibilitätstest

Wir wollen im folgenden die Anzahl und insbesondere die relative Häufigkeit von irreduziblen Polynomen studieren. Ist q eine Primzahlpotenz, so bezeichnen wir mit $I(n,q)$ die Anzahl der irreduziblen normierten Polynome vom Grad n über $\mathbb{F}_q$. Es ist $p_n := \frac{I(n,q)}{q^n}$ die relative Häufigkeit der normierten irreduziblen Polynome unter den normierten Polynomen vom Grad n und dies ist insbesondere die Wahrscheinlichkeit dafür, dass ein normiertes Polynom vom Grad n irreduzibel ist, wenn man es gleichverteilt auswählt.

Satz 10.1.5. *Es sei q eine Primzahlpotenz und $n \in \mathbb{N}_{\geq 1}$. Dann gilt für $I(n,q)$*

$$\frac{q^n - 2q^{n/2}}{n} \leq I(n,q) \leq \frac{q^n}{n} \ . \tag{10.1}$$

Gilt insbesondere $q^n \geq 16$ so erhält man für p_n

$$\frac{1}{2n} \leq \frac{1}{n}\left(1 - \frac{2}{q^{n/2}}\right) \leq p_n \leq \frac{1}{n} \ . \tag{10.2}$$

Die Wahrscheinlichkeit p_n liegt also in etwa bei $\frac{1}{n}$. Um nun ein irreduzibles normiertes Polynom n-ten Grades zu finden, kann man ein beliebiges normiertes Polynom auswählen und dies aus Irreduzibilität testen. Im Mittel sollten wir damit nach $2n$ Wiederholungen ein irreduzibles Polynom gefunden haben. Es ergibt sich also mit Lemma 2.3.1 eine erwartete Laufzeit in $\mathcal{O}\left(n\mathbf{M}(n)\log q + n^{2.688+\varepsilon}\right)$. Der folgende Algorithmus ist wesentlich schneller.

```
> CreateIrreduc := proc(n, x, p, α := 1)
      local f, g, i, j, k, q;
      k := AlgebraicDegree(α); q := p^k;
      for j do
          f := RandomPolyDegreeEqual(n, x, p, α);
          for i to floor(n/2) do
              g := mod(Powmod(x, q^i, f, x), p);
              if mod(Gcdex(g − x, f, x), p) ≠ 1 then break; end if;
          end do;
          if i = floor(n/2) + 1 then return f; end if;
      end do;
  end proc:
```

Abbildung 10.2: Finden eines irreduziblen Polynoms

Lemma 10.1.6. *Es sei q eine Primzahlpotenz und $n \in \mathbb{N}$. Der Erwartungswert für den Grad des kleinsten Primfaktors eines gleichverteilt ausgewählten Polynoms vom Grad n liegt in $\mathcal{O}(\log n)$.*

Mit Satz 2.3.5 benötigt der Rumpf der innersten Schleife $\mathcal{O}\left(\mathbf{M}(n)\log(nq)\right)$ Körperoperationen. Die obigen Überlegungen würden eine erwartete Anzahl von $\mathcal{O}\left(n^2\mathbf{M}(n)\log(nq)\right)$ Körperoperationen implizieren. Da wir aber das Polynom gleichverteilt auswählen, erwarten wir mit Lemma 10.1.6 nur $\mathcal{O}\left(\log n\right)$ Iterationen für die innerste Schleife. Die erwartete Gesamtlaufzeit liegt damit bei $\mathcal{O}\left(n\mathbf{M}(n)\log(nq)\log n\right)$ Körperoperationen. Insbesondere können wir damit einen Erweiterungskörper $\mathbb{F}_{q^n}$ von $\mathbb{F}_q$ in $\mathcal{O}\left(n\mathbf{M}(n)\log(nq)\log n\right)$ erwarteter Zeit konstruieren.

Literatur

[1] Berlekamp, Elwyn R.: *Algebraic Coding Theory.* Aegean Park Press, 1984.

[2] Bosch, Siegfried: *Algebra.* Springer Berlin Heidelberg, 2006.

[3] Cantor, D. und H. Zassenhaus: *A New Algorithm for Factoring Polynomials over Finite Fields.* In: *Mathematics of Computation*, Band 36, Seiten 587–592, 1981.

[4] Coppersmith, Don und Shmuel Winograd: *Matrix multiplication via arithmetic progressions.* J. Symb. Comput., 9(3):251–280, 1990.

[5] Flajolet, P., X. Gourdon und D. Panario: *The complete analysis of a polynomial factorization algorithm over finite fields.* J. Algorithms, 40(1):37–81, 2001.

[6] Flajolet, Philippe, Xavier Gourdon und Daniel Panario: *Random Polynomials and Polynomial Factorization.* In: *ICALP '96: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, Seiten 232–243, London, UK, 1996. Springer-Verlag.

[7] Fürer, Martin: *Faster integer multiplication.* In: *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, Seiten 57–66, New York, NY, USA, 2007. ACM.

[8] Gathen, Joachim von zur und Jürgen Gerhard: *Modern computer algebra.* Cambridge University Press, New York, NY, USA, 1999.

[9] Gathen, Joachim von zur und Jürgen Gerhard: *Modern Computer Algebra - Solutions to selected exercises*, 1999–2003.

[10] Gianni, P. und B. Trager: *Square-free algorithms in positive characteristic.* Applicable Algebra in Engineering, Communication and Computing, 7:1–14, 1996.

[11] Knuth, Donald E.: *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[12] Schönhage, Arnold: *Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2.* Acta Informatica, 7:395–398, 1977.

[13] Schönhage, Arnorld und Volker Strassen: *Schnelle Multiplikation großer Zahlen.* Computing, 7:281–292, 1971.

[14] Schulze-Pillot, Rainer: *Elementare Algebra und Zahlentheorie.* Springer Berlin Heidelberg, 2007.

[15] Wikipedia: *Index calculus algorithm — Wikipedia, The Free Encyclopedia*, 2008. [Online; accessed 16-June-2008].

[16] Yun, David Y.Y.: *On square-free decomposition algorithms.* In: *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, Seiten 26–35, New York, NY, USA, 1976. ACM.