Algebra-Praktikum, Sommersemester 2004

Prof. Dr. U. Schoenwaelder, Dr. F. Lübeck

Einleitung

Sie dürfen eine beliebige Auswahl der folgenden Aufgabenblätter bearbeiten. Auch Teillösungen werden akzeptiert und gemäß dem angegebenen Punkteschema bewertet. Als Programmiersprachen werden Maple und GAP-4 empfohlen (ansonsten nach Absprache).

Zur Lösung einer Aufgabe gehört nicht nur ein lauffähiges, ausreichend getestetes Programm, sondern auch eine ausführliche Dokumentation. Diese soll nicht nur die einzelnen Schritte eines Algorithmus erläutern, sondern auch seinen mathematischen Hintergrund beschreiben. Ihrem Wesen nach soll diese Dokumentation also eine kurze schriftliche Seminararbeit sein. Sie kann in Form von Kommentaren in den Quellcode Ihres Programmes eingearbeitet sein.

Sie können, soweit in den Hinweisen zu den einzelnen Aufgaben angegeben, in GAP-4 oder Maple vorhandene Befehle verwenden, zum Beispiel die Befehle zum Berechnen von größten gemeinsamen Teilern in Euklidischen Ringen. Dies gilt natürlich nicht, wenn die Aufgabe gerade darin besteht, einen solchen speziellen Befehl zu implementieren. Falls Sie im Zweifel sind, was Sie verwenden dürfen und was nicht, fragen Sie uns bitte.

Es ist zulässig und wird sogar empfohlen, Aufgaben zu zweit zu bearbeiten.

Zum Erwerb eines Seminarscheins sind mindestens 100 Punkte aus mindestens 5 teilweise bearbeiteten Aufgaben erforderlich sowie ein 45-minütiger Vortrag zu einer der bearbeiteten Aufgaben.

In einigen Aufgaben finden Sie Formulierungen wie *„soweit Sie mit Ihrem Programm kommen“*. In solchen Fällen setzen wir einen Preis unserer Wahl für die Lösung aus, die am weitesten kommt. (Natürlich ist hier der Rechtsweg ausgeschlossen.)

Bei Fragen erreichen Sie uns im Sammelbau, Templergraben 64, Zimmer 202/228.
e-mail: Frank.Luebeck@Math.RWTH-Aachen.De
WWW: http://www.math.rwth-aachen.de/~AlgPrakSS04

Auf die folgenden Bücher wird in einigen Aufgaben verwiesen. Falls sie nicht in der Bibliothek zu finden sind, können sie bei uns ausgeliehen oder angesehen werden.

Außerdem sind sicher auch viele andere Algebra-Bücher nützlich, solche sollten Ihnen aus der Algebra-Vorlesung bekannt sein und sind hier nicht aufgeführt.

Literatur

[Coh93] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate texts in mathematics*. Springer-Verlag, 1993.

[GCL92] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer Academic Publishers, 1992.

[vdW67] B. L. van der Waerden. *Algebra*. Springer-Verlag, Berlin, Heidelberg, New York, 1967.

Aufgabe 1: Der Euklidische Algorithmus

Hintergrund: In dieser Aufgabe sollen Sie sich mit einem der grundlegendsten algebraischen Algorithmen beschäftigen, dem *Euklidischen Algorithmus*. Dieser ist die Grundlage für das Rechnen etwa mit ganzen Zahlen und Polynomen über Körpern. Der Euklidische Algorithmus berechnet zu zwei Elementen $a, b \in R$ aus einem euklidischen Ring R den größten gemeinsamen Teiler g. Der *Erweiterte Euklidische Algorithmus* liefert zusätzlich zwei Elemente $x, y \in R$ mit $g = xa + yb$. Dies finden Sie in den meisten Algebra-Büchern erklärt.

Hauptaufgabe: (a) Programmieren Sie den Euklidischen Algorithmus und den Erweiterten Euklidischen Algorithmus für die Ringe $R = \mathbb{Z}$ und $R = K[X]$, wobei K ein Körper ist. *5 Punkte*

(b) Informieren Sie sich in [GCL92, 7.7] über den *Heuristischen Euklidischen Algorithmus*, der (manchmal) den größten gemeinsamen Teiler zweier Polynome aus $\mathbb{Q}[X]$ berechnet. Der Einfachheit halber brauchen Sie den Text nur im Hinblick auf univariate Polynome anzusehen. Der entscheidende Satz ist [GCL92, Theorem 7.9] und der GCDHEU genannte Algorithmus. Programmieren Sie diesen Algorithmus, wobei Sie Ihr Programm aus (a) für ganze Zahlen verwenden. *10 Punkte*

Anwendungen: Vergleichen Sie Ihr Programm aus (b) mit Ihrem Programm aus (a) für Polynome an einigen Beispielen. *5 Punkte*

Sie können Ihre hier geschriebenen Programme für mehrere der anderen Aufgaben weiterverwenden.

Hinweise: In Teil (a) dürfen Sie die in den Systemen eingebauten Funktionen für Division mit Rest verwenden. *zusammen 20 Punkte*

Datei: aufg09.tex

Aufgabe 2: Pseudoprimzahlen

Hintergrund: Seit langer Zeit beschäftigen sich Leute mit der Frage, wie man natürliche Zahlen faktorisieren kann oder feststellen kann, ob eine Zahl Primzahl ist. Dies ist in den letzten Jahren eine auch in der Praxis sehr wichtige Fragestellung, da viele moderne Verschlüsselungsverfahren auf der Annahme beruhen, dass es sehr schwierig ist, Faktoren einer Zahl mit mehreren großen Primfaktoren zu finden.

Sei $p \in \mathbb{N}$ ungerade und $p - 1 = 2^e u$ mit ungeradem u. Für ein $a \in \mathbb{N}$, das teilerfremd zu p ist, sagen wir, dass p *starke Pseudoprimzahl zur Basis* a ist, wenn $a^u = 1 (\bmod p)$ oder $a^{2^s u} = -1 (\bmod p)$ für ein $0 \leq s < e$ ist.

Hauptaufgabe: (a) Begründen Sie, dass jede ungerade Primzahl p starke Pseudoprimzahl zu jeder Basis a mit $1 \leq a \leq p - 1$, ist. *2 Punkte*

(b) Schreiben Sie ein Programm, dass bei Eingabe von $p, a \in \mathbb{N}$ feststellt, ob p starke Pseudoprimzahl zur Basis a ist. *5 Punkte*

Anwendungen: (a) Finden Sie für $a \in \{2, 3, 5\}$ möglichst viele zusammengesetzte Zahlen, die starke Pseudoprimzahlen zur Basis a sind. *3 Punkte*

(b) Finden Sie einen Kandidaten für eine Primzahl mit 200 Dezimalstellen, indem Sie nach einer Zahl dieser Größenordnung suchen, die für jede von mindestens 25 Basen, die Sie ausprobieren, eine starke Pseudoprimzahl ist. *5 Punkte*

Hinweise: Nach einem Ergebnis von Miller und Rabin (siehe etwa *Rabin, M. O. "Probabilistic Algorithm for Testing Primality.", J. Number Th. 12, 128-138, 1980.*) gibt es zu einer ungeraden zusammengesetzten Zahl p mindestens $3/4 \cdot p$ verschiedene $1 \leq a \leq p - 1$, für die p keine Pseudoprimzahl zur Basis a ist. *zusammen 15 Punkte*

Aufgabe 3: Rechnen in algebraischen Körpererweiterungen

Hintergrund: Algebraische Körpererweiterungen lassen sich etwa durch Restklassenringe von Polynomringen realisieren. Informationen hierzu finden Sie in jedem Textbuch zur Algebra.

Hauptaufgabe: Sei K ein Körper, in dem wir bereits Elemente hinschreiben und mit diesen rechnen können, Sie dürfen der Einfachheit halber etwa $K = \mathbb{Q}$ annehmen. Sei $f(X)$ ein nicht-konstantes Polynom über K und $(f(X))$ das davon erzeugte Ideal im Polynomring $K[X]$.

(a) Überlegen Sie sich, wie Sie mit dem Computer Elemente im Restklassenring $R = K[X]/(f(X))$ darstellen können. *1 Punkt*

(b) Programmieren Sie eine Arithmetik zum Rechnen in R. Diese sollte Addition, Subtraktion, Multiplikation und den Test auf Gleichheit für je zwei Elemente aus R einschließen. (Das Polynom f können Sie entweder jeder Operation als Argument mitgeben, oder sie speichern es in der Datenstruktur für die Elemente aus dem Restklassenring.) *5 Punkte*

(c) Welche Elemente aus R haben ein Inverses bezüglich der Multiplikation? Schreiben Sie ein Programm, das von einem Element feststellt, ob es invertierbar ist, und gegebenenfalls das Inverse berechnet. Unter welcher Bedingung an f ist R ein Körper? *4 Punkte*

(d) Sei R ein Körper. Wie bestimmen Sie für ein beliebiges Element $x \in R$ sein Minimalpolynom über K? *5 Punkte*

Anwendungen: (a) Sei ζ_5 eine primitive 5-te Einheitswurzel über $\mathbb{Q}$. Wie können Sie mit Ihrem Programm im Körper $L := \mathbb{Q}(\zeta_5)$ rechnen? *2 Punkte*

(b) Der Körper L enthält die Quadratwurzeln $\pm\sqrt{5}$. Wie schreiben Sie in Ihrem Programm die Elemente ζ_5 und $\pm\sqrt{5}$, und können Sie entscheiden, welches Element $+\sqrt{5}$ ist? *2 Punkte*

(c) Wie lautet das Minimalpolynom von $\zeta_5 + \zeta_5^4$ über $\mathbb{Q}$? *3 Punkte*

(d) Sei $L(i)$ eine minimale Körpererweiterung von L, in der das Polynom $X^2 + 1 \in \mathbb{Q}[X]$ eine Nullstelle hat. Bestimmen Sie ein primitives Element der Erweiterung $L(i)/\mathbb{Q}$ und sein Minimalpolynom über $\mathbb{Q}$. *3 Punkte*

Hinweise: Wenn Sie die Programmierung in Maple ausführen möchten, könnten folgende Hilfeseiten nützlich sein: `polynomials`, `array`, `list`.

Entsprechende Informationen in GAP-4 sind unter `lists` und `polynomials and rational functions` zu finden.

Wenn Sie eine vorhandene Polynomarithmetik verwenden, dürfen Sie auch die darin vorhandenen Funktionen für größte gemeinsame Teiler und Division mit Rest benutzen. *zusammen 25 Punkte*

Datei: aufg01.tex

Aufgabe 4: Endliche Körper und Conway Polynome

Hintergrund: Zu jeder Primzahl p ist $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ ein endlicher Körper mit p Elementen. Zu jedem $q = p^n$, $n \in \mathbb{N}_{\geq 1}$, gibt es eine Körpererweiterung $\mathbb{F}_q/\mathbb{F}_p$ vom Grad n und diese ist bis auf Isomorphie eindeutig.

In der Praxis ist die Kenntnis von $\mathbb{F}_q$ „bis auf Isomorphie“ manchmal nicht ausreichend. Wenn man etwa Elemente von $\mathbb{F}_q$ hinschreiben will, so muss man zum Beispiel ein irreduzibles Polynom über $\mathbb{F}_p$ vom Grad n angeben, das die Körpererweiterung beschreibt. (Vergleiche Aufgabe 3.)

In dieser Aufgabe geht es darum, ganz bestimmte solcher Polynome - die *Conway Polynome* - zu finden. Sie erlauben es, Rechnungen in verschiedenen Körpererweiterungen von $\mathbb{F}_p$ durchzuführen, ohne sämtliche vorkommenden Elemente von Beginn an in einen gemeinsamen großen Körper einbetten zu müssen.

Hauptaufgabe: Wir schreiben die Elemente in $\mathbb{F}_p$ als $0, \ldots, p-1$ und legen durch $0 < 1 < \ldots < p-1$ eine Anordnung fest. Damit definieren wir auch eine Anordnung der Polynome vom Grad n über $\mathbb{F}_p$, $f = \sum_{i=0}^{n} a_i X^i \in \mathbb{F}_p[X]$, über den lexikographischen Vergleich der Koeffizientenlisten $[(-1)^n a_n, (-1)^{n-1} a_{n-1}, \ldots, a_0]$.

Das *Conway-Polynom* $\phi_n^p(X) \in \mathbb{F}_p[X]$ ist rekursiv definiert als das kleinste Polynom vom Grad n über $\mathbb{F}_p$, das

- normiert und irreduzibel ist,
- primitiv ist (das heißt, eine Nullstelle des Polynoms hat Ordnung $p^n - 1$, erzeugt also die multiplikative Gruppe von $\mathbb{F}_{p^n}$) und
- für jeden echten Teiler m von n gilt $\phi_m^p(X^{(p^n-1)/(p^m-1)}) = 0 (\bmod \phi_n^p(X))$ (das heißt, die passende Potenz einer Nullstelle von ϕ_n^p ist Nullstelle von ϕ_m^p).

Geben Sie eine andere Beschreibung der Conway-Polynome $\phi_1^p(X)$. Was haben diese mit primitiven Elementen in $\mathbb{F}_p$ zu tun? *1 Punkt*

Schreiben Sie ein Programm, das für gegebenes p und n das Conway-Polynom $\phi_n^p(X)$ ausgibt. *10 Punkte*

Wieso gibt es zu jedem p und n ein Conway-Polynom? *5 Punkte*

Anwendungen: Berechnen Sie die Conway-Polynome zu $p \in 2, 5, 1009$ und $n = 1, 2, \ldots$, soweit Sie kommen. *4 Punkte*

Hinweise: Die wichtigsten Tatsachen über endliche Körper, auch die Existenz primitiver Elemente, sind in fast jedem Algebra-Buch zu finden.

Für die Existenz der Conway-Polynome muß nur die Existenz irgendeines Polynoms mit den oben aufgeführten Eigenschaften gezeigt werden. Hierzu kann man zuerst ein primitives Elements von $\mathbb{F}_q$ betrachten und eine geeignete Potenz davon bestimmen, die dann die dritte Eigenschaft erfüllt. *zusammen 20 Punkte*

Datei: aufg10.tex

Aufgabe 5: Partitionen

Hintergrund: Wir bezeichnen hier mit $\mathbb{N}$ die Menge der natürlichen Zahlen ohne die Null. Seien $n \in \mathbb{N}$ und $M \subset \mathbb{N}$. Eine *M-Partition von n* ist eine Folge $m_1, m_2, \ldots, m_r$ mit:

1. $m_i \in M$, $i = 1, \ldots, r$
2. $m_1 \geq m_2 \geq \ldots \geq m_r$
3. $m_1 + m_2 + \ldots + m_r = n$

Wenn $M = \mathbb{N}$ ist , so spricht man auch einfacher von einer *Partition von n*.

Hauptaufgabe: (a) Schreiben Sie ein Programm, das zu gegebenem n und M sämtliche M-Partitionen von n ausgibt (M kann übrigens immer durch die endliche Menge $M \cap \{1, 2, \ldots, n\}$ ersetzt werden). *7 Punkte*

(b) Ein zweites Programm soll die Anzahl der M-Partitionen von n berechnen (und zwar auch noch für solche n, für die die Ausgabe des ersten Programms viel zu groß wäre). *7 Punkte*

Bei diesen Programmen sollen Sie auf Effizienz Wert legen. Wenn Sie mehrere Ideen für einen Algorithmus haben, probieren Sie diese aus und vergleichen sie.

Anwendungen: (a) Auf wieviele Arten läßt sich ein Geldbetrag von 10^i EUR mit den verbreiteten Münzen und Geldscheinen stückeln? ($i = 0$ bis soweit Sie mit Ihrem Programm kommen.) *2 Punkte*

(b) Sei p eine Primzahl und $n \in \mathbb{N}$. Wieviele paarweise nicht isomorphe abelsche Gruppen der Ordnung p^n gibt es? Bestimmen Sie eine Liste solcher Gruppen der Ordnung 17^{10}. Wieviele dieser Gruppen enthalten eine zyklische Untergruppe der Ordnung 17^3? Und wieviele paarweise nicht isomorphe Gruppen der Ordnung 13^{100} gibt es, die eine zyklische Gruppe der Ordnung 13^{90} enthalten? *4 Punkte*

Hinweise: In dieser Aufgabe können Sie die Verwendung und/oder die Vermeidung von *rekursiven* Funktionen studieren. Versuchen Sie zuerst, selbst eine Lösung zu finden. Danach können Sie auch untersuchen, ob das von Ihnen verwendete System bereits Lösungen dieser Aufgaben eingebaut hat und diese mit Ihrer Lösung vergleichen. *zusammen 20 Punkte*

Datei: aufg02.tex

Aufgabe 6: Polynom-Interpolation

Hintergrund: Sie wissen aus der Algebra, daß ein Polynom $f(X)$ vom Grad n über einem Körper K höchstens n Nullstellen hat. Daraus folgt, daß es zu $n+1$ paarweise verschiedenen Elementen $x_0, \ldots, x_n \in K$ sowie $n+1$ beliebigen Elementen $y_0, \ldots, y_n \in K$ genau ein Polynom $f(X) \in K[X]$ mit $f(x_i) = y_i$ für $i = 0, \ldots, n$ gibt, das höchstens den Grad n hat.

Hauptaufgabe: (a) Schreiben Sie ein Programm, das bei Eingabe von $x_0, \ldots, x_n$ und $y_0, \ldots, y_n$ wie oben, das Polynom f ausgibt. *5 Punkte*

(b) Wir betrachten die Funktionen $f_k : \mathbb{N} \to \mathbb{N}$ mit $f_k(n) := \sum_{i=1}^{n} i^k$. Sie kennen vermutlich die Übungsaufgabe, per Induktion die Gleichung $f_1(n) = \frac{1}{2}n(n+1)$ zu zeigen. Tatsächlich ist es richtig, daß sämtliche $f_k(n)$ Polynomfunktionen in n sind. Schreiben Sie ein Programm, mit dem Sie diese Polynomfunktionen für $k = 2, 3, \ldots$ finden und deren Richtigkeit beweisen können. Benutzen Sie hierzu das Programm aus (a). *5 Punkte*

Anwendungen: Berechnen Sie

$$\left(\sum_{i=1}^{100^{100}} i^{100} \right) \bmod 123456789.$$

5 Punkte

Hinweise: Falls Sie in (a) keine Idee haben, schauen Sie in Algebra-Büchern (oder in GAP oder Maple) unter dem Stichwort *Interpolation* nach. *zusammen 15 Punkte*

Datei: aufg05.tex

Aufgabe 7: Smith-Normalform und Hermite-Normalform

Hintergrund: Seien m und n positive natürliche Zahlen und A eine $m \times n$-Matrix über $\mathbb{Z}$. Dann kann A durch elementare Zeilen- und Spaltenumformungen in eine Matrix B der Gestalt

$$B = \left[\begin{array}{ccccc|c} e_1 & 0 & \cdots & 0 & 0 & \\ 0 & e_2 & \cdots & 0 & 0 & \\ 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & \cdots & e_{r-1} & 0 & \\ 0 & 0 & \cdots & 0 & e_r & \\ \hline & & 0 & & & 0 \end{array}\right] \tag{1}$$

übergeführt werden. Die Diagonaleinträge von B erfüllen $e_i \in \mathbb{Z}$ und $e_i > 0$ für $1 \leq i \leq r$, und $e_i \mid e_{i+1}$ für $1 \leq i < r$. Hierbei ist r eine nicht-negative ganze Zahl (wobei $r = 0$ natürlich genau dann gilt, wenn A die Nullmatrix ist). Die Matrix B mit diesen Eigenschaften ist eindeutig durch A bestimmt. Sie heißt die *Smith-Normalform* von A. Die Einträge $e_1, \ldots, e_r$ heißen die *Invariantenteiler* von A.

Durch eine Folge elementarer Spaltenumformungen läßt sich A in eine *reduzierte Spalten-Treppenmatrix* B überführen. Das soll heißen: Die ersten r Spalten von B sind alle ungleich 0, die restlichen Spalten sind gleich 0 (auch hier bedeutet $r = 0$ natürlich wieder, dass A die Nullmatrix ist). Der erste von 0 verschiedene Eintrag b_j in Spalte j ist positiv und steht tiefer als der entsprechende Eintrag in Spalte $j - 1$. Jeder Eintrag b links von b_j erfüllt $0 \leq b < b_j$ $(1 < j \leq r)$. Die Matrix B mit diesen Eigenschaften ist eindeutig durch A bestimmt. Sie heißt die *Hermite-Normalform* von A.

Als elementare Umformung sind hier zugelassen:
(1) Vertauschung von Spalten.
(2) Addition eines ganzzahligen Vielfachen einer Spalte zu einer anderen.
(3) Multiplikation einer Spalte mit -1.
(4) Im Fall der Smith-Normalform auch die analogen Zeilenumformungen.

Hauptaufgabe: Programmieren Sie Algorithmen zum Berechnen
(a) der Smith-Normalform, und *10 Punkte*
(b) der Hermite-Normalform *10 Punkte*
einer ganzzahligen Matrix. Experimentieren Sie dabei mit verschiedenen Strategien, die Einträge der Matrizen auszuräumen. Ziel ist es, möglichst wenig Schritte zu brauchen und die Zahlen in den während der Rechnung auftretenden Matrizen möglichst klein zu halten.

(c) Modifizieren Sie ihre Algorithmen, so dass mit den jeweiligen Normalformen auch die transformierenden Matrizen ausgerechnet werden. Im Fall (a) sollen also Matrizen $U \in GL_m(\mathbb{Z})$ und $V \in GL_n(\mathbb{Z})$ bestimmt werden, so dass UAV die Smith-Normalform von A ist. In (b) soll eine Matrix $U \in GL_m(\mathbb{Z})$ berechnet werden, so dass AU die Hermite-Normalform von A ist. *5 Punkte*

Anwendungen: (a) **Die Smith-Normalform**. Sei $A \in \mathbb{Z}^{m \times n}$. Der *Spaltenraum* $S(A)$ von A ist die Menge aller ganzzahligen Linearkombinationen der Spalten von A, also

$$S(A) := \{Az \mid z \in \mathbb{Z}^{n \times 1}\}.$$

Dies ist eine endlich erzeugte Untergruppe der freien abelschen Gruppe $\mathbb{Z}^{m\times 1}$, und als solche wieder frei und von endlichem Rang.

Die Faktorgruppe

$$CK(A) := \mathbb{Z}^{m\times 1}/S(A) \tag{2}$$

ist eine endlich erzeugte abelsche Gruppe. Jede endlich erzeugte abelsche Gruppe ist isomorph zu einer Gruppe der Form (2) mit geeigneten $m, n \in \mathbb{N}$ und $A \in \mathbb{Z}^{m\times n}$. Sind $e_1, \ldots, e_r$ die Invariantenteiler von A, dann ist

$$CK(A) \cong \mathbb{Z}/e_1\mathbb{Z} \oplus \mathbb{Z}/e_2\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}/e_r\mathbb{Z} \oplus \mathbb{Z}^{(m-r)\times 1}.$$

Sei q eine Primzahlpotenz und $\mathbb{F}_q$ der endliche Körper mit q Elementen. Sei weiter $0 < k \in \mathbb{N}$ und $n := (q^k - 1)/(q - 1)$. Die Zeilen und Spalten der $n \times n$-Matrix $A(k, q)$ seien indiziert mit den 1-dimensionalen Teilräumen von $V := \mathbb{F}_q^{1\times k}$. Sind $\langle v\rangle$ und $\langle w\rangle$ zwei solche Teilräume, dann sei der $(\langle v\rangle, \langle w\rangle)$-Eintrag a_{vw} in $A(k, q)$ definiert durch

$$a_{vw} := \begin{cases} 0, & \text{falls } v \cdot w \neq 0, \\ 1, & \text{falls } v \cdot w = 0. \end{cases}$$

Hierbei steht $v \cdot w$ für das Standard-Skalarprodukt zwischen v und w. Offensichtlich ist die Definition von a_{vw} unabhängig von den gewählten Repräsentanten der 1-dimensionalen Teilräume $\langle v\rangle$ bzw. $\langle w\rangle$. Die Matrizen $A(k, q)$ treten etwa in der Codierungstheorie auf.

Berechnen Sie die Invariantenteiler möglichst vieler der Matrizen $A(k, q)$. *10 Punkte*

(b) **Die Hermite-Normalform**. Sei $A \in \mathbb{Z}^{m\times n}$. Der *Rechts-Nullraum* $K(A)$ von A ist die Menge

$$K(A) := \{z \in \mathbb{Z}^{n\times 1} \mid Az = 0\},$$

mit anderen Worten, die Lösungsmenge des ganzzahligen homogenen linearen Gleichungssystems

$$Ax = 0.$$

Offensichtlich ist das ganzzahlige inhomogene lineare Gelichungssystem

$$Ax = z$$

genau dann lösbar, wenn z aus dem Spaltenraum von A ist. Zur Behandlung linearer ganzzahliger Gleichungssysteme ist also die Bestimmung des Nullraums und des Spaltenraums von A erforderlich. Da beide Räume freie abelsche Gruppen von endlichem Rang sind, genügt es jeweils, eine Basis dafür anzugeben. Genau dieses leistet die Hermite-Normalform einer Matrix.

Sei B die Hermite-Normalform von A und sei $U \in GL_n(\mathbb{Z})$ mit $B = AU$. Dann bilden die ersten r Spalten von B (mit r wie in der Definition der Hermite-Normalform) eine Basis von $S(A)$, und die Spalten mit den Nummern $r+1, \ldots, n$ von U eine Basis von $K(A)$. Wegen der Stufengestalt von B können Sie damit einem Element $z \in \mathbb{Z}^{m\times 1}$ sofort ansehen, ob es in $S(A)$ liegt. Können Sie diese Aussagen beweisen?

Für $0 < n \in \mathbb{N}$ sei $A(n)$ die folgende $(n+1) \times 2n$-Matrix über $\mathbb{Z}$:

$$A(n) := \begin{bmatrix} 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & \cdots \\ 2 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 2 & -1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 2 & -1 & -1 & 1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 2 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 0 & 0 & 0 & \cdots \\ 2 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & 0 & \cdots \\ 2 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Bestimmen Sie die Hermite-Normalform von $A(n)$ für n soweit Sie kommen. *10 Punkte*

Hinweise: Sie finden weitere Informationen zu diesem Thema in [Coh93, 2.4]. Extrem hilfreich könnte folgende Web-Adresse sein, insbesondere die Literaturangaben zu diesem Problemkreis:
http://www-history.mcs.st-and.ac.uk/~havas/. *zusammen 45 Punkte*

Datei: aufg18.tex

Aufgabe 8: Backtrack-Algorithmen und Gruppenoperationen

Hintergrund: Wenn Sie eine gute Lösung für die folgenden Aufgaben finden, haben Sie vermutlich *Backtrack*-Algorithmen programmiert. Dies ist eine wichtige Technik, bei der man zum Finden einer oder aller Lösungen einer Aufgabe einen „Suchbaum“ durchläuft und dabei möglichst früh erkennen möchte, welche „Äste“ man gar nicht weiter verfolgen muß.

Hauptaufgaben: (a) Auf ein Spielfeld mit $n \times n$ Schachbrettmuster sollen n Spielfiguren (ähnlich den Damen beim Schach) so verteilt werden, daß in jeder Zeile, Spalte, Haupt- und Nebendiagonale nur eine Figur steht. Sei D_8 die Gruppe der Spiegelungen und Drehungen der Spielfeldebene, die das Feld auf sich abbilden. Es ist klar, daß eine Lösung der Aufgabe nach Anwendung einer solchen Abbildung wieder eine Lösung ergibt. Solche Lösungen nennen wir *äquivalent*. Schreiben Sie ein Programm, das für gegebenes n aus jeder Äquivalenzklasse von Lösungen einen Repräsentanten ausgibt. *15 Punkte*

(b) Wir bezeichnen mit den Zahlen $\{1, \ldots, n\}$ die Ecken eines regelmäßigen n-Ecks gegen den Uhrzeigersinn und mit $\{i, j\}$ für $1 \leq i < j \leq n$ die Verbindungsstrecke zwischen den Ecken i und j. Ein *n-Graph* ist eine Teilmenge der Menge aller solcher Verbindungsstrecken. Die symmetrische Gruppe auf den n Eckpunkten induziert eine Operation auf der Menge der n-Graphen (jede Verbindungsstrecke zwischen zwei Punkten wird auf diejenige zwischen den zwei Bildpunkten abgebildet). Wir nennen zwei n-Graphen *isomorph*, wenn sie in derselben Bahn unter dieser Operation liegen. Schreiben Sie ein Programm, das für gegebenes n aus jeder Isomorphieklasse von n-Graphen einen Repräsentanten ausgibt. *15 Punkte*

Anwendungen: Erzeugen Sie mit den beiden Programmen eine Liste, die für $n = 1, 2, \ldots$ die Anzahlen der Äquivalenzklassen beziehungsweise der Isomorphieklassen angibt. Wie weit kommen Sie? Können Sie jeweils etwas über die vorkommenden Bahnlängen sagen? *zusammen 30 Punkte*

Datei: aufg04.tex

Aufgabe 9: Chinesischer Restsatz

Hintergrund: Seien $m_0, \ldots, m_n \in \mathbb{N}_{\geq 2}$ paarweise teilerfremd und seien $u_0, \ldots, u_n \in \mathbb{Z}$. Sei weiter $m := \prod_{k=0}^{n} m_k$ und R ein Repräsentantensystem von $\mathbb{Z}/m\mathbb{Z}$ (zum Beispiel $R = \{u \in \mathbb{Z} |\ 0 \leq u < m\}$). Der *chinesische Restsatz* besagt, daß es ein eindeutiges $u \in R$ mit $u \equiv u_k \bmod m_k$ für $0 \leq k \leq n$ gibt.

Dies bedeutet, daß eine ganze Zahl etwa dadurch bestimmt ist, daß für deren Betrag eine obere Schranke bekannt ist und deren Rest modulo genügend vieler paarweise teilerfremder Zahlen bekannt ist.

Hauptaufgabe: (a) Beschreiben Sie einen Algorithmus, der für gegebene m_i und u_i wie oben das eindeutige $u \in \mathbb{N}$ mit $0 \leq u < m$ liefert. Benutzen Sie hierzu folgende Darstellung von u:

$$u = b_0 + b_1 m_0 + b_2(m_0 m_1) + \cdots + b_n(m_0 \cdots m_{n-1})$$

mit $0 \leq b_k < m_k$ für $0 \leq k \leq n$.

(b) Programmieren Sie den Algorithmus. Was sollten Sie tun, wenn Sie das Programm mit festen $m_0, \ldots, m_n$ aber vielen Tupeln $u_0, \ldots, u_n$ aufrufen wollen? *10 Punkte*

Anwendungen: Sie haben vielleicht in anderen Aufgaben schon gesehen, daß es beim Rechnen mit ganzzahligen Matrizen (Normalformen, Gauß-Algorithmus, Berechnung der Determinante) das Problem der *Explosion der Einträge* gibt. Schreiben Sie ein Programm, das die Determinante einer ganzzahligen Matrix modulo *genügend vieler* (was heißt das hier?) nicht zu großer Primzahlen ausrechnet und dann die Ergebnisse mit dem Programm aus (b) zur richtigen Determinante der Matrix zusammensetzt. Auf diese Weise wird bis auf diesen letzten Schritt jedes Rechnen mit langen Zahlen vermieden. (Besonders interessant ist dieses Vorgehen natürlich, wenn man für die Determinante der betrachteten Matrix schon irgendwo her eine kleine Schranke kennt.) *10 Punkte*

Hinweise: Für die Anwendung dürfen Sie die in den Systemen eingebauten Funktionen für die Berechnung der Determinanten modulo Primzahlen verwenden, in GAP sind vielleicht die Hilfeseiten zu `Finite Field Elements` und `IntFFE` nützlich. *zusammen 20 Punkte*

Datei: aufg08.tex

Aufgabe 10: Symmetrische Polynome

Hintergrund: Sei $f(x_1, \ldots, x_n) \in R[x_1, \ldots, x_n]$ ein Polynom in n Unbestimmten über einem Ring R. Dann heißt f *symmetrisch*, falls für jede Permutation σ von $\{x_1, \ldots, x_n\}$ die Gleichheit $f(x_1, \ldots, x_n) = f(\sigma(x_1), \ldots, \sigma(x_n))$ gilt. Sei Z eine weitere Unbestimmte über R und

$$\prod_{i=1}^{n}(Z - x_i) = \sum_{j=0}^{n}(-1)^{n-j} s_{n-j}(x_1, \ldots, x_n) Z^j.$$

Dann heißt das Koeffizientenpolynom $s_j(x_1, \ldots, x_n)$ das *j-te elementarsymmetrische Polynom* in $R[x_1, \ldots, x_n]$. Der Hauptsatz über elementarsymmetrische Polynome besagt, daß es für jedes symmetrische Polynom $f(x_1, \ldots, x_n)$ ein Polynom $\tilde{f} \in R[x_1, \ldots, x_n]$ gibt, das nach Einsetzen von s_i für x_i gerade f ergibt; also $\tilde{f}(s_1, \ldots, s_n) = f(x_1, \ldots, x_n)$. Dies wird in zahlreichen Algebra-Büchern bewiesen.

Hauptaufgabe: Schreiben Sie ein Programm, das bei Eingabe eines symmetrischen Polynoms f das zugehörige Polynom $\tilde{f}$ ausgibt.

Darin sollten Sie für symmetrische Polynome eine platzsparende Datenstruktur verwenden. Bei der Operation der symmetrischen Gruppe auf den Variablen von f werden etwa Monome wieder auf Monome abgebildet. Es würde ausreichen, für ein symmetrisches Polynom immer nur Repräsentanten der Bahnen dieser Operation auf den Monomen abzuspeichern. *15 Punkte*

Anwendungen: Sei K ein algebraisch abgeschlossener Körper und

$$g(X) = X^n + \sum_{i=0}^{n-1} a_i X^i = \prod_{j=1}^{n}(X - \alpha_j) \in K[X].$$

Die Größe $\mathrm{discr}(g) = \prod_{i,j:1\le i<j\le n}(\alpha_i - \alpha_j)^2$ heißt *Diskriminante* von g. Verwenden Sie Ihr Programm, um für $n = 2, 3, \ldots$ die Diskriminante $\mathrm{discr}(g)$ als Polynom in den Koeffizienten a_i von g auszudrücken. *5 Punkte*

zusammen 20 Punkte

Datei: aufg03.tex

Aufgabe 11: Rechnen in Permutationsgruppen

Hintergrund: Hier sollen Sie einen Eindruck bekommen, warum man auf dem Computer gut mit endlichen Permutationsgruppen rechnen kann. Dieses sind Untergruppen einer endlichen symmetrischen Gruppe, die wir uns durch eine Menge von erzeugenden Permutationen gegeben denken. In GAP können Permutationen auf natürlichen Zahlen in Zykelschreibweise ein- und ausgegeben werden (siehe `?Permutations`) und es gibt zahlreiche Funktionen, die Berechnungen in und mit von solchen Elementen erzeugten Gruppen ausführen.

Hauptaufgabe: Sei M eine Menge von Permutationen von $\{1,\ldots,n\}$, G die davon erzeugte Gruppe und $p \in \{1,\ldots,n\}$.

(a) Schreiben Sie ein Programm, das aus gegebenem M und p die Bahn B des Punktes p unter der Operation von G, sowie für jeden Punkt $k \in B$ ein Element $x_k \in G$, das p auf k abbildet, ausgibt. *10 Punkte*

(b) Beweisen Sie Schreiers Lemma: In der Situation von (a) wird der Stabilisator von p in G erzeugt von den Elementen $x_k s x_{s(k)}^{-1}$ mit $k \in B$ und $s \in M$. Erweitern Sie dann das Programm in (a), so daß es zusätzlich Erzeugende des Stabilisators von p ausgibt. *5 Punkte*

(c) Sie können nun das Programm aus (a) und (b) auf die Erzeugerelemente des Stabilisators von p und einen anderen Punkt $p' \in \{1,\ldots,n\}$ anwenden und dies rekursiv fortsetzen, bis der Stabilisator nur noch aus dem trivialen Element `()` besteht. (Warum passiert das nach endlich vielen Schritten?) Die Folge der Ausgaben der obigen Programme heißt *Stabilisatorkette* von G. Schreiben Sie ein Programm, das aus gegebener Erzeugermenge M eine Stabilisatorkette von G ausgibt. *5 Punkte*

(d) Wie können Sie aus der Stabilisatorkette von G die Ordnung von G bestimmen? Wie können Sie von einer beliebigen Permutation von $\{1,\ldots,n\}$ feststellen, ob sie ein Element von G ist. *5 Punkte*

Anwendungen: (a) Wenden Sie das Programm für die Stabilisatorkette auf

$$M := \{(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23),$$
$$(3,17,10,7,9)(4,13,14,19,5)(8,18,11,12,23)(15,20,22,21,16)\}$$

an. Berechnen Sie die Ordnung der davon erzeugten Gruppe G und stellen Sie fest, ob die Transposition `(1,2)` in G liegt. *5+5 Punkte*

Hinweise: Sie haben sicher festgestellt, daß es beim obigen Verfahren ein Problem gibt: die Anzahl der Erzeuger für die Stabilisatoren in der Stabilisatorkette kann sehr stark wachsen. In Wirklichkeit genügen meist viel weniger Elemente, um die Stabilisatoren zu erzeugen. Es gibt viele technische Tricks, um das obige Verfahren, vor allem in diesem Punkt, zu verbessern, etwa den *Schreier-Sims Algorithmus*. Solche Optimierungen sind die Grundlage für viele effiziente Algorithmen für Permutationsgruppen in GAP. Interessante Beispiele sind etwa in

http://www.math.rwth-aachen.de/~GAP/WWW/Intro/rubik.html

zu finden. *zusammen 35 Punkte*

Datei: aufg06.tex

Aufgabe 12: Galoisgruppen

Hintergrund: Gegeben sei ein irreduzibles separables Polynom $f \in K[X]$ über einem Körper K. Dazu gibt es bis auf Isomorphie über K einen eindeutigen Zerfällungskörper L. Die Körpererweiterung L/K ist galoissch und wir bezeichnen ihre Galoisgruppe mit G_f. Hier sollen Sie lernen, daß es nicht so leicht ist, etwa im Fall $K = \mathbb{Q}$ aus gegebenem f die Gruppe G_f zu berechnen. Übrigens ist umgekehrt nicht bekannt, ob es zu einer beliebigen endlichen Gruppe G ein irreduzibles Polynom f über $\mathbb{Q}$ mit $G_f \cong G$ gibt.

Hauptaufgabe: (a) Benutzen Sie die Ideen aus [vdW67, I §66], um für $n = 2, 3, \ldots, 15$ ein *normiertes* irreduzibles Polynom $f(X) \in \mathbb{Z}[X]$ zu finden, dessen Galoisgruppe über $\mathbb{Q}$ isomorph zur symmetrischen Gruppe auf n Punkten ist.

Hierzu müssen Sie insbesondere irreduzible Polynome von passenden Graden über endlichen Körpern finden. *15 Punkte*

(b) Finden Sie mit Hilfe von GAP heraus, wieviele bis auf Isomorphie verschiedene Galoisgruppen es zu irreduziblen Polynomen vom Grad $n = 2, 3, \ldots, 15$ gibt - den Grundkörper K legen wir hier nicht fest. Welche Information, die Sie unter `??Library` in GAP finden, ist hier von Bedeutung? *5 Punkte*

(c) In Maple gibt es eine Funktion `galois`, die zu einem irreduziblen Polynom über $\mathbb{Q}$ vom Grad höchstens 8 die Galoisgruppe berechnet. Benutzen Sie diese Funktion, um für $n = 3, 4, 5$ und jeden möglichen Isomorphietyp von G_f ein Polynom f mit dieser Galoisgruppe zu finden. *10 Punkte*

Hinweise: Wenn Sie näheres zur Berechnung von Galoisgruppen wissen möchten, können Sie etwa in das Buch [Coh93] sehen. Dort wird zum Beispiel gezeigt, daß für ein irreduzibles Polynom $f \in K[X]$ die Gruppe G_f genau dann in der alternierenden Gruppe A_n liegt, wenn die Diskriminante von f ein Quadrat in K ist. *zusammen 30 Punkte*

Datei: aufg07.tex

Aufgabe 13: Quadratfreie Faktorisierung

Hintergrund: Sei R ein faktorieller Ring. Dann ist bekanntlich auch der Polynomring $R[X]$ faktoriell. Ein Polynom $0 \neq f \in R[X]$ heißt *quadratfrei*, wenn es nicht durch ein Polynom der Form g^2 mit $g \in R[X] \setminus R$ teilbar ist. Insbesondere sind konstante Polynome ungleich 0 quadratfrei.

Sei nun $0 \neq f \in R[X]$. Dann existieren quadratfreie Polynome $f_i \in R[X]$, $i = 1, \ldots, k$ mit $\mathrm{ggT}(f_i, f_j) = 1$ für $i \neq j$ und $f_k \notin R$, mit

$$f = \prod_{i=1}^{k} f_i^i. \tag{3}$$

Diese Produkt-Zerlegung heißt *quadratfreie Faktorisierung* von f.

Hauptaufgabe: Programmieren Sie Algorithmen zur quadratfreien Faktorisierung von Polynomen

(a) für $R = \mathbb{Q}$ und *10 Punkte*

(b) für den Fall, dass R ein endlicher Körper ist. *10 Punkte*

Die Algorithmen sollen bei gegebenem $0 \neq f \in R[X]$ die quadratfreien Faktoren aus der Zerlegung (3) finden. Wo liegen die Unterschiede in den beiden Fällen?

Anwendungen: Berechnen Sie die quadratfreien Faktorisierungen der

(a) charakteristischen Polynome aller 6×6 Permutationsmatrizen über $\mathbb{Q}$, sowie *2 Punkte*

(b) der Polynome aus (a) über $\mathbb{F}_3$ betrachtet, und *1 Punkte*

(c) der charakteristischen Polynome der Permutationsmatrizen zu folgenden Permutationen aus S_{28} über $\mathbb{Q}$:

$(1,2,3,4)(5,6,7,8,9,10,11,12,13,14,15,16)(17,18,19,20,21,22,23,24,25,26,27,28)$,
$(1,19,15,28,26,18,10,12,5,6,14,21,24,16,8,7)(2,23,25,9,22,17,27,4,13,11,3,20)$,
$(1,19,26,25,8,7,28,4,11,13,10,22,14,23,15,3)(2,24,9,12,20,16,27,18,21,17,5,6)$,
$(1,4,6,24,11,14,21,17,19,23,16,12,5,18,28,26,25,27,22,13,15,8,2,10,3,9)(7,20)$,
$(1,2,3)(4,5,6,7,8)(9,10,11,12,13)(14,15,16,17,18,19,20,21,22,23,24,25,26,27,28)$,
$(1,2)(3,4,5)(6,7,8,9)(10,11,12,13,14)(15,16,17,18,19,20)(21,22,23,24,25,26,27)$,
$(2,20,15,16,7,17,26,27,24,18,4,19,8,6,14,9,28,12,10,5)(3,23,13,22,11,25)$,
$(1,13,11,5,17,20,28,18,25,19,26,22,10,3,4,15,12,24,7,9)(2,14)(6,16,21,8,23)$,
$(1,2,3,4)(5,6,7,8,9,10,11,12)(13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28)$,
$(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28)$. *2 Punkte*

Die Permutationen stehen als GAP-lesbare Liste in der Datei:
`http://www.math.rwth-aachen.de/~AlgPrakSS04/perma16`

Hinweise: Die gesuchten Algorithmen beruhen auf einer Untersuchung des größten gemeinsamen Teilers eines Polynoms f und seiner formalen Ableitung $D(f)$. Sie finden die Algorithmen etwa in [GCL92, 8.2 und 8.3] beschrieben.

Sie dürfen die in Maple oder GAP existierenden Algorithmen zur Berechnung der größten gemeinsamen Teiler verwenden.

Bei den Anwendungen dürfen Sie die vorhandenen Programme zur Berechnung von charakteristischen Polynomen verwenden (in GAP-4 etwa der Befehl `CharacteristicPolynomial`).

Weiterhin gibt es Programme, die eine Permutation in Zykelschreibweise in eine Permutationsmatrix verwandeln (in GAP-4 etwa der Befehl `PermutationMat`). *zusammen 25 Punkte*

Datei: aufg16.tex

Aufgabe 14: Polynomfaktorisierung nach Kronecker

Hintergrund: Moderne Verfahren zur Faktorisierung von Polynomen über den ganzen (beziehungweise rationalen) Zahlen benutzen meist die Reduktion der Polynome zu solchen über endlichen Körpern, führen dort die Faktorisierung durch (siehe etwa Aufgabe 15) und *liften* dann das Ergebnis zurück zu Polynomen über den ganzen Zahlen.

Man kann aber nach Kronecker die Zerlegung auch finden, indem man zunächst für eine Unbestimmte Elemente des Koeffizientenringes einsetzt, die Ergebnisse faktorisiert (im Koeffizientenring) und daraus mit Polynominterpolation (siehe Aufgabe 6) die Faktorisierung des ursprünglichen Polynoms gewinnt.

Hauptaufgabe: Das oben angedeutete Verfahren ist ausführlich in [vdW67, §32] erklärt. Schreiben Sie ein Programm mit dem mittels dieses Kronecker-Verfahrens multivariate Polynome über den ganzen Zahlen faktorisiert werden können. *10 Punkte*

Anwendungen: (a) Lösen Sie die Aufgaben in [vdW67, §32] mit Hilfe Ihres Programms. *2 Punkte*

(b) Machen Sie einige Experimente mit Ihrem Programm. Sie können etwa Beispiel-Polynome durch Multiplikation erzeugen und dann das Wiederfinden der Faktoren mit Ihrem Programm und mit der in Maple eingebauten Faktorisierung vergleichen. *3 Punkte*

zusammen
15 Punkte

Datei: aufg12.tex

Aufgabe 15: Berlekamp-Algorithmus

Hintergrund: Faktorisierungs-Algorithmen für ganzzahlige Polynome beinhalten oft als ersten Schritt Faktorisierungen über endlichen Körpern. Der Berlekamp-Algorithmus faktorisiert quadratfreie Polynome über endlichen Körpern. Ein Polynom ungleich 0 heißt quadratfrei, wenn es nicht durch das Quadrat eines nicht-konstanten Polynoms teilbar ist. Sie finden die mathematischen Hintergründe und Beschreibungen des Berlekamp-Algorithmus etwa in [GCL92, 8.4] und [Coh93, 3.4.9–3.4.11].

Hauptaufgabe: Programmieren Sie den Berlekamp-Algorithmus für beliebige endliche Körper. Ihr Programm soll insbesondere für einen Körper $K := \mathbb{F}_q$ und ein Polynom $f \in K[X]$ vom Grad n die folgenden Schritte ausführen:

(a) Test, ob f nicht-konstant und quadratfrei ist.

(b) Berechnung der Matrix $Q \in K^{n\times n}$, deren i-te Zeile gegeben ist durch die Koeffizienten von

$$X^{q(i-1)} \bmod f,$$

$i = 1, \ldots, n$. Hierbei bedeutet $g \bmod f$ den (eindeutig bestimmten) Rest, der bei der Division von $g \in K[X]$ durch $0 \neq f \in K[X]$ bleibt.

(c) Bestimmung einer Basis $\{b^{(1)}, \ldots, b^{(k)}\}$ mit $b^{(1)} = (1, 0, \ldots, 0)$ für den (Links-)Nullraum W von $Q - E_n$, d.h. der Menge aller $(a_0, \ldots, a_{n-1}) \in K^{1\times n}$ mit $(a_0, \ldots, a_{n-1})(Q - E_n) = 0$. Die Dimension k dieses Nullraums ist die Anzahl der irreduziblen Faktoren von f. Insbesondere ist f genau dann irreduzibel, wenn $k = 1$ ist.

(d) Der eigentliche Algorithmus: Das Auffinden von Faktoren von f durch Berechnen von

$$\mathrm{ggT}(b^{(i)} - a, u),$$

wobei u alle bisher gefundenen Faktoren von f durchläuft, a alle Elemente aus K, und $b^{(i)}$ alle in (c) gefundenen Basiselemente von W (und der Zeilenvektor $b^{(i)}$ nun als Polynom vom Grad $\leq n$ interpretiert wird). Überlegen Sie sich gut, wie Sie diesen Teil optimieren können, um nicht zu viele unnötige Tests durchzuführen. *25 Punkte*

Anwendungen: (a) Sei

$$f = X^6 - 3X^5 + X^4 - 3X^3 - X^2 - 3X + 1.$$

Faktorisieren Sie f, falls möglich, d.h., falls f quadratfrei ist, für alle Körper mit höchstens 11 Elementen. *5 Punkte*

(b) Faktorisieren Sie, für Primzahlpotenzen $q = p^n$ soweit Sie kommen, das Polynom $X^q - X$ über dem Primkörper $\mathbb{F}_p$. Dieses Polynom ist bekanntlich separabel, also insbesondere quadratfrei. *5 Punkte*

Hinweise: Ist das zu faktorisierende Polynom nicht quadratfrei, dann hilft der Algorithmus *Quadratfreie Faktorisierung* von Aufgabenblatt 13. In (c) dürfen Sie etwa die Maple-Prozedur `nullspace` oder eine vergleichbare aus GAP verwenden. *zusammen 35 Punkte*

Datei: aufg17.tex