

• **Mathematisches Handwerkszeug**

1. **Induktion**

- (a) Beweisen Sie durch Induktion, dass für jede natürliche Zahl  $n$  die Gleichung

$$1 + 3 + 5 + \dots + (2n - 1) = n^2$$

gilt

- (b) Beweisen Sie durch Induktion, dass für jede natürliche Zahl  $n$  die Gleichung

$$n^2 - (n - 1)^2 + (n - 2)^2 - (n - 3)^2 \pm \dots + (-1)^{n-1} = \frac{n(n + 1)}{2}$$

gilt

- (c) Zeigen Sie, dass für  $0 < a < 1$  und jede natürliche Zahl  $n$  stets folgende Abschätzung gilt:

$$(1 - a)^n \geq 1 - na$$

- (d) Die harmonischen Zahlen

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

spielen in vielen Komplexitätsuntersuchungen eine Rolle. Beweisen Sie folgende einfache Abschätzung:

$$H_{2^m} \geq 1 + \frac{m}{2}$$

2. **Einfache Rekursionen**

- (a) Bestimmen Sie die exakte Lösung der folgenden Rekursionsgleichungen

i.  $n \cdot a_n = (n - 2) \cdot a_{n-1} + 2$  ( $n > 1$ ),  $a_1 = 1$

ii.  $a_n = 2a_{n-1} + 1$  ( $n > 1$ ),  $a_1 = 1$

iii.  $a_n = 3a_{n-1} + n - 1$  ( $n \geq 1$ ),  $a_0 = 0$

iv.  $n \cdot a_n = (n + 1) \cdot a_{n-1} + 2n$  ( $n > 0$ ),  $a_0 = 0$

v.  $a_n = \frac{n}{n+1} \cdot a_{n-1} + 1$  ( $n > 0$ ),  $a_0 = 1$

- (b) Lösen Sie die folgenden Rekursionsgleichungen vom divide-and-conquer-Typ durch Zurückführung auf Rekursionen 1. Ordnung

i.  $a_n = 3a_{n/3} + n^2$  ( $n > 1$ ),  $a_1 = c$

ii.  $a_n = 3a_{n/3} + n$  ( $n > 1$ ),  $a_1 = c$

iii.  $a_n = \sqrt{n} \cdot a_{\sqrt{n}} + c \cdot n$

- (c) Lösen Sie die folgenden Rekursionsgleichungen 2. Ordnung exakt

i.  $a_n = 5a_{n-1} - 6a_{n-2}$  ( $n \geq 2$ ),  $a_0 = 0$ ,  $a_1 = 1$

ii.  $a_n = 2a_{n-1} - a_{n-2}$  ( $n \geq 2$ ),  $a_0 = 1, a_1 = 2$

iii.  $a_n = 2a_{n-1} - a_{n-2}$  ( $n \geq 2$ ),  $a_0 = 1, a_1 = 1$

Welches Wachstumsverhalten beobachten Sie?

(d) Bestimmen Sie die Gesamtheit aller Lösungen der folgenden Rekursionsgleichungen

i.  $a_n = 5a_{n-1} - 6a_{n-2}$  ( $n \geq 2$ )

ii.  $a_n = 2a_{n-1} - a_{n-2} + 2a_{n-3}$  ( $n \geq 3$ )

(e) Für welche Werte von  $u$  und  $v$  werden sich die Lösungen der Rekursionsgleichung  $a_{n+1} = u \cdot a_{n-1} + v \cdot a_{n-2}$  unabhängig von den Anfangswerten asymptotisch wie  $o(n)$  verhalten?

• **Eigenschaften der Fibonacci-Zahlen**

In dieser Aufgabe bezeichnet  $(f_n)_{n \geq 0} = (0, 1, 1, 2, 3, 5, 8, 13, \dots)$  die Folge der FIBONACCI-Zahlen. In allen Teilen dieser Aufgabe ist folgende Beziehung nützlich, die per Induktion aus der rekursiven Definition folgt:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

Man kann z.B. die Determinante auf beiden Seiten bilden und erhält

$$(-1)^n = f_{n+1}f_{n-1} - f_n^2$$

woraus man, wenn man es als BÉZOUT-Beziehung liest, sofort  $\text{ggT}(f_{n+1}, f_n) = 1$  schließen kann.

1. Konvergenz zum goldenen Schnitt

Die Glieder der Folge  $(f_{n+1}/f_n)_{n \geq 1}$  liegen alle in dem Intervall  $[1, 2]$  der reellen Geraden, die Folge muss dort also mindestens einen Häufungspunkt haben. Tatsächlich liegt sogar Konvergenz vor, genauer

$$\lim_{n \rightarrow \infty} \frac{f_{n+1}}{f_n} = \phi = \frac{1 + \sqrt{5}}{2}$$

Beweisen Sie dies! Betrachten Sie dazu die Folge mit folgenden Hilfsbezeichnungen:

$$x_n := \frac{f_{2n}}{f_{2n-1}} \quad y_n := \frac{f_{2n+1}}{f_{2n}} \quad (n \geq 1)$$

Dann schreibt sich die Folge  $(f_{n+1}/f_n)_{n \geq 1}$  als  $x_1, y_1, x_2, y_2, x_3, y_3, \dots$ . Bestimmen Sie die relative Lage und den Abstand zweier aufeinanderfolgender Folgenglieder und leiten Sie daraus ab, dass sich die Situation geometrisch folgendermassen darstellt:

$$x_1 < x_2 < x_3 < \dots < \phi < \dots < y_3 < y_2 < y_1$$

d.h.  $(x_i)_{i \geq 1}$  ist streng monoton steigend,  $(y_i)_{i \geq 1}$  ist streng monoton fallend, jedes  $x_i$  ist kleiner als jedes  $y_j$ , also müssen beide Folgen konvergieren, und wegen der Abstände müssen sie sogar den gleichen Grenzwert haben.

2. Benutzen Sie die Matrix-Darstellung, um zu zeigen, dass

$$f_{m+n} = f_{m+1}f_n + f_m f_{n-1}$$

3. Benutzen Sie die Matrix-Darstellung, um zu zeigen, dass

$$f_n \equiv f_{m-1}^q \cdot f_r \pmod{f_m} \text{ falls } n = q \cdot m + r$$

Wegen  $\text{ggT}(f_m, f_{m-1}) = 1$  folgt daraus

$$\text{ggT}(f_n, f_m) = \text{ggT}(f_m, f_r) \text{ falls } n \equiv r \pmod{m}$$

und das hat (euklidischer Algorithmus!) die Konsequenz

$$\text{ggT}(f_n, f_m) = f_{\text{ggT}(n,m)}$$

4. Bestimmen Sie die Eigenwerte und Eigenvektoren der Matrix  $F = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

Antwort : Eigenwerte sind  $\phi$  und  $\hat{\phi}$ ; die zugehörigen Eigenvektoren sind  $\begin{pmatrix} \phi \\ 1 \end{pmatrix}$  und  $\begin{pmatrix} \hat{\phi} \\ 1 \end{pmatrix}$

Als Konsequenz daraus erhält man die Darstellung

$$F^n = \frac{1}{\sqrt{5}} \begin{pmatrix} \phi & \hat{\phi} \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \phi^n & 0 \\ 0 & \hat{\phi}^n \end{pmatrix} \begin{pmatrix} 1 & -\hat{\phi} \\ -1 & \phi \end{pmatrix}$$

Aus

$$\begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix} \begin{pmatrix} \phi \\ 1 \end{pmatrix} = F^n \begin{pmatrix} \phi \\ 1 \end{pmatrix} = \phi^n \begin{pmatrix} \phi \\ 1 \end{pmatrix}$$

erhält man

$$\phi^{n+1} = \phi \cdot f_{n+1} + f_n \text{ und } \phi^n = \phi \cdot f_n + f_{n-1}$$

Die zweite Gleichung, mit  $n + 1$  an Stelle von  $n$ , dann durchmultipliziert mit  $\hat{\phi}$ , liefert  $\phi^n = f_{n+1} - \hat{\phi} \cdot f_n$ .

## • Asymptotik und Wachstumsabschätzungen

### 1. Wachstumsordnungen

Ordnen Sie die folgenden Terme, gelesen als Funktionen von  $n$ , nach ihrer Wachstumsordnung:

$$2^{\sqrt{n}}, e^{\log n^3}, n^{3.01}, 2^{n^2}, n^{1.6}, \log n^3 + 1, \sqrt{n!}, n^{3 \log n}, n^3 \log n, (\log \log n)^3, n^{0.5} 2^n, (n+4)^{12}$$

### 2. Landau-Symbole — weitere Beispiele

Beweisen oder widerlegen Sie die folgenden Aussagen

- (a)  $n^3 + 2^{256} \cdot n^2 \in \Theta(n^3)$
- (b)  $\frac{9n \log n - 4n}{3n \cdot \sqrt{n}} \in \Omega(n)$
- (c)  $\frac{9n \log n - 4n}{3n - \sqrt{n}} \in \Omega(n)$
- (d)  $3^{3n} \in O(3^n)$
- (e)  $n \log n \in o(\log(n^7))$

### 3. Vergleich von Laufzeiten

Zur Berechnung eines Problems stehen zwei Algorithmen  $\mathcal{A}$  und  $\mathcal{B}$  zur Verfügung. Für deren jeweilige Komplexität (= Laufzeit auf Instanzen der Grösse  $n$ ) gelte

$$t_{\mathcal{A}}(n) = \sqrt{n} \quad \text{bzw.} \quad t_{\mathcal{B}}(n) = 2^{\sqrt{\log_2 n}}$$

- (a) Welcher der beiden Algorithmen ist asymptotisch besser (= schneller)?
- (b) Wo liegt der break-even-point, d.h. von welchem Wert der Instanzengrösse  $n$  an ist der asymptotisch bessere Algorithmus immer im Vorteil?

### 4. Iterierter Funktionsaufruf

Betrachten Sie ein Programmkonstrukt

$$\mathcal{A}(n) : \text{for } i \text{ from } 1 \text{ to } n \text{ do } \mathcal{B}(i)$$

- (a) Nehmen Sie an, dass das Programm  $\mathcal{B}$  bei input  $n$  eine Laufzeit  $\in \Theta(n)$  hat. Zeigen Sie, dass dann die Laufzeit von  $\mathcal{A}(n) \in \Theta(n^2)$  ist (unter der vernünftigen Annahme, dass der Aufwand für die Schleifenkontrolle  $\in \Theta(n)$  liegt).
- (b) Nehmen Sie nun an, dass das Programm  $\mathcal{B}(n)$  eine Laufzeit  $\in \Theta(g(n))$  hat, wobei  $g(n)$  eine wachsende Funktion ist. Was kann man dann über die Laufzeit  $f(n)$  von  $\mathcal{A}(n)$  sagen?  
 Gilt immer  $f(n) \in \mathcal{O}(n \cdot g(n))$  ?  
 Gilt immer  $f(n) \in \Omega(n \cdot g(n))$  ?  
 Was passiert, wenn  $g(n)$  exponentielles Wachstum hat?

## • Sortieren und Suchen

### 1. Die Turniermethode

Bei einem Tennisturnier wird der beste Spieler in einem k.o.-Verfahren ermittelt. Unter der idealisierenden Annahme, dass die Spieler während eines Turniers eine konstante Spielstärke haben und sich diese durch Zahlenwerte darstellen lässt, so dass bei einem Spiel nur die Spielstärke der beteiligten Spieler das Resultat bestimmt, kann man den Verlauf des Turniers mit einem binären Baum darstellen. An dessen Knoten sind Spielernahmen notiert. Die Blätter des Baums sind mit den verschiedenen Spielern notiert, die Notierung eines jeden inneren Knotens ergibt sich induktiv aus den Notierungen seiner beiden Kinder anhand der Spielstärke der dort genannten Spieler (verschiedene Spieler sollen unterschiedliche Spielstärke haben). Champion ist der Spieler, der die Wurzel erreicht. Die

Struktur des Baumes hat einen Einfluss auf die Turnierorganisation, aber die Frage, wer Champion wird, ist von der Struktur unabhängig.

- (a) Klar ist, dass bei einem solchen Turnier der Verlierer des Endspiels nicht notwendig der zweitbeste Spieler ist. Aber wie ermittelt man denn dann den zweitbesten Spieler? Überlegen Sie sich, wie man durch (möglichst wenige! wie wenige?) zusätzliche Spiele auch noch den zweitbesten Spieler ermitteln kann.
- (b) Wieviele paarweise Vergleiche benötigt man, um den Median einer drei- bzw. vier-elementigen totalgeordneten Menge zu ermitteln?

## 2. Quickselect

- (a) Entwerfen und implementieren Sie unter Verwendung der `partition`-Methode, die Sie bei `quicksort` kennengelernt haben, einen Algorithmus `quickselect`, der bei Aufruf `quickselect(A, k)` mit einer Liste  $A$  von ganzen Zahlen und einer Zahl  $k$ , ( $1 \leq k \leq \text{length}(A)$ ) das der Größe nach  $k$ -te Element von  $A$  findet, ohne die Liste zu sortieren.
- (b) Demonstrieren sie das Funktionieren dieser Methode, indem Sie die schrittweise Berechnung des Medians der Liste

[97, 50, 79, 56, 49, 63, 57, -59, 45, -8, -93, 92, 43, -62, 77, 66, 54, -5, 99, -61]

durchführen.

- (c) Zeigen sie, dass `quickselect` im schlechtesten Fall eine Anzahl von Vergleichsoperationen benötigt, die proportional zum Quadrat der Listenlänge ist.
- (d) Sei  $T(n)$  die mittlere Anzahl von Vertauschungs-Operationen, die Ihr Algorithmus benötigt. Überprüfen Sie, ob folgende Abschätzung gilt:

$$T(n) \leq \frac{2}{n} \sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k) + \mathcal{O}(n)$$

- (e) Zeigen Sie, dass die Abschätzung in der vorigen Teilaufgabe durch den Ansatz  $T(n) \leq c \cdot n$  mit geeigneter Konstante  $c$  gelöst wird.
- (f) Konsultieren Sie ggf. für diese Aufgabe den Abschnitt 10.2 im Buch von CORMEN/LEISERSON/RIVEST oder den Abschnitt 3.1 des Buches von HEUN

## 3. Paralleles Sortieren

Ein Feld  $A[1..2n]$  der Länge  $2n$  mit Elementen aus einer totalgeordneten Menge heisst  $n$ -sortiert, wenn  $A[k] \leq A[n+k]$  für  $1 \leq k \leq n$  gilt.

Durch paralleles Sortieren

$$A[1..n] \mapsto B[1..n] \quad \text{und} \quad A[n+1..2n] \mapsto B[n+1..2n]$$

der Teilfelder  $A[1..n]$  bzw.  $A[n+1..2n]$  der Länge  $n$  entsteht das Feld  $B[1..2n]$ .

- (a) Geben Sie für  $n = 4$  ein konkretes Beispiel eines 4-sortierten Feldes  $A[1..8]$  an, dessen Elemente die Zahlen  $1, 2, \dots, 8$  in einer geeigneten Umordnung sind, so dass das durch paralleles Teilsortieren entstehende Feld  $B[1..8]$  nicht vollständig sortiert ist.
- (b) Zeigen sie allgemein (für beliebiges  $n$  also), dass das Feld  $B[1..2n]$  immer noch  $n$ -sortiert ist, falls dies für das ursprüngliche Feld  $A[1..2n]$  galt.
- (c) Wieviele Inversionen kann  $B[1..2n]$  im ungünstigsten Fall noch haben?

#### 4. Erwartungswert und Varianz

- (a) Bei den Algorithmen, die das Prinzip der “schnellen Exponentiation” verwenden, spielt die Funktion  $\sharp_1 : \mathcal{B}_n \mapsto \mathbb{N}$  eine wesentliche Rolle. Betrachten Sie diese Funktion als eine Zufallsvariable auf  $\mathcal{B}_n$  (mit Gleichverteilung) und bestimmen sie deren Erwartungswert und Varianz.
- (b) Eine typische Implementierung von Insertionsort hat auf Feldern  $A[1..n]$  (mit paarweise verschiedenen Elementen, der Einfachheit halber) der Länge  $n$  eine Laufzeit proportional zu  $T(\sigma) = 3(n - \text{1rm}(\sigma)) + 8 \text{inv}(\sigma) + 7n - 6$ , wenn  $\sigma \in \mathfrak{S}_n$  die sortierende Permutation ist, d.h.  $A_{\sigma^{-1}(1)} < A_{\sigma^{-1}(2)} < \dots < A_{\sigma^{-1}(n)}$ . Bestimmen Sie die beste, schlechteste und mittlere Laufzeit (im Permutationsmodell mit Gleichverteilung).

Wenn Sie ambitioniert sind, berechnen Sie auch noch die Varianz.

Hinweis: die Varianzberechnung kann man nach dem Vorbild entsprechender Berechnungen für  $\text{1rm}_n$  und  $\text{inv}_n$  elegant mittels der Polynome  $t_n(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{T(\sigma)}$  durchführen, denn auch diese Polynome haben eine Darstellung in Produktform:

$$t_n(x) = x^{7n-6} \prod_{j=2}^n \left( 1 + \sum_{i=1}^{j-1} x^{3+8 \cdot i} \right)$$

#### • Entropie und Quellcodierung

1. Ist  $t$  ein binärer Baum mit inneren Knoten  $I(t)$  Blättern  $E(t)$  und ist  $v : E(t) \rightarrow \mathbb{R}$  eine auf den Blättern von  $t$  definierte Funktion, so kann man diese rekursiv zu einer Funktion  $v : E(t) \cup I(t) \rightarrow \mathbb{R}$  fortsetzen, die auf allen Knoten von  $t$  definiert ist:
  - der Wert  $v(b)$  für  $b \in I(t)$  ist die Summe der Werte  $v(b_\ell) + v(b_r)$  für seine beiden Nachfolgerknoten (Kinder)  $b_\ell$  und  $b_r$ .

Zeigen Sie, dass dann gilt:

$$\sum_{b \in E(t)} v(b) \cdot h(b, t) = \sum_{b \in I(t)} v(b)$$

Wie kann man sich das bei der Berechnung mittlerer Codewortlängen bei HUFFMAN-Codes zunutze machen?

2. Betrachten Sie eine Quelle auf dem Alphabet  $A = \{a, b, c, d, e, f, g, h\}$  mit den Wahrscheinlichkeiten

$$\langle p_a, p_b, p_c, p_d, p_e, p_f, p_g, p_h \rangle = \left\langle \frac{1}{30}, \frac{1}{30}, \frac{1}{30}, \frac{2}{30}, \frac{3}{30}, \frac{5}{30}, \frac{5}{30}, \frac{12}{30} \right\rangle$$

und bestimmen Sie drei verschiedene binäre HUFFMAN-Codes für diese Quelle. “Verschieden” soll heissen, dass die Vektoren der Wortlängen verschieden sind.

3. Da die mittlere Codewortlänge einer Quelle in offensichtlicher Weise der Erwartungswert einer Zufallsvariablen ist, macht es auch einen Sinn, in diesem Kontext von der *Varianz* eines Codes zu sprechen.
  - (a) Verschiedene optimale HUFFMAN-Codes zu einer Quelle haben die gleiche mittlere Wortlänge, können aber durchaus verschiedene Varianz haben. Zeigen sie dies, indem sie für die Quelle über dem Alphabet  $\{a, b, c, d, e\}$  mit Wahrscheinlichkeiten

$$\langle p_a, p_b, p_c, p_d, p_e \rangle = \langle 0.4, 0.2, 0.2, 0.1, 0.1 \rangle$$

zwei verschiedenen HUFFMAN-Codes mit verschiedener Varianz konstruieren.

- (b) Berechnen sie mittlere Wortlänge und Varianz optimaler HUFFMAN-Codes für Quellen mit Gleichverteilung über 5,6,7 und 8 Symbolen.
- (c) Welcher technische Aspekt in der Benutzung von Codes mit variabler Wortlänge steht im Zusammenhang mit der Varianz?

## • Divide-and-Conquer

### 1. Einfache divide-and-conquer-Rekursionen

Betrachten Sie für  $k = 1, 2, 3$  die divide-and-conquer-Rekursion

$$t_k(2n) = k \cdot t_k(n) + b \cdot n + c, \quad t(1) = d$$

wobei  $b, c, d$  (nichtnegative) Konstante sind. Sinn dieser Aufgabe ist es, sich am konkreten Beispiel davon zu überzeugen, dass diese drei Fälle ein unterschiedliches Wachstumsverhalten der Lösungen zeigen. So wie die Definition gegeben ist, macht es natürlich nur Sinn,  $t(n)$  für  $n =$  Potenz von 2 zu betrachten.

- (a) Setzen Sie  $t_k(2^m)$  in der Form

$$t_k(2^m) = \beta_k(m) \cdot b + \gamma_k(m) \cdot c + \delta_k(m) \cdot d$$

an (d.h. lineare Abhängigkeit von den Konstanten  $b, c, d$ ) und berechnen Sie aus der Rekursion die gesuchten Funktionen  $\beta_k(m), \gamma_k(m), \delta_k(m)$  für  $k = 1, 2, 3$ .

- (b) Welche Folgerungen für das asymptotische Verhalten von  $t(n)$  (wenigstens, soweit  $n$  die Potenzen durchläuft, ergeben sich daraus)?
- (c) (Für mathematisch Interessierte)  
Eine elegante Methode, die Lösung zu finden, besteht darin, eine Potenzreihe

$$\tau_k(z) := \sum_{m \geq 0} t_k(2^m) \cdot z^m$$

mit den gesuchten Werten als Koeffizienten anzusetzen. Dann wird man nachzuweisen, dass diese Potenzreihen die Funktionalgleichungen

$$\tau_k(z) = d + z \cdot k \cdot \tau_k(z) + \frac{b \cdot z}{1 - 2z} + \frac{c \cdot z}{1 - z}$$

erfüllen. Schliesslich kann man diese Gleichungen explizit lösen und aus dieser Lösung die gesuchten  $t_k(2^m)$  direkt ablesen.

Führen Sie diesen Vorschlag durch.

## 2. Alternierende Listen

Eine Liste  $A[1..n]$  von ganzen Zahlen heisst *alternierend*, wenn

$A[1] < A[2] > A[3] < A[4] > \dots$  gilt, genauer:  $A[2i-1] < A[2i] > A[2i+1]$  für  $1 \leq i \leq n/2$ .

Entwerfen Sie einen möglichst effizienten divide-and-conquer-Algorithmus, der jede gegebene Liste (verschiedener) ganzer Zahlen in eine alternierende Liste umordnet.

Bestimmen Sie die Komplexität Ihres Verfahrens, gemessen in Vergleichsoperationen. Ihr Algorithmus soll mit  $o(n \log n)$  Vergleichsoperationen auskommen — Sie sollen also nicht sortieren!

Hinweis: Nehmen Sie der Einfachheit halber an, daß die Listenlängen  $n$  immer Potenzen von 2 sind.

## 3. Optimierung von Intervallsummen

Für ein Feld  $A[1..n]$  von ganzen Zahlen (positive und negative!) und Indices  $i, j$  mit  $1 \leq i \leq j \leq n$  sei  $a_{i,j} = \sum_{i \leq k \leq j} A[k]$  die Summe der Elemente von  $A[i..j]$ . Dabei wird die leere Summe wie üblich  $=0$  gesetzt, d.h.  $a_{i,i} = 0$ . Gesucht ist die maximale Intervallsumme, also  $M(A) = \max_{1 \leq i \leq j \leq n} a_{i,j}$ .

Überlegen Sie sich ein möglichst effizientes Verfahren zur Berechnung von  $M(A)$ , d.h., entwerfen Sie nicht nur ein korrektes Verfahren, sondern beurteilen Sie auch dessen Komplexität (hierbei können arithmetische, Vergleichs- und Zuweisungsoperationen von Interesse sein).

(Hinweis: es gibt viele verschiedene Möglichkeiten, dieses Problem algorithmisch anzugehen — und die können sehr unterschiedliche Komplexität haben! Eine anregende Quelle: Column 7 in den *Programming Pearls* von J. BENTLEY, Addison-Wesley, 1989.)

## 4. Optimierung eines Suchverfahrens

Es geht um das Problem, in einer sortierten (!) Liste  $L = [L_1, L_2, \dots, L_n]$  der Länge  $n$  von (verschiedenen) Elementen aus einer totalgeordneten Menge  $M$  die Position eines gesuchten Elementes  $X \in M$  zu bestimmen, falls es in der Liste enthalten ist bzw. Fehlanzeige zu melden, falls es nicht in der Liste enthalten ist. Es geht also um die Bestimmung eines  $\ell$  mit  $1 \leq \ell \leq n$  und  $X = L_\ell$ , indem man  $X$  mit Elementen der Liste vergleicht. Falls  $X$  nicht in  $L$  enthalten ist, soll  $\ell = 0$  die Antwort sein.

Eine mögliche Strategie besteht darin, mittels Sprüngen der Länge  $k$  ein Teilintervall  $[L_{i*k+1}, L_{i*k+2}, \dots, L_{(i+1)*k}]$  der Länge  $k$  zu bestimmen, in dem sich das gesuchte Element mit Sicherheit aufhält (“Grobsuche”), um dann dieses Teilintervall linear zu durchsuchen (“Feinsuche”).

(a) Formalisieren Sie diese Idee als Algorithmus (pseudocode)

(b) Zeigen sie, dass der Aufwand, gemessen in Vergleichsoperationen, im schlech-

testen Fall

$$f(n, k) = \lfloor \frac{n}{k} \rfloor + k - 1$$

beträgt.

- (c) Man dieses Verfahren dadurch optimieren, dass man die Sprungweite  $k$  optimal an die Listenlänge  $n$  anpasst. Berechnen Sie  $f(n, k)$  für  $1 \leq k \leq n \leq 8$  und bestimmen Sie den optimalen Wert von  $k$ .
- (d) Wie sollte man allgemein  $k$  in Abhängigkeit von  $n$  optimal wählen?  
Hinweis: Optimierung verlangt hier die Bestimmung von  $k_0$  mit

$$f(n, k_0) = \min_{1 \leq k \leq n} \left\{ \left\lfloor \frac{n}{k} \right\rfloor + k - 1 \right\}$$

Da diskrete Optimierung schwieriger ist als kontinuierliche, kann man sich näherungsweise erst einmal damit befassen, die Funktion

$$f : \mathbb{R}_{>0} \rightarrow \mathbb{R}_{>0} : x \mapsto \frac{n}{x} + x - 1$$

zu betrachten und  $x_0 \in [1, n] \subset \mathbb{R}$  mit

$$f(x_0) = \min_{1 \leq x \leq n} \left\{ \frac{n}{x} + x - 1 \right\}$$

zu bestimmen ( $x$  variiert also über das kontinuierliche Intervall  $[1, n] \subset \mathbb{R}$ ). In einem zweiten Schritt sollte man sich dann überlegen, wie gross der Fehler ist, wenn man als ganzzahlige Antwort  $\lceil x_0 \rceil$  gibt.

- (e) Wenn Sie die vorige Teilaufgabe gelöst haben, werden Sie feststellen, dass bei der optimalen Lösung die Kosten von Grobsuche und Feinsuche in etwa gleich sind (ein nützliches Prinzip, nicht nur hier!). Jetzt könnten Sie auf die Idee kommen, diesen Ansatz weiterzutreiben und als Methode der “sukzessiven Verfeinerung” über mehrere Stufen zu treiben. Was kommt quantitativ dabei heraus?
- (f) Mathematisch gesehen erfordert die optimierende Lösung der vorigen Teilaufgabe die Berechnung von

$$\min_{x \in \mathbb{R}_{>0}} x \cdot \sqrt{x/n}$$

Was kommt dabei heraus?

## • Euklidischer Algorithmus

1. Untersuchen Sie die Gleichungen

$$120x + 252y = 48 \quad \text{und} \quad 20x + 50y = 510$$

auf Lösbarkeit in  $\mathbb{Z} \times \mathbb{Z}$  und  $\mathbb{N} \times \mathbb{N}$ . Veranschaulichen Sie die Lösungen in einer graphischen Darstellung ( $\mathbb{R}^2$  mit ganzzahligem Gitter  $\mathbb{Z}^2$ ).

- Es seien  $a, b$  teilerfremde natürliche Zahlen und es sei  $n > a \cdot b$ . Zeigen Sie, dass die Gleichung  $a \cdot x + b \cdot y = n$  immer eine *positive* Lösung  $(x, y) \in \mathbb{N}^2$  hat.  
Hinweis: Orientieren Sie sich an der graphischen Interpretation der Lösungsmengen linearer Gleichungen.  
Geben Sie einen effizienten Algorithmus an, der eine solche Lösung findet.
- Es stehen Ihnen als Münzen 2-Cent-Stücke, 5-Cent-Stücke und 10-Cent-Stücke zur Verfügung. Wie viele verschiedene Möglichkeiten gibt es, daraus einen Betrag von 1 Euro zusammenzustückeln?
- Die folgende Aufgabe, entnommen dem Buch *Computational Number Theory* von D. BRESSOUD und S. WAGON, Springer Verlag, 1999, wird dort als das mutmasslich am häufigsten studierte diophantische<sup>1</sup> Problem bezeichnet. Sie wurde von einem B. E. WILLIAMS im Jahr 1926 in der Zeitschrift *Saturday Evening Post* veröffentlicht und deren Leser sollen 20.000 Lösungen eingeschickt haben.

*Five sailors on an island gather a pile of coconuts and decide to divide them among themselves the next day. During the night, one of the sailors awoke, counted the coconuts, gave one to the monkey that was hanging around camp, and took exactly  $1/5$  of the rest for himself. Later that night, a second sailor awoke and did exactly the same thing. And so on, in turn, with none of the sailors being aware of what the others were doing. In the morning, all awoke and, after tossing one coconut to the waiting monkey, divided the remaining coconuts into five equal parts. How many coconuts had the sailors gathered?*

Zeigen Sie, wie man die Bedingungen in einer Gleichung  $ax + by = c$  mit ganzzahligen Koeffizienten formulieren kann, in der  $x$  die Gesamtzahl der Kokosnüsse darstellt. Welches ist also der kleinste mögliche (ganzzahlige) Wert für  $x$ ?

### • Modulare Arithmetik, CRA

- Bestimmen Sie die Menge der Lösungen in  $\mathbb{Z}$  der folgenden Kongruenzensysteme.

$$(a) \begin{cases} x \equiv 5 \pmod{9} \\ x \equiv 11 \pmod{15} \end{cases}$$

$$(b) \begin{cases} x \equiv 8 \pmod{9} \\ x \equiv 4 \pmod{15} \end{cases}$$

- Aus einem alten indischen Rechenbuch:

Aus Früchten werden 63 gleich grosse Haufen gelegt, 7 Stück bleiben übrig. Es kommen 23 Reisende, unter denen die Früchte gleichässig verteilt werden, so dass keine übrig bleibt. Wieviele waren es?

- In einem alten chinesischen Rechenbuch findet sich folgende Aufgabe:

---

<sup>1</sup>Zu Ehren des Mathematikers DIOPHANTUS VON ALEXANDRIA, etwa 250 A.D., bezeichnet man Probleme des ganzzahligen Gleichungslösens weithin als *diophantische* Probleme.

Eine Bande von 17 Räufern stahl einen Sack mit Goldstücken. Als sie ihre Beute teilen wollten, blieben 3 Goldstücke übrig. Beim Streit darüber, wer ein Goldstück mehr erhalten sollte, wurde ein Räuber erschlagen. Jetzt blieben bei der Verteilung 10 Goldstücke übrig. Erneut kam es zum Streit, und wieder verlor ein Räuber sein Leben. Jetzt liess sich endlich die Beute gleichmässig verteilen. Wieviele Goldstücke waren mindestens in dem Sack?

4. Aus einem alten indischen Lehrbuch der Astronomie und Mathematik:

Man bestimme die kleinste positive Zahl, die bei Division durch 3,4,5 und 6 die Reste 2,3,4, bzw. 5 lässt.

5. Aus FIBONACCIS *Liber abbaci*:

Man bestimme die kleinste positive Zahl, die bei Division durch 2,3,4,5,6 jeweils den Rest 1 lässt und durch 7 teilbar ist

6. Die Partialbruchzerlegung rationaler Zahlen:

Ist  $a/m$  eine rationale Zahl und  $m = m_1 m_2 \cdot \dots \cdot m_r$  eine Zerlegung von  $m$  in paarweise teilerfremde Faktoren, dann lässt sich der Bruch  $a/m$  auf genau eine Weise in der Form

$$\frac{a}{m} = b + \frac{a_1}{m_1} + \frac{a_2}{m_2} + \dots + \frac{a_r}{m_r}$$

mir  $a_1, a_2, \dots, a_r, b \in \mathbb{Z}$  und  $0 \leq a_i < m_i$  ( $1 \leq i \leq r$ ) schreiben.

Zeigen Sie dies und entwerfen (und implementieren) Sie einen Algorithmus, der dies leistet.

## • RSA-Kryptographie

1. **Berechnung des privaten Schlüssels**

Für einen Teilnehmer an einem public-key RSA-Kryptosystem werden die Symbole, aus denen die Nachrichten zusammengesetzt sind, mittels der Zahlen  $x \in \mathbb{Z}_{377}$  dargestellt, mittels der Abbildung  $x \mapsto x^e \bmod 377$  verschlüsselt und mittels der Abbildung  $x \mapsto x^d \bmod 377$  entschlüsselt. Der öffentliche Schlüssel sei  $e = 59$ .

Welches ist der private Schlüssel  $d$ ?

## • DFT/FFT

### Diskrete Fourier-Transformation

1. Ein Polynom  $p(z)$  mit  $\deg p(z) < 4$  sei durch folgende Werte festgelegt:

$$p(1) = 1, p(i) = 2, p(-1) = 3, p(-i) = 4$$

Bestimmen Sie die Koeffizientendarstellung dieses Polynoms.

Berechnen Sie die Werte  $p(5)$  und  $p(-2 + 3i)$

2. Berechnen Sie das Produkt der Polynome

$$a(z) = 1 - 3z^2 + 5z^3 \quad \text{und} \quad b(z) = 2 + z - z^2 + 4z^3,$$

indem Sie geeignet auswerten und interpolieren.

3. Es bezeichne  $V_n = V(\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1})$  die Matrix der diskreten Fourier-Transformation  $\text{DFT}_n$ ; dabei ist  $\omega_n = e^{2\pi i/n}$  die  $n$ -te (Haupt-)Einheitswurzel. Aus der Definition folgt  $\det V_n \neq 0$ , aber wie groß ist die Determinante  $\det V_n$  denn wirklich?

Berechnen Sie  $|\det V_n|$ . Eine Möglichkeit besteht darin, herauszufinden, wie die Matrix  $V_n^2$  aussieht; eine andere Möglichkeit wäre, die Kenntnis der inversen Matrix  $V_n^{-1}$  zu verwenden.

4. Die Fourier-Transformierte eines *reellen* Vektors  $\mathbf{a} = (a_0, a_1, \dots, a_{n-1}) \in \mathbb{R}^n$  kann natürlich durchaus komplexe (also nicht notwendig reelle) Komponenten haben.

Zeigen Sie, dass die *Realteile* von  $\mathbf{b} = (b_0, b_1, \dots, b_{n-1}) = V_n \mathbf{a}^t$  in diesem Fall eine Symmetrieeigenschaft haben:  $\mathcal{R}(b_k) = \mathcal{R}(b_{n-k})$  für  $1 \leq k < n$  ( $\mathcal{R}$  = Realteil). Was passiert, wenn der input-Vektor  $\mathbf{a}$  nur rein-imaginäre Komponenten hat?

- NP-Vollständigkeit

Zur Terminologie: Ist  $G = (V, E)$  ein ungerichteter Graph, d.h.  $E \subseteq \binom{V}{2}$  = Menge der zweielementigen Teilmengen von  $V$ , und ist  $v \in V$  ein Knoten von  $G$ , so bezeichne  $\Delta(v) = \Delta_G(v)$  die Menge der *Nachbarn* von  $v$  in  $G$ , d.h.  $\Delta_G(v) = \{u \in V; \{u, v\} \in E\}$ , sowie  $\delta(v) = \delta_G(v) = \#\Delta_G(v)$  den *Grad* von  $v$  in  $G$ , also die Anzahl der Nachbarn.

1. Ist  $G = (V, E)$  ein ungerichteter Graph, so bezeichnet man jede Teilmenge  $C \subseteq V$  mit  $\binom{C}{2} \subseteq E$  als *Clique* von  $G$ , sowie jede Teilmenge  $C \subseteq V$  mit  $\binom{C}{2} \cap E = \emptyset$  als *Co-Clique* von  $G$ . (Der Übergang vom Graphen  $G$  zum komplementären Graphen  $\bar{G} = (V, \binom{V}{2} \setminus E)$  transformiert Cliques in Co-Cliques und umgekehrt — in diesem Sinne sind diese Begriffe als gleichwertig).

Die *CoCliquenzahl*  $\alpha(G)$  eines Graphen  $G$  ist die maximale Grösse einer Co-Clique von  $G$ . Das Problem der Berechnung von  $\alpha(G)$  ist offenbar (mindestens) so schwierig wie das NP-vollständige Entscheidungsproblem COCLIQUE.

- (a) Für  $i = 0, 1, 2$  seien “Graphen vom Typ  $i$ ” definiert als Graphen  $G$  mit  $\max_{v \in V} \delta_G(v) = i$ . Beschreiben Sie anschaulich die Graphen  $G$  dieser drei Typen.
- (b) Wie berechnet man (effizient!)  $\alpha(G)$  für Graphen der Typen 0, 1, 2 (siehe oben)?
- (c) Ein *divide-and-conquer*-Ansatz zur Berechnung von  $\alpha(G)$  sieht so aus: Ist  $G = (V, E)$  ein (ungerichteter) Graph und  $v \in V$  ein Knoten von  $G$ , so bezeichne:
- $G_v^1$  den Graphen, der entsteht, wenn man aus  $G$  den Knoten  $v$  und die zugehörigen Kanten entfernt (aber nicht die anderen Endpunkte, also die Nachbarn von  $v$  in  $G$ ), d.h.

$$G_v^1 = \left( V \setminus \{v\}, E \cap \binom{V \setminus \{v\}}{2} \right)$$

- $G_v^2$  den Graphen, der entsteht, wenn man aus  $G$  den Knoten  $v$ , dessen

Nachbarn  $\Delta_G(v)$  und die zugehörigen Kanten entfernt, d.h.

$$G_v^1 = \left( V \setminus N, E \cap \binom{V \setminus N}{2} \right)$$

mit  $N = \Delta_G(v) \cup \{v\}$ .

Zeigen Sie:

$$\forall v \in V : \alpha(G) = \max(\alpha(G_v^1), 1 + \alpha(G_v^2))$$

- (d) Formulieren Sie einen *divide-and-conquer*-Algorithmus zur Berechnung von  $\alpha(G)$ .

Hinweis: Sie müssen sich festlegen, bei welchen “einfachen” Graphen die obige Zerlegung nicht mehr weitergeführt werden soll. Dafür bieten sich die Graphen der Typen 0, 1, 2 an.

- (e) Untersuchen Sie die *worst-case*-Komplexität Ihres Verfahrens!

Eine rekursive Abschätzung für die Zeit-Komplexität  $t(n)$  (für Graphen mit  $n$  Knoten) sollte folgende Gestalt haben

$$t(n) \leq t(n-1) + t(n-2-i) + cn^2 \quad (i \in \{0, 1, 2\})$$

wobei der Parameter  $i$  angibt, bei welchem Typ von “einfachen” Graphen man das divide-and-conquer stoppt.

Zeigen Sie, dass sich Lösungen dieser Rekursion exponentiell verhalten, also von der Form  $\mathcal{O}(\beta_i^n)$  sind, wobei  $\beta_0 = 1.61\dots$ ,  $\beta_1 = 1.46\dots$ ,  $\beta_2 = 1.38\dots$  ist.

## 2. Graphen, Färbungen und Erfüllbarkeit

- (a) Zeigen Sie, dass ein (ungerichteter) Graph genau dann einen Eulerschen Kreis besitzt, wenn er zusammenhängend ist und jeder Knoten einen *geraden* Grad hat.
- (b) Zeigen Sie, dass ein (ungerichteter) Graph  $G$  genau dann 2-färbbar ist, wenn jeder Kreis (geschlossene Pfad) in  $G$  *gerade* Länge hat. Formulieren Sie einen effizienten Algorithmus zum Testen der 2-Färbbarkeit von Graphen.
- (c) Zeigen Sie, dass das Problem 2-(CNF)-SAT der Erfüllbarkeit von Klauselmengen, bei denen jede Klausel aus höchstens zwei Literalen besteht, effizient entscheidbar ist.

Hinweis: Ist  $X$  die Menge der in den Klauseln vorkommenden Variablen, so untersuchen Sie den (gerichteten) Graphen, dessen Knotenmenge aus allen Literalen  $x$  und  $\bar{x}$  ( $x \in X$ ) besteht und wo jeder Klausel  $\{\bar{\alpha}, \beta\}$  eine Kante  $\alpha \rightarrow \beta$  und eine Kante  $\bar{\beta} \rightarrow \bar{\alpha}$  zugeordnet wird. Beachte  $\bar{\bar{\alpha}} = \alpha$ .

Wie formuliert sich die (Un-)Erfüllbarkeit als Eigenschaft dieses Graphen?

## 3. Zum Problem SATISFIABILITY

- (a) Es seien  $x_1, x_2, \dots, x_k, z_1, z_2, \dots, z_{k-3}$  Boolesche Variable. Betrachten Sie die folgenden Disjunktionen (Klauseln)

$$c \equiv x_1 \vee x_2 \vee \dots \vee x_k$$

$$c_1 \equiv x_1 \vee x_2 \vee z_1$$

$$c_j \equiv x_{j+1} \vee \overline{z_{j-1}} \vee z_j \quad (2 \leq j \leq k-3)$$

$$c_{k-2} \equiv x_{k-1} \vee \overline{z_{k-3}} \vee x_k$$

und zeigen Sie:

- jede Boolesche Bewertung von  $x_1, \dots, x_k$ , welche die Klausel  $c$  erfüllt, kann zu eine Booleschen Bewertung von  $x_1, \dots, x_k, z_1, \dots, z_{k-3}$  erweitert werden, die alle Klauseln  $c_1, \dots, c_{k-2}$  erfüllt.
  - die Bewertung  $x_i = \mathbf{false}$  ( $1 \leq i \leq k$ ), welche die Klausel  $c$  nicht erfüllt, kann auf keine Weise zu einer Bewertung von  $x_1, \dots, x_k, z_1, \dots, z_{k-3}$  erweitert werden, die alle Klauseln  $c_1, \dots, c_{k-2}$  erfüllt.
- (b) Überlegen Sie sich, wie man zu jedem Booleschen Schaltkreis (oder straight-line Programm) mit input-Variablen  $x_1, \dots, x_n$  eine Klauselmenge (mit Literalen zu den Variablen  $x_1, \dots, x_n$  und zusätzlichen Variablen für jeden input- bzw. output-Knoten und jedes Gatter) mit maximal drei Literalen pro Klausel konstruieren kann, so dass Instanzen von CIRCUIT VALUE in Instanzen von SATISFIABILITY transformiert werden. Stellen Sie sicher, dass Ihre Konstruktion mit logarithmischem Platz auskommt.

#### 4. Polynomgleichungen und Färbungen von Graphen

Das Problem der Lösbarkeit von Polynomgleichungen (bzw. Systemen)

$$p(x_1, \dots, x_n) = 0 \quad \text{bzw.} \quad p_i(x_1, \dots, x_n) = 0 \quad (1 \leq i \leq m)$$

(mit ganzzahligen Koeffizienten, in mehreren Variablen) (“algebraische Gleichungen”) hat die Mathematiker seit der Antike (DIOPHANTUS, 3. Jh.) beschäftigt. Wirklich “einfach” sind dabei nur die beiden Spezialfälle

- es tritt nur eine Variable auf: das ist die Frage nach den Nullstellen eines Polynoms in einer Variablen
- alle Terme haben den Grad 1: das ist der Fall eines linearen Gleichungssystems

Allgemein gilt folgendes:

- es kann prinzipiell kein algorithmisches Verfahren geben, mit dem man die Frage nach der Existenz *ganzzahliger* Nullstellen beliebiger Polynome beantworten kann (negative Lösung des berühmten 10. Problems von D. HILBERT, gestellt im Jahr 1900, gelöst 1970, nach Vorarbeiten von M. DAVIS, H. PUTNAM, J. ROBINSON, von dem damals 20-jährigen Y. MATJASEVIC). Die “Lösung” besteht darin, einen sehr tiefen Zusammenhang mit der Theorie der Berechenbarkeit und dem Halteproblem herzustellen (“*jede rekursiv-aufzählbare Menge natürlicher Zahlen ist diophantisch*” — die umgekehrte Aussage ist einfach).
- für die Frage nach der Existenz *reeller* bzw. *komplexer* Lösungen von polynomialen Gleichungen (und Gleichungssystemen) gibt es ein hoch-komplexes Entscheidungsverfahren von A. TARSKI (etwa um 1940 gefunden, aber zeitbedingt erst viel später veröffentlicht).

Bei diesem Stand der Dinge verwundert es nicht, dass auch lösbare Spezialfälle dieses Problems i.a. sehr komplex sind. Hier soll ein Zusammenhang mit dem NP-vollständigen Problem der 3-Färbbarkeit von Graphen hergestellt werden.

Ist  $G = (V, E)$  ein Graph, so sei  $\{x_v; v \in V\}$  eine die Knoten von  $G$  repräsentierende Menge von Variablen. Man betrachte dann das polynomiale Gleichungssystem

$$\begin{aligned} x_v^3 &= 1 & (v \in V) \\ x_u^2 + x_u x_v + x_v^2 &= 0 & ((u, v) \in E) \end{aligned}$$

von insgesamt  $\#V + \#E$  Gleichungen.

Zeigen Sie: dieses System ist genau dann lösbar in den komplexen Zahlen  $\mathbb{C}$ , wenn der Graph  $G$  3-färbbar ist.

- **Ergänzungen**

1. **DFT und FFT über Restklassenringen**

In dieser Aufgabe soll exemplarisch dargestellt werden, dass DFT und FFT nicht an das Rechnen mit komplexen Zahlen gebunden sind, sondern dass das Wesen dieser Transformationen und Algorithmen im Umgang mit den Einheitswurzeln, also den Lösungen der Gleichung  $X^n = 1$ , liegt. Lösungen dieser Gleichung kann man aber auch in ganz anderen Bereichen als den komplexen Zahlen finden - auch in solchen, wo man rundungsfehlerfrei rechnen kann. Ziel dieser Aufgabe ist es, die Berechnung der Faltungsoperation über Restklassenringen mittels DFT und FFT darzustellen. Dieses ist der Ansatz für das bislang schnellste Integer-Multiplikationsverfahren von Schönhage und Strassen.

Die Bearbeitung dieser Aufgabe setzt voraus, dass Sie mit den Eigenschaften der Restklassenringe  $\mathbb{Z}_m = (\{0, 1, 2, \dots, m\}, +_m, \times_m, 0, 1)$  einigermaßen vertraut sind. Dabei bedeuten “ $+_m$ ” und “ $\times_m$ ” die Addition bzw. Multiplikation “modulo  $m$ ”.

Falls Sie die allgemeinen Aussagen nicht gleich verstehen, sollten Sie versuchen, diese erst einmal anhand der weiter unten folgenden konkreten Beispiele nachzuvollziehen.

(a) Ist  $R$  irgendein Ring und  $0 \neq \omega \in R$ , ist ferner  $n = 2^k$  mit  $k \geq 1$ , so gilt

$$\sum_{i=0}^{n-1} \omega^{ip} = \prod_{j=0}^{k-1} (1 + \omega^{2^j p}) \quad (1 \leq p < n)$$

(b) Ist nun  $R = \mathbb{Z}_m$ , ferner  $0 \neq \omega \in \mathbb{Z}_m$  und  $n = 2^k$  mit  $k \geq 1$ , und gilt dann noch  $\omega^{n/2} + 1 = 0$  in  $\mathbb{Z}_m$ , so ist

$$\sum_{i=0}^{n-1} \omega^{ip} = 0 \quad (1 \leq p < n)$$

Hinweis: schreiben Sie  $p = 2^s \cdot q$  mit ungeradem  $q$  und finden Sie ein geeignetes  $j$  mit  $1 + \omega^{2^j p} = 0$  in  $\mathbb{Z}_m$ .

- (c) Nun sei  $R = \mathbb{Z}_m$ ,  $0 \neq \omega = 2^t \in \mathbb{Z}_m$  mit  $t \geq 1$ ,  $n = 2^k$  mit  $k \geq 1$ , sowie  $\omega^{n/2} + 1 = 0$  in  $\mathbb{Z}_m$ .

Dann sind  $n$  und  $\omega$  invertierbar in  $\mathbb{Z}_m$ , d.h.  $\text{ggT}(n, m) = 1 = \text{ggT}(\omega, m)$ .

- (d) Unter den Voraussetzungen der vorigen Teilaufgabe sind die  $n$  Zahlen

$$\omega^0 = 1, \omega^1, \omega^2, \omega^3, \dots, \omega^{n-1}$$

$n$  verschiedene (!) Lösungen der Gleichung  $X^n = 1$  in  $\mathbb{Z}_m$ .

- (e) Die vorige Aussage erlaubt es, unter den dort gegebenen Bedingungen eine DFT zu definieren

$$\text{DFT}_{m,n,\omega} : \mathbb{Z}^n \rightarrow \mathbb{Z}^n : \mathbf{a} \mapsto (a(\omega^0), a(\omega^1), a(\omega^2), \dots, a(\omega^{n-1}))$$

wobei  $\mathbf{a} = (a_0, a_1, a_2, \dots, a_{n-1})$  und  $a(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$ . Zeigen Sie, dass diese (lineare) Transformation umkehrbar ist und geben Sie die Transformationsmatrix für  $\text{DFT}_{m,n,\omega}$  und ihre Inverse an.

- (f) Betrachten Sie die Situation für  $m = 5, n = 4, \omega = 2$ . Berechnen Sie die Transformationsmatrix für  $\text{DFT}_{5,4,2}$  und  $\text{DFT}_{5,4,2}^{-1}$ . Berechnen Sie die DFT von  $\mathbf{a} = (0, 1, 2, 3)$  und kontrollieren Sie das Resultat per Rücktransformation.
- (g) Sind  $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$  und  $\mathbf{b} = (b_0, b_1, \dots, b_{n-1})$  Vektoren aus  $R^n$  (wobei  $R$  ein Ring ist), so bezeichnet man den Vektor

$$c = (c_0, c_1, \dots, c_{2n-1}) \in R^{2n} \quad \text{wobei } c_k = \sum_{i+j=k} a_i b_j \quad (0 \leq k < 2n)$$

als die *Faltung* von  $\mathbf{a}$  und  $\mathbf{b}$ .

Skizzieren Sie, wie man die DFT bzw. FFT über Restklassenringen  $R = \mathbb{Z}_m$  zur Berechnung von Faltungsoperationen verwenden kann.

- (h) Berechnen Sie konkret die Faltung der beiden Vektoren  $\mathbf{a} = (1, 2, 3, 4)$  und  $\mathbf{b} = (4, 3, 2, 1)$  über dem Ring  $\mathbb{Z}_{17}$ . Verwenden Sie  $\text{DFT}_{m,n,\omega}$  mit den Parametern  $m = 17, n = 8, \omega = 2$ .

## 2. Transitiv Hülle und reguläre Sprachen

Diese Aufgabe knüpft an das in der Vorlesung behandelte Thema *Boolesche Matrixmultiplikation und transitive Hülle* an. Aus dem Ansatz, Pfade in Graphen zu betrachten, ergibt sich unmittelbar der Zusammenhang mit der Theorie endlichen Automaten. Erinnern Sie sich für diese Aufgabe also bitte an die Vorlesung ETI-I, Abteilung "Endliche Automaten und reguläre Ausdrücke/Sprachen". Beachten Sie: ist  $\Sigma$  ein (endliches) Alphabet, so bildet die Menge der formalen Sprache  $L \subseteq \Sigma^*$  mit den Operationen  $\cup$  (=Vereinigung) und  $\cdot$  (=Konkatenation) einen Halbbring. Eine wichtige Operation für formale Sprachen ist der Kleene-Operator  $A \mapsto A^*$  der iterierten Konkatenation.

$$A^* = A^0 \cup A \cup A^2 \cup A^3 \cup \dots$$

Die Familie der *regulären* Sprachen ist die kleinste Familie von Sprachen, die alle endlichen Sprachen enthält und unter der “rationalen” Operationen der Vereinigung, der Konkatenation und dem Kleene-Operator abgeschlossen ist. Sie wissen sicher noch, wie man diese Sprachen mit Hilfe von endliche Automaten charakterisiert (Theorem von Kleene).

Sind nun  $\mathbf{A} = [A_{i,j}]_{1 \leq i,j \leq n}$  und  $\mathbf{B} = [B_{i,j}]_{1 \leq i,j \leq n}$  Matrizen von formalen Sprachen, so definiert man als Matrixprodukt

$$\mathbf{A} \cdot \mathbf{B} = \mathbf{C} = [C_{i,j}]_{1 \leq i,j \leq n} \text{ mit } C_{i,j} = \bigcup_{1 \leq k \leq n} A_{i,k} \cdot B_{k,j}$$

Sind  $\mathbf{A}$  und  $\mathbf{B}$  Matrizen von rationalen Sprachen, so ist natürlich auch  $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$  eine Matrix von rationalen Sprachen.

- (a) Es sei  $G = (V, E)$  ein endlicher Graph mit  $V = \{1, 2, \dots, n\}$ . Fassen Sie die Kantenmenge  $E$  als Alphabet auf und entsprechend die Adjazenzmatrix

$$\mathbf{E} = [E_{i,j}]_{1 \leq i,j \leq n} \text{ mit } E_{i,j} = \begin{cases} \{(i, j)\} & \text{falls } (i, j) \in E \\ \emptyset & \text{falls } (i, j) \notin E \end{cases}$$

als Matrix von (sehr einfachen) formalen Sprachen. Zeigen Sie: die Matrix  $\mathbf{E}^n$  (im Sinne der oben eingeführten Matrixmultiplikation) enthält als Einträge (reguläre Ausdrücke für) die regulären Sprachen, die gerade die Pfade der Länge  $n$  (mit passendem Anfangs- und Endpunkt) in dem Graphen  $G$  beschreiben.

- (b) Man kann also auch die Matrix  $\mathbf{E}^*$  betrachten? Deren Einträge codieren also die Mengen aller Pfade zwischen passenden Anfangs- und Endpunkten. Zeigen Sie, dass die Einträge in dieser Matrix rationale Sprachen sind. Berechnen Sie diese Matrix für das Beispiel aus der Vorlesung (Seite 248 im Buch von Savage).
- (c) Zeigen Sie, dass allgemein folgende Aussage gilt: ist  $\mathbf{A}$  eine Matrix von rationalen Sprachen, so ist auch  $\mathbf{A}^*$  eine Matrix von rationalen Sprachen.
- (d) Ist  $\Sigma$  ein endliches Alphabet, so wird für jedes Wort  $w \in \Sigma^*$  mit  $|w|$  die Länge von  $w$  bezeichnet. Die Abbildung  $\phi : \Sigma^* \rightarrow \mathbb{Z}[t] : w \mapsto t^{|w|}$  ist ein multiplikativer Homomorphismus, der sich zu einem Homomorphismus

$$\phi : 2^{\Sigma^*} \rightarrow \mathbb{Z}[[t]] : L \mapsto \ell_L(t) := \sum_{w \in L} t^{|w|}$$

von Halbringen fortsetzen lässt.  $\phi(L)$  ist eine also Potenzreihe

$$\ell_L(t) = \ell_0 + \ell_1 t + \ell_2 t^2 + \ell_3 t^3 + \dots$$

wobei der Koeffizient  $\ell_n$  von  $\ell_L(t)$  angibt, wieviele Wörter in  $L$  die Länge  $n$  haben:

$$\ell_n = \#\{w \in L; |w| = n\}$$

$\ell_L(t)$  wird auch als “erzeugende Funktion” von  $L$  bezeichnet. ein fundamentales Resultat aus der Theorie der formalen Sprachen besagt, dass die erzeugende Funktion einer rationalen Sprache immer eine rationale Funktion ist, d.h. sich als Quotient von zwei Polynomen schreiben lässt.

[ Das wäre ganz einfach zu beweisen, wenn die “erwünschten” Beziehungen

$$\ell_{A \cup B}(t) = \ell_A(t) + \ell_B(t) , \ell_{A \cdot B}(t) = \ell_A(t) \cdot \ell_B(t) , \ell_{A^*}(t) = \frac{1}{1 - \ell_A(t)}$$

immer gelten würden. Leider ist das nicht der Fall! Können Sie sehen, wo das Problem liegt? Versuchen Sie, Gegenbeispiele zu diesen Beziehungen zu konstruieren. Dass die zitierte Aussage dennoch gilt, folgt übrigens daraus, dass man jede reguläre Sprache durch einen *deterministischen* endlichen Automaten erkennen kann.]

Hier geht es um einen speziellen Aspekt dieser Situation: Ist wieder  $G = (V, E)$  ein endlicher Graph, so kann man den Homomorphismus  $\phi$  auch die Matrizen  $\mathbf{E}^n$  und  $\mathbf{E}^*$  anwenden.

- i. Beweisen Sie die Beziehung

$$\phi(\mathbf{E}^n) = t^n \mathbf{A}^n$$

wobei  $\mathbf{A}$  die Adjazenzmatrix des Graphen  $G$  ist und die Potenz  $\mathbf{A}^n$  in der üblichen ganzzahligen Arithmetik (also nicht mit Boolescher Matrixmultiplikation) berechnet wird.

- ii. Wie kann man diese Beziehung verwenden, um die Einträge der Matrix  $\phi(\mathbf{E}^*)$  zu berechnen. Beachten Sie: an der Position  $(i, j)$  dieser Matrix steht gerade die erzeugende Funktion der Sprache aller Pfade (beliebiger Länge) von  $i$  nach  $j$  im Graphen  $G$ .
- iii. Führen Sie diese Berechnung für den schon oben behandelten Beispielfgraphen aus. (Sie werden diese Berechnung vielleicht lieber nicht per Hand machen wollen. Dazu gibt es Computeralgebra-System, wie Maple und Mathematica).

### 3. Reguläre Sprachen und lineare Gleichungssysteme

Diese Aufgabe gehört zusammen mit der vorigen in den Bereich der Automaten-theorie und ist als Ergänzung zum Stoff von “Theoretische Informatik I” gedacht. Sie sollte Anregung bieten, sich wieder einmal mit diesem Bereich auseinanderzusetzen.

- (a) Es sei  $\Sigma$  ein endliches Alphabet und es seien  $A, B \subset \Sigma^*$  Sprachen über  $\Sigma$ . Beweisen Sie die als “ARDENS Lemma” bekannte Aussage:
  - i. Die Sprache  $A^* \cdot B$  ist die (bezüglich Inklusion) kleinste Lösung des linearen Gleichungssystems für formale Sprachen:

$$X = A \cdot X + B$$

ii. Gehört  $\lambda$  (= leeres Wort) nicht zu  $A$ , so ist  $A^* \cdot B$  auch die *einzig*e Lösung dieser Gleichung.

Beachten Sie: sind  $A, B$  selbst reguläre (= rationale) Sprachen, so ist auch die Lösung des Gleichungssystems regulär.

(b) Bevor Sie verallgemeinern,

i. versuchen Sie das folgende lineare Gleichungssystem zu lösen:

$$X = a \cdot X + b \cdot Y + a \quad , \quad Y = a \cdot X + a \cdot Y + b$$

ii. Ein endlicher Automat  $\mathcal{A}$  über dem Alphabet  $\Sigma = \{a, b\}$  habe drei Zustände  $A, B, C$ , dabei sei  $B$  Anfangszustand und  $A, B$  seien akzeptierende Zustände. Die Übergangsfunktion  $\delta : \{A, B, C\} \times \{a, b\} \rightarrow \{A, B, C\}$  sei gegeben durch

$$\delta(A, a) = B, \delta(A, b) = \delta(B, a) = A, \delta(B, b) = C, \delta(C, a) = \delta(C, b) = C$$

Beschreiben Sie die von dem endlichen Automaten akzeptierte Sprache mit Hilfe eines linearen Gleichungssystems und lösen Sie dieses.

(c) Formulieren und beweisen Sie eine allgemeine Aussage über die Lösungen von linearen Gleichungssystemen, deren Koeffizienten und Lösungen (reguläre) formale Sprachen sind. Übertragen Sie diese Aussage auf endliche Automaten (Theorem von KLEENE).

#### 4. Divisionseigenschaft für Polynome

– Formulieren Sie einen Algorithmus, der die Divisionseigenschaft für Polynome realisiert:

Ist  $k$  ein Körper und sind  $f, g \in k[X]$  Polynome mit  $g \neq 0$ , so gibt es Polynome  $q, r \in k[X]$  mit  $r = 0$  oder  $\deg r < \deg g$ , so dass

$$f = g \cdot q + r$$

gilt.

Zeigen Sie, dass  $q$  und  $r$  eindeutig bestimmt sind.

– Zeigen Sie, dass in  $k[X]$  die BEZOUT-Eigenschaft gilt:

Zu  $f, g \in k[X]$  gibt es  $u, v \in k[X]$  mit

$$u \cdot f + v \cdot g = \text{ggT}(f, g)$$

und dass dabei  $u$  und  $v$  eindeutig bestimmt sind, wenn man noch

$$\deg u < \deg g - \deg \text{ggT}(f, g) \quad \text{und} \quad \deg v < \deg f - \deg \text{ggT}(f, g)$$

fordert.

– Zeigen Sie folgende erweiterte BEZOUT-Eigenschaft:

Sind  $f, g, h \in k[X]$  mit  $\text{ggT}(f, g) \mid h$ , so gibt es eindeutig bestimmte  $u, v \in k[X]$  mit

$$u \cdot f + v \cdot g = h \quad \text{und} \quad \deg u < \deg g - \deg \text{ggT}(f, g)$$

Falls  $\deg h < \deg f + \deg g - \deg \text{ggT}(f, g)$  gilt, so ist auch  $\deg v < \deg f - \deg \text{ggT}(f, g)$

- Partialbruchzerlegung für echte rationale Funktionen.

Eine *rationale Funktion* wird dargestellt durch einen Quotienten  $f/g$  von zwei Polynomen  $f, g \in k[X]$  mit  $g \neq 0$ . Ein solcher Quotient heisst *echte* rationale Funktion, wenn  $\deg f < \deg g$  oder  $f = 0$  gilt. (Wie im Fall der rationalen Zahlen identifiziert man Quotienten  $f_1/g_1$  und  $f_2/g_2$ , falls  $f_1 \cdot g_2 = f_2 \cdot g_1$  gilt.) Folgende Aussage wird als “Partialbruchzerlegung” (für echte rationale Funktionen) angesprochen:

Sind  $f, g, h \in k[X]$  Polynome, wobei  $\text{ggT}(g, h) = 1$  und  $\deg f < \deg g + \deg h$ , so gibt es eindeutig bestimmte Polynome  $a, b \in k[X]$  mit  $\deg a < \deg g$  und  $\deg b < \deg h$  und

$$\frac{f}{g \cdot h} = \frac{a}{g} + \frac{b}{h}$$

Wie berechnet man  $a$  und  $b$  ?

- **Notiz zur Laufzeit von Insertionsort**

Eine Implementierung von Insertionsort könnte typischerweise eine Laufzeit haben, die für eine sortierende Permutation  $\sigma \in \mathfrak{S}_n$  proportional zu

$$T(\sigma) = 3(n - \text{lrm}(\sigma)) + 8 \text{inv}(\sigma) + 7n - 6$$

ist.

Der “best case” tritt ein für  $\sigma' = 1\,2\,3 \dots n$  und ergibt wegen  $\text{inv}(\sigma') = 0$  und  $\text{lrm}(\sigma') = n : T_{\text{best}}(n) = 7n - 6$ .

Der “worst case” tritt ein für  $\sigma'' = n(n-1)(n-2) \dots 1$  und ergibt wegen  $\text{inv}(\sigma'') = \binom{n}{2}$  und  $\text{lrm}(\sigma'') = 1 : T_{\text{worst}}(n) = 4n^2 + 6n - 9$ .

Im “average case” hat man wegen der Linearität der Mittelwertbildung

$$\begin{aligned} \mathbf{E}(T_n) &= 3(n - \mathbf{E}(\text{lrm}_n)) + 8 \mathbf{E}(\text{inv}_n) + 7n - 6 \\ &= 3(n - H_n) + 8 \frac{n(n-1)}{4} + 7n - 6 \\ &= 2n^2 + 8n - 3H_n - 6 \end{aligned}$$

Die Varianz der Zufallsvariablen  $T(\sigma)$  (mit Gleichverteilung auf  $\mathfrak{S}_n$ ) kann mittels der Polynome

$$t_n(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{T(\sigma)}$$

berechnen:

$$\mathbf{V}(T_n) = \frac{t_n''(1)}{t_n(1)} + \frac{t_n'(1)}{t_n(1)} - \left( \frac{t_n'(1)}{t_n(1)} \right)^2$$

Um eine brauchbare Darstellung für diese Polynome zu erhalten, betrachte man zunächst einmal die Abbildung

$$\sigma = \sigma_1 \sigma_2 \dots \sigma_n \mapsto n - \mathbf{lrm}(\sigma) + \mathbf{inv}(\sigma)$$

Klassifiziert man nach dem letzten Symbol  $\sigma_n$ , so erkennt man

$$n - \mathbf{lrm}(\sigma) + \mathbf{inv}(\sigma) = \begin{cases} (n-1) - \mathbf{lrm}(\sigma_1 \dots \sigma_{n-1}) + \mathbf{inv}(\sigma_1 \dots \sigma_{n-1}) & \text{falls } \sigma_n = n \\ (n-1) - \mathbf{lrm}(\sigma_1 \dots \sigma_{n-1}) + \mathbf{inv}(\sigma_1 \dots \sigma_{n-1}) + n - k + 1 & \text{falls } \sigma_n = k < n \end{cases}$$

Mittels der Polynome

$$\tau_n(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{n - \mathbf{lrm}(\sigma) + \mathbf{inv}(\sigma)}$$

formuliert sich dies als

$$\tau_n(x) = \tau_{n-1}(x) \cdot (1 + x^2 + x^3 + \dots + x^n)$$

woraus mittels Induktion (Anfang:  $\tau_1(x) = 1$ )

$$\tau_n(x) = \prod_{2 \leq j \leq n} (1 + x^2 + x^3 + \dots + x^j)$$

folgt. Ganz analog ergibt sich, wenn  $a$  und  $b$  Parameter sind:

$$\tau_n^{(a,b)}(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{a \cdot (n - \mathbf{lrm}(\sigma)) + b \cdot \mathbf{inv}(\sigma)} = \prod_{2 \leq j \leq n} \left( 1 + \sum_{i=1}^{j-1} x^{a+b \cdot i} \right)$$

Man erkennt die beiden bekannten Spezialfälle

– Links-Rechts-Maxima ( $a = 1, b = 0$ )

$$\tau_n^{(1,0)}(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{n - \mathbf{lrm}(\sigma)} = \prod_{2 \leq j \leq n} (1 + (j-1)x)$$

– Inversionen ( $a = 0, b = 1$ )

$$\tau_n^{(0,1)}(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{\mathbf{inv}(\sigma)} = \prod_{2 \leq j \leq n} (1 + x + x^2 + \dots + x^{j-1})$$

Die Produktdarstellung macht nunmehr die Berechnung der Varianz von  $T_n$  zu einer Routineangelegenheit: setzt man

$$\phi_j^{(a,b)}(x) = 1 + \sum_{i=1}^{j-1} x^{a+b \cdot i}$$

so ist

$$\tau_n^{(a,b)}(x) = \prod_{j=2}^n \phi_j^{(a,b)}(x)$$

und mittels logarithmischer Ableitung ergibt sich

$$\frac{\tau_n^{(a,b)''}(1)}{\tau_n^{(a,b)}(1)} + \frac{\tau_n^{(a,b)'}(1)}{\tau_n^{(a,b)}(1)} - \left( \frac{\tau_n^{(a,b)'(1)}}{\tau_n^{(a,b)}(1)} \right)^2 = \sum_{j=2}^n \left[ \frac{\phi_j^{(a,b)''}(1)}{\phi_j^{(a,b)}(1)} + \frac{\phi_j^{(a,b)'(1)}}{\phi_j^{(a,b)}(1)} - \left( \frac{\phi_j^{(a,b)'(1)}}{\phi_j^{(a,b)}(1)} \right)^2 \right]$$

Die Summanden auf der rechten Seite lassen sich leicht ausrechnen. Wegen

$$t_n(x) = \sum_{\sigma \in \mathfrak{S}_n} x^{T(\sigma)} = x^{7n-6} \cdot \tau_n^{(3,8)}(x)$$

erhält man dann  $\mathbf{V}(T_n)$ , indem man  $a = 3, b = 8$  setzt (der multiplikative Term  $x^{7n-6}$  trägt natürlich nichts zur Varianz bei!). Einsetzen und geduldiges (bzw. maschinenunterstütztes) Ausrechnen liefert schliesslich

$$\mathbf{V}(T_n) = \frac{16}{9} n^3 + \frac{8}{3} n^2 + \frac{176}{9} n - \frac{112}{3} - 15 H_n + 9 H_n^{(2)}$$

wobei  $H_n^{(2)} = \sum_{i=1}^n \frac{1}{i^2} \rightarrow_{n \rightarrow \infty} \frac{\pi^2}{6}$ .