Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M.Sc. Stefan Karch
M.Sc. Mariia Sukhova

21.11.2023

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 6

Bearbeitungszeitraum: 04.12.2023 – 15.12.2023

Aufgabe 1 (Pflichtaufgabe) *Das zweistufige Horner–Schema*

Polynome sind Funktionen $f : \mathbb{R} \to \mathbb{R}$ von der Form

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} \cdots + a_1 x + a_0 \qquad \text{für alle } x \in \mathbb{R}. \tag{I}$$

Die reellen Zahlen $a_0, \ldots, a_n$ werden die *Koeffizienten* des Polynoms genannt, wobei $a_n \neq 0$ vorausgesetzt wird. Die Zahl n nennt man den *Grad* des Polynoms. Sei eine Stelle $x \in \mathbb{R}$ fest gewählt, so kann man durch sukzessives Ausklammern folgende Darstellung für den Funktionswert $f(x)$ gewinnen:

$$f(x) = \big(\cdots\big((a_n x + a_{n-1})x + a_{n-2}\big)x \cdots + a_1\big)x + a_0. \tag{II}$$

Definiert man die reellen Zahlen $b_0, b_1, \ldots, b_{n-2}, b_{n-1}$ rekursiv durch

$$b_{n-1} := a_n, \qquad b_{k-1} := x b_k + a_k \qquad \text{für } k = n-1, n-2, \ldots, 0,$$

so kann man für den Funktionswert $f'(x)$ der Ableitung des Polynoms die folgende Darstellung herleiten:

$$f'(x) = \big(\cdots(b_{n-1}x + b_{n-2})x \cdots + b_1\big)x + b_0. \tag{III}$$

Dies ist die Grundlage des sogenannten *zweistufigen Horner–Schemas*, einem Verfahren, mit dem Polynome und deren Ableitungen effizient ausgewertet werden können. Das zweistufige Hornerschema wertet die Klammern in Formel (II) und (III) quasi „von innen nach außen“ aus. Es kann als Algorithmus folgendermaßen beschrieben werden:

(1) Sei $x \in \mathbb{R}$ gegeben. Setze $y_0 := a_n$ und $z_0 := a_n$.

(2) Für $k = 1, 2, \ldots, n$ setze $y_k := x y_{k-1} + a_{n-k}$, und für $k = 1, 2, \ldots, n-1$ setze $z_k := x z_{k-1} + y_k$.

(3) Es gilt $f(x) = y_n$ und $f'(x) = z_{n-1}$.

Schreiben Sie ein Java–Programm, welches ein Polynom und dessen Ableitung nach dem Horner–Schema auswertet. Gehen Sie dabei folgendermaßen vor:

- Erstellen Sie eine öffentliche Klasse namens `Hornerschema` mit einer `main`-Methode. Lesen Sie in der `main`-Methode zunächst den Grad n eines Polynoms f ein. Erzeugen Sie als nächstes ein Feld der Länge $n+1$, welches Gleitkommazahlen speichert. Lesen Sie die Koeffizienten $a_0, \ldots, a_n$ von der Konsole ein, und speichern Sie diese im Feld ab. Verwenden Sie dazu eine **`for`**-Schleife.
- Lesen Sie eine Stelle x von der Konsole ein, und berechnen Sie den zugehörigen Wert des Polynoms $f(x)$ sowie den Wert der Ableitung $f'(x)$ mit dem zweistufigen Horner–Schema, und geben Sie beide Ergebnisse auf der Konsole aus. Um die Werte y_k und z_k zwischen zu speichern sollten keine Felder sondern lediglich zwei einfache Variablen verwendet werden.
- Testen Sie Ihr Programm mit den Polynomen $f : \mathbb{R} \to \mathbb{R},\ f(x) := x^3 + 2x^2 + 1$. Es gilt

$\boldsymbol{x}$	-2	-1	0	1	2
$\boldsymbol{f(x)}$	1	2	1	4	17
$\boldsymbol{f'(x)}$	4	-1	0	7	20

Fragen 1 *Das zweistufige Horner–Schema*

- Wann geschehen Typumwandlungen in Java implizit?

Aufgabe 2 *Vektoriteration*

In einem Land haben alle Familien genau einen Sohn und eine Tochter, und jeder heiratet genau einmal. Ferner lässt sich die Bevölkerung in Generationen aufteilen. In der n-ten Generation gibt es x_n Funktionäre, y_n Arbeiter und z_n Bauern. Ehefrauen gehören dem Stand ihres Mannes an. Funktionärssöhne lehnen den Beruf ihres Vaters strikt ab und werden Arbeiter, Söhne von Arbeitern werden Bauern, während Bauernsöhne zu je einem Drittel den genannten drei Bevölkerungsgruppen angehören. Ist dieses System über viele Generationen hinweg stabil? Die mathematische Formulierung dieses einfachen Modells aus der Populationstheorie lautet wie folgt:

$$u^{[n+1]} := \begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1/3 \\ 1 & 0 & 1/3 \\ 0 & 1 & 1/3 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} =: Au^{[n]}, \quad n = 0, 1, 2, \ldots.$$

Der Vektor $u^{[n]}$ repräsentiert hierbei die n-te Generation mit den Bevölkerungsgruppen x_n, y_n und z_n. Die Matrix A stellt die Umstrukturierung in der Gesellschaft dar. Die n-te Generation ergibt sich demnach durch n-maliges Multilpizieren der Matrix A mit sich selbst und dem Produkt dieser Matrix mit dem Startvektor gemäß

$$u^{[n]} = A^n u^{[0]}.$$

Veranschaulichen Sie sich zunächst das Zustandekommen der Matrix A. Beachten Sie, dass die Summe der Elemente in einer Spalte immer gleich eins ist. Erstellen Sie ein Klasse `Vektoriteration` , mit der Sie die Bevölkerungsentwicklung beurteilen können:

- Erstellen Sie zunächst eine Methode `readVec` mit einem eindimensionalen Feld vom Gleitkommatyp als Rückgabewert und der Dimension DIM als Übergabeparameter, die den Startvektor $u^{[0]}$ einliest. Erstellen Sie außerdem eine ebenfalls Funktion `readMat`, die die Matrix A einliest. Die Rückgabe soll ein zweidimensionales Feld vom Gleitkommatyp sein. Der Übergabeparameter ist ebenfalls die ganzzahlige Dimension DIM.

- Schreiben Sie eine Methode `writeVec` ohne Rückgabewert, die als Argument ein Feld vom Gleitkommatyp erhält. Diese Methode soll die Elemente des Vektors auf der Konsole ausgeben.

- Erstellen Sie eine rückgabelose Methode `matPot`, die die n-te Potenz A^n der Matrix A bestimmt. Als Argumente sollen die Matrix A, die ganzzahlige Dimension DIM und die ganze Zahl n übergeben werden. Die Methode soll ein zweidimensionales Feld vom Gleitkommatyp zurückgeben. Zur Berechnung wird eine Hilfsmatrix $Temp$ und die Ergebnismatrix An benötigt, in die mithilfe einer tiefen Kopie die Matrix A abgespeichert wird. Beide Matrizen sind zweidimensionale Felder vom Gleitkommatyp. Anschließend sollen mittels einer for-Schleife, die n mal durchlaufen wird, die Matrizen $Temp$ und A miteinander multipliziert werden und in An abgespeichert werden. An soll am Anfang jedes Schleifendurchlaufs 0 gesetzt werden und am Ende des Durchlaufs soll $Temp$ gleich An gesetzt werden. Am Ende wird An zurückgegeben.

  Hinweis: Für die Multiplikation von zweidimensionalen Matrizen müssen auch zwei for-Schleifen durchlaufen werden.

- Die Funktion `matVecMult` soll das Ergebnis b der Multiplikation der Matrix A mit dem Vektor u, d.h. $Au = b$ berechnen. Dafür sollen die Matrix A und der Vektor u und die ganzzahlige Dimension DIM als Argumente übergeben werden. Die Funktion soll das eindimensionale Feld vom Gleitkommatyp b zurückgeben.

- Schreiben Sie das Hauptprogramm. Definieren Sie eine ganzzahlige Variable DIM, die den Wert 3 haben soll. Erzeugen Sie die benötigten zweidimensionalen Felder vom Gleitkommatyp A und An und eindimensionalen Felder vom Gleitkommatyp $u0$ und u. Zudem soll einer ganzzahlige Variable n, die die Iterationszahl abspeichert, zunächst der Wert 1 zugewiesen werden. Lesen Sie die Matrix A im Hauptprogramm nur einmal ein. Erstellen Sie eine `while`-Schleife, in der Sie das Einlesen des Startvektors $u0$ und von n solange wiederholen bis die Eingabe vom Benutzer mit $n \leq 0$ abgebrochen wird. In der Schleife soll mithilfe des eingelesenen Startvektors das Ergebnis u^n berechnet und ausgegeben werden. Dazu werden die vorher definierten Methoden benötigt.

- Rechnen Sie mit Bevölkerungsanteilen, d.h. für den Startvektor $u^{[0]}$ gilt $0 \leq x_0, y_0, z_0 \leq 1$ und $x_0 + y_0 + z_0 = 1$. Testen Sie verschiedene Startvektoren und verschiedene Werte für n. Welcher stationäre Zustand stellt sich unabhängig vom Startvektor für hinreichend große n ein? Wie hoch ist der Anteil der Bauern?

Musterlösung: stationärer Zustand:

$$u_n = \begin{pmatrix} 1/6 \\ 1/3 \\ 1/2 \end{pmatrix}$$

Aufgabe 3 *ggT und kgV*

Je zwei natürliche Zahlen m und n besitzen einen *größten gemeinsamen Teiler* $\mathrm{ggT}(m,n)$ und ein *kleinstes gemeinsames Vielfaches* $\mathrm{kgV}(m,n)$. Der größte gemeinsame Teiler kann mit dem *Euklidischen Algorithmus* berechnet werden, der folgendermaßen definiert ist:

(1) Falls $m > n$ gilt, setze $p_0 := m$ und $q_0 := n$, ansonsten $p_0 := n$ und $q_0 := m$.

(2) Setze $r_0 := p_0 \bmod q_0$.

(3) Solange $r_k \neq 0$ gilt, setze

$$\begin{aligned} p_{k+1} &:= q_k, \\ q_{k+1} &:= r_k, \\ r_{k+1} &:= p_{k+1} \bmod q_{k+1}, \end{aligned}$$

und erhöhe den formalen Iterationsindex k um Eins.

(4) Der größte gemeinsame Teiler von m und n ist q_{k+1}.

Mit $p_k \bmod q_k$ wird der Rest bezeichnet, der bei ganzzahliger Division von p_k durch q_k entsteht. Für das kleinste gemeinsame Vielfache von m und n gilt

$$\mathrm{kgV}(m,n) = \frac{mn}{\mathrm{ggT}(m,n)}.$$

Schreiben Sie ein Java-Programm, welches den größten gemeinsamen Teiler und das kleinste gemeinsame Vielfache zweier Zahlen berechnet und auf der Konsole ausgibt. Gehen Sie dabei folgendermaßen vor:

(a) Erstellen Sie eine öffentliche Klasse namens `GGTundKGV`. Definieren Sie in dieser Klasse eine öffentliche Klassenmethode namens `zahlenOK` mit zwei formalen Parametern vom Typ **`int`**. In der Methode soll geprüft werden, ob die beiden übergebenen Parameterwerte positive Zahlen sind. Das Ergebnis dieser Überprüfung soll als Wert vom Typ **`boolean`** von der Methode zurück gegeben werden.

(b) Definieren Sie eine öffentliche Klassenmethode namens `ggT` mit zwei formalen Parametern vom Typ **`int`**. Berechnen Sie in dieser Methode den größten gemeinsamen Teiler der übergebenen Parameterwerter. Implementieren Sie dazu den Euklidischen Algorithmus. Der größte gemeinsame Teiler soll von der Methode als Wert vom Typ **`int`** zurück gegeben werden.

Hinweis: Die Werte p_k, q_k und r_k können in jeweils einer einzelnen Variable gespeichert werden. Der formale Iterationsindex k braucht nicht gespeichert zu werden.

(c) Definieren Sie eine öffentliche Klassenmethode namens `kgV` mit zwei formalen Parametern vom Typ **`int`**. Berechnen sie in dieser Methode das kleinste gemeinsame Vielfache der übergebenen Parameterwerte. Verwenden Sie dazu die Methode `ggT`. Das kleinste gemeinsame Vielfache soll von der Methode als Wert vom Typ **`int`** zurück gegeben werden.

(d) Definieren Sie eine öffentliche Klassenmethode namens `ausgabe` ohne Rückgabewert mit zwei formalen Parametern vom Typ **`int`**. Die Methode soll den größten gemeinsamen Teiler und das kleinste gemeinsame Vielfache der übergebenen Parameterwerte auf der Konsole ausgeben.

(e) Definieren Sie als letztes die `main`-Methode des Programms. Lesen in dieser zwei Zahlen vom Typ **`int`** von der Konsole ein. Überprüfen Sie zunächst, ob beide Zahlen positiv sind. Wenn ja, so geben Sie den größten gemeinsamen Teiler und das kleinste gemeinsame Vielfache beider Zahlen auf der Konsole aus. Wenn nicht, so geben Sie eine entsprechende Meldung auf der Konsole aus. Verwenden Sie die Methoden `zahlenOK` und `ausgabe`.

(f) Testen Sie ihr Programm mit dem Zahlenpaar $(12, 3)$, dessen $\mathrm{ggT}\ 3$ ist, mit dem kgV von 12.