Institut für Angewandte und
Numerische Mathematik

Dr. Mathias J. Krause
M. Sc. Albert Mink
M. Sc. Zoltan Veszelka

4.12.2017

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 7

Bearbeitungszeitraum: 4.12.2017 – 15.12.2017

Aufgabe 20 *Die Matrix-Vektor-Multiplikation*

Sei $A \in \mathbb{R}^{m,n}$ eine *Matrix* mit m *Zeilen* und n *Spalten* und $x \in \mathbb{R}^n$ ein n-dimensionaler Vektor, so ist das *Matrix-Vektor-Produkt* Ax ein m-dimensionaler Vektor. Das Matrix-Vektor-Produkt ist folgendermaßen definiert: Sei $y := Ax$, mit

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix},$$

so gilt

$$y_i := \sum_{j=1}^{n} a_{ij} x_j \qquad \text{für } i = 1, 2, \dots, m.$$

Schreiben Sie ein Java-Programm, in dem die Matrix-Vektor-Multiplikation realisiert wird. Gehen Sie dabei folgendermaßen vor.

(a) Erstellen Sie eine öffentliche Klasse namens `MatrixVektor`. Definieren Sie in dieser Klasse zunächst eine Klassenmethode namens `matrixEinlesen` mit zwei formalen Parametern vom Typ **`int`**. Die Methode soll mittels zweier **`for`**–Schleifen eine Matrix mit m Zeilen und n Spalten von der Konsole einlesen und als Wert vom Typ **`double`**`[][]` zurückgeben. Die Matrixdimensionen m und n sollen dabei über die formalen Parameter übergeben werden. Definieren Sie in analoger Weise eine Klassenmethode namens `vektorEinlesen` mit einem formalen Parameter vom Typ **`int`**, die einen n-dimensionalen Vektor von der Konsole einliest und als Wert vom Typ **`double`**`[]` zurück gibt.

(b) Definieren Sie nun eine Klassenmethoden namens `vektorAusgeben` mit einem formalen Parameter vom Typ **`double`**`[]` ohne Rückgabewert. Die Methode soll das übergebene Feld auf der Konsole ausgeben.

(c) Definieren Sie eine Klassenmethode namens `matrixvektorProdukt` mit zwei formalen Parametern vom Typ **`double`**`[][]` und **`double`**`[]`. Über den ersten Parameter soll eine Matrix A, über den zweiten ein Vektor x übergeben werden. Berechnen Sie in der Methode das Matrix-Vektor-Produkt Ax, und geben Sie das Ergebnis als Wert vom Typ **`double`**`[]` zurück.

Hinweis: Vorsicht! Matrix- und Vektorkomponenten werden üblicherweise mit Eins beginnend durchnumeriert. Die Komponenten eines Feldes hingegen werden in Java mit Null beginnend durchnumeriert.

(d) Definieren Sie nun die `main`-Methode des Programms. Lesen Sie in dieser Methode zunächst die Dimensionen m und n von der Konsole ein. Lesen Sie danach eine Matrix $A \in \mathbb{R}^{m,n}$ und einen Vektor $x \in \mathbb{R}^n$ von der Konsole ein. Berechnen Sie das Matrix-Vektor-Produkt Ax, und geben Sie dieses auf der Konsole aus.

(e) Testen Sie ihr Programm mit folgenden Eingaben.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \qquad x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \qquad Ax = \begin{pmatrix} 14 \\ 32 \\ 50 \end{pmatrix}.$$

und

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \qquad x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad Ax = \begin{pmatrix} 5 \\ 11 \\ 17 \end{pmatrix}.$$

Aufgabe 21 (Pflichtaufgabe) *Spiel des Lebens*

Das „Spiel des Lebens“ soll für eine Matrix von $n \times n$ ($n = 10$) Zellen programmiert werden, d.h. man hat n Zeilen mit je n Zellen. Eine Zelle kann sich in genau einem der Zustände `LEBT` oder `TOT` befinden. Als Nachbarn einer Zelle bezeichnet man alle Zellen, die links, rechts, oberhalb, unterhalb oder diagonal versetzt zu dieser Zelle liegen. (**Vorsicht**: am Rand existieren nicht alle diese Zellen.) Eine Zelle im Innern hat also genau acht, Randzellen haben fünf und Eckzellen haben genau drei Nachbarzellen.

Die Zellen sind zeilenweise wie folgt indiziert:

$$\begin{matrix} (1,1) & (1,2) & \dots & (1,n) \\ (2,1) & (2,2) & \dots & (2,n) \\ \vdots & \vdots & & \vdots \\ (n,1) & (n,2) & \dots & (n,n) \end{matrix}$$

Beispielsweise hat die Eckzelle $(1,1)$ die Nachbarn $(1,2)$, $(2,1)$ und $(2,2)$.

Ausgehend von einer Anfangsgeneration (jede Zelle ist entweder im Zustand `TOT` oder `LEBT`; der Anfangszustand soll zufällig bestimmt werden) wird eine neue Zellgeneration nach folgenden Regeln erzeugt:

- Eine Zelle wird (unabhängig von ihrem derzeitigen Zustand) in der nächsten Generation `TOT` sein, wenn sie in der jetzigen Generation weniger als zwei (Vereinsamung) oder mehr als drei (Überbevölkerung) lebende Nachbarn besitzt.

- Eine Zelle mit genau zwei lebenden Nachbarn ändert ihren Zustand nicht.
- Eine Zelle mit genau drei lebenden Nachbarn wird sich in der nächsten Generation im Zustand `LEBT` befinden.

Hinweis: Man darf erst dann die alte Generation ändern, wenn alle Entscheidungen für die neue Generation getroffen sind.

(a) Erstellen Sie eine Klasse `Zellen` mit einer Methode `Genesis`. Hier soll eine boolsche Matrix erzeugt werden, welche den Startzustand der Zellkolonie repräsentiert. Dabei steht der boolsche Wert **`true`** für den Zustand `LEBT` und **`false`** für `TOT`. Diese Matrix wird zum Schluss zurückgegeben – der Rückgabetyp ist also **`boolean`**`[][]`. Als Argument erhält die Methode die Dimension `N` der Matrix sowie die Anzahl der zu Beginn lebenden Teilchen, welche zufällig auf die Matrix verteilt werden. Es müssen also folgende Schritte durchgeführt werden:

- Erstellen Sie eine $N \times N$ Matrix vom Typ **`boolean`**.
- Erzeugen Sie sich zunächst zufällige Indizes `i` und `j`.
  Mit dem Ausdruck `(`**`int`**`)(Math.random()*N)` erhalten Sie eine zufällige ganze Zahl im Bereich $\{0, ..., N-1\}$.
- Prüfen Sie zunächst, ob dieses Zelle nicht bereits lebendig ist, bevor Sie ihr den Wert **`true`** zuweisen. So können Sie mitzählen, wie viele Zellen bisher aktiviert wurden.
- Wiederholen Sie dies so lange, bis genügend Zellen lebendig sind.

(b) Schreiben Sie eine Methode `AnzNachbarn`, welche bestimmen soll, wie viele (lebendige) Nachbarn eine bestimmte Zelle besitzt. Übergeben Sie dazu die boolsche Zellmatrix sowie die Koordinaten der Zelle als Argument. Achten Sie beim Zählen darauf, dass Randzellen weniger Nachbarn besitzen und die Methode nur auf die gültigen Bereiche der Matrix zugreift. Weiterhin zählt eine Zelle selbst *nicht* als einer ihrer Nachbarn. Geben Sie die Anzahl der lebenden Nachbarzellen anschließend zurück.

(c) Schreiben Sie eine Methode `NeueGeneration` in der aus einer gegebenen Zellmatrix die Matrix der nachfolgenden Generation berechnet und zurückgegeben werden soll. Tragen Sie die neuen Zustände in einer neu erstellten boolschen Matrix ein. Nutzen Sie für jede Koordinate die Methode `AnzNachbarn` um zu wissen, wie der neue Zustand der gegebenen Zelle lauten muss.

(d) Schreiben Sie eine Methode `Ausgabe` vom Typ **`void`**. Diese soll eine übergebene boolsche Zellmatrix anschaulich auf der Konsole ausgeben. Beispielsweise können Sie lebende Zellen durch ein Sternchen ($*$) und tote Zellen durch ein Leerzeichen veranschaulichen.

(e) Schreiben Sie die main-Methode der Klasse. Hier soll eine entsprechende Kolonie erstellt und ihre Entwicklung simuliert werden. Geben Sie nach jeder neuen Generation die aktuelle Zellmatrix auf dem Bildschirm aus.