Prof. Andreas Goerdt
M. Pippig, M. Riedel, U. Schönke

TU Chemnitz
Wintersemester 2003/2004

Algorithmen und Programmierung

8. Übung – Lösungsvorschläge

1. Aufgabe:

```
import java.util.*;
import Prog1Tools.IOTools;
public class BinSucheFelder{
  public static void main(String[] args) {

    int n, i;
    int [] feld;
    int links, rechts, mitte, gesucht;
    int zzAlt, zzNeu, a, c, m;

    /***************** Initialisierungen *****************/
    // Initialisierung des Zufallszahlengenerators
    System.out.println("Initialisierung␣des␣ZZ-Generators");
    zzAlt = IOTools.readInteger("Bitte␣den␣Seed␣eingeben:␣");
    a = IOTools.readInteger("Bitte␣a␣eingeben:␣");
    c = IOTools.readInteger("Bitte␣c␣eingeben:␣");
    m = IOTools.readInteger("Bitte␣m␣eingeben:␣");
    System.out.println();

    // Initialisierung des Feldes
    n = IOTools.readInteger("Bitte␣die␣Feldgroesse␣eingeben:␣");
    feld = new int[n];
    for (i=0; i<feld.length; i++) {
      zzNeu = ((a * zzAlt) + c) % m;
      // Achtung! Werden nicht nur pos. Zahlen gewuenscht,
      // dann Folgendes auskommentieren:
      if (zzNeu < 0)
        zzNeu = zzNeu + m;
      zzAlt=zzNeu;
      feld[i]=zzNeu;
    }
    System.out.println();

    // Feld sortieren
    Arrays.sort(feld);
```

```
    for (i=0; i<feld.length; i++)
      System.out.print(feld[i] + " ");
    System.out.println();

    // gesuchtes Element einlesen
    gesucht = IOTools.readInteger("Gesuchtes Element eingeben: ");

    /****************** binaere Suche ******************/
    // Initialisierung
    links = 0;
    rechts = feld.length - 1;

    // Suche
    while (links < rechts) {
      mitte = (links + rechts) / 2;
      if (gesucht > feld[mitte])
          links = mitte + 1;
      else
          rechts = mitte;
    }
    if (gesucht == feld[links])
      System.out.println("Gefunden an Pos. " + links);
    else
      System.out.println("Element ist nicht enthalten");

    return;
  }
}
```

Die Laufzeit der binären Suche ist logarithmisch, während die Laufzeit des einfachen sequentiellen Durchsuchens linear ist.

2. Aufgabe:

Seien f_l das Feld `feld` sowie i_l und j_l die Werte der Variablen i und j nach dem l-ten Schleifendurchlauf (der äußeren Schleife).

Inv1

Induktionsanfang

Für $l = 0$ (vor dem ersten Schleifendurchlauf) gilt:

$$\begin{aligned} i_0 &= -1 \\ j_0 &= \texttt{feld.length} \\ x &= f_0[k] \end{aligned}$$

Da $i_0 < k < j_0$, gilt mit $i' = j' = k$ die Invariante.

Induktionsschritt

Für $l \geq 1$ und $i_{l-1} < j_{l-1}$ gilt nach Voraussetzung (Invariante):

$$\begin{array}{ll} \exists i' \geq i_{l-1} & \text{mit} \quad f_{l-1}[i'] \geq x \\ \exists j' \leq j_{l-1} & \text{mit} \quad f_{l-1}[j'] \leq x \end{array}$$

Wir betrachten nun den l-ten Durchlauf (nicht der letzte!):

Es gelte

$$i_l < j_l$$

(sonst Betrachtung der Invariante nicht möglich).

Die erste `do-while`-Schleife hält beim ersten Element von links, das größer oder gleich x ist, an. Dessen Index ist zugleich das i_l:

$$i_{l-1} < i_l \leq i' \text{ mit } f_l\left[i_l\right] \geq x$$

Analog hält die zweite `do-while`-Schleife beim ersten Element von rechts, das kleiner oder gleich x ist, an. Dessen Index ist das j_l:

$$j' \leq j_l < j_{l-1} \text{ mit } f_l\left[j_l\right] \leq x$$

Da $i_l < j_l$, werden die Werte $f_l\left[i_l\right]$ und $f_l\left[j_l\right]$ getauscht. i_l und j_l werden dabei nicht verändert, daher gilt die Invariante mit

$$\begin{array}{rcl} i' & = & j_l \\ j' & = & i_l \end{array}$$

Inv2

Sei $n =$ `feld.length`.

Induktionsanfang

Die Invariante gilt für i_0, j_0 und f_0 per Definition, da

$$\begin{array}{lcl} f_0[0], \ldots, f_0[-1] & \Rightarrow & \text{leere Folge} \\ f_0[n], \ldots, f_0[n-1] & \Rightarrow & \text{leere Folge} \end{array}$$

Induktionsschritt

Für $l \geq 1$ und $i_{l-1} \leq j_{l-1}$ gilt nach Voraussetzung (Invariante):

$$\begin{array}{lcl} f_{l-1}[0], \ldots, f_{l-1}\left[i_{l-1}\right] & \leq & x \\ f_{l-1}\left[j_{l-1}\right], \ldots, f_{l-1}[n-1] & \geq & x \end{array}$$

Wir betrachten nun den l-ten Durchlauf: Die erste `do-while`-Schleife hält beim ersten Element von links, das größer oder gleich x ist, an. Dessen Index ist zugleich das i_l:

$$i_{l-1} < i_l \leq i' \text{ mit } f\left[i_l\right] \geq x$$

Analog hält die zweite `do-while`-Schleife beim ersten Element von rechts, das kleiner oder gleich x ist, an. Dessen Index ist das j_l:

$$j' \leq j_l < j_{l-1} \text{ mit } f\left[j_l\right] \leq x$$

Dann ist

$$\begin{array}{lll} f_l\left[0\right], \ldots, f_l\left[i_l - 1\right] & \leq & x \\ f_l\left[j_l + 1\right], \ldots, f_l\left[n-1\right] & \geq & x \end{array}$$

Falls $i_l < j_l$, dann werden die Werte $f_l\left[i_l\right]$ und $f_l\left[j_l\right]$ getauscht. Damit ist

$$\begin{array}{lll} f_l\left[0\right], \ldots, f_l\left[i_l\right] & \leq & x \\ f_l\left[j_l\right], \ldots, f_l\left[n-1\right] & \geq & x \end{array}$$

und die Invariante gilt.

Falls jedoch $i_l = j_l$, dann ist

$$\begin{array}{lll} f_l\left[i_l\right] & \geq & x \\ f_l\left[j_l\right] & \leq & x \quad \text{und} \\ i_l = j_l \Rightarrow f_l\left[i_l\right] = f_l\left[j_l\right] = x & & \end{array}$$

und die Invariante gilt mit

$$\begin{array}{lll} f_l\left[0\right], \ldots, f_l\left[i_l\right] & \leq & x \\ f_l\left[j_l\right], \ldots, f_l\left[n-1\right] & \geq & x \end{array}$$

Zum Beweis der Korrektheit des gesamten Algorithmus müsste der letzte Schleifendurchlauf noch gesondert betrachtet werden.

3. Aufgabe:

   Seien f_l das Feld `feld` sowie i_l und j_l die Werte der Variablen i und j nach dem l-ten Schleifendurchlauf (der äußeren Schleife). Weiterhin seien $i_{l,temp}$ und $j_{l,temp}$ die Werte der Variablen i und j nach Abarbeitung der beiden inneren `while`-Schleifen während des l-ten Durchlaufes der äußeren Schleife.

   Induktionsanfang

   Für $l = 0$ gilt:

$$\begin{aligned} i_0 &= 0 \\ j_0 &= n-1, \quad n \geq 1 \text{ (sonst nicht sinnvoll)} \end{aligned}$$

   Es ist $0-(n-1) = -n+1 \leq 0 < 2$, also gilt die Invariante.

   Induktionsschritt

   Für $l \geq 1$ gilt nach Voraussetzung (Invariante): $i_{l-1} - j_{l-1} \leq 2$

   Wir betrachten nun den l-ten Durchlauf:

   Die erste `while`-Schleife hält beim ersten Element von links, das größer oder gleich x ist, an:

$$i_{l-1} \leq i_{l,temp} \text{ mit } f\left[i_{l,temp}\right] \geq x$$

   Analog hält die zweite `while`-Schleife beim ersten Element von rechts, das kleiner oder gleich x ist, an:

$$j_{l,temp} \leq j_{l-1} \text{ mit } f\left[j_{l,temp}\right] \leq x$$

   Falls $i_{l,temp} < j_{l,temp}$, dann gilt

$$\begin{aligned} i_l &= i_{l,temp}+1 \\ j_l &= j_{l,temp}-1 \end{aligned}$$

   i_l kann um höchstens 1 größer sein als j_l (nämlich genau bei $i_{l,temp} = j_{l,temp} - 1$), somit ist $i_l - j_l \leq 1$ und die Invariante gilt.

   Falls $i_{l,temp} = j_{l,temp}$, dann gilt

$$\begin{aligned} i_l &= i_{l,temp}+1 \\ j_l &= j_{l,temp}-1 \end{aligned}$$

   und $i_l - j_l = i_{l,temp} + 1 - \left(i_{l,temp} - 1\right) = 2$, folglich gilt die Invariante.

   Falls $i_{l,temp} > j_{l,temp}$, dann gilt

$$\begin{aligned} i_l &= i_{l,temp} \\ j_l &= j_{l,temp} \end{aligned}$$

   In diesem Fall sind alle Elemente von $f_l[0]$ bis $f_l[j_l]$ kleiner gleich x und alle Elemente von $f_l[i_l]$ bis $f_l[n-1]$ größer gleich x. i_l und j_l sind benachbarte Indizes, also
   $i_l - j_l = 1 \leq 2$, womit wiederum die Invariante gilt.

4. Aufgabe:

```
import Prog1Tools.IOTools;

public class GgtUndKgv {
  public static void main(String[] args) {

    int a, b;
    int u, v, x, y;

    a = IOTools.readInteger("Bitte a eingeben: ");
    b = IOTools.readInteger("Bitte b eingeben: ");

    x = a; y = b;
    u = a; v = b;

    while ( x != y) {
      if (x < y) {
        y = y - x;
        v = v + u;
      }
      else {
        x = x - y;
        u = u + v;
      }
      System.out.println("u = " + u + " und v = " + v);
    }
    System.out.println("Der ggT ist: " + x);
    System.out.println("Das kgV ist: " + (u + v) / 2);
    return;
  }
}
```

Der Algorithmus von GILL berechnet den größten gemeinsamen Teiler und das kleinste gemeinsame Vielfache von *a* und *b*.

5. Aufgabe:

```
import Prog1Tools.IOTools;
public class Tuerme {
  public static void main(String[] args) {
    int n; // Groesse
    int [][] matrix;
    int i, j;
    int [] zeile, spalte;
    boolean ok;

    // Initialisierung der Matrix
    n = IOTools.readInteger("Dimension des Feldes eingeben: ");
    matrix = new int [n][n];
    for (i=0; i<matrix.length; i++)
      for (j=0; j<matrix.length; j++)
        matrix[i][j]=IOTools.readInteger(
                    "matrix[" + i + "][" + j + "] = ");

    // Initialisierung der Hilfsvektoren
    zeile = new int[n];
    spalte = new int[n];
    for (i=0; i<matrix.length; i++) {
      zeile[i] = 0;
      spalte[i] = 0;
    }

    // Tuerme in jeder Zeile und Spalte zaehlen
    for (i=0; i<matrix.length; i++)
      for (j=0; j<matrix.length; j++) {
        zeile[i] = zeile[i] + matrix[i][j];
        spalte[j] = spalte[j] + matrix[i][j];
      }

    // Auswertung: in jeder Zeile und Spalte genau 1 Turm?
    ok = true;
    for (i=0; i<matrix.length; i++)
      ok = ok && (zeile[i] == 1) && (spalte[i] == 1);

    // Ausgabe
    if (ok)
      System.out.println("Alles okay");
    else
      System.out.println("Bedingungen nicht erfuellt");
    return;
  }
}
```

```
import Prog1Tools.IOTools;
public class Tuerme2 {
  public static void main(String[] args) {
    int n; // Groesse
    int [][] matrix;
    int i, j;
    int zeilensumme, spaltensumme;
    boolean ok;

    // Initialisierung der Matrix
    n = IOTools.readInteger("Dimension des Feldes eingeben: ");
    matrix = new int [n][n];
    for (i=0; i<matrix.length; i++)
      for (j=0; j<matrix.length; j++)
        matrix[i][j]=IOTools.readInteger(
                    "matrix[" + i + "][" + j + "] = ");

    // Ausgabe der Matrix
    for (i=0; i<matrix.length; i++) {
      for (j=0; j<matrix.length; j++)
        System.out.print(matrix[i][j] + " ");
      System.out.println();
    }

    // Tuerme in jeder Zeile und Spalte zaehlen und Auswertung
    ok = true;
    for (i=0; i<matrix.length; i++) {
      zeilensumme = 0;
      spaltensumme = 0;
      for (j=0; j<matrix.length; j++) {
        zeilensumme = zeilensumme + matrix[i][j];
        spaltensumme = spaltensumme + matrix[j][i];
      }
      if ((zeilensumme != 1) || (spaltensumme !=1))
        ok = false;
    }

    // Ausgabe
    if (ok)
      System.out.println("Alles okay");
    else
      System.out.println("Bedingungen nicht erfuellt");

    return;
  }
}
```