Westfälische Wilhelms-Universität Münster
Übungen zur Vorlesung „Datenstrukturen und Algorithmen“ im SoSe 2017

Prof. Dr. Klaus Hinrichs
Aaron Scherzinger

Blatt 2

Abgabe im Learnweb bis zum 11.05.2017 um 10 Uhr

Aufgabe 5: (5+5=10 Punkte) Betrachten Sie ein System $\mathcal{S}$ von Gleitkommazahlen mit drei Oktalziffern in der Mantisse und einer Oktalziffer im Exponenten: Jede Zahl $x \in \mathcal{S}$ ist somit von der Form $x = \pm 0.d_1d_2d_3 \cdot 8^e$ mit $d_1, d_2, d_3, e \in \{0, \ldots, 7\}$.

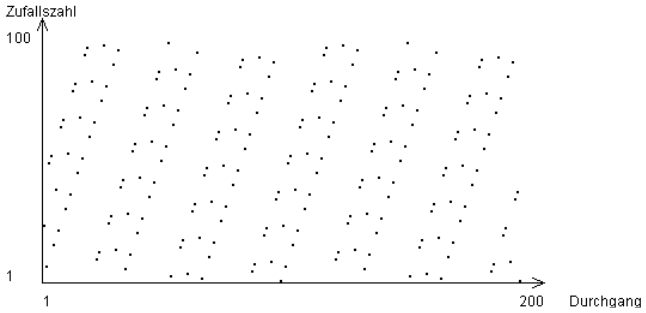
(a) Wie viele verschiedene Zahlendarstellungen gibt es in dem System $\mathcal{S}$?

(b) Wie viele verschiedene Zahlen können in dem System $\mathcal{S}$ dargestellt werden? Begründen Sie Ihre Lösung.

Aufgabe 6: (8+8+8=24 Punkte) Bearbeiten Sie die folgenden Aufgabenteile!

(a) Entwickeln Sie ein `JAVA`-Programm zur Visualisierung der Einträge eines Arrays `int[] a`, das ganze Zahlen im Bereich $[1, \ldots, 100]$ enthält. Ihre Bildschirmausgabe soll eine Matrix mit 100 Zeilen und `a.length` Spalten darstellen, in der in der i-ten Spalte der j-te Eintrag genau dann markiert ist, wenn `a[i]=j` gilt.

Die Ausgabe der Matrix soll grafisch mithilfe des zur Verfügung gestellten Programmrahmens realisiert werden. Fügen Sie der dort vorhandenen Klasse `ZufallszahlenVisualisierung` hierzu zunächst das Array `a` als Attribut hinzu. Für die Visualisierung der Zahlen müssen sie die Methode `paintComponent` der Klasse implementieren. Machen Sie sich hierzu mit der Funktionalität der Klassen `java.awt.Graphics`, `java.awt.Color` und `java.awt.Point` vertraut. In den Kommentaren zum Programmrahmen finden Sie bereits einige Beispiele. Beachten Sie, dass das Fenster skaliert werden kann und behandeln Sie dies, indem Sie die Größe der Zeichenfläche in der Methode `paintComponent` in Ihre Visualisierung einbeziehen. Eine Möglichkeit der grafischen Visualisierung ist nachfolgend vorgestellt. Hier wurde das Array `a` mit den ersten 200 Werten gefüllt, die der Zufallszahlengenerator $r_{i+1} = (21 \cdot r_i + 23) \mod 100$ liefert.



Applikation von H. Kösters, WWU Münster

(b) Testen Sie Ihre Umgebung, indem Sie das Array mit den ersten n Werten füllen, die die folgenden Zufallszahlengeneratoren liefern:

- Der Zufallszahlengenerator, der durch die Gleichung $r_{i+1} = (21 \cdot r_i + 23) \mod 100$ und den Startwert $r_0 = 1$ gegeben ist.
- Der Zufallszahlengenerator, der durch die Gleichung $r_{i+1} = (21 \cdot r_i + 7) \mod 100$ und den Startwert $r_0 = 1$ gegeben ist.
- Ein Zufallszahlengenerator, der von `JAVA` bereitgestellt wird (Wie kann man die dort erzeugten Werte auf den Bereich $[1, \ldots, 100]$ abbilden?).

Modellieren Sie hierzu eine abstrakte Klasse `ZufallszahlenGenerator`, die die Verwendung der oben genannten Generatoren vereinheitlicht.

(c) Verwenden Sie Ihre bisher erzielten Ergebnisse, um alle 95 Zufallszahlgeneratoren, die nach der linearen Kongruenzmethode durch Inkrement $c = 0$, Startwert $r_0 = 1$ und Multiplikator $1 < a < 97$

für den Modulus $m = 97$ gegeben sind, zu untersuchen. Identifizieren Sie dabei diejenigen Zufallszahlgeneratoren, die sich bereits auf Grundlage eines einfachen visuellen Tests für den praktischen Einsatz ausschließen lassen. Sie können davon ausgehen, dass die Länge einer auftretenden Periode maximal 97 beträgt.

Aufgabe 7: (4+4+(2+3+5+2)=20 Punkte) Die Folge der Fibonacci-Zahlen

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, \ldots$$

ist definiert durch $x_0 = 0$, $x_1 = 1$ und $x_n = x_{n-1} + x_{n-2}$ $\forall n > 2$. Bearbeiten Sie dazu die folgenden Aufgaben!

(a) Entwerfen Sie eine rekursive Methode in `JAVA`, die zu einer gegebenen ganzen Zahl n die n-te Fibonacci-Zahl ermittelt. Ihre Methode sollte folgende Signatur besitzen:

```
public int recursiveFib(short n)
```

Testen Sie Ihre Methode (mindestens) für $n \in \{-1, 0, 1, 2, 5, 20, 40\}$.

(b) Entwerfen Sie eine iterative Methode in `JAVA`, die zu einer gegebenen ganzen Zahl n die n-te Fibonacci-Zahl ermittelt. Ihre Methode sollte folgende Signatur besitzen:

```
public int iterativeFib(short n)
```

Testen Sie Ihre Methode (mindestens) für $n \in \{-1, 0, 1, 2, 5, 20, 40\}$.

(c) Gehen Sie nun davon aus, dass Ihre Programmiersprache `JAVA` nur 16-Bit-Arithmetik (Datentyp `short`) unterstützt. Um dennoch alle Fibonacci-Zahlen bis zur Größe 10^8 bestimmen zu können. greifen Sie daher auf modulare Arithmetik zurück.

(i) Welches Intervall ganzer (positiver) Zahlen kann eindeutig durch ein 3-Tupel $[r_1, r_2, r_3]$ der zugehörigen Reste bezüglich der Moduli $m_1 = 999$, $m_2 = 1000$ und $m_3 = 1001$ dargestellt werden?

(ii) Beschreiben Sie umgangssprachlich und formal die Berechnung des 3-Tupel $[r_1, r_2, r_3]$, das eine Zahl r aus dem zuvor bestimmten Intervall bezüglich der angegebenen Moduli eindeutig repräsentiert.

(iii) Enwickeln Sie ausgehend von Ihrer Lösung aus Aufgabenteil (b) eine Methode in `JAVA`, die zu einer gegebenen ganzen Zahl n die n-te Fibonacci-Zahl gemäß modularer Arithmetik ermittelt. Als Rückgabe dieser Methode soll ein Array vom Typ `short` dienen, das die drei Reste r_1, r_2, r_3 enthält, die die n-te Fibonacci-Zahl repräsentieren. Ihre Methode sollte daher folgende Signatur besitzen:

```
public short[] modularFib(short n)
```

Testen Sie Ihre Methode (mindestens) für $n \in \{-1, 0, 1, 2, 5, 20, 40\}$.

(iv) Bestimmen Sie die größte Fibonacci-Zahl, die sich in durch Ihre Methode mit modularer Arithmetik ermitteln lässt. Um die wievielte Fibonacci-Zahl handelt es sich dabei?