Übungen Programmieren in Clojure Serie 5

1. Zufallszahlen

(a) Schreiben Sie eine Funktion, die eine verzögerte Folge (*lazy seq*) von Zufallszahlen im Bereich $[0, n)$ erzeugt.

(b) Simulieren Sie die Ziehung der Lottozahlen 6 aus 49.

2. Skalarprodukt

Gegeben zwei Vektoren (von Zahlen) $x = [x_1, x_2, \ldots, x_n]$ und $[y = y_1, y_2, \ldots, y_n]$ gleicher Länge n, berechnet man das Skalarprodukt (englisch auch *dot product*) als

$$x \cdot y = \sum_{i=1}^{n} x_i y_i$$

Schreiben Sie eine Funktion, die das Skalarprodukt zweier Vektoren berechnet.

3. Collatz-Folge

Die sogenannte Collatz[1]-Folge zu einer natürlichen Zahl n wird folgendermaßen gebildet:

$$n_1 = n$$
$$n_{i+1} = \begin{cases} n_i/2 & \text{falls } n_i \text{ gerade} \\ 3n_i + 1 & \text{falls } n_i \text{ ungerade} \end{cases}$$

Startet man etwa mit der Zahl 7 erhält man

$$7\ 22\ 11\ 34\ 17\ 52\ 26\ 13\ 40\ 20\ 10\ 5\ 16\ 8\ 4\ 2\ 1\ 4\ 2\ 1\ \ldots$$

Wie man sieht, geht die Folge schließlich in den Zyklus $1, 4, 2$ über. Die *Collatz-Vermutung*[2] besagt, dass dies für jeden Startwert n der Fall ist, d.h. jede Collatz-Folge erreicht irgendwann den Wert 1.

(a) Konstruieren Sie eine verzögerte Folge für die Collatz-Folge beginnend mit der natürlichen Zahl $n > 0$.

(b) Verwenden Sie diese Folge, um zu einer Zahl n zu berechnen, wieviele Schritte benötigt werden, bis die Collatz-Folge den Wert 1 erreicht.

4. Primzahlen

(a) Schreiben Sie eine Funktion `(prime? n)`, die testet, ob die natürliche Zahl n eine Primzahl ist.

(b) Schreiben Sie eine Funktion `(primes n)`, die eine Folge der ersten n Primzahlen erzeugt.

[1] Lothar Collatz, deutscher Mathematiker 1910 - 1990
[2] Die Vermutung wurde bisher nicht bewiesen.

(c) Konstruieren Sie eine verzögerte Folge `(primes)` aller Primzahlen.

(d) Kann man `(prime? n)` effizienter machen, indem man nur Primzahlen zum Test nimmt, ob sie die Zahl n teilen?

5. Perfekte Zahlen

Eine natürliche Zahl $n > 0$ heißt *perfekt*, wenn sie die Summe ihrer echten Teiler ist. Die kleinste perfekte Zahl ist 6, es gilt $6 = 1 + 2 + 3$.

(a) Schreiben Sie eine Funktion `(perfect? n)`, die testet, ob eine natürliche Zahl $n > 0$ perfekt ist.

(b) Konstruieren Sie eine verzögerte Folge der perfekten Zahlen.

(c) Geben Sie die ersten 4 perfekten Zahlen aus und messen Sie die Laufzeit. (Probieren Sie es besser nicht mit 5, es sei denn, Sie sind sehr geduldig.)

6. Kettenbrüche

Ein regulärer Kettenbruch ist ein fortgesetzter Bruch der Form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots}}}}$$

Man beschreibt einen regulären Kettenbruch oft dadurch, dass man die Folge der a_i angibt, typischerweise in der Literatur so:

$$[a_0; a_1, a_2, a_3, \ldots]$$

Zu einem solchen Kettenbruch kann man eine Folge von Näherungen bilden, indem man die Näherung für die ersten n der a_i berechnet. Dies ergibt einen Näherungsbruch $\frac{p_n}{q_n}$, den man nach folgenden Formeln berechnen kann[3]:

$$p_{n+1} = a_{n+1} p_n + p_{n-1} \text{ mit } p_{-1} = 1 \text{ und } p_{-2} = 0$$

$$q_{n+1} = a_{n+1} q_n + q_{n-1} \text{ mit } q_{-1} = 0 \text{ und } q_{-2} = 1$$

Reguläre Kettenbrüche sind interessant, weil sie interessante Zahlen approximieren. So gilt z.B.

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{\ddots}}}}$$

[3] siehe etwa das Skript von Tomas Sauer, JLU Gießen aus dem Wintersemester 2004/05 über Kettenbrüche, http://www.uni-giessen.de/tomas.sauer/homepage_d.html.

D.h. die Folge $[1; 2, 2, 2, 2, \dots]$ führt zu Näherungsbrüchen, die nach $\sqrt{2}$ konvergieren:

$$1 \quad \frac{3}{2} \quad \frac{7}{5} \quad \frac{17}{12} \quad \frac{41}{29} \quad \frac{99}{70} \quad \frac{239}{169} \quad \frac{577}{408} \quad \frac{1393}{985} \quad \frac{3363}{2378} \quad \dots$$

Die ersten 6 Nachkommastellen des letzten Bruchs stimmen mit der Dezimaldarstellung von $\sqrt{2}$ überein.

Auch die Eulersche Zahl e kann durch einen Kettenbruch approximiert werden, nämlich durch den Kettenbruch mit folgender Folge der a_i[4]:

$$[2; 1, 2, 1,\ 1, 4, 1,\ 1, 6, 1,\ \dots,\ 1, 2k, 1,\ \dots]$$

(a) Schreiben Sie eine Funktion `(continued-frac a-seq)`, die zu einer Folge der a_i eine verzögerte Folge der Näherungsbrüche (wie oben definiert) erzeugt.

(b) Definieren Sie die Folge der a_i für $\sqrt{2}$ und verwenden Sie `continued-frac`, um die ersten 10 Näherungsbrüche zu berechnen.

(c) Definieren Sie die Folge der a_i für die Eulersche Zahl e und verwenden Sie `continued-frac`, um die ersten 10 Näherungsbrüche zu berechnen.

7. Funktionen aus Mathematica

Mathematica hat zwei oft benötigte Funktionen:

„Many programs you write will involve operations that need to be iterated several times. Nest and NestList are powerful constructs for doing this.

`Nest[f,x,n]` apply the function f nested n times to x.
`NestList[f,x,n]` generate the list `{x, f[x], f[f[x]],...}`, where f is nested up to n deep.“

[http://reference.wolfram.com/language/tutorial/ApplyingFunctionsRepeatedly.html]

Schreiben Sie die beiden Funktionen in Clojure.

8. FizzBuzz

Auf http://www.c2.com/cgi/wiki?FizzBuzzTest findet man folgendes:

The „Fizz-Buzz test“ is an interview question designed to help filter out the 99.5% of programming job candidates who can't seem to program their way out of a wet paper bag. The text of the programming assignment is as follows:

> „Write a program that prints the numbers from 1 to 100. But for multiples of three print `Fizz` instead of the number and for the multiples of five print `Buzz`. For numbers which are multiples of both three and five print ‚FizzBuzz'.“

Lösen Sie die Aufgabe mit Clojure.

Rev 4.0 – 11. August 2017

[4] siehe auch Problem 65 des Projekts Euler http://projecteuler.net