Übung zur VL „Grundlagen der Programmierung“

5. Übung

Dr. Zubow

Arrays (Felder)

- Aufgabe:
  - Geben Sie den Inhalt des Arrays nach Ausführung der folgenden Anweisungen an:

```
def a = new int[99];

for (def i = 0; i < 99; i++)
     a[i] = 98-i;

for (def i = 0; i < 99; i++)
     a[i] = a[a[i]];
```

Geburtstagsparadoxon

Personen betreten nacheinander einen leeren Raum solange bis zwei Personen am gleichen Tag Geburtstag haben. Wie lange wird es durchschnittlich dauern?

- Annahme: Geburstage sind gleichverteilte Zufallszahlen zwischen 0 und N-1.
- `days[d] = true` wenn am Tag `d` bereits jemand Geburtstag hat, ansonsten `false`

N = 365

```
def N = 365;
def days = new boolean[N];
def count = 0;
while (true) {
 count++;
 def d = (int) (N * Math.random());
 if (days[d]) break;
 days[d] = true;
}
println(count);
```

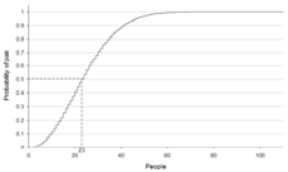
Voreinstellung: false

Zwischen 0 und N-1

Geburtstagsparadoxon (analytisch)

- Sei $\bar{p}(n)$ die Wahrscheinlichkeit, dass alle n Geburstage unterschiedlich sind:
  - $p(n)$=0 wenn $n > 365$
  - Wenn $n \leq 365$:

$$\bar{p}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right)$$
$$= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}$$
$$= \frac{365!}{365^n (365 - n)!}$$



- Die Wahrscheinlichkeit, dass mindestens 2 der n Personen am selben Tag Geburtstag haben ist:

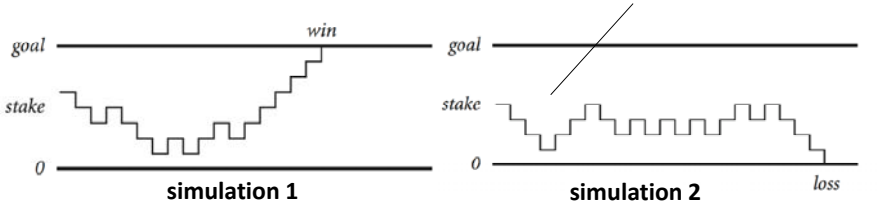
$$p(n) = 1 - \bar{p}(n).$$

Monte Carlo Simulation

Gambler's ruin. Der Spieler startet mit $stake und platziert $1 Einsätze (bets) bis er entweder bankrott ist oder sein Ziel ($goal) erreicht hat.

- Wie hoch ist die Chance zu gewinnen?
- Wieviele Einsätze muss er tätigen?
- Ein Ansatz: Monte Carlo Simulation.
- Werfe eine digitale Münze und schaue was passiert.
- Wiederhole das Experiment und berechne Statistiken.

Ähnlich der Brownsche Bewegung



simulation 1
Resultat: Sieg

simulation 2
Resultat: Niederlage

Gambler's Ruin

In Groovy:

```
// input arguments from command line
int stake  = Integer.parseInt(args[0]);
int goal   = Integer.parseInt(args[1]);
int trials = Integer.parseInt(args[2]);

def wins = 0;
for (i in 1..trials) {
    def t_money = stake;
    while (t_money > 0 && t_money < goal) {
        if (Math.random() < 0.5)
            t_money++;
        else
            t_money--;
    }
    if (t_money == goal) wins++;
}
println(wins + " wins of " + trials);
```

Simulation and Analysis

```
stake  goal  trials
% groovy gambler 5 25 1000
191 wins of 1000

% groovy gambler 5 25 1000
203 wins of 1000

% groovy gambler 500 2500 1000
197 wins of 1000
```

Nach mehreren Stunden Berechnung…

Tatsache: Wahrscheinlichkeit für Sieg = stake / goal.

Tatsache: Erwartete Anzahl der Einsätze = stake * desired gain.

Beispiel: 20% Chance um $500 in $2500 zu verwandeln;
es können jedoch bis zu 1 Million ($1 bets) Einsätze nötig sein.

Hinweis: Beide Tatsachen können mathematisch bewiesen werden.
Für komplexere Szenarien sind jedoch Computersimulationen geeigneter.

Russische Bauernmultiplikation

- Die Multiplikation von natürlichen Zahlen (m, n > 0) kann man wie folgt auf das Addieren, Subtrahieren, Halbieren und Verdoppeln zurückführen (Russische Bauernmultiplikation):

$$m \cdot n = \begin{cases} m & \text{falls n=1} \\ m \cdot (n-1) + m & \text{falls n>1, ungerade} \\ 2m \cdot \frac{n}{2} & \text{falls n gerade} \end{cases}$$

1. Berechnen Sie 257 * 37 und geben Sie alle Zwischenergebnisse an!
2. Schreiben Sie eine Funktion ohne Verwendung des Multiplikationsoperators, das die obige Rechenregel benutzt.

Russische Bauernmultiplikation

- Bsp.

257 * 37
= 257 + 257*36
= 257 + 514*18
= 257 + 1028*9
= 257 + 1028 + 1028*8
= 257 + 1028 + 2056*4
= 257 + 1028 + 2056*4
= 257 + 1028 + 4112*2
= 257 + 1028 + 8224
= 9509

Restprodukt	Summand
257*37	
257*36	257
514*18	
1028*9	
1028*8	1028
2056*4	
4112*2	
8224*1	
	8224
Summe	9509

$$m \cdot n = \begin{cases} m & \text{falls n=1} \\ m \cdot (n-1) + m & \text{falls n>1, ungerade} \\ 2m \cdot \frac{n}{2} & \text{falls n gerade} \end{cases}$$

Arrays (Felder)

- Aufgabe:
  - Berechne alle Primzahlen kleiner N.
- Idee:
  - Sieb des Eratosthenes:
    - Verwende ein boolesches Array, welches angibt, ob eine Zahl i eine Primzahl ist (a[i] = true) oder nicht (a[i] = false).
    - Zunächst sind alle Zahlen potentielle Primzahlen (a[i] = true, für 1<i<=N).
    - Ist eine Zahl i ein Vielfaches einer bekannten Primzahl, so wird a[i] = false gesetzt.
    - Wenn a[i] immer noch true ist, nachdem alle Vielfachen von kleineren Primzahlen auf false gesetzt wurden, dann wissen wir, dass es sich um eine Primzahl handelt.

Arrays (Felder)

In Groovy:

```
def N = 100;
def a = new boolean[N];

for (i in 2..N-1)
  a[i] = true; // alle Zahlen sind pot. Kandidaten

for (i in 2..N-1) {
  if (a[i] != false) { // Primzahl gefunden
    for (j = i; j*i < N; j++) {
      a[i*j] = false; // Vielfaches einer Primzahl
   }
 }
}
for (i in 2..N-1) { // Primzahlen ausgeben
 if (a[i]) {
    print(" " + i);
 }
}
```

i	2	3	a[i]
2	1		1
3	1		1
4	1	0	
5	1		1
6	1	0	
7	1		1
8	1	0	
9	1	0	
10	1	0	
11	1		1
12	1	0	0
13	1		1
14	1	0	

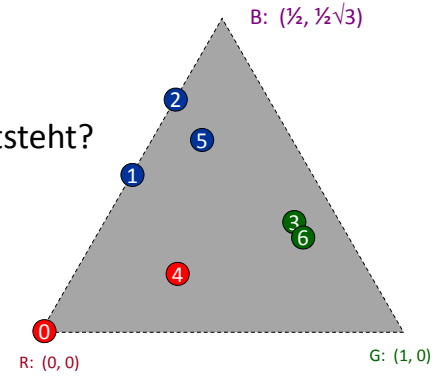
Chaos Game

Chaos game: Gleichseitiges Dreieck mit den Ecken R, G, B.

- Starte bei R.
- Wiederhole das folgende N mal:
  - Wähle eine zufällige Ecke
  - Bewege dich in diese Richtung; stoppe auf halbem Wege
  - Zeichne einen Knoten

B B G R B G …

Frage: Welches Bild entsteht?



Chaos Game

In Java:

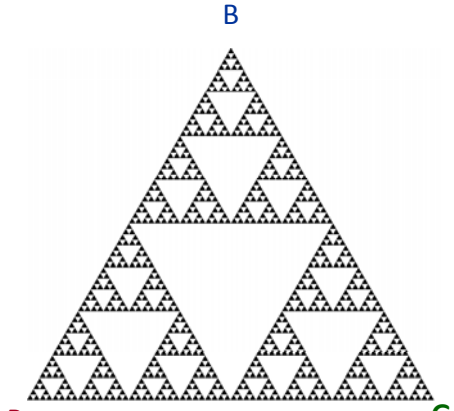
```
public class Chaos {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      double[] cx = { 0.000, 1.000, 0.500 };
      double[] cy = { 0.000, 0.000, 0.866 };

      double x = 0.0, y = 0.0;
      for (int i = 0; i < N; i++) {
         int r = (int) (Math.random() * 3);
         x = (x + cx[r]) / 2.0;
         y = (y + cy[r]) / 2.0;
         StdDraw.point(x, y);
      }
   }
}
```

½√3

- cx[i], cy[i] sind die Koordinaten der 3 Eckpunkte

Chaos Game

Resultat:



Sierpinski-Dreieck

Backup

Stars

- Schreiben Sie eine Funktion, welche die natürlichen Zahlen von 1 bis 99 wie folgt ausgibt:
  - Zahlen, die entweder durch 7 teilbar sind oder deren Quersumme gleich 7 ist, werden durch einen Stern (*) ersetzt, alle anderen werden einfach ausgegeben.

```
int s,j;

for (def i = 1; i <= 99; i++) {
   // Fuer alle Zahlen im Intervall [1,99]
   s = 0; j = i;
   while (j > 0) { //solange j noch Ziffern >0 enthaelt
      s = s + (j % 10); //letzte Stelle von j zu Quersumme s addieren
      j = j / 10; //letzte Stelle von j abschneiden
   }
   if ( (s == 7) || /*Wenn Quersumme=7 oder*/
     (i % 7 == 0) ) {/*Zahl durch 7 teilbar,*/
      print(" *"); /*dann "*" ausgeben,*/
   } else {
      print(i); /*sonst die Zahl selbst*/
   }
   println();
}
```