Praktikum V – Zufallszahlen

GTI SoSe 2016 Prof. A. Siebert

Aufgabe 1. Zufallszahlengeneratoren.
Implementieren Sie in Java die folgenden Zufallszahlengeneratoren, jeweils auf dem Datentyp `int`, d.h. auf vorzeichenbehafteten 32-bit Zahlen.

a. Implementieren Sie den Linearen Kongruenzgenerator

$$z_{i+1} = a \cdot z_i \mod m$$

Vermeiden Sie Überläufe. Negative Zahlen dürfen ebenfalls nicht auftreten.
Wie im Skript beschrieben muss sich für die Parameter a=16807, m=$2^{31}-1$ und z_0=1 die Folge `16807, 282475249, 1622650073, 984943658, ...` ergeben.

b. Implementieren Sie den Verzögerten Generator

$$z_i = z_{i-24} + z_{i-55} \mod m$$

mit m=2^{30}. Die Modulo-Operation lässt sich mit dem Operator `%` oder mit der Bitoperation `x & 0x3fffffff` durchführen.
Halten Sie die jeweils benötigten Werte in einem Feld (*Array*) der Länge 55 vor.
Wenn Sie die ersten 55 Werte mit dem LKG aus (a) initialisieren (also z_1=`16807`, z_2=`282475249, ...` z_{55}=`771515668`), so sollten die Zufallszahlen ab z_{56} die Werte `563082809, 846088761, 650837516, ...` haben.

c. Implementieren Sie den XORShift Generator mit den Verschiebeparametern (21, 35, 4).
Generieren Sie zunächst die Zufallszahlen auf dem Datentyp `long` und verwenden Sie später nur die letzten 31 bits.
Wenn Sie die mit dem Startwert $z_1 = 1$ beginnen, dann sollten der Generator auf `long` die Werte `35651601, 1130297953386881, -9204155794254196429, ...` durchlaufen.
Hieraus leiten sich die 31-bit Zufallszahlen `35651601, 33153, 572526899, ...` ab.

Messen Sie bei allen drei Generatoren, wie lange es dauert, jeweils n=10 000 000 (= 10 Millionen) Zufallszahlen zu generieren.
[Meine Messwerte: LKG ≈ 100 ms, Verzög. Generator ≈ 30 ms, XORShift ≈ 25 ms.]

Code zur Zeitmessung in Java:

```
long startTime, diffTime;
startTime = System.nanoTime();
   *** your code here ***
diffTime  = (System.nanoTime() - startTime)/1000000; // in milliseconds
```

Aufgabe 2. Verteilung der Lauflängen.
Bestimmen Sie für die oben generierten Zufallszahlen die Verteilung der Lauflängen, wobei Sie (wie im Skript)

- alle Lauflängen r≥7 zusammenfassen
- die erste Zahl eines Laufes überspringen (es sei denn, es handelt sich um den allerersten Lauf).

Zum Testen: Für obige Generatoren und Startwerte (beim verzögerten Generator verwende ich die Zufallszahlen ab z_{56}) ergeben sich die Häufigkeiten

r	LKG	Verzögert	XORShift
1	1840369	1839430	1839821
2	1226224	1226798	1226930
3	460197	460062	459579
4	121997	122683	122630
5	25543	25180	25333
6	4380	4288	4316
≥7	724	735	725

Aufgabe 3. χ^2-Test.
Sind die Zufallszahlen, die wir oben erzeugt haben, einigermaßen brauchbar?
Das ist alles andere als sicher, da wir Generatoren, die für *unsigned* Datentypen entworfen wurden, "einfach so" auf einem *signed* Datentyp verwenden.

a. Wenden Sie den χ^2-Test an auf das Histogramm der obigen Zufallszahlen, d.h. teilen Sie den Wertebereich der Zufallszahlen einmal in 10 und einmal in 100 Intervalle ein, zählen Sie die Anzahl der Zufallszahlen in jedem Intervall, berechnen Sie hierfür die jeweilige χ^2-Kennzahl und vergleichen Sie diese mit den zugehörigen Quantilen für 0.01, 0.05, 0.95, 0.99.
Ihr Programm soll entsprechend eine Meldung *Die Zufallszahlen sind zurück zu weisen / suspekt / anscheinend ok* ausgeben.

b. Wenden Sie den χ^2-Test auf die Lauflängen aus Aufgabe 2 an, analog zu (a).
Beim Vergleich von n_r mit $N_r \cdot p_r$ darf letzteres nicht gerundet werden.

Sie sollten also folgende Tabelle füllen:

χ^2	LKG	Verzögert	XORShift
histo, #=10			
histo, #=100			
Lauflängen			

Wenn Sie obige Tests mit einem schlechten Zufallszahlengenerator bzw. mit unpassenden Parametern wiederholen, dann sollte mindestens ein Test dies bemerken!