Random number generators (RNG)

RNGs needed for:
- Monte Carlo simulations
- decision making / lottery
- graphics
- tests of analysis tools

Note: random ≠ „free of detectable patterns"

examples for patterns:
- pairs, triplets... in lottery 6/49
- $\pi = 3.14159265358979323846264338327950\ldots$

„conveys entire history of human race" (Knuth, Vol 2)

Def: A RNG (or pseudo-RNG) is an algorithm or device which produces a series/stream of numbers with a specified (e.g. uniform) distribution, in which subsequent values appear independent.

Different classes:
- physical devices using thermal fluctations or atomic decay
- mathematical algorithms

Essential requirements for pseudo-RNGs:
- correct distribution (uniform)
- long periods (much longer than runtime)

Usual approach on computer: work with positive integers, convert to real by $x_n = i_n / i_{max}$

Note: maximal period ensures uniform distribution

1) Early example: mid-square method (von Neumann, 1946)

Example: 10 digits

$i_1 = 5772156649 \rightarrow i_1^2 = 33317792380594909201$

i_2

Very simple, but not useful: fixed points, short cycles

2) Complicated: Knuth's super-random generator K

See, e.g.: http://www.perlmonks.org/index.pl?node_id=452508

3) Common choice today: linear congruental generator (lcg)

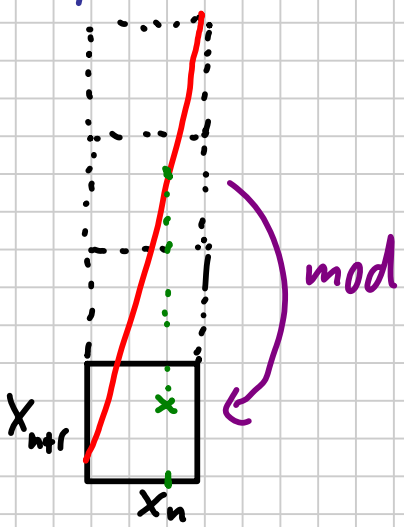
Lehmer, 1949

$$X_{n+1} = (a X_n + c) \bmod m \qquad 0 \le a < m \qquad 0 \le c < m$$

multiplier, increment, modulus

+ starting value (seed) X_0

Special case: $c = 0 \rightarrow$ multiplicative lcg – mlcg



Example: lcgs $X_{n+1} = (aX_n + c) \bmod m$ for $m = 4$

trivial cases: $a=0$: $X_n \to c$ $\quad$ $a=1$: cyclic shift

$a=2$: $c=0$

$0 \to 0$
$1 \to 2$
$2 \to 0$
$3 \to 2$

$1 \searrow$
$3 \nearrow$ $2 \to 0 \circlearrowright$

allg:

$1-c \searrow$
$3-c \nearrow$ $2-c \to -c \circlearrowright$ (mod 4)

$a=3$: $c=0$

$0 \circlearrowright$
$1 \rightleftarrows 3$
$2 \circlearrowright$

$c=1$

$0 \rightleftarrows 1$
$2 \rightleftarrows 3$

$c=2$

$0 \rightleftarrows 2$
$1 \circlearrowright$
$3 \circlearrowright$

$c=3$

$0 \rightleftarrows 3$
$1 \rightleftarrows 2$

Conclusions:
- lcg's can have fixed points and short cycles
- the least significant digits/bits are not very random (for $m=2^n$ ⇝ above example)

The lcg has maximum period m if (and only if)

i) c is relative prime to m
ii) $b = a-1$ is a multiple of p for every prime factor p of m
iii) b is a multiple of 4 if m is multiple of 4.

Widely used lcg implementations:

DRAND48: lcg (2^{48}, 25214903917, 11)
(m, a, c)

rand() in ANSI C: lcg(2^{31}, 1103515245, 12345)

See also: http://crypto.mat.sbg.ac.at/results/karl/server/node4.html

General problems with lcgs:
- n-tupels on few hyperplanes
- maximum period: m

Longer periods require longer „memory" of RNG:

4) Lagged Fibonacci generator

e.g. additive: $X_n = (X_{n-55} + X_{n-24}) \bmod m$

Mitchell + Moore (1958)

with m even, $X_0, \ldots, X_{55}$ not all even

⇝ period $2^{55}-1$ for least significant bit

Many possible test for RNGs, but ultimate test: own application – check using different RNGs!

I use: SPRNG – scalable parallel RNG library.