Random number generators (RNGs)

- Old-school: dice, coinflips, ...
- Hardware RNGs: Generates numbers based on some unpredictable feature of physical environment (e.g. thermal noise)
- Pseudo RNGs: Deterministic algorithms that generate numbers that are predetermined but appear random (unpredictable).

  What we will focus on!

  Initialised by the starting number (seed)

  → Reproducible!

- Desired properties for a pseudo RNG:

  1) Produce numbers that are distributed uniformly on $[0,1]$, i.e. samples from $U(0,1)$
  2) Negligible correlations between numbers (Knowing a previous number shouldn't help you guess the next number — unless you know the algorithm of course...)
  3) The period before repetition should be as long as possible
  4) Computationally fast algorithm

⊗ End Nov 11

- Classic algorithm : Linear Congruential Generator (LCG)

(~ 1950s)

$$N_{i+1} = (a N_i + c) \bmod (m)$$

1) scale and shift current number

2) Use modulus operator

a : multiplier $\quad 0 < a < m$

c : increment $\quad 0 \leq c < m$

m : the modulus $\quad 0 < m$

N_0 : the seed $\quad 0 \leq N_0 < m$

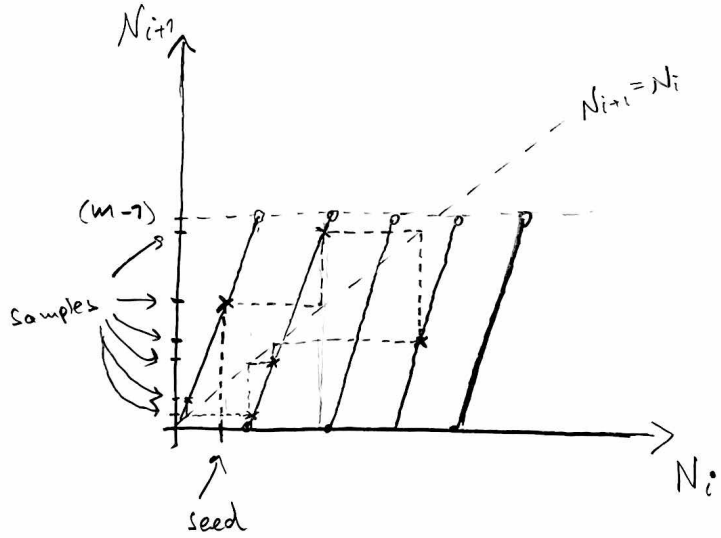
modulus operator :
"what's the remainder ?"

$13 \bmod (2) = 1$

$17 \bmod (5) = 2$

$8 \bmod (8) = 0$

$16 \bmod (17) = 16$



Other example :
12/24 hour clock

23:00 + 2hours → 01:00

$(23 + 2) \bmod (24) = 1$

- To get numbers on $[0, 1)$: $X_i = \frac{N_i}{m}$

- How good the generator is depends critically on the choice of parameters : a, c, m (and potentially N_0)
  - There's a lot of research on such parameter choices for LCGs and other RNG algorithms !

- Examples :

$m = 9$, $a = 2$, $c = 0$, $N_0 = 1$ } $1, 2, 4, 8, 7, 5, (1)$ → repeats

Period : 6

$m = 9$, $a = 2$, $c = 0$, $N_0 = 3$ } $3, 6, (3)$

Period : 2 !

$m = 9$, $a = 4$, $c = 1$, $N_0 = 0$ } $0, 1, 5, 3, 4, 8, 6, 7, 2, (0)$

Period : 9

- More realistic case :

$m = 2^{32} = 4\,294\,967\,296$ (more than 4.2×10^9)

$a = 1664525$

$c = 1013904223$ → [Rand1, Numerical Recipes]

- Always look up period of a RNG !

  Famous RNGs have had surprisingly small periods

  ⇒ cannot trust results !

- RANDU (IBM, 1960s), famous worst-case example (samples in 3D would fall on distinct 2D planes ...)

- Period is not only concern!

  What happens for $a=1$, $c=1$, m = large number ?

  Answer: get a "modulus counter": $N_0, N_0+1, N_0+2, N_0+3, \ldots$

  - Long period, but does it look random? No!

- There are collection of statistical randomness tests used to test RNGs. (They all fail some ...)

[Diehard tests, Test U01]

[Donald Knuth (TeX inventor) was the first to propose a set of such tests...]

- Other RNG examples:

  - "Shift-register"

$$N_{i+1} = (aN_{i-j} + cN_{i-k}) \bmod (m)$$

Uses more than just the preceeding number!

  - A standard choice today: Mersenne Twister (MT19937)

    - Developed in 1997 [Matsumoto, Nishimura]
    - Available in <random> in C++11
    - Period of $2^{19937}-1$

- Pitfall when using RNGs with parallelization:

  Make sure that RNGs on different threads get different seeds!!

  (Don't want different threads generating the exact same numbers)

- Can use thread number to modify a "base seed" such that all threds have a unique seed.

  ⇒ Result is still reproducible, if you use the same number of threds

- [Show (random) examples]