RANDOM NUMBER GENERATORS RNGS

More appropriately called

Pseudo-Random Number Generators

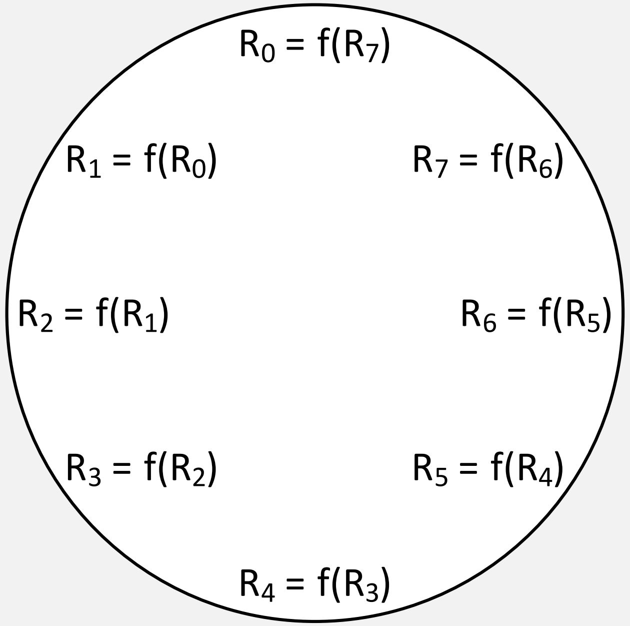
Delroy A. Brinkerhoff

PSEUDO-RANDOM NUMBER GENERATORS

- *Correct* computer programs are deterministic
- Given the same input, they produce the same output

PSEUDO-RANDOM NUMBER GENERATORS

- *Correct* computer programs are deterministic
- Given the same input, they produce the same output
- Software RNGS produce a long, repeating cycle of numbers
- The numbers “look” random (they pass some statistical tests of randomness)



$R_0 = f(R_7)$

$R_1 = f(R_0)$

$R_7 = f(R_6)$

$R_2 = f(R_1)$

$R_6 = f(R_5)$

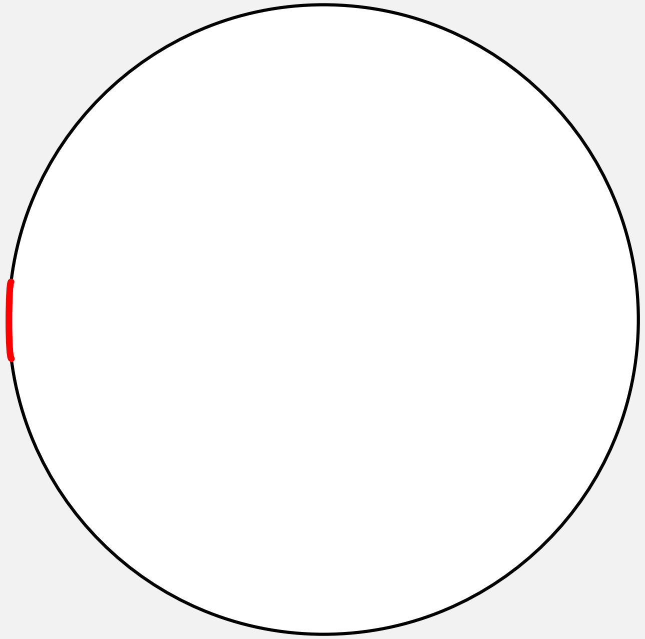
$R_3 = f(R_2)$

$R_5 = f(R_4)$

$R_4 = f(R_3)$

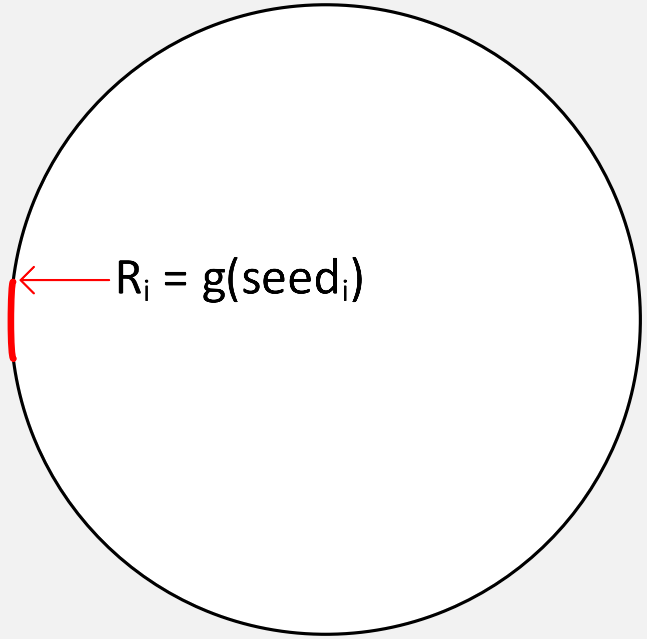
USING PSEUDO-RANDOM SEQUENCES

- RNGS have *very* long cycles (i.e., a long sequence before repeating)
- Programs typically use a small part of the cycle (i.e., a short sub-sequence)



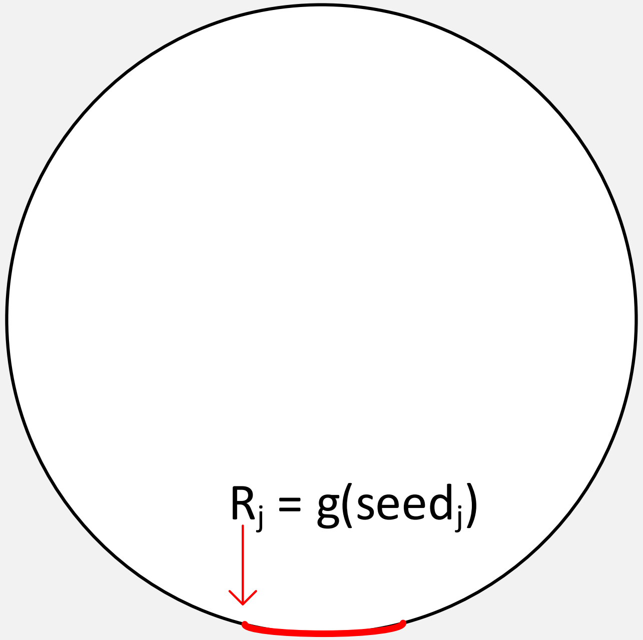
USING PSEUDO-RANDOM SEQUENCES

- RNGS have *very* long cycles (i.e., a long sequence before repeating)
- Programs typically use a small part of the cycle (i.e., a short sub-sequence)
- Programs start a sub-sequence with a "seed" value
- The same seed always produces the same sub-sequence



USING PSEUDO-RANDOM SEQUENCES

- RNGS have *very* long cycles (i.e., a long sequence before repeating)
- Programs typically use a small part of the cycle (i.e., a short sub-sequence)
- Programs start a sub-sequence with a "seed" value
- The same seed always produces the same sub-sequence
- A different seed produces a different sub-sequence

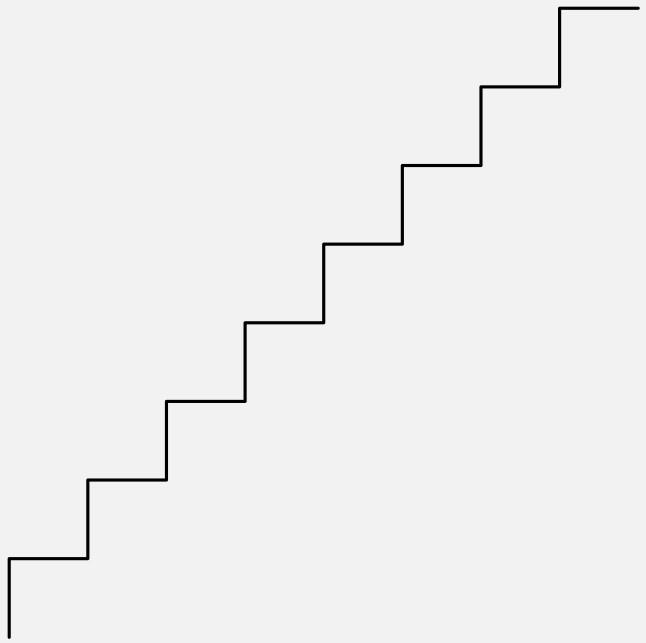


SEEDING A RNG

- Programmers want a different random sequence each time the program runs
- They need a source of unique seeds

SEEDING A RNG

- Programmers want a different random sequence each time the program runs
- They need a source of unique seeds
- The computer clock maintains the time since the epoch
  - Jan 1, 1970 (Unix, Linux, macOS)
  - Jan 1, 1980 (Windows)
- Time is a monotonically increasing value



INHERITED C
RANDOM NUMBER GENERATOR

```
#include <stdlib.h>
#include <time.h>

srand((unsigned)time(nullptr));

for (int i = 0; i < 10; i++)
        numbers[i] = rand() % 100;
```

C++ RNGS AND DISTRIBUTIONS

```
#include <random>
#include <chrono>

default_random_engine
    rng((unsigned)(chrono::system_clock::now().time_since_epoch().count()));

for (int i = 0; i < 10; i++)
    numbers[i] = rng();

uniform_int_distribution<int>    range(1, 100);

for (int i = 0; i < 10; i++)
    numbers[i] = range(rng);
```