



UNIT 9A

Randomness in Computation: Random Number Generators

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

1

Randomness

- Some computations are based on randomness.
 - games, encryption, simulations
- A sequence is *random* if, for any value in the sequence, the next value in the sequence is totally independent of the current value.

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

2

Random numbers in Ruby

- To generate random numbers in Ruby, we can use the **rand** function.
- The **rand** function take a positive integer argument (n) and returns an integer between 0 and n-1.

```
>> rand(15110)
=> 1239
>> rand(15110)
=> 7320
>> rand(15110)
=> 84
```

Is **rand** truly random?

- The function **rand** uses some algorithm to determine the next integer to return.
- If we knew what the algorithm was, then the numbers generated would not be truly random.
- We call **rand** a pseudo-random number generator (PRNG) since it generates numbers that appear random but are not truly random.

Creating a PRNG

- Consider a pseudo-random number generator **prng1** that takes an argument specifying the length of a random number sequence and returns an array with that many “random” numbers.

```
>> prng1(9)
```

```
=> [0, 7, 2, 9, 4, 11, 6, 1, 8]
```

- Does this sequence look random to you?

Creating a PRNG

- Let's run **prng1** again:

```
>> prng1(15)
```

```
=> [0, 7, 2, 9, 4, 11, 6, 1, 8, 3,  
    10, 5, 0, 7, 2]
```

- Now does this sequence look random to you?
- What do you think the 16th number in the sequence is?

Another PRNG

- Let's try another PRNG function:

```
=> prng2(15)
```

```
>> [0, 8, 4, 0, 8, 4, 0, 8, 4, 0,  
     8, 4, 0, 8, 4]
```

- Does this sequence appear random to you?
- What do you think is the 16th number in this sequence?

PRNG Period

- Let's define the PRNG period as the number of values in a pseudo-random number generator sequence before the sequence repeats.

```
[0, 7, 2, 9, 4, 11, 6, 1, 8, 3,  
 10, 5, 0, 7, 2]
```

period = 12

```
[0, 8, 4, 0, 8, 4, 0, 8, 4, 0,  
 8, 4, 0, 8, 4]
```

period = 3

Looking at `prng1`

```
def prng1(n)
  seq = [0]           ; seed (starting value)
  for i in 1..n-1 do
    seq << (seq.last + 7) % 12
  end
  return seq
end

>> prng1(15)
=> [0, 7, 2, 9, 4, 11, 6, 1, 8, 3,
    10, 5, 0, 7, 2]
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

9

Looking at `prng2`

```
def prng2(n)
  seq = [0]           ; seed (starting value)
  for i in 1..n-1 do
    seq << (seq.last + 8) % 12
  end
  return seq
end

>> prng2(15)
=> [0, 8, 4, 0, 8, 4, 0, 8, 4, 0,
    8, 4, 0, 8, 4]
```

15110 Principles of Computing,
Carnegie Mellon University - CORTINA

10

Linear Congruential Generator (LCG)

- A more general version of the PRNG used in these examples is called a linear congruential generator.
- Given the current value x_i of PRNG using the linear congruential generator method, we can compute the next value in the sequence, x_{i+1} , using the formula $x_{i+1} = (a \times x_i + c) \text{ modulo } m$ where a , c , and m are pre-determined constants.
 - **prng1:** $a = 1, c = 7, m = 12$
 - **prng2:** $a = 1, c = 8, m = 12$

Picking the constants a , c , m

- If we choose a large value for m , and appropriate values for a and c that work with this m , then we can generate a very long sequence before numbers begin to repeat.
 - Ideally, we could generate a sequence with a maximum period of m .

Picking the constants a, c, m

- The LCG will have a period of m for all seed values if and only if:
 - c and m are *relatively prime* (i.e. the only positive integer that divides both c and m is 1)
 - a-1 is divisible by all prime factors of m
 - if m is a multiple of 4, then a-1 is also a multiple of 4
- Example: prng1 (a = 1, c = 7, m = 12)
 - Factors of c: 1, 7 Factors of m: 1, 2, 3, 4, 6, 12
 - 0 is divisible by all prime factors of 12 → true
 - if 12 is a multiple of 4, then 0 is also a multiple of 4 → true

Example

$$x_{i+1} = (a \times x_i + c) \text{ modulo } m$$

$$x_0 = 4 \quad a = 5 \quad c = 3 \quad m = 8$$

- Compute x_1, x_2, \dots , for this LCG formula.
- What is the period of this formula?
 - If the period is maximum, does it satisfy the three properties for maximal LCM?

LCMs in the Real World

- glibc (used by the c compiler gcc):
a = 1103515245, c = 12345, m = 2^{32}
- *Numerical Recipes* (popular book on numerical methods and analysis):
a = 1664525, c = 1013904223, m = 2^{32}
- Random class in Java:
a = 25214903917, c = 11, m = 2^{48}
- The PRNG built into Ruby has a period of 2^{19937} .

Using RubyLabs for Random Numbers

```
>> include RandomLab
=> Object
>> p = PRNG.new(1, 7, 12)
=> #<RandomLab::PRNG a: 1 c: 7 m: 12>
>> p.seed(0)
=> 0
>> p.advance
=> 7
>> p.advance
=> 2
>> p.state
=> 2
```