Random Number Generators

- Historically there are three types of generators
  - table look-up generators
  - hardware generators
  - algorithmic (software) generators
- Algorithmic generators are widely accepted because they meet all of the following criteria:
  - randomness - output passes all reasonable statistical tests of randomness
  - controllability - able to reproduce output, if desired
  - portability - able to produce the same output on a wide variety of computer systems
  - efficiency - fast, minimal computer resource requirements
  - documentation - theoretically analyzed and extensively tested

Algorithmic Generators

- An *ideal* random number generator produces output such that *each* value in the interval $0.0 < u < 1.0$ is *equally likely* to occur
- A *good* random number generator produces output that is (almost) statistically indistinguishable from an ideal generator
- We will construct a good random number generator satisfying all our criteria

Conceptual Model

- Conceptual Model:
  - Choose a *large* positive integer m. This defines the set $\mathcal{X}_m = \{1, 2, \ldots, m-1\}$
  - Fill a (conceptual) urn with the elements of $\mathcal{X}_m$
  - Each time a random number u is needed, draw an integer x "at random" from the urn and let $u = x/m$
- Each draw *simulates* a sample of an independent identically distributed sequence of $\mathit{Uniform}(0, 1)$
- The possible values are $1/m, 2/m, \ldots (m-1)/m$.
- It is important that m be large so that the possible values are densely distributed between 0.0 and 1.0

Conceptual Model

- 0.0 and 1.0 are impossible
  - This is important for some random variates
- We would like to draw from the urn with replacement
- For practical reasons, we will draw without replacement
  - If m is large and the number of draws is small relative to m, then the distinction is largely irrelevant

Lehmer's Algorithm

- *Lehmer's algorithm* for random number generation is defined in terms of two fixed parameters:
  - *modulus* m, a fixed large *prime* integer
  - *multiplier* a, a fixed integer in $\mathcal{X}_m$
- The integer sequence $x_0, x_1, \ldots$ is defined by the iterative equation

  $$x_{i+1} = g(x_i)$$

  with

  $$g(x) = ax \bmod m$$

- $x_0 \in \mathcal{X}_m$ is called the *initial seed*

Let $m = 13$. Calculate the sequence

$$x_{i+1} = ax_i \bmod m$$

in your learning group.

Let $m = 13$. Calculate the sequence

$$x_{i+1} = ax_i \bmod m$$

in your learning group.

$$\begin{aligned}
\text{Outliers}: &\quad a = 5 \text{ and } x_0 = 1\\
\text{AskAI}: &\quad a = 7 \text{ and } x_0 = 1\\
\text{Crosswalk Crusaders}: &\quad a = 5 \text{ and } x_0 = 4\\
\text{Jonah's Group}: &\quad a = 6 \text{ and } x_0 = 1\\
: &\quad a = 5 \text{ and } x_0 = 2\\
: &\quad a = 9 \text{ and } x_0 = 3
\end{aligned}$$

Let $m = 13$. Calculate the sequence

$$x_{i+1} = ax_i \bmod m$$

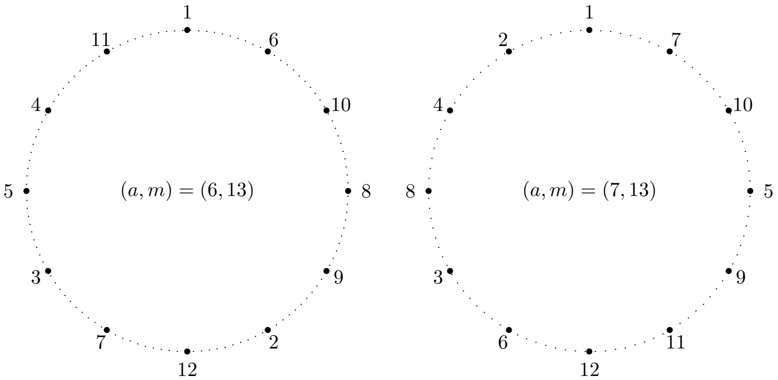
in your learning group.

$$\begin{aligned}
\text{Outliers}: &\quad a = 5 \text{ and } x_0 = 1 \\
\text{AskAI}: &\quad a = 7 \text{ and } x_0 = 1 \\
\text{Crosswalk Crusaders}: &\quad a = 5 \text{ and } x_0 = 4 \\
\text{Jonah's Group}: &\quad a = 6 \text{ and } x_0 = 1 \\
: &\quad a = 5 \text{ and } x_0 = 2 \\
: &\quad a = 9 \text{ and } x_0 = 3
\end{aligned}$$

Congrats, you are all pRNGs. What you see are the cyclic periods of various as, finding an a with big cycles (large periods, ρ) is important for good pRNGs.

Example 2.1.2

- Full-period multipliers generate a virtual *circular list* with $m - 1$ distinct elements.



Parameter Considerations

- The choice of m is dictated, in part, by system considerations
  - On a system with 32-bit 2's complement integer arithmetic, $2^{31} - 1$ is a natural choice
  - With 16-bit or 64-bit integer representation, the choice is not obvious
  - In general, we want to choose m to be the largest representable prime integer
- Given m, the choice of a must be made with great care

Central Issues

- For a chosen (a, m) pair, does the function $g(\cdot)$ generate a full-period sequence?
- If a full period sequence is generated, how random does the sequence appear to be?
- Can $ax \bmod m$ be evaluated efficiently and correctly?
  - Integer overflow can occur when computing ax

Sshhh! It's a Secret Expression

We can think of the Lehmer generator equation as a simplification ($n = 1$) of the following form

$$y = l(a, x_i, n, m) = a x_i^n \bmod m \big|_{n=1}$$

This equation may look familiar to you. If $a = 1$ it's the fundamental equation that several different PKI (public key cryptography) algorithms use. Namely, our inability to devise an inverse (called the *discrete log*)

$$x_i = l^*(y, a, n, m)$$

is what gives (non-elliptic curve) public key algorithms (part of) their strength (the other is very large numbers).

Crypto systems don't use this equation for their pRNG — crypto systems need random *bit values*. Lehmer generatators are not good enough for that. In general crypto systems DON'T use pRNGs designed for simulation.

Given a Lehmer based pRNG:

- m a prime number
- a the multiplier in the generating equation

$$x_{i+1} = ax_i \bmod m$$

How do m and a affect the features of the pRNG?

When a programmer chooses a "seed", what is actually being chosen?

Given a Lehmer based pRNG:

- m a prime number
  m dictates **two things**: the maximum number of values in the set ("urn") χ_m, and (based on the prime factorization of $m - 1$) *how many* full period modifiers (the a's) exist for the values in χ_m.
- a the multiplier in the generating equation

$$x_{i+1} = ax_i \bmod m$$

  a dictates **two things**: the *specific* size of the random sequence (its *period*) and the actual sequence of the values x_i. We want the size to be big, like $m - 1$, but not all multipliers will do this.

How do m and a affect the features of the pRNG?

When a programmer chooses a "seed", what is actually being chosen? The programmer is choosing x_0, that specifies the starting point in the sequence of random values dictated by a.

Randomness

- Choose the FPMC multiplier that gives "most random" sequence
- No universal definition of randomness
- In 2-space, $(x_0, x_1), (x_1, x_2), (x_2, x_3), \ldots$ form a *lattice* structure
- For any integer $k \geq 2$, the points

  $$(x_0, x_1, \ldots, x_{k-1}), (x_1, x_2, \ldots, x_k), (x_2, x_3, \ldots, x_{k+1}), \ldots$$

  form a lattice structure in k-space
- Numerically analyze *uniformity* of the lattice

  E.g., Knuth's spectral test

Random Numbers Falling In The Planes