Statistics 243: *class notes*

William J. De Meo

10/1/97

1 Uniform Random Number Generator

1. Linear Congruential U(0,1) Generator This generator is based on four numbers

- m = modulus
- c = increment
- a = multiplier
- X_0 = seed

The formula for this generator is

$$X_{n+1} = \mathrm{mod}(aX_n + c, m)$$

For X_0 system clock value is one possible source of a starting value. We can set X_0 to the clock value with the line:

```
seed = time();
```

If c = 0, the method is called a *multiplicative congruential generator*.

By their very nature, these generators have a cycle, or period which is the number of unique vlues which are produced before it starts repeating. We want a period equal to the number of unique values in the computer. So we should pick the largest unsigned integer for m, or 2^p where p is the number of bits on the computer. But this is the same as just using the formula

$$X_{n+1} = aX_n + c$$

It can be shown that a mixed congruential generator will have period m iff

1. c is relatively prime to m
2. mod(a,p) = 1 for all prime factors p of m
3. mod(a,4) = 1 if 4 is a factor of m

1.1 Composite Generators

$$X_{n+1} = \mathrm{mod}(a_1X_n + c_1, m)$$

$$Y_{n+1} = \mathrm{mod}(a_2Y_n + c_2, m)$$

$$W_n = \mathrm{mod}(X_n + Y_n, m)$$

1.2 Quadratic Congruential Generator

$$X_{n+1} = \mathrm{mod}(aX_n^2 + aX_n + c, m)$$

1.3 Additive Generators

see `man 3 random`

$$X_n \mathrm{mod}(X_{n-r_1} + X_{n-r_2}, m), n \geq max(r_1, r_2)$$

To start, the first $max(r_1, r_2)$ numbers are chosen arbitrarilly $r_1 = 24, r_2 = 55$ possibly good starting values.

1.4 Feedback shift Register Techniques (Tausworthe generators)

Linear recurrence relation among the bits of the random number.

$$a_k = mod((c_p a_{k-p} + c_{p-1} a_{k-p+1} + \cdots + c_1 a_{k-1}, 2)$$

The $\{c_i\}$ are fixed and equal to 0 or 1.

1.5 Shuffling

We can make any random number generator more random by using *shuffling*. The procedure is as follows:

1. Initialization
   - Generaute an array of, say, 100 random numbers
   - You should have it automatically initialized
2. Generate another random number y to start the process.
3. Each time you want a random number, use y to find an index into v: `index = (int)( 100 * y((double)m)` where m is the modulus.
4. Set y = v[index]
5. Replace v[index] with a new random number
6. Return y.