

13 Random Numbers

The great metaphysical truth in the generation of random numbers is this: If you want a function that is reasonably random in behavior, then take any more or less random function modulo a prime number.

13.1 Arithmetic Modulo Prime Numbers

Details of specifics can be found in several sources, including Knuth, but here is the underlying mathematics.

Let p be a prime number. There are p (*linear*) *residues* modulo p , namely the integers 0 through $p-1$. Multiplicatively, the $p-1$ nonzero residues form a cyclic *group*, generated by any *primitive root*.

We define the *Euler totient* or more commonly the *Euler ϕ function* $\phi(n)$ as follows:

1. If p is prime, then $\phi(p) = p - 1$
2. If p is prime, then $\phi(p^k) = (p - 1)p^{k-1}$
3. If n and m are relatively prime (have no factors in common), then $\phi(nm) = \phi(n)\phi(m)$.

With this definition, $\phi(n)$ for any n is the number of residues modulo n that have no factors in common with n .

If p is a prime, then there are $\phi(p)$ primitive roots modulo p .

For example, consider the case $p = 23$. We can take powers of 2 as follows.

$$\begin{aligned}
5, & & 5^7 = 40 = 17, \\
5^2 = 25 = 2, & & 5^8 = 85 = 16, \\
5^3 = 10, & & 5^9 = 80 = 11, \\
5^4 = 50 = 4, & & 5^{10} = 55 = 9, \\
5^5 = 20, & & 5^{11} = 45 = 22. \\
5^6 = 100 = 8, & &
\end{aligned}$$

We notice at this point that $22 = -1$, so the succeeding powers are

$$\begin{aligned}
5^{12} = -5 & & 5^{18} = -17, \\
5^{13} = -2, & & 5^{19} = -16, \\
5^{14} = -10, & & 5^{20} = -11, \\
5^{15} = -4, & & 5^{21} = -9, \\
5^{16} = -20, & & 5^{22} = 1. \\
5^{17} = -8, & &
\end{aligned}$$

The multiplication table modulo 23 is now easy to read off. Instead of dealing with the residues, work with the exponents of 5. For example, to multiply 11 times 13, we just note that $11 = 5^9$ and $13 = -10 = 5^{14}$ and from that we read off that $11 \times 13 = 5^{9+14} = 5^{23} = 5^1 = 5$. Similarly, we note that the powers of 5 that are relatively prime to 22, namely 1,3,5,7,9,13,15,17,19, and 21, are all primitive roots and generate a full cycle multiplicatively modulo 23, and the powers that are not relatively prime to 22 do not generate a full cycle. The two residues 1 and $5^{11} = -1$ generate a short cycle of length 2, and the eleven even powers $5^2, 5^4, 5^6, 5^8, 5^{10}, 5^{12}, 5^{14}, 5^{16}, 5^{18}, 5^{20}$, and $5^{22} = 1$ generate a short cycle of length eleven.

Now, what this means for random numbers is the following. For the most part, if you start with a primitive root modulo p , the sequence of powers

of that primitive root are a reasonably random permutation of the nonzero linear residues 1 through $p - 1$. Given an integer y modulo a prime p , and given a primitive root r for p , determining the discrete logarithm k such that $r^k = y$ modulo p is a relatively tough problem. This is actually at the heart of the use of discrete logs in some problems in public key cryptography.

For generating pseudorandom numbers, it turns out that we don't want to take just powers of a primitive root. Such a choice would not provide sufficient independence from one pseudorandom number to the next.

Instead, we can use a *mixed linear congruential* random number generator that begins with a seed x_0 , a modulus m , and fixed values a and b taken modulo m and then computes

$$x_{n+1} = ax_n + b \pmod{m}$$

. There are rules for choosing m , a , and b . Some of these rules can be found in Knuth. We generally want to choose m to be prime and a and b about the square root of m and with other properties.

Any deterministic iterated function modulo m must be periodic, since there are only $m - 1$ residues modulo m . With the right choices of parameters, we can maximize that period. We note that $2^{31} - 1$ is prime, and this has been a convenient fact for a long time, since it allows 32-bit computers to have a convenient modulus to work with. Modulo $2^{31} - 1$, we have

$$A \times 2^{31} + B = A \times 2^{31} - A + A + B = A \times (2^{31} - 1) + A + B = A + B.$$

So if we iterate a single precision function like the mixed congruential function

above, we get a double precision result and can reduce the result modulo m by adding the left 32 bits to the right 32 bits and thus avoiding having to divide. This speeds things up a lot.

It is reasonable to ask why we don't know exactly which functions to use for random number generation. Think about this for a minute. For a single prime about 2^{31} in size, the number of possibilities for A and B are each about 2^{31} for a total number of 2^{62} possibilities. This number is about $10^{18.7}$. If you then need to iterate the function about half the full cycle in order to get the repeat, this would be about 10^{27} total iterations of the mixed congruential function. If a 1 GHz processor can execute 3×10^{17} ops in a year, this is about 3×10^9 computers running for a year each to exhaust the number of possibilities for a single prime.

13.2 Arithmetic Modulo Primitive Polynomials

Now, it turns out of course that iterating the mixed congruential function takes much longer than just one cycle on a fast processor. Sometimes we either don't want to invest that much in a random number generator, and sometimes we only need to generate one bit at a time. The way to do this is with a *linear shift register*.

Consider polynomials with coefficients taken modulo a prime. Consider polynomials with coefficients taken modulo the prime 2, for example. Some of these polynomials can be factored mod 2, and some cannot. For example, we have

$$(x^2 + x + 1)^2 = x^4 + x^2 + 1$$

so $x^4 + x^2 + 1$ is called *reducible*, while $f(x) = x^3 + x + 1$ cannot be factored mod 2, and is called *irreducible*.

Among the irreducible polynomials are the *primitive* polynomials. These are the polynomials that act like prime integers, in the following way. Consider the sequence of powers of x taken modulo $f(x)$, and note that working modulo $f(x)$ means using the relation $x^3 + x + 1 = 0$, which is the same as $x^3 = x + 1$.

$$1$$

$$x$$

$$x^2$$

$$x^3 = x + 1$$

$$x^4 = x^2 + x$$

$$x^5 = x^3 + x^2 = x^2 + x + 1$$

$$x^6 = x^3 + x^2 + x = x^2 + x + x + 1 = x^2 + 1$$

$$x^7 = x^3 + x = x + x + 1 = 1$$

Analogous to the situation with prime numbers, the powers of x form a single multiplicative cycle. If $f(x)$ is a primitive polynomial of degree d with coefficients taken modulo a prime p , then the cycle will have length $p^d - 1$. In this case, $p = 2$ and $d = 3$, and we have 7 elements in the cycle.

Note also that if we write the coefficients of the polynomials as a binary number, then we cycle through all the possible combinations in what is apparently a random order:

$$001 = 1$$

$$010 = x$$

$$100 = x^2$$

$$011 = x^3 = x + 1$$

$$110 = x^4 = x^2 + x$$

$$111 = x^5 = x^3 + x^2 = x^2 + x + 1$$

$$101 = x^6 = x^3 + x^2 + x = x^2 + x + x + 1 = x^2 + 1$$

$$001 = x^7 = x^3 + x = x + x + 1 = 1$$

Looked at differently, we note that if we take $x^3 + x + 1$ as zero, then this is the *four* bit number 1011, so we can add this number to our bit representations coefficient-wise. Multiplication by x is a left shift of the bits, and when we get a leftmost bit that is a 1, we add in zero (that is, 1011) mod 2 so as to cancel the leftmost 1.

$$0001$$

$$0010$$

$$0100$$

$$1000 = 1000 + 1011 = 0011$$

$$0110$$

$$1100 = 1100 + 1011 = 0111$$

$$0111$$

$$1110 = 1110 + 1011 = 0101$$

$$1010 = 1010 + 1011 = 0001$$

We use this as generator of random bits by tapping the leftmost bit as the random bit, thus getting the sequence 0,0,0,1,0,1,0,1,1. And then everything repeats.