Portable Random Number Generators

Park and Miller [1] have surveyed a large number of random number generators that have been used over the last 30 years or more. Along with a good theoretical review, they present an anecdotal sampling of a number of inadequate generators that have come into widespread use. The historical record is nothing if not appalling.

There is good evidence, both theoretical and empirical, that the simple multiplicative congruential algorithm

$$I_{j+1} = aI_j \pmod{m} \tag{7.1.2}$$

can be as good as any of the more general linear congruential generators that have $c \neq 0$ (equation 7.1.1) — *if* the multiplier a and modulus m are chosen exquisitely carefully. Park and Miller propose a "Minimal Standard" generator based on the choices

$$a = 7^5 = 16807 \qquad m = 2^{31} - 1 = 2147483647 \tag{7.1.3}$$

First proposed by Lewis, Goodman, and Miller in 1969, this generator has in subsequent years passed all new theoretical tests, and (perhaps more importantly) has accumulated a large amount of successful use. Park and Miller do not claim that the generator is "perfect" (we will see below that it is not), but only that it is a good minimal standard against which other generators should be judged.

It is not possible to implement equations (7.1.2) and (7.1.3) directly in a high-level language, since the product of a and $m-1$ exceeds the maximum value for a 32-bit integer. Assembly language implementation using a 64-bit product register is straightforward, but not portable from machine to machine. A trick due to Schrage [2,3] for multiplying two 32-bit integers modulo a 32-bit constant, without using any intermediates larger than 32 bits (including a sign bit) is therefore extremely interesting: It allows the Minimal Standard generator to be implemented in essentially any programming language on essentially any machine.

Schrage's algorithm is based on an *approximate factorization* of m,

$$m = aq + r, \quad \text{i.e.,} \quad q = [m/a],\ r = m \bmod a \tag{7.1.4}$$

with square brackets denoting integer part. If r is small, specifically $r < q$, and $0 < z < m - 1$, it can be shown that both $a(z \bmod q)$ and $r[z/q]$ lie in the range $0, \ldots, m-1$, and that

$$az \bmod m = \begin{cases} a(z \bmod q) - r[z/q] & \text{if it is} \geq 0, \\ a(z \bmod q) - r[z/q] + m & \text{otherwise} \end{cases} \tag{7.1.5}$$

The application of Schrage's algorithm to the constants (7.1.3) uses the values $q = 127773$ and $r = 2836$.

Here is an implementation of the Minimal Standard generator:

```
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define MASK 123459876

float ran0(long *idum)
"Minimal" random number generator of Park and Miller. Returns a uniform random deviate
between 0.0 and 1.0. Set or reset idum to any integer value (except the unlikely value MASK)
to initialize the sequence; idum must not be altered between calls for successive deviates in
a sequence.
{
    long k;
    float ans;

    *idum ^= MASK;                      XORing with MASK allows use of zero and other
    k=(*idum)/IQ;                           simple bit patterns for idum.
    *idum=IA*(*idum-k*IQ)-IR*k;         Compute idum=(IA*idum) % IM without over-
    if (*idum < 0) *idum += IM;             flows by Schrage's method.
    ans=AM*(*idum);                     Convert idum to a floating result.
    *idum ^= MASK;                      Unmask before return.
    return ans;
}
```

The period of ran0 is $2^{31} - 2 \approx 2.1 \times 10^9$. A peculiarity of generators of the form (7.1.2) is that the value 0 must never be allowed as the initial seed — it perpetuates itself — and it never occurs for any nonzero initial seed. Experience has shown that users always manage to call random number generators with the seed idum=0. That is why ran0 performs its exclusive-or with an arbitrary constant both on entry and exit. If you are the first user in history to be proof against human error, you can remove the two lines with the $\wedge$ operation.

Park and Miller discuss two other multipliers a that can be used with the same $m = 2^{31} - 1$. These are $a = 48271$ (with $q = 44488$ and $r = 3399$) and $a = 69621$ (with $q = 30845$ and $r = 23902$). These can be substituted in the routine ran0 if desired; they may be slightly superior to Lewis *et al.*'s longer-tested values. No values other than these should be used.

The routine ran0 is a Minimal Standard, satisfactory for the majority of applications, but we do not recommend it as the final word on random number generators. Our reason is precisely the simplicity of the Minimal Standard. It is not hard to think of situations where successive random numbers might be used in a way that accidentally conflicts with the generation algorithm. For example, since successive numbers differ by a multiple of only 1.6×10^4 out of a modulus of more than 2×10^9, very small random numbers will tend to be followed by smaller than average values. One time in 10^6, for example, there will be a value $< 10^{-6}$ returned (as there should be), but this will *always* be followed by a value less than about 0.0168. One can easily think of applications involving rare events where this property would lead to wrong results.

There are other, more subtle, serial correlations present in ran0. For example, if successive points (I_i, I_{i+1}) are binned into a two-dimensional plane for $i = 1, 2, \ldots, N$, then the resulting distribution fails the χ^2 test when N is greater than a few $\times 10^7$, much less than the period $m - 2$. Since low-order serial correlations have historically been such a bugaboo, and since there is a very simple way to remove