爆破线性同余随机算法

线性同余随机数发生器 (LCG) 使用以下公式生成随机数：

$$X_{n+1} = (aX_n + c) \bmod m$$

在编程中，通常把 m 取为 2 的幂，这样 CPU 就能自动完成除模运算了。根据维基百科，当 m 是 2 的幂且 $m \geq 4$ 时，必须满足 $a \bmod 4 \equiv 1$ 且 $c \bmod 2 \equiv 1$ ，才能使随机数的循环周期为 m.

虽然未经严格证明，但当上述条件成立时，下列计算存在较快的算法：

1. 已知 n 和X_0，计算X_n；
2. 已知X_{n+1}计算X_n；
3. 已知$X_0 = 0$，给定X_n计算 n.

下面就来讲解这些算法。

一、 已知 n 和X_0，计算X_n

当 m 为2^n时，计算公式可简化为：

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

通过这个公式，我们可以从第 n 个随机数推出第 n+1 个随机数。简便起见，我们把这个公式称做“把随机数向前推了 1 次”。设把随机数向前推 k 次的公式为 $F_k(x) = a_k x + c_k$（没错，所有F_k都有类似 LCG 公式的形式），则有 $F_1(x) = ax + c$，即$a_1 = a$，$c_1 = c$.

然后我们有：

$$F_2(x) = F_1\big(F_1(x)\big) = a_1(a_1 x + c_1) + c_1 = a_1^2 x + a_1 c_1 + c_1$$

即$a_2 = a_1^2$，$c_2 = a_1 c_1 + c_1$.可以注意到 $F_2(x)$ 仍然是 LCG 公式，只是循环周期并不是2^n.

以此类推有：

$$a_4 = a_2^2, c_4 = a_2 c_2 + c_2$$
$$a_8 = a_4^2, c_8 = a_4 c_4 + c_4$$
$$a_{16} = a_8^2, c_{16} = a_8 c_8 + c_8$$
$$\cdots$$
$$a_{2^{n-1}} = a_{2^{n-2}}^2,\ c_{2^{n-1}} = a_{2^{n-2}} c_{2^{n-2}} + c_{2^{n-2}}$$

注意在电脑程序中，CPU 会把所有a_k和c_k对2^n取模，但这并不影响结果。

那么，对于任意的 k，怎样计算F_k呢？只需把 k 写成许多 2 的幂之和，然后将所有的 F_{2^i} 复合即可。例如，如果我们要求F_3，由于$3 = 1 + 2$，我们有：

$$F_3(x) = F_2\big(F_1(x)\big) = a_2(a_1 x + c_1) + c_2$$

你可能会注意到 $F_2\big(F_1(x)\big) = a_2(a_1 x + c_1) + c_2$ 和 $F_1\big(F_2(x)\big) = a_1(a_2 x +$

$c_2) + c_1$ 并不相同。然而，如果把所有的a_2和c_2展开为原始的 a 和 c，那么两者就完全相同。虽然没有证明，但我认为对所有 k，F_k总是能且只能写成唯一一种 F_{2^i} 的复合。

计算出F_k后，就能通过 $X_k = F_k(X_0)$ 快速计算X_k了。

二、 已知X_{n+1}，计算X_n

设 $F_{-1}(x)$ 为将随机数向前推 1 次的公式，即 $F_{-1}(x) = F_1^{-1}(x)$. 注意到原来的 LCG 算法的循环周期为2^n，故有 $F_{-1}(x) = F_{2^n-1}(x)$. 我们可以用上一节提到的方法来计算 F_{2^n-1}. 然后就可以通过公式 $X_k = F_{-1}(X_{k+1})$ 从X_{k+1}计算出X_k了。

由上一节可知，F_{-1} 也是一个 LCG 公式。因此我们可以用相同的方法计算出F_{-2}、F_{-4}等等。

三、 已知$X_0 = 0$，给定X_n计算 n

这个比较麻烦。首先引入 2 个未经证明的引理：当一个 LCG 算法的模为2^n，且循环周期为2^n时：

1. 随机数 X 的二进制的第 i 位（从低到高，下同）的周期为2^i;
2. 如果X_k的二进制的第 i 位是 0, 则 $X_{k+2^{i-1}}$ 的二进制的第 i 位是 1，反之亦然。

虽然没有证明，但我觉得它们是对的。至少没有找到反例。根据这两条引理我们就能找到一个从X_k计算出 k 的快速算法：

首先需要一个累加器。考察X_k的二进制的最低位。如果是 1，我们就把X_k向前推 1 次，并给累加器加上 1。令 $X_{k'} = F_{-1}(X_k)$，根据引理 2，$X_{k'}$的最低位应该是 0. 如果X_k的最低位是 0，那么令$X_{k'} = X_k$，并且不要在累加器上加任何数。无论哪一种情况，$X_{k'}$的最低位总是 0.

然后考察$X_{k'}$的第 2 位。如果是 1，我们就把$X_{k'}$向前推 2 次，并给累加器加上 2。令 $X_{k''} = F_{-2}(X_{k'})$，根据引理 2，$X_{k''}$ 的第 2 位应该是 0，又根据引理 1，$X_{k''}$的最低为会保持为 0. 如果$X_{k'}$的第 2 位是 0，那么令$X_{k''} = X_{k'}$，并且不要在累加器上加任何数。无论哪一种情况，$X_{k''}$ 的最低位和第 2 位总是 0.

不停重复这个过程，每次都会把特定的位变成 0，并且比这一位更低的位都会保持为 0。由于X_k只有有限位，最终就会变成 0，此时我们就得出 k 就等于向前推的次数，而这个次数被记录在了累加器中。

我们也可以用 F_1, F_2, F_4, … 而非 F_{-1}, F_{-2}, F_{-4}, … 来把随机数往后推。算法是相同的，只是在变成 0 之前是向后推了 k 次，而非向前推。此时我们要求的结果是$2^n - k$而不是 k.