Tree-Structured Pseudo-Random Sequences

TR88-003

January, 1988

John H. Halton

The University of North Carolina at Chapel Hill
Department of Computer Science
Sitterson Hall, 083A
Chapel Hill, NC 27599-3175

TREE-STRUCTURED PSEUDO-RANDOM SEQUENCES

by

John H. Halton

Computer Science Department
The University of North Carolina
Chapel Hill, NC 27599

ABSTRACT

A class of families of linear congruential pseudo-random sequences is defined, for which it is possible to *branch* at any event without changing the sequence of random numbers used in the original random walk, and for which the sequences in different branches show properties analogous to mutual statistical *independence*. This is a hitherto unavailable, and computationally desirable, tool.

1. INTRODUCTION

During the last forty or fifty years, the *Monte Carlo method* has been used with considerable success, to solve large mathematical problems too computationally complicated to yield to the classical numerical methods developed during the previous four centuries. For general discussions, the reader is referred to, e.g., BUS 62, HAM 64, HAL 70, ERM 71, SOB 73, KLE 75, YAK 77, or RUB 81 [references in this format are to the Bibliography at the end of this paper]. In particular, there is an extensive history of the effective application of the Monte Carlo method to *particle-transport problems*, such as arise in the design of radiation shielding, nuclear reactors, and fission and fusion bombs (see, e.g., CAR 75, SPA 69).

While the method was originally conceived in terms of *representing the solution of a problem as a parameter of a hypothetical population, and using a* [truly] *random sequence of numbers to* *construct a sample of the population, from which statistical estimates of the parameters can be obtained* (see HAL 70); it soon became apparent, from the point of view of the need, both for repeatable results to 'debug' the Monte Carlo computer programs and for a large, stable supply of suitable 'random numbers', that certain deterministic sequences exhibiting some of the properties of truly random sequences would be more useful in practice. These became known as *pseudo-random sequences* (and, by corruption of terms, as sequences of 'pseudo-random numbers') (see the above-mentioned references, and also LEH 51, HUL 62, TAU 65, JAN 66, and NIE 78). Somewhat later, even less 'random-looking' sequences, dubbed *quasi-random*, having exceptionally good uniformity properties and leading to fast convergence of the resulting Monte Carlo estimates, were proposed (see HAM 60, HAL 60, ZAR 66, and HAL 72). The uniformity of distribution of the pseudo-random sequences was found to be imperfect when they were used to define points in several dimensions (FRA 63, GRE 65, MAR 72), and several non-statistical approaches were developed for error-analysis.

One of the most successful classes of pseudo-random number-generators is the so-called *linear-congruential* algorithm (originally due to Lehmer; see LEH 51). The sequence $[\xi_0, \xi_1, \xi_2, \xi_3, \ldots] = [\xi_j]_{j=0}^{\infty}$ of *canonical pseudo-random numbers*, which should be independently uniformly distributed in the semi-open unit interval [0, 1), is obtained from an integer sequence $[x_0, x_1, x_2, x_3, \ldots] = [x_j]_{j=0}^{\infty}$, by

$$\xi_j = x_j/2^M; \tag{1}$$

and the x_j are uniquely determined by selecting M, a, b, and x_0, and taking

$$(\forall j \geq 0) \quad 0 \leq x_j < 2^M, \quad x_{j+1} \equiv ax_j + b \pmod{2^M}. \tag{2}$$

Given the integer parameters a and b and an initial integer x_0; each successive x_{j+1} is the *residue* of $ax_j + b$ *modulo* 2^M (i.e., the remainder when $ax_j + b$ is integer-divided by 2^M). Given integers Z and $Q > 0$, we shall henceforth write

$$R = \langle Z|Q\rangle \quad \Leftrightarrow \quad \{0 \leq R < Q, \quad R \equiv Z \pmod{Q}\}. \tag{3}$$

(When, as here, $Q = 2^M$ and we perform a binary computation, such as is now universally used in digital computers, this residue is easily obtained, as the integer consisting of the M least significant bits of Z.) Therefore (2) will take the form:

$$(\forall j \geq 0) \quad x_{j+1} = \langle a x_j + b \,|\, 2^M \rangle. \qquad (4)$$

Many calculations using the Monte Carlo method (including those of particle transport alluded to above) involve the use of long sequences of pseudo-random numbers to generate sequential histories of flights and collisions, usually referred-to as *random walks*. By averaging appropriately-selected *scores* (functions of single random walks generated in this way) over large numbers of such random histories, it is possible to estimate the parameters of interest with considerable accuracy.

It is clear that different random sequences will, in general, produce different random-walk histories; and these latter, in turn, will generally lead to different scores. While it is inherent in the Monte Carlo method that its results should show random fluctuations, it is extremely convenient to be able to reproduce a given computational result exactly, when we wish to do so. In particular, this is important in the initial 'debugging' stage of developing a new program (or program-module), when we need to separate the effects of desirable randomness from those of undesirable programming errors, so as to ensure that the program or module will do correctly what the programmer intends; and it is also useful when several runs must be made, to develop intentionally-correlated random samples, all depending on the same random walk. Some of these ends can be achieved by storing, and later retrieving, the values of the thousands, millions, or even billions, of random numbers required; but it is clearly much more convenient to redesign the random generator (algorithm) in such a way that no such mass-storage is required. The original invention of pseudo-random sequences was partly motivated by this need.

When one attempts to refine the physics underlying a particle-transport computation, by taking into account the concomitant generation and subsequent motion of additional particles or radiation, it is useful to compare the scores obtained with and without these refinements, for the same random walks. Since this leads to situations in which the random walks *branch* in a tree-like manner, requiring random sequences of differing lengths and unpredictable relationships, the problem becomes far more complex. We are now required to be able to generate a *tree-structure* of pseudo-random numbers, with good uniformity properties within each branch and good properties of independence between branches. In a typical conventional particle-transport calculation, using non-branching random walks, we may compute some 10^3 - 10^5 random walks, averaging perhaps 10^2 - 10^4 steps each, with every step requiring around 10 random numbers; this adds up to a need for something of the order of 10^6 - 10^{10} random numbers. With current generators having periods of the order of 10^{14}, such a requirement is acceptable; and techniques are available to increase the periods (without unacceptably increasing the time required to generate the random numbers) to the order of 10^{60} or so.

However, if our model is expanded to allow branching at every step, a comparable tree-structured calculation would, in principle, need perhaps $10^4 \times 2^{10^5}$, or about 10^{3000} random numbers. It is, of course, entirely out of the question, in any case, to *use* this many random numbers; since, according to current astrophysical thought, the calculation would hardly have begun when the Sun, in its red-giant phase, would consume the Earth, just a mere 10^{26} - 10^{27} nanoseconds from now! The problem is, rather, to provide theoretical access to suitably-distributed random numbers; so that they will be available as and when needed. The actual consumption of random numbers in a computation of this kind could hardly exceed some 10^{16} or so, unless computer technology makes rather remarkable progress even in comparison with its astonishing record; thus, we must rely on sampling techniques such as 'Russian roulette' to keep the overall needs down. Nevertheless, we must be able to generate those random numbers that we *do* need, with appropriate properties of distribution. The present development is an attempt to address this potential need. The problem was first raised by Warnock (see WAR 83, FRE 84) and useful suggestions of a general and heuristic nature were made by him as to its solution. In the present paper (expanding on ideas first presented in HAL 87), I propose a possible explicit approach to the task of generating a large number of branching pseudo-random sequences which are mutually independent in a rigorously specified manner.

2. PRELIMINARIES

For any positive integer n and real a, let

$$S_0(a) = 0 \quad \text{and} \quad S_n(a) = 1 + a + a^2 + a^3 + \ldots + a^{n-1}. \tag{5}$$

This is consistent, since the sum $S_n(a)$ has n terms. Then

$$S_n(a) = n, \quad \text{if} \quad a = 1, \tag{6}$$

and

$$S_n(a) = (a^n - 1)/(a - 1), \quad \text{if} \quad a \neq 1. \tag{7}$$

Lemma 1. *For any non-negative integer m and real z,*

$$S_{2m}(z) = (1 + z)\, S_m(z^2). \tag{8}$$

≪By (5), if $m = 0$, then (8) is immediate; and, otherwise,

$$S_{2m}(z) = (1 + z) + (z^2 + z^3) + \ldots + (z^{2m-2} + z^{2m-1})$$
$$= (1 + z)\,(1 + z^2 + z^4 + \ldots + z^{2m-2}); \tag{9}$$

which yields (8) at once.≫ [Proofs will, throughout this paper, be enclosed between ≪ and ≫.]

Definition 1. If N is any positive integer, then we express the fact that another positive integer k is a factor of N [i.e., integer-divides it, without remainder] by the usual notation

$$k \mid N. \tag{10}$$

We now see, in particular, that there is a *unique* non-negative integer u, such that k^u divides N, but k^{u+1} does not. We shall write

$$k^u \Uparrow N \tag{11}$$

to express this situation. If $v \le u$, then we also have, as in (10), that

$$k^v \mid N. \tag{12}$$

We extend the notation (11) to $N = 0$ by writing, for any $k > 0$,

$$k^\infty \Uparrow 0. \tag{13}$$

The notation defined in (11) and (13) is slightly tricky: while $k \mid N$ is a relation between *two* integers, k and N; $k^u \Uparrow N$ is a relation between *three* integers, k, u, and N. When we use an abbreviation, such as "$8 \Uparrow x$", it will be understood to mean "$2^3 \Uparrow x$": the member on the left of the symbol $\Uparrow$ will always be a pure power of one uniquely determined k. Hereinafter, we shall particularly make use of the special case, when $k = 2$.

Lemma 2. *For any odd positive integer a, there are unique positive integers q and r, such that*

$$a = (2r - 1)\,2^q - 1. \tag{14}$$

≪Since a is odd, $a + 1$ is necessarily even. Thus, there is a unique *maximum* q for which $2^q \mid (a + 1)$, and $q \ge 1$. For this q, we have $2^q \Uparrow (a + 1)$. Also, the quotient, when we divide $(a + 1)$ by 2^q, is odd; whence it can be expressed uniquely in the form $(2r - 1)$. This immediately yields (14).≫

$k^v \mid N.$ (12)

We extend the notation (11) to $N = 0$ by writing, for any $k > 0$,

Lemma 3. *With a, q, and r defined as in Lemma 2; if $u \geq 0$ and $v \geq 0$ are the unique integers such that $2^u \Uparrow n$ and $2^v \Uparrow S_n(a)$, then $v = u + q - 1$; that is,*

$$2^{u+q-1} \Uparrow S_n(a) \quad \textit{if and only if} \quad 2^u \Uparrow n. \tag{15}$$

≪By repeated application of Lemma 1, we get that

$$\begin{aligned} S_n(a) &= (1 + a)\, S_{n/2}(a^2) = (1 + a)(1 + a^2)\, S_{n/4}(a^4) = \ldots \\ &= (1 + a)(1 + a^2)(1 + a^4) \ldots (1 + a^{2^{u-1}})\, S_{n/2^u}(a^{2^u}). \end{aligned} \tag{16}$$

Also, by (14), $2^q \Uparrow (1 + a)$, and $q \geq 1$; and every binomial factor on the right of (16), after the first one, is of the form $1 + a^{2m}$, with integer $m \geq 1$. Since a is odd, either $a \equiv 1$ or $a \equiv 3 \pmod 4$; whence $a^2 \equiv 1 \pmod 4$; and, therefore,

$$(\forall m \geq 1) \quad a^{2m} \equiv 1 \pmod 4. \tag{17}$$

Hence, $(\forall m \geq 1)\ 1 + a^{2m} \equiv 2 \pmod 4$; i.e., $(\forall m \geq 1)\ 2 \Uparrow (1 + a^{2m})$. Therefore, the product of all the binomial factors on the right of (16) is divisible by 2 exactly $q + (u - 1)$ times. Finally, we observe that, since a is odd by our hypothesis, every power of a is odd too; whence, by (5), the last factor on the right of (16) is the sum of an odd number, $n/2^u$, of odd numbers, and so must itself be odd. Thus, when u and v are defined as stated, $v = q + u - 1$, and (15) follows immediately.≫

Definition 2. If $[x_0, x_1, x_2, \ldots] = [x_j]_{j=0}^{\infty}$ is a sequence of numbers, and if we are given that, for some $0 \leq i < j$,

$$(\forall k \geq 0) \quad x_{j+k} = x_{i+k}, \tag{18}$$

then we say that the sequence is *periodic*. If λ is the *least* value of the difference $j - i$, for which (18) holds, then we say that the *period* is λ.

If h is the *least* value of i satisfying (18) for $j - i = \lambda$, we say that the periodicity *starts at* index h; and if $h = 0$, then we say that the sequence is *completely periodic.*

Note that, if the sequence $[x_j]_{j=0}^{\infty}$ is periodic, with period λ, starting at index h; then, for any *offset* α, the same is true of the sequence $[x_j - \alpha]_{j=0}^{\infty}$.

Lemma 4. *Given that the sequence $[x_j]_{j=0}^{\infty}$ is periodic with period λ, starting at index h, and given i and j, with $i < j$, satisfying the relation* (18); *it follows that $\mu = j - i$ is an integer multiple of λ; that is,*

$$\lambda \mid \mu. \tag{19}$$

≪Since λ is minimal, we have $0 < \lambda \leq \mu$. Because the sequence is periodic with period λ, starting at index h; it is clear from (18) that $x_{h+k} = x_{(h+\lambda)+k} = x_{h+(\lambda+k)} = x_{(h+\lambda)+(\lambda+k)} = x_{h+(2\lambda+k)} = \ldots$; that is, by induction on integers r,

$$(\forall k \geq 0)\ (\forall r \geq 0) \quad x_{h+r\lambda+k} = x_{h+k}; \tag{20}$$

and, similarly, by (18) for i and j, by induction on integers s,

$$(\forall k \geq 0)\ (\forall s \geq 0) \quad x_{i+s\mu+k} = x_{i+k}. \tag{21}$$

Write $n = \max\{i, h\}$, so that $n \geq h$ and $n \geq i$; and replace k, throughout (20), by $k + n - h$ and, throughout (21), by $k + n - i$. Then, whatever is true with the resulting universal quantifiers, namely, $(\forall k \geq h - n)$ and $(\forall k \geq i - n)$, is also true with the quantifier $(\forall k \geq 0)$; so that

$$(\forall k \geq 0)\ (\forall r \geq 0)\ (\forall s \geq 0)\ x_{n+r\lambda+k} = x_{n+k} = x_{n+s\mu+k}. \tag{22}$$

The **Euclidean Algorithm Theorem** states that, if γ denotes the g.c.d. of positive λ and μ (so that $\gamma \mid \lambda$ and $\gamma \mid \mu$, and γ is *maximal*), there are integers U_0 and V_0 such that $\gamma = U_0\lambda + V_0\mu$. *Proof:* ≪Let $\mathbb{Z}$ be the set of all integers. The set $\Theta = \{\theta = U\lambda + V\mu \colon U \in \mathbb{Z}, V \in \mathbb{Z}\}$, has a subset $\Theta^+ = \{\theta = U\lambda + V\mu \colon U \in \mathbb{Z}, V \in \mathbb{Z}, \theta > 0\}$, which is non-empty, since $0 < \lambda = 1 \times \lambda + 0 \times \mu \in \Theta^+$ and $0 < \mu = 0 \times \lambda + 1 \times \mu \in \Theta^+$. Let $\kappa = U_0\lambda + V_0\mu$ be the *least* $\theta \in \Theta^+$. Integer-divide λ by κ; then $\lambda = \sigma\kappa + \rho$ (where $0 \leq \rho < \kappa$), and so $\rho = \lambda - \sigma\kappa = (1 - \sigma U_0)\lambda - \sigma V_0\mu \in \Theta$. Since $\rho < \kappa$, and κ is minimal in Θ^+, $\rho \notin \Theta^+$; and therefore $\rho = 0$ (i.e., $\kappa \mid \lambda$). Integer-divide μ by κ, to show, similarly, that $\kappa \mid \mu$; whence $\kappa \mid \gamma$, since γ is the maximal divisor. Since we also know that $\gamma \mid \lambda$, $\gamma \mid \mu$, and $\kappa \in \Theta$; $\gamma \mid \kappa$. Therefore, $\kappa = \gamma$. This proves the theorem.≫ Now, U_0 and V_0 must have opposite signs, since we have that $0 < \gamma \leq \lambda \leq \mu$; so that there must be non-negative integers r_0 and s_0, such that either (i) $r_0\lambda - s_0\mu = \gamma$ or (ii) $s_0\mu - r_0\lambda = \gamma$. In both cases, take $r = r_0$ and $s = s_0$

in (22); then, in case (i), replace n by $\nu - s_0\mu$; in case (ii), replace n by $\nu - r_0\lambda$. Either way, we see that

$$(\forall k \geq 0) \quad x_{\nu+\gamma+k} = x_{\nu+k}. \tag{23}$$

But this means that the sequence is periodic, with period at most γ. Since λ is minimal, by Definition 2, we must have $\lambda \leq \gamma$. Thus, $\gamma = \lambda$, and the lemma follows at once.≫

This means that the period of a periodic sequence is unique.

Definition 3. Given a semi-open interval $[A, B)$ on the real line, and a set J of Q points $z_1 < z_2 < \ldots < z_Q$ in it, we say that the points are *cyclically equally spaced* in $[A, B)$ if

$$z_{h+1} - z_h = (B - A)/Q \quad \text{for} \quad h = 1, 2, \ldots, Q - 1. \tag{24}$$

Note that this implies that $(z_1 - A) + (B - z_Q) = (B - A)/Q$ also, since $z_Q - z_1 = (Q - 1)(B - A)/Q$. If we imagine the interval $[A, B)$, with the points of J in it, wrapped around a circle; then these Q points would be equally-spaced around the circle. Note, too, that, if the set J is cyclically equally spaced in $[A, B)$, so is any *offset* set of points $z_h - \alpha$ (reduced, modulo $B - A$, to fall in the interval).

Definition 4. Given a set J of Q points cyclically equally spaced in an interval $[A, B)$; if the sequence $[x_j]_{j=0}^{\infty}$ is periodic, with period λ, starting at index h, and if the set $K_0 = \{x_j\}_{j=h}^{\infty}$ of values taken by the x_j, once the periodicity is established, is a subset of J, with P distinct points in it, and $P = \lambda$; and if, further, these P values are also cyclically equally spaced in the interval $[A, B)$; then we say that the sequence is *uniform* in J, with *coarseness* Q/P.

Lemma 5. *In the situation described in Definition 4,*

$$P \mid Q; \tag{25}$$

so that the coarseness of a uniform sequence is always a positive integer.

≪The points of J may be thought of as equally spaced around a circle of circumference $B - A$; the points of K (which are also in J) are also equally spaced around the circle. Thus, there is an integer G, such that adjacent points of K have a spacing just G times as great as that of adjacent points of J; that is, $PG = Q$; whence (25) follows. G is therefore the coarseness of the sequence in J.>>

Note that, if the period of the sequence $[x_j]_{j=0}^{\infty}$ passes through *all* the points of J (that is, if $P = Q$), then the coarseness of the sequence in J takes its minimum possible value, namely, 1.

Definition 5. Given a set J of Q points cyclically equally spaced in an interval $[A, B)$; if two sequences $[x_j]_{j=0}^{\infty}$ and $[x^{\dagger}_j]_{j=0}^{\infty}$ are such, that the difference-sequence, $[\delta_j]_{j=0}^{\infty}$, where

$$(\forall j \geq 0) \quad \delta_j = \langle x_j - x^{\dagger}_j \,|\, 2^M \rangle, \tag{26}$$

is periodic, and is uniform in J with coarseness G; then we say, by analogy with the definition of uniformity and coarseness, that the two sequences are *independent* with respect to J, and that their *consonance* is G.

3. ANALYSIS OF LINEAR CONGRUENTIAL GENERATORS

We are interested in generating a canonical pseudo-random sequence $[\xi_j]_{j=0}^{\infty}$ of numbers in $[0, 1)$, for use in Monte Carlo computations. We therefore want the ξ_j to take a large number of distinct values, distributed with near-constant density in $[0, 1)$. Our present consideration will be limited to the *linear congruential* sequences, which are related through (1) to the integer sequences $[x_j]_{j=0}^{\infty}$ defined in (2) or (4), with M a non-negative integer. This implies that, if we write (as we shall do henceforth)

$$2^M = Q, \tag{27}$$

then $$(\forall j \geq 0) \quad x_j \in J = \{0, 1, 2, \ldots, Q-1\}, \tag{28}$$

and therefore

$$(\forall j \geq 0) \quad \xi_j \in F = \{0, 1/Q, 2/Q, \ldots, (Q-1)/Q\}. \tag{29}$$

In the terminology of Definition 3, the sets J and F are cyclically equally spaced, in the semi-open intervals $[0, Q)$ and $[0, 1)$, respectively.

Note that we may assume, without loss of generality, that a and b are also integers selected from J. We further assume henceforth that $a \neq 0$. [If $a = 0$, then, clearly, by (4), for all $j \geq 1$, $x_j = b$.]

Lemma 6. *The recurrence relation* (4) *is satisfied, for all* $n \geq 0$, *by*

$$x_n = \langle a^n x_0 + S_n(a)\, b \,|\, Q \rangle; \tag{30}$$

where $S_n(a)$ *is defined as the sum in* (5).

$\ll$When $n = 0$, we know that $a^n = 1$ and the sum $S_n(a) = 0$; so that, in fact, $x_n = a^n x_0 + S_n(a) b$. Suppose that the relation holds for $n = k$, say (this is initially true when $k = 0$). Then, by (4) with (3), we have that

$$\begin{aligned} x_{k+1} &= \langle a x_k + b \,|\, Q \rangle = \langle a[a^k x_0 + S_k(a)\, b] + b \,|\, Q \rangle \\ &= \langle a^{k+1} x_0 + [a\, S_k(a) + 1] b \,|\, Q \rangle; \end{aligned} \tag{31}$$

and, by (5), it is easily seen that

$$a\, S_k(a) + 1 = S_{k+1}(a); \tag{32}$$

whence the congruence will also hold for $n = k + 1$. The lemma follows by induction.$\gg$

Lemma 7. *The sequence* $[x_j]_{j=0}^{\infty}$ *is periodic, with period not exceeding* Q.

$\ll$By (28), there are at most Q possible distinct values of x_j; among the $Q + 1$ numbers $x_0, x_1, x_2, \ldots, x_Q$, there must be two values alike, and we can always further specify that all intermediate values different from these and each-other: $x_i = x_j$, say, with $0 \leq i < j$ and $x_i, x_{i+1}, x_{i+2}, \ldots, x_{j-1}$ all different [if some intermediate value $x_k = x_i$, say, replace j by k; if two intermediate values $x_h = x_k$, say, replace i by h and j by k]. It is now clear from the form of (4) that (18) will hold, since each member of the sequence is determined solely and uniquely by its immediate predecessor, without regard to its position in the sequence. Hence, the sequence is periodic and, by Lemma 4, $j - i$ is a multiple of the period, which thus, clearly, cannot exceed Q.≫

Lemma 8. *If a is any even integer, the sequence $[x_j]_{j=0}^{\infty}$ is periodic, with period* 1.

≪We have already seen that the period is 1 when $a = 0$. For any even a, clearly $a^M \equiv 0 \pmod{Q}$; so there will be a unique minimal h, such that $a^h \equiv 0 \pmod{Q}$. If $n \geq h$; then, by (5),

$$S_n(a) \;=\; S_h(a) + a^h S_{n-h}(a) \;\equiv\; S_h(a) \pmod{Q}. \tag{33}$$

Therefore, in particular, by (30) and (33),

$$x_{h+1} \;=\; \langle a^{h+1} x_0 + S_{h+1}(a)\, b \,|\, Q\rangle \;=\; \langle S_h(a)\, b \,|\, Q\rangle$$

$$= \langle a^h x_0 + S_h(a)\, b \,|\, Q\rangle \;=\; x_h; \tag{34}$$

whence, by Definition 2, the sequence is periodic, starting at index h, with period 1.≫

Of course, a period of length 1 is of very little use for the generation of pseudo-random numbers; so we shall henceforth assume that a is odd.

Lemma 9. *If a is any odd integer, then the sequence $[x_j]_{j=0}^{\infty}$ is completely periodic.*

≪Consider the Q integers $1, a, a^2, \ldots, a^Q$, reduced modulo Q. Their values must lie in the set J; so, arguing exactly as in proving Lemma 7, we see that we must have $0 \leq i < j \leq Q$, such that $\langle a^i | Q\rangle = \langle a^j | Q\rangle$, while $\langle a^i | Q\rangle, \langle a^{i+1} | Q\rangle, \langle a^{i+2} | Q\rangle, \ldots, \langle a^{j-1} | Q\rangle$ are all different. Thus, $a^j - a^i = a^i(a^{j-i} - 1)$ must be divisible by Q; and since a is odd, it follows that $Q \mid (a^{j-i} - 1)$; so that there must be a positive integer $m = j - i \leq Q$, such that

$$a^m \;\equiv\; 1 \pmod{Q}. \tag{35}$$

By (2) and (35), we have that $x_{j-1} \equiv a^m x_{j-1} \equiv a^{m-1}(x_j - b) \pmod{Q}$; so that, writing $c = a^{m-1}$ and $d = -cb$, we have

$$(\forall j \;\geq\; 1) \quad x_{j-1} \;\equiv\; cx_j + d \pmod{Q}, \tag{36}$$

or, by (3), $$(\forall j \geq 1) \quad x_{j-1} = \langle c x_j + d \,|\, Q \rangle. \tag{37}$$

Thus, each member of the sequence is determined solely and uniquely by its immediate successor, without regard to its position in the sequence, and the equation (18) also holds for negative k, so long as the index $i + k \geq 0$. This extends the periodicity of the sequence (already established in Lemma 7) to the starting index 0, proving the present lemma.$\gg$

From now on, we shall always suppose that a is *odd*, satisfying (14) and thereby uniquely defining positive integers q and r, as stated in Lemma 2. Since we also suppose (without loss of generality) that $a \in J$, we see, by (28), that $1 \leq (2r - 1)\, 2^q - 1 \leq 2^M - 1$; whence $r \geq 1$, and therefore $2^q \leq 2^M$. Since $q \geq 1$, we conclude that

$$1 \leq q \leq M. \tag{38}$$

Now write $$W = \langle x_1 - x_0 \,|\, Q \rangle = \langle (a-1)x_0 + b \,|\, Q \rangle; \tag{39}$$

and, by appeal to Definition 1, put

$$2^c \Uparrow b, \quad 2^s \Uparrow x_0, \quad 2^d \Uparrow (a-1), \quad \text{and} \quad 2^g \Uparrow W. \tag{40}$$

Since (again without loss of generality) we also suppose that $b \in J$ and $x_0 \in J$, it now follows that, unless $b = 0$ [$c = \infty$] or $x_0 = 0$ [$s = \infty$],

$$0 \leq c < M \quad \text{and} \quad 0 \leq s < M; \tag{41}$$

and, since a is odd, $a - 1$ is even, whence $d \geq 1$.

Lemma 10. *The period λ of the completely periodic sequence $[x_j]_{j=0}^{\infty}$ is given by*

$$\lambda = 2^u, \quad \textit{where} \quad u = \max\{0, M - g - q + 1\}, \tag{42}$$

and g is defined uniquely by (39) *and* (40).

$\ll$By Definition 2 and (30), λ is the least j for which

$$x_0 = x_j = \langle a^j x_0 + S_j(a)\, b \,|\, Q \rangle. \tag{43}$$

If $a \neq 1$, by (7), $a^j x_0 - x_0 = S_j(a)\,(a-1)\,x_0$; whence, by (3), (39), and (43),

$$S_j(a)\, W \equiv 0 \pmod{Q}. \tag{44}$$

If $a = 1$, we note that $W = b$, and so (43) implies (44) directly. Thus (44) is true for all a. Therefore, either

$$W \equiv 0 \pmod{Q} \tag{45}$$

(i.e., $g \geq M$, including the possibility that $W = 0$ and $g = \infty$); or $g < M$, and

$$2^{M-g} \mid S_j(a). \tag{46}$$

If (45) holds, then clearly, by (4) and (39), $x_1 = x_0$; so that $\lambda = 1$. Thus, $u = 0$ and $M - g - q + 1 \leq 0$ [since, by the assumption of (45), $g \geq M$, and, by (38), $q \geq 1$]; so that (42) is satisfied.

If, instead, $g < M$ and (46) holds, we observe that, by Lemma 3, $2^{u+q-1} \Uparrow S_j(a)$ if and only if $2^u \Uparrow j$; whence there is an integer $u \geq 0$, such that $u + q - 1 \geq M - g$ and $2^u \Uparrow \lambda$. Thus, since the period λ is minimal, u will be the *least* non-negative solution of

$$\lambda = 2^u \quad \text{and} \quad u + q - 1 \geq M - g. \tag{47}$$

Clearly, this is given by (42).$\gg$

Lemma 11. *With g defined by* (39) *and* (40);

(i) *if $c < s + d$, then $g = c$;*

(ii) *if $c = s + d$, then $g > c$;*

(iii) *if $c > s + d$, then $g = s + d$.*

$\ll$By (40), $2^{s+d} \Uparrow (a-1)x_0$ and $2^c \Uparrow b$. Write $(a-1)x_0 = 2^{s+d}\,U$ and $b = 2^c\,V$, where U and V are odd integers. By (39), there are now three cases, characterized as in our lemma. (i) If $c < s + d$, then $W = \langle (a-1)x_0 + b \,|\, Q\rangle = \langle 2^c\{2^{s+d-c}\,U + V\} \,|\, Q\rangle = 2^c\,X_1$, and the factor X_1 is *odd*; so that $g = c$. (ii) If $c = s + d$, then $W = \langle 2^c\{U + V\} \,|\, Q\rangle = 2^c\,X_2$, and the factor X_2 is *even*, being the sum of two odd numbers; so that $2^{c+1} \mid W$ (that is, $g > c$). (iii) If $c > s + d$, then $W = \langle 2^{s+d}\{U + 2^{c-s-d}\,V\} \,|\, Q\rangle = 2^{s+d}\,X_3$, and the factor X_3 is *odd*; so that $g = s + d$.$\gg$

The value of M is mainly machine-dependent ($M = 48$ is typical of 'supercomputers', and then $Q = 2^{48} \approx 2.8 \times 10^{14}$). As we shall see later, it is not always possible to control the parity of b; but we can, and do, control the value of a (and thus the parity of $a - 1$). We naturally seek to make the period of the sequence as long as possible. The absolute maximum is clearly $Q = 2^M$, but this cannot always be attained. Referring to Lemma 10, we see that both q and g should be as small as possible; and, since, by (38), $q \geq 1$, we stipulate that

$$q = 1. \tag{48}$$

By the definition (14) of q and r, this is equivalent to $a = (2r - 1)2 - 1 = 4(r - 1) + 1$; so that

$$a \equiv 1 \pmod{4}. \tag{49}$$

By the definition (40) of d, we have that, for some integer r',

$$a = (2r' - 1)2^d + 1 \tag{50}$$

[compare (14)], which implies that

$$a \equiv 1 \pmod{2^d}. \tag{51}$$

Now, we have (above) that $a - 1 = 4(r - 1)$; so that, by (50),

$$d \geq 2. \tag{52}$$

Conversely, by (50), if we assume (52), $a - 1 = (2r' - 1)2^d = 4r''$, which implies (49); further, $a = (2r'' + 1)2 - 1$, which yields (48), by (14).]

First, let us consider what happens when $b \neq 0$.

Lemma 12. *Under the conditions of Lemmas 10 and 11, if we impose the restrictions* (50) *and* (52) *on the parameter* a *and suppose that* $b \neq 0$, *then*

(i) *if* $c \leq s + d - 1$, *the period of the sequence is* $2^{M-c} \geq 2$;

(ii) *if* $c = s + d$, *the period of the sequence is* $\max\{1, 2^{M-g}\}$, *where* $g \geq c + 1$;

(iii) *if* $c \geq s + d + 1$, *the period of the sequence is* $2^{M-s-d} \geq 4$.

≪As we have seen, (50) and (52) imply that $q = 1$. Thus, (42) reduces to

$$\lambda = 2^u, \quad \text{where} \quad u = \max\{0, M - g\}; \tag{53}$$

and the three cases of Lemma 11 are the same as those of the present lemma.

(i) If $c \leq s + d - 1$, then $g = c$. By (41), since $b \neq 0$, $c < M$, and it follows that $M - g = M - c \geq 1$; so that, by (53), $\lambda = 2^{M-c} \geq 2^1 = 2$.

(ii) If $c = s + d$, then $g > c$; and, by (53), $\lambda = \max\{1, 2^{M-g}\}$.

(iii) If $c \geq s + d + 1$, then $g = s + d$. Since $b \neq 0$, by (41) and our hypothesis, $s + d < c < M$, so we get that $M - g = M - s - d \geq 2$; so that, by (53), $\lambda = 2^{M-s-d} \geq 2^2 = 4$.≫

Now we turn to the omitted case, when $b = 0$ and $c = \infty$. By (4) or (30), we see that

$$x_n = \langle a^n x_0 | Q \rangle. \tag{54}$$

Therefore, if $x_0 = 0$, every $x_n = 0$ too; so that $\lambda = 1$. If, on the other hand, $x_0 \neq 0$, so that $2^s \Uparrow x_0$, with $0 \leq s < M$; we can write $x_0 = 2^s \omega_0$, where ω_0 is odd, and we see that (since a is odd) $2^s \Uparrow x_n$ too; so that, for all n,

$$x_n = 2^s \omega_n, \tag{55}$$

where ω_n is odd. Thus, (54) reduces, on division by 2^s, to

$$\omega_n = \langle a^n \omega_0 | 2^{M-s} \rangle. \tag{56}$$

We are therefore led to examine the dependence on $m = M - s$ of the period λ_m of the sequence $[\omega_j]_{j=0}^{\infty}$ with ω_0 (and therefore all the ω_j) odd, when all numbers are reduced modulo 2^m. By (56), this problem is seen to be equivalent to that of finding the least n for which

$$a^n \equiv 1 \pmod{2^m}. \tag{57}$$

By (50) and (52), and since, clearly, if $u \geq v$,

$$X \equiv Y \pmod{2^u} \Rightarrow X \equiv Y \pmod{2^v}; \tag{58}$$

it follows that the λ_m are nondecreasing as $m \to \infty$, and that

$$\lambda_1 = \lambda_2 = \ldots = \lambda_d = 1. \tag{59}$$

As a further preliminary, we need the following result.

Lemma 13. *When a satisfies* (50) *and* (52), *the least value of* n *for which* (57) *holds is* 2^{m-d}, *for all* $m \geq d$.

≪Since λ_m is the least n for which (57) holds; for each m, there is an integer q_m, such that

$$a^{\lambda_m} = 1 + q_m 2^m. \tag{60}$$

Suppose it known that $\lambda_m = 2^{m-d}$ for all $d \leq m \leq h$; by (59), this is certainly true for $h = d$. Putting $\lambda_h = 2^{h-d}$ in (60), we get that $a^{2^{h-d}} = 1 + q_h 2^h$; and, on squaring, this yields

$$a^{2^{h+1-d}} = \left(a^{2^{h-d}}\right)^2 = \left(1 + q_h 2^h\right)^2 = 1 + q_h 2^{h+1} + q_h^2 2^{2h}.$$

Therefore, since $h \geq d \geq 2$, by (52); we get that

$$a^{2^{h+1-d}} \equiv 1 \pmod{2^{h+1}}; \tag{61}$$

whence $\lambda_{h+1} \leq 2^{h+1-d}$. Further, since the λ_m are nondecreasing, we get $\lambda_{h+1} \geq \lambda_h = 2^{h-d}$. If we let $X = \lambda_{h+1} - 2^{h-d}$, so that $0 \leq X \leq 2^{h-d}$, then

$$a^{\lambda_{h+1}} = a^{X+2^{h-d}} = a^X a^{2^{h-d}} = a^X (1 + q_h 2^h). \tag{62}$$

Let $a^X = Y + s\, 2^h$, with $0 \leq Y < 2^h$. Then $a^{\lambda_{h+1}} = (Y + s\, 2^h)\,(1 + q_h 2^h) \equiv Y + (Y q_h + s\,)2^h \equiv Y + Z\, 2^h \pmod{2^{h+1}}$, where $Z = \langle Y q_h + s \,|\, 2 \rangle$ is 0 or 1. Since $a^{\lambda_{h+1}} \equiv 1 \pmod{2^{h+1}}$ and $0 \leq Y < 2^h$, it is clearly necessary that $Y = 1$ and $Z = 0$; so that $a^X \equiv 1 \pmod{2^h}$; whence $X \geq 2^{h-d}$. Since we also have $X \leq 2^{h-d}$, it follows that $X = 2^{h-d}$; whence $\lambda_{h+1} = 2^{h-d} + X = 2^{h-d} + 2^{h-d} = 2^{h+1-d}$. The lemma now follows by induction.≫

Lemma 14. *When a satisfies* (50) *and* (52) *and* $b = 0$, *the period of the sequence* $[x_j]_{j=0}^{\infty}$ *is* $\max\{1, 2^{M-s-d}\}$.

≪(i) If $x_0 = 0$, $s = \infty$ and, as we have seen, $\lambda = 1$, agreeing with the lemma. (ii) If $x_0 \neq 0$ and $M - s - d \leq 0$; then $1 \leq M - s \leq d$, by (41). Since $m = M - s$ in (57), we get by (59) that $\lambda = \lambda_{M-s} = 1$, again agreeing with the lemma. (iii) Otherwise, $x_0 \neq 0$ and $M - s - d > 0$, and the lemma asserts that the sequence $[x_j]_{j=0}^{\infty}$ has a period 2^{M-s-d}. Now, the period of the sequence $[x_j]_{j=0}^{\infty}$, given by (54), is clearly, by

(55) and (56), the same as that of the sequence $[\omega_j]_{j=0}^{\infty}$ with ω_0 odd; and this, in turn, equals the least n for which (57) holds, when $m = M - s$. By Lemma 13, this is 2^{M-s-d}, completing the proof of our lemma.$\gg$

Lemmas 12 and 14 show the general desirability of using odd values of b. Then, $c = 0$, and we are in Case (i) of Lemma 12, with $\lambda = 2^M$, the optimal situation. However, as we shall see later, this will not always be possible to achieve.

It is interesting to see under what circumstances the least desirable situation (namely, when $\lambda = 1$) occurs. We already know, by Lemma 8, that this can happen when a is *even*. Lemma 12 now tells us that, when a is *odd* and satisfies (50) and (52), and $b \neq 0$, it can only happen in Case (ii), when $c = s + d$. Let us write

$$x_0 = 2^M - \theta, \quad a - 1 = 2^M - \alpha, \quad b = 2^M - \beta; \tag{63}$$

where, by (50), $\alpha = 2^d\,(2U - 1)$ with $1 \le U \le 2^{M-d-1}$; and, since b and x_0 are in J, $\beta = 2^{s+d}\,(2V - 1)$ with $1 \le V \le 2^{M-s-d-1}$, and $\theta = 2^s\,(2X - 1)$ with $1 \le X \le 2^{M-s-1}$. Then, by (39),

$$W = \langle 2^{2M} - 2^M(\alpha + \theta - 1) + \alpha\theta - \beta \,|\, Q\rangle, \tag{64}$$

and therefore, by (53), we get that $\lambda = 1$ if and only if $g \ge M$; i.e., if and only if

$$\beta = \alpha\theta, \quad \text{or} \quad V = 2UX - U - X + 1. \tag{65}$$

Finally, Lemma 14 tells us that we can have $\lambda = 1$ when $b = 0$, either if $x_0 = 0$ or if x_0 is a multiple of 2^{M-d}.

Lemma 15. *If the sequence* $[x_j]_{j=0}^{\infty}$ *is generated by* (4), *with the parameter a odd, then, given* (40), *we have that*

(a) *if* $c \le s - 1$, $\quad (\forall j \ge 0)\ \left\{2^{c+1} \mid x_{2j} \ \text{ and } \ 2^c \Uparrow x_{2j+1}\right\}$;

(b) *if* $c = s$, $\quad (\forall j \ge 0)\ \left\{2^c \Uparrow x_{2j} \ \text{ and } \ 2^{c+1} \mid x_{2j+1}\right\}$;

(c) *if* $c \ge s + 1$ *or* $c = s = \infty$, $\quad (\forall j \ge 0)\ 2^s \Uparrow x_j$.

$\ll$For all $j \geq 0$, define the powers t_j by

$$2^{t_j} \Uparrow x_j. \tag{66}$$

Then, by (4), since a is odd, $2^{t_j} \Uparrow ax_j$, and, by (40), $2^c \Uparrow b$. We recall that it is possible for b or any x_j to vanish, yielding that $c = \infty$ or $t_j = \infty$, respectively [see (13)]. Using an argument exactly analogous to that used in proving Lemma 11, we see that (i) if $b = x_j = 0$, then $c = t_j = \infty$, and, in fact, *every* $x_n = 0$ (including $x_0 = 0$); so that $s = \infty$ and $(\forall j \geq 0)$ $2^s \Uparrow x_j$; (ii) if $c < t_j$, then $2^c \Uparrow x_{j+1}$; (iii) if $c = t_j$, then x_{j+1} must be an *even* multiple of 2^c, so that $2^{c+1} \mid x_{j+1}$; and (iv) if $c > t_j$, then $2^{t_j} \Uparrow x_{j+1}$. Thus,

$$t_j > c \Rightarrow t_{j+1} = c; \quad t_j = c \Rightarrow t_{j+1} > c; \quad t_j < c \Rightarrow t_{j+1} = t_j. \tag{67}$$

But the sequence $[t_j]_{j=0}^{\infty}$ begins with $t_0 = s$; whence the lemma follows immediately.$\gg$

Lemma 16. *Given $M > 0$, with $Q = 2^M$ and $L = [0, Q)$; define the set J by* (28), *and let the sequence $[x_j]_{j=0}^{\infty}$ be periodic, with period λ, starting at index h. Let the set $K_0 = \{x_j\}_{j=h}^{\infty}$, of values of the x_j, once the periodicity has started, be a subset of J, consisting of just λ distinct values. Then a sufficient condition for the sequence $[x_j]_{j=0}^{\infty}$ to be uniform in J, is that there be integers α and p, with $0 \leq p \leq M$, such that*

$$\lambda = 2^{M-p} \quad \textit{and} \quad (\forall j \geq h) \;\; 2^p \mid (x_j - \alpha). \tag{68}$$

$\ll$Since the sequence $[x_j]_{j=0}^{\infty}$ is periodic, with period λ, starting at index h; the sequence $[x_j - \alpha]_{j=0}^{\infty}$, offset from the first by $-\alpha$, is also periodic, with the same period λ, starting at the same index h, as is noted after Definition 2. That the set K_0 has just λ distinct elements indicates that the period has no repeated values. Let $K_\alpha = \{x_j - \alpha\}_{j=h}^{\infty}$ be the set of periodic offset values, *reduced modulo* Q; clearly, these are also just λ in number. If we write

$$J_0 = J \quad \text{and} \quad J_p = \{r\,2^p:\ 0 \le r \le 2^{M-p} - 1\}, \tag{69}$$

then J_p is obviously the set of all integer multiples of 2^p in J (and so in L). Hence, the total number of such multiples is 2^{M-p}, and J_p is cyclically equally spaced in L (by Definition 3, since adjacent points are $2^p = (Q - 0)/2^{M-p}$ apart). If (68) holds, then K_α is clearly a subset of J_p, since 2^p divides every $x_j - \alpha$; and so, since $\lambda = 2^{M-p}$, K_α must equal J_p. Thus, K_α is cyclically equally spaced in L; and therefore so is the original set K_0, offset from K_α by $+\alpha$, as is noted after Definition 3. Thus, by Definition 4, the sequence $[x_j]_{j=0}^{\infty}$ is uniform in J, with coarseness Q/λ.>>

Lemma 17. *The period λ of the sequence $[x_j]_{j=0}^{\infty}$ generated by* (4) *equals the number P of distinct values in the set $K_0 = \{x_j\}_{j=h}^{\infty}$.*

<<We refer to the proof of Lemma 7. The $j - i$ values x_i, x_{i+1}, x_{i+2}, . . . , x_{j-1} are all different, and thereafter the values repeat, because, by (2) or (4), equal predecessors in the sequence have equal immediate successors, and because $x_i = x_j$. Thus, $P = j - i$. Therefore, by Lemma 4, P is a multiple of λ. But, since all P values in the above list differ, λ cannot be less than P; whence $\lambda = P$.>>

In Lemmas 9, 10, 12, 14, 15, 16, and 17, we have now marshalled all the facts we need to prove our main result.

Theorem 1. *Given the set J defined in* (28) *and the sequence $[x_j]_{j=0}^{\infty}$ generated by* (4) *with parameter a satisfying* (50) *and* (52); *the sequence is uniform in J, in the sense of Definition 4. When g is defined by* (39) *and* (40), *the coarseness of the sequence is given by*

(i) 2^c, *if* $c \le s + d - 1$;

(ii) $\min\{2^M, 2^g\}$, *if* $c = s + d$;

(iii) $\min\{2^M, 2^{s+d}\}$, *if* $c \ge s + d + 1$ *or* $c = s = \infty$.

≪≪Lemma 9 tells us that the sequence $[x_j]_{j=0}^{\infty}$ generated by (4) is completely periodic, since the parameter a is odd; and Lemma 17 tells us that the number, P, of distinct values of x_j in the period of the sequence equals its period, λ (i.e., the period consists of λ different values, with no repetitions). Lemma 16 gives sufficient conditions for the sequence to be uniform; and, in the present case, all of that lemma's preliminaries are satisfied, with $h = 0$ and $K_0 = \{x_j\}_{j=0}^{\infty}$, so that $|K_0| = \lambda$, as required. By (53) [see Lemmas 10 and 12], λ takes the form 2^u with $0 \le u \le M$; which translates, if we write $u = M - p$, into the first part, $\lambda = 2^{M-p}$, of the condition (68) of Lemma 16. Further, Lemmas 12 and 14 specify the corresponding values of p. Therefore, the second part of the condition (68), which becomes

$$(\forall j \ge 0) \quad 2^p \mid (x_j - \alpha), \tag{70}$$

alone remains to be verified, with the help of Lemma 15. Given that the sequence is indeed uniform in J, it then follows from Definition 4 that the coarseness of the sequence is $Q/P = Q/\lambda = 2^M/2^{M-p} = 2^p$. An examination of Lemmas 12, 14, and 15 indicates that there are six cases to be considered. Necessary correspondences between cases are shown in Table 1, below.

TABLE 1

Case	*Lemma 12*	*Lemma 14*	*Lemma 15*
(I) $c \le s - 1$	(i) $p = c < M$	—	(a)
(II) $c = s$	(i) $p = c < M$	—	(b)
(III) $c \ge s + 1$ and $c \le s + d - 1$	(i) $p = c < M$	—	(c)
(IV) $c = s + d < g$	(ii) $p = \min\{M, g\}$	—	(c)
(V) $c \ge s + d + 1$	(iii) $p = s + d < M$	$p = \min\{M, s + d\}$	(c)
(VI) $c = s = \infty$	—	$p = M$	(c)

(I) If $c \le s - 1$, then we have Case (a) of Lemma 15: members x_{2j} of the sequence $[x_j]_{j=0}^{\infty}$ are *even* multiples of 2^c, and members x_{2j+1} are *odd* multiples of 2^c; so all x_j are multiples of 2^c. Thus, (70) holds, if we take $\alpha = 0$ and apply $p = c$ (from Lemma 12). The sequence is therefore uniform in J, with coarseness 2^c.

(II) If $c = s$, then we have Case (b) of Lemma 15; members x_{2j} of the sequence are *odd* multiples of 2^c, and members x_{2j+1} are *even* multiples of 2^c; so that, again, all x_j are multiples of 2^c; whence, as before, since $p = c$, the sequence is uniform in J, with coarseness 2^c.

In all remaining cases, we have Case (c) of Lemma 15: all the x_j are *odd* multiples of 2^s.

(III) If $s + 1 \le c \le s + d - 1$, then, once more, $p = c$. Now, every *odd* multiple of 2^s clearly equals 2^s plus *some* multiple of $2^{s+1} = 2^c$; thus, if we take $\alpha = 2^s$, (70) follows; so that the sequence is, once again, uniform in J, with coarseness 2^c. This completes the proof of Part (i) of our theorem.

(IV) If $c = s + d$ then, by Lemma 12, $p = \min\{M, g\}$, where g is defined by (39) and (40). By Lemma 6, with equation (5),

$$x_j = \langle x_0 + (a^j - 1)x_0 + S_j(a)b \,|\, Q\rangle = \langle x_0 + S_j(a)W \,|\, Q\rangle. \tag{71}$$

It follows from this that every $x_j - x_0$ is divisible by the g.c.f. of 2^g and 2^M, i.e., by 2^p. Taking $\alpha = x_0$, we see that (70) holds; whence the sequence $[x_j]_{j=0}^{\infty}$ is uniform in J, with coarseness $2^p = \min\{2^M, 2^g\}$. This proves Part (ii) of our theorem.

(V) If $c \ge s + d + 1$, then $p = \min\{M, s + d\}$, by both Lemmas 12 and 14. In Lemma 12, Case (iii), we have shown that $M - s - d \ge 2$; so that, if $p = M \le s + d$, then we must have $b = 0$, as treated in Lemma 14, and therefore $1 \le M - s \le d$ [since we are not in Case (VI) (i.e., $x_0 \ne 0$) $s < M$]. In this case, $\lambda = 1$, and therefore all the x_j are equal; whence we see that every $x_j - x_0 = 0$, which is divisible by 2^M. Thus, taking $\alpha = x_0$, we derive (70); whence our sequence is uniform in J, with coarseness $2^p = 2^M = \min\{2^M, 2^{s+d}\}$.

If $p = s + d < M$, on the other hand, then, by (52), $M - s > d \ge 2$. Let us write

$$x_j = 2^s X_j, \tag{72}$$

where every X_j is an *odd* number. Then, by (2),

$$X_{j+1} \equiv aX_j + 2^{-s} b \pmod{2^{M-s}}, \tag{73}$$

with $2^{-s}\,b$ divisible by 2^d [recall that, in the present case, $c \geq s + d + 1$, and that, by (40), b is divisible by 2^c]. Hence, by (51),

$$X_{j+1} \equiv X_j \pmod{2^d}; \tag{74}$$

so that all the X_j are not only *odd*, but congruent to the *same* odd number, modulo 2^d. This means that every x_j equals $x_0 = 2^s X_0$ plus a multiple of $2^{s+d} = 2^p$. Taking $\alpha = x_0$, we see that (70) holds, and therefore, again, the sequence $[x_j]_{j=0}^{\infty}$ is uniform in J, with coarseness $2^p = 2^{s+d} = \min\{2^M, 2^{s+d}\}$.

(VI) Finally, if $c = s = \infty$, then, as we have seen, every $x_j = 0$; whence $\lambda = 1$ and so $p = M$. By the same token, (70) holds for $\alpha = 0$; so that our sequence is indeed uniform in J, with coarseness Q. This completes the proof of our theorem.≫

Corollary 1. *The coarseness of the sequence* $[x_j]_{j=0}^{\infty}$, *defined as in Theorem 1, attains its minimum possible value, namely,* 1, *if and only if* $c = 0$.

≪It is clear from the definitions (39) and (40) underlying Theorem 1 that $s \geq 0$ and $c \geq 0$. In Case (i) of the theorem, the coarseness $2^c = 1$ only when $c = 0$; implying that $s + d - 1 \geq 0$ and thus in no way restricting the allowable values of s [since $s \geq 0$ anyway, and, by (52), $d \geq 2$]. In Case (ii), $g \geq c + 1 = s + d + 1$ and the coarseness is $\min\{2^M, 2^g\}$. Now, by (52), $g \geq d + 1 \geq 3$, since $s \geq 0$, and $5 \leq 2^d + 1 \leq a = (2r - 1)\,2^d + 1 < 2^M$, by (50) and since $a \in J$; whence

$$M \geq 3. \tag{75}$$

Thus, either way, the coarseness is at least 8. In Case (iii), similarly, by (75) and because $s \geq 0$ and $d \geq 2$, the coarseness $\min\{2^M, 2^{s+d}\}$ is at least 4. Thus, the absolutely best coarseness, 1, is attained when and only when $c = 0$ [in Case (i)].≫

Corollary 2. *Given the set* F *defined in* (28) *and the sequence* $[\xi_j]_{j=0}^{\infty}$, *defined by* (1) *and* (2), *with parameter* a *satisfying* (50) *and* (52); *the sequence is uniform in* F, *in the sense of Definition 4, and the coarseness of the sequence is given by the values in Cases* (i), (ii), *and* (iii) *of Theorem 1.*

≪Both sets, J and F, have Q members (points) and are respectively cyclically equally spaced, in $[0, Q)$ and $[0, 1)$. The sequence $[x_j]_{j=0}^{\infty}$ stands in the same relation to J as does $[\xi_j]_{j=0}^{\infty}$ to F, and the corresponding sets $K_0 = \{x_j\}_{j=0}^{\infty}$ and $K_1 = \{\xi_j\}_{j=0}^{\infty}$ both have just λ members. Thus, by Definition 4 and Theorem 1, the corollary follows.≫

We have now collected sufficient information, on the uniformity properties of linear-congruential pseudo-random sequences, to enable us to move on to the main purpose of our study; namely, the generation and analysis of *tree-structured* families of generators. We shall discover that the results, embodied, for the most part, in Theorem 1 and its corollaries, which tell us about the uniformity and coarseness of a single sequence, suffice to analyze the properties of independence and consonance between members of families of such sequences.

4. TREE-STRUCTURED FAMILIES OF GENERATORS

We now proceed to consider tree-like *branching* processes. We take particle-transport problems as important and typical paradigms. The model often used has steps representing the rectilinear (or, in the presence of effective force-fields, curved) particle flight across empty physical space (using a statistical Poisson distribution of path-length, determined by the 'mean free path' parameter); alternating with steps representing 'collision' events, terminating such free flights. Collision events include elastic or inelastic rebound-collisions and various nuclear reactions, which often generate new particles (of matter or radiation); these last lead to a branching of the particle histories. The creation of 'virtual particles' (used, for example, in the Monte Carlo 'particle-splitting' technique, and in obtaining Monte Carlo scores at small-aperture detectors) also leads to branching. Since each step in a particle-history (or random-walk) may typically require about 10 random numbers, we may expect our pseudo-random sequence to entail branching at every T-th term, where T is of the order of 10. While it is certainly feasible to allow branching at *every* random number, it is likely to be more economical to pick such a T and only allow branching at every T-th step of the random sequence. The price we pay is that T must be an over-estimate, so as to ensure that, at least, most of the time, T random numbers suffice to compute a random-walk step [if more are needed, in a particular step, then we must allocate an integer multiple of T random numbers to this step]; thus, quite a few random numbers will be wasted in the process.

Before we can move forward, we must consider the behavior of the sequence $[x_{T-1}, x_{2T-1}, x_{3T-1}, x_{4T-1}, \ldots] = [x_{jT-1}]_{j=1}^{\infty}$ corresponding to the branch-points of the process (x_{jT-1} is the current pseudo-random number last obtained, when T numbers have been generated and a branch may occur).

Lemma 18. *The behavior of the sequence* $[x_{jT-1}]_{j=1}^{\infty}$ *of branch-points is given by*

$$X_{j+1} = \langle A\, X_j + B \,|\, Q \rangle, \tag{76}$$

when we write

$$A = \langle a^T | Q \rangle, \quad B = \langle S_T(a) b \,|\, Q \rangle, \quad \text{and} \quad X_j = x_{jT-1}. \tag{77}$$

≪By Lemma 6, the relation (30) holds; so that, using (5), we see that, modulo Q,

$$\begin{aligned} x_{(j+1)T-1} &\equiv a^{(j+1)T-1}\, x_0 + S_{(j+1)T-1}(a)\, b \\ &\equiv a^{(j+1)T-1}\, x_0 + \left(a^{(j+1)T-2} + a^{(j+1)T-3} + \ldots + a^2 + a + 1\right) b \\ &\equiv a^T \left[a^{jT-1}\, x_0 + \left(a^{jT-2} + a^{jT-3} + \ldots + a^2 + a + 1\right) b\right] \\ &\qquad + \left(a^{T-1} + a^{T-2} + a^{T-3} + \ldots + a^2 + a + 1\right) b \\ &\equiv a^T \left[a^{jT-1}\, x_0 + S_{jT-1}(a)\, b\right] + S_T(a)\, b \\ &\equiv a^T\, x_{jT-1} + S_T(a)\, b. \end{aligned} \tag{78}$$

With the notations of (3) and (77), (78) takes the form (76).≫

The recurrence relation (76) is exactly of the same form as (4); so that all our earlier analysis applies here. By Corollary 1, we observe that *odd* values of B are preferable; and, clearly, by (77) with Lemma 3, B will be odd, if and only if both b and T are odd. It is easily seen, by (51), that

$$A = a^T \equiv 1 \pmod{2^d}, \tag{79}$$

that an equally-spaced subsequence $[x_{jT-1}]_{j=1}^{\infty}$ of $[x_j]_{j=0}^{\infty}$ may well be what we are really dealing with.

The recurrence relation (4), with parameters a, b, and x_0 (we take Q and M as fixed), generates a *linear-congruential* sequence $[x_j]_{j=0}^{\infty}$ of integers in J. It constitutes a pseudo-random *generator*, which we may denote by $\Phi = \$(a, b, x_0)$. Having analyzed the periodic behavior and uniformity of a single linear congruential sequence, we can now consider a pair of such sequences: (i) $[x_j]_{j=0}^{\infty}$, with generator $\Phi = \$(a, b, x_0)$, characterized by (4), and (ii) $[x^{\dagger}{}_j]_{j=0}^{\infty}$, with generator $\Phi^{\dagger} = \$(a^{\dagger}, b^{\dagger}, x^{\dagger}{}_0)$, say, characterized by

$$(\forall j \geq 0) \quad x^{\dagger}{}_{j+1} = \langle a^{\dagger}x^{\dagger}{}_j + b^{\dagger} \,|\, Q\rangle. \tag{80}$$

We may now define the difference-sequence $[\delta_j]_{j=0}^{\infty}$ as we did in (26), and observe at once that

$$(\forall j \geq 0) \quad \delta_{j+1} = \langle a\delta_j + (a - a^{\dagger})x^{\dagger}{}_j + (b - b^{\dagger}) \,|\, Q\rangle. \tag{81}$$

Further, by applying (71), we see that

$$\begin{aligned}\delta_n &= \langle \delta_0 + S_n(a)\,[(a-1)x_0 + b] - S_n(a^{\dagger})\,[(a^{\dagger}-1)x^{\dagger}{}_0 + b^{\dagger}] \,|\, Q\rangle \\ &= \langle \delta_0 + S_n(a)\,W - S_n(a^{\dagger})\,W^{\dagger} \,|\, Q\rangle,\end{aligned} \tag{82}$$

where $W^{\dagger}$ is the counterpart, for the generator $\Phi^{\dagger}$, of W, defined in (39). This formula is rather difficult to analyze for the period and uniformity of the difference-sequence; but a particular case proves to be more tractable. Suppose that we restrict our consideration to $a^{\dagger} = a$; then

$$(\forall j \geq 0) \quad \delta_{j+1} = \langle a\delta_j + (b - b^{\dagger}) \,|\, Q\rangle, \tag{83}$$

which is exactly similar to (4), except that b is replaced by $\beta = \langle b - b^{\dagger} \,|\, Q\rangle$. It follows that all the results obtained so far (up to and including Theorem 1 and its corollaries) for the sequence $[x_j]_{j=0}^{\infty}$ apply also to the sequence $[\delta_j]_{j=0}^{\infty}$. It is just another linear-congruential sequence, whose generator may be written as $\Delta = \$(a, \beta, \delta_0)$.

All this can now be generalized to a *family* of generators, which we may denote by $\Phi_\mu = \$(a_\mu, b_\mu, x_{\mu 0})$, with parameters a_μ, b_μ, and $x_{\mu 0}$, satisfying

$$(\forall j \geq 0) \quad x_{\mu(j+1)} = \langle a_\mu x_{\mu j} + b_\mu | Q \rangle. \tag{84}$$

We restrict our consideration, by taking $(\forall \mu)\ a_\mu = a$, and write

$$\beta_{\mu\nu} \equiv \langle b_\mu - b_\nu | Q \rangle \quad \text{and} \quad \delta_{\mu\nu j} \equiv \langle x_{\mu j} - x_{\nu j} | Q \rangle. \tag{85}$$

Then $$(\forall j \geq 0) \quad \delta_{\mu\nu(j+1)} = \langle a\, \delta_{\mu\nu j} + \beta_{\mu\nu} | Q \rangle. \tag{86}$$

It is reasonable to minimize the coarseness of each individual sequence; and, by Corollary 1, the absolute minimum, 1, is attainable when and only when every $c_\mu = 0$, i.e., every b_μ is *odd*. The values of the parameters $x_{\mu 0}$ and a, subject only to (50) and (52), are arbitrary. This means that we have at our disposal fully half of *all* possible linear-congruential sequences; altogether 2^{M-1} sequences, for each choice of x_0, when a is fixed. However, this does entail that every $\beta_{\mu\nu}$ will now be *even*. (There is *no* choice of the b_μ which will permit us to get all odd $\beta_{\mu\nu}$.)

Now let us consider the kind of *branching random walk* for which the present study is intended to provide effective pseudo-random generators. In Figure 1, we see the first five levels of a *binary tree* with the nodes numbered in a simple, systematic manner. The caption explains the system. From any odd-numbered node, say $N_\mu = 2\mu + 1$, $(\mu = 0, 1, 2, \ldots)$, we define a *random walk*, or sequence of nodes,

$$\Gamma_\mu = [N_\mu \to 2N_\mu \to 4N_\mu \to \ldots \to 2^m N_\mu \to \ldots], \tag{87}$$

obtained by taking the left-slanting branch at every node (i.e., going from parent to left-child, every time), which will correspond, for example, in the case of a particle-transport problem, to the history of a single particle.

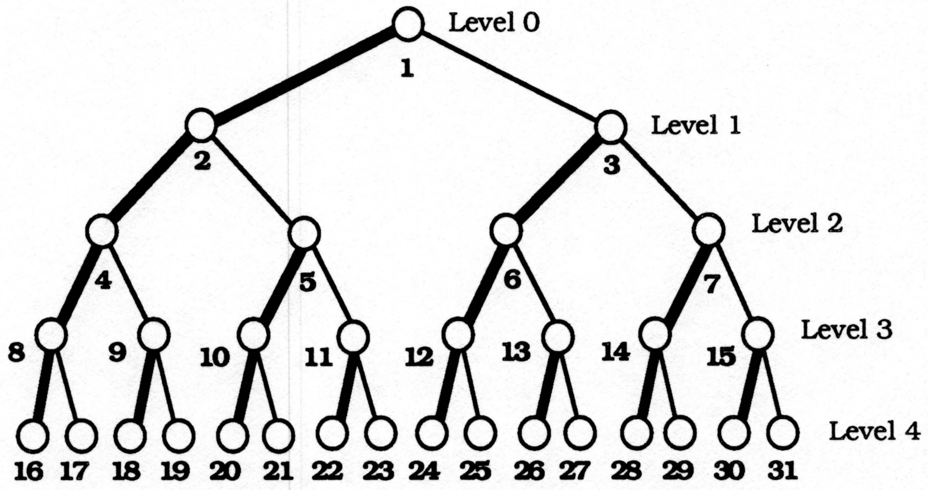


Figure 1.

Binary Tree Structure.

Level k has 2^k nodes, numbered $2^k, 2^k+1, 2^k+2, \ldots, 2^{k+1}-1$. Children of node number n are nodes numbers $2n$ (on left) and $2n+1$ (on right). *Left branches* are shown thicker; they denote continuing random walks, generated by single linear-congruential generators.

Associated with the walk Γ_μ, there will be, at each node, an array or other data-structure, giving the properties of the corresponding event, e.g., of a collision in the particle-history. The statistical samples occurring at every node of the random walk will be computed using pseudo-random numbers coming from a single generator of type $\Phi_\mu = \mathfrak{S}(a, b_\mu, x_{\mu0})$, satisfying (84), with parameters $x_{\mu0}$, b_μ, and $a_\mu = a$, satisfying (50) and (52). When an additional particle is generated at node number v, this will correspond to taking a right-slanting branch, to the child-node numbered $N_v = 2v + 1$, where a new pseudo-random generator $\Phi_v = \mathfrak{S}(a, b_v, x_{v0})$, with parameters a, b_v, and x_{v0}, initiates a concurrent history, Γ_v.

Since it is typical that branching does not actually occur at every node (and, indeed, since, as has been explained in §1, it would be totally impossible, in practice, to perform the computations needed if every branch did occur), it is of great practical utility, that the generator Φ_v, needed on branching at node number v, should be identified by appeal only to the index v, or, at worst, to a small number of parameters computed and stored at the node v.

Suppose that

$$2^m \Uparrow \nu, \quad \text{so that} \quad \nu = 2^m(2\mu + 1), \tag{88}$$

since such is the form of all nodes on the random walk Γ_μ. Then we may associate, with the node ν, a *record*,

$$\mathbb{R}_\nu = (2^m N_\mu, b_\mu, x_{\mu m}), \tag{89}$$

consisting of (i) the current node number, $\nu = 2^m N_\mu$ (whence both m and μ can be uniquely determined); (ii) the value of the parameter b_μ of the current generator Φ_μ (remember that the parameter a is supposed to be common to all random generators in this scheme, or *family*); and (iii) the current random number $x_{\mu m}$. We now begin the new random walk Γ_ν, with new parameters, b_ν and $x_{\nu 0}$, and the particular scheme that we adopt is specified when we define the functional relationships between these new parameters and the record:

$$\left.\begin{aligned} b_\nu &= \mathcal{B}(2^m N_\mu, b_\mu, x_{\mu m}) = \mathcal{B}(\mathbb{R}_\nu) \\ x_{\nu 0} &= \mathcal{X}(2^m N_\mu, b_\mu, x_{\mu m}) = \mathcal{X}(\mathbb{R}_\nu) \end{aligned}\right\}. \tag{90}$$

This can also be formalized by putting:

$$\mathbb{R}_{N_\nu} = (N_\nu, b_\nu, x_{\nu 0}) = \mathcal{M}(2^m N_\mu, b_\mu, x_{\mu m}) = \mathcal{M}(\mathbb{R}_\nu). \tag{91}$$

The mapping $\mathcal{M}$ (or, more explicitly, the functions $\mathcal{B}$ and $\mathcal{X}$ comprising it) determine the particular algorithm we choose.

Consider, first, the relationship between two segments of the *same* random sequence $[x_j]_{j=0}^{\infty}$. If the separation between them is H, say, then we may take

$$x^\dagger_j = x_{H+j} \quad \text{and} \quad \Phi^\dagger = \mathcal{S}(a, b, x_H); \tag{92}$$

so that, by (83) with $b^\dagger = b$, we see that the sequence $[\delta_j]_{j=0}^{\infty}$ defined in (23) has generator $\Delta = \mathcal{S}(a, 0, x_0 - x_H)$.

Lemma 19. *Given the set J defined in* (28), *the sequence $[x_j]_{j=0}^{\infty}$ generated by* (4) *with parameter a satisfying* (50) *and* (52), *and given any positive integer H; the sequences $[x_j]_{j=0}^{\infty}$ and $[x_j]_{j=H}^{\infty}$ (which differ only by the positional offset H) are independent with respect to J, in the sense of Definition 5. When c, s, d, and g are defined by* (39) *and* (40), *and*

$$2^{\kappa} \Uparrow H, \tag{93}$$

the two sequences have consonance $\min\{2^M, 2^{\kappa+g+d}\}$.

≪Applying Theorem 1 to the generator Δ, we see at once that the sequence $[\delta_j]_{j=0}^{\infty}$ is *uniform* in J; and therefore, by Definition 5, we immediately conclude that the two sequences $[x_j]_{j=0}^{\infty}$ and $[x_j]_{j=H}^{\infty}$ are *independent* with respect to J. Since, by (13), $2^{\infty} \Uparrow 0$, and the second parameter of the generator is 0, the corresponding 'power of divisibility' of that parameter is ∞; so that, by Theorem 1, the *coarseness* of $[\delta_j]_{j=0}^{\infty}$ is $G = \min\{2^M, 2^{\sigma+d}\}$, where d is defined by (50) and (52), and σ is defined by $2^{\sigma} \Uparrow (x_0 - x_H)$. Hence, by Definition 5, the *consonance* of $[x_j]_{j=0}^{\infty}$ and $[x_j]_{j=H}^{\infty}$ is G.

Now, by (15) with (48), $2^{\kappa} \Uparrow S_H(a)$; by (39) and (40), $2^g \Uparrow W$; and, finally, by (71), $x_0 - x_H = \langle -S_H(a)W \,|\, Q \rangle$. Therefore, we see that $\sigma = \kappa + g$; and so $G = \min\{2^M, 2^{\kappa+g+d}\}$.≫

Just as we stipulated, first, that the parameter a be odd, and then that it should satisfy (50) and (52), so as to minimize the coarseness [that is, maximize the uniformity] of the individual sequences; so we now seek to minimize the consonance G of a pair of sequences. To this end, we may minimize d, subject to (52), by

$$d = 2, \tag{94}$$

so that, by (50), this is equivalent to $a = (2r' - 1)4 + 1 = 8(r' - 1) + 5$, or

$$a \equiv 5 \pmod 8. \tag{95}$$

Corollary 3. *Under the conditions of Lemma 19, if we impose the additional constraint* (95), *and choose the parameter b to be odd; then the consonance of the two sequences becomes* $\min\{2^M, 2^{\kappa+2}\}$.

≪Since (95) is equivalent to (94), direct substitution of 2 for d in the formula given by the lemma yields $G = \min\{2^M, 2^{\kappa+g+2}\}$. Since b is made odd, so that $c = 0$, we have $c < s + 2 = s + d$, by (94); whence Case (i) of Lemma 11 yields that $g = c = 0$. The corollary now follows immediately.≫

Warnock (see WAR 83) proposes, in our notation, that all 'left-slanting' generators Φ_μ should share common parameters a and b. Thus, his function of type $\mathcal{B}$, say $\mathcal{B}_{\mathbb{W}}$, is the projection of the second argument, unchanged:

$$b_\nu = \mathcal{B}_{\mathbb{W}}(2^m N_\mu, b_\mu, x_{\mu m}) = b_\mu = b. \tag{96}$$

His function of type $\mathcal{X}$, say $\mathcal{X}_{\mathbb{W}}$, applies a step of type (4), with its own independent parameters, $a^\dagger$ and $b^\dagger$, say, to go from the last random number $x_{\mu m}$ to the first one of the new sequence:

$$x_{\nu 0} = \mathcal{X}_{\mathbb{W}}(2^m N_\mu, b_\mu, x_{\mu m}) = \langle a^\dagger x_{\mu m} + b^\dagger \,|\, Q \rangle. \tag{97}$$

Since all the left-slanting generators in Warnock's scheme have the same parameters a and b, if we select a satisfying (95) [and so (50) and (52)] and b odd [$c = 0$]; then, by Corollary 1, all the resulting sequences will be uniform in J, with minimal coarseness 1. Thus, the period has length Q; that is, *every* value in J occurs in *each* such sequence. Consequently, *all* the possible sequences are just positional offsets of each other; and therefore, by Corollary 3, if a pair of such sequences has positional offset H satisfying (93), it will exhibit the consonance $\min\{2^M, 2^{\kappa+2}\}$.

Unfortunately, it is impossible to improve the situation optimally by making all $\kappa = 0$. This is because of the structure of the integers, with respect to divisibility by powers of 2. The sequence of κ-values takes the form shown in Figure 2.

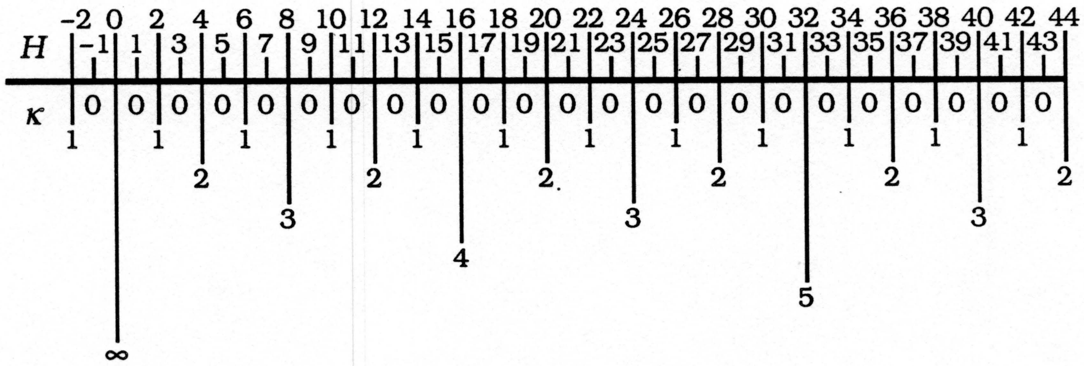


Figure 2.

Divisibility of the Integer Sequence by Powers of 2.

For a segment of the sequence of integers H, the corresponding values of κ, such that $2^{\kappa} \Uparrow H$, are tabulated, with each row having a single value of κ. (The structure is reminiscent of the "Sieve of Eratosthenes" used to find prime numbers.) Observe that, to be as far as possible from $\kappa \geq \kappa_0$, one has to be close to a value of H with $\kappa = \kappa_0 - 1$. For example, with $\kappa_0 = 4$ and H lying between 16 and 32, it is best to choose $H = 21$ or 23, somewhat closer to the lesser of the extreme κ-values, 4 and 5. The two extreme values will, of course, never be equal.

We can, at best, hope that sequences corresponding to immediately adjacent events will have low values of κ. The generator Φ_μ begins at the node numbered $N_\mu = 2\mu + 1$ and passes, in a left-slanting direction, through the node numbered $\nu = 2^m N_\mu$; from which a branch goes to its right-child node, numbered $N_\nu = 2\nu + 1$. The generator Φ_ν begins there, and its sequence $[x_{\nu j}]_{j=0}^{\infty}$, in Warnock's scheme, is, as we have seen, just a positionally-offset copy of the sequence $[x_{\mu j}]_{j=0}^{\infty}$ of generator Φ_μ. We certainly want the two histories, beginning at the left and right children of node ν, to have the least possible consonance; so we would like the offset, between $x_{\mu(m+1)} = \langle a x_{\mu m} + b \,|\, Q \rangle$ and $x_{\nu 0} = \langle a^{\dagger} x_{\mu m} + b^{\dagger} \,|\, Q \rangle$, to be *odd*; this is clearly equivalent to having the offset between $x_{\mu m}$ and $x_{\nu 0}$ *even*. By an obvious extension of (30), we require that there be an integer n, such that

$$a^\dagger x_{\mu m} + b^\dagger \equiv a^{2n} x_{\mu m} + S_{2n}(a)\, b \pmod{Q}, \tag{98}$$

or, by (3), $a^\dagger = \langle a^{2n} | Q \rangle$ and $b^\dagger = \langle S_{2n}(a) b | Q \rangle$. (99)

Since (99) is independent of $x_{\mu m}$, we see that a single transformation of the form (97) will work in all cases.

However, our advantage is somewhat brittle. It is physically desirable that the history, generated by Φ_ν and beginning at node N_ν, should also have small consonance with histories beginning at nodes *neighboring* node 2ν in the chain generated by Φ_μ; i.e., corresponding to sequential (positional) offsets close to, but different from, $2n - 1$ (with n the same as that in (99)). These offsets will be even, in about half the cases, and examination of the sequence of κ-values in Figure 2 indicates that, if we wish to avoid $\kappa \geq \kappa_0$, say, we shall certainly have a near-neighbor with $\kappa = \kappa_0 - 1$. As for more distant histories, across the tree, these will have a variety of offsets, but this is hardly to be avoided. After all, we are looking at a universe of only 2^M distinct sequences, to fill 2^{k-1} histories [left-slanting branches], in a binary tree of height k, with $2^k - 1$ branch-points and 2^{k+1} nodes. Since a typical value of this k is perhaps $10^2 - 10^4$, while a typical value of M is about 48, the capacity of the scheme is evidently overloaded.

The plausible argument, that computational runs requiring some $10^5 - 10^8$ random numbers should be pretty unrelated, when taken from random segments of a pseudo-random sequence with period of the order of $2^{48} \approx 3 \times 10^{14}$, at least three million times longer, turns out not to be entirely valid. However, in mitigation, it should be pointed out that, until now, no rigorous analysis of the algorithm was available.

If one nevertheless decides to adopt this scheme, the indication is strong that one should adopt a satisfying (95), b odd, and $a^\dagger$ and $b^\dagger$ satisfying (99), with values of n such as 11, 12, 22, 23, or 24 (for H = 21, 23, 43, 45, or 47, respectively).

We now leave Warnock's algorithm, and return to our consideration of the more general relationship between two sequences, $[x_j]_{j=0}^{\infty}$ and $[x^\dagger_j]_{j=0}^{\infty}$, whose respective generators are $\Phi = \mathfrak{S}(a, b, x_0)$ and $\Phi^\dagger = \mathfrak{S}(a, b^\dagger, x^\dagger_0)$, and whose difference $[\delta_j]_{j=0}^{\infty}$, with $\delta_j = \langle x_j - x^\dagger_j | Q \rangle$, satisfies (83), with $\beta = \langle b - b^\dagger | Q \rangle \neq 0$. We shall assume that both b and $b^{\dagger}$ are *odd* integers, and that a satisfies (95). Following (39) and (40), let

$$\Omega = \langle \delta_1 - \delta_0 | Q \rangle = \langle (a-1)\delta_0 + \beta | Q \rangle, \tag{100}$$

and $$2^{\gamma} \Uparrow \beta, \quad 2^{\sigma} \Uparrow \delta_0, \quad \text{and} \quad 2^{\tau} \Uparrow \Omega. \tag{101}$$

By Theorem 1 and Definition 5, we now obtain:

Theorem 2. *Given the set J defined in (28) and the parameters a, with (95), and b and $b^{\dagger}$, both odd, with $\langle b - b^{\dagger} | Q \rangle \neq 0$; the sequences $[x_j]_{j=0}^{\infty}$ and $[x^{\dagger}_j]_{j=0}^{\infty}$, with generators $\Phi = \$(a, b, x_0)$ and $\Phi^{\dagger} = \$(a, b^{\dagger}, x^{\dagger}_0)$, respectively, are independent with respect to J. If Ω, γ, σ, and τ are defined as in (100) and (101), then the consonance of the sequences is given by*

(i) 2^{γ}, *if* $\gamma \leq \sigma + 1$;

(ii) $\min\{2^M, 2^{\tau}\}$ *with* $\tau > \gamma$, *if* $\gamma = \sigma + 2$;

(iii) $2^{\sigma+2}$, *if* $\gamma \geq \sigma + 3$.

<<By (83) and (95), which implies (94), we have the conditions of Theorem 1, with $d = 2$, and Ω, γ, σ, and τ respectively taking the places of W, c, s, and g. By our assumptions,

$$1 \leq \gamma < M; \tag{102}$$

whence the case of $\gamma = \sigma = \infty$ is impossible, and $M > \gamma > \sigma + 2$ [compare Lemma 12]. Theorem 2 follows immediately.>>

Corollary 4. *Under the conditions of Theorem 2, if $\delta_0 = x_0 - x^{\dagger}_0 = 0$, then the consonance of the sequences is 2^{γ}.*

<<If $\delta_0 = 0$, then, by (101) with (13), $\sigma = \infty$. Thus, by (102), we are in Case (i) of Theorem2; and the corollary is immediate.>>

It is instructive to note the dependence on σ, for any given γ, of the consonance determined by Theorem 2. This is sketched in Figure 3.

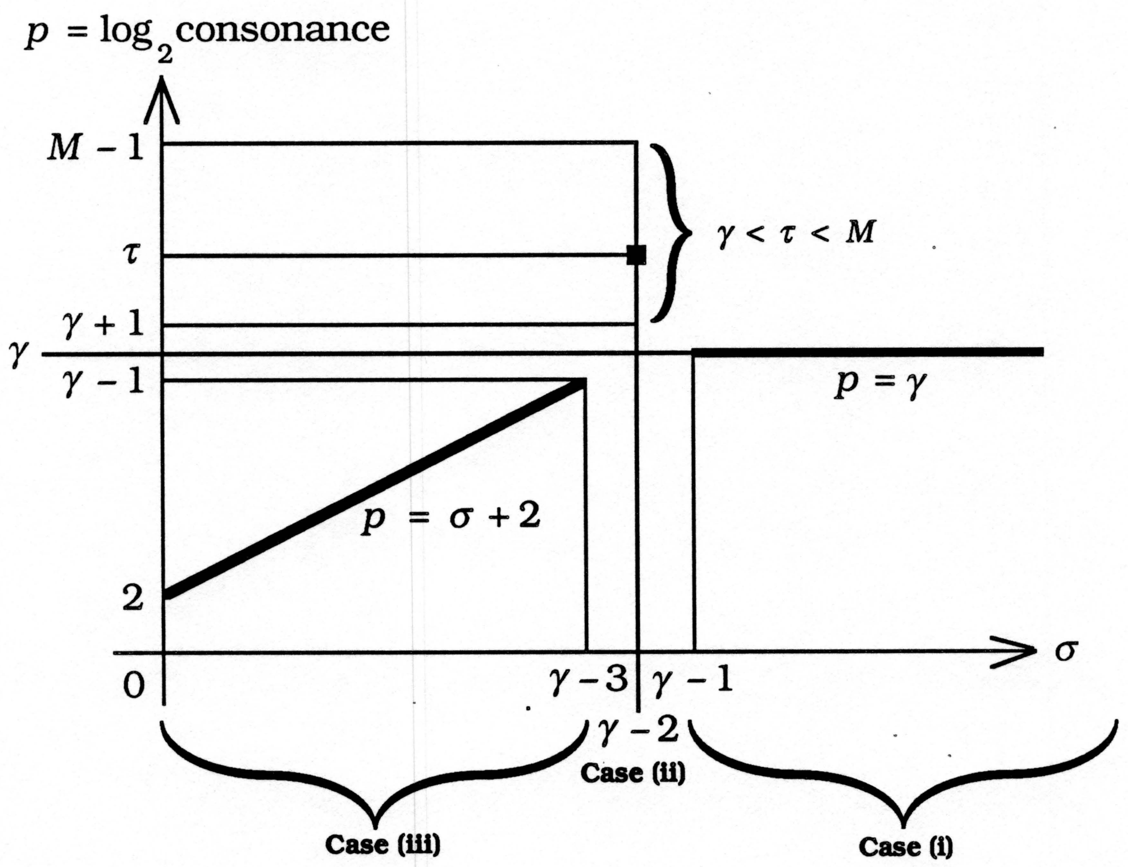


Figure 3.

Consonance as a Function of σ, for Fixed γ.

The logarithm to base 2 of the consonance of two sequences, $[x_j]_{j=0}^{\infty}$ and $[x^\dagger{}_j]_{j=0}^{\infty}$, with generators $\Phi = \mathfrak{S}(a, b, x_0)$ and $\Phi^\dagger = \mathfrak{S}(a, b^\dagger, x^\dagger{}_0)$, respectively, is plotted against σ (where $2^\sigma \Uparrow \delta_0 = \langle x_0 - x^\dagger{}_0 | Q \rangle$), for given γ (where $2^\gamma \Uparrow \beta = \langle b - b^\dagger | Q \rangle \neq 0$). Cases indicated are those used for classification of results in Theorem 2.

In the general situation described by (84) – (91), in which a family of generators Φ_μ, with a single common parameter a, satisfying (95), and with all their individual parameters b_μ odd, is matched to the odd-numbered nodes N_μ and left-slanting random walks Γ_μ of a binary tree; we seek, as ever, to minimize the consonance between the sequences generated by the different Φ_μ, and especially between those sequences close to each other in the tree. The closest physical relationship will be between the sequences originating at the two child-nodes of any given node; for example, if the parent node is numbered v, the sequences are $[x_{\mu j}]_{j=m+1}^{\infty} = [x_{\mu(i+m+1)}]_{i=0}^{\infty}$, beginning at node $2v$, and $[x_{vj}]_{j=0}^{\infty}$, begining at node $2v + 1 = N_v$. If, in Theorem 2, we take

$$x_j = x_{\mu(m+j+1)}, \quad b = b_\mu, \quad x^\dagger_j = x_{vj}, \quad b^\dagger = b_v; \tag{103}$$

so that $$\beta = <b - b^\dagger | \mathcal{Q}> = <b_\mu - b_v | \mathcal{Q}> = \beta_{\mu v} \neq 0 \tag{104}$$

and $$\delta_j = <(x_j - x^\dagger_j | \mathcal{Q}) = <x_{\mu(m+j+1)} - x_{vj} | \mathcal{Q}>; \tag{105}$$

then the conditions of the theorem are satisfied and the conclusions of the theorem hold, for all indices v and functions $\mathcal{B}$ and $\mathcal{X}$.

Let us write

$$\Delta_{\mu v h} = <x_{\mu(m+h)} - x_{v0} | \mathcal{Q}> \tag{106}$$

(the notation makes sense, since, by (88), μ and v determine m). Then we note, by (105), that, in particular,

$$\delta_0 = \Delta_{\mu v 1}. \tag{107}$$

We shall denote the *logarithmic consonance* (i.e., the logarithm to base 2 of the consonance) of our two sequences by $p_{\mu v 1}$.

Corollary 5. *If $\Delta_{\mu v 1}$ is odd, then the logarithmic consonance of the sequences $[x_{\mu(i+m+1)}]_{i=0}^{\infty}$ and $[x_{vj}]_{j=0}^{\infty}$ is given by*

(i) $p_{\mu v 1} = 1$, *if* $\gamma = 1$;

(ii) $2 < p_{\mu v 1} \leq M$, *if* $\gamma = 2$;

(iii) $p_{\mu v 1} = 2$, *if* $\gamma \geq 3$;

where γ is defined by (101).

≪≪If $\Delta_{\mu\nu 1}$ is odd, then $\sigma = 0$, and, by (102), the three cases of Theorem 2 become those listed above; whence the values of $p_{\mu\nu 1}$ are as stated.≫≫

This result suggests that we should take $\Delta_{\mu\nu 1}$ odd and

$$\gamma \geq 3; \tag{108}$$

the latter condition is easily satisfied, e.g., by taking every $b_\mu \equiv 1 \pmod 8$. Note that, if $\Delta_{\mu\nu 1}$ is even and not zero, it is much harder to confine the values of $p_{\mu\nu 1}$.

Now consider, as we did for Warnock's scheme, what happens if we compare the sequences $[x_{\mu(i+m+H)}]_{i=0}^{\infty}$ and $[x_{\nu j}]_{j=0}^{\infty}$, with a positional offset in one sequence. Then β (and therefore also γ) is unaffected; but δ_0 (and therefore also Ω, σ, and τ) will depend on H, since now

$$\delta_0 = \langle x_{\mu(m+H)} - x_{\nu 0} \,|\, Q \rangle = \Delta_{\mu\nu H}. \tag{109}$$

Theorem 2 will clearly still apply. For different values of H, the logarithmic consonance of our two sequences, which is denoted by $p_{\mu\nu H}$, will depend on σ as shown in Figure 3, with an isolated maximum-value 'spike' when $\sigma = \gamma - 2$ and $\gamma < p_{\mu\nu H} < M$. The dependence of σ on H will be scattered, rather as in Figure 2; and, as for Warnock's algorithm, this creates a problem.

By (71) and (106), we see that

$$\Delta_{\mu\nu h} = \langle \Delta_{\mu\nu 0} + S_h(a) W_{\mu m} \,|\, Q \rangle; \tag{110}$$

where
$$W_{\mu m} = \langle (a-1)x_{\mu m} + b_\mu \,|\, Q \rangle \tag{111}$$

is analogous to W in (39). Thus, by (110) with $h = 1$,

$$\Delta_{\mu\nu 1} = \langle \Delta_{\mu\nu 0} + S_1(a) W_{\mu m} \,|\, Q \rangle. \tag{112}$$

Since, by (111) with (49) and because all the b_μ are going to be odd in our present discussion, $W_{\mu m}$ is odd; and since, also, $S_1(a) = 1$; we see that

$$\Delta_{\mu\nu 1} \text{ is odd} \quad \text{if and only if} \quad \Delta_{\mu\nu 0} \text{ is even.} \tag{113}$$

Further, by (93), (48), and Lemma 3, we have that $2^{\kappa} \Uparrow S_H(a)W_{\mu m}$. Now, let us write

$$2^{\sigma_{\mu\nu h}} \Uparrow \Delta_{\mu\nu h}. \tag{114}$$

Then, our usual line of argument, applied to (110) with $h = H$, yields that:

$$\left.\begin{array}{llll} \text{(a)} & \sigma_{\mu\nu H} = \sigma_{\mu\nu 0} & \text{if } \sigma_{\mu\nu 0} < \kappa; \\ \text{(b)} & \sigma_{\mu\nu H} > \kappa & \text{if } \sigma_{\mu\nu 0} = \kappa; \\ \text{(c)} & \sigma_{\mu\nu H} = \kappa & \text{if } \sigma_{\mu\nu 0} > \kappa. \end{array}\right\} \tag{115}$$

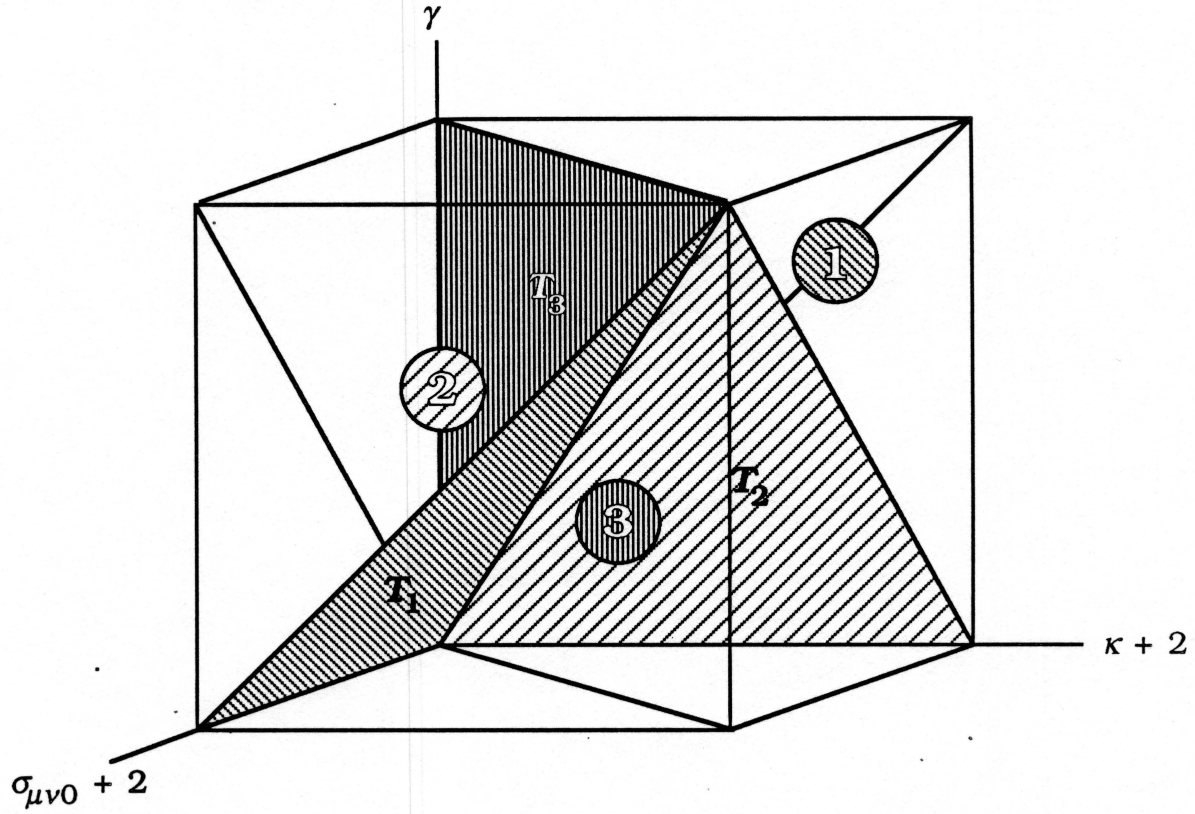
Using Figure 3 as a guide, it is not too hard to derive, from (115) and Theorem 2 with $\sigma = \sigma_{\mu\nu H}$, the relationships shown in Figure 4. The consequences are shown in Table 2, below. (D denotes the diameter of the cube in which the triangles T_1, T_2, and T_3 meet.)

TABLE 2

Region	*Case in* (115)	*Case in Theorem 2*	$p_{\mu\nu H}$
'1'	(a)	(iii)	$\sigma_{\mu\nu 0} + 2$
'2'	(c)	(iii)	$\kappa + 2$
'3'	(a), (c)	(i)	γ
T_1	(c)	(ii)	$p > \gamma$
T_2	(a)	(ii)	$p > \gamma$
T_3	(b)	*indeterminate*	?
D	(b)	(i)	γ

Theorem 3. *Define* γ *by* (101), κ *by* (93), $\Delta_{\mu\nu h}$ *by* (106), *and* $\sigma_{\mu\nu h}$ *by* (114); *and let* $\Delta_{\mu\nu 1}$ *be odd (i.e.,* $\sigma_{\mu\nu 1} = 0$). *Then* $\Delta_{\mu\nu 0}$ *is even; and, if a clear minimum occurs (i.e., one of* $\sigma_{\mu\nu 0} + 2$, $\kappa + 2$, *and* γ, *is strictly smaller than the other two), then*

$$p_{\mu\nu H} = \min\{\sigma_{\mu\nu 0} + 2,\ \kappa + 2,\ \gamma\}. \tag{116}$$



T_1	▧	$\gamma = \kappa + 2 < \sigma_{\mu\nu0} + 2$	1	$\kappa + 2 > \sigma_{\mu\nu0} + 2 < \gamma$
T_2	▨	$\gamma = \sigma_{\mu\nu0} + 2 < \kappa + 2$	2	$\sigma_{\mu\nu0} + 2 > \kappa + 2 < \gamma$
T_3	▥	$\kappa + 2 = \sigma_{\mu\nu0} + 2 < \gamma$	3	$\kappa + 2 > \gamma < \sigma_{\mu\nu0} + 2$

Figure 4.

Logarithmic Consonance as a Function of $\sigma_{\mu\nu0}$, κ, and γ.

The numbered solid regions, '1', '2', and '3', are pyramidal portions of the cube, bounded by faces of the cube and by triangular plane regions shaded otherwise than their own shading-key, and lying opposite to the similarly-shaded triangles: Region '1' is bounded by T_2 and T_3, and lies opposite to T_1; Region '2' is bounded by T_3 and T_1, and lies opposite to T_2; and Region '3' is bounded by T_1 and T_2, and lies opposite to T_3. The resulting values of the logarithmic consonance $p_{\mu\nu H}$ are given in Table 2.

≪The first conclusion of the theorem, that $\Delta_{\mu\nu 0}$ is even, follows at once from (113), since $\Delta_{\mu\nu 1}$ is postulated to be odd. The result (116) follows immediately from Table 2 and the information in Figure 4, where we see that a "clear minimum" occurs precisely when we are in the interior of one of the regions '1', '2', or '3'.≫

When $H = 1$ (so that $\kappa = 0$), Theorem 3 yields that $\sigma_{\mu\nu 0} > \kappa$; whence $p_{\mu\nu 1} = \gamma$ if $\gamma < \kappa + 2 = 2$, and $p_{\mu\nu 1} = \kappa + 2 = 2$ if $\gamma < \kappa + 2$. Thus we recover Cases (i) and (iii) of Corollary 5.

TABLE 3

$\sigma_{\mu\nu 0} + 2$ *and* γ	κ	*Region*	$p_{\mu\nu H}$
$\sigma_{\mu\nu 0} + 2 < \gamma$	$\kappa + 2 < \sigma_{\mu\nu 0} + 2$	'2'	$\kappa + 2$
	$\kappa + 2 = \sigma_{\mu\nu 0} + 2$	T_3	?
	$\sigma_{\mu\nu 0} + 2 < \kappa + 2 < \gamma$	'1'	$\sigma_{\mu\nu 0} + 2$
	$\kappa + 2 = \gamma$	'1'	$\sigma_{\mu\nu 0} + 2$
	$\kappa + 2 > \gamma$	'1'	$\sigma_{\mu\nu 0} + 2$
$\sigma_{\mu\nu 0} + 2 = \gamma$	$\kappa + 2 < \gamma$	'2'	$\kappa + 2$
	$\kappa + 2 = \gamma$	D	γ
	$\kappa + 2 > \gamma$	T_2	$p > \gamma$
$\sigma_{\mu\nu 0} + 2 > \gamma$	$\kappa + 2 < \gamma$	'2'	$\kappa + 2$
	$\kappa + 2 = \gamma$	T_1	$p > \gamma$
	$\gamma < \kappa + 2 < \sigma_{\mu\nu 0} + 2$	'3'	γ
	$\kappa + 2 = \sigma_{\mu\nu 0} + 2$	'3'	γ
	$\kappa + 2 > \sigma_{\mu\nu 0} + 2$	'3'	γ

We observe that we can, to some extent, control the values, first, of γ, and then, of $\sigma_{\mu\nu 0}$; but we have no control over κ, since H is arbitrary. Table 3 (based on Figure 4 and Table 2) shows the dependence of the logarithmic consonance $p_{\mu\nu H}$ on all three parameters, for all their possible relative magnitudes. The 'bad' cases arise in the triangular plane regions T_1, T_2, and T_3; and, clearly, the least damage is done, if trouble arises for as few values of H as possible. Now, half the values of H are odd ($\kappa = 0$), a quarter are divisible by 2 but not by 4 ($\kappa = 1$), an eighth are divisible by 4 but not by 8 ($\kappa = 2$), and so on; with the value κ accounting for $2^{-\kappa-1}$ of all values of H; and with all values greater than κ accounting for the *same* fraction. Thus, if $\sigma_{\mu\nu 0} + 2 < \gamma$, a fraction $2^{-\sigma_{\mu\nu 0}-1} > 2^{-\gamma+1}$ of the H-values (when $\kappa = \sigma_{\mu\nu 0}$) are bad; if $\sigma_{\mu\nu 0} + 2 = \gamma$, the fraction is $2^{-\gamma+1}$ (when $\kappa > \gamma - 2$); and if $\sigma_{\mu\nu 0} + 2 > \gamma$, the fraction is again $2^{-\gamma+1}$ (when $\kappa = \gamma - 2$). We therefore see that it is desirable to take, first, $\Delta_{\mu\nu 1}$ odd [$\sigma_{\mu\nu 1} = 0$] and $\gamma \geq 3$ [as noted in (108)], and then

$$\sigma_{\mu\nu 0} \geq \gamma - 2. \tag{117}$$

Since the fraction of 'bad' values of H is then $2^{-\gamma+1}$, it is probably wise to exceed the criterion in (108) somewhat, to make this fraction smaller. A reasonable condition might be

$$\gamma \geq 8, \tag{118}$$

yielding a fraction 2^{-7} (less than 1%) of bad values of H. This is achieved, for example, by taking every $b_\mu \equiv 1 \pmod{256}$. As the lower bound on γ increases, (a) the 'good' values of H yield somewhat less desirable consonances, and (b) the numbers of available distinct values of b_μ and of $x_{\mu 0}$ decrease correspondingly; so there is a trade-off here, as in so many such situations, and an 'engineering solution' (i.e., a compromise) is indicated.

Note the special solution, when

$$\Delta_{\mu\nu 0} = 0; \quad \text{i.e.,} \quad \sigma_{\mu\nu 0} = \infty. \tag{119}$$

Then, as is pointed out in Corollary 4, we have $p_{\mu\nu H} = \gamma$ for *all* H; but at the cost of no choice of $\sigma_{\mu\nu 0}$.

We must not overemphasize the importance of the consonances of positionally offset pairs of sequences. The unfortunate results can, to some extent, be minimized by suitably avoiding unfavorable offsets;

but only at the cost of wasting some random numbers which might otherwise be put to use. Again, suitable compromises are indicated.

Finally, we consider the consonance between sequences *not* arising from the same branch-point. Of course, Theorem 2 still applies. Let Φ_μ and Φ_ν begin at nodes numbered $N_\mu = 2\mu + 1$ and $N_\nu = 2\nu + 1$, respectively, but now *without* the relation (88). Then N_μ will be at Level m, if and only if

$$2^{m-1} \le \mu \le 2^m - 1; \quad \text{i.e.,} \quad m = \lceil \log_2(\mu + 1) \rceil, \tag{120}$$

and N_ν will be at Level n, determined similarly. Now, β will still be defined by (104), and γ by (101); but the appropriate δ_0 will be given by

$$k = \max\{m, n\}, \quad \delta_0 = x_{\mu(k-m)} - x_{\nu(k-n)}. \tag{121}$$

If $m \le n$, then $k = n$ and $\delta_0 = x_{\mu(n-m)} - x_{\nu 0}$. Whatever condition, such as (108) or (118), we apply to the b_μ will still hold here; but the conditions on the $x_{\mu 0}$ will no longer work for us here. Thus σ will be effectively out of our control; and so the consonance of the sequences will float freely, in accordance to Theorem 2 and Figure 3, with logarithmic consonance not greater than γ, except for the 'bad' cases, when $\sigma = \gamma - 2$. Again, this will tend to occur about $2^{-\gamma+1}$ of the time.

5. SPECIFIC PROCEDURES

We now have all the underlying machinery that we shall need, to select specific procedures, to generate tree-structured families of linear congruential pseudo-random generators, yielding sequences which are individually uniform, with minimal coarseness, and which are mutually independent, with acceptably low consonances.

To put things in perspective, we observe that, for a given fixed choice of the parameter a [which we have supposed to satisfy (95)], there are 2^{M-3} distinct possible values of b_μ satisfying (108) [and 2^{M-8} distinct possible values satisfying (118)], and altogether 2^M distinct possible values of $x_{\mu 0}$ [if we choose to make $\Delta_{\mu\nu 1}$ odd ($\sigma_{\mu\nu 1} = 0$), then this number is halved]. The possible distinct pseudo-random sequences are thus in any case no more than 2^{2M-3} in number; and probably less, in any given procedure (e.g., in Warnock's algorithm, there are only at most 2^M distinct sequences). Since the sequences begin at all the odd-numbered nodes (numbered $N_\mu = 2\mu + 1$) of a binary tree [see Figure 1], it is clear that *there must be at least one repetition in the first* $2M - 1$ *levels*, and thereafter, more and more frequently within each level (since Level $2M - 2$ alone has 2^{2M-2} nodes, and so 2^{2M-3} odd-numbered nodes; and each level has twice as many nodes as its immediate predecessor). We thus cannot expect to avoid the recurrence of the same pseudo-random sequences at scattered points in our binary tree. (Even if we were to exploit every possible sequence of the form (4) in our tree, there would still have to be at least one repetition in the first $3M + 2$ levels.) In practice, it is extremely difficult to avoid the occurrence of repetitions somewhat more frequent than these extreme bounds. However, we must recall that the nodes of our binary tree correspond to batches of T consecutive pseudo-random numbers [see Lemma 18], one of which usually suffices to generate a single physical event; and these events will rarely lead to actual branching (or, as has been pointed out, the resulting computations would be enormously, impossibly, too laborious). Thus only a very sparse, random sample of the branches is actually exploited in any realistic calculation. This is what saves us, in practice. Nevertheless, any repetitions that do occur must be minimized with respect to quantity, and dispersed as far as possible in their distribution over the tree.

Perhaps the simplest hypothesis to adopt would be that

$$x_{\nu 0} = \mathcal{X}_{\mathbb{S}}(2^m N_\mu, b_\mu, x_{\mu m}) = x_{\mu m}. \qquad (122)$$

This is comparable in economy to Warnock's definition of $\mathcal{B}_{\mathbb{W}}$ in (96), and is equivalent, of course, by (106), to (119). It then follows, by (113) and Corollary 5, that, if (108) holds, the logarithmic consonance of the sequences is 2: a highly satisfactory result.

Now, all the parameters b_ν are postulated to be odd, with (108) holding; which we can ensure by choosing $c = 0, 1, 2,$ or 3, and then taking

$$(\forall \nu \geq 0) \quad b_\nu \equiv 2c + 1 \pmod 8. \qquad (123)$$

Since every starting node of a new generator Φ_ν has an odd number, $N_\nu = 2\nu + 1$, with all the ν different, of course; it is natural to adopt the simple relation

$$b_\nu = \mathcal{B}_{\mathbb{N},3}(2^m N_\mu, b_\mu, x_{\mu m}) = \langle 8\nu + 2c + 1 \,|\, \mathcal{Q} \rangle. \qquad (124)$$

As a slight generalization, we may consider

$$b_\nu \;=\; \mathcal{B}_{\mathbb{N},q}(2^m N_\mu,\, b_\mu,\, x_{\mu m}) \;=\; \langle 2^q \nu + 2c + 1 \,|\, Q\rangle, \qquad (125)$$

where $q \geq 3$. Since $b_\nu \in J$, we see that there will be exactly 2^{M-q} distinct possible values of b_ν satisfying (125). Obviously, for (125),

$$\beta_{\mu\nu} \;=\; \langle 2^q(\mu - \nu) \,|\, Q\rangle; \qquad (126)$$

whence, $\gamma \geq q$. This result would indicate that, in fact, (124), with $q = 3$, is the best choice; though the considerations leading to (118) would suggest something closer to $q = 8$, instead.

Corollary 5 tells us that, so long as the parameters a, b_μ, and b_ν satisfy (95) and (123), and so long as, using (113), we require that $\Delta_{\mu\nu 0} = \langle x_{\mu m} - x_{\nu 0} \,|\, Q\rangle$ be *even*; the consonance between 'parallel' sequences, $[x_{\mu(i+m+1)}]_{i=0}^{\infty}$, beginning at node $2\nu = 2^{m+1}(2\mu + 1)$, and $[x_{\nu j}]_{j=0}^{\infty}$, begining at node $2\nu + 1$, will be $2^2 = 4$, which is just fine.

Consider now the situation (125), for some suitable q. A little thought indicates that any given b-value will occur just once in the triangular *apex* of the binary tree, consisting of Levels 0 through $M - q$ (in this part of the tree, there will be $2^{M-q+1} - 1$ nodes in all; of these, $2^{M-q} - 1$ will be even-numbered, and just 2^{M-q} will be odd-numbered). The same value will clearly recur, by (125), at intervals of 2^{M-q} in the node-index ν, which correspond to intervals of 2^{M-q+1} in the node-number $2\nu + 1$. Consequently, since Level k of the binary tree contains 2^k consecutively-numbered nodes, half of which are odd-numbered [see Figure 1]; we see that the same b-value will occur just once in Level $M - q + 1$, twice in Level $M - q + 2$, four times in Level $M - q + 3, \ldots,$ and $2^{k-M+q-1}$ times in Level $k > M - q$. Thus, the total number of occurrences of each possible b-value is $1 + 1 + 2 + 4 + \ldots + 2^{k-M+q-1} = 2^{k-M+q}$, as one would expect, since Levels 0 through k of the tree contain just 2^k odd-numbered nodes.

Now let us count how many x-values are generated altogether, in these $k + 1$ first levels of the tree, for a given b-value. The single sequence, with this b-value, starting in the apex may begin in any of its levels; so that the number r_k of x-values it generates satisfies

$$k - M + q + 1 \;\leq\; r_k \;\leq k + 1. \qquad (127)$$

The single sequence starting in Level $M - q + 1$ will generate just $k - M + q$ values, the two sequences starting in Level $M - q + 2$ will generate $2(k - M + q - 1)$ values, and so on. The total number of *x*-values generated in Levels 0 through *k*, using any one given *b-value* is thus

$$N_k = r_k + (k - M + q) + 2(k - M + q - 1) + 4(k - M + q - 2)$$
$$+ \ldots + 2^{h-M+q}(k - h) + \ldots + 2^{k-M+q-1}. \qquad (128)$$

Now, there are altogether $Q = 2^M$ possible *x*-values, and every sequence with parameter *a* satisfying (95) and odd *b*-value passes through all of them; so let us suppose that we arrange for the sequences listed above to occupy disjoint pieces of their single common period. Since all $\Delta_{\mu\nu 0}$ should be even, and the *x*-values in any sequence alternate in parity, we can arrange for each sequence to begin with an *x*-value of the correct parity if we allow a possible additional step. Allowing for this, we must have a range of $N_k + 2^{k-M+q}$ available *x*-values, and this cannot exceed 2^M. It follows that we can rearrange (128), with (127), noting that each term may be multiplied by 2 − 1, to yield that

$$\begin{aligned} N_k + 2^{k-M+q} \le \{\, k + 1 \,\} &+ 2(k - M + q) - \{\, k - M + q \,\} \\ &+ 4(k - M + q - 1) - 2(k - M + q - 1) \\ &+ 8(k - M + q - 2) - 4(k - M + q - 2) \\ &+ \ldots \\ &+ \{\, 2^{k-M+q} \,\} - 2^{k-M+q-1} \\ &+ \{\, 2^{k-M+q} \,\} \le 2^M, \end{aligned}$$

if *k* is to be adequate for *every* *b*-value. The middle expression above partially 'telescopes' (excepting the terms in { . . . }) to yield

$$\{\, M - q + 1 \,\} + 2 + 4 + 8 + \ldots + 2^{k-M+q-1} + \{\, 2^{k-M+q+1} \,\} \le 2^M,$$

and this, in turn, sums, if

$$-\ 2^{k-M+q} < M - q - 1 \le 2^{k-M+q}, \qquad (129)$$

to give, since the upper bound in (129) is an integer power of 2, that

$$M - q - 1 + 2^{k-M+q} + 2^{k-M+q+1}$$
$$= M - q - 1 + 3 \times 2^{k-M+q} \le 2^{k-M+q+2} \le 2^M, \quad (130)$$

is sufficient for all *b*-values. Let us write

$$M' = M - q \quad \text{and} \quad k' = k - M' = k - M + q; \qquad (131)$$

then, remembering that $k > M - q$, we see that the inequalities (129) and (130) become

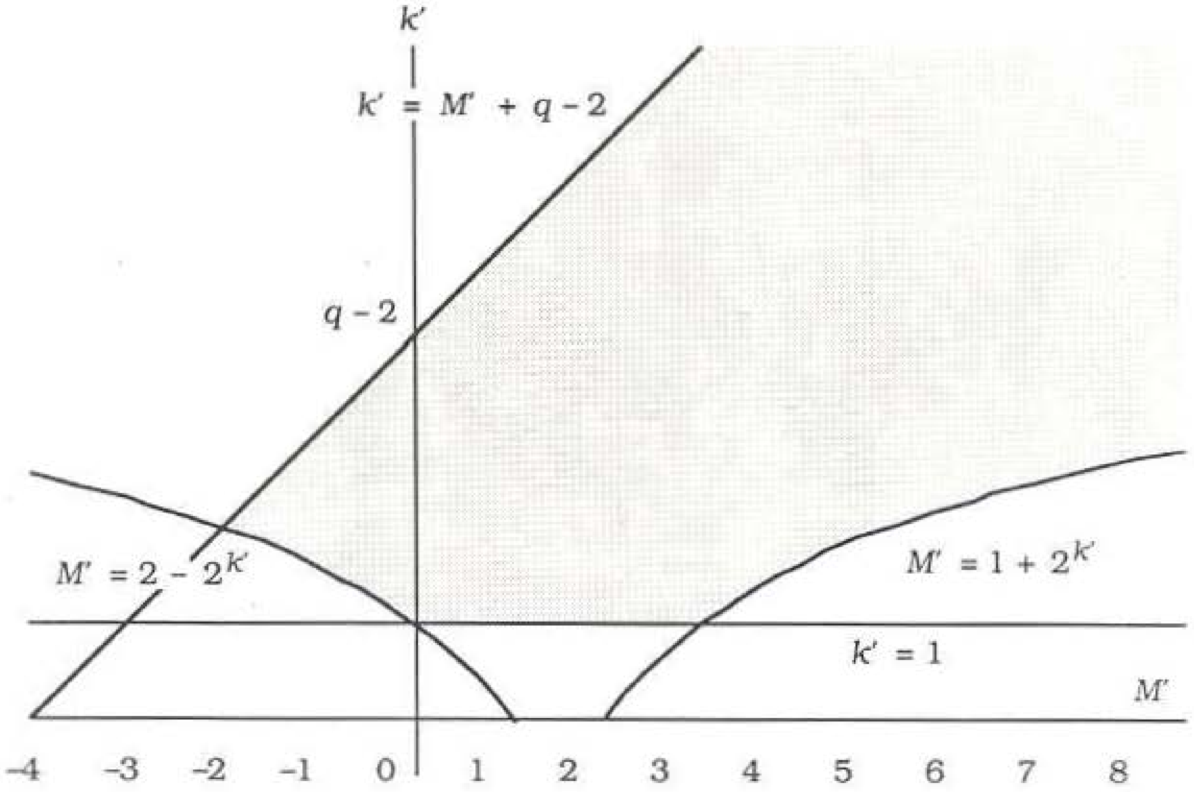
$$2 - 2^{k'} \le M' \le 1 + 2^{k'} \qquad (132)$$

and
$$1 \le k' \le M' + q - 2. \qquad (133)$$

These inequalities are illustrated in Figure 5, in which we see that (since we seek to avoid repetitions as long as possible, and therefore want k, and so also k', as large as possible, given M and q) the operative bound is $k' = M' + q - 2$, or

$$k = 2M - q - 2. \qquad (134)$$

If we choose $q = 3$, as in (124), then the optimum is $k = 2M - 5$; if $q = 8$, then it is $k = 2M - 10$.



<u>**Figure 5.**</u>

Graph of allowed (M' , k') region.

$M' = M - q$ and $k' = k - M + q$ satisfy the inequalities (132) and (133). This makes allowable the region shaded in the figure. Since M and q are given first, and we seek the greatest possible k, it is clear that the sloping line $k' = M' + q - 2$, or $k = 2M - q - 2$ yields the best value of k.

Now, let us suppose that the first occurrence of a particular b-value, say b^*, is at the node numbered $N_{v_0} = 2v_0 + 1$. This is in the apex of the tree, and so $v_0 < 2^{M-q}$. The initial x-value, $x_{v_0 0}$, is arbitrary, say x^*. By (125), subsequent occurrences of b^* will then be at nodes numbered $N_{v_s} = 2v_s + 1$, with $v_s = v_0 + s\,2^{M-q}$ $(s = 1, 2, \ldots)$. For the second occurrence of b^*, in Level $M - q + 1$ $(s = 1)$, we must select the initial x-value $x_{v_1 0}$ to be either $x_{v_0(k+1)}$ or $x_{v_0(k+2)}$, so as to match the parity of the x-value at the parent-node, numbered v_1, and to skip over all possible x-values (in the first $k + 1$ levels of the tree) of the sequence beginning at N_{v_0}. We generalize to the s-th occurrence of b^*, where the initial x-value $x_{v_s 0}$ must skip over all previously accounted-for x-values (allowing for the correct parity) by jumping to the value $x_{v_0 T_s}$ or $x_{v_0(T_s+1)}$. This means that we must have available all the coefficients [compare (77)]

$$A_s = \langle a^{T_s} | Q \rangle, \quad S_s = S_{T_s}(a), \quad B_s = \langle S_s b^* | Q \rangle; \qquad (135)$$

and, in particular, we see that

$$\text{Level } M - q + 1: \quad T_1 = k + 1 \quad = 2M - q - 1. \qquad (136)$$

Note that A_s, S_s, and T_s are all independent of the b-value b^*. For successive occurrences of b^*, it is easily verified that

$$\begin{aligned}
\text{Level } M - q + 2: \quad T_2 &= T_1 + 1 + (k - M + q) \\
&= T_1 + M - 1 = 3M - q - 2, \\
T_3 &= T_2 + 1 + (k - M + q - 1) \\
&= T_2 + M - 2 = 4M - q - 4; \\
\text{Level } M - q + 3: \quad T_4 &= T_3 + M - 2 = 5M - q - 6, \\
T_5 &= T_4 + M - 3 = 6M - q - 9, \\
T_6 &= T_5 + M - 3 = 7M - q - 12, \\
T_7 &= T_6 + M - 3 = 8M - q - 15; \\
\text{Level } M - q + 4: \quad T_8 &= T_7 + M - 3 = 9M - q - 18, \\
T_9 &= T_8 + M - 4 = 10M - q - 22, \\
T_{10} &= T_9 + M - 4 = 11M - q - 26, \qquad (137)
\end{aligned}$$

and so on. Clearly, the rule is:

$$T_{s+1} - T_s = M - n_s, \quad \text{where} \quad 2^{n_s-1} \le s < 2^{n_s}, \tag{138}$$

with n_s evidently unique. It follows that

$$\begin{aligned} T_s &= T_1 + \sum_{r=1}^{s-1} (T_{r+1} - T_r) = T_1 + \sum_{r=1}^{s} (M - n_r) - M + n_s \\ &= (s+1)M + n_s - q - 1 - \sum_{r=1}^{s} n_r . \end{aligned} \tag{139}$$

We observe that, for any integer $n > 0$, $n_r = n$, when $r = 2^{n-1}, 2^{n-1} + 1, 2^{n-1} + 2, \ldots, 2^n - 1$ (i.e., for 2^{n-1} consecutive values of r). Thus,

$$\begin{aligned} \sum_{r=1}^{s} n_r &= 1 + 2 \times 2 + 2^2 \times 3 + 2^3 \times 4 + \ldots \\ &\qquad + 2^{n_s-2}(n_s - 1) + (s + 1 - 2^{n_s-1})\, n_s \\ &= \begin{array}{llll} 2 & \times 1 & -1 & \times 1 \\ +2^2 & \times 2 & -2 & \times 2 \\ +2^3 & \times 3 & -2^2 & \times 3 \\ +2^4 & \times 4 & -2^3 & \times 4 \\ + & & \ldots & \end{array} \\ &\qquad + 2^{n_s-1}(n_s - 1) - 2^{n_s-2}(n_s - 1) + n_s(s + 1 - 2^{n_s-1}) \\ &= 2^{n_s-1}(n_s - 1) + n_s(s + 1 - 2^{n_s-1}) - \{1 + 2 + 2^2 + \ldots + 2^{n_s-2}\} \\ &= n_s(s+1) - \{1 + 2 + 2^2 + \ldots + 2^{n_s-1}\} = n_s(s+1) - 2^{n_s} + 1, \end{aligned}$$

where, again, we have taken advantage of the 'telescoping' trick used in deriving (130); so that

$$T_s = (s+1)M - n_s s + 2^{n_s} - q - 2. \tag{140}$$

It is easily verified that this agrees with the values in (137).

By (134), the total number of occurrences of b^* in Levels 0 through $k = 2M - q - 2$ is, as we have seen, $2^{k-M+q} = 2^{M-2}$, and so s will run from 0 through $2^{M-2} - 1$. Even though, as we have seen, we can economise by using the same parameters for all values of b^* [see (135)], it is still not practical to store such a large number of coefficients (typically, as we have noted, $M = 48$ and $2^{M-2} \approx 7 \times 10^{13}$).

so they must be computable when needed. To do this, we return to the concept of a *node record* $\mathbb{R}_\nu$ [see (89)] carrying all the information needed to generate both the left-slanting 'regular' branch/sequence, and any right-slanting children whenever the latter are appropriate. In order to continue the left-slanting branch, according to the generator $\Phi_\mu = \mathbb{S}(a, b_\mu, x_{\mu 0})$, it suffices that $\mathbb{R}_\nu$ should carry b_μ and $x_{\mu m}$ when $\nu = 2^m(2\mu + 1)$. In addition, $\mathbb{R}_\nu$ should carry sufficient information to yield b_ν and $x_{\nu 0}$ at the right-child node N_ν. By (125), q (or 2^q) and $b_0 = 2c + 1$ are universal parameters of the algorithm, like M (or $Q = 2^M$) and a; so $\mathbb{R}_\nu$ need only carry the index ν to enable us to compute $b_\nu = \langle 2^q\nu + b_0 | Q\rangle = b^*$.

If we adopt the simplest algorithm, embodied by (122), then (89) suffices; but, as we have seen, there will be many identical replications of sequences.

Algorithm 1. *The procedure carries at each node, numbered* $\nu = 2^m N_\mu = 2^m(2\nu + 1)$, *a record* $\mathbb{R}_\nu = (2^m N_\mu, b_\mu, x_{\mu m})$, *the transformation for which, on passage to the two child-nodes is given by*

$$\mathbb{R}_{2\nu} = \mathcal{L}_{S,q}(\mathbb{R}_\nu), \quad \mathbb{R}_{2\nu+1} = \mathbb{R}_{N_\nu} = \mathcal{M}_{S,q}(\mathbb{R}_\nu). \tag{141}$$

These mappings are defined by

$$\mathcal{L}_{S,q}(\mathbb{R}_\nu) = \left\{\begin{array}{r} (2\nu = 2^{m+1}N_\mu) \\ b\text{-value at } 2\nu \\ x_{\mu(m+1)} \end{array}\right\} = \left\{\begin{array}{l} 2 \times (\nu = 2^m N_\mu) \\ (b\text{-value at } \nu = b_\mu) \\ \langle a x_{\mu m} + b_\mu | Q\rangle \end{array}\right\} \tag{142}$$

and

$$\mathcal{M}_{S,q}(\mathbb{R}_\nu) =$$

$$\left\{\begin{array}{r} (2\nu + 1 = 2^{m+1}N_\mu + 1) \\ (b\text{-value at } N_\nu = b_\nu) \\ x_{\nu 0} \end{array}\right\} = \left\{\begin{array}{l} 2 \times (\nu = 2^m N_\mu) + 1 \\ \langle 2^q\nu + b_0 | Q\rangle \\ x_{\mu m} \end{array}\right\}. \tag{143}$$

This agrees with (122) and (125). Thus, $x_{\nu 0} = \mathcal{X}_{\mathrm{S}}(\mathbb{R}_\nu)$ and $b_\nu = \mathcal{B}_{\mathrm{N},q}(\mathbb{R}_\nu)$.

Now consider, instead, reducing the frequency and proximity of repetitions, by the scheme outlined above, in which

$$\begin{aligned} x_{\nu 0} &= \mathcal{X}_{\mathrm{T}}(\mathbb{R}_\nu) \\ &= \left\{ \begin{matrix} x_{\nu_0 T_s} \\ x_{\nu_0 (T_s+1)} \end{matrix} \right\} \left\{ \begin{matrix} \text{whichever has the} \\ \text{same parity as } x_{\mu m} \end{matrix} \right\}, \end{aligned} \tag{144}$$

where
$$\nu_0 = \langle \nu | 2^{M-q} \rangle, \quad s = (\nu - \nu_0)/2^{M-q}, \tag{145}$$

and T_s is given by (138) and (140), with $T_0 = 0$. The choice of the initial x-value $x^* = x_{\nu_0 0}$ for the first occurrence of any b^* is still open; so, looking to (122), let us determine m_0 and μ_0 (uniquely) by

$$\nu_0 = 2^{m_0}(2\mu_0 + 1), \tag{146}$$

and arbitrarily select, say:

$$\left. \begin{aligned} x_{00} &= f_0 : \quad \text{at the root of the tree;} \\ x_{\nu_0 0} &= \left\{ \begin{matrix} \langle 2^{q+1}\nu_0 + f_0 | Q \rangle = x^* \\ \langle a x^* + b^* | Q \rangle \end{matrix} \right\} \left\{ \begin{matrix} \text{whichever has the} \\ \text{same parity as } x_{\mu_0 m_0} \end{matrix} \right\}. \end{aligned} \right\} \tag{147}$$

Here, f_0, like b_0, is a universal parameter of the algorithm. Note that, fortunately, this does *not* disturb the x-value counts developed above; since the only case in which r_k attains its upper bound, $k + 1$ [see (127)], is that of the root-node, when an extra step is *never* required by (147). The only modification is that, because $x_{\nu_0 0}$ now does *not* necessarily equal x^*, we have to select the exact form of $x_{\nu 0}$; following (135), we choose to have

$$x_{\nu 0} = \langle A_s x^* + S_s b^* | Q \rangle, \tag{148}$$

which makes the formula accessible without keeping a record of the actual values of all $x_{\nu_0 0}$.

Let us express the index v in *binary notation* as

$$v = \{B_{k-1} \dots B_{M-q}\, B_{M-q-1} \dots B_0\}; \qquad (149)$$

where the B_j are the uniquely determined (0 or 1) *bits* of v, given by

$$B_j = \lfloor <v|2^{j+1}>/2^j \rfloor; \qquad (150)$$

and, since there are just 2^k odd-numbered nodes (numbered from 0 through $2^k - 1$) in Levels 0 through k, the k bits shown in (149) suffice. Then, by (88),

$$B_0 = B_1 = \dots = B_{m-1} = 0, \quad \text{but} \quad B_m = 1, \qquad (151)$$

and

$$\mu = \{B_{k-1} \dots B_{m+1}\}; \qquad (152)$$

while, by (142),

$$s = \{B_{k-1} \dots B_{M-q}\} \quad \text{and} \quad v_0 = \{B_{M-q-1} \dots B_0\}; \qquad (153)$$

and, by (138), n_s is defined as the *number of significant bits in s*, i.e., $B_{k-1} = \dots = B_{M-q+n_s} = 0$, and

$$s = \{B_{M-q+n_s-1} \dots B_{M-q}\} \quad \text{with} \quad B_{M-q+n_s-1} = 1. \qquad (154)$$

The node record will now have to be somewhat extended, beyond $\mathbb{R}_v$, to carry all the information needed to continue the process from the parent-node v to its left and right children; and we shall denote this superset by $\mathbb{R}^*_v$. Clearly, the information in the subset $\mathbb{R}_v$ suffices, to generate both the subset $\mathbb{R}_{2v}$ and the first two components, the node-number and b-value, of $\mathbb{R}_{N_v}$. However, to determine x_{v0} by (147) and (148), we need, apart from the universal parameters $b_0 = \{C_{q-1} \dots C_0\}$ and $f_0 = \{F_{q-1} \dots F_0\}$ of the algorithm, to have

$$b^* = \{B_{M-q-1} \dots B_0\, C_{q-1} \dots C_0\}, \qquad (155)$$

$$x^* = \{B_{M-q-2} \dots B_0\, F_{q-1} \dots F_0\}, \qquad (156)$$

and, in $\mathbb{R}^*_v$, the coefficients A_s and S_s.

Observe, by (135), that A_s and S_s depend only on s (through T_s); and that, in passing from a parent-node v to its children $2v$ and $2v + 1$, we change

$$\left.\begin{array}{ll} & \{0\ 0\ \ldots\ 0\ 0\ \blacklozenge\ \mathrm{B}_{M-q-1}\ \cdots\ \mathrm{B}_0\} \\ \text{into} & \{0\ 0\ \ldots\ 0\ \mathrm{B}_{M-q-1}\ \blacklozenge\ \mathrm{B}_{M-q-2}\ \cdots\ \mathrm{B}_0\ 0\} \\ \text{and} & \{0\ 0\ \ldots\ 0\ \mathrm{B}_{M-q-1}\ \blacklozenge\ \mathrm{B}_{M-q-2}\ \cdots\ \mathrm{B}_0\ 1\} \end{array}\right\}, \tag{157}$$

so long as $v < 2^{M-q}$ (i.e., $s = 0$; nodes numbered $2v$ and $2v + 1$ are in the apex of the binary tree); thereafter, when $v \geq 2^{M-q}$ (i.e., $s > 0$), we change

$$\left.\begin{array}{ll} & \{0\ 1\ \mathrm{B}_{M-q+n_s-2}\ \cdots\ \mathrm{B}_{M-q}\ \blacklozenge\ \mathrm{B}_{M-q-1}\ \cdots\ \mathrm{B}_0\} \\ \text{into} & \{1\ \mathrm{B}_{M-q+n_s-2}\ \cdots\ \mathrm{B}_{M-q}\ \mathrm{B}_{M-q-1}\ \blacklozenge\ \mathrm{B}_{M-q-2}\ \cdots\ \mathrm{B}_0\ 0\} \\ \text{and} & \{1\ \mathrm{B}_{M-q+n_s-2}\ \cdots\ \mathrm{B}_{M-q}\ \mathrm{B}_{M-q-1}\ \blacklozenge\ \mathrm{B}_{M-q-2}\ \cdots\ \mathrm{B}_0\ 1\} \end{array}\right\}; \tag{158}$$

the diamond ($\blacklozenge$) marks the separation between the bits of s and of v_0; so that, if we denote the values of s and n_s for the left child by s' and $n_{s'}$, and for the right child by s'' and $n_{s''}$, then

$$\left.\begin{array}{l} n_{s'} = n_{s''} = \begin{cases} n_s + \mathrm{Y}, & \text{if } s = 0, \\ n_s + 1, & \text{if } s \geq 1, \end{cases} \\ s' = s'' = 2s + \mathrm{Y}. \end{array}\right\} \tag{159}$$

where, for brevity, we write Y for B_{M-q-1}. Consequently, the children will share the values of $T_{s'} = T_{s''}$, $A_{s'} = A_{s''}$, and $S_{s'} = S_{s''}$; and, by (140), once $v \geq 2^{M-q-1}$ (i.e., the children are out of the apex),

$$\begin{aligned} T_{s'} = T_{s''} &= (2s + \mathrm{Y} + 1)M - (n_s + 1)(2s + \mathrm{Y}) + 2^{n_s+1} - q - 2 \\ &= 2T_s - 2s - \mathrm{Y}(M - n_s - 1) - (M - q - 2). \end{aligned} \tag{160}$$

Until then, n_s and s both remain zero.

We note, by Lemma 13, that $a^{2^{M-2}} \equiv 1 \pmod{2^M}$; so that

$$\hat{a} = a^{2^{M-2}-1} \tag{161}$$

acts as the *reciprocal* of a, modulo $2^M = Q$; whence any factor a^{-r}, appearing in integer-valued algebraic expressions reduced modulo Q, may be replaced by a factor $\hat{a}^r$. Thus, by (135), (140), and (161),

$$\begin{aligned} A_s = \langle a^{T_s} \,|\, \mathcal{Q} \rangle &= \langle a^{(s+1)M - n_s s + 2^{n_s} - q - 2} \,|\, \mathcal{Q} \rangle \\ &= \langle a^{M-q-2}\, a^{Ms}\, \hat{a}^{n_s s}\, a^{2^{n_s}} \,|\, \mathcal{Q} \rangle. \end{aligned} \tag{162}$$

When we turn to the other coefficient, $S_s = S_{T_s}(a)$, that we shall need to carry at every node, we need to establish some straightforward properties of the function $S_n(z)$.

Lemma 20. *For any non-negative integers p and q, and real z,*

$$S_{p+q}(z) = S_p(z) + z^p S_q(z), \tag{163}$$

$$S_{p-q}(z) = S_p(z) - z^{p-q} S_q(z), \tag{164}$$

$$S_{pq}(z) = S_p(z)\, S_q(z^p); \tag{165}$$

and, in particular,

$$S_{2p}(z) = (1 + z)\, S_p(z^2) = (1 + z^p)\, S_p(z). \tag{166}$$

≪≪We refer to the definition (5). If $z = 1$, then, by (6), $(\forall n \geq 0)$ $z^n = 1$ and $S_n(1) = n$; whence (163) - (166) all hold, as is trivial to verify. Suppose, therefore, that $z \neq 1$. Then,

$$\begin{aligned} S_{p+q}(z) &= 1 + z + z^2 + \ldots + z^{p-1} + z^p + z^{p+1} + \ldots + z^{p+q-1} \\ &= (1 + z + z^2 + \ldots + z^{p-1}) + z^p (1 + z + z^2 + \ldots + z^{q-1}), \end{aligned}$$

from which (163) follows at once. Replacing p by p' in (163) and rearranging terms, we get

$$S_{p'}(z) = S_{p'+q}(z) - z^{p'} S_q(z); \tag{167}$$

whence (164) follows immediately, when we write $p' = p - q$. Now, by repeated application of (163), we see that

$$\begin{aligned} S_{pq}(z) &= S_{p+p(q-1)}(z) = S_p(z) + z^p S_{p(q-1)}(z) \\ &= S_p(z) + z^p S_p(z) + z^{2p} S_{p(q-2)}(z) \\ &= S_p(z) + z^p S_p(z) + z^{2p} S_p(z) + z^{3p} S_{p(q-3)}(z) \\ &= \ldots = S_p(z)\, \{1 + z^p + z^{2p} + z^{3p} + \ldots + z^{(q-1)p}\}, \end{aligned}$$

which yields (165). Finally, we note that the equality of the first and second members of (166) is Lemma 1 [equation (8)], while, if we put $q = p$ in (163), we get the equality of the first and third members of (166). Also, the same two identities are obtained, respectively, by putting $p = 2$ (and then replacing q by p) and by putting $q = 2$, in (165).$\gg\gg$

By (140), and (163) and (164) of Lemma (20), we see that

$$\begin{aligned} S_{T_s}(a) &= S_{(s+1)M-n_s s+2^{n_s}-q-2}(a) \\ &= S_{Ms-n_s s+2^{n_s}}(a) + a^{Ms-n_s s+2^{n_s}} S_{M-q-2}(a) \\ &= S_{Ms-n_s s}(a) + a^{Ms-n_s s} (S_{2^{n_s}}(a) + a^{2^{n_s}} S_{M-q-2}(a)) \\ &= S_{Ms}(a) - a^{Ms-n_s s} \left\{S_{n_s s}(a) - S_{2^{n_s}}(a) - a^{2^{n_s}} S_{M-q-2}(a)\right\} \\ &= S_M(a)\, S_s(a^M) \\ &\qquad - a^{Ms-n_s s} \left\{S_{n_s s}(a) - S_{2^{n_s}}(a) - a^{2^{n_s}} S_{M-q-2}(a)\right\}; \end{aligned} \tag{168}$$

so that, by (135),

$$\begin{aligned} S_s = S_{T_s}(a) &= <S_M(a)\, S_s(a^M) \\ &\qquad - a^{Ms}\, \hat{a}^{n_s s} \left\{S_{n_s s}(a) - S_{2^{n_s}}(a) - a^{2^{n_s}} S_{M-q-2}(a)\right\} \mid Q>. \end{aligned} \tag{169}$$

An examination of (162) and (169) reveals the parameters which need to be carried in the record $\mathbb{R}^*_\nu$, and updated from father-node to children, to execute the algorithm. The supplementary universal parameters of the algorithm,

$$\left.\begin{aligned} K_0 &= S_M(a), & K_0^* &= S_{M-q-2}(a), \\ K_1 &= a^M, & K_1^* &= a^{M-q-2}, \\ K_2 &= a^2, & K_2^* &= a^{2^{M-2}-2} = \hat{a}^2 \pmod{Q}, \end{aligned}\right\}, \tag{170}$$

are computed once and for all, and stored with M (and $Q = 2^M$), a, q, b_0, f_0, and $\hat{a}$, to be used at all nodes. This leaves thirteen coefficients to be added to $\mathbb{R}_\nu$ to make up $\mathbb{R}^*_\nu$; namely,

$$\left.\begin{array}{lll} U_s = a^{2s}, & V_s = a^{n_s}, & W_s = a^{n_s s}, \\ U_s^* = \hat{a}^{2s}, & V_s^* = \hat{a}^{n_s}, & W_s^* = \hat{a}^{n_s s}, \\ X_s = a^{2^{n_s}}, & Y_s = S_{2s}(a), & Z_s = S_{n_s}(a), \\ X_s^* = a^{Ms}, & Y_s^* = S_s(a^M), & Z_s^* = S_{n_s s}(a), \\ X_s\dagger = S_{2^{n_s}}(a). & & \end{array}\right\} \tag{171}$$

Using (159) and Lemma 20, we can now compute the update-relations for these. First, observe that, when $0 \le \nu < 2^{M-q}$, $n_s = s = 0$; whence, for all such ν,

$$\left.\begin{array}{c} U_s = V_s = W_s = U_s^* = V_s^* = W_s^* = X_s^* = X_s\dagger = 1, \\ Y_s = Y_s^* = Z_s = Z_s^* = 0, \quad \text{and} \quad X_s = a. \end{array}\right\} \tag{172}$$

Next, note that the bit $\mathsf{Y} = \mathsf{B}_{M-q-1}$ can only be 0 or 1, and $S_0(z) = 0$ and $S_1(z) = 1$, so that

$$S_{\mathsf{Y}}(z) = \mathsf{Y} \quad \text{and} \quad S_{n\mathsf{Y}}(z) = \mathsf{Y}\, S_n(z). \tag{173}$$

Also, an additive term having a factor Y, or a multiplicative factor whose exponent has a factor Y, is disregarded unless $\mathsf{Y} = 1$ (and, in that case, the factor Y may be omitted). Thus, for all $\nu \ge 2^{M-q-1}$, we see that:

$$U_{s'} = U_{s''} = a^{2(2s+\mathsf{Y})} = U_s^2\, K_2^{\mathsf{Y}}, \tag{174}$$

$$V_{s'} = V_{s''} = a^{n_s+1} = a^{n_s}\, a = V_s\, a, \tag{175}$$

$$\begin{aligned} W_{s'} = W_{s''} = a^{(n_s+1)(2s+\mathsf{Y})} &= a^{2n_s s+2s+n_s\mathsf{Y}+\mathsf{Y}} \\ &= W_s^2\, U_s\, (V_s\, a)^{\mathsf{Y}} = W_s^2\, U_s\, V_{s'}^{\mathsf{Y}}, \end{aligned} \tag{176}$$

$$U_{s'}^* = U_{s''}^* = \hat{a}^{2(2s+\mathsf{Y})} = U_s^{*2}\, K_2^{*\mathsf{Y}}, \tag{177}$$

$$V_{s'}^* = V_{s''}^* = \hat{a}^{n_s+1} = \hat{a}^{n_s}\, \hat{a} = V_s^*\, \hat{a}, \tag{178}$$

$$\begin{aligned} W_{s'}^* = W_{s''}^* = \hat{a}^{(n_s+1)(2s+\mathsf{Y})} &= \hat{a}^{2n_s s+2s+n_s\mathsf{Y}+\mathsf{Y}} \\ &= W_s^{*2}\, U_s^*\, (V_s^*\, \hat{a})^{\mathsf{Y}} = W_s^{*2}\, U_s^*\, V_{s'}^{*\mathsf{Y}}, \end{aligned} \tag{179}$$

$$X_{s'} = X_{s''} = a^{2^{n_s+1}} = (a^{2^{n_s}})^2 = X_s^2, \tag{180}$$

$$\begin{aligned}
Y_{s'} = Y_{s''} &= S_{2(2s+\mathrm{Y})}(a) = (1 + a^{2s+\mathrm{Y}})\, S_{2s+\mathrm{Y}}(a) \\
&= (1 + U_s\, a^{\mathrm{Y}})\, [S_{2s}(a) + a^{2s}\, S_{\mathrm{Y}}(a)] \\
&= (1 + U_s\, a^{\mathrm{Y}})\, (Y_s + U_s\, \mathrm{Y}). \qquad (181)
\end{aligned}$$

$$Z_{s'} = Z_{s''} = S_{n_s+1}(a) = S_{n_s}(a) + a^{n_s}\, S_1(a) = Z_s + V_s. \qquad (182)$$

$$X_{s'}{}^{*} = X_{s''}{}^{*} = a^{M(2s+\mathrm{Y})} = X_s{}^{*2}\, K_1{}^{\mathrm{Y}}. \qquad (183)$$

$$\begin{aligned}
Y_{s'}{}^{*} = Y_{s''}{}^{*} &= S_{2s+\mathrm{Y}}(a^M) = S_{2s}(a^M) + a^{2Ms}\, S_{\mathrm{Y}}(a^M) \\
&= (1 + a^{Ms})\, S_s(a^M) + X_s{}^{*2}\, \mathrm{Y} \\
&= (1 + X_s{}^{*})\, Y_s{}^{*} + X_s{}^{*2}\, \mathrm{Y}. \qquad (184)
\end{aligned}$$

$$\begin{aligned}
Z_{s'}{}^{*} = Z_{s''}{}^{*} &= S_{(n_s+1)(2s+\mathrm{Y})}(a) = S_{2n_s s+2s+n_s\mathrm{Y}+\mathrm{Y}}(a) \\
&= S_{2n_s s+2s+n_s\mathrm{Y}}(a) + a^{2n_s s+2s+n_s\mathrm{Y}}\, S_{\mathrm{Y}}(a) \\
&= S_{2n_s s+2s}(a) + a^{2n_s s+2s}\, S_{n_s\mathrm{Y}}(a) + W_s{}^2\, U_s\, V_s{}^{\mathrm{Y}}\, \mathrm{Y} \\
&= S_{2n_s s}(a) + a^{2n_s s}\, S_{2s}(a) + W_s{}^2\, U_s\, \mathrm{Y}\, (Z_s + V_s) \\
&= (1 + a^{n_s s})\, S_{n_s s}(a) + W_s{}^2\, [Y_s + U_s\, \mathrm{Y}\, (Z_s + V_s)] \\
&= (1 + W_s)\, Z_s{}^{*} + W_s{}^2\, [Y_s + U_s\, \mathrm{Y}\, (Z_s + V_s)]. \qquad (185)
\end{aligned}$$

$$X_{s'}\dagger = X_{s''}\dagger = S_{2^{n_s+1}}(a) = S_{2\times 2^{n_s}}(a) = (1 + X_s)\, X_s\dagger. \qquad (186)$$

In terms of these coefficients, we now see that (162) and (169) become

$$A_s = \langle K_1{}^{*}\, X_s{}^{*}\, W_s{}^{*}\, X_s \,|\, Q\rangle \qquad (187)$$

and
$$S_s = \langle K_0\, Y_s{}^{*} - X_s{}^{*}\, W_s{}^{*}\, (Z_s{}^{*} - X_s\dagger - X_s\, K_0{}^{*}) \,|\, Q\rangle. \qquad (188)$$

Algorithm 2. *The procedure carries at each node, numbered* $\nu = 2^m N_\mu = 2^m(2\nu + 1)$, *a record*

$$\left.\begin{aligned}
R^{*}{}_{\nu} &= [R_\nu;\ C_\nu] \\
&= [2^m N_\mu,\ b_\mu,\ x_{\mu m}; \\
&\qquad U_s,\ V_s,\ W_s,\ U_s{}^{*},\ V_s{}^{*},\ W_s{}^{*}, \\
&\qquad X_s,\ Y_s,\ Z_s,\ X_s{}^{*},\ Y_s{}^{*},\ Z_s{}^{*},\ X_s\dagger],
\end{aligned}\right\} \qquad (189)$$

the transformation for which, on passage to the two child-nodes is given by

$$\mathbb{R}^*_{2\nu} = \mathcal{L}_{T,q}(\mathbb{R}^*_\nu), \quad \mathbb{R}^*_{2\nu+1} = \mathbb{R}^*_{N_\nu} = \mathcal{M}_{T,q}(\mathbb{R}^*_\nu). \qquad (190)$$

These mappings are defined as follows.

(a) *for* $\mathbb{R}_\nu$:

$$\mathcal{L}_{T,q}(\nu = 2^m N_\mu) = (2\nu = 2^{m+1} N_\mu),$$

$$\mathcal{L}_{T,q}(b\text{-value at } \nu = b_\mu) = b_\mu,$$

$$\mathcal{L}_{T,q}(x\text{-value at } \nu = x_{\mu m}) = (x_{\mu(m+1)} = \langle a\, x_{\mu m} + b_\mu | Q\rangle);$$

and

$$\mathcal{M}_{T,q}(\nu = 2^m N_\mu) = (2\nu + 1 = 2^{m+1} N_\mu + 1),$$

$$\mathcal{M}_{T,q}(b\text{-value at } \nu = b_\mu) = \langle 2^q \nu + b_0 | Q\rangle,$$

$$\mathcal{M}_{T,q}(x\text{-value at } \nu = x_{\mu m}) = (x_{\nu 0} = \langle A_s\, x^* + S_s\, b^* | Q\rangle),$$

where $x^* = \langle 2^{q+1} v_0 + f_0 | Q\rangle$, $b^* = \langle 2^q \nu + b_0 | Q\rangle$, *and* A_s *and* S_s *are computed from* (187) *and* (188), *using the coefficients in* $\mathbb{C}_\nu$.

(b) *for* $\mathbb{C}_\nu$: *the coefficients remain at the values in* (172), *so long as* $\nu < 2^{M-q}$; *thereafter, when* $\nu \geq 2^{M-q-1}$,

$$\mathcal{L}_{T,q}(U_s) = U_{s'} = \mathcal{M}_{T,q}(U_s) = U_{s''} = \langle U_s^2 K_2^{Y} | Q\rangle,$$

$$\mathcal{L}_{T,q}(V_s) = V_{s'} = \mathcal{M}_{T,q}(V_s) = V_{s''} = \langle V_s\, a | Q\rangle,$$

$$\mathcal{L}_{T,q}(W_s) = W_{s'} = \mathcal{M}_{T,q}(W_s) = W_{s''} = \langle W_s^2 U_s V_{s'}^{Y} | Q\rangle,$$

$$\mathcal{L}_{T,q}(U_s^*) = U_{s'}^* = \mathcal{M}_{T,q}(U_s^*) = U_{s''}^* = \langle U_s^{*2} K_2^{*Y} | Q\rangle,$$

$$\mathcal{L}_{T,q}(V_s^*) = V_{s'}^* = \mathcal{M}_{T,q}(V_s^*) = V_{s''}^* = \langle V_s^*\, \hat{a} | Q\rangle,$$

$$\mathcal{L}_{T,q}(W_s^*) = W_{s'}^* = \mathcal{M}_{T,q}(W_s^*) = W_{s''}^* = \langle W_s^{*2} U_s^* V_{s'}^{*Y} | Q\rangle.$$

$$\mathcal{L}_{T,q}(X_s) = X_{s'} = \mathcal{M}_{T,q}(X_s) = X_{s''} = \langle X_s^2 \,|\, Q \rangle,$$

$$\mathcal{L}_{T,q}(Y_s) = Y_{s'} = \mathcal{M}_{T,q}(Y_s) = Y_{s''}$$
$$= \langle (1 + U_s\, a^{\mathrm{Y}})\,(Y_s + U_s\, \mathrm{Y}) \,|\, Q \rangle,$$

$$\mathcal{L}_{T,q}(Z_s) = Z_{s'} = \mathcal{M}_{T,q}(Z_s) = Z_{s''} = \langle Z_s + V_s \,|\, Q \rangle,$$

$$\mathcal{L}_{T,q}(X_s{}^*) = X_{s'}{}^* = \mathcal{M}_{T,q}(X_s{}^*) = X_{s''}{}^* = \langle X_s{}^{*2}\, K_1{}^{\mathrm{Y}} \,|\, Q \rangle,$$

$$\mathcal{L}_{T,q}(Y_s{}^*) = Y_{s'}{}^* = \mathcal{M}_{T,q}(Y_s{}^*) = Y_{s''}{}^*$$
$$= \langle (1 + X_s{}^*)\, Y_s{}^* + X_s{}^{*2}\, \mathrm{Y} \,|\, Q \rangle,$$

$$\mathcal{L}_{T,q}(Z_s{}^*) = Z_{s'}{}^* = \mathcal{M}_{T,q}(Z_s{}^*) = Z_{s''}{}^*$$
$$= \langle (1 + W_s)\, Z_s{}^* + W_s{}^2\, [Y_s + U_s\, \mathrm{Y}\, (Z_s + V_s)] \,|\, Q \rangle,$$

$$\mathcal{L}_{T,q}(X_s{}^\dagger) = X_{s'}{}^\dagger = \mathcal{M}_{T,q}(X_s{}^\dagger) = X_{s''}{}^\dagger = \langle (1 + X_s)\, X_s{}^\dagger \rangle;$$

where the various symbols are defined in (170) *and* (171).

A multiplication-count [there are no divisions, and we may suppose that the reductions modulo Q are performed by truncation of binary computer-words; also, we do not count multiplications by powers of 2, which can be performed by fast bit-shifts] yields 1 for generating the three components of $\mathbb{R}_{2\nu}$ by $\mathcal{L}_{T,q}$, and 9 for generating the three components of $\mathbb{R}_{2\nu+1}$ by $\mathcal{M}_{T,q}$; while, for generating the thirteen components of $\mathbb{C}_{2\nu}$ and $\mathbb{C}_{2\nu+1}$ (which are identical), we require 15 multiplications when $\mathrm{Y} = 0$ and 22 when $\mathrm{Y} = 1$. Thus, from Level $M - q - 1$ on, the algorithm takes, altogether, an average of 28.5 multiplications to generate the records $\mathbb{R}^*$ of both children of any given node (considerably less in the apex of the tree), an average of 14.25 multiplications per node. Since these nodes occur only every T pseudo-random numbers, as we explained earlier [see Lemma 18], and we expect T to be of the order of 10, with other steps taking 1 multiplication to perform, by (4); the overall expected number of multiplications per pseudo-random number generated will only be about 2.325; that is, between 2 and 3 times the time required by the highly-efficient linear congruential generator itself, without any tree-structure. This would appear to be highly satisfactory.

6. COMPUTATIONAL RESULTS

A program was written in "C" to execute **Algorithm 1**. A typical complete output [part of which is condensed into two-column format to save space] is given in Appendix A, and the listing of the program is given in Appendix B. The section which inputs and computes the universal parameters of the algorithm and initializes the first record takes essentially *10 commands* ["`for (...)`" and "`while (...)`" are each counted as one command, additional to what they control; and inessential commands, used to keep track of what is happening in this test-program, are not counted]. The section which "builds the tree", i.e., computes the records at the child-nodes of a given parent-node, takes essentially *6 commands* [note that such variables as "`space`", "`level`", "`mu`", "`power`", "`BS`", "`XS`", and "`NS`" are used for bookkeeping functions in the test-program and are ignored].

With $M = 6$ and $q = 3$, the computations covered the first 255 nodes of the tree (8 levels). Eight sets of data were tried, yielding the following results, with respect to repetitions.

```
DATA-SET 1

a = 21,  b0 = 3,  f0 = 7

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all

DATA-SET 2

a = 37,  b0 = 63,  f0 = 57

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all
```

```
DATA-SET 3

a = 5,  b0 = 7,  f0 = 5

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all

DATA-SET 4

a = 53,  b0 = 1,  f0 = 1

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all
```

```
DATA-SET 5

a = 45,  b0 = 11,  f0 = 37

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all
```

```
DATA-SET 6

a = 13,  b0 = 33,  f0 = 33

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all
```

```
DATA-SET 7

a = 21,  b0 = 11,  f0 = 0

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all
```

```
DATA-SET 8

a = 5,  b0 = 33,  f0 = 42

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     0 repetitions
Level 4:     3 repetitions
Level 5:     7 repetitions
Level 6:    16 repetitions
Level 7:    35 repetitions

 61 repetitions in all
```

Since the values of a [subject only to (95)], b_0 [odd], and f_0 were chosen quite aimlessly, the recurring pattern of repetitions suggests that a theorem underlies it; the number of repetitions at each level is probably a constant, depending only on M and q. Further experimentation, varying M and q, supports this conjecture. For example, covering the first 511 nodes, when $M = 7$ and $q = 5$, we get:

```
DATA-SET 9

a = 5,  b0 = 5,  f0 = 5

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     2 repetitions
Level 4:     4 repetitions
Level 5:     9 repetitions
Level 6:    17 repetitions
Level 7:    22 repetitions
Level 8:    21 repetitions

 75 repetitions in all
```

```
DATA-SET 10

a = 37,  b0 = 23,  f0 = 30

Level 0:     0 repetitions
Level 1:     0 repetitions
Level 2:     0 repetitions
Level 3:     2 repetitions
Level 4:     4 repetitions
Level 5:     9 repetitions
Level 6:    17 repetitions
Level 7:    22 repetitions
Level 8:    21 repetitions

 75 repetitions in all
```

Another program was written in "C" to execute **Algorithm 2**. Again, a typical complete output is given in Appendix A, partly abbreviated, and the listing of the program is given in Appendix B. The section which inputs and computes the universal parameters of the algorithm and initializes the first record now takes essentially *47 commands*, and now the section which "builds the tree" (a simpler section builds the 'apex' of the tree; we consider the much more complicated section which builds the lower part of the tree) takes essentially *59 commands* ["`if (...)`" or "`if (...)...else...`" is counted as an additional command, wherever it occurs, as are "`for (...)`" and "`while (...)`"; inessential commands are ignored]. Thus, the avoidance of repetitions in the first $k + 1 = 2M - q - 1$ levels of the tree (and commensurate avoidance of repetitions thereafter, within $2^{k-M+q} = 2^{M-2}$ occurrences of any b-value) costs a factor of $59/6 \approx 10$ in program-complexity.

With $M = 6$ and $q = 3$, the computations again covered the first 255 nodes (8 levels) of the tree. The same eight sets of data were tried, yielding the theoretically predicted absence of repetitions in the first $k + 1 = 2M - q - 1 = 8$ levels. Only the first two outputs are given below; the other six are identical. Clearly, the same conjectured theorem applies here, too.

```
DATA-SET 1                              DATA-SET 2

a = 21,  b0 = 3,  f0 = 7                a = 37,  b0 = 63,  f0 = 57

Level 0:     0 repetitions              Level 0:     0 repetitions
Level 1:     0 repetitions              Level 1:     0 repetitions
Level 2:     0 repetitions              Level 2:     0 repetitions
Level 3:     0 repetitions              Level 3:     0 repetitions
Level 4:     0 repetitions              Level 4:     0 repetitions
Level 5:     0 repetitions              Level 5:     0 repetitions
Level 6:     0 repetitions              Level 6:     0 repetitions
Level 7:     0 repetitions              Level 7:     0 repetitions

  0 repetitions in all                    0 repetitions in all
```

7. BIBLIOGRAPHY

BUS 62 N. P. BUSLENKO, D. I. GOLENKO, YU. A. SHREIDER, I. M. SOBOL', V. G. SRAGOVICH. *The Method of Statistical Trials—The Monte Carlo Method.* Edited by YU. A. SHREIDER; Fizmatgiz, Moscow (1962) [in Russian]; Elsevier, Amsterdam (1964) 312 pp.; Pergamon Press, Oxford (1966) 390 pp. [two different translations].

CAR 75 L. L. CARTER, E. D. CASHWELL. *Particle Transport Simulation with the Monte Carlo Method.* Technical Information Center, Energy Research and Development Administration [ERDA], Oak Ridge, TN (1975) 121 pp.

ERM 71 S. M. ERMAKOV. *The Monte Carlo Method and Contiguous Questions.* Nauka, Moscow; First Edition (1971) 328 pp.; Second Edition (1975) 472 pp. [in Russian].

FRA 63 J. N. FRANKLIN. Deterministic simulation of random processes. *Math. Comp.* 17 (1963) pp. 28–59.

FRE 84 P. FREDERICKSON, R. HIROMOTO, T. L. JORDAN, B. SMITH, T. WARNOCK. Pseudo-random trees in Monte Carlo. *Parallel Computing* 1 (1984) pp. 175–180.

GRE 65 M. GREENBERGER. Method in randomness. *Commun. ACM* 8 (1965) pp. 177–179.

HAL 60 J. H. HALTON. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* 2 (1960) pp. 84–90.

HAL 70 J. H. HALTON. A retrospective and prospective survey of the Monte Carlo method. *SIAM Review* 12 (1970) pp. 1–63.

HAL 72 J. H. HALTON. Estimating the accuracy of quasi-Monte Carlo integration. *Applications of Number Theory to Numerical Analysis.* Edited by S. K. ZAREMBA; Academic Press, New York (1972) pp. 345–360.

HAL 87 J. H. HALTON. On a new class of independent families of linear congruential pseudo-random sequences. Univ. N. C., Comp. Sci. Dept., Tech. Report No. 87–001 (1987) 22 pp.

HAM 60 J. M. HAMMERSLEY. Monte Carlo methods for solving multivariable problems. *Ann. New York Acad. Sci.* 86 (1960) pp. 844–874.

HAM 64 J. M. HAMMERSLEY, D. C. HANDSCOMB. *Monte Carlo Methods.* Methuen, London; John Wiley, New York; (1964) 185 pp.

HUL 62 T. E. HULL, A. R. DOBELL. Random number generators. *SIAM Review* 4 (1962) pp. 230–254.

JAN 66 B. JANSSON. *Random Number Generators.* Doctoral Thesis, Faculty of Mathematics and Natural Sciences, University of Stockholm (1966) 205 pp. [Distributed by Almquist and Wiksell, Stockholm.]

KLE 75 J. P. C. KLEIJNEN. *Statistical Techniques in Simulation.* Marcel Dekker, New York; *Part I* (1974) 300 pp.; *Part II* (1975) 503 pp.

LEH 51 D. H. LEHMER. Mathematical methods in large-scale computing units. *Proc. Second Symposium on Large-Scale Digital Calculating Machinery, 1949.* Harvard University, Cambridge, MA (1951) pp. 141–146.

NIE 78 H. NIEDERREITER. Quasi-Monte Carlo methods and pseudo-random numbers. *Bull. Amer. Math. Soc.* 84 (1978) pp. 957–1041.

MAR 72 G. MARSAGLIA. The structure of linear congruential sequences. *Applications of Number Theory to Numerical Analysis.* Edited by S. K. ZAREMBA; Academic Press, New York (1972) pp. 249–285.

RUB 81 R. Y. RUBINSTEIN. *Simulation and the Monte Carlo Method.* John Wiley, New York (1981) 293 pp.

SOB 73 I. M. SOBOL'. *Monte Carlo Computational Methods.* Nauka, Moscow (1973) 312 pp. [in Russian].

SPA 69 J. SPANIER, E. M. GELBARD. *Monte Carlo Principles and Neutron Transport Problems.* Addison-Wesley, Reading, MA (1969) 248 pp.

TAU 65 R. C. TAUSWORTHE. Random numbers generated by linear recurrence modulo 2. *Math. Comp.* 19 (1965) pp. 201–209.

WAR 83 T. T. WARNOCK. Synchronization of random number generators. *Congressus Numerantium* 37 (1983) pp. 135–144.

YAK 77 S. J. YAKOWITZ. *Computational Probability and Simulation.* Addison-Wesley, Reading, MA (1977) 262 pp.

ZAR 66 S. K. ZAREMBA. Good lattice points, discrepancy, and numerical integration. *Ann. Math. Pura Appl.* IV: 73 (1966) pp. 293–317.

ZAR 68 S. K. ZAREMBA. The mathematical basis of Monte Carlo and quasi-Monte Carlo methods. *SIAM Review.* 10 (1968) pp. 303–314.

ACKNOWLEDGEMENTS

I am grateful to several colleagues, and especially to Dr T. T. Warnock of Cray Research, Inc., for some interesting and stimulating discussions of the applications, desirable properties, and necessary limitations of tree-structured random generators of the type developed here. I am also much indebted to Dr T. E. Booth of Los Alamos National Laboratory for an exceptionally careful reading of earlier versions of the manuscript, and for many relevant comments.

APPENDIX A

Computational Results

An example of the output of the program for Algorithm 1 is given below.

```
M = 6, a = 21, b0 = 3, f0 = 7, q = 3

 Node  Level   mu   power   b     x      Node  Level   mu   power   b     x
   1     0      0     0     3     7       41     5     20     0     35     8
   2     1      0     1     3    22       42     5     10     1     19     8
   3     1      1     0    11     7       43     5     21     0     43    33
   4     2      0     2     3    17       44     5      5     2     43    24
   5     2      2     0    19    22       45     5     22     0     51    57
   6     2      1     1    11    30       46     5     11     1     27    41
   7     2      3     0    27     7       47     5     23     0     59    22
   8     3      0     3     3    40       48     5      1     4     11    43
   9     3      4     0    35    17       49     5     24     0      3    32
  10     3      2     1    19    33       50     5     12     1     35    56
  11     3      5     0    43    22       51     5     25     0     11     1
  12     3      1     2    11     1       52     5      6     2     51    16
  13     3      6     0    51    30       53     5     26     0     19    41
  14     3      3     1    27    46       54     5     13     1     43    33
  15     3      7     0    59     7       55     5     27     0     27    30
  16     4      0     4     3    11       56     5      3     3     27    16
  17     4      8     0     3    40       57     5     28     0     35    33
  18     4      4     1    35     8       58     5     14     1     51    57
  19     4      9     0    11    17       59     5     29     0     43    46
  20     4      2     2    19     8       60     5      7     2     59    33
  21     4     10     0    19    33       61     5     30     0     51    14
  22     4      5     1    43    57       62     5     15     1     59    14
  23     4     11     0    27    22       63     5     31     0     59     7
  24     4      1     3    11    32       64     6      0     6      3    53
  25     4     12     0    35     1       65     6     32     0      3    42
  26     4      6     1    51    41       66     6     16     1      3    42
  27     4     13     0    43    30       67     6     33     0     11    11
  28     4      3     2    27    33       68     6      8     2      3    42
  29     4     14     0    51    46       69     6     34     0     19    11
  30     4      7     1    59    14       70     6     17     1     11    19
  31     4     15     0    59     7       71     6     35     0     27    40
  32     5      0     5     3    42       72     6      4     3     35    10
  33     5     16     0     3    11       73     6     36     0     35    11
  34     5      8     1     3    11       74     6     18     1     19    59
  35     5     17     0    11    40       75     6     37     0     43     8
  36     5      4     2    35    11       76     6      9     2     11    59
  37     5     18     0    19     8       77     6     38     0     51    48
  38     5      9     1    11    48       78     6     19     1     27     0
  39     5     19     0    27    17       79     6     39     0     59    17
  40     5      2     3    19    59       80     6      2     4     19    42
```

Node	Level	mu	power	b	x
81	6	40	0	3	59
82	6	20	1	35	11
83	6	41	0	11	8
84	6	10	2	19	59
85	6	42	0	19	8
86	6	21	1	43	32
87	6	43	0	27	33
88	6	5	3	43	35
89	6	44	0	35	24
90	6	22	1	51	32
91	6	45	0	43	57
92	6	11	2	27	56
93	6	46	0	51	41
94	6	23	1	59	9
95	6	47	0	59	22
96	6	1	5	11	18
97	6	48	0	3	43
98	6	24	1	3	35
99	6	49	0	11	32
100	6	12	2	35	59
101	6	50	0	19	56
102	6	25	1	11	32
103	6	51	0	27	1
104	6	6	3	51	3
105	6	52	0	35	16
106	6	26	1	19	48
107	6	53	0	43	41
108	6	13	2	43	32
109	6	54	0	51	33
110	6	27	1	27	17
111	6	55	0	59	30
112	6	3	4	27	43
113	6	56	0	3	16
114	6	28	1	35	24
115	6	57	0	11	33
116	6	14	2	51	32
117	6	58	0	19	57
118	6	29	1	43	49
119	6	59	0	27	46
120	6	7	3	59	48
121	6	60	0	35	33
122	6	30	1	51	25
123	6	61	0	43	14
124	6	15	2	59	33
125	6	62	0	51	14
126	6	31	1	59	14
127	6	63	0	59	7
128	7	0	7	3	28
129	7	64	0	3	53
130	7	32	1	3	53
131	7	65	0	11	42
132	7	16	2	3	53
133	7	66	0	19	42
134	7	33	1	11	50
135	7	67	0	27	11
136	7	8	3	3	53
137	7	68	0	35	42
138	7	34	1	19	58
139	7	69	0	43	11
140	7	17	2	11	26
141	7	70	0	51	19
142	7	35	1	27	35
143	7	71	0	59	40
144	7	4	4	35	53
145	7	72	0	3	10
146	7	36	1	35	10
147	7	73	0	11	11
148	7	18	2	19	42
149	7	74	0	19	59
150	7	37	1	43	19
151	7	75	0	27	8
152	7	9	3	11	34
153	7	76	0	35	59
154	7	38	1	51	35
155	7	77	0	43	48
156	7	19	2	27	27
157	7	78	0	51	0
158	7	39	1	59	32
159	7	79	0	59	17
160	7	2	5	19	5
161	7	80	0	3	42
162	7	40	1	3	26
163	7	81	0	11	59
164	7	20	2	35	10
165	7	82	0	19	11
166	7	41	1	11	51
167	7	83	0	27	8
168	7	10	3	19	42
169	7	84	0	35	59
170	7	42	1	19	59
171	7	85	0	43	8
172	7	21	2	43	11
173	7	86	0	51	32
174	7	43	1	27	16
175	7	87	0	59	33
176	7	5	4	43	10
177	7	88	0	3	35
178	7	44	1	35	27
179	7	89	0	11	24
180	7	22	2	51	19
181	7	90	0	19	32
182	7	45	1	43	24
183	7	91	0	27	57
184	7	11	3	27	51
185	7	92	0	35	56
186	7	46	1	51	16
187	7	93	0	43	41
188	7	23	2	59	56
189	7	94	0	51	9
190	7	47	1	59	9

Node	Level	mu	power	b	x
191	7	95	0	59	22
192	7	1	6	11	5
193	7	96	0	3	18
194	7	48	1	3	10
195	7	97	0	11	43
196	7	24	2	3	34
197	7	98	0	19	35
198	7	49	1	11	43
199	7	99	0	27	32
200	7	12	3	35	58
201	7	100	0	35	59
202	7	50	1	19	43
203	7	101	0	43	56
204	7	25	2	11	43
205	7	102	0	51	32
206	7	51	1	27	48
207	7	103	0	59	1
208	7	6	4	51	50
209	7	104	0	3	3
210	7	52	1	35	51
211	7	105	0	11	16
212	7	26	2	19	3
213	7	106	0	19	48
214	7	53	1	43	8
215	7	107	0	27	41
216	7	13	3	43	11
217	7	108	0	35	32
218	7	54	1	51	40
219	7	109	0	43	33
220	7	27	2	27	0
221	7	110	0	51	17
222	7	55	1	59	49
223	7	111	0	59	30
224	7	3	5	27	34
225	7	112	0	3	43
226	7	56	1	3	19
227	7	113	0	11	16
228	7	28	2	35	27
229	7	114	0	19	24
230	7	57	1	11	0
231	7	115	0	27	33
232	7	14	3	51	19
233	7	116	0	35	32
234	7	58	1	19	0
235	7	117	0	43	57
236	7	29	2	43	48
237	7	118	0	51	49
238	7	59	1	27	33
239	7	119	0	59	46
240	7	7	4	59	43
241	7	120	0	3	48
242	7	60	1	35	24
243	7	121	0	11	33
244	7	30	2	51	0
245	7	122	0	19	25
246	7	61	1	43	17
247	7	123	0	27	14
248	7	15	3	59	48
249	7	124	0	35	33
250	7	62	1	51	25
251	7	125	0	43	14
252	7	31	2	59	33
253	7	126	0	51	14
254	7	63	1	59	14
255	7	127	0	59	7

(b,	x)	at node (level)
(3,	10)	145 (7); 194 (7)
(3,	11)	16 (4); 33 (5), 34 (5)
(3,	35)	98 (6); 177 (7)
(3,	40)	8 (3); 17 (4)
(3,	42)	32 (5); 65 (6), 66 (6), 68 (6), 161 (7)
(3,	43)	97 (6); 225 (7)
(3,	53)	64 (6); 129 (7), 130 (7), 132 (7), 136 (7)
(11,	1)	12 (3); 51 (5)
(11,	11)	67 (6); 147 (7)
(11,	16)	211 (7); 227 (7)
(11,	32)	24 (4); 99 (6), 102 (6)
(11,	33)	115 (6); 243 (7)
(11,	43)	48 (5); 195 (7), 198 (7), 204 (7)
(11,	59)	76 (6); 163 (7)
(19,	8)	20 (4); 37 (5), 42 (5), 85 (6)
(19,	11)	69 (6); 165 (7)
(19,	33)	10 (3); 21 (4)
(19,	42)	80 (6); 133 (7), 148 (7), 168 (7)
(19,	48)	106 (6); 213 (7)
(19,	59)	40 (5); 74 (6), 84 (6), 149 (7), 170 (7)

```
(  b,     x)   at    node (level)
( 27,     0)           78 (6);   220 (7)
( 27,     8)          151 (7);   167 (7)
( 27,    16)           56 (5);   174 (7)
( 27,    17)           39 (5);   110 (6)
( 27,    33)           28 (4);    87 (6),   231 (7),   238 (7)
( 27,    41)           46 (5);   215 (7)
( 27,    46)           14 (3);   119 (6)
( 35,     8)           18 (4);    41 (5)
( 35,    10)           72 (6);   146 (7),   164 (7)
( 35,    11)           36 (5);    73 (6),    82 (6)
( 35,    24)           89 (6);   114 (6),   242 (7)
( 35,    27)          178 (7);   228 (7)
( 35,    32)          217 (7);   233 (7)
( 35,    33)           57 (5);   121 (6),   249 (7)
( 35,    56)           50 (5);   185 (7)
( 35,    59)          100 (6);   153 (7),   169 (7),   201 (7)
( 43,     8)           75 (6);   171 (7),   214 (7)
( 43,    11)          139 (7);   172 (7),   216 (7)
( 43,    14)          123 (6);   251 (7)
( 43,    24)           44 (5);   182 (7)
( 43,    32)           86 (6);   108 (6)
( 43,    33)           43 (5);    54 (5),   219 (7)
( 43,    41)          107 (6);   187 (7)
( 43,    48)          155 (7);   236 (7)
( 43,    57)           22 (4);    91 (6),   235 (7)
( 51,     0)          157 (7);   244 (7)
( 51,    14)           61 (5);   125 (6),   253 (7)
( 51,    16)           52 (5);   186 (7)
( 51,    19)          141 (7);   180 (7),   232 (7)
( 51,    25)          122 (6);   250 (7)
( 51,    32)           90 (6);   116 (6),   173 (7),   205 (7)
( 51,    41)           26 (4);    93 (6)
( 51,    57)           45 (5);    58 (5)
( 59,     7)           15 (3);    31 (4),    63 (5),   127 (6),   255 (7)
( 59,     9)           94 (6);   190 (7)
( 59,    14)           30 (4);    62 (5),   126 (6),   254 (7)
( 59,    17)           79 (6);   159 (7)
( 59,    22)           47 (5);    95 (6),   191 (7)
( 59,    30)          111 (6);   223 (7)
( 59,    33)           60 (5);   124 (6),   175 (7),   252 (7)
( 59,    48)          120 (6);   248 (7)
```

An example of the output of the program for Algorithm 2, for the same data as above, is given below.

```
M = 6, a = 21, b0 = 3, f0 = 7, q = 3, Q = 2^M = 64, 2^q = 8, 2^(M-q) = 8
a^ = 61, K0 = 50, K0* = 1, K1 = 41, K1*= 21, K2 = 57, K2* = 9
```

Node	Level	mu	power	b	x	s	ns	T	A	S	XXX
U	V	W	UU	VV	WW	X	Y	Z	XX	YY	ZZ
1	0	0	0	3	7	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	3	22	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	0	11	23	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
4	2	0	2	3	17	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
5	2	2	0	19	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
6	2	1	1	11	46	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
7	2	3	0	27	55	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
8	3	0	3	3	40	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
9	3	4	0	35	7	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
10	3	2	1	19	17	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
11	3	5	0	43	14	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
12	3	1	2	11	17	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
13	3	6	0	51	38	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
14	3	3	1	27	30	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
15	3	7	0	59	55	0	0	0	0	0	1
1	1	1	1	1	1	21	0	0	1	0	0
16	4	0	4	3	11	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
17	4	8	0	3	62	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1
18	4	4	1	35	54	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
19	4	9	0	11	31	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1
20	4	2	2	19	56	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
21	4	10	0	19	47	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1
22	4	5	1	43	17	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
23	4	11	0	27	6	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1
24	4	1	3	11	48	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
25	4	12	0	35	15	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1

Node	Level	mu	power	b	x	s	ns	T	A	S	XXX
U	V	W	UU	VV	WW	X	Y	Z	XX	YY	ZZ
26	4	6	1	51	17	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
27	4	13	0	43	54	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1
28	4	3	2	27	17	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
29	4	14	0	51	14	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1
30	4	7	1	59	62	1	1	0	0	0	22
57	21	21	9	61	61	57	22	1	41	1	1
31	4	15	0	59	63	1	1	8	33	56	22
57	21	21	9	61	61	57	22	1	41	1	1

Node	Level	mu	power	b	x	s	ns	T
32	5	0	5	3	42	2	2	0
33	5	16	0	3	61	2	2	13
34	5	8	1	3	25	2	2	0
35	5	17	0	11	10	2	2	13
36	5	4	2	35	17	2	2	0
37	5	18	0	19	2	2	2	13
38	5	9	1	11	22	2	2	0
39	5	19	0	27	29	2	2	13
40	5	2	3	19	43	2	2	0
41	5	20	0	35	50	2	2	13
42	5	10	1	19	46	2	2	0
43	5	21	0	43	29	2	2	13
44	5	5	2	43	16	2	2	0
45	5	22	0	51	61	2	2	13
46	5	11	1	27	25	2	2	0
47	5	23	0	59	26	2	2	13
48	5	1	4	11	59	3	2	0
49	5	24	0	3	38	3	2	17
50	5	12	1	35	30	3	2	0
51	5	25	0	11	33	3	2	17
52	5	6	2	51	24	3	2	0
53	5	26	0	19	33	3	2	17
54	5	13	1	43	25	3	2	0
55	5	27	0	27	46	3	2	17
56	5	3	3	27	0	3	2	0
57	5	28	0	35	33	3	2	17
58	5	14	1	51	25	3	2	0
59	5	29	0	43	30	3	2	17
60	5	7	2	59	17	3	2	0
61	5	30	0	51	54	3	2	17
62	5	15	1	59	38	3	2	0
63	5	31	0	59	33	3	2	17
64	6	0	6	3	53	4	3	0
65	6	32	0	3	58	4	3	21
66	6	16	1	3	4	4	3	0
67	6	33	0	11	37	4	3	21
68	6	8	2	3	16	4	3	0
69	6	34	0	19	5	4	3	21
70	6	17	1	11	29	4	3	0
71	6	35	0	27	34	4	3	21

Node	Level	mu	power	b	x	s	ns	T
72	6	4	3	35	8	4	3	0
73	6	36	0	35	5	4	3	21
74	6	18	1	19	61	4	3	0
75	6	37	0	43	18	4	3	21
76	6	9	2	11	25	4	3	0
77	6	38	0	51	10	4	3	21
78	6	19	1	27	60	4	3	0
79	6	39	0	59	37	4	3	21
80	6	2	4	19	26	5	3	0
81	6	40	0	3	31	5	3	24
82	6	20	1	35	61	5	3	0
83	6	41	0	11	38	5	3	24
84	6	10	2	19	25	5	3	0
85	6	42	0	19	62	5	3	24
86	6	21	1	43	12	5	3	0
87	6	43	0	27	15	5	3	24
88	6	5	3	43	59	5	3	0
89	6	44	0	35	46	5	3	24
90	6	22	1	51	52	5	3	0
91	6	45	0	43	47	5	3	24
92	6	11	2	27	40	5	3	0
93	6	46	0	51	63	5	3	24
94	6	23	1	59	29	5	3	0
95	6	47	0	59	54	5	3	24
96	6	1	5	11	34	6	3	0
97	6	48	0	3	35	6	3	27
98	6	24	1	3	33	6	3	0
99	6	49	0	11	40	6	3	27
100	6	12	2	35	25	6	3	0
101	6	50	0	19	48	6	3	27
102	6	25	1	11	0	6	3	0
103	6	51	0	27	51	6	3	27
104	6	6	3	51	43	6	3	0
105	6	52	0	35	0	6	3	27
106	6	26	1	19	8	6	3	0
107	6	53	0	43	19	6	3	27
108	6	13	2	43	56	6	3	0
109	6	54	0	51	3	6	3	27
110	6	27	1	27	33	6	3	0
111	6	55	0	59	24	6	3	27
112	6	3	4	27	27	7	3	0
113	6	56	0	3	20	7	3	30
114	6	28	1	35	24	7	3	0
115	6	57	0	11	45	7	3	30
116	6	14	2	51	0	7	3	0
117	6	58	0	19	13	7	3	30
118	6	29	1	43	33	7	3	0
119	6	59	0	27	12	7	3	30
120	6	7	3	59	32	7	3	0
121	6	60	0	35	13	7	3	30
122	6	30	1	51	33	7	3	0
123	6	61	0	43	28	7	3	30
124	6	15	2	59	25	7	3	0
125	6	62	0	51	4	7	3	30
126	6	31	1	59	48	7	3	0

Node	Level	mu	power	b	x	s	ns	T
127	6	63	0	59	45	7	3	30
128	7	0	7	3	28	8	4	0
129	7	64	0	3	49	8	4	33
130	7	32	1	3	5	8	4	0
131	7	65	0	11	14	8	4	33
132	7	16	2	3	23	8	4	0
133	7	66	0	19	38	8	4	33
134	7	33	1	11	20	8	4	0
135	7	67	0	27	49	8	4	33
136	7	8	3	3	19	8	4	0
137	7	68	0	35	22	8	4	33
138	7	34	1	19	60	8	4	0
139	7	69	0	43	49	8	4	33
140	7	17	2	11	44	8	4	0
141	7	70	0	51	49	8	4	33
142	7	35	1	27	37	8	4	0
143	7	71	0	59	30	8	4	33
144	7	4	4	35	11	9	4	0
145	7	72	0	3	8	9	4	35
146	7	36	1	35	12	9	4	0
147	7	73	0	11	27	9	4	35
148	7	18	2	19	20	9	4	0
149	7	74	0	19	11	9	4	35
150	7	37	1	43	37	9	4	0
151	7	75	0	27	32	9	4	35
152	7	9	3	11	24	9	4	0
153	7	76	0	35	43	9	4	35
154	7	38	1	51	5	9	4	0
155	7	77	0	43	48	9	4	35
156	7	19	2	27	7	9	4	0
157	7	78	0	51	56	9	4	35
158	7	39	1	59	4	9	4	0
159	7	79	0	59	59	9	4	35
160	7	2	5	19	53	10	4	0
161	7	80	0	3	10	10	4	37
162	7	40	1	3	14	10	4	0
163	7	81	0	11	53	10	4	37
164	7	20	2	35	36	10	4	0
165	7	82	0	19	21	10	4	37
166	7	41	1	11	41	10	4	0
167	7	83	0	27	50	10	4	37
168	7	10	3	19	32	10	4	0
169	7	84	0	35	21	10	4	37
170	7	42	1	19	41	10	4	0
171	7	85	0	43	34	10	4	37
172	7	21	2	43	39	10	4	0
173	7	86	0	51	26	10	4	37
174	7	43	1	27	22	10	4	0
175	7	87	0	59	53	10	4	37
176	7	5	4	43	2	11	4	0
177	7	88	0	3	47	11	4	39
178	7	44	1	35	41	11	4	0
179	7	89	0	11	36	11	4	39
180	7	22	2	51	55	11	4	0
181	7	90	0	19	12	11	4	39

Node	Level	mu	power	b	x	s	ns	T
182	7	45	1	43	6	11	4	0
183	7	91	0	27	31	11	4	39
184	7	11	3	27	35	11	4	0
185	7	92	0	35	28	11	4	39
186	7	46	1	51	30	11	4	0
187	7	93	0	43	63	11	4	39
188	7	23	2	59	28	11	4	0
189	7	94	0	51	15	11	4	39
190	7	47	1	59	41	11	4	0
191	7	95	0	59	20	11	4	39
192	7	1	6	11	21	12	4	0
193	7	96	0	3	30	12	4	41
194	7	48	1	3	34	12	4	0
195	7	97	0	11	57	12	4	41
196	7	24	2	3	56	12	4	0
197	7	98	0	19	57	12	4	41
198	7	49	1	11	19	12	4	0
199	7	99	0	27	38	12	4	41
200	7	12	3	35	48	12	4	0
201	7	100	0	35	57	12	4	41
202	7	50	1	19	3	12	4	0
203	7	101	0	43	22	12	4	41
204	7	25	2	11	11	12	4	0
205	7	102	0	51	46	12	4	41
206	7	51	1	27	10	12	4	0
207	7	103	0	59	57	12	4	41
208	7	6	4	51	58	13	4	0
209	7	104	0	3	51	13	4	43
210	7	52	1	35	35	13	4	0
211	7	105	0	11	56	13	4	43
212	7	26	2	19	59	13	4	0
213	7	106	0	19	0	13	4	43
214	7	53	1	43	58	13	4	0
215	7	107	0	27	3	13	4	43
216	7	13	3	43	3	13	4	0
217	7	108	0	35	16	13	4	43
218	7	54	1	51	50	13	4	0
219	7	109	0	43	35	13	4	43
220	7	27	2	27	16	13	4	0
221	7	110	0	51	19	13	4	43
222	7	55	1	59	51	13	4	0
223	7	111	0	59	40	13	4	43
224	7	3	5	27	18	14	4	0
225	7	112	0	3	29	14	4	45
226	7	56	1	3	39	14	4	0
227	7	113	0	11	42	14	4	45
228	7	28	2	35	27	14	4	0
229	7	114	0	19	34	14	4	45
230	7	57	1	11	60	14	4	0
231	7	115	0	27	61	14	4	45
232	7	14	3	51	51	14	4	0
233	7	116	0	35	18	14	4	45
234	7	58	1	19	36	14	4	0
235	7	117	0	43	61	14	4	45
236	7	29	2	43	32	14	4	0

Node	Level	mu	power	b	x	s	ns	T
237	7	118	0	51	29	14	4	45
238	7	59	1	27	23	14	4	0
239	7	119	0	59	58	14	4	45
240	7	7	4	59	27	15	4	0
241	7	120	0	3	36	15	4	47
242	7	60	1	35	52	15	4	0
243	7	121	0	11	7	15	4	47
244	7	30	2	51	40	15	4	0
245	7	122	0	19	23	15	4	47
246	7	61	1	43	55	15	4	0
247	7	123	0	27	28	15	4	47
248	7	15	3	59	8	15	4	0
249	7	124	0	35	55	15	4	47
250	7	62	1	51	7	15	4	0
251	7	125	0	43	44	15	4	47
252	7	31	2	59	43	15	4	0
253	7	126	0	51	20	15	4	47
254	7	63	1	59	44	15	4	0
255	7	127	0	59	39	15	4	47

APPENDIX B

Programs

The first program executes Algorithm 1, given by (141) - (143).

```
/**********************************************************************
***********************************************************************
             PROGRAM FOR ALGORITHM 1
***********************************************************************
**********************************************************************/

#include <stdio.h>
#include <math.h>

#define TMAX   256
#define TMAX2  TMAX / 2

long M, Q, a, b0, f0, q, qq, R, h, i, j, k, w,
     rep[10], space,
     level[TMAX], mu[TMAX],power[TMAX], bval[TMAX], xval[TMAX],
     BS[TMAX], XS[TMAX], NS[TMAX];

main()

  { long res();

    FILE *f, *fopen();

    f = fopen("tsrn", "w");

/**********************************************************************
             INPUT AND PREPROCESS PARAMETERS
**********************************************************************/

    printf("Type M a b0 f0 q:  ");
    scanf("%ld %ld %ld %ld %ld", &M, &a, &b0, &f0, &q);
    Q = 1; for (i = 0; i < M; i++) Q = Q * 2;
    qq = 1; for (i = 0; i < q; i++) qq = qq * 2;
    printf("\n  M     a    b0    f0     q   Q = 2^M\n");
    printf("%31d  %31d  %31d  %31d  %31d       %31d\n\n",
      M, a, b0, f0, q, Q);
    fprintf(f, "  M     a    b0    f0     q   Q = 2^M\n");
    fprintf(f, "%31d  %31d  %31d  %31d  %31d       %31d\n\n",
      M, a, b0, f0, q, Q);
```

```
/************************************************************************
            INITIALIZE ALL RECORD-ARRAYS
************************************************************************/

    level[1] = 0; mu[1] = 0; power[1] = 0;
    bval[1] = b0; xval[1] = f0;
    BS[1] = b0; XS[1] = f0; NS[1] = 1;
    space = 1; j = 1;

/************************************************************************
            BUILD THE TREE
************************************************************************/

    for (i = 1; i < TMAX2; i++)
      { j++;
        level[j] = level[i] + 1;
        mu[j] = mu[i];
        power[j] = power[i] + 1;
        bval[j] = bval[i];
        xval[j] = res(a * xval[i] + bval[i]);
        insert(j, bval[j], xval[j]);
        j++;
        level[j] = level[i] + 1;
        mu[j] = i;
        power[j] = 0;
        bval[j] = res(qq * i + b0);
        xval[j] = xval[i];
        insert(j, bval[j], xval[j]);
      }

/************************************************************************
            PRINT-OUT ALL NODES GENERATED
************************************************************************/

    fprintf(f, " Node  Level  mu   power   b     x\n\n");
    for (i = 1; i < TMAX; i++)
      { fprintf(f, "%4ld  %4ld  %4ld  %4ld  %4ld  %4ld\n",
          i, level[i], mu[i], power[i], bval[i], xval[i]);
      }
    fprintf(f, "\n\n");

/************************************************************************
            PRINT AND COUNT ALL REPETITIONS OF (b, x)
************************************************************************/

    R = 0; for (i = 0; i < 10; i++) rep[i] = 0;
    h = -1; j = -1; k = -1; w = 0;
    fprintf(f, "(  b,    x)  at   node (level)\n\n");
    for (i = 1; i < TMAX; i++)
      if (j == BS[i] && k == XS[i])
        if (w == 0)
          { fprintf(f, "(%3ld, %3ld)        ", j, k);
            fprintf(f, "%3ld (%1ld);  %3ld (%1ld)",
              h, level[h], NS[i], level[NS[i]]);
            R++; rep[level[NS[i]]]++; w = 1;
          }
        else fprintf(f, ",  %31d (%11d)", NS[i], level[NS[i]]);
      else
        { h = NS[i];
          j = BS[i];
          k = XS[i];
          if (w > 0)
            { fprintf(f, "\n");
              w = 0;
            }
        }
    fprintf(f, "\n\n");
    fprintf(f, "  M    a   b0   f0    q   Q = 2^M\n");
    fprintf(f, "%31d  %31d  %31d  %31d  %31d        %31d\n\n",
      M, a, b0, f0, q, Q);
    for (i = 0, j = 1; j <= TMAX2; i++, j = 2 * j)
      { printf("Level %11d:  %31d repetitions\n", i, rep[i]);
        fprintf(f, "Level %11d:  %31d repetitions\n", i, rep[i]);
      }
    printf("\n%31d repetitions in all\n\n", R);
    fprintf(f, "\n%31d repetitions in all\n\n", R);
    fclose(f);
  }

/**************************************************************************
            INSERT A NEW NODE-RECORD INTO THE REPETITIONS-LIST
**************************************************************************/

insert(n, p, q)

  long n, p, q;

    { long i, j, k;

      space++;
      k = space; i = 1;
      while (k == space && i < space)
      if (p < BS[i] || p == BS[i] && q < XS[i])
        { k = i;
          for (j = space; j > k; j--)
            { BS[j] = BS[j - 1];
              XS[j] = XS[j - 1];
              NS[j] = NS[j - 1];
            }
        }
      else i++;
      BS[k] = p; XS[k] = q; NS[k] = n;
    }

/**************************************************************************
            POSITIVE RESIDUE OF P MODULO Q
**************************************************************************/

long res(P)

  long P;
```

```
    { while (P < 0) P = P + Q;
      return(P % Q);

    }
```

The second program executes Algorithm 2, given by (170) - (190).

```
/**********************************************************************
***********************************************************************
            PROGRAM FOR ALGORITHM 2
***********************************************************************
**********************************************************************/

#include <stdio.h>
#include <math.h>

#define TMAX   256
#define TMAX2  TMAX / 2

long M, Q, Q0, Q1, QQ, a, b0, f0, q, qq, R, h, i, j, k, u, v, w,
     aa, b, x, y, z, rep[10], space, K0, KK0, K1, KK1, K2, KK2,
     level[TMAX], mu[TMAX],power[TMAX], bval[TMAX], xval[TMAX],
     BS[TMAX], XS[TMAX], NS[TMAX],
     U[TMAX], UU[TMAX], V[TMAX], VV[TMAX], W[TMAX], WW[TMAX],
     X[TMAX], XX[TMAX], Y[TMAX], YY[TMAX], Z[TMAX], ZZ[TMAX],
     XXX[TMAX], st[TMAX], nt[TMAX], T[TMAX],
     A[TMAX], S[TMAX];

main()

  { long res();

    FILE *f, *fopen();

    f = fopen("tsrn", "w");

/**********************************************************************
            INPUT AND PREPROCESS PARAMETERS
**********************************************************************/

    printf("Type M a b0 f0 q:  ");
    scanf("%ld %ld %ld %ld %ld", &M, &a, &b0, &f0, &q);
    QQ = 1; for (i = q + 1; i < M; i++) QQ = QQ * 2;
    Q0 = QQ * 2; Q1 = Q0 * 2; qq = 1;
    for (i = 0; i < q; i++) qq = qq * 2;
    Q = qq * Q0;
    printf("\n  M     a    b0    f0     q   Q = 2^M  2^q  2^(M-q)\n");
    printf("%31d  %31d  %31d  %31d  %31d       %31d  %31d  %31d\n\n",
      M, a, b0, f0, q, Q, qq, Q0);
    fprintf(f, "  M     a    b0    f0     q   Q = 2^M  2^q  2^(M-q)\n");
    fprintf(f,
     "%31d  %31d  %31d  %31d  %31d       %31d  %31d  %31d\n\n",
      M, a, b0, f0, q, Q, qq, Q0);
    KK0 = 0; u = M - q - 2; v = 1;
    for (i = 0; i < u; i++)
      { KK0 = res(KK0 + v);
        v = res(v * a);
      }
    KK1 = v; K0 = KK0;
    for (i = u; i < M; i++)
      { K0 = res(K0 + v);
        v = res(v * a);
      }
    K1 = v; K2 = res(a * a);
    aa = a; u = a;
    for (i = 3; i < M; i++)
      { u = res(u * u);
        aa = res(aa * u);
      }
    KK2 = aa; KK2 = res(KK2 * KK2);
    printf(" a^   K0   K0*  K1   K1*  K2   K2*\n");
    printf("%3ld  %3ld  %3ld  %3ld  %3ld  %3ld  %3ld\n\n",
      aa, K0, KK0, K1, KK1, K2, KK2);
    fprintf(f, " a^   K0   K0*  K1   K1*  K2   K2*\n");
    fprintf(f, "%3ld  %3ld  %3ld  %3ld  %3ld  %3ld  %3ld\n\n",
      aa, K0, KK0, K1, KK1, K2, KK2);

/***********************************************************************
            INITIALIZE ALL RECORD-ARRAYS
***********************************************************************/

    level[1] = 0; mu[1] = 0; power[1] = 0;
    bval[1] = b0; xval[1] = f0;
    BS[1] = b0; XS[1] = f0; NS[1] = 1;
    space = 1; j = 1;
    for (i = Q0; i < Q1; i++)
      { U[i] = 1; V[i] = 1; W[i] = 1; UU[i] = 1; VV[i] = 1; WW[i] = 1;
        X[i] = a; Y[i] = 0; Z[i] = 0; XX[i] = 1; YY[i] = 0; ZZ[i] = 0;
        st[i] = 0; nt[i] = 0; XXX[i] = 1;
      }

/***********************************************************************
            BUILD THE APEX OF THE TREE
***********************************************************************/

    for (i = 1; i < Q0; i++)
      { u = qq * i;
        b = res(u + b0);
        x = res(u * 2 + f0);
        j++;
        level[j] = level[i] + 1;
        mu[j] = mu[i];
        power[j] = power[i] + 1;
        bval[j] = bval[i];
        xval[j] = res(a * xval[i] + bval[i]);
        insert(j, bval[j], xval[j]);
        j++;
        level[j] = level[i] + 1;
        mu[j] = i;
        power[j] = 0;
        bval[j] = b;
        xval[j] = x;
        if ((xval[i] + x) % 2 == 1)
             xval[j] = res(a * x + b);
        else xval[j] = x;
        insert(j, bval[j], xval[j]);
      }

/************************************************************************
            BUILD THE REST OF THE TREE
************************************************************************/

    for (i = Q0; i < TMAX2; i++)
      { z = i % Q1;
        y = z / Q0;
        u = res(qq * i);
        b = res(u + b0);
        x = res(u * 2 + f0);
        w = res(W[i] * W[i]);
        z = res(XX[i] * XX[i]);
        j++;
        st[j] = 2 * st[i] + y;
        nt[j] = nt[i] + 1;
        level[j] = level[i] + 1;
        mu[j] = mu[i];
        power[j] = power[i] + 1;
        bval[j] = bval[i];
        U[j] = res(U[i] * U[i]);
        V[j] = res(V[i] * a);
        W[j] = res(w * U[i]);
        UU[j] = res(UU[i] * UU[i]);
        VV[j] = res(VV[i] * aa);
        WW[j] = res(WW[i] * WW[i] * UU[i]);
        X[j] = res(X[i] * X[i]);
        if (y == 1) Y[j] = res(Y[i] + U[i]);
        else        Y[j] = Y[i];
        if (y == 1) u = res(U[i] * a);
        else        u = U[i];
        Y[j] = res((1 + u) * Y[j]);
        Z[j] = res(Z[i] + V[i]);
        XX[j] = z;
        YY[j] = res((1 + XX[i]) * YY[i]);
        if (y == 1) u = res(Y[i] + U[i] * (Z[i] + V[i]));
        else        u = Y[i];
        ZZ[j] = res((1 + W[i]) * ZZ[i] + w * u);
        XXX[j] = res((1 + X[i]) * XXX[i]);
        if (y == 1)
          { U[j] = res(U[j] * K2);
            W[j] = res(W[j] * V[j]);
            UU[j] = res(UU[j] * KK2);
            WW[j] = res(WW[j] * VV[j]);
            XX[j] = res(XX[j] * K1);
            YY[j] = res(YY[j] + z);
          }
        xval[j] = res(a * xval[i] + bval[i]);
        insert(j, bval[j], xval[j]);
        j++;
        st[j] = st[j - 1];
        nt[j] = nt[j - 1];
        level[j] = level[j - 1];
        mu[j] = i;
        power[j] = 0;
        bval[j] = b;
        U[j] = U[j - 1];
        V[j] = V[j - 1];
        W[j] = W[j - 1];
        UU[j] = UU[j - 1];
        VV[j] = VV[j - 1];
        WW[j] = WW[j - 1];
        X[j] = X[j - 1];
        Y[j] = Y[j - 1];
        Z[j] = Z[j - 1];
        XX[j] = XX[j - 1];
        YY[j] = YY[j - 1];
        ZZ[j] = ZZ[j - 1];
        XXX[j] = XXX[j - 1];
        A[j] = res(KK1 * XX[j] * WW[j] * X[j]);
        S[j] = res(K0 * YY[j] - XX[j] * WW[j]
          * (ZZ[j] - XXX[j] - X[j] * KK0));
        xval[j] = res(A[j] * x + S[j] * b);
        if ((xval[i] + xval[j]) % 2 == 1)
          xval[j] = res(a * xval[j] + b);
        insert(j, bval[j], xval[j]);

/*************************************************************************
          VERIFY THE 13 COEFFICIENT VALUES BY DIRECT EVALUATION
*************************************************************************/

        k = 1; v = 0; w = 1; u = st[j];
        for (h = 0; h < u; h++)
          { v = res(v + w);
            w = res(w * K1);
            k = res(k * KK2);
          }
        if (YY[j] != v)
          printf("Node %4ld: YY-values: %4ld, %4ld\n", j, YY[j], v);
        if (XX[j] != w)
          printf("Node %4ld: XX-values: %4ld, %4ld\n", j, XX[j], w);
        if (UU[j] != k)
          printf("Node %4ld: UU-values: %4ld, %4ld\n", j, UU[j], k);
        v = 0; w = 1; u = 2 * st[j];
        for (h = 0; h < u; h++)
          { v = res(v + w);
            w = res(w * a);
          }
        if (Y[j] != v)
          printf("Node %4ld: Y-values: %4ld, %4ld\n", j, Y[j], v);
        if (U[j] != w)
          printf("Node %4ld: U-values: %4ld, %4ld\n", j, U[j], w);
        k = 1; v = 0; w = 1; u = nt[j];
        for (h = 0; h < u; h++)
          { v = res(v + w);
```

```
            w = res(w * a);
            k = res(k * aa);
          }
        if (Z[j] != v)
          printf("Node %4ld: Z-values: %4ld, %4ld\n", j, Z[j], v);
        if (V[j] != w)
          printf("Node %4ld: V-values: %4ld, %4ld\n", j, V[j], w);
        if (VV[j] != k)
          printf("Node %4ld: VV-values: %4ld, %4ld\n", j, VV[j], k);
        k = 1; v = 0; w = 1; u = st[j] * nt[j];
        for (h = 0; h < u; h++)
          { v = res(v + w);
            w = res(w * a);
            k = res(k * aa);
          }
        if (ZZ[j] != v)
          printf("Node %4ld: ZZ-values: %4ld, %4ld\n", j, ZZ[j], v);
        if (W[j] != w)
          printf("Node %4ld: W-values: %4ld, %4ld\n", j, W[j], w);
        if (WW[j] != k)
          printf("Node %4ld: WW-values: %4ld, %4ld\n", j, WW[j], k);
        v = 0; w = 1; u = 1;
        for (h = 0; h < nt[j]; h++) u = u * 2;
        for (h = 0; h < u; h++)
          { v = res(v + w);
            w = res(w * a);
          }
        if (XXX[j] != v)
          printf("Node %4ld: XXX-values: %4ld, %4ld\n", j, XXX[j], v);
        if (X[j] != w)
          printf("Node %4ld: X-values: %4ld, %4ld\n", j, X[j], w);

/***********************************************************************
          VERIFY THE As AND Ss PARAMETERS BY DIRECT EVALUATION FROM Ts
***********************************************************************/

        u = 1; for (h = 0; h < nt[j]; h++) u = u * 2;
        T[j] = (st[j] + 1) * M - nt[j] * st[j] + u - q - 2;
        u = T[j]; v = 0; w = 1;
        for (h = 0; h < u; h++)
          { v = res(v + w);
            w = res(w * a);
          }
        if (A[j] != w)
          printf("Node %4ld: A-values: %4ld, %4ld\n", j, A[j], w);
        if (S[j] != v)
          printf("Node %4ld: S-values: %4ld, %4ld\n", j, S[j], v);
      }

/***********************************************************************
          PRINT-OUT ALL NODES GENERATED
***********************************************************************/

    fprintf(f, " Node  Level  mu   power   b     x");
    fprintf(f, "   s    ns     T     A     S    XXX\n");
    fprintf(f, "   U     V     W    UU    VV    WW");
    fprintf(f, "   X      Y      Z    XX    YY    ZZ\n\n");
    for (i = 1; i < TMAX; i++)
      { fprintf(f, "%4ld  %4ld  %4ld  %4ld  %4ld  %4ld",
          i, level[i], mu[i], power[i], bval[i], xval[i]);
        fprintf(f, "%4ld  %4ld  %4ld  %4ld  %4ld  %4ld\n",
          st[i], nt[i], T[i], A[i], S[i], XXX[i]);
          fprintf(f, "%4ld  %4ld  %4ld  %4ld  %4ld  %4ld",
          U[i], V[i], W[i], UU[i], VV[i], WW[i]);
        fprintf(f, "%4ld  %4ld  %4ld  %4ld  %4ld  %4ld\n\n",
          X[i], Y[i], Z[i], XX[i], YY[i], ZZ[i]);
      }
    fprintf(f, "\n\n\n");

/***********************************************************************
          PRINT AND COUNT ALL REPETITIONS OF (b, x)
***********************************************************************/

    R = 0; for (i = 0; i < 10; i++) rep[i] = 0;
    h = -1; j = -1; k = -1; w = 0;
    fprintf(f, "(  b,   x)  at   node (level)\n\n");
    for (i = 1; i < TMAX; i++)
      if (j == BS[i] && k == XS[i])
        if (w == 0)
          { fprintf(f, "(%3ld, %3ld)        ", j, k);
            fprintf(f, "%3ld (%1ld);  %3ld (%1ld)",
              h, level[h], NS[i], level[NS[i]]);
            R++; rep[level[NS[i]]]++; w = 1;
          }
        else fprintf(f, ",  %3ld (%1ld)", NS[i], level[NS[i]]);
      else
        { h = NS[i];
          j = BS[i];
          k = XS[i];
          if (w > 0)
            { fprintf(f, "\n");
              w = 0;
            }
        }
    fprintf(f, "\n\n");
    fprintf(f, " M    a   b0   f0    q   Q = 2^M  2^q  2^(M-q)\n");
    fprintf(f,
      "%3ld  %3ld  %3ld  %3ld  %3ld       %3ld  %3ld  %3ld\n\n",
      M, a, b0, f0, q, Q, qq, Q0);
    for (i = 0, j = 1; j <= TMAX2; i++, j = 2 * j)
      { printf("Level %1ld:  %3ld repetitions\n", i, rep[i]);
        fprintf(f, "Level %1ld:  %3ld repetitions\n", i, rep[i]);
      }
    printf("\n%3ld repetitions in all\n\n", R);
    fprintf(f, "\n%3ld repetitions in all\n\n", R);
    fclose(f);
  }

/***********************************************************************
          INSERT A NEW NODE-RECORD INTO THE REPETITIONS-LIST
***********************************************************************/
```

```
insert(n, p, q)

  long n, p, q;

    { long i, j, k;

      space++;
      k = space; i = 1;
      while (k == space && i < space)
      if (p < BS[i] || p == BS[i] && q < XS[i])
        { k = i;
          for (j = space; j > k; j--)
            { BS[j] = BS[j - 1];
              XS[j] = XS[j - 1];
              NS[j] = NS[j - 1];
            }
        }
      else i++;
      BS[k] = p; XS[k] = q; NS[k] = n;
    }

/**********************************************************************
            POSITIVE RESIDUE OF P MODULO Q
**********************************************************************/

long res(P)

  long P;

    { while (P < 0) P = P + Q;
      return(P % Q);

    }
```