THIRD BASE

Brian Hayes

A reprint from

American Scientist

the magazine of Sigma Xi, the Scientific Research Society

Volume 89, Number 6
November–December, 2001
pages 490–494

THIRD BASE

Brian Hayes

People count by tens and machines count by twos—that pretty much sums up the way we do arithmetic on this planet. But there are countless other ways to count. Here I want to offer three cheers for base 3, the ternary system. The numerals in this sequence—beginning 0, 1, 2, 10, 11, 12, 20, 21, 22, 100, 101—are not as widely known or widely used as their decimal and binary cousins, but they have charms all their own. They are the Goldilocks choice among numbering systems: When base 2 is too small and base 10 is too big, base 3 is just right.

Cheaper by the Threesome

Under the skin, numbering systems are all alike. Numerals in various bases may well look different, but the numbers they represent are the same. In decimal notation, the numeral 19 is shorthand for this expression:

$$1 \times 10^1 + 9 \times 10^0.$$

Likewise the binary numeral 10011 is understood to mean:

$$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0,$$

which adds up to the same value. So does the ternary version, 201:

$$2 \times 3^2 + 0 \times 3^1 + 1 \times 3^0.$$

The general formula for a numeral in any positional notation goes something like this:

$$\ldots d_3 r^3 + d_2 r^2 + d_1 r^1 + d_0 r^0 \ldots.$$

Here r is the base, or *radix*, and the coefficients d_i are the digits of the number. Usually, r is a positive integer and the digits are integers in the range from 0 to $r-1$, but neither of these restrictions is strictly necessary. (You can build perfectly good numbers on a negative or an irrational base, and below we'll meet numbers with negative digits.)

To say that all bases represent the same numbers, however, is not to say that all numeric representations are equally good for all purposes. Base 10 is famously well suited to those of us who count on our fingers. Base 2 dominates computing technology because binary devices are simple and reliable, with just two stable states—on or off, full or empty. Computer circuitry also exploits a coincidence between binary arithmetic and binary logic: The same signal can represent either a numeric value (1 or 0) or a logical value (*true* or *false*).

The cultural preference for base 10 and the engineering advantages of base 2 have nothing to do with any intrinsic properties of the decimal and binary numbering systems. Base 3, on the other hand, does have a genuine mathematical distinction in its favor. By one plausible measure, it is the most efficient of all integer bases; it offers the most economical way of representing numbers.

How do you measure the cost of a numeric representation? If you simply count digits, then the biggest base will always win; for example, base 1,000,000 can represent any number between 0 and decimal 999,999 in a single digit. The trouble is, that single digit can be any of a million different symbols, all of which you must somehow recognize. At the opposite pole are unary, or base-1, numbers. The unary representation of decimal 1,000,000 needs only one type of symbol, but that symbol is repeated a million times. (Unary notation is in a category apart from other bases—it's not really a positional number system—but in the present context it serves as a useful limiting case.)

Among all possible ways of writing the numbers up to a million, neither base 1,000,000 nor base 1 seems ideal; as a matter of fact, you could hardly do worse than either of these choices. Minimizing the number of digits causes an explosion in the alphabet of symbols, and vice versa; when you squish down one factor, the other squirts out. Evidently we need to optimize some joint measure of a number's width (how many digits it has) and its depth (how many different symbols can occupy each digit position). An obvious strategy is to minimize the product of these two quantities. In other words, if r is the radix and w is the width in digits, we want to minimize rw while holding r^w constant.

Curiously, this problem is easier to solve if r and w are treated as continuous rather than integer variables—that is, if we allow a fractional base and a fractional number of digits. Then it turns out (*see Figure 1*) that the optimum radix is

Brian Hayes is Senior Writer for American Scientist. *Address: 211 Dacian Avenue, Durham, NC 27701; bhayes@amsci.org*

e, the base of the natural logarithms, with a numerical value of about 2.718. Because 3 is the integer closest to *e*, it is almost always the most economical integer radix (*see Figure 2*).

Consider again the task of representing all numbers from 0 through decimal 999,999. In base 10 this obviously requires a width of six digits, so that $rw = 60$. Binary does better: 20 binary digits suffice to cover the same range of numbers, for $rw = 40$. But ternary is better still: The ternary representation has a width of 13 digits, so that $rw = 39$. (If base e were a practical choice, the width would be 14 digits, yielding $rw = 38.056$.)

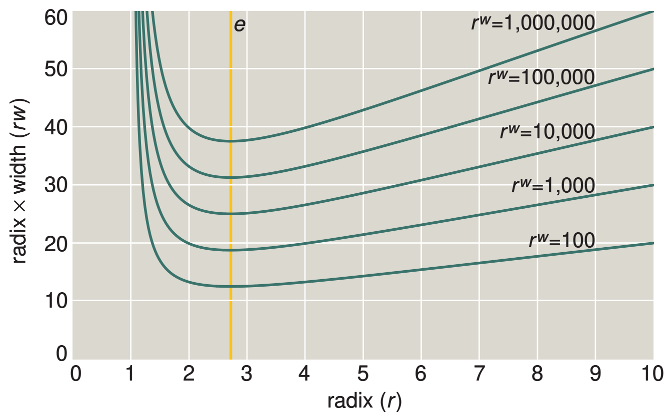

Figure 1. Most economical radix for a numbering system is e (about 2.718) when economy is measured as the product of the radix and the width, or number of digits, needed to express a given range of values. Here both the radix and the width are treated as continuous variables.

Trit by Trit by Trit

This special property of base 3 attracted the notice of early computer designers. On the hypothesis that a computer's component count would be roughly proportional both to the width and to the depth of the numbers being processed, they suggested that rw might be a good predictor of hardware cost, and so ternary notation would make the most efficient use of hardware resources. The earliest published discussion of this idea I've been able to find appears in the 1950 book *High-speed Computing Devices*, a survey of computer technologies compiled on behalf of the U.S. Navy by the staff of Engineering Research Associates.

At about the same time as the ERA survey, Herbert R. J. Grosch proposed a ternary architecture for the Whirlwind computer project at MIT. Whirlwind evolved into the control system for a military radar network, which stood vigil over North American airspace through 30 years of the Cold War. Whirlwind was also the proving ground for several novel computer technologies—including magnetic core memory—but ternary arithmetic was not among the innovations tested; Whirlwind and its successors were binary machines.

As it happens, the first working ternary computer was built on the other side of the Iron Curtain. The machine was designed by Nikolai P. Brusentsov and his colleagues at Moscow State University and was named Setun, for a river that flows near the university campus. Some 50 machines were built between 1958 and 1965. Setun operated on numbers composed of 18 ternary digits, or *trits*, giving the machine a numerical range of 387,420,489. A binary computer would need 29 bits to reach this capacity; in terms of rw, the ternary design wins 54 to 58.

Unfortunately, Setun did not realize the potential of base 3 to reduce component counts. Each trit was stored in a pair of magnetic cores, wired in tandem so that they had three stable states. A pair of cores could have held two binary bits, which amounts to more information than a single trit, and so the ternary advantage was squandered.

Along with ternary arithmetic, a computer built of base-3 hardware can also exploit ternary logic. Consider the task of comparing two numbers. In a machine based on binary logic, comparison is often a two-stage process. First you ask, "Is x less than y?"; depending on the answer, you may then have to ask a second question, such as "Is x equal to y?" Ternary logic simplifies the process: A single comparison can yield any of three possible outcomes: "less," "equal" and "greater."

Ternary computers were a fad that faded, though not quickly. In the 1960s there were several more projects to build ternary logic gates and memory cells, and to assemble these units into larger components such as adders. In 1973 Gideon Frieder and his colleagues at the State University of New York at Buffalo designed a complete base-3 machine they called TERNAC, and created a software emulator of it. Since then the idea of ternary computing has had occasional revivals, but you're not going to find a ternary minitower in stock at CompUSA.

Why did base 3 fail to catch on? One easy guess is that reliable three-state devices just didn't exist or were too hard to develop. And once binary technology became established, the tremendous investment in methods for fabricating binary chips would have overwhelmed any small theoretical advantage of other bases. Furthermore, it's

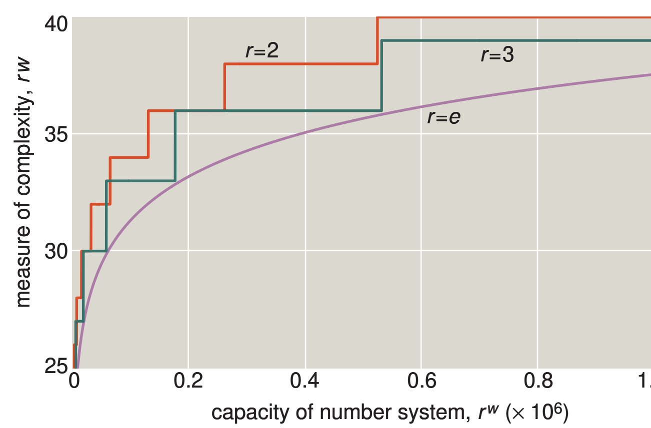

Figure 2. Most economical integer radix is almost always 3, the integer closest to e. If the capacity of a numbering system is r^w, and the cost of a representation is rw, then $r=3$ is the best integer radix for all but a finite set of capacities. Specifically, ternary is inferior to binary only for 8,487 values of r^w; ternary is superior for infinitely many values.

only a hypothesis that such an advantage exists. Everything hinges on the assumption that rw is a proper measure of hardware complexity, or in other words that the incremental cost of increasing the radix is the same as the incremental cost of increasing the number of digits.

But even if ternary circuits don't find a home in computer hardware, the Goldilocks argument favoring base 3 may apply in other contexts. Suppose you are creating one of those dreadful telephone menu systems—Press 1 to be inconvenienced, Press 2 to be condescended to, and so forth. If there are many choices, what is the best way to organize them? Should you build a deep hierarchy with lots of little menus that each offer just a few options? Or is it better to flatten the structure into a few long menus? In this situation a reasonable goal is to minimize the number of options that the wretched caller must listen to before finally reaching his or her destination. The problem is analogous to that of representing an integer in positional notation: The number of items per menu corresponds to the radix r, and the number of menus is analogous to the width w. The average number of choices to be endured is minimized when there are three items per menu.

Turning to Ternary Dust

Although numbers are the same in all bases, some properties of numbers show through most clearly in certain representations. For example, you can see at a glance whether a binary number is even or odd: Just look at the last digit. Ternary also distinguishes between even and odd, but the signal is subtler: A ternary numeral represents an even number if the numeral has an even number of 1s. (The reason is easy to see when you count powers of 3, which are invariably odd.)

More than 20 years ago, Paul Erdős and Ronald L. Graham published a conjecture about the ternary representation of powers of 2. They observed that 2^2 and 2^8 can be written in ternary without any 2s (the ternary numerals are 11 and 100111 respectively). But every other positive power of 2 seems to have at least one 2 in its ternary expansion; in other words, no other power

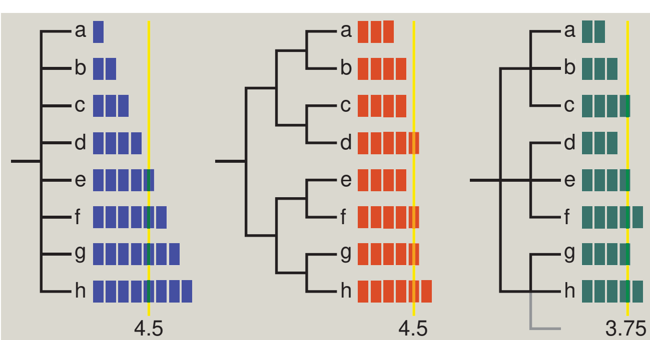

Figure 3. Ternary structure may offer the quickest path through a telephone menu system. Putting eight choices (assumed to be equally likely) in a single octonary menu *(left)* forces the caller to listen to 4.5 menu items on average. A binary structure *(middle)* has the same performance, but the ternary tree *(right)* reduces the average to 3.75.

of 2 is a simple sum of powers of 3. Ilan Vardi of the Institut des hautes études scientifiques has searched up to $2^{6973568802}$ without finding a counterexample, but the conjecture remains open.

The digits of ternary numerals can also help illuminate a peculiar mathematical object called the Cantor set, or Cantor's dust. To construct this set, draw a line segment and erase the middle third; then turn to each of the resulting shorter segments and remove the middle third of those also, and continue in the same way. After infinitely many middle thirds have been erased, does anything remain? One way to answer this question is to label the points of the original line as ternary numbers between 0 and 0.222.… (The repeating ternary fraction 0.222… is exactly equal to 1.0.) Given this labeling, the first middle third to be erased consists of those points with coordinates between 0.1 and 0.122…, or in other words all coordinates with a 1 in the first position after the radix point. Likewise the second round of erasures eliminates all points with a 1 in the second position after the radix point. The pattern continues, and the limiting set consists of points that have no 1s anywhere in their ternary representation. In the end, almost all the points have been wiped out, and yet an infinity of points remain. No two points are connected by a continuous line, but every point has neighbors arbitrarily close at hand. It's hard to form a mental image of such an infinitely perforated object, but the ternary description is straightforward.

The Jewel in the Triple Crown

"Perhaps the prettiest number system of all," writes Donald E. Knuth in *The Art of Computer Programming*, "is the balanced ternary notation." As in ordinary ternary numbers, the digits of a balanced ternary numeral are coefficients of powers of 3, but instead of coming from the set {0, 1, 2}, the digits are –1, 0 and 1. They are "balanced" because they are arranged symmetrically about zero. For notational convenience the negative digits are usually written with a vinculum, or overbar, instead of a prefixed minus sign, thus: $\bar{1}$.

As an example, the decimal number 19 is written $1\bar{1}01$ in balanced ternary, and this numeral is interpreted as follows:

$$1 \times 3^3 - 1 \times 3^2 + 0 \times 3^1 + 1 \times 3^0,$$

or in other words 27 – 9 + 0 + 1. Every number, both positive and negative, can be represented in this scheme, and each number has only one such representation. The balanced ternary counting sequence begins: 0, 1, $1\bar{1}$, 10, 11, $1\bar{1}\bar{1}$, $1\bar{1}0$, $1\bar{1}1$. Going in the opposite direction, the first few negative numbers are $\bar{1}$, $\bar{1}1$, $\bar{1}0$, $\bar{1}\bar{1}$, $\bar{1}11$, $\bar{1}10$, $\bar{1}1\bar{1}$. Note that negative values are easy to recognize because the leading trit is always negative.

The idea of balanced number systems has quite a tangled history. Both the Setun machine and the Frieder emulator were based on balanced ternary, and so was Grosch's proposal for the Whirlwind project. In 1950, Claude E. Shannon published an account of symmetrical signed-digit systems, including ternary and other bases. But none of these 20th-century inventors was the first. In 1840, Augustin Cauchy discussed signed-digit numbers in various bases, and Léon Lalanne immediately followed up with a discourse on the special virtues of balanced ternary. Twenty years earlier, John Leslie's remarkable *Philosophy of Arithmetic* had set forth methods of calculating in any base with either signed or unsigned digits. Leslie in turn was anticipated a century earlier by John Colson's brief essay on "negativo-affirmative arithmetick." Earlier still, Johannes Kepler used a balanced-ternary scheme modeled on Roman numerals. There is even a suggestion that signed-digit arithmetic was already implicit in the Hindu Vedas, which would make the idea very old indeed!

What makes balanced ternary so pretty? It is a notation in which everything seems easy. Positive and negative numbers are united in one system, without the bother of separate sign bits. Arithmetic is nearly as simple as it is with binary numbers; in particular, the multiplication table is trivial. Addition and subtraction are essentially the same operation: Just negate one number and then add. Negation itself is also effortless: Change every $\overline{1}$ into a 1, and vice versa. Rounding is mere truncation: Setting the least-significant trits to 0 automatically rounds to the closest power of 3.

The best-known application of balanced ternary notation is in mathematical puzzles that have to do with weighing. Given a two-pan balance, you are asked to weigh a coin known to have some integral weight between 1 gram and 40 grams. How many measuring weights do you need? A hasty answer would be six weights of 1, 2, 4, 8, 16 and 32 grams. If the coin must go in one pan and all the measuring weights in the other, you can't do better than such a powers-of-2 solution. If the weights can go in either pan, however, there's a ternary trick that works with just four weights: 1, 3, 9 and 27 grams. For instance, a coin of 35 grams—$110\overline{1}$ in signed ternary—will balance on the scale when weights of 27 grams and 9 grams are placed in the pan opposite the coin and a weight of 1 gram lies in the same pan as the coin. Every coin up to 40 grams can be weighed in this way. (So can all helium balloons weighing no less than –40 grams.)

James Allwright, who maintains a Web site promoting balanced ternary notation, suggests a monetary system based on the same principle. If both a merchant and a customer have just one bill or coin in each power-of-3 denomination, they can make exact change for any transaction.

Martha Stewart's File Cabinet

Some weeks ago, rooting around in files of old clippings and correspondence, I made a discovery of astonishing obviousness and triviality. What I found had nothing to do with the content of the files; it was about their arrangement in the drawer.

Imagine a fastidious office worker—a Martha Stewart of filing—who insists that no file folder lurk in the shadow of another. The protruding tabs on the folders must be arranged so that adjacent folders always have tabs in different positions. Achieving this staggered arrangement is easy if you're setting up a new file, but it gets messy when folders are added or deleted at random.

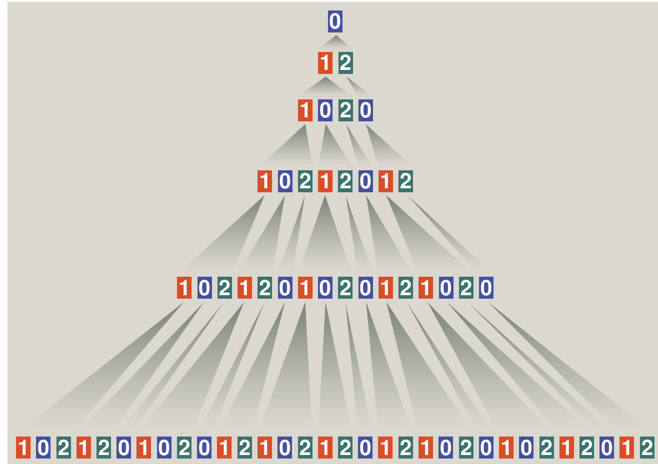

Figure 4. Ternary sequence devised by Axel Thue can be extended indefinitely without ever generating two adjacent identical subsequences of any length. No binary sequence has this property. The sequence is defined by three replacement rules: 0 → 12, 1 → 102, 2 → 0.

A drawer filled with "half-cut" folders, which have just two tab positions, might initially alternate *left-right-left-right*. The pattern is spoiled, however, as soon as you insert a folder in the middle of the drawer. No matter which type of folder you choose and no matter where you put it (except at the very ends of the sequence), every such insertion generates a conflict. Removing a folder has the same effect. Translated into a binary numeral with *left* = 0 and *right* = 1, the pristine file is the alternating sequence …0101010101.… An insertion or deletion creates either a 00 or a 11—a flaw much like a dislocation in a crystal. Although in principle the flaw could be repaired—either by introducing a second flaw of the opposite polarity or by flipping all the bits between the site of the flaw and the end of the sequence—even the most maniacally tidy record-keeper is unlikely to adopt such practices in a real file drawer.

In my own files I use third-cut rather than half-cut folders; the tabs appear in three positions, *left*, *middle* and *right*. Nevertheless, I had long thought—or rather I had assumed without bothering to think—that a similar analysis would apply, and that I couldn't be sure of avoiding conflicts between adjacent folders unless I was willing to shift files to new folders after every insertion. Then came my Epiphany of the File Cabinet a few weeks ago: Suddenly I understood that going from half-cut to third-cut folders makes all the difference.

It's easy to see why; just interpret the drawerful of third-cut folders as a sequence of ternary digits. At any position in any such sequence, you can always insert a new digit that differs from both of its neighbors. Base 3 is the smallest base that has this property. Moreover, if you build up a ternary sequence by consistently inserting digits that avoid conflicts, then the choice of which symbol to insert is always a forced one; you never have to make an arbitrary selection among two or more legal possibilities. Thus, as a file drawer fills up, it is not only possible to maintain perfect Martha Stewart order; it's actually quite easy.

Deletions, regrettably, are more troublesome than insertions. There is no way to remove arbitrary elements from either a binary or a ternary sequence with a guarantee that two identical digits won't be brought together. (On the other hand, if you're fussy enough to fret about the positions of tabs on file folders, you probably never throw anything away anyhow.)

The protocol for avoiding conflicts between third-cut file folders is so obvious that I assume it must be known to file clerks everywhere. But in half a dozen textbooks on filing—admittedly a small sample of a surprisingly extensive literature—I found no clear statement of the principle.

Strangely enough, my trifling observation about arranging folders in file drawers leads to some mathematics of wider interest. Suppose you seek an arrangement of folders in which you not only avoid putting any two identical tabs next to each other, but you also avoid repeating any longer patterns. This would rule out not only 00 and 11 but also 0101 and 021021. Sequences that have no adjacent repeated patterns of any length are said to be "square free," by analogy to numbers that have no duplicated prime factors.

In binary notation, the one-digit sequences 0 and 1 are obviously square free, and so are 01 and 10 (but not 00 or 11); then among sequences three bits long there are 010 and 101, but none of the other six possibilities is square free. If you now try to create a four-digit square-free binary sequence, you'll find that you're stuck. No such sequences exist.

What about square-free ternary sequences? Try to grow one digit by digit, and you're likely to find your path blocked at some point. For example, you might stumble onto the sequence 0102010, which is square free but cannot be extended without creating a square. Many other ternary sequences also lead to such dead ends. Nevertheless, the Norwegian mathematician Axel Thue proved almost a century ago that unbounded square-free ternary sequences exist, and he gave a method for constructing one. The heart of the algorithm is a set of digit replacement rules: $0 \rightarrow 12$, $1 \rightarrow 102$, $2 \rightarrow 0$. At each stage in the construction of the sequence, the appropriate rule is applied to each digit, and the result becomes the starting point for the next stage. Figure 4 shows a few iterations of this process. Thue showed that if you start with a square-free sequence and keep applying the rules, the sequence will grow without bound and will never contain a square.

More recently, attention has turned to the question of how many ternary sequences are square free. Doron Zeilberger of Rutgers University, in a paper co-authored with his computer Shalosh B. Ekhad, established that among the 3^n n-digit ternary sequences at least $2^{n/17}$ are square free. Uwe Grimm of the Universiteit van Amsterdam has tightened this lower bound somewhat; he has also found an upper bound and has counted all the n-digit sequences up to $n = 110$. It turns out there are 50,499,301,907,904 ways of arranging 110 ternary digits that avoid all repeated patterns. I'll have to choose one of them when I set up my square-free file drawer.

Bibliography

Allwright, James. 1996. Balanced ternary web pages. http://perun.hscs.wmin.ac.uk/~jra/ternary/

Bharati Krsna Tirtha, Swami. 1965. *Vedic Mathematics, or Sixteen Simple Mathematical Formulae from the Vedas (For One-line Answers to All Mathematical Problems)*. Varanasi, India: Hindu Vishvavidyalaya Sanskrit Publication Board.

Cauchy, Augustin. 1840. Sur les moyens d'éviter les erreurs dans les calculs numériques. *Comptes rendus hebdomadaires des séances de l'Académie des sciences* 11:789–798.

Colson, John. 1726. A short account of negativo-affirmative arithmetick. *Philosophical Transactions of the Royal Society of London* 34:161–173.

Ekhad, Shalosh B., and Doron Zeilberger. 1998. There are more than $2^{n/17}$ n-letter ternary square-free words. *Journal of Integer Sequences*, Vol. 1, Article 98.1.9.

Engineering Research Associates, Inc. 1950. *High-speed Computing Devices*. New York: McGraw-Hill.

Erdős, P., and R. L. Graham. 1980. *Old and New Problems and Results in Combinatorial Number Theory*. Geneva: L'Enseignement Mathématique Université de Genève.

Frieder, G., A. Fong and C. Y. Chow. 1973. A balanced-ternary computer. *Conference Record of the 1973 International Symposium on Multiple-valued Logic*, pp. 68–88.

Gardner, Martin. 1964. The "tyranny of 10" overthrown with the ternary number system. *Scientific American* 210(5):118–124.

Grimm, Uwe. 2001. Improved bounds on the number of ternary square-free words. http://arxiv.org/abs/math.CO/0105245.

Grosch, Herbert R. J. 1991. *Computer: Bit Slices from a Life*. Novato, Calif.: Third Millennium Books.

European Museum on Computer Science and Technology. 1998. Nikolai Brusentsov—the creator of the trinary computer. http://www.icfcst.kiev.ua/museum/Brusentsov.html

Knuth, Donald E. 1981. *The Art of Computer Programming. Vol. 2: Seminumerical Algorithms*. Second edition. Reading, Mass: Addison-Wesley, pp. 190–193.

Lalanne, Léon. 1840. Note sur quelques propositions d'arithmologie élémentaire. *Comptes rendus hebdomadaires des séances de l'Académie des sciences* 11:903–905.

Leslie, John. 1820. *The Philosophy of Arithmetic, Exhibiting a Progressive View of the Theory and Practice of Calculation, with Tables for the Multiplication of Numbers as Far as One Thousand*. Edinburgh: William and Charles Tait.

Rine, David C. (ed.) 1984. *Computer Science and Multiple-Valued Logic: Theory and Applications*. Second edition. Amsterdam: North-Holland Publishing Company.

Shannon, C. E. 1950. A symmetrical notation for numbers. *American Mathematical Monthly* 57:90–93.

Thue, Axel. 1912. Über die gegenseitige lage gleicher teile gewisser Zeichenreihen. In *Selected Mathematical Papers of Axel Thue*, pp. 413–477. Oslo: Universitetsforlaget.

Vardi, Ilan. 1991. The digits of 2^n in base three. In *Computational Recreations in Mathematica*. Reading, Mass.: Addison-Wesley, pp. 20–25.