COMPUTER RECREATIONS

On the ups and downs of hailstone numbers

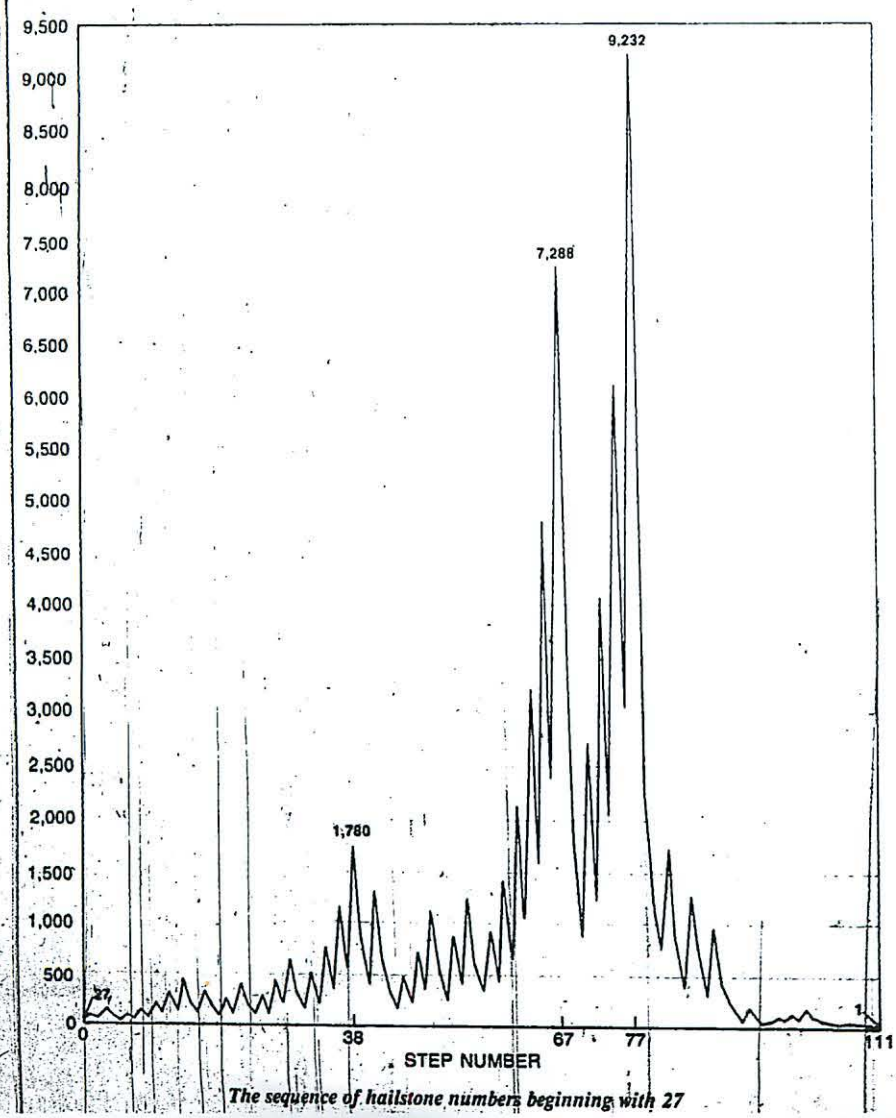
by Brian Hayes

Three steps forward and two steps back: it is not the most efficient way to travel, but it seems certain to get you there in the end. A curious unsolved problem in the theory of numbers puts that conclusion in doubt. The problem can be stated as follows. Choose any positive integer (any whole number greater than zero) and call it N. If the number is odd, triple it and add 1, or in other words replace N by $3N + 1$. If the number is even, divide it by 2, replacing N by $N/2$. In either case the result is the new value of N and the procedure is repeated. After many iterations do the numbers tend to grow larger or smaller? Do they converge on some particular value or diverge toward infinity? How long does it take to settle the "fate" of a number?

For any given value of N, answering these questions calls for nothing more than simple arithmetic. For example, if N is 27, an odd number, the next value is $(3 \times 27) + 1$, or 82; it is followed by 41 and then by 124. Evidently there will be many ups and downs in this series of numbers; the value goes up whenever N is odd and down whenever it is even. The reader is invited to extend the series to see where it leads.

The difficult task is not evaluating the series for a given N but finding a general solution, one that applies to all possible values of N. As yet no general solution has been devised. A great many numbers have been tested explicitly, and they all follow the same pattern, but no one has been able to prove that every number conforms to the pattern. It is hardly the most important unsolved problem in number theory, but it is one of the most irksome. The procedure is easy to describe and to carry out, but it is remarkably difficult to understand what is going on.

The problem illustrates well both the utility and the limitations of the digital computer as a mathematical instrument. To explore beyond the smallest integers some mechanical aid to computation is needed, but almost any computer will do, even a programmable calculator. On the other hand, extending the calculation to a significantly larger range of numbers is practical only with the most powerful computing machinery. When it comes to the very deepest questions, it is not certain any computer can be of help. For the most part the computer is a tool of "experimental" mathematics: it generates examples and counterexamples. Discovering a principle in the peregrinations of N seems to call for theorem proving rather than number crunching.



The sequence of hailstone numbers beginning with 27

When the transformation rule is applied repeatedly to an arbitrary number, what outcome can be expected? Here are three naive hypotheses:

The first argument runs thus: There are equal numbers of odd and even integers, and so in any long series of calculations odd and even values of N should come up equally often. When N is odd, it is increased by a factor of 3 (and a little more), but when N is even, it is decreased by only a factor of 2. Hence the value of N after many iterations should increase without limit. On the average the value should increase by $(3N + 1)/2$ per iteration. For large values of N that is essentially $3/2\ N$.

The second hypothesis relies on the notion that what goes up must come down. This line of reasoning begins with the observation that whenever the cal-

culation happens to yield an exact power of 2, the series of numbers immediately cascades back down to a value of 1. (When any power of 2 except 2 itself is divided by 2, the result is necessarily an even number, so that the descending branch of the calculation is invariably selected.) There are infinitely many exact powers of 2 among the infinite counting numbers, and a calculation that is continued long enough is certain to alight on one of them. Very large values of N might well be reached in the course of a calculation, but eventually there must be a crash.

The third argument is similar in form to the second, but leads to a different conclusion. Note that whenever the calculation changes direction, such as when an odd number is encountered after a series of even ones, it reenters territory it has been in before. Indeed, in wandering up and down the number line it can return to a finite domain of numbers arbitrarily often. Eventually it can be expected to stumble onto a value it has visited before, and once that happens the entire future of the calculation is fixed. Because the procedure for choosing a next step is fully deterministic, any duplicated value of N must lead into a loop that will thereafter be repeated endlessly.

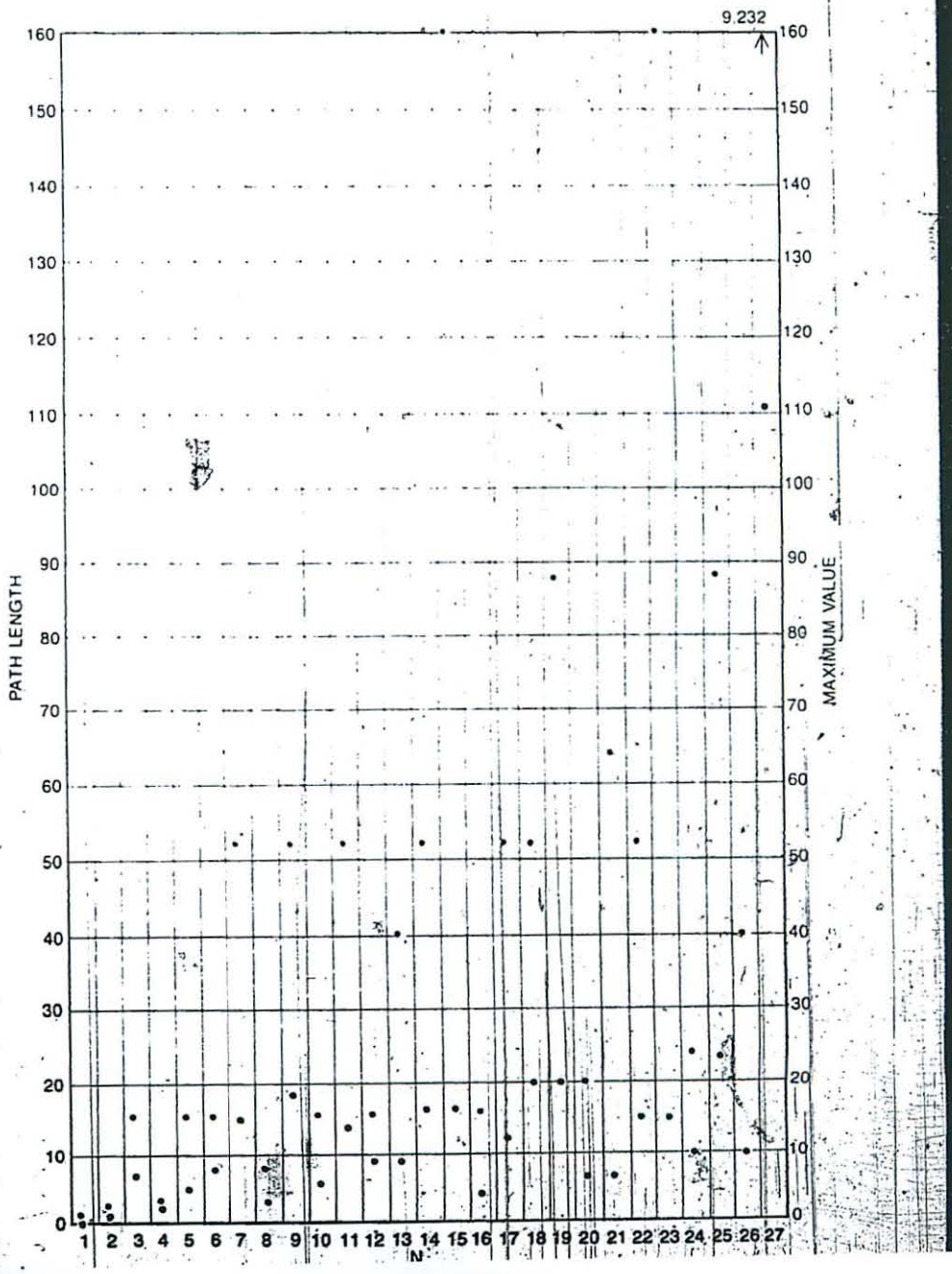
The three hypotheses presented here should not be taken too seriously. They cannot all be right. Some of their premises are definitely open to question. In particular, all three theories rely on a probabilistic analysis, but the series of numbers generated by applying the rule is not a random one. What does mathematical experiment have to say about the matter?

The place to begin the calculation is at the beginning, with 1. It is an odd number, and so the instructions call for multiplying it by 3 and adding 1. The result, 4, is even and is therefore divided by 2, yielding another even number; dividing by 2 again brings the calculation back to 1. Hence with the first computation two of the speculative theories cited above are given handsome support. As the crash hypothesis predicts, the calculation stumbles on a power of 2; it does so after just one iteration. As the cyclical theory predicts, the calculation becomes trapped in an endless loop; the values 4, 2 and 1 will be repeated indefinitely.

Among all the counting numbers 1 is very special: it is the first and the smallest. The results obtained when N is equal to 1 may therefore be atypical; before reaching any conclusions one ought to check further. Since the fate of 2 and 4 are known already from the calculation for $N = 1$, the obvious candidate is 3. It is odd, and so the next value is $(3 \times 3) + 1$, or 10. Dividing by 2 yields 5, and then multiplying by 3 and adding has turned up, and the series cascades through $N = 8$ into the 4–2–1 loop.

After examining the first four natural numbers the trend seems clear, and yet there is still reason for doubt. In the calculations made so far two quantities of interest stand out: the highest value of N reached during a calculation and the path length, which I shall define as the total number of iterations needed to reach a value of 1. For 1 itself the maximum value is 1 and the path length is zero. For 2 the peak is 2 and the length is 1. For 3 the maximum is 16 and the length is 7. The example of 3 suggests that the maximum value reached and the length of the series can be much larger than the initial value of N, and so perhaps the function will turn out to be unbounded for some values of N.

Consider again the series generated when the initial value is 27. As noted above, the first three numbers are 82, 41 and 124, but two successive divisions bring the series back down to 31. Hence after five steps almost no progress has been made. As the calculation continues, however, the three-steps-forward, two-steps-back mechanism gives rise to a series of oscillations of ever larger amplitude. New peaks are reached at 142, 214, 322 and 484. There are further setbacks (at step 19 the value has dropped to 91), but the trend continues to be upward. The calculation passes through 700, through 1,186 and through 2,158



and by the 77th iteration has reached the substantial value of 9,232. It seems we are on our way. As it turns out, however, the path ends at 1 after a total of 111 steps, never having risen higher than 9,232. (The complete path is shown in the illustration on page 10.)

Calculations of the kind I have just traced have been made for all the integers in an extremely wide range. Nabuo Yoneda of the University of Tokyo has tested all values up to 2^{40}, or 1.2×10^{12}. In every case the result has been the same: after a finite number of steps the series subsides into the 4–2–1 loop, where it must stay forever. Among the first 50 integers 27 has the longest path

N	PATH LENGTH	MAXIMUM VALUE
1	0	1
2	1	2
3	7	16
6	8	16
7	16	52
9	19	52
18	20	52
25	23	88
27	111	9,232
54	112	9,232
73	115	9,232
97	118	9,232
129	121	9,232
171	124	9,232
231	127	9,232
313	130	9,232
327	143	9,232
649	144	9,232
703	170	250,504
871	178	190,996
1,161	181	190,996
2,223	182	250,504
2,463	208	250,504
2,919	216	250,504
3,711	237	481,624
6,171	261	975,400
10,971	267	975,400
13,255	275	497,176
17,647	278	11,003,416
23,529	281	11,003,416
26,623	307	106,358,020
34,239	310	18,976,192
35,655	323	41,163,712
52,527	339	106,358,020
77,031	350	21,933,016

Sequence of longest paths up to N = *100,000*

N	PATH LENGTH	MAXIMUM VALUE
1	0	1
2	1	2
3	7	16
7	16	52
15	17	160
27	111	9,232
255	47	13,120
447	97	39,364
639	131	41,524
703	170	250,504
1,819	161	1,276,936
4,255	201	6,810,136
4,591	170	8,153,620
9,663	184	27,114,424
20,895	255	50,143,264
26,623	307	106,358,020
31,911	160	121,012,864
60,975	334	593,279,152
77,671	231	1,570,824,736

Sequence of peak values up to N = *100,000*

back to 1 (although 41 and 31 are not much shorter and reach the same peak value, for reasons that should be apparent from the information given above). No positive integer has been found to generate a series that goes off toward infinity, and no loops other than the 4–2–1 loop have been found. Nevertheless, the conjecture that all positive numbers conform to the same pattern remains without a secure theoretical basis.

The $3N + 1$ problem, as it is generally called, has a murky history, but it does not seem to be of great antiquity. Over the past 30 years or so it has turned up repeatedly in various university departments of mathematics and computer science, its comings and goings seeming to be as capricious as the advances and recessions of the numbers themselves. Jeffrey C. Lagarias of Bell Laboratories, who has recently looked into the origins of the problem and the prospects for solving it, notes that it may have been invented several times. In the 1930's Lothar Collatz, who was then a student at the University of Hamburg, investigated a class of problems that includes the $3N + 1$ problem, although the work was not published until many years later. In 1952 the British mathematician B. Thwaites independently discovered the problem, and a few years later it was invented yet again by Richard Vernon Andree of the University of Oklahoma at Norman.

Lagarias cites some 20 research articles on the $3N + 1$ problem and its generalizations, most of them published within the past 10 years, but the problem had circulated by word of mouth long before. Collatz' colleague Helmut Hasse introduced it at Syracuse University in the 1950's, and Stanislaw Ulam took it to Los Alamos and elsewhere. Shizuo Kakutani, who first heard of the problem in about 1960, reported to Lagarias: "For a month everybody at Yale worked on it, with no result. A similar phenomenon happened when I mentioned it at the University of Chicago. A joke was made that the problem was part of a conspiracy to slow down mathematical research in the U.S."

Another sustained attack on the problem, with an emphasis on computer-aided numerical calculations, was made in the early 1970's by a group in the Artificial Intelligence Laboratory at M.I.T. The problem is recorded as Item 133 in the group's informal (and unpublished) transactions, called HAKMEM, or "hackers' memorandum."

In its wanderings the problem has been known by many names. Calling it the $3N + 1$ problem does not seem entirely satisfactory, in that it gives undue attention to one half of the procedure and slights the other half. Of the various alternatives the one I find most congenial identifies the numbers generated from a given starting value as "hailstone numbers." The path the series follows is rather like the trajectory of a hailstone through a storm cloud, rising in updrafts and then falling under its own weight.

A computer program for calculating hailstone numbers can be written in a few lines of a higher-level programming language such as BASIC. Indeed, the central algorithm can be expressed in a single statement. In BASIC it might be

```
IF N MOD 2 = 0 THEN N = N/2
ELSE N = 3*N + 1
```

Here the first operation is one that people (but not computers) are capable of doing without explicit calculation: determining whether N is odd or even. N MOD 2 is a modulus operation, which computes the remainder when N is divided by 2. If the remainder is 0, the THEN part of the statement is executed and N is set equal to $N/2$; otherwise the ELSE part is executed, setting N equal to $3N + 1$.

A program in BASIC serves well enough for generating hailstone numbers from the first few hundred integers, but if more extensive calculations are undertaken, it becomes intolerably slow. The BASIC statement calls for a division (as part of the modulus operation), a comparison and then either a second division or a multiplication and an addition. Division and multiplication are time-consuming operations, particularly in a small computer system. There is much to be gained here by speaking directly to the central processing unit in its own language. All the division and multiplication operations can thereby be eliminated.

The illustration on the opposite page gives a schematic account of such a machine-language program. It is assumed that the value of N is initially in a register designated AX, which also serves as an "accumulator" where arithmetic operations are done. The value at the start of the procedure is the binary representation of the decimal number 27.

The first step is to save a copy of the initial value in another register, here labeled BX. The division operation is avoided by exploiting a property of the binary number system: shifting a binary number to the right one position is equivalent to dividing it by 2, just as shifting a decimal number to the right divides it by 10. In the course of the shift the rightmost digit (the units digit) is preserved in a one-bit storage location called the carry flag. Testing the carry flag determines whether the original number was odd or even, since in binary notation every odd number ends in a 1 and every even number ends in a 0.

If N is even, the calculation is now finished. The value remaining in register AX after shifting to the right one place

the quotient $N/2$. In this case, however, N is odd and further computations are needed. First the original value of N is recovered from register BX. Then, instead of multiplying by 3, the value is added to itself twice; even though this requires two machine instructions instead of one, it is done appreciably faster. The final step is to increment the number in AX by 1. In the instruction set of one microprocessor the entire program takes 20 cycles of the computer's clock when N is even and 18 cycles when N is odd. At a clock frequency of roughly five megahertz the program fragment could in principle be executed some 250,000 times per second. (A few more clock cycles could be saved, at some cost in program clarity.) The equivalent algorithm employing division and multiplication instructions takes 175 cycles for even N and 286 cycles for odd N.

In the illustration registers are shown as being eight bits wide and can therefore accommodate numbers no larger than 2^8, or 256. In most microprocessors the registers are actually 16 bits wide, so they can hold numbers up to 65,536. Even that limit is a severe constraint: a program employing 16-bit arithmetic could not calculate hailstone numbers beyond $N = 702$. Achieving a higher capacity requires multiple-precision arithmetic, in which a single number is split between two or more registers or memory locations. With 32 bits of precision numbers up to about four billion can be represented; 64 bits extend the limit to 18 digits. Each increase in precision, however, exacts a penalty in speed.

The algorithm for calculating one value of N is only a fragment of a working program. In addition there must be routines or facilities for getting input values into the machine and for displaying results. A practical set of programs for exploring the hailstone numbers ought to do a good deal more. For example, it should be possible to print out the entire sequence of numbers generated by a given starting value, or to list the path length and maximum value associated with all the integers in a given range. Another program could be set up to search for numbers yielding progressively longer paths or larger peak values. There are many other possibilities.

Variations on the $3N + 1$ formula employing different coefficients and constants are also worth exploring. R. William Gosper and Richard Schroeppel, when they were members of the HAKMEM group, investigated the $3N - 1$ problem and showed that it is equivalent to the $3N + 1$ problem with negative values of N. Every number they checked terminates in one of three loops: the largest loop begins at $N = 17$ and has a length of 18 steps.

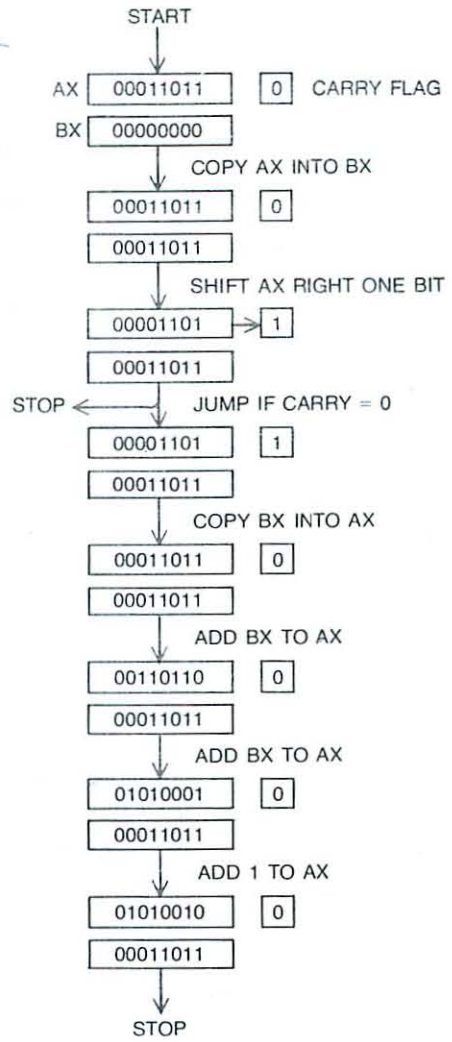
A program whose only aim is to search for numbers that do not fall into the 4–2–1 loop can be greatly streamlined. If numbers are checked in succession beginning with 1, only odd numbers need to be examined. Any even number is immediately reduced by half, and so the path it generates would already have been detected. For similar reasons there is no need to follow the path of a number all the way to 1; once the value of N falls below the initial value the candidate can be dismissed. Still more effective rules for narrowing the search have been developed by William H. Henneman, a student in the HAKMEM group who is now at Boston University.

Although no proof has yet been discovered, a hint of an explanation may lie in a heuristic argument more refined than the three naive hypotheses given above. There it was noted that in any stage of the calculation N has an equal probability of being multiplied by 3 or divided by 2, leading to the suggestion that the value should tend to increase by a factor of $3/2$ per iteration. Lagarias points out, however, that one-fourth of all the integers are divisible not only by 2 but also by 4; one-eighth of them are divisible by 8, one-sixteenth by 16 and so on. Taking into account divisions by all possible powers of 2 yields a prediction that N should decrease by a factor of $3/4$ per iteration. The empirical evidence supports the prediction.

Even if it turns out that all positive integers fall into the 4–2–1 loop, the hailstone numbers offer an abundance of curiosities. Perhaps the most intriguing properties of the numbers are conspicuous patterns in the distribution of path lengths and peak values. If a number as small as 27 can keep the ball in the air for 111 steps and reach a height of 9,232, one might well expect that the path length and the peak value would grow rapidly as N increased. Actually the path length grows very slowly; the increase in the maximum value is faster, but it is also quite erratic.

Among the first 100 integers the longest path is 118 steps (at $N = 97$); among the first 100,000 integers the longest path is just 350 steps (at $N = 77{,}031$). Thus increasing N by a factor of 1,000 increases the path length by a factor of only 3; the relation appears to be a logarithmic one. The record maximum of 9,232 set at $N = 27$ is not exceeded until $N = 255$, which reaches a peak of 13,120. New maximums are recorded at quite irregular intervals. The hailstone sequence for $N = 77{,}671$ reaches the extraordinary height of 1,570,824,736.

It is easy to see that the peak value reached in a hailstone calculation must invariably be an even number. It can also be proved that only an odd value of N can set a new record for maximum height (with the possible exception of $N = 2$). In the case of numbers that set new records for path length there is no



The hailstone algorithm

theoretical argument I know of that requires them to be either odd or even. Nevertheless, among the first 100,000 integers path-length records are set almost exclusively by odd values of N.

A listing of the path length and maximum value for a range of numbers has a frustrating mixture of regularity and disorder: it is definitely not random, but the pattern resists interpretation. For instance, certain maximum values are much commoner than others and far too common to be explained by any statistical process. The outstanding example is 9,232, the number first reached at $N = 27$. Of the first 1,000 integers more than 350 have their maximum at 9,232.

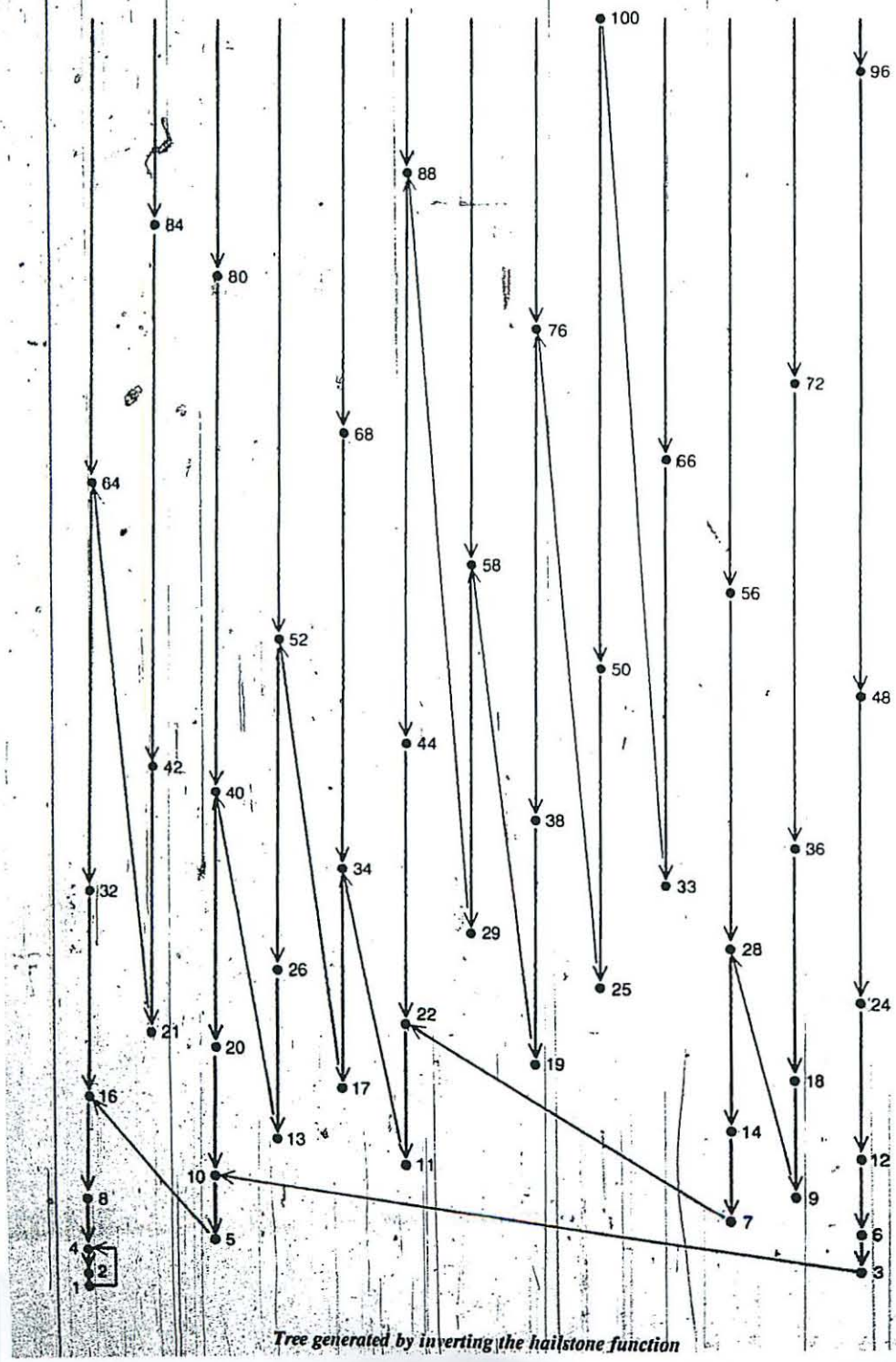
The distribution of path lengths is equally peculiar. Every possible length can be produced (by the successive exact powers of 2), but again some numbers appear far more often than others. Moreover, both the path lengths and the maximum values show a strong tendency to form clusters. In 1976 Fred Gruenberger of California State University in Northridge published a list of such clusters; the largest was a string of 52 consecutive numbers that all have the same path length. Can two consecutive val-

ues of N have the same path length and the same maximum? The question can be settled algebraically, but readers who prefer a numerical demonstration might want to examine the hailstone sequences for $N = 386$ through $N = 391$.

One illuminating way to look at the hailstone problem is to turn it upside down. Suppose it is true that all positive numbers ultimately fall into the 4–2–1 loop. They must then form an unbroken chain through which any number in the infinite counting sequence is connected to the bottom of the loop. Accordingly it should be possible to invert the hailstone function: to begin with 1 and apply the transformation "backward" in order to generate every larger number. If some number cannot be reached in this way, by following the river upstream, the number cannot yield 1 as its final value.

The method might well yield a general solution of the hailstone problem, if only it could be carried to completion. As it turns out, the procedure is not as straightforward as it seems. The normal hailstone function is deterministic: a value of N at any point in the calculation can have only one possible successor. If N is 40, for example, the next number can only be 20. When the path is traced in reverse, there are ambiguities. When the value $N = 20$ is encountered, it is known it could only have been generated from 40, which must therefore be the next value. At 40, however, the next value could be either 80 or 13, the stream splits, and both tributaries must be explored. There is a bifurcation at every number of the form $6K + 4$, where K can be zero or any positive integer.

A branching system of this kind can be traced only to a finite depth. A single branch must be followed until some predetermined limit is reached and then attention must be diverted to another branch. When the limit is set at 100, 13 branches are explored and 49 numbers are confirmed to be connected to the system of numerical rills and rivulets. With the limit at 1,000 there are 84 branches, but only 340 numbers are counted. A limit of 10,000 yields 1,065 branches, which pass through 4,235 numbers. Note that more than half of the numbers seem to lie in the interstices between the branches of the stream. As the limit is increased more numbers are included, but even more are missed. If the ramifications of the system could be explored to infinite depth, would all positive integers ultimately find a place in it? That is the big question yet to be answered.



Tree generated by inverting the hailstone function

In October two combinatorial problems were mentioned as being unlikely candidates for solution by "nonalgorithmic" methods with an electronic spreadsheet. A number of readers were quick to show that it can be done.

Spreadsheet solutions to the Tower of Hanoi problem were sent by David Behar, John B. Jones, Jr., George Arthur Miller, J. B. Sladen, Alun Wyn-jones and others. The techniques employed were similar. An algorithm in which odd-numbered disks circulate clockwise and even-numbered ones counterclockwise was disassembled so that a temporal sequence of instructions became a spatial array of them.

Behar, Miller and Sladen also solved the eight-queens problem. Here the main difficulty is the need to backtrack when a developing solution fails. I had supposed some record of previous unsuccessful attempts would have to be kept, but that is not the case. D. H. Fremlin, in a commentary on the spreadsheet approach to the problems, pointed out "a method that will generate all the moves in turn without any memory other than what is displayed.... The formulas are complicated and involve sequential algorithms if they are done in their natural forms, but all these 'look for' subroutines can be represented on the spreadsheet by local calculations."