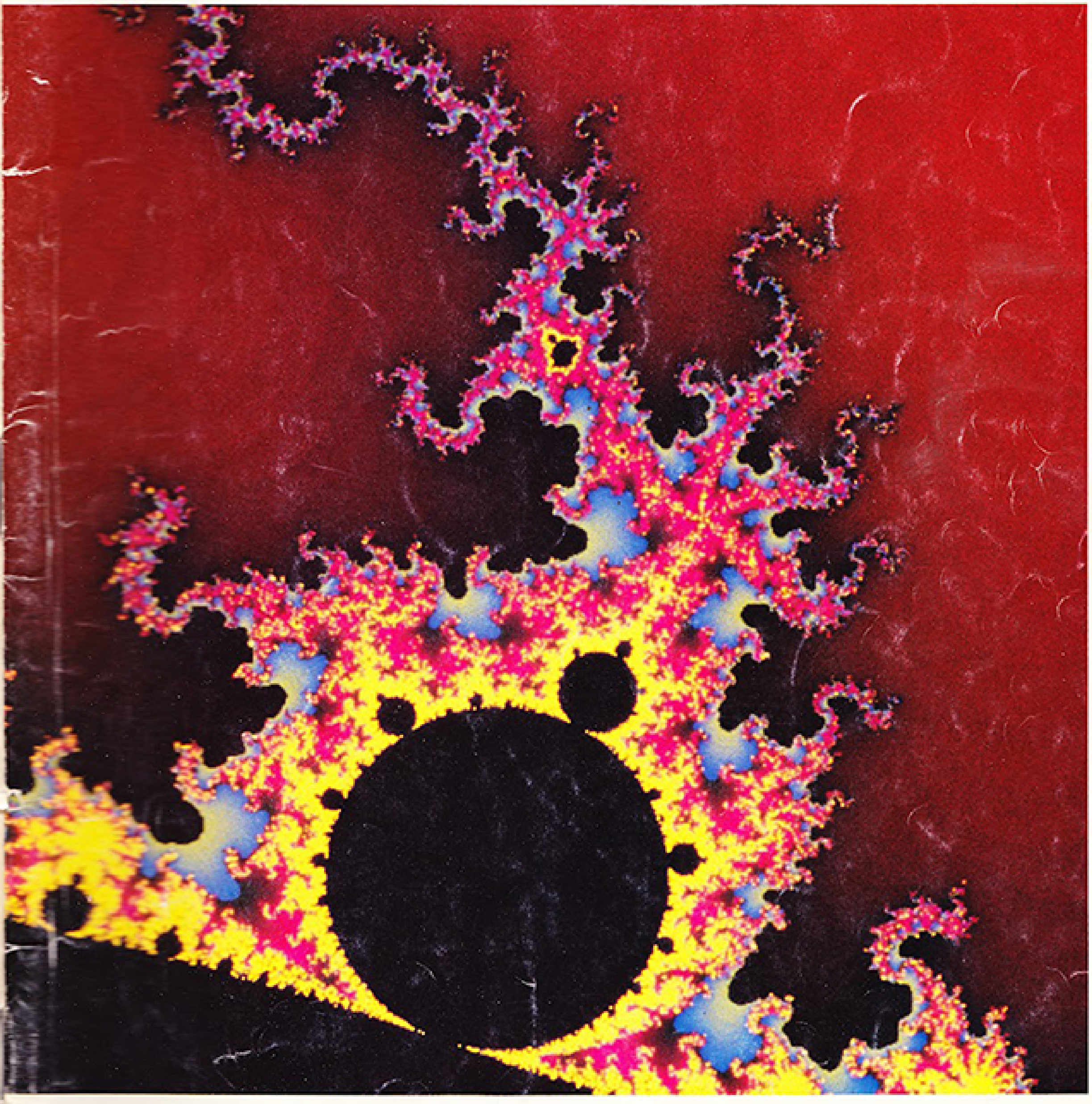
SCIENTIFIC AMERICAN



EXPLORING THE MANDELBROT SET

$2.50

August 1985

Scientific American August 1985

COMPUTER RECREATIONS

A computer microscope zooms in for a look at the most complex object in mathematics

by A. K. Dewdney

The Mandelbrot set broods in silent complexity at the center of a vast two-dimensional sheet of numbers called the complex plane. When a certain operation is applied repeatedly to the numbers, the ones outside the set flee to infinity. The numbers inside remain to drift or dance about. Close to the boundary minutely choreographed wanderings mark the onset of the instability. Here is an infinite regress of detail that astonishes us with its variety, its complexity and its strange beauty.

The set is named for Benoit B. Mandelbrot, a research fellow at the IBM Thomas J. Watson Research Center in Yorktown Heights, N.Y. From his work with geometric forms Mandelbrot has developed the field he calls fractal geometry, the mathematical study of forms having a fractional dimension. In particular the boundary of the Mandelbrot set is a fractal, but it is also much more.

With the aid of a relatively simple program a computer can be converted into a kind of microscope for viewing the boundary of the Mandelbrot set. In principle one can zoom in for a closer look at any part of the set at any magnification [*see cover of this issue and illustrations on pages 17–19*]. From a distant vantage the set resembles a squat, wart-covered figure eight lying on its side. The inside of the figure is ominously black. Surrounding it is a halo colored electric white, which gives way to deep blues and blacks in the outer reaches of the plane.

Approaching the Mandelbrot set, one finds that each wart is a tiny figure shaped much like the parent set. Zooming in for a close look at one of the tiny figures, however, opens up an entirely different pattern: a riot of organic-looking tendrils and curlicues sweeps out in whorls and rows. Magnifying a curlicue reveals yet another scene: it is made up of pairs of whorls joined by bridges of filigree. A magnified bridge turns out to have two curlicues sprouting from its center. In the center of this center, so to speak, is a four-way bridge with four more curlicues, and in the center of these curlicues another version of the Mandelbrot set is found.

The magnified version is not quite the same Mandelbrot set. As the zoom continues, such objects seem to reappear, but a closer look always turns up differences. Things go on this way forever, infinitely various and frighteningly lovely.

Here I shall describe two computer programs, both of which explore the effects of iterated operations such as the one that leads to the Mandelbrot set. The first program generated the colored illustrations appearing in this month's column. The program can be adapted to run on personal computers that have the appropriate hardware and software for generating graphics. It will create satisfying images even if one has access only to a monochrome display. The second program is for readers who, like me, need an occasional retreat from infinite complexity to the apparent simplicity of the finite.

The word "complex" as used here has two meanings. The usual meaning is obviously appropriate for describing the Mandelbrot set, but the word has a second and more technical sense. A number is complex when it is made up of two parts, which for historical reasons are called real and imaginary. These terms no longer have any special significance: the two parts of a complex number might as well be called Humpty and Dumpty. Thus $7 + 4i$ is a complex number with real part 7 (Humpty) and imaginary part $4i$ (Dumpty). The italic i next to the 4 shows which part of the complex number is imaginary.

Every complex number can be represented by a point in the plane; the plane of complex numbers is called the complex plane. To find $7 + 4i$ in the complex plane, start at the complex number 0, or $0 + 0i$, and measure seven units east and four units north. The resulting point represents $7 + 4i$. The complex plane is an uncountable infinity of such numbers. Their real parts and their imaginary parts can be either positive or negative and either whole numbers or decimal expansions.

Adding or multiplying two complex numbers is easy. To add $3 - 2i$ and $7 + 4i$, add the parts separately; the sum is $10 + 2i$. Multiplying complex numbers is only slightly more difficult. For example, if the symbol i is treated like the x in high school algebra, the product of $3 - 2i$ and $7 + 4i$ is $21 + 12i - 14i - 8i^2$. At this stage a special property of the symbol i must be brought into play: it happens that i^2 equals -1. Thus the product can be simplified by collecting the real and the imaginary parts: it is $29 - 2i$.

It is now possible to describe the iterative process that generates the Mandelbrot set. Begin with the algebraic expression $z^2 + c$, where z is a complex number that is allowed to vary and c is a certain fixed complex number. Set z initially to be equal to the complex number 0. The square of z is then 0 and the result of adding c to z^2 is just c. Now substitute this result for z in the expression $z^2 + c$. The new sum is $c^2 + c$. Again substitute for z. The next sum is $(c^2 + c)^2 + c$. Continue the process, always making the output of the last step the input for the next one.

Strange things happen when the iterations are carried out for particular values of c. For example, here is what happens when c is $1 + i$:

first iteration,	$1 + 3i$
second iteration,	$-7 + 7i$
third iteration,	$1 - 97i$

Note that the real and the imaginary parts may grow, shrink or change sign. If this process of iteration continues, the resulting complex numbers get progressively larger.

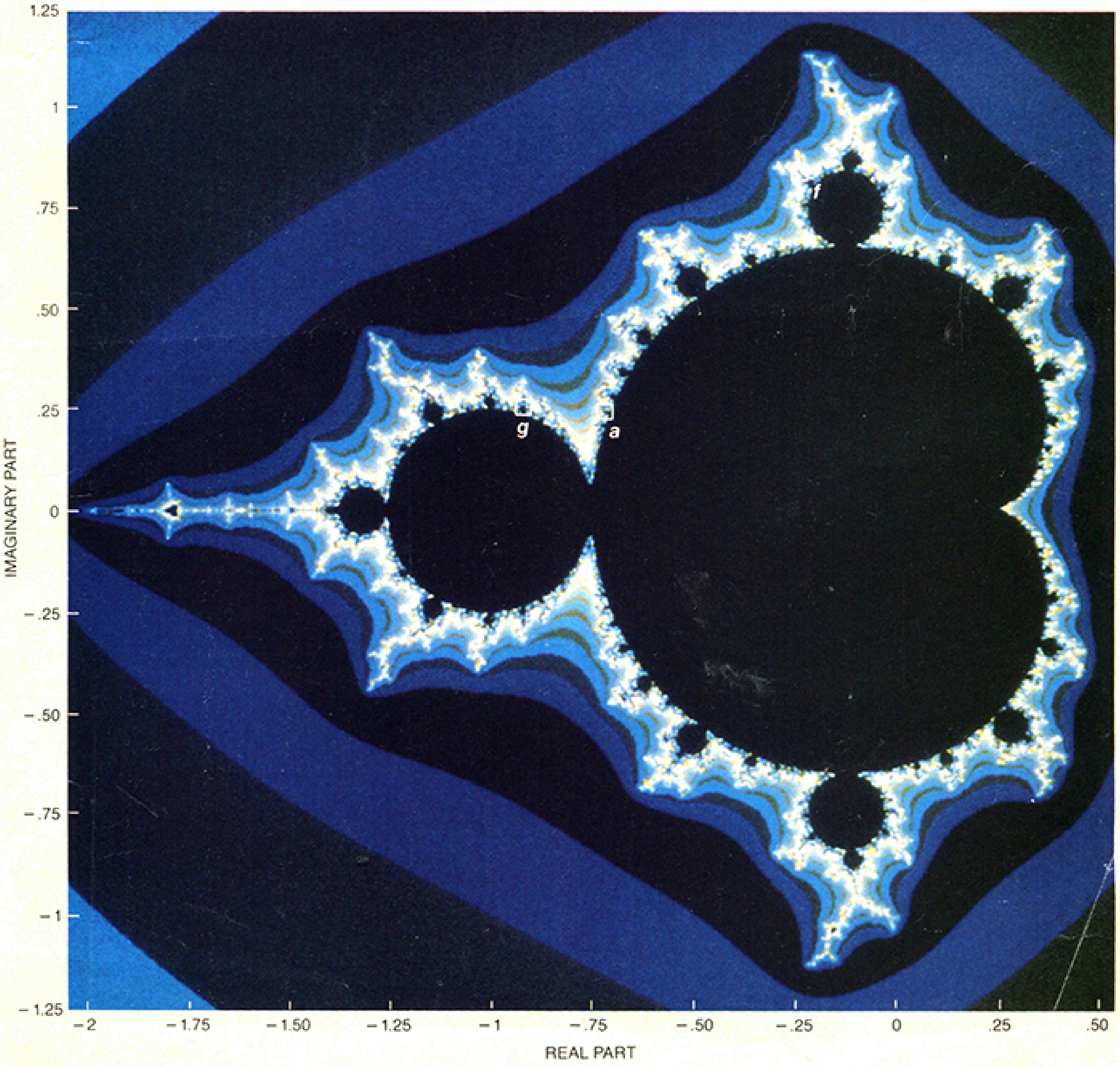
What exactly is meant by the size of a complex number? Since complex numbers correspond to points in the plane, ideas of distance apply. The size of a complex number is just its distance from the complex number 0. That distance is the hypotenuse of a right triangle whose sides are the real and the imaginary parts of the complex number. Hence to find the size of the number square each of its parts, add the two squared values and take the square root of the sum. For example, the size of the complex number

$7 + 4i$ is the square root of $7^2 + 4^2$, or approximately 8.062. When complex numbers reach a certain size under the iterative process I have just described, they grow very quickly: indeed, after a few more iterations they exceed the capacity of any computer.

Fortunately I can ignore all the complex numbers c that run screaming off to infinity. The Mandelbrot set is the set of all complex numbers c for which the size of $z^2 + c$ is finite even after an indefinitely large number of iterations. The program I am about to describe searches for such numbers. I am indebted in all of this to John H. Hubbard, a mathematician at Cornell University. Hubbard is an authority on the Mandelbrot set, and he was one of the first people to make computer-generated images of it. Most of the images in this article were made by Heinz-Otto Peitgen and his colleagues at the University of Bremen. Peitgen learned the art from Hubbard.

Hubbard's program has inspired a program I call MANDELZOOM. The program sets up an array called *pic*, which is needed for saving pictures. The entries of *pic* are separate picture elements called pixels, which are arranged in a grid pattern. Hubbard's array has 400 columns and 400 rows, and Peitgen's is even larger. Readers who want to adapt MANDELZOOM for personal use must choose an array suited to their equipment and temperament. Larger arrays impose a longer wait for the pictures, but they improve the resolution.

In the first part of MANDELZOOM one may select any square region of the complex plane to be examined. Specify the southwest corner of the square with the complex number to which it corresponds. Two variables in the program, *acorner* and *bcorner*, enable one to enter the real part and the imaginary part of the number respectively. Specify the length of each side of the



The Mandelbrot set and its coordinates in the complex plane. The details shown on the cover and on the next two pages are outlined square by entering a value for a variable called *side*.

The second part of the program adjusts the array *pic* to match the square of interest by computing the size of a variable called *gap*. *Gap* is the distance within the square between adjacent pixels. To obtain *gap* divide *side* by the number of rows (or columns) in *pic*.

The heart of the program is its third part. Here a search is made for the complex numbers *c* in the Mandelbrot set, and colors are assigned to the numbers that are, in a special sense, nearby. The procedure must be carried out once for every pixel; thus Hubbard's 400-by-400 array requires 160,000 separate computations. Assume the program is currently working on the pixel in row *m* and column *n*; the third part then breaks down into four steps:

1. Calculate one complex number *c* that is assumed to represent the pixel: add $n \times gap$ to *acorner* to obtain the real part *ac* of *c*; add $m \times gap$ to *bcorner* to obtain the imaginary part *bc* of *c*. It is not necessary to include the imaginary number *i* in the program.

2. Set a complex variable *z* (which has parts *az* and *bz*) equal to $0 + 0i$. Set an integer variable called *count* equal to 0.

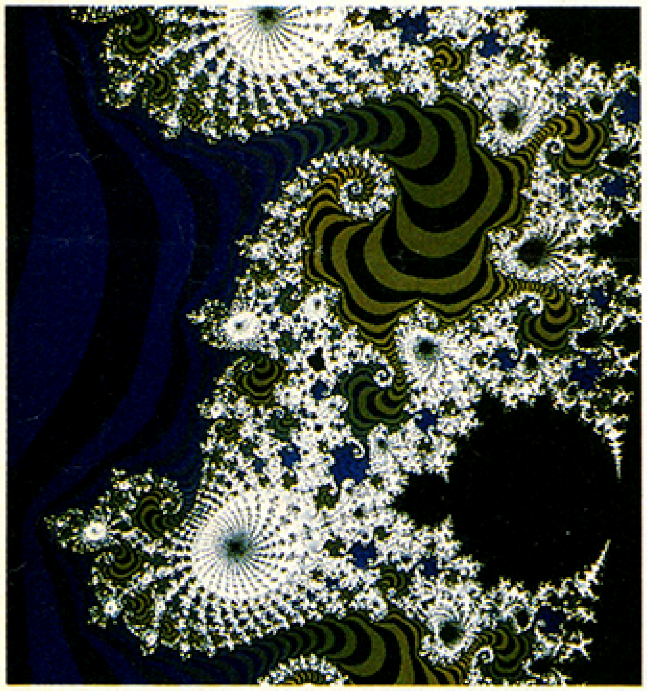
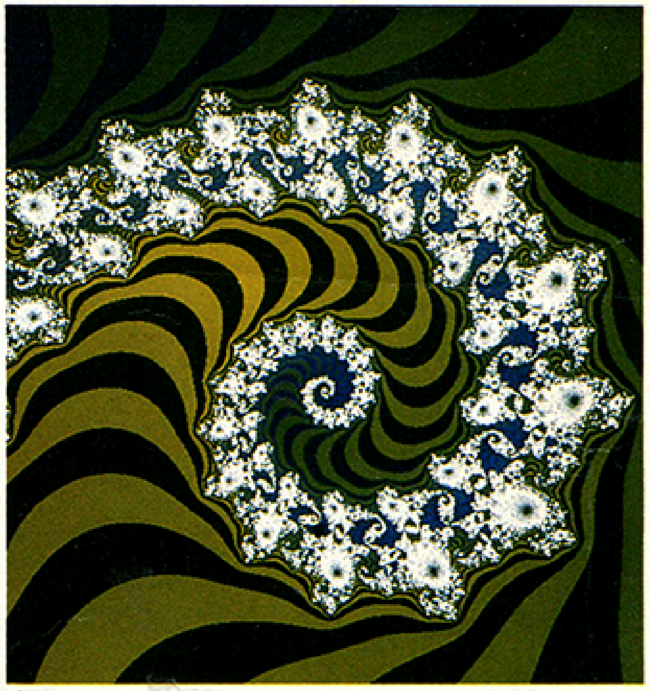
3. Carry out the following three steps repeatedly, until either the size of *z* exceeds 2 or the size of *count* exceeds 1,000, whichever comes first:

```
      z ← z² + c
  count ← count + 1
   size ← size of z
```

Why is the number 2 so important? A straightforward result in the theory of complex-number iterations guarantees that the iterations will drive *z* to infinity if and only if at some stage *z* reaches a size of 2 or greater. It turns out that relatively many points with an infinite destiny reach 2 after only a few iterations. Their slower cousins become increasingly rare at higher values of the variable *count*.

4. Assign a color to *pic* (*m,n*) according to the value reached by *count* at the end of step 3. Display the color of the corresponding pixel on the screen. Note that the color of a pixel depends on only one complex number within its tiny domain, namely the one at its northeast corner; the behavior of this number then represents the behavior of the entire pixel.

The scheme for assigning colors requires that the range of *count* values attained within the array be grouped into subranges, one subrange for each

a

b

c

d

Successive enlargements of the "shepherd's crook" in region a *of the image on the preceding page* color. Pixels for which the size of z reaches 2 after only a few iterations are colored red. Pixels for which the size of z reaches 2 after relatively many iterations are colored violet, at the other end of the spectrum. Pixels for which the size of z is less than 2 even after 1,000 iterations are assumed to lie in the Mandelbrot set; they are colored black.

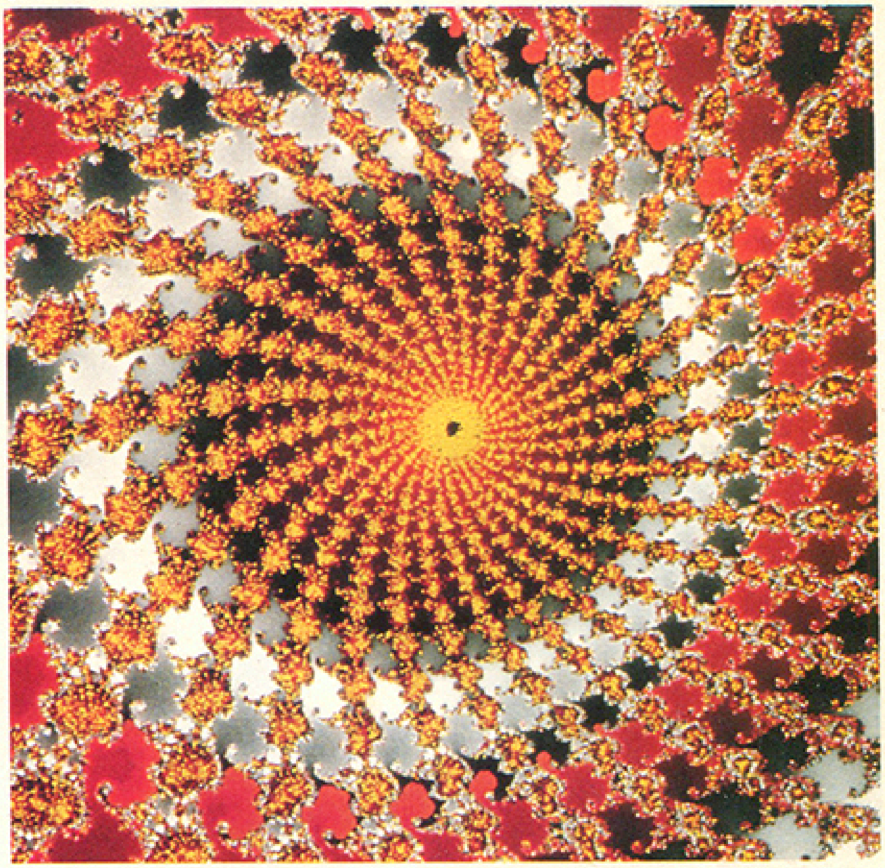
It makes sense to leave the colors unspecified until the range of *count* values in a particular square has been determined. If the range is narrow, the entire color spectrum can then be assigned within that range. Thus Hubbard suggests that in step 4 only the value of *count* be assigned to each array element of *pic*. A separate program can then scan the array, determine the high and low values of *count* and assign the spectrum accordingly. Readers who get this far will certainly find workable schemes.

The reader who does not have a color monitor can still take part in black and white. Complex numbers for which z is larger than 2 after r iterations are colored white. The rest are colored black. Adjust r to taste. To avoid all-night runs the array can be, say, 100 rows by 100 columns. Hubbard also suggests it is perfectly reasonable to reduce the maximum number of iterations per point from 1,000 to 100. The output of such a program is a suggestive, pointillistic image of its colored counterpart [*see illustration on next page*].

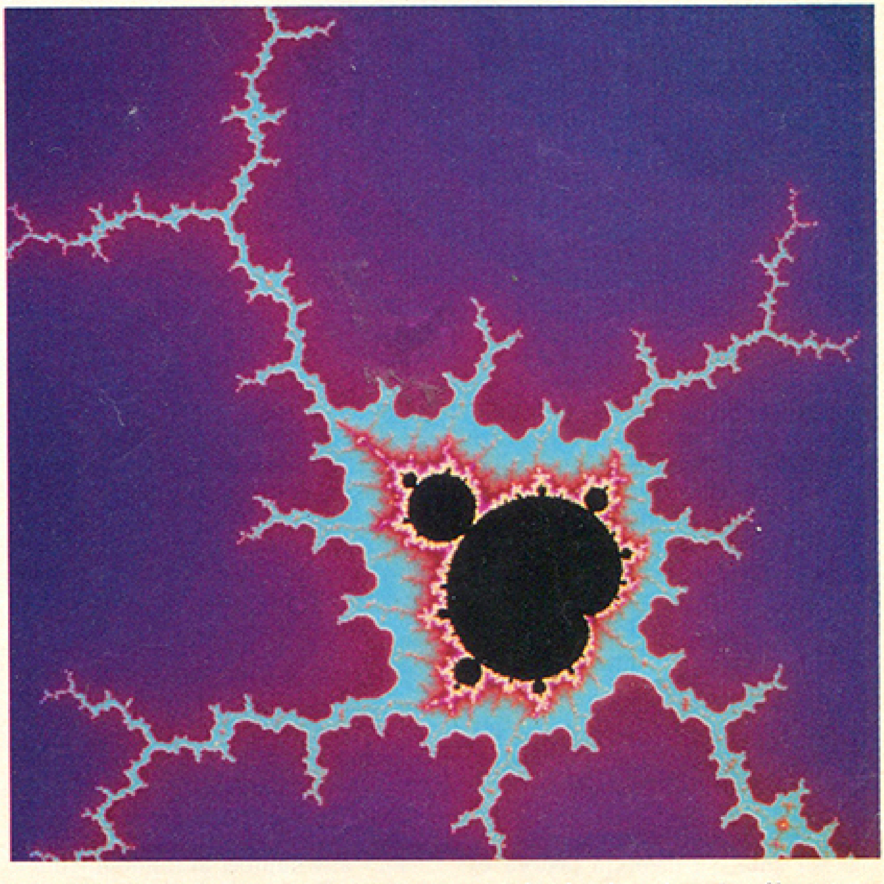
How powerful is the "zoom lens" of a personal computer? It depends to some degree on the effective size of the numbers the machine can manipulate. For example, according to Magi (my microcomputer amanuensis at the University of Western Ontario), the IBM PC uses the 8088 microprocessor, a chip manufactured by the Intel Corporation designed to manipulate 16-bit numbers. A facility called double precision makes it possible to increase the length of each number to 32 bits. With such double precision Magi and I calculate that magnifications on the order of 30,000 times can be realized. Higher precision software that in effect strings these numbers together can enhance the numerical precision to hundreds of significant digits. The magnification of the Mandelbrot set theoretically attainable with such precision is far greater than the magnification needed to resolve the nucleus of the atom.

Where should one explore the complex plane? Near the Mandelbrot set, of course, but where precisely? Hubbard says that "there are zillions of beautiful spots." Like a tourist in a

e


A compound eye peering out from a region of image d *on the opposite page*

f


A miniature Mandelbrot in region f *on page 17, tethered to the main set by a filament*

Pointillist, miniature Mandelbrot generated by a monochrome monitor

land of infinite beauty, he bubbles with suggestions about places readers may want to explore. They do not have names like Hawaii or Hong Kong: "Try the area with the real part between .26 and .27 and the imaginary part between 0 and .01." He has also suggested two other places:

Real Part	*Imaginary Part*
−.76 to −.74	.01 to .03
−1.26 to −1.24	.01 to .03

The reader who examines the color images accompanying this article should bear in mind that any point having a color other than black does not belong to the Mandelbrot set. Much of the beauty resides in the halo of colors assigned to the fleeing points. Indeed, if one were to view the set in isolation, its image might not be so pleasing: the set is covered all over with filaments and with miniature versions of itself.

In fact none of the miniature Mandelbrots are exact copies of the parent set and none of them are exactly alike. Near the parent set there are even more miniature Mandelbrots, apparently suspended freely in the complex plane. The appearance is deceiving. An amazing theorem proved by Hubbard and a colleague, Adrian Douady of the University of Paris, states that the Mandelbrot set is connected. Hence even the miniature Mandelbrots that seem to be suspended in the plane are attached by filaments to the parent set. The minatures are found almost everywhere near the parent set and they come in all sizes. Every square in the region includes an infinite number of them, of which at most only a few are visible at any given magnification. According to Hubbard, the Mandelbrot set is "the most complicated object in mathematics."

Readers with a simple appetite for more color images of the Mandelbrot set and other mathematical objects can write to Hubbard for a brochure (Department of Mathematics, Cornell University, Ithaca, N.Y. 14853). The brochure includes an order form with which one can buy 16-inch-square color prints that are similar in quality to the Peitgen images shown here.

Confronted with infinite complexity it is comforting to take refuge in the finite. Iterating a squaring process on a finite set of ordinary integers also gives rise to interesting structures. The structures are not geometric but combinatorial.

Pick any number at random from 0 through 99. Square it and extract the last two digits of the result, which must also be a number from 0 through 99. For example, 59^2 is equal to 3,481; the last two digits are 81. Repeat the process and sooner or later you will generate a number you have already encountered. For example, 81 leads to the sequence 61, 21, 41 and 81, and this sequence of four numbers is then repeated indefinitely. It turns out that such loops always arise from iterative processes on finite sets. Indeed, it is easy to see there must be at least one repeated number after 100 operations in a set of 100 numbers; the first repeated number then leads to a loop. There is a beautiful program for detecting the loops that requires almost no memory, but more of this later.

It takes only an hour to diagram the results of the squaring process. Represent each number from 0 through 99 by a separate point on a sheet of paper. If the squaring process leads from one number to a new number, join the corresponding points with an arrow. For example, an arrow should run from point 59 to point 81. The first few connections in the diagram may lead to tangled loops, and so it is a good idea to redraw them from time to time in such a way that no two arrows cross. A nonintersecting iteration diagram is always possible.

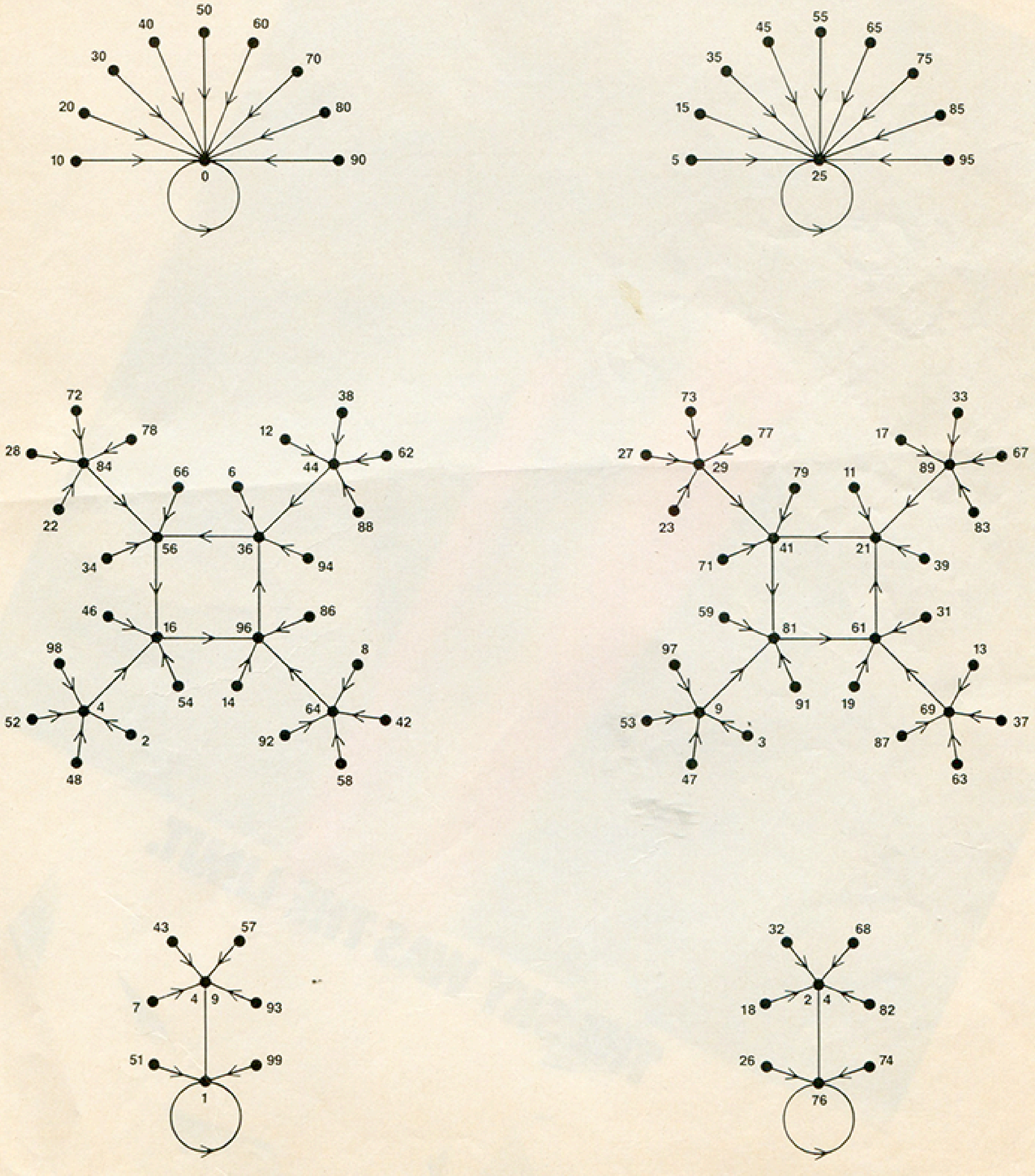
One can go even further. Separate subdiagrams often arise, and they can be displayed in a way that highlights some of the symmetries arising from the iterations. For example, the nonintersecting iteration diagram for the squaring process on the integers from 0 through 99 includes six unconnected subdiagrams. The pieces come in identical pairs and each piece is highly symmetrical [*see illustration on opposite page*]. Can the reader explain the symmetry? What would happen if the integers from 0 through 119 were used instead? Is there a relation between the number of unconnected pieces found in the diagram and the largest integer in the sequence?

Similar patterns of iteration hold for some of the complex numbers in the Mandelbrot set: for certain values of c repeated iterations of $z^2 + c$ can lead to a finite cycle of complex numbers. For example, the complex number $0 + 1i$ leads to an indefinite oscillation between the two complex numbers $-1 + 1i$ and $0 - 1i$. The cycle may even have only one member. Whether such cycles are found in a finite set or in the infinite Mandelbrot set, they are called attractors.

Each of the six parts of the iteration diagram for the integers 0 through 99 includes one attractor. Geometrically the attractor can be represented as a polygon, and the sets of numbers that lead into it can be represented as trees.

One way to find an attractor by computer is to store each newly generated number in a specially designated array. Compare the new number with all the numbers previously stored in the array. If a match is found, print all the numbers in the array from the matching number to the number just created. The method is straightforward and easy to program. Nevertheless, it can take a long time if the array is large. An attractor cycle within an array that includes *n* numbers would take on the order of n^2 comparisons to discover: each new number must be compared with up to *n* numbers in the array.

There is a clever little program that will find an attractor much faster. The program requires not *n* words of memory but only two, and it can be encoded on the simplest of programmable pocket calculators. The program is found in a remarkable book titled *Mathematical Recreations for the Programmable Calculator,* by Dean Hoffman of Auburn University and Lee Mohler of the University of Alabama. Needless to say, many of the topics that are covered in the book can be



The six components of the iteration diagram for squaring the first 100 integers readily adapted to computer programs.

The program is called RHOP because the sequence of numbers that eventually repeats itself resembles a piece of rope with a loop at one end. It also resembles the Greek letter rho (ρ). There are two variables in the program called *slow* and *fast*. Initially both variables are assigned the value of the starting number. The iterative cycle of the program includes just three instructions:

$$\begin{aligned} fast &\leftarrow fast \times fast \pmod{100} \\ fast &\leftarrow fast \times fast \pmod{100} \\ slow &\leftarrow slow \times slow \pmod{100} \end{aligned}$$

The operation mod 100 extracts the last two digits of the products. Note that the squaring is done twice on the number *fast* but only once on the number *slow*. *Fast* makes its way from the tail to the head of the rho twice as fast as *slow* does. Within the head *fast* catches up with *slow* by the time *slow* has gone partway around. The program exits from its iterative cycle when *fast* is equal to *slow*.

The attractor is identified by reiterating the squaring process for the number currently assigned to *slow*. When that number recurs, halt the program and print the intervening sequence of numbers.

I should be delighted to see readers' diagrams that explore the effects of iterative squaring on finite realms of varying size. The diagrams can be done on a computer or by hand. Discrete iteration is a newly developing mathematical field with applications in computer science, biomathematics, physics and sociology. Theorists might watch for a book on the subject by François Robert of the University of Grenoble.

The two-dimensional beings who inhabit the planet Arde are deeply grateful to the many readers who tried to improve the crossover circuit I described in May. That circuit is made up of 12 two-input *nand*-gates. I asked readers to find the minimum number of *nand*-gates—and *nand*-gates only—from which a crossover circuit can be built. Most of the circuits submitted have 10 gates, a mild improvement, but three readers found an eight-gate crossover [*see illustration on this page*].

In the eight-gate circuit there is one three-input *nand*-gate and two single-input *nand*-gates. The latter act as inverters, converting a 0 signal into a 1 signal and vice versa. The three readers who discovered the eight-gate solution are Eric D. Carlson of Cambridge, Mass., Dale C. Koepp of San Jose, Calif., and Steve Sullivan of Beaverton, Ore. I have passed their names along with the improved crossover circuit to my Ardean friends. Believe it or not, the same crossover circuit appears under U.S. Patent 3,248,573 (April 26, 1966). Robert L. Frank, who is a systems consultant in Birmingham, Mich., wrote that the patent was awarded to Lester M. Spandorfer of Cheltenham, Pa., Albert B. Tonik of Dresher, Pa., and Shimon Even of Cambridge, Mass.

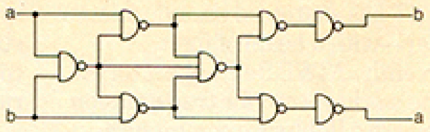
It seems natural to wonder whether the circuit actually appears in any present-day device. It is also natural to wonder whether there is an even smaller *nand* crossover. One supposes not.

C. Walter Johnson of Long Beach, Calif., wrote to me describing a wide variety of planar circuits that incorporate several types of gate. Apparently it is possible to build not only crossover circuits in two dimensions but also planar flip-flops. The flip-flops provide memory for a two-dimensional computer.

One-dimensional computers in the form of cellular automata have been investigated by Stephen Wolfram of the Institute for Advanced Study in Princeton. It is too early to say what contributions readers may have made to this field after reading Wolfram's "Glider Gun Guidelines," but I can pass along some initial reactions. A sin of commission sent a few readers off chasing gliders in the line automaton code-numbered 792. Wolfram and I meant to specify code 357. A sin of omission was my decision not to mention the line automata known to be capable of universal computation. I thought of describing such a line automaton, first constructed by Alvy Ray Smith in 1970. At the time Smith was a graduate student at Stanford University. I was afraid that the description of Smith's universal line automaton would unduly complicate the article: the automaton has 18 states ($k = 18$) and three-cell neighborhoods ($r = 1$).

Arthur L. Rubin of Los Angeles has made a sensible suggestion for defining the speed of light in an arbitrary line automaton. Rubin's suggestion corrects a defect in an earlier definition that sets the speed of light equal to one cell per unit of time. The old definition ignores the possibility that not all automata can attain such speeds. The revised speed of light is "the maximum speed of propagation of any impulse (say to the right)." The leading edge of the impulse is defined by the condition that only 0's can lie to its right. Rubin goes on to prove that the speed of light is 1/3 for the line automaton code-numbered 792.

In my May column I also asked whether the line automaton called



A crossover circuit with eight **nand**-*gates*

Ripple has a one-way glider gun. Gliders fired from such a gun would spew out unendingly to the right but never to the left. William B. Lipp of Milford, Conn., has made a simple and charming argument against the existence of such a gun. "Consider a pattern," he writes, "that never has nonzero values to the left of some block labeled 0. Observe that the leftmost nonzero value in the pattern must always be a 1. If it were a 2, the 2 would ripple to the left forever, thus contradicting the assumption that no nonzero entries lie to the left of block 0. But the leftmost 1 must become 0 on the next cycle, moving the left boundary of the pattern at least one block to the right." Thus either a glider ripples to the left or its gun is eaten away by 0's.

Other readers sought to show that Ripple is not capable of universal computation. For some automata one can prove a sufficient condition, namely that the halting problem is decidable. Ripple halts when all its cells contain 0's, but the halting conditions for any universal computing machine it might contain could be quite different.

Several readers attempted constructions of line automata capable of universal computation, among them Frank Adams of East Hartford, Conn., Jonathan Amsterdam of Cambridge, Mass., Kiyoshi Igusa of Brandeis University and Carl Kadie of East Peoria, Ill. The constructions are all straightforward and believable, but Kadie, not content with his one-dimensional automaton, went on to suggest a zero-dimensional one. It would consist of a single cell, and it seems reasonable to call it a point automaton. Readers with a theoretical bent might enjoy pondering the universality of a point automaton. Is it possible?

Alvy Ray Smith published his proof for the existence of a computation-universal line automaton in 1971, in *Journal of the Association for Computing Machinery*. One might have thought a career with such an auspicious start would today be blossoming in some well-known academic institution. Instead it has blossomed in a quite different setting: Smith is director of computer-graphics research for Lucasfilm, Ltd., in San Rafael, Calif. In a future column I hope to report on some of the amazing cinematic effects produced at the Lucasfilm laboratory.