17 December 1976, Volume 194, Number 4271 SCIENCE

Mathematics and Computer Science: Coping with Finiteness

Advances in our ability to compute are bringing us substantially closer to ultimate limitations.

Donald E. Knuth

A well-known book entitled *One, Two, Three, . . . Infinity* was published by Gamov about 30 years ago (*1*), and he began by telling a story about two Hungarian noblemen. It seems that the two gentlemen were out riding, and one suggested to the other that they play a game: Who can name the largest number. "Good," said the second man, "you go first." After several minutes of intense concentration, the first nobleman announced the largest number he could think of: "Three." Now it was the other man's turn, and he thought furiously, but after about a quarter of an hour he gave up. "You win," he said.

In this article I will try to assess how much further we have come, by discussing how well we can now deal with large quantities. Although we have certainly narrowed the gap between three and infinity, recent results indicate that we will never actually be able to go very far in practice. My purpose is to explore relationships between the finite and the infinite, in the light of these developments.

Some Large Finite Numbers

Since the time of Greek philosophy, men have prided themselves on their ability to understand something about infinity; and it has become traditional in some circles to regard finite things as essentially trivial, too limited to be of any interest. It is hard to debunk such a notion, since there are no accepted standards for demonstrating that something is interesting, especially when something finite is compared with something transcendent. Yet I believe that the climate of thought is changing, since finite processes are proving to be such fascinating objects of study.

In the first place, it is important to understand that finite numbers can be extremely large. Let us start with some very familiar and fairly small numbers: the value of xn is $x + x + \cdots + x$, added n times. Similarly we can define a number I shall write as $x\uparrow n$, which means $xx\cdots x$ multiplied n times. For example, $10\uparrow 10 = 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10\cdot 10 = 10{,}000{,}000{,}000$ is 10 billion; this is usually written 10^{10}, but it will be clear in a minute why I prefer to use an upward arrow. In fact, the next step uses two arrows

$$x\uparrow\uparrow n = x\uparrow(x\uparrow(\cdots\uparrow x)\cdots))$$

where we take powers n times. For example

$$10\uparrow\uparrow 10 = 10^{10^{10^{10^{10^{10^{10^{10^{10^{10}}}}}}}}}$$

$$= 1 \text{ followed by } 10^{10^{10^{10^{10^{10^{10^{10^{10^{10}}}}}}}}} \text{ zeros}$$

This is a pretty big number; at least, if a monkey sits at a typewriter and types at random, the average number of trials before he types perfectly the entire text of Shakespeare's *Hamlet* would be much, much less than this: it is merely a 1 followed by about 40,000 zeros. The general rule is

$$x\overbrace{\uparrow\uparrow\ldots\uparrow}^{k\text{ arrows}}n = \underbrace{x\overbrace{\uparrow\ldots\uparrow}^{k-1}(x\overbrace{\uparrow\ldots\uparrow}^{k-1}(\cdots\overbrace{\uparrow\ldots\uparrow}^{k-1}x)\cdots))}_{n\text{ times}}$$

Thus, one arrow is defined in terms of none, two in terms of one, three in terms of two, and so on.

In order to see how these arrow functions behave, let us look at a very small example

$$10\uparrow\uparrow\uparrow\uparrow 3$$

This is equal to

$$10\uparrow\uparrow\uparrow(10\uparrow\uparrow\uparrow 10)$$

so we should first evaluate $10\uparrow\uparrow\uparrow 10$. This is

$$10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow 10)))))))$$

and that is

$$10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow 10^{10^{10^{10^{10^{10^{10^{10^{10}}}}}}}})))))))$$

$$= 10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow 10^{10^{\cdot^{\cdot^{\cdot^{10}}}}})))))))$$

where the stack of 10's is $10\uparrow\uparrow 10$ levels tall. We take the huge number at the right of this formula, which I cannot even write down without using the arrow notation, and repeat the double-arrow operation, getting an even huger number, and then we must do the same thing again and again. Let us call the final

The author is professor of computer science at Stanford University, Stanford, California 94305.

result $\mathcal{H}$. (It is such an immense number, we cannot use just an ordinary letter for it.)

Of course we are not done yet, we have only evaluated $10\uparrow\uparrow\uparrow\uparrow 10$; to complete the job we need to stick this gigantic number into the formula for $10\uparrow\uparrow\uparrow\uparrow 3$, namely

$$10\uparrow\uparrow\uparrow\uparrow 3 = 10\uparrow\uparrow\uparrow \mathcal{H}$$
$$= \underbrace{10\uparrow\uparrow(10\uparrow\uparrow(10\uparrow\uparrow\cdots\uparrow\uparrow(10\uparrow\uparrow 10)\cdots))}_{\mathcal{H}\text{ times}}$$

The three dots " ··· " here suppress a lot of detail—maybe I should have used four dots. At any rate it seems to me that the magnitude of this number $10\uparrow\uparrow\uparrow\uparrow 3$ is so large as to be beyond human comprehension.

On the other hand, it is very small as finite numbers go. We might have used $\mathcal{H}$ arrows instead of just four, but even that would not get us much further—almost all finite numbers are larger than this. I think this example helps open our eyes to the fact that some numbers are very large even if they are merely finite. Thus, mathematicians who stick mostly to working with finite numbers are not really limiting themselves too severely.

Realistic Numbers

This discussion has set the stage for the next point I want to make, namely that our total resources are not actually very large. Let us try to see how big the known universe is. Archimedes began such an investigation many years ago, in his famous discussion of the number of grains of sand that would completely fill the earth and sky; he did not have the benefit of modern astronomy, but his estimate was qualitatively the same as what we would say today. The distance to the farthest observable galaxies is thought to be at most about 10 billion light years. On the other hand, the fundamental nucleons that make up matter are about 10^{-12} centimeter in diameter. In order to get a generous upper bound on the size of the universe, let us imagine a cube that is 40 billion light years on each side, and fill it with tiny cubes that are smaller than protons and neutrons, say 10^{-13} cm on each side (see Fig. 1). The total number of little cubes comes to less than 10^{125}. We might say that this is an "astronomically large" number, but actually it has only 125 digits.

Instead of talking only about large numbers of objects, let us also consider the time dimension. Here the numbers are much smaller; for example, if we take as a unit the amount of time that

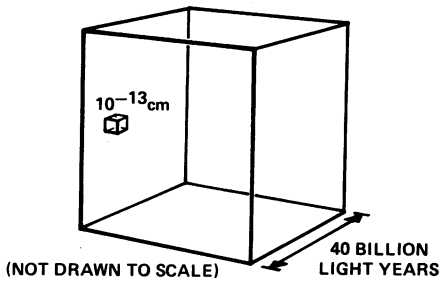


Fig. 1. The known universe fits inside this box.

light rays take to travel 10^{-13} cm, the total number of time units since the dawn of the universe is only one fourth the number of little cubes along a single edge of the big cube in Fig. 1, assuming that the universe is 10 billion years old.

Coming down to earth, it is instructive to consider typical transportation speeds.

Snail	0.006	mile/hour
Man walking	4	mile/hour
U.S. automobile	55	mile/hour
Jet plane	600	mile/hour
Supersonic jet	1200	mile/hour

I would never think of walking from California to Boston, but the plane flight is only 150 times faster. Compare this to the situation with respect to the following computation speeds, given 10-digit numbers.

Man (pencil and paper)	0.2/sec
Man (abacus)	1/sec
Mechanical calculator	4/sec
Medium-speed computer	200,000/sec
Fast computer	200,000,000/sec

A medium-fast computer can add 1 million times faster than we can, and the fastest machines are 1000 times faster yet. Such a ratio of speeds is unprecedented in history: consider how much a mere factor of 10 in speed, provided by the automobile, has changed our lives, and note that computers have increased our calculation speeds by six orders of magnitude; that is more than the ratio of the fastest airplane velocity to a snail's pace.

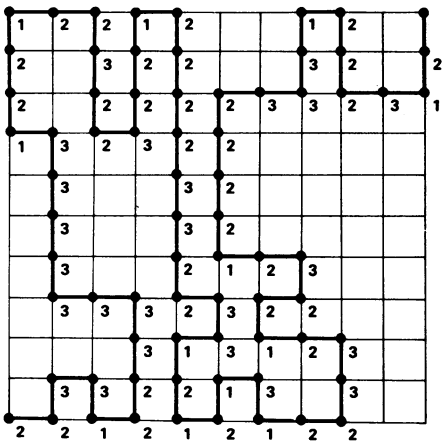


Fig. 2. A "random" path from the lower left corner to the upper right corner of a 10 × 10 grid.

I do not mean to claim that computers do everything a million times faster than people can; mere mortals like us can do some things much better. For example, you and I can even recognize the face of a friend who has recently grown a moustache; and for tasks like filing, a computer may be only ten or so times faster than a good secretary. But when it comes to arithmetic, computers appear to be almost infinitely fast compared with people.

As a result, we have begun to think about computational problems that used to be unthinkable. Our appetite for calculation has caused us to deal with finite numbers much larger than those we considered before, and this has opened up a rich vein of challenging problems, just as exciting as the problems about infinity which have inspired mathematicians for so many centuries.

Of course, computers are not infinitely fast, and our expectations have become inflated even faster than our computational capabilities. We are forced to realize that there are limits beyond which we cannot go. The numbers we can deal with are not only finite, they are very finite, and we do not have the time or space to solve certain problems even with the aid of the fastest computers. Thus, the theme of this article is coping with finiteness: What useful things can we say about these finite limitations? How have people learned to deal with the situation?

Advances in Technology and Techniques

During the last 15 years computer designers have made computing machines about 1000 times faster. Mathematicians and computer scientists have also discovered a variety of new techniques, by which many problems can now be solved enormously faster than they could before. I will present several examples of this; the first one, which is somehow symbolic of our advances in arithmetic ability, is the following factorization of a very large number, completed in 1970 by Morrison and Brillhart (*2*).

340,282,366,920,
938,463,463,374,607,431,768,211,457
= 5,704,689,200,685,129,054,721 ×
59,649,589,127,497,217

The point, of course, is not simply to compute the exact 39-digit product of these two large numbers; that is trivial and takes only a few millionths of a second. The problem is to start with the big 39-digit number and to discover its prime factors. (The big number is $2^{128} + 1$, and its factors are of use, for example, in the design of codes of a type used for space communications.) The number of microseconds per year is only 31,556,952,000,000, a 14-digit number, so even if we could test 1 million factors every second it would take about 2000 years to discover the smaller factor. The factorization actually took about 1½ hours of computer time; it was achieved by a combination of sophisticated methods representing a culmination of mathematical developments that began about 160 years earlier.

Latin Squares

Now let us look at another kind of example. Here is a so-called latin square of order 8, an arrangement of eight numbers in eight rows and eight columns so that each number appears in each row and each column.

1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1

On top of this square we can overlay another latin square of order 8, using italic numbers; again, there is one italic digit of every kind in every row and in every column.

1*1*	2*2*	3*3*	4*4*	5*5*	6*6*	7*7*	8*8*
2*3*	1*4*	4*1*	3*2*	6*7*	5*8*	8*5*	7*6*
3*5*	4*6*	1*7*	2*8*	7*1*	8*2*	5*3*	6*4*
4*7*	3*8*	2*5*	1*6*	8*3*	7*4*	6*1*	5*2*
5*4*	6*3*	7*2*	8*1*	1*8*	2*7*	3*6*	4*5*
6*2*	5*1*	8*4*	7*3*	2*6*	1*5*	4*8*	3*7*
7*8*	8*7*	5*6*	6*5*	3*4*	4*3*	1*2*	2*1*
8*6*	7*5*	6*8*	5*7*	4*2*	3*1*	2*4*	1*3*

These two latin squares are called orthogonal, since the superposition shows that every pair of roman and italic numbers occurs exactly once. Thus we have roman 1 with italic *1* (in the upper left corner), roman 1 with italic *2* (near the lower right corner), and so on; all 8 × 8 possibilities appear. Latin squares and orthogonal latin squares are commonly used in the design of statistical experiments and for such things as crop rotation.

The great 18th-century mathematician Euler showed how to construct pairs of orthogonal latin squares of all sizes except for order 2, 6, 10, 14, 18, and so on, and he stated his belief that orthogonal latin squares of these missing orders do not exist (*3*). It is easy to verify this for order 2; and in 1900, an exhaustive analysis by a French mathematician (*4*) showed that orthogonal latin squares of order 6 are indeed impossible. About 20 years later, an American mathematician (*5*) published a proof that Euler was right in the remaining cases 10, 14, 18, . . . ; but unfortunately his "proof" had a serious flaw so the question was still not settled. Finally computers were invented, and an attempt was made to test Euler's conjecture in the smallest remaining case, order 10.

In 1952, a group of mathematicians at the University of California at Los Angeles (UCLA) decided to see if there were any latin squares orthogonal to the following one of order 10.

0	1	2	3	4	5	6	7	8	9
1	8	3	2	5	4	7	6	9	0
2	9	5	6	3	0	8	4	7	1
3	7	0	9	8	6	1	5	2	4
4	6	7	5	2	9	0	8	1	3
5	0	9	4	7	8	3	1	6	2
6	5	4	7	1	3	2	9	0	8
7	4	1	8	0	2	9	3	5	6
8	3	6	0	9	1	5	2	4	7
9	2	8	1	6	7	4	0	3	5

This particular square was selected more or less at random, using a procedure analogous to one discussed in the next example below; the probability that the above latin square will be generated (*6*) turns out to be about 10^{-26}, so I imagine that there are extremely many 10 × 10 latin squares, something like 10^{26} at least. However, the computer at UCLA ran for many hours trying to find an orthogonal mate for this square; finally, having produced no answers, it was shut off (*7*). This was consistent with Euler's conjecture that no mates exist, but the investigators realized that several hundred more years of calculation would be required to show this exhaustively—and then they would have to try to find mates for the other 10^{26} or so initial squares.

The method used in this experiment was to look for a mate by filling in the entries row by row, one entry at a time in all possible ways, without violating the definition of orthogonal latin squares. Furthermore, they used the fact that the leftmost column of the orthogonal mate can be assumed to contain the digits 0 to 9 in order. Five years later Parker (*8*) discovered a far better way to look for orthogonal mates. His idea was to find all ways to put ten 0's into an orthogonal mate for a particular square; this means finding one entry in each row and each column so that no two entries contain the same digit. This is a much easier problem, and it turned out that there were roughly 100 ways to do it, using any cell in the first column. The remaining problem is to combine a solution for the 0's with a solution for the 1's and a solution for the 2's, and so forth, and again this is comparatively simple. Parker was able to deduce that there is exactly one latin square orthogonal to the one studied at UCLA, namely the italic digits in the following array.

0*0*	1*2*	2*8*	3*5*	4*9*	5*4*	6*7*	7*3*	8*6*	9*1*
1*1*	8*7*	3*4*	2*9*	5*3*	4*6*	7*5*	6*0*	9*2*	0*8*
2*2*	9*5*	5*6*	6*4*	3*8*	0*7*	8*0*	4*1*	7*9*	1*3*
3*3*	7*6*	0*9*	9*0*	8*4*	6*5*	1*8*	5*2*	2*1*	4*7*
4*4*	6*8*	7*1*	5*7*	2*5*	9*3*	0*6*	8*9*	1*0*	3*2*
5*5*	0*1*	9*7*	4*8*	7*0*	8*2*	3*9*	1*4*	6*3*	2*6*
6*6*	5*9*	4*0*	7*2*	1*7*	3*1*	2*3*	9*8*	0*4*	8*5*
7*7*	4*3*	1*5*	8*1*	0*2*	2*0*	9*4*	3*6*	5*8*	6*9*
8*8*	3*0*	6*2*	0*3*	9*6*	1*9*	5*1*	2*7*	4*5*	7*4*
9*9*	2*4*	8*3*	1*6*	6*1*	7*8*	4*2*	0*5*	3*7*	5*0*

And the total time for his program to be completed, on a slow computer in 1959, was less than 1 minute.

This example, together with the previous example about factoring, illustrates an important point: we should never expect that the first way we try to do something on a computer is the best way. Good programming is much more subtle than that; chances are that an expert can find a method that will go considerably faster than that of a novice, especially in combinatorial problems where there have been significant advances in techniques during recent years. By analyzing Parker's method statistically, I estimate that his approach runs about 100 billion times faster than the original method used by the extremely competent mathematicians who studied this problem at UCLA; that is 11 orders of magnitude faster, because of a better idea.

By now many sets of orthogonal latin squares of order 10 have been found, and orthogonal pairs are known to exist for all orders greater than 6. But computers were of little help in discovering these facts; the constructions were discovered by hand (by Parker himself in many cases), generalizing from patterns observed in the smaller cases (*9*). For order 14 the problem is so much larger that even Parker's method would no longer be fast enough to search for all orthogonal mates by computer. This illustrates another point about combinatorial problems: the computation time often increases greatly when the size of the input to the problem has gone up only slightly.

Counting the Paths on a Grid

The next examples are all based on a single diagram, namely a grid of 100 squares; it is the diagram we would obtain if we drew boxes around the ele-

ments of a 10 × 10 latin square. (Incidentally, there are many possible examples that illustrate the points I wish to make, so it was necessary for me to find some way to narrow down the selection. Since a 10 × 10 array fits nicely on a page, I have decided to stick mostly to examples that are based somehow on this one diagram.)

First let us consider how many ways there are to go along the lines of such a grid from the lower left corner to the upper right corner, without touching the same point twice. Problems like this have been studied by chemists and physicists concerned with the behavior of large molecules (*10*); it seems to be a difficult problem, and no way is known to calculate the exact number of such paths in a reasonable amount of time. However, it is possible to obtain approximate solutions which are correct with high probability.

The idea is to construct a "random" path from the starting point to the finishing point. First we must go up or to the right; by flipping a coin or rolling some dice we might decide to go right. Again there are two choices, and half the time we will go up. From here there are three possibilities, and we may choose from these at random, say to the right. And so on. Figure 2 shows the first random path I generated in this way. At each choice point of Fig. 2, I have written the number of alternatives present when the path got that far. For example, the 1's at the edges mean that there is only one way to go, since the other way either is already occupied or leads into a trap.

The probability that this particular path would be obtained by such a random procedure is the product of all the individual probabilities at each choice point, namely

$$\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{3}\cdot\frac{1}{3}\cdot\ldots\cdot\frac{1}{3}\cdot\frac{1}{1}\cdot\frac{1}{2}$$

$$= 2^{-34}3^{-24}$$

$$= 1/4{,}852{,}102{,}490{,}441{,}335{,}701{,}504$$

about one chance in 5×10^{21}. So I am pretty sure that you have never seen this particular path before, and I doubt if I will ever generate it again.

In a similar vein, it is interesting to note that the great Mozart wrote a considerable amount of music that has never yet been performed. In one of his more playful moments, he specified 11 possibilities for each of the 16 bars of a waltz (*11*); the idea was that people from the audience should roll dice 16 times, obtaining a sequence of 16 numbers between 2 and 12 inclusive, and the performers would play the 16 bars corresponding to these respective rolls. The total number of ways to play Mozart's dice waltz is $2 \times 11^{14} = 759{,}499{,}667{,}966{,}482$ (*12*); so it is safe to say that fewer than one out of every million of Mozart's melodies will ever be heard by human ear.

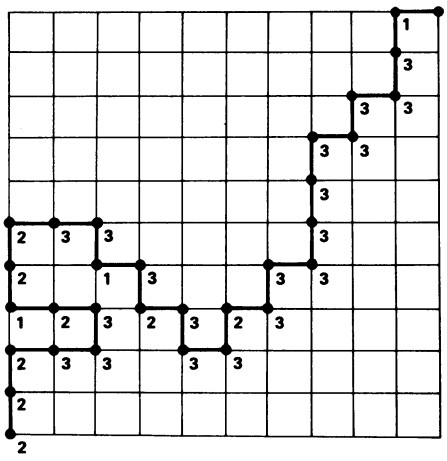


Fig. 3. A second path, which would be obtained with probability $\approx 3 \times 10^{-12}$.

Actually I have a phonograph record that contains 36 randomly selected waltzes from Mozart's scheme (*13*), and after hearing the fifth one or so I began to feel that the rest all sounded about the same. We might suspect that a similar thing will happen in this random path problem; all random paths from lower left to upper right might tend to look approximately like the first few. Figure 3 shows the second path I generated by making random choices; note that this one has quite a different character, and the strange thing is that the probability of obtaining it is more than ten orders of magnitude larger than we saw before. But still the probability is "negligibly small."

The third path I generated in this way decided to get into a corner and to hug the edge. The fourth one had its own twist; and the fifth was reminiscent of the first. These paths are shown in Fig. 4; of course I am displaying here each path exactly as I obtained it, not suppressing any that were uninteresting or unexpected, because the experiment must be unbiased.

The difference between this game and Mozart's dice music is that we know of no way to generate a truly random path, in the sense that each path should occur with the same probability. Although we have seen that each path occurs with extremely small probability, virtually zero, the actual probabilities differ from each other by many orders of magnitude.

If we want to estimate the total number of possible paths, solely on the basis of these data, a theorem of statistics tells us that the best estimate is obtained by using the average value of the reciprocals of the probabilities observed. Thus, although three of these five paths had probabilities around 10^{-11}, suggesting that there are about 10^{11} possible paths, the much lower probabilities in the other two cases imply that it is much better to guess that there are about 10^{22} paths in all. Based on the five experiments I have described, the best estimate of the average length of path will be about 70; and the best estimate of the chance that the point in the middle occurs somewhere on the path is that it almost always occurs, even though three-fifths of the experiments said the opposite. When large numbers like this are involved, we get into paradoxical situations, where the rules of statistics tell us that the best estimates are made by throwing away most of our data.

As you might expect, five experiments are not enough to determine the answers reliably. But by using a computer to generate several thousand random paths in the same way, I am fairly confident that the total number of possible paths from lower left to upper right is $(1.6 \pm 0.3) \times 10^{24}$, and that the average length of path is 92 ± 5. Conflicting evidence was obtained about the chance of hitting the center, but it seems that 81 ± 10 percent of all paths do hit the center point. Of course, I have only generated an extremely small fraction of these paths, so I cannot really be sure; perhaps nobody will ever know the true answer.

The Shortest Paths

For the next examples we will add weights to the lines in the grid. The basic diagram is shown in Fig. 5, where a random digit has been placed beside each line; these digits may be thought of as the lengths of roads between adjacent points of intersection. Thus, there are three roads of length 4 on the bottom line, and the upper part of the diagram contains three adjacent roads of length 0. Actually I must admit that the sequences of numbers are not completely arbitrary; for example, the reader might recognize 1.414213562... in the top line as the square root of 2, and π appears down the second column. For our purposes these digits will be random enough.

The first problem we might ask about such a network of roads is: What is the shortest route from the lower left corner

to the upper right corner? We have estimated that there are some 10^{24} possible paths, and we might want to know which of these is shortest, using the given lengths.

Fortunately we do not have to try all possible paths to find the shortest; there is a simple method due to Dijkstra (*14*) which can be used to solve this problem by hand in less than half an hour. The answer (see Fig. 6) is a curious sort of path, which might very well be missed if one does not use a systematic method; it is the only way to go from southwest to northeast in a path of length 43.

The idea underlying Dijkstra's method is rather simple. Suppose that at some stage we have found all positions at distance 20 or less, say, from the southwest corner. By looking at the roads connecting these points to the others it will be easy to see which points will be at distance 21, and so on. You can imagine a fluid spreading over the diagram at the rate of one unit of length per minute.

Connecting Points in a Network

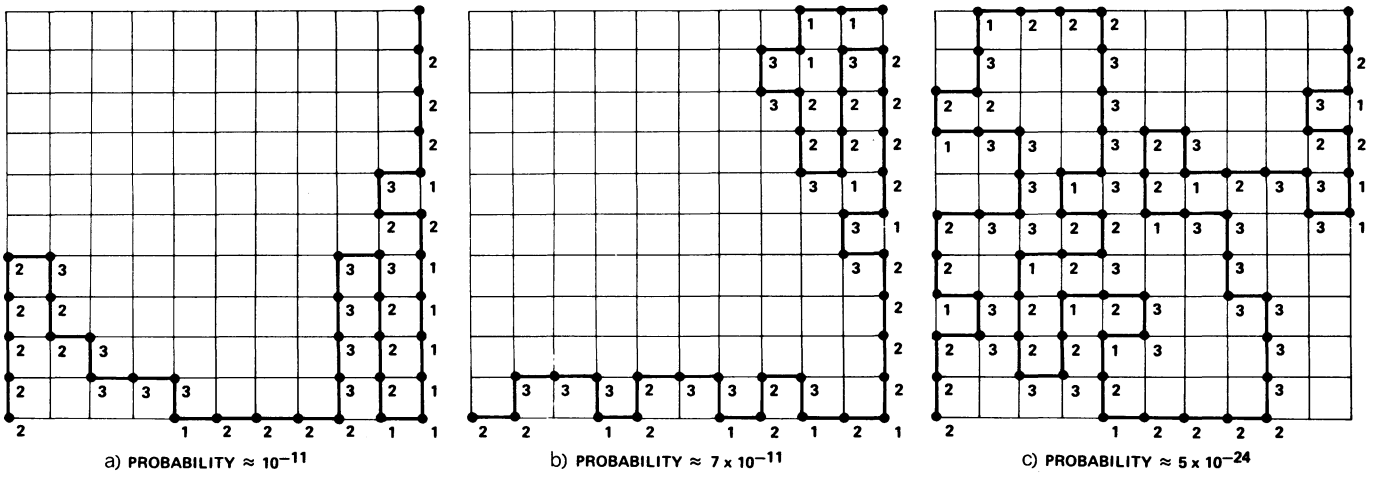
The next problem is somewhat harder. Suppose we want to construct electrical connections between all four of the corner points in Fig. 5: What is the shortest electrical network joining these four points, sticking to the lines and distances shown? Such a network is usually called a Steiner tree (*15*), and Fig. 7 shows the shortest possible one.

The number of possible Steiner trees connecting the four corners is much larger than the number of paths, but still I am sure that this is the shortest. In this case I do not know how to compute the shortest by hand, but a properly programmed computer can do it in a few seconds.

We say that we have a "good" algorithm for some problem if the time to solve it increases only as a polynomial in the size of the inputs; in other words, if doubling the size of the problem increases the solution time by at most a constant factor. There is a good algorithm to find Steiner trees connecting up to five points; it takes roughly n^3 steps, where n is the total number of points in the network of roads (*16*). But if we want to connect larger numbers of points by Steiner trees, the computation rapidly gets larger; and when the number of points to be connected is, say, as large as $n/10$, no good algorithm is known.

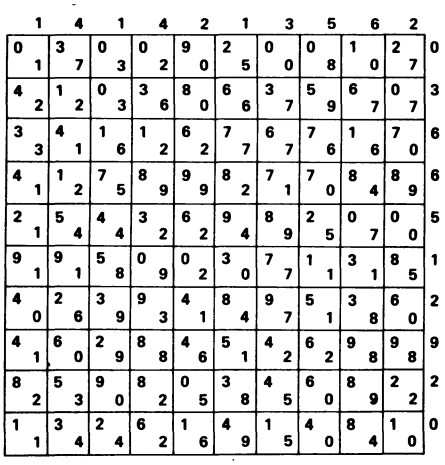
On the other hand, when our job is to find the shortest way to connect up all n of the points in the network, a good algorithm is available, again one that is so good it can be performed by hand in half an hour.

A minimal connection of all points in a network is called a spanning tree, and in the particular network we are considering it is possible to prove that the number of possible spanning trees is really huge, more than 4×10^{52}; in fact, the exact number (*17*) is 40,325,021,721,404, 118,513,276,859,513,497,679,249,183,623, 593,590,784



a) PROBABILITY ≈ 10^{-11} b) PROBABILITY ≈ 7×10^{-11} c) PROBABILITY ≈ 5×10^{-24}

Fig. 4. Three more randomly generated paths, with their associated probabilities.



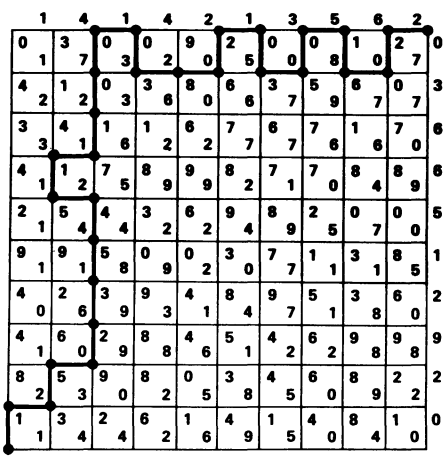
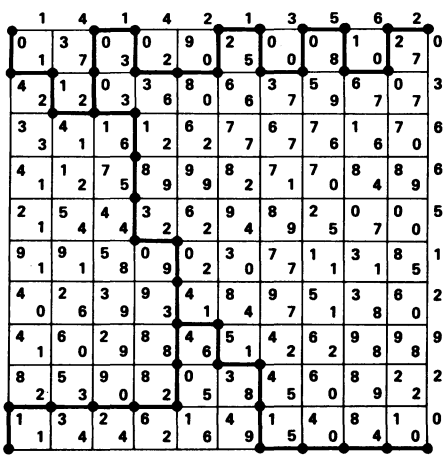
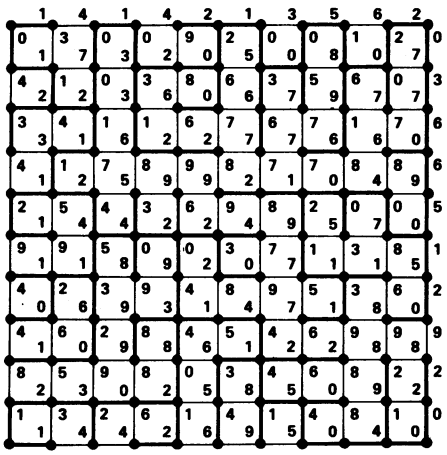
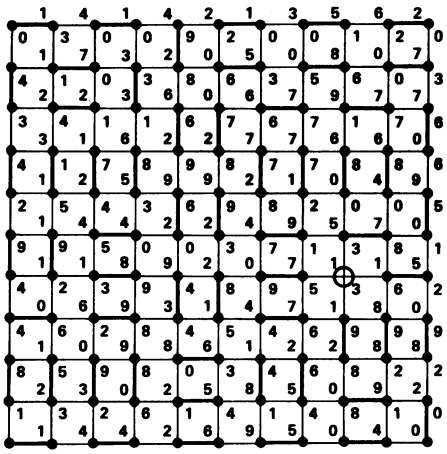


Fig. 5 (left). Network to be used in subsequent examples, based on 20 mathematical constants. Fig. 6 (middle). The shortest route from lower left to upper right in this network of roads. Fig. 7 (right). The shortest way to connect the four corners.







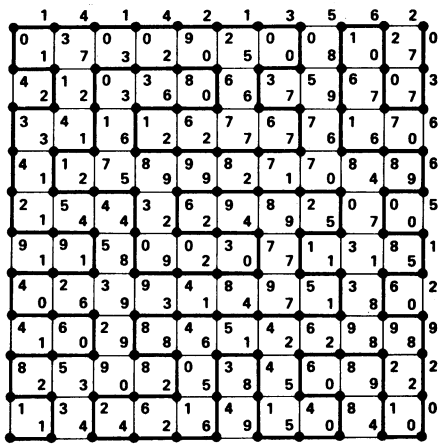


Fig. 8 (left). A minimum spanning tree. Fig. 9 (middle). The best choice of 60 nonoverlapping lines in the diagram. Fig. 10 (right). A shortest path from lower left to upper right, touching each point just once.

Yet we can find the best one, in a remarkably easy way discovered by Kruskal (*18*): simply consider all the lines one by one in order of increasing length, starting with the shortest one, then the next shortest, and so on. In case of ties between lines of the same length, use either order. The rule is to include each line in the spanning tree if and only if it connects at least two points that are not connected by a path of previously selected lines. This is called the greedy algorithm because it is based on the idea of trying the best conceivable possibilities first. Such a policy does not always solve a combinatorial problem—we know that greed does not always pay off in the long run—but in the case of spanning trees the idea works perfectly (see Fig. 8).

Maximum Matching

Another problem on this network for which a good algorithm is available is to choose 60 of the lines with the maximum possible sum, no two overlapping. We may think now of the points as people, instead of as cities, and the numbers now measure the amount of happiness generated between one person and his or her neighbor. The idea is to pair off the people so as to get the maximum total happiness. If men and women alternate in the diagram, with men at the corners, there will be 61 men and 60 women in all, so one man will have no partner; he makes a personal sacrifice for the greater good of the group as a whole. There are exactly 1,801,272,981,919,008 ways to do such a pairing, according to a mathematical theory worked out to solve a physical problem about crystals (*19*); Fig. 9 exhibits the best one.

It turns out that the circled man is the best to omit, and the others should pair up in this way. Once again we are able to find the optimum solution in 1 or 2 seconds on a computer if we use a suitable algorithm, even though the number of possible arrangements is far too large to examine exhaustively. In this case the algorithm is somewhat more subtle than the ones I have discussed earlier, but it is based on simple ideas. First we add a "dummy" woman who will be paired with the man who gets no real woman. The happiness rating is 0 between the dummy woman and every man. Then if we add or subtract some number from all the happiness ratings touching any particular person, the solution to the problem does not change. A clever way of adjusting these scores can be used so that all 61 of the ratings for the couples matched here are 9, and all the other ratings are 9 or less (*20*).

An Apparently Harder Problem

From these examples, one might get the idea that a good algorithm can be found for virtually any combinatorial problem. Unfortunately this does not appear to be true, although I did want to demonstrate that considerable progress has been made toward finding good methods. The next problem seems to be much harder: What is the shortest path from the lower left corner to the upper right corner that passes through all 121 points of the grid exactly once?

This is called the traveling salesman problem, because we might think of a salesman who wants to visit each city with minimum travel time. The problem arises frequently in industry—for example, when the goal is to find the best order in which to do n jobs, based on the costs of changing from one job to another. But it has resisted all attacks; we know how to solve medium-sized problems, but the algorithms are not good in the technical sense since the running time goes up rapidly on large cases.

The traveling salesman's path shown in Fig. 10 is as short as possible, and it required several minutes of computer time to verify the fact. To my knowledge, this is the largest network for which the traveling salesman problem has yet been solved exactly. I used a method suggested in 1971 by Held and Karp (*21*), based on a combination of ideas we have applied to other problems: it is possible to add or subtract numbers from all the lines which touch a particular point, without changing the shape of the minimum tour, and we can use the greedy algorithm to construct a minimum spanning tree for the changed distances. The minimum spanning tree is no longer than the shortest tour, since every tour is a spanning tree; but by properly modifying the distances we can make the minimum spanning tree very nearly a tour, so comparatively few possibilities need to be tried. I extended the Held and Karp method to take advantage of the fact that each point has at most four neighbors. In this way it was possible to verify at reasonable cost that this tour is optimum; but if I were faced with a larger problem, having say twice as many points to visit, there would be no known method to get the answer in a reasonable amount of time.

In fact, it may well be possible in a few years to prove that no good algorithm exists for the traveling salesman problem. Since so many people have tried for so many years to find a good algorithm, without success, the trend is now to look

for a proof that success in this endeavor is impossible. It is analogous to the question of solving polynomial equations: quadratic equations were resolved in ancient Mesopotamia, and the solution of cubic and quartic equations was found at the beginning of the Renaissance, but nobody was able to solve arbitrary equations of the fifth degree. Finally, during the first part of the 19th century, Abel and Galois proved conclusively that there is no way to solve fifth degree equations in general, using ordinary arithmetic (*22*). It is now believed that there is no good algorithm for the general traveling salesman problem, and we are awaiting another Abel or Galois to prove it.

In support of this belief, several important things have already been proved, notably that the traveling salesman problem is computationally equivalent to hundreds of other problems of general interest (*23*). If there is a good algorithm for any one of these problems, which for technical reasons are called NP-complete problems, then there will be good algorithms for all the NP-complete problems. Thus, for example, a good algorithm for the traveling salesman problem would lead immediately to a good algorithm for many other difficult problems, such as the optimum scheduling of high school classes, the most efficient way to pack things into boxes, or the best Steiner trees connecting a large number of points. A good solution to any one of these problems will solve them all, so if any one of them is hard they all must be.

A Provably Harder Problem

In recent years, certain problems have, in fact, been shown to be intrinsically hard, in the sense that there will never be a fast way to solve them. Probably the most interesting example of this type was developed in 1974 by A. Meyer and L. J. Stockmeyer (*24*). The problem is to decide whether or not certain logical statements about whole numbers 0,1,2,.... are true or false, even when the form of these statements is severely restricted.

Here are some examples of the sort of statements we must deal with.

$$048 \leq 1063$$

a statement which is clearly true.

$$\forall n\, \exists m(m < n+1)$$

This is logical shorthand that can be translated as follows, for people who are not familiar with the new math: "For all numbers n there exists a number m such that m is less than $n + 1$." It is clearly a true statement, since we may take m equal to n.

$$\forall n\, \exists m(m+1 < n)$$

"For all numbers n there exists a number m such that $m + 1$ is less than n." This statement is false, for if $n = 0$ there is no number less than zero; we are considering only statements about nonnegative numbers.

The next example is a little more complicated.

$$\forall x\, \forall y(y \geq x+2 \rightarrow \exists z(x < z \wedge z < y))$$

"For all numbers x and all numbers y, if y is greater than or equal to $x + 2$ then there exists a number z such that x is less than z and z is less than y." In other words, if y is at least 2 more than x, there is a number z between x and y, and this is obviously true.

Finally we can also make statements about sets of numbers; for example

$$\forall S(\exists x(x \in S) \rightarrow$$
$$\exists y(y \in S \wedge \forall z(z \in S \rightarrow y \leq z))$$

"For all sets S of numbers, if there exists a number x such that x is in S then there exists a number y such that y is in S and for all numbers z in S we have $y \leq z$." Informally, the statement says that every set of numbers which is not empty has a smallest element, and this is true.

The logical statements we shall be concerned with cannot be essentially any harder than these examples; they may not involve subtraction, multiplication, or division; they cannot even involve addition except addition of a constant. (They cannot involve the formula $x + y$.) Thus the statements must be very simple—much, much simpler than those used every day by mathematicians constructing proofs of theorems.

According to a well-known theorem of Büchi (*25*), it is possible to decide in a finite number of steps whether or not any statement of the simple kind we have described is true or false, even though these logical statements may concern infinitely many cases.

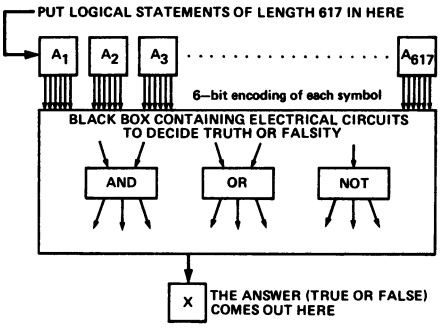


Fig. 11. Electrical network to decide the correctness of logical statements containing 617 characters or less.

But the new theorem says that it is impossible actually to do this in the real world, even if we limit ourselves to statements that can be written in no more than 617 symbols: "No realistic algorithm will ever be able to decide truth or falsity for arbitrary given statements of length 617 or less."

In order to understand exactly what this theorem means, we have to know what it means to speak of a "realistic" algorithm. The theorem of Meyer and Stockmeyer is based on the fact that anything that can be done by computer can be done by constructing an electrical network, and so they envisage a setup like that shown in Fig. 11. At the top of such a device, one can insert any statement whose truth is to be tested. The logical language involved here makes use of 63 different symbols, including upper and lower case letters and a blank symbol, so we can place the statement (followed if necessary by blanks) into a sequence of 617 positions. Each position is converted into six electrical pulses, whose configuration of "on" and "off" identifies the corresponding character; thus, the letter X might be represented by the six pulses "off, on, on, off, off, on." The resulting 6×617 pulses now enter an electrical network or "black box" consisting of AND, OR, and NOT circuits; AND produces a signal that is "on" only when both inputs to AND are "on," OR produces a signal that is "on" when either or both of its inputs are "on," and NOT changes "on" to "off" and vice versa. At the bottom of the network, a pulse comes out which is "on" or "off" according to whether the given logical statement of length 617 was true or false.

According to Büchi's theorem, it is possible to construct such an electrical network with finitely many components, in a finite amount of time. But Meyer and Stockmeyer (*24*) have proved that every such network must use at least 10^{125} components, and we have seen that this is much larger than the number of protons and neutrons in the entire known universe.

Thus it is hopeless to find an efficient algorithm for this finite problem. We have to face the fact that it can never be done—no matter how clever we may become, or how much money and energy is invested in the project.

What should we do in the face of such limitations? Whenever something has been proved impossible, there is an aspect of the human spirit that encourages us to find some way to do it anyway. In this particular case, we might try the following sneaky approach: We could build an electric circuit which gives the correct answer in all simple cases and which gives a random answer, true or false, in the other cases. Since the problem is so hard, nobody will be able to know the difference.

But this is obviously unsatisfactory. A better approach would be to distinguish between levels of truth; for example, the answer might be "true," "false," or "maybe." And we could give various shades of "maybe," saying perhaps that the statement is true in lots of cases.

Let us consider the traveling salesman problem again. It is reasonably likely that, some day, somebody will prove that no good algorithm exists for this problem. If so, that will be a truly great theorem; but what should we do when we actually need to solve such a problem?

The answer, of course, is to settle for a tour that is not known to be the shortest possible one, but is pretty close. It has recently been observed that we can quickly find a traveling salesman's tour that is guaranteed to be no worse than 50 percent longer than the shortest possible tour, if the distances satisfy the triangle inequality. And algorithms have recently been developed for other problems that give answers which are probably correct, where the degree of probability can be specified, but the answer is not certain.

In this way, computer scientists and mathematicians have been learning how to cope with our finite limitations.

Summary

By presenting these examples, I have tried to illustrate four main points.

1) Finite numbers can be really enormous, and the known universe is very small. Therefore the distinction between finite and infinite is not as relevant as the distinction between realistic and unrealistic.

2) In many cases there are subtle ways to solve very large problems quickly, in spite of the fact that they appear at first to require examination of too many possibilities.

3) There are also cases where we can prove that a fairly natural problem is intrinsically hard, far beyond our conceivable capabilities.

4) It takes a good deal of skill to decide whether a given problem is in the easy or hard class; but even if a problem does turn out to be hard there are useful and interesting ways to change it into one that can be done satisfactorily.

References and Notes

1. G. Gamov, *One, Two, Three, . . . Infinity* (Viking, New York, 1947).
2. M. A. Morrison and J. Brillhart, *Math. Comput.* **29**, 183 (1975).
3. L. Euler, *Verh. Uitg. Zeeuwsch. Genoot. Wet. Vlissingen* **9**, 85 (1782); *Leonardi Euleri Opera Omnia* 7, 291 (1923).
4. G. Tarry, *Mathésis* **20** (Suppl.), 23 (1900).
5. H. F. MacNeish, *Ann. Math.* **23**, 221 (1922).
6. M. Hall and D. E. Knuth, *Am. Math. Mon.* **72** (part 2, *Computers and Computing*), 21 (1965).
7. C. Tompkins, *Proc. Symp. Appl. Math.* **6**, 195 (1956); L. J. Paige and C. Tompkins, *ibid.* **10**, 71 (1960).
8. E. T. Parker, *ibid.* **15**, 73 (1963).
9. For a complete survey see J. Dénes and A. D. Keedwell, *Latin Squares and Their Applications* (Academic Press, New York, 1974).
10. See, for example, M. N. Barber and B. W. Ninham, *Random and Restricted Walks* (Gordon & Breach, New York, 1970), chap. 7.
11. W. A. Mozart, *Musikalisches Würfelspiel* (Schott, Mainz, 1956), K 516 f Anh. C 30.01; this was first published in 1793.
12. The 11 possibilities for bar 8 are all identical, and Mozart gave only two distinct possibilities for bar 16, so the total number of waltzes is 2×11^{14} rather than 11^{16}.
13. T. H. O'Beirne, *Dice-Composition Music* (Barr & Stroud, Glasgow, 1967).
14. E. W. Dijkstra, *Numer. Math.* 1, 269 (1959).
15. See, for example, N. Christofides, *Graph Theory, An Algorithmic Approach* (Academic Press, London, 1975), sect. 7.4.
16. First construct the matrix of distances between all pairs of points, then try all possible intermediate junction points.
17. A determinant formula that specifies the number of spanning trees in a particular graph was discovered by C. W. Borchardt [*J. Reine Angew. Math.* **57**, 111 (1860)]. When the graph is a square grid, with m rows and m columns, the number of spanning trees seems to be always of the form mx^2 or $2mx^2$, where all the prime factors $> m$ of x are rather small numbers of the form $km \pm 1$; at least this is true when $m \leq 12$. For example, the large number cited in the text corresponds to the case $m = 11$, and in factored form it equals $2^{15} \cdot 11^2 \cdot 23^2 \cdot 89 \cdot 109 \cdot 199 \cdot 241 \cdot 373 \cdot 397 \cdot 419$. This curious circumstance, which I noticed while preparing the present article, still awaits a theoretical explanation.
18. J. B. Kruskal, Jr., *Proc. Am. Math. Soc.* 7, 48 (1956).
19. E. W. Montroll, in *Applied Combinatorial Mathematics*, E. F. Beckenbach, Ed. (Wiley, New York, 1964), sect. 4.4.
20. This is the well-known "Hungarian method" for the assignment problem; for example, see (*15*, sect. 12.4).
21. M. Held and R. M. Karp, *Math. Prog.* 1, 6 (1971).
22. See, for example, C. B. Boyer, *A History of Mathematics* (Wiley, New York, 1968), pp. 555 and 641.
23. R. M. Karp, in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. (Plenum, New York, 1972), p. 85. See also A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Mass., 1974), chap. 10. For a popular account of related work, see G. B. Kolata, *Science* **186**, 520 (1974).
24. L. J. Stockmeyer, *The Complexity of Decision Problems in Automata Theory and Logic* (report MAC TR-133, Massachusetts Institute of Technology, Cambridge, 1974), chap. 6.
25. J. R. Büchi, *Z. Math. Logik Grundl. Math.* **6**, 66 (1960).

26. The preparation and publication of this article was supported in part by NSF grant DCR 72-03752 A02, ONR contract NR 044-402, and the IBM Corporation. Some of the computations were done with the MACSYMA system, supported by the Defense Advanced Research Projects Agency under ONR contract N00014-75-C-0661; others were done on the SUMEX-AIM computer, supported by NIH RR00785; still others were done at the Université de Montréal, Centre des Recherches Mathématiques, where a preliminary version of this article was prepared under the auspices of the Chaire Aisenstadt. A lecture based on this material was presented at the AAAS annual meeting in Boston, 22 February 1976, in the session entitled "The Frontiers of the Natural Sciences" organized by R. M. Sinclair.