Mathematics

Discrete Mathematics: The New Mathematics of Science

The computer revolution has made discrete mathematics as indispensable as the calculus to science and technology

Anthony Ralston

Views

"Calculus has emerged as the principal language of science and technology. I have found that those people who speak calculus . . . can, with a little coaching, learn any science or technology relatively rapidly. Conversely, people who do not speak calculus seem to find it extraordinarily difficult to learn anything very substantive or quantitative about science and technology."

If these words had been spoken in, say, 1960, they would have been unexceptionable. Calculus was then, in a very real sense, *the* mathematics of science and technology. But the quotation is from a 1985 speech by the president of a major American university who himself comes from a technical background. I believe that anyone espousing this point of view in 1985 was seriously misinformed. Calculus is, indeed, as important in science and technology as it ever was. But it is no longer preeminent; it has an equal partner called discrete mathematics.

The reason calculus is no longer the only mathematics providing an entree to science and technology is due to the development of computers and the discipline of computer science. Despite the virtually universal awareness of the computer revolution, few people are as yet aware of the changes in the world of mathematics which are happening as a result. In this article I will outline the content of discrete mathematics and discuss some of its applications. And I will consider some of the effects of the growth in importance of discrete mathematics on mathematics research and mathematics education.

The distinction between discrete mathematics and the calculus

At the most basic level a model for the distinction between discrete mathematics and the continuous mathematics of the calculus and the branches of mathematical analysis that depend on calculus is the distinction between the integers and the real numbers. The latter are the basis of all mathematics which, like calculus, is concerned mainly with the properties of *continuous* functions. But digital computers (referred to hereafter simply as "computers") are *discrete* machines. And while it would be an oversimplification to say that computers deal only with integers, it is correct to say that the numbers with which computers do deal are, like the integers, a discrete set of points on the number line, whereas the real numbers are dense on this line. The mathematics necessary to deal with functions defined on a discrete rather than a continuous set of points is quite different in many respects from that remarkable mathematical edifice known as analysis, of which the calculus is the base, which deals mainly with continuous functions.

Some history should help illustrate the argument that there really is a need for a nontraditional kind of mathematics in dealing with computers. Although physical systems are composed of very many discrete particles—atoms and molecules—it is convenient and highly accurate to deal with matter as if it were continuous. Hence, the standard way of modeling the laws of physics is through calculus, typically in the form of differential equations. This approach has been such a wonderful success because the results derivable from it are, for almost all intents and purposes, essentially exact. Prior to the advent of the electromechanical desk calculators which preceded computers, the calculus model could generally only be manipulated analytically—that is, by finding exact or, perhaps, series solutions of the differential equations.

With the advent of digital calculators and then computers, it became possible to try to solve differential and other equations numerically. Some readers may have had the experience of solving an ordinary differential equation using a desk calculator by filling up large sheets of ruled paper with calculations. Such calculations always involved replacing the derivatives and integrals of the calculus with discrete approximations, because only the latter could be manipulated numerically. The mathematical justification for doing this was that if the discrete approximation to the continuous quantity was sufficiently good, the resulting computed solution would be close enough to the true solution of the continuous problem to be usable, say, in later scientific or engineering work. This was the beginning of the flowering of numerical analysis, that branch of mathematics which, in essence, deals with the solution of the problems of analysis by means of arithmetic.

With the coming of computers, numerical analysis

Anthony Ralston is Professor of Computer Science and Mathematics at the State University of New York at Buffalo. He received his S.B. and Ph.D. in mathematics from MIT and is a former president of the Association for Computing Machinery and of the American Federation of Information Processing Societies. His current interests involve the effects of discrete mathematics and computer science on the mathematics curricula in elementary and secondary schools as well as in college. Address: Department of Computer Science, State University of New York, 226 Bell Hall, Buffalo, NY 14260.

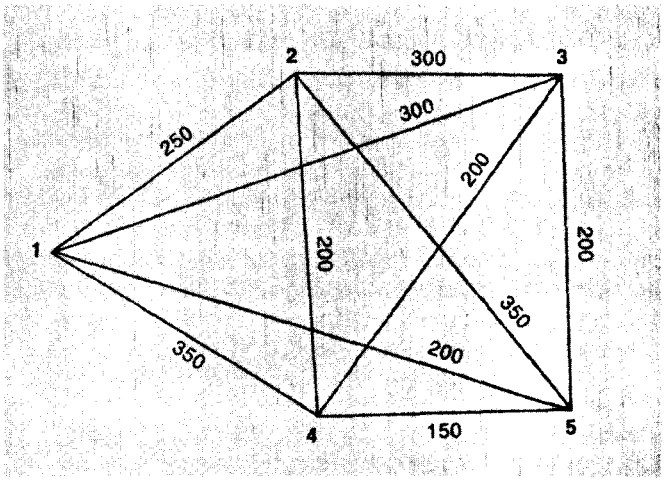


Figure 1. A salesperson who lives in city 1 wishes to visit four other cities once and only once by the shortest possible route before returning home. In this schematic version of the traveling-salesperson problem (not drawn to scale), the shortest route, 1-2-4-3-5-1 with a length of 1,050, does not even take advantage of the shortest distance between two cities, the 150 between cities 4 and 5. And the *greedy* routes that always proceed to the nearest next city, 1-5-4-3-2-1 or 1-5-4-2-3-1, have lengths of 1,100 and 1,150, respectively.

became the most rapidly growing branch of mathematics through the 1960s and early 1970s. It is still growing rapidly but now can fairly be called a mature branch of mathematics. The standard paradigm of problems solved by the techniques of numerical analysis is to take the calculus formulation of a physical problem (which, as noted above, is typically an approximation of a discrete physical situation) and then convert it back into discrete form in order to solve it numerically. This works so well because both the continuous approximation of the underlying discrete physics and the discrete approximation of the differential equation are accurate enough to enable the final results to be useful. It is important to realize that the scales of the calculus approximation and the subsequent discrete formulation of it are very different; the former involves the aggregation of a vast number of atoms into a single entity with an error of one part in 10^{20} or more, whereas the latter conversion into discrete form is much more crude, involving typically an accuracy of one part in 10^3 to 10^8.

If the need for discrete mathematics were confined to numerical analysis, then it could not be argued that such mathematics plays a role comparable to that of the calculus. Numerical analysis, although of very wide applicability, is a subject for specialists and should not influence significantly the mathematics taught to college undergraduates. Although numerical analysis does provide the main interface between continuous and discrete mathematics, it encompasses today only a very small and still decreasing proportion of the applications of discrete mathematics.

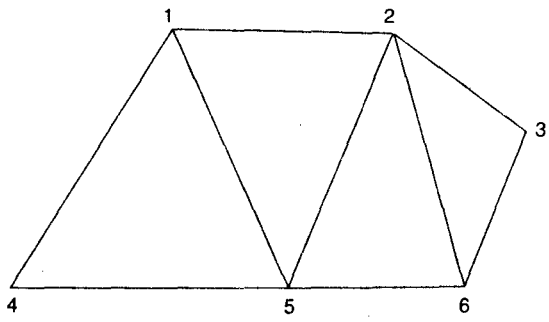
The real impetus to the growth of discrete mathematics has come from computer science itself and from the needs of other disciplines such as economics and biology. Economists and biologists, as they have tried to become more quantitative and mathematical, began by using the models provided by calculus because nothing else seemed to be available, even though the underlying situations being modeled were often discrete. As computers have become more available and as discrete mathematics has provided useful and accessible tools, such disciplines are increasingly using discrete rather than continuous models. We give some examples of these models in the following sections.

Computer science as a discipline is barely twenty years old. Nevertheless, as it continues to grow rapidly, it provides today and will provide for many years to come the single greatest source of problems for applied mathematics. Since almost all of these problems require discrete mathematics for their solution, the focus of applied mathematics will move increasingly in that direction. Whether or not discrete mathematics will become more important than analysis in applied mathematics is not the point; the crux of the matter is that both will be sufficiently important so that no one planning a career in mathematics or any discipline which depends heavily on mathematics will be able to afford to be ignorant of either classical applied mathematics or discrete mathematics, the new applied mathematics.

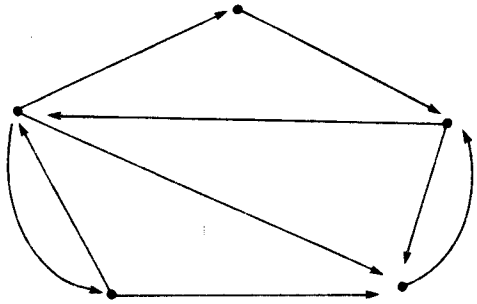
The kinds of mathematics which fall under the rubric of discrete mathematics and the types of problems to which this mathematics can be applied can best be understood through a series of examples, some of which will be purely mathematical and some concerned with practical problems. These examples will by no means exhaust the branches of mathematics comprised by discrete mathematics; they are intended simply to provide a flavor of discrete mathematics and its applications. But all the branches of discrete mathematics we discuss below have a wide variety of practical applications. Before proceeding, it should be noted that the distinction between discrete and continuous mathematics is not hard and fast. Some branches of mathematics—linear algebra is perhaps the best example—contain elements of both the discrete and the continuous. Also, it is worth noting here that there is much to be said pedagogically for teaching discrete and continuous mathematics together rather than, as is normally the case now, teaching them in distinct courses.

Graph theory

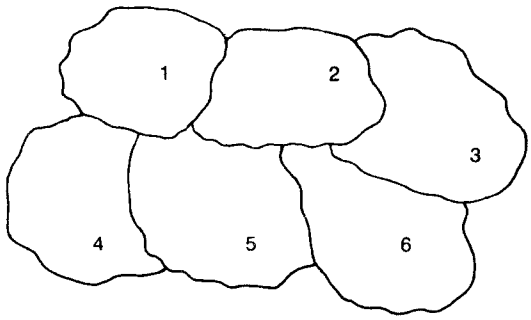
One of the most widely applicable branches of discrete mathematics, graph theory is an active area of research among mathematicians and computer scientists. To a professional mathematician, a graph is not the thing we drew in high school to display the behavior of a function like a quadratic. Graph theory deals with structures made up of vertices and edges:



The edges may be directionless or they may have directions, in which case we speak of a directed graph, or digraph:



One of the most famous problems in graph theory is the four-color problem, even though it is not about graphs but rather about maps:



Or imagine a map of the 48 contiguous United States. The problem is, What is the smallest number of colors needed to color a map so that no two areas with a common boundary (of more than a single point) have the same color? Although this is a problem of much more mathematical than practical importance, it does affect, for example, how many different colors of ink a printer of an atlas might need. The four-color theorem states that four colors are sufficient to color any map which can be drawn on a piece of paper.

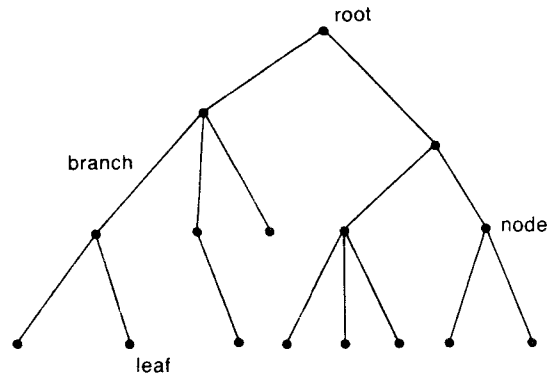
This problem was originally posed in the first half of the nineteenth century and was only solved—using graph theory—just over ten years ago by two mathematicians at the University of Illinois, Kenneth Appel and Wolfgang Haken. How does the four-color theorem become a theorem of graph theory? By replacing each area in our map example with a vertex and joining two vertices with an edge only when there is a boundary between the corresponding areas, the map is converted into a graph; indeed, our sample graph corresponds to the map, with the vertex numbers in the former corresponding to the numbers in the areas in the latter. Appel and Haken proved the theorem using a fast computer to analyze a large number of possible cases, a number shown by mathematical analysis to be sufficient to prove the theorem. Thus, a problem which for over a century had withstood the assaults of some of the greatest mathematicians of their day fell to a computer analysis based on advances in the mathematics of graph theory.

A much more practical problem of discrete mathematics is the traveling-salesperson problem illustrated in Figure 1: a salesperson must visit each of many cities (the vertices in the graph) once only and return home; the object is to find the shortest route. Easy, you may think—just enumerate all possible routes and calculate the distance of each. This is easy enough when there are only five cities as in the figure. But suppose there were 20 cities. How many possibilities are there then? The answer for any number n of cities is $(n-1)!$, since there are $n-1$ ways to choose the first city to visit, $n-2$ choices for the next city, and so on. But if $n = 20$, then $19!$ is about 10^{17}, a vast number. And if $n = 50$, the brute-force approach of calculating all possibilities would take longer than the life of the universe on any known or imaginable computer.

Various techniques may be used to speed up the calculation. For example, never continue to investigate a path whose length is already longer than one you have previously found. And never investigate a path which goes in the opposite direction to one already studied. Furthermore, there are techniques using random sampling of paths which result in good approximations to the best paths. Indeed, some very sophisticated computational algorithms have been devised for this problem, which is also a model for practical situations other than trips by salespeople. But none of these overcome the essential property alluded to above, the exponential growth in the size of the problem as n grows.

Might some algorithm exist capable of finding the shortest route in the traveling-salesperson problem in which the computation does not grow exponentially in the number of vertices? We don't know. The answer lies at the heart of the most important unsolved problem in theoretical computer science, the technical name of which is the $P = NP$? problem. The subdiscipline of computer science which studies questions of this kind is called *computational complexity*.

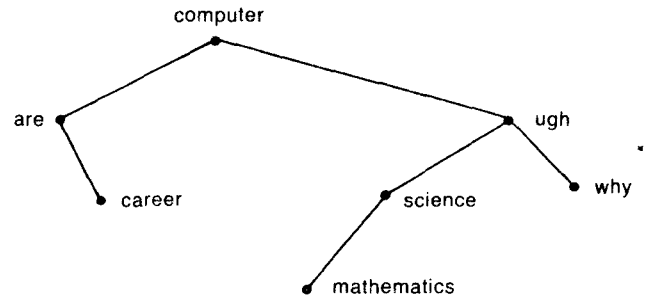
There is a special kind of graph called a *tree*, which is particularly important in computer science:



Although no directions are shown on the edges, a tree is a digraph because the direction of each edge is implicitly downward. (It is not pure perversity on the part of computer scientists that causes them to grow trees downward from the root; it is just easier, when working with pencil and paper or on a computer display screen, to work from the top down.) What properties must a digraph have to be a tree? As implied by our example, there can be no cycles (closed paths), and every vertex has only one edge entering it from the top except for the topmost vertex (the root), which has no edges entering it.

One application of trees is in game playing. The game tree in Figure 2 diagrams all possible moves in a simple game in which there are four sticks at the start, each player must take one or two sticks at each turn, and the player who takes the last stick wins. Should this be the first or second player if both play as well as possible? The figure provides the answer. Such trees play an important role in computer programs which play games like chess or checkers.

This game tree is a binary tree because each node has at most two branches emanating from it. Binary trees have applications to many problems in computer science, among which is the common problem of sorting a list. For example, consider the following words to be listed in alphabetical order: computer, ugh, science, why, mathematics, are, career; as each word is read in turn into the computer, its position on the tree is determined by starting at the root and going left or right, depending on whether the new word comes, respectively, earlier or later alphabetically than the word already at the vertex, until a vertex is reached with which no word is yet associated.



As drawn, the list of words can be read off in alphabetical order left to right, but there are more effective ways of doing this than to depend on how carefully the tree is drawn.

Difference equations

The discrete analogs of differential equations are difference equations. A simple example is the following equation, which models the situation in which you invest P at a fixed yearly interest rate r. Then with P_k being the value of the investment after k years

$$P_{k+1} = P_k(1 + r), \qquad k > 0, \qquad P_0 = P \qquad (1)$$

This is called a difference equation or a recurrence relation because it relates values of members of a sequence $\{P_i\}$ at different values of the subscripts. As with differential equations, the condition $P_0 = P$ is called the initial condition. The solution of Eq. 1 is given by a term of a geometric series as

$$P_k = (1 + r)^k P \qquad (2)$$

If, for example, in year 0 you put amount P into an individual retirement account (IRA) and withdrew it after N years (without penalty), then, using Eq. 2, the amount A you would have is $A = (1 - t)P_N = (1 - t)(1 + r)^N P$, where t is your tax rate at the time of withdrawal. Of course, the assumptions of a fixed interest rate and a single tax rate (with no tax brackets) simplify the problem, but both of these could be removed at the expense of making Eq. 1 more complicated and difficult to solve.

If instead of putting your money into an IRA you put it into a savings account or money-market account at the same interest rate r, the equation corresponding to Eq. 1 is $P_{k+1} = (1 + r)P_k - rtP_k$, with $k > 0$ and $P_0 = P(1 - t)$, because now you must pay taxes before you invest the initial principal, and you must pay taxes on the interest each year as you earn it. Again this difference equation is not hard to solve and yields a value after N years of

$$A = P_N = [1 + r(1 - t)]^N P(1 - t) \qquad (3)$$

for t (a 30% tax rate) and $N = 40$ years, the ratio of the value of A derived from Eq. 2 to that given by Eq. 3 is 3.04. That is, you would have three times as much money after 40 years with an IRA than with a normal savings account on which you paid taxes every year. Even with our simplifying assumptions, the advantage of the IRA is clear. Using the difference equations above, it can also be shown that, even if you withdrew your funds from the IRA before you were 59 $\frac{1}{2}$ years old, thereby incurring a 10% penalty, you would still come out ahead of the savings-account route if you kept your funds in the IRA for about 6 years (the precise number depending on the true interest and tax rates). I have used a calculation such as this to (try to) convince undergraduates that they cannot afford *not* to take out an IRA as soon as they get their first job (although the recent tax reform changes things somewhat).

Another area where difference equations are useful is in population studies, in which the most appropriate models are discrete because typically the interest is in measurements of populations at discrete—and, usually, constant—intervals. Consider, for example, the case of one species that preys on another in a particular area, say wolves and rabbits. The equation

$$R_n = aR_{n-1} - bW_{n-1} \qquad a,b > 0 \qquad (4)$$

says that the number of rabbits R_n in time period n (say, a year) is proportional to the number there were in period $n-1$ and to the number of wolves in period $n-1$. The coefficient a would normally be greater than 1, reflecting a birth rate greater than the (natural) death rate of rabbits, while the value of b would reflect how many rabbits each wolf would be expected to kill in the time period. A corresponding equation for the number of wolves in period n might be

$$W_n = cW_{n-1} \qquad (5)$$

if neither the availability of food in the form of rabbits nor the food eaten by the rabbits is essential for wolves in the particular ecological system. The coefficient c, analogous to a in Eq. 4, reflects the net birth-versus-death rate of wolves. Eqs. 4 and 5 are a pair of simultaneous difference equations whose solutions may be found by first solving Eq. 5, which has a form like Eq. 1, and substituting this into Eq. 4, which can then be solved to give $R_n = Ra^n - bW(c^n - a^n)/(c - a)$ and $W_n = Wc^n$, where R and W are the initial populations of rabbits and wolves (when $n = 0$). Although this is a very simple model, it may nevertheless be used to get some insight into what ratio of rabbits to wolves (R/W) will

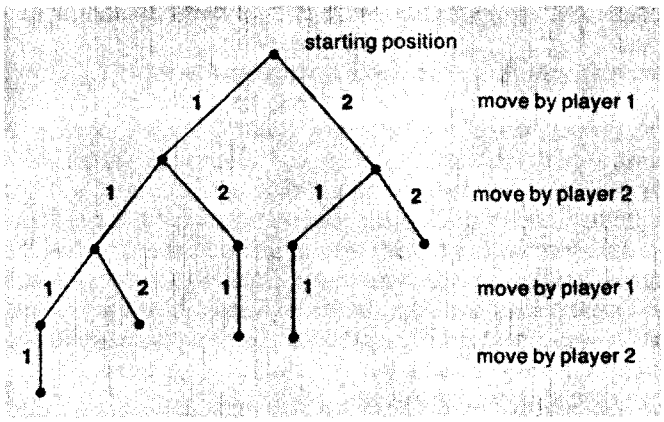


Figure 2. This game tree outlines all the possible moves in a game in which two players begin with four sticks and alternately pick up either one or two sticks. The object of the game is to pick up the last stick; thus, a winning strategy for the first player requires a pick up of only one stick on the first move, since picking up two would allow the other player to take the remaining two on the next move and win the game.

result in a stable population of rabbits. Suppose, for example, that $c = 1$ so that the wolf population is a constant (W). Then $R_n = Ra^n - bW(1 - a^n)/(1 - a)$ and, more usefully, $R_n - R_{n-1} = a^{n-1}[(a - 1)R - bW]$. Therefore, depending on whether $(a - 1)R - bW$ is less than, equal to, or greater than 0, the population of rabbits will, respectively, decrease, remain stable, or increase.

A more realistic predator-prey model would lead to a more complicated system of difference equations which would not be solvable analytically, except in special cases. Note, however, that with difference equations we can always compute the solution for successive values of n from 0 on directly from the equations themselves. For classical physical models expressed in terms of differential equations, if the equations cannot be solved analytically, then the solution cannot be computed directly. To compute it at all, it is generally necessary first to approximate the differential equations by difference equations and then compute. Thus, when a model can be *naturally* stated in terms of difference equations, as is common in computer science as well as in economics, biology, and other disciplines, this is usually to be preferred over a model stated in terms of differential equations.

Combinatorics

A venerable branch of mathematics, combinatorics is introduced to most students in high school when they study permutations and combinations. Recent years have seen an explosive growth in this area both because computers allow combinatorial calculations which were impossible previously and because so many of the mathematical problems posed by computer-science research require combinatorial methods for their solution. The two examples we give here are characteristic of combinatorial problems. Actually we used combinatorial reasoning in the traveling-salesperson problem to *count* the number of possible routes that could be taken among n cities.

As our first example illustrates, combinatorial problems often require the use of difference equations for their solution. Let X be an n-set, that is, just some unordered collection of n objects which we shall assume are all distinct. A partition of X is a collection of subsets which contain all the elements of X and is such that each subset has at least one member. For instance, if $X = \{1, 2, A, B\}$, a partition of X is $\{1, A\}, \{2\}, \{B\}$. How many distinct partitions are there (where the order in which the subsets are listed is irrelevant) of an n-set into k subsets? Let this number be denoted by $S(n, k)$. We reason as follows: $S(n, 1) = 1$, because the only partition into one subset is the n-set itself; $S(n, n) = 1$, because the only partition into n subsets puts one element of the n-set into each subset; $S(n, k) = 0$ if $k > n$, because in this case no partition is possible. Now suppose we knew $S(n-1, k)$ for all positive values of k. Then denoting the elements of the n-set by $x_1, x_2, \ldots, x_n$, we have the following partitions of the n-set into k subsets: those in which x_n is in a subset by itself, this number being $S(n-1, k-1)$, because the other $n-1$ members must go into $k-1$ subsets; and those in which x_n is not in a subset by itself, this number being $kS(n-1, k)$, because the other $n-1$ elements must be partitioned into k subsets and then x_n can be put into any one of these k subsets. Thus

$$S(n, k) = S(n-1, k-1) + kS(n-1, k) \qquad (6)$$

This equation is called a partial difference equation because, by analogy with partial differential equations, it involves more than one (specifically, two—n and k) discrete variables. The equation cannot be solved analytically, but, using the boundary counditions above, it is quite easy to compute the values of $S(n, k)$ (which are called *Stirling numbers of the second kind*, after the eighteenth-century Scottish mathematician, John Stirling, who first identified them).

The greatest need for combinatorial mathematics in computer science is in the analysis of algorithms, one of the most important and most mathematical subdisciplines of computer science. (An algorithm is just a procedure, a rule for performing a well-defined task; the study of algorithms lies at the heart of computer science.) As an example, consider again that very common problem of the need to sort a list of n items into numerical or alphabetical order. This is perhaps the most common task performed on computers, since it is basic to many data-processing applications. Therefore, efficiency in performing this task is very important. A well-known algorithm for sorting is called bubble sort. The procedure, as illustrated in Figure 3, is as follows: Compare the first two items in the list and, if they are not in order, interchange them. Then compare the second (which may have been first originally) and third items and do the same. Continue in this way until the bottom of the list is reached. At this point the largest element will be in the n^{th} position. Then go back to the beginning and repeat the process, ignoring at each pass the elements known to be in their correct positions. After at most $n-1$ passes, the list will be sorted. The name of this algorithm comes from the fact that the smaller (or alphabetically earlier) elements "bubble" up to the top of the list.

Figure 4 displays an algorithm for bubble sort. How efficient is it? To answer this question we focus on the

Initial list	After first pass	After second pass	After third pass	After fourth pass	After fifth pass	After sixth pass
15	12	9	9	6	6	2
12	9	12	6	9	2	6
9	15	6	12	2	8	8
17	6	12	2	8	9	9
6	12	2	8	12	12	12
12	2	8	12	12	12	12
2	8	15	15	15	15	15
8	16	16	16	16	16	16
16	17	17	17	17	17	17

Figure 3. A simple algorithm for putting a scrambled list of n numbers into numerical order is to compare the first two items on the list and interchange them if they are out of order, then do the same for the second and third items, then the third and fourth, and so on until the bottom of the list. In this example, 15 is exchanged with 12 and then with 9; then 17 is exchanged with each of the items initially below it. After this pass through all the elements, the largest number must be at the bottom. Each repetition of the procedure would then guarantee that the next item up would be in order. Thus, $n-1$ passes are needed to guarantee that the whole list is in order, although fewer may be required; in this case, the list is ordered after six passes, but the computer, following the algorithm in Figure 4, would go through two more passes in which nothing would change.

key action in this algorithm, labeled "comparison," since this is the action performed most often. (Note that the "interchange" in the next line is performed only if the comparison indicates that the elements are not in their proper order.) For each value of i from 1 to $n-1$, the comparison is performed i times. Therefore, the total number of comparisons is $1 + 2 + 3 + \ldots + n-2 + n-1$. This is the sum of the first $n-1$ integers, which can be expressed in closed form as $n(n-1)/2$. Thus, the number of comparisons is always proportional to n^2 or, to use a more common notation, $0(n^2)$, where the "0" should read "order of."

Can the efficiency of this algorithm be improved? Yes, by marking the last place where an interchange was performed on each pass, because below this all elements are in their proper order and need not be examined on the next pass. How does this change the analysis above? Not at all in the worst case since, if the initial order is n, $n-1$, $n-2$, . . . , 3, 2, 1, then $n(n-1)/2$ comparisons are

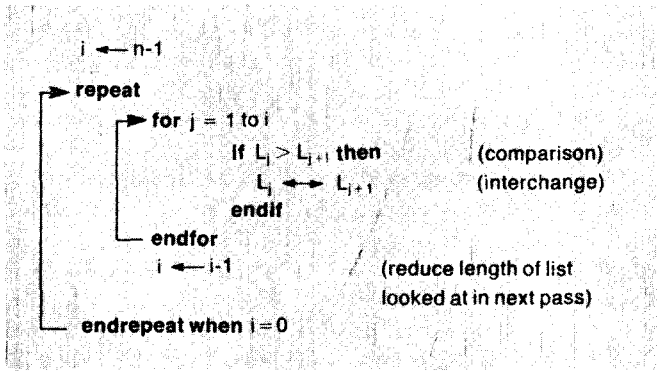


Figure 4. This algorithm for bubble sort will put a list $L[1{:}n]$ of n items into numerical order. The algorithm is inefficient, as can be seen in Figure 3, because the repeat-endrepeat loop is apt to be performed several times after the list is ordered, and because the "comparison" is performed needlessly many times on the ordered part of the list as it grows from the bottom up.

still needed. But, on the average, fewer comparisons are needed. However, the average number is still proportional to n^2, although the constant of proportionality is smaller than $\frac{1}{2}$.

Are there methods of sorting for which the number of comparisons is a function of n that grows more slowly than n^2 in the worst or average cases? This is an important question because efficiency becomes significant for large values of n, when the difference between n^2 and some more slowly growing function of n could be considerable. The answer to the question is, yes, there are methods of sorting (of which the two best known are called quicksort and heapsort) for which the number of comparisons grows proportionally to $n \log n$ in the average case and (for heapsort but not quicksort) in the worst case, too. With logarithms taken to the base 2, the constants of proportionality do not differ much from that for bubble sort. Since for $n = 10{,}000$, for example, n^2 is 100 million while $n \log n$ is only about 135,000, bubble sort is indeed a very poor method of sorting and should never be used for any but quite small n.

Mathematical logic

Discrete mathematics includes mathematical logic because the propositions, or predicates, which are the key entities of logic typically have the "value" *true* or *false*—or, when translated into the binary-number system, 1 or 0, respectively. (A predicate is a function, such as "the positive integer i is prime" or "$x > 2$," which is true or false for each substitution of a value for the variable, i in the first example, x in the second.) Although logic has historically been important mainly as a tool for studying the theoretical foundations of mathematics, it was applied to the design of computer hardware from almost the earliest days of such hardware. The form in which logic was applied to hardware design is called Boolean algebra, after the nineteenth-century British mathematician George Boole, who first made significant progress in developing an algebra of logic.

Consider, for example, the problem illustrated in Figure 5, in which we wish to design a *logical circuit*—which can then be implemented by electronic circuitry—to add two "bits" (i.e., digits) x_i and y_i of two binary numbers X and Y, ignoring the carry bit from the previous $(i-1)$ stage. The truth table displays all possible values of the inputs x_i and y_i and gives the corresponding values of the sum and carry outputs s_i and c_i. From this table we can deduce the Boolean equations, which are shown both informally and in their usual mathematical notation. Figure 5 displays the appropriate logic circuit for this half adder using the standard symbols for "and," "or," and "not." To design a full adder which would also take into account the carry from the previous $(i-1)$ stage, we need two of these half adders and an additional "or" circuit. Boolean algebraic techniques are still used in the design of many complex integrated circuits.

More recently, mathematical logic has become important in two other branches of computer science. In one, the *verification* of algorithms and computer programs, the language of logic is used to describe the input to and the output from an algorithm or program and, as well, to make *assertions* about the algorithm or program

itself. These are then used to try to construct a proof that the algorithm or program does what it purports to do. For example, using the algorithm in Figure 4, an assertion which could be placed before the "for" loop is

$$\forall k\colon 1 \leq k \leq j\colon x_k \leq x_{j+1} \qquad (7)$$

which should be read "For all $(\forall)k$ from 1 to j, x_k is less than or equal to x_{j+1}." Thus, the assertion claims that just before the $j+1$ execution of the loop, the $j+1$ item is at least as great as all the preceding items, which is, after all, what is supposed to happen in bubble sort. Before the loop is entered the first time, j is implicitly 0, so that there are no values of k from 1 to j and so the assertion is empty. An assertion like Eq. 7 is called a loop invariant because it states a condition that is supposed to be unchanged after each execution of the loop. Assertions like these are intended to be used by a mechanical theorem prover (that is, by a computer program designed to prove theorems) to prove that an algorithm or a program is correct. This is a fairly new area of computer-science research which holds great promise. It has already contributed to good programming methodology by showing how the design of programs affects how hard or easy it will be to prove they are correct.

The other still more recent use of mathematical logic is directly as a programming language for so-called logic programming. Interest and research in this area was given a major boost in 1981 when the Japanese announced that a logic-programming language, PROLOG, would be the basic software tool in their development of a new generation of (so-called fifth-generation) computers. Among the many kinds of symbolic computations to which PROLOG may be applied is its application to large databases, in which, besides just retrieving and updating data, it is often desired to answer questions about the relationships among the objects in the database. For example, consider the seven items in the following database (which departs a bit from the syntax of PROLOG): male (ronsenior); male (ronjunior); female (nancy); female (maureen); parents (ronjunior, ronsenior, nancy); parents (maureen, ronsenior, nancy); may-enter-politics(X) :- female(X) and father (ronsenior). The first six items define relationships and the objects that satisfy those relationships, whereas the last item is a rule which may be used in querying the database. Suppose, for example, you enter the query "may-enter-politics(X)?" Then PROLOG would respond "maureen."

Although this simple example does not illustrate it, PROLOG uses sophisticated tools of mathematical logic to answer queries and to perform the other applications of which it is capable. It is too soon to say how important logic programming will become, but it is clear that the use of mathematical logic in a programming context provides facilities which are useful in many applications.

Mathematics research and education

Research in mathematics is mainly an internally driven system. Mathematicians do research in areas that interest them. The funding agencies, like the National Science Foundation, do affect to some extent what areas of research are emphasized, but they are more likely to be followers of trends than leaders of them. Nevertheless, despite what has been called "the decay in the idea that the purpose of mathematics is to serve the sciences" (Mandelkern 1985), applied mathematicians are considerably influenced in their research by the problems which need solving in other areas of science. Although there are still many problems in the physical sciences which need applied-mathematics research for their solution, the major growth area in applied mathematics is in problems which come from computer science. Since these problems are overwhelmingly in discrete mathematics, we may expect to see a gradual increase in research in this area of applied mathematics relative to classical applied mathematics. Indeed, that this is already happening is clear from the increase in the volume of literature published on the applications of discrete mathematics.

It is, however, in mathematics education that the first effects of the growing importance of discrete mathematics will be seen by most nonmathematicians. The first two years of college mathematics are moving increasingly toward having discrete mathematics play a role equal to the calculus. This trend has profound

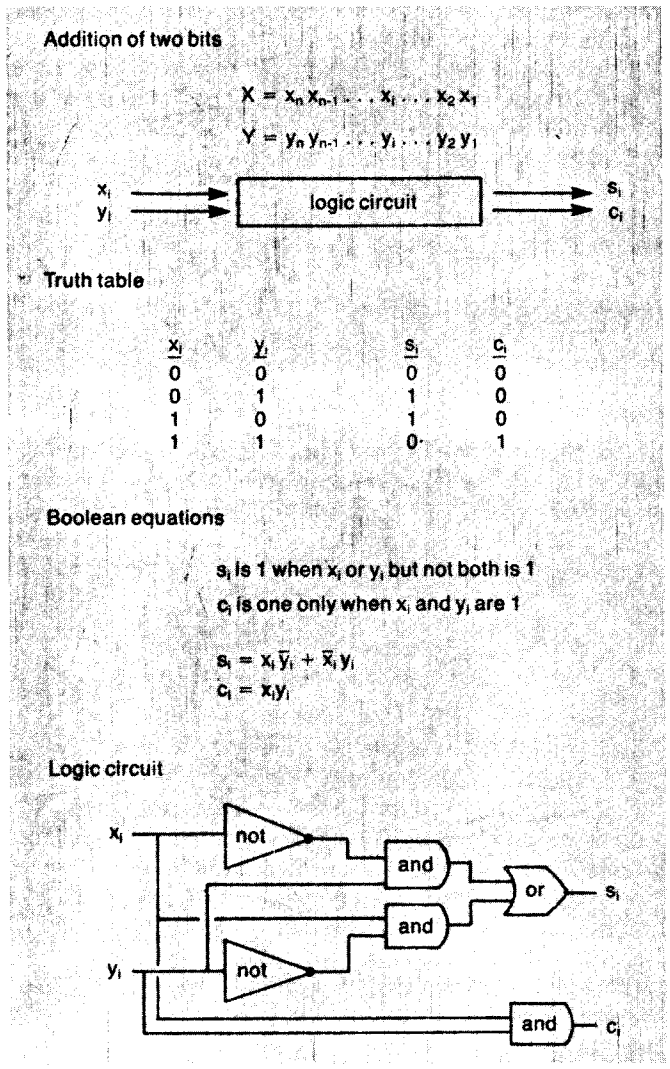


Figure 5. A logic circuit to add two bits x_i and y_i of binary numbers X and Y is designed by first converting these operations into the equations of Boolean algebra. The Boolean equations here summarize all the possible values of adding x_i and y_i, which are outlined in the truth table; when both x_i and y_i are one, for example, the sum is zero and the carry forward is one.

implications for the mathematics curriculum. If it continues, it will certainly affect the kinds of mathematics which will be emphasized in graduate work and, therefore, it will affect the emphasis in mathematics research. It will also have important implications for how calculus is taught and for those disciplines in the physical sciences and engineering which traditionally have required calculus to support their courses. There is a growing literature on the desirability of including more discrete mathematics in the first two years of college mathematics (Ralston 1981; Ralston and Young 1983). Quite significant is the increasing number of textbooks on discrete mathematics intended for freshman or sophomore courses—at least a dozen were published between 1983 and 1985, and a still larger number may be expected to appear in the next three years (e.g., Johnsonbaugh 1984; Ross and Wright 1985).

A trend toward discrete mathematics at the college level will inevitably have important effects on mathematics at the secondary and even the primary levels. Whether or not this presages another "new mathematics" is not yet clear (Ralston 1985). Certainly the trauma of the new mathematics of the 1960s is still very much with us. But if discrete mathematics becomes the new mathematics of the 1990s, it will be because changes in science and technology require that the mathematics taught to our children be relevant to the professions they will enter.

References

Johnsonbaugh, R. 1984. *Discrete Mathematics*. Macmillan.

Mandelkern, M. 1985. Constructive mathematics. *Math. Mag.* 58:272–79.

Ralston, A. 1981. Computer science, mathematics, and the undergraduate curricula in both. *Am. Math. Monthly* 88:472–85.

———. 1985. Discrete mathematics—the really new mathematics. *Yearbook of the National Council of Teachers of Mathematics.*

Ralston, A., and G. S. Young. 1983. *The Future of College Mathematics.* Springer-Verlag.

Ross, K. A., and C. R. B. Wright. 1985. *Discrete Mathematics.* Prentice-Hall.

"We did the whole room over in fractals."