Curricular Implications of Microcomputers for School Mathematics

Georg Schrage, Dortmund University
and Jerry P. Becker, Southern Illinois University

EDITOR'S NOTE: This article represents an abridged version of a recent unpublished manuscript presented to the National Council Supervisors of Mathematics, Washington, D.C, April 1986.

We are all aware that the microcomputer is a potentially valuable tool in mathematics teaching. Indeed, there are many who feel the microcomputer is a very powerful educational tool. But to explore the potential of this new technology means that we must also examine its limitations. In this paper, emphasis will be placed on what cannot or should not be done with computers in mathematics education. There are three types of limitations concerning microcomputers in Grades K through 12:

1. limitations due to educational responsibility
2. limitations due to practical technical restrictions
3. limitations due to logical and conceptual restrictions

Limitations Due to Educational Responsibility

The first point is certainly the most problematic, maybe even controversial. There is no doubt that microcomputers are affecting *what* students are learning and *how* they are learning. But the more sophisticated a tool, the more we must care about its use; and it is exactly the power and versatility of microcomputers which threaten a danger of their misuse.

Propaganda about the educational use of microcomputers is pervasive in our society today. We are referring to the vast promotion of educational software for curricular subjects. Currently, the number of educational programs available is estimated at 80,000, and and that number is doubling each year! Most of this software is of tutorial or animated drill and practice type, which is usually very good from a technical point of view, but is of questionable educational value. We must use great care in selecting software in mathematics teaching or else we may be risking misuse of the microcomputer.

We strongly support a reasonable use of microcomputers in the classroom. As mathematics teachers, we would like to have a microcomputer, connected to a screen, available to use in making demonstrations which might become objects of discussion for the whole class. Further, we believe the microcomputer can be very useful for:

1. stimulating mathematical thinking and supporting mathematical problem solving,
2. visualizing mathematical concepts, and
3. simulating processes that can be handled by mathematical models.

The computer enables us to handle mathematical objects, operate with numbers, present and transform geometrical figures, and visualize relationships among data. That microcomputers can be very useful in supporting and enriching mathematical problem solving is very clear. Problem solving is central to any mathematical activity. The computer enhances our problem solving capacities, and that is what students should experience in today's mathematical education, regardless of whether this occurs via BASIC, LOGO, PASCAL, or some other programming language. (Of course, we do not mean to imply that we regard all programming languages as equally suitable.)

Solving a problem using a computer is typically a sophisticated process that includes very different kinds of activities; for example, developing mathematical models, generating data for a problem, analyzing relationships, designing alogrithms, and constructing programs. With these kinds of activities, students can get to the mathematical heart of the matter. If the problems are well chosen, students will experience the intellectual challenge of mathematics, as well as the satisfaction provided by the solution of a difficult problem.

In mathematics, the solution to a problem is not nearly as interesting as the *method* used to get it, and many problems can be solved by quite different methods. Teachers should encourage their students to look for different approaches to a problem and to compare and evaluate them. We illustrate this by two examples below.

EXAMPLE 1:

The following is a nice problem for students at the elementary or junior high school level:

> On a farm there are 178 animals - cows and geese. Altogether they have 562 legs. How many cows and how many geese are on the farm?

There are many ways to solve this problem.

(a) We can actually make a list and check all combinations:

cows	geese	legs
0	178	356
1	177	358
2	176	360
.	.	.
.	.	.
.	.	.
50	128	456
.	.	.
.	.	.
.	.	.
90	88	536
.	.	.
.	.	.
.	.	.
103	75	562

(b) We can use a computer to generate the list. The following LOGO program does the job:

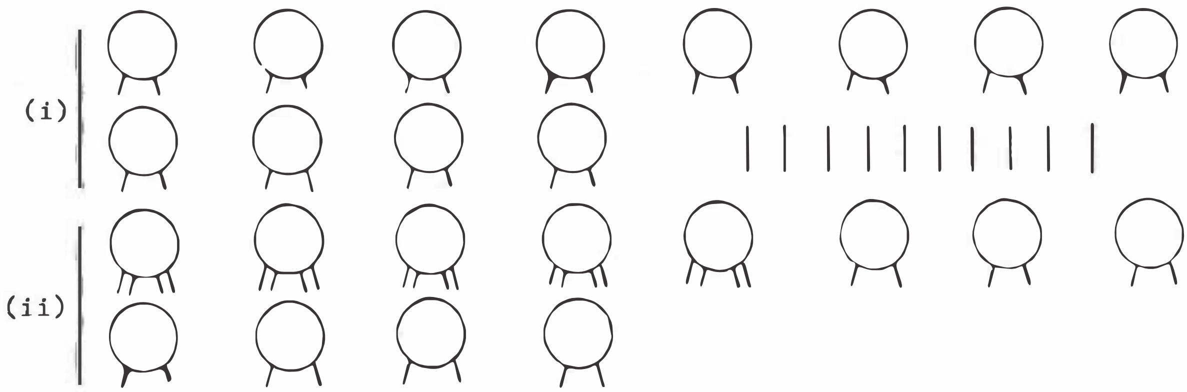
```
TO RANCH :ANIMALS :LEGS :COWS
MAKE "GEESE :ANIMALS-:COWS
(PRINT :COWS :GEESE 2*: GEESE +
  4*:COWS)
IF 2* :GEESE + 4*:COWS = :LEGS
  [STOP]
RANCH :ANIMALS :LEGS :COWS + 1
END
```

If we type in "RANCH 178 562 0," then the computer prints the above list and, therefore, solves the problem. This is a simple program, and the algorithm is brief and easy to understand and write.

(c) Some students attack the problem by using linear equations:

$$x + y = 178$$
$$4x + 2y = 562$$

(d) We know an eight-year old girl who solved the problem with smaller parameters - 12 animals and 34 legs - in the following manner:



She symbolized the animals by circles, gave two legs to each, and assigned the remaining ten legs in pairs. Once she had solved this "smaller" problem and *understood the concept*, she was then able to solve the problem for arbitrary parameters.

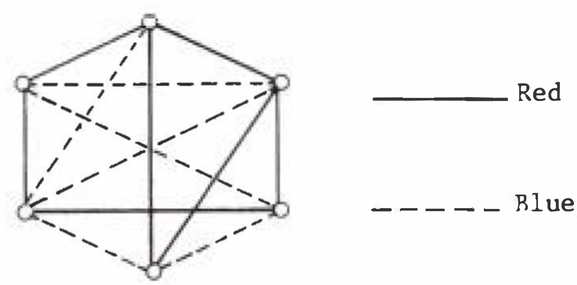
Which solution is the "best" one? We think the last, because it is the *simplest* and most straighforward one. It avoids the tedious work of the first, as well as the advanced algebraic tool of the third one. Further, it is available to an elementary school student, and we think it is the best solution from the point of view of a mathematician, too. The second (computer) solution represents a possible approach, too, but it should not stand alone.

We cannot define mathematical beauty, except to say that is has to do with simplicity and the use of straightforward arguments, simple but powerful ideas, and avoiding the use of sophisticated tools. Teachers should always strive to help students get a feeling for the beauty of mathematical ideas and methods. This is all the more important when we have a computer which we can program to supply solutions very quickly.

EXAMPLE 2:

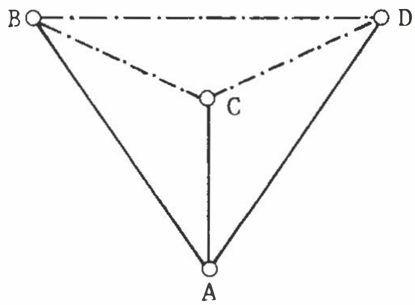
The following is a strategy game for students:

We start with six vertices of a hexagon. Two players alternately take turns, each time connecting two so far unconnected vertices. The first player uses a red pencil, and the second player uses a blue one. A player loses if he or she generates a triangle with all sides in his or her color. The result is a draw if all possible 15 lines have been drawn without producing a one-colored triangle.



After some experimentation, it is observed that someone always loses. The following conjecture arises: if each of the 15 connecting lines of a hexagon is colored either red or blue, then there will be at least one red or blue triangle.

If we have a computer available, the conjecture can be proven by systematically checking all possible blue and red colorings. If no coloring without a one-colored triangle is found in this process, then we are done. But, compare this with the following method. Consider an arbitrary vertex of the hexagon, say A. There are 5 emanating lines. At least three must have the same color. Without loss of generality, let us assume that there are three red lines. The end points of these three lines may be labeled B, C, and D.



If one of the lines BC, BD, or CD is red, then there is a red-colored triangle; if these three lines are all blue, then we have a blue triangle (that is, BCD). We think this is a beautiful proof, demonstrating the superiority of mathematical reasoning over brute computer force.

What should we learn from these examples? When teaching problem solving, we should always encourage our students to look for different ways to get solutions. They should also be aware of the tools available and select the most suitable one(s). It isn't necessary to use a bomb to kill a fly!

Limitations Due to Technical Restrictions

Virtually any finite mathematical problem can be solved by a computer, simply by checking all possible states of the problem. But many mathematical problems, especially combinatorial ones, are of such exploding complexity that even the most powerful computer may never be able to handle them. Let us demonstrate this, again by a simple example:

EXAMPLE 3:

Consider the numbers 1 and 2. There are two different arrangements to write these numbers in sequence, namely 12 and 21, and each such arrangement is called a permutation. There are six permutations of the three numbers 1,2,3: 123, 132, 213, 231, 312, 321.

It is not difficult to write a program to find out all possible permutations of n elements. But suppose we want to get a list of all possible arrangements of the numbers 1 to 15. This would seem to be a simple problem for a computer. The number of permutations of n elements is n!=1*2*3*... *(n-1)*n. The table below provides the values 1! to 15!

n	# Permutations
1	1
2	2
3	6
4	24
5	120
6	720
7	5,040
8	40,320
9	362,880
10	3,628,800
11	39,916,800
12	479,001,600
13	6,227,020,800
14	87,178,291,200
15	1,307,666,368,000

Now, imagine a very powerful computer, capable of determining and listing 1,000 different arrangements each second. This computer would have to work 40,000 years to finish its job!

There are many problems having important applications, which are practically unsolvable because of their algorithmic complexity. It is a subject of greatest scientific and economical importance to determine the complexity of algorithms and, if possible, to find algorithms for a given class of problems by which solutions can be obtained in a reasonable amount of time. We should strive to help students learn by simple examples from, say, combinatorics and number theory, that there are many problems which cannot be solved by computers for practical reasons, even if it is easy to develop a program that seems to solve the problem.

Limitations Due to Logical and Conceptual Restrictions

There are other problems which have been proven to be unsolvable by any computer. The best known of these problems is the so-called "halting problem." The unsolvability of the halting problem means that it is impossible to construct an algorithm which can decide for any arbitrary program and its data if it will ever stop or if it will be caught in a never ending loop. We think that the treatment of the halting problem and related problems is, perhaps, beyond the usual mathematics curriculum, though it is not really difficult. But there are other problems by which students can become aware of what a computer actually can and cannot provide in order to find a solution.

EXAMPLE 4:

Recently, we asked some students in a problem solving course to prove that $\sqrt{2}$ is irrational; that it cannot be represented as p/q with integers p and q. One student wrote the following: "With the help of a computer, we can determine that $\sqrt{2}$ is equal to 1.414213 . . . , never ending and never repeating. Therefore, it cannot be a rational number." The student, of course, had a fundamental misunderstanding of the conceptual potency of computers.

EXAMPLE 5:

The Collatz Problem (also known as the Ulam Problem, the Syracuse Problem, or the Hasse-Kakutani Problem). Consider the following algorithm in Pascal:

```
INPUT N
WHILE N > 1 DO
IF ODD (N) THEN N:=3*N+1 ELSE N:=
  N DIV 2
END.
```

(NOTE: DIV denotes whole number division.)

The input number N is the *seed* of a sequence either ending with 1 or never ending. Here are some examples:

(a) 10 5 16 8 4 2 1

(b) 42 21 64 32 16 8 4 2 1

(c) 120 60 30 15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1

It is still an unproven conjecture that for any positive integer N the algorithm will come to 1 eventually. Recent issues of the *American Mathematical Monthly* and *The Mathematical Intelligencer* contain papers devoted to this problem.

A good deal of experimental work has been done concerning this problem. Using powerful computers, it has been proven that the algorithm stops for any positive integer $N < 2^{40} \approx 1.2*10^{12}$. This gives certain evidence about the conjecture, but it doesn't prove anything concerning the general problem. Why, then, all this effort? There are two possibilities:

(a) The conjecture is true. This can never be proven by computer experimentation, because we can only check a finite number of integers, and therefore, an infinite number of possible seed numbers will forever remain unchecked.

(b) The conjecture is false; that is, there is a positive integer input N such that the algorithm never comes to 1. This can be due to either of the following reasons:

- There is a seed number N such that the sequence generated by N diverges to infinity. The existence of such a number can never be proven by running the algorithm, because you have to stop this calculation after awhile not knowing if, at sometime in the future, the algorithm would come to the end.

- There is a seed number N such that the sequence N, N_1, N_2, . . . generated by N is caught in a loop; that is, after a while, part of the sequence will be periodically repeated.

Such a loop can be detected by a computer, thus proving that the conjecture is wrong. (If, in the above algorithm, N: 3*N+1 is replaced by N:=3*N-1, we can find seed numbers producing infinite sequences. For example, 80 40 20 10 5 14 7 20 10 5 14 7 . . .) These are the only logical possibilities.

Conclusion

There is presently a great deal of discussion about computer literacy. A major factor in computer literacy, we believe, is the competence to make reasonable use of the power of computers, which means to be aware of the computer's limitations. We have all heard the term "computer revolution" in education. One characterisitic of a revolution is that it completely changes traditional values, structures, and ideas. The computer is a powerful tool that can affect *what* students will learn and *how* they will learn. But we should not forget the great mathematical ideas as developed by Euclid, Archimedes, Euler, Gauss, and others over hundreds of years. They are still the great ideas of tomorrow and tomorrow's tomorrow. Further, the importance of these ideas continues to grow. We must not allow the availability of computers to make mathematics superfluous; on the contrary, it requires improved mathematical education. The computer itself can help us to improve and enrich the curriculum. If we make sensible use of the computer, its impact should result in a permanent educational *evolution*, instead of revolution.

Georg Schrage is a professor of mathematics at Dortmund University, Federal Republic of Germany. Dr. Becker has been employed by Southern Illinois University since 1979, where he is a professor of mathematics, Curriculum and Instruction and Media Department. The authors have an interest in the use of microcomputers in education, particularly in their application to problem solving.

REFERENCES

Lagarias, J. "The 3X+1 Problem and Its Generalizations." American Mathematical Monthly 92, no. 1 (1985): 323.

Wagan S. "The Collatz Problem." The Mathematical Intelligencer 7, no. 1 (1985): 72-76.