Algorithms and Technology

Lynn Gordon Calvert

Part 1

Scenario 1:	I wonder what would work better for this problem a spreadsheet or a computer program?
Scenario 2:	The program and instructions are for Logo and the only graphing software I have is Geometer's Sketchpad. What would it take to convert this to a Geometer's Sketchpad script?

Both scenarios point to the ever-increasing variety of technology available to us in our homes and in our classrooms. Modeling a mathematical situation when several technological devices are available requires decision making about ease of use, efficiency of algorithm and the output desired or preferred. To make an informed decision, creating or re-creating an algorithm using different tools requires a global understanding of (1) the problem, (2) the algorithm used to simulate the situation and (3) common features and processes used across technology. As mathematics educators, we must promote the use of specific technological devices and provide opportunities for students to develop the ability to adapt to the technology that is or becomes available to them. We must also decide which tool is best suited for modeling the mathematical situation at hand.

The following radioactive decay simulation illustrates the benefits, limitations and decision making involved when choosing different technological tools to simulate the same event.

Radioactive Decay

A rich site for mathematics is the simulation of radioactive decay. Exploration into this area may initially be simulated with dice (Lovitt and Lowe 1993). Starting with 50 dice, all active "particles" are rolled. "Six's" are particles that have "decayed" in that year and are removed from the active ones. This process is repeated until all the particles have decayed which usually takes somewhere between 14 to 35 years. Exponential decay, the graphical analysis of half-life, rate of decay, and experimental vs. theoretical probability all come into play. Exploring the phenomenon further—that is, to approximate the average number of years for 50 particles to decay, determine the distribution of this event, compare the half-life and total decay for 50, 100 or 20,000 particles or change the rate of decay—is too time consuming or simply impossible to do with dice . . . but not with technology.

Creating an algorithm with technology to simulate this event requires an awareness of the process or iterative algorithm involved and the application of this process to the technology available. Spreadsheets and programmable calculators are used here to illustrate two possible sources to model the radioactive decay problem.

Spreadsheet

Display

Figure 1: Spreadsheet Display

	A	B	C	...	AG	AR	AS
Atom 1	1	1	0		0	3	6
Atom 2	1	1	1		0	3	2
Atom 3	1	0	0		0	6	1
...							
Remaining	50	43	32	...	0		
Year	1	2	3	...	25		

This spreadsheet is set up so that columns A, B, C and so on show individual particles that are active ('1') and particles that have decayed ('0'). Column A represents the beginning of year one. All particles are active. Column B represents the beginning of the second year. Connected to column B is column AR which is a series of randomly generated numbers from 1 to 6. If a '6' was generated, the particle was said to decay in that year.

Figure 2: Spreadsheet Chart Display

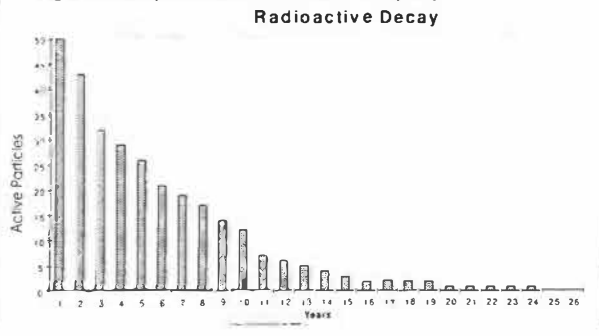


Figure 3: Spreadsheet Formulas

Year 1 All active	**Year 2** Checks random number value	**All other years** Checks previous year and then random number value	**Random Numbers** (1–6)
1	=IF(AR2<5,0,1)	=IF(B2=0,0, IF(AS2>5,0,1))	=INT (RAND()*(5)+1)

TI-82 Calculator

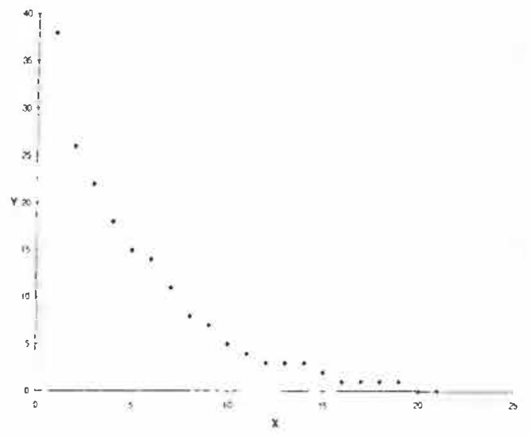
Display

Figure 4: Calculator Screen Display

```
YEARS
                                    20
{38   26      22      18      15    ...
                                  Done
```

This program calculates how many years it took for all 50 particles to decay (20 years) and it also contains a list of particles remaining after each reiteration of the program. That is, after year 1 there were 38 particles active, after year 2 there were 26 particles active and so on.

Figure 5: Calculator Graph Display



Program

The variables used include the following:

A	=	the number of active particles remaining
T	=	the number of decayed particles (a counter for the number of times 6 occurs)
P	=	the number of years or loops required to get to zero active particles
R	=	a random number between 1 and 6
L1	=	the list of the number of active particles remaining at the end of each year
K	=	an index counter in the "for" loop

Figure 6: TI-82 Decay Program

```
PROGRAM: DECAY
: ClrHome
: ClrDraw
: 0 → a → T
: 0 → P
: 50 → A
: While A>0
: 0 → T
: P + 1 → P
: For (K, 1, A)
: int (rand*6+1) → R
: If R = 6
: Then
: T + 1 → T
: End
: End
: A – T → A
: Pt-On (P, A)
: A → L1 (P)
: End
: Pause
: Disp "YEARS", P
: Disp L1
```

Note: Active particles remaining are totaled at the beginning of each year in the spreadsheet and at the end of the year in the calculator program.

Modeling radioactive decay using different forms of technology and providing an opportunity for students to display and discuss their results develop a richer understanding of the situation and of the iterative algorithm involved. Similar features of the algorithms include generating a random number between 1 and 6 and finding its integer value. Related outcomes include determining whether a particle decays or not and counting the number of active particles remaining each year; however, the procedures used to determine these are significantly different depending on the technology used. For instance, a nested loop is used on the spreadsheet to determine particles' activity individually. The algorithm is to initially reference the previous column to see if the particle decayed in the previous year (If B2 = 0 then 0); then, if it is still active it references the random number generated and decides whether it decays or remains active (If AR2 > 5 then 0, otherwise 1).

The "For" loop and "List 1" in the calculator program does this quite differently. The algorithm here keeps a count of the particles remaining (A), randomly generates a number A times, counts how many 6s occurred and subtracts that from the previous total to determine the number remaining this year.

In discussions, students choosing to use a spreadsheet will find that defining the steps needed to simulate the situation and create a graph are simpler to complete than those choosing a programmable calculator; however, changing the parameters of the problem, such as the number of initial particles and the rate of decay, is much more cumbersome making it more difficult to explore related situations. The discussions surrounding the connections made between the displayed results, about the similarities and differences between the algorithms created and about the possibilities and constraints of the tools chosen become the most interesting and perhaps the most valuable part of this activity.

Part 2

Scenario 3: Hey, I've written a program on my TI-82 for something like this before. If I change it a bit and add a few lines, I can probably use it for this problem.

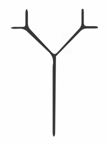
The scenario above approaches algorithms and technology from a perspective different from the previous discussion. Rather than starting from scratch, previously known algorithms can be altered or extended to model related events or to create new ones. Providing students with opportunities to recognize and analyze relationships between different situations and their algorithms promotes the development of mathematical connections. It also provides more creative opportunities in the mathematics classroom by allowing students to explore algorithms they are familiar with to create or pose new problems.

The following activity explores the extension of a basic recursive algorithm to create new mathematical objects. Recursion is an interesting phenomenon found in stories ("For my third wish," said the peasant to the genie, "I want three more wishes."), in art (Escher prints) and in nature (fractals). Its appeal and complexity are inherent in its structure: "a recursive definition is a circular definition that manages to avoid paradox. When something is defined recursively, it is defined in terms of itself" (Poundstone 1985, 123). Recursive programs are usually compact because they reduce the given problem to a simpler one or a subroutine of the same structure. However, this is also the cause of their complexity. After providing instruction on recursion, Harvey (1992, 432) stated that "almost all of the students could understand recursive procedures that [he] presented, but not all could reliably write their own recursive procedures." It may be inappropriate to expect that all secondary students be able to write their own recursive programs, but that should not exclude recursion from the curriculum. Students are quite capable of understanding basic recursive algorithms and making appropriate changes or additions necessary for the task.

Fractal trees can be created using technological devices that support graphics and recursion. For example, Logo and Geometer's Sketchpad can both be used to create binary trees. The basic geometric structure of the binary tree is Y-shaped; that is, each branch divides into two new smaller branches half the size and form a 90° angle symmetric about the previous branch. The Terrapin Logo program and the binary tree (level 3) are shown in Figure 7.

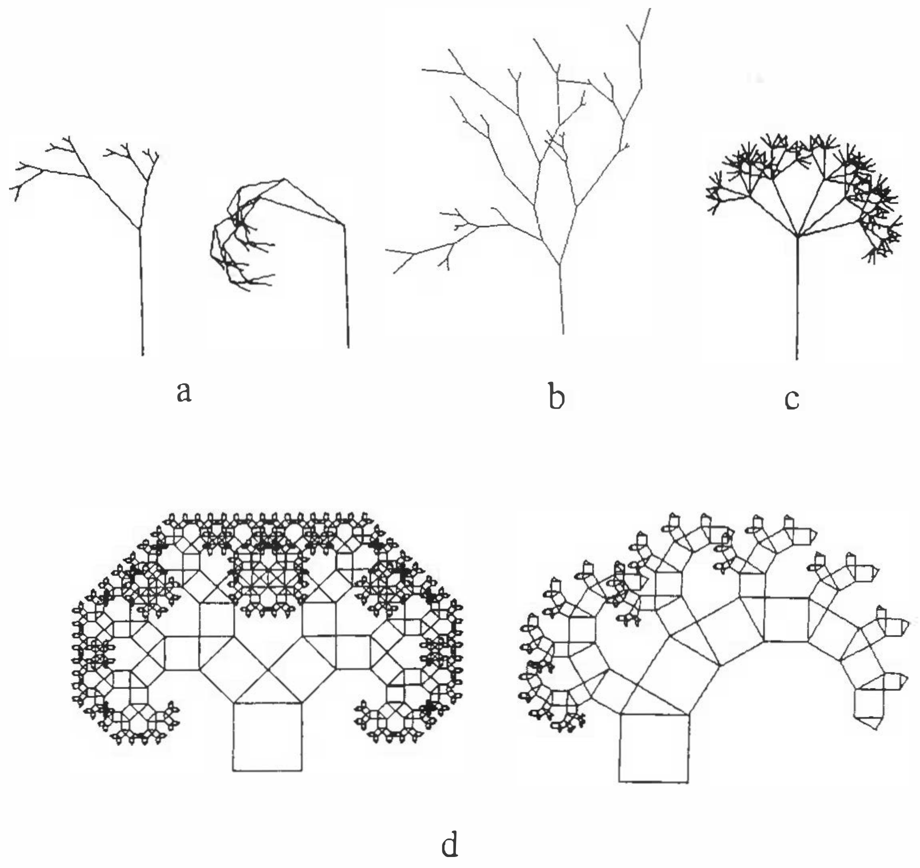
Figure 7: Binary Tree Logo Program

```
TO TREE :SIZE :LEVEL
  IF :LEVEL < 1 [STOP]
  FORWARD :SIZE
  LEFT 45
    TREE :SIZE/2 :LEVEL - 1
  RIGHT 90
    TREE :SIZE/2 :LEVEL - 1
  LEFT 45
  BACK :SIZE
END
```


Binary Tree in Logo and Geometer's Sketchpad

Again, it is of interest to compare the algorithms and products created from different technological tools (as we did in Part 1), but this particular activity also allows a number of creative possibilities by altering and extending the basic algorithm to create new but related objects. Once an algorithm is understood, a student is free to experiment by making changes and additions to it. For example, the Logo program above provided the basic structure of the fractal tree. Simple adjustments were made that altered the length and angle of the branches (Figure 8a); a random number generator available in Logo was used to produce branches of random lengths (Figure 8b); adding another call for recursion in the algorithm created trees or bushes with three or more branches extending from each node (Figure 8c); and numerous other variations are limited only by the imagination (Figure 8d Bosman's Pythagorus tree and a lopsided version).

Figure 8: Alterations and Extensions to Binary Tree



Using a basic algorithm and then adjusting it to fit alternate needs is a technique that computer programmers use frequently. It requires an understanding of the original program and problem solving skills to change it appropriately. Through investigation, experimentation and exploration, it is possible for algorithms to be used toward creative mathematical thinking.

Conclusion

Although we have broadened our view of the place of algorithms in mathematics, our increased access to a variety of technological tools requires that we push our thinking even further. The recent attention to discrete mathematics promotes the view of algorithms as useful problem solving aids. We presently encourage students "to develop and analyze algorithms" (NCTM 1989, 176) and to compare the efficiency of algorithms (Maurer and Ralston 1991); however, there is a need to include activities that allow students to make decisions as to which technological device best suits their purposes, to alter one algorithm to fit the constraints of a different tool and to alter and extend previously known algorithms to model related events or to create new ones.

References

Harvey, B. "Avoiding Recursion." In Learning Mathematics and Logo, edited by C. Hoyles and R. Noss, 393 428. Cambridge, Mass.: MIT Press, 1992.

Lovitt, C., and I. Lowe. Chance and Data Investigations. Vol. I. Australia: Curriculum Corporation, 1993.

Maurer, S. B., and A. Ralston. "Algorithms: You Cannot Do Discrete Mathematics Without Them." In Discrete Mathematics Across the Curriculum, K 12, 1991 Yearbook, edited by M. Kenney and C. Hirsch. Reston, Va.: National Council of Teachers of Mathematics, 1991.

National Council of Teachers of Mathematics (NCTM). Curriculum and Evaluation Standards for School Mathematics. Reston, Va.: NCTM, 1989.

Poundstone, W. The Recursive Universe: Cosmic Complexity and the Limits of Scientific Knowledge. Chicago: Contemporary Books, 1985.