25 Preparing for the computer age at an early age

Harriet Fell
College of Computer Science, Northeastern University
Boston, Massachusetts, USA

Abstract

In developing a secondary-school mathematics curriculum, it is important to consider the kinds of experiences children should have to help underpin the development of the more abstract notions they are likely to encounter in later years of schooling. Many sophisticated ideas from modern mathematics and computer science are accessible to children and adults with very little mathematical background. The ideas are made more accessible by experiential projects with links to such topics as finite state automata, fractals, graph theory and cryptography. Scientists and mathematicians can bring their knowledge to the schools though clubs and demonstrations which will eventually infiltrate the classroom.

Keywords

Creativity, discrete mathematics, experimental, parents, transfer.

BEFORE SECONDARY EDUCATION

In developing a secondary school mathematics curriculum, it is important for us to consider the kinds of exposure children may or should have before they reach high school. We must start early if we are to enable students to reach the high of levels mathematical sophistication they will need to survive in and contribute to our increasingly technological society. I present here a series of projects that invite students to study mathematics with their heads and hands. The goal of these projects is to get young students involved in discovering and discussing mathematics. The projects allow students to study many examples, formulate conjectures, find counter-examples, and sometimes find impossibilities. In some cases the projects have direct applications to technology, in others technology

Information and Communications Technologies in School Mathematics
J.D. Tinsley & D.C. Johnson (Eds.)

helps in presenting or exploring a mathematical idea, yet others are just to foster independent mathematical thought, with or without technology.

From Math Club to Classroom

We cannot expect elementary school teachers to keep up, on their own, with the rapid scientific advances in today's world. Parents with training in math and science can partake in their own children's education and contribute to their school's programs by running science, math, and computer clubs within their children's schools. The examples, presented in this paper, are from an elementary school math club that I ran for two years. Yes, that means the children involved were bright kids from a privileged neighbourhood, kids who already knew that they liked math. It is my hope that projects, like those I describe here, can be integrated into ordinary classrooms. Our projects did not stop at our Monday morning math club. Enthused children showed their work to classroom teachers and many of our projects were adopted and adapted for classroom use. Several teachers started making appearances at our meetings.

Math Club to Home

Several parents came to help out at our club, some regularly and others occasionally. The youngest children (6 years old) were expected to bring a parent along. The students often elaborated on their work at home and brought the results in to share. Three years have passed since my youngest child finished elementary school, the club is still alive, run by parents who helped out when I was there. The projects I contributed are about to be reused with a new group of students.

SAMPLE PROJECTS

Some of these projects started in college computer science classes and were then presented to six- to twelve-year-old children at my local elementary school math club. Other materials were designed for the math club but could easily serve as projects for high school groups.

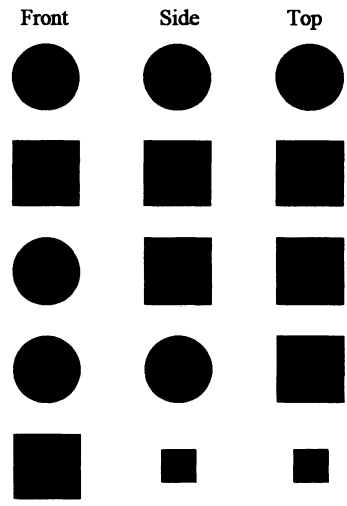
Modelling clay mathematics – spatial perception

It may seem that this project has little to do with technology but actually it is a result of thoughts on computer-aided machining. Modelling clay is a great material for exploring surfaces and solids. We first used clay to study ways of describing 3-dimensional objects in 2-dimensions. Though there is wonderful software available for visualising three-dimensional objects, I think it is important for children to have the experience of visualising and manipulating three-dimensional objects in three-space to fully understand the two-dimensional projections they see.

Students were asked to build convex solids given the top, side, and front views (see figure 1). Every child quickly made a sphere for the first problem and a cube for the second. Could they find any other solutions? ... Yes. This means that the three projections do not supply enough information for a computer to direct a machine to construct the object, even with the convexity condition.

Many came up with a short cylinder for the third set of projections. The fourth problem stumped them all but they kept trying and were able to see and discuss why their models didn't solve the problem.

The (only?) solution is a solid that commonly appears as a calculus volume problem, the intersection of two cylinders. I showed them one, made of brass, that I machined when I was in postgraduate school.



For each row, make a solid with no holes or dents that has the front, side and top views shown.

Figure 1 Front, side and top views

The fifth problem has no solution and even young students could articulate why it couldn't be done.

In a second exercise, we built elaborate terrains from contour maps. Finally, the contour maps were replaced with series of level curves of a surface, the result being a Klein bottle, imbedded in three-space.

Point, line, square, cube, tesseract – spatial perception, discrete mathematics

This is another project that doesn't directly relate to computers but as the students carry out the combinatorial analysis of the objects they build, they are getting a taste of empirical analysis, deduction, and induction that play a part in discrete mathematics.

Using small Styrofoam balls for the vertices (I realised later that gumdrops are cheaper and work better) and wires for edges, students build a line, square, cube, and projection of a tesseract. This is really a study in counting, finding patterns, and expressing the results by formula. As they build each object, the students record the number of vertices, edges, faces, volumes it contains. They realise that they join two squares to form a cube and two cubes to form a tesseract. They see that the number of vertices keeps doubling. Several saw that a cube came from two squares plus four new edges, one for each vertex in a square; a tesseract came from two cubes plus eight new edges, one for each vertex in a cube and thus predicted the number of vertices and edges in a 5-dimensional hypercube.

Finite state automata – symbolic manipulation systems

Automata theory and formal languages provide a way of thinking mathematically about computers and are deeply related to specific fields of computer science, e.g. lexical analysis in compilers. Though these subjects are commonly taught to upper level computer science majors, many aspects of them are accessible with no special mathematical background (Papadimitriou, 1981). They offer an excellent opportunity for mathematical description, discussion, and informal proof.

In our study of finite-state automata, children acted out the machine. Each actor played a state. A simple costume showed the state's name and whether it was an accepting state. Each state actor had a script telling them what to do (where to pass the tape) for each alphabet symbol (see figure 2). An input string was written out on adding machine tape and placed in a paper coffee cup with a slit in the lid. The cup was handed to the start state who read the first symbol and handed the cup to the appropriate state actor who read the next symbol and so on. When the tape ran out, the string was deemed good if the last state was accepting and bad otherwise. The good strings were taped on one side of the blackboard, the bad ones opposite. The audience had standard schematics of the automaton and together with the actors tried to describe the pattern of the accepted strings. Everybody wanted a chance to be an actor. Everyone participated in proposing descriptions of the accepted language, and coming up with counter examples of proofs that the description was correct.

Figure 2 A Machine that accepts 20¢; Alphabet = {5¢, 10¢, 25¢}. The 15¢ state (Tova, on the right) is reading the tape.

The children are wearing signs that tell their states. In this case, you can see the '5' and '10' states. The girl on the right is holding a coffee cup with a tape and reading her script that tells what to do depending on what she reads. She is either the '15' or '20' state and the boy, second from the left, is the other. Off the right side of the picture, is another child/state, the accepting state '20'. Any amount greater that 20 is also accepted.

After three weeks of play-acting we switched to: "Try to design automata to do the jobs described below."

M1	alphabet = { m, n, o } accepts only the string mom.
M2	alphabet = {a, b} accepts any string that doesn't end with bb.
M3	alphabet = {a, b} accepts any string of length greater than 4.
M4	alphabet = {0, 1} accepts any string that has 000 in it.

About a quarter of my thirty six- to twelve-year-olds were able to solve these problems (e.g., see figure 3)—not all of my senior computer science majors are able to do this. The other students still loved the exercise. They all made reasonable starts at the problems and then found counterexamples of mishandled strings for each other's designs.

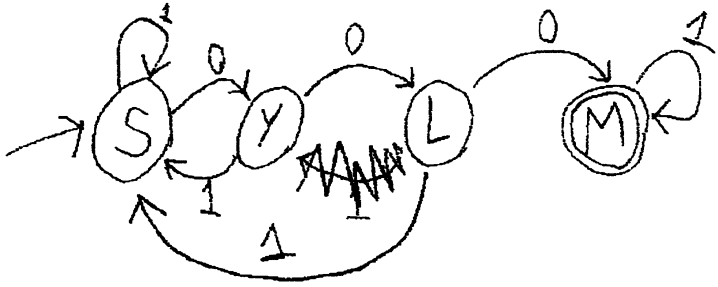


Figure 3: A solution for M4 designed by an elementary school student.

Fractals - geometry, discrete mathematics, direct manipulation

Fractals, popularised by Benoit Mandelbrot in the mid 70s, now play an important role in modelling natural phenomena and in generating realistic computer graphics (Mandelbrot, 1977). Today's school-age children have grown up with the fractal terrains of science fiction movies, many are aware of the Mandelbrot set. They are anxious to learn more about these marvels. We did several exercises involving fractals some with drawing or building, others with counting.

Building fractal trees

This is an introduction to how fractals can generate plant-like structures. In addition to paper and pencils, students were given small coloured stickers, a penny, and a clear plastic 'tree tool'. With a few pictures and a minimum of instruction, they set off to create trees. The stickers are for leaves or flowers at the ends of the branches.

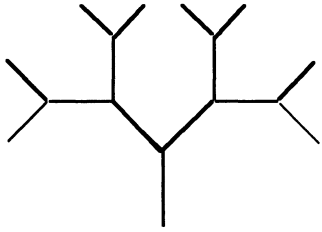


Figure 4 A Tree

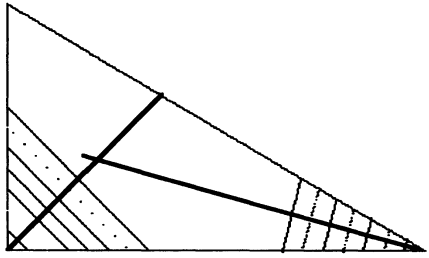


Figure 5 The Tree tool

The instructions: Place the tree tool at the tip of a branch so the dark line lines up with the branch. Add two new branches by drawing along the edges of the tree tool (see figures 4 and 5). There are different ways to make a fractal tree. Try some of these:

1. Always use the right (90°) angle and the same length for the branches.
2. Always use the right (90°) angle but make the branches get shorter the farther out you go.
3. Try the same things, always using the 30° angle (see figure 6a).
4. Each time you are about to add two branches, toss a coin. Use 90° if it came up heads and 30° for tails (see figure 6b).
5. Find a way to use coin tosses to decide on the lengths of branches.

With dice and a vast supply of 1/4" thick squares of varying sizes, students were able to build fractal terrains, not as convincing as those of *Lucas Films*, but they got the point. In both these exercises, students experienced the way in which randomness can perturb a deterministic algorithm to generate variety in nature.

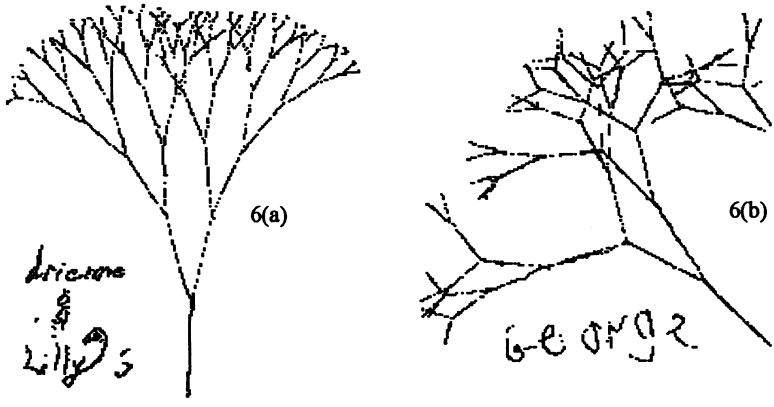


Figure 6 Student trees.

Students met Sierpinski's Triangle by drawing many levels on a triangular grid and then rediscovered Sierpinski's Triangle in Pascal's Triangle. They also drew many levels of snowflake curves. While drawing, they counted edges or triangles and deduced formulas to tell how many occurred at each level. This exercise like the counting of edges and vertices while building a tesseract is great preparation for the combinatorial analysis they will need if they ever have to analyze algorithms. It is also a clear introduction to recursion.

Graph theory—algorithms

It is easy to explain the problems of finding the shortest or cheapest path, or minimal spanning tree in a graph to someone with no mathematical background beyond simple arithmetic. It is also very easy for children to understand that these are relevant problems to scheduling airlines or even to routing packets on a computer network. Graphs are easy to visualise, draw, and talk about. Children can propose algorithms and try carrying out each other's proposals. They don't necessarily come up with perfect algorithms but it is the act of proposing a method, and stating it clearly enough so that someone else is able to perform it that is important.

A program we have our computer science freshmen write in Algorithms Data Structures II draws graphs and animates several standard graph algorithms. Printouts of the drawing provide a large source of sample graphs for elementary school children to experiment with (e.g., see figure 7). They can, eventually, watch an animation and try to deduce the algorithm being executed.

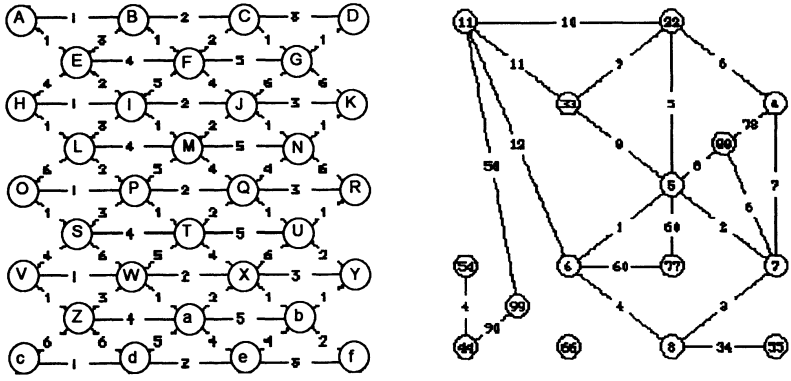
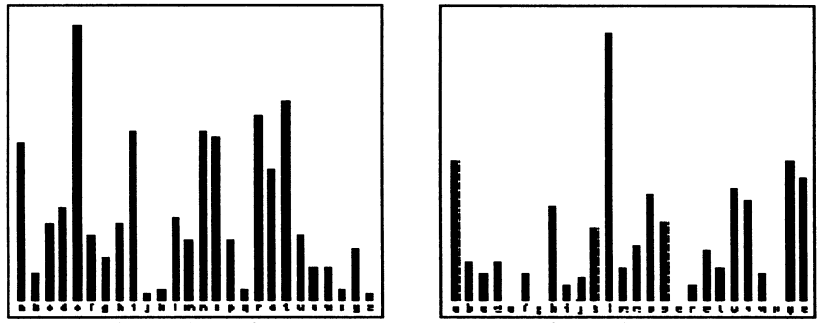


Figure 7 Sample graphs for cheapest path and minimal spanning tree searches.

Cryptography—statistics

Secret codes have always appealed to children, and encryption is crucial to the success of communication technology. In this project, children create a histogram of occurrences of characters in a passage and match their result to a bar chart showing the probability of letters in common English. From the match they

deduce the Caesar shift necessary to decode their personal messages. They gain a sense of how a probability distribution and an exact count can be close to each other but not exactly the same. This is another project that came from my freshman computer science class. The college students had to write programs to draw the histogram, the children did it by hand.



Frequency of letters in English

Frequency of letters in message

Figure 8 Frequency graphs.

Encrypted Message:
ulcly ohkdl zlluz vthuf mpylm splzj vunyl nhalk vuvul zwvaa olfms pjrlk
aoyvb noaol ayllz puzdh ytzao lfjyh dslkv uaoln yhzza olibz olzhu kaolv
spcla yburz aolfk ypmal kpuzd hytzv clyvb yolhk zhuks huklk vuaol ybnzs
prlny llult ilyzn spaal ypunz aylht zvmao ltmsl dvbav clyao lihfz dpysp
unvcl yaold halyh ukaol uypno avujb laolw vywvp zlzhw wlhyl k

Decrypted message:
never had we seen so many fireflies congregated on one spot they flicked
throu ghthe trees inswa rmsth eycra wledo ntheg rasst hebus hesan dtheo
livet runks theyd rifte dinsw armso verou rhead sandl anded onthe rugsl
ikegr eenem bersg litte rings tream softh emfle wouto verth ebays wirli
ngove rthew atera ndthe nrigh toncu ethep orpoi sesap peare d

Figure 9 Breaking a secret code.

Game of Life - algorithm, simulation

The Game of Life is an example of cellular automata that has entertained mathematicians and computer scientists since it was invented by John Horton Conway in 1970. The game is a simulation of living cells evolving according to simple rules. Simple rules, however, can lead to a very complex structure. In addition to just following the rules for life and death, students get to design creatures that will live, die, move, grow, go through planned changes, or just do things they never imagined.

Though the rules for going from one generation to the next are simple to describe, they are not trivial to follow as you must not confuse the new generation with the last. We used Xs on a grid to represent the last generation and bright pink squares for the new generation. Then we copied the new generation as Xs onto a new grid. We also used a lovely interactive computer program of the game designed by Richard Rasala (Internet) for yet another computer science exercise.

OTHER TOPICS

There were many other equally sophisticated topics we worked on: plotting functions of one variable—lines, parabolas, circles, the Serpentine Curve, and the Witch of Agnesi (Hodgman, 1959)—affine functions of two variables (finding the image of the unit square with the letter P written in it.); binary numbers; infinite series (Cohen, 1991); π (Beckmann, 1971); LOGO; Möbius Strips; Fibonacci Numbers and the Golden Ratio—including sunflowers and pineapples (Thompson, 1992)—Patterns in Numbers (Burns, 1982); and Sound.

OBSERVATIONS AND CONCLUSION

Our math club was a success, 20 to 30 children, three to six parents, and two or three teachers showed up early Monday morning just to do math. The math club projects filtered into the classroom. More girls showed up than boys—that may not be a measure of success but it does say that there was something in my methods that appealed to girls. A twelve-year-old student wrote that Math Club,

> "... has altered first thing Monday, traditionally a dreaded time, into a meeting with a cheerful atmosphere and compelling exercises that are actually fun to do. It has changed the image of Math in mind from boring, photo-copied sheets full of tedious numbers and operation signs into exciting projects such as calculating how to efficiently put board-walks across a city or how to make an authentic tree out of fractals or to make a 3-D mountain range that could possibly be a setting for a Science Fiction movie. ... It has shown me that there are NO limits to mathematics; they can stretch as far as the mind will allow." (Adrienne Newberg, 1993).

There are many topics in mathematics and computer science that are accessible without much background. They inspire mathematical exploration, description, discussion, conjecture, and proof. They allow children to get an early start at the abstract thought that is necessary for higher mathematics and computer science.

Clubs run by parent-scientists fit into any level of elementary or secondary school. They can bolster the work of teachers and without interfering with classroom curriculum can help introduce ideas from the forefront of technology.

FUTURE WORK

The Internet now makes it possible to spread these ideas. We should create depositories to make project materials available to parents or teachers. Such a site should, however, be backed up by people willing to answer questions. That still leaves the issue of such a site not being universally accessible, but though there may be a gap now, I believe that gap will continue to narrow and we must plan on computer technology becoming at least as accessible as books are now.

REFERENCES:

Beckmann, P. (1971). *A History of Pi*, Dorset, New York.
Burns, M. (1982). *Math for Smarty Pants*, Little, Brown and Company, Boston.
Cohen, D. (1991). *Calculus by and for young people*, Don-Cohen—The Mathman®, Champaign, IL.
Hodgman, C. (ed), (1959). *C.R.C. Standard Mathematical Tables*, 12th. Edition, Chemical Rubber Publishing Company, Cleveland.
Lewis, H and Papadimitriou, C. (1981). *Elements of the Theory of Computation*, Prentice-Hall, New Jersey.
Mandelbrot, B. (1977). *Fractals, Form, Chance, and Dimension*, Freeman, San Francisco.
Newberg, A. (1993). Letter in the *Horace Mann News Notes*, June 18.
Thompson, D. (1992). *On Growth and Form: The Complete Revised Edition*, Dover, New York.

Internet reference
Rasala, R. *The Game of Life Project*, rasala@ccs.neu.edu.

Harriet Fell has been on the faculty at Northeastern University for twenty-five years, first in mathematics and later in computer science. She has directed many students in the creation of systems for persons with disabilities. She is co-developer, with Linda Ferrier, of the Baby-Babble-Blanket, an interface for infants with severe motor disabilities, and is now working on an Early Vocalisation Analyzer. With colleagues Viera Proulx and Richard Rasala, she is developing a laboratory-based curriculum for computer science freshmen. She has run an elementary school math club, a Brownie Scout math project, and is computer badge counsellor for Boy Scout Troop 205.