Programming and Problem Solving with the TI-83 Plus: The Structured Search

A. Craig Loewen

Problem solving has always been one of the most difficult areas of the mathematics curriculum to teach well. Despite 25 years of research devoted to improving problem-solving skills, many students still struggle with how to start a problem and how to be flexible (that is, switching from an ineffective strategy to a new one). Our students do not seem to have a wide range of problem-solving strategies at their disposal and instead focus on such strategies as guess-and-test when no algorithm seems immediately available. Guess-and-test seems to have become even more popular in an age when calculators and computers aid in making quick, accurate computations.

Now, don't get me wrong. I'm not speaking against guess-and-test as a strategy. It is a legitimate, recognized and viable strategy, but in some ways it seems a bit inelegant and it is often inefficient.

There are, of course, many different problem-solving strategies, and we know that more sophisticated solvers combine these strategies. For example, the elimination strategy works well only if you are able to generate an effective and comprehensive list, and drawing a picture usually helps identify an appropriate formula. In general, all problem-solving strategies become more powerful when they are blended with other strategies. Add to that the power of the minicomputer. When we can harness the ability of the hand-held computer to quickly make repetitive computations, we may further enhance several of these strategies. The computer can be used to quickly generate lists, check conditions in the problem and complete exhaustive searches. All this can be done much more quickly than we can do on our own. In general, with a few simple programming skills (which are easily mastered by a high school student), we can blend these strategies and tools to create something we could call a *structured search.*

The Structured Search

In a structured search, we would set up a loop that effectively generates a list of possible values that can then be tested against the conditions of the problem. The TI-83 Plus gives us at least two efficient ways to create this kind of loop. This simple program uses the "for"-loop and writes the values 1–5 on the calculator screen:

```
PROGRAM:ABA
:1→A
:While A<6
:Disp A
:A+1→A
:End
:
```

This following program, which uses a "while"-loop, does the same thing:

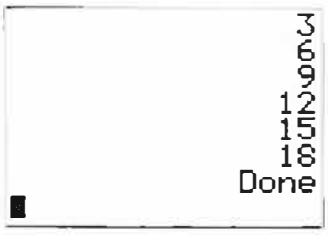
```
PROGRAM:ABA
:For(A,1,5)
:Disp A
:End
:
```

Although the differences between the two types of loops are quite small, the "for"-loop is a little more elegant and much easier to enter. The "for"-loop also forces a fixed number of repetitions of the commands within the loop, whereas the second type of loop is more flexible, continuing indefinitely until a condition is met.

To complete the structured search, we also need to know how to enter simple logical checks, which are called "if"-statments. Let's modify our "for"-loop program above to do the following:

- Consider all the values from 1 through 19, using the command For(A,1,19)
- Print on the screen all of the values of A that are divisible by 3, using the command If A/3 = int(A/3):Disp A

Here is the program:



Here is the screen display when the program is run:

```
PROGRAM:ABA
:For(A,1,20)
:If A/3=int(A/3)
:Disp A
:End
:
```

Knowledge of the other mathematical functions available on the TI-83 Plus, together with these simple commands (and a little exploration), will provide us with another means of tackling a variety of problems.

Cryptarithms

A common type of problem appearing in many magazines and newspapers (and even on the Internet) is cryptarithms. A cryptarithm is an arithmetic statement where the digits have been replaced by letters. Here is a classic cryptarithm:

ABCD
× 4
CDBA

In this problem, each of the letters A through D needs to be replaced by a single digit to create a four-digit number that, when multiplied by 4, produces the same four digits in reverse order. Note that both As must be the same digit, which is true for all four letters.

It is a lot of fun to solve this problem by hand, but it is fun to solve it with a structured search, too. We need to set up four loops (one for each variable, A, B, C and D), use them to build the number ABCD, multiply that value by 4 and see what happens.

- Is it possible that there are no solutions to this problem? Is it possible there is more than one solution?
- How would you approach this problem if you were solving it without programming? What is a reasonable strategy?
- How long would you estimate it would take to solve this problem by hand?

Here is a structured search that could solve this problem:

```
PROGRAM:ABCD
:For(A,1,9)
:For(B,0,9)
:For(C,0,9)
:For(D,1,9)
:1000*A+100*B+10
*C+D→E
:1000*D+100*C+10
*B+A→F
:If E*4=F:Disp E

:End
:End
:End
:End
:
```

- Why do the A and D loops run from 1 to 9, while the B and C loops run from 0 to 9?
- Could this program generate any extraneous solutions? How could you tell?
- How many solutions does the program generate?
- How many values will this program consider in all?
- What does the line A × 1000 + B × 100 + C × 10 + D → E do?
- What are the values that are stored in the variables E and F?
- Why are there four "end"-statements at the bottom of the program?
- Adapt the program above to solve this similar problem: ABCD × 9 = DCBA.

Notice how long the calculator takes to run through the entire list of possible solutions. An obvious disadvantage of the structured search is the time required to execute it (although it is still much quicker than doing it by hand). A significant advantage is that the search has considered all possibilities, something we would never be willing to do by hand.

But wait a minute! Does the program have the computer test possibilities that are not reasonable? In other words, is there a way to further limit the number of possibilities considered and thus speed up the program? We already limited A to the values 1 through 9 because 0 cannot appear as the lead digit. However, consider that A must be either 1 or 2. If A is 3 or greater, a digit will be carried into the 10,000s place when A is multiplied by 4—and the product must not involve more than four place values. This one change significantly delimits the number of possibilities we need to consider. We can change the line For(A, 1, 9) to read For(A, 1, 2).

- How many possibilities does this one change eliminate?
- Are there other letters that could be further limited?

Strangely, with this process, we are slowly moving toward greater emphasis on another strategy: applying logical reasoning. Again, we see how strategies become more powerful when blended. If we continue with this process, we may find that logical reasoning leads us through to another, even more elegant solution to the problem. This is the joy of problem solving—identifying a number of ways to attack a problem and implementing a broad range of skills and strategies, moving gracefully between and among them as the solution is built.

Another important idea emerges as we play with this problem. A critical problem-solving skill is being able to identify when a strategy is not appropriate or does not apply. Unless we can significantly limit the number of checks the program needs to make, it could become a very tedious process. For example, when a problem requires five or more variables, it would probably be better to turn to a more powerful computer to engage the search or turn to a different strategy altogether.

Another Example

Let's look at another example of a cryptarithm to explore how we can modify our program to check for solutions:

```
  AB
+ BA
 CDC
```

The first thing we notice is that there are still four variables, but neither A, B nor C can equal zero. We will need to modify our "for"-loops. Obviously, we also need to modify our "if"-statements.

- What would the new "if"-statements look like? How many would we need?
- How many possible solutions could this search generate? How would we check for extraneous solutions?

One possible program looks like this:

```
PROGRAM:ABBA
:For(A,1,9)
:For(B,1,9)
:For(C,1,9)
:For(D,0,9)
:11*A+11*B→E
:101*C+10*D→F
:If E=F:Disp A*1
000+B*100+C*10+D

:End
:End
:End
:End
:
```

When run, the program generates several possible solutions, but be sure to check for ineligible ones; that is, solutions where two letters are assigned the same digit.

Here, another important quality of the problem-solving process is reinforced. Looking back is critical, although it is often overlooked. It is tempting to think that, because the program generated all of these results, they are all viable. This is not the case. Looking back helps confirm which possible solutions are real solutions. It is at the looking-back stage that we are most likely to catch our mistakes (computational or logical) and thus learn from our experience.

Challenges

- How many solutions are there to the equation ABC + CBA = DDD?
- Program your TI-83 Plus to solve the following equation: ABCDE × 4 = EDCBA. Estimate how long you think the calculator will require to generate its results. Use what you know from the ABCD × 4 = DCBA problem. Can you think of ways to limit the search?
- Try to build a routine in your program to eliminate ineligible answers
- Try to generate your own substitution problems that can be solved through a structured search.

A Money Problem

Consider another familiar problem that can be effectively solved using a structured search:

> Uri has 48 coins in his pocket, all nickels and pennies. Altogether, he has exactly $1.72 in change. How many nickels and how many pennies does he have?

We could set up a system of linear equations to solve this problem, but it is also fun to write a simple program as an alternative solution. The program looks like this:

```
PROGRAM:CENTS
:For(N,0,34)
:48-N→P
:If 5*N+P=172:Di
sp N,P
:End
:
```

- How are the dimensions of the N-loop (nickels) determined?
- Why does this program require only one loop?
- How many possible solutions are there?
- Describe the connection between the system of linear equations related to this problem and the program above.

- Assume that Uri also has some dimes. Is there a solution to this problem? How many solutions are there to this problem? Construct a structured search to solve this revised problem.
- Write a money problem of your own that could be solved using a structured search.

Some Challenging Problems

There are many different problems that can be solved using a variation of the structured search. Here are two somewhat more challenging problems with related programs. You may wish to try solving the problems yourself with or without your calculator, or you may find it interesting to work your way through the program, trying to determine the effect of each line.

Find the smallest number that, in each case, produces a remainder that is one less than the divisor when divided by each of the values 2 through 10.

```
PROGRAM:AZ
:ClrHome
:11→N
:0→C
:While C=0
:1→C
:For(A,2,10)
:N-int(N/A)*A→T
:If T≠A-1:Then
:0→C:10→A:N+1→N
:Output(1,1,N)
:End
:End
:End
:Disp N
:
```

A perfect number is defined as a number that equals the sum of all its factors. For example, the first perfect number is 6 because its factors 1, 2 and 3 have a sum of 6. What are the next two perfect numbers?

```
PROGRAM:PERFECT
:2→C
:ClrHome
:For(A,2,500)
:Output(1,1,A)
:0→T
:For(B,1,√(A))
:If A/B=int(A/B)
:T+B+A/B→T
:End
:If √(A)=int(√(A
)):T-√(A)→T
:If T-A=A:Then:O
utput(C,1,A):C+1
→C:End
:End
:
```

The topic of perfect numbers has captivated many mathematicians over the centuries. It is worth reading about these numbers and finding other algorithms that have been defined for identifying them more easily.

Cryptarithms and Alphametics

Included below is a collection of cryptarithms and alphametics. An alphametic is a special type of cryptarithm in which the letters used to replace the digits in an equation also form comprehensible words. Sometimes the words themselves form phrases. Here is a familiar alphametic:

```
  SEND
+ MORE
MONEY
```

It is not practical to solve the above alphametic with a hand-held computer because it involves eight different letters and thus eight different loops. How many different possibilities would the calculator have to consider in order to solve this problem?

The following puzzles were taken from www.freepuzzles.com.

How many solutions are there for each of the following?

AB + B = BA

C + C + C = DC

WAS × S = ASAW

A × C × AC = CCC

Here are two slightly more challenging puzzles taken from the same website. Each of these problems could also be solved with a structured search.

$A^2 + B^2 + C^2 = D^2 + E^2$

In the equation above, the letters represent consecutive positive integers. Find the corresponding value for each letter.

$(30 + 25)^2 = 3{,}025$

Break the number 3,025 into two parts, 30 and 25. The square of (30 + 25) equals 3,025, as shown. Two more numbers share the same property. Can you find them?

The following website has a huge number of cryptarithms and alphametics, as well as an aid to solve the puzzles: www.tkcs-collins.com/truman/alphamet/alphamet.shtml.

TO + GO = OUT

I + DID = TOO

Conclusion

Doing problems and puzzles like these is a fun and motivating activity that can be easily implemented into the mathematics classroom. The structured search provides another problem-solving tool for effectively approaching these problems.