ARML Lecture: Recursion and Generating Functions
Arvind Thiagarajan, 2007-2008
Thursday, April 24, 2008

1 Recursion

Recursion is a fairly common topic, both on higher-level contests and on short answer timed contests like ARML. A recursion is essentially a sequence of elements where each element can be calculated from the previous elements in the sequence. That is, a recursion is a sequence $a_1, a_2, a_3$ etc. such that $a_n = f(a_i)$ for all $i$ less than $n$ and for some function $f$. Perhaps the most well known recurrence is the Fibonacci Sequence, given by the recurrence $a_n = a_{n-1} + a_{n-2}$ and the base case $a_0 = a_1 = 1$. Now, you might have learned from SCT that the most efficient way to compute the $n$th element of a recursive sequence is by dynamic programming. Dynamic Programming (DP) is the method in which you calculate all $a_i$ for $i < n$ before you calculate $a_n$. This might seem like a pretty obvious method so we won’t stress it much. We’ll just show how it applies for the Fibonacci Sequence and then let you try out some practice problems. So for the Fibonacci Sequence, suppose we want the 9th term. Using dynamic programming, we start computing the sequence from the bottom up:

$$
\begin{align*}
  a_2 &= a_1 + a_0 = 2 \\
  a_3 &= a_2 + a_1 = 3 \\
  a_4 &= a_3 + a_2 = 5 \\
  a_5 &= a_4 + a_3 = 8 \\
  a_6 &= a_5 + a_4 = 13 \\
  a_7 &= a_6 + a_5 = 21 \\
  a_8 &= a_7 + a_6 = 34 \\
\end{align*}
$$

and we’re done. All we really did here was list the numbers we knew so far and to find the next term in the sequence we added the most recent two terms that we’d found. Whenever you have a recursion and you decide to use DP on it, you must try to just keep the recurrence relation in mind and efficiently calculate. Thus, as long as the term of the recursive sequence that you’re looking for is low enough in the sequence (i.e. the index $n$ of $a_n$ is relatively small) then the DP doesn’t take much effort. Thus, the main part of any problem involving recursion is finding the recurrence relation. The best way to learn how to do this is to do problems. Before you start though, here are a couple of tips:

- Often your recurrence will not be entirely sequential. You’ll see what I mean in a concrete sense on problem 7 below, but understand that for certain problems you will really have two recurrence relations that depend on each other, i.e sequences $a_n = f(a_i, b_i)$ and $b_n = g(a_i, b_i)$ for all $i$ less than $n$. In this case, don’t be daunted; just do the DP but calculate terms together, i.e calculate $a_1$ and $b_1$ first together, then calculate $a_2$ and $b_2$, etc.

- If you have a problem with a certain dimension that is large enough to make the problem seem complicated but small enough to seem like it might be an index in a recursive problem, try setting up a recurrence where the answer depends on the answer to the same problem with smaller dimensions. (See problem 2)
At times, when you have found a recursion and the value of $n$ is obscenely high but finding a closed form expression for $a_n$ is either unfeasible or annoying, look for cycles. If you find a cycle in your sequence of length $l$, then $a_n = a_i$ iff $n ≡ i \pmod{l}$.

2 Recursion Problems

1. (Traditional) Consider a 2-player game where each player takes turns removing between 1 and 7 stones. Whoever can't remove a stone loses. There are 2006 stones and you go first. How many stones do you remove in order to win?

2. (Mandelbrot 4 2006, 6) Two rows of ten pegs are lined up and adjacent pegs are spaced 1 unit apart. How many ways can ten rubber-bands be looped around the pegs so that no peg does not contain a rubber band? (Rubber bands cannot stretch more than $\sqrt{2}$ units.)

3. (MStueben) In how many ways can you make change for a quarter with pennies, nickels, and dimes?

4. (AIME I 2006, 11) You have 8 cubes of size 1 through 8. You must build a tower with the constraint that the cube on top of a cube of size $k$ must have a size of at most $k + 2$. How many different towers can be constructed?

5. (HMMT Guts) How many 10-bit binary strings can be made so that no three consecutive bits are the same?

6. (JSteinhardt) I have seven bowls of varying sizes. I cannot put a larger bowl in a smaller bowl. How many possible arrangements of bowls can I make (multiple stacks are allowed)?

7. (TJARML 2008) I have green beads and blue beads and I want to put them on a string such that nowhere on the string are there either two consecutive green beads or three consecutive blue beads. How many ways can I construct a string of beads of length 10 given this condition.

8. How many different strings of ones and zeros are there of length 12 such that no two ones are adjacent?

9. (AIME I 2008, 11) Consider sequences that consist entirely of A’s and B’s and that have the property that every run of consecutive A’s has even length, and every run of consecutive B’s has odd length. Examples of such sequences are AA, B, and AABAA, while BBAB is not such a sequence. How many such sequences have length 14?

10. (AIME II 2008, 9) A particle is located on the coordinate plane at $(5, 0)$. Define a move for the particle as a counterclockwise rotation of $\frac{\pi}{4}$ radians about the origin followed by a translation of 10 units in the positive $x$-direction. Given that the particle’s position after 150 moves is $(p, q)$, find the greatest integer less than or equal to $|p| + |q|$. (This isn’t what would be traditionally classified as a recursion problem, but meh.)

3 Generating Functions

Eventually you’ll reach a point where you’ll have a recursion problem with a high value of $n$ or a problem that ask you to find $a_n$ in terms of $n$. The first of these is infeasible to do by DP and the second one is impossible to do by DP. We need a new way to do this, and so we stumble upon this generating function business. So:
The ordinary generating function $A(x)$ of a sequence $a_n$ is defined as $A(x) = \sum_{n=0}^{\infty} a_n x^n$. Given this, there are three properties of generating functions that we particularly value in the context of this lecture:

1. $A(x) + B(x)$ is the generating function for the sequence $a_n + b_n$

2. $A(x)B(x)$ is the generating function of the sequence $c_n = \sum_{i=0}^{n} a_i b_{n-i}$ (this is a convolution, and whenever you realize that it’s in a problem, you should probably consider using generating functions)

3. $x^i A(x)$ is the generating function for the sequence $a_{n-i}$

You should verify all of these properties on your own. Now, we return again to our recursion problem. The problems we can solve with generating functions are limited to recurrences of the form $a_n = \sum_{i=0}^{n-1} c_i a_i$. Let’s look at this for a simpler case, $a_n = Ma_{n-1} + Ka_{n-2}$. This can be rewritten as $a_n - Ma_{n-1} - Ka_{n-2} = 0$. Now, if the two sequences in this equation are equal, their generating functions must be equal. The generating function equation then is $A(x)(1 - Mx - Kx^2) = O(x)$ where $O(x)$ is simply an term that is linear in $x$ that accounts for the fact that the generation functions for $a_{n-1}$ and $a_{n-2}$ do not start at $x = 0$. Now, we have $A(x) = \frac{O(x)}{1 - Mx - Kx^2}$. Now, use partial fraction decomposition on the right side and then Taylor expand what remains. Then write the newly expanded right side as one compact series in the form $\sum_{n=0}^{\infty} (\text{stuff})x^n$.

Then, you will have that that said stuff is equivalent to your sequence, and thus you will have solved for the general form of $a_n$. In fact, you will find that $a_n = C_1 r_1^n + C_2 r_2^n$ where $C_1, C_2$ are constants determined by the base case of the recursion and $r_1, r_2$ are the roots of $r^2 - Mr - K$.

Now, we turn to a generating function which is more likely to be relevant for ARML. This is the dice generating function. First we impose the constraint that the dies may only have non-negative integers on them and that a die is the same regardless of the orientation of its faces. The motivation behind making this generating function the way we will make it comes from three things: 

1. We want a generating function which tells us how many ways a given number can be obtained by rolling the dice.

2. We want the generating function to be unique, i.e each die has a unique generating function associated with it.

3. We want the product of the generating functions of two dies to tell us how many ways a given number can be obtained by rolling both the dies simultaneously.

Thus, given these goals, we must choose what specific sequence we wish to encode. Uniqueness dictates that a strict ordering be imposed on the sequence for a die. Thus, we choose the sequence to be $a_n$ such that $a_i$ represents the number of times $i$ appears on the dice. We find this extremely convenient, for we already recognized that rolling two dies simultaneously requires us to take the convolution of the sequences that we have just defined, and so our generating function will behave exactly as we wish for it to. Using this generating function, we have that the generating function for the regular, well-loved dice is $(x + x^2 + x^3 + x^4 + x^5 + x^6)$. Now, since we’re probably running out of time around now, we’ll do a few generating functions problems.
4 Generating Functions Problems

1. Find, with proof, the general solution of $a_n = \sum_{i=0}^{n-1} c_i a_i$.

2. In how many ways can two dice be chosen such that, when rolled together, the probability of obtaining a sum of $i$ is the same as the probability of obtaining $i$ if two normal six sided dice are rolled?

3. In how many ways can two six sided dice be chosen such that, when rolled together, the probability of obtaining a sum of $i$ is the same as the probability of obtaining $i$ if two normal six sided dice are rolled?

4. (Matthew Crawford) Virginia has a pair of fair 8 sided dice with faces numbered 1-8 on each die. Montana has a pair of fair 8 sided dice with faces numbered with positive integers in such a way that when her pair of dice is rolled, the probability of any particular sum occurring is the same as when Virginia rolls her dice. The largest number on any face of either of Montana's dice is 11. Find the sum of the numbers on the faces of the die whose faces are all less than 11.

Finally, I apologize for the lack of 5s having been promised.