COT 3100: Induction Preliminaries

Mathematical induction is a powerful proof technique that can be used in many areas of mathematics. Fundamentally, the technique can be used to prove open statements about the non-negative or positive integers. (More generally, it can be used to prove any statement for all integers greater than or equal to some integer n_0.)

In this course, in order to learn induction, we'll have you practice the technique on several types of problems:

a) Summations
b) Divisibility Problems
c) Matrix Problems
d) Recurrence Relations

This is NOT exhaustive (you may see problems from other realms), but many of the induction questions we use in the course come from one of these four categories.

In the course, you've already received ample background in divisibility and mod, but we haven't seen any background information on summations, matrices or recurrence relations. Thus, the goal of this lecture is to provide background information on each of these topics, so that you can handle the usual algebra that comes up in induction questions within these realms.

Summations

In many instances, analyzing an algorithm to determine its efficiency requires adding up many numbers. This amounts to completing a summation. Sums are so common, that mathematicians have developed some short-hand notation, so that it's easier to unambiguously represent a long sum that might otherwise take a long time to write out fully.

Here is an example of a sum written out:

5 + 7 + 9 + 11 + 13 + 15 + 17 + 19 + 21 + 23 + 25 + 27 + 29

Here is how that sum is written out in summation notation:

$$\sum_{k=2}^{14}(2k+1)$$

Given a summation, to determine what it represents, do the following:

1) Create a running total set to 0.
2) Set the variable in the bottom of the sum equal to the initial value given, (in this case, 2).
3) Plug this value into the expression. (In this case, 2k+1).
4) Add this to your running "total".
5) If your variable equals the last value listed, in this case, 14, stop and your answer is what is stored in total. Otherwise, plug in the next integer value for the variable and go to step 3.

In code, we would have something like this:

```
int total = 0;
for (k=2; k<=14; k++)
  total += (2*k+1);
```

In general, we would say the following:

$$\sum_{k=a}^{b} f(k) = f(a) + f(a+1) + f(a+2) \ldots + f(b)$$

where a is less than or equal to b and both are integers.

The very first sum for which we can get a formula is as follows:

$$\sum_{k=a}^{b} c = (b - a + 1)c$$

because we are adding c exactly b – a + 1 times and repeated addition is multiplication.

Now, let's consider a more difficult sum:

S = 1 + 2 + 3 + 4 + … + (n-1) + n

Our answer will be in terms of n. A nifty trick that will help us is pretending that we want to add the numbers backwards. Now, if we add up both of these equations, we notice a peculiar simplification for the sum of each column:

S = 1 + 2 + 3 + 4 + … + (n-1) + n
S = n +(n-1) + … + 2 + 1

2S = (n+1) + (n+1) + … +(n+1) + (n+1)
2S = n(n+1)
S = n(n+1)/2

What we notice is that each column adds up to the same exact thing! The reason for this is that to move from column to column, we add 1 on the top row, and subtract 1 from the bottom row. This net movement (+1, -1) amounts to no net gain or loss. So, if the sum of the values in the first column is n+1, the sum of the values in the second column must be as well. Since there are n columns, we get n(n+1) when we add up all of these numbers. We get the final result when we divide by 2. In summation notation we have:

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Now, let's look at a few quick uses of this formula:

$$\sum_{k=1}^{100} \mathrm{k} = \frac{100(100+1)}{2} = 5050$$

$$\sum_{k=1}^{2n} \mathrm{k} = \frac{(2\mathrm{n})(2\mathrm{n}+1)}{2} = \mathrm{n}(2\mathrm{n}+1)$$

$$\sum_{k=1}^{4n-1} \mathrm{k} = \frac{(4\mathrm{n}-1)(4\mathrm{n}-1+1)}{2} = 2\mathrm{n}(4\mathrm{n}-1)$$

The next formulas that we will examine are

$\sum_{k=a}^{b} \mathrm{cf(k)} = \mathrm{c}\sum_{\mathrm{k=a}}^{\mathrm{b}} \mathrm{f(k)}$, where c is constant with respect to k.

$$\sum_{k=a}^{b} (\mathrm{f(k)} + \mathrm{g(k)}) = \sum_{\mathrm{k=a}}^{\mathrm{b}} \mathrm{f(k)} + \sum_{\mathrm{k=a}}^{\mathrm{b}} \mathrm{g(k)}$$

The reason this is true is because we can always factor out a constant in all terms of a sum. For example, 3(4) + 3(5) + 3(6) = 3(4 + 5 + 6), using factoring. The first is a sum with the 3 inside each term, while the sum is a sum that is then multiplied by 3.

Here are quick examples that use these rules:

$$\sum_{k=1}^{2n} 4\mathrm{k} = 4\sum_{\mathrm{k=1}}^{\mathrm{2n}} \mathrm{k} = \frac{4(2\mathrm{n})(2\mathrm{n}+1)}{2} = 4\mathrm{n}(2\mathrm{n}+1)$$

$$\sum_{k=1}^{n} (\mathrm{k}+3) = \sum_{\mathrm{k=1}}^{\mathrm{n}} \mathrm{k} + \sum_{\mathrm{k=1}}^{\mathrm{n}} 3 = \frac{\mathrm{n}(\mathrm{n}+1)}{2} + 3\mathrm{n} = \frac{\mathrm{n}^2 + 7\mathrm{n}}{2}$$

In problems where the mathematical induction technique is used, the following "split" idea is used:

$$\sum_{k=a}^{c} f(k) = \sum_{k=a}^{b} f(k) + \sum_{k=b+1}^{c} f(k)$$

This formula is valid as long as a ≤ b < c. The reason this formula is used (splitting a sum), is because sometimes the value of $\sum_{k=a}^{b} f(k)$ is a known quantity. Thus, when given a summation of the form, $\sum_{k=a}^{c} f(k)$, one critical trick because figuring out the value of b that is appropriate and helpful, to split one sum into two, in this manner.

Index Shift Idea

Another idea to tackle a sum is realizing that something like $\sum_{k=20}^{40} \mathrm{f(k)}$ can also be written as $\sum_{k=1}^{21} \mathrm{f(k+19)}$. If you plug into both of these summations, it's fairly easy to see that they are adding up the terms f(20) + f(21) + f(22) + … + f(40). This is called an index shift. In general, we can express an index shift as follows, assuming that 1 ≤ a ≤ b:

$$\sum_{k=a}^{b} \mathrm{f(k)} = \sum_{k=1}^{b-a+1} f(k+a-1) = \sum_{i=0}^{b-a} f(k+a)$$

The latter two are included just to show an equivalent sum that starts at 1 or another one that starts at 0. Here is the index shift applied a sum:

$$\sum_{k=n-1}^{2n+1} (3\mathrm{k}-4) = \sum_{k=0}^{n+2} (3(k+n-1)-4)$$

$$= \sum_{k=0}^{n+2} (3k+3n-3-4)$$

$$= \sum_{k=0}^{n+2} (3k+3n-7)$$

$$= \sum_{k=0}^{n+2} (3k) + \sum_{k=0}^{n+2} (3n-7)$$

$$= \frac{3(n+2)(n+3)}{2} + \frac{2(n+3)(3n-7)}{2}$$

$$= \frac{(n+3)}{2}(3n+6+6n-14)$$

$$= \frac{(n+3)(9n-8)}{2}$$

$$= \frac{9}{2}n^2 + \frac{19}{2}n - 12$$

Hybrid Sum

In recitation, we learned how to handle both arithmetic and geometric sums. Now, let's look at a sum that is neither arithmetic nor geometric:

$$S = \sum_{i=1}^{n} i2^{i-1} = 1(2^0) + 2(2^1) + 3(2^2) + \cdots + n2^{n-1}$$

Let this sum be S. Now, multiply S by two and we get:

$$2S = 2\sum_{i=1}^{n} i2^{i-1} = 1(2^1) + 2(2^2) + 3(2^3) + \cdots + n2^n$$

Now, subtract the second equation from the first:

$$S - 2S = 1(2^0) + 1(2^1) + 1(2^2) + \cdots + 1(2^{n-1}) - n2^n$$

Notice that for each term of the form 2^i, the coefficients differ by 1 in the two sums, so what we have left is largely a regular geometric sum:

$$-S = \frac{1-2^n}{1-2} - n2^n$$

$$S = n2^n - \frac{2^n - 1}{2-1}$$

$$= n2^n - (2^n - 1) = n2^n - 2^n + 1 = (n-1)2^n + 1$$

Matrices

Are grids of numbers…they are really, really useful to computer scientists…at the end of the day, an image in its raw representation, is usually just a grid of numbers, where those numbers specify colors.

A matrix has some number rows and columns. Here is a 2 by 3 matrix:

$$\begin{bmatrix} 2 & -1 & 3 \\ 4 & 2 & 9 \end{bmatrix}$$

Adding Matrices

Dimensions have to be identical:

$$\begin{bmatrix} 2 & -1 & 3 \\ 4 & 2 & 9 \end{bmatrix} + \begin{bmatrix} 6 & 8 & 2 \\ -5 & 1 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 7 & 5 \\ -1 & 3 & 16 \end{bmatrix}$$

Just add corresponding terms to get the resultant term

Subtraction is the same subtract first term minus second term for each slot:

$$\begin{bmatrix} 2 & -1 & 3 \\ 4 & 2 & 9 \end{bmatrix} - \begin{bmatrix} 6 & 8 & 2 \\ -5 & 1 & 7 \end{bmatrix} = \begin{bmatrix} -4 & -9 & 1 \\ 9 & 1 & 2 \end{bmatrix}$$

Multiplication of Matrices

Let M1 have r1 rows, c1 cols

Let M2 have r2 rows, c2 cols

We can multiply M1 x M2, if and only if c1 = r2. (Note: it's possible that we can multiply M1 times M2, but NOT be allowed to multiply M2 times M1.)

The result of the multiplication is always has r1 rows and c2 columns:

$$\begin{bmatrix} 1 & 2 & 3 \\ -2 & 4 & -1 \end{bmatrix} \times \begin{bmatrix} -3 & 5 \\ -4 & 6 \\ 0 & -5 \end{bmatrix}$$

$$= \begin{bmatrix} 1(-3) + 2(-4) + 3(0) & 1(5) + 2(6) + 3(-5) \\ -2(-3) + 4(-4) + (-1)(0) & -2(5) + 4(6) + (-1)(-5) \end{bmatrix}$$

$$= \begin{bmatrix} -11 & 2 \\ -10 & 19 \end{bmatrix}$$

In general, to get the entry in row x, column y, we "multiply" row x by column y. What it means to multiply a row by a column, is roughly the definition of a dot product, if you happen to be familiar with that concept. If you are not, what we do is multiply each corresponding term and add those values.

$$\text{Result[x][y]} = \sum_{i=1}^{c1} M1[x][i] * M2[i][y]$$

Matrix Multiplication in code:

```
public static int[][] mult(int[][] m1, int[][]
m2) {
    int[][] res = new
int[m1.length][m2[0].length];
    for (int i=0; i<m1.length; i++)
       for (int j=0; j<m2[0].length; j++)
           for (int k=0; k<m1[0].length; k++)
               res[i][j] += (m1[i][k]*m2[k][j]);
    return res;
}
```

One other thing to note is that matrix exponentiation is just like regular exponentiation (and can only be done on square matrices). Thus, if M is a matrix, the following statement is true:

$M^{a+b} = M^a M^b$

Using this rule turns out to be very common in matrix induction problems.

Note: A vast majority of induction problems with matrices involve square matrices only.

Recursively Defined Sequences

Fibonacci – wondered about multiplying rabbits. In his problem, a pair of rabbits appeared in month one that were new. It took that pair one month to mature. Then, starting in month 3, that mature pair of rabbits would give birth to a new pair every month. All the other pairs of rabbits would operate in this fashion. His question was, after 12 months, how many pairs of rabbits would there be?

Month 1 – 1 pair of rabbits (new)
Month 2 – 1 pair of rabbits (mature)
Month 3 – 2 pairs of rabbits (1 mature, 1 new)
Month 4 – 3 pairs of rabbits (2 mature, 1 new)
Month 5 – 5 pairs of rabbies (3 mature, 2 new)

Not too hard to see…all rabbits in month k-1 become mature in month k. All rabbits alive in month k-2 give birth to new rabbits in month k.

Let F_k = # of pairs of rabbits alive at month k.
Then for k >= 3, by the definition of the story, we have

$F_1 = 1$, $F_2 = 1$, for all $k > 2$, we have $F_k = F_{k-1} + F_{k-2}$

This is a recursively defined sequence! Hallmarks of a recursively defined sequence:

Some initial terms are given.
To generate the rest of the terms, a formula is given to find the k^{th} term in terms of previous terms of the sequence.

Here is a recursive definition of factorials:

$0! = 1$
for all $k > 0$, $k! = k * (k-1)!$

Here is a recursive definition of the combination function:

$C(n, 0) = C(n, n) = 1$, for all non-negative integers n.
$C(n, k) = C(n-1, k-1) + C(n-1, k)$ for all $n > 1$ and $0 < k < n$.

The most simple way to list a particular term in a recursively defined sequence is to build up the sequence until the desired term:

What is F_{12}? (Fibonacci's original question!)

1,1,2,3,5,8,13,21,34,55,89,144

Answer is 144…