Chapter 1
Overview

Sherri Shulman

Math-linux.com

Oct 15, 2013

Preliminaries/Math review

1. Logarithms
2. $x^A x^B = X^{A+B}$
3. $\frac{x^A}{X^B} = x^{A-B}$
4. $(X^A)^B = X^{AB}$
5. $X^N + X^N = 2X^N$
6. $2^N + 2^N = 2^{N+1}$

1. $X^A = B$ iff $\log_X B = A$ $A, B, C > 0, A \neq 1$
2.

$$\log_A B = \frac{\log_C B}{\log_C A}$$
$$X = \log_C B, Y = \log_C A, Z = \log_A B$$
$$\text{So } C^X = B, C^Y = A, A^Z = B$$
$$C^{Y^Z} = C^{YZ} = B = C^X \text{ so } X = YZ, \text{ so } Z = \frac{X}{Y}$$
$$\text{so } \log_A B = \frac{\log_C B}{\log_C A}$$

1. $\log AB = \log A + \log B,\ A, B > 0$ (Assume log base 2)
2. $$\begin{aligned} X = \log A,\ Y &= \log B, Z = \log AB \\ \text{then } 2^X = A, 2^Y &= B, 2^Z = AB \\ \text{so } 2^X 2^Y = 2^Z &= AB = 2^{X+Y} \\ \text{so } X + Y &= Z \end{aligned}$$

1. others ... which you can derive similarly:
2. $\log A/B = \log A - \log B$
3. $\log(A^B) = B \log A$
4. $\log X < X$ for all $X > 0$
5. $\log 1 = 0, \log 2 = 1, \log 1024 = 10$

1. Series
2. $\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$
3. $\sum_{i=0}^{N} A^i = \frac{A^{N+1}-1}{A-1}$
4. If $0 < A < 1$ $\sum_{i=0}^{N} A^i \leq \frac{1}{1-A}$
5. When $n \to \infty$ this sum approaches $\frac{1}{1-A}$

$$\begin{aligned} S &= 1 + A + A^2 + A^3 + \cdots \\ AS &= A + A^2 + A^3 + \cdots \\ S - AS &= 1 \\ S &= \frac{1}{1-A} \end{aligned}$$

1. Or $\sum_{i=1}^{\infty}(\frac{i}{2^i})$

$$S = \frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots$$
$$2S = 1 + \frac{2}{2} + \frac{3}{2^2} + \cdots$$
$$2S - S = S = 1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} \cdots$$
$$= 2 \text{ by the series closed form for } 0 < A < 1$$

1. $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$
2. There are a number of proofs of this one. It's very common.
3. There are a couple of other series/closed forms on p 5
4. I'll also post a scan of some others for your use.

1. Modular arithmetic.
2. $A \equiv B \pmod{N}$ if n divides $A - B$.
3. The remainder is the same.
4. $A + C \equiv B + C$ if $A \equiv B \pmod{N}$
5. $AD \equiv BD$ if $A \equiv B \pmod{N}$

1. Proof by induction.
2. Basic form:
3. Base case
4. Inductive Hypothesis for a value k
5. Proof of the proposition for $k+1$ based on the hypothesis that it is true for k.

1. Suppose we want to prove that the Fib series converges: $F_i < (5/3)^i$
2. If you had me for DM, you know that I always say some definitions of Fib start at 0 some at 1. It is the same, just shifts the series. We'll start at 0: $F_0 = 0, F_1 = 1, F_{n+2} = F_{n-1} + f_{n-2}$.
3. For the base case we choose the two base cases of the defn:

$$F_0 = 0 < (5/3)^0$$
$$F_1 = 1 < (5/3)^1$$

4. Now we assume the proposition is true for arbitrary $k \geq 1$.
5. Now prove for $k + 1$:

1. $k+1$:

$$\begin{aligned}
F_{k+1} &= F_k + F_{k-1} \\
&< (5/3)^k + (5/3)^{k-1} \\
&< (3/5)(5/3)(5/3)^k + (3/5)(5/3)(5/3)^{k-1} \\
&< (3/5)(5/3)^{k+1} + (3/5)(5/3)^k \\
&< (3/5)(5/3)^{k+1} + (3/5)(3/5)(5/3)(5/3)^k \\
&< (3/5)(5/3)^{k+1} + (3/5)(3/5)(5/3)^{k+1} \\
&< (3/5)(5/3)^{k+1} + (3/5)^2(5/3)^{k+1} \\
&< ((3/5) + (3/5)^2)(5/3)^{k+1} \\
&< (24/25)(5/3)^{k+1} \\
&< (5/3)^{k+1}
\end{aligned}$$

1. Again ... this is a review.
2. So the above was induction proofs.
3. We also use proofs by counterexample. (If you assert that something is universally true, just need a counterexample to disprove.)
4. Proof by Contradiction:
5. To prove $p \rightarrow q$, Assume p, then assum $\neg q$. Show that creates a contradiction, so you can conclude q.
6. I just did this prime number example in DM. (p 7 in your book).

1. Intro to recursion
2. If you've taken Haskell, your're set
3. Otherwise ... very small sample of a recursive function in C (note that problem 1.4 really requires recursion as well and we already saw it in Rob Pike's grep program.)

```
int f( int x )
{
    if( x == 0 )
        return 0;
    else
        return 2 * f( x - 1 ) +  x * x;
}
```

```
int bad( int n )
{
    if( n == 0 )
        return 0;
    else
        return bad( n / 3 + 1 ) + n - 1;
}
```

```
void printOut( int n )  // Print nonnegative n
{
    if( n >= 10 )
        printOut( n / 10 );
    printDigit( n % 10 );
}
```

1. Recursion and induction
2. The base case of the recursion ist he base case of the induction.
3. The recursive case follows the recursion ... we assume the program works and then prove that it works based on this assumption.

1