COMP 220: Data Structures **Lecture Notes 1**

Logarithms

Student experience with logarithms varies widely. In these notes we look over the core properties of logs with an eye toward algorithm analysis.

Exponentials

Before we talk logarithms, let's talk exponentials. At this point in your life you should be pretty familiar with raising things to a power. In algebra you worked with a lot of variables with fixed exponents like x^2 or x^3. Exponentiation is generally thought of as repeated multiplication where we multiple the base, x, by itself for as many times as the exponent. This idea alone lets you derive a lot of the basic rules for exponent manipulation.

The product rules tells us we can add exponents when multiplying two exponentiations with the same base. To see this just expand out the two terms, x^a and x^b, of the product into repeated multiplications of x to see that the end result is the product of $a + b$ instances of x.

$$\begin{aligned} x^a * x^b &= (x_1 * x_2 * \ldots * x_a) * (x_1 * x_2 * \ldots * x_b) \\ &= x_1 * x_2 * \ldots * x_{a+1} * x_{a+2} * \ldots * x_{a+b} \\ &= x^{a+b} \end{aligned} \tag{1}$$

Similarly, we can manipulate the product of two exponentials if the base differs but the exponents match. Once again, expand out the individual terms of the product and then reorganize the terms of the expanded product[1] in order to pair up the two bases.

[1] because multiplication is commutative

$$\begin{aligned} x^a * y^a &= (x_1 * x_2 * \ldots * x_a) * (y_1 * y_2 * \ldots * y_a) \\ &= (x_1 * y_1) * (x_2 * y_2) * \ldots * (x_a * y * a) \\ &= (x * y)^a \end{aligned} \tag{2}$$

Raising an exponentiation to a power is equivalent to multiplying the exponents. We see this by once again expanding the exponentiation and examining the resultant product. The result is b terms of multiplying x by itself a times or all together as x multiplied by itself $a * b$ times.

$$\begin{aligned} (x^a)^b &= (x_1^a * x_2^a * \ldots * x_b^a) \\ &= ((x_0 * x_1 * \ldots * x_a)_1 * \ldots * (x_0 * x_1 * \ldots * x_a)_b) \\ &= x^{a*b} \end{aligned} \tag{3}$$

If you wanted to get formal about this you could identify the recursive structure in the repeated multiplication[2] taking special note of the base cases[3] and then use proof by induction to prove all of the

[2] $x^a = x * x^{a-1}$

[3] $x^0 = 1$ and $x^1 = x$

above properties.

Along the way you probably encountered negative exponents as a shorthand for fractions [4] and fractional exponents as a shorthand for roots [5]. Consider fractions as negative exponents. For any x and a we know that $\frac{x^a}{x^a} = \frac{1}{x^a}x^a = 1$ because of the properties of multiplicative inverses. Representing $\frac{1}{x^a}$ as x^{-a} and then applying the product rule yields:

$$\begin{aligned} \frac{1}{x^a}x^a &= x^{-a} * x^a \\ &= x^{a-a} \\ &= x^0 = 1 \end{aligned} \tag{4}$$

The math checks out because we choose the notation in such a way that it must!

[4] $x^{-2} = \frac{1}{x^2}$

[5] $x^{\frac{1}{2}} = \sqrt{x}$

Roots

Roots deserve their own section because they highlight a general relationship between functions that is important for understanding logarithms: *inverse functions*. We know that raising any number to the second power is typically referred to as the *square* function. The square root function is the function that inverts this operations such that you retain the original base of the square $\sqrt{x^2} = x$. The reverse[6] is also true if we allow for complex numbers. If we're only talking about positive, real valued numbers, then the n^{th} root[7] is the inverts, or is the inverse of, the n^{th} power and vice versa.

[6] $(\sqrt{x})^2$

[7] $\sqrt[n]{}$

Now back to exponent notation. Why use fractional exponents for roots. Remember that the inverse relation between a power and it's root tells us that for positive, real valued numbers we get $\sqrt[n]{x^n} = (\sqrt[n]{x})^n = x$. To work this out such that taking the root has an exponent based representation we must find the right value for the root and the right operation for the composition of a power and and a root. If taking the root is a special form of exponentiation then composition is just continued exponentiation[8]. If m is the exponent value for the n^{th} root, then we must find a value for m such that the following holds:

[8] $(x^n)^m$

$$(x^n)^m = (x^m)^n = x \tag{5}$$

That value must be $m = \frac{1}{n}$. Why?

$$\begin{aligned} (x^n)^{\frac{1}{n}} &= \left(x^{\frac{1}{n}}\right)^n \\ &= x^{\frac{1}{n}*n} \\ &= x^{\frac{n}{n}} \\ &= x^1 = x \\ &= \left(x^{\frac{1}{n}}\right)^n \end{aligned} \tag{6}$$

Logarithms

When determining the complexity of an algorithm we very often want to know how many times something will repeat[9] until it terminates. Very often this value varies with the some feature of our data or problem[10]. For example, let's say we're working with a vector and the size of that vector, n, can be different every time we run our algorithm. If we wanted to repeatedly cut the size of the vector in half until there were only one item left[11], how many times would we need to cut it in half?

[9] through iteration or recursion

[10] array/vector/data set size

[11] see Binary Search

Initially we have n items. After one cut, we have $\frac{n}{2}$. Then we cut that half in half to get $\frac{n}{4}$. But wait! This is just continued multiplication by $\frac{1}{2}$, a.k.a. repeated exponentiation by -2. After k cuts we have $\frac{n}{2^k}$ elements in our vector. So our original question is then for what value of k is the following true[12]:

[12] we're going to assume that n is a power of 2 for now

$$\begin{aligned} 1 &= n\frac{1}{2^k} \\ 2^k &= n \end{aligned}$$

The problem we face is that the variable we care about, k, is in the exponent not the base. If it were in the base, say k^{-2^n} then no problem, we can use roots to isolate k from n. This is not what we're dealing with though. We need a function such that $f(n^a) = a$ not $f(n^a) = n$. Where the k^{th} root gives us the the base of an k^{th} power where k is some known value, we need a function that gives us the unknown *exponent* of a constant number raised to the power. That function is the *base k logarithm* or $\log_k$.

When we refer to the *exponential function* we are referring to the act of raising some fixed base b to some variable power n, b^n. The base b logarithm[13] is the inverse of this function and vice versa. This means:

[13] $\log_b$

$$\log_b b^k = b^{\log_b k} = k \tag{7}$$

In English: the log base b of some number is *the power to which you would raise b in order to obtain that number*. Notice this means that for any b, $\log_b 0$ is undefined and $\log_b 1 = 0$ as no power of b is 0 and for all b, $b^0 = 1$. So, if you can only remember one thing about logarithms it should be this relationship with exponentiation. It combined with the exponent rules allows you to derive anything you need to know about logarithms.

Now back to our problem. If we have exactly $n = 2^k$ elements in our vector then after cutting it in half $k = \log_2 n$ times we have only one item remaining. Similarly, if we wanted to grow from size from 1 up to some $n = 2^k$ by successive doubling[14] then we'd need to double the size $k = \log_2 n$ times. What if we aren't dealing with powers of two? Perhaps I need to know the number of doublings that are need to get some n equal to or larger than 357? It can't be $\log_2 357$ as that's around 8.5. The answer must be 9 as strictly 8 is too few. This is easy to figure without a calculator and without knowing

[14] see dynamic arrays

the exact value of $\log_2 357$ if we understand the log/exponentiation relationship and know our powers of 2. From this perspective we're asking: "What is the powers of 2 bound 357 and which is the upper bound?" A quick check reveals our bounds are $256 = 2^8$ and $512 = 2^9$. This means $8 < \log_2 357 < 9$[15]. From here we can decide which bound solves our problem. In this case it's 9. More generally, for $n < 2^k$ we need to do $\lceil \log_2 n \rceil$ doublings and $\lfloor \log_2 n \rfloor$ cuts in half to reach our goals[16].

[15] $256 < 2^{\log_2 357} < 512$

[16] do you see why we round down when cutting but up when growing?

Algebra and Logarithms

It's often fun to find more verbose ways of expressing a number or expression in a mathematical statement. For example, notice that for any number $n \neq 0$ we can pick an arbitrary base $b \neq 0$ and $n = b^{\log_b n}$ or $n = \log_b b^n$. Why is this fun? Because from there we can start to derive and prove all the important algebraic properties of logarithms. Let's begin with the product rule.

$$\begin{aligned} \log_b (n * m) &= \log_b \left(b^{\log_b n} * b^{\log_b m} \right) \\ &= \log_b b^{\log_b n + \log_b m} \\ &= \log_b n + \log_b m \end{aligned} \tag{8}$$

The fact that the log of a product is equal to the sum of the logs of the terms is one of the oldest applications of logarithms[17]. From here we can have some real fun. What about division? Given that $\frac{a}{b} = a * \frac{1}{b} = ab^{-1}$, all we need to do is work out how to handle simple multiplicative inverses like $\frac{1}{n} = n^{-1}$. The key here is remembering that $n * n^{-1} = 1$.

[17] see slide rules

$$\begin{aligned} \log_b (n^{-1} * n) &= \log_b 1 = 0 \\ &= \log_b (n * n^{-1}) \\ &= \log_b n + \log_b n^{-1} \end{aligned} \tag{9}$$

$$\log_b n^{-1} = -\log_b n$$

In general we see that $\log_b n/m = \log_b n - \log_b m$.

What about raising numbers to a power in the presence of logarithms? Just remember that raising to a power is repeated multiplication and apply the product rule for logarithms.

$$\begin{aligned} \log_b n^m &= \log_b n_1 * \cdots * n_m \\ &= \log_b n_1 + \cdots + \log_b n_m \\ &= m \log_b n \end{aligned} \tag{10}$$

Notice that given what we proved about division, this rule works for an m, positive or negative[18].

[18] $\log_b n^{-1} = -1 * \log_b n$

Now what about roots? It's helpful to step back and recall a defining property for roots[19] and $n * n^{-1} = 1$. That is, $(n^p)^{\frac{1}{p}} = n$.

[19] like we did with division

$$\begin{aligned} \log_b (n^p)^{\frac{1}{p}} &= \log_b n \\ &= p \log_b n^{\frac{1}{p}} \end{aligned} \tag{11}$$

$$\log_b n^{\frac{1}{p}} = \frac{\log_b n}{p}$$

Once again, the exponent rule applies when roots are dealt with as fractional powers.

Notice that all these rules dealt with some operation that can be understood in terms of multiplication and the relationship that follow from it. This shouldn't be a surprise as the logarithm is the inverse of exponentiation which in turn can be viewed as repeated multiplication. What's worth noting is we have no algebraic rules for $\log_b(n+m)$. When addition occurs before the logarithm, then we're kind of stuck.

Base Conversion

In computer science we love powers of two and therefore love logarithms base 2. If all you have is a basic calculator or a programming library without generic logarithms built in then you're often stuck with $\log_{10}$[20] and the natural logarithm, or $\ln = \log_e$. This means we as computer scientists have a real vested interest in knowing how to convert between logarithm bases. Once again, we can prove/derive the formula by playing around with long-winded ways of writing down basic expressions. In this case we'll start from a way to say $n = n$ in terms of the logs and exponents and go from there.

[20] typically listed as just log

$$\begin{aligned} b^{\log_b n} &= a^{\log_a n} \\ \log_a b^{\log_b n} &= \log_a a^{\log_a n} \\ \log_b n * \log_a b &= \log_a n \\ \log_b n &= \frac{\log_a n}{\log_a b} \end{aligned} \tag{12}$$

An important consequence of this is that because any two logarithms differ by at most a constant factor[21], then in complexity space $\log_a n = O(\log_b n)$ for all bases a and b. For this reason we typically just say "Logarithmic Time" and ignore the base of the logarithm.

[21] $\log_a b$ or its inverse

Conclusion

We'll do some simple exercises to develop your intuition and thinking about logarithms. They will pop up time and time again this semester so start familiarizing yourself with all the properties we proved in these notes. Beyond getting to know your logs, these notes should also illustrate the value of really knowing and understanding some fundamental principles and definitions such that other important properties can be proved form those principles. We mostly teased out what you need to know about logarithms by playing around with algebraic properties of multiplication and some basic definitions. The technique that enabled this more often than not is looking and alternative, often verbose ways of writing down simple mathematical statements. When dealing with proofs, you often need to expand and manipulate rather than simplify.