Prereq Review #1: Logarithms

Ian Tullis, CS161 Summer 2022

You can't study algorithms for long before encountering logarithms.[1] They arise in a very natural way when studying the performance of many algorithms and data structures – for example, binary search (the subject of another of our prereq reviews) takes logarithmic, rather than linear, time. As the quarter goes on, we'll develop more of an intuition for *why* logarithms show up so often in CS theory, but our purpose here is to review how to manipulate them.

Definition

Taking a logarithm is the inverse of exponentiation. For positive real numbers b (the *base*) and x, we say

$$y = \log_b x$$

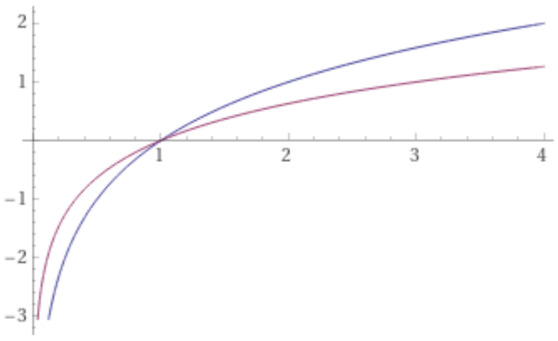
if and only if

$$x = b^y$$

For example, $\log_3 9 = 2$ because $9 = 3^2$.

Here are a couple of useful observations about logarithms that hold regardless of the base:

- $\log_b 1 = 0$, which makes sense because any quantity raised to the 0th power is 1. The log of any positive number less than 1 is a negative number, and the log of any number greater than 1 is a positive number. Here's a plot with $\log_2 x$ in blue and $\log_3 x$ in red:



- As the graph suggests, the log function is *strictly increasing*: for two values x_1 and x_2, $f(x_1) > f(x_2)$ if and only if $x_1 > x_2$, and $f(x_1) = f(x_2)$ if and only if $x_1 = x_2$.[2]

[1] The words *algorithm* and *logarithm* even happen to be anagrams!

[2] You may have encountered one use of this in CS109, for example – if you're trying to find a value that maximizes some quantity, it is equivalent to finding the value that maximizes the log of that quantity.

Just what is $\log x$?

If you have worked with logarithms in various classes at Stanford – especially across multiple departments – you have probably seen "log x" (without an explicit base) used somewhat confusingly to refer to all three of these things:

- $\log_{10} x$. For me, this was the most common log in high school.
- $\log_e x$ (AKA "ln x"). When I got to college and started studying chemistry, biology, etc., log suddenly switched to meaning this by default.
- $\log_2 x$, which computer scientists sometimes abbreviate as "lg x". This and the above are the most common logs in CS.

So which one will we focus on in CS161? Refreshingly, the answer is usually *none of them* – or, more accurately, every one everywhere all at once. As we will see, when we use asymptotic ("big-O") notation, the base of the log will not matter. However, sometimes we may run into logs in situations in which we care about, e.g., the exact number of steps needed for a procedure to complete, and then we will be careful to specify which base we mean. But if you see me just saying "$\log x$" and not specifying a base, you can safely assume that the base does not matter.

Regular world and log world

You may have heard the old joke about logarithms and snakes – logarithms "help adders multiply". What does that mean? Consider the following hierarchy of operations:

- Addition: $2 + 3 = 2 + 1 + 1 + 1$
- Multiplication: $2 \cdot 3 = 2 + 2 + 2$
- Exponentiation: $2^3 = 2 \cdot 2 \cdot 2$

and so on.[3] Working with logarithms (going into "log world") lets you treat operations on one of these levels in "regular world" as if they were one level earlier:

- $\log_b(x_1 x_2) = \log_b x_1 + \log_b x_2$ (a product in regular world becomes a sum in log world)
- $\log_b(x^a) = a \log_b x$ (an exponent in regular world becomes a product in log world)

[3]The next level up is called *tetration* – continuing with our pattern, it would be 2^{2^2}, i.e., $2^{(2^2)}$. This is not important for CS161, but it actually does come up in CS theory.

Some log tricks

Here's a list of a few properties that can be derived from the definition of the logarithm. I am generally more of a fan of re-derivation than memorization, but these are so ubiquitous that it's worth becoming comfortable with them:

- $\log_b \frac{1}{x} = -\log_b x$. Note that this is just a special case of the rule for the log of an exponentiated term (from the previous page), with $a = -1$.
- $\log_b \frac{x_1}{x_2} = \log_b x_1 - \log_b x_2$. You can see this by rewriting the left side as $\log_b x_1 (x_2)^{-1}$, then applying the rule for the log of a product (from the previous page) to get $\log_b x_1 + \log_b (x_2)^{-1}$, then using the rule for the log of an exponentiated term.
- $b^{\log_b x} = x$. To verify this, we fall back on the definition of a logarithm. Let y equal $b^{\log_b x}$, whatever it is. Then we can write

  $$y = b^{\log_b x}$$
  $$\log_b x = \log_b y$$

  And therefore $y = x$, since (as we saw at the end of page 1) two variables that have the same log value (in the same base) must be the same. Intuitively, this is what we would hope would happen – since taking a logarithm is the inverse of exponentiating, the two operations should cancel each other out.

  This one may seem particularly arcane, but it's fairly common in CS theory to try to stuff everything up into an exponent in order to make an argument easier.
- $\log_b a = \frac{\log_c a}{\log_c b}$, for *any* base c. To see this, rewrite the equation as $\log_c a = \log_b a \log_c b$, then use the definition of the logarithm on the left: $a = c^{\log_b a \log_c b} = (c^{\log_c b})^{\log_b a}$. Then, by the previous bullet point, we have $b^{\log_b a}$, which is just a.

Some common mistakes

These are written in red so that you don't try them.

- $\log_b (x_1 + x_2) \neq \log_b x_1 + \log_b x_2$. This is trying to treat a sum in regular world as a sum in log world. But regular world and log world are not on the same level.
- $\frac{\log_b x_1}{\log_b x_2} \neq \log_b \frac{x_1}{x_2}$. Again, this is trying to treat a quotient (a kind of multiplication) in regular world as a quotient in log world.