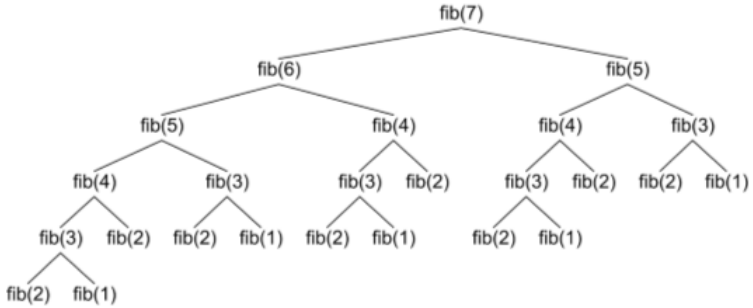
Recursive Efficiency

So, how efficient is the `fib()` function? What we mean by that is, how much memory does it use and how fast does it run? (Those are called the **space and time** measures of efficiency.) You can get a quick idea by simply calling `fib(42)`. It seems to take forever! **Why?**

This particular recursive implementation of `fib()` is **extremely inefficient**, because the function makes many **redundant calls**, calculating **exactly the same term in the sequence** several times. Given that the Fibonacci sequence can be implemented quickly and efficiently using iteration, this is more than a little disturbing.



The problem here is **not recursion**, but the naïve way in which is implemented. In this case, we are repeatedly calculating the same value. By using a different strategy, we can write a recursive version of `fib()` where all of these redundant calls disappear. You'll learn how to do that in the next lesson.