So the last big oh that we're going to look at is O of log n.

So I'm going to explain this with a sorted array.

And in order to do what I'm about to do here you have to have sorted data.

Let's say you want to find a number in this array.

And we're going to look for the number one.

What would be the most efficient way to find that?

What we're going to do here is we're going to take the array, we're going to cut it in half and say,

is the number one in the first half or the second half?

Well, it's not in the second half.

So we don't even have to look at any of those numbers.

And that may not seem like a big deal when you have an array with eight items in it.

But if you have an array with a million items in it, you just made it where you have a half a million

items that you don't have to look at.

So then we do this again.

We cut this in half.

It's not in the second half.

We're going to remove that and then we do it again.

And now we have found the number one.

This is a technique called divide and conquer.

So let's bring everything back here and count the number of steps that we went through to find that

item.

It was one.

Two.

Three.

And remember we had eight items in the array.

And two to the third power.

Equals eight.

So I'm going to turn this into a logarithm.

And if you're not a math person don't worry.

We're not going to go deep into this.

But look at the three numbers we have here the two, the three and the eight.

Log sub two of eight.

Equals three.

I'm going to back this up.

And we'll look at this again.

Log sub two of eight equals three.

We're basically saying two to the what power equals eight.

Two to the what.

Power equals eight.

Well, that number is three.

To put it another way, if we take the number eight and we repeatedly cut it in half, we divide it

by two.

How many times do we have to divide it by two to get down to one item?

And that is three times.

And the real power of this is when we get to very large numbers, let's say we have log sub two of this

number and this is over a billion.

How many times would you have to cut this number in half to get down to one item.

And that would be.

31 times.

To put it another way, two to the 31st power equals this number here.

That's over a billion.

So if you had an array with a billion items in it, and you were going to iterate through that array

linearly to find something and say what you were looking for was the last item, you would have to look

at a billion items to find it.

But if you use Divide and conquer, you could find any item in that array in 31 steps.

And that is the power of o of log n.

So now let's look at this on a graph.

And see that this is very flat.

Not as flat as O of one, of course, but very flat, very efficient compared to O of n and O of n squared.

At this point I want to show one other thing on the graph.

I'm not going to talk about this in a lot of detail, but I just want to show this and that is O of

n times log n that is used in some sorting algorithms.

And this is the most efficient that you can make a sorting algorithm unless you're sorting only numbers.

But if you're going to sort multiple types of data, you're going to sort strings.

This is as efficient as you can make a sorting algorithm.

We will talk about this and how we came up with n times log n when we get to those sorting algorithms,

because it will make more sense then.

So that's why I'm not going to talk about this now.

I'm going to take this back out.

So for the rest of the course, everything that we look at is going to be one of these four.

So that.

Is O of log N.