So I just want to wrap up our section on big O by looking at how these time complexities compare to

one another.

So I'm going to bring up the graph.

And let's say n is equal to 100.

If n is equal to 100, what are all of these going to be equal to.

Well o of one is always going to be equal to one.

O of log n is going to be approximately seven if n is 100.

Obviously, and is going to be equal to 100 and O of n squared is going to be equal to 10,000.

So you can already see a big disparity between these numbers.

But let's say we increase n to a thousand.

O of one is still going to be one.

O of log N is going to be approximately ten.

So two to the 10th power is 1024, which is close to 1000.

So O of log N of 1000 is going to be approximately ten.

Oh event is going to be a thousand, but O of n squared is a million.

So you can see that one is going to grow very, very fast.

So there are different words and phrases that you'll hear used with each of these.

So o of n squared.

Is going to be a loop within a loop.

O of n, you'll hear the word proportional.

O of log in, you'll hear the phrase divide and conquer.

And with O of one.

You'll hear the phrase constant time.

So these are our big four that we're going to look at in this course.

So now I'm going to take us over to a website that I think is a great resource for learning big O.

It is big O cheat sheet.com.

So at the top of the page you have this graph.

These two were not going to see in this course.

In fact, O of n factorial.

Is really something that can only happen if you're writing bad code on purpose.

So unlikely that you're going to see that, but you can see that O of n squared is in the horrible category.

Remember o of n times log n is going to be when we're using certain sorting algorithms.

And down here is where we want to stay most of the time.

So let's scroll down a little bit here has these tables.

So these are common data structures.

So we have time complexity here and it's broken into average.

Notice these are all beginning with the Greek letter theta and worst which begins with the Greek letter

Omicron or O for big O.

And space complexity over here.

Notice they're all the same except for one of them, which we're not going to build in this course.

But everything is O of N except for that one.

So everything that we're going to build for data structures have identical space complexities.

And this is one of the reasons why we're going to spend more time on time complexity than space complexity,

because for data structures it's all going to be the same.

But it's different when we scroll down.

To the sorting algorithms.

This one for the time complexity we have best, which is omega average theta worst.

Oh, let's see, the space complexities are all over the place.

So you have something like selection sort.

That has a terrible time complexity, but a great space complexity.

Versus merge sort, which has a great time complexity and not nearly as good of a space complexity.

So these time complexities down here only work if you're sorting numbers, if you are going to sort

strings or different kinds of data other than numbers, the best time complexity you can get for a sorting

algorithm is O of n times log n.

I will also include a link to this site.

This is a great resource and I encourage you to go check this out.

And with that, that wraps up our section on big O.