# Raw Transcript: It is essential that you understand big O when you go into a technical interview, because this will

# absolutely come up.

# Understanding this will also make you a better coder.

# So what is it?

# So think of it like this.

# We have two sets of code.

# We have code one and we have code two.

# How do we determine which one is better.

# Now both of these are going to accomplish the same thing but they're written differently.

# Is code one better.

# You know maybe it is easier to read or is code two better?

# Well, you know, maybe there are other things about that that make code two better.

# Well, big O is a way to mathematically figure out which of these two is better, which one runs more

# efficiently.

# So obviously we can't measure this with a scale.

# So I'm going to bring in a stopwatch to make the point.

# Let's say we run code one, and when we run it, we start a stopwatch and it runs for 15 seconds.

# Then we reset the stopwatch and we run code two.

# And code two.

# Runs a lot longer than 15 seconds.

# Well, obviously we want our code to run as quickly as possible, be as efficient as possible.

# So by this measure, code one is the better code.

# This is what is called time complexity.

# But we don't measure time complexity in time.

# We measure it in the number of operations.

# And the reason we do that is if you took the same code and we ran it on a faster computer.

# Obviously it would complete in less time, but the number of operations would be the same.

# So either way, on the fast computer or the slow computer, the time complexity would be the same.

# The number of operations would be the same.

# Also, with big O, we measure what is called space complexity.

# And space complexity is the amount of memory that something uses.

# So let's say code one, while you know it's very fast and runs in 15 seconds or comparatively fast.

# Maybe it uses a lot of memory.

# And then we reset all of this and we look at code two.

# And while that runs a really long time, the full minute, maybe it uses less memory.

# And if this is what's most important to you, maybe code two is better.

# Now we are going to look mostly at time complexity.

# But space complexity will come up in interviews.

# They'll ask you, okay, you wrote this code in such a way, maybe during a whiteboard session or something

# like that, and they say, great, you've optimized for time complexity.

# But what if we care about space complexity more?

# How would you write it?

# Then you will have to understand both concepts.

# But what we will focus on mostly in this course will be time complexity.

# 