INTRO (SELF)
Hello! My name is Braden and I'm here to talk to you about some computer science.

CUT TO SLIDES

So today I hope to teach about three topics. First we will get straight what a binary search tree is, then talk about what breadth-first search is, how it functions, and how it compares to the similar system called depth-first-search.
And ultimately I want you to understand what differs depth first search and breadth first search from each other.//

1.
So here is a binary search tree. Treat this as a visualization of digital data as we will be working with this graphic throughout the lesson. These circles will represent individual data out of a collection that we have to search through using these little lines we have. These letters themselves we will view as placeholder values to signify that they are data.
But when you look at this binary search tree, I bet you see a collection of circles with letters in them. Automatically, your brain might try to make sense of this and sort it. When you see the alphabet you think "Okay, there's an A, there's a B, there's C" We have our own system for understanding this graphic because of the nature of what the placeholders mean to us as humans. But let's put that aside. Remember these circles can be anything, just as if we were a computer system.

CUT TO SELF

So how can we tell a computer what order to process the entries of this data, and when it **should** process any given entry? And this is important because the order it chooses to go through our alphabetical entries will vary the efficiency of our search. Which is ultimately the goal of this exercise. We want to search through that tree as effectively as we can.

Our binary search tree showed the circles with letters of the alphabet. For my demonstration today I will be highlighting to you the order that the different searching algorithms produce so you can keep track of what is going on. For example, the final product of this system might look like this. It will make more sense in a moment.

2./

/
We are going to walk through the binary search tree and solve it as if we were the computer using these different search algorithms so you can see how they differ.

We will abide by these four rules of the binary search tree so we can consistently get the same /structure to our result.

I am going to explain breadth first search to begin with. Breadth-first-search will try to navigate the tree by always viewing every viable option that we could possibly do.

Because of rule 1, we need to start at G, and by starting at G we have marked it. Or perhaps you could say we have “processed it,” and it has been resolved. From G, what are our options? We have lines connecting us to E, B, and A. So we have multiple options, and because of rule 3 of the binary search tree, we will prioritize alphabetical order and choose A.

So now we have marked both G, and A. This means E and B are both still options, since we have G, and now D is an option since we have A. Again, rule 3, so we choose B. Now in alphabetical order I wish I could choose C, but in the rules of the tree I cannot connect to it since I have not connected to S or E. So instead we have E, F, and D. Let’s choose D, which opens up S. S is pretty far down the prioritization order, so let’s choose E.

After that, our valid options are C, S, and F - we choose C, and then F, and finally S. Great, now we have ended up with an output that reads G-A-B-D-E-C-F-S. I bring up this output, because we can use this string of letters to help us understand how it differs from our outcome once we complete a depth-first search. Let’s get to that!

In a depth-first-search, we will first seek to reach the bottom of the graphic before we go back up and vary out the breadth of our search. Let me show you what I mean.

Again, we start at G, and we have the same choice of E, B, or A. We still choose A all the same. But with depth-first-search, we have to only operate out of the piece we just marked until we have nowhere to go. That means we are at A, and we only have one choice: D. From D, we have to go to S, since we cannot go back and ponder the choices of G. From S, this is a different story. We now have two options again. We can choose C, or F. Alphabetical order says C, and then from C we must choose E.

Well now here’s a conundrum, we are on E, and the only connecting circles are ones we have marked. This means we have to use rule 4, and return back to C, and then back to S. back at S, we once once again have a valid circle! So we go to F, and then to B. And now, we have completed our search, but the order that it was done differs from the order of the breadth-first search.

These two systems created different outputs of their searching because of the manner in which they search, this all being in the name. The “depth” of depth-first search has it intentionally exclude connections next to it in favor of finding harder-dug entries. Breadth-first search scans the entries most apparent to ensure they are found quickly. This means you will want to use them in different situations based on the needs of the problem at hand.

CUT TO SELF

Today I hope you learned the basic principles of traversing a binary-search-tree and the fundamental difference between two core searching systems. Shifting the order you navigate the tree in by changing the searching system you use can make for a more effective search, but you have to know what you’re looking for. Thank you for watching my video, and get searching!