

YouTube Summarizer

Transcript:

Hey guys Greg here, let's solve path sum, Lico number 112. Now we're given the root of a binary tree and another integer called target sum. Now we need to return true. If the tree has a root to leaf path such that adding up all of the values along the path equals the target sum. Okay, so in this example here, we have this tree and our target sum is 22. So if you go five plus four is nine, plus 11 is 20, plus two is 22, particularly from the root all the way to the leaf exactly. The sum is precisely 22 and so we just need to return true because there exists some path like that. It could have been like down here or down here as long as it's somewhere from the root including all of those values precisely to the bottom, then you're good to go. And this example here, the target sum is five. This is a path sum of four. This is a path sum of three. A path sum of that target sum does not exist so we'd return false. Okay, I'd have to say from a visualization standpoint, this is going to be pretty straightforward and what you'd expect. We're going to do a DFS and we're going to keep a current sum. So we're going to initialize that to zero and we're going to see the first node which is the root. Our current sum is going to go up by that node's value which is five. We're going to look over to the left and so our current sum goes up by four to make it nine. We'd go over to the left. It goes up 11 to make it 20. We'd go over to the left and we'd hit a leaf node with a current sum of 27. Since we see this is a leaf node and we could confirm that by making sure that it doesn't have a left and that it doesn't have a right. Since that is true, we could see, okay, does our current sum match our target? No, it does not. So we just need to go back up here. And so at this level here, our sum is going to be 20. So then over here, we would increase it by two and so we have a current sum of 22. We could again say, okay, is this node a leaf? Yes it is because it has no left and no right. Does our current sum match that target? Yes it does. Okay, so overall we would return true here. So we would just do a pretty standard DFS traversal which looks something like this. And every time you hit a leaf, which is defined by not having a left and a right, you would see if our current sum matches if it does you'd return true. And if it doesn't, you'd kind of locally return false. You don't necessarily return false overall. But if you ever find a path, you want to overall return true and kind of in these micro decisions of when it's wrong, you kind of want to return false. So we'll work those details out in the code. That's the idea. Okay, so we'll get a helper function, which I'll just call has some and it's going to take a root as well as the current sum associated at that level. So if it's not a root, well, then clearly we went too far. So in this case, we're just going to return false. Otherwise, we're at a valid node and the current sum that we have should go up by the current node's value. Now, is it a leaf node? So if we don't have a root dot left and we don't have a root dot right, that's the definition of a leaf node. So we need to return the result of Cur sum is equal equal to the target sum. So you'd want to return true here if they match and if they don't match, you'd want to return false. If we get over here, that means you were not a leaf node. So we have a chance of finding it. We need to see if you can find it on your left side or your right side. So we'll return that we can find a sum on the root dot left with the same sum that we currently have or if we can find that sum on the right with again, that current sum. And that really is the code there. You would just want to start that from the initial root. So we will return that we have a sum on the root with a current sum of zero. This overall will tell us if we found it anywhere because of that or and if we found it literally nowhere then it would return false. So this is going to have a time complexity of big of n where n is the number of nodes in the tree because we just do a pretty standard DFS. And it's going to have a space complexity of the height of the tree due to the recursive call stack. And so that's going to be big of h or in the worst case, that'll also be big of n. Drop a like, it's helpful guys. I hope it was and have a great day. Bye bye.

Summary

Lico number 112: We're given the root of a binary tree and another integer called target sum . Now we need to return true if the tree has a root to leaf path such that adding up all of the values along the path equals the target sum. So in this example here, our target sum is 22. So if you go five plus four is nine, plus 11 is 20, plus two is 22, particularly from the root all the way to the leaf exactly. A path sum of that target sum does not exist so we'd return false.