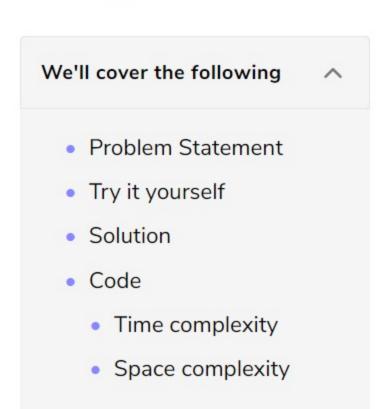


Q Search Course

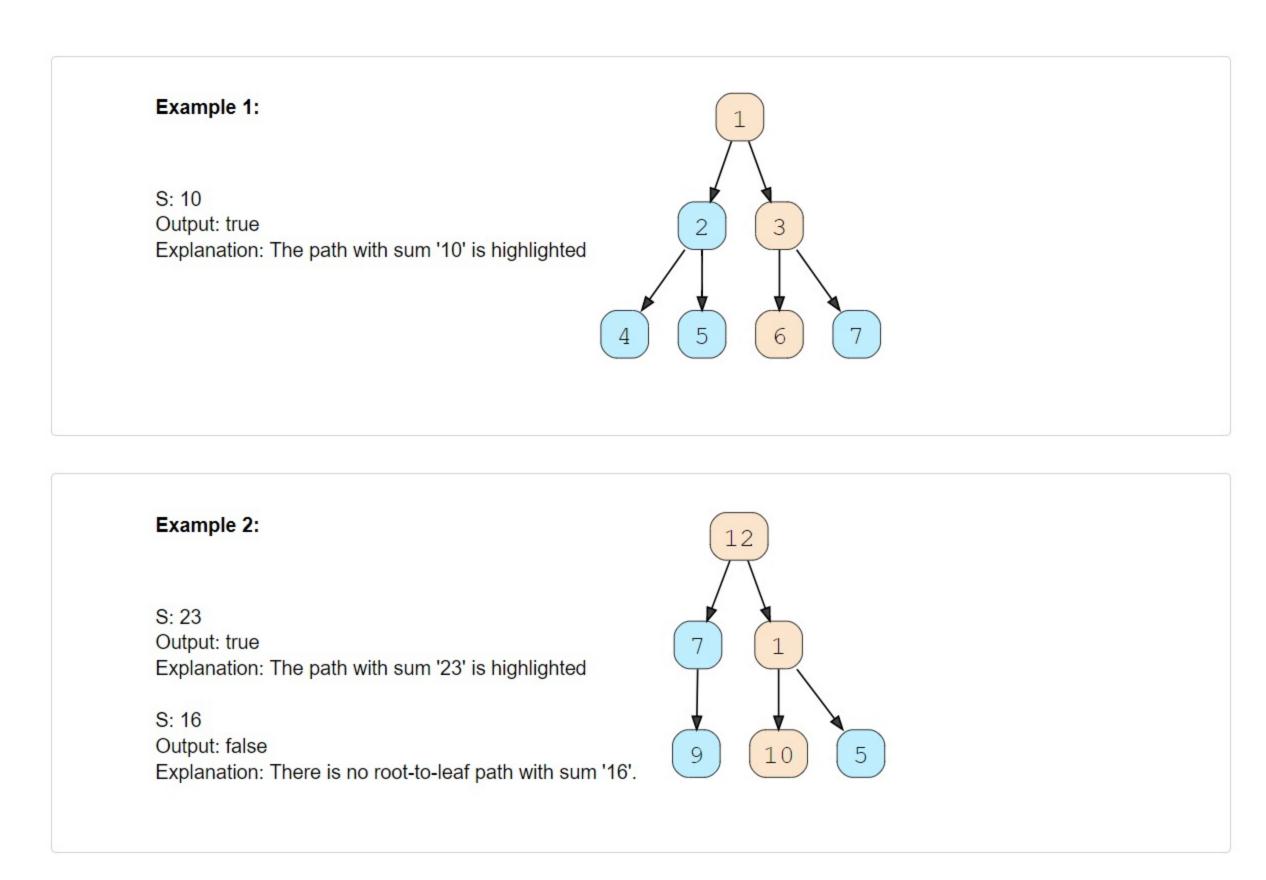
Binary Tree Path Sum (easy)



Problem Statement

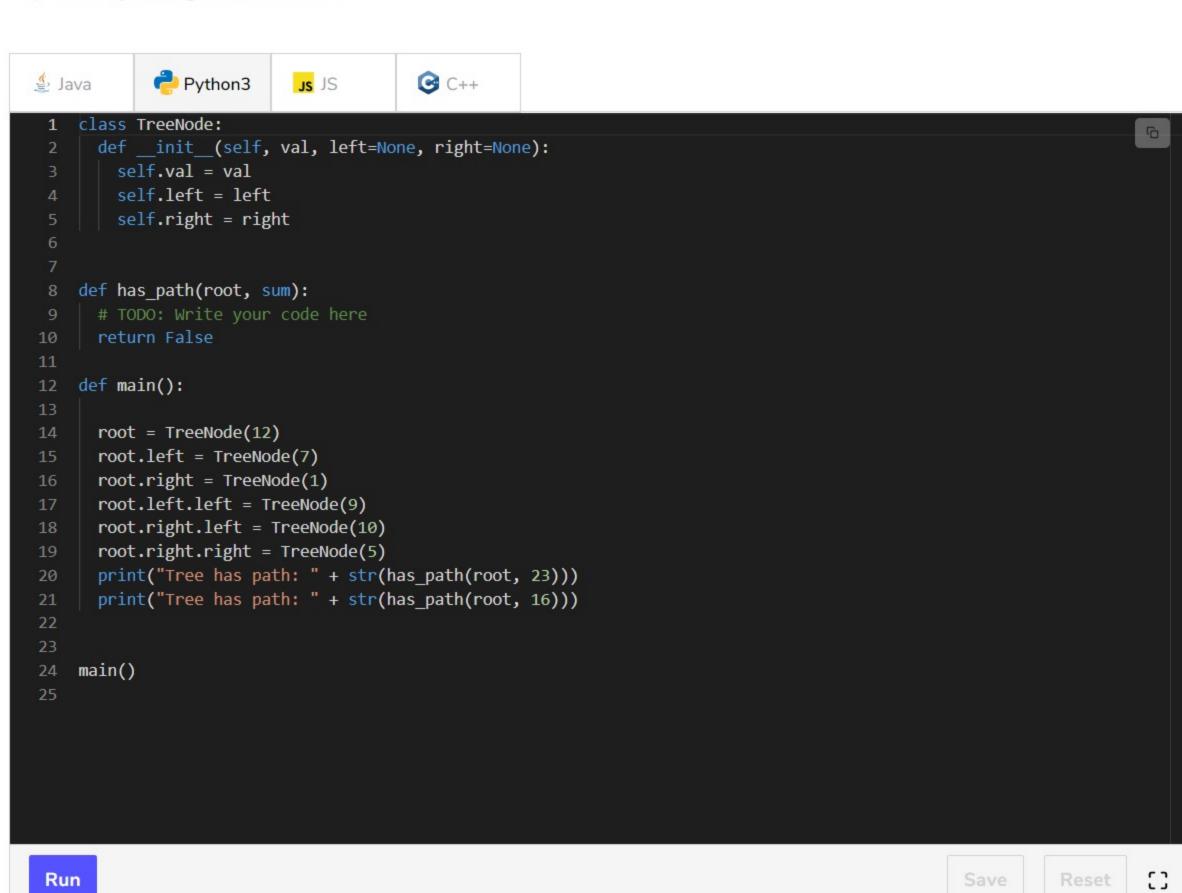
Given a binary tree and a number 'S', find if the tree has a path from root-to-leaf such that the sum of all the node values of that path equals 'S'.

₿



Try it yourself

Try solving this question here:



Solution

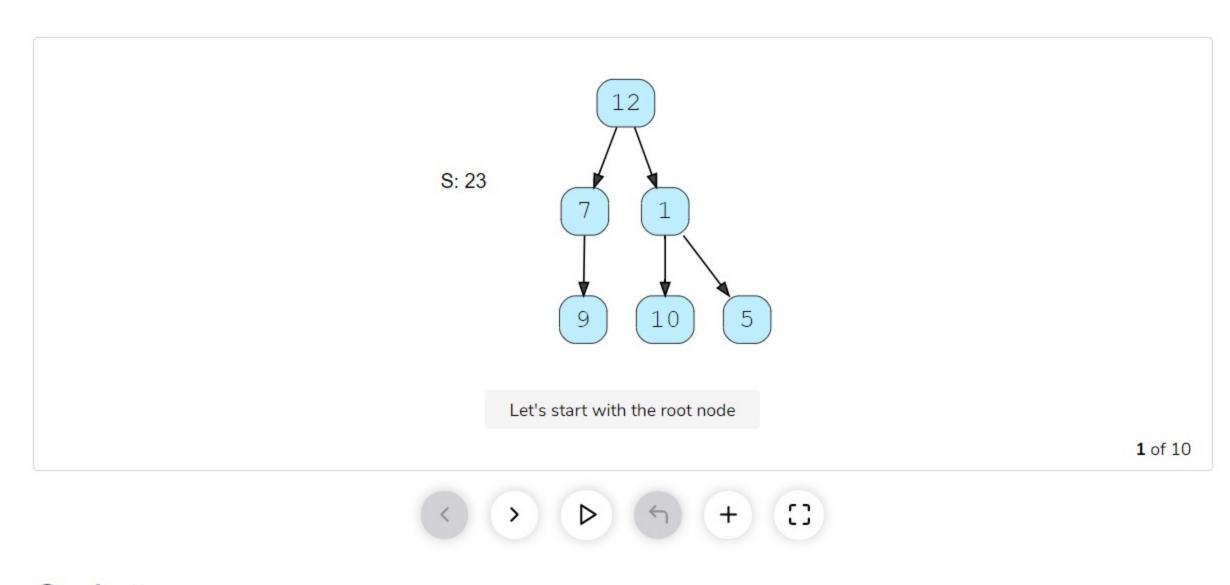
As we are trying to search for a root-to-leaf path, we can use the **Depth First Search (DFS)** technique to solve this problem.

To recursively traverse a binary tree in a DFS fashion, we can start from the root and at every step, make two recursive calls one for the left and one for the right child.

Here are the steps for our Binary Tree Path Sum problem:

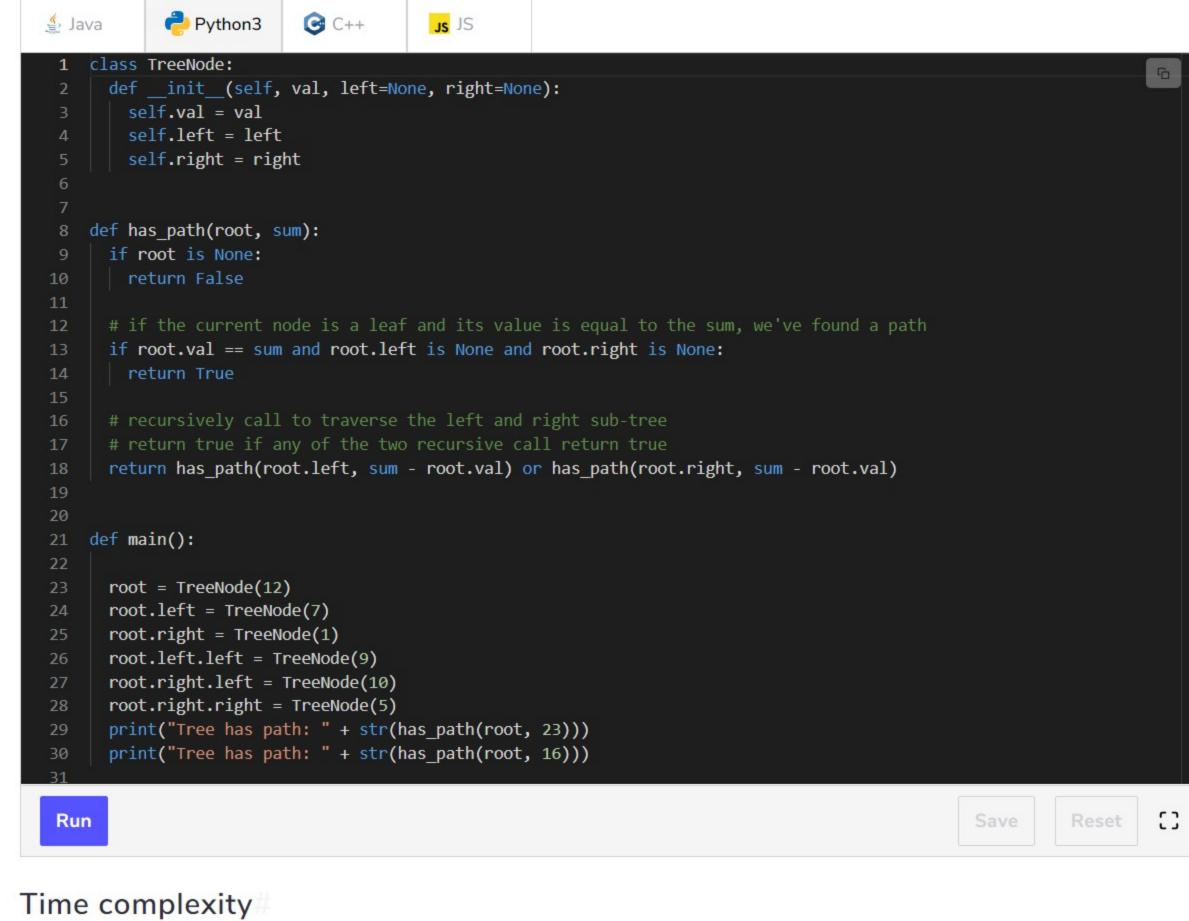
- 1. Start DFS with the root of the tree.
- 2. If the current node is not a leaf node, do two things:
 - Subtract the value of the current node from the given number to get a new sum => s = s node.value
- Make two recursive calls for both the children of the current node with the new number calculated in the previous step.
 3. At every step, see if the current node being visited is a leaf node and if its value is equal to the given
- number 'S'. If both these conditions are true, we have found the required root-to-leaf path, therefore return true.4. If the current node is a leaf but its value is not equal to the given number 'S', return false.

Let's take the example-2 mentioned above to visually see our algorithm:



Code

Here is what our algorithm will look like:



The time complexity of the above algorithm is O(N), where 'N' is the total number of nodes in the tree. This

is due to the fact that we traverse each node once.

Space complexity

The space complexity of the above algorithm will be O(N) in the worst case. This space will be used to store the recursion stack. The worst case will happen when the given tree is a linked list (i.e., every node has only one child).

