



Master algorithms together on [Binary Search](#)! Create a room, invite your friends, and race to finish the problems.

---

Daily Coding Problem

[Blog](#)

---

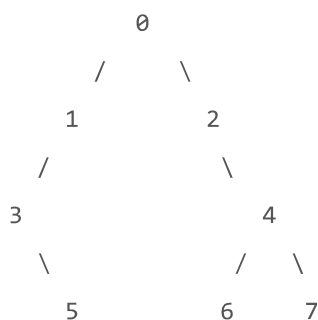
## Daily Coding Problem #254

### Problem

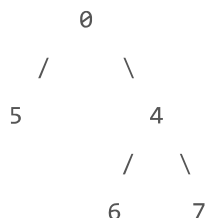
This problem was asked by Yahoo.

Recall that a full binary tree is one in which each node is either a leaf node, or has two children. Given a binary tree, convert it to a full one by removing nodes with only one child.

For example, given the following tree:



You should convert it to:



## Solution

To start out, notice that it will be easier to work from the bottom of the tree upwards than the reverse. For example, we can delete a leaf node without having any impact on the rest of the tree, but deleting or replacing a higher-up node will be more complicated.

When must we delete a node? This will only be the case when the node has exactly one child. If the node is a leaf node, or has two children, we don't need to make any changes.

With these two facts in mind, our solution will be to perform a post-order traversal, working our way from the bottom to the top of the tree, and replacing a node with its child if only one child exists.

A recursive implementation is shown below.

```
def convert(root):
    if not root:
        return None

    root.left = convert(root.left)
    root.right = convert(root.right)

    if not root.left and not root.right:
        return root

    if root.left and root.right:
        return root

    if not root.left:
        root = root.right

    else:
        root = root.left
```

Since we visit each node once, this will take  $O(N)$  time. The space complexity will be how far down we recur, which is equal to the height of the tree.

© Daily Coding Problem 2019

[Privacy Policy](#)

[Terms of Service](#)

[Press](#)