Daily Coding Problem #146

Problem

This question was asked by BufferBox.

Given a binary tree where all nodes are either `0` or `1`, prune the tree so that subtrees containing all `0`s are removed.

For example, given the following tree:

```
   0
  / \
 1   0
    / \
   1   0
  / \
 0   0
```

should be pruned to:

```
   0
  / \
 1   0
    /
   1
```

We do not remove the tree at the root or its left child because it still has a `1` as a descendant.

Solution

If we think about `prune` recursively and assume that it works, then we can run `prune` on our input tree's `left` and `right` child and it should get rid of all subtrees that are wholly `0`s. By that logic, if there is still a `left` or `right` child then this the current tree cannot be wholly `0`s. So the only remaining cases are:

- When the root itself is null, just return null
- When the root's value is 0 and it is a leaf, then it should return null

Remember to prune the tree first before checking its subtrees.

```
def prune(root):
    if root is None:
        return root

    root.left, root.right = prune(root.left), prune(root.right)

    if root.left is None and root.right is None and root.val == 0:
        return None

    return root
```

This takes O(n) time and O(h) space, since we traverse the entire tree.