LeetCode 156

Description

Given a binary tree where all the right nodes are either leaf nodes with a sibling (a left node that shares the same parent node) or empty, flip it upside down and turn it into a tree where the original right nodes turned into left leaf nodes. Return the new root.

For example:
Given a binary tree {1,2,3,4,5},

```
  1
```

/ \
2 3
/ \
4 5

return the root of the binary tree [4,5,2,#,#,3,1].

4
/ \
5 2
/ \
3 1

Thought

By draw the tree, we can find that all we need to do is to:

1. find the new root
2. reset the pointer from new root to its left child and right child
3. call the operation recursively until there is no node or the left child of a node is null

Solution

```
public TreeNode upsideDownBinaryTree(TreeNode root) {
  //corner case
    if(root == null || root.left == null) {
```

```
            return root;
        }

        TreeNode newRoot = upsideDownBinaryTree(root.left);
        root.left.left = root.right;
        root.left.right = root;
        root.left = null;
        root.right = null;
        return newRoot;
    }
```

Takeaways

- Divide big problem into the same smaller problems and solve recursively