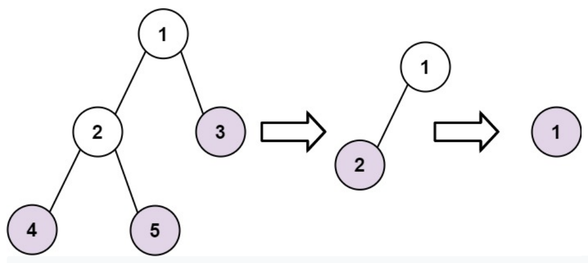
366. Find Leaves of Binary Tree

Medium 1748 30 Add to List Share

Given the `root` of a binary tree, collect a tree's nodes as if you were doing this:

- Collect all the leaf nodes.
- Remove all the leaf nodes.
- Repeat until the tree is empty.

Example 1:




```
Input: root = [1,2,3,4,5]
Output: [[4,5,3],[2],[1]]
Explanation:
[[3,5,4],[2],[1]] and [[3,4,5],[2],[1]] are also considered correct answers since
per each level it does not matter the order on which elements are returned.
```

Example 2:

```
Input: root = [1]
Output: [[1]]
```

Constraints:

- The number of nodes in the tree is in the range `[1, 100]`.
- `-100 <= Node.val <= 100`

Accepted 117,304 | Submissions 155,523

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Google | 23 LinkedIn | 13

Related Topics

Tree Depth-First Search Binary Tree

Java Autocomplete

```java
class Solution {
    private List<Pair<Integer, Integer>> pairs;

    private int getHeight(TreeNode root) {

        // return -1 for null nodes
        if (root == null) return -1;

        // first calculate the height of the left and right children
        int leftHeight = getHeight(root.left);
        int rightHeight = getHeight(root.right);

        // based on the height of the left and right children, obtain the height of the current (parent) node
        int currHeight = Math.max(leftHeight, rightHeight) + 1;

        // collect the pair -> (height, val)
        this.pairs.add(new Pair<Integer, Integer>(currHeight, root.val));

        // return the height of the current node
        return currHeight;
    }

    public List<List<Integer>> findLeaves(TreeNode root) {
        this.pairs = new ArrayList<>();

        getHeight(root);

        // sort all the (height, val) pairs
        Collections.sort(this.pairs, Comparator.comparing(p -> p.getKey()));

        int n = this.pairs.size(), height = 0, i = 0;

        List<List<Integer>> solution = new ArrayList<>();

        while (i < n) {
            List<Integer> nums = new ArrayList<>();
            while (i < n && this.pairs.get(i).getKey() == height) {
                nums.add(this.pairs.get(i).getValue());
                i++;
            }
            solution.add(nums);
            height++;
        }
        return solution;
    }
}
```