Description Solution Discuss (565) Submissions

545. Boundary of Binary Tree

Medium 1223 1958 Add to List Share

The **boundary** of a binary tree is the concatenation of the **root**, the **left boundary**, the **leaves** ordered from left-to-right, and the **reverse order** of the **right boundary**.

The **left boundary** is the set of nodes defined by the following:

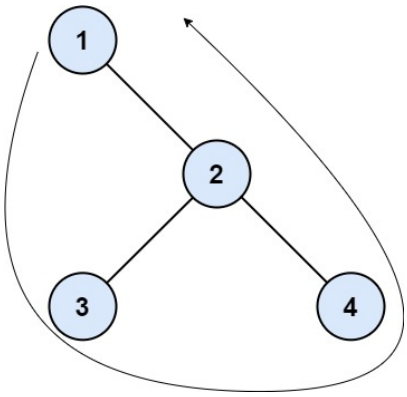
- The root node's left child is in the left boundary. If the root does not have a left child, then the left boundary is **empty**.
- If a node in the left boundary and has a left child, then the left child is in the left boundary.
- If a node is in the left boundary, has **no** left child, but has a right child, then the right child is in the left boundary.
- The leftmost leaf is **not** in the left boundary.

The **right boundary** is similar to the **left boundary**, except it is the right side of the root's right subtree. Again, the leaf is **not** part of the **right boundary**, and the **right boundary** is empty if the root does not have a right child.

The **leaves** are nodes that do not have any children. For this problem, the root is **not** a leaf.

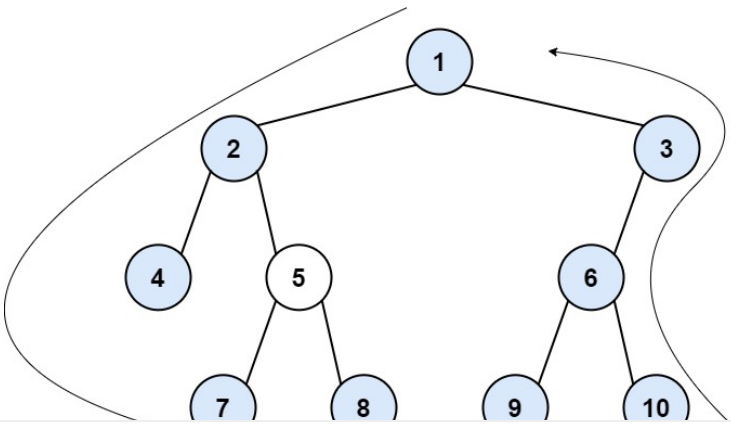
Given the `root` of a binary tree, return *the values of its* ***boundary***.

Example 1:




```
Input: root = [1,null,2,3,4]
Output: [1,3,4,2]
Explanation:
- The left boundary is empty because the root does not have a left child.
- The right boundary follows the path starting from the root's right child 2 -> 4.
  4 is a leaf, so the right boundary is [2].
- The leaves from left to right are [3,4].
Concatenating everything results in [1] + [] + [3,4] + [2] = [1,3,4,2].
```

Example 2:



C++ Autocomplete

```
26
        res.push_back(root->val);
27
        boundaryLeft(root->right,res);
28          }
29      }
30
31      void boundaryRight(TreeNode* root,vector<int> &res)
32      {
33          if(!root)
34              return;
35
36          if(root->right)
37          {
38
        boundaryRight(root->right,res);
39
        res.push_back(root->val);
40          }
41          else if(root->left)
42          {
43
        boundaryRight(root->left,res);
44
        res.push_back(root->val);
45          }
46      }
47
48      void leaves(TreeNode* root,vector<int> &res)
49      {
50          if(!root)
51              return;
52
53          if(!root->left and !root->right)
54          {
55
        res.push_back(root->val);
56              return;
57          }
58
59          leaves(root->left,res);
60          leaves(root->right,res);
61      }
62
63
64      vector<int> boundaryOfBinaryTree(TreeNode * root) {
65          vector<int> res;
66          if(!root)
67              return res;
68
69          res.push_back(root->val);
70
71          boundaryLeft(root->left,res);
72          leaves(root-
```

NEW

Testcase Run Code Result Debugger

Accepted Runtime: 3 ms

Your input [1,null,2,3,4]

Output [1,3,4,2] Diff

Expected [1,3,4,2]

Problems Pick One Prev 545/2509 Next xample ases Run Code Submit