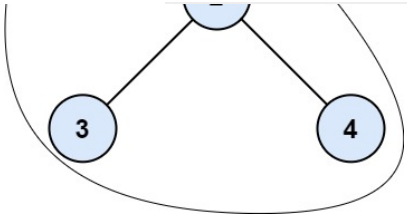
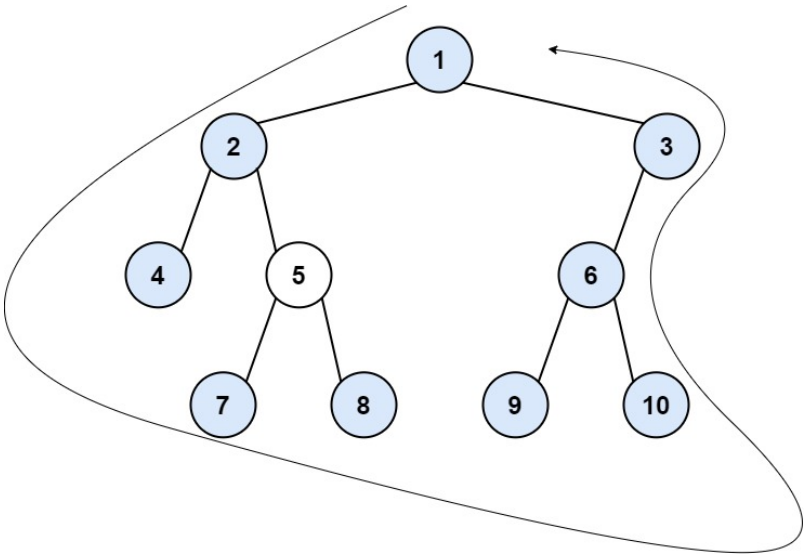
Description Solution Discuss (565) Submissions



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:



Input: root = [1,2,3,4,5,6,null,null,null,7,8,9,10]
Output: [1,2,4,7,8,9,10,6,3]
Explanation:
- The left boundary follows the path starting from the root's left child 2 -> 4.
  4 is a leaf, so the left boundary is [2].
- The right boundary follows the path starting from the root's right child 3 -> 6 -> 10.
  10 is a leaf, so the right boundary is [3,6], and in reverse order is [6,3].
- The leaves from left to right are [4,7,8,9,10].
Concatenating everything results in [1] + [2] + [4,7,8,9,10] + [6,3] = [1,2,4,7,8,9,10,6,3].

Constraints:

- The number of nodes in the tree is in the range $[1, 10^4]$.
- $-1000 <= Node.val <= 1000$

Accepted 116,992 Submissions 263,733

Seen this question in a real interview before? Yes No

Companies

Related Topics

Similar Questions

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