1261. Find Elements in a Contaminated Binary Tree

Medium | Topics | Companies | Hint

Given a binary tree with the following rules:

1. `root.val == 0`
2. For any `treeNode`:
   1. If `treeNode.val` has a value `x` and `treeNode.left != null`, then `treeNode.left.val == 2 * x + 1`
   2. If `treeNode.val` has a value `x` and `treeNode.right != null`, then `treeNode.right.val == 2 * x + 2`

Now the binary tree is contaminated, which means all `treeNode.val` have been changed to `-1`.

Implement the `FindElements` class:

- `FindElements(TreeNode* root)` Initializes the object with a contaminated binary tree and recovers it.
- `bool find(int target)` Returns `true` if the `target` value exists in the recovered binary tree.

Python3 | Auto

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class FindElements:

    def __init__(self, root: Optional[TreeNode]):



    def find(self, target: int) -> bool:


# Your FindElements object will be instantiated and called as such:
# obj = FindElements(root)
# param_1 = obj.find(target)
```

Ln 10, Col 9 | Saved

Testcase | Test Result

All Submissions

Accepted 31 / 31 testcases passed

AndrewC275 submitted at Feb 21, 2025 09:31

Editorial | Solution

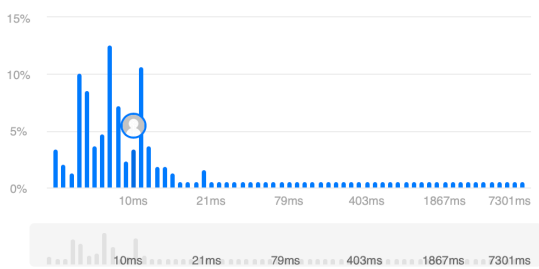
Runtime

10 ms | Beats **43.58%**

Analyze Complexity

Memory

21.83 MB | Beats **86.90%**



Code | Python3

```python
# Definition for a binary tree node.
```

Python3 | Auto

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class FindElements:

    def __init__(self, root: Optional[TreeNode]):
        self.val_set = set()
        self.recover(root, 0)

    def recover(self, node, v):
        if not node:
            return
        node.val = v
        self.val_set.add(v)
        self.recover(node.left, 2 * v + 1)
        self.recover(node.right, 2 * v + 2)

    def find(self, target: int) -> bool:
        if target in self.val_set:
            return True
        else:
            return False


# Your FindElements object will be instantiated and called as such:
# obj = FindElements(root)
# param_1 = obj.find(target)
```

Summary when to use self.

1. accessing instance variables stored in self.
2. modifying instance variables, updating attributes of objects
3. calling other instance methods inside a method