Problem 919: Complete Binary Tree Inserter

Problem Information

Difficulty: Medium
Acceptance Rate: 64.84%
Paid Only: No
Tags: Tree, Breadth-First Search, Design, Binary Tree

Problem Description

A **complete binary tree** is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

Design an algorithm to insert a new node to a complete binary tree keeping it complete after the insertion.

Implement the `CBTInserter` class:

* `CBTInserter(TreeNode root)` Initializes the data structure with the `root` of the complete binary tree. * `int insert(int v)` Inserts a `TreeNode` into the tree with value `Node.val == val` so that the tree remains complete, and returns the value of the parent of the inserted `TreeNode`. * `TreeNode get_root()` Returns the root node of the tree.

Example 1:

Input ["CBTInserter", "insert", "insert", "get_root"] [[[1, 2]], [3], [4], []] **Output** [null, 1, 2, [1, 2, 3, 4]] **Explanation** CBTInserter cBTInserter = new CBTInserter([1, 2]); cBTInserter.insert(3); // return 1 cBTInserter.insert(4); // return 2 cBTInserter.get_root(); // return [1, 2, 3, 4]

Constraints:

* The number of nodes in the tree will be in the range `[1, 1000]`. * `0 <= Node.val <= 5000` * `root` is a complete binary tree. * `0 <= val <= 5000` * At most `104` calls will be made to

`insert` and `get_root`.

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class CBTInserter {
public:
CBTInserter(TreeNode* root) {

}

int insert(int val) {

}

TreeNode* get_root() {

}
};

/**
* Your CBTInserter object will be instantiated and called as such:
* CBTInserter* obj = new CBTInserter(root);
* int param_1 = obj->insert(val);
* TreeNode* param_2 = obj->get_root();
*/
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class CBTInserter {

public CBTInserter(TreeNode root) {

}

public int insert(int val) {

}

public TreeNode get_root() {

}
}

/**
* Your CBTInserter object will be instantiated and called as such:
* CBTInserter obj = new CBTInserter(root);
* int param_1 = obj.insert(val);
* TreeNode param_2 = obj.get_root();
*/
```

Python3:

```
# 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 CBTInserter:

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


def insert(self, val: int) -> int:


def get_root(self) -> Optional[TreeNode]:



# Your CBTInserter object will be instantiated and called as such:
# obj = CBTInserter(root)
# param_1 = obj.insert(val)
# param_2 = obj.get_root()
```