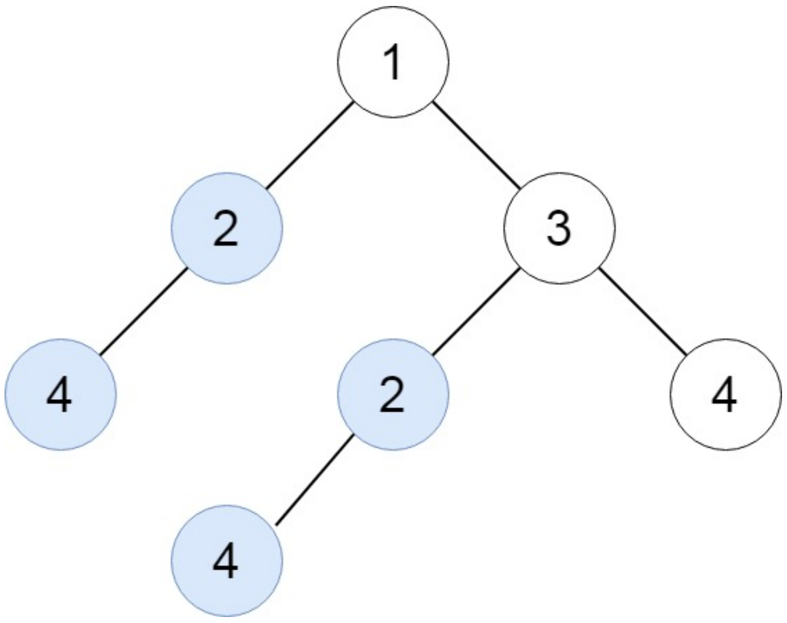
# 652. Find Duplicate Subtrees

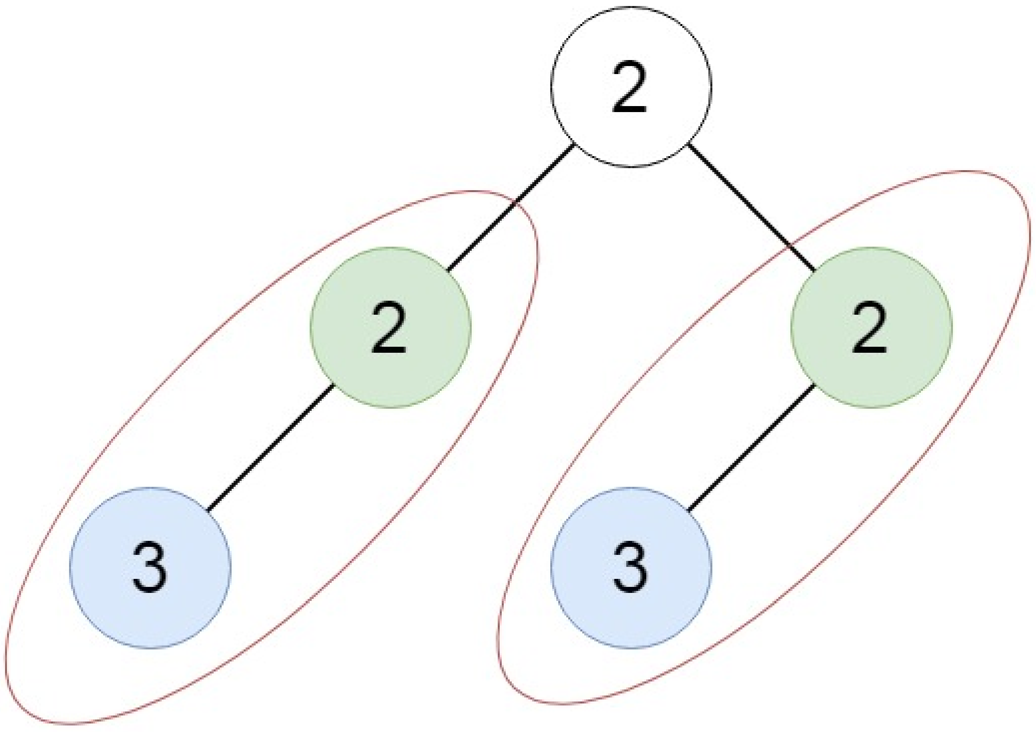
Given the `root` of a binary tree, return all **duplicate subtrees**.

For each kind of duplicate subtrees, you only need to return the root node of any **one** of them.

Two trees are **duplicate** if they have the **same structure** with the **same node values**.



**Input:** root = [1,2,3,4,null,2,4,null,null,4]
**Output:** [[2,4],[4]]



**Input:** root = [2,2,2,3,null,3,null]
**Output:** [[2,3],[3]]

```
def findDuplicateSubtrees(self, root: TreeNode) -> List[TreeNode]:
        res = {}
        ans = []
        self.helper(root,ans,res)
        return ans


    def helper(self,root,ans,res):
        if root is None:
            return '#'
        lt = self.helper(root.left,ans,res)
        rt = self.helper(root.right,ans,res)
        temp = str(root.val)+ ","+ lt+ "," +rt
        res[temp] = res.get(temp,0)+1
        if res[temp]==2:
            ans.append(root)
        return temp
```

**Approach #1: Depth-First Search [Accepted]**

**Intuition**

We can serialize each subtree. For example, the tree

```
 1
  / \
 2    3
    / \
   4    5
```

can be represented as the serialization `1,2,#,#,3,4,#,#,5,#,#`, which is a unique representation of the tree.

**Algorithm**

Perform a depth-first search, where the recursive function returns the serialization of the tree. At each node, record the result in a map, and analyze the map after to determine duplicate sub-trees.