1110. Delete Nodes And Return Forest

Solved

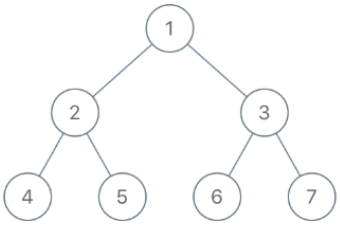
Medium | Topics | Companies

Given the `root` of a binary tree, each node in the tree has a distinct value.

After deleting all nodes with a value in `to_delete`, we are left with a forest (a disjoint union of trees).

Return the roots of the trees in the remaining forest. You may return the result in any order.

Example 1:




```
Input: root = [1,2,3,4,5,6,7], to_delete = [3,5]
Output: [[1,2,null,4],[6],[7]]
```

Example 2:

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

```
/**
* Definition for a binary tree node.
* public class TreeNode {
*     public var val: Int
*     public var left: TreeNode?
*     public var right: TreeNode?
*     public init() { self.val = 0; self.left = nil; self.right =
nil; }
*     public init(_ val: Int) { self.val = val; self.left = nil;
self.right = nil; }
*     public init(_ val: Int, _ left: TreeNode?, _ right:
TreeNode?) {
*         self.val = val
*         self.left = left
*         self.right = right
*     }
* }
*/
class Solution {
```

```
    func delNodes(_ root: TreeNode?, _ to_delete: [Int]) ->
[TreeNode?] {
        var ans = [TreeNode?]()
        var hash = Set(to_delete)
        func dfs(_ root: TreeNode?) {
            guard let root = root else {
                return
            }
            dfs(root.left)
            dfs(root.right)
            if(hash.contains(root.val)){
                if(root.left != nil ) {
                    ans.append(root.left)
                }
                if(root.right != nil) {
                    ans.append(root.right)
                }
                root.left = nil
                root.right = nil
            }
            if(root.left != nil && hash.contains(root.left!.val)) {
                root.left = nil
            }
            if(root.right != nil && hash.contains(root.right!.val))
{
                root.right = nil
            }
        }
        if(root != nil && !hash.contains(root!.val)){
            ans.append(root!)
        }

        dfs(root)

        var ans2:[TreeNode?] = []
        for i in ans where !hash.contains(i!.val) {
            ans2.append(i)
        }
        return ans2
    }
}
```