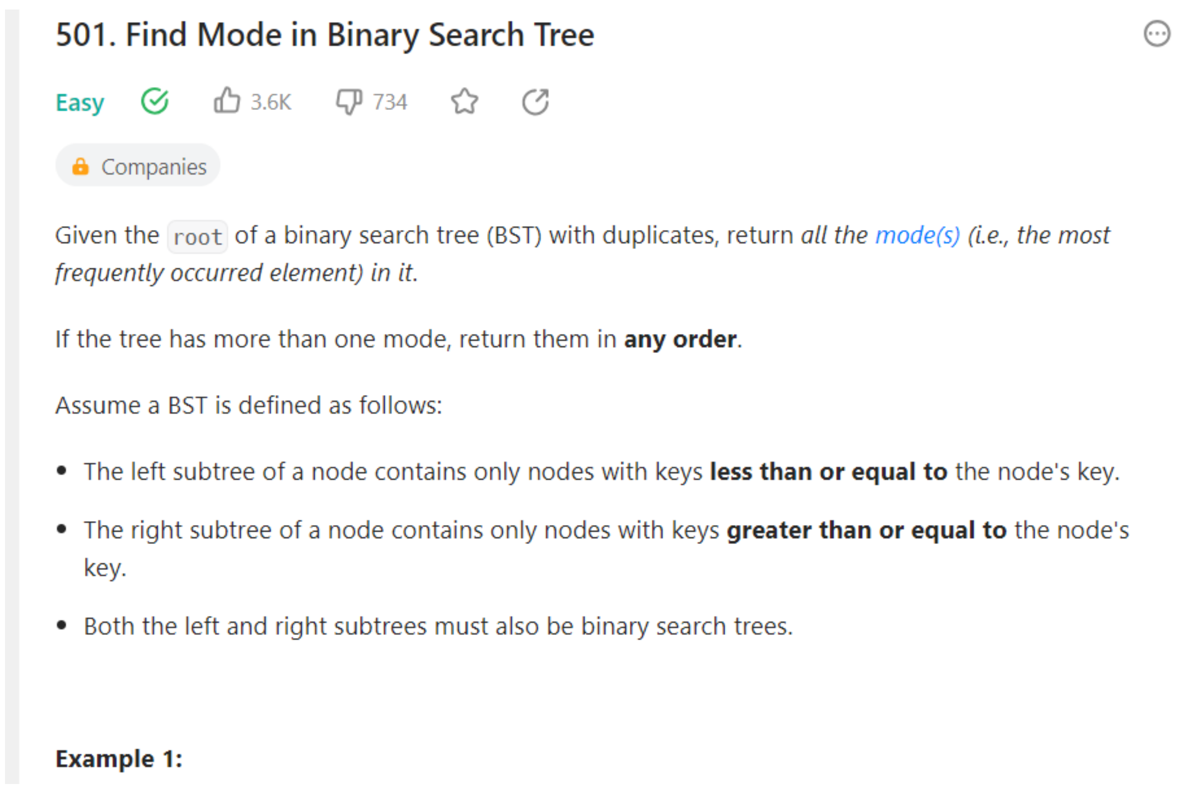

501. Find Mode in Binary Search Tree

Given the `root` of a binary search tree (BST) with duplicates, return *all the mode(s) (i.e., the most frequently occurred element) in it.*

If the tree has more than one mode, return them in **any order**.

Assume a BST is defined as follows:

- The left subtree of a node contains only nodes with keys **less than or equal to** the node's key.
- The right subtree of a node contains only nodes with keys **greater than or equal to** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:

Code

```
class Solution {
    var hash :[Int:Int] = [:]
    func findMode(_ root: TreeNode?) -> [Int] {
        guard let root = root else {
            return []
        }
        findocc(root)
        var maxval = 0
        if let maxValue = Array(hash.values).max() {
            maxval = maxValue
        } else {
            maxval = 0
        }
        var res:[Int] = []
        for (i,v) in hash {
            if(v == maxval) {
                res.append(i)
            }
        }
        return res
    }

    func findocc(_ root: TreeNode){
        if let temp = hash[root.val] {
            hash[root.val] = temp + 1
        } else  {
            hash[root.val] = 1
        }

        if let left = root.left {
            findocc(left)
        }
        if let right = root.right {
            findocc(right)
        }
    }

}
```