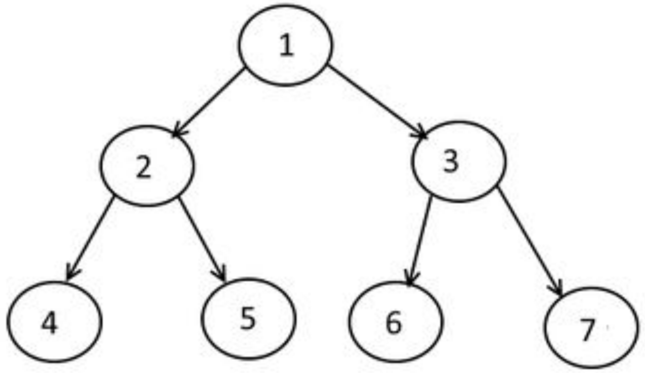
BINARY TREES



Inorder Traversal: 4 2 5 1 6 3 7

Preorder Traversal: 1 2 4 5 3 6 7

Postorder Traversal: 7 6 3 5 4 2 1

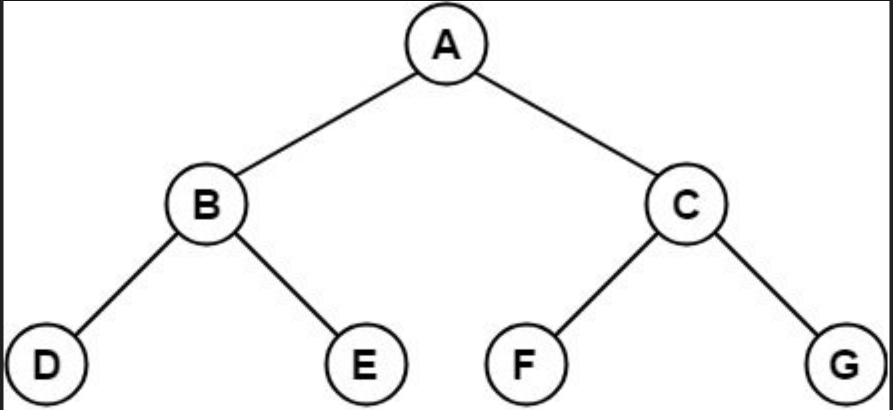
Breadth-First Search: 1 2 3 4 5 6 7

Depth-First Search: 1 2 4 5 3 6 7

Question: Given a binary tree, return the inorder traversal of its nodes' values.

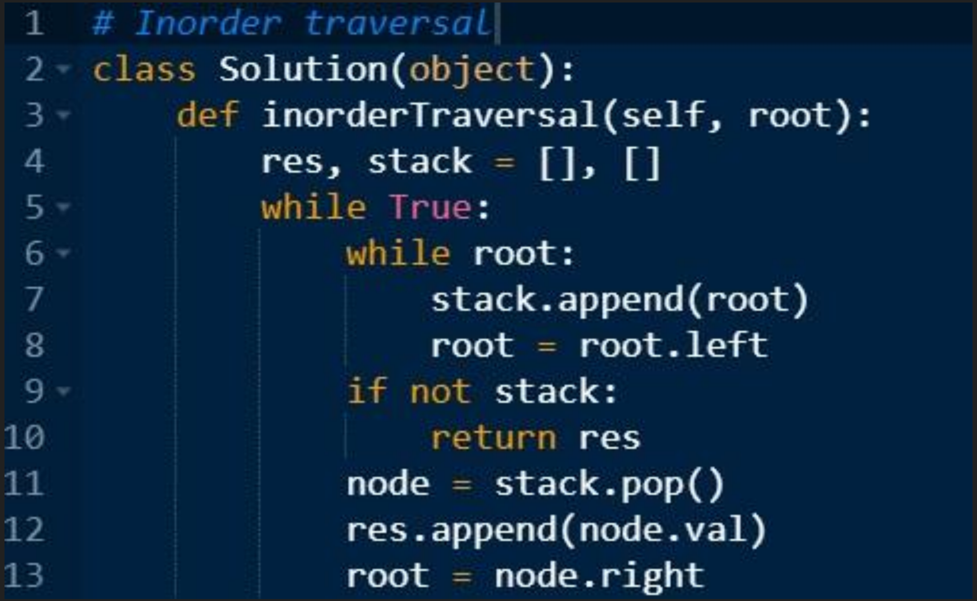
YOU HAVE 15 MINUTES

https://leetcode.com/problems/binary-tree-inorder-traversal/



Inorder Traversal : D , B , E , A , F , C , G

USE STACK AS A PRIMARY DATA STRUCTURE



```python
# Inorder traversal
class Solution(object):
    def inorderTraversal(self, root):
        res, stack = [], []
        while True:
            while root:
                stack.append(root)
                root = root.left
            if not stack:
                return res
            node = stack.pop()
            res.append(node.val)
            root = node.right
```

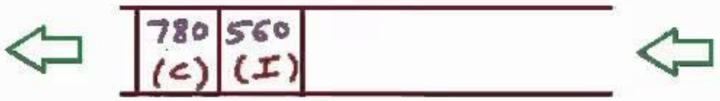
Question: Given a binary tree, return the level order traversal of its nodes' values. (ie, from left to right, level by level).

YOU HAVE 15 MINUTES

https://leetcode.com/problems/binary-tree-level-order-traversal/

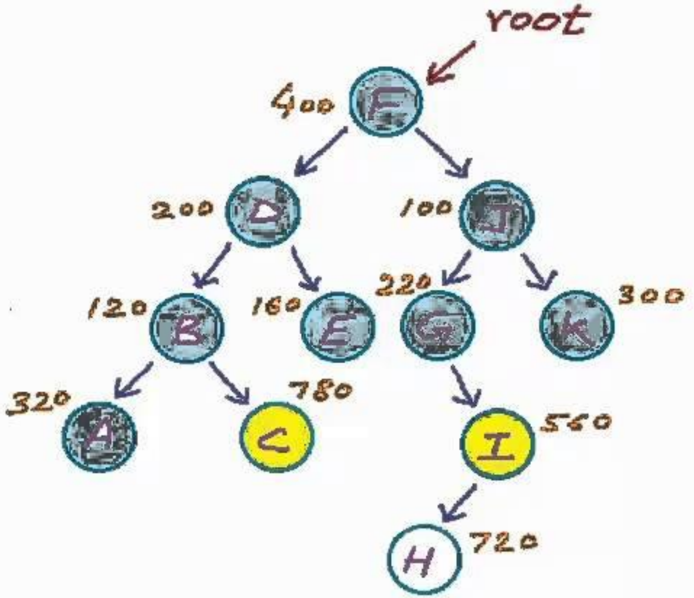
USE QUEUE AS A PRIMARY DATA STRUCTURE

Level-order Traversal

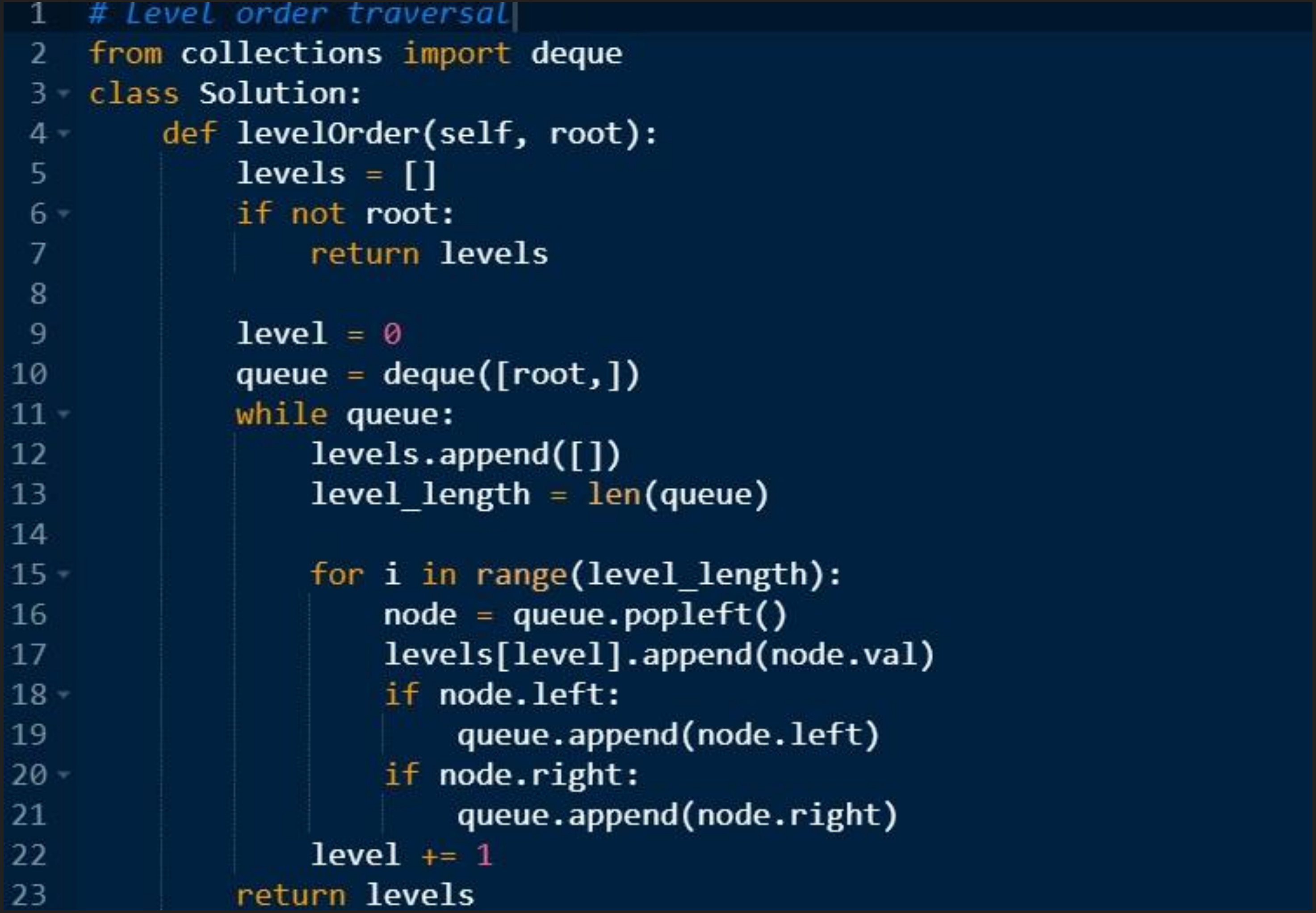


Queue (FIFO)

F, D, J, B, E, G, K, A, C



mycodeschool.com

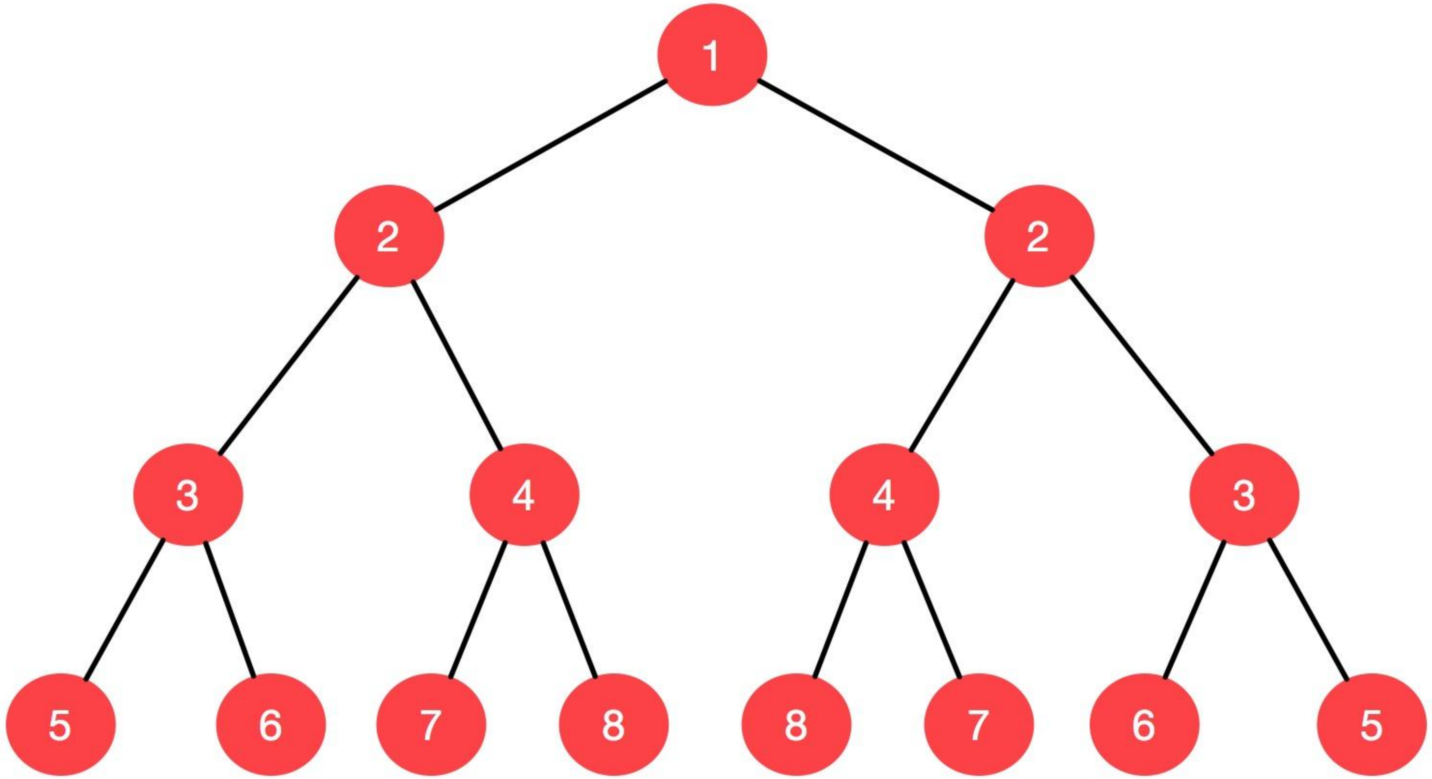


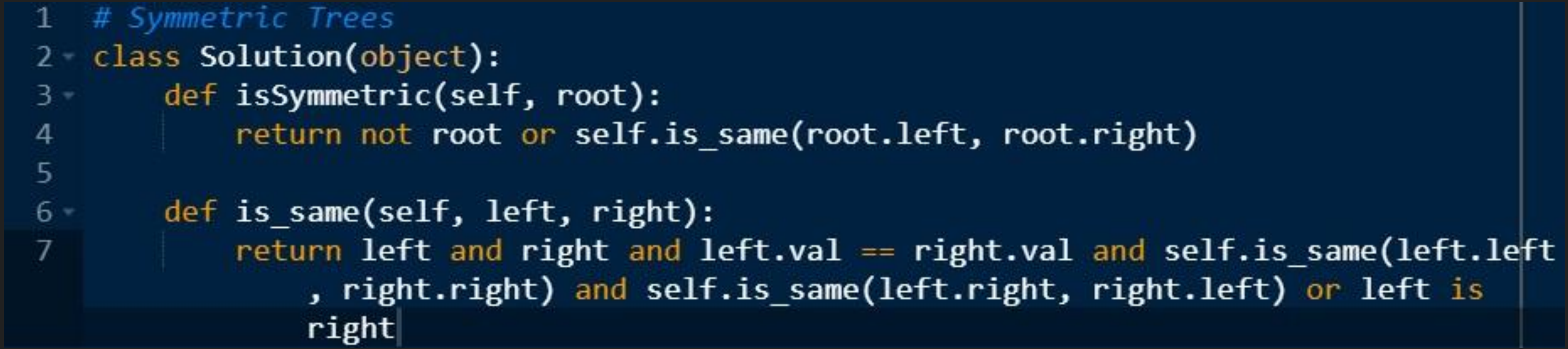
```python
# Level order traversal
from collections import deque
class Solution:
    def levelOrder(self, root):
        levels = []
        if not root:
            return levels

        level = 0
        queue = deque([root,])
        while queue:
            levels.append([])
            level_length = len(queue)

            for i in range(level_length):
                node = queue.popleft()
                levels[level].append(node.val)
                if node.left:
                    queue.append(node.left)
                if node.right:
                    queue.append(node.right)
            level += 1
        return levels
```

Question: Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

YOU HAVE 15 MINUTES

https://leetcode.com/problems/symmetric-tree/


1
2
2
3
4
4
3
5
6
7
8
8
7
6
5

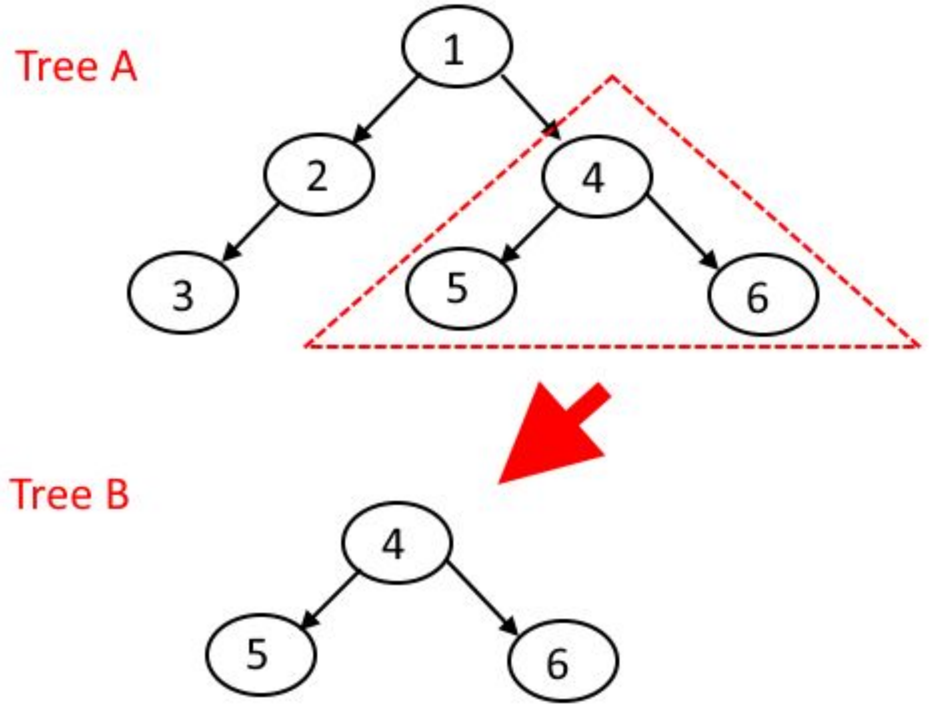


```python
# Symmetric Trees
class Solution(object):
    def isSymmetric(self, root):
        return not root or self.is_same(root.left, root.right)

    def is_same(self, left, right):
        return left and right and left.val == right.val and self.is_same(left.left
            , right.right) and self.is_same(left.right, right.left) or left is
            right
```

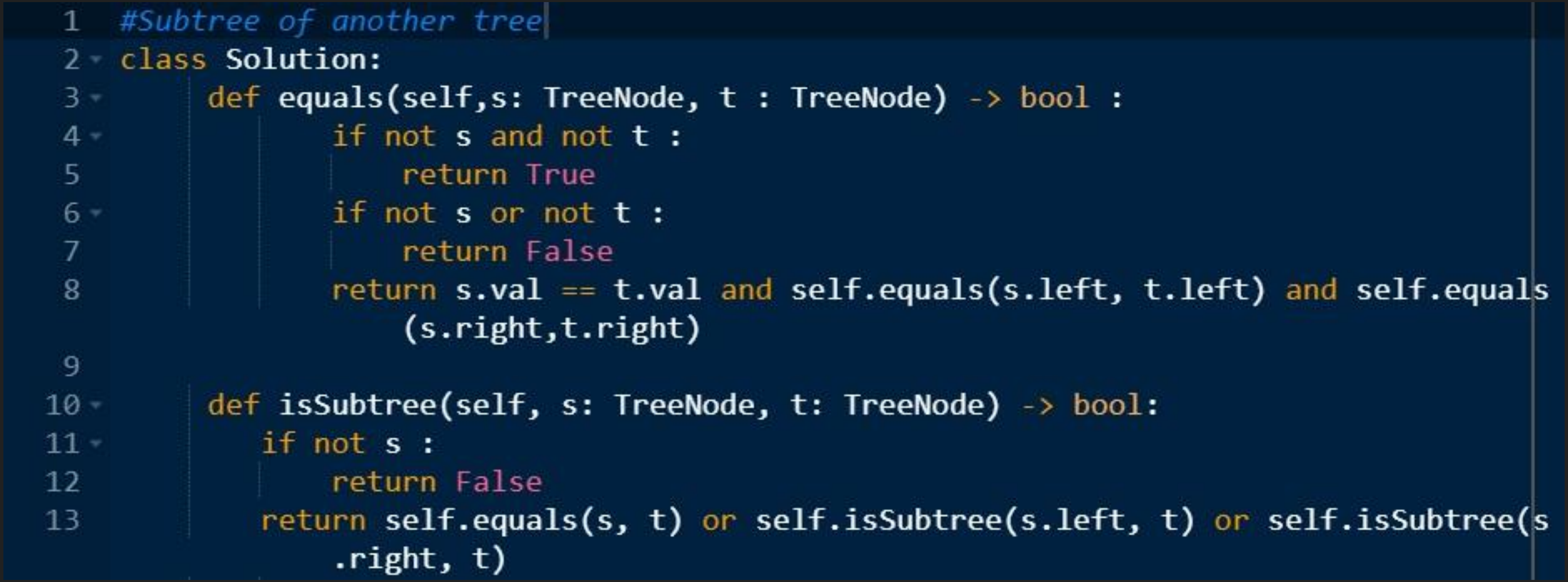
Question: Given two non-empty binary trees s and t, check whether tree t has exactly the same structure and node values with a subtree of s. A subtree of s is a tree consists of a node in s and all of this node's descendants. The tree s could also be considered as a subtree of itself.

YOU HAVE 15 MINUTES

https://leetcode.com/problems/subtree-of-another-tree/



Tree B is a subtree of Tree A



```python
#Subtree of another tree
class Solution:
        def equals(self,s: TreeNode, t : TreeNode) -> bool :
                if not s and not t :
                    return True
                if not s or not t :
                    return False
                return s.val == t.val and self.equals(s.left, t.left) and self.equals(s.right,t.right)

        def isSubtree(self, s: TreeNode, t: TreeNode) -> bool:
            if not s :
                return False
            return self.equals(s, t) or self.isSubtree(s.left, t) or self.isSubtree(s.right, t)
```

Question: Given a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you can see ordered from top to bottom.

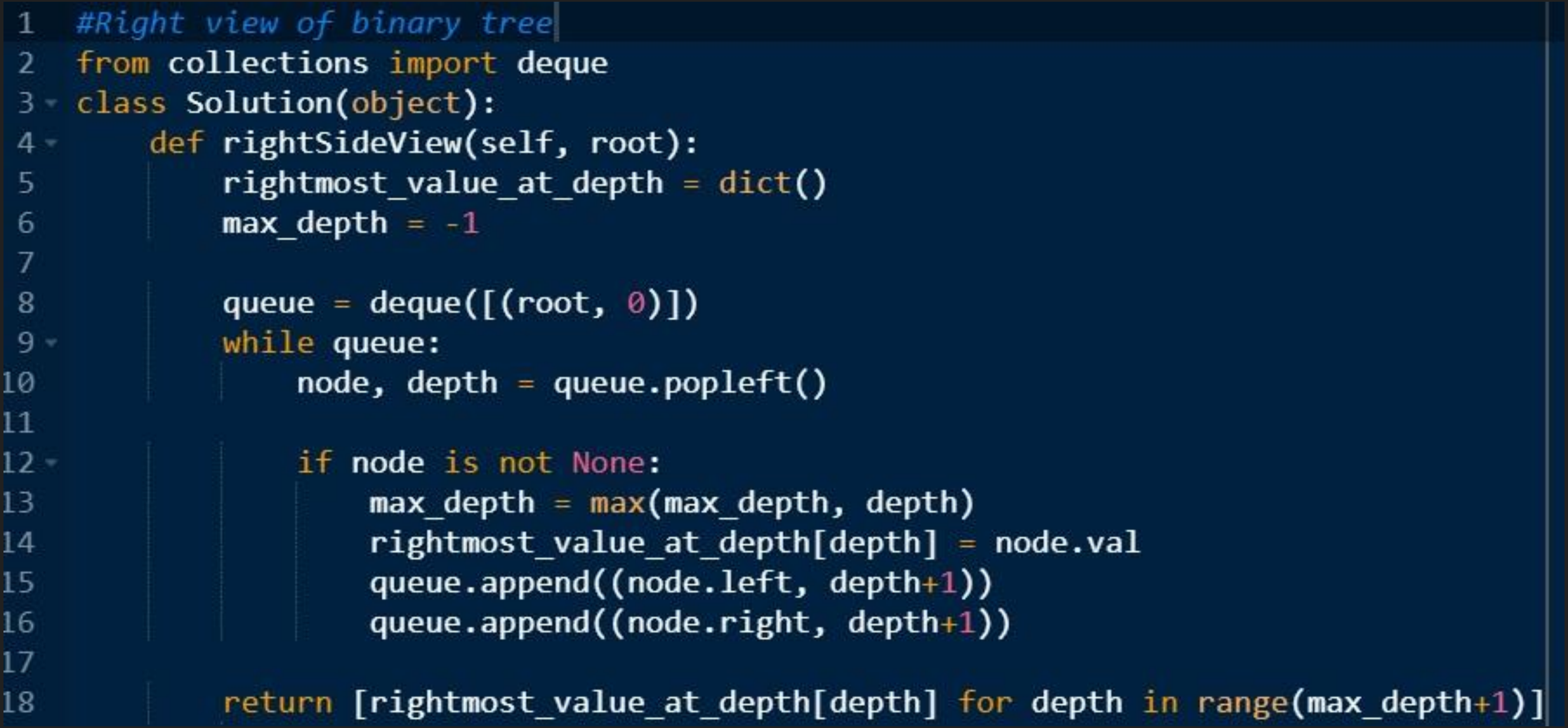
YOU HAVE 15 MINUTES

https://leetcode.com/problems/binary-tree-right-side-view/

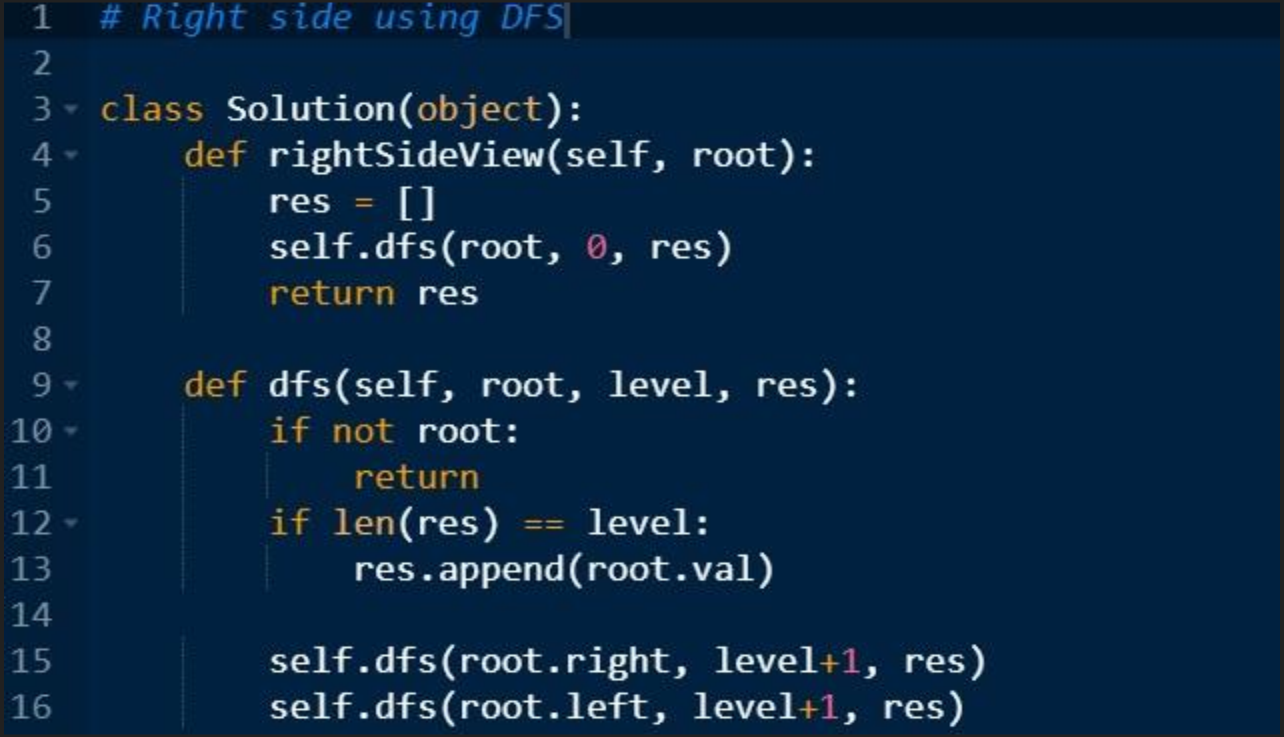
```
       20
     /    \
   10       25
  /  \      / \
 8    15   23   28
     /  \
    14   18
   /
  12
 /
11
```

Left View of following tree is
20, 10, 8, 14, 12, 11

Right View of following tree is
20, 25, 28, 18, 12, 11



```python
#Right view of binary tree
from collections import deque
class Solution(object):
    def rightSideView(self, root):
        rightmost_value_at_depth = dict()
        max_depth = -1

        queue = deque([(root, 0)])
        while queue:
            node, depth = queue.popleft()

            if node is not None:
                max_depth = max(max_depth, depth)
                rightmost_value_at_depth[depth] = node.val
                queue.append((node.left, depth+1))
                queue.append((node.right, depth+1))

        return [rightmost_value_at_depth[depth] for depth in range(max_depth+1)]
```

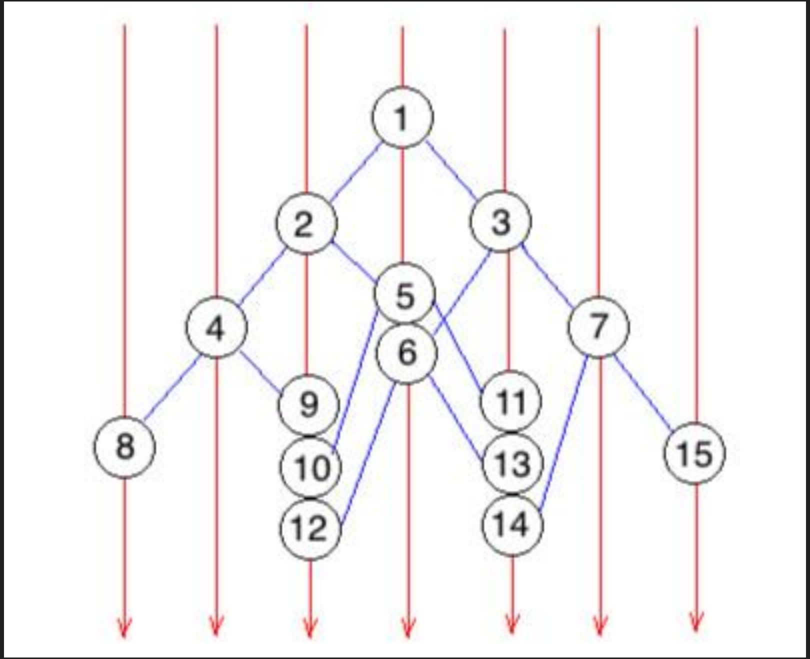


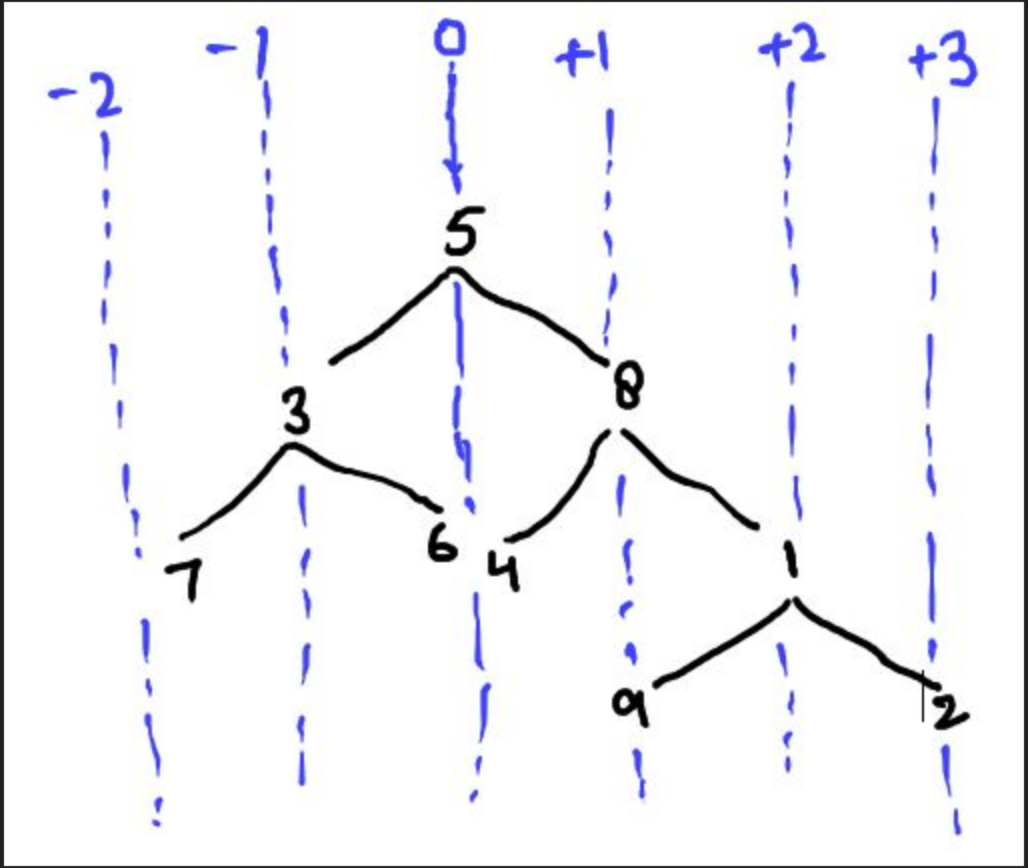
```python
# Right side using DFS

class Solution(object):
    def rightSideView(self, root):
        res = []
        self.dfs(root, 0, res)
        return res

    def dfs(self, root, level, res):
        if not root:
            return
        if len(res) == level:
            res.append(root.val)

        self.dfs(root.right, level+1, res)
        self.dfs(root.left, level+1, res)
```

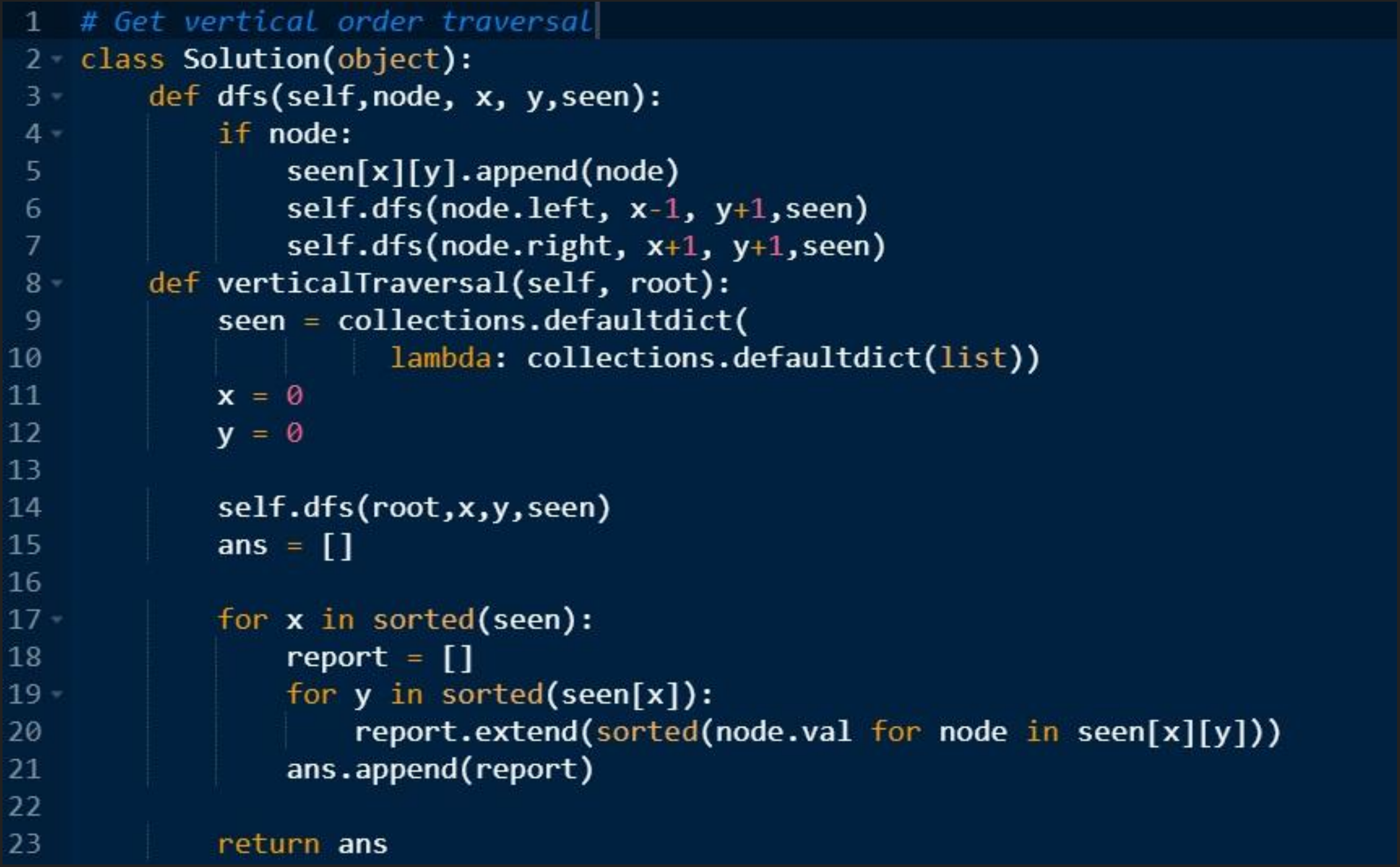
Question: Given a binary tree, return the vertical order traversal of its nodes values.

YOU HAVE 15 MINUTES

https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

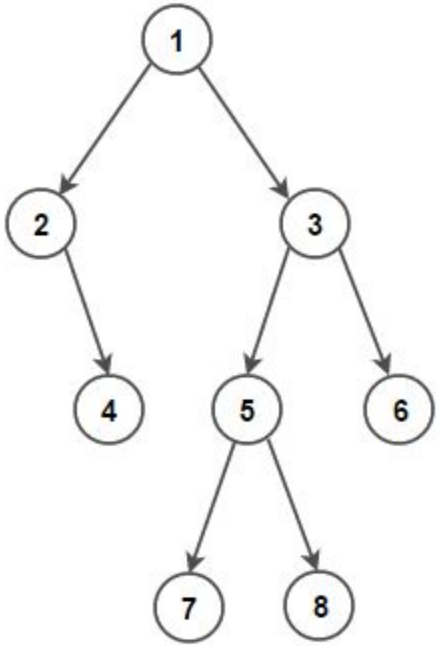

-2
-1
0
+1
+2
+3
5
3
8
7
6
4
1
9
2


```python
# Get vertical order traversal
class Solution(object):
    def dfs(self,node, x, y,seen):
        if node:
            seen[x][y].append(node)
            self.dfs(node.left, x-1, y+1,seen)
            self.dfs(node.right, x+1, y+1,seen)
    def verticalTraversal(self, root):
        seen = collections.defaultdict(
                    lambda: collections.defaultdict(list))
        x = 0
        y = 0

        self.dfs(root,x,y,seen)
        ans = []

        for x in sorted(seen):
            report = []
            for y in sorted(seen[x]):
                report.extend(sorted(node.val for node in seen[x][y]))
            ans.append(report)

        return ans
```

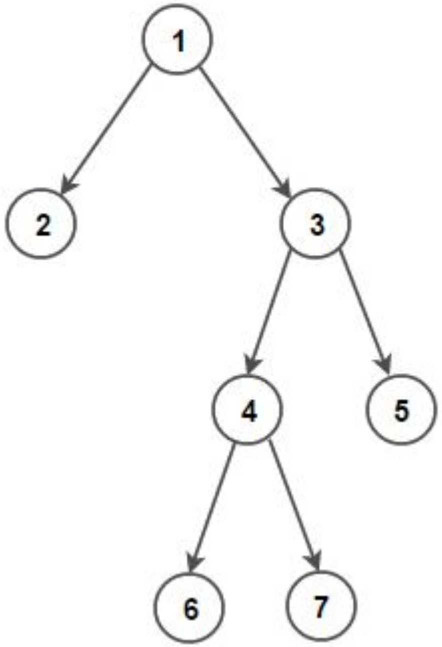
Question: Given a binary tree, determine if it is height-balanced.

YOU HAVE 15 MINUTES

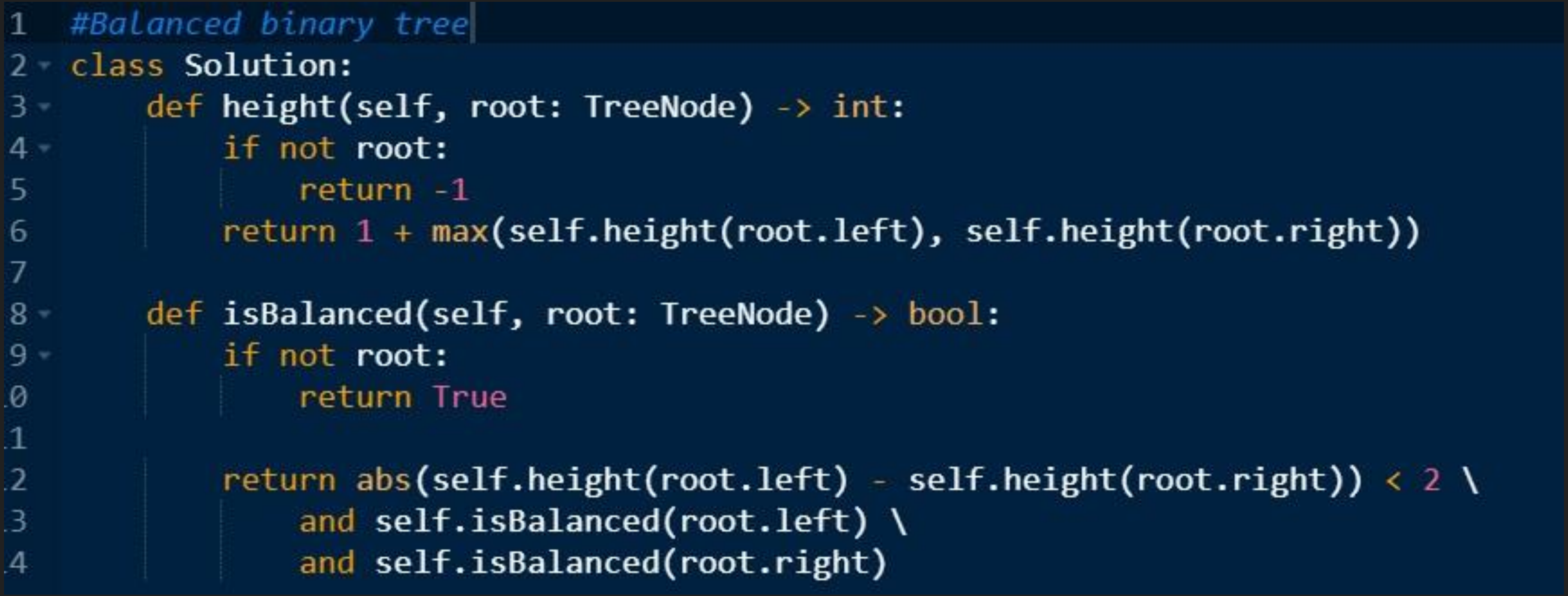
https://leetcode.com/problems/balanced-binary-tree/



Height Balanced Tree



Not a Height Balanced Tree

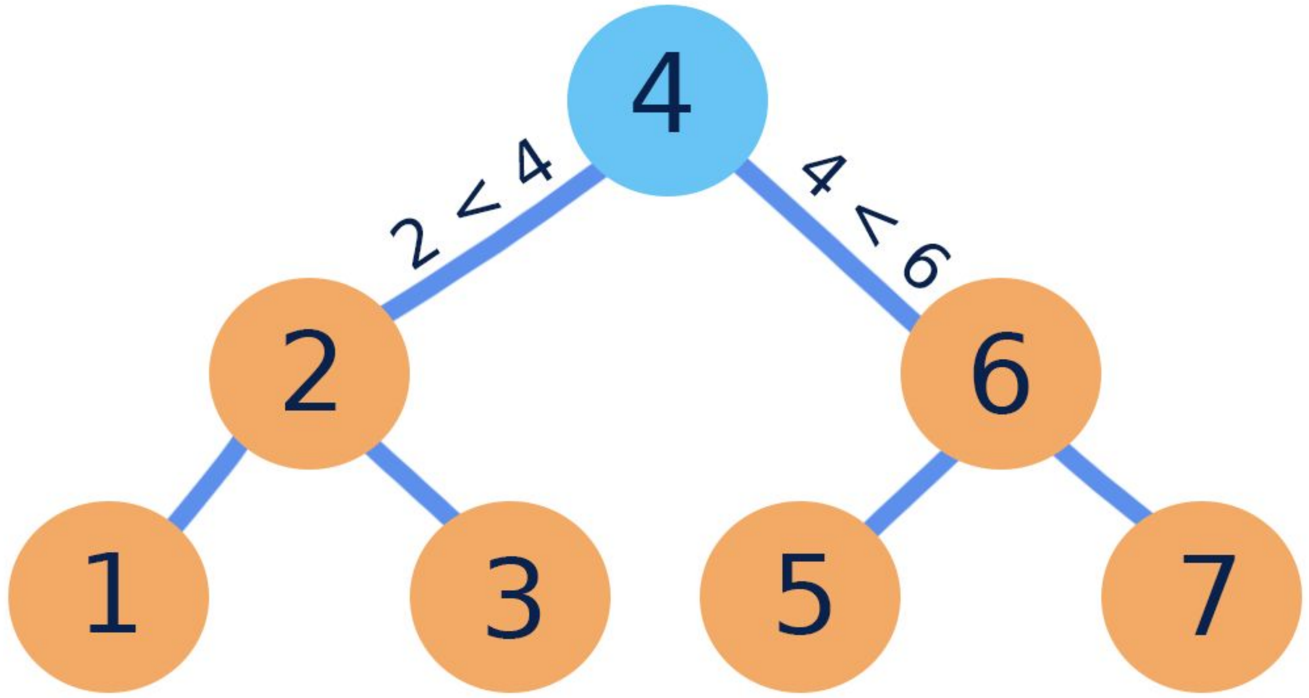


```python
#Balanced binary tree
class Solution:
    def height(self, root: TreeNode) -> int:
        if not root:
            return -1
        return 1 + max(self.height(root.left), self.height(root.right))

    def isBalanced(self, root: TreeNode) -> bool:
        if not root:
            return True

        return abs(self.height(root.left) - self.height(root.right)) < 2 \
            and self.isBalanced(root.left) \
            and self.isBalanced(root.right)
```

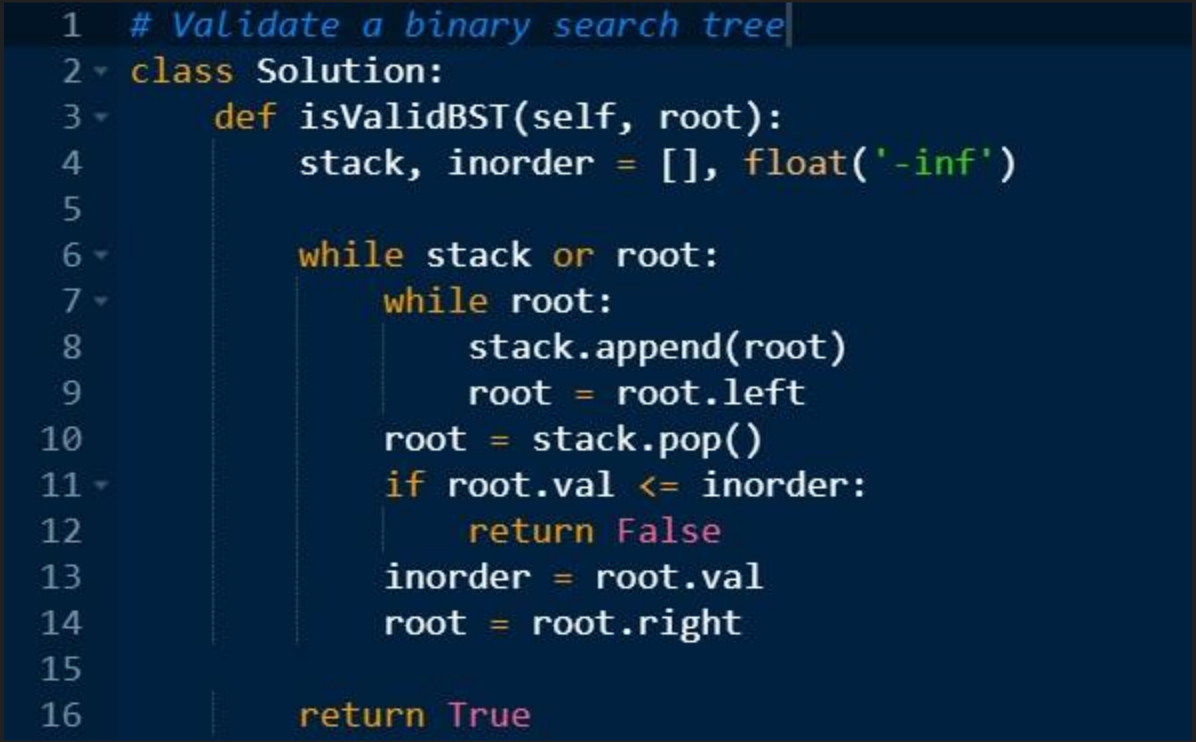
Question: Given a binary tree, determine if it is a valid binary search tree (BST).

YOU HAVE 15 MINUTES

https://leetcode.com/problems/validate-binary-search-tree/



In Order Traversal: 1 2 3 4 5 6 7



```python
# Validate a binary search tree
class Solution:
    def isValidBST(self, root):
        stack, inorder = [], float('-inf')

        while stack or root:
            while root:
                stack.append(root)
                root = root.left
            root = stack.pop()
            if root.val <= inorder:
                return False
            inorder = root.val
            root = root.right

        return True
```