Trees Live Class

INTERVIEW KICKSTART

Instructor Introduction

Chao Zhang

Tech Lead & Manager at Google (Bay Area)

Ex: Workday, Salesforce, LinkedIn

10+ yrs experience

Engineering, tech lead, team management

Summary of preclass videos

1. The first section discusses the problem of coming up with an efficient data structure to represent dictionaries and sets. Arrays and linked lists provide a brute-force implementation, which is then improved upon by the use of hash tables and finally **binary search trees**. Binary search tree operations like search, insert, delete, predecessor/successor, min, max are discussed along with iterative pseudocode that can be easily mapped to working code.

Summary of preclass videos

2. The second section discusses the use of **binary trees** (and N-ary trees) to represent hierarchical data. Two broad ways of traversing trees are discussed: BFS and DFS. Special types of DFS traversals like Preorder, Inorder and Postorder traversals are discussed with their simple recursive implementations. BFS is discussed using a queue-based iterative pseudocode. Finally, there is some discussion on how to reconstruct a binary tree from its traversals - preorder + inorder, or postorder + inorder. (No pseudocode given for it though)

Agenda

Tree Traversal - BFS - live coding - break

Tree Traversal - DFS - live coding - break

Top - Bottom - live coding - break

Bottom - Up - live coding - break

Tree Construction - live coding - break

Final Q&A

Remote class best practices

Keep your video camera on

Keep track of time

Mute your mics when not speaking

Participants for interactions, speak up once you are asked

Sudo code for coding examples, focus on logic not syntax

Post class Q&A

{ik} INTERVIEW KICKSTART

Trees Patterns

Tree *traversal* patterns:

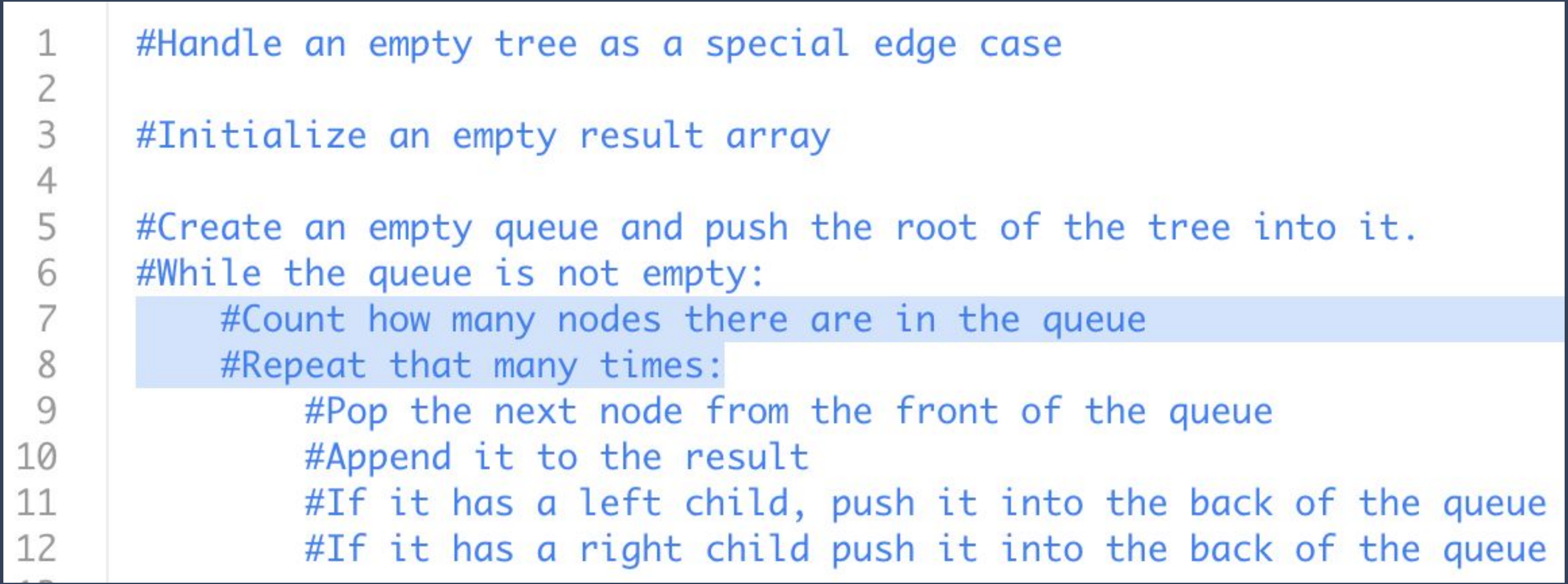
1. BFS
2. DFS

a) Top-down

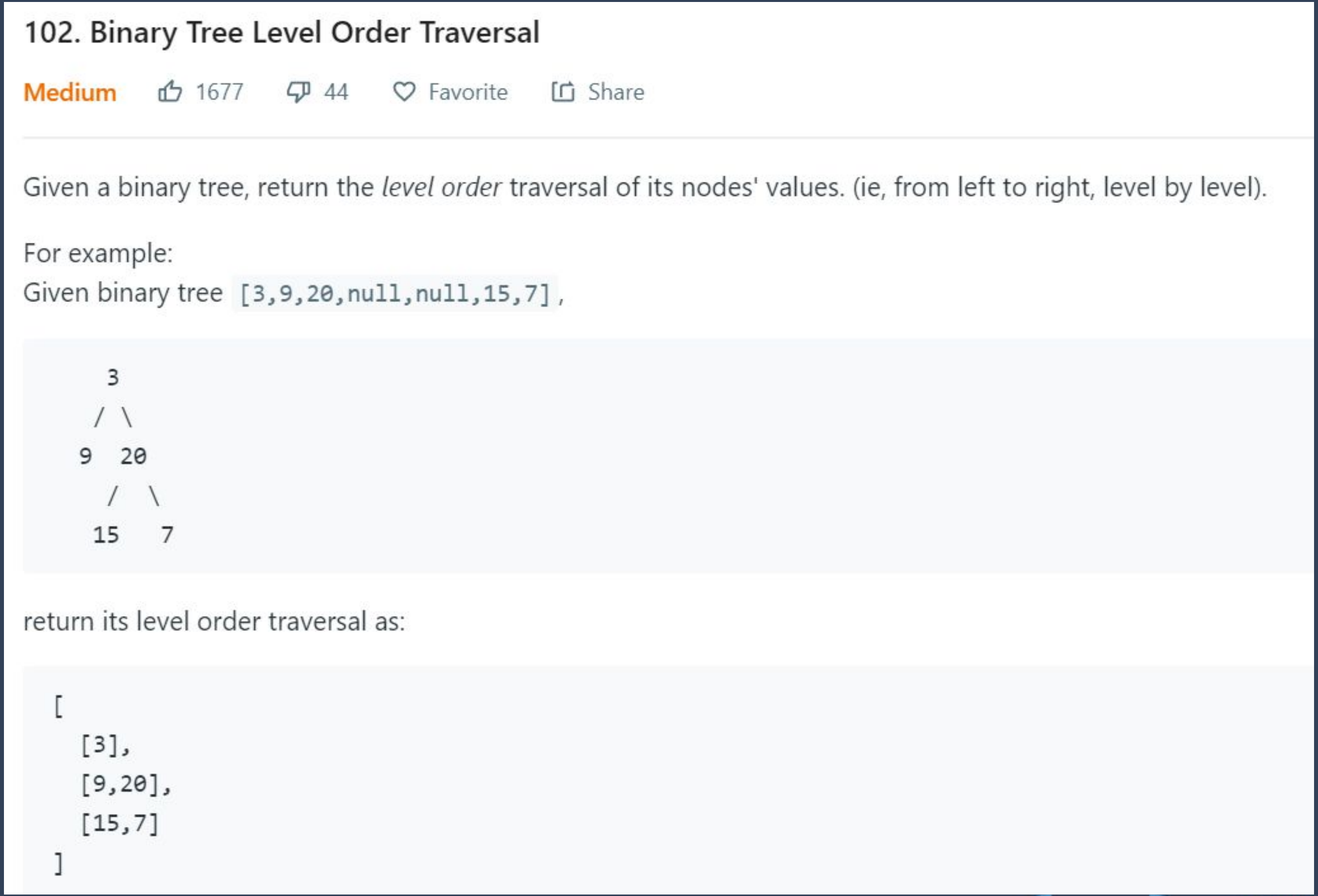
b) Bottom-up

Tree *construction* patterns

1. Top-down
2. Left-to-right (inorder)



```
#Handle an empty tree as a special edge case

#Initialize an empty result array

#Create an empty queue and push the root of the tree into it.
#While the queue is not empty:
    #Count how many nodes there are in the queue
    #Repeat that many times:
        #Pop the next node from the front of the queue
        #Append it to the result
        #If it has a left child, push it into the back of the queue
        #If it has a right child push it into the back of the queue
```



102. Binary Tree Level Order Traversal

Medium 1677 44 Favorite Share

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

For example:
Given binary tree `[3,9,20,null,null,15,7]`,

```
    3
   / \
  9  20
    /  \
   15   7
```

return its level order traversal as:

```
[
  [3],
  [9,20],
  [15,7]
]
```

Live Coding Session

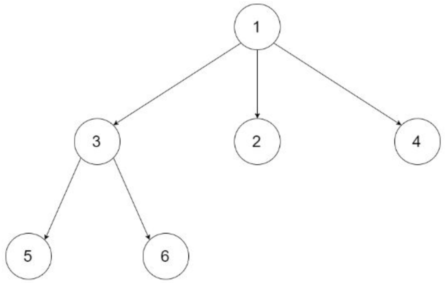
Live Coding from Chao

429. N-ary Tree Level Order Traversal

Easy 297 34 Favorite Share

Given an n-ary tree, return the level order traversal of its nodes' values. (ie, from left to right, level by level).

For example, given a `3-ary` tree:



We should return its level order traversal:

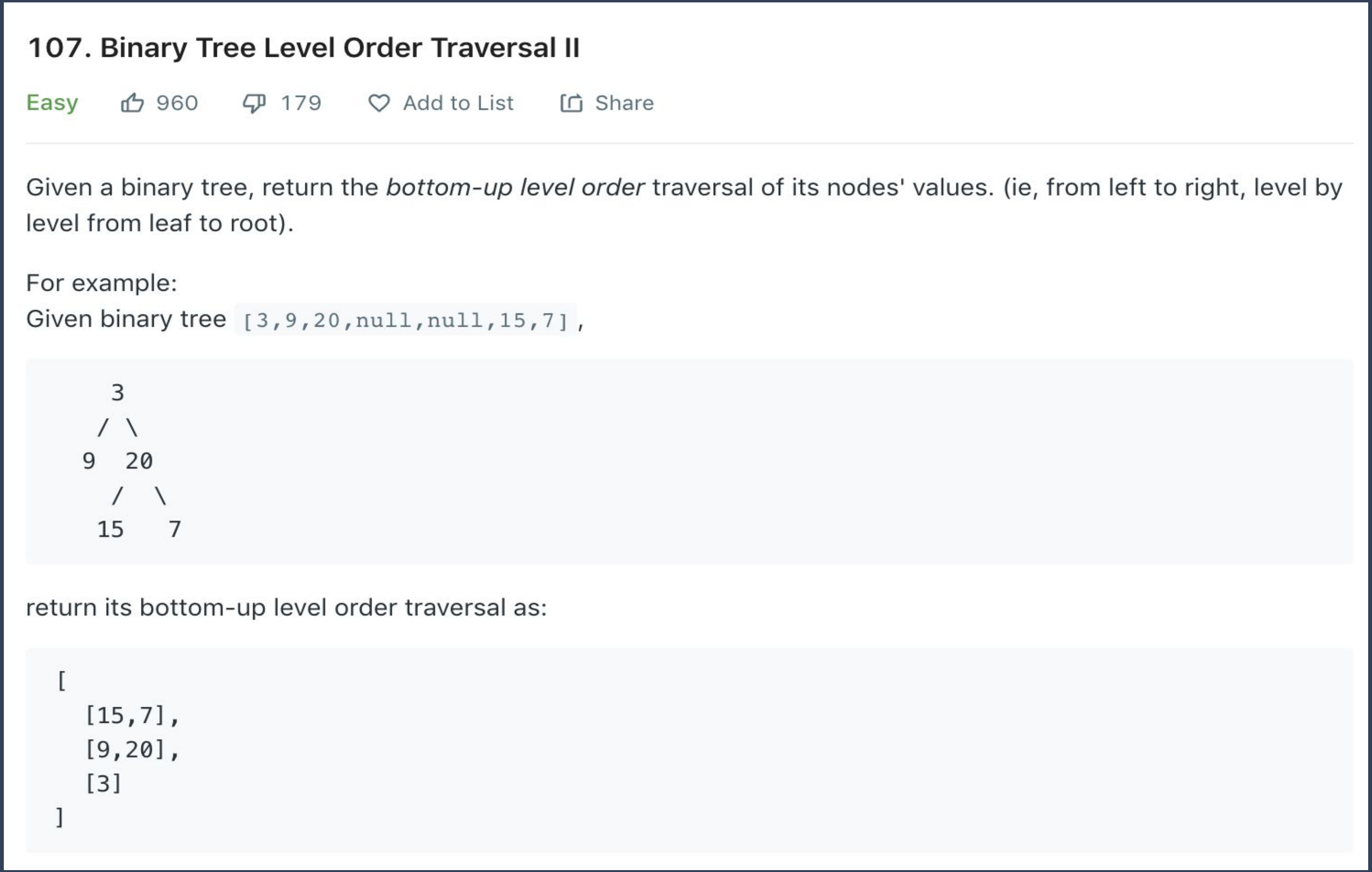
```
[
     [1],
     [3,2,4],
     [5,6]
]
```

Note:

1. The depth of the tree is at most `1000`.
2. The total number of nodes is at most `5000`.

Live Coding Session

Live Coding from Chao



107. Binary Tree Level Order Traversal II

Easy 960 179 Add to List Share

Given a binary tree, return the *bottom-up level order* traversal of its nodes' values. (ie, from left to right, level by level from leaf to root).

For example:
Given binary tree `[3,9,20,null,null,15,7]`,

```
    3
   / \
  9  20
    /  \
   15   7
```

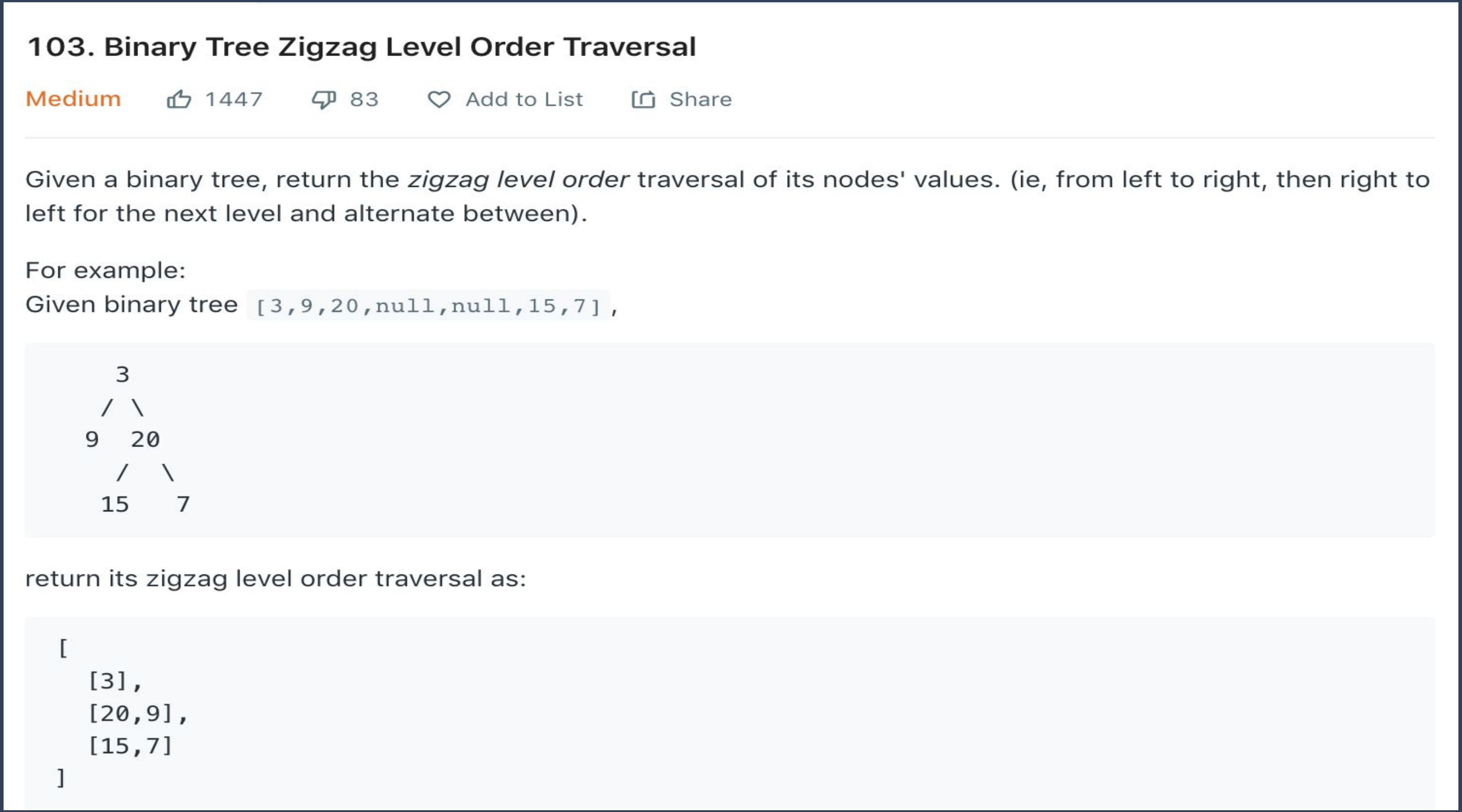
return its bottom-up level order traversal as:

```
[
  [15,7],
  [9,20],
  [3]
]
```

INTERVIEW KICKSTART

Live Coding Session

Live Coding from Chao



103. Binary Tree Zigzag Level Order Traversal

Medium 1447 83 Add to List Share

Given a binary tree, return the *zigzag level order* traversal of its nodes' values. (ie, from left to right, then right to left for the next level and alternate between).

For example:
Given binary tree `[3,9,20,null,null,15,7]`,

```
    3
   / \
  9  20
    /  \
   15   7
```

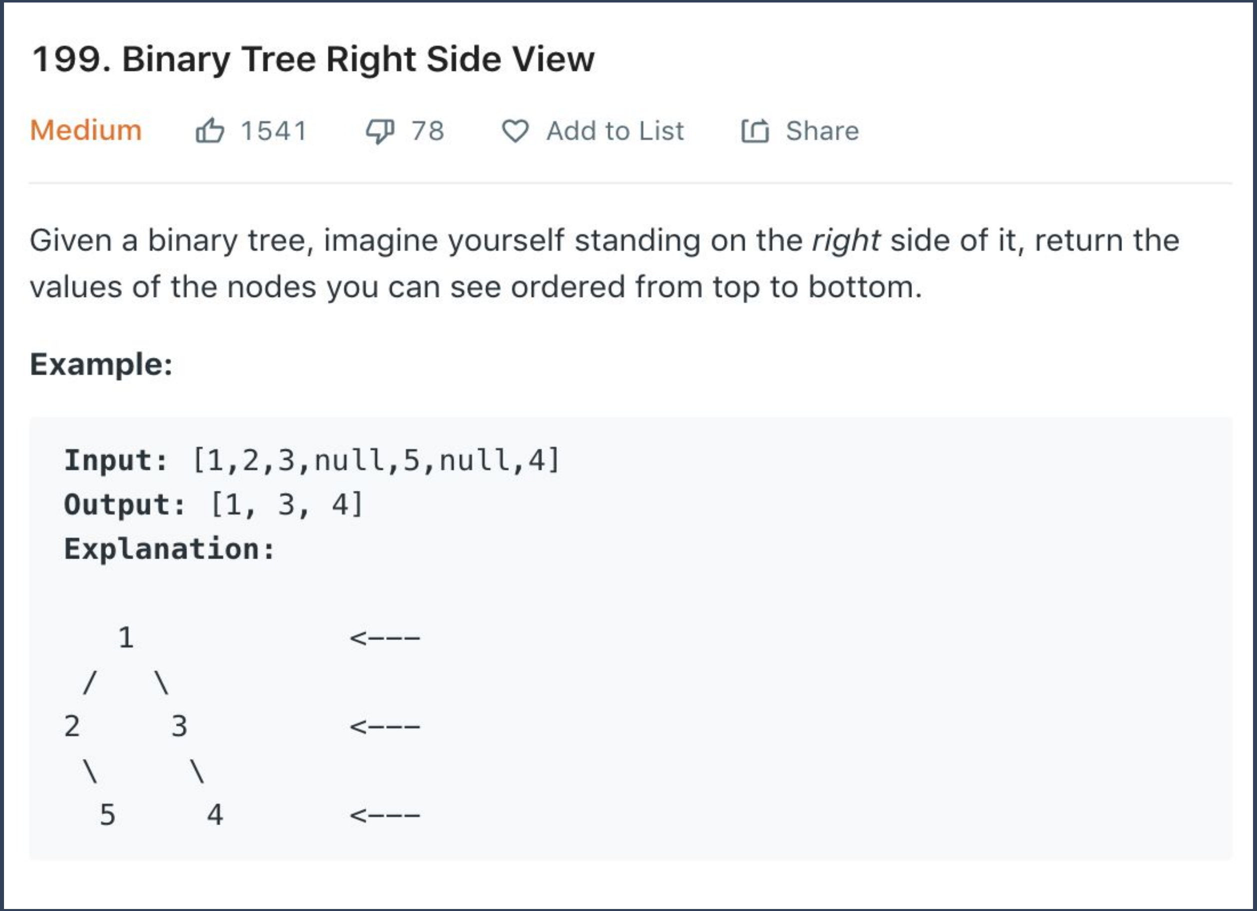
return its zigzag level order traversal as:

```
[
  [3],
  [20,9],
  [15,7]
]
```

{ik} INTERVIEW KICKSTART

Live Coding Session

Live Coding from Chao



199. Binary Tree Right Side View

Medium 1541 78 Add to List Share

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.

Example:

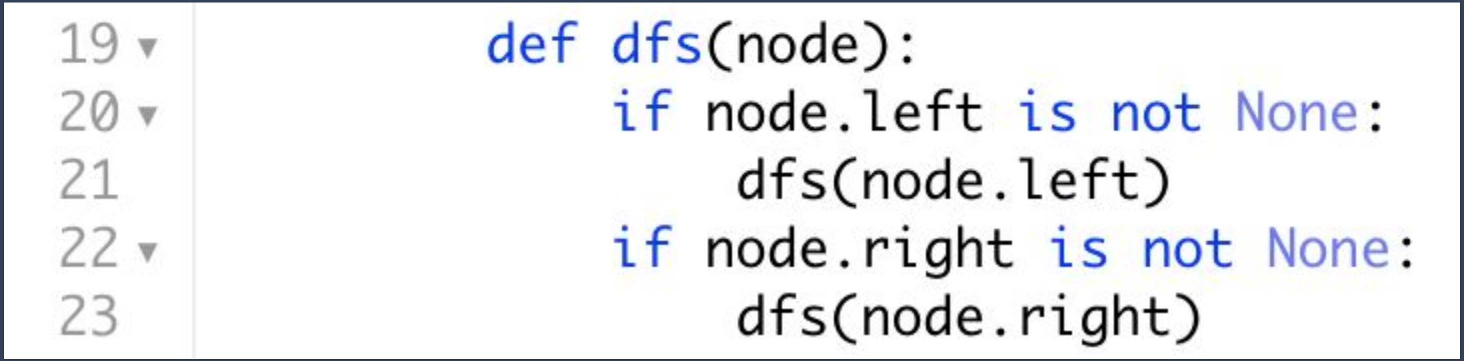
```
Input: [1,2,3,null,5,null,4]
Output: [1, 3, 4]
Explanation:

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

Live Coding Session

Live Coding from Chao

Trees DFS template

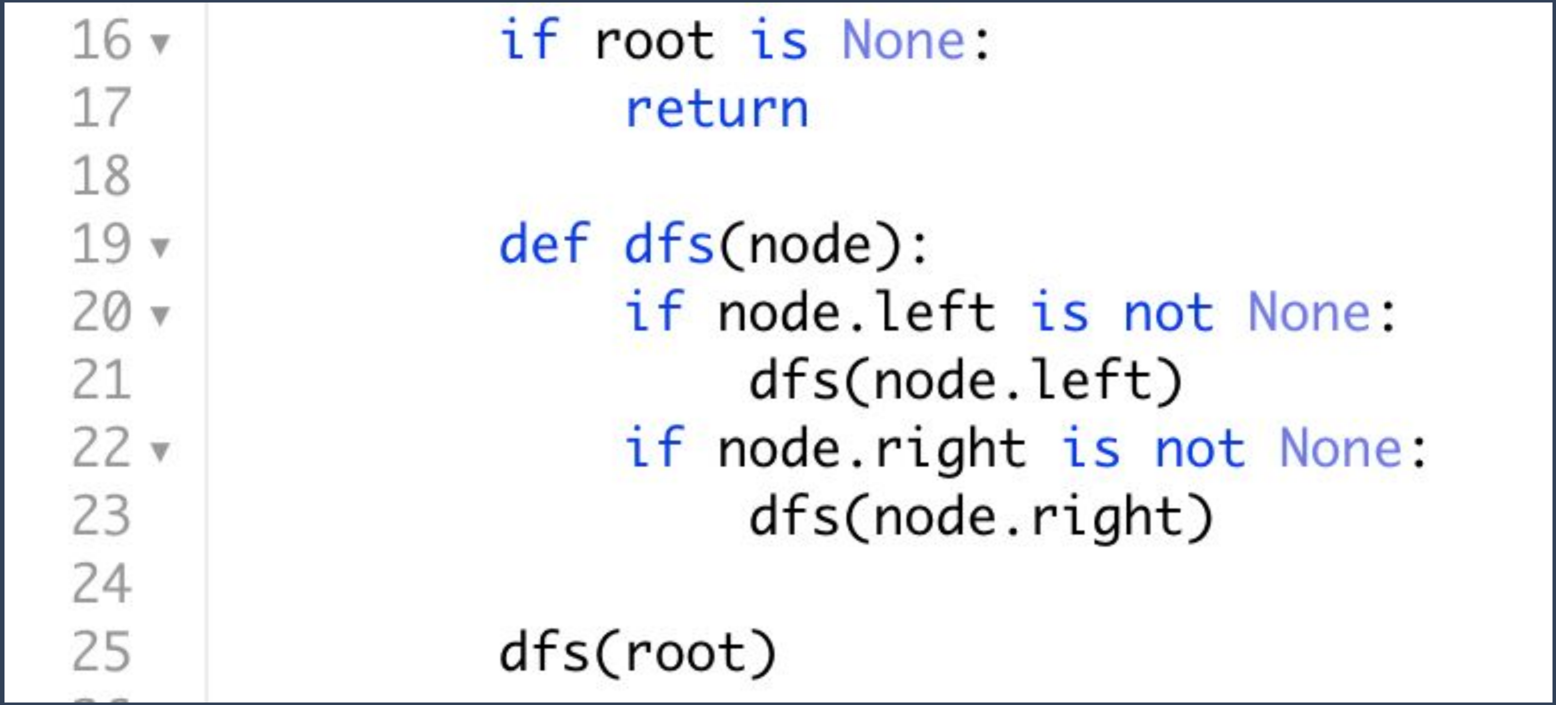
Note that according to this template, DFS is only called on a non-null node (like Graph DFS)



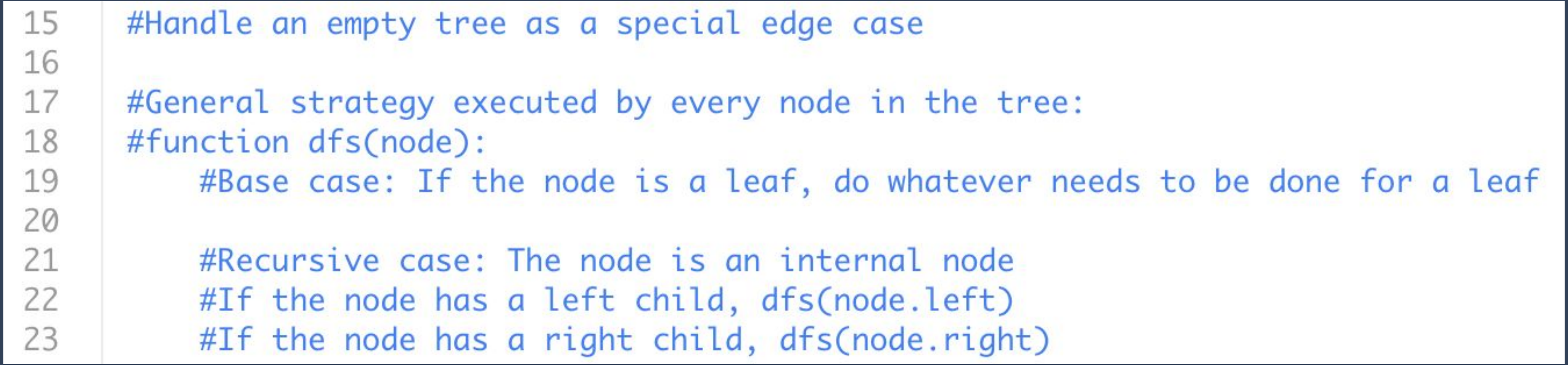
```python
def dfs(node):
    if node.left is not None:
        dfs(node.left)
    if node.right is not None:
        dfs(node.right)
```

Trees DFS template

Handle a null tree as a special case outside of DFS code...



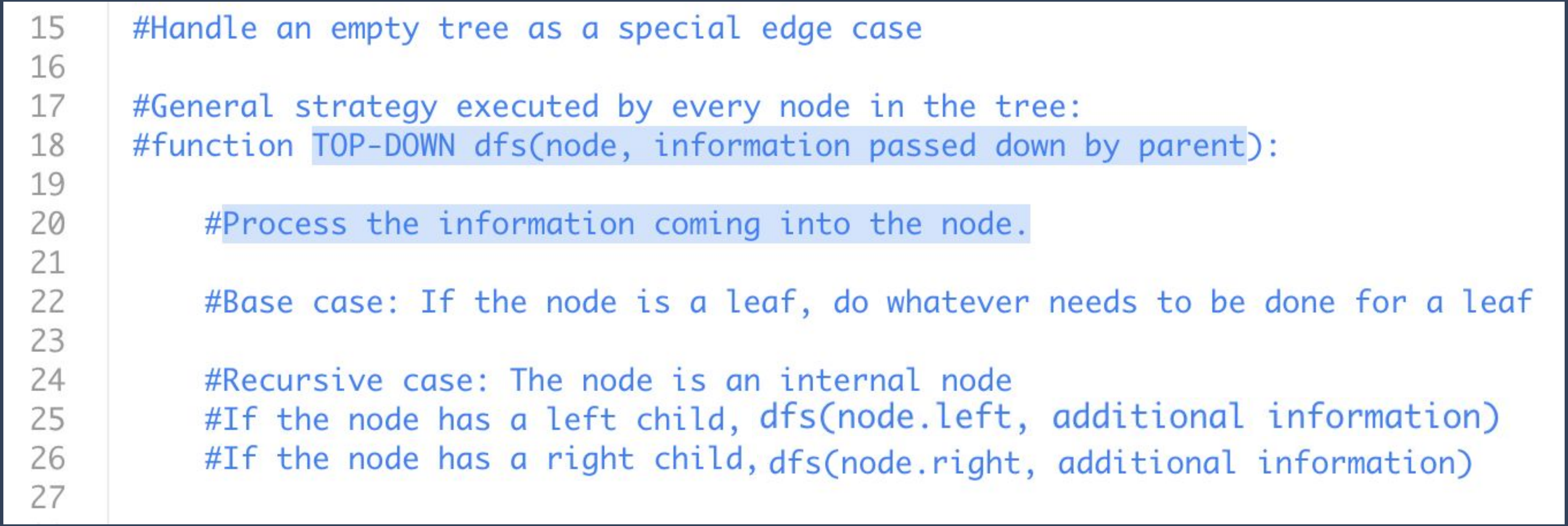
```python
if root is None:
    return

def dfs(node):
    if node.left is not None:
        dfs(node.left)
    if node.right is not None:
        dfs(node.right)

dfs(root)
```



```
#Handle an empty tree as a special edge case

#General strategy executed by every node in the tree:
#function dfs(node):
    #Base case: If the node is a leaf, do whatever needs to be done for a leaf

    #Recursive case: The node is an internal node
    #If the node has a left child, dfs(node.left)
    #If the node has a right child, dfs(node.right)
```

Top-down DFS: The flow of information is from top to bottom, like a waterfall.

Since the information is processed by the node before passing it down to either child, it is “pre-order”.



```
#Handle an empty tree as a special edge case

#General strategy executed by every node in the tree:
#function TOP-DOWN dfs(node, information passed down by parent):

    #Process the information coming into the node.

    #Base case: If the node is a leaf, do whatever needs to be done for a leaf

    #Recursive case: The node is an internal node
    #If the node has a left child, dfs(node.left, additional information)
    #If the node has a right child, dfs(node.right, additional information)

```

112. Path Sum

Easy 1198 380 Favorite Share

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

Note: A leaf is a node with no children.

Example:

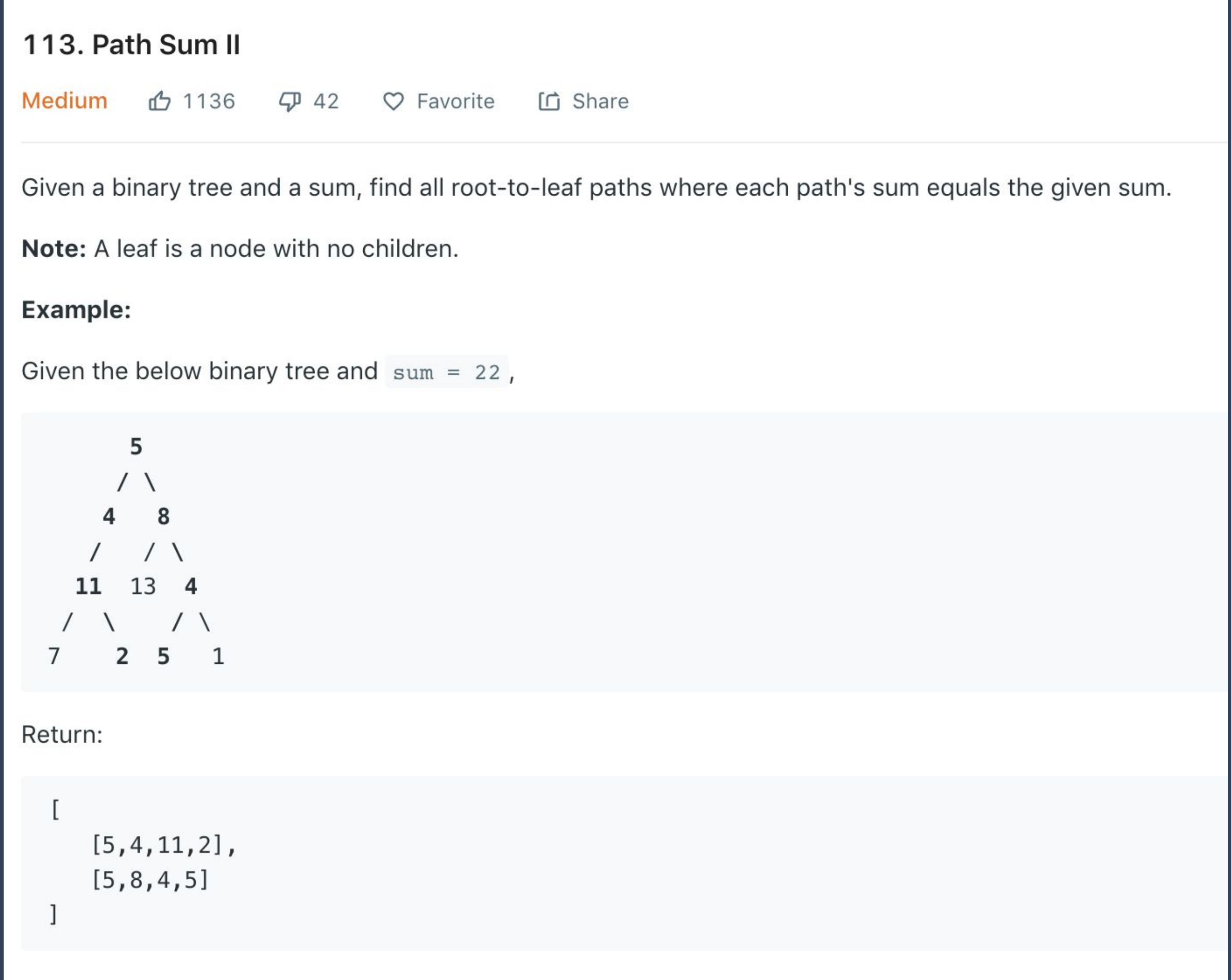
Given the below binary tree and `sum = 22`,

```
      5
     / \
    4   8
   /   / \
  11  13  4
 /  \      \
7    2      1
```

return true, as there exist a root-to-leaf path `5->4->11->2` which sum is 22.

Live Coding Session

Live Coding from Chao



113. Path Sum II

Medium 1136 42 Favorite Share

Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.

Note: A leaf is a node with no children.

Example:

Given the below binary tree and `sum = 22`,

```
      5
     / \
    4   8
   /   / \
  11  13  4
 /  \    / \
7    2  5   1
```

Return:

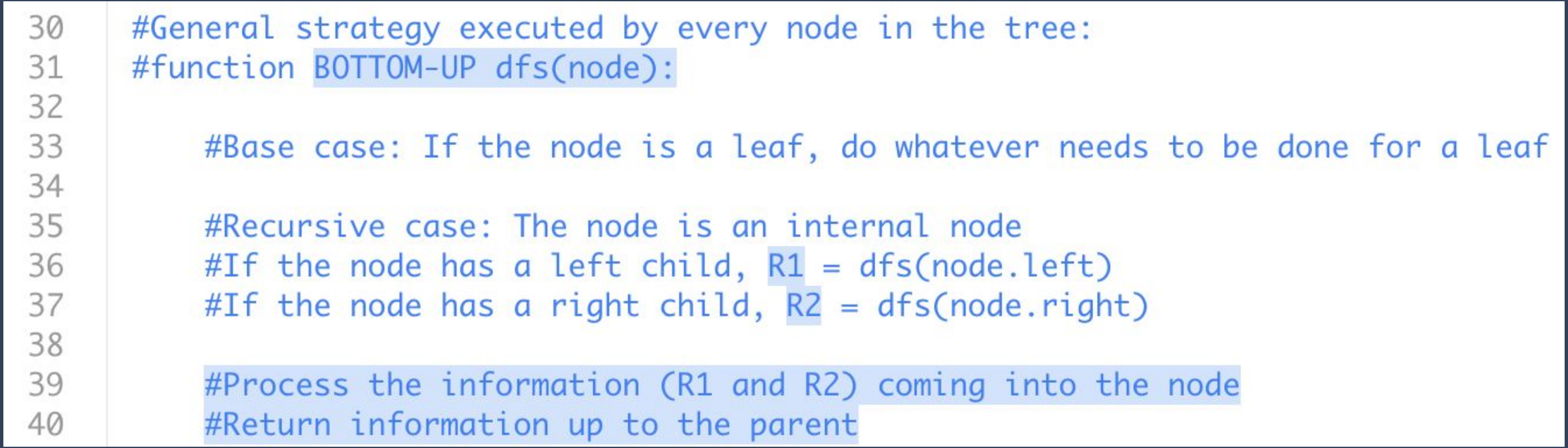
```
[
   [5,4,11,2],
   [5,8,4,5]
]
```

Live Coding Session

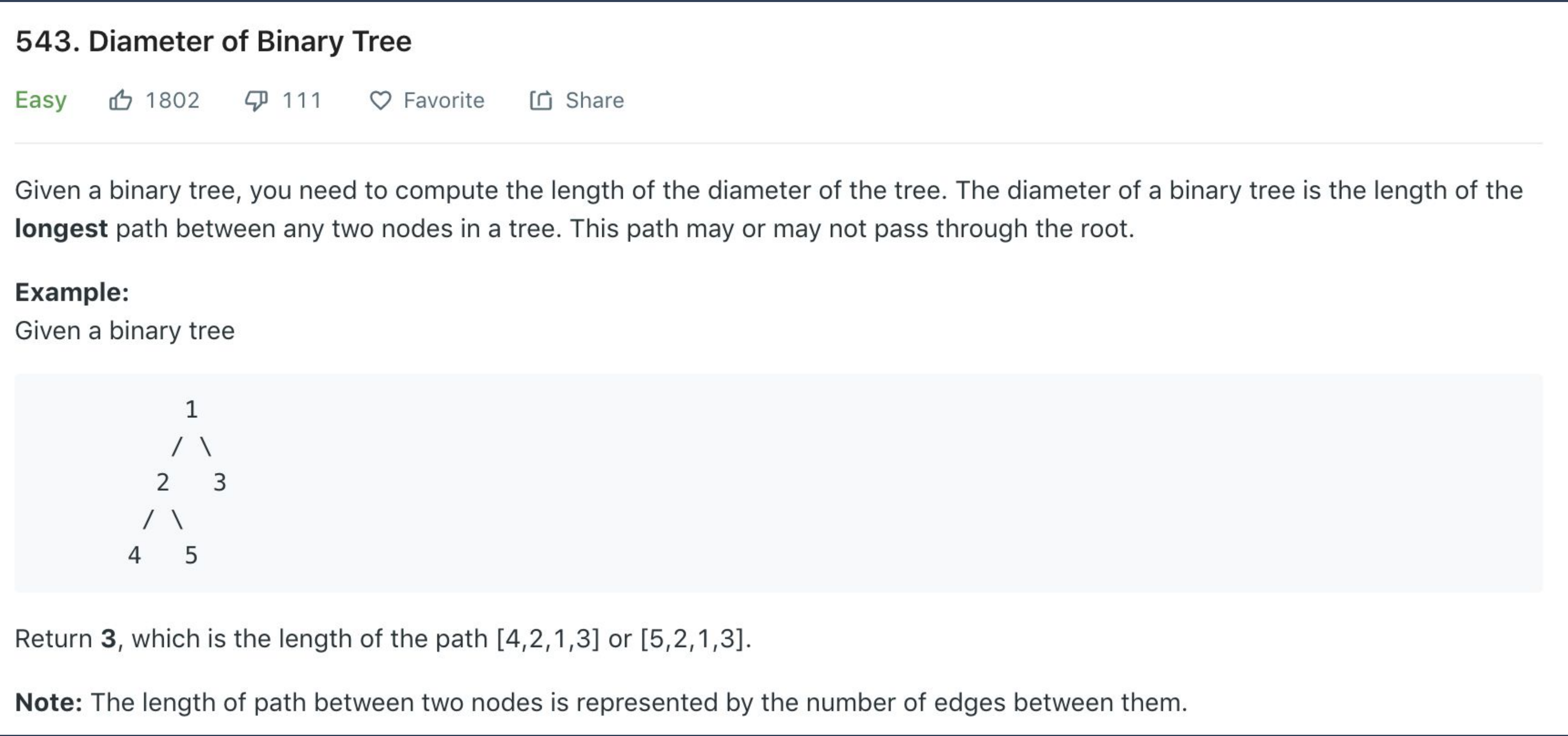
Live Coding from Chao

Bottom-up DFS: The flow of information is from bottom to top, like smoke rising up through vents.

Since the information is processed by the node after getting it from either child, it is “post-order”.



```
#General strategy executed by every node in the tree:
#function BOTTOM-UP dfs(node):

    #Base case: If the node is a leaf, do whatever needs to be done for a leaf

    #Recursive case: The node is an internal node
    #If the node has a left child, R1 = dfs(node.left)
    #If the node has a right child, R2 = dfs(node.right)

    #Process the information (R1 and R2) coming into the node
    #Return information up to the parent
```



543. Diameter of Binary Tree

Easy 1802 111 Favorite Share

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the **longest** path between any two nodes in a tree. This path may or may not pass through the root.

Example:
Given a binary tree

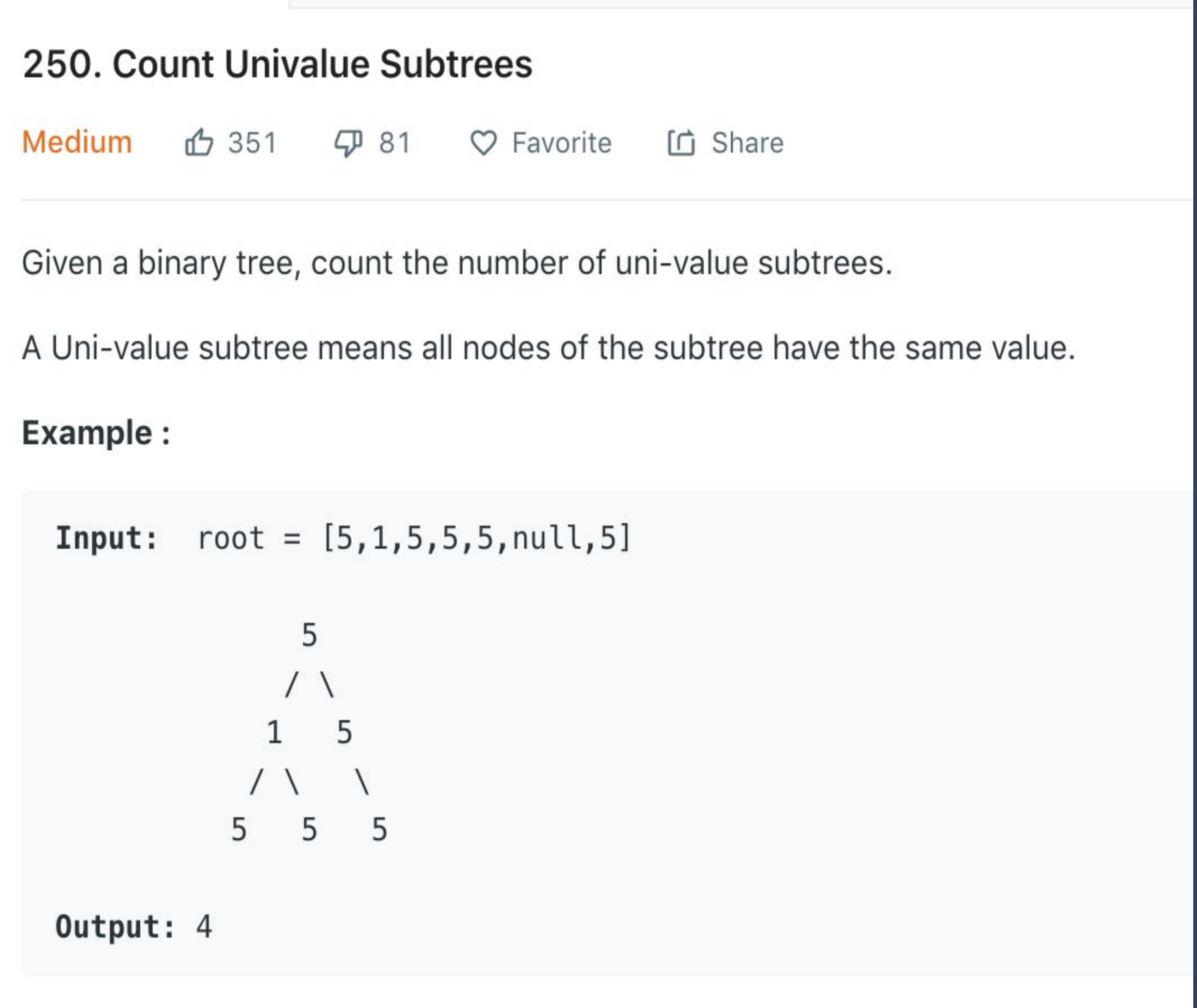
```
          1
         / \
        2   3
       / \
      4   5
```

Return **3**, which is the length of the path [4,2,1,3] or [5,2,1,3].

Note: The length of path between two nodes is represented by the number of edges between them.

Live Coding Session

Live Coding from Chao



250. Count Univalue Subtrees

Medium 351 81 Favorite Share

Given a binary tree, count the number of uni-value subtrees.

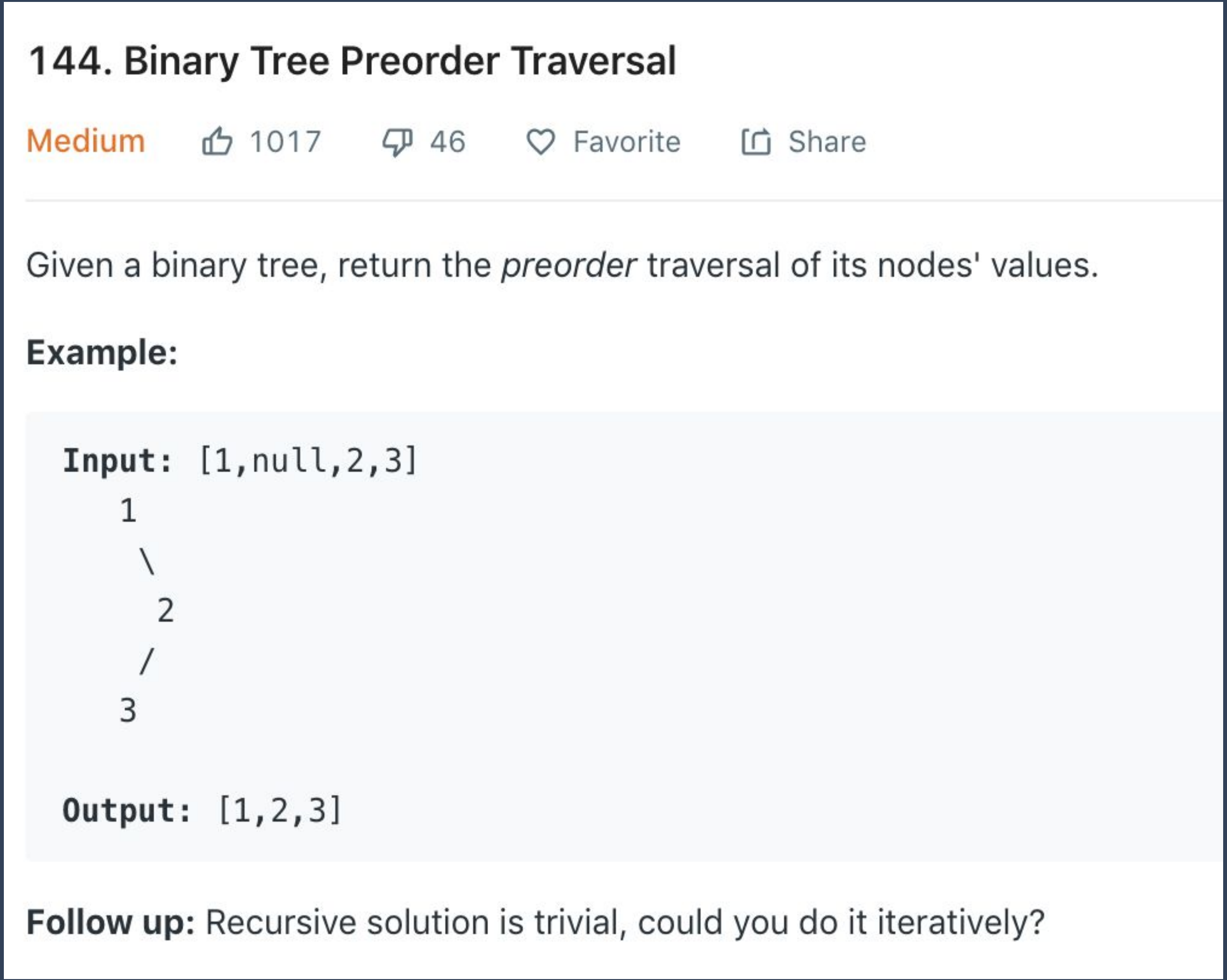
A Uni-value subtree means all nodes of the subtree have the same value.

Example :

```
Input:  root = [5,1,5,5,5,null,5]

              5
             / \
            1   5
           / \   \
          5   5   5

Output: 4
```

Live Coding Session

Live Coding from Chao



144. Binary Tree Preorder Traversal

Medium 1017 46 Favorite Share

Given a binary tree, return the *preorder* traversal of its nodes' values.

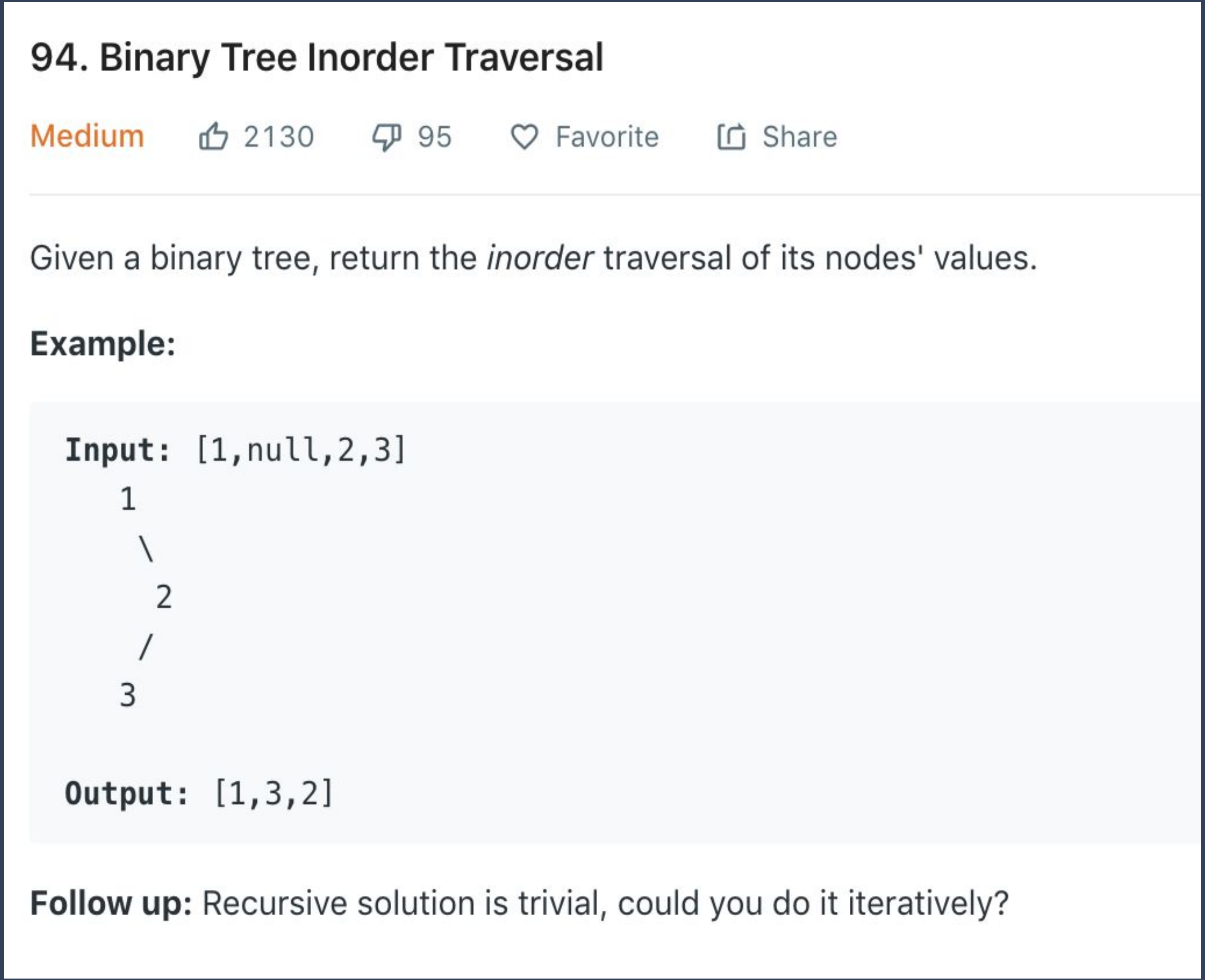
Example:

```
Input: [1,null,2,3]
   1
    \
     2
    /
   3

Output: [1,2,3]
```

Follow up: Recursive solution is trivial, could you do it iteratively?

Live Coding Session

Live Coding from Chao



94. Binary Tree Inorder Traversal

Medium 2130 95 Favorite Share

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

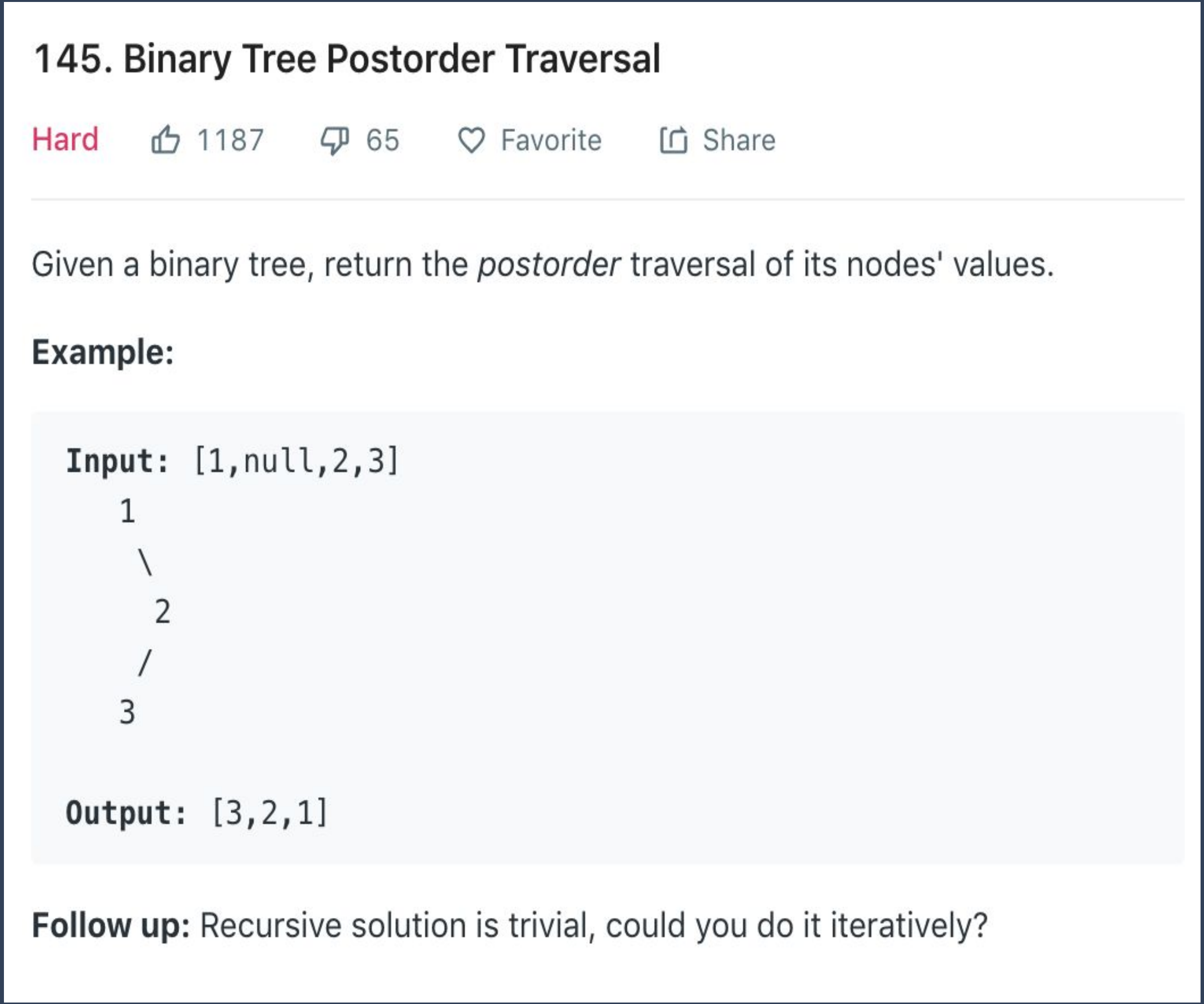
Example:

```
Input: [1,null,2,3]
   1
    \
     2
    /
   3

Output: [1,3,2]
```

Follow up: Recursive solution is trivial, could you do it iteratively?

Live Coding Session

Live Coding from Chao



145. Binary Tree Postorder Traversal

Hard 1187 65 Favorite Share

Given a binary tree, return the *postorder* traversal of its nodes' values.

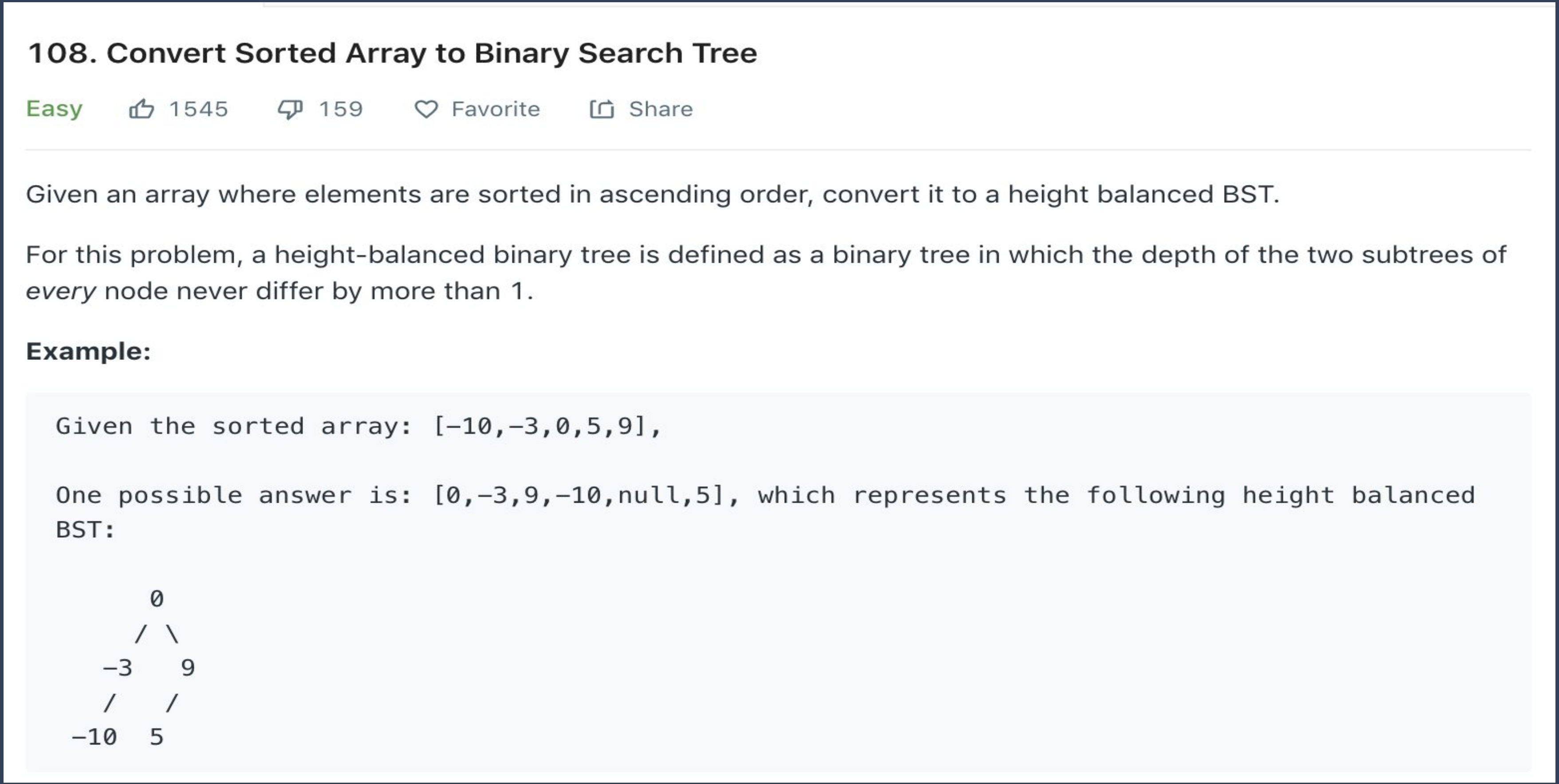
Example:

```
Input: [1,null,2,3]
   1
    \
     2
    /
   3

Output: [3,2,1]
```

Follow up: Recursive solution is trivial, could you do it iteratively?

Live Coding Session

Live Coding from Chao



108. Convert Sorted Array to Binary Search Tree

Easy 1545 159 Favorite Share

Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

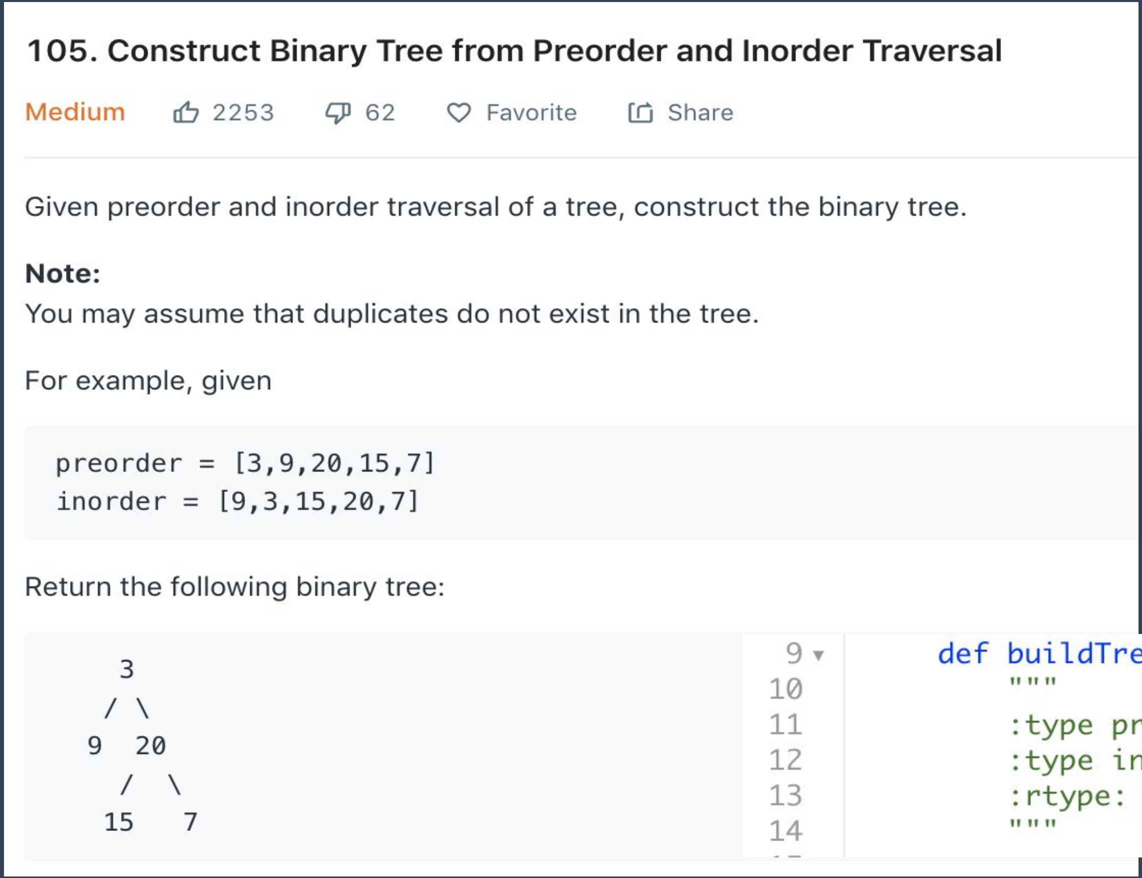
For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

Example:

```
Given the sorted array: [-10,-3,0,5,9],

One possible answer is: [0,-3,9,-10,null,5], which represents the following height balanced
BST:

      0
     / \
   -3   9
   /   /
 -10  5
```

Live Coding Session

Live Coding from Chao



105. Construct Binary Tree from Preorder and Inorder Traversal

Medium 2253 62 Favorite Share

Given preorder and inorder traversal of a tree, construct the binary tree.

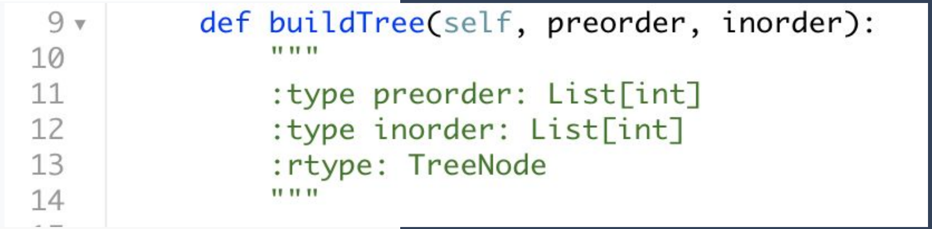
Note:
You may assume that duplicates do not exist in the tree.

For example, given

```
preorder = [3,9,20,15,7]
inorder = [9,3,15,20,7]
```

Return the following binary tree:

```
    3
   / \
  9  20
    /  \
   15   7
```



```
    def buildTree(self, preorder, inorder):
        """
        :type preorder: List[int]
        :type inorder: List[int]
        :rtype: TreeNode
        """
```

Live Coding Session

Live Coding from Chao

1008. Construct Binary Search Tree from Preorder Traversal

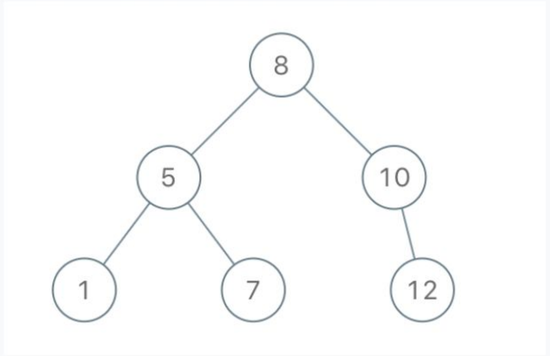
Medium 427 18 Favorite Share

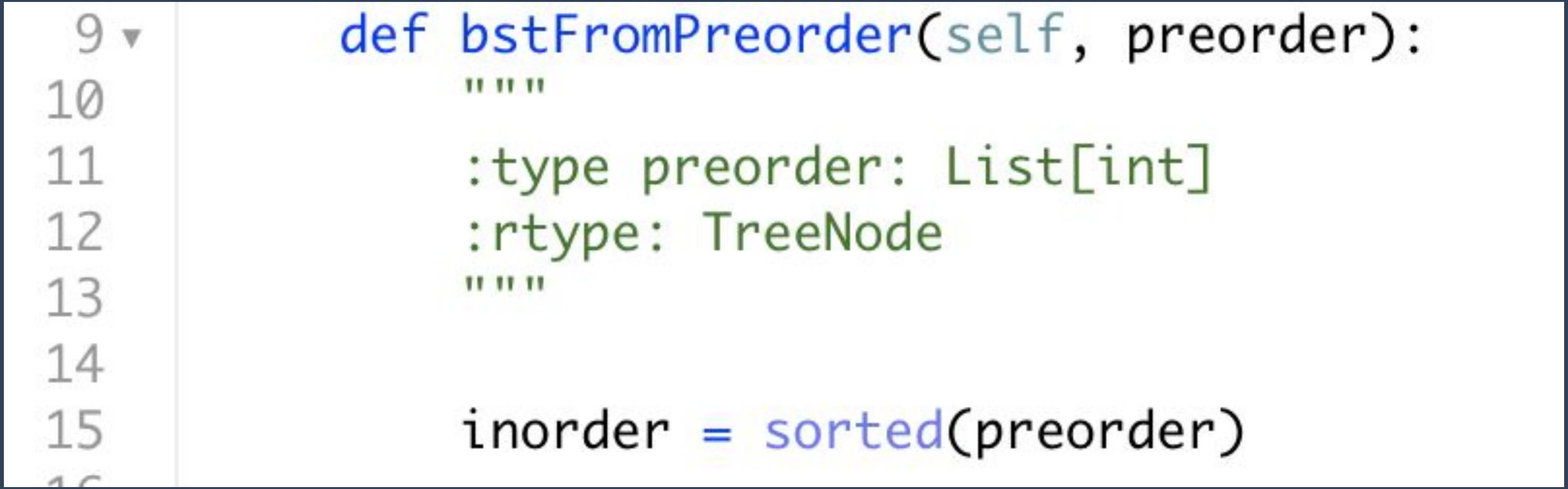
Return the root node of a binary **search** tree that matches the given `preorder` traversal.

(Recall that a binary search tree is a binary tree where for every `node`, any descendant of `node.left` has a value `<` `node.val`, and any descendant of `node.right` has a value `>` `node.val`. Also recall that a preorder traversal displays the value of the `node` first, then traverses `node.left`, then traverses `node.right`.)

Example 1:

```
Input: [8,5,1,7,10,12]
Output: [8,5,10,1,7,null,12]
```



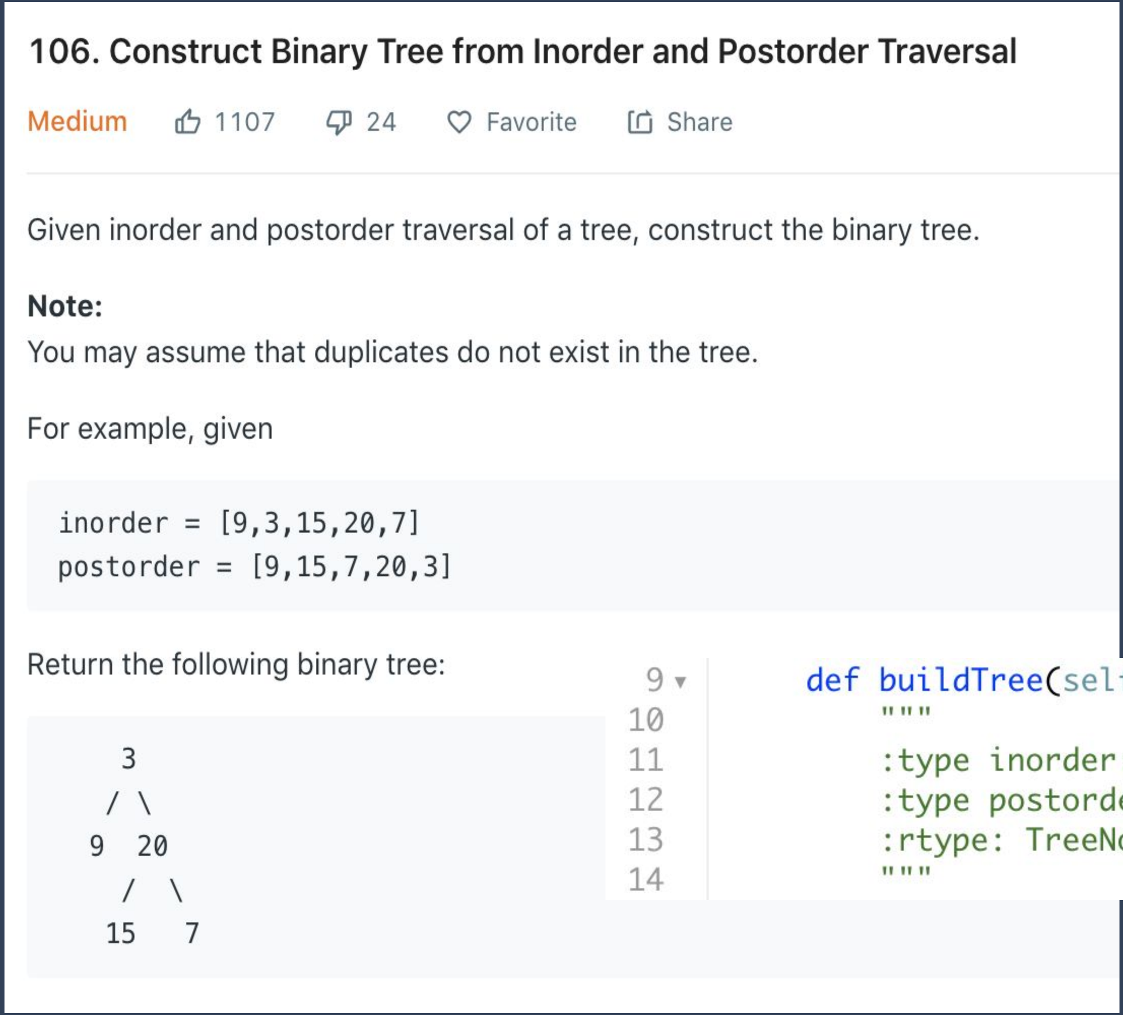


```
    def bstFromPreorder(self, preorder):
        """
        :type preorder: List[int]
        :rtype: TreeNode
        """

        inorder = sorted(preorder)
```

(same code as before here)

Live Coding Session

Live Coding from Chao



106. Construct Binary Tree from Inorder and Postorder Traversal

Given inorder and postorder traversal of a tree, construct the binary tree.

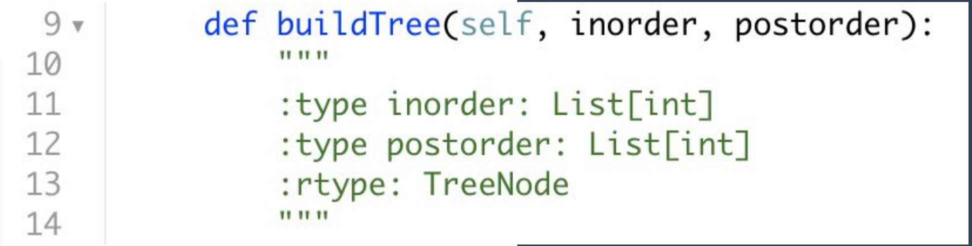
Note:
You may assume that duplicates do not exist in the tree.

For example, given

```
inorder = [9,3,15,20,7]
postorder = [9,15,7,20,3]
```

Return the following binary tree:

```
    3
   / \
  9  20
    /  \
   15   7
```



```
    def buildTree(self, inorder, postorder):
        """
        :type inorder: List[int]
        :type postorder: List[int]
        :rtype: TreeNode
        """
```

Live Coding Session

Live Coding from Chao

Slides Created by:

Omkar Deshpande

{ik} INTERVIEW KICKSTART