894.All Possible Full Binary Trees

894.All Possible Full Binary Trees

Medium

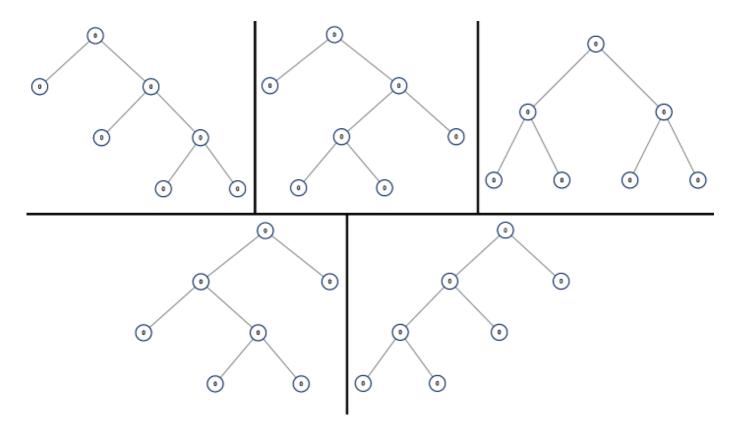
2064161Add to ListShare

Given an integer n, return a list of all possible **full binary trees** with n nodes. Each node of each tree in the answer must have node.val == 0.

Each element of the answer is the root node of one possible tree. You may return the final list of trees in **any order**.

A **full binary tree** is a binary tree where each node has exactly 0 or 2 children.

Example 1:



```
Input: n = 7
Output: [[0,0,0,null,null,0,0,null,null,0,0],[0,0,0,null,null,0,0,0],
[0,0,0,0,0,0],[0,0,0,0,null,null,null,null,null,0,0],
[0,0,0,0,0,null,null,0,0]]
```

Example 2:

```
Input: n = 3
Output: [[0,0,0]]
```

Constraints:

• [1 <= n <= 20]

```
# Definition for a binary tree node.
# class TreeNode:
    def init (self, val=0, left=None, right=None):
          self.val = val
         self.left = left
#
          self.right = right
class Solution:
   def allPossibleFBT(self, n: int) -> List[Optional[TreeNode]]:
        if n%2 == 0:
           return []
        if n==1:
           return [TreeNode(0)]
        return self.helper(n)
   def helper(self,n):
        if n==1:
            return [TreeNode(0)]
        ans = []
        for i in range (1, n, 2):
            leftTrees = self.helper(i)
            rightTrees = self.helper(n-i-1)
           for leftNode in leftTrees:
                for rightNode in rightTrees:
                    node = TreeNode(0)
                    node.left = leftNode
                    node.right = rightNode
                    ans.append(node)
        return ans
```