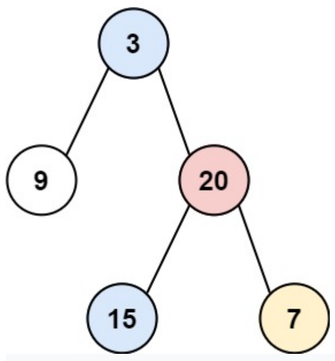
314. Binary Tree Vertical Order Traversal

Medium 1850 235 Add to List Share

Given the `root` of a binary tree, return ***the vertical order traversal*** *of its nodes' values*. (i.e., from top to bottom, column by column).

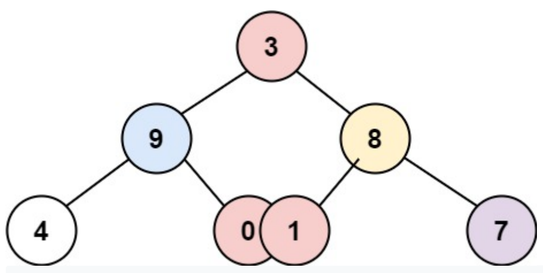
If two nodes are in the same row and column, the order should be from **left to right**.

Example 1:



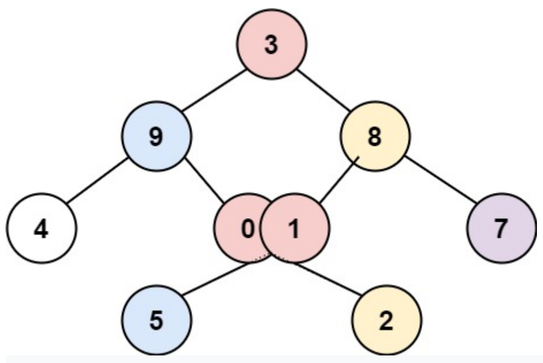

```
Input: root = [3,9,20,null,null,15,7]
Output: [[9],[3,15],[20],[7]]
```

Example 2:




```
Input: root = [3,9,8,4,0,1,7]
Output: [[4],[9],[3,0,1],[8],[7]]
```

Example 3:




```
Input: root = [3,9,8,4,0,1,7,null,null,null,2,5]
Output: [[4],[9,5],[3,0,1],[8,2],[7]]
```

Example 4:

```
Input: root = []
Output: []
```

Constraints:

- The number of nodes in the tree is in the range `[0, 100]`.
- `-100 <= Node.val <= 100`

Accepted 203,907 | Submissions 412,709

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 63 Bloomberg | 6 Amazon | 4 Apple | 2 Salesforce | 2

Related Topics

Hash Table Tree Depth-First Search Breadth-First Search Binary Tree

Similar Questions

Binary Tree Level Order Traversal Medium

Java Autocomplete

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
  Map<Integer, ArrayList<Pair<Integer, Integer>>> columnTable = new HashMap();
  int minColumn = 0, maxColumn = 0;

  private void DFS(TreeNode node, Integer row, Integer column) {
    if (node == null)
      return;

    if (!columnTable.containsKey(column)) {
      this.columnTable.put(column, new ArrayList<Pair<Integer, Integer>>());
    }

    this.columnTable.get(column).add(new Pair<Integer, Integer>(row, node.val));
    this.minColumn = Math.min(minColumn, column);
    this.maxColumn = Math.max(maxColumn, column);
    // preorder DFS traversal
    this.DFS(node.left, row + 1, column - 1);
    this.DFS(node.right, row + 1, column + 1);
  }

  public List<List<Integer>> verticalOrder(TreeNode root) {
    List<List<Integer>> output = new ArrayList();
    if (root == null) {
      return output;
    }

    this.DFS(root, 0, 0);

    // Retrieve the resuts, by ordering by column and sorting by row
    for (int i = minColumn; i < maxColumn + 1; ++i) {

      Collections.sort(columnTable.get(i), new Comparator<Pair<Integer, Integer>>() {
        @Override
        public int compare(Pair<Integer, Integer> p1, Pair<Integer, Integer> p2) {
          return p1.getKey() - p2.getKey();
        }
```