[**Binary Tree Vertical Order Traversal**](https://leetcode.com/problems/binary-tree-vertical-order-traversal/)

**import** java.util.\*;

**public** **class** BTVerticalTraversal {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

TreeNode root = **new** TreeNode(10);

root.left = **new** TreeNode(5);

root.left.left = **new** TreeNode(3);

root.left.right = **new** TreeNode(7);

root.right = **new** TreeNode(15);

root.right.right = **new** TreeNode(18);

System.***out***.println(*verticalOrder*(root));

}

**public** **static** List<List<Integer>> verticalOrder(TreeNode root) {

List<List<Integer>> result = **new** ArrayList<>();

**if**(root == **null**) {

**return** result;

}

Queue<TreeNode> qNode = **new** LinkedList<>();

qNode.offer(root);

Queue<Integer> qDistance = **new** LinkedList<>();

qDistance.offer(0);

HashMap<Integer , List<Integer>> map = **new** HashMap<>();

**int** min = 0;

**int** max = 0;

**while**(!qNode.isEmpty()) {

**int** size = qNode.size();

**for**(**int** i = 0 ; i < size ; i++) {

TreeNode tempNode = qNode.poll();

**int** tempDistance = qDistance.poll();

**if**(!map.containsKey(tempDistance)) {

map.put(tempDistance, **new** ArrayList<>());

}

map.get(tempDistance).add(tempNode.val);

**if**(tempNode.left != **null**) {

qNode.offer(tempNode.left);

qDistance.offer(tempDistance - 1);

min = Math.*min*(min, tempDistance - 1);

}

**if**(tempNode.right != **null**) {

qNode.offer(tempNode.right);

qDistance.offer(tempDistance + 1);

max = Math.*max*(max, tempDistance + 1);

}

}

}

**for**(**int** i = min ; i <= max ; i++) {

**if**(map.containsKey(i)) {

result.add(map.get(i));

}

}

**return** result;

}

}

Time complexity : O(n). n is Number of nodes in tree

Space Complexity : O(2n), n is Number of nodes in tree