

Experiment 3

Student Name: Khushmn Sangha UID: 22BCS14585
Branch: BE-CSE Section/Group: 602/A

Semester: 6th Date of Performance: 14-02-25 Subject Name: Advanced Programming - II Subject Code: 22CSP-351

Aim: To solve the following problems on Leetcode, with the goal of optimizing solutions in terms of time complexity and space efficiency:

- (I) 94.<u>Binary Tree Inorder Traversal</u>
- (II) 101.Symmetric Tree
- (III) 104. Maximum Depth of Binary Tree
- (IV) 98. Validate Binary Search Tree
- (V) 230.Kth Smallest Element in a BST
- (VI) 102. Binary Tree Level Order Traversal
- (VII) 107. Binary Tree Level Order Traversal II
- (VII) 103.Binary Tree Zigzag Level Order Traversal
- (VIII) 199. Binary Tree Right Side View
- (IX) 106. Construct Binary Tree from Inorder and Postorder Traversal
- (X) 513.Find Bottom Left Tree Value
- (XI) 124. Binary Tree Maximum Path Sum
- (XII) 987. Vertical Order Traversal of a Binary Tree
- 1. **Objective:** To efficiently solve a set of algorithmic problems on Leetcode, optimizing for time and space complexity, by implementing advanced techniques such as bit manipulation, binary search, sorting algorithms, and dynamic programming, with the goal of enhancing problem-solving skills and achieving optimal solutions for real-world applications.

2. Implementation/Code:

```
(VIII) Problem No. 94.Binary Tree Inorder Traversal
  Code: class Solution {
    public List<Integer>
    inorderTraversal(TreeNode root) {
```

```
List<Integer> ans = new
ArrayList<>();
             Deque<TreeNode> stack = new
ArrayDeque<>();
             while (root != null ||
!stack.isEmpty()) {
                   while (root != null) {
                          stack.push(root);
                          root = root.left;
                   root = stack.pop();
                   ans.add(root.val);
                   root = root.right;
             return ans;
    ♦ E Problem List 〈 〉 💢
                                                                                                                                                                                                                                        🖺 🕨 Run 🚹 Submit 🔯 🖵

■ Description | Submissions | Submissions | Submissions | Submissions | Description | Descripti
                                                                                                                                                                                                                                                                                     ⟨/> Code | ☑ Testcase | > Test Result
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   = 0002
   ← All Submissions
                                                                                                                                                                                                                                                                    0
                                                                                                                                                                                                                                                                                                                         uto

'ireeMoode(int val) { this.val = val; }

TreeMode(int val, TreeMode left, TreeMode right) {

this.val = val;

this.left = left;

this.right = right;
       Accepted 71 / 71 testcases passed
      M Khushmn Sangha submitted at Feb 14, 2025 11:54
                 O Runtime
                                                                                                                                        Memory
                 0 ms | Beats 100.00% 🗃
                                                                                                                                           41.76 MB | Beats 43.97%
                                                                                                                                                                                                                                                                                                                ss Solution {
public List(Integer> inorderTraversal(TreeNode root) {
   List(Integer> ans = new ArrayList();
Deque(TreeNode> stack = new ArrayDeque(>();
                                                   while (root != null || !stack.isEmpty()) {
  while (root != null) {
    stack.push(root);
    root = root.left;
}
                                                                                                                                                                                                                                                                                                                      root = stack.pop();
ans.add(root.val);
root = root.right;
      Code Java
              * Definition for a binary tree node.
                * public class TreeNode {
* int val;
               * TreeNode left;
```

(IX) Problem No. 101. Symmetric Tree Code: class Solution {

```
public boolean isSymmetric(TreeNode root) {
                    return isSymmetric(root, root);
private boolean isSymmetric(TreeNode p, TreeNode q) {
        if (p == null || q == null)
            return p == q;
        return p.val == q.val && isSymmetric(p.left, q.right) && isSymmetric(p.right, q.left);
♦ Froblem List < > >
                                                                                                                                                                                                      🍒 ▶ Run 🚹 Submit 🔯 🖵
                                                                                                                                                                                                                                                                                                                                                                                     88 ⊗ ∆ 13 2+ Invite

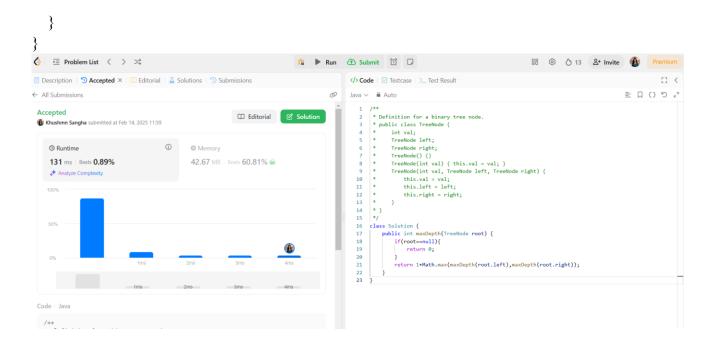
■ Description | S Accepted × | ■ Editorial | ■ Solutions | Submissions
                                                                                                                                                                                                                                             ⟨/> Code | ☑ Testcase | >_ Test Result
                                                                                                                                                                                                                                                                                                                                                                                                                                                            三口()り 27
                                                                                                                                                                                                                                                             * public class TreeNode {

* int val:
   Accepted 199 / 199 testcases passed
                                                                                                                                                                                                                                                                          plic class TreeMode {
int val;
TreeMode Left;
TreeMode right;
TreeMode() ()
TreeMode(int val) { this.val = val; }
TreeMode(int val, TreeMode Left, TreeMode(int val, TreeMode Left, TreeMo
                                                                                                                                                      ☐ Editorial ☑ Solution
   M Khushmn Sangha submitted at Feb 14, 2025 11:57
           0 ms | Beats 100.00% 🞳
                                                                                                                     42.16 MB | Beats 19.77%
                                                                                                                                                                                                                                                                    public boolean isSymmetric(TreeNode root) {
                                                                                                                                                                                                                                                                             return isSymmetric(root, root);
                                                                                                                                                                                                                                                                     rivate boolean isSymmetric(TreeNode p, TreeNode q) {
  if (p == null || q == null)
                                                                                                                                                                                                                                                                          eturn p.val == q.val && isSymmetric(p.left, q.right) && isSymmetric(p.right, q.left);
         * Definition for a binary tree node.
          * public class TreeNode {
* int val;
```

(X) Problem No. 104. Maximum Depth of Binary Tree
Code:
class Solution {
 public int maxDepth(TreeNode root) {
 if(root==null){
 return 0;
 }

TreeNode left;

return 1+Math.max(maxDepth(root.left),maxDepth(root.right));

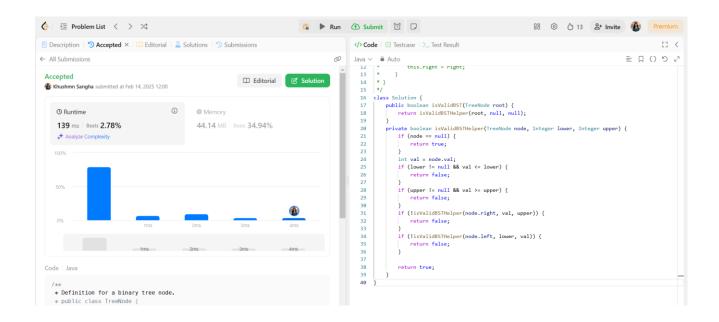


(XI) *Problem No.* 98. <u>Validate Binary Search Tree</u> *Code:*

```
class Solution {
   public boolean isValidBST(TreeNode root) {
      return isValidBSTHelper(root, null, null);
   }
   private boolean isValidBSTHelper(TreeNode node, Integer lower, Integer upper) {
      if (node == null) {
        return true;
      }
      int val = node.val;
      if (lower != null && val <= lower) {
        return false;
      }
      if (upper != null && val >= upper) {
        return false;
      }
      if (!isValidBSTHelper(node.right, val, upper)) {
        return false;
      }
    }
}
```

```
if (!isValidBSTHelper(node.left, lower, val)) {
    return false;
}

return true;
}
```



(XII) *Problem No.* 230. Kth Smallest Element in a BST *Code*:

```
class Solution {
  public int kthSmallest(TreeNode root, int k) {
    final int leftCount = countNodes(root.left);

  if (leftCount == k - 1)
    return root.val;
  if (leftCount >= k)
    return kthSmallest(root.left, k);
  return kthSmallest(root.right, k - 1 - leftCount); // leftCount < k
}

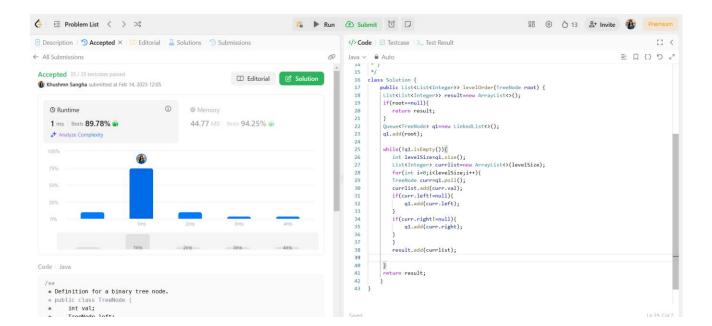
private int countNodes(TreeNode root) {</pre>
```

```
if (root == null)
       return 0;
    return 1 + countNodes(root.left) + countNodes(root.right);
♦ E Problem List 〈 > >
                                                                                                                      🏗 ▶ Run 街 Submit 🔯 🔘
                                                                                                                                                                                                                             98 ② ♦ 13 &+ Invite 1
                                                                                                                                                                                                                                                                       "<sub>3</sub> C'() □ ≡
                                                                                                                                                              Treelode right;
Treelode() ()
Treelode() ()
Treelode(int val) ( this.val = val; )
Treelode(int val), Treelode left, Treelothis.val = val;
this.val = val;
this.left = left;
this.right = right;
  Accepted
                                                                                                                                                                                                                  eNode right) {
                                                                     Memory
      O Runtime
       131 ms | Beats 4.29%
                                                                     44.24 MB Beats 84.70% 🐠
                                                                                                                                                      class Solution {
    public int kthSmallest(TreeNode root, int k) {
        final int leftCount = countNodes(root.left);
}
                                                                                                                                                         if (leftCount == k - 1)
    return noot.val;
if (leftCount >= k)
    return kthSmallest(root.left, k);
    return kthSmallest(root.right, k - 1 - leftCount); // leftCount < k</pre>
                                                                                                                                                        private int countlodes(TreeNode root) {
   if (root == null)
        return 0;
   return 1 + countNodes(root.left) + countNodes(root.right);
 Code Java
    /**
 * Definition for a binary tree node.
     * public class TreeNode {
* int val;
              TreeNode left:
```

Code: class Solution { public List<List<Integer>> levelOrder(TreeNode root) { List<List<Integer>> result=new ArrayList<>(); if(root==null){ return result; } Queue<TreeNode> q1=new LinkedList<>(); q1.add(root);

(XIII) Problem No.: 102. Binary Tree Level Order Traversal

```
while(!q1.isEmpty()){
 int levelSize=q1.size();
 List<Integer> currlist=new ArrayList<>(levelSize);
 for(int i=0;i<levelSize;i++){</pre>
 TreeNode curr=q1.poll();
 currlist.add(curr.val);
 if(curr.left!=null){
    q1.add(curr.left);
  }
 if(curr.right!=null){
    q1.add(curr.right);
 result.add(currlist);
}
return result;
```

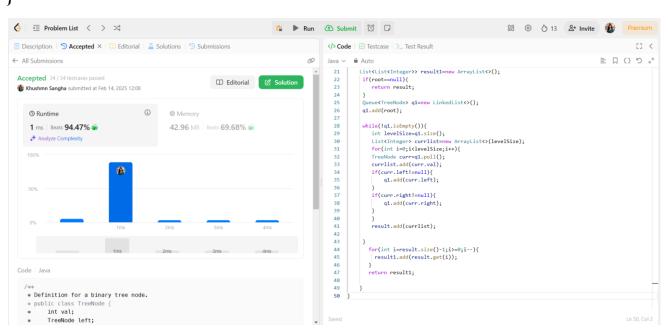


```
(XIV) Problem No. 107.Binary Tree Level Order Traversal II
Code: class Solution {
   public List<List<Integer>>
levelOrderBottom(TreeNode root) {

   List<List<Integer>> result=new
ArrayList<>();
   List<List<Integer>> result1=new
ArrayList<>();
   if(root==null) {
      return result;
   }
}
```

```
Queue<TreeNode> q1=new
LinkedList<>();
   q1.add(root);
   while(!q1.isEmpty()){
    int levelSize=q1.size();
    List<Integer> currlist=new
ArrayList<>(levelSize);
     for(int i=0;i<levelSize;i++){</pre>
    TreeNode curr=q1.poll();
    currlist.add(curr.val);
    if(curr.left!=null){
       q1.add(curr.left);
    if(curr.right!=null){
       q1.add(curr.right);
    result.add(currlist);
    for(int i=result.size()-1;i>=0;i--){
     result1.add(result.get(i));
    return result1;
```

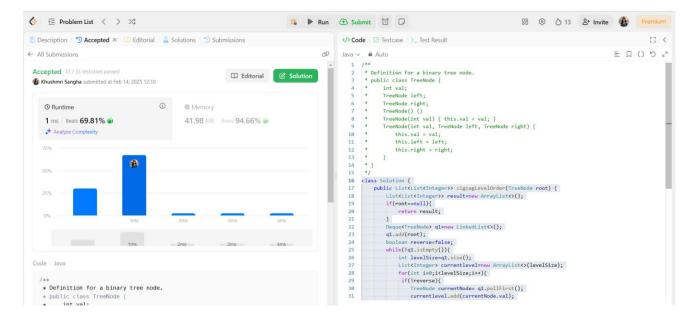
} }



```
(XIII) Prolem No. 103.Binary Tree Zigzag Level Order Traversal
Code: class Solution {
   public List<List<Integer>>
zigzagLevelOrder(TreeNode root) {
    List<List<Integer>> result=new ArrayList<>();
   if(root==null) {
      return result;
   }
   Deque<TreeNode> q1=new LinkedList<>();
   q1.add(root);
   boolean reverse=false;
```

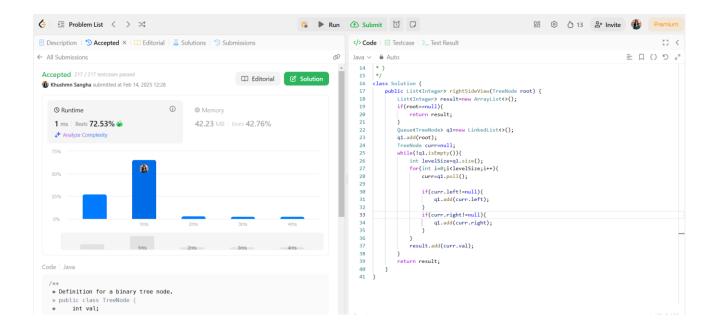
```
while(!q1.isEmpty()){
       int levelSize=q1.size();
       List<Integer> currentlevel=new
ArrayList<>(levelSize);
       for(int i=0;i<levelSize;i++){
       if(!reverse){
         TreeNode currentNode= q1.pollFirst();
         currentlevel.add(currentNode.val);
         if(currentNode.left!=null){
            q1.addLast(currentNode.left);
         }if(currentNode.right!=null){
            q1.addLast(currentNode.right);
         }
        }else{
        TreeNode currentNode= q1.pollLast();
       currentlevel.add(currentNode.val);
       if(currentNode.right!=null){
            q1.addFirst(currentNode.right);
       }if(currentNode.left!=null){
            q1.addFirst(currentNode.left);
         }
```

```
result.add(currentlevel);reverse=!reverse;
}
return result;
}
```



```
(XIV) Problem No. 199.Binary Tree Right Side View
Code:
class Solution {
  public List<Integer> rightSideView(TreeNode root) {
    List<Integer> result=new ArrayList<>();
    if(root==null) {
      return result;
    }
    Queue<TreeNode> q1=new LinkedList<>();
```

```
q1.add(root);
     TreeNode curr=null;
     while(!q1.isEmpty()){
       int levelSize=q1.size();
       for(int i=0;i<levelSize;i++){
          curr=q1.poll();
          if(curr.left!=null){
            q1.add(curr.left);
          }
          if(curr.right!=null){
            q1.add(curr.right);
          }
       }
       result.add(curr.val);
    return result;
}
```



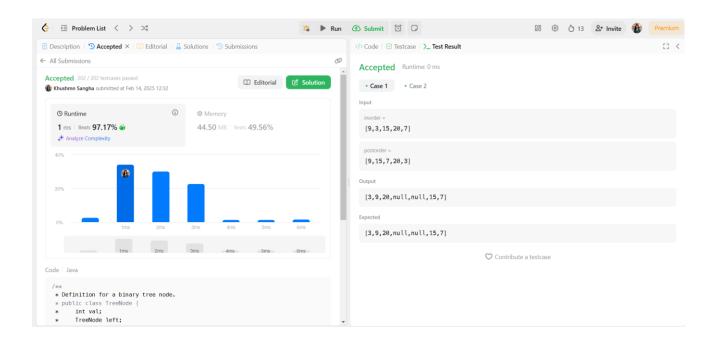
```
(XV) Problem No. 106.Construct Binary Tree from Inorder and Postorder Traversal
Code: class Solution {
    private int postIndex;
    private Map<Integer, Integer>
inorderMap;

public TreeNode buildTree(int[]
inorder, int[] postorder) {
    postIndex = postorder.length - 1;
    inorderMap = new HashMap<>();

for (int i = 0; i < inorder.length;
i++) {</pre>
```

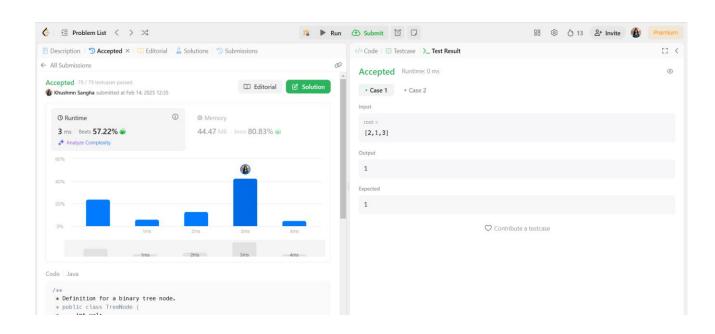
```
inorderMap.put(inorder[i], i);
    return constructTree(postorder, 0,
inorder.length - 1);
  }
  private TreeNode constructTree(int[]
postorder, int left, int right) {
    if (left > right) return null;
    int rootValue =
postorder[postIndex--];
     TreeNode root = new
TreeNode(rootValue);
    int inorderIndex =
inorderMap.get(rootValue);
    root.right =
constructTree(postorder, inorderIndex
+ 1, right);
    root.left =
constructTree(postorder, left,
inorderIndex - 1);
    return root;
```

```
}
```



```
(XVI) Problem No. 513.Find Bottom Left Tree Value
Code: class Solution {
  public int
findBottomLeftValue(TreeNode root) {
    Queue<TreeNode> q = new
ArrayDeque<>();
    q.offer(root);
    int ans = 0;
    while (!q.isEmpty()) {
        ans = q.peek().val;
        for (int i = q.size(); i > 0; --i) {
```

```
TreeNode node = q.poll();
    if (node.left != null) {
        q.offer(node.left);
    }
    if (node.right != null) {
        q.offer(node.right);
    }
}
return ans;
}
```



```
(XVII) Problem No. 124. Binary Tree Maximum Path Sum
Code:
class Solution {
  public int maxPathSum(TreeNode root)
{
  maxPathSumDownFrom(root);
  return ans;
private int ans = Integer.MIN_VALUE;
  private int
maxPathSumDownFrom(TreeNode root)
{
  if(root == null)
   return 0;
  final int 1 =
Math.max(maxPathSumDownFrom(root.l
eft), 0);
  final int r =
Math.max(maxPathSumDownFrom(root.r
ight), 0);
  ans = Math.max(ans, root.val + 1 + r);
  return root.val + Math.max(l, r);
```

```
♦ E Problem List < > >
                                                                                                                                                                                    🖺 🕨 Run 🚹 Submit 🔯 🔘
⟨/> Code | ☑ Testcase | >_ Test Result
                                                                                                                                                                                                                       三口()口 三
← All Submissions
                                                                                                                                  TreeNode() {}
 Accepted 96 / 96 testcases passed
                                                                                                                                 TreeMode() {}
TreeMode(int val) { this.val = val; }
TreeMode(int val, TreeMode left, TreeMode left, TreeMode this.val = val;
this.left = left;
this.right = right;
                                                                        (b) Khushmn Sangha submitted at Feb 14, 2025 12:41
                                                                                                                                                                           eNode right) {
                                                                                                                      " this...
" }
" }
" class Solution {
public int maxPathSum(TreeHode root) {
| maxPathSumOonFrom(root);
return ans;
     © Runtime ③
                                                       Memory
                                                                                                                    13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
     1 ms | Beats 11.98%
                                                        44.19 MB | Beats 87.50% 🞳
     * Analyze Complexity
                                                                                                                              private int maxPathSumDownFrom(TreeNode root) {
                                                                                                                               if (root == null)
return 0;
                                                                                                                              final int 1 = Math.max(maxPathSumDownFrom(root.left), 0);
final int r = Math.max(maxPathSumDownFrom(root.right), 0);
     * Definition for a binary tree node.
     * public class TreeNode {
            int val:
```

```
(XVIII) Problem No. 987: .Vertical Order Traversal of a Binary Tree

Code: class Solution {
    public List<List<Integer>>
    verticalTraversal(TreeNode root) {
        // TreeMap to store nodes by
    vertical level
        TreeMap<Integer,
    TreeMap<Integer,
PriorityQueue<Integer>>> map =
    new TreeMap<();
        Queue<Tuple> queue = new

LinkedList<>();
```

```
// Start BFS
    queue.offer(new Tuple(root, 0,
0));
    while (!queue.isEmpty()) {
       Tuple tuple = queue.poll();
       TreeNode node = tuple.node;
       int x = tuple.x; // Vertical
level
       int y = tuple.y; // Horizontal
level
       // Add node value to TreeMap
       map.putIfAbsent(x, new
TreeMap<>());
       map.get(x).putIfAbsent(y,
new PriorityQueue<>());
       map.get(x).get(y).offer(node.
val);
       // Traverse left and right
children
       if (node.left != null) {
         queue.offer(new
Tuple(node.left, x - 1, y + 1);
       }
```

```
if (node.right != null) {
         queue.offer(new
Tuple(node.right, x + 1, y + 1));
    // Prepare result list
    List<List<Integer>> result =
new ArrayList<>();
    for (TreeMap<Integer,
PriorityQueue<Integer>> vertical:
map.values()) {
       List<Integer> verticalList =
new ArrayList<>();
       for (PriorityQueue<Integer>
nodes : vertical.values()) {
         while (!nodes.isEmpty()) {
            verticalList.add(nodes.p
oll());
       result.add(verticalList);
    return result;
  }
```

```
class Tuple {
    TreeNode node;
    int x; // Vertical level
    int y; // Horizontal level

    public Tuple(TreeNode node, int
x, int y) {
        this.node = node;
        this.x = x;
        this.y = y;
    }
}
```

```
♦ Problem List < > >
                                                                                                                                                   🖺 🕨 Run 🚹 Submit 🔯 🗔

■ Description | Saccepted × | □ Editorial | ■ Solutions | Submissions
← All Submissions
                                                                                                                                                                     0
                                                                                                                                                                                                                                                                                                                                          E □ () □ ≡
                                                                                                                                                                                           /**

* Definition for a binary tree node.

* public class TreeNode {

* int val;

* TreeNode left;

* TreeNode right;

* TreeNode(int val) { this.val = val; }

* TreeNode(int val) { this.val = val; }

* TreeNode(int val, TreeNode left, TreeNode right) {

* this.val = val;

* this.right = left;

* this.right = right;

* }
   Accepted 34 / 34 testcases passed
  Khushmn Sangha submitted at Feb 14, 2025 12:44
        O Runtime
        3 ms | Beats 82.70% 🞳
                                                                                        42.75 MB | Beats 34.13%
         Analyze Complexity
                                                                                                                                                                                   10
11
12
13
14
15
16
17
                                                                                                                                                                                            class Solution {
                                                                                                                                                                                                  public List<Lint<Integer>> verticalTraversal(TreeHode root) {
   // TreeMap to store nodes by vertical level
   TreeMap<Integer, TreeMap<Integer, PriorityQueue(Integer>>> map = new TreeMap<\();
   Queue(Tupla> queue = new LinkedList<\();
}</pre>
                                                                                                                                                                                                         queue.offer(new Tuple(root, 0, 0));
                                                                                                                                                                                                         while (!queue.isEmpty()) {
                                                                                                                                                                                                               Tuple tuple = queue.poll();
TreeNode node = tuple.node;
int x = tuple.x; // Vertical level
int y = tuple.y; // Horizontal level
                                                                                                                                                     0 2
       * Definition for a binary tree node.
       * public class TreeNode {
* int val;
```

