Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

45% completed

Search Course

Solution Review: Problem Challenge 1
Problem Challenge 2
Solution Review: Problem Challenge 2

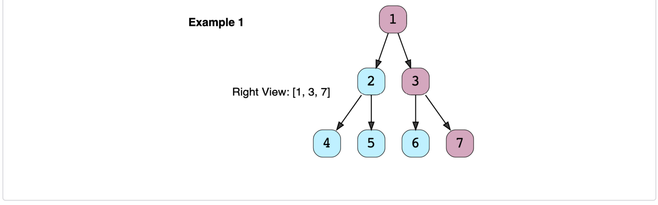
# Solution Review: Problem Challenge 2

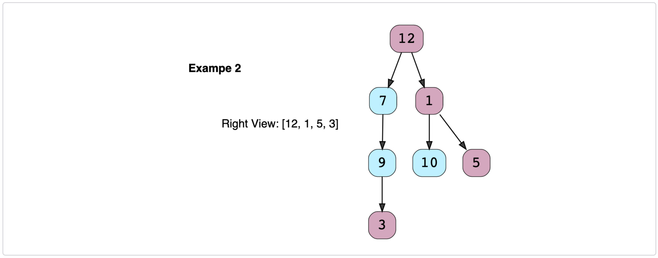
We'll cover the following



## Right View of a Binary Tree (easy)

Given a binary tree, return an array containing nodes in its right view. The right view of a binary tree is the set of **nodes visible when the tree is seen from the right side**.




## Solution

This problem follows the Binary Tree Level Order Traversal pattern. We can follow the same **BFS** approach. The only additional thing we will be doing is to append the last node of each level to the result array.

## Code

Here is what our algorithm will look like; only the highlighted lines have changed:

Java | Python3 | C++ | JS

```
26          if (i == levelSize - 1)
27            result.add(currentNode);
28          // insert the children of current node in the queue
29          if (currentNode.left != null)
30            queue.offer(currentNode.left);
31          if (currentNode.right != null)
32            queue.offer(currentNode.right);
33        }
34      }
35
36      return result;
37    }
38
39    public static void main(String[] args) {
40      TreeNode root = new TreeNode(12);
41      root.left = new TreeNode(7);
42      root.right = new TreeNode(1);
43      root.left.left = new TreeNode(9);
44      root.right.left = new TreeNode(10);
45      root.right.right = new TreeNode(5);
46      root.left.left.left = new TreeNode(3);
47      List<TreeNode> result = RightViewTree.traverse(root);
48      for (TreeNode node : result) {
49        System.out.print(node.val + " ");
50      }
51    }
52  }
53
```

Run | Save | Reset

## Time complexity

The time complexity of the above algorithm is $O(N)$, where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

## Space complexity

The space complexity of the above algorithm will be $O(N)$ needed for the return list. We will also need $O(N)$ space for the queue. Since we can have a maximum of $N/2$ nodes at any level (this could happen only at the lowest level), therefore we will need $O(N)$ space to store them in the queue.

## Similar Questions

**Problem 1:** Given a binary tree, return an array containing nodes in its left view. The left view of a binary tree is the set of nodes visible when the tree is seen from the left side.

**Solution:** We will be following a similar approach, but instead of appending the last element of each level, we will be appending the first element of each level to the output array.