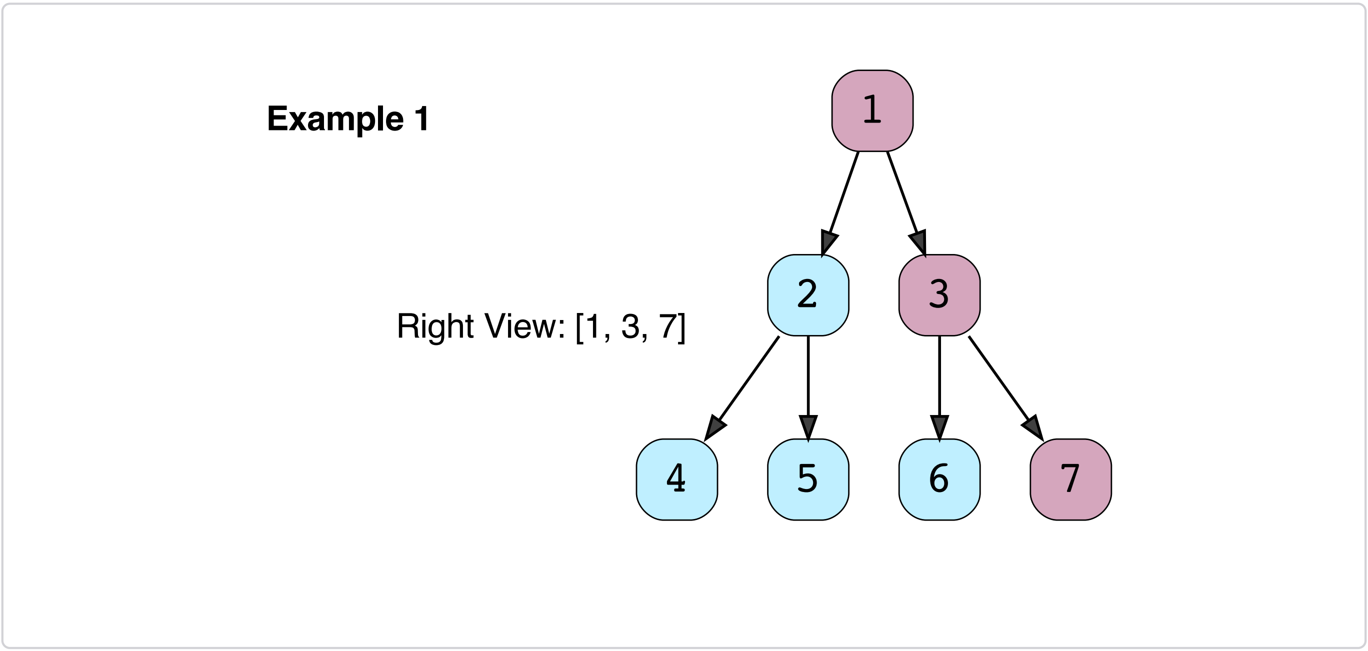
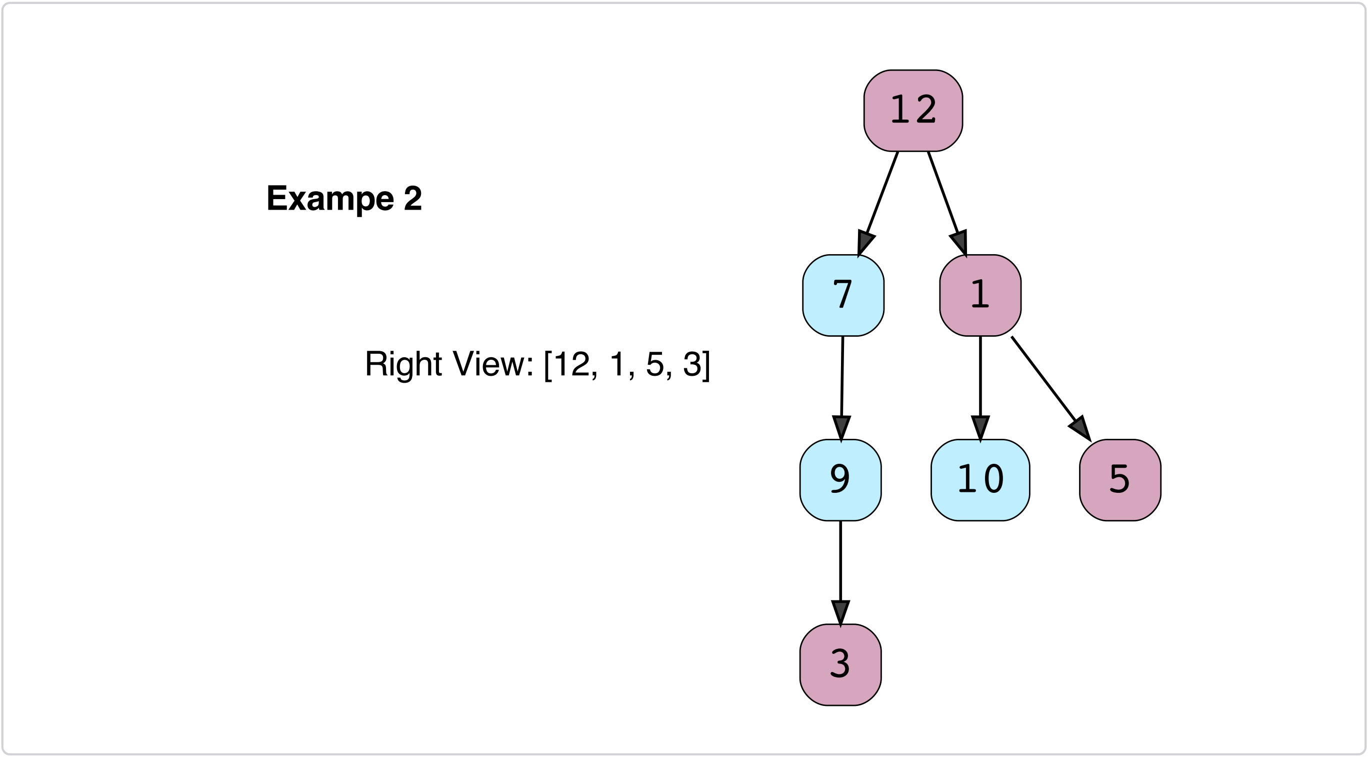
Solution Review: Problem Challenge 2

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**.


Example 1
Right View: [1, 3, 7]
1
2
3
4
5
6
7


Exampe 2
Right View: [12, 1, 5, 3]
12
7
1
9
10
5
3

Solution#

This problem follows the Binary Tree Level Order Traversal (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5726607939469312/) 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

```
const Deque = require('./collections/deque'); //http://www.collectionsjs.c


class TreeNode {
  constructor(val) {
    this.val = val;
    this.left = null;
    this.right = null;
  }
}


function tree_right_view(root) {
  result = [];
  if (root === null) {
    return result;
  }

  const queue = new Deque();
  queue.push(root);
  while (queue.length > 0) {
    const levelSize = queue.length;
    for (i = 0; i < levelSize; i++) {
      currentNode = queue.shift();
      // if it is the last node of this level, add it to the result
      if (i === levelSize - 1) {
        result.push(currentNode);
      }
```

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 $O(N)$ as we need to return a list containing the level order traversal. 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.

← Back
Problem Challenge 2

Next →
Introduction

✓ Mark as Completed

Report an Issue