Grokking the Coding Interview: Patterns for Coding Questions

30% completed

Search Course

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

Pattern: Tree Breadth First Search

- Introduction
- Binary Tree Level Order Traversal (easy)
- Reverse Level Order Traversal (easy)
- Zigzag Traversal (medium)
- Level Averages in a Binary Tree (easy)
- Minimum Depth of a Binary Tree (easy)
- Level Order Successor (easy)
- Connect Level Order Siblings (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Tree Depth First Search

- Introduction
- Binary Tree Path Sum (easy)
- All Paths for a Sum (medium)
- Sum of Path Numbers (medium)
- Path With Given Sequence (medium)
- Count Paths for a Sum (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Two Heaps

- Introduction
- Find the Median of a Number Stream (medium)
- Sliding Window Median (hard)
- Maximize Capital (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

Pattern: Subsets

- Introduction
- Subsets (easy)
- Subsets With Duplicates (easy)
- Permutations (medium)
- String Permutations by changing

Reverse Level Order Traversal (easy)

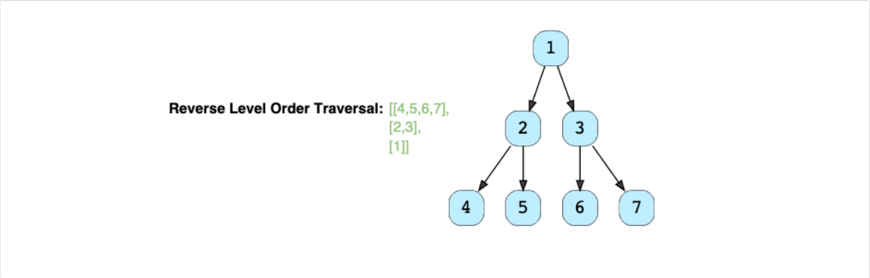
We'll cover the following

- Problem Statement
- Try it yourself
- Solution
- Code
  - Time complexity
  - Space complexity

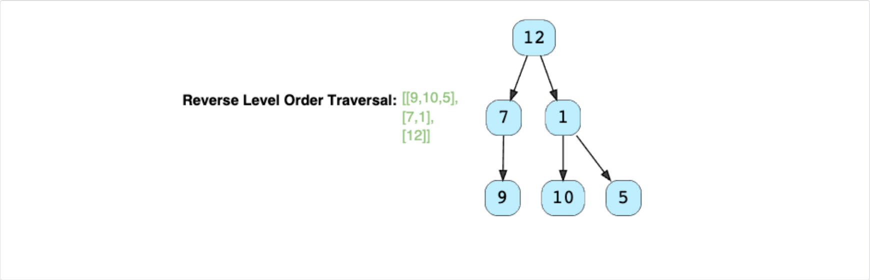
Problem Statement

Given a binary tree, populate an array to represent its level-by-level traversal in reverse order, i.e., the **lowest level comes first**. You should populate the values of all nodes in each level from left to right in separate sub-arrays.

Example 1:



Example 2:



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
class TreeNode {

  constructor(value) {
    this.value = value;
    this.left = null;
    this.right = null;
  }
};

const traverse = function(root) {
  result = [];
  // TODO: Write your code here
  return result;
}

var root = new TreeNode(12)
root.left = new TreeNode(7)
root.right = new TreeNode(1)
root.left.left = new TreeNode(9)
root.right.left = new TreeNode(10)
root.right.right = new TreeNode(5)
console.log(`Reverse level order traversal: ${traverse(root)}`)
```

RUN | SAVE | RESET

case (medium)
Balanced Parentheses (hard)
Unique Generalized Abbreviations (hard)
Problem Challenge 1
Solution Review: Problem Challenge 1
Problem Challenge 2
Solution Review: Problem Challenge 2
Problem Challenge 3
Solution Review: Problem Challenge 3

Pattern: Modified Binary Search

Introduction
Order-agnostic Binary Search (easy)
Ceiling of a Number (medium)
Next Letter (medium)
Number Range (medium)
Search in a Sorted Infinite Array (medium)
Minimum Difference Element (medium)
Bitonic Array Maximum (easy)
Problem Challenge 1
Solution Review: Problem Challenge 1
Problem Challenge 2
Solution Review: Problem Challenge 2
Problem Challenge 3
Solution Review: Problem Challenge 3

Pattern: Bitwise XOR

Introduction
Single Number (easy)
Two Single Numbers (medium)
Complement of Base 10 Number (medium)
Problem Challenge 1
Solution Review: Problem Challenge 1

Pattern: Top 'K' Elements

Introduction
Top 'K' Numbers (easy)
Kth Smallest Number (easy)
'K' Closest Points to the Origin (easy)
Connect Ropes (easy)
Top 'K' Frequent Numbers (medium)
Frequency Sort (medium)
Kth Largest Number in a Stream (medium)
'K' Closest Numbers (medium)
Maximum Distinct Elements (medium)
Sum of Elements (medium)
Rearrange String (hard)
Problem Challenge 1
Solution Review: Problem Challenge 1
Problem Challenge 2

MW
Explore
Tracks
My Courses
Edpresso
Refer a Friend
Create

Solution

This problem follows the Binary Tree Level Order Traversal pattern. We can follow the same **BFS** approach. The only difference will be that instead of appending the current level at the end, we will append the current level at the beginning of the result list.

Code

Here is what our algorithm will look like; only the highlighted lines have changed. Please note that, for **Java**, we will use a `LinkedList` instead of an `ArrayList` for our result list. As in the case of `ArrayList`, appending an element at the beginning means shifting all the existing elements. Since we need to append the level array at the beginning of the result list, a `LinkedList` will be better, as this shifting of elements is not required in a `LinkedList`. Similarly, we will use a double-ended queue (deque) for **Python**, **C++**, and **JavaScript**.

Java | Python3 | C++ | JS

```
  while (queue.length > 0) {
    levelSize = queue.length;
    currentLevel = [];
    for (i = 0; i < levelSize; i++) {
      currentNode = queue.shift();
      // add the node to the current level
      currentLevel.push(currentNode.val);
      // insert the children of current node in the queue
      if (currentNode.left !== null) {
        queue.push(currentNode.left);
      }
      if (currentNode.right !== null) {
        queue.push(currentNode.right);
      }
    }
    result.unshift(currentLevel);
  }
  return result;
}


const root = new TreeNode(12);
root.left = new TreeNode(7);
root.right = new TreeNode(1);
root.left.left = new TreeNode(9);
root.right.left = new TreeNode(10);
root.right.right = new TreeNode(5);
console.log(`Reverse level order traversal: ${traverse(root).toArray()}`);
```

RUN SAVE RESET

Close

Output 5.265s

```
Reverse level order traversal: 9,10,5,7,1,12
```

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.

☑ MARK AS COMPLETED

← Back
Binary Tree Level Order Traversal (easy)

Next →
Zigzag Traversal (medium)

Report an Issue