Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

44% completed

Search Course

Connect Level Order Siblings (medium)
Problem Challenge 1
Solution Review: Problem Challenge 1

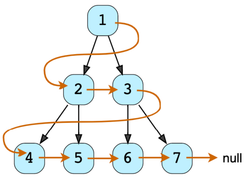
Solution Review: Problem Challenge 1

We'll cover the following

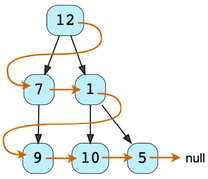
Connect All Level Order Siblings (medium)

Given a binary tree, connect each node with its level order successor. The last node of each level should point to the first node of the next level.

Example 1:



Example 2:



Solution

This problem follows the Binary Tree Level Order Traversal pattern. We can follow the same **BFS** approach. The only difference will be that while traversing we will remember (irrespective of the level) the previous node to connect it with the current node.

Code

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

Java | Python3 | C++ | JS

```
import java.util.*;

class TreeNode {
  int val;
  TreeNode left;
  TreeNode right;
  TreeNode next;

  TreeNode(int x) {
    val = x;
    left = right = next = null;
  }

  // tree traversal using 'next' pointer
  public void printTree() {
    TreeNode current = this;
    System.out.print("Traversal using 'next' pointer: ");
    while (current != null) {
      System.out.print(current.val + " ");
      current = current.next;
    }
  }
};

class ConnectAllSiblings {
  public static void connect(TreeNode root) {
    if (root == null)
      return;
```

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 $O(N)$ which is required 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.

Interviewing soon? We've partnered with Hired so that companies apply to you, instead of the other way around. See how