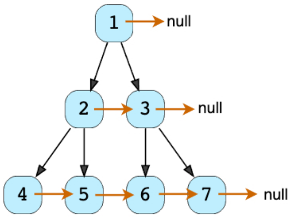
Connect Level Order Siblings (medium)

We'll cover the following

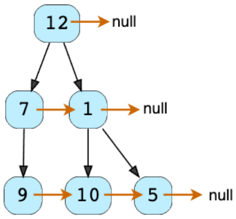
Problem Statement

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

Example 1:



Example 2:



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
import java.util.*;

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

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

  // level order traversal using 'next' pointer
  void printLevelOrder() {
    TreeNode nextLevelRoot = this;
    while (nextLevelRoot != null) {
      TreeNode current = nextLevelRoot;
      nextLevelRoot = null;
```

```java
      while (current != null) {
        System.out.print(current.val + " ");
        if (nextLevelRoot == null) {
          if (current.left != null)
            nextLevelRoot = current.left;
          else if (current.right != null)
            nextLevelRoot = current.right;
        }
        current = current.next;
```

Run Save Reset

Solution

This problem follows the Binary Tree Level Order Traversal pattern. We can follow the same **BFS** approach. The only difference is that while traversing a level we will remember 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

```java
import java.util.*;

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

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

  // level order traversal using 'next' pointer
  public void printLevelOrder() {
    TreeNode nextLevelRoot = this;
    while (nextLevelRoot != null) {
      TreeNode current = nextLevelRoot;
      nextLevelRoot = null;
      while (current != null) {
        System.out.print(current.val + " ");
        if (nextLevelRoot == null) {
          if (current.left != null)
            nextLevelRoot = current.left;
          else if (current.right != null)
            nextLevelRoot = current.right;
        }
        current = current.next;
```

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.

← Back
Level Order Successor (easy)

Next →
Problem Challenge 1

☑ Mark as Completed

Report an Issue Ask a Question