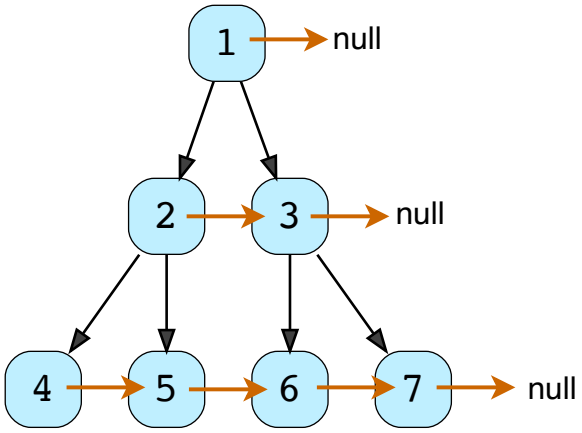
Connect Level Order Siblings (medium)

We'll cover the following

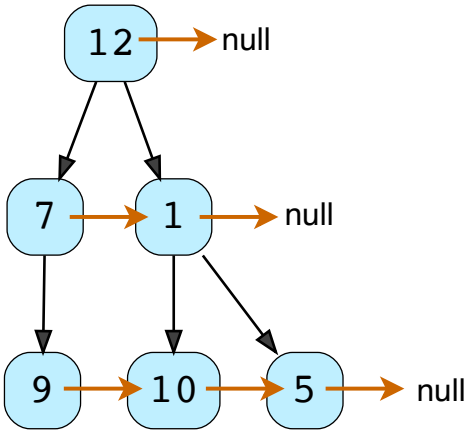
Problem Statement

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++

```
19            if current.left:
20              nextLevelRoot = current.left
21            elif current.right:
```

```
            nextLevelRoot = current.right
          current = current.next
        print()


def connect_level_order_siblings(root):
  # TODO: Write your code here
  queue = [root,]
  while queue:
    prevNode = None
    size = len(queue)
    for _ in range(size):
      curr = queue.pop(0)
      if prevNode:
        prevNode.next = curr
      prevNode = curr
      if curr.left:
        queue.append(curr.left)
      if curr.right:
        queue.append(curr.right)
  return root

def main():
  root = TreeNode(12)
  root.left = TreeNode(7)
```

Output 0.33s

```
Level order traversal using 'next' pointer:
12
7 1
9 10 5
```

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 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

```
            nextLevelRoot = current.left
          elif current.right:
            nextLevelRoot = current.right
        current = current.next
      print()


def connect_level_order_siblings(root):
  if root is None:
    return

  queue = deque()
  queue.append(root)
  while queue:
    previousNode = None
    levelSize = len(queue)
    # connect all nodes of this level
    for _ in range(levelSize):
      currentNode = queue.popleft()
      if previousNode:
        previousNode.next = currentNode
      previousNode = currentNode

      # insert the children of current node in t
      if currentNode.left:
        queue.append(currentNode.left)
      if currentNode.right:
        queue.append(currentNode.right)
```

```
Output                                                    0.23s

 Level order traversal using 'next' pointer:
 12
 7 1
 9 10 5
```

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

Ask a Question

Report