

## Connect Level Order Siblings (medium)

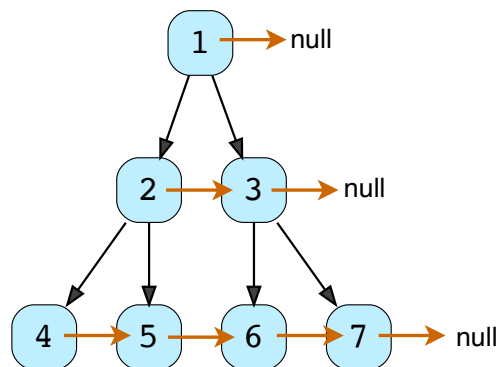
We'll cover the following ^

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

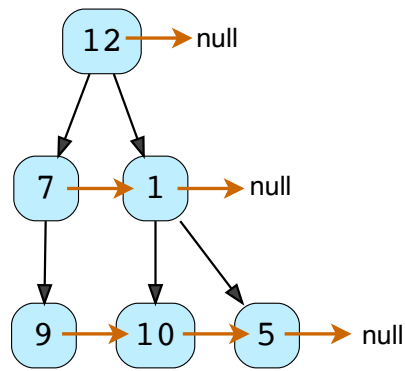
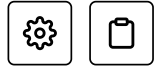
### Problem Statement #

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

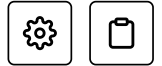
Python3

JS

C++

```
1 from __future__ import print_function
2 from collections import deque
3
4
5 class TreeNode:
6     def __init__(self, val):
7         self.val = val
8         self.left, self.right, self.next = None, None, None
9
10 # level order traversal using 'next' pointer
11 def print_level_order(self):
12     nextLevelRoot = self
13     while nextLevelRoot:
14         current = nextLevelRoot
15         nextLevelRoot = None
16         while current:
17             print(str(current.val) + " ", end='')
18             if not nextLevelRoot:
19                 if current.left:
20                     nextLevelRoot = current.left
21                 elif current.right:
22                     nextLevelRoot = current.right
23             current = current.next
24         print()
25
26
27 def connect_level_order_siblings(root):
28     # TODO: Write your code here
```





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

```
37     for _ in range(levelSize):
38         currentNode = queue.popleft()
39         if previousNode:
40             previousNode.next = currentNode
41             previousNode = currentNode
42
43         # insert the children of current node in the queue
44         if currentNode.left:
45             queue.append(currentNode.left)
46         if currentNode.right:
47             queue.append(currentNode.right)
48
49
50 def main():
51     root = TreeNode(12)
52     root.left = TreeNode(7)
53     root.right = TreeNode(1)
54     root.left.left = TreeNode(9)
55     root.right.left = TreeNode(10)
56     root.right.right = TreeNode(5)
57     connect_level_order_siblings(root)
58
59     print("Level order traversal using 'next' pointer: ")
60     root.print_level_order()
61
62
63 main()
64
```

×

Output1.484s

Level order traversal using 'next' pointer:  
12  
7 1



### 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](#)[Next →](#)

Level Order Successor (easy)

Problem Challenge 1



**Mark as Completed**



Report  
an Issue



Ask a Question

([https://discuss.educative.io/tag/connect-level-order-siblings-medium\\_\\_pattern-tree-breadth-first-search\\_\\_grokking-the-coding-interview-patterns-for-coding-questions](https://discuss.educative.io/tag/connect-level-order-siblings-medium__pattern-tree-breadth-first-search__grokking-the-coding-interview-patterns-for-coding-questions))