medium.com

# LeetCode : Populating Next Right Pointers in Each Node I and II — [Medium]

*Pruthvik Reddy*

3-4 minutes

---

Problem Link:

## Problem Description:

You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to `NULL`.

Initially, all next pointers are set to `NULL`.

**Example 1:**

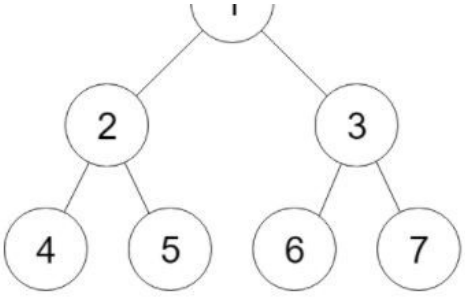

Figure A

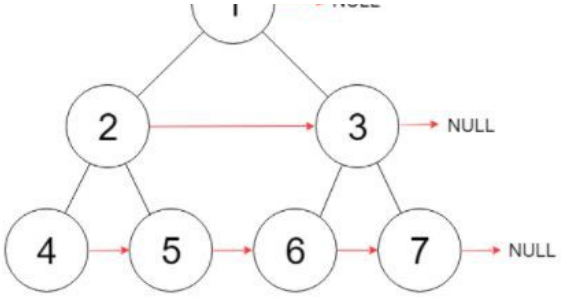

Figure B

```
Input: root = [1,2,3,4,5,6,7]
Output: [1,#,2,3,#,4,5,6,7,#]
Explanation: Given the above perfect binary tree (Figure A), your function should
populate each next pointer to point to its next right node, just like in Figure B. The
serialized output is in level order as connected by the next pointers, with '#'
signifying the end of each level.
```

**Constraints:**

- The number of nodes in the given tree is less than `4096`.
- `-1000 <= node.val <= 1000`

## Solution

If you look at the example closely, we are just manipulating the next pointer of nodes at every level in the tree. So, basically a level order traversal problem. Once you know how to get the nodes in a sequence from left to right, the solution is pretty straightforward.

How do you get nodes of binary tree level wise? If you know bfs traversal, then it is only a slight extension to the bfs. Using the iterative approach, count the number of nodes at current level, append the children of the node to the queue. Also while dequeueing the node from the queue add it to a list (named “rows” in code) so that all nodes at a level are stored in the list “rows”

Time Complexity : O(n)

## Follow up:

Can it be done using constant extra space? Yes. The following question is similar to this one and the solution for the follow up question is discussed there….[keep Reading]

## Populating Next Right Pointers in Each Node II

Problem Link

## Problem Description:

Given a binary tree

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to `NULL`.

Initially, all next pointers are set to `NULL`.

**Example 1:**

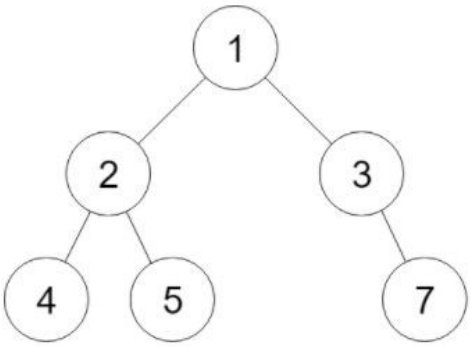

Figure A

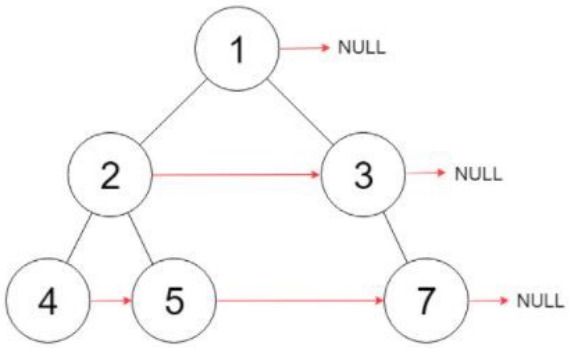

Figure B

```
Input: root = [1,2,3,4,5,null,7]
Output: [1,#,2,3,#,4,5,7,#]
Explanation: Given the above binary tree (Figure A), your function should populate each
next pointer to point to its next right node, just like in Figure B. The serialized
```

```
output is in level order as connected by the next pointers, with '#' signifying the end
of each level.
```

**Constraints:**

- The number of nodes in the given tree is less than `6000`.
- `-100 <= node.val <= 100`

## Solution

It is similar to the question in the first part of the post. The same code also works as we have not restricted our code to perfect binary tree. So the code provided above works.

## Follow up:

Can it be done using constant extra space? Yes.

To do the problem without using queue, we first need to come up with a different logic which doesn't involve storing nodes in a list.

1. At each level, we want all the nodes at that level
2. Once we get all nodes at a level, we move to the next level

   So two while loops. One loops for moving to next level and other inner while loop to traverse all nodes at that level.

   We make use of two pointers childhead and child which are NULL at the start. We use childhead pointer to move to the next level i.e we assign it to root so that the root moves to the first node at the next level.

   It will be slightly difficult to understand the code without understanding the logic. So I suggest you to consider an example and frame the logic first before heading to the code.