116. Populating Next Right Pointers in Each Node

Solved ✓

Medium Topics Companies

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:

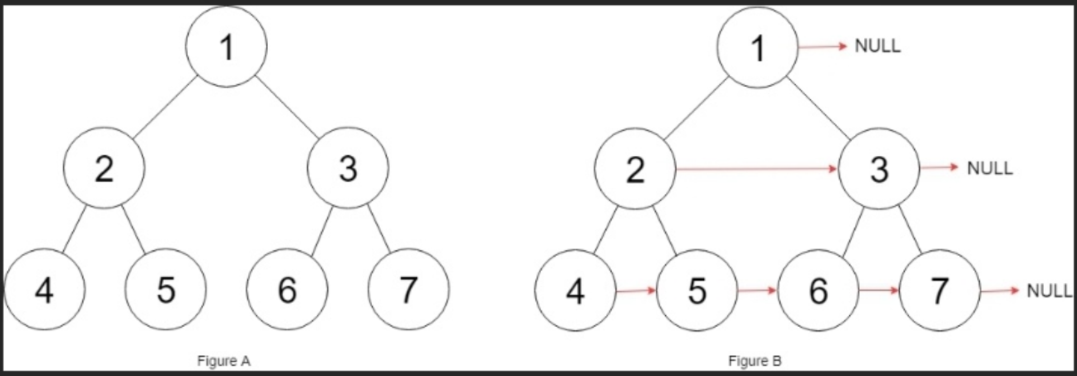


Figure A

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.
```

Example 2:

```
Input: root = []
Output: []
```

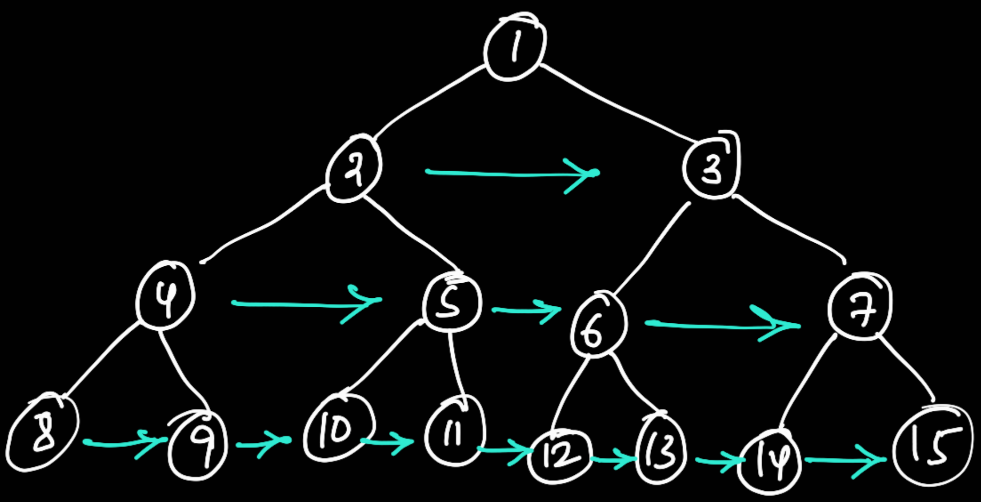
Constraints:

- The number of nodes in the tree is in the range `[0, 2^12 - 1]`.
- `-1000 <= Node.val <= 1000`

Follow-up:

- You may only use constant extra space.
- The recursive approach is fine. You may assume implicit stack space does not count as extra space for this problem.

Approach 1: BFS



level	links
0	0
1	1
2	3
3	7

The no. of links at any level i $(i>0)$ is

$= \text{links at level } i-1 + 2^{i-1}$

This is the constraint that we use to avoid linking of ③ → ④ (or) ⑦ → ⑧.

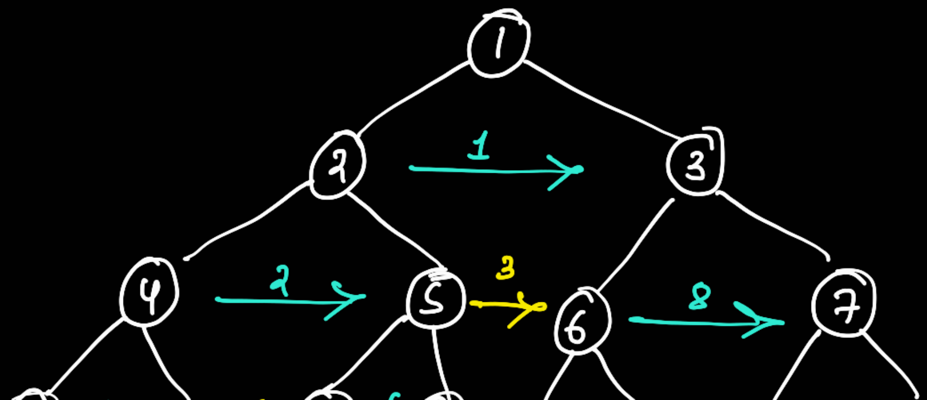
Algorithm:

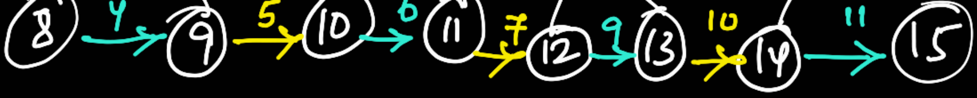
```
→ i = 0, prev = null, links = 0, limit = 0

→ Push root to queue
→ while q is not empty
  {
        curr = Pop( )
        if ( curr has non null children)
          push both children into queue

        if (prev is not null)
        {
            Prev → next = curr
            links ++
        }

        if ( links == limit)
        {
            Prev = null
            links = 0
            limit = limit + $2^i$
            i++
        }
        else
            Prev = curr
}
```

$T(n) : O(n)$

$S(n) : O(n)$ becoz last level contains $\simeq n/2$ nodes

Approach 2: DFS





links like → are pretty straight forward to create while doing Dfs.

i.e. at every non leaf node, do

node→left→next = node→right

The main task is to do links like → becoz you cannot have node ⑥ address when you are at node ④ or ⑤. So how can we Link them.

The order in which links are created is mentioned in the diagram.

you are at node ② and link 1 is already created now you need to create link 3

if we pay attention, we can see that we can access node ⑥ address through link 1

let node = ②

node→right→next = node→next→left

This is it. This is the way to create links.

Algorithm:

```
link (root)
{
    if (root is null or root is leafnode)
            return
    root→left→next = root→right
    if (root→next) //To avoid runtime error at
                     nodes ③ , ⑦
            root -> right -> next   = root -> next -> left
        link (root -> left)
        link (root -> right)
    }
```

T(n) : O (n)

S(n) : O (1)