Smart Coding & Interview Series
Top-20 Basic Program
(Binary Tree Problems)

First, understand the solution building strategies and coding for the problems in LIVE/VIDEO session and then you apply those strategies discussed in LIVE/VIDEO session to solve the following problems. Use your favourite language(C/C++/Java/C#/Python/Scala) for coding.

1) Find Bottom Left Tree Value: Given a binary tree, find the leftmost value in the last row of the tree. You may assume the tree (i.e., the given root node) is not NULL.

Example:

```
Input:
   2
  / \
 1   3
Output: 1
```

Source: https://leetcode.com/problems/find-bottom-left-tree-value/description/

2) Binary Tree Zigzag Level Order Traversal: Given a binary tree, return the zigzag level order traversal of its nodes' values. (ie, from left to right, then right to left for the next level and alternate between).

Example:

Given binary tree `[3,9,20,null,null,15,7]`,

```
   3
  / \
 9  20
   /  \
  15   7
```

return its zigzag level order traversal as:

```
[
  [3],
  [20,9],
  [15,7]
]
```

Source : https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/description/

Smart Coding & Interview Series
Top-20 Basic Program
(Binary Tree Problems)

3) Path Sum: Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

Example:

Given the below binary tree and sum = 22,

```
    5
   / \
  4   8
 /   / \
11  13  4
/ \     \
7   2     1
```

return true, as there exist a root-to-leaf path 5->4->11->2 which sum is 22

Source : *https://leetcode.com/problems/path-sum/description/*

4) Flatten Binary Tree to Linked List: Given a binary tree, flatten it to a linked list in-place.

Example, given the following tree:

```
   1
  / \
 2   5
/ \   \
3   4   6
```

The flattened tree should look like:

```
1
 \
  2
   \
    3
     \
      4
       \
        5
         \
          6
```

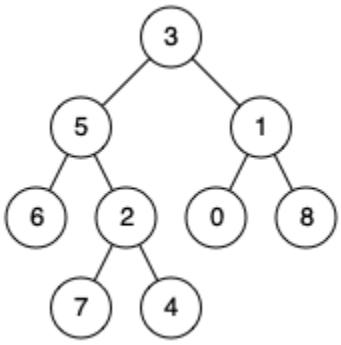
Source : *https://leetcode.com/problems/flatten-binary-tree-to-linked-list/description/*

5) Lowest Common Ancestor of a Binary Tree: Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree. The lowest common ancestor is

defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow a node to be a descendant of itself).

Example:

Given the following binary tree: root = [3,5,1,6,2,0,8,null,null,7,4]



Input: *root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1*
Output: *3*
Note:

- *All of the nodes' values will be unique.*
- *p and q are different and both values will exist in the binary tree.*

Source: *https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/description/*

6) Populating Next Right Pointers in Each Node: 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 :

Smart Coding & Interview Series
Top-20 Basic Program
(Binary Tree Problems)

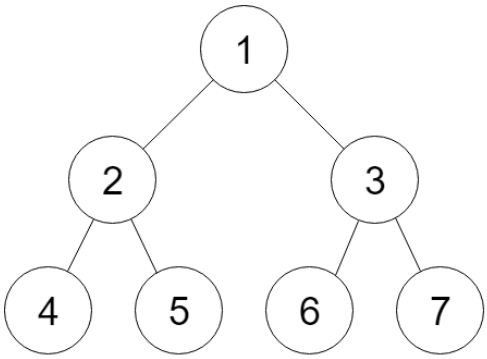


Figure A

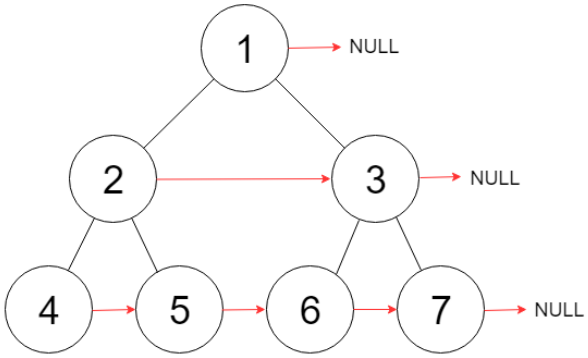


Figure B

Input: root = [1,2,3,4,5,6,7]
Output: [1,#,2,3,#,4,5,6,7,#]

Source:[*https://leetcode.com/problems/populating-next-right-pointers-in-each-node/description/*](https://leetcode.com/problems/populating-next-right-pointers-in-each-node/description/)