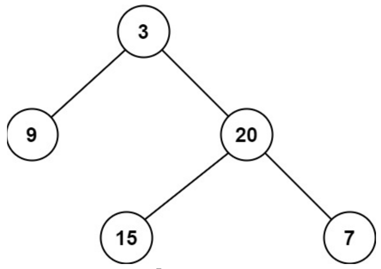
2023-07-15 - Handout – Trees

Q1. Minimum Depth of Binary Tree

Link: https://leetcode.com/problems/minimum-depth-of-binary-tree/

Given a binary tree, find its minimum depth. The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.
Note: A leaf is a node with no children.

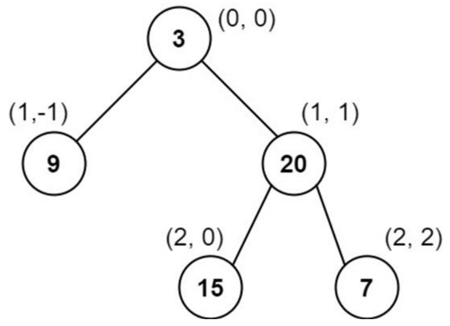


Input: root = [3, 9, 20, null, null, 15, 7]
Output: 2
Input: root = [2, 3, 4, 5, null, 6, null, 7]
Output: 3
Input: root = [2, null, 3, null, 4, null, 5, null, 6]
Output: 5
Constraints:
The number of nodes in the tree is in the range [0, 10^5]
-1000 <= Node.val <= 1000

Q2. Vertical Order Traversal of a Binary Tree

Link: https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/

Given the root of a binary tree, calculate the vertical order traversal of the binary tree. For each node at position (row, col), its left and right children will be at positions (row + 1, col – 1) and (row + 1, col + 1) respectively. The root of the tree is at (0, 0).
The vertical order traversal of a binary tree is a list of top-to-bottom orderings for each column index starting from the leftmost column and ending on the rightmost column. There may be multiple nodes in the same row and same column. In such a case, sort these nodes by their values.
Return the ***vertical order traversal*** of the binary tree.

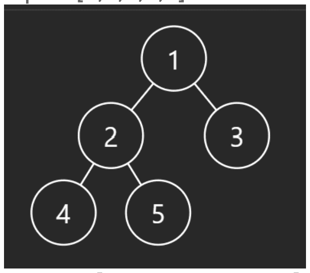


Input: root = [3, 9, 20, null, null, 15, 7]
Output: [[9], [3, 15], [20], [7]]
Explanation:
Column -1: Only node 9 is in the column.
Column 0: Node 3 and 15 are in the column in that order from top to bottom.
Column 1: Only node 20 is in this column.
Column 2: Only node 7 is in this column.
Input: root = [1,2,3,4,5,6,7]
Output: [[4], [2], [1,5,6], [3], [7]]
Constraints: The number of nodes in the tree is in the range [1, 1000].

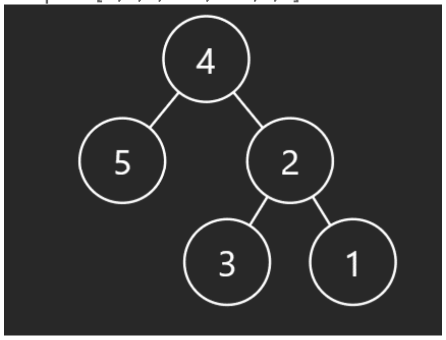
Q3. Binary Tree Upside Down

Link: https://leetcode.com/problems/binary-tree-upside-down/

Given a binary tree where all the right nodes are either leaf nodes with a sibling (a left node that shares the same parent node) or empty, flip it upside down and turn it into a tree where the original right nodes turned into left leaf nodes. Return the new root.
Input: [1,2,3,4,5]



Output: [4,5,2,null,null,3,1]

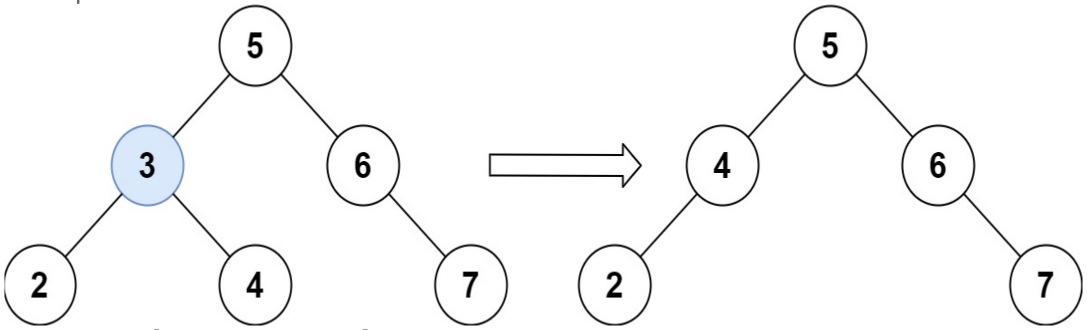


Q4. Delete Node in a BST

Link: https://leetcode.com/problems/delete-node-in-a-bst/

Given a root node reference of a BST and a key, delete the node with the given key in the BST. Return the ***root node reference*** (possibly updated) of the BST.
Basically, the deletion can be divided into two stages:
- Search for a node to remove.
- If the node is found, delete the node.

Example1:



Input: root = [5, 3 ,6, 2, 4, null, 7], key = 3
Output: [5, 4, 6, 2, null, null, 7]
Explanation: Given key to delete is 3. So we find the node with value 3 and delete it.
One valid answer is [5, 4, 6, 2, null, null, 7], shown in the above BST. Please notice that another valid answer is [5, 2, 6, null, 4, null, 7] and it's also accepted.
Example 2: root = [], key = 0
Output: []
Constraints:
The number of nodes in the tree is in the range [0, 10^4]
-10^5<= Node.val <= 10^5
Each node has a unique value
root is a valid binary search tree
-10^5 <= key <= 10^5