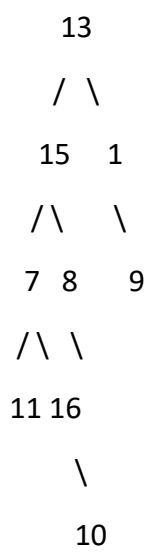
Minimum Height of Binary Tree LeetCode

The program for finding the minimum height of a binary tree

Example:



Output: The Minimum Height of Binary Tree: 3

Approach 1: Function to find the minimum height of the binary tree recursively

- In the **minHeightRecursively** function, we calculate the minimum height of the binary tree recursively.
- If the tree is empty (root is null), its height is 0.
- If there's no left subtree, we return the height of the right subtree plus 1.
- If there's no right subtree, we return the height of the left subtree plus 1.
- Otherwise, we return the minimum height between the left and right subtrees, plus 1 for the current level.

Time Complexity: O(N), where N is the number of nodes in the binary tree. We visit each node once.

Space Complexity: O(H), where H is the height of the binary tree due to the function call stack.

Approach 2: Function to find the minimum height of the binary tree iteratively

- In the **minHeightIteratively** function, we calculate the minimum height of the binary tree iteratively using a queue.

- We start with a height of 1 since the root is at level 1.
- We enqueue all nodes at the current level and return the current height as soon as we encounter a leaf node (a node with no children).

Time Complexity: O(N), where N is the number of nodes in the binary tree. We visit each node once.

Space Complexity: O(N), where N is the number of nodes in the binary tree. In the worst case, when the tree is fully unbalanced (a skewed tree), all nodes need to be enqueued in the queue, resulting in O(N) space complexity.