**Problem Name:** Minimum Depth of Binary tree

**Topics:**

**Companies:**

**Level:** Hard

**Language:** C++

**Problem Statement:**

**Input Format:**

**Output Format:**

**Constraints:**

**Examples:**

**Iterative approach Solution:**

**Explanation:** We can use [Breadth-First Search(BFS)](https://en.wikipedia.org/wiki/Breadth-first_search) to find the first node which is a leaf and return its depth from the root. We can use a queue to store a pair of the nodes and their depths from the root node. As we receive a node which is a leaf, we return its depth.

**Code:**

**Time Complexity**: **O(N)**, as we traverse the whole tree once again.

**Space Complexity: O(N)**, as we use a queue to store details of every node.

**Optimized Solution:**

**Explanation:**

This problem is structurally same as finding the height of a binary tree but in this case, we need to find the minimum height/depth between the root and any leaf in the tree. We can retrieve the minimum depths of left and right subtrees of the root using [Depth First Search(DFS)](https://en.wikipedia.org/wiki/Depth-first_search) and then return the minimum of the two depths. We need to consider some cases when either of the left and right subtrees of a node is **NULL**.  If the left subtree is **NULL,**it will return a height equal to **0**but as we have found no leaf in this subtree, so this **0**will be a wrong answer. Hence, we only need to call the recursive function when the node it is called upon is **NOT** null.

**Code:**

**Time Complexity**: O(N) as we traverse the whole tree once even in the worst case.

**Space Complexity:** O(N) When the binary tree is skewed, the recursive stack frames may consume upto O(N) space.