```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    public int minDepth(TreeNode root) {
        if (root == null) {
            return 0;
        }
        return h(root, 0);
    }

    private int h(TreeNode root, int level) {
        if (root.left == null && root.right == null) {
            return level + 1;
        }
        int l = root.left != null ? h(root.left, level+1) : Integer.MAX_VALUE;
        int r = root.right != null ? h(root.right, level+1) : Integer.MAX_VALUE;
        return Math.min(r, l);
    }
}
```

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    public int minDepth(TreeNode root) {
        if (root == null)
            return 0;
        if (root.left == null) {
            return 1 + minDepth(root.right);
        }
        if (root.right == null) {
            return 1 + minDepth(root.left);
        }
        return 1 + Math.min(minDepth(root.left), minDepth(root.right));
    }
}
```

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.

Example:

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

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

return its minimum depth = 2.