

111 Minimum Depth of Binary Tree

2018年3月31日 19:48

Question:

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.

来自 <https://leetcode.com/problems/minimum-depth-of-binary-tree/description/>

给定一个二叉树，找出其最小深度。

最小深度是从根节点到最近叶节点的最短路径的节点数量。

Solution for Python3:

```
1  # Definition for a binary tree node.
2  # class TreeNode:
3  #     def __init__(self, x):
4  #         self.val = x
5  #         self.left = None
6  #         self.right = None
7
8  class Solution1:
9      def minDepth(self, root):
10         """
11         :type root: TreeNode
12         :rtype: int
13         """
14         if not root:
15             return 0
16         if not root.left:
17             return 1 + self.minDepth(root.right)
18         if not root.right:
19             return 1 + self.minDepth(root.left)
20         return 1 + min(self.minDepth(root.left), self.minDepth(root.right))
21
22 class Solution2:
23     def minDepth(self, root):
24         """
25         :type root: TreeNode
26         :rtype: int
27         """
28         if not root: return 0
29         d = map(self.minDepth, (root.left, root.right))
30         return 1 + (min(d) or max(d))
31         # 我们需要加上最小子树的深度，除了该子树深度为0即该子树为空。
32         # 例如，左子树为空，右子树为1，当前节点深度为1+右子树的深度。
33
34 class Solution2:
35     def minDepth(self, root):
36         """
37         :type root: TreeNode
38         :rtype: int
```

```

39         .type: int
40         """
41         if not root: return 0
42         d, D = sorted(map(self.minDepth, (root.left, root.right)))
43         return 1 + (d or D)

```

Solution for C++:

```

1  /**
2   * Definition for a binary tree node.
3   * struct TreeNode {
4   *     int val;
5   *     TreeNode *left;
6   *     TreeNode *right;
7   *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8   * };
9   */
10 class Solution1 {
11 public:
12     int minDepth(TreeNode* root) {
13         if (root == NULL) {
14             return 0;
15         }
16         if (root->left == NULL) {
17             return 1 + minDepth(root->right);
18         }
19         if (root->right == NULL) {
20             return 1 + minDepth(root->left);
21         }
22         return 1 + min(minDepth(root->left), minDepth(root->right));
23     };
24
25 class Solution2 {
26 public:
27     int minDepth(TreeNode* root) {
28         if (!root) return 0;
29         int L = minDepth(root->left), R = minDepth(root->right);
30         return 1 + (min(L, R) ? min(L, R) : max(L, R));
31         // return 1 + (L && R ? min(L, R) : max(L, R));
32         // return 1 + (!L - !R ? max(L, R) : min(L, R));
33     }
34 };

```