

111. Minimum Depth of Binary Tree

Oct. 28, 2018 | 32.3K views

Average Rating: 4.17 (23 votes)

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]`.



return its minimum depth = 2.

Solution

Tree definition

First of all, here is the definition of the `TreeNode` which we would use.

JavaPythonCopy

```
1 // Definition for a binary tree node.
2 public class TreeNode {
3     int val;
4     TreeNode left;
5     TreeNode right;
6
7     TreeNode(int x) {
8         val = x;
9     }
10 }
```

Approach 1: Recursion

Algorithm

The intuitive approach is to solve the problem by recursion. Here we demonstrate an example with the DFS (Depth First Search) strategy.

JavaPythonCopy

```
1 class Solution {
2     public int minDepth(TreeNode root) {
3         if (root == null) {
4             return 0;
5         }
6
7         if ((root.left == null) && (root.right == null)) {
8             return 1;
9         }
10
11         int min_depth = Integer.MAX_VALUE;
12         if (root.left != null) {
13             min_depth = Math.min(minDepth(root.left), min_depth);
14         }
15         if (root.right != null) {
16             min_depth = Math.min(minDepth(root.right), min_depth);
17         }
18         return min_depth + 1;
19     }
20 }
21 }
```

Complexity analysis

- Time complexity : we visit each node exactly once, thus the time complexity is $\mathcal{O}(N)$, where N is the number of nodes.
- Space complexity : in the worst case, the tree is completely unbalanced, e.g. each node has only one child node, the recursion call would occur N times (the height of the tree), therefore the storage to keep the call stack would be $\mathcal{O}(N)$. But in the best case (the tree is completely balanced), the height of the tree would be $\log(N)$. Therefore, the space complexity in this case would be $\mathcal{O}(\log(N))$.

Approach 2: DFS Iteration

We could also convert the above recursion into iteration, with the help of stack.

The idea is to visit each leaf with the DFS strategy, while updating the minimum depth when we reach the leaf node.

So we start from a stack which contains the root node and the corresponding depth which is `1`. Then we proceed to the iterations: pop the current node out of the stack and push the child nodes. The minimum depth is updated at each leaf node.

JavaPythonCopy

```
1 class Solution {
2     public int minDepth(TreeNode root) {
3         LinkedList
```

Complexity analysis

- Time complexity : each node is visited exactly once and time complexity is $\mathcal{O}(N)$.
- Space complexity : in the worst case we could keep up to the entire tree, that results in $\mathcal{O}(N)$ space complexity.

Approach 3: BFS Iteration

The drawback of the DFS approach in this case is that all nodes should be visited to ensure that the minimum depth would be found. Therefore, this results in a $\mathcal{O}(N)$ complexity. One way to optimize the complexity is to use the BFS strategy. We iterate the tree level by level, and the first leaf we reach corresponds to the minimum depth. As a result, we do not need to iterate all nodes.

JavaPythonCopy

```
1 class Solution {
2     public int minDepth(TreeNode root) {
3         LinkedList
```

Complexity analysis

- Time complexity : in the worst case for a balanced tree we need to visit all nodes level by level up to the tree height, that excludes the bottom level only. This way we visit $N/2$ nodes, and thus the time complexity is $\mathcal{O}(N)$.
- Space complexity : is the same as time complexity here $\mathcal{O}(N)$.

Rate this article: ★★★★★

Previous

Next

Comments: 26

Sort By

- 🔥

Type comment here... (Markdown is supported)

🔍 Preview

Post
- Alok_k ★87 🕒 December 20, 2018 6:12 AM

4 lines Simplest recursion beats 100%

```
public int minDepth(TreeNode root) {
    if(root==null){return 0;}
    if(root.left==null){return minDepth(root.right)+1;}
}
```

56 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 3 REPLIES
- haoyangfan ★897 🕒 November 26, 2018 7:18 AM

More concise BFS solution

```
class Solution {
    public int minDepth(TreeNode root) {
        if (root == null) return 0;
    }
}
```

23 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 1 REPLY
- v1s1on ★499 🕒 January 22, 2019 10:57 AM

Both the DFS & BFS implementations are unnecessarily complicated. Here's my stab at them w/ C++.

Both run in 4ms which seems to be the best time so far.

DFS

5 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 1 REPLY
- stekale ★4 🕒 January 31, 2019 9:10 AM

The minimum depth in the example given should have the output as 1 right, why the output is given as 2? can someone clarify?

4 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 2 REPLIES
- michalya ★6 🕒 October 29, 2018 1:30 AM

I think that it's better to solve it with BFS approach. If the tree is unbalanced then we will find the min height when we first reach a leaf, without exploring longer paths.

4 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 1 REPLY
- dyckia ★199 🕒 November 11, 2019 8:36 PM

Here is a concise DFS solution without using pair

```
public int minDepth(TreeNode root) {
    // use two stacks, one for storing the nodes to be examined
    // one for recording the corresponding depth
}
```

1 🗳️ | 🔄 Share | 🗨️ Reply
- LittleYoki ★31 🕒 February 17, 2019 2:31 AM

I think in the worst case of the BFS method, we have a tree like this:

```
1
 \
 2
```

1 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 1 REPLY
- suraj_ch77 ★30 🕒 October 29, 2018 7:55 AM

BFS uses queue, right? How is stack solving the problem of BFS here?

1 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 3 REPLIES
- NideeshT ★576 🕒 April 23, 2019 9:44 AM

(Java) Youtube Video Explanation - accepted

<https://youtu.be/QR58z5LSJul>

Hi everyone, I'm making Youtube videos to help me study/review solved problems. Wanted to share if it helps!

Note I claim no rights to this question. All rights belong to Leetcode. The company tags in the video

0 🗳️ | 🔄 Share | 🗨️ Reply
- drpepper07 ★84 🕒 April 1, 2019 9:10 AM

Why so complicated? Why not extend maxDepth to handle the edge cases here?

Beats 100 % with 0 ms runtime.

```
/**
 * The minimum depth is the number of nodes along the shortest path ***from the
 */
```

0 🗳️ | 🔄 Share | 🗨️ Reply

SHOW 1 REPLY