

111. Minimum Depth of Binary Tree [\(/problems/minimum-depth-of-binary-tree/\)](/problems/minimum-depth-of-binary-tree/)

Oct. 28, 2018 | 17.5K views

Average Rating: 4.71 (14 votes)

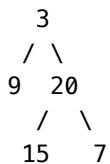
Given a binary tree, find its minimum depth.

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.

Java

Python

 Articles > 111. Minimum Depth of Binary Tree  Copy

```
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
```

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.

Java

Python

 Copy

```
1 class Solution:
2     def minDepth(self, root):
3         """
4         :type root: TreeNode
5         :rtype: int
6         """
7         if not root:
8             return 0
9
10        children = [root.left, root.right]
11        # if we're at leaf node
12        if not any(children):
13            return 1
14
15        min_depth = float('inf')
16        for c in children:
17            if c:
18                min_depth = min(self.minDepth(c), min_depth)
19        return min_depth + 1
```

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.

Java

Python

 Copy

```

1 class Solution:
2     def minDepth(self, root):
3         """
4         :type root: TreeNode
5         :rtype: int
6         """
7         if not root:
8             return 0
9         else:
10            stack, min_depth = [(1, root)], float('inf')
11
12            while stack:
13                depth, root = stack.pop()
14                children = [root.left, root.right]
15                if not any(children):
16                    min_depth = min(depth, min_depth)
17                for c in children:
18                    if c:
19                        stack.append((depth + 1, c))
20
21            return min_depth

```

Complexity analysis

Articles > 111. Minimum Depth of Binary Tree ▼

- 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.

Java

Python

Copy

```
1 from collections import deque
2 class Solution:
3     def minDepth(self, root):
4         """
5         :type root: TreeNode
6         :rtype: int
7         """
8         if not root:
9             return 0
10        else:
11            node_deque = deque([(1, root)])
12
13        while node_deque:
14            depth, root = node_deque.popleft()
15            children = [root.left, root.right]
16            if not any(children):
17                return depth
18            for c in children:
19                if c:
20                    node_deque.append((depth + 1, c))
```

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 $O(N)$

Articles > 111. Minimum Depth of Binary Tree ▼

Analysis written by @liaison (<https://leetcode.com/liaison/>) and @andvary (<https://leetcode.com/andvary/>)

Rate this article:

Previous (/articles/beautiful-array/)

Next (/articles/n-ary-tree-preorder-traversal/)

Comments: 14

Sort By ▼



Type comment here... (Markdown is supported)

Preview

Post



(/alok_k)

Alok_k (alok_k) ★ 38 🕒 December 19, 2018 4:42 PM

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;}  
}
```

Read More

23 ▲ ▼ 📄 Share ↩ Reply



(/haoyangfan)

haoyangfan (haoyangfan) ★ 384 🕒 November 25, 2018 5:48 PM

More concise BFS solution

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

Read More

5 ▲ ▼ 📄 Share ↩ Reply

SHOW 1 REPLY



(/stekale)

stekale (stekale) ★ 3 🕒 January 30, 2019 7:40 PM

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

3 ▲ ▼ 📄 Share ↩ Reply

SHOW 2 REPLIES



(/v1s1on)

v1s1on (v1s1on) ★ 216 ⌚ January 21, 2019 9:27 PM

Articles > 111. Minimum Depth of Binary Tree ▾

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

Read More

2 ^ ▾ | Share | Reply

SHOW 1 REPLY



(/michalya)

michalya (michalya) ★ 3 ⌚ October 28, 2018 1:00 PM

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.

1 ^ ▾ | Share | Reply

SHOW 1 REPLY



(/suraj_ch77)

suraj_ch77 (suraj_ch77) ★ 24 ⌚ October 28, 2018 7:25 PM

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

0 ^ ▾ | Share | Reply

SHOW 2 REPLIES



(/s961206)

s961206 (s961206) ★ 316 ⌚ July 5, 2019 4:24 PM

I think there is no needs to use Pair for BFS solution:

```
public int minDepth(TreeNode root) {
    if(root == null) return 0;
    Queue<TreeNode> q = new LinkedList<>();
```

Read More

0 ^ ▾ | Share | Reply



(/nideesht)

NideeshT (nideesht) ★ 172 ⌚ April 22, 2019 9:14 PM

(Java) Youtube Video Explanation - accepted

<https://youtu.be/QR5Bz5LSHul> (<https://youtu.be/QR5Bz5LSHul>)

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

Read More

0 ^ ▾ | Share | Reply



(/drpepper07)

drpepper07 (drpepper07) ★ 37 ⌚ March 31, 2019 8:40 PM



Articles > 111. Minimum Depth of Binary Tree ▾

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 root node to the leaf node.
```

[Read More](#)

0 ^ v ... Share ... Reply

SHOW 1 REPLY



(/littleyoki)

LittleYoki (littleyoki) ★ 24 ⌚ February 16, 2019 1:01 PM

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

```
1
 \
  2
```

[Read More](#)

0 ^ v ... Share ... Reply

SHOW 1 REPLY

[<](#) [1](#) [2](#) [>](#)