559. Maximum Depth of n-ary Tree (/problems/maximum-depth-of-n-ary-tree/)

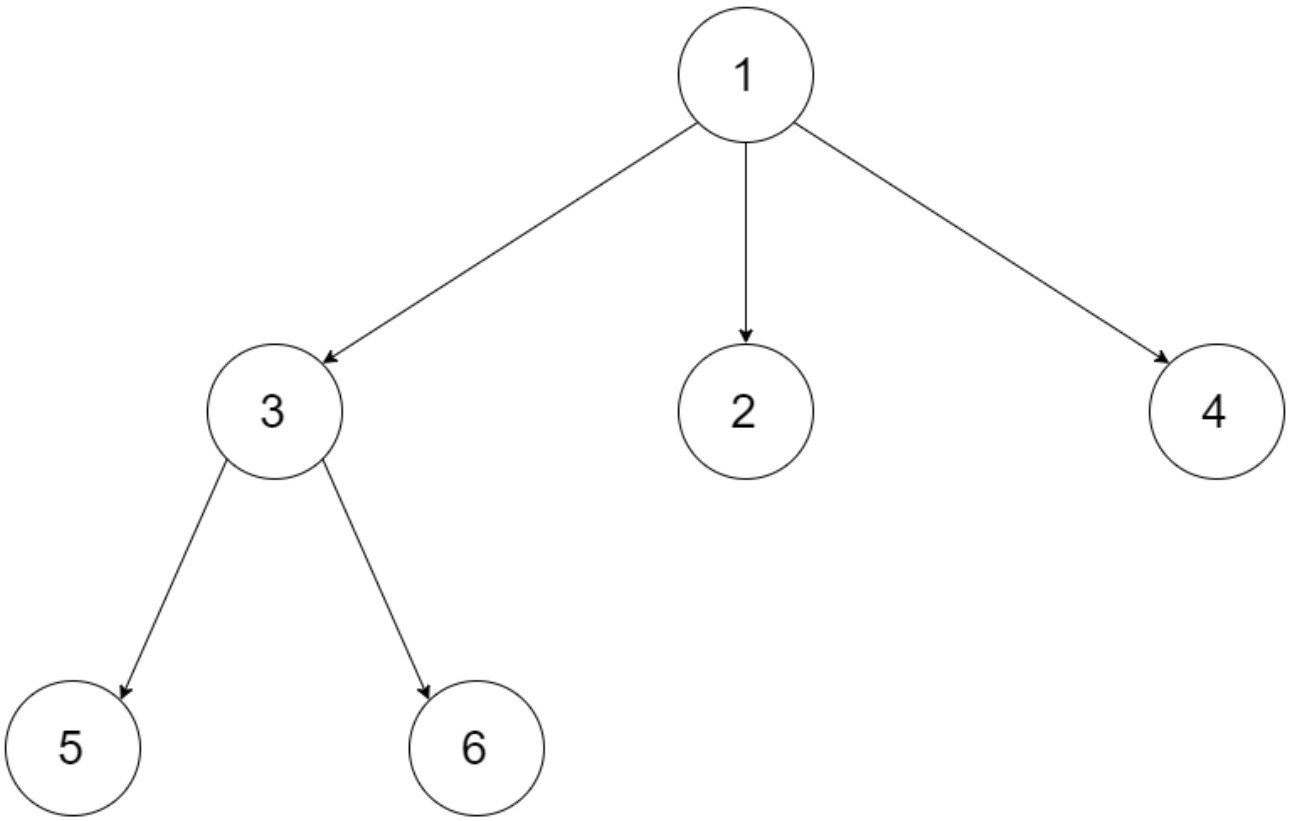
Oct. 2, 2018 | 12.4K views

Average Rating: 4 (5 votes)

Given a n-ary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

For example, given a `3-ary` tree:



We should return its max depth, which is 3.

Note:

1. The depth of the tree is at most `1000`.
2. The total number of nodes is at most `5000`.

Solution

Tree definition

First of all, please refer to this article (https://leetcode.com/articles/maximum-depth-of-binary-tree/) for the solution in case of binary tree. This article offers the same ideas with a bit of generalisation.

Here is the definition of the `TreeNode` which we would use.

Java | Python

Copy

```
# Definition for a Node.
class Node(object):
    def __init__(self, val, children):
        self.val = val
        self.children = children
```

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

Articles > 559. Maximum Depth of n-ary Tree

Copy

```python
class Solution(object):
    def maxDepth(self, root):
        """
        :type root: Node
        :rtype: int
        """
        if root is None:
            return 0
        elif root.children == []:
            return 1
        else:
            height = [self.maxDepth(c) for c in root.children]
            return max(height) + 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: Iteration

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

> The idea is to visit each node with the DFS strategy, while updating the max depth at each visit.

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 depth is updated at each step.

Java | Python

Articles > 559. Maximum Depth of n-ary Tree

Copy

```python
class Solution(object):
    def maxDepth(self, root):
        """
        :type root: Node
        :rtype: int
        """
        stack = []
        if root is not None:
            stack.append((1, root))

        depth = 0
        while stack != []:
            current_depth, root = stack.pop()
            if root is not None:
                depth = max(depth, current_depth)
                for c in root.children:
                    stack.append((current_depth + 1, c))

        return depth
```

Complexity analysis

- Time complexity : $\mathcal{O}(N)$.
- Space complexity : $\mathcal{O}(N)$.

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

Rate this article:

Previous (/articles/maximum-depth-of-binary-tree/) Next (/articles/remove-invalid-parentheses/)

Comments: 11

Sort By

Type comment here... (Markdown is supported)

Preview Post

(/vitalii_kravets)

Vitalii_Kravets (vitalii_kravets) ★ 16 October 8, 2018 11:40 AM

I don't think we need a null-check

```
if (root != null) {
```

inside the while of the iterative approach.

7 Share Reply

(/heronalps)

Heronalps (heronalps) ★ 108 October 26, 2018 1:30 PM

The second solution is basically BFS with a queue.

6 Share Reply

(/sand91)

SanD91 (sand91) ★ 295 February 1, 2019 12:26 PM

Easy java solution without using collections. Beats 100%

```
class Solution {
    public int maxDepth(Node root) {
        if (root == null) return 0;
```

Read More

5 Share Reply

(/grimwall)

grimwall (grimwall) ★ 2 November 6, 2018 6:04 AM

Second solution is BFS, the visiting order for the default test case is: 1, 3, 2, 4, 5, 6;

2 Share Reply

SHOW 3 REPLIES

(/hy_huaiyu)

HY_huaiyu (hy_huaiyu) ★ 95 November 3, 2018 8:43 AM

recursion approach, I think this one is easier to understand :

```
class Solution {
    int max = 0;
    public int maxDepth(Node root) {
```

Read More

2 Share Reply

(/try101)

try101 (try101) ★ 6 November 7, 2018 5:43 PM

how is space complexity is o(logn) in case of balanced tree .there is function call for each node so I would think o(n) . can some one explain this?

1 Share Reply

SHOW 3 REPLIES

(/kotak86)

kotak86 (kotak86) ★ 27 July 7, 2019 5:24 PM

@liaison (https://leetcode.com/liaison) and @andvary for the second approach, Space complexity for the best case should log(n) correct ?
As we never store all the nodes in queue if tree is balanced

0 Share Reply

(/haoyangfan)

haoyangfan (haoyangfan) ★ 384 April 29, 2019 8:41 PM

Why the iterative solution is running so slow compared with the recursive solution?
Here is my implementation of iterative solution

```
/*
// Definition for a Node.
```

Read More

0 Share Reply

SHOW 1 REPLY

(/leetbunny)

leetbunny (leetbunny) ★ 28 January 7, 2019 5:21 AM

Is time complexity for DFS O(nlogn) given the sort/get max instead of O(n)?

0 Share Reply

(/sjw214)

sjw214 (sjw214) ★ 112 November 28, 2018 3:15 PM

Javascript solution:

```
class Node {
  constructor(val) {
      this.val = val;
```

Read More

0 Share Reply

‹ 1 2 ›