❮ Previous (/articles/binary-tree-preorder-transversal/) Next ❯ (/articles/utf-8-validation/)

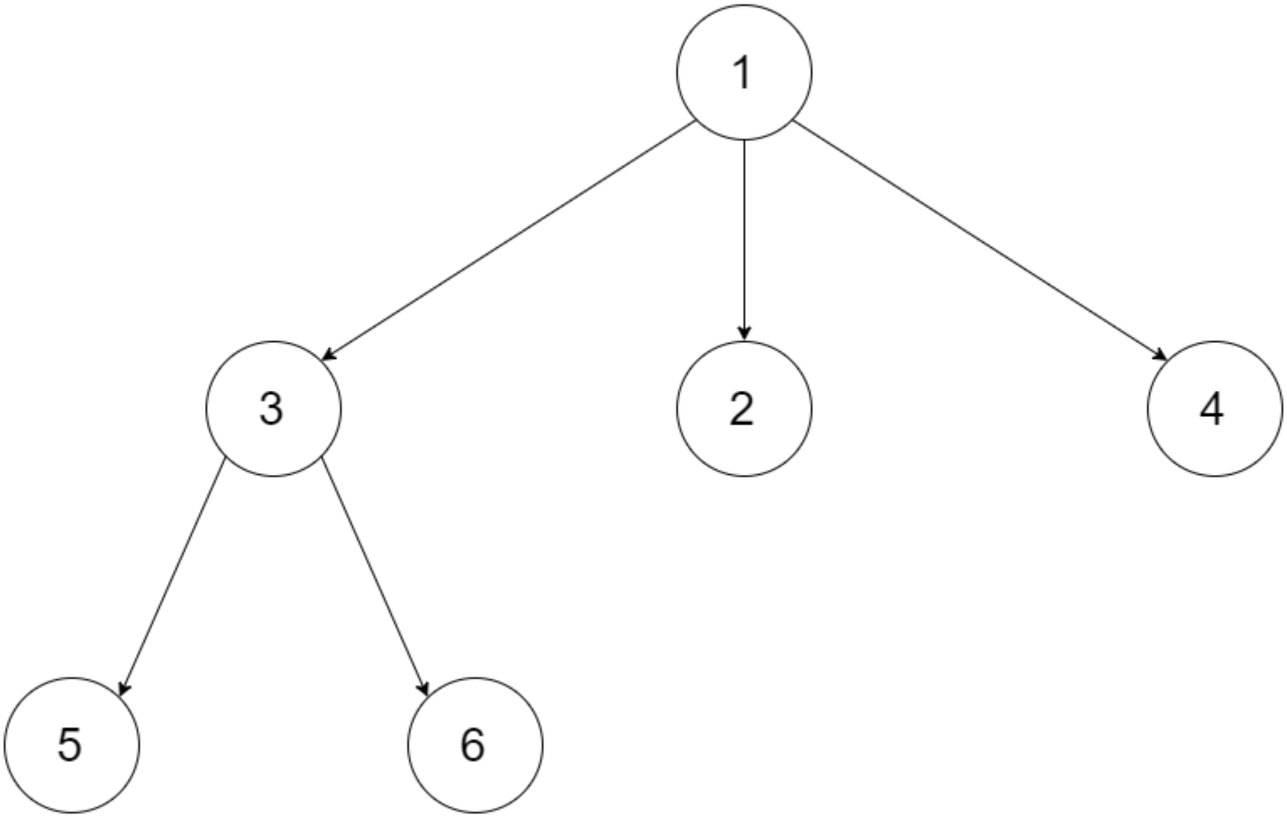
590. N-ary Tree Postorder Traversal (/problems/n-ary-tree-postorder-traversal/)

Oct. 21, 2018 | 8.4K views

Average Rating: 5 (7 votes)

Given an n-ary tree, return the *postorder* traversal of its nodes' values.

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



Return its postorder traversal as: `[5,6,3,2,4,1]` .

Note:

Articles › 590. N-ary Tree Postorder Traversal

Recursive solution is trivial, could you do it iteratively?

Solution

How to traverse the tree

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

There are two general strategies to traverse a tree:

- *Breadth First Search* (`BFS`)

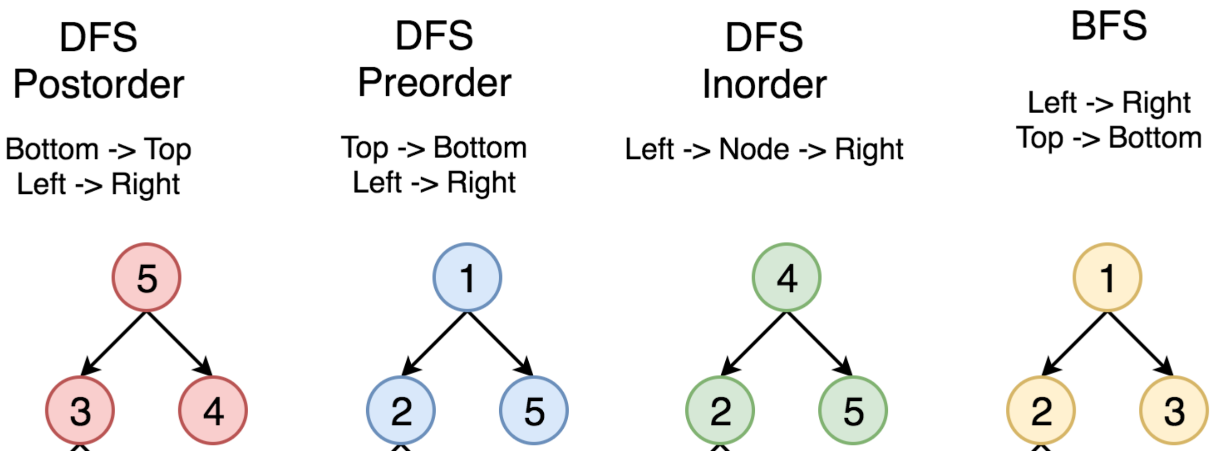
  We scan through the tree level by level, following the order of height, from top to bottom. The nodes on higher level would be visited before the ones with lower levels.

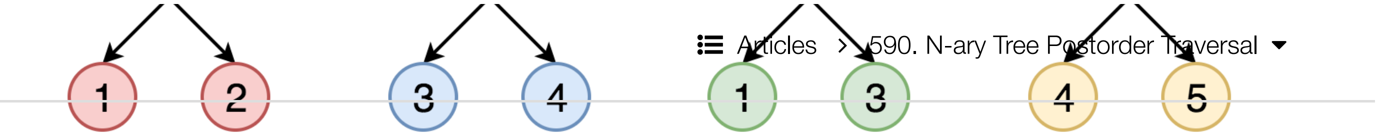
- *Depth First Search* (`DFS`)

  In this strategy, we adopt the `depth` as the priority, so that one would start from a root and reach all the way down to certain leaf, and then back to root to reach another branch.

  The DFS strategy can further be distinguished as `preorder`, `inorder`, and `postorder` depending on the relative order among the root node, left node and right node.

On the following figure the nodes are numerated in the order you visit them, please follow `1-2-3-4-5` to compare different strategies.





Here the problem is to implement postorder traversal using iterations.

Approach 1: Iterations

Algorithm

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

Java Python Copy

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

Let's start from the root and then at each iteration pop the current node out of the stack and push its child nodes. In the implemented strategy we push nodes into stack following the order `Top->Bottom` and `Left->Right`. Since DFS postorder traversal is `Bottom->Top` and `Left->Right` the output list should be reverted after the end of loop.

Java Python

☰ Articles › 590. N-ary Tree Postorder Traversal

Copy

```
class Solution(object):
    def postorder(self, root):
        """
        :type root: Node
        :rtype: List[int]
        """
        if root is None:
            return []

        stack, output = [root, ], []
        while stack:
            root = stack.pop()
            if root is not None:
                output.append(root.val)
            for c in root.children:
                stack.append(c)

        return output[::-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, *i.e.* the size of tree.
- Space complexity : depending on the tree structure, we could keep up to the entire tree, therefore, the space complexity is $\mathcal{O}(N)$.

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

Rate this article:

Previous (/articles/binary-tree-preorder-transversal/)

Next (/articles/utf-8-validation/)

Comments: 5

Sort By ▾

Type comment here... (Markdown is supported)

Preview Post

(/woshi471983103)

woshi471983103 (woshi471983103) ★ 1 January 21, 2019 2:55 PM

thx bro nice work

1 Share Reply

(/donellejr)

donellejr (donellejr) ★ 4 April 28, 2019 11:37 AM

C# Iterative version. Time complexity: O(N) Space complexity: O(N).

```
public IList<int> Postorder(Node root) {
        IList<int> result = new List<int>();
        if (root == null) return result;
```

Read More

0 Share Reply

(/hackerjatt)

hackerjatt (hackerjatt) ★ 2 March 11, 2019 10:12 AM

def postorder(self, root: 'Node') -> List[int]:
if not root:
return self.res
for child in root.children:
self.postorder(child)

Read More

0 Share Reply

(/danielerhabor)

danielerhabor (danielerhabor) ★ 0 February 22, 2019 12:38 AM

Are you sure this solution works for
1
/ |
2 5 4
/ |

Read More

0 Share Reply

deleted_user ★ 31 January 31, 2019 1:57 PM

A recursive version:

```
class Solution {
    public List<Integer> postorder(Node root) {
        return f(root, new LinkedList<>());
```

Read More

0 Share Reply

Help Center (/support/) | Students (/students) | Terms (/terms/) | Privacy