



What is Morris traversal?

 Educative Answers Team 

Morris (InOrder) traversal is a tree traversal algorithm that does *not* employ the use of recursion or a stack. In this traversal, links are created as successors and nodes are printed using these links. Finally, the changes are reverted back to restore the original tree.

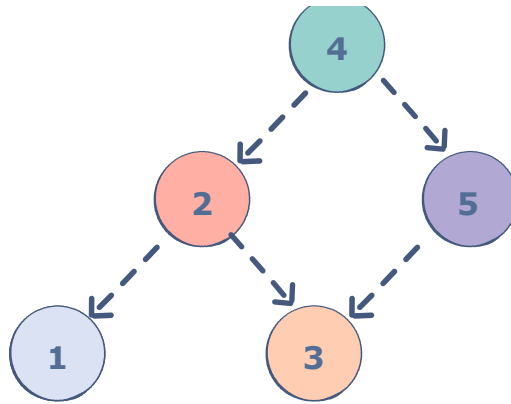
Algorithm

- Initialize the `root` as the current node `curr`.
- While `curr` is *not* `NULL`, check if `curr` has a left child.
- If `curr` does not have a left child, print `curr` and update it to point to the node on the right of `curr`.
- Else, make `curr` the right child of the rightmost node in `curr`'s left subtree.
- Update `curr` to this left node.

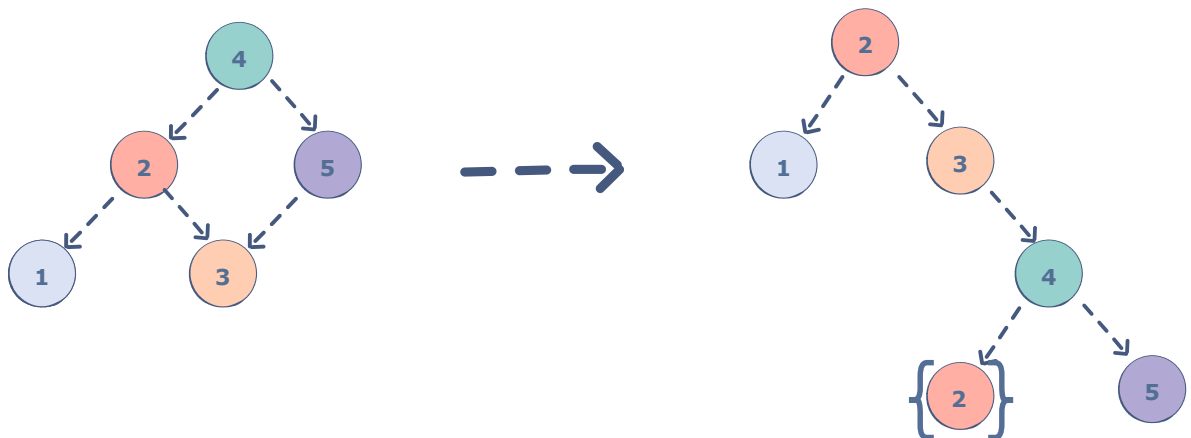
Demo

Let's take the binary tree given below and traverse it using Morris (InOrder) traversal.

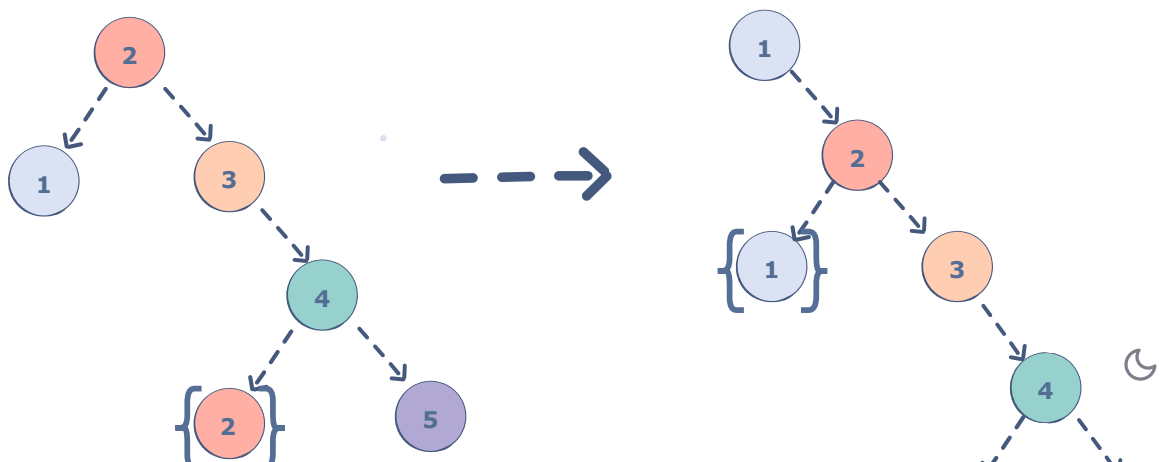


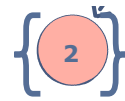


4 is the root, so it is initialized as **curr**. 4 has a left child, so it is made the rightmost right child of it's left subtree (the immediate predecessor to 4 in an InOrder traversal). Finally, 4 is made the right child of 3 and **curr** is set to 2.



The {2} above refers to 2 and all of its children. Now that the tree has a link back to 4, the traversal continues.





1 is printed because it has no left child and `curr` is returned to 2, which was made to be 1's right child in the previous iteration. On the next iteration, 2 has both children. However, the dual-condition of the loop makes it stop when it reaches itself; this is an indication that its left subtree has already been traversed. So, it prints itself and continues with its right subtree (3). 3 prints itself, and `curr` becomes 3 and goes through the same checking process that 2 did. It also realizes that its left subtree has been traversed and continues with the 4. The rest of the tree follows this same pattern.

Code

The above algorithm is implemented in the code below:

C++

 Python

Java

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int data;
6      struct Node* left_node;
7      struct Node* right_node;
8  };
9
10 void Morris(struct Node* root)
11 {
12     struct Node *curr, *prev;
13
14     if (root == NULL)
15         return;
16
17     curr = root;
18
19     while (curr != NULL) {
20
21         if (curr->left node == NULL) {
```

```

22         cout << curr->data << endl;
23         curr = curr->right_node;
24     }
25
26     else {
27
28         /* Find the previous (prev) of curr */
29         prev = curr->left_node;
30         while (prev->right_node != NULL && prev->right_node != curr)
31             prev = prev->right_node;

```

Run

RELATED TAGS

tree

morris traversal

Copyright ©2022 Educative, Inc. All rights reserved



Related Courses



Educative

Data Structures in JavaScript:



Malcolm Maclean



D3 Tips and Tricks: In