Iterative Postorder Traversal | Set 1 (Using Two Stacks)

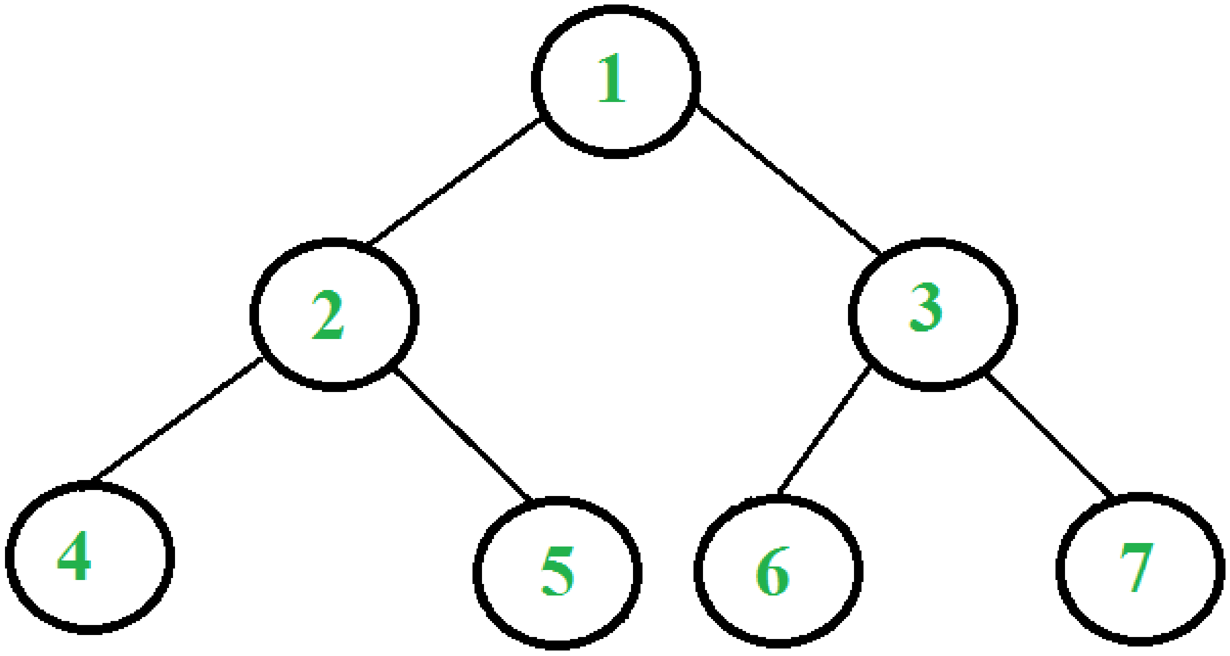
Last Updated : 11 Sep, 2023

•

We have discussed iterative inorder and iterative preorder traversals. In this post, iterative postorder traversal is discussed, which is more complex than the other two traversals (due to its nature of non-tail recursion, there is an extra statement after the final recursive call to itself). Postorder traversal can easily be done using two stacks, though. The idea is to push reverse postorder traversal to a stack. Once we have the reversed postorder traversal in a stack, we can just pop all items one by one from the stack and print them; this order of printing will be in postorder because of the LIFO property of stacks. Now the question is, how to get reversed postorder elements in a stack – the second stack is used for this purpose. For example, in the following tree, we need to get 1, 3, 7, 6, 2, 5, 4 in a stack. If we take a closer look at this sequence, we can observe that this sequence is very similar to the preorder traversal. The only difference is that the right child is visited before left child, and therefore the sequence is "root right left" instead of "root left right". So, we can do something like iterative preorder traversal with the following differences:
a) Instead of printing an item, we push it to a stack.
b) We push the left subtree before the right subtree.
Following is the complete algorithm. After step 2, we get the reverse of a postorder traversal in the second stack. We use the first stack to get the correct order.

```
1. Push root to first stack.
2. Loop while first stack is not empty
   2.1 Pop a node from first stack and push it to second stack
   2.2 Push left and right children of the popped node to first stack
3. Print contents of second stack
```

Let us consider the following tree



Following are the steps to print postorder traversal of the above tree using two stacks.