

Iterative Postorder Traversal | Set 2 (Using One Stack)

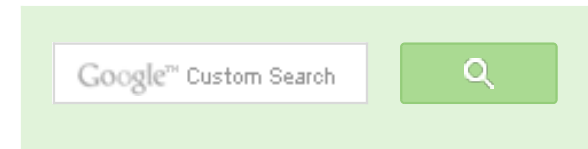
We have discussed a simple [iterative postorder traversal using two stacks](#) in the previous post. In this post, an approach with only one stack is discussed.

The idea is to move down to leftmost node using left pointer. While moving down, push root and root's right child to stack. Once we reach leftmost node, print it if it doesn't have a right child. If it has a right child, then change root so that the right child is processed before.

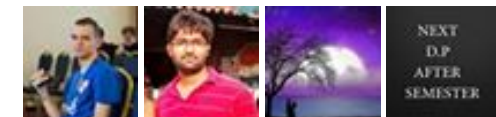
Following is detailed algorithm.

- 1.1 Create an empty stack
- 2.1 Do following while root is not NULL
 - a) Push root's right child and then root to stack.
 - b) Set root as root's left child.
- 2.2 Pop an item from stack and set it as root.
 - a) If the popped item has a right child and the right child is at top of stack, then remove the right child from stack, push the root back and set root as root's right child.
 - b) Else print root's data and set root as NULL.
- 2.3 Repeat steps 2.1 and 2.2 while stack is not empty.

Let us consider the following tree



52,731 people like [GeeksforGeeks](#).



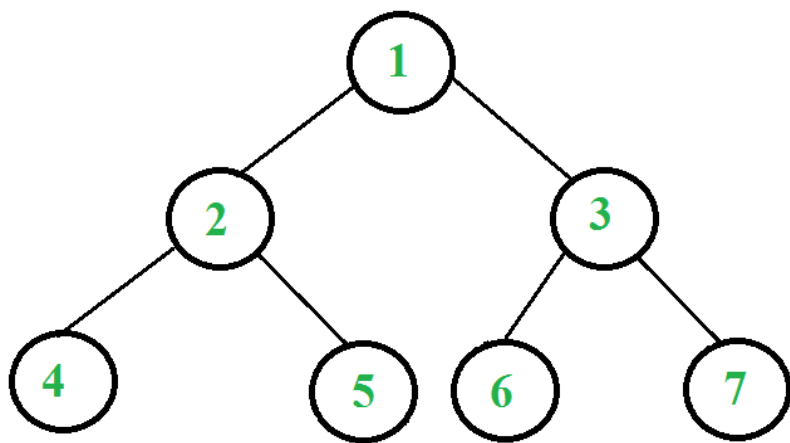
[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)



Following are the steps to print postorder traversal of the above tree using one stack.

1. Right child of 1 exists.
Push 3 to stack. Push 1 to stack. Move to left child.
Stack: 3, 1
2. Right child of 2 exists.
Push 5 to stack. Push 2 to stack. Move to left child.
Stack: 3, 1, 5, 2
3. Right child of 4 doesn't exist. '
Push 4 to stack. Move to left child.
Stack: 3, 1, 5, 2, 4
4. Current node is NULL.
Pop 4 from stack. Right child of 4 doesn't exist.
Print 4. Set current node to NULL.
Stack: 3, 1, 5, 2
5. Current node is NULL.
Pop 2 from stack. Since right child of 2 equals stack top element,
pop 5 from stack. Now push 2 to stack.
Move current node to right child of 2 i.e. 5
Stack: 3, 1, 2

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

Shouldn't you expect
a cloud with:

SYSTEM MONITORING

Plus the experts to help run it?

TRY MANAGED CLOUD ►

 **rackspace**
the open cloud company

Popular Posts

All permutations of a given string

Memory Layout of C Programs

Understanding "extern" keyword in C

Median of two sorted arrays

Tree traversal without recursion and without
stack!

Structure Member Alignment, Padding and

Data Packing

Intersection point of two Linked Lists

6. Right child of 5 doesn't exist. Push 5 to stack. Move to left child.

Stack: 3, 1, 2, 5

7. Current node is NULL. Pop 5 from stack. Right child of 5 doesn't exist.
Print 5. Set current node to NULL.

Stack: 3, 1, 2

8. Current node is NULL. Pop 2 from stack.
Right child of 2 is not equal to stack top element.
Print 2. Set current node to NULL.

Stack: 3, 1

9. Current node is NULL. Pop 1 from stack.
Since right child of 1 equals stack top element, pop 3 from stack.
Now push 1 to stack. Move current node to right child of 1 i.e. 3

Stack: 1

10. Repeat the same as above steps and Print 6, 7 and 3.
Pop 1 and Print 1.

```
// C program for iterative postorder traversal using one stack
#include <stdio.h>
#include <stdlib.h>

// Maximum stack size
#define MAX_SIZE 100

// A tree node
struct Node
{
    int data;
    struct Node *left, *right;
};

// Stack type
struct Stack
{
    int size;
    int top;
```

```

    struct Node* *array;
};

// A utility function to create a new tree node
struct Node* newNode(int data)
{
    struct Node* node = (struct Node*) malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

// A utility function to create a stack of given size
struct Stack* createStack(int size)
{
    struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack))
    stack->size = size;
    stack->top = -1;
    stack->array = (struct Node**) malloc(stack->size * sizeof(struct Node));
    return stack;
}

// BASIC OPERATIONS OF STACK
int isFull(struct Stack* stack)
{ return stack->top - 1 == stack->size; }

int isEmpty(struct Stack* stack)
{ return stack->top == -1; }

void push(struct Stack* stack, struct Node* node)
{
    if (isFull(stack))
        return;
    stack->array[++stack->top] = node;
}

struct Node* pop(struct Stack* stack)
{
    if (isEmpty(stack))
        return NULL;
    return stack->array[stack->top--];
}

struct Node* peek(struct Stack* stack)
{
    if (isEmpty(stack))
        return NULL;

```

695



Subscribe

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

[Backtracking | Set 7 \(Sudoku\)](#) · 35 minutes ago

RVM Can someone please elaborate this Qs from above...

[Flipkart Interview | Set 6](#) · 55 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

[Software Engineering Lab, Samsung Interview | Set 2](#) · 55 minutes ago

@meya Working solution for question 2 of 4f2f round....

[Amazon Interview | Set 53 \(For SDE-1\)](#) · 1 hour ago
sandeep void rearrange(struct node *head)
{...

[Given a linked list, reverse alternate nodes and append at the end](#) · 2 hours ago

Neha I think that is what it should return as, in...

[Find depth of the deepest odd level leaf node](#) · 3 hours ago

```

    return stack->array[stack->top];
}

// An iterative function to do postorder traversal of a given binary tree
void postOrderIterative(struct Node* root)
{
    // Check for empty tree
    if (root == NULL)
        return;

    struct Stack* stack = createStack(MAX_SIZE);
    do
    {
        // Move to leftmost node
        while (root)
        {
            // Push root's right child and then root to stack.
            if (root->right)
                push(stack, root->right);
            push(stack, root);

            // Set root as root's left child
            root = root->left;
        }

        // Pop an item from stack and set it as root
        root = pop(stack);

        // If the popped item has a right child and the right child is
        // processed yet, then make sure right child is processed before
        if (root->right && peek(stack) == root->right)
        {
            pop(stack); // remove right child from stack
            push(stack, root); // push root back to stack
            root = root->right; // change root so that the right
                                // child is processed next
        }
        else // Else print root's data and set root as NULL
        {
            printf("%d ", root->data);
            root = NULL;
        }
    } while (!isEmpty(stack));
}

// Driver program to test above functions
int main()


```

AdChoices 

[► Java to C++](#)

[► Java Stack](#)

[► 7 Stack](#)

AdChoices 

[► Node](#)

[► Java Array](#)

[► C++ Empty Stack](#)

AdChoices 

[► Stack Array](#)

[► System Stack](#)

[► A Stack Of](#)

```
{  
    // Let us construct the tree shown in above figure  
    struct Node* root = NULL;  
    root = newNode(1);  
    root->left = newNode(2);  
    root->right = newNode(3);  
    root->left->left = newNode(4);  
    root->left->right = newNode(5);  
    root->right->left = newNode(6);  
    root->right->right = newNode(7);  
  
    postOrderIterative(root);  
  
    return 0;  
}
```

Output:

4 5 2 6 7 3 1

This article is compiled by [Aashish Barnwal](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

 LexisNexis® [Learn More](#) 

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree



24

Tweet

3



3

Writing code in comment? Please use ideone.com and share the link here.

14 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



Debabrata · 2 months ago

//Using double stack is simpler

```
struct Stack{
Node *treeNode;
struct Stack *next;
};
typedef struct Stack Stack;
void push(Stack **root,Node *t_Node){
Stack *head = *root,*tmp;
tmp = NALLOC(1,Stack);
tmp->treeNode = t_Node;
tmp->next = (*root);
(*root) = tmp;
}
```

```
Node* pop(Stack **root){
Stack *tmp = *root;
Node* ret;
if(!tmp){
```

[see more](#)

^ | v • Reply • Share ›



Arko • 8 months ago

will this code work??/

```
private static void iterativePostorderSingleStack(Node<integer> root) {
if (root != null) {
StackArray<node<integer>> array = new StackArray<node<integer>>();
array.push(root);
while (!array.isEmpty()) {
Node<integer> node = array.pop();
if (node.left == null && node.right == null)
System.out.print(node.item + " ");
else {
Node<integer> left = node.left;
```

[see more](#)

^ | v • Reply • Share ›



Sumit Gera · 9 months ago

How can I reason for the terminating condition of the outer do-while loop?

^ | v · Reply · Share ›



Trilok Sharma · 10 months ago

```
/*
if (root->right && !isEmpty(stack) && peek(stack) == root->right)
should be used in place of
if (root->right && peek(stack) == root->right)
//-----
// CORRECT CODE IS

void postOrderIterative(struct Node* root)
{
    // Check for empty tree
    if (root == NULL)
        return;

    stack<struct Node *> mystack;

    do
    {
        // Move to leftmost node
```

[see more](#)

1 ^ | v · Reply · Share ›



Trilok Sharma · 10 months ago

```
/*

if (root->right && !isEmpty(stack) && peek(stack) == root->right)
should be used in place of
```

```
if (root->right && peek(stack) == root->right)
```

```
*/
```

^ | v • Reply • Share ›



Geek86 • 10 months ago

Please validate my code for iterative post order traversal..
I have used 2 stacks..

[sourcecode language="java"]

```
public class Client {
```

```
public static void postOrderIterate(BTNode node) {
```

```
Stack<BTNode> processingStack = new Stack<BTNode>();
```

```
Stack<BTNode> resultStack = new Stack<BTNode>();
```

```
processingStack.push(node);
```

```
while (!processingStack.isEmpty()) {
```

```
BTNode processingNode = processingStack.pop();
```

```
if (processingNode.getLeft() != null) {
```

```
processingStack.push(processingNode.getLeft());
```

[see more](#)

1 ^ | v • Reply • Share ›



EOF • 10 months ago

Much simpler solution if we are allowed to destroy the tree :D
Here is a java code.

//JAVA Code

```
static void traverse(Node root){
    Stack<Node> s = new Stack<>();
    Node temp;

    s.push(root);
    while(!s.empty()){
        temp = s.peek();
        if(temp.left != null){
            s.push(temp.left);
            temp.left = null;
        }else if(temp.right != null){
            s.push(temp.right);
            temp.right = null;
        }else if(temp.left == null && temp.right == null){
            System.out.print(temp.keyC + " ");
            s.pop();
        }
    }
}
```

^ | v • Reply • Share ›



Anonym → EOF • a month ago

looks like you could have used a bool to track the visited nodes

^ | v • Reply • Share ›



EOF → EOF • 10 months ago

We can copy the tree before traversing..

^ | v • Reply • Share ›



abhishek08aug · 11 months ago

Intelligent :D

^ | v · Reply · Share ›



Lin Guo · a year ago

[sourcecode language="Java"]

```
static void postOrderTraverse(TreeNode root).
```

```
{
```

```
if(root==null) return;
```

```
Stack<TreeNode> stack=new Stack<TreeNode>();
```

```
stack.push(root);
```

```
TreeNode prevNode=null;
```

```
while(! stack.isEmpty())
```

```
{
```

```
TreeNode curNode=stack.peek();
```

```
if(prevNode==null || prevNode.left==curNode || prevNode.right==curNode).
```

```
{
```

```
if(curNode.left!=null)
```

```
stack.push(curNode.left);
```

```
else if(curNode.right!=null)
```

```
stack.push(curNode.right);
```

```
}else if(curNode.left==prevNode)
```

```
{
```

[see more](#)

^ | v · Reply · Share ›



Viswanath Krishnamurthy · a year ago

There is a another way of doing it, in much simpler way, using only one while I

[sourcecode language="C"]

```
int PostOrderTraversalWithoutRecursion( struct bnode* root ).
```

```
{
```

```

struct bnode* tempptr = root;.
/* Check if the tree is empty */.
if( root == NULL ).
{
printf("ERROR: Tree is Empty n");.
return 0;
}

push( tempptr );.
while (! IsstackEmpty() )
{
tempptr = pop();.
if( tempptr == NULL ).
{

```

[see more](#)

^ | v • Reply • Share ›



Sibendu Dey • a year ago

```

#include
#include

```

```

using namespace std;

```

```

struct node {
int data;
struct node *leftchild;
struct node *rightchild;
};

```

```

struct node * create_node(int data) {
struct node *temp=(struct node *)malloc(sizeof(struct node));
temp->data=data;

```

```
temp->leftchild=temp->rightchild=NULL;
}
```

```
void inorder(struct node *root) {
struct node **stack=(struct node **)malloc(sizeof *stack * 20);
```

[see more](#)

^ | v • Reply • Share ›



Kumar • a year ago

// An iterative function to do postorder traversal of a given binary

```
void postOrderIterative(struct Node* root)
{
    if(root==NULL)
        return;

    struct Stack* stack = createStack(MAX_SIZE);
    push(stack, root);
    struct Node *prev, *curr;
    prev=NULL;

    while(!isEmpty(stack))
    {
        curr=top(stack);
        if(prev==NULL || prev->left==curr || prev->right==curr) //top to bott
        {
            if(curr->left)
                push(stack, curr->left);
```

[see more](#)

^ | v • Reply • Share ›

 [Subscribe](#)

 [Add Disqus to your site](#)

@geeksforgeeks, **Some rights reserved**

[Contact Us!](#)

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team