Inorder Tree Traversal without Recursion

Using Stack is the obvious way to traverse tree without recursion. Below is an algorithm for traversing binary tree using stack. See this for step wise step execution of the algorithm.

```
1) Create an empty stack S.
2) Initialize current node as root
3) Push the current node to S and set current = current->left until current is NULL
4) If current is NULL and stack is not empty then
     a) Pop the top item from stack.
     b) Print the popped item, set current = current->right
     c) Go to step 3.
5) If current is NULL and stack is empty then we are done.
```

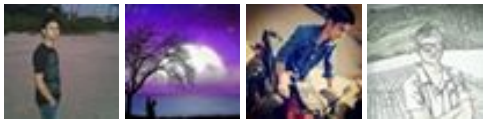
Let us consider the below tree for example

```
            1
          /   \
        2      3
      /  \
    4     5

Step 1 Creates an empty stack: S = NULL

Step 2 sets current as address of root: current -> 1

Step 3 Pushes the current node and set current = current->left until current is NULL
     current -> 1
```

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
     push 1: Stack S -> 1
     current -> 2
     push 2: Stack S -> 2, 1
     current -> 4
     push 4: Stack S -> 4, 2, 1
     current = NULL

Step 4 pops from S
     a) Pop 4: Stack S -> 2, 1
     b) print "4"
     c) current = NULL /*right of 4 */ and go to step 3
Since current is NULL step 3 doesn't do anything.

Step 4 pops again.
     a) Pop 2: Stack S -> 1
     b) print "2"
     c) current -> 5/*right of 2 */ and go to step 3

Step 3 pushes 5 to stack and makes current NULL
     Stack S -> 5, 1
     current = NULL

Step 4 pops from S
     a) Pop 5: Stack S -> 1
     b) print "5"
     c) current = NULL /*right of 5 */ and go to step 3
Since current is NULL step 3 doesn't do anything

Step 4 pops again.
     a) Pop 1: Stack S -> NULL
     b) print "1"
     c) current -> 3 /*right of 5 */

Step 3 pushes 3 to stack and makes current NULL
     Stack S -> 3
```


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
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
    current = NULL

Step 4 pops from S
    a) Pop 3: Stack S -> NULL
    b) print "3"
    c) current = NULL /*right of 3 */

Traversal is done now as stack S is empty and current is NULL.
```

Implementation:

```
#include<stdio.h>
#include<stdlib.h>
#define bool int

/* A binary tree tNode has data, pointer to left child
   and a pointer to right child */
struct tNode
{
   int data;
   struct tNode* left;
   struct tNode* right;
};

/* Structure of a stack node. Linked List implementation is used for
   stack. A stack node contains a pointer to tree node and a pointer t
   next stack node */
struct sNode
{
  struct tNode *t;
  struct sNode *next;
};

/* Stack related functions */
void push(struct sNode** top_ref, struct tNode *t);
struct tNode *pop(struct sNode** top_ref);
bool isEmpty(struct sNode *top);

/* Iterative function for inorder tree traversal */
void inOrder(struct tNode *root)
{
  /* set current to root of binary tree */
```

```
  struct tNode *current = root;
  struct sNode *s = NULL;  /* Initialize stack s */
  bool done = 0;

  while (!done)
  {
    /* Reach the left most tNode of the current tNode */
    if(current !=  NULL)
    {
      /* place pointer to a tree node on the stack before traversing
        the node's left subtree */
      push(&s, current);
      current = current->left;
    }

    /* backtrack from the empty subtree and visit the tNode
       at the top of the stack; however, if the stack is empty,
      you are done */
    else
    {
      if (!isEmpty(s))
      {
        current = pop(&s);
        printf("%d ", current->data);

        /* we have visited the node and its left subtree.
          Now, it's right subtree's turn */
        current = current->right;
      }
      else
        done = 1;
    }
  } /* end of while */
}

/* UTILITY FUNCTIONS */
/* Function to push an item to sNode*/
void push(struct sNode** top_ref, struct tNode *t)
{
  /* allocate tNode */
  struct sNode* new_tNode =
            (struct sNode*) malloc(sizeof(struct sNode));

  if(new_tNode == NULL)
  {
     printf("Stack Overflow \n");
     getchar();
```


Recent Comments

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

Backtracking | Set 7 (Sudoku) · 43 minutes ago

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

Flipkart Interview | Set 6 · 1 hour ago

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

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour 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 · 3 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

```
      exit(0);
   }

   /* put in the data  */
   new_tNode->t  = t;

   /* link the old list off the new tNode */
   new_tNode->next = (*top_ref);

   /* move the head to point to the new tNode */
   (*top_ref)    = new_tNode;
}

/* The function returns true if stack is empty, otherwise false */
bool isEmpty(struct sNode *top)
{
   return (top == NULL)? 1 : 0;
}

/* Function to pop an item from stack*/
struct tNode *pop(struct sNode** top_ref)
{
   struct tNode *res;
   struct sNode *top;

   /*If sNode is empty then error */
   if(isEmpty(*top_ref))
   {
      printf("Stack Underflow \n");
      getchar();
      exit(0);
   }
   else
   {
      top = *top_ref;
      res = top->t;
      *top_ref = top->next;
      free(top);
      return res;
   }
}

/* Helper function that allocates a new tNode with the
   given data and NULL left and right pointers. */
struct tNode* newtNode(int data)
{
   struct tNode* tNode = (struct tNode*)
```

```
                        malloc(sizeof(struct tNode));
  tNode->data = data;
  tNode->left = NULL;
  tNode->right = NULL;

  return(tNode);
}

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

  /* Constructed binary tree is
            1
          /   \
        2       3
      /  \
    4     5
  */
  struct tNode *root = newtNode(1);
  root->left        = newtNode(2);
  root->right       = newtNode(3);
  root->left->left  = newtNode(4);
  root->left->right = newtNode(5);

  inOrder(root);

  getchar();
  return 0;
}
```

Time Complexity: O(n)

References:
http://web.cs.wpi.edu/~cs2005/common/iterative.inorder
http://neural.cs.nthu.edu.tw/jang/courses/cs2351/slide/animation/Iterative%20Inorder%20Traversal.pps

See this post for another approach of Inorder Tree Traversal without recursion and without stack!

Please write comments if you find any bug in above code/algorithm, or want to share more information about stack based Inorder Tree Traversal.

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

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

23 Comments GeeksforGeeks

Sort by Newest

Babaji • 2 months ago

My solution solves all traversals using a simple and elegant 'dummy' node me
and a dummy node onto the stack according to our traversal order and everyt
dummy node represents a leaf node where both right and left children are null
stack, just print its value and continue onto the next iteration of the loop. This n
recursion actually works. Here is some code:

```
struct Node
{
Node* left;
Node* right;
int data;

Node(int d) : left(nullptr), right(nullptr), data(d) { }

bool isLeaf() { return ((left == nullptr) && (right == nullptr)); }
};

void InOrder(Node* root, Stack s)
{
```

see more

• Reply • Share ›

bikash • 4 months ago

I guess the above code can be simplified a little:

```
void inOrder(struct tNode *root)
{
/* set current to root of binary tree */
struct tNode *current = root;
```

```
struct sNode *s = NULL; /* initialize stack s */

while (!isEmpty(s) || current)
{
if(current)
{
/* place pointer to a tree node on the stack before traversing
the node's left subtree */
push(&s, current);
current = current->left;
} else {
current = pop(&s);
printf("%d ", current->data);
/* we have visited the node and its left subtree. Now, it's right subtree's turn */
current = current->right;
}
}
}
```

1 • Reply • Share ›

superaghu • 6 months ago

```
public static void InOrderIterative(TreeNode root)
{
Console.WriteLine("In Order Iterative");
var stack = new Stack<treenode>();
var node = root;
if (node != null)
{
stack.Push(node);
while (stack.Count > 0)
{
//stack.Push(node);
```

```
while (node.LeftChild != null)
{
node = node.LeftChild;
stack.Push(node);
}
node = stack.Pop();
Console.Write(node.Data + " -->");
if (node.RightChild != null)
{
node = node.RightChild;
stack.Push(node);
}
}
}
}
```

• Reply • Share ›

chinnisasi • 10 months ago

```
void inOrderTraversalWithOutStack(struct node* node)   {
        std::stack<struct node*> stk;
        struct node* current = node;
        while(current)  {
                while(current)  {
                        stk.push(current);
                        current = current->left;
                }
                while(current == NULL && !stk.empty())  {
                        current = stk.top();
                        stk.pop();
                        printf("%d \t", current->data);
                        current = current->right;
                }
```

```
        }
}
```

7 • Reply • Share ›

Ashish → chinnisasi • a month ago

error, as when ur left side node of right subtree of left child .:)

• Reply • Share ›

Nitesh • 10 months ago

```
 void pre_ordr(node *tree)
{
     vector<node*> v;
     if(tree == NULL)
       return;

     node *root = tree;
     v.push_back(root);
     while(v.size())
     {
          while(root->left != NULL)
          {
             v.push_back(root->left);
             root = root->left;
          }
          node *tmp = v.back();
          v.pop_back();
          cout<<"Data = "<<tmp->data<<endl;
```

see more

• Reply • Share ›

alexander.korobeynikov • 11 months ago

So many variations here, but still no proper/optimal implementation. I say "opti

```
 template <typename TreeNode, typename Visitor>
void binary_tree_traverse_inorder(TreeNode* node, Visitor visit)
{
    std::stack<TreeNode*> stack;
    while (node || !stack.empty()) {
        if (node) {
            // go left as far as possible, push to the stack
            stack.push(node);
            node = node->left;
        } else {
            // we are at the bottom, pop from the stack
            node = stack.top();
            stack.pop();
            visit(node);
            node = node->right;
        }
    }
}
```

• Reply • Share ›

samsammy ➦ alexander.korobeynikov • 7 months ago

node=stack.top() ; and stack.pop(); can be correctly written as a single node=stack.pop();

As pop operation in stack deletes the object in top and returns it.

• Reply • Share ›

moussa → samsammy • 3 months ago

r u sure?

http://www.cplusplus.com/refer...

• Reply • Share ›

Apoorv Srivastava • 11 months ago

how come in step 4c of the stepwise execution of algorithm current -> 3 /*righ

• Reply • Share ›

Zain Kazami • 11 months ago

you are right amit kumar.

• Reply • Share ›

Choudhary Amit Kumar • a year ago

Are yaar yes data structure to ple hi nhi padti.

• Reply • Share ›

Vijay Sharma • a year ago

```
PrintInorder(struct node* head).
{
If head==NULL return;.
struct stack *s=createstack();.
while(1)
{
while(head)
{
push(s, head);
head=head->left;
}
If IsEmpty(S) break;.
head=pop(s);
```

```
printf("%d", head->data);
head=head->right;
}
deletestack(s);
}
```

• Reply • Share ›

Sakthi Kumaran Suriya • a year ago

Here is the simplified non-recursive traversal (functionally equivalent to recursi

void traverse(link h, void visit(link)) {.

STACK<link> s(max);.

s.push(h);.

while (! s.empty()) {

visit(h = s.pop());.

if (h->r!= 0) s.push(h->r);.

if (h->l!= 0) s.push(h->l);.

}.
}

this is pre-order traversal, note that r should be pushed before l (unlike recursi

For level order traversal, just change the underlying data structure from stack

• Reply • Share ›

Abhi • a year ago

```
template <class T>
```

```
void BinaryTree<T>::IterativeInorder(T* & outAr, const int& size)
{
    BinaryTreeNode<T>* current = m_Root;
    stack<BinaryTreeNode<T>*> traversalStack;


    int iInorder = 0;
    DeleteAr(outAr);
    outAr = new int [size];

    cout << "\nInorder iterative traversal: ";

    while (current || traversalStack.empty() == false)
    {
        if (!current)
        {
            while (current == NULL && traversalStack.empty() == false
```

see more

Reply • Share ›

Avinash • 2 years ago

```
PrintInorder(struct node* head)
{
        If head==NULL return;
        struct stack *s=createstack();
        while(1)
        {
                while(head)
                {
                        push(s,head);
```

```
                }
                If IsEmpty(S) break;
                head=pop(s);
                printf("%d",head->data);
                head=head->right;
        }
        deletestack(s);
}
```

• Reply • Share ›

bhupender • 3 years ago

well i think this code will crash as you are referencing a NULL pointer at
/* place pointer to a tree node on the stack before traversing
the node's left subtree
*/
push(&s, current);

• Reply • Share ›

manishj • 3 years ago

Although the above algorithm works , it is not eqivalent to the corresponding re
difference is that it pops an element from stack , before its right sub-tree is vis
an element is popped from stack only after its right-subtree has been traverse

We need to change the algorithm as follows to work correctly :
Step 4 don't pop the value from stack , but read the value from top of stack .
Step5 check if value has already been visited , if it is visited only-then pop the
NULL. Repeat from step 3.Else mark it visited and ,set current = current->righ

Following is the program that illustrates the point :

```
void inorder(btree *root)
{
        btree * current = root;
        stack<btree *> st;
        bool done = false;

        int index = 0;
```

see more

Reply • Share ›

dev • 4 years ago

```
template<typename E,typename K>
void  BST<E,K>::InOrderIterative(BSTNode<K> *root)
{
        BSTNode<K>* current = root->leftChild;
        std::stack<BSTNode<K>*> stk;
        stk.push(root);
        while(!stk.empty()) {
                while(current != NULL) {
                        stk.push(current);
                        current = current->leftChild;
                }
                BSTNode<K>* temp = stk.top();
                stk.pop();
                std::cout<<temp->GetData()<<"\n";
                current = temp->rightChild;
        }
}
```

Reply • Share ›

alveko ➦ dev • 11 months ago

there is a bug in the first while-condition. it shall also include (... || curre

• Reply • Share ›

gauravs • 4 years ago

instead of writing

```
if(current !=  NULL)
    {

      push(&s, current);
      current = current->left;
    }
```

we can write

```
while(current !=  NULL)
    {
      push(&s, current);
      current = current->left;
    }
```

• Reply • Share ›

tech.login.id2 • 4 years ago

Wouldn't the below be a simpler implementation of the algo?

```
void inOrder (Tree *current) {
```

```
while (current) {
    if (current->left) {
        push (current);
        current = current->left;
    } else {
        print (current);
        while (current && current->right == NULL)
            current = pop ();

        current = current->right;
    }
}

}
```

1 • Reply • Share ›

amitmitra83 ➦ tech.login.id2 • 3 years ago

This is fine for C 0r C++, but if you are jumping in java with this simpler with STACK Exception.

• Reply • Share ›

Subscribe Add Disqus to your site