GeeksforGeeks

A computer science portal for geeks

Placements Practice GATE CS IDE Q&A GeeksQuiz

Inorder Non-threaded Binary Tree Traversal without Recursion or Stack

We have discussed Thread based Morris Traversal. Can we do inorder traversal without threads if we have parent pointers available to us?

```
Input: Root of Below Tree [Every node of
       tree has parent pointer also]
        10
      /    \
     5      100
           /  \
          80   120
Output: 5 10 80 100 120
The code should not extra space (No Recursion
and stack)
```

We strongly recommend you to minimize your browser and try this yourself first.

In inorder traversal, we follow "left root right". We can move to children using left and right pointers. Once a node is visited, we need to move to parent also. For example, in the above tree, we need to move to 10 after printing 5. For this purpose, we use parent pointer. Below is algorithm.

```
1. Initialize current node as root
2. Initialize a flag: leftdone = false;
3. Do following while root is not NULL
     a) If leftdone is false, set current node as leftmost
        child of node.
     b) Mark leftdone as true and print current node.
     c) If right child of current nodes exists, set current
        as right child and set leftdone as false.
     d) Else If parent exists, If current node is left child
        of its parent, set current node as parent.
        If current node is right child, keep moving to ancestors
        using parent pointer while current node is right child
        of its parent.
     e) Else break (We have reached back to root)
```

Illustration:

```
Let us consider below tree for illustration.
        10
      /    \
     5      100
           /  \
          80  120

Initialize: Current node = 10, leftdone = false

Since leftdone is false, we move to 5 (3.a), print it
and set leftdone = true.

Now we move to parent of 5 (3.d). Node 10 is
printed because leftdone is true.

We move to right of 10 and set leftdone as false (3.c)

Now current node is 100. Since leftdone is false, we move
to 80 (3.a) and set leftdone as true.  We print current
node 80 and move back to parent 100 (3.d).  Since leftdone
is true, we print current node 100.

Right of 100 exists, so we move to 120 (3.c).   We print
current node 120.

Since 120 is right child of its parent we keep moving to parent
while parent is right child of its parent.  We reach root. So
we break the loop and stop
```

Below is C++ implementation of above algorithm. Note that the implementation uses Binary Search Tree instead of Binary Tree. We can use the same function **inorder()** for Binary Tree also. The reason for using Binary Search Tree in below code is, it is easy to construct a Binary Search Tree with parent pointers and easy to test the outcome (In BST inorder traversal is always sorted).

```
// C++ program to print inorder traversal of a Binary Search
// Tree (BST) without recursion and stack
#include <bits/stdc++.h>
using namespace std;

// BST Node
struct Node
{
    Node *left, *right, *parent;
    int key;
};

// A utility function to create a new BST node
Node *newNode(int item)
```

```
{
    Node *temp = new Node;
    temp->key = item;
    temp->parent = temp->left = temp->right = NULL;
    return temp;
}

/* A utility function to insert a new node with
   given key in BST */
Node *insert(Node *node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

    /* Otherwise, recur down the tree */
    if (key < node->key)
    {
        node->left  = insert(node->left, key);
        node->left->parent = node;
    }
    else if (key > node->key)
    {
        node->right = insert(node->right, key);
        node->right->parent = node;
    }

    /* return the (unchanged) node pointer */
    return node;
}

// Function to print inorder traversal using parent
// pointer
void inorder(Node *root)
{
    bool leftdone = false;

    // Start traversal from root
    while (root)
    {
        // If left child is not traversed, find the
        // leftmost child
        if (!leftdone)
        {
            while (root->left)
                root = root->left;
        }

        // Print root's data
        printf("%d ", root->key);

        // Mark left as done
        leftdone = true;

        // If right child exists
        if (root->right)
        {
            leftdone = false;
            root = root->right;
        }

        // If right child doesn't exist, move to parent
        else if (root->parent)
        {
            // If this node is right child of its parent,
            // visit parent's parent first
            while (root->parent &&
                   root == root->parent->right)
                root = root->parent;
            if (!root->parent)
                break;
            root = root->parent;
        }
        else break;
    }
}

int main(void)
{
    Node * root = NULL;

    root = insert(root, 24);
    root = insert(root, 27);
    root = insert(root, 29);
    root = insert(root, 34);
    root = insert(root, 14);
    root = insert(root, 4);
    root = insert(root, 10);
    root = insert(root, 22);
    root = insert(root, 13);
    root = insert(root, 3);
    root = insert(root, 2);
    root = insert(root, 6);

    printf("Inorder traversal is \n");
    inorder(root);

    return 0;
}
```

Run on IDE

Output:

```
Inorder traversal is
2 3 4 6 10 13 14 22 24 27 29 34
```

This article is contributed by **Rishi Chhibber**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

7 Comments Category: Trees

Related Posts:

- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- Construct a Binary Search Tree from given postorder
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Check if leaf traversal of two Binary Trees is same?
- Closest leaf to a given node in Binary Tree
- Locking and Unlocking of Resources arranged in the form of n-ary Tree
- Find all possible binary trees with given Inorder Traversal

(Login to Rate and Mark)

4 Average Difficulty : **4/5.0**
Based on **3** vote(s)

□ Add to TODO List

□ Mark as DONE

Like Share 8 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

7 Comments **GeeksforGeeks** 1 **Login**

Recommend 1 Share Sort by Newest

Join the discussion…

LOGAN • 19 days ago

http://www.universeforyour.com...
http://www.universeforyour.com...
http://www.universeforyour.com...

⌃ | ⌄ • Reply • Share ›

copperfield • a month ago

just the one that I was searching for, I first find the Morris one but I don't like that it modify the try even if temporary, I like this better :)

⌃ | ⌄ • Reply • Share ›

.NetGeek • 2 months ago

@GeeksforGeeks: It doesn't work, if you add a Left Node in any of the node in RightSubtree. e.g. Adding this root = insert(root, 25); wont work as it result into:

Inorder traversal is
2 3 4 6 10 13 14 22 24 25 27 29 34 24

Please correct.

⌃ | ⌄ • Reply • Share ›

Rishi Chhibber ➦ .NetGeek • 2 months ago

The output is correct even with 25 added.

I tested it just now at ideone.

http://ideone.com/e.js/1XjDDM

⌃ | ⌄ • Reply • Share ›

.NetGeek ➦ Rishi Chhibber • 2 months ago

This is working fine now. However, it was giving above mentioned output after adding node(25) yesterday :(.
Moreover, this(http://ideone.com/e.js/1XjDDM) is giving run time error if I just execute it with 1 or 2 node. Please check.

⌃ | ⌄ • Reply • Share ›

GeeksforGeeks Mod ➦ .NetGeek • 2 months ago

The produced output seems correct. Please note that the implementation uses Binary Search Tree. And Inorder traversal of Binary Search Tree is always sorted order. Please correct me if I am wrong.

⌃ | ⌄ • Reply • Share ›

.NetGeek ➦ GeeksforGeeks • 2 months ago iNetGeek → GeeksforGeeks • 2 months ago

I agree that inorder traversal must produce output in sorted order. However, the implementation of BST or Inorder traversal (as is done here) could go wrong.

^ | v • Reply • Share ›