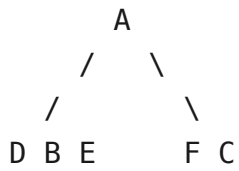
Construct Tree from given Inorder and Preorder traversals

Let us consider the below traversals:

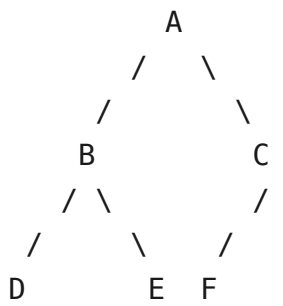
Inorder sequence: D B E A F C
Preorder sequence: A B D E C F

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

In a Preorder sequence, leftmost element is the root of the tree. So we know 'A' is root for given sequences. By searching 'A' in Inorder sequence, we can find out all elements on left side of 'A' are in left subtree and elements on right are in right subtree. So we know below structure now.



We recursively follow above steps and get the following tree.



pick next element in next recursive call.

2) Create a new tree node tNode with the data as picked element.

3) Find the picked element's index in Inorder. Let the index be inIndex.

4) Call buildTree for elements before inIndex and make the built tree as left subtree of tNode.

5) Call buildTree for elements after inIndex and make the built tree as right subtree of tNode.

6) return tNode.

Thanks to Rohini and Tushar for suggesting the code.

C

```c
/* program to construct tree using inorder and preorder traversals */
#include <stdio.h>
#include <stdlib.h>

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

/* Prototypes for utility functions */
int search(char arr[], int strt, int end, char value);
struct node* newNode(char data);

/* Recursive function to construct binary of size len from
   Inorder traversal in[] and Preorder traversal pre[].  Initial values
   of inStrt and inEnd should be 0 and len -1.  The function doesn't
   do any error checking for cases where inorder and preorder
   do not form a tree */
struct node* buildTree(char in[], char pre[], int inStrt, int inEnd)
{
    static int preIndex = 0;

    if (inStrt > inEnd)
        return NULL;

    /* Pick current node from Preorder traversal using preIndex
    and increment preIndex */
    struct node* tNode = newNode(pre[preIndex++]);

    /* If this node has no children then return */
    if (inStrt == inEnd)
        return tNode;

    /* Else find the index of this node in Inorder traversal */
    int inIndex = search(in, inStrt, inEnd, tNode->data);
```

```
    tNode->right = buildTree(in, pre, inIndex + 1, inEnd);

    return tNode;
}

/* UTILITY FUNCTIONS */
/* Function to find index of value in arr[start...end]
   The function assumes that value is present in in[] */
int search(char arr[], int strt, int end, char value)
{
    int i;
    for (i = strt; i <= end; i++) {
        if (arr[i] == value)
            return i;
    }
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(char data)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

/* This funtcion is here just to test buildTree() */
void printInorder(struct node* node)
{
    if (node == NULL)
        return;

    /* first recur on left child */
    printInorder(node->left);

    /* then print the data of node */
    printf("%c ", node->data);

    /* now recur on right child */
    printInorder(node->right);
}

/* Driver program to test above functions */
int main()
{
    char in[] = { 'D', 'B', 'E', 'A', 'F', 'C' };
    char pre[] = { 'A', 'B', 'D', 'E', 'C', 'F' };
    int len = sizeof(in) / sizeof(in[0]);
    struct node* root = buildTree(in, pre, 0, len - 1);
```

```
    getchar();
}
```

Java

```
// Java program to construct a tree using inorder and preorder traversal

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
class Node {
    char data;
    Node left, right;

    Node(char item)
    {
        data = item;
        left = right = null;
    }
}

class BinaryTree {
    Node root;
    static int preIndex = 0;

    /* Recursive function to construct binary of size len from
       Inorder traversal in[] and Preorder traversal pre[].
       Initial values of inStrt and inEnd should be 0 and len -1.
       The function doesn't do any error checking for cases where
       inorder and preorder do not form a tree */
    Node buildTree(char in[], char pre[], int inStrt, int inEnd)
    {
        if (inStrt > inEnd)
            return null;

        /* Pick current node from Preorder traversal using preIndex
           and increment preIndex */
        Node tNode = new Node(pre[preIndex++]);

        /* If this node has no children then return */
        if (inStrt == inEnd)
            return tNode;

        /* Else find the index of this node in Inorder traversal */
        int inIndex = search(in, inStrt, inEnd, tNode.data);

        /* Using index in Inorder traversal, construct left and
           right subtress */
        tNode.left = buildTree(in, pre, inStrt, inIndex - 1);
        tNode.right = buildTree(in, pre, inIndex + 1, inEnd);
```

```
    /* Function to find index of value in arr[start...end]
     The function assumes that value is present in in[] */
    int search(char arr[], int strt, int end, char value)
    {
        int i;
        for (i = strt; i <= end; i++) {
            if (arr[i] == value)
                return i;
        }
        return i;
    }

    /* This funtcion is here just to test buildTree() */
    void printInorder(Node node)
    {
        if (node == null)
            return;

        /* first recur on left child */
        printInorder(node.left);

        /* then print the data of node */
        System.out.print(node.data + " ");

        /* now recur on right child */
        printInorder(node.right);
    }

    // driver program to test above functions
    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();
        char in[] = new char[] { 'D', 'B', 'E', 'A', 'F', 'C' };
        char pre[] = new char[] { 'A', 'B', 'D', 'E', 'C', 'F' };
        int len = in.length;
        Node root = tree.buildTree(in, pre, 0, len - 1);

        // building the tree by printing inorder traversal
        System.out.println("Inorder traversal of constructed tree is : ");
        tree.printInorder(root);
    }
}

// This code has been contributed by Mayank Jaiswal
```

Python

```
# Python program to construct tree using inorder and
```

```
    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

"""Recursive function to construct binary of size len from
   Inorder traversal in[] and Preorder traversal pre[].  Initial values
   of inStrt and inEnd should be 0 and len -1.  The function doesn't
   do any error checking for cases where inorder and preorder
   do not form a tree """
def buildTree(inOrder, preOrder, inStrt, inEnd):

    if (inStrt > inEnd):
        return None

    # Pich current node from Preorder traversal using
    # preIndex and increment preIndex
    tNode = Node(preOrder[buildTree.preIndex])
    buildTree.preIndex += 1

    # If this node has no children then return
    if inStrt == inEnd :
        return tNode

    # Else find the index of this node in Inorder traversal
    inIndex = search(inOrder, inStrt, inEnd, tNode.data)

    # Using index in Inorder Traversal, construct left
    # and right subtrees
    tNode.left = buildTree(inOrder, preOrder, inStrt, inIndex-1)
    tNode.right = buildTree(inOrder, preOrder, inIndex + 1, inEnd)

    return tNode

# UTILITY FUNCTIONS
# Function to find index of vaue in arr[start...end]
# The function assumes that value is rpesent in inOrder[]

def search(arr, start, end, value):
    for i in range(start, end + 1):
        if arr[i] == value:
            return i

def printInorder(node):
    if node is None:
        return

    # first recur on left child
    printInorder(node.left)
```

```
    printInorder(node.right)

# Driver program to test above function
inOrder = ['D', 'B', 'E', 'A', 'F', 'C']
preOrder = ['A', 'B', 'D', 'E', 'C', 'F']
# Static variable preIndex
buildTree.preIndex = 0
root = buildTree(inOrder, preOrder, 0, len(inOrder)-1)

# Let us test the build tree by priting Inorder traversal
print "Inorder traversal of the constructed tree is"
printInorder(root)

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

C#

```
// C# program to construct a tree using
// inorder and preorder traversal
using System;

/* A binary tree node has data, pointer
to left child and a pointer to right child */
public class Node
{
    public char data;
    public Node left, right;

    public Node(char item)
    {
        data = item;
        left = right = null;
    }
}

class GFG
{
public Node root;
public static int preIndex = 0;

/* Recursive function to construct binary
of size len from Inorder traversal in[]
and Preorder traversal pre[]. Initial values
of inStrt and inEnd should be 0 and len -1.
The function doesn't do any error checking for
cases where inorder and preorder do not form a tree */
public virtual Node buildTree(char[] arr, char[] pre,
                              int inStrt, int inEnd)
{
    if (inStrt > inEnd)
```

```
    /* Pick current node from Preorder traversal
     using preIndex and increment preIndex */
    Node tNode = new Node(pre[preIndex++]);

    /* If this node has no children then return */
    if (inStrt == inEnd)
    {
        return tNode;
    }

    /* Else find the index of this
       node in Inorder traversal */
    int inIndex = search(arr, inStrt,
                         inEnd, tNode.data);

    /* Using index in Inorder traversal,
    construct left and right subtress */
    tNode.left = buildTree(arr, pre, inStrt, inIndex - 1);
    tNode.right = buildTree(arr, pre, inIndex + 1, inEnd);

    return tNode;
}

/* UTILITY FUNCTIONS */

/* Function to find index of value in arr[start...end]
The function assumes that value is present in in[] */
public virtual int search(char[] arr, int strt,
                          int end, char value)
{
    int i;
    for (i = strt; i <= end; i++)
    {
        if (arr[i] == value)
        {
            return i;
        }
    }
    return i;
}

/* This funtcion is here just to test buildTree() */
public virtual void printInorder(Node node)
{
    if (node == null)
    {
        return;
    }

    /* first recur on left child */
    printInorder(node.left);
```

```
    printInorder(node.right);
}

// Driver Code
public static void Main(string[] args)
{
    GFG tree = new GFG();
    char[] arr = new char[] {'D', 'B', 'E', 'A', 'F', 'C'};
    char[] pre = new char[] {'A', 'B', 'D', 'E', 'C', 'F'};
    int len = arr.Length;
    Node root = tree.buildTree(arr, pre, 0, len - 1);

    // building the tree by printing inorder traversal
    Console.WriteLine("Inorder traversal of " +
                      "constructed tree is : ");
    tree.printInorder(root);
}
}

// This code is contributed by Shrikant13
```

Output :

```
Inorder traversal of constructed tree is :
D B E A F C
```

Time Complexity: O(n^2). Worst case occurs when tree is left skewed. Example Preorder and Inorder traversals for worst case are {A, B, C, D} and {D, C, B, A}.

Efficient Approach :

We can optimize the above solution using hashing (unordered_map in C++ or HashMap in Java). We store indexes of inorder traversal in a hash table. So that search can be done O(1) time.

C++

```
/* C++ program to construct tree using inorder
   and preorder traversals */
#include <bits/stdc++.h>
using namespace std;
```

```
    struct Node* left;
    struct Node* right;
};

struct Node* newNode(char data)
{
    struct Node* node = new Node;
    node->data = data;
    node->left = node->right = NULL;
    return (node);
}

/* Recursive function to construct binary of size
len from Inorder traversal in[] and Preorder traversal
pre[]. Initial values of inStrt and inEnd should be
0 and len -1. The function doesn't do any error
checking for cases where inorder and preorder
do not form a tree */
struct Node* buildTree(char in[], char pre[], int inStrt,
                       int inEnd, unordered_map<char, int>& mp)
{
    static int preIndex = 0;

    if (inStrt > inEnd)
        return NULL;

    /* Pick current node from Preorder traversal using preIndex
    and increment preIndex */
    int curr = pre[preIndex++];
    struct Node* tNode = newNode(curr);

    /* If this node has no children then return */
    if (inStrt == inEnd)
        return tNode;

    /* Else find the index of this node in Inorder traversal */
    int inIndex = mp[curr];

    /* Using index in Inorder traversal, construct left and
    right subtress */
    tNode->left = buildTree(in, pre, inStrt, inIndex - 1, mp);
    tNode->right = buildTree(in, pre, inIndex + 1, inEnd, mp);

    return tNode;
}

// This function mainly creates an unordered_map, then
// calls buildTree()
struct Node* buldTreeWrap(char in[], char pre[], int len)
{
    // Store indexes of all items so that we
    // we can quickly find later
```

```
    return buildTree(in, pre, 0, len - 1, mp);
}

/* This funtcion is here just to test buildTree() */
void printInorder(struct Node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%c ", node->data);
    printInorder(node->right);
}

/* Driver program to test above functions */
int main()
{
    char in[] = { 'D', 'B', 'E', 'A', 'F', 'C' };
    char pre[] = { 'A', 'B', 'D', 'E', 'C', 'F' };
    int len = sizeof(in) / sizeof(in[0]);

    struct Node* root = buldTreeWrap(in, pre, len);

    /* Let us test the built tree by printing
      Insorder traversal */
    printf("Inorder traversal of the constructed tree is \n");
    printInorder(root);
}
```

Time Complexity : O(n)

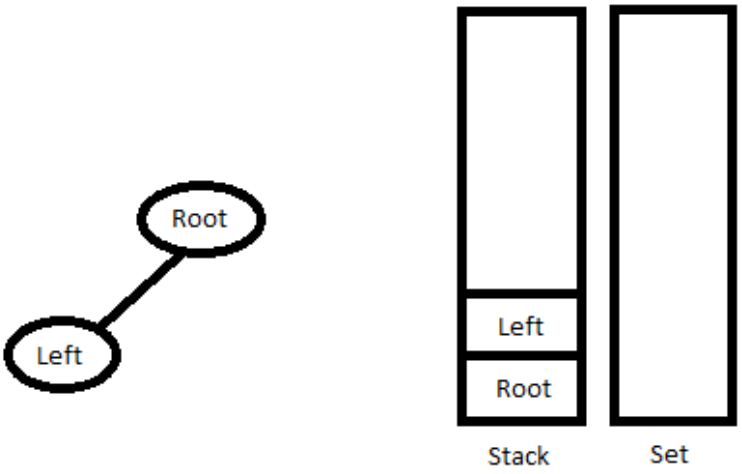
Another approach :

Use the fact that InOrder traversal is Left-Root-Right and PreOrder traversal is Root-Left-Right. Also, first node in the PreOrder traversal is always the root node and the first node in the InOrder traversal is the leftmost node in the tree.

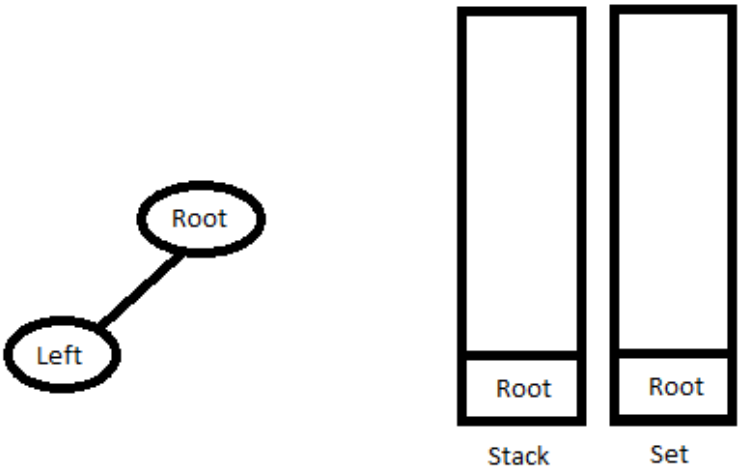
Maintain two data-structures : Stack (to store the path we visited while traversing PreOrder array) and Set (to maintain the node in which the next right subtree is expected).

1. Do below until you reach the leftmost node.
Keep creating the nodes from PreOrder traversal
If the stack's topmost element is not in the set, link the created node to the left child of stack's

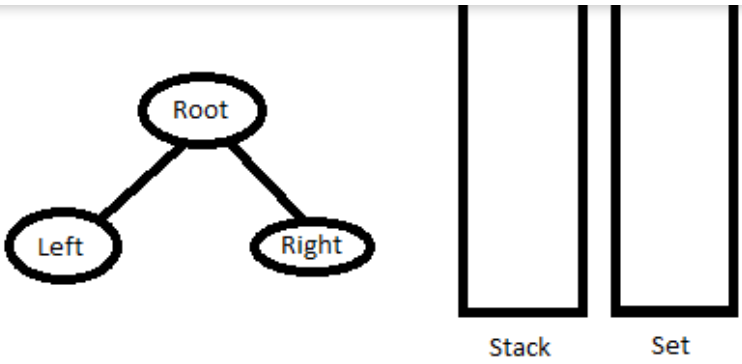
topmost element from the set and the stack.
Push the node to a stack.



2. Keep popping the nodes from the stack until either the stack is empty, or the topmost element of stack compares to the current element of InOrder traversal. Once the loop is over, push the last node back into the stack and into the set.



3. Goto Step 1.




```
// Java program to construct a tree using inorder and preorder traversal
import java.util.*;

public class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int x) { val = x; }
}

class BinaryTree {
    static Set<TreeNode> set = new HashSet<>();
    static Stack<TreeNode> stack = new Stack<>();

    // Function to build tree using given traversal
    public TreeNode buildTree(int[] preorder, int[] inorder)
    {

        TreeNode root = null;
        for (int pre = 0, in = 0; pre < preorder.length;) {

            TreeNode node = null;
            do {
                node = new TreeNode(preorder[pre]);
                if (root == null) {
                    root = node;
                }
                if (!stack.isEmpty()) {
                    if (set.contains(stack.peek())) {
                        set.remove(stack.peek());
                        stack.pop().right = node;
                    }
                    else {
                        stack.peek().left = node;
                    }
                }
                stack.push(node);
            } while (preorder[pre++] != inorder[in] && pre < preorder.length);
```

```
                in++;
            }

            if (node != null) {
                set.add(node);
                stack.push(node);
            }
        }

        return root;
    }

    // Function to print tree in Inorder
    void printInorder(TreeNode node)
    {
        if (node == null)
            return;

        /* first recur on left child */
        printInorder(node.left);

        /* then print the data of node */
        System.out.print(node.val + " ");

        /* now recur on right child */
        printInorder(node.right);
    }

    // driver program to test above functions
    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();

        int in[] = new int[] { 9, 8, 4, 2, 10, 5, 10, 1, 6, 3, 13, 12, 7 };
        int pre[] = new int[] { 1, 2, 4, 8, 9, 5, 10, 10, 3, 6, 7, 12, 13 };
        int len = in.length;

        TreeNode root = tree.buildTree(pre, in);

        tree.printInorder(root);
    }
}
```

Output :

```
 9 8 4 2 10 5 10 1 6 3 13 12 7
```

Thanks Hardik Agarwal for suggesting this approach.

the same problem.

Recommended Posts:

Tree Traversals (Inorder, Preorder and Postorder)
Check if given Preorder, Inorder and Postorder traversals are of same tree
Construct a tree from Inorder and Level order traversals | Set 2
Construct a tree from Inorder and Level order traversals | Set 1
Construct Full Binary Tree from given preorder and postorder traversals
Preorder from Inorder and Postorder traversals
Print Postorder traversal from given Inorder and Preorder traversals
Construct Full Binary Tree using its Preorder traversal and Preorder traversal of its mirror tree
Construct the full k-ary tree from its preorder traversal
Construct a special tree from given preorder traversal
Construct a Binary Tree from Postorder and Inorder
Construct Special Binary Tree from given Inorder traversal
Construct BST from given preorder traversal | Set 2
Iterative Preorder Traversal of an N-ary Tree
Preorder Traversal of N-ary Tree Without Recursion

Improved By : shrikanth13

Article Tags : Tree cpp-unordered_map Inorder Traversal Microsoft Preorder Traversal tree-traversal

Practice Tags : Microsoft Tree

/

☐ To-do ☐ Done

3.5

Based on **332** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

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

Load Comments | Share this post!