

# Construct a Binary Tree from Postorder and Inorder

Given Postorder and Inorder traversals, construct the tree.

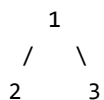
Examples:

Input :

in[] = {2, 1, 3}

post[] = {2, 3, 1}

Output : Root of below tree

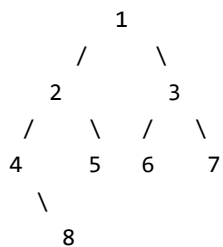


Input :

in[] = {4, 8, 2, 5, 1, 6, 3, 7}

post[] = {8, 4, 5, 2, 6, 7, 3, 1}

Output : Root of below tree



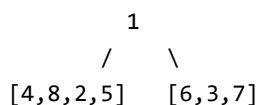
**Recommended: Please solve it on “PRACTICE” first, before moving on to the solution.**

We have already discussed construction of tree from given Inorder and Preorder traversals. The idea is similar.

Let us see the process of constructing tree from  $in[] = \{4, 8, 2, 5, 1, 6, 3, 7\}$  and  $post[] = \{8, 4, 5, 2, 6, 7, 3, 1\}$

**1)** We first find the last node in  $post[]$ . The last node is “1”, we know this value is root as root always appear in the end of postorder traversal.

**2)** We search “1” in  $in[]$  to find left and right subtrees of root. Everything on left of “1” in  $in[]$  is in left subtree and everything on right is in right subtree.



**3)** We recur the above process for following two.

....**b)** Recur for  $in[] = \{6, 3, 7\}$  and  $post[] = \{6, 7, 3\}$

.....Make the created tree as right child of root.

....**a)** Recur for  $in[] = \{4, 8, 2, 5\}$  and  $post[] = \{8, 4, 5, 2\}$ .

.....Make the created tree as left child of root.

Below is C++ implementation of above idea. One important observation is, we recursively call for right subtree before left subtree as we decrease index of postorder index whenever we create a new node.

## C++

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

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

// Utility function to create a new node
```

```

Node* newNode(int data);

/* Prototypes for utility functions */
int search(int arr[], int strt, int end, int value);

/* Recursive function to construct binary of size n
   from Inorder traversal in[] and Preorder traversal
   post[]. Initial values of inStrt and inEnd should
   be 0 and n -1. The function doesn't do any error
   checking for cases where inorder and postorder
   do not form a tree */
Node* buildUtil(int in[], int post[], int inStrt,
               int inEnd, int *pIndex)
{
    // Base case
    if (inStrt > inEnd)
        return NULL;

    /* Pick current node from Preorder traversal using
       postIndex and decrement postIndex */
    Node *node = newNode(post[*pIndex]);
    (*pIndex)--;

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

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

    /* Using index in Inorder traversal, construct left and
       right subtress */
    node->right = buildUtil(in, post, iIndex+1, inEnd, pIndex);
    node->left = buildUtil(in, post, inStrt, iIndex-1, pIndex);

    return node;
}

// This function mainly initializes index of root
// and calls buildUtil()
Node *buildTree(int in[], int post[], int n)
{
    int pIndex = n-1;
    return buildUtil(in, post, 0, n - 1, &pIndex);
}

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

/* Helper function that allocates a new node */
Node* newNode(int data)
{

```

```

Node* node = (Node*)malloc(sizeof(Node));
node->data = data;
node->left = node->right = NULL;
return (node);
}

/* This function is here just to test */
void preOrder(Node* node)
{
    if (node == NULL) return;
    printf("%d ", node->data);
    preOrder(node->left);
    preOrder(node->right);
}

// Driver code
int main()
{
    int in[] = {4, 8, 2, 5, 1, 6, 3, 7};
    int post[] = {8, 4, 5, 2, 6, 7, 3, 1};
    int n = sizeof(in)/sizeof(in[0]);

    Node *root = buildTree(in, post, n);

    cout << "Preorder of the constructed tree : \n";
    preOrder(root);

    return 0;
}

```

Run on IDE

## Java

```

// Java program to construct a tree using inorder
// and postorder traversals

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

    public Node(int data)
    {
        this.data = data;
        left = right = null;
    }
}

// Class Index created to implement pass by reference of Index
class Index
{
    int index;
}

class BinaryTree
{
    /* Recursive function to construct binary of size n

```

```

from Inorder traversal in[] and Preorder traversal
post[]. Initial values of inStrt and inEnd should
be 0 and n -1. The function doesn't do any error
checking for cases where inorder and postorder
do not form a tree */
Node buildUtil(int in[], int post[], int inStrt,
               int inEnd, Index pIndex)
{
    // Base case
    if (inStrt > inEnd)
        return null;

    /* Pick current node from Preorder traversal using
       postIndex and decrement postIndex */
    Node node = new Node(post[pIndex.index]);
    (pIndex.index)--;

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

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

    /* Using index in Inorder traversal, construct left and
       right subtress */
    node.right = buildUtil(in, post, iIndex + 1, inEnd, pIndex);
    node.left = buildUtil(in, post, inStrt, iIndex - 1, pIndex);

    return node;
}

// This function mainly initializes index of root
// and calls buildUtil()
Node buildTree(int in[], int post[], int n)
{
    Index pIndex = new Index();
    pIndex.index = n - 1;
    return buildUtil(in, post, 0, n - 1, pIndex);
}

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

/* This function is here just to test */
void preOrder(Node node)
{
    if (node == null)
        return;
    System.out.print(node.data + " ");
    preOrder(node.left);
    preOrder(node.right);
}

```

```
}

public static void main(String[] args)
{
    BinaryTreeNode tree = new BinaryTreeNode();
    int in[] = new int[]{4, 8, 2, 5, 1, 6, 3, 7};
    int post[] = new int[]{8, 4, 5, 2, 6, 7, 3, 1};
    int n = in.length;
    Node root = tree.buildTree(in, post, n);
    System.out.println("Preorder of the constructed tree : ");
    tree.preOrder(root);
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

[Run on IDE](#)

Output :

```
Preorder of the constructed tree :
1 2 4 8 5 3 6 7
```

Time Complexity :  $O(n^2)$

**Asked in:** [Adobe](#), [Amazon](#)

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

## GATE CS Corner    Company Wise Coding Practice

Trees

### Recommended Posts:

[Construct Tree from given Inorder and Preorder traversals](#)

[Check if two trees are Mirror](#)

[Maximum width of a binary tree](#)

[Construct Full Binary Tree from given preorder and postorder traversals](#)

[Given a binary tree, print all root-to-leaf paths](#)

([Login](#) to Rate and Mark)

**3.3**

Average Difficulty : **3.3/5.0**  
Based on **52** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use [ide.geeksforgeeks.org](http://ide.geeksforgeeks.org), generate link and share the link here.

Load Comments

Share this post!

@geeksforgeeks, [Some rights reserved](#)

[Privacy Policy](#)

[Contact Us!](#)

[About Us!](#)

[Advertise with us!](#)

