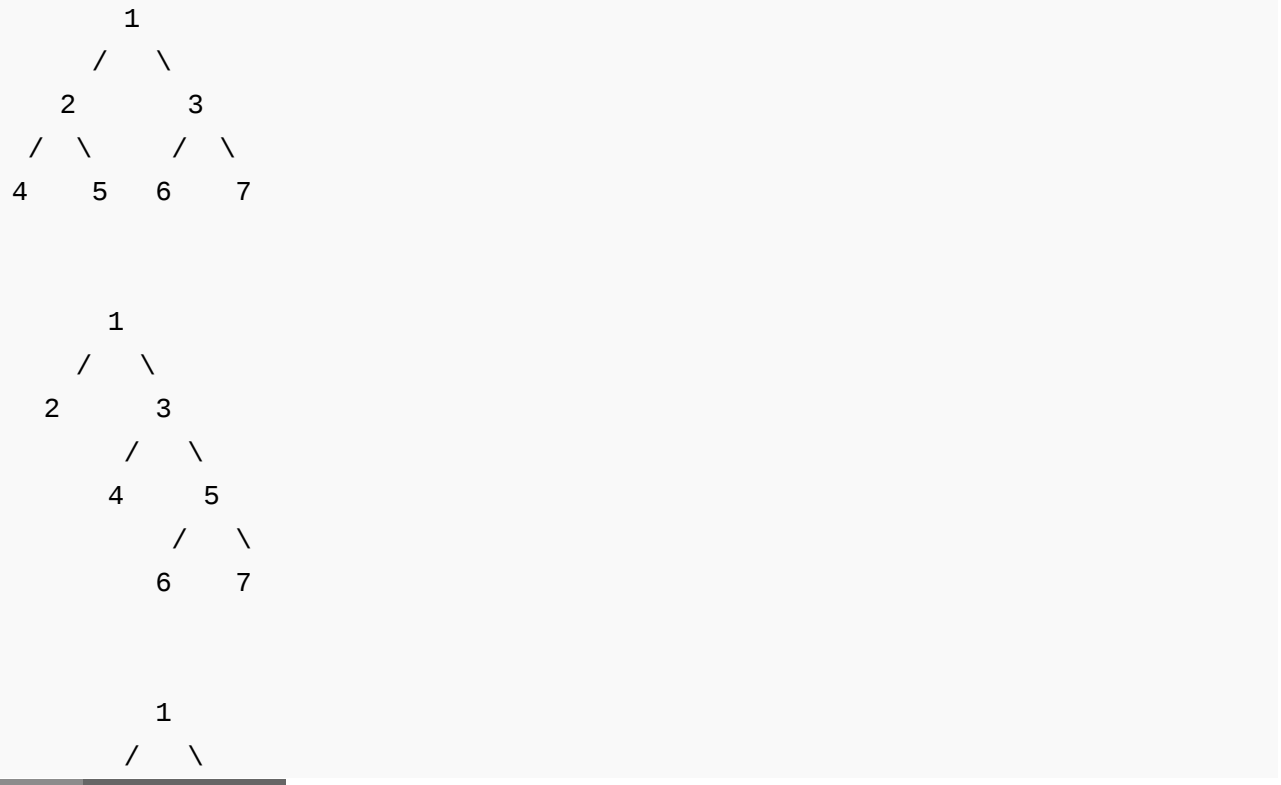
Construct Full Binary Tree from given preorder and postorder traversals

Given two arrays that represent preorder and postorder traversals of a full binary tree, construct the binary tree.

A **Full Binary Tree** is a binary tree where every node has either 0 or 2 children

Following are examples of Full Trees.



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

```
      2        3
    /   \      /   \
   4     5    6     7
 /   \
8     9
```

It is not possible to construct a general Binary Tree from preorder and postorder traversals (See this). But if know that the Binary Tree is Full, we can construct the tree without ambiguity. Let us understand this with the help of following example.

Let us consider the two given arrays as pre[] = {1, 2, 4, 8, 9, 5, 3, 6, 7} and post[] = {8, 9, 4, 5, 2, 6, 7, 3, 1};
In pre[], the leftmost element is root of tree. Since the tree is full and array size is more than 1. The value next to 1 in pre[], must be left child of root. So we know 1 is root and 2 is left child.
How to find the all nodes in left subtree? We know 2 is root of all nodes in left subtree. All nodes before 2 in post[] must be in left subtree. Now we know 1 is root, elements {8, 9, 4, 5, 2} are in left subtree, and the elements {6, 7, 3} are in right subtree.

```
                1
              /   \
             /     \
    {8, 9, 4, 5, 2}     {6, 7, 3}
```

We recursively follow the above approach and get the following tree.

```
          1
        /   \
      2       3
    /   \     /   \
   4     5   6     7
  / \
 8   9
```

```
/* program for construction of full binary tree */
```


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

```c
#include <stdio.h>
#include <stdlib.h>

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

// A utility function to create a node
struct node* newNode (int data)
{
    struct node* temp = (struct node *) malloc( sizeof(struct node) );

    temp->data = data;
    temp->left = temp->right = NULL;

    return temp;
}

// A recursive function to construct Full from pre[] and post[].
// preIndex is used to keep track of index in pre[].
// l is low index and h is high index for the current subarray in post
struct node* constructTreeUtil (int pre[], int post[], int* preIndex,
                                int l, int h, int size)
{
    // Base case
    if (*preIndex >= size || l > h)
        return NULL;

    // The first node in preorder traversal is root. So take the node
    // preIndex from preorder and make it root, and increment preIndex
    struct node* root = newNode ( pre[*preIndex] );
    ++*preIndex;

    // If the current subarry has only one element, no need to recur
    if (l == h)
        return root;

    // Search the next element of pre[] in post[]
    int i;
    for (i = l; i <= h; ++i)
        if (pre[*preIndex] == post[i])
            break;
```

```
    // Use the index of element found in postorder to divide postorder
    // two parts. Left subtree and right subtree
    if (i <= h)
    {
        root->left = constructTreeUtil (pre, post, preIndex, l, i, siz
        root->right = constructTreeUtil (pre, post, preIndex, i + 1, h
    }

    return root;
}

// The main function to construct Full Binary Tree from given preorder
// postorder traversals. This function mainly uses constructTreeUtil()
struct node *constructTree (int pre[], int post[], int size)
{
    int preIndex = 0;
    return constructTreeUtil (pre, post, &preIndex, 0, size - 1, size);
}

// A utility function to print inorder traversal of a Binary Tree
void printInorder (struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}

// Driver program to test above functions
int main ()
{
    int pre[] = {1, 2, 4, 8, 9, 5, 3, 6, 7};
    int post[] = {8, 9, 4, 5, 2, 6, 7, 3, 1};
    int size = sizeof( pre ) / sizeof( pre[0] );

    struct node *root = constructTree(pre, post, size);

    printf("Inorder traversal of the constructed tree: \n");
    printInorder(root);

    return 0;
}
```

Output:

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

```
Inorder traversal of the constructed tree:
8 4 9 2 5 1 6 3 7
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

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.

34 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

prashant jha • 14 hours ago

since u have to make cmplete binary tree so u must have to open two recursiv
will obviously be the next index of preorder and right child index can be found b
the node in postorder and find its index in the preorder

```
#include<iostream>
using namespace std;
struct tnode
{
tnode* lchild;
int data;
tnode* rchild;
tnode(int d)
{
lchild=NULL;
data=d;
rchild=NULL;
}
};
```

see more

Reply • Share ›

Tapan Avasthi • a month ago

struct node * buildCompletePrePostTree(int *Pe, int sPe, int ePe, int *Po, int s

if(sPe>ePe || sPo>ePo || Pe==NULL || Po==NULL)

return NULL;

struct node *root=newNode(Pe[sPe]);

if(sPe==ePe || sPo==ePo) //size is 1

return root;//subtree doesn't have a child node

root->left=buildCompletePrePostTree(Pe,sPe+1,searchIndex(Pe,sPe,ePe,Po
1,Po,sPo,searchIndex(Po,sPo,ePo,Pe[sPe+1]));

root->right=buildCompletePrePostTree(Pe, searchIndex(Pe,sPe,ePe,Po[ePo-
searchIndex(Po,sPo,ePo,Pe[sPe+1])+1,ePo-1);

return root;

}//end of buildCompletePrePostTree method

see more

• Reply • Share ›

Sriharsha g.r.v • 6 months ago

we hav used array "pre" and searched the corresponding value in "pos"
and then solved the problem.the other is way is
possible and its simple to analyse with above logic,
i.e use "pos" and searched the corresponding value in "pre"

#include <stdio.h>
#include <stdlib.h>

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

```
struct node
{
int data;
struct node *left;
struct node *right;
};

struct node* newNode (int data)
```

see more

Reply • Share ›

Guest • 6 months ago

here we hav used array "pre" and searched the corresponding value in "pos" and then solved the problem.can any one help if the other is way is possible.

i actually made this change
initialising preorder with n-1 and

```
if (i <= h && i>=0)
{
root->right = constructTreeUtil (pre, post, i , h, size);
root->left = constructTreeUtil (pre, post, l, i-1, size);

}
```

explanation with code is highly appreciated.thanq

Reply • Share ›

Guest • 6 months ago

here we hav used array pre and searched the corresponding value in pos and help if the other is way is possible.
explanation with code is highly appreciated.thanq

explanation with code is highly appreciated.thanq

Reply • Share ›

prakash • 8 months ago

root->right = constructTreeUtil (pre, post, preIndex, i + 1, h, size);
in this statement upper bound arg for post array must be (h-1) not h. hecause order array which is already created

Reply • Share ›

Nilesh Agrawal • 9 months ago

Great Example

Reply • Share ›

Sarthak Mall 'shanky' • 10 months ago

okk..got it .. I was seeing it as perfect binary tree by mistake... :)

1 Reply • Share ›

GeeksforGeeks • 10 months ago

The definition in fact seems to be matching with Wikipedia. The wiki page says binary tree or 2-tree or strictly binary tree) is a tree in which every node other t perhaps more clearly, every node in a binary tree has exactly (strictly) 0 or 2 c ambiguously defined as a perfect tree (see next). Physicists define a binary tre

Reply • Share ›

Sarthak Mall 'shanky' • 10 months ago

First of all your def of full binary tree is incomplete and last 2 examples are not http://en.wikipedia.org/wiki/B....

Reply • Share ›

Emie Al-Ansi • 10 months ago

thank you so0o0 much^^

Reply • Share ›

jayant • 10 months ago

```
int pree[9] = {1, 2, 4, 8, 9, 5, 3, 6, 7};
  int post[9] = {8, 9, 4, 5, 2, 6, 7, 3, 1};
  int curr=0,pos=0,len=9;


  node* fullbt()
  {
      int i;
      node *root=(node*)malloc(sizeof(node));
      root->data=pree[pos];
      root->left=NULL;
      root->right=NULL;

      for(i=0;i<len;i++)
      if(post[i]==pree[pos])break;

      pos++;
```

see more

• Reply • Share ›

sap • 11 months ago

```
node constructfullbinary(int pre[],int post[],int i,int j,int p,int q
      int t;
      if(i>j||p>q)
      return NULL;
      node temp=getnode(pre[i]);
      if(i==j)
      return temp;
```

```
        for(t=p;;t++)
        {
                if(pre[i+1]==post[t])
                break;
        }

        temp->left=constructfullbinary(pre,post,i+1,i+1+t-p,p,t);
        temp->right=constructfullbinary(pre,post,i+1+t-p+1,j,t+1,q-1)


return temp;
}
```

Reply • Share ›

4m7u1 • 11 months ago

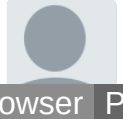
if pre[] = {1, 2, 4, 8, 9, 5, 3, 6, 7} and post[] = {8, 9, 4, 5, 2, 6, 7, 3, 1};
when we do the firt recurison,
root => 1
root->left={8,9,4,5,2}
and using the below function
root->right = constructTreeUtil (pre, post, preIndex, i + 1, h, size);
root->right should be 6,7,3,1 right? as h= size-1 which points to post[8]=0.... c

Reply • Share ›

abhishek08aug • 11 months ago

Intelligent :D

Reply • Share ›



Deepa Kumari • a year ago

nice.....

Viky • a year ago

I think we can get rid of if(l<h) condition as we are already comparing it in the b

Sreenivas Doosa Viky • a year ago

Hey Viky,

Please have a closer look. In the second condition, it is not 'l' it is 'i'

ravik • a year ago

Please check this and correct me if i am wrong.

```
 /* Paste your code here (You may delete these lines if not writing c
struct node *construct(int pre[], int post[], int start, int end, int
{
        int i;
        struct node *temp;
        if(start>end)
        {
                index--;
                return NULL;
        }
        temp = newNode(post[end]);
        for( i = end - 1; i >= start; i--)
                if(pre[index] == post[i])
                        break;
        temp->left      = construct(pre, post, start, i, ++index);
        temp->right     = construct(pre, post, i+1, end-1, ++index);
        return temp;
```

```
}
```

Reply • Share ›

Priyank • a year ago

For populating the right child of a node, the endIndex for post[] should be h -1 i

```
root->right = constructTreeUtil (pre, post, preIndex, i + 1, h -1, s
```

The last element in post[] will always be the root itself, hence we need to exclu
child.

Reply • Share ›

Nitin • a year ago

I think if you see here:
root->left = constructTreeUtil (pre, post, preIndex, l, i, size);
root->right = constructTreeUtil (pre, post, preIndex, i + 1, h, size);
For first time, preIndex will have the value '0' which will be updated to '1'.
Here, you are passing index '1' for both left and right child.
So, in their respective recursion, root will be '1' and both of children will have '2

//struct node* root = newNode (pre[*preIndex]);

I think, we need to do something so that for left child, index '1'(value = 2) will be
'6'(value = 3) will be passed.

Please, correct me if I'm wrong.
I don't know much about c++ but Java.
So, I could have missed something here.

Reply • Share ›

Abhishek ➚ Nitin • a year ago

By the time, preIndex would be passed to the second call, preindex wo
Remember, the first call is a recursion call. It will make other construct
preIndex, covering all the nodes under the left subtree.

root->left = constructTreeUtil (pre, post, preIndex, l, i, size);
root->right = constructTreeUtil (pre, post, preIndex, i + 1, h, size);

  • Reply • Share ›

Palash • a year ago

Time complexity seems to be O(n^2), worst case.

  • Reply • Share ›

neha • 2 years ago

i should start from 0;
for (i = 0; i <= h; ++i)

  • Reply • Share ›

codinggeek16 ➚ neha • 11 months ago

You are right otherwise it will fail for second example of full binary tree

  • Reply • Share ›

Lucy ➚ neha • 2 years ago

@neha..See root is fixed which is 1 here..we need to search for 2..Ist v
index 0 is occupied..and is the root of the tree

```
/* Paste your code here (You may delete these lines if not writ
```

  • Reply • Share ›

codinggeek16 ➚ Lucy • 11 months ago

try running it for second example of full binary tree examples

try running it for second example of full binary tree examples.

Reply • Share ›

neha • 2 years ago

i should start from 0

for (i = l; i <= h; ++i)

Reply • Share ›

swiyu • 2 years ago

Why need complete binary tree? I think full binary tree is enough

```
/* Paste your code here (You may delete these lines if not writing co
```

Reply • Share ›

GeeksforGeeks ➦ swiyu • 2 years ago

@swiju: Thanks for pointing this out. The full tree is good enough. We

Reply • Share ›

Jayanta • 2 years ago

One small correction:

It will be "int post[]" in constructTree() as 2nd argument, it can not be int post.

struct node *constructTree (int pre[], int post[], int size)

Reply • Share ›

GeeksforGeeks ➦ Jayanta • 2 years ago

Thanks for pointing this out. We have corrected the program. Keep it u

Reply • Share ›

sreeram • 2 years ago

i think in the preorder and post order you missed out 8

• Reply • Share ›

GeeksforGeeks ➚ sreeram • 2 years ago

Thanks for pointing this out. We have added 8.

• Reply • Share ›

Subscribe Add Disqus to your site