Construct a special tree from given preorder traversal

Given an array 'pre[]' that represents Preorder traversal of a spacial binary tree where every node has either 0 or 2 children. One more array 'preLN[]' is given which has only two possible values 'L' and 'N'. The value 'L' in 'preLN[]' indicates that the corresponding node in Binary Tree is a leaf node and value 'N' indicates that the corresponding node is non-leaf node. Write a function to construct the tree from the given two arrays.

Source: Amazon Interview Question

Example:

```
Input:  pre[] = {10, 30, 20, 5, 15},  preLN[] = {'N', 'N', 'L', 'L', 'L'}
Output: Root of following tree
          10
         /  \
        30   15
       /  \
      20   5
```

The first element in pre[] will always be root. So we can easily figure out root. If left subtree is empty, the right subtree must also be empty and preLN[] entry for root must be 'L'. We can simply create a node and return it. If left and right subtrees are not empty, then recursively call for left and right subtrees and link the returned nodes to root.

```
/* A program to construct Binary Tree from preorder traversal */
#include<stdio.h>

/* A binary tree node structure */
struct node
```

Google™ Custom Search

GeeksforGeeks
52,731 people like GeeksforGeeks.

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

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

/* Utility function to create a new Binary Tree node */
struct node* newNode (int data)
{
    struct node *temp = new struct node;
    temp->data = data;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

/* A recursive function to create a Binary Tree from given pre[]
   preLN[] arrays. The function returns root of tree. index_ptr is use
   to update index values in recursive calls. index must be initially
   passed as 0 */
struct node *constructTreeUtil(int pre[], char preLN[], int *index_ptr
{
    int index = *index_ptr; // store the current value of index in pre

    // Base Case: All nodes are constructed
    if (index == n)
        return NULL;

    // Allocate memory for this node and increment index for
    // subsequent recursive calls
    struct node *temp = newNode ( pre[index] );
    (*index_ptr)++;

    // If this is an internal node, construct left and right subtrees
    if (preLN[index] == 'N')
    {
      temp->left  = constructTreeUtil(pre, preLN, index_ptr, n);
      temp->right = constructTreeUtil(pre, preLN, index_ptr, n);
    }

    return temp;
}

// A wrapper over constructTreeUtil()
struct node *constructTree(int pre[], char preLN[], int n)
{
    // Initialize index as 0. Value of index is used in recursion to m
```

Geometric Algorithms

Dynamic DevOps Suite
Fast, flexible and hands-free DevOps from build through Deploy
OM Software
Click to Learn More >

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

```
    // the current index in pre[] and preLN[] arrays.
    int index = 0;

    return constructTreeUtil (pre, preLN, &index, n);
}


/* This function is used only for testing */
void printInorder (struct node* node)
{
    if (node == NULL)
        return;

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

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

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

/* Driver function to test above functions */
int main()
{
    struct node *root = NULL;

    /* Constructing tree given in the above figure
          10
         /  \
        30   15
       /  \
      20   5 */
    int pre[] = {10, 30, 20, 5, 15};
    char preLN[] = {'N', 'N', 'L', 'L', 'L'};
    int n = sizeof(pre)/sizeof(pre[0]);

    // construct the above tree
    root = constructTree (pre, preLN, n);

    // Test the constructed tree
    printf("Following is Inorder Traversal of the Constructed Binary T
    printInorder (root);

    return 0;
    return 0;
}
```

Output:

```
Following is Inorder Traversal of the Constructed Binary Tree:
20 30 5 10 15
```

Time Complexity: O(n)

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

Better Than Hadoop.
HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.
LexisNexis
Learn More

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)

695 Subscribe

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...
Backtracking | Set 7 (Sudoku) · 37 minutes ago

RVM Can someone please elaborate this Qs from above...
Flipkart Interview | Set 6 · 57 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...
Software Engineering Lab, Samsung Interview | Set 2 · 57 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 · 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

- 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

2 Tweet 0 0

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

20 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

prashant · a day ago
below standard question
Reply · Share ›

prakash · 4 months ago
"If left subtree is empty, the right subtree must also be empty and preLN[] entr
mentioned in the question. Am I missing something
Reply · Share ›

rahul · 5 months ago
Do we really need to index_ptr as pointer to maintain the index? Just passing i
Reply · Share ›

Amit Bgl · 9 months ago
wow code :D
Reply · Share ›

eric wu · 9 months ago

No need to use n, the recursion will exit by itself when it reaches all the leaves

Reply · Share ›

eric wu · 9 months ago

No need to use n, the recursion will exit by itself when it reaches all the leaves

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

Reply · Share ›

Harkirat Singh · 11 months ago

Did any of you guys, trying doing it without recursion. I did try, the code turns p
straight logic.

You got a pretty straightforward solution. Liked It!

Reply · Share ›

abhishek08aug · a year ago

Intelligent :D

Reply · Share ›

xiaoc10 · a year ago

```
        if (index == n)
            return NULL;
```

Why the above two lines are necessary?

Reply · Share ›

eric wu xiaoc10 · 9 months ago

No need to use n, the recursion will exit after it reaches all the leaves.

^ | v • Reply • Share ›

L • 2 years ago

```
It shud be
   if (preLN[index] == 'N')
   {
     temp->left  = constructTreeUtil(pre, preLN, index_ptr, n);
     temp->right = constructTreeUtil(pre, preLN, index_ptr, n);
   } else {
     temp->left = NULL;
     temp->right = NULL;
   }
```

^ | v • Reply • Share ›

Sreenivas Doosa → L • 2 years ago

@L:
You don`t need to add the else condition to set left and right child to NU
NULL when you create a New Node..

^ | v • Reply • Share ›

Gopika • 2 years ago

I dont under stand where is 'n' coming from. Can you please explain.

^ | v • Reply • Share ›

wakeup123 → Gopika • 10 months ago

n is the size of the array pre[], as well as preLN[]. it is being passed to calling the function in the main. As you can see below......

int n = sizeof(pre)/sizeof(pre[0]);

// construct the above tree
root = constructTree (pre, preLN, n);

Reply • Share ›

Ankit Gupta → Gopika • 2 years ago

If you are talking about the 'n' in the order O(n). It is from the running tir
constructTreeUtil(). @var index_ptr takes values in the range [0, n). He

Reply • Share ›

Priyank • 2 years ago

Why is this true: "If left subtree is empty, the right subtree must also be empty

Reply • Share ›

ritesh → Priyank • 9 months ago

Thats because every node has 2 or 0 children as per question.So it mu
as per its preorder style of traversal,if the left subtree is empty then its
subtree.There has to have a leftsubtree at first.

Reply • Share ›

kartik → Priyank • 2 years ago

As per the problem statement, every node has either 0 or 2 children.

Reply • Share ›

Gopika → kartik • 2 years ago

I am not clear about where is 'n' coming from.
Can you please explain.

Thanks.

```
/* Paste your code here (You may delete these lines if
```

Reply • Share ›

kartik ➜ Gopika • 2 years ago

n is size of input arrays and size of tree.

^ | ⌄ • Reply • Share ›

Subscribe   Add Disqus to your site