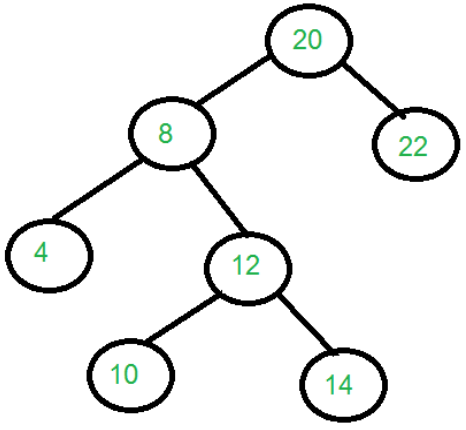
Construct a tree from Inorder and Level order traversals

Given inorder and level-order traversals of a Binary Tree, construct the Binary Tree. Following is an example to illustrate the problem.

```
Input: Two arrays that represent Inorder
       and level order traversals of a
       Binary Tree
in[]    = {4, 8, 10, 12, 14, 20, 22};
level[] = {20, 8, 22, 4, 12, 10, 14};

Output: Construct the tree represented
        by the two arrays.
        For the above two arrays, the
        constructed tree is shown in
        the diagram on right side
```



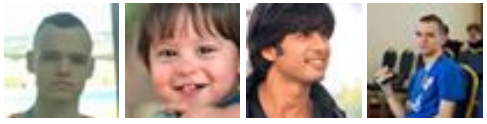
We strongly recommend to minimize the browser and try this yourself first.
The following post can be considered as a prerequisite for this.

Construct Tree from given Inorder and Preorder traversals

Let us consider the above example.

in[] = {4, 8, 10, 12, 14, 20, 22};
level[] = {20, 8, 22, 4, 12, 10, 14};

In a Levelorder sequence, the first element is the root of the tree. So we know '20' is root for given



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

sequences. By searching ’20′ in Inorder sequence, we can find out all elements on left side of ‘20’ are in left subtree and elements on right are in right subtree. So we know below structure now.

```
            20
          /    \
         /      \
 {4,8,10,12,14}  {22}
```

Let us call {4,8,10,12,14} as left subarray in Inorder traversal and {22} as right subarray in Inorder traversal.
In level order traversal, keys of left and right subtrees are not consecutive. So we extract all nodes from level order traversal which are in left subarray of Inorder traversal. To construct the left subtree of root, we recur for the extracted elements from level order traversal and left subarray of inorder traversal. In the above example, we recur for following two arrays.

```
// Recur for following arrays to construct the left subtree
In[]    = {4, 8, 10, 12, 14}
level[] = {8, 4, 12, 10, 14}
```

Similarly, we recur for following two arrays and construct the right subtree.

```
// Recur for following arrays to construct the right subtree
In[]    = {22}
level[] = {22}
```

Following is C++ implementation of the above approach.

```
/* program to construct tree using inorder and levelorder traversals *
#include <iostream>
using namespace std;

/* A binary tree node */
struct Node
{
    int key;
    struct Node* left, *right;
};

/* Function to find index of value in arr[start...end] */
int search(int arr[], int strt, int end, int value)
{
```


Popular Posts

```
    for (int i = strt; i <= end; i++)
        if (arr[i] == value)
            return i;
    return -1;
}

// n is size of level[], m is size of in[] and m < n. This
// function extracts keys from level[] which are present in
// in[].  The order of extracted keys must be maintained
int *extrackKeys(int in[], int level[], int m, int n)
{
    int *newlevel = new int[m], j = 0;
    for (int i = 0; i < n; i++)
        if (search(in, 0, m-1, level[i]) != -1)
            newlevel[j] = level[i], j++;
    return newlevel;
}

/* function that allocates a new node with the given key  */
Node* newNode(int key)
{
    Node *node = new Node;
    node->key = key;
    node->left = node->right = NULL;
    return (node);
}

/* Recursive function to construct binary tree of size n from
   Inorder traversal in[] and Level Order traversal level[].
   inSrt and inEnd are start and end indexes of array in[]
   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 levelorder do not form a tree */
Node* buildTree(int in[], int level[], int inStrt, int inEnd, int n)
{

    // If start index is more than the end index
    if (inStrt > inEnd)
        return NULL;

    /* The first node in level order traversal is root */
    Node *root = newNode(level[0]);

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

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

    // Extract left subtree keys from level order traversal
    int *llevel  = extrackKeys(in, level, inIndex, n);

    // Extract right subtree keys from level order traversal
    int *rlevel  = extrackKeys(in + inIndex + 1, level, n-inIndex-1, n

    /* construct left and right subtress */
    root->left = buildTree(in, llevel, inStrt, inIndex-1, n);
    root->right = buildTree(in, rlevel, inIndex+1, inEnd, n);

    // Free memory to avoid memory leak
    delete [] llevel;
    delete [] rlevel;

    return root;
}

/* Uti;ity function to print inorder traversal of binary tree */
void printInorder(Node* node)
{
    if (node == NULL)
       return;
    printInorder(node->left);
    cout << node->key << " ";
    printInorder(node->right);
}

/* Driver program to test above functions */
int main()
{
    int in[]    = {4, 8, 10, 12, 14, 20, 22};
    int level[] = {20, 8, 22, 4, 12, 10, 14};
    int n = sizeof(in)/sizeof(in[0]);
    Node *root = buildTree(in, level, 0, n - 1, n);

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

    return 0;
}
```

Output:

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 25 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 45 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 45 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 · 2 hours ago

```
Inorder traversal of the constructed tree is
4 8 10 12 14 20 22
```

An upper bound on time complexity of above method is $O(n^3)$. In the main recursive function, extractNodes() is called which takes $O(n^2)$ time.

The code can be optimized in many ways and there may be better solutions. Looking for improvements and other optimized approaches to solve this problem.

This article is contributed by **Abhay Rathi**. 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)
- 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
- Print all nodes that are at distance k from a leaf node

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

18 Comments GeeksforGeeks

Sort by Newest

prashant • 3 days ago

for each node if there exist a node after it in level ordr traversal but before that is present in its left subtree
http://ideone.com/B2cmtJ

```
#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 ›

RandomGuy • 8 days ago

If we just pass the current root node to the extractKeys method and take only t
less/more than the current root node, then also we can get the desired result.
basis of boolean values, find the smaller/greater values for left/right level order

Reply • Share ›

alien • 25 days ago

nice solution

Reply • Share ›

disqus_0z6aYV2hDC • 25 days ago

For lines:
root->left = buildTree(in, llevel, inStrt, inIndex-1, n);
root->right = buildTree(in, rlevel, inIndex+1, inEnd, n);

The last argument, 'n' is the size of the level order array passed. But llevel anc
get array out of bounds error in the calls?

Reply • Share ›

Gaurav • a month ago

Here is N^2 algo that I developed.
Space Complexity N

```
public class BuildATree {

int in[] = { 4, 8, 10, 12, 14, 20, 22 };

int level[] = { 20, 8, 22, 4, 12, 10, 14 };

private HashMap<integer, integer=""> map = new HashMap<integer, integer="
```

```
private Node root;

class Node {

int data;

Node left;

Node right;
```

see more

Reply • Share ›

Gaurav Gaurav • a month ago

here is the code

http://ideone.com/NCpOWN

Reply • Share ›

Gaurav Gulzar • a month ago

http://ideone.com/CLC9VI

Reply • Share ›

Isha • a month ago

Some part of the code is not displayed properly in the comment box and I am

Reply • Share ›

GeeksforGeeks Mod Isha • a month ago

Isha, we have removed the previous comment. Could you please post link here?

Reply • Share ›

Isha ➚ GeeksforGeeks • a month ago

Here is the link:

http://ideone.com/WxZurW

Reply • Share ›

Isha • a month ago

We can do it in O(n^2) time complexity and O(n) space complexity.

Algorithm: We create three queues, one will store the tree nodes and other wil ending index of the range(as derived from inorder array) for that node. For ex:

int in[] = {4, 8, 10, 12, 14, 20, 22};

int lev[] = {20, 8, 22, 4, 12, 10, 14};

For root, 20 starting index will be 0 and ending index will be 6. For node 8 the s index will be 4, likewise. So basically range of a node N implies the index range the nodes in the left and right subtree of that node N.

First we create the root(lev[0]) and then push it in the queue for tree node, and queues. Then we move to the next element in the level order array and check node that is at the rear of the queue. IF it does then we create the tree node cc this node the left/right child depending on its positioning before/after the root's was the right child of the root(node at the rear of the queue) then we popup the completed the creation of its child. We push the child node in the queue along

see more

Reply • Share ›

meh • a month ago

I have an O(n^2) solution that could be made O(n) when using a hashmap to fi level_order in the inorder collection.

The idea is basically to keep track of the items in the current level and the next time keeping the indexes in the inorder collection where the sub-trees of the cu

The solution is iterative similar to the approach when trying to print a tree level

Time complexity: O(n^2) // May be improved to O(n) with hashmap

Space complexity: O(lgn) // May be worsened to O(n) with hashmap

Code: https://ideone.com/UoaEe1

Reply • Share ›

gg • a month ago

No need for extractLeaves function

http://ideone.com/CLC9Vl

Reply • Share ›

L_Earner • a month ago

//Program to create a binary tree from inorder and level order traversal

#include<stdio.h>

#include<stdlib.h>

int in[] = {4, 8, 10, 12, 14, 20, 22};

int level[] = {20, 8, 22, 4, 12, 10, 14};

int n;

//Declare a tree node

struct tree

{

int data;

struct tree *left;

see more

Reply • Share ›

ATUL • a month ago

use map in extractKeys function to reduce its complexity to O(n) instead of O(

Reply • Share ›

Ravi • a month ago

well done abhay! solution looks good time complexity wise, you can save som

Reply • Share ›

abhay Ravi • a month ago

could you provide more details, how can space be saved?

Reply • Share ›

pramendra rathi abhay • 25 days ago

yoou can save space by removing the llevel and rlevel array. ar
levelorder array in function as you are passing for inorder array

Reply • Share ›

Subscribe Add Disqus to your site