GeeksforGeeks

A computer science portal for geeks

Placements Practice GATE CS IDE Q&A GeeksQuiz

Construct a Binary Search Tree from given postorder

Given postorder traversal of a binary search tree, construct the BST.

For example, if the given traversal is {1, 7, 5, 50, 40, 10}, then following tree should be constructed and root of the tree should be returned.

```
      10
    /   \
   5     40
  /  \      \
 1    7      50
```

Method 1 (O(n^2) time complexity)

The last element of postorder traversal is always root. We first construct the root. Then we find the index of last element which is smaller than root. Let the index be ‘i’. The values between 0 and ‘i’ are part of left subtree, and the values between ‘i+1′ and ‘n-2′ are part of right subtree. Divide given post[] at index “i” and recur for left and right sub-trees.

For example in {1, 7, 5, 40, 50, 10}, 10 is the last element, so we make it root. Now we look for the last element smaller than 10, we find 5. So we know the structure of BST is as following.

```
              10
            /    \
           /      \
   {1, 7, 5}       {50, 40}
```

We recursively follow above steps for subarrays {1, 7, 5} and {40, 50}, and get the complete tree.

Method 2 (O(n) time complexity)

The trick is to set a range {min .. max} for every node. Initialize the range as {INT_MIN .. INT_MAX}. The last node will definitely be in range, so create root node. To construct the left subtree, set the range as {INT_MIN …root->data}. If a values is in the range {INT_MIN .. root->data}, the values is part part of left subtree. To construct the right subtree, set the range as {root->data .. INT_MAX}.

Following code is used to generate the exact Binary Search Tree of a given post order traversal.

```
/* A O(n) program for construction of BST from
   postorder traversal */
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node *left, *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 BST from post[].
// postIndex is used to keep track of index in post[].
struct node* constructTreeUtil(int post[], int* postIndex,
                         int key, int min, int max, int size)
{
    // Base case
    if (*postIndex < 0)
        return NULL;

    struct node* root = NULL;

    // If current element of post[] is in range, then
    // only it is part of current subtree
    if (key > min && key < max)
    {
        // Allocate memory for root of this subtree and decrement
        // *postIndex
        root = newNode(key);
        *postIndex = *postIndex - 1;

        if (*postIndex > 0)
        {

          // All nodes which are in range {key..max} will go in right
          // subtree, and first such node will be root of right subtree.
          root->right = constructTreeUtil(post, postIndex, post[*postIndex],
                                          key, max, size );

          // Contruct the subtree under root
          // All nodes which are in range {min .. key} will go in left
          // subtree, and first such node will be root of left subtree.
          root->left = constructTreeUtil(post, postIndex, post[*postIndex],
                                         min, key, size );
        }
    }
    return root;
}

// The main function to construct BST from given postorder
// traversal. This function mainly uses constructTreeUtil()
struct node *constructTree (int post[], int size)
{
    int postIndex = size-1;
    return constructTreeUtil(post, &postIndex, post[postIndex],
                             INT_MIN, INT_MAX, 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 post[] = {1, 7, 5, 50, 40, 10};
    int size = sizeof(post) / sizeof(post[0]);

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

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

    return 0;
}
```

Run on IDE

Output:

```
Inorder traversal of the constructed tree:
1 5 7 10 40 50
```

Note that the output to the program will always be a sorted sequence as we are printing the inorder traversal of a Binary Search Tree.

Reference: http://qa.geeksforgeeks.org/6613/construct-from-postorder-traversal-can-one-help-doing-this

This article is contributed by **Aditya Goel**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

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

8 Comments Category: Trees

Related Posts:

- Check sum of Covered and Uncovered nodes of Binary Tree
- Lowest Common Ancestor in a Binary Tree | Set 2 (Using Parent Pointer)
- BFS vs DFS for Binary Tree
- Maximum difference between node and its ancestor in Binary Tree
- Inorder Non-threaded Binary Tree Traversal without Recursion or Stack
- Check if leaf traversal of two Binary Trees is same?
- Closest leaf to a given node in Binary Tree
- Locking and Unlocking of Resources arranged in the form of n-ary Tree

(Login to Rate and Mark)

3.5 Average Difficulty : **3.5/5.0**
Based on **4** vote(s)

□ Add to TODO List

□ Mark as DONE

Like Share 18 people like this.

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

8 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

MPK • 16 days ago

Can't we just scan the array from the right and just do an insert() into a Binary Search Tree?

```
for(int i = arr.length -1; i >=0; i--)
{
BSTree.insert(arr[i]);
}
```

Not sure if I'm missing anything.. can someone pls comment, if wrong, with an example? Thanks..

⌃ | ⌄ • Reply • Share ›

Ajinkya Kale • 18 days ago

Hello, This is my solution

```
public Node construct(int[] postorder,int start, int end){

if(start>end){
return null;
}

Node root = new Node(postorder[end]);
int i = start;

for(;i<end;i++){ if(postorder[i]="">postorder[end]){
break;
}
}

root.left = construct(postorder,start,i-1);
root.right = construct(postorder,i,end-1);
return root;
}
```

⌃ | ⌄ • Reply • Share ›

Dim131 • 18 days ago

I think that in the constructTreeUtil function there is no need for the parameter size.

⌃ | ⌄ • Reply • Share ›

LOGAN • 19 days ago

super

http://www.universeforyour.com...

http://www.universeforyour.com...

http://www.universeforyour.com

http://www.universeforyour.com...

• Reply • Share ›

yesh_02 • 20 days ago

https://ideone.com/LZE3hx - exponential
https://ideone.com/QzneVd - Linear

1 • Reply • Share ›

piyush bansal • 22 days ago

I have a java solution and it works fine for the given problem statement:
NOTE: Assumption all values in the TREE are unique

```
/*

* In the Given problem statement we are given a post-order traversal and,

* from this we need to construct a Binary Tree

* */

public static void main(String[] args) {

int post[] = {1, 7, 5, 50, 40, 10}; //The order or traversal while creating this array was LEFT-RIGHT-ROOT

BinaryTree bt = new BinaryTree();

bt.root = constructBinaryTreeFromPostOrder(post, 0, post.length - 1);

bt.printBinaryTreeInOrder(bt.root);
```

see more

• Reply • Share ›

TeamG → piyush bansal • 20 days ago

Thanks Piyush.Would you Please drop a code.geeksforgeeks.org generated URL with similar variable and function names.

• Reply • Share ›

piyush bansal → TeamG • 20 days ago

http://code.geeksforgeeks.org/...

Thanks for the update :)

• Reply • Share ›