Construct BST from given preorder traversal | Set 1

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

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

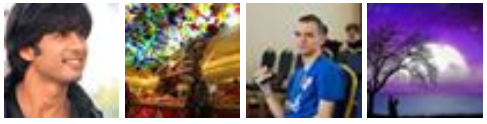
```
     10
   /   \
  5     40
 /  \      \
1    7      50
```

Method 1 (O(n^2) time complexity)
The first element of preorder traversal is always root. We first construct the root. Then we find the index of first element which is greater than root. Let the index be 'i'. The values between root and 'i' will be part of left subtree, and the values between 'i+1′ and 'n-1′ will be part of right subtree. Divide given pre[] at index "i" and recur for left and right sub-trees.
For example in {10, 5, 1, 7, 40, 50}, 10 is the first element, so we make it root. Now we look for the first element greater than 10, we find 40. So we know the structure of BST is as following.

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

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

```
/* A O(n^2) program for construction of BST from preorder traversal */
```

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

```
#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[]. preIndex is used
// to keep track of index in pre[].
struct node* constructTreeUtil (int pre[], int* preIndex,
                                int low, int high, int size)
{
    // Base case
    if (*preIndex >= size || low > high)
        return NULL;

    // The first node in preorder traversal is root. So take the node
    // preIndex from pre[] and make it root, and increment preIndex
    struct node* root = newNode ( pre[*preIndex] );
    *preIndex = *preIndex + 1;

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

    // Search for the first element greater than root
    int i;
    for ( i = low; i <= high; ++i )
        if ( pre[ i ] > root->data )
            break;
```


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

```
    // Use the index of element found in postorder to divide postorder
    // two parts. Left subtree and right subtree
    root->left = constructTreeUtil ( pre, preIndex, *preIndex, i - 1,
    root->right = constructTreeUtil ( pre, preIndex, i, high, size );

    return root;
}

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

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

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

    return 0;
}
```

Output:

```
1 5 7 10 40 50
```

Time Complexity: O(n^2)

Method 2 (O(n) time complexity)

The idea used here is inspired from method 3 of this post. The trick is to set a range {min .. max} for every node. Initialize the range as {INT_MIN .. INT_MAX}. The first 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..max .. INT_MAX}.

```
/* A O(n) program for construction of BST from preorder 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;
    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 BST from pre[]. preIndex is used
// to keep track of index in pre[].
struct node* constructTreeUtil( int pre[], int* preIndex, int key,
                                int min, int max, int size )
{
    // Base case
    if( *preIndex >= size )
        return NULL;

    struct node* root = NULL;

    // If current element of pre[] is in range, then
```

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

```
    // only it is part of current subtree
    if( key > min && key < max )
    {
        // Allocate memory for root of this subtree and increment *pre
        root = newNode ( key );
        *preIndex = *preIndex + 1;

        if (*preIndex < size)
        {
            // Contruct the subtree under root
            // All nodes which are in range {min .. key} will go in le
            // subtree, and first such node will be root of left subtr
            root->left = constructTreeUtil( pre, preIndex, pre[*preInd
                                        min, key, size );

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

    return root;
}

// The main function to construct BST from given preorder traversal.
// This function mainly uses constructTreeUtil()
struct node *constructTree (int pre[], int size)
{
    int preIndex = 0;
    return constructTreeUtil ( pre, &preIndex, pre[0], INT_MIN, INT_MA
}

// 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[] = {10, 5, 1, 7, 40, 50};
```

```
    int size = sizeof( pre ) / sizeof( pre[0] );

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

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

    return 0;
}
```

Output:

```
1 5 7 10 40 50
```

Time Complexity: O(n)

We will soon publish a O(n) iterative solution as a separate post.

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.