Binary Tree to Binary Search Tree Conversion

Given a Binary Tree, convert it to a Binary Search Tree. The conversion must be done in such a way that keeps the original structure of Binary Tree.

Examples.

```
Example 1
Input:
          10
         /  \
        2    7
       / \
      8   4
Output:
          8
         /  \
        4    10
       / \
      2   7


Example 2
Input:
          10
         /  \
        30   15
       /      \
```

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

```
      20       5
Output:
          15
         /  \
       10    20
       /      \
      5        30
```

Solution

Following is a 3 step solution for converting Binary tree to Binary Search Tree.
1) Create a temp array arr[] that stores inorder traversal of the tree. This step takes O(n) time.
2) Sort the temp array arr[]. Time complexity of this step depends upon the sorting algorithm. In the following implementation, Quick Sort is used which takes (n^2) time. This can be done in O(nLogn) time using Heap Sort or Merge Sort.
3) Again do inorder traversal of tree and copy array elements to tree nodes one by one. This step takes O(n) time.

Following is C implementation of the above approach. The main function to convert is highlighted in the following code.

```
/* A program to convert Binary Tree to Binary Search Tree */
#include<stdio.h>
#include<stdlib.h>

/* A binary tree node structure */
struct node
{
    int data;
    struct node *left;
    struct node *right;
};

/* A helper function that stores inorder traversal of a tree rooted
  with node */
void storeInorder (struct node* node, int inorder[], int *index_ptr)
{
    // Base Case
    if (node == NULL)
        return;

    /* first store the left subtree */
```

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

```
    storeInorder (node->left, inorder, index_ptr);

    /* Copy the root's data */
    inorder[*index_ptr] = node->data;
    (*index_ptr)++;  // increase index for next entry

    /* finally store the right subtree */
    storeInorder (node->right, inorder, index_ptr);
}

/* A helper function to count nodes in a Binary Tree */
int countNodes (struct node* root)
{
    if (root == NULL)
     return 0;
    return countNodes (root->left) +
           countNodes (root->right) + 1;
}

// Following function is needed for library function qsort()
int compare (const void * a, const void * b)
{
    return ( *(int*)a - *(int*)b );
}

/* A helper function that copies contents of arr[] to Binary Tree.
   This functon basically does Inorder traversal of Binary Tree and
   one by one copy arr[] elements to Binary Tree nodes */
void arrayToBST (int *arr, struct node* root, int *index_ptr)
{
    // Base Case
    if (root == NULL)
      return;

    /* first update the left subtree */
    arrayToBST (arr, root->left, index_ptr);

    /* Now update root's data and increment index */
    root->data = arr[*index_ptr];
    (*index_ptr)++;

    /* finally update the right subtree */
    arrayToBST (arr, root->right, index_ptr);
}

// This function converts a given Binary Tree to BST
void binaryTreeToBST (struct node *root)
```

```
{
    // base case: tree is empty
    if(root == NULL)
       return;

    /* Count the number of nodes in Binary Tree so that
       we know the size of temporary array to be created */
    int n = countNodes (root);

    // Create a temp array arr[] and store inorder traversal of tree i
    int *arr = new int[n];
    int i = 0;
    storeInorder (root, arr, &i);

    // Sort the array using library function for quick sort
    qsort (arr, n, sizeof(arr[0]), compare);

    // Copy array elements back to Binary Tree
    i = 0;
    arrayToBST (arr, root, &i);

    // delete dynamically allocated memory to avoid meory leak
    delete [] arr;
}
```

```
/* 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;
}

/* Utility function to print inorder traversal of Binary Tree */
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);
```

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

```
    /* 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    */
    root = newNode(10);
    root->left = newNode(30);
    root->right = newNode(15);
    root->left->left = newNode(20);
    root->right->right = newNode(5);

    // convert Binary Tree to BST
    binaryTreeToBST (root);

    printf("Following is Inorder Traversal of the converted BST: \n");
    printInorder (root);

    return 0;
}
```

Output:

```
Following is Inorder Traversal of the converted BST:
5 10 15 20 30
```

We will be covering another method for this problem which converts the tree using O(height of tree) extra space.

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

3 Tweet 0 1

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

13 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion

Join the discussion…

Sudarshan Kj • 2 months ago

what is the use of compare fuction there?
anyone pls tell me

• Reply • Share ›

Harry ➦ Sudarshan Kj • 24 days ago

It used by the c function "qsort", refer to http://www.cplusplus.com/refe

• Reply • Share ›

Vivek • 6 months ago

O(n^2) with space O(1)

```
/* A program to convert Binary Tree to Binary Search Tree */

#include<stdio.h>

#include<stdlib.h>



/* A binary tree node structure */

struct node

{

    int data;
```

see more

• Reply • Share ›

Vinodhini • 7 months ago

could you guys post the O(height of tree) solution?

• Reply • Share ›

Amit Bgl • 9 months ago

wow code :D

• Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

• Reply • Share ›

prakash_ntk • a year ago

```
//Here is a solution for converting a Binary tree to BST without //us.



Convert_BST(node root, int cur_max)
{
        If(NULL==root)
                return ;
        Convert_BST(root->right, cur_max);
        node * max_node=find_next _max(root,cur_max);
        int temp=root->data;
        root->data=max_node->data;
        max_node->data=temp;
        cur_max= max_node->data;
        Convert_BST(root->left, cur_max);
}
```

```
node find_next_max(node root,int pre_max)
```

see more

1 ^ | ˅ • Reply • Share ›

algobard • 2 years ago

Can you guys please post the approach which uses O(ht.) space?

^ | ˅ • Reply • Share ›

Arun • 2 years ago

```
 traverse the tree in pre-order , make the root node correct (mean ro
do it n(no. of node) times. the tree will be BST .
Time complexity is nlogn.
```

^ | ˅ • Reply • Share ›

Dipanjan ➦ Arun • 2 years ago

```
 Your solution won't retain the original Binary tree structure
```

^ | ˅ • Reply • Share ›

Pramod • 2 years ago

```
import java.util.*;
/**
 * Given a Binary Tree, convert it to a Binary Search Tree. The conve
 * be done in such a way that keeps the original structure of Binary
 *
 * @author ppatil
 *
 * Example 1
Input:
```

```
          10
         /  \
        2    7
       / \
      8   4
Output:
          8
         /  \
```

see more

• Reply • Share ›

Venki • 2 years ago

I guess we can use modified head sort method. Use an explicit pointer to poin
from the top of heap. Algorithmically,

1. Min heapify the binary tree (you need parent pointer).
2. Set inorder successor pointer to left most element.
3. Copy root to successor node.
4. Move the successor node to next node in inorder traversal.
5. Heapify the binary tree (excluding subtrees of inorder successor, little trick r
6. Repeat 3 to 5 till all nodes are placed.

Binary tree is nothing but random shuffle or data, the any algorithm must atlea

• Reply • Share ›

Keshava • 2 years ago

Would it not be better to make the BST balanced while we are at it?

selecting arr[n/2] as root recursively would do that

• Reply • Share ›