Find median of BST in O(n) time and O(1) space

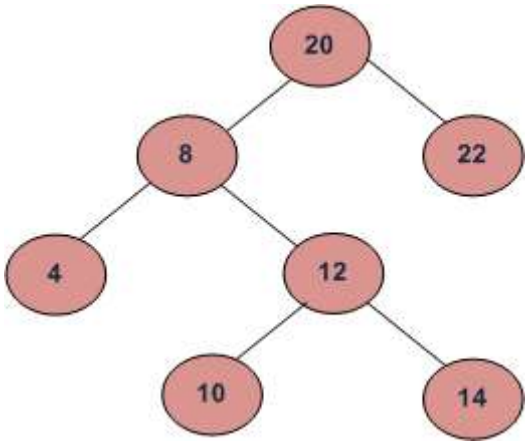
Given a Binary Search Tree, find median of it.

4.3

If no. of nodes are even: then median = ((n/2th node + (n+1)/2th node) /2

If no. of nodes are odd : then median = (n+1)th node.

For example, median of below BST is 12.



More Examples:

```
 Given BST(with odd no. of nodes) is :
                    6
                  /    \
                 3       8
               /   \    /  \
              1     4  7    9

Inorder of Given BST will be : 1, 3, 4, 6, 7, 8, 9
So, here median will 6.

Given BST(with even no. of nodes) is :
                    6
                  /    \
                 3       8
               /   \    /
              1     4  7
```

```
Inorder of Given BST will be : 1, 3, 4, 6, 7, 8
So, here median will  (4+6)/2 = 5.
```

Asked in : Google

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

To find the median, we need to find the Inorder of the BST because its Inorder will be in sorted order and then find the median i.e.

The idea is based on K'th smallest element in BST using O(1) Extra Space

The task is very simple if we are allowed to use extra space but Inorder traversal using recursion and stack both uses Space which is not allowed here. So, the solution is to do Morris Inorder traversal as it doesn't require any extra space.

```
Implementation:
1- Count the no. of nodes in the given BST
   using Morris Inorder Traversal.
2- Then Perform Morris Inorder traversal one
   more time by counting nodes and by checking if
   count is equal to the median point.
   To consider even no. of nodes an extra pointer
   pointing to the previous node is used.
```

```
/* C++ program to find the median of BST in O(n)
   time and O(1) space*/
#include<iostream>
using namespace std;

/* A binary search 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 new BST node
struct Node *newNode(int item)
{
    struct Node *temp =  new Node;
    temp->data = item;
    temp->left = temp->right = NULL;
    return temp;
}

/* A utility function to insert a new node with
   given key in BST */
struct Node* insert(struct Node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

    /* Otherwise, recur down the tree */
    if (key < node->data)
        node->left  = insert(node->left, key);
    else if (key > node->data)
        node->right = insert(node->right, key);

    /* return the (unchanged) node pointer */
```

```
    return node;
}

/* Function to count nodes in a  binary search tree
   using Morris Inorder traversal*/
int counNodes(struct Node *root)
{
    struct Node *current, *pre;

    // Initialise count of nodes as 0
    int count = 0;

    if (root == NULL)
        return count;

    current = root;
    while (current != NULL)
    {
        if (current->left == NULL)
        {
            // Count node if its left is NULL
            count++;

            // Move to its right
            current = current->right;
        }
        else
        {
            /* Find the inorder predecessor of current */
            pre = current->left;

            while (pre->right != NULL &&
                   pre->right != current)
                pre = pre->right;

            /* Make current as right child of its
               inorder predecessor */
            if(pre->right == NULL)
            {
                pre->right = current;
                current = current->left;
            }

            /* Revert the changes made in if part to
               restore the original tree i.e., fix
               the right child of predecssor */
            else
            {
                pre->right = NULL;

                // Increment count if the current
                // node is to be visited
                count++;
                current = current->right;
            } /* End of if condition pre->right == NULL */
        } /* End of if condition current->left == NULL*/
    } /* End of while */

    return count;
}


/* Function to find median in O(n) time and O(1) space
   using Morris Inorder traversal*/
int findMedian(struct Node *root)
{
   if (root == NULL)
        return 0;

    int count = counNodes(root);
    int currCount = 0;
    struct Node *current = root, *pre, *prev;

    while (current != NULL)
    {
```

```
        if (current->left == NULL)
        {
            // count current node
            currCount++;

            // check if current node is the median
            // Odd case
            if (count % 2 != 0 && currCount == (count+1)/2)
                return prev->data;

            // Even case
            else if (count % 2 == 0 && currCount == (count/2)+1)
                return (prev->data + current->data)/2;

            // Update prev for even no. of nodes
            prev = current;

            //Move to the right
            current = current->right;
        }
        else
        {
            /* Find the inorder predecessor of current */
            pre = current->left;
            while (pre->right != NULL && pre->right != current)
                pre = pre->right;

            /* Make current as right child of its inorder predecessor */
            if (pre->right == NULL)
            {
                pre->right = current;
                current = current->left;
            }

            /* Revert the changes made in if part to restore the original
              tree i.e., fix the right child of predecssor */
            else
            {
                pre->right = NULL;

                prev = pre;

                // Count current node
                currCount++;

                // Check if the current node is the median
                if (count % 2 != 0 && currCount == (count+1)/2 )
                    return current->data;

                else if (count%2==0 && currCount == (count/2)+1)
                    return (prev->data+current->data)/2;

                // update prev node for the case of even
                // no. of nodes
                prev = current;
                current = current->right;

            } /* End of if condition pre->right == NULL */
        } /* End of if condition current->left == NULL*/
    } /* End of while */
}

/* Driver program to test above functions*/
int main()
{

    /* Let us create following BST
                  50
               /     \
              30      70
             /  \    /  \
           20   40  60   80 */
    struct Node *root = NULL;
    root = insert(root, 50);
    insert(root, 30);
```

```
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);

    cout << "\nMedian of BST is "
         << findMedian(root);
    return 0;
}
```

Run on IDE

Output:

```
Median of BST is 50
```

Reference:
https://www.careercup.com/question?id=4882624968392704

This article is contributed by **Sahil Chhabra**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or 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.

GATE CS Corner Company Wise Coding Practice

Binary Search Tree | Amazon | Google | Order-Statistics | statistical-algorithms

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

K'th smallest element in BST using O(1) Extra Space
Leaf nodes from Preorder of a Binary Search Tree
Remove all leaf nodes from the binary search tree
Find k-th smallest element in BST (Order Statistics in BST)
Convert BST to Min Heap
Iterative searching in Binary Search Tree
Largest number less than or equal to N in BST (Iterative Approach)
Maximum Unique Element in every subarray of size K
Count greater nodes in AVL tree
Left Leaning Red Black Tree (Insertion)

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

Load Comments

Share this post!