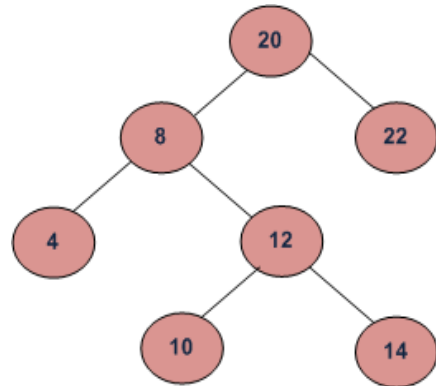


Lowest Common Ancestor in a Binary Search Tree.

Given values of two nodes in a Binary Search Tree, write a c program to find the **Lowest Common Ancestor (LCA)**. You may assume that both the values exist in the tree.

The function prototype should be as follows:

```
struct node *lca(node* root, int n1, int n2)
n1 and n2 are two given values in the tree with given root.
```



For example, consider the BST in diagram, LCA of 10 and 14 is 12 and LCA of 8 and 14 is 8.

Following is definition of LCA from Wikipedia:

Let T be a rooted tree. The lowest common ancestor between two nodes n1 and n2 is defined as the lowest node in T that has both n1 and n2 as descendants (where we allow a node to be a descendant of itself).

The LCA of n1 and n2 in T is the shared ancestor of n1 and n2 that is located farthest from the root. Computation of lowest common ancestors may be useful, for instance, as part of a procedure for determining the distance between pairs of nodes in a tree: the distance from n1 to n2 can be computed as the distance from the root to n1, plus the distance from the root to n2, minus twice the distance from the root to their lowest common ancestor. (Source [Wiki](#))

Solutions:

If we are given a BST where every node has **parent pointer**, then LCA can be easily determined by traversing up using parent pointer and printing the first intersecting node.

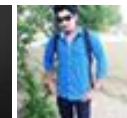
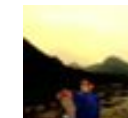
Google™ Custom Search



GeeksforGeeks



52,731 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

We can solve this problem using BST properties. We can **recursively traverse** the BST from root. The main idea of the solution is, while traversing from top to bottom, the first node n we encounter with value between $n1$ and $n2$, i.e., $n1 < n < n2$ or same as one of the $n1$ or $n2$, is LCA of $n1$ and $n2$ (assuming that $n1 < n2$). So just recursively traverse the BST in, if node's value is greater than both $n1$ and $n2$ then our LCA lies in left side of the node, if it's is smaller than both $n1$ and $n2$, then LCA lies on right side. Otherwise root is LCA (assuming that both $n1$ and $n2$ are present in BST)

```
// A recursive C program to find LCA of two nodes n1 and n2.
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node* left, *right;
};

/* Function to find LCA of n1 and n2. The function assumes that both
n1 and n2 are present in BST */
struct node *lca(struct node* root, int n1, int n2)
{
    if (root == NULL) return NULL;

    // If both n1 and n2 are smaller than root, then LCA lies in left
    if (root->data > n1 && root->data > n2)
        return lca(root->left, n1, n2);

    // If both n1 and n2 are greater than root, then LCA lies in right
    if (root->data < n1 && root->data < n2)
        return lca(root->right, n1, n2);

    return root;
}

/* Helper function that allocates a new node with the given data.*/
struct node* newNode(int data)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = node->right = NULL;
    return (node);
}
```



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

```

/* Driver program to test mirror() */
int main()
{
    // Let us construct the BST shown in the above figure
    struct node *root = newNode(20);
    root->left = newNode(8);
    root->right = newNode(22);
    root->left->left = newNode(4);
    root->left->right = newNode(12);
    root->left->right->left = newNode(10);
    root->left->right->right = newNode(14);

    int n1 = 10, n2 = 14;
    struct node *t = lca(root, n1, n2);
    printf("LCA of %d and %d is %d \n", n1, n2, t->data);

    n1 = 14, n2 = 8;
    t = lca(root, n1, n2);
    printf("LCA of %d and %d is %d \n", n1, n2, t->data);

    n1 = 10, n2 = 22;
    t = lca(root, n1, n2);
    printf("LCA of %d and %d is %d \n", n1, n2, t->data);

    getchar();
    return 0;
}

```

Output:

LCA of 10 and 14 is 12

LCA of 14 and 8 is 8

LCA of 10 and 22 is 20

Time complexity of above solution is $O(h)$ where h is height of tree. Also, the above solution requires $O(h)$ extra space in function call stack for recursive function calls. We can avoid extra space using **iterative solution**.

```

/* Function to find LCA of n1 and n2. The function assumes that both
n1 and n2 are present in BST */
struct node *lca(struct node* root, int n1, int n2)
{
    while (root != NULL)
    {
        // If both n1 and n2 are smaller than root, then LCA lies in

```

Shouldn't
you expect
a cloud with:

**ONE-CLICK
DEPLOYMENTS**

Experience the
Managed Cloud
Difference

TRY TODAY ►

 **rackspace**
the open cloud company

```

if (root->data > n1 && root->data > n2)
    root = root->left;

// If both n1 and n2 are greater than root, then LCA lies in r
else if (root->data < n1 && root->data < n2)
    root = root->right;

else break;
}
return root;
}

```

See [this](#) for complete program.

You may like to see [Lowest Common Ancestor in a Binary Tree](#) also.

Exercise

The above functions assume that n1 and n2 both are in BST. If n1 and n2 are not present, then they may return incorrect result. Extend the above solutions to return NULL if n1 or n2 or both not present in BST.

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



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

LexisNexis®

Learn More

Recent Comments

karthik it should have been max_wrap=
max_wrap -...

[Maximum circular subarray sum](#) · 1 minute ago

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

[Backtracking | Set 7 \(Sudoku\)](#) · 45 minutes ago

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

[Flipkart Interview | Set 6](#) · 1 hour ago

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


[Software Engineering Lab, Samsung Interview | Set 2](#) · 1 hour 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

AdChoices 

[▶ Code C++](#)

[▶ Binary Tree](#)

[▶ Java Tree](#)

[▶ XML Tree Viewer](#)[▶ Root Tree](#)[▶ Java to C++](#)[▶ JavaScript Tree](#)[▶ Tree Control](#)[▶ Ancestor Tree](#)

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](#)



24



Tweet

1



4

Writing code in comment? Please use [ideone.com](#) and share the link here.

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team