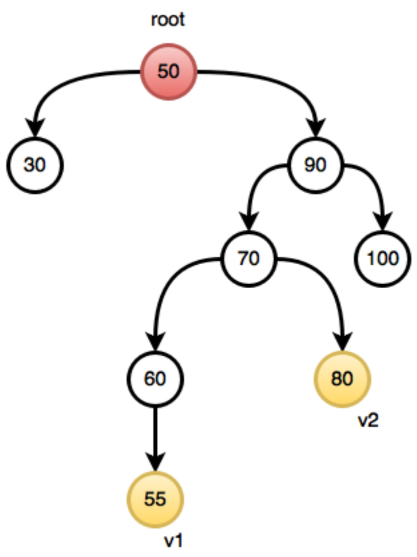
Binary Search Tree : Lowest Common Ancestor ☆

Problem Submissions Leaderboard Editorial

Editorial by Shafaet

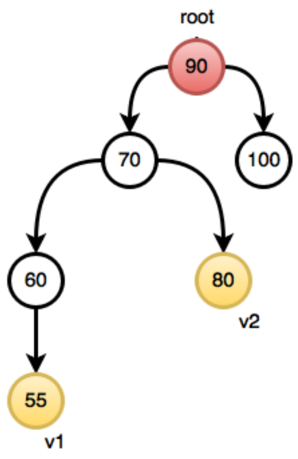
You can take advantage of the properties of a binary search tree to find the LCA. You have to consider 3 cases to solve this problem.

Case 1: Both v_1 and v_2 are on the right of the current root



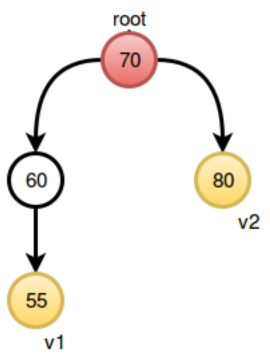
In this case, you can be sure that the LCA is situated in the right subtree. Go the right child and discard the left-side of the tree.

Case 2: Both v_1 and v_2 are on the left of the current root



Now in this case, you can be sure that the LCA is situated in the left subtree. Discard the right subtree and go to the left child.

Case 3: v_1 and v_2 are on two different subtrees or one of them is the current root.



This is the terminal case. The current root is the LCA so you don't need to search anymore!

Time complexity is O(depth of the tree).

Featured Solutions

Java

```
static Node lca(Node root,int v1,int v2)
{
    //Decide if you have to call recursively
    //Samller than both
    if(root.data < v1 && root.data < v2){
        return lca(root.right,v1,v2);
    }
    //Bigger than both
    if(root.data > v1 && root.data > v2){
        return lca(root.left,v1,v2);
    }

    //Else solution already found
    return root;
}
```

JP Tested by John Pierce

Problem Tester's code:

```
# Python 3 iterative solution
def lca(root , v1 , v2):
    # make sure v2 > v1
    if v1 > v2: v1, v2 = v2, v1
    # traverse until terminal
    while True:
        if v1 < root.info and v2 < root.info:
            root = root.left
        elif v1 > root.info and v2 > root.info:
            root = root.right
        else:
            return(root)
```

Feedback

Was this editorial helpful?

Yes No

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature