Home » Technical Interview Questions » Tree Interview Questions » Inorder Successor of a node in Binary Tree

Inorder Successor of a node in Binary Tree

Difficulty Level Hard

Frequently asked in Amazon Expedia Morgan Stanley OYO Rooms Snapchat

Tags Binary Search Tree Tree Views 127

Recent Posts

Longest Common Subsequence LeetCode Solution

Range Sum Query 2D – Immutable LeetCode Solution

Palindrome Number LeetCode Solution

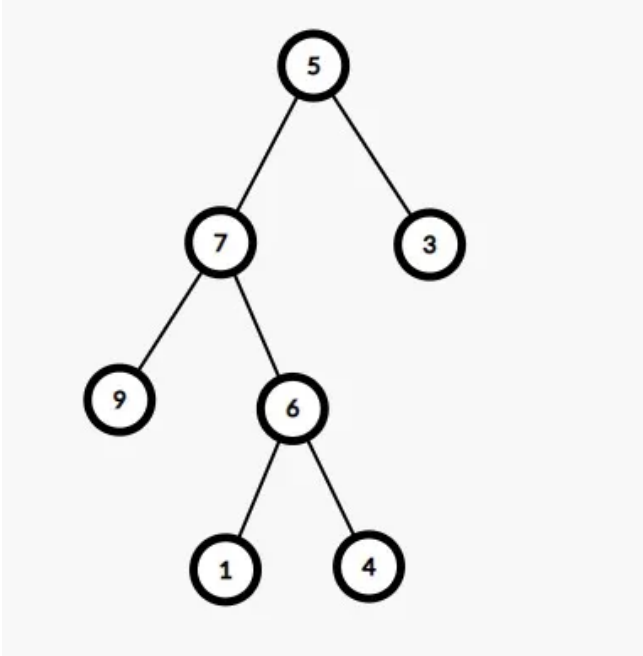
Count Sub Islands LeetCode Solution

Find the Town Judge LeetCode Solution

Table of Contents

The problem asks to find “Inorder Successor of a node in Binary Tree”. An inorder successor of a node is a node in the binary tree that comes after the given node in the inorder traversal of the given binary tree.

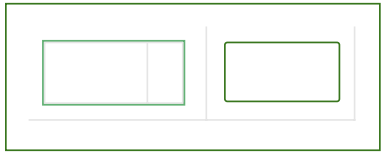
Example



Output

```
Inorder successor of 6 is 4
```

Explanation



The inorder traversal of the tree is 9 7 1 6 4 5 3. Thus the inorder successor of 1 is 6.

Approach

Generally, we are told to find the next node in inorder traversal of a binary search tree. But that is different from that of a binary tree. So one thing which we should note is that the inorder traversal of a general binary tree is not in ascending order. Now let's move ahead. So if we have a node then there are 3 cases which we should look upon.

The 3 cases which we should note are related to its right child or if the current node itself is a rightmost child. So if the node has a right child. Then inorder successor is simply the leftmost child in its right subtree. But if it does not have the right child. Then find the lowest ancestor of the given node such that the given node lies in the left subtree of the ancestor. For doing this, recursively find the node, and when we move back from recursion store the parent from where we have chosen the left direction.

Now the last case is if the node is the rightmost child. If that happens there does not exist any inorder successor for the node.

Code

Code

```
#include<bits/stdc++.h>
using namespace std;

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

node* create(int data){
  node* tmp = new node();
  tmp->data = data;
  tmp->left = tmp->right = NULL;
  return tmp;
}

node* findLeftMostNode(node* root){
  while(root && root->left) root = root->left;
  return root;
}

node* findRightMostNode(node* root){
  while(root && root->right) root = root->right;
  return root;
}

node* findAncestorSuccessor(node* root, node* given)
{
  if(root){
    if(root == given)
      return root;
    node* tmp = findAncestorSuccessor(root->left, given);
    if(tmp){
      if(root->left == tmp){
        cout<<"Inorder Successor of "<<given->data<<" is "<<root->d
        return NULL;
      }
```

```
    if(tmp){
      if(root->left == tmp){
        cout<<"Inorder Successor of "<<given->data<<" is "<<root->d
        return NULL;
      }
      return root;
    }
  }
    return NULL;
}

void findInorderSuccesor(node* root, node* given)
{
    // if right child exists
    if(given->right)
    {
      node* leftmost = findLeftMostNode(given);
      cout<<"Inorder Succesor of "<<given->data<<" is "<<leftmost->
      return;
    }
    // if right child does not exists
    if(given->right == NULL)
    {
        node* rightmost = findRightMostNode(root);
        if(rightmost == given)
            cout<<"Inorder Successor does not exists";
        else
          findAncestorSuccessor(root, given);
    }
}

int main()
{
  node* root = create(5);
  root->left = create(7);
  root->right = create(3);
  root->left->left = create(9);
  root->left->right = create(6);
  root->left->right->left = create(1);
```

Output

```
Inorder Successor of 1 is 6
```

Java code to find Inorder Successor of a node in Binary Tree

Code

```
import java.util.*;

class node{
  int data;
  node left, right;
}

class Main{
  static node create(int data){
    node tmp = new node();
    tmp.data = data;
    tmp.left = tmp.right = null;
    return tmp;
  }

  static node findLeftMostNode(node root){
    while(root != null && root.left != null) root = root.left;
    return root;
  }

  static node findRightMostNode(node root){
    while(root != null && root.right != null) root = root.right;
    return root;
  }

  static node findAncestorSuccessor(node root, node given)
```

```
      return root;
    node tmp = findAncestorSuccessor(root.left, given);
    if(tmp != null){
      if(root.left == tmp){
        System.out.print("Inorder Successor of "+given.data+" is
        return null;
      }
      return root;
    }
    tmp = findAncestorSuccessor(root.right, given);
    if(tmp != null){
      if(root.left == tmp){
        System.out.print("Inorder Successor of "+given.data+" is
        return null;
      }
      return root;
    }
  }
    return null;
}

static void findInorderSuccesor(node root, node given)
{
    // if right child exists
    if(given.right != null)
    {
      node leftmost = findLeftMostNode(given);
      System.out.print("Inorder Successor of "+given.data+" is "+
      return;
    }
    // if right child does not exists
    else
    {
        node rightmost = findRightMostNode(root);
        if(rightmost == given)
            System.out.print("Inorder Successor does not exists")
        else
          findAncestorSuccessor(root, given);
    }
```

```
    {
      node root = create(5);
      root.left = create(7);
      root.right = create(3);
      root.left.left = create(9);
      root.left.right = create(6);
      root.left.right.left = create(1);
      root.left.right.right = create(4);
      findInorderSuccesor(root, root.left.right.left);
    }
  }
```

Output

```
Inorder Successor of 1 is 6
```

Complexity Analysis

Time Complexity

O(N), because in worst cases we may have to traverse over all of the nodes.

Space Complexity

O(H), since we have used recursion. Thus if we consider the space taken by compiler stack. The space complexity is dependent on the height of the tree.

Tree Interview Questions

Amazon, Binary Search Tree, Expedia, Hard, Morgan Stanley, OYO Rooms, Snapchat, Tree

‹ Given an Array of Pairs Find all Symmetric Pairs in it

› Find postorder traversal of BST from preorder traversal