Introduction to Binary Search Tree

02 October 2024 16:53

```
class Solution {
   static boolean isBSTTraversal(int arr[])
   {
      for(int i=0;i<(arr.length-1);i++)
      {
         if(arr[i]>=arr[i+1])
         {
            return false;
         }

      }
      return true;
      // code here
   }
}
```

Search in a Binary Search Tree

02 October 2024 16:54

```
class Solution {
    public TreeNode searchBST(TreeNode root, int val)
    {
        if(root==null)
        {
            return null;
        }

        if(root.val<val)
        {
            return searchBST(root.right,val);
        }
        else if(root.val>val)
        {
            return searchBST(root.left,val);
        }
        return root;

    }
}
```

Find Min/Max in BST

02 October 2024 16:54

```
class Solution {
    // Function to find the minimum element in the given BST.
    int minValue(Node root)
    {
        Node curr=root;

        if(root==null)
        {
            return -1;
        }
        while(curr.left!=null)
        {
            curr=curr.left;
        }
        return curr.data;
    }
}
```

Ceil in a Binary Search Tree

02 October 2024 18:26

```
class Tree {
    // Function to return the ceil of given number in BST.
    int  ans=Integer.MAX_VALUE;  =====> use this method always
    void find(Node root,int key)
    {
        if(root==null)
        {
            return;
        }
        if(root.data==key)
        {
            ans=root.data;
            return;
        }
        if(root.data<key)
        {
            find(root.right,key);
        }
        if(root.data>key)
        {
            ans=root.data;
            find(root.left,key);
        }
        return;
    }

    int findCeil(Node root, int key) {
        if (root == null) return -1;
        //Node t=new Node();

        find(root,key);
        if(ans==Integer.MAX_VALUE)
        {
            return -1;
        }
        return ans;
        // Code here
    }
}
```

Floor in a Binary Search Tree

02 October 2024 18:27

```
class Solution {
   public static int  ans;
   static void find(Node root,int key)
   {
      if(root==null)
      {
         return;
      }
      if(root.data==key)
      {
         ans=root.data;
         return;
      }
      if(root.data<key)
      {
         ans=root.data;
         find(root.right,key);
      }
      if(root.data>key)
      {

         find(root.left,key);
      }


      return;
   }
   public static int floor(Node root, int key)
   {

      if (root == null) return -1;
      ans=-1;

      find(root,key);

      return ans;
   }
}
```

Insert a given Node in Binary Search Tree

02 October 2024 18:27

```
class Solution {
    public TreeNode insertIntoBST(TreeNode root, int val)
    {
        if(root==null)
        {
            TreeNode node=new TreeNode(val);
            return node;
        }
        if(root.val<val)
        {
            root.right=insertIntoBST(root.right,val);
        }
        else if(root.val>val)
        {
            root.left=insertIntoBST(root.left,val);
        }
        return root;

    }
}
```

Delete a Node in Binary Search Tree***

02 October 2024 18:27

```
class Solution {

    public TreeNode delete(TreeNode root, int key) {
        if (root == null) {
            return null;
        }

        // Find the node to be deleted
        if (root.val < key) {
            root.right = delete(root.right, key);
        } else if (root.val > key) {
            root.left = delete(root.left, key);
        } else {
            // Node to be deleted is found
            if (root.left == null && root.right == null) {
                // Case 1: No children, just remove the node
                return null;
            } else if (root.left == null) {
                // Case 2: One child (right child)
                return root.right;
            } else if (root.right == null) {
                // Case 3: One child (left child)
                return root.left;
            } else {
                // Case 4: Two children
                // Find the minimum value in the right subtree (or max value in the left subtree)
                TreeNode minNode = findMin(root.right);
                root.val = minNode.val;  // Replace node's value with the min node's value
                root.right = delete(root.right, minNode.val);  // Delete the min node from the right subtree
            }
        }
        return root;
    }

    // Helper function to find the minimum value node in the tree
    private TreeNode findMin(TreeNode root) {
        while (root.left != null) {
            root = root.left;
        }
        return root;
    }

    public TreeNode deleteNode(TreeNode root, int key) {
        return delete(root, key);
    }
}
```

Find K-th smallest/largest element in BST

02 October 2024 18:27

```
class Solution {
    int ans=Integer.MIN_VALUE;
    int count = 0;
    void find(TreeNode root,int k)
    {
        if(root==null)
        {
            return;
        }
        find(root.left,k);
        count++;
        if(k==count)
        {
            ans=root.val;
            return;
        }


        find(root.right,k);
        //k--;
    }
    public int kthSmallest(TreeNode root, int k)
    {
        find(root,k);
        return ans;



    }
}
```

```
int ans;
    void find(TreeNode root,int[] k)
    {
        if(root==null||k[0]<0)
        {
            return;
        }
        find(root.left,k);
        k[0]--;
        if(k[0]==0)
        {
            ans=root.val;
            return;
        }


        find(root.right,k);
        //k--;
    }
    public int kthSmallest(TreeNode root,
int k)
    {
        int[]karray=new int[1];
        karray[0]=k;

        find(root,karray);
        return ans;
    }
```

Check if a tree is a BST or BT***

02 October 2024 18:27

```
class Solution {
    Long a=Long.MIN_VALUE;
    Long b=Long.MAX_VALUE;
    boolean find(TreeNode root,long a,long b)  //yaha long ki jagah Long karne pe error aa ja rha hai
    {
        if(root==null)
        {
            return true;
        }
        if(root.val<=a || root.val>=b)
        {
            return false;
        }
        if(a>b)//Commenting this if line also working
        {
            return false;
        }
        return find(root.left,a,root.val)&&find(root.right,root.val,b);
    }
    public boolean isValidBST(TreeNode root)
    {
        if(root==null)
        {
            return true;
        }
        return find(root,a,b);

    }
}
```

LCA in Binary Search Tree

02 October 2024 18:27

```
class Solution {
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) 
    {
        if(root==null)
        {
            return null;
        }
        if(root.val<p.val&&root.val<q.val)
        {
            return lowestCommonAncestor(root.right,p,q);
        }
        else if(root.val>p.val&&root.val>q.val)
        {
            return lowestCommonAncestor(root.left,p,q);
        }
        else
        {
             return root;
        }

    }
}
```

Construct a BST from a preorder traversal

02 October 2024 18:28

```
TreeNode build(int []preorder, int l,int r)
{
    if(l>r)
    {
        return null;
    }
    TreeNode root=new TreeNode(preorder[l]);
    int i;
    for(i=l+1;i<=r;i++)
    {
        if(preorder[i]>root.val)
        {
            break;
        }
        //idx=i;
    }
    root.left=build(preorder,l+1,i-1);
    root.right=build(preorder,i,r);
    return root;
}
    public TreeNode bstFromPreorder(int[] preorder)
    {
        int l=0;
        int r=preorder.length-1;
        return build(preorder,l,r);

    }
```

Inorder Successor/Predecessor in BST

02 October 2024   18:28

```
int suc=-1;

   void find(Node root,int x)
   {
      if(root==null)
      {
         return;
      }
      if(root.data>x)
      {
         suc=root.data;
         find(root.left,x);
      }
      else if(root.data<x)
      {
         find(root.right,x);
      }
      else
      {
         find(root.right,x);
      }
      return;
   }
   public int inorderSuccessor(Node root, Node x)
   {

      find(root,x.data);
      return suc;
   }
```

Merge 2 BST's***************

02 October 2024 18:28

```
class Solution {
   // Function to return a list of integers denoting the node
   // values of both the BST in a sorted order.
   class BSTIterator{
      Stack<Node> st;
      BSTIterator(Node root){
         st = new Stack<>();
         insert(root);
      }

      void insert(Node node){
         while(node != null){
            st.add(node);
            node = node.left;
         }
      }

      int get(){
         return st.peek().data;
      }
      boolean hasNext(){
         return st.size() > 0;
      }
      int next(){
         Node node = st.pop();
         insert(node.right);
         return node.data;
      }
   }
   public ArrayList<Integer> merge(Node root1, Node root2) {
      // Write your code here
      BSTIterator it1 = new BSTIterator(root1);
      BSTIterator it2 = new BSTIterator(root2);
      ArrayList<Integer> list = new ArrayList<>();

      while(it1.hasNext() && it2.hasNext()){
         int val1 = it1.get();
         int val2 = it2.get();
         if(val1 < val2){
            list.add(val1);
            it1.next();
         }else{
            list.add(val2);
            it2.next();
         }
      }

      while(it1.hasNext()){
         list.add(it1.get());
         it1.next();
      }
```

```
        while(it2.hasNext()){
            list.add(it2.get());
            it2.next();
        }

        return list;
    }
}
```

Two Sum In BST | Check if there exists a pair with Sum K***************

03 October 2024 19:26

```
class Solution {
   boolean findTarget(Node root, int target) {
      // Write your code here
      HashSet<Integer> set = new HashSet<>();
      return inorder(root, set, target);
   }

   private boolean inorder(Node node, HashSet<Integer> set, int target) {
      if (node == null) return false;
      if (inorder(node.left, set, target)) return true;
      if (set.contains(target - node.data)) return true;
      set.add(node.data);
      return inorder(node.right, set, target);
   }
}
```

```
/ { Driver Code Starts
#include <bits/stdc++.h>
using namespace std;
#define MAX_HEIGHT 100000
#define MAX_SIZE 100000

// Tree Node
struct Node {
   int data;
   Node *left;
   Node *right;

   Node(int val) {
      data = val;
      left = right = NULL;
   }
};


int isPairPresent(Node *root, int k);

// Function to Build Tree
Node* buildTree(string str)
{
  // Corner Case
  if(str.length() == 0 || str[0] == 'N')
     return NULL;

  // Creating vector of strings from input
  // string after spliting by space
  vector<string> ip;
```

```
    istringstream iss(str);
    for(string str; iss >> str; )
        ip.push_back(str);

    // Create the root of the tree
    Node* root = new Node(stoi(ip[0]));

    // Push the root to the queue
    queue<Node*> queue;
    queue.push(root);

    // Starting from the second element
    int i = 1;
    while(!queue.empty() && i < ip.size()) {

        // Get and remove the front of the queue
        Node* currNode = queue.front();
        queue.pop();

        // Get the current node's value from the string
        string currVal = ip[i];

        // If the left child is not null
        if(currVal != "N") {

            // Create the left child for the current node
            currNode->left = new Node(stoi(currVal));

            // Push it to the queue
            queue.push(currNode->left);
        }

        // For the right child
        i++;
        if(i >= ip.size())
            break;
        currVal = ip[i];

        // If the right child is not null
        if(currVal != "N") {

            // Create the right child for the current node
            currNode->right = new Node(stoi(currVal));

            // Push it to the queue
            queue.push(currNode->right);
        }
        i++;
    }

    return root;
}

int main() {

    int t;
    string tc;
```

```
   getline(cin, tc);
   t=stoi(tc);
   while(t--)
   {
       string s;
      getline(cin, s);
      Node* root = buildTree(s);

      getline(cin, s);
      int k = stoi(s);
       //getline(cin, s);

      cout << isPairPresent(root, k) << endl;
      //cout<<"~"<<endl;
   }
   return 0;
}
// } Driver Code Ends


/*Complete the function below
Node is as follows
struct Node {
    int data;
    Node *left;
    Node *right;

    Node(int val) {
        data = val;
        left = right = NULL;
    }
};
*/

// root : the root Node of the given BST
// target : the target sum
void pushElements(Node* root, stack<Node*> &s, bool direction){
    if(direction){
        while(root){
            s.push(root);
            root = root->left;
        }
        return;
    }
    while(root){
        s.push(root);
        root = root->right;
    }
}

int isPairPresent(struct Node *root, int target)
{
    if(!root) return 0;
    stack<Node*> s1, s2;
    int x, y;
    Node* temp;
    pushElements(root,s1, true);
    pushElements(root,s2,false);
    while(!s1.empty() && !s2.empty() && s1.top()->data < s2.top()->data){
        x = s1.top()->data;
        y = s2.top()->data;
        if(x+y == target) return 1;
        if(x+y<target){
            temp = s1.top();
            s1.pop();
            pushElements(temp->right,s1,true);
        }
        else{
            temp = s2.top();
            s2.pop();
            pushElements(temp->left,s2, false);
        }
    }
    return 0;
}
```

Recover BST | Correct BST with two nodes swapped

03 October 2024 19:26

Largest BST in Binary Tree

03 October 2024 19:26