import java.util.\*;

import java.io.\*;

public class BST\_Operations {

class Node{

int data;

Node left, right;

public Node(int data){

this.data = data;

left = right = null;

}

}

Node root;

// Constructors

BST\_Operations(){

root = null;

}

BST\_Operations(int data){

root = new Node(data);

}

void insert(int data){

root = insertVal(root, data);

}

// function to insert an element in BST

Node insertVal(Node root, int data){

if(root == null){

root = new Node(data);

return root;

}

else if(data < root.data){

root.left = insertVal(root.left, data);

}

else{

root.right = insertVal(root.right, data);

}

return root;

}

// function to search an element in a BST

Node search(Node root, int data){

if(root == null || root.data == data){

return root;

}

if(root.data < data){

return search(root.right, data);

}

return search(root.left, data);

}

// function to delete the node in BST

void delete(int data){

root = deleteNode(root, data);

}

Node deleteNode(Node root, int data){

if(root == null){

return root;

}

if(root.data > data){

root.left = deleteNode(root.left, data);

}

else if(root.data < data){

root.right = deleteNode(root.right, data);

}

else{

// Case 1: Node with no child or only one child

if(root.left == null){

return root.right;

}

else if(root.right == null){

return root.left;

}

// Case 2: Node with two children

Node inorderSucc = minValNode(root.right);

root.data = inorderSucc.data;

root.right = deleteNode(root.right, inorderSucc.data);

}

return root;

}

// function to find the minimum value node in BST

Node minValNode(Node node){

Node curr = node;

while(curr.left != null){

curr = curr.left;

}

return curr;

}

// function to display the results

void inOrder(){

inOrderRec(root);

}

void inOrderRec(Node root){

if(root != null){

inOrderRec(root.left);

System.out.print(root.data + " ");

inOrderRec(root.right);

}

}

// Driver method

public static void main(String[] args){

BST\_Operations tree = new BST\_Operations();

tree.insert(100);

tree.insert(50);

tree.insert(150);

tree.insert(70);

tree.insert(20);

tree.insert(200);

tree.inOrder();

System.out.println(" ");

int data = 170;

if(tree.search(tree.root, data) == null){

System.out.println(data + " is not found");

}

else{

System.out.println(data + " is found");

}

tree.delete(50);

tree.delete(200);

tree.delete(100);

tree.inOrder();

System.out.println(" ");

}

}

PPT Batch

1. Sort Colors

class Solution:

def sortColors(self, nums: List[int]) -> None:

"""

Do not return anything, modify nums in-place instead.

"""

p0 = current = 0

p2 = len(nums) - 1

while current <= p2:

if nums[current] == 0:

nums[p0], nums[current] = nums[current], nums[p0]

p0 += 1

current += 1

elif nums[current] == 2:

nums[p2], nums[current] = nums[current], nums[p2]

p2 -= 1

else:

current += 1

2. Rotate Array

class Solution:

def reverse(self, nums: List[int], low:int, high:int):

while(low < high):

nums[low],nums[high] = nums[high],nums[low]

low = low + 1

high = high - 1

def rotate(self, nums: List[int], k: int) -> None:

"""

Do not return anything, modify nums in-place instead.

"""

n = len(nums)

k = k % n

self.reverse(nums,0,n-k-1)

self.reverse(nums,n-k,n-1)

self.reverse(nums,0,n-1)