**Date**: 25/11/24

**Binary Search Tree**

1. **BST creation using Collections**

package BST;

import java.util.TreeSet;

public class BinarySearchTree{

public static void main(String[] args){

TreeSet<Integer> bst=new TreeSet<>();

bst.add(90);

bst.add(20);

bst.add(70);

bst.add(23);

bst.add(7);

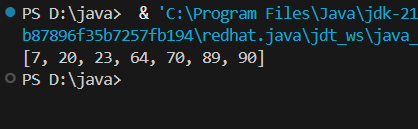
bst.add(89);

bst.add(64);

System.out.print(bst);

}

}



**Time Complexity**: O (log n)

1. **Manual Creation of BST**

package BST;

class ManualBST {

static class Node {

int data;

Node left, right;

Node(int data) {

this.data = data;

}

}

Node root;

void insert(int data) {

root = insertdata(root,data);

}

Node insertdata(Node root, int data) {

if (root == null) return new Node(data);

if (data < root.data) root.left = insertdata(root.left, data);

else if (data > root.data) root.right = insertdata(root.right, data);

return root;

}

void inorder(Node root) {

if (root != null) {

inorder(root.left);

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

inorder(root.right);

}

}

public static void main(String[] args) {

ManualBST tree = new ManualBST();

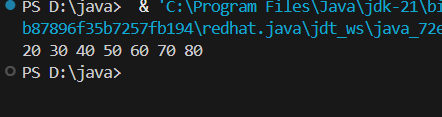
int[] num = {50, 30, 20, 40, 70, 60, 80};

for (int data : num)tree.insert(data);

tree.inorder(tree.root);

}

}



**Time Complexity**: O(n^2)

1. **Validate BST**

package BST;

import java.util.Stack;

class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

left=right=null;

}

}

public class ValidateBST{

public boolean isValid(Node root){

if(root==null){

return true;

}

Stack<Node> stack1=new Stack<>();

Node prev=null;

while(root!=null || !stack1.isEmpty()){

while(root!=null){

stack1.push(root);

root=root.left;

}

root=stack1.pop();

if(prev!=null && root.data>=prev.data){

return false;

}

prev=root;

root=root.right;

}

return true;

}

public static void main(String[] args){

Node root=new Node(10);

root.left=new Node(20);

root.right=new Node(34);

root.left.left=new Node(56);

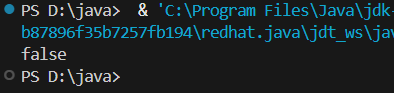
root.right.left=new Node(98);

ValidateBST bst=new ValidateBST();

System.out.print(bst.isValid(root));

}

}



**Time Complexity:** O(n)

1. **Left, Right, Top, Bottom View of BST**

package BST;

public class ViewBST{

class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

}

}

Node root;

void insert(int data){

root=insertval(root,data);

}

Node insertval(Node root,int data){

if(root==null) return new Node(data);

if(data<root.data) root.left=insertval(root.left,data);

if(data>root.data) root.right=insertval(root.right,data);

return root;

}

void printtop(){

if(root!=null){

System.out.print(root.data);

}

else{

System.out.print(-1);

}

}

void printbottom(Node root){

if(root!=null){

if(root.left==null && root.right==null){

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

}

printbottom(root.left);

printbottom(root.right);

}

}

void printleft(Node root){

if(root!=null){

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

printleft(root.left);

}

}

void printright(Node root){

if(root!=null){

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

printright(root.right);

}

}

public static void main(String[] args){

ViewBST bst=new ViewBST();

int[] num={50, 30, 70, 20, 40, 60, 80};

for(int n:num){

bst.insert(n);

}

bst.printtop();

System.out.println();

bst.printbottom(bst.root);

System.out.println();

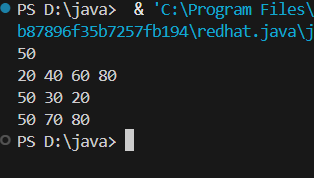
bst.printleft(bst.root);

System.out.println();

bst.printright(bst.root);

}

}



**Time Complexity:** O(n^2)

1. **Segment Tree**

package BST;

public class Segmenttree{

int st[];

Segmenttree(int arr[],int length){

int h=(int)(Math.ceil(Math.log(length)/Math.log(2)));

int maxsize=2\*(int)(Math.pow(2,h)-1);

st=new int[maxsize];

construct(arr,0,length-1,0);

}

int calculatemid(int x,int y){

return x+(y-x)/2;

}

int construct(int arr[],int x,int y,int i){

if(x==y){

st[i]=arr[x];

return arr[x];

}

int mid=calculatemid(x,y);

st[i]=construct(arr, x, mid,i\*2+1)+construct(arr,mid+1,y,i\*2+2);

return st[i];

}

void printst(){

for(int num:st){

if(num!=0){

System.out.print(num+" ");

}

}

}

public static void main(String[] args){

int arr[]={2,5,43,68,98,45};

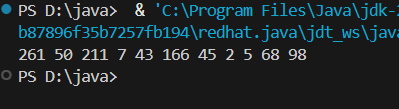
int length=arr.length;

Segmenttree tree=new Segmenttree(arr,length);

tree.printst();

}

}



**Time Complexity:** O(n)