**Binary Search Tree**

package bst;

import java.util.\* ;

// @author Paddi

public class BST {

public static void main(String[] args) {

//lets make a tree

int n=0,i=0,j=0,x=0,lvl=0,ht=3 ;

node root,curr,prev ;

root=new node(0,null,null,null,0) ;

Scanner sc=new Scanner(System.in);

System.out.println("Enter number nodes: ");

n=sc.nextInt() ;

System.out.println("Enter numbers: ");

for(i=0;i<n;i++)

{

x=sc.nextInt() ;

if(i==0)

{

root.val=x ;

}

else

{

curr=root ;

prev=root ;

lvl=0 ;

node leg =new node(x,null,null,null,0) ;

while(true)

{

if(x>curr.val)

{

prev=curr ;

curr=curr.right ;

lvl++ ;

}

if(curr==null)

{

prev.right=leg ;

leg.parent=prev ;

leg.lvl=lvl ;

break ;

}

if(x<curr.val)

{

prev=curr ;

curr=curr.left ;

lvl++ ;

}

if(curr==null)

{

prev.left=leg ;

leg.parent=prev ;

leg.lvl=lvl ;

break ;

}

// lvl++ ;

}

}

}

//DFS

node.dfs(root);

System.out.println(" ");

int h=node.nonleaf(root);

System.out.print("Recursive Implementation:\n");

System.out.print(h+" non-leaf nodes\n ");

int l=0,r=0 ;

curr=root ;

ht=node.height(root) ;

System.out.print("height of BST: "+ht+"\n\n ");

System.out.println("now Iteratively! ") ;

node.itrnonleaf(root,ht) ;

node.itrheight(root);

}

}

class node {

int val ;

node parent ;

node left ;

node right ;

int lvl ;

node(int v,node p,node l,node r,int lv)

{

val=v ;

parent=p ;

left=l;

right=r ;

lvl=lv ;

}

static int height(node temp)

{

int l=0,r=0 ;

if(temp.left!=null)

l=height(temp.left)+1;

if(temp.right!=null)

r=height(temp.right)+1;

return Math.max(l,r) ;

}

static void dfs(node r)

{

if(r==null)

return ;

dfs(r.left) ;

System.out.print(" "+r.val+"("+r.lvl+")");

dfs(r.right);

}

static int nonleaf(node r)

{

int cnt=0 ;

if(r==null)

return cnt ;

cnt+=nonleaf(r.left) ;

cnt+=nonleaf(r.right);

if(r.left!=null || r.right!=null)

return cnt+1 ;

return cnt ;

}

static void itrnonleaf(node r,int ht)

{

int l=0,i ;

int ri=1 ;

int cnt=0 ;

node arr[]=new node[100] ;

int lvl[]=new int[100] ;

for(i=0;i<100;i++)

{

lvl[i]=0 ;

}

arr[l]=r ;

while(l!=ri)

{

r=arr[l] ;

if(r.left!=null)

arr[ri++]=r.left ;

if(r.right!=null)

arr[ri++]=r.right ;

if(r.left!=null || r.right!=null)

{

cnt++ ;

lvl[r.lvl]++ ;

}

l++ ;

}

System.out.println("non leaf nodes: "+cnt) ;

for(i=0;i<=ht;i++)

{

System.out.println("no. of nonleaf nodes at level "+i+": "+lvl[i]);

}

}

static void itrheight(node root)

{

node curr=root ;

int l=0,r=0 ;

while(true)

{

curr=curr.left ;

if(curr!=null)

l++ ;

else

break ;

}

curr=root ;

while(true)

{

curr=curr.right ;

if(curr!=null)

r++ ;

else

break ;

}

if(r>l)

{

System.out.println("Height is: "+r);

}

else

{

System.out.println("Height is: "+l);

}

}

}

**OUTPUT :**

Enter number nodes:

11

Enter numbers:

60

20

80

10

30

70

90

65

75

85

95

10(2) 20(1) 30(2) 60(0) 65(3) 70(2) 75(3) 80(1) 85(3) 90(2) 95(3)

Recursive Implementation:

5 non-leaf nodes

height of BST: 3

now Iteratively!

non leaf nodes: 5

no. of nonleaf nodes at level 0: 1

no. of nonleaf nodes at level 1: 2

no. of nonleaf nodes at level 2: 2

no. of nonleaf nodes at level 3: 0

Height is: 3