20-March-2020

Link for practice:

https://thecodingsimplified.com/binary-tree/

[Complete at least first 20 ques from the given link over the weekend.]

Content:

20-March-2020

1. **Check if two trees are mirror to each other**
2. **Convert a tree into its mirror tree**
3. **Check if two trees have the same structure**
4. **Check if two trees are IsoMorphic**
5. **Get the height of a tree itereatively without using recursion**

H.W

1. **Find the level of a given node.**
2. **Complete at least first 20 ques from the given link over the weekend.**

Code:

```
package mbatch;
import java.util.*;
class Node
{
	int data;
	Node left;
	Node right;
	Node(int data){
		this.data=data;
		left=null;
		right=null;
	}
}
class BinaryTree{
```

```
	Node root;
	BinaryTree(){

		root=null;
	}
	BinaryTree(int data){
		this.root=new Node(data);
	}
	int TreeSum(Node root)//to calculate the sum of all nodes
in a tree
	{
		if(root==null) return 0;
		return
root.data+TreeSum(root.left)+TreeSum(root.right);
	}
	int countNodes(Node root)//to calculate the number of nodes
in a tree
	{
		if(root==null) return 0;
		return 1+countNodes(root.left)+countNodes(root.right);
	}
	int leafNodes(Node root)//to calculate the number of Leaf
Nodes in a tree
	{
		if(root==null) return 0;
		if(root.left==null && root.right==null) return 1;
		return leafNodes(root.left)+leafNodes(root.right);
	}
	int sumleafNodes(Node root)//to calculate the sum of Leaf
Nodes in a tree
	{
		if(root==null) return 0;
		if(root.left==null && root.right==null) return
root.data;
		return
sumleafNodes(root.left)+sumleafNodes(root.right);
	}
	int height(Node root)
	{
		if(root==null) return -1;
		return 1+Math.max(height(root.left),
height(root.right));
	}
	void printAtLevel(Node root,int level)
	{
		if(root==null) return;
		if(level==1)
```

```
        {
                System.out.print(root.data+" ");
                return;
        }
        printAtLevel(root.left,level-1);
        printAtLevel(root.right,level-1);
}
void levrec(Node root)
{
        if(root==null) return;
        int h=height(root);
        for(int i=1;i<=h+1;i++)
        {
                printAtLevel(root, i);
                System.out.println();
        }
}
void levitr(Node root)
{
        if(root==null) return;
        Queue<Node> q=new java.util.LinkedList<>();
        q.add(root);
        while(!q.isEmpty())
        {
                Node temp=q.remove();
                System.out.print(temp.data+" ");
                if(temp.left!=null)
                {
                        q.add(temp.left);
                }
                if(temp.right!=null)
                {
                        q.add(temp.right);
                }
        }
        System.out.println();
}
void levlineitr(Node root)
{
        if(root==null) return;
        Queue<Node> q=new java.util.LinkedList<>();
        q.add(root);
        while(true)
        {
                int size=q.size();
                if(size==0) break;
                //while(size>0)
```

```
            for(int i=0;i<size;i++)
            {
                  Node temp=q.remove();
                  System.out.print(temp.data+" ");
                  if(temp.left!=null)
                  {
                        q.add(temp.left);
                  }
                  if(temp.right!=null)
                  {
                        q.add(temp.right);
                  }
                  //size--;
            }
            System.out.println();
      }
}
boolean isIdentical(Node root1, Node root2)
{
      if(root1==null && root2==null) return true;
      if(root1==null||root2==null) return false;
      return root1.data==root2.data
                  && isIdentical(root1.left,root2.left)
                  && isIdentical(root1.right,root2.right);

}
boolean isMirror(Node root1, Node root2)
{
      if(root1==null && root2==null) return true;
      if(root1==null||root2==null) return false;
      return root1.data==root2.data
                  && isMirror(root1.left,root2.right)
                  && isMirror(root1.right,root2.left);

}
Node toMirror(Node root)
{
      if(root==null) return null;
      Node temp=root.left;
      root.left=root.right;
      root.right=temp;
      toMirror(root.left);
      toMirror(root.right);
      return root;
}
boolean sameStructure(Node root1, Node root2)
{
```

```
            if(root1==null && root2==null) return true;
            if(root1==null||root2==null) return false;
            return sameStructure(root1.left,root2.left)
                        && sameStructure(root1.right,root2.right);

      }
      boolean isIsomorphic(Node root1, Node root2)
      {
            if(root1==null && root2==null) return true;
            if(root1==null||root2==null) return false;
            if(root1.data!=root2.data) return false;
            return (isIsomorphic(root1.left,root2.left) &&
isIsomorphic(root1.right,root2.right))
            || (isIsomorphic(root1.left,root2.right) &&
isIsomorphic(root1.right,root2.left));

//          return root1.data==root2.data
//                      && ((isIsomorphic(root1.left,root2.left) &&
isIsomorphic(root1.right,root2.right))
//                      || (isIsomorphic(root1.left,root2.right) &&
isIsomorphic(root1.right,root2.left)));

      }
      int heightitr(Node root)
      {
            if(root==null) return -1;
            Queue<Node> q=new java.util.LinkedList<>();
            q.add(root);
            int level=0;
            while(true)
            {
                  int size=q.size();
                  if(size==0) break;
                  //while(size>0)
                  for(int i=0;i<size;i++)
                  {
                        Node temp=q.remove();
                        //System.out.print(temp.data+" ");
                        if(temp.left!=null)
                        {
                              q.add(temp.left);
                        }
                        if(temp.right!=null)
                        {
                              q.add(temp.right);
                        }
                        //size--;
```

```
                }
                level++;
                //System.out.println();
            }
            return level-1;
        }
}
public class btree {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        BinaryTree bt=new BinaryTree(2);//BT with root node 2
        bt.root.left=new Node(3);//linking explicitly
        bt.root.right=new Node(5);
        bt.root.left.right=new Node(9);
        bt.root.right.left=new Node(7);//Required Tree Created
        System.out.println("Sum of all Nodes:
"+bt.TreeSum(bt.root));
        System.out.println("Total Nodes:
"+bt.countNodes(bt.root));
        System.out.println("Leaf Nodes:
"+bt.leafNodes(bt.root));
        System.out.println("Height: "+bt.height(bt.root));
        System.out.print("Nodes at level 1: ");
        bt.printAtLevel(bt.root,1);
        System.out.println();
        System.out.print("Nodes at level 2: ");
        bt.printAtLevel(bt.root,2);
        System.out.println();
        System.out.print("Nodes at level 3: ");
        bt.printAtLevel(bt.root,3);
        System.out.println();
        System.out.print("Nodes at level 4: ");
        bt.printAtLevel(bt.root,4);
        System.out.println();
        System.out.println("Sum of Leaf Nodes:
"+bt.sumleafNodes(bt.root));
        bt.levrec(bt.root);
        bt.levitr(bt.root);
        bt.levlineitr(bt.root);

        BinaryTree bt2=new BinaryTree(2);//BT with root node 2
        bt2.root.left=new Node(5);//linking explicitly
        bt2.root.right=new Node(3);
        bt2.root.left.right=new Node(7);
        bt2.root.right.left=new Node(9);
        System.out.println("Identical:
"+bt.isIdentical(bt.root, bt2.root));
```

```
            System.out.println("Mirror: "+bt.isMirror(bt.root,
bt.root));
            System.out.println("Mirror: "+bt.isMirror(bt.root,
bt2.root));
            Node temp=bt.toMirror(bt.root);
            System.out.println(temp.data+" "+temp.left.data+"
"+temp.right.data);
            System.out.println("sameStructure:
"+bt.sameStructure(bt.root, bt2.root));
            System.out.println("isIsomorphic:
"+bt.isIsomorphic(bt.root, bt.root));
            System.out.println("isIsomorphic:
"+bt.isIsomorphic(bt.root, bt2.root));
            BinaryTree bt3=new BinaryTree(2);//BT with root node 2
            bt3.root.left=new Node(3);//linking explicitly
            bt3.root.right=new Node(5);
            bt3.root.left.left=new Node(7);
//          bt3.root.right.left=new Node(7);
            bt3.root.right.right=new Node(9);
            System.out.println("isIsomorphic:
"+bt.isIsomorphic(bt.root, bt3.root));
            System.out.println("Height: "+bt.heightitr(bt.root));
      }
}
```