https://course.acciojob.com/idle?question=249439ec-42db-41ff-af2b-d14a69763904

- EASY
- Max Score: 30 Points

Same Tree

Given two `root` nodes of two separate binary trees `a` and `b`, check whether the trees are identical.

Two binary trees are considered the same if they have identical shapes, and the corresponding nodes have the same value.

Input Format

You are given two separate binary trees `a` and `b` in separate lines.

Output Format

For each test case print "YES" if the trees are identical, else print "NO"

Example 1

Input

```
1 2 3
1 2 3
```

Output

```
YES
```

Explanation

```
1st tree
     1
    / \
   2   3

2nd tree
     1
    / \
   2   3
```

Example 2

Input

```
1 2 3
1 3 2
```

Output

```
NO
```

Explanation

```
1st tree
     1
    / \
   2   3

2nd tree
     1
    / \
   3   2
```

Constraints

1 <= size(a), size(b) <= 100

Value of any node is less than 2^32

Topic Tags

- Trees

My code

```
// in java
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
import java.util.*;



class Main {
    static Node buildTree(String str) {
        if (str.length() == 0 || str.charAt(0) == 'N') {
            return null;
        }
        String ip[] = str.split(" ");
        Node root = new Node(Integer.parseInt(ip[0]));
        Queue<Node> queue = new LinkedList<>();
        queue.add(root);
        int i = 1;
        while (queue.size() > 0 && i < ip.length) {
            Node currNode = queue.peek();
            queue.remove();
            String currVal = ip[i];
            if (!currVal.equals("N")) {
                currNode.left = new Node(Integer.parseInt(currVal));
                queue.add(currNode.left);
```

```
            }
            i++;
            if (i >= ip.length) break;
            currVal = ip[i];
            if (!currVal.equals("N")) {
                currNode.right = new Node(Integer.parseInt(currVal));
                queue.add(currNode.right);
            }
            i++;
        }

        return root;
    }
    void inOrder(Node node) {
        if (node == null) {
            return;
        }
        inOrder(node.left);
        System.out.print(node.data + " ");
        inOrder(node.right);
    }

    public static void main(String[] args) throws IOException {
        BufferedReader br =new BufferedReader(new
InputStreamReader(System.in));
        String s1 = br.readLine();
        Node root1 = buildTree(s1);
        String s2 = br.readLine();
        Node root2 = buildTree(s2);
        Solution g = new Solution();
        g.areTreesIdentical(root1, root2);
    }
```

```
}


class Node {
    int data;
    Node left;
    Node right;
    Node(int data) {
        this.data = data;
        left = null;
        right = null;
    }
}

class Solution {
    static int fun(Node root1, Node root2)
        {
                if(root1==null && root2 ==null)
                  return 1;
                if(root1==null && root2 !=null)
                  return 0;
                if(root1!=null && root2 ==null)
                  return 0;
               int a= fun(root1.left,root2.left);
                int b=fun(root1.right,root2.right);
                return a*b;

        }
    public static void areTreesIdentical(Node root1, Node root2) {
        //Your code here
        int ans=fun(root1,root2);
           if(ans==0)
```

```
                System.out.print("NO");
               else  System.out.print("YES");
    }
}
```