

<https://course.acciojob.com/idle?question=19b6bff8-dded-4cc9-a13d-cda60887b20f>

● EASY

● Max Score: 30 Points

## Check if Tree is Isomorphic

Given two Binary Trees `root1` and `root2`, write a function that returns true if they are Isomorphic or not.

Note: You just need to implement the `isIsomorphic()` function and return true if two trees are isomorphic, else false.

Note:- Two trees are called isomorphic if one can be obtained from another by a series of flips, i.e. by swapping left and right children of several nodes. Any number of nodes at any level can have their children swapped. Two empty trees are isomorphic.

### Input Format

First line contains a string representing the tree with `root1`. Second line contains a string representing the tree with `root2`.

The values in the string are in the order of level order traversal of the tree where, numbers denote node values, and a character "N" denotes NULL child.

### Output Format

Print true if two trees are isomorphic.

### Example 1

Input

```
9 8 N 6 1 N N N N
9 N 8 1 6 N N N N
```

Output

true

Explanation

The first tree can be represented as:-



The second tree can be represented as:-



The two trees are isomorphic.

## Example 2

Input

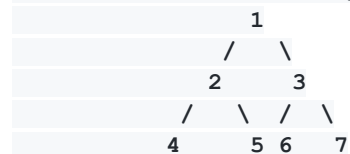
```
1 2 3 4 5 6 7 N N N N N N N N
3 6 7 N N N N
```

Output

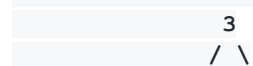
false

Explanation

The first tree can be represented as:-



The second tree can be represented as:-



The two trees are not isomorphic.

## Constraints

The number of nodes in the both the trees are in the range [1, 500]

$-500 \leq \text{Node.data} \leq 500$

### Topic Tags

- Recursion
- Trees

# My code

// in java

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

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

```
}
```

```
class Main {  
    static Node buildTree(String str){  
        // System.out.print(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));
        Scanner sc = new Scanner(System.in);

        String s = sc.nextLine();
        String s1 = sc.nextLine();

        Node root1 = buildTree(s);
        Node root2 = buildTree(s1);
        Solution tree = new Solution();
        boolean ans = tree.isIsomorphic(root1,root2);
        System.out.println(ans);
    }
}

class Solution{

```

```
public static boolean isIsomorphic(Node root1, Node root2) {  
    // Write your code here  
    if(root1==null && root2==null)  
        return true;  
    if(root1.data !=root2.data)  
        return false;  
    if(root1!=null && root2==null)  
        return false;  
    if(root1==null && root2!=null)  
        return false;  
    return  
isIsomorphic(root1.left,root2.right)&&isIsomorphic(root1.right,root2.left)  
;  
    }  
}
```