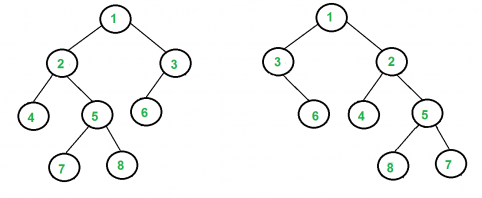
**Check if Tree is Isomorphic: -**

Easy Accuracy: 49.11% Submissions: 78K+ Points: 2

Given two Binary Trees. Check whether they are Isomorphic or not.

**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.  
For example, the following two trees are isomorphic with the following sub-trees flipped: 2 and 3, NULL and 6, 7 and 8.  
[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/ISomorphicTrees-e1368593305854.png)

**Example 1:**

**Input:**

**T1** 1 **T2:** 1

  / \ / \

  2 3 3 2

  / /

  44

**Output:** No

**Example 2:**

**Input:**

**T1** 1 **T2:** 1

  / \ / \

  2 3 3 2

  / \

  44

**Output:** Yes

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function**isomorphic()**that takesthe root nodes of both the Binary Trees as its input and returns True if the two trees are isomorphic. Else, it returns False. (The driver code will print Yes if the returned values are true, otherwise false.)

**Expected Time Complexity:**O(min(M, N)) where M and N are the sizes of the two trees.  
**Expected Auxiliary Space:**O(min(H1, H2)) where H1 and H2 are the heights of the two trees.

**Constraints:**  
1<=Number of nodes<=105

**Code: -**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

struct Node {

int data;

Node \*left;

Node \*right;

Node(int val) {

data = val;

left = right = NULL;

}

};

Node\* buildTree(string str)

{

// Corner Case

if(str.length() == 0 || str[0] == 'N')

return NULL;

// Creating vector of strings from input

// string after spliting by space

vector<string> ip;

istringstream iss(str);

for(string str; iss >> str; )

ip.push\_back(str);

// Create the root of the tree

Node\* root = new Node(stoi(ip[0]));

// Push the root to the queue

queue<Node\*> queue;

queue.push(root);

// Starting from the second element

int i = 1;

while(!queue.empty() && i < ip.size()) {

// Get and remove the front of the queue

Node\* currNode = queue.front();

queue.pop();

// Get the current node's value from the string

string currVal = ip[i];

// If the left child is not null

if(currVal != "N") {

// Create the left child for the current node

currNode->left = new Node(stoi(currVal));

// Push it to the queue

queue.push(currNode->left);

}

// For the right child

i++;

if(i >= ip.size())

break;

currVal = ip[i];

// If the right child is not null

if(currVal != "N") {

// Create the right child for the current node

currNode->right = new Node(stoi(currVal));

// Push it to the queue

queue.push(currNode->right);

}

i++;

}

return root;

}

// } Driver Code Ends

class Solution{

public:

// Return True if the given trees are isomotphic. Else return False.

bool isIsomorphic(Node \*r1,Node \*r2){

if(!r1 and !r2) return true;

else if(!r1 or !r2) return false;

if(r1->data != r2->data) return false;

if(isIsomorphic(r1->left, r2->left) == false)

if(isIsomorphic(r1->left, r2->right) == false)

return false;

if(isIsomorphic(r1->right, r2->left) == false)

if(isIsomorphic(r1->right, r2->right) == false)

return false;

return true;

}

};

//{ Driver Code Starts.

int main()

{

int t;

scanf("%d ",&t);

while(t--)

{

string s1,s2;

getline(cin,s1);

getline(cin,s2);

Node\* root1 = buildTree(s1);

Node\* root2 = buildTree(s2);

Solution obj;

if(obj.isIsomorphic(root1,root2))

cout<<"Yes"<<endl;

else

cout<<"No"<<endl;

}

return 0;

}

// } Driver Code Ends