7-1 树的同构 (25 分)

给定两棵树T1和T2。如果T1可以通过若干次左右孩子互换就变成T2，则我们称两棵树是“同构”的。例如图1给出的两棵树就是同构的，因为我们把其中一棵树的结点A、B、G的左右孩子互换后，就得到另外一棵树。而图2就不是同构的。

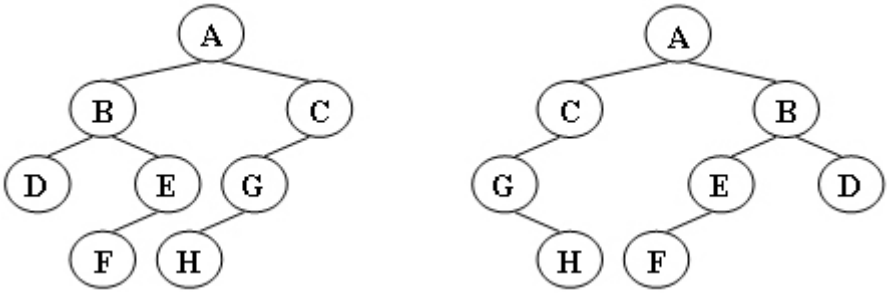

图1

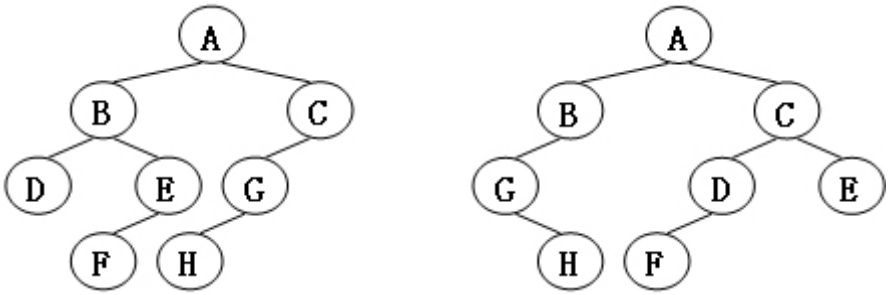

图2

现给定两棵树，请你判断它们是否是同构的。

输入格式:

输入给出2棵二叉树树的信息。对于每棵树，首先在一行中给出一个非负整数N (≤ 10)，即该树的结点数（此时假设结点从0到$N-1$编号）；随后N行，第i行对应编号第i个结点，给出该结点中存储的1个英文大写字母、其左孩子结点的编号、右孩子结点的编号。如果孩子结点为空，则在相应位置上给出“-”。给出的数据间用一个空格分隔。注意：题目保证每个结点中存储的字母是不同的。

输出格式:

如果两棵树是同构的，输出“Yes”，否则输出“No”。

输入样例1（对应图1）：

```
8
A 1 2
B 3 4
C 5 -
D - -
E 6 -
G 7 -
F - -
H - -
8
G - 4
B 7 6
F - -
```

```
A 5 1
H - -
C 0 -
D - -
E 2 -
```
输出样例1:
```
Yes
```
输入样例2（对应图2）：
```
8
B 5 7
F - -
A 0 3
C 6 -
H - -
D - -
G 4 -
E 1 -
8
D 6 -
B 5 -
E - -
H - -
C 0 2
G - 3
F - -
A 1 4
```
输出样例2:
```
No
```

```
#include<stdio.h>
#include<stdlib.h>

#define MAX 10;

typedef struct{
    char data;
    int left;
    int right;
}Node;

typedef struct{
    Node* node;
    int length;
}Tree;

Tree* build();

int main(){
```

```
    Tree* tree1;
    Tree* tree2;
    tree1=build();
    tree2=build();



    int flag=iftong(tree1,tree2);
    if(flag==1){
        printf("YES");
    }else{
        printf("NO");
    }

}


Tree* build(){
    Tree* tree;
    int num;
    scanf("%d",&num);
    tree=(Tree*)malloc(num*sizeof(Tree));
    tree->node=(Node*)malloc(sizeof(Node)*(num+1));
    tree->length=num;
    char data;
    char left;
    char right;
    int ldata,rdata;
    for(int i=0;i<num;i++){
        scanf("\n%c %c %c",&data,&left,&right);
        tree->node[i].data=data;
        if(left=='-'){
            tree->node[i].left=num;
        }else{
            ldata=left-'0';
            tree->node[i].left=ldata;
        }
        if(right=='-'){
            tree->node[i].right=num;
        }else{
            rdata=right-'0';
            tree->node[i].right=rdata;
        }
    }
    tree->node[num].data='\\';
```

```
    return tree;

}

int iftong(Tree* tree1,Tree* tree2){
    int flag=1;
    for(int i=0;i<tree1->length;i++){
        for(int j=0;j<tree2->length;j++){
            if(tree1->node[i].data==tree2->node[j].data){
                int left1=tree1->node[i].left;
                int left2=tree2->node[j].left;
                int right1=tree1->node[i].right;
                int right2=tree2->node[j].right;
                if(tree1->node[left1].data==tree2->node[left2].data&&tree1-
>node[right1].data==tree2->node[right2].data){
                   flag=1;
                }else if(tree1->node[left1].data==tree2->node[right2].data&&tree1-
>node[right1].data==tree2->node[left2].data){
                     flag=1;
                }else{
                     flag=0;
                }
                break;
            }

        }
         if(flag==0){
            return 0;
         }
    }
    return 1;

}
```