04-树4 是否同一棵二叉搜索树（25 分）

给定一个插入序列就可以唯一确定一棵二叉搜索树。然而，一棵给定的二叉搜索树却可以由多种不同的插入序列得到。例如分别按照序列{2, 1, 3}和{2, 3, 1}插入初始为空的二叉搜索树，都得到一样的结果。于是对于输入的各种插入序列，你需要判断它们是否能生成一样的二叉搜索树。

输入格式:

输入包含若干组测试数据。每组数据的第1行给出两个正整数*N* (≤10)和*L*，分别是每个序列插入元素的个数和需要检查的序列个数。第2行给出*N*个以空格分隔的正整数，作为初始插入序列。最后*L*行，每行给出*N*个插入的元素，属于*L*个需要检查的序列。

简单起见，我们保证每个插入序列都是1到*N*的一个排列。当读到*N*为0时，标志输入结束，这组数据不要处理。

输出格式:

对每一组需要检查的序列，如果其生成的二叉搜索树跟对应的初始序列生成的一样，输出“Yes”，否则输出“No”。

输入样例:

4 2 //第一组数据，第一行，每个序列有四个元素，有两个序列跟第二行进行比较

3 1 4 2 //第二行，用来进行比较

3 4 1 2 //跟第二行进行比较，结果 Yes

3 2 4 1 //跟第二行进行比较 ，结果 No

2 1 //第二组数据，的第一行，，每个序列有2个元素，有1个序列跟第二行进行比较

2 1 //第二行

1 2 //进行比较，结果No

输出样例:

Yes

No

No

04-树5 Root of AVL Tree（25 分）

AVL树是一个自平衡二叉搜索树。在AVL树中，任意节点的两个子子树的高度最多相差一个; 如果在任何时候它们相差超过一个，则重新平衡以恢复这个财产。图1-4说明了旋转规则。

 



现在给出一系列的插入，你应该告诉结果AVL树的根。

输入规格：

每个输入文件都包含一个测试用例。对于每种情况，第一行包含一个正整数*N*（≤ 2 0），其是将被插入的键的总数。然后下一行给出了*N个*不同的整数键。 一个空格分开。

输出规格：

对于每个测试用例，在一行中打印生成的AVL树的根。

示例输入1：

5

88 70 61 96 120

示例输出1：

70

示例输入2：

7

88 70 61 96 120 90 65

示例输出2：

88