# 列车调度(Train)

**Description**

Figure 1 shows the structure of a station for train dispatching.

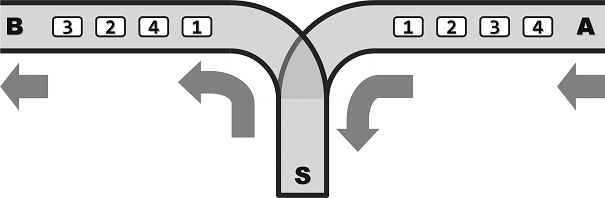


Figure 1

In this station, A is the entrance for each train and B is the exit. S is the transfer end. All single tracks are one-way, which means that the train can enter the station from A to S, and pull out from S to B. Note that the overtaking is not allowed. Because the compartments can reside in S, the order that they pull out at B may differ from that they enter at A. However, because of the limited capacity of S, no more that m compartments can reside at S simultaneously.

Assume that a train consist of n compartments labeled {1, 2, …, n}. A dispatcher wants to know whether these compartments can pull out at B in the order of {a1, a2, …, an} (a sequence). If can, in what order he should operate it?

**Input**

Two lines:

1st line: two integers n and m;

2nd line: n integers separated by spaces, which is a permutation of {1, 2, …, n}. This is a compartment sequence that is to be judged regarding the feasibility.

**Output**

If the sequence is feasible, output the sequence. “Push” means one compartment goes from A to S, while “pop” means one compartment goes from S to B. Each operation takes up one line.

If the sequence is infeasible, output a “no”.

**Example 1**

Input

5 2

1 2 3 5 4

Output

push

pop

push

pop

push

pop

push

push

pop

pop

**Example 2**

Input

5 5

3 1 2 4 5

Output

No

**Restrictions**

1 <= n <= 1,600,000

0 <= m <= 1,600,000

Time: 2 sec

Memory: 256 MB

**描述**

某列车调度站的铁道联接结构如Figure 1所示。

其中，A为入口，B为出口，S为中转盲端。所有铁道均为单轨单向式：列车行驶的方向只能是从A到S，再从S到B；另外，**不允许超车**。因为车厢可在S中驻留，所以它们从B端驶出的次序，可能与从A端驶入的次序不同。不过S的容量有限，同时驻留的车厢不得超过m节。

设某列车由编号依次为{1, 2, ..., n}的n节车厢组成。调度员希望知道，按照以上交通规则，这些车厢能否以{a1, a2, ..., an}的次序，重新排列后从B端驶出。如果可行，应该以怎样

的次序操作?

**输入**

共两行。

第一行为两个整数n，m。

第二行为以空格分隔的n个整数，保证为{1, 2, ..., n}的一个排列，表示待判断可行性的驶出序列{a1，a2，...，an}。

**输出**

若驶出序列可行，则输出操作序列，其中push表示车厢从A进入S，pop表示车厢从S进入B，每个操作占一行。

若不可行，则输出No。

**样例**

见英文题面

**限制**

1 ≤ n ≤ 1,600,000

0 ≤ m ≤ 1,600,000

时间：2 sec

空间：256 MB

# 隧道(Tunel)

**Description**

A tunnel is single and one-way. Each vehicle can only enter the tunnel at one end and exit it at the other end. Overtaking is not allowed. The tunnel has the limitation on the height of each vehicle. At any time, the administrator would like to know how high the highest vehicle is at this time.

Now you need to maintain the record of vehicles in and out with the query highest vehicle function.

**Input**

The 1st line contains only one integer, i.e., the number of the operation of both height query and vehicles in and out.

The following n lines indicate n operations in order. Each line is in one of the following formats.

1. E x //One vehicle with the height of x enters the tunnel //(x is an integer).

2. D //One vehicle exits the tunnel.

3. M //Query the height of the highest vehicle currently

**Output**

If the amount of D operation and M operation is m, output m lines. For each D operation, output the height of the vehicle that exits the tunnel. For each M operation, output all the queried maximum heights.

**Example**

Input

9

E 5

E 6

M

E 2

M

D

M

D

M

Output

6

6

5

6

6

2

**Restrictions**

0 <= n <= 2,000,000

0 <= x <= 231 - 1

Ensure that all vehicles in and out are legal.

Time: 2 sec

Memory: 256 MB

**Hints**

How to simulate a queue by multiple stacks. Please refer to the exercises in Chapter 4.

How to make a stack through which one can query the largest number?

How to make a queue through which one can query the largest number?

Please refer to the Note XA in Chapter 4 and Exercise 10-19, 10-20 in *Exercise Analysis*.

**描述**

现有一条**单向单车道**隧道，每一辆车从隧道的一端驶入，另一端驶出，不允许超车

该隧道对车辆的高度有一定限制，在任意时刻，管理员希望知道此时隧道中最高车辆的高度是多少

现在请你维护这条隧道的车辆进出记录，并支持查询最高车辆的功能

**输入**

第一行仅含一个整数，即高度查询和车辆出入操作的总次数n

以下n行，依次这n次操作。各行的格式为以下几种之一：

1. E x //有一辆高度为x的车进入隧道（x为整数）

2. D //有一辆车离开隧道

3. M //查询此时隧道中车辆的最大高度

**输出**

若D和M操作共计m次，则输出m行

对于每次D操作，输出离开隧道车辆的高度

对于每次M操作，输出所查询到的最大高度

**样例**

见英文题面

**限制**

0 ≤ n ≤ 2,000,000

0 ≤ x ≤ 231 - 1

保证车辆的进出序列是合法的

时间：2 sec

空间：256 MB

**提示**

如何由多个栈来模拟一个队列？可参考第四章末尾的某习题。

如何实现一个能够高效获取最大值的栈？

如何实现一个可以高效获取最大值的队列？

可参考第04章XA节的讲义以及《习题解析》的[10-19]题、[10-20]题

# 真二叉树重构(Proper Rebuild)

**Description**

In general, given the preorder traversal sequence and postorder traversal sequence of a binary tree, we cannot determine the binary tree.

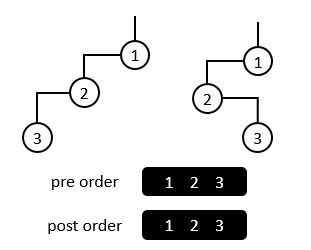


Figure 1

In Figure 1 for example, although they are two different binary tree, their preorder traversal sequence and postorder traversal sequence are both of the same.

But for one proper binary tree, in which each internal node has two sons, we can uniquely determine it through its given preorder traversal sequence and postorder traversal sequence.

Label n nodes in one binary tree using the integers in [1, n], we would like to output the inorder traversal sequence of a binary tree through its preorder and postorder traversal sequence.

**Input**

The 1st line is an integer n, i.e., the number of nodes in one given binary tree,

The 2nd and 3rd lines are the given preorder and postorder traversal sequence respectively.

**Output**

The inorder traversal sequence of the given binary tree in one line.

**Example**

Input

5

1 2 4 5 3

4 5 2 3 1

Output

4 2 5 1 3

**Restrictions**

For 95% of the estimation, 1 <= n <= 1,000,00

For 100% of the estimation, 1 <= n <= 4,000,000

The input sequence is a permutation of {1,2...n}, corresponding to a legal binary tree.

Time: 2 sec

Memory: 256 MB

**Hints**

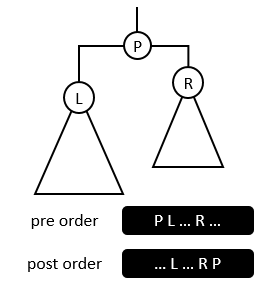
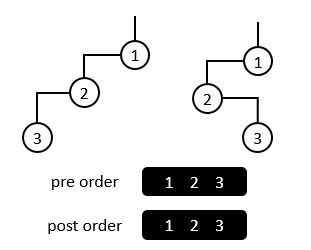


Figure 2

In Figure 2, observe the positions of the left and right children in preorder and postorder traversal sequence.

**描述**

一般来说，给定二叉树的先序遍历序列和后序遍历序列，并不能确定唯一确定该二叉树。



（图一）

比如图一中的两棵二叉树，虽然它们是不同二叉树，但是它们的先序、后序遍历序列都是相同的。

但是对于“真二叉树”（每个内部节点都有两个孩子的二叉树），给定它的先序、后序遍历序列足以完全确定它的结构。

将二叉树的n个节点用[1, n]内的整数进行编号，输入一棵真二叉树的先序、后序遍历序列，请输出它的中序遍历序列。

**输入**

第一行为一个整数n，即二叉树中节点的个数。

第二、三行为已知的先序、后序遍历序列。

**输出**

仅一行，给定真二叉树的中序遍历序列。

**样例**

见英文题面

**限制**

对于95%的测例：1 ≤ n ≤ 1,000,000

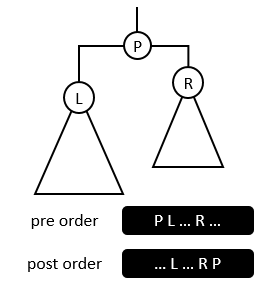
对于100%的测例：1 ≤ n ≤ 4,000,000

输入的序列是{1,2...n}的排列，且对应于一棵合法的真二叉树

时间：2 sec

空间：256 MB

**提示**



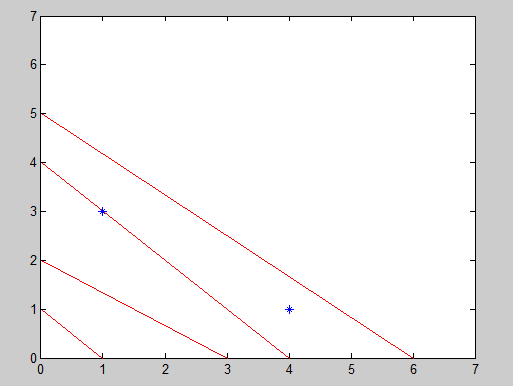
观察左、右孩子在先序、后序遍历序列中的位置

重温视频05e5-3

# THU2015 spring 2-1 Alarm

**描述**

Duck公司在仓库安装了红外报警装置，如图所示，所有红外线互不相交。n个发射器和n个接收器将平面分成n+1个区域，从左到右分别记作0、1、…、n。现在技术人员正在进行调试，对于每个点，需要快速知道它处于哪个区域。若正好处于红外线上，则视为处于**右**边的区域。



**输入**

第一行两个数n、m，表示有n条直线、m个点

接下来n行，每行两个数a、b，表示一组报警装置的发射器安装在(a, 0)，接收器安装在(0, b)。每行a、b都比前一行的大

接下来m行，每行两个数x、y，表示求点(x, y)所在的区域

**输出**

输出m行，每行一个数，对应每个点所在区域的编号

**输入样例**

4 2

1 1

3 2

4 4

6 5

1 3

4 1

**输出样例**

3

3

**限制**

1 <= n <=100,000，n为整数

1 <= m <= 1,000,000，m为整数

1 <= {a, b, x, y} <2^32，a,b,x,y均为整数

时间：2sec

空间：256MB

**提示**

\*\*一级提示\*\*

1.向量积 http://zh.wikipedia.org/wiki/%E5%8F%89%E7%A7%AF

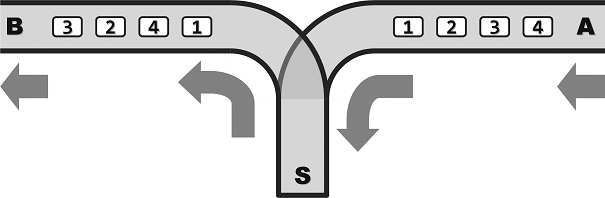
2.向量积在计算几何学中的应用 http://blog.csdn.net/william001zs/article/details/6213485

3.考虑高效的搜索方法

# THU2015 spring 2-2 Train

**描述**

某列车调度站的铁道联接结构如图所示。



其中，A为入口，B为出口，S为中转盲端。所有铁道均为单轨单向式：列车行驶的方向只能是从A到S，再从S到B；另外，不允许超车。因为车厢可在S中驻留，所以它们从B端驶出的次序，可能与从A端驶入的次序不同。不过S的容量有限，同时驻留的车厢**不得超过m节**。

设某列车由编号依次为{1, 2, ..., n}的n节车厢组成。调度员希望知道，按照以上交通规则，这些车厢能否以(a1, a2, ..., an)的次序，重新排列后从B端驶出。

**输入**

共两行。

第一行为两个数n，m。

第二行为以空格分隔的n个数，表示待判断可行性的驶出序列(a1，a2，...，an)。

**输出**

仅一行。若驶出序列可行，则输出Yes；否则输出No。

**输入样例1**

5 2

1 2 3 5 4

**输出样例1**

Yes

**输入样例2**

5 5

3 1 2 4 5

**输出样例2**

No

**限制**

1 <= n <= 100,000，n为整数

0 <= m <= 100,000，m为整数

待判断的驶出序列保证为{1, 2, ..., n}的一个排列，

时间：1sec

空间：256MB

**提示**

**一级提示**

栈混洗

# THU2015 spring 2-3 Rebuild

**描述**

某二叉树的n个节点已经用[1, n]内的整数进行了编号。现给定该二叉树的**先序**遍历序列和**中序**遍历序列，试输出其对应的后序遍历序列。

**输入**

第一行为一个数n。

第二、三行，即已知的先序、中序遍历序列，数字之间以空格分隔。

**输出**

仅一行。

若所给的先序、中续遍历序列的确对应于某棵二叉树，则输出其后序遍历序列，数字之间以空格分隔。否则，输出-1。

**输入样例1**

5

1 2 4 5 3

4 2 5 1 3

**输出样例1**

4 5 2 3 1

**输入样例2**

4

2 3 1 4

4 2 1 3

**输出样例2**

-1

**输入样例3**

8

5 2 4 1 3 6 7 8

4 2 1 5 3 7 6 8

**输出样例3**

4 1 2 7 8 6 3 5

**限制**

1 <= n <= 500,000，n为整数

输入和输出的遍历序列均为[1, n]内整数的一个排列，整数间均以空格分隔。

时间：1sec

空间：256MB

**提示**

\*\*一级提示\*\*

注意观察特殊节点在不同遍历序列中的位置

# THU2015 spring 2-4 Queap

**描述**

通过扩展已实现的基本数据结构，模拟更为复杂的数据结构，是一种常见而实用的技巧。这里，我们将利用栈结构，来模拟一种功能增强的队列。

具体来说，你必须首先实现一个元素类型为int的栈，该栈**能且只能**支持以下操作接口：

bool empty(); //判断是否栈空

void push(int); //向栈中压入一个整数

int pop(); //弹出栈顶元素，并返回其数值

int top(); //仅获取栈顶元素的值，但不弹出

接下来，你必须借助以上栈结构，模拟所谓的队堆（queap）结构，它至少支持以下操作接口：

void enqueue(int); //新元素作为尾元素入队

int dequeue(); //队首元素出队，并返回其数值（调用时输入数据保证队列非空）

int min(); //返回队列内的最小元素（调用时输入数据保证队列非空）

可见，这种结构之所以称作队堆，是因为它兼具一定的队列（queue）和堆（heap）的功能。

最后，你还需要通过一系列真实的查询和修改操作，对所实现的队堆结构进行测试。

**输入**

第一行仅含一个整数，即查询和修改操作的总数n。

以下n行，依次给出对初始为空的队堆所实施的n次操作。各行的格式为以下几种之一：

1. E x //即enqueue(int x)

2. D //即x = dequeue()

3. M //即x = min()

**输出**

若D和M操作共计m <=n 次，则共m行，依次给出各D/M操作所对应的返回值。

**输入样例**

9

E 5

E 6

M

E 2

M

D

M

D

M

**输出样例**

5

2

5

2

6

2

**限制**

0 <=m<= n <= 1,000,000，m,n均为整数

所有元素需要存储在底层栈中，即除了底层栈之外的空间使用应为O(1)。

实现队堆所用到的栈的个数应为**常数**。

不能向该存储数据的栈中添加其他public成员函数（构造、析构函数除外）或变量。

你可以根据需要包装该底层栈形成其他数据结构，其他不存储数据的结构的成员函数和变量可以自由选取。

**请通过拓展给定的模版**[**queap.cpp**](http://dsa.cs.tsinghua.edu.cn/oj/attachment/3ffa/3ffaf6a8704e3d075df704ff9cbe9100ab2df8d4.cpp)**来实现。**

时间：1sec

空间：256MB

**提示**

\*\*一级提示\*\*

考虑如何使用多个栈分别模拟:

1.可以快速返回最值的栈

2.队列