Problem C. Distribution Center

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

The factory of the Impractically Complicated Products Corporation has many manufacturing lines and the same number of corresponding storage rooms. The same number of conveyor lanes are laid out in parallel to transfer goods from manufacturing lines directly to the corresponding storage rooms. Now, they plan to install a number of robot arms here and there between pairs of adjacent conveyor lanes so that goods in one of the lanes can be picked up and released down on the other, and also in the opposite way. This should allow mixing up goods from different manufacturing lines to the storage rooms.

Depending on the positions of robot arms, the goods from each of the manufacturing lines can only be delivered to some of the storage rooms. Your task is to find the number of manufacturing lines from which goods can be transferred to each of the storage rooms, given the number of conveyor lanes and positions of robot arms.

Input

The input consists of a single test case, formatted as follows. An integer n ($2 \le n \le 2 \cdot 10^5$) in the first line is the number of conveyor lanes. The lanes are numbered from 1 to n, and two lanes with their numbers differing by 1 are adjacent. All of them start from the position $x = 0$ and end at $x = 10^5$. The other integer m ($1 \le m < 10^5$) is the number of robot arms.

The following m lines indicate the positions of the robot arms by two integers x_i ($0 < x_i < 10^5$) and y_i ($1 \le y_i < n$). Here, x_i is the x-coordinate of the i-th robot arm, which can pick goods on either the lane y_i or the lane $y_i + 1$ at position $x = x_i$, and then release them on the other at the same x-coordinate.

You can assume that positions of no two robot arms have the same x-coordinate, that is, $x_i \neq x_j$ for any $i \neq j$.

Output

Output n integers separated by a space in one line. The i-th integer is the number of the manufacturing lines from which the storage room connected to the conveyor lane i can accept goods.

Examples

standard input	standard output
4 3 1000 1 2000 2 3000 3	2 3 4 4
4 3 1 1 3 2 2 3	2 4 4 2

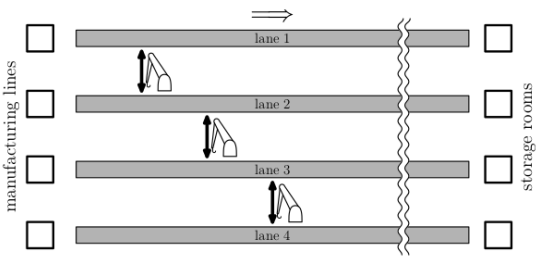

Figure C.1. Illustration of Sample Input 1

Problem J. Jogging in the Park

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The Outer Park is very well suited for jogging. Inside the park, there are n glades, conveniently numbered from 1 to n. The glades are connected by m trails, the i-th trail connects glades a_i and b_i and has length c_i meters. All trails can be walked in both directions. The main entrance is situated at glade 1.

Your friends decided to have a nice run together over the park. Each of them prepared her own route starting from the entrance and moving to subsequent glades by trails.

At the end of jogging, all your friends want to reach glade n with a beautiful cafe at the same time. Not everyone planned her route accordingly, though: some of the routes do not end at glade n, and the routes might have different length as well. Luckily, all your friends run at the same speed.

You decided to help your friends and write a program to provide each of them with an *extended* route — that is, a route which starts with the same sequence of glades as her original one, but ends at glade n and has exactly the same length, in meters, as all the other extended routes. Note that both the original and the extended routes may contain the same trail multiple times.

Input

The first line of the input contains two integers n and m ($2 \le n \le 50$; $1 \le m \le \frac{n(n-1)}{2}$) — the number of glades and trails in the park. Each of the next m lines contains three integers a_i, b_i and c_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$; $1 \le c_i \le 10^6$) — the indices of glades connected by the i-th trail and its length, respectively.

The next line of the input contains a single integer k ($2 \le k \le 50$) — the number of your jogging friends. Each of the next k lines contains an integer l_i ($2 \le l_i \le 50$) — the number of glades in the route of your i-th friend, followed by l_i integers $g_{i,1}, g_{i,2}, \ldots, g_{i,l_i}$ ($1 \le g_{i,j} \le n$) — the indices of glades in the route.

All unordered pairs (a_i, b_i) are distinct. For every i, $g_{i,1} = 1$. For every j between 1 and $l_i - 1$, there exists a trail between glades $g_{i,j}$ and $g_{i,j+1}$.

Output

If it is impossible to extend all the routes according to the problem statement, output a single integer -1. Otherwise, print k lines. The i-th of these lines must contain an integer p_i ($p_i \ge l_i$) followed by p_i integers $h_{i,1}, h_{i,2}, \ldots, h_{i,p_i}$ ($1 \le h_{i,j} \le n$) — the indices of glades in the i-th extended route.

For every i, h_{i,p_i} must be equal to n. For every j between 1 and $p_i - 1$, there must exist a trail between glades $h_{i,j}$ and $h_{i,j+1}$. For every j between 1 and l_i, $h_{i,j}$ must be equal to $g_{i,j}$. The total length of all the extended routes, in meters, must be the same.

The sum of all p_i must not exceed $2 \cdot 10^6$. It is guaranteed that if a valid route extension exists, there also exists one with the sum of all p_i not exceeding $2 \cdot 10^6$.

Example

standard input	standard output
4 4 1 2 10 2 3 30 2 4 30 1 4 100 3 2 1 2 3 1 2 3 2 1 4	5 1 2 4 2 4 5 1 2 3 2 4 2 1 4

Problem E. HDRF

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

If you love Big Data, you should be familiar with running code in a distributed manner. This always requires lots of infrastructure elements working together to make the parallel computations possible. One of such elements is usually a scheduler that decides which scheduled tasks are to be started now in some "fair" and "efficient" way.

Based on the nature of the tasks (testing, long-running, real-time, etc.), they are organized into hierarchical structure which can be represented as a rooted tree.

The following problem is inspired by one of the modern scheduling algorithms called Hierarchical Dominant Resource Fairness (HDRF).

You are given a rooted tree T with root at vertex 1 which consists of n vertices. Each vertex i of the tree gets a unique priority v_i. For each vertex, we can compute the value r_i: the smallest v_i in the subtree of vertex i including itself.

Consider the following tree traversal algorithm:

1. Start at the root vertex.
2. Choose the direct child of the current vertex which has the smallest value r_i.
3. Go to this child.
4. If the current vertex is a leaf, write it down and remove it from the tree (when we delete a vertex, we recompute all r_i). Otherwise, go to step 2.

Repeat the above procedure starting from step 1 until the tree is empty.

Given a tree T and the numbers v_i, compute the order in which vertices will be written down.

Input

The first line contains an integer n ($2 \le n \le 100\,000$), the number of vertices in the tree.

The second line contains $n - 1$ integers, where i-th integer p_i ($1 \le p_i \le n$) is the parent of vertex $(i + 1)$ in the tree. Vertices are numbered by integers from 1 to n. It is guaranteed that the input forms a valid rooted tree with root at vertex 1.

The third line contains n distinct integers $v_1, v_2, \ldots, v_n$ ($0 \le v_i \le 10^9$), the priorities of vertices.

Output

Output n vertices in the order they will be written down by the algorithm.

Example

standard input	standard output
5 4 4 1 1 3 5 2 1 4	3 2 4 5 1

Problem A. Hacker Cups and Balls

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

The dark side of Dreamoon, Ademnoor, wants to play an interesting game with you. Did you hear about the game "cups and balls"? Here is the hacker version of it!

There are n cups and n balls, both are numbered $1, 2, \ldots, n$. At each moment of time, there is exactly one ball in each cup. Initially, a_i-th ball is placed in the i-th cup. Admenoor will perform m magic operations on these balls and cups. The i-th operation will sort all the balls in the cups numbered between l_i and r_i, inclusive. The sorting could be performed in either ascending order or descending order. After these m operations, you need to answer which ball is placed in the center cup. We guarantee that n will be an odd integer, so the center cup means the $\frac{n+1}{2}$-th cup.

For example, consider $n = 5$, $m = 2$ and $a = [5, 1, 4, 2, 3]$. If the first operation is to sort the balls in the cups numbered between 1 and 4 in ascending order, then a would become $[1, 2, 4, 5, 3]$. If the second operation is to sort the balls in the cups numbered between 2 and 5 in descending order, then a would become $[1, 5, 4, 3, 2]$. In this example, the number of the ball in the center cup after all operations is 4.

Input

The first line of input contains two integers n and m. The following line contains n integers $a_1, a_2, \ldots, a_n$. Each of the following m lines contains two integers l_i and r_i. If $l_i < r_i$, Admenoor will sort that balls in ascending order in this operation; otherwise, the balls will be sorted in descending order in this operation.

- $1 \le n \le 99\,999$
- n is an odd integer
- $0 \le m \le 10^5$
- $1 \le a_i \le n$
- $\langle a_i \rangle$ is a permutation of $1, 2, \ldots, n$
- $1 \le l_i, r_i \le n$

Output

Output a single line with the number of the ball in the center cup after all operations.

Examples

standard input	standard output
3 2 1 3 2 1 3 3 1	2
5 2 5 1 4 2 3 1 4 5 2	4