

Codeforces Round #808 (Div. 1)

A. Doremy's IQ

time limit per test: 1 second
 memory limit per test: 256 megabytes
 input: standard input
 output: standard output

Doremy is asked to test n contests. Contest i can only be tested on day i . The difficulty of contest i is a_i . Initially, Doremy's IQ is q . On day i Doremy will choose whether to test contest i or not. She can only test a contest if her current IQ is strictly greater than 0.

If Doremy chooses to test contest i on day i , the following happens:

- if $a_i > q$, Doremy will feel she is not wise enough, so q decreases by 1;
- otherwise, nothing changes.

If she chooses not to test a contest, nothing changes.

Doremy wants to test as many contests as possible. Please give Doremy a solution.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n and q ($1 \leq n \leq 10^5$, $1 \leq q \leq 10^9$) — the number of contests and Doremy's IQ in the beginning.

The second line contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$) — the difficulty of each contest.

It is guaranteed that the sum of n over all test cases does not exceed 10^5 .

Output

For each test case, you need to output a binary string s , where $s_i = 1$ if Doremy should choose to test contest i , and $s_i = 0$ otherwise. The number of ones in the string should be maximum possible, and she should never test a contest when her IQ is zero or less.

If there are multiple solutions, you may output any.

Example

input
5
1 1
1
2 1
1 2
3 1
1 2 1
4 2
1 4 3 1
5 2
5 1 2 4 3
output
1
11
110
1110
01111

Note

In the first test case, Doremy tests the only contest. Her IQ doesn't decrease.

In the second test case, Doremy tests both contests. Her IQ decreases by 1 after testing contest 2.

In the third test case, Doremy tests contest 1 and 2. Her IQ decreases to 0 after testing contest 2, so she can't test contest 3.

B. Difference Array

time limit per test: 1 second
 memory limit per test: 256 megabytes
 input: standard input
 output: standard output

You are given an array a consisting of n non-negative integers. It is guaranteed that a is sorted from small to large.

For each operation, we generate a new array $b_i = a_{i+1} - a_i$ for $1 \leq i < n$. Then we sort b from small to large, replace a with b , and decrease n by 1.

After performing $n - 1$ operations, n becomes 1. You need to output the only integer in array a (that is to say, you need to output a_1).

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains one integer n ($2 \leq n \leq 10^5$) — the length of the array a .

The second line contains n integers a_1, a_2, \dots, a_n ($0 \leq a_1 \leq \dots \leq a_n \leq 5 \cdot 10^5$) — the array a .

It is guaranteed that the sum of n over all test cases does not exceed $2.5 \cdot 10^5$, and the sum of a_n over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output the answer on a new line.

Example

input
5 3 1 10 100 4 4 8 9 13 5 0 0 0 8 13 6 2 4 8 16 32 64 7 0 0 0 0 0 0 0
output
81 3 1 2 0

Note

To simplify the notes, let $\text{sort}(a)$ denote the array you get by sorting a from small to large.

In the first test case, $a = [1, 10, 100]$ at first. After the first operation, $a = \text{sort}([10 - 1, 100 - 10]) = [9, 90]$. After the second operation, $a = \text{sort}([90 - 9]) = [81]$.

In the second test case, $a = [4, 8, 9, 13]$ at first. After the first operation, $a = \text{sort}([8 - 4, 9 - 8, 13 - 9]) = [1, 4, 4]$. After the second operation, $a = \text{sort}([4 - 1, 4 - 4]) = [0, 3]$. After the last operation, $a = \text{sort}([3 - 0]) = [3]$.

C. DFS Trees

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a connected undirected graph consisting of n vertices and m edges. The weight of the i -th edge is i .

Here is a wrong algorithm of finding a [minimum spanning tree](#) (MST) of a graph:

```
vis := an array of length n
s := a set of edges

function dfs(u):
    vis[u] := true
    iterate through each edge (u, v) in the order from smallest to largest edge weight
        if vis[v] = false
            add edge (u, v) into the set (s)
            dfs(v)

function findMST(u):
    reset all elements of (vis) to false
```

```
reset the edge set (s) to empty
dfs(u)
return the edge set (s)
```

Each of the calls `findMST(1)`, `findMST(2)`, ..., `findMST(n)` gives you a spanning tree of the graph. Determine which of these trees are minimum spanning trees.

Input

The first line of the input contains two integers n, m ($2 \leq n \leq 10^5, n - 1 \leq m \leq 2 \cdot 10^5$) — the number of vertices and the number of edges in the graph.

Each of the following m lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$), describing an undirected edge (u_i, v_i) in the graph. The i -th edge in the input has weight i .

It is guaranteed that the graph is connected and there is at most one edge between any pair of vertices.

Output

You need to output a binary string s , where $s_i = 1$ if `findMST(i)` creates an MST, and $s_i = 0$ otherwise.

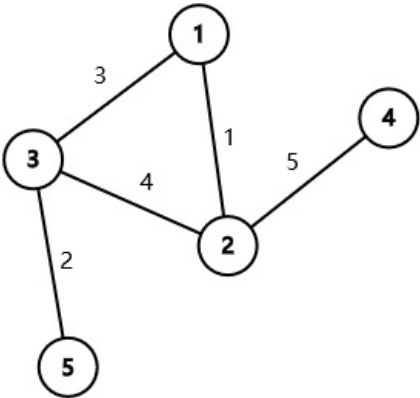
Examples

input
5 5 1 2 3 5 1 3 3 2 4 2
output
01111

input
10 11 1 2 2 5 3 4 4 2 8 1 4 5 10 5 9 5 8 2 5 7 4 6
output
0011111011

Note

Here is the graph given in the first example.



There is only one minimum spanning tree in this graph. A minimum spanning tree is $(1, 2), (3, 5), (1, 3), (2, 4)$ which has weight $1 + 2 + 3 + 5 = 11$.

Here is a part of the process of calling `findMST(1)`:

- reset the array `vis` and the edge set `s`;
- calling `dfs(1)`;
- `vis[1] := true`;
- iterate through each edge $(1, 2), (1, 3)$;
- add edge $(1, 2)$ into the edge set `s`, calling `dfs(2)`:
 - `vis[2] := true`
 - iterate through each edge $(2, 1), (2, 3), (2, 4)$;

- because `vis[1] = true`, ignore the edge $(2, 1)$;
- add edge $(2, 3)$ into the edge set s , calling `dfs(3)`:
- ...

In the end, it will select edges $(1, 2), (2, 3), (3, 5), (2, 4)$ with total weight $1 + 4 + 2 + 5 = 12 > 11$, so `findMST(1)` does not find a minimum spanning tree.

It can be shown that the other trees are all MSTs, so the answer is 01111.

D. Partial Virtual Trees

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Kawashiro Nitori is a girl who loves competitive programming. One day she found a rooted tree consisting of n vertices. The root is vertex 1. As an advanced problem setter, she quickly thought of a problem.

Kawashiro Nitori has a vertex set $U = \{1, 2, \dots, n\}$. She's going to play a game with the tree and the set. In each operation, she will choose a vertex set T , where T is a *partial virtual tree* of U , and change U into T .

A vertex set S_1 is a partial virtual tree of a vertex set S_2 , if S_1 is a subset of S_2 , $S_1 \neq S_2$, and for all pairs of vertices i and j in S_1 , $\text{LCA}(i, j)$ is in S_1 , where $\text{LCA}(x, y)$ denotes the **lowest common ancestor** of vertices x and y on the tree. Note that a vertex set can have many different partial virtual trees.

Kawashiro Nitori wants to know for each possible k , if she performs the operation **exactly** k times, in how many ways she can make $U = \{1\}$ in the end? Two ways are considered different if there exists an integer z ($1 \leq z \leq k$) such that after z operations the sets U are different.

Since the answer could be very large, you need to find it modulo p . It's guaranteed that p is a prime number.

Input

The first line contains two integers n and p ($2 \leq n \leq 2000$, $10^8 + 7 \leq p \leq 10^9 + 9$). It's guaranteed that p is a prime number.

Each of the next $n - 1$ lines contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n$), representing an edge between u_i and v_i .

It is guaranteed that the given edges form a tree.

Output

The only line contains $n - 1$ integers — the answer modulo p for $k = 1, 2, \dots, n - 1$.

Examples

input
4 998244353 1 2 2 3 1 4
output
1 6 6

input
7 100000007 1 2 1 3 2 4 2 5 3 6 3 7
output
1 47 340 854 880 320

input
8 1000000007 1 2 2 3 3 4 4 5 5 6 6 7 7 8
output
1 126 1806 8400 16800 15120 5040

Note

In the first test case, when $k = 1$, the only possible way is:

- 1. $\{1, 2, 3, 4\} \rightarrow \{1\}$.

When $k = 2$, there are 6 possible ways:

- 1. $\{1, 2, 3, 4\} \rightarrow \{1, 2\} \rightarrow \{1\}$;
- 2. $\{1, 2, 3, 4\} \rightarrow \{1, 2, 3\} \rightarrow \{1\}$;
- 3. $\{1, 2, 3, 4\} \rightarrow \{1, 2, 4\} \rightarrow \{1\}$;
- 4. $\{1, 2, 3, 4\} \rightarrow \{1, 3\} \rightarrow \{1\}$;
- 5. $\{1, 2, 3, 4\} \rightarrow \{1, 3, 4\} \rightarrow \{1\}$;
- 6. $\{1, 2, 3, 4\} \rightarrow \{1, 4\} \rightarrow \{1\}$.

When $k = 3$, there are 6 possible ways:

- 1. $\{1, 2, 3, 4\} \rightarrow \{1, 2, 3\} \rightarrow \{1, 2\} \rightarrow \{1\}$;
- 2. $\{1, 2, 3, 4\} \rightarrow \{1, 2, 3\} \rightarrow \{1, 3\} \rightarrow \{1\}$;
- 3. $\{1, 2, 3, 4\} \rightarrow \{1, 2, 4\} \rightarrow \{1, 2\} \rightarrow \{1\}$;
- 4. $\{1, 2, 3, 4\} \rightarrow \{1, 2, 4\} \rightarrow \{1, 4\} \rightarrow \{1\}$;
- 5. $\{1, 2, 3, 4\} \rightarrow \{1, 3, 4\} \rightarrow \{1, 3\} \rightarrow \{1\}$;
- 6. $\{1, 2, 3, 4\} \rightarrow \{1, 3, 4\} \rightarrow \{1, 4\} \rightarrow \{1\}$.

E. Replace

time limit per test: 1.5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

You are given an integer array a_1, \dots, a_n , where $1 \leq a_i \leq n$ for all i .

There's a "replace" function f which takes a pair of integers (l, r) , where $l \leq r$, as input and outputs the pair

$$f((l, r)) = (\min\{a_l, a_{l+1}, \dots, a_r\}, \max\{a_l, a_{l+1}, \dots, a_r\}).$$

Consider repeated calls of this function. That is, from a starting pair (l, r) we get $f((l, r))$, then $f(f((l, r)))$, then $f(f(f((l, r))))$, and so on.

Now you need to answer q queries. For the i -th query you have two integers l_i and r_i ($1 \leq l_i \leq r_i \leq n$). You must answer the minimum number of times you must apply the "replace" function to the pair (l_i, r_i) to get $(1, n)$, or report that it is impossible.

Input

The first line contains two positive integers n, q ($1 \leq n, q \leq 10^5$) — the length of the sequence a and the number of the queries.
The second line contains n positive integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq n$) — the sequence a .
Each line of the following q lines contains two integers l_i, r_i ($1 \leq l_i \leq r_i \leq n$) — the queries.

Output

For each query, output the required number of times, or -1 if it is impossible.

Examples

input
5 6 2 5 4 1 3 4 4 1 5 1 4 3 5 4 5 2 3
output
-1 0 1 2 3 4
input
6 3 2 3 4 6 1 2 5 6 2 5 2 3

output
5 1 3
input
5 3 3 2 2 4 1 2 5 1 3 1 5
output
-1 -1 0

Note

In the first example, $n = 5$ and $a = [2, 5, 4, 1, 3]$.

For the first query: $(4, 4) \rightarrow (1, 1) \rightarrow (2, 2) \rightarrow (5, 5) \rightarrow (3, 3) \rightarrow (4, 4) \rightarrow \dots$, so it's impossible to get $(1, 5)$.

For the second query, you already have $(1, 5)$.

For the third query: $(1, 4) \rightarrow (1, 5)$.

For the fourth query: $(3, 5) \rightarrow (1, 4) \rightarrow (1, 5)$.

For the fifth query: $(4, 5) \rightarrow (1, 3) \rightarrow (2, 5) \rightarrow (1, 5)$.

For the sixth query: $(2, 3) \rightarrow (4, 5) \rightarrow (1, 3) \rightarrow (2, 5) \rightarrow (1, 5)$.

F. Bugaboo

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

A transformation of an array of positive integers a_1, a_2, \dots, a_n is defined by replacing a with the array b_1, b_2, \dots, b_n given by $b_i = a_i \oplus a_{(i \bmod n)+1}$, where \oplus denotes the bitwise XOR operation.

You are given integers n , t , and w . We consider an array c_1, c_2, \dots, c_n ($0 \leq c_i \leq 2^w - 1$) to be *bugaboo* if and only if there exists an array a_1, a_2, \dots, a_n such that after transforming a for t times, a becomes c .

For example, when $n = 6$, $t = 2$, $w = 2$, then the array $[3, 2, 1, 0, 2, 2]$ is bugaboo because it can be given by transforming the array $[2, 3, 1, 1, 0, 1]$ for 2 times:

$$\begin{aligned} [2, 3, 1, 1, 0, 1] &\rightarrow [2 \oplus 3, 3 \oplus 1, 1 \oplus 1, 1 \oplus 0, 0 \oplus 1, 1 \oplus 2] = [1, 2, 0, 1, 1, 3]; \\ [1, 2, 0, 1, 1, 3] &\rightarrow [1 \oplus 2, 2 \oplus 0, 0 \oplus 1, 1 \oplus 1, 1 \oplus 3, 3 \oplus 1] = [3, 2, 1, 0, 2, 2]. \end{aligned}$$

And the array $[4, 4, 4, 4, 0, 0]$ is not bugaboo because $4 > 2^2 - 1$. The array $[2, 3, 3, 3, 3, 3]$ is also not bugaboo because it can't be given by transforming one array for 2 times.

You are given an array c with some positions lost (only m positions are known at first and the remaining positions are lost). And there are q modifications, where each modification is changing a position of c . A modification can possibly change whether the position is lost or known, and it can possibly redefine a position that is already given.

You need to calculate how many possible arrays c (with arbitrary elements on the lost positions) are bugaboos after each modification. Output the i -th answer modulo p_i (p_i is a given array consisting of q elements).

Input

The first line contains four integers n , m , t and w ($2 \leq n \leq 10^7$, $0 \leq m \leq \min(n, 10^5)$, $1 \leq t \leq 10^9$, $1 \leq w \leq 30$).

The i -th line of the following m lines contains two integers d_i and e_i ($1 \leq d_i \leq n$, $0 \leq e_i < 2^w$). It means the position d_i of the array c is given and $c_{d_i} = e_i$. It is guaranteed that $1 \leq d_1 < d_2 < \dots < d_m \leq n$.

The next line contains only one number q ($1 \leq q \leq 10^5$) — the number of modifications.

The i -th line of the following q lines contains three integers f_i , g_i , p_i ($1 \leq f_i \leq n$, $-1 \leq g_i < 2^w$, $11 \leq p_i \leq 10^9 + 7$). The value $g_i = -1$ means changing the position f_i of the array c to a lost position, otherwise it means changing the position f_i of the array c to a known position, and $c_{f_i} = g_i$. The value p_i means you need to output the i -th answer modulo p_i .

Output

The output contains q lines, denoting your answers.

Examples

input
3 2 1 1 1 1 3 1 4 2 0 123456789 2 1 111111111 1 -1 987654321 3 -1 555555555
output
1 0 1 2

input
24 8 5 4 4 4 6 12 8 12 15 11 16 7 20 2 21 9 22 12 13 2 13 11 3 15 12 5 7 13 9 3 14 10 5 15 11 15 16 13 14 17 14 1 18 18 9 19 19 6 20 23 10 21 24 8 22 21 13 23
output
1 4 9 2 1 0 1 10 11 16 16 0 16

Note

In the first example, $n = 3$, $t = 1$, and $w = 1$. Let $?$ denote a lost position of c .

In the first query, $c = [1, 0, 1]$. The only possible array $[1, 0, 1]$ is bugaboo because it can be given by transforming $[0, 1, 1]$ once. So the answer is $1 \bmod 123\,456\,789 = 1$.

In the second query, $c = [1, 1, 1]$. The only possible array $[1, 1, 1]$ is not bugaboo. So the answer is $0 \bmod 111\,111\,111 = 0$.

In the third query, $c = [?, 1, 1]$. There are two possible arrays $[1, 1, 1]$ and $[0, 1, 1]$. Only $[0, 1, 1]$ is bugaboo because it can be given by transforming $[1, 1, 0]$ once. So the answer is $1 \bmod 987\,654\,321 = 1$.

In the fourth query, $c = [?, 1, ?]$. There are four possible arrays. $[0, 1, 1]$ and $[1, 1, 0]$ are bugaboos. $[1, 1, 0]$ can be given by performing $[1, 0, 1]$ once. So the answer is $2 \bmod 555\,555\,555 = 2$.