Problem A. Ancestors And Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	512 mebibytes

You are given a rooted tree with nodes numbered $1, 2, \ldots, n$, where the root is node 1.

You are also given a sequence $a_1, a_2, \ldots, a_m$ of length m, consisting of integers from 1 to n, each element representing a node of the tree.

You have to answer q queries. In each query, you are given an interval of the sequence and a node of the tree. You need to find the maximum node number obtained by taking the LCA of the given tree node and each node selected from the interval of the sequence.

Formally, let $\mathrm{LCA}(u, v)$ denote the Lowest Common Ancestor of nodes u and v in the tree. For a query (ℓ, r, u), you need to compute $\max_{\ell \le k \le r} \mathrm{LCA}(a_k, u)$.

Input

The first line of input contains three integers: n, m, and q ($1 \le n, m, q \le 5 \cdot 10^5$).

Each of the next $n - 1$ lines contains two integers, u and v, representing an edge between nodes u and v ($1 \le u, v \le n$). The resulting graph is a tree.

The next line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$).

Each of the next q lines contains three integers: ℓ, r, and u ($1 \le \ell \le r \le m$, $1 \le u \le n$), representing a query about the interval $[a_\ell, a_r]$ of the sequence and the tree node u.

Output

For each query, output one line containing a single integer: the answer to that query.

Example

standard input	*standard output*
10 12 20 1 10 1 9 9 4 9 5 4 8 4 7 5 2 7 6 2 3 10 8 6 4 3 2 5 7 1 4 6 7 5 8 1 1 12 1 5 6 2 1 3 2 5 5 3 5 7 3 8 12 4 1 5 4 1 1 5 5 6 5 11 12 6 1 2 6 9 12 7 7 9 7 1 4 8 6 11 8 1 1 9 9 10 9 2 12 10 1 1 10	1 1 2 9 3 5 4 9 1 5 7 4 7 9 8 9 1 9 1 10

Problem B. Borders And Center

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

You are given a string s of length n. The *center* of a substring $s[i \dots j]$ is defined as $(i + j)/2$.

Now you need to answer q queries. In each query, given an integer k, you are to compute the sum of the number of *borders* over all substrings whose center is $k/2$.

A non-empty string t is a *border* of another string s if and only if t is both a prefix and a suffix of s. For example, for any non-empty string s, s itself is a border of s.

Input

The first line contains two integers, n ($1 \le n \le 10^6$) and q ($1 \le q \le 20$), denoting the length of the string s and the number of queries.

The second line contains a string s of length n, consisting of lowercase English letters.

Each of the following q lines contains an integer k ($2 \le k \le 2n$), representing a query.

Output

Print q lines, the i-th line being the answer to the i-th query.

Example

standard input	*standard output*
11 6 ccccddddcdc 22 19 12 13 16 17	1 3 10 11 5 4

Note

Explanation for the example:

- When $k = 22$, the only substring with center $k/2$ is $s[11 \dots 11] =$ `c`, whose number of borders is 1.
- When $k = 19$, the substrings with center $19/2$ are $s[9 \dots 10] =$ `cd` and $s[8 \dots 11] =$ `dcdc`, whose numbers of borders are 1 and 2, respectively.
- When $k = 12$, the substrings with center $12/2$ are:

$$s[6 \dots 6] = \texttt{d}\ (1),$$
$$s[5 \dots 7] = \texttt{ddd}\ (3),$$
$$s[4 \dots 8] = \texttt{cdddd}\ (1),$$
$$s[3 \dots 9] = \texttt{ccddddc}\ (2),$$
$$s[2 \dots 10] = \texttt{cccddddcd}\ (1),$$
$$s[1 \dots 11] = \texttt{ccccddddcdc}\ (2).$$

The numbers in parentheses are the respective border counts.

Problem C. Covering Game

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

WCat is playing a covering game.

He has an $n \times n$ grid, where the cell in row i and column j is labelled (i, j). Exactly n cells are black, and the rest are white. Moreover, any two black cells are in different rows and different columns.

For each positive integer k, WCat has infinitely many red rectangles of size $1 \times k$ and infinitely many blue rectangles of size $k \times 1$. WCat needs to cover the grid with these rectangles.

In this covering game, red rectangles can only be placed horizontally, and blue rectangles can only be placed vertically.

More precisely:

- For a $1 \times k$ red rectangle, WCat may choose positive integers i and j such that $1 \le i, j + k - 1 \le n$, and cover the cells $(i, j + \ell)$ for $\ell = 0, 1, \ldots, k - 1$.
- For a $k \times 1$ blue rectangle, WCat may choose positive integers i and j such that $1 \le i + k - 1, j \le n$, and cover the cells $(i + \ell, j)$ for $\ell = 0, 1, \ldots, k - 1$.

WCat must cover all white cells with these rectangles without overlapping, and no rectangle may cover any black cell. If WCat uses exactly $2n - 2$ rectangles, he wins the game.

Two rectangles are considered *identical* in two covering schemes if they have the same color and cover exactly the same set of white cells; otherwise, they are considered *different*.

WCat wants to know how many distinct covering schemes allow him to win the covering game. Two covering schemes are considered different if and only if there exists a white cell such that the rectangle covering it in the first scheme is different from the rectangle covering it in the second scheme. Since the number of schemes may be large, you only need to output the result modulo $10^9 + 7$.

Input

The first line contains an integer n ($2 \le n \le 2 \cdot 10^5$), the side length of the grid.

The second line contains a sequence of n integers $a_1, a_2, \ldots, a_n$, where the black cell in column i is in row a_i. This sequence is a permutation of $1, 2, \ldots, n$.

Output

Print one integer: the number of covering schemes that allow WCat to win the game, modulo $10^9 + 7$.

Examples

standard input	*standard output*
2 1 2	4
3 1 2 3	16
3 1 3 2	12

Note

In the first example, WCat can only cover with 1×1 rectangles. Each such rectangle can be either red or blue, so there are $2 \cdot 2 = 4$ covering schemes.

Problem D. DAGPower

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

City K is located between a warm current and mountainous terrain, enjoying unique meteorological conditions that support a thriving traditional handicraft industry. Due to the influence of water vapor, the city's average annual sunshine duration is lower than the national average, so its electricity supply has long relied mainly on thermal power generation. Additionally, its rich historical heritage has made cultural tourism an important pillar industry. Large-scale installation of solar photovoltaic panels might affect the visual appeal of tourist attractions. With the decreasing cost of solar power generation systems, represented by photovoltaic power generation, the city has decided to replace some of its high-consumption, low-efficiency thermal power generation capacity with solar power, in order to better achieve energy conservation, emission reduction, and green low-carbon development goals.

The power transmission network of City K can be viewed as a directed acyclic graph with n nodes and m edges. Nodes without incoming edges represent thermal power plants currently in operation. The remaining nodes with incoming edges represent facilities that require electricity supply. As the operator of City K's power grid, DAGPower Company plans to temporarily convert some thermal power plants into solar power plants for a trial grid operation.

Let S_t denote the set of nodes that remain as thermal power plants during the trial, and S_p denote the set of nodes that DAGPower temporarily converts into solar power plants. DAGPower originally had a temporary conversion plan, but during equipment configuration, they discovered that this plan might easily cause large-scale failures. Specifically, during the trial, there may be facilities that are directly or indirectly connected to both S_t and S_p, receiving power from two different types of power plants simultaneously, which significantly increases the risk of power coupling anomalies and other emergencies. The total risk is estimated as the sum of peak power loads for all such facilities.

However, due to the tight schedule, DAGPower cannot redesign the plan and redeploy equipment from scratch. To minimize the impact on the production and daily life of City K during the trial, DAGPower hopes to develop an emergency conversion plan based on the original scheme. The goal of the emergency plan is to isolate the two power supply types as much as possible, so that the total risk to the city's power grid is minimized. For that, some power plants can be urgently converted to the other type: thermal into solar, or solar back into thermal. However, urgent conversion itself increases the risk, as some power generation capacity would be lost.

Formally, for node i which is a facility, let a_i amperes be its peak power load, and for node i which is a power plant, let a_i amperes be the loss of power generation capacity in case of urgent conversion. The emergency plan works as follows:

- First, DAGPower urgently converts some power plants to the other type.
- After the urgent conversion, let S'_t denote the set of thermal power plants, and S'_p denote the set of solar power plants. For the emergency plan, DAGPower allows S'_t or S'_p to be empty.
- Now, the total risk is estimated as the sum of a_i of facilities that are simultaneously connected to both S'_t and S'_p, plus the sum of a_i for all power plants that were urgently converted to the other type.

Your task is to determine the minimal possible total risk after following an emergency plan.

Input

The first line of input contains two integers, n and m: the number of nodes and the number of direct transmission lines in City K's power network ($3 \le n \le 5000$, $1 \le m \le \min(50\,000, n \cdot (n-1)/2)$.

The second line contains n integers $d_1, \ldots, d_n$, indicating the type of each node:

- If $d_i = -1$, node i is a power-consuming facility.
- If $d_i = 0$, node i is a power plant which will produce thermal power in the original plan ($i \in S_t$).
- If $d_i = 1$, node i is a power plant which will produce solar power in the original plan ($i \in S_p$).

At least one d_i is 1.

The third line contains n integers $a_1, \ldots, a_n$, representing the peak power load (in amperes) of each facility node, or the power generation loss (in amperes) incurred when urgently converting a power plant to the other type ($1 \le a_i \le 40$).

Each of the following m lines contains two integers, u_i and v_i, meaning there is a transmission line in the grid that directly delivers power from u_i to v_i ($1 \le u_i, v_i \le n$). There are no duplicate edges. The input graph is a directed acyclic graph with at least two vertices of in-degree zero. The value d_i equals -1 if and only if vertex i has an in-degree greater than zero.

Output

Print one integer: the total risk (in amperes) of the optimal emergency conversion plan.

Examples

standard input	*standard output*
6 6 0 1 0 1 -1 -1 11 17 1 13 2 28 1 5 2 5 3 5 2 6 3 6 4 6	3
6 7 0 1 0 -1 -1 -1 11 17 1 13 2 28 1 4 2 4 2 5 2 6 3 5 4 6 5 6	12

Problem E. Edges of LCP Weight

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

For a string s of length n consisting only of lowercase English letters, consider an undirected weighted graph with n vertices, where between any two distinct vertices i and j, there is an edge of weight $\mathrm{LCP}(\mathrm{suf}_i, \mathrm{suf}_j)$. Here, LCP denotes the length of the longest common prefix of two strings, and suf_i is the i-th suffix of the string s: the string $s_i s_{i+1} \dots s_n$.

Egrade defines the *value* of the string s as the total weight of edges in the **maximum** spanning tree of this graph: the subgraph which is a tree and has the maximum possible total weight of edges.

Egrade asks you to find, among all strings of length n consisting only of lowercase English letters, any string whose value is the k-th smallest.

Input

The first line of input contains an integer t ($1 \le t \le 2 \cdot 10^5$), the number of test cases.

Each test case is given on a single line with two integers, n and k ($1 \le n \le 4 \cdot 10^5$, $1 \le k \le \min(26^n, 10^{15})$).

The sum of all n is at most $4 \cdot 10^5$.

Output

For each test case:

- On the first line, output the k-th smallest value.
- On the second line, output a string of length n whose value is the k-th smallest. If multiple strings satisfy the condition, output any one of them.

Example

standard input	*standard output*
3 2 1 2 676 3 16000	0 hi 1 gg 1 qwq

Note

Explanation of the example:

- For the first test case, among strings of length 2, the smallest (first) value is 0. One string satisfying the condition is "`hi`". Strings like "`ab`" or "`yz`" also satisfy the condition.
- For the second test case, among strings of length 2, the 676-th smallest (the largest) value is 1. One string satisfying the condition is "`gg`". Strings like "`aa`" or "`zz`" also satisfy the condition.
- For the third test case, among strings of length 3, the 16 000-th smallest value is 1. One string satisfying the condition is "`qwq`". Strings like "`cpp`" or "`lol`" also satisfy the condition.

Problem F. Find The XOR Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Mr. Mej has a strange tree in his garden.

Each vertex of this tree has a label: the vertex with label 1 is the root, and every positive integer appears exactly once as a vertex label.

The vertex with label i has exactly i child vertices, and the labels of these i children are consecutive integers. Formally, let $mn(i)$ be the smallest label among the children of vertex i, and $mx(i)$ be the largest label among them. Then, $mx(i) - mn(i) = i - 1$, and every integer in the range $mn(i)$ to $mx(i)$ appears exactly once.

Moreover, for any $1 \le i < j$, we have $mx(i) < mn(j)$.

These properties uniquely determine the structure of the strange tree. For example, vertex 1 has child 2; vertex 2 has children 3 and 4; vertex 3 has children 5, 6, and 7; and so on.

However, because Mr. Mej does not really like infinite trees, he recently cut almost all vertices from the tree: he only kept the vertices with labels between 1 and n inclusive.

Mr. Mej can draw magical power from this strange tree. Specifically, each vertex on the tree has a *magic value*, which is initially 0. Mr. Mej can choose a vertex x to gather magic, at which time he will obtain the XOR sum of the magic values of all vertices in the subtree of x, including x itself.

Mr. Mej also performs maintenance on the tree. If he chooses a vertex x and a value c, he can cast a maintenance spell that performs a bitwise OR with c on the magic values of all vertices in the subtree of x, including x itself.

Now, Mr. Mej has performed q operations in total, each operation being either maintenance or gathering magic. He wants to know the amount of magic obtained each time he gathers magic.

Input

The first line of the input contains two integers, n and q ($2 \le n \le 10^{18}$, $1 \le q \le 10^6$), representing the size of the tree and the number of operations.

The next q lines each start with an integer op ($op \in \{1, 2\}$), denoting the type of operation:

- If $op = 1$, then two integers x and c follow ($1 \le x \le n$, $1 \le c < 2^{60}$), meaning to apply bitwise OR with c to the magic values of all vertices in the subtree of x.
- If $op = 2$, then one integer x follows ($1 \le x \le n$), meaning to query the XOR sum of the magic values of all vertices in the subtree of x.

Output

For each query (operation with $op = 2$), print a line with a single integer: the XOR sum being asked.

Example

standard input	*standard output*
11 6 1 3 931 1 4 209 1 2 28 2 1 1 8 287 2 4	193 479

Note

In the first example, initially, the magic values of all vertices are 0.

- First operation: apply bitwise OR with 931 to the subtree of vertex 3. The magic values become: 0, 0, 931, 0, 931, 931, 931, 0, 0, 0.
- Second operation: apply bitwise OR with 209 to the subtree of vertex 4. The magic values become: 0, 0, 931, 209, 931, 931, 931, 209, 209, 209.
- Third operation: apply bitwise OR with 28 to the subtree of vertex 2. The magic values become: 0, 28, 959, 221, 959, 959, 959, 221, 221, 221, 221.
- Fourth operation: query the XOR sum of the subtree of vertex 1, which is: $0 \oplus 28 \oplus 959 \oplus 221 \oplus 959 \oplus 959 \oplus 959 \oplus 221 \oplus 221 \oplus 221 \oplus 221 = 193$.
- Fifth operation: apply bitwise OR with 287 to the subtree of vertex 8. The magic values become: 0, 28, 959, 221, 959, 959, 959, 479, 221, 221, 221.
- Sixth operation: query the XOR sum of the subtree of vertex 4, which is: $221 \oplus 479 \oplus 221 \oplus 221 \oplus 221 = 479$.

Problem G. Get Two Strings From One

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

Given a binary string of length n, you need to partition it into two subsequences (which may be empty) so that the sum of the two subsequences, each interpreted as a binary number, is minimized. In particular, an empty subsequence is treated as the binary number 0. Output this minimal sum in binary form.

Input

The first line of the input contains an integer t ($1 \le t \le 10^5$), the number of test cases. For each test case:

- The first line contains an integer n ($1 \le n \le 5 \cdot 10^5$).
- The second line contains a binary string of length n.

The sum of all n does not exceed $5 \cdot 10^5$.

Output

For each test case, print a line with the answer in binary form without extra leading zeroes. Print zero as a single '0'.

Example

standard input	*standard output*
3 4 0101 3 000 4 0011	10 0 10

Note

Explanation for the example:

- For the first test case, one optimal partition is this: take the characters at positions 1 and 2 as the first subsequence, and the characters at positions 3 and 4 as the second subsequence. The answer is $01_2 + 01_2 = 10_2$.
- For the second test case, the answer is obviously 0. Note that you must output "0" and not an empty line.
- For the third test case, one optimal partition is this: take the characters at positions 1 and 3 as the first subsequence, and the characters at positions 2 and 4 as the second subsequence. The answer is $01_2 + 01_2 = 10_2$.

Problem H. Help For Future Navigators

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	1024 mebibytes

Little Q is a teacher of the light-speed spaceship course at Interstellar Elementary School. She plans to purchase n distinct energy stones from the school's Interstellar Energy Store. Each energy stone contains one unit of energy and bears a unique integer label between 1 and n. She wants to distribute them among m students to help them practice flying light-speed spaceships.

During the distribution, each energy stone is given to exactly one student, and a student may receive no stones. Let c_i denote the total number of energy stones assigned to student i, so that $\sum_{i=1}^{m} c_i = n$ and $c_i \geq 0$. Two distribution schemes are considered *essentially different* if and only if there exists a student whose set of stone labels differs between the two schemes. For example, with two stones and two students: giving stone 1 to student 1 and stone 2 to student 2 is essentially different from giving stone 1 to student 2 and stone 2 to student 1, because student 1's assigned set is $\{1\}$ in the first scheme and $\{2\}$ in the second.

Every student wants to receive more energy stones for the light-speed spaceship than the others, so if a student gets more stones than another student, they will have greater satisfaction.

After distributing the n stones, we compute students' satisfaction. Satisfaction of each student i is computed as follows. Initially, it is set to 1. Then we consider $m-1$ other students j ($j \neq i$) one by one:

- If $c_i > c_j$, student i's satisfaction is multiplied by $A_{i,j}$;
- If $c_i = c_j$, student i's satisfaction is multiplied by $B_{i,j}$;
- If $c_i < c_j$, student i's satisfaction does not change.

The values satisfy $1 < B_{i,j} < A_{i,j} \leq 10^9$ for any $j \neq i$.

Little Q defines the *weight* of a distribution scheme as the product of the satisfactions of all m students after the distribution. Little Q wants to know, for all essentially different distribution schemes, the sum of their weights modulo 317. However, Little Q is a light-speed spaceship teacher, not a light-speed computation teacher, so she asks you, an expert in light-speed calculation, to quickly tell her the result modulo 317.

Moreover, because Little Q did not test the performance of the spaceships beforehand, the energy requirement may vary with different spaceship performances. Thus she will ask q possible total numbers of stones n to purchase. For each n, you need to tell her the sum of the weights of all distribution schemes if she purchases n stones, modulo 317.

Input

The first line contains an integer m ($1 \leq m \leq 12$), the number of students.

The next m lines contain the matrix A of size $m \times m$. Each line contains m integers; the number in row i, column j is $A_{i,j}$. When $i = j$, $A_{i,j} = 0$. When $i \neq j$, $2 < A_{i,j} \leq 10^9$.

The next m lines contain the matrix B of size $m \times m$. Each line contains m integers; the number in row i, column j is $B_{i,j}$. When $i = j$, $B_{i,j} = 0$. When $i \neq j$, $1 < B_{i,j} < A_{i,j}$.

The following line contains an integer q ($1 \leq q \leq 100$), the number of queries.

Each of the next q lines contains an integer n ($1 \leq n \leq 10^{18}$), asking for the answer when the total number of stones is n.

Output

Print q lines, each containing an integer in $[0, 317)$: the sum of the weights of all distribution schemes modulo 317.

Example

standard input	*standard output*
3 0 7 5 5 0 7 4 6 0 0 4 4 3 0 4 3 2 0 4 1 2 10 1000000000000000000	37 303 46 148

Note

Explanation for the example:

For $n = 1$, there are 3 essentially different distribution schemes: giving the only stone to student 1, 2, or 3. The weights of these schemes are 280, 420, and 288 respectively. The total weight is 988, and $988 \bmod 317 = 37$.

For $n = 2$, there are 9 essentially different distribution schemes. For every ordered pair (i, j) with $1 \le i, j \le 3$, the scheme "stone 1 to student i, stone 2 to student j" corresponds to a weight shown in the table below (row i, column j gives the weight):

280	420	504
420	420	160
504	160	288

The sum of all weights is 3156, and $3156 \bmod 317 = 303$.

Problem I. Incident in the Ruins

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

Yellow sand dances and spreads across the desolate land. The dim red light of a broken signal lamp flickers weakly. Two figures, one tall and one short, walk side by side through a concrete ruin that should not exist. The gray sky enveloping them reflects the feelings in their hearts.

"Are you sure it's around here?" asked O, captain of the special operations team DC-1025.

"Should be. The last location of the distress signal from the two of them appeared near here," said T, looking at the screen of his COMPASS device.

DC-1025 originally planned to investigate this ruin as a whole team, but because O and T received an urgent mission, S and K went ahead alone. Unexpectedly, the two juniors sent a distress signal and then vanished. After handling the urgent mission, O and T immediately rushed to the rescue, but what greeted them was only lifeless broken walls and debris.

"Look at this." Sharp-eyed T noticed a glimmer, bent down, and picked up a small black plastic piece from the sand-covered road. On it was engraved a familiar insignia dotted with dark blue squares — undoubtedly DC-1025's dedicated storage chip "Proof". It seemed this chip was most likely left by S and K; but whether it was intentionally placed or accidentally dropped could not be inferred.

T gently wiped the sand off the chip and inserted it into the matching device he carried. After a few seconds, an unsettling prompt popped up on the screen: chip damaged, read error.

O and T barely managed to read some binary information from the undamaged parts of the nearly-wrecked chip. Below is an example of a piece of binary information read, where "." indicates a binary bit that could not be accurately read:

```
+--------+-------------------------------------------------+
|.00.....|0..10011...........00011....1001.................|
|.00.111.|...10100...10101........01110101...10010...01001|
|.00.011.|...10011...00001...11001.......1................1|
|.00.0...|...................10011...10101...11010...10101|
+--------+-------------------------------------------------+
```

In the information format shown above, the left side is an unsigned integer representing the current timestamp in binary, and the right side is a string recording the specific event.

They read a log recording n events. Each timestamp in the log is an integer represented by m binary bits. Since the event content is difficult to recover directly, they plan to start by restoring the time points of the events. If the event recorder itself had no malfunction, the N events should be recorded in chronological order; a later event's timestamp should be no earlier than a previous one.

Now given these n damaged binary timestamps s_1, ..., s_n, each of length m, O and T want to know the earliest possible time of the n-th event in the log, and, under the condition that this event occurs at the earliest possible time, the total number of all possible timestamp restoration schemes.

Input

The first line of the input contains two integers n and m: the number of events and the length of each timestamp, respectively ($2 \le n, m \le 200$).

Each of the next n lines contains a string s_i of length m, consisting only of characters '0', '1', and '.', representing the timestamp of the i-th event, where '.' indicates a damaged binary bit.

Output

If there exists a way to replace every '.' in all s_i with either '0' or '1' so that the n timestamps form a non-decreasing sequence, then output a string of length m consisting only of '0' and '1' and a non-negative integer, separated by a space. The string should be the smallest possible timestamp that s_n can be restored to among all valid restoration schemes, and the integer should be the total number of valid restoration schemes modulo $10^9 + 7$.

If no restoration scheme exists, output a single integer -1.

Examples

standard input	*standard output*
3 4 ..10 .1.0 ...1	0101 1
3 4 ..1. .1.. ...1	0101 4
3 4 111. 011. 0...	-1
3 8 .00.111. .00.011. .00.0...	00010110 2

Note

In the first example, it can be proved that. among all restoration schemes satisfying the chronological order, s_3 can be restored to 0101 at the earliest. When s_3 corresponds to 0101, there is only one restoration scheme, with timestamps 0010, 0100, and 0101, respectively.

In the second example, s_1 can be either 0010 or 0011, and s_2 can be either 0100 or 0101, so there are 4 valid restoration schemes in total.

Problem J. Just Check The Possibility

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

You need to answer several queries.

In each query, given two integers n and m, determine whether it is possible to construct a sequence $a_1, a_2, \ldots, a_n$ consisting of positive integers such that:

- There exists an index j from 1 to n such that $a_j = m$;
- There exists a non-negative integer t satisfying

$$\prod_{i=1}^{n}(a_i + a_{i+1}) = 2^{2t+1},$$

  where $a_{n+1} = a_1$.

If such a sequence can be constructed, output "`YES`"; otherwise, output "`NO`".

Input

The first line contains an integer t ($1 \le t \le 10^6$), the number of queries.

Each of the next t lines contains two integers n and m ($1 \le n \le 2 \cdot 10^6$, $1 \le m \le 2^{62} - 1$), meaning you need to check whether there exists a positive integer sequence of length n containing the value m.

Output

For each query, output one line containing the string "`YES`" or "`NO`", indicating whether such a construction is possible.

Example

standard input	*standard output*
2 3 3 2 1	YES NO

Problem K. King of Monkeys and Two Sets

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	128 mebibytes

Sun Wukong, the King of Monkeys, has two sets A and B of sizes n and m respectively. Their elements are integers in the interval $[0, \ell]$, and both 0 and ℓ are included in each set. Now he wants to find the smallest set C that satisfies the following conditions:

- $0, \ell \in C$.
- After sorting the elements of $A \cap C$ in increasing order, the difference between any two consecutive elements is at most a.
- After sorting the elements of $B \cap C$ in increasing order, the difference between any two consecutive elements is at most b.

Sun Wukong wants you to tell him the size of C, or determine that no such C exists.

Input

The first line of the input contains five integers: n, m, ℓ, a, b ($2 \le n, m \le 10^6$, $1 \le a, b \le \ell \le 10^{18}$).

The second line contains n distinct integers in increasing order, representing the elements of set A. The first number is 0 and the last number is ℓ.

The third line contains m distinct integers in increasing order, representing the elements of set B. The first number is 0 and the last number is ℓ.

Output

Print a single integer: the size of the desired set C. If no such C exists, print -1.

Examples

standard input	*standard output*
10 10 30 8 6 0 2 5 8 11 18 20 21 23 30 0 1 6 12 14 18 21 24 28 30	9
3 4 10 6 4 0 6 10 0 2 7 10	-1

Note

In the first example, a possible minimal set C is $\{0, 6, 8, 11, 12, 18, 23, 24, 30\}$. Hence, the answer is 9.

Problem L. Luxuries and Aesthetics

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

Xiao Lan has n gems, where the brightness of the i-th gem is a_i. Now Xiao Lan wants to divide these gems into several groups, so that each gem will belong to exactly one group.

For a group of gems, if the group contains k gems with brightness values $b_1, b_2, \ldots, b_k$, then Xiao Lan defines the *aesthetic value* of this group as $(\sum_{i=1}^{k} b_i)^2/k^2$. For a grouping scheme, Xiao Lan defines its aesthetic value as the sum of the aesthetic values of all groups.

Now Xiao Lan has q questions. Each question asks: if the number of gems in each group must be within $[\ell, r]$, what is the maximum possible aesthetic value among all grouping schemes that satisfy this requirement? Find this value, or determine that there is no such grouping scheme.

Since the answer can be represented as a rational number a/b, for convenience, you only need to output the result modulo $10^9 + 7$. That is, you need to find an integer c in $[0, 10^9 + 7)$ such that $a \equiv b \cdot c \pmod{10^9 + 7}$. Under the constraints of this problem, it can be proved that such a value c always exists and is unique.

Input

The first line of the input contains two integers, n and q ($1 \le n, q \le 5 \cdot 10^5$), the total number of gems and the number of questions.

The second line contains n non-negative integers, where the i-th integer is the brightness a_i of the i-th gem ($0 \le a_i \le 10^8$).

The next q lines each contain two integers, ℓ and r ($1 \le \ell \le r \le n$), meaning the question: when each group must contain between ℓ and r gems (inclusive), what is the maximum aesthetic value achievable among all admissible grouping schemes?

Output

Print q lines. The i-th line should contain an integer, the answer to the i-th question: the maximum aesthetic value (modulo $10^9 + 7$) when each group's size is restricted to the interval given in the i-th question. If no grouping scheme satisfies the condition, output -1.

Example

standard input	*standard output*
8 4 17 4 12 9 15 3 8 6 1 7 5 8 2 4 3 3	864 62500086 500000431 -1

Note

Explanation for the example:

- For the first question, the optimal grouping is to put each gem into a separate group: $\{17\}, \{4\}, \{12\}, \{9\}, \{15\}, \{3\}, \{8\}, \{6\}$. The aesthetic value is 864, which is the maximum.
- For the second question, the only admissible grouping is $\{17, 4, 12, 9, 15, 3, 8, 6\}$. The aesthetic value reaches its maximum $74^2/8^2$, and $(74^2/8^2) \bmod (10^9 + 7) = 62\,500\,086$.

- For the third question, the optimal grouping is $\{17, 15\}, \{12, 9\}, \{8, 6\}, \{4, 3\}$. The aesthetic value reaches its maximum $(32^2 + 21^2 + 14^2 + 7^2)/2^2$, and modulo $10^9 + 7$ the result is $500\,000\,431$.
- For the fourth question, each group must have size exactly 3, but the total number of gems 8 is not a multiple of 3. Hence some gems would always remain ungrouped, so no admissible grouping exists; output -1.