Problem A. Chords

Imagine a mystical forest where a magic piano with n keys resides. Each key of this piano represents a note, denoted by positive integers from 1 through n. In musical terms, a chord is defined as a set of notes played together in harmony. The type or class of a chord is categorized based on the difference between its root note (the lowest note) and the other notes in the chord.

For instance, the chord class known as *minor triad* includes all chords of the format $\{x, x+3, x+7\}$. Similarly, the chord class *dominant seventh* includes all chords that follow the pattern $\{x, x+4, x+7, x+10\}$.

For each key x on the piano (which produces the note x), you can play it with a_x different strengths, each of which produces a unique sound. As a result, you can produce a_x distinct sounds from that note. More interestingly, for a minor triad with root note x as an example, you can produce $a_x \times a_{x+3} \times a_{x+7}$ unique sounds.

Now, consider a given chord class represented by $\{x+b_1, x+b_2, ..., x+b_m\}$. Your task is to calculate the total number of different sounds you can produce when you are free to choose any root note x.

Input

The first line of the input contains a single integer T $(1 \le T \le 10^3)$, denoting the number of test cases.

The first line of each test case contains two integers n, m $(1 \le m \le n \le 5 \times 10^5)$ separated by space, denoting the number of keys in the piano and the number of notes in the given chord class.

The second line of each test case contains n positive integers $a_1, \ldots, a_n$ $(0 < a_x < 1062125569)$, each representing the number of different strengths that can be applied to the corresponding key x.

The third line of each test case contains m ascending integers $b_1, \ldots, b_m$ $(0 = b_1 < b_2 < \ldots < b_m < n)$, denoting the given chord class.

It is guaranteed that the sum of n over all test cases will not exceed 10^6.

Output

For each test case, output the answer modulo 1062125569 in a single line.

Example

standard input	standard output

1
5 2
3 4 2 5 1
0 2
	28

Problem B. Delivery Robot

A delivery robot is an autonomous machine designed to provide delivery services. On many campuses, these robots can offer significant convenience to students.

A delivery robot typically integrates various assistance systems, such as collision avoidance systems. These systems should enable the robot to automatically avoid both static obstacles and other robots. Due to the non-immediate nature of velocity control, collision avoidance systems typically allocate a time interval Δt. If a collision is predicted to occur within Δt at the current velocity, the system adjusts the robot's velocity accordingly, including its speed and direction.

We assume that the velocity can change instantly at the beginning of Δt but remains constant throughout Δt. In this problem, we set $\Delta t = 1$, meaning you only need to consider the first $\Delta t = 1$ from the initial position, without considering the subsequent process.

We envision the delivery robot as a moving convex polygon on a two-dimensional plane with n vertices, with its initial vertex coordinates given. In addition, there is a stationary convex polygonal obstacle on the plane with m vertices, and its vertex coordinates are also given.

As a developer of the collision avoidance system, you want to conduct q tests, each time setting an initial velocity vector $\mathbf{v}$ for the robot, and the robot will start from the given initial position. According to the principle of the collision avoidance system, if the robot will collide with the obstacle after traveling for $\Delta t = 1$ at the current velocity, the robot's velocity should be changed immediately at the beginning, that is, $\mathbf{v}$ should be modified to $\mathbf{v}'$.

Without an upper limit on speed, it is evident that a suitable $\mathbf{v}'$ must exist to prevent a collision within $\Delta t = 1$. Depending on the situation, different strategies can be used to select $\mathbf{v}'$. The objective of this problem is to choose a $\mathbf{v}'$ that minimizes the magnitude of the difference between $\mathbf{v}$ and $\mathbf{v}'$, i.e. $\mathbf{v}' = \arg\min_{\mathbf{u}} \|\mathbf{v} - \mathbf{u}\|$. The magnitude of a vector $\mathbf{v} = (x, y)$ is defined as $\|\mathbf{v}\| = \sqrt{x^2 + y^2}$.

Input

The first line contains an integer T ($1 \le T \le 20$), indicating the number of test cases.

The first line of each test case contains three integers n, m, q ($3 \le n, m, q \le 1000$), indicating the number of vertices of the delivery robot and the obstacle, respectively.

Each of the next n lines contains two integers x, y ($-10^6 \le x, y \le 10^6$), indicating the coordinates of a vertex of the delivery robot. The coordinates are given in counter-clockwise order.

Each of the next m lines contains two integers x, y ($-10^6 \le x, y \le 10^6$), indicating the coordinates of a vertex of the obstacle. The coordinates are given in counter-clockwise order. It is guaranteed that the delivery robot and the obstacle are strictly disjoint.

Each of the next q lines contains two integers x, y ($-10^6 \le x, y \le 10^6$), indicating the initial velocity vector $\mathbf{v} = (x, y)$ of the delivery robot.

It is guaranteed that $\sum n \le 2000$ and $\sum m \le 2000$ and $\sum q \le 5000$ over all test cases.

Output

For each test, output the **square** of the minimum magnitude of $\mathbf{v} - \mathbf{v}'$, i.e.min $\|\mathbf{v} - \mathbf{v}'\|^2$, in a single line. You should output the answer as an irreducible fraction p/q, where p, q are integers and $q > 0$.

Example

standard input	standard output
`1`	`0/1`
`4 3 3`	`1/2`
`-1 -1`	`18/5`
`-2 -1`	
`-2 -2`	
`-1 -2`	
`0 1`	
`1 0`	
`2 2`	
`1 1`	
`2 2`	
`3 3`	

Note

You can use `__int128` in your `C++` code.

Problem C. Congruences

Your task is to solve a system of m congruences, each one represented in the following format:

$$x^{p_i} \equiv q_i \pmod{n} \quad (i = 1, 2, ..., m)$$

Here, p_i refers to **pairwise distinct** prime numbers, n is the product of all p_i (i.e., $n = \prod_{i=1}^{m} p_i$), and q_i is an integer within the range of $[0, n)$.

The goal is to find the **smallest positive** integer x that fulfills all given congruences. If there is no solution exists for the given system of congruences, output -1.

Input

The first line contains a positive integer T ($1 \le T \le 10^4$), indicating the number of test cases.

For each test case, the first line contains a positive integer m, and each of the following m lines contains two integers p_i and q_i ($2 \le p_i \le 10^{18}$, $0 \le q_i < n$). It is guaranteed that $n = \prod_{i=1}^{m} p_i \le 10^{18}$.

Output

For each test case, output a single **positive** integer x representing the answer or output -1.

Example

standard input	standard output

3
2
3 5
2 1
1
13 3
2
3 4
2 4
	5
	3
	4

Note

You can use `__int128` in your `C++` code.

Problem D. GG and YY's Binary Number

GG and YY are honing their skills with binary numbers. They have a set V comprising n binary numbers: $V = \{v_1, v_2, \ldots, v_n\}$. Additionally, they have m queries determined by intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_m, r_m]$. For each query interval $[l_i, r_i]$, they aim to compute the count of unique subsets U taken from V such that the xor sum of the elements of U lies within $[l_i, r_i]$. The results should be provided modulo $10^9 + 7$.

Specifically, for each $[l_i, r_i]$, compute:

$$\left(\sum_{j \in [l_i, r_i]} \sum_{\{u_1, u_2, \ldots, u_k\} \subseteq V} [u_1 \oplus u_2 \oplus \ldots \oplus u_k = j] \right) \bmod (10^9 + 7)$$

where $\oplus$ represents the xor operation and square brackets $[\cdot]$ denote the Iverson bracket. Please note that the xor sum of an empty set is 0.

In the Iverson bracket notation, if the condition inside the bracket is true, the value is 1; otherwise, it is 0. For instance, $[u_1 \oplus u_2 \oplus \ldots \oplus u_k = j]$ is 1 if the xor sum of $u_1, u_2, \ldots, u_k$ equals j, and 0 otherwise.

Input

The first line contains one integer t ($1 \leq t \leq 100$), indicating the number of test cases.

The first line of each case contains two integers n and m ($1 \leq n, m \leq 250$), indicating the size of the sets V and W.

The following line contains n binary integers $v_1, \ldots, v_n$ ($0 \leq v_i < 2^{250}$), indicating the binary integers in V.

The following m lines present the m queries. The i-th line continas two binary integers l_i, r_i ($0 \leq l_i \leq r_i < 2^{250}$), indicating the query interval.

It is guaranteed that there will be no leading zeros in the binary numbers, with the exception of 0. For instance, the binary number 110 corresponds to the decimal number 6.

Output

For each test case, output m integers in m lines, indicating the results of the queries.

Example

standard input	standard output

1
4 2
100 0 1 101
1 10
10 100
	4
	4

Problem E. 0 vs 1

Two players named Zero and One are playing a strategic game with a string of characters consisting of only `0`s and `1`s.

The rules of the game are as follows:

- The players take turns removing a single character from either the left or the right end of the string, starting with the player named Zero.
- If Zero picks a `1`, he lose the game. Similarly, if One picks a `0`, he loses the game.
- If all characters in the string are removed and no one has lost, the game ends in a draw.

If both players are playing optimally, your task is to determine who will win the game, or whether the game will end in a draw.

Input

The first line contains a single integer T $(1 \leq T \leq 150)$, denoting the number of test cases.

The first line of each test case consists of an integer n $(1 \leq n \leq 10^5)$, denoting the length of the string.

The second line contains a string of length n consisting of only `0`s and `1`s, denoting the initial string of the game.

It is guaranteed that there are no more than 50 test cases with $n > 100$.

Output

For each test case, output a integer in a single line. If One will win the game, output 1. If Zero will win the game, output 0. If the game will end in a draw, output -1.

Example

standard input	standard output
2 3 110 5 01010	1 -1

Note

In the first test case, Zero can only pick up the character 0 from the right in his first turn. Then, it is One's turn to pick up a character. When it comes back to Zero's turn, he can only pick up the character 1, so Zero loses.

Problem F. Nested String

Given two strings T_1 and T_2, a string X is T-nested if and only if X can be represented as the string obtained by inserting T_2 at a position in T_1. For example, if $T_1 =$ `abcd` and $T_2 =$ `ef`, then `efabcd`, `aefbcd` and `abcdef` are all T-nested strings.

Given strings S, T_1, and T_2, your task is to compute the count of distinct T-nested substrings in S. Two nested substrings are considered distinct if they either occur at different positions in S, or **the insert positions of T_2 in T_1 are different**. A substring of S means a continuous sequence of characters from string S.

Input

Ensure to use cin/cout and disable synchronization with stdio to avoid unexpected TLE verdict.

The first line contains a single integer T ($1 \le T \le 20$), denoting the number of test cases.

The first line of each test case contains two strings T_1 and T_2 ($|T_1| + |T_2| \le |S|$) separated by a single space.

The second line of each test case contains a single string S ($|S| \le 10^7$).

It is guaranteed that S, T_1, and T_2 all consist only of lowercase letters and $\sum |S| \le 2 \times 10^7$. Here, $|S|$ means the length of string S.

Output

For each test case, output an integer in a single line representing the number of distinct T-nested substrings in S.

Example

standard input	standard output
3	6
abab ab	5
abababab	5
ab a	
aaabbaabaa	
aba ab	
ababaabbabaab	

Note

In the first test case, the 6 T-nested substrings are (the substring is underlined and the part from T_2 is highlighted in bold):

1. $\underline{\textbf{ab}\text{abab}}\text{ab}$
2. $\underline{\text{ab}\textbf{ab}\text{ab}}\text{ab}$
3. $\underline{\text{abab}\textbf{ab}}\text{ab}$
4. $\text{ab}\underline{\textbf{ab}\text{abab}}$
5. $\text{ab}\underline{\text{ab}\textbf{ab}\text{ab}}$
6. $\text{ab}\underline{\text{abab}\textbf{ab}}$

Problem G. Solubility

In the thriving city, various industries are fueled by the extensive utilization of different liquids. The metropolis houses n unique types of liquids, each with specific characteristics and applications ranging from energy to pharmaceuticals.

The Center for Chemical and Physical Combinations (CCPC) plays a pivotal role in researching and categorizing these liquids. A crucial part of their research focuses on understanding the miscibility relations among the liquids. They have identified m pairs of miscibility relations, each pair representing two types of liquids that are miscible (mutually soluble).

However, miscibility is not always a straightforward property. It exhibits a transitive relation, meaning if liquid type A is miscible with type B, and type B is miscible with type C, then type A will also be miscible with type C.

Whether developing a new kind of fuel or a groundbreaking medical serum, the right combination of liquids is paramount. An incorrect mixture could lead to immiscible components, which could be ineffective or even dangerous. Therefore, if it is not possible to deduce that two types of liquids are miscible based on the given miscibility relations and their transitivity, then those two types of liquids are considered immiscible.

The industries often require the creation of substances involving a specific set of k types of liquids, all of which must be miscible. Your role as a computational scientist at CCPC is to develop an algorithm that will efficiently determine if a given set of k types of liquids are miscible. Specifically, for any two distinct types of liquids within this set of k types, if they are all miscible with each other, then the k types of liquids are considered to be miscible.

Input

The first line contains an integer T ($1 \le T \le 20$), representing the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n, m \le 10^5$), representing the total number of liquids and the number of solubility relations, respectively.

Each of the next m lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), where u and v represent the types of two liquids that are mutually soluble.

The following line contains a single integer k ($1 \le k \le n$), indicating the number of liquids being considered.

The last line of each test case includes k different integers within $[1, n]$, each denoting a type of liquid.

Output

For each test case, output a single line containing either `YES` if all k types of liquids are mutually soluble, or `NO` if they are not.

Example

standard input	standard output
3	YES
3 2	NO
1 2	YES
1 3	
3	
1 2 3	
4 2	
1 2	
3 4	
3	
1 2 4	
6 4	
1 2	
3 4	
5 6	
1 5	
2	
1 6	

Problem H. Expectation of Rank

Let p be a prime number and $\mathbb{F}_p$ be the finite field with order p. Suppose A is a square matrix of order n and each of its entry is a random variable that uniformly distributed on $\mathbb{F}_p$. Please calculate the expectation of the rank of A, i.e., $\mathbb{E}[\text{rank}(A)]$.

In mathematics, a finite field, also known as a Galois field, is a set that contains a finite number of elements. These elements follow the operations of multiplication, addition, subtraction, and division, all of which satisfy the basic rules of arithmetic. The most common examples of finite fields are given by the integers mod p when p is a prime number.

When we say $\mathbb{F}_p$ is a finite field with order p, it means that $\mathbb{F}_p$ contains exactly p distinct elements, with p being a prime number. The elements of $\mathbb{F}_p$ are the integers $0, 1, 2, ..., p-1$, and the operations of the field are performed modulo p. For instance, if $p = 5$, then $\mathbb{F}_5$ is the set $\{0, 1, 2, 3, 4\}$, and in this field, $2 + 3 = 0$ and $4 \times 4 = 1$.

Input

The first line of the input contains an integer T ($1 \le T \le 50$), denoting the number of test cases.

Each of the following T lines contains two integers n, p ($1 \le n \le 5000$, $2 \le p \le 10^9$), denoting the order of the square matrix A and the prime number p.

It's guaranteed that the sum of n in all test cases will not exceed 5000.

Output

For each test case, output the expectation of the rank of A. You should output the answers modulo 10^9+7. That is, if the answer is $\frac{P}{Q}$, you should output $P \cdot Q^{-1} \bmod 10^9 + 7$, where Q^{-1} denotes the multiplicative inverse of Q modulo $10^9 + 7$. It can be proved that the answer can always be expressed in this form.

Example

standard input	standard output

2
2 2
2 3
	812500007
	802469143

Note

The rank of the matrix

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

in $\mathbb{F}_3$ is 1.

Problem I. Diagonal Fancy

Given a matrix A with n rows and m columns, your objective is to compute the total number of continuous sub-square matrices B that are **diagonal fancy**.

A square matrix B is designated as diagonal fancy if it satisfies the subsequent criteria:

- For any indices i_1, j_1, i_2, and j_2, if $i_1 - j_1 = i_2 - j_2$, then $B_{i_1,j_1} = B_{i_2,j_2}$.
- For any indices i_1, j_1, i_2, and j_2, if $i_1 - j_1 \neq i_2 - j_2$, then $B_{i_1,j_1} \neq B_{i_2,j_2}$.

Here, $B_{i,j}$ signifies the element located at the i-th row and j-th column of matrix B.

A continuous sub-square matrix from matrix A with n rows and n columns is defined as the matrix derived from A by selecting continuous n rows and continuous n columns.

Input

Ensure to use cin/cout and disable synchronization with stdio to avoid unexpected TLE verdict.

The input consists of multiple test cases. The first line of the input contains an integer T ($1 \le T \le 100$), which represents the number of test cases.

The first line of each test case contains two integers n and m ($1 \le n, m \le 1000$), representing the number of rows and columns in the matrix A.

Each of the next n lines contains m space-separated integers $A_{i,1}, A_{i,2}, \ldots, A_{i,m}$ ($1 \le A_{i,j} \le n \times m$), representing the elements of the matrix A for that particular test case.

It is guaranteed that $\sum n \times m \le 10^7$ over all test cases.

Output

For each test case, output a single integer in a single line, which represents the count of continuous sub-square matrices in matrix A that are diagonal fancy.

Example

standard input	standard output
3	5
2 2	4
1 2	14
3 1	
2 2	
1 2	
2 1	
3 3	
1 2 3	
4 1 2	
5 4 1	

Note

In the first test case, there are 5 diagonal fancy subsquares in total. They are listed in bold below.

$$\begin{matrix} \mathbf{1} & 2 \\ 3 & 1 \end{matrix} \qquad \begin{matrix} 1 & \mathbf{2} \\ 3 & 1 \end{matrix} \qquad \begin{matrix} 1 & 2 \\ \mathbf{3} & 1 \end{matrix} \qquad \begin{matrix} 1 & 2 \\ 3 & \mathbf{1} \end{matrix} \qquad \begin{matrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{1} \end{matrix}$$

Problem J. Rikka with Square Numbers

Rikka struggles with math. Observing this, Yuta decides to help her improve by assigning some math practice tasks. Here is one such task.

The task requires transforming one integer, a, into another, b, where $a \neq b$. In a single operation, Rikka can either add a positive square number to a, or subtract a positive square number from a. The goal is to convert a into b by executing the least number of operations.

A square number is an integer that is the product of some integer with itself. For example, 1, 4, 9, and 16 are square numbers, as they are the squares of 1, 2, 3, and 4, respectively.

The problem might seem complex for Rikka. Can you assist Rikka in solving this?

Input

The first line contains one integer T $(1 \leq T \leq 10^3)$, indicating the number of test cases.

For each test case, the only line contains two integers a, b $(1 \leq a, b \leq 10^9$, $a \neq b)$.

Output

For each test case, output a single line containing one integer, indicating the minimum number of operations required.

Example

standard input	standard output

2
1 4
5 1
	2
	1

Note

Consider the first sample where Rikka can add $2^2 = 4$ to a, and then subtract $1^2 = 1$ from a. After executing two operations, a becomes 4.