

Problem A

Spiral Game

Max no. of test cases: 40
Time limit: 1 second

A Spiral game is a board game in which players take turn to move along a path of n squares (numbered from 1 to n) on the game board. Some squares are designated as “must stops”, meaning players must stop on it, while other squares can simply pass through without actually stopping. The game is played by m players, numbered from 1 to m . Each player takes turn to roll a dice to determine the number of steps (between 1 and 6) to take. But if the steps taken pass through a “must stop” square, and not stopping on that square, then no steps can be taken. That player must wait for the next turn to roll the dice and hope to roll a number that can stop on that “must stop” square. First player to stop at the last square (square n) wins the game.

For example, in the following game, there are 10 squares with squares 5 and 9 as “must stops”.

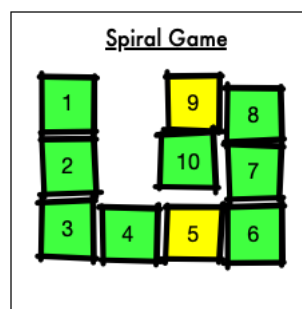


Figure 1: A sample Spiral Game board

Then a game between two players may go as follows:

Table 1: A simple spiral game between two players.

Dice Roll Sequence	-	1	2	3	4	5	6	7	8	9	10	11	12
Steps	-	4	5	6	2	1	3	4	2	3	1	2	1
player 1 on square	0	4	4	4	4	5	5	9	9	9	9	9	10
player 2 on square	0	0	5	5	7	7	7	7	9	9	10	10	10

Note that in this game, although player 1 rolled 4 and then a 6, he must remain at square 4 because moving 6 squares will pass through “must stop” square 5. All players start at square 0 at the beginning of the game.

Please simulate this game and determine the winning player.

Input File Format

The input file contains multiple test cases. The first line of the input consists of an integer indicating the number of test cases.

Each test case begins with a line containing two integers n and m , representing the number of squares in this game and the number of players ($10 \leq n \leq 5,000$, $2 \leq m \leq 150$). The next line contains a list of “must stop” squares in ascending order; the trailing (last) number is 0 to indicate end of the list (the number of “must stops” is between 0 and n). The next line is a sequence of dice rolls, which are numbers between 1 and 6; the trailing (last) number is 0 to indicate end of the game.

Output Format

For each test case, output the winning player number. If no one wins the game, output 0.

Sample Input

```
3
10 2
5 9 0
4 5 6 2 1 3 4 2 3 1 2 1 1 0
10 4
1 0
2 3 4 5 6 2 3 1 2 3 4 6 2 3 4 3 1 2 3 0
10 2
0
1 2 3 4 0
```

Output for the Sample Input

```
2
4
0
```

Problem B

Valid Arithmetic Expressions

Max no. of test cases: 20

Time limit: 1 second

In many modern programming languages, one may consider those valid arithmetic expressions, without parentheses, that are made up of the digits $0,1,2,\dots,9$ and the binary operators $+, -, *, /$. Given the alphabet set $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /\}$, a project leader of a software company, Dr. H, who is a mathematician and an excellent programmer, formally defined the valid arithmetic expression AE using the following rules:

Rule 1: Zero or any positive integers is in AE. Note that negative integers are not allowed in AE.

Rule 2: If x and y are in AE, then so are:

1. $x + y$
2. $x - y$
3. $x * y$
4. x / y

Following the above rules, $1 + 2$ and $3 + 4 * 5$ are valid arithmetic expressions; $8 + /9$ and $5 - *6$ are not. Note that $1 + 2 * 3 = 7$ because there is a precedence of operations: Multiplication and division are performed before addition. Operations at the same level are performed in the order of occurrences as the expression is scanned from left to right. For a positive integer n , let $e(n)$ be the total number of these valid arithmetic expressions that are composed by n symbols. Then, $e(1) = 10$ because of the arithmetic expressions of one symbol are the 10 digits. Next, $e(2) = 90$ for the expressions $10, 11, \dots, 99$.

Your task is to write a computer program to determine, $e(n)$, the total number of valid arithmetic expressions that are composed by n symbols. If the result is larger than or equal to $10^9 + 7$, you should output the value modulo $10^9 + 7$, that is, the remainder obtained using the actual value divided by $10^9 + 7$.

Technical Specification

1. $1 \leq n \leq 100000000$.
2. Unnecessary leading zeros, e.g., $\underbrace{0\dots0}_k 0$, $\underbrace{0\dots0}_k 1$, $\underbrace{0\dots0}_k 2, \dots$, and $\underbrace{0\dots0}_k 9$, where $k \geq 1$, are not allowed.
3. Divided by zero, $/0$, is not allowed.
4. There are no unnecessary leading plus or minus signs.

Input Format

The first line of the input file contains an integer indicating the number of test cases to follow. Each test case has the following format: a positive integer n denoting valid arithmetic expressions that are composed by n symbols.

Output Format

The total number of valid arithmetic expressions that are composed by n symbols. If the result is larger than or equal to $10^9 + 7$, you should output the value modulo $10^9 + 7$, that is, the remainder obtained using the actual value divided by $10^9 + 7$.

Sample Input

```
2
1
2
```

Sample Output for the Sample Input

```
10
90
```

Problem C

Enumerating Trees

Max no. of test cases: 11
Time limit: 1 second

An n -node tree T has n nodes, one of which is designated as the root. Each node of T has several (possibly zero or one) children. The children of a node are ordered in the sense that there is the first child, the second child, etc. We want to enumerate all n -node trees.

Denote the root of T by r . The NCPC representation $\text{NCPC}(T)$ of T is the output of $\text{DFS}(r)$, where the pseudocode of $\text{DFS}(u)$ is as follows for any node u of T :

```

1: for  $i = 1$  to the number of children of  $u$  do
2:   Set  $u_i$  to be the  $i$ th child of  $u$ ;
3:   Output  $i$ ;
4:    $\text{DFS}(u_i)$ ;
5: end for
6: if  $u \neq r$  then
7:   Output 0;
8: end if

```

It is not hard to see that different n -node trees do not share a common NCPC representation.

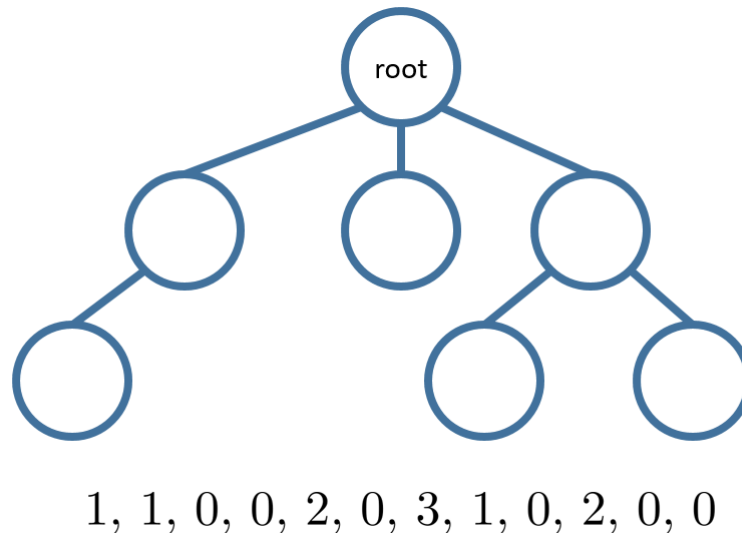


Figure 1: A tree and its NCPC representation, with commas added only for visual clarity

Figure 1 illustrates $\text{NCPC}(T)$ for a sample tree T , where the j th child of a node is shown on the left of the $(j + 1)$ st, for all j . Figure 2 illustrates all five 4-node trees and their NCPC representations.

Please find $\text{NCPC}(T)$ for each n -node tree T .

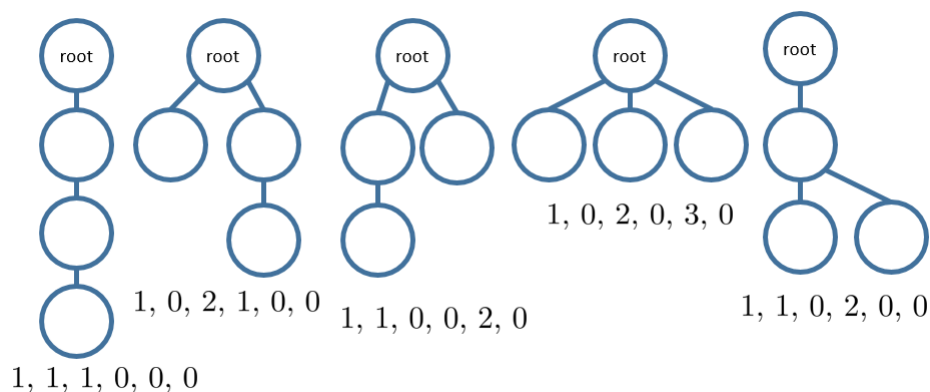


Figure 2: All five 4-node trees and their NCPC representations, with commas added only for visual clarity

Input File Format

The first line is the number of test cases, which is at most 11. Each test case consists of an integer $2 \leq n \leq 12$.

Output Format

Please do the following for each test case: For each n -node tree T , output $\text{NCPC}(T)$ in one line. The NCPC representations should be in the lexicographic order. For example, “1, 0, 2, 0, 3, 0” should appear before “1, 0, 2, 1, 0, 0,” and “1, 1, 0, 0, 2, 0” should appear before “1, 1, 0, 2, 0, 0” (where underlines and commas are added only for visual clarity). Separate two consecutive numbers in $\text{NCPC}(T)$ by a space.

Sample Input

2
4
3

Output for the Sample Input

1 0 2 0 3 0
1 0 2 1 0 0
1 1 0 0 2 0
1 1 0 2 0 0
1 1 1 0 0 0
1 0 2 0
1 1 0 0

Problem D

Jacobi Symbol

Max no. of test cases: 10
Time limit: 1 second

Finite multiplicative groups play an important role in modern cryptography. For example, the RSA cryptosystem is based on a finite multiplicative group generated by integers modulo n , where n is the product of two large primes.

The finite multiplicative group generated by integers modulo n is usually denoted as \mathbf{Z}_n^* . The multiplication of any two of elements x and y in \mathbf{Z}_n^* is defined as

$$x * y = (x \times y) \bmod n,$$

where “ $x * y$ ” is the multiplication operation in \mathbf{Z}_n^* , and “ $x \times y$ ” is the multiplication of integers, and “ $\alpha \bmod n$ ” is the remainder of dividing α by n . For example, in \mathbf{Z}_7^* ,

$$2 * 3 = 6 \bmod 7 = 6, \quad 3 * 4 = 12 \bmod 7 = 5.$$

For simplicity, the multiplication operator “ $*$ ” can be omitted. We use x^k to represent $\underbrace{x * x * \dots * x}_k$.

The multiplicative group \mathbf{Z}_n^* has interesting properties. For example, not all integers \mathbf{Z}_n^* has “square roots”. The squares of all elements in \mathbf{Z}_7^* are:

$$\begin{aligned} 1^2 &= 1, & 2^2 &= 4, & 3^2 &= 2, \\ 4^2 &= 2, & 5^2 &= 4, & 6^2 &= 1. \end{aligned}$$

It can be seen that only $a = 1, 2, 4$ in \mathbf{Z}_7^* have square roots.

Let p , $p > 2$, be a prime. If there is an element x in \mathbf{Z}_p^* that satisfies $x^2 = a$, then a is called a *quadratic residue* modulo p , otherwise, it is *quadratic non-residue*. In 1798, a French mathematician Legendre defined a symbol $\left(\frac{a}{p}\right)$ with values 1, -1, 0:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a = kp, \text{ for some integer } k, \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p. \end{cases}$$

The Legendre symbol $\left(\frac{a}{p}\right)$ can also be viewed as a function with two inputs a and p , whose output is 0, -1, or 1. Legendre’s original definition was by means of the explicit formula

$$\left(\frac{a}{p}\right) = a^{(p-1)/2}.$$

It has been proved that these two definitions are equivalent.

In 1837, a German mathematician Jacobi generalize the Legendre symbol, called *Jacobi symbol*. For any integer a and any positive odd integer n , the Jacobi symbol $\left(\frac{a}{n}\right)$ is defined as the product of the Legendre symbols corresponding to the prime factors of n :

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_r}\right)^{e_r},$$

where $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ is the prime factorization of n . For example,

$$\left(\frac{a}{45}\right) = \left(\frac{a}{3}\right)^2 \left(\frac{a}{5}\right), \quad \text{since } 45 = 3^2 5.$$

The Jacobi symbol is useful in computational number theory, especially in primality testing and integer factorization; these in turn are important in cryptography.

The value of $\left(\frac{a}{n}\right)$ can be computed directly from the definition. However, the factorization of n cannot be computed efficiently when the value of n is large. We need an algorithm to compute $\left(\frac{a}{n}\right)$ without factoring n .

The following rules can be used to efficiently compute $\left(\frac{a}{n}\right)$ without factoring n .

1. $\left(\frac{a}{n}\right) = \begin{cases} 0 & \text{if } \gcd(a, n) \neq 1, \\ \pm 1 & \text{if } \gcd(a, n) = 1. \end{cases}$

Where $\gcd(a, n)$ is the greatest common divisor of a and n .

2. If n is odd then $\left(\frac{1}{n}\right) = \left(\frac{n}{1}\right) = 1$.

3. If n is odd then $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}} = \begin{cases} 1 & \text{if } n \bmod 8 = 1 \text{ or } 7, \\ -1 & \text{if } n \bmod 8 = 3 \text{ or } 5. \end{cases}$

4. If n is odd and $a_1 \bmod n = a_2 \bmod n$ then $\left(\frac{a_1}{n}\right) = \left(\frac{a_2}{n}\right)$.

5. If n is odd then $\left(\frac{a_1 a_2}{n}\right) = \left(\frac{a_1}{n}\right) \left(\frac{a_2}{n}\right)$

For example, if $a = 2^k \cdot t$ then $\left(\frac{a}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$

6. (The law of quadratic reciprocity) If a and n are both odd then

$$\left(\frac{a}{n}\right) \left(\frac{n}{a}\right) = (-1)^{\frac{a^2-1}{2} \frac{n^2-1}{2}} = \begin{cases} -1 & \text{if } a \bmod 4 = n \bmod 4 = 3, \\ 1 & \text{otherwise.} \end{cases}$$

Based on the above rules, write a program to compute the value of $\left(\frac{a}{n}\right)$.

Input File Format

Each test case consists of a modulus n , followed by some a 's. The value of n is odd and $1 < n < 2^{62}$. The last a is followed by a 0, indicating the end of a . For example,

```
15 3 10 6 11 0
```

means $n = 15$, and $a = 4, 2, 5, 11$.

The input data for each test case is given in one line, and the last line of the input file contains only an integer 0 to indicate the end of the input.

Output Format

For each test case, print the value of $\left(\frac{a}{n}\right)$ for each a in one line:

```
1 1 0 -1
```

Print exactly one space in front of each output data, including the first one, and no other spaces.

Sample Input

```
15 4 2 5 11 0
137 25 3 57 23 7 0
2339 56 8 23 34 0
0
```

Output for the Sample Input

```
1 1 0 -1
1 -1 -1 -1 1
1 -1 -1 1
```

Problem E

Josephus Problem

Max no. of test cases: 50
Time limit: 3 seconds

The Josephus problem is a well-known problem in computer science. It involves a group of n soldiers who are about to be captured by the enemy. Instead of surrendering, the soldiers decide to form a circle and execute every k^{th} soldier until only one survivor remains. The soldiers are numbered from 1 to n .

Let's consider an example with $n = 7$ and $k = 3$. The soldiers are numbered 1, 2, 3, 4, 5, 6, and 7. The first soldier executed is number 3, and the second one is number 6. The remaining soldiers are numbers 7, 1, 2, 4, and 5. The 3rd soldier (number 2) is executed next, and the circle continues, so the next soldier executed is number 7. Finally, number 4 is the last survivor.

But now, the variable k has a bit of difference and is not a constant anymore. Given k_0, k_1, \dots, k_{m-1} , the i^{th} access of variable k is $k_{(i-1) \pmod m}$.

Take $n = 7$, $k_0 = 2$, and $k_1 = 3$ as an example. The first soldier executed is number 2 ($k_0 = 2$), and the second one is number 5 ($k_1 = 3$). The remaining soldiers are numbers 6, 7, 1, 3, and 4. The 3rd soldier (number 7) ($k_0 = 2$) is executed next, and the circle continues, so the next soldier executed is number 4. Finally, number 6 is the last survivor.

Your task is to write a program that determines the number of the last survivor.

Input File Format

There are at most 50 test cases. Each test case consists of several integers separated by a space and written on a single line. They are n , m , and m integers k_i for $0 \leq i < m$. The constraints are as follows: $1 \leq n \leq 10^{18}$, $1 \leq m \leq 5$ and $1 \leq k_i \leq 10^5$.

Output Format

For each test case, the program should output the number of the last survivor on a separate line.

Sample Input

7 1 1
7 1 2
7 1 3
7 1 4
7 2 2 2
7 2 2 3
7 2 2 4
7 2 2 5

Output for the Sample Input

7
7
4
2
7
6
5
5

Problem F

Roundtrip

Max no. of test cases: 10
Time limit: 1 second

A special robot is tasked with cleaning a rectangular factory floor. This factory floor is divided into $m \times n$ sections. The robot is programmed to move either horizontally or vertically between sections and is incapable of moving diagonally.

Within the factory, some sections have machinery placed on them, so the robot cannot enter nor pass through these sections. Certain sections are designated as “dirty” due to previous work done in those sections; thus need cleaning. As the robot traverses the factory floor, it will clean every section it enters, irrespective of the section’s dirty status.

Charging stations are placed at the grid positions $(0, 0)$ and $(m-1, n-1)$. Upon completing a full charge, the robot has the capacity to clean a total of $(m+n-1)$ sections, inclusive of the sections containing the charging station. If the robot runs out of power, it will stop moving. Every morning, the robot initiates its cleaning routine from the $(0, 0)$ section and determines its cleaning path based on the provided map detailing the factory floor’s conditions. This factory floor map is represented as a $m \times n$ grid. Sections occupied by machinery are denoted by -1, dirty sections are denoted by 1, and all other sections are marked as 0.

The robot begins at the section $(0, 0)$, move toward the charging station at section $(m-1, n-1)$ to recharge and subsequently returns to section $(0, 0)$ to complete the day’s cleaning. Please write a program to determine the maximum number of dirty sections that the robot can clean after this roundtrip of floor cleaning?



		<u>Column</u>		
		0	1	2
<u>R</u> <u>O</u> <u>W</u>	0	0 	1	-1
	1	1	0	-1
	2	1	1	1 

Figure 1: Sample floor plan

Input File Format

First line of input has one integer, indicating the number of test cases. For each test case, the first line contains two integers n and m , $3 \leq n \leq 500$, $3 \leq m \leq 500$, which denote that the floor is divided into n rows by m columns sections. The next n lines each has m integers, representing status of each section. Note only -1, 0 or 1 are used.

Output Format

For each test case, output an integer on a single line, indicating the maximum number of dirty sections that the robot can clean in one roundtrip.

Sample Input

```
2
3 3
0 1 -1
1 0 -1
1 1 1
7 5
1 1 1 0 1
0 0 1 -1 0
1 0 1 0 0
0 0 -1 0 0
0 1 0 0 0
1 -1 1 0 0
0 -1 1 -1 1
```

Output for the Sample Input

```
5
9
```

Problem G

TWIN

Max no. of test cases: 10
Time limit: 3 seconds

You get a request to implement a next-generation random number generator with code name TWIN. The idea of TWIN is as follows. Suppose that you are given a list $A = [(x_i, y_i) \mid 1 \leq i \leq n]$ that contains n pairs of 32-bit nonnegative integers. For a pair, we will use x -value to denote its first coordinate value and y -value for the second. When we compare the order of two pairs, there are two modes: the x -major and the y -major in the lexicographic way. In x -major, you first compare their x -values; if they are equal, then you compare their y -values. In y -major, you first compare the y -values and then the x -values if required. During the process of TWIN, the first step is to extract pairs of A that lie roughly within the second and third quartiles according to x -major, and then find the median by y -major in this extracted set. Specifically, if you sort pairs of A by x -major, you get a totally ordered list. In this list, remove the first and last $\lfloor n/4 \rfloor$ pairs, and there are $m = n - 2 \lfloor n/4 \rfloor$ pairs left. Next, find the median in the remaining set by y -major. You know that, when m is odd, its median is unique. However, when m is even, generally speaking, there are two different candidates for the median. In this case, always get the smaller one according to y -major. We will use $f(A)$ to denote the resultant pair.

In addition to implement TWIN, you need an extra function

$$h(z) = ((a_1 \cdot z + b_1) \bmod 2^{32}, (a_2 \cdot z + b_2) \bmod 2^{32}),$$

for some integers a_1, b_1, a_2, b_2 . Now, set $A_0 = A$ and we iterate through A_k for $k \geq 0$. Given A_k , let s_k be the x -value of $f(A_k)$. Remove $f(A_k)$ from A_k and insert $h(s_k)$ into A_k ; call the result as A'_k . Set s'_k to be the x -value of $f(A'_k)$. Insert $h(s'_k)$ into A'_k , and you get A_{k+1} . Notice that size of A_{k+1} is one more than the size of A_k . For a given k , please evaluate A_k and s_k from A_0 .

For example, set $A_0 = [(0, 4), (1, 4), (2, 3), (3, 2), (3, 2)]$. Let $h(z) = ((z + 1) \bmod 2^{32}, (z + 2) \bmod 2^{32})$. Notice that all pairs in A_0 are already sorted in x -major and duplicate pairs are allowed here. Since $n = 5$, the truncation length is equal to $\lfloor 5/4 \rfloor = 1$. Next, we extract the middle part $[(1, 4), (2, 3), (3, 2)]$ from A_0 , and then sort them in y -major and finally get $[(3, 2), (2, 3), (1, 4)]$. Notice that its median is $(2, 3)$ and this is the required $f(A_0)$. Thus, $s_0 = 2$, which comes from the x -value of $(2, 3)$. Now remove $f(A_0)$, which is $(2, 3)$, from A_0 and insert $h(s_0) = h(2) = (3, 4)$ back to it, and we get $A'_0 = [(0, 4), (1, 4), (3, 2), (3, 2), (3, 4)]$. You can see further that $f(A'_0) = (3, 2)$, and thus $s'_0 = 3$. Hence, $A_1 = [(0, 4), (1, 4), (3, 2), (3, 2), (3, 4), (4, 5)]$.

Input File Format

The first line gives you the number of test cases, which is a number that is at most 10. For each test case, it starts with a line that specifies space-delimited 32-bit unsigned integers

n, k, a_1, b_1, a_2, b_2 , where $1 \leq n \leq 10$ and $1 \leq k \leq 10^6$. In the next n lines, the i th line gives you x_i and y_i , respectively. The next test case follows immediately after the previous one. Notice that the sum of all k 's in the test cases is less than 2×10^6 .

Output Format

For each test case, output its s_k in one line.

Sample Input

```
2
5 1 1 1 1 2
0 4
1 4
2 3
3 2
3 2
5 2 1 1 1 2
0 4
1 4
2 3
3 2
3 2
```

Output for the Sample Input

```
3
1
```


Problem H

Kindergarten

Max no. of test cases: 10
Time limit: 1 second

In a kindergarten there are n children, numbered from 1 to n . Every two children are either mutual friends or unfamiliar with each other. You are a teacher in this kindergarten. One day there is a school trip. You ask the children to line up and wait for the buses. Every child sees that **all his/her friends standing in front of him/her are also mutual friends with each other** so everyone pushes forward. It's dangerous to do so, and you decide to arrange the children into different lines so that children in a line are unfamiliar with each other. Given that there are m pairs of mutual friends, and assuming that children are numbered as the lined-up order before your arrangement, please write a program to determine the least number of lines needed.

For example, assume that there are 6 children, lined up as 1, 2, 3, 4, 5, 6 with 6 being the front, and there are 9 pairs of mutual friends: (1, 3), (1, 5), (2, 4), (2, 5), (2, 6), (3, 5), (4, 5), (4, 6), and (5, 6). We need 4 lines so that children in each line are unfamiliar with each other. One possible arrangement is

- line 1: 1, 2
- line 2: 3, 4
- line 3: 5
- line 4: 6

Input File Format

There are more than one test cases in the input file. The first line of the input is an integer, specifying the number of test cases. For each test case, the first line contains two integers n ($2 \leq n \leq 500$) and m ($2 \leq m \leq 2 \times 10^4$), which are the number of children and the number of pairs that are mutual friends. Two consecutive numbers in a line are separated by a space.

Output Format

For each test case, output an integer, which is the least number of lines needed as requested.

Sample Input

1
6 9
1 3
1 5
2 4
2 5
2 6
3 5
4 5
5 6
6 4

Output for the Sample Input

4

Problem I

Cities

Max no. of test cases: 20
Time limit: 1 second

We want to connect cities into the minimum number of zones. We have n cities and m one-way roads where each road connects two cities. The roads have no cycles, so you can not start from a city and return to it. A zone consists of cities on a road path, i.e., a sequence of roads where the end of a road is the beginning of the next road. Note that the length of a path could be zero, so a city can be a zone by itself. We also assume that a city can only be in one zone. Now given all the roads, please determine the minimum number of zones that include all cities.

Input File Format

There is more than one test case in the input file. The first line has the number of test cases. The first line of a test case is the number of cities (n). The second line of a test case is the number of the roads (m). Each of the following lines has two integers for a road. The first integer is the index of the starting city, and the second integer is the index of the end city.

Output Format

Each line of the output has the minimum number of zones for a test case.

Sample Input

```
2
4
3
0 1
2 1
3 1
5
4
0 1
1 2
2 3
3 4
```

Output for the Sample Input

3
1

Problem J

Collision Simulation

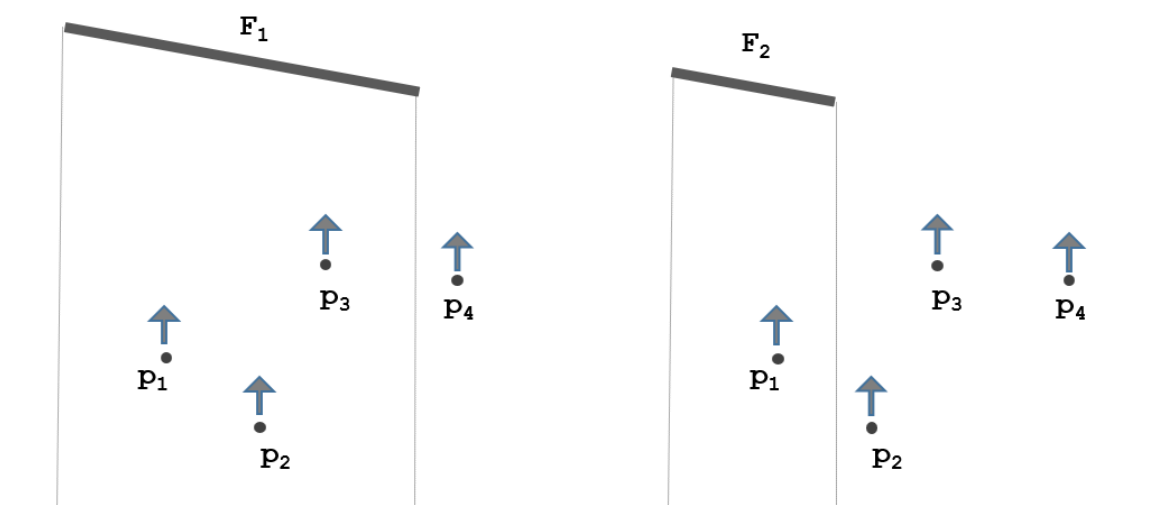
Max no. of test cases: 18

Time limit: 21 seconds

In this problem, you are to simulate the process of firing a set of projectiles against a number of defending fences. The scenario is as follows.

Consider the 2-D plane. There are n machine guns which are represented as points in the plane, and you are given the coordinates of them. In the simulation, each of the machine guns will fire a projectile towards the north direction simultaneously at the same time, and the projectiles will move at the same speed.

On the other hand, there is a defending fence, represented as a line segment, somewhere in the plane. The defending fence is equipped with a protecting shield, which is capable of neutralizing the attack of one projectile. Further than that, the fence will simply be destroyed. The goal of this simulation is to determine the machine gun that can destroy the given fence in the plane, i.e., the machine gun that will hit the fence secondly.



For example, in the above figure, there are 4 machines guns p_1, \dots, p_4 in the plane, and two simulations are performed. In the first one, p_3 will hit F_1 first and will be neutralized. p_1 will hit secondly and destroy F_1 . In the second simulation, only p_1 will hit F_2 , and F_2 will not be destroyed.

Input File Format

The first line has an integer indicating the number of test cases to follow.

For each test case, the first line consists of two integers n and m , the number of machine guns and the number of simulations. Each of the next n lines contains two integers, which are the coordinates of the n machine guns. Then there are m lines, each of which contains four integers x_1, y_1, x_2, y_2 which represent the coordinates of the two endpoints of the fence.

You may assume that

- $1 \leq n \leq 2 \times 10^5$.
- $1 \leq m \leq 2 \times 10^5$.
- No machines are located at the same place, i.e., we have $(x_i, y_i) \neq (x_j, y_j)$ for any two machine guns $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$.
- All the input coordinates are within the range of $\pm 10^8$.
- For any input fence with endpoints $(x_1, y_1), (x_2, y_2)$ and any machine gun (x, y) , we always have $y < \min(y_1, y_2)$.
- The machine guns are numbered from 1 to n according to the order of input.

Output Format

For each simulation of each test case, print the index of the machine gun whose projectile will destroy the fence in a line. If there are multiple possible answers for an simulation, output the possibility with the smallest index. If it is not possible to destroy the fence, output -1 instead.

Sample Input

```
1
4 2
1 2
2 1
3 4
5 4
0 7 4 6
0 7 1 6
```

Output for the Sample Input

```
1
-1
```

Problem K

Forming groups

Max no. of test cases: 10
Time limit: 1 second

As the semester comes to its last month, many courses, including Prof. Miller’s *C++ Programming*, have begun forming final project groups. In this course, N students are enrolled, where N is an even number. The professor aims to create $N/2$ disjoint groups, with each group consisting of exactly two students. Before pairing up, each student is required to submit a strict preference list, which is an ordered list of all other students in the class, indicating their preferences for potential teammates.

The professor seeks to find a *stable matching*, which satisfies the following conditions:

1. Each student is paired up with exactly one other student, forming $N/2$ disjoint groups.
2. No two students, who are not paired together, both prefer each other over their current partners (avoiding a “block pair”).

For example, let’s consider four students in the class with their preference lists:

Table 2: An example of complete stable matching.

	1st	2nd	3rd
A	<u>B</u>	C	D
B	<u>A</u>	C	D
C	<u>D</u>	A	B
D	<u>C</u>	B	A

In this example, a stable matching would be (A, B) and (C, D) , as there are no block pairs. However, in any other matching, such as (A, C) and (B, D) , the pair (A, B) forms a block pair, as both prefer each other over their current partners.

In some cases, a “complete” stable matching may not exist, where every student can be paired up without forming any block pairs. For example:

Table 3: An example of incomplete stable matching.

	1st	2nd	3rd
A	B	C	D
B	C	A	D
C	A	B	D
D	A	B	C

In this example, it is not possible to form a complete stable matching. We can always identify a block pair in any matching:

- $(A, B), (C, D) \rightarrow (B, C)$ is a block pair
- $(A, C), (B, D) \rightarrow (A, B)$ is a block pair
- $(A, D), (B, C) \rightarrow (A, C)$ is a block pair

However, we can still find the maximum number of students that can be paired up stably among themselves. In the provided example, the number of maximum stable matching is 2.

Your task is to write a program that takes N preference lists as input and finds the maximum number of students that form a stable matching, ensuring that no block pairs exist.

Input File Format

The input file contains multiple test cases. The first line of the input consists of an integer indicating the number of test cases.

Each test case begins with a line containing an even integer N ($2 \leq N \leq 100$) representing the number of students in the course.

Following that, there will be N lines, each representing the preference list of a student. Each of these lines contains $N - 1$ space-separated integers, denoting the order of preference of the students. The j -th integer on the i -th line represents the student preferred in the j -th position by student i . The students are represented by integers from 1 to N .

Output Format

For each test case, please print the number of students that form a maximum stable matching.

Sample Input

```
2
4
2 3 4
1 3 4
4 1 2
3 2 1
4
2 3 4
3 1 4
1 2 4
1 2 3
```


Output for the Sample Input

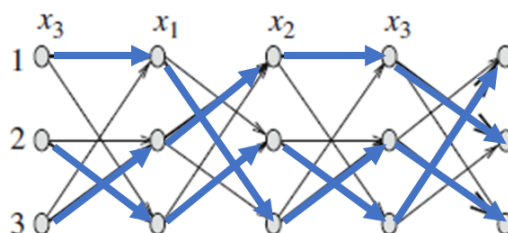
4
2

Problem L

Permutation Branching

Max no. of test cases: 5
Time limit: 1 second

NexCore Processors Corporation (NCPC) is a leading semiconductor company specializing in the design, development, and manufacturing of advanced processors. With a focus on cutting-edge technology and innovation, NCPC delivers high-performance semiconductor solutions that power a wide range of applications. Ted is a team leader of a research group at NCPC in charge of developing efficient computing devices. Recently, he has been studying a method of efficiently evaluating Boolean functions, as the following diagram shows how to evaluate a Boolean function with three input variables $x_1, x_2, x_3 \in \{0, 1\}$. Each of the columns has three nodes indexed from 1 to 3 in increasing order and is labeled with a variable, except the last one. There are two permutations between any two adjacent columns.



As above, between the first two adjacent columns labeled with x_3 and x_1 , the boldface arcs indicate a permutation π_1 with $\pi_1(1) = 1, \pi_1(2) = 3, \pi_1(3) = 2$. The other permutation π_0 is $\pi_0(1) = 3, \pi_0(2) = 2, \pi_0(3) = 1$. If $x_3 = 1$, then it follows the boldface permutation (π_1); else it follows the other permutation (π_0) to reach the next column. Therefore, a set of input values will determine each node in the leftmost column to end up with the nodes in the rightmost column. For example, if $(x_1, x_2, x_3) = (0, 1, 1)$, then it yields an identity permutation $1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3$. If $(x_1, x_2, x_3) = (0, 1, 0)$, then it yields a permutation $1 \rightarrow 3, 2 \rightarrow 1, 3 \rightarrow 2$.

Specifically, we define that a permutation branching diagram P as a $w \times \ell$ array of nodes, where we call w the width and ℓ the length of the diagram, respectively. All ℓ levels (columns) have w nodes, and all nodes at a given level are labeled by the same variable. The above diagram has $w = 3, \ell = 5$. Moreover, at each level (column), the directed edges going to the next level form two injective mappings (permutations) from $\{1, \dots, w\}$ to $\{1, \dots, w\}$. If that level is labeled by a variable x_i , then one of these mappings is given by edges corresponding to tests with $x_i = 1$, and the other to tests with $x_i = 0$. The length of P is the number of levels (columns) in it.

We can view such a diagram as a sequence of instructions $\langle i_j, \pi_1, \pi_0 \rangle$, where $1 \leq j < \ell$, x_{i_j} is the variable tested at the corresponding level j and π_1, π_0 are the two permutations defined on $\{1, 2, \dots, w\}$ corresponding to whether $x_{i_j} = 0$ or 1. I.e., for $b \in \{0, 1\}$, π_b

will be used to reach the next level if $x_{i_j} = b$. A permutation is cyclic if it is composed of a single cycle on all its elements. For example, $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \end{pmatrix}$ is a cyclic permutation, which can be denoted as $1 \rightarrow 3 \rightarrow 5 \rightarrow 4 \rightarrow 2 \rightarrow 1$, or simply $\sigma = (1\ 3\ 5\ 4\ 2)$.

Consider a $w \times \ell$ permuting branching diagram P . Any input vector $x \in \{0, 1\}^n$ yields a permutation $P(x)$ which is the composition of the selected permutations at each level. For a Boolean function f and a cyclic permutation σ , we say that the branching diagram P σ -computes f if for every input x :

$$P(x) = \begin{cases} \sigma & \text{if } f(x) = 1, \\ e & \text{if } f(x) = 0, \end{cases}$$

where e is the identity permutation. For example, the following gives a permuting branching diagram (3 5 2 1 4)-compute the Boolean formula $x_1 \wedge x_2$. Each row with 11 integers follows the format $\langle i_j, \pi_1, \pi_0 \rangle$, where the first integer indicates the test Boolean variable x_{i_j} to determine which permutation to apply, the following 5 integers indicate the permutation π_1 , and the next 5 integers indicate the permutation π_0 . It can be verified that only when $x_1 = x_2 = 1$ can yield the permutation (3 5 2 1 4).

```
2 2 3 4 5 1 1 2 3 4 5
1 3 1 5 2 4 1 2 3 4 5
2 5 1 2 3 4 1 2 3 4 5
1 2 4 1 5 3 1 2 3 4 5
```

Ted knows that any Boolean function can be evaluated with a width-5 permuting branching diagram. Your task is to write a program to help Ted design a width-5 permuting branching diagram for a Boolean function such that the length is as small as possible. We will use the 3-Conjunctive Normal Form (3-CNF) to denote a Boolean function f for this problem. Any n -variable Boolean function f can be written as $C_1 \wedge C_2 \wedge \dots \wedge C_m$ and each clause C_i is in the form $z_1 \vee \dots \vee z_k$, where $1 \leq k \leq 3$ and each z_i can be a Boolean variable or its negation.

Input File Format

We use positive integer i to indicate the i -th Boolean variable x_i and $-i$ for $\neg x_i$. For example, the clause $(x_1 \vee \neg x_3 \vee x_5)$ will be given as a sequence of three integers 1 -3 5 separated by space. The variables are indexed from 1.

The first line in the input file has a positive integer T , which indicates there are T (5) test cases. Then for each case, the first line has two positive integers, n (< 15) and m (< 20), which indicates, respectively, the number of Boolean variables and the number of clauses of a Boolean function. Then in each of the following m lines there is a sequence of at most three distinct integers in $\{1, 2, \dots, n, -1, \dots, -n\}$ indicating a clause. Thus, the conjunction of these m clauses indicates an input Boolean function. Note for those clause with fewer than 3 literals will end with 0 in the input file.

Output Format

For each test case, the first line should contain an integer ℓ and a cyclic permutation σ of $\{1,2,3,4,5\}$, indicating the length of the permuting branching diagram and the diagram σ -computing the corresponding input Boolean function. The answer is not unique. You only need to output any one with $\ell < 4100$. Then, in the k -th line of the following $\ell - 1$ lines, output $i \ \pi_1 \ \pi_0$ in order (exactly 11 positive integers), where i indicates level k is labeled with the variable x_i , and π_1, π_0 are permutations of $\{1, 2, 3, 4, 5\}$ and π_1 will be applied to reach the next level when $x_i = 1$, and π_0 will be applied when $x_i = 0$. The format of π_1, π_0 is $a_1 \ a_2 \ a_3 \ a_4 \ a_5$ indicating a permutation in the form: $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ a_1 & a_2 & a_3 & a_4 & a_5 \end{pmatrix}$, i.e. $\pi_b(i) = a_i$, for $b \in \{0, 1\}, 1 \leq i \leq 5$.

Sample Input

```
3
1 1
-1 0
2 2
1 0
2 0
3 2
1 2 0
2 -3 0
```

Note: the corresponding test Boolean functions are $f(x_1) = \neg x_1$, $f(x_1, x_2) = (x_1) \wedge (x_2)$ and $f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (x_2 \vee \neg x_3)$.

Output for the Sample Input

```
1 2 3 4 5 1
1 1 2 3 4 5 2 3 4 5 1
4 3 5 2 1 4
2 2 3 4 5 1 1 2 3 4 5
1 3 1 5 2 4 1 2 3 4 5
2 5 1 2 3 4 1 2 3 4 5
1 2 4 1 5 3 1 2 3 4 5
16 3 5 2 1 4
2 1 4 5 2 3 2 5 1 3 4
3 3 1 5 2 4 1 2 3 4 5
2 1 2 3 4 5 5 1 2 3 4
3 5 3 2 4 1 2 5 1 3 4
1 1 3 4 2 5 2 4 5 3 1
2 1 2 3 4 5 3 1 5 2 4
```

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

Problem M

Conflicting Sets

Max no. of test cases: 20
Time limit: 2 seconds

Peter has a collection of n sets $\Phi = \{S_1, S_2, \dots, S_n\}$. Each set in Φ is a subset of $\{1, 2, \dots, m\}$ and each set contains at least two integers. The relation between two sets X and Y in Φ is classified into three types: X and Y are disjoint if $X \cap Y = \emptyset$; X and Y are nested if $X \supseteq Y$ or $X \subseteq Y$; and otherwise, X and Y are in conflict. We say that Φ is conflict-free if no two sets in Φ are in conflict. Peter wants to know whether Φ is conflict-free; and if it is not, he wants to know whether he can remove an integer $r \in \{1, 2, \dots, m\}$ from each set such that the resulting collection $\{S_1 - \{r\}, S_2 - \{r\}, \dots, S_n - \{r\}\}$ is conflict-free.

For example, let $n = 3$, $m = 5$, $S_1 = \{1, 2\}$, $S_2 = \{1, 2, 3\}$, and $S_3 = \{4, 5\}$. In this example, the collection is conflict-free, since S_1 and S_2 are nested, S_1 and S_3 are disjoint, and S_2 and S_3 are disjoint. Consider another example, in which $n = 6$, $m = 10$, $S_1 = \{2, 5\}$, $S_2 = \{1, 2, 3, 4\}$, $S_3 = \{1, 2, 4\}$, $S_4 = \{2, 4\}$, $S_5 = \{1, 4\}$, and $S_6 = \{1, 2\}$. Since S_1 and S_2 are in conflict, this collection is not conflict-free. In this example, Peter may remove $r = 2$ to obtain the collection $\Phi' = \{\{5\}, \{1, 3, 4\}, \{1, 4\}, \{4\}, \{1, 4\}, \{1\}\}$. It is not hard to check that Φ' is conflict-free.

Please write a program to help Peter.

Technical Specification

1. The number of test cases is at most 20.
2. $1 \leq n \leq 10^5$ and $1 \leq m \leq 2 \times 10^5$.
3. The total size of all sets, that is, $|S_1| + |S_2| + \dots + |S_n|$, is at most 2×10^5 .
4. Each input set contains at least two integers.

Input File Format

The first line of the input contains an integer indicating the number of test cases. Each test case begins with a line containing two integers n and m . Each of the next n lines gives a set: the i -th line begins with a positive integer $|S_i|$ indicating the size of S_i ; then $|S_i|$ distinct integers follow, which are the integers in S_i . The integers in each set are given in ascending order.

Output Format

If the input collection is conflict-free, print "0". Otherwise, if a solution exists, print the smallest solution, that is, the smallest integer r such that the collection $\{S_1 - \{r\}, S_2 - \{r\}, \dots, S_n - \{r\}\}$ is conflict-free; and if there is no solution, print "-1".

Sample Input

```
4
3 5
2 1 2
3 1 2 3
2 4 5
6 10
2 2 5
4 1 2 3 4
3 1 2 4
2 2 4
2 1 4
2 1 2
4 200000
2 1 2
2 2 3
2 1 3
2 1 3
4 4
2 1 2
2 2 3
2 3 4
2 1 4
```

Output for the Sample Input

```
0
2
1
-1
```

Problem N

Closing Bridges

Max no. of test cases: 20
Time limit: 1 second

Θ is a city consisting of n villages and m roads where n is odd. Each road joins two villages. We say two villages A and B are connected if at least one of the following two rules hold:

- there exists a road R that joins A and B ;
- a village C and a road R exist so that A and C are connected and R joins B and C .

We assume that every two villages in Θ are connected. We say that a road in Θ is a **bridge** if removing the road from Θ disconnects Θ . The bridges in Θ are by design so that each of them can be either open or closed. If a bridge is closed, the villages joined by the bridge are no longer connected. In addition, given the rules mentioned above, some other pairs of villages may be disconnected when the bridge is closed.

Currently, a disease is spreading in Θ . The city government does not have sufficient vaccines, so they are considering implementing an isolation policy. The isolation policy will find a set of bridges to close. For each bridge, if it is closed, then the city Θ is partitioned into two areas so that in each of the two areas every two villages are connected. We define **major area** of a bridge as the area that contains more villages between the two areas separated by closing the bridge. This is well-defined because n is odd. Note that the major area with respect to a bridge only depends on how the villages in Θ are connected, and it does not depend on whether other bridges are closed. Considering the capacity of the medical services, the city government wants to find a set of bridges to close so that the intersection of their major areas consists of k villages for some k in a given range $[a, b]$; that is, the set of integers from a to b , inclusive. If no bridge is closed, then we define the intersection of their major areas contains all villages.

Input File Format

The first line contains only one integer $t \in [1, 20]$, indicating that there are t testcases. Then t testcases follow. The first line of each testcase contains exactly four integers n, m, a , and b . The integer n indicates the number of villages in Θ where n is odd and in the range $[2, 10^3]$. The integer m indicates the number of roads in Θ where $n - 1 \leq m \leq n(n - 1)/2$. The integers a and b indicate the given range where $1 \leq a \leq b \leq n$. The subsequent lines of each testcase give the m pairs of roads. Each road is a pair of integers x, y that indicate the identifiers of the joined villages where $1 \leq x, y \leq n$ and $x \neq y$. No two roads join the same pair of villages.

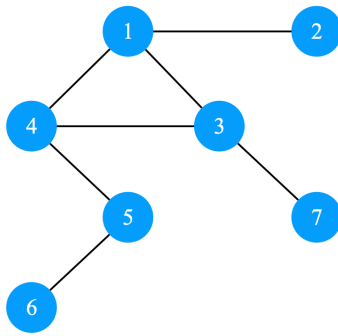


Figure 1: In the above, each node represents a village in Θ and each edge represents a road that connects two villages. The bridges are $\{1, 2\}, \{3, 7\}, \{4, 5\}, \{5, 6\}$. If the given range is $[a, b] = [3, 3]$, then the government can close either all bridges or all bridges except $\{5, 6\}$.

Output Format

If the number of villages in the intersection of the major areas for any set of bridges is outside the given range, then output “-1.” Otherwise, output the minimum number of bridges to close so as to satisfy the requirements.

Sample Input

```

2
7 7 3 3
1 2
1 3
3 7
3 4
1 4
4 5
5 6
3 3 3 3
1 2
2 3
1 3

```

Output for the Sample Input

```

3
0

```