Problem A. Permutation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

Chiaki has a permutation $p_1, p_2, \ldots, p_n$ of integers $1, 2, \ldots, n$ with some unknown positions. She would like to know the number of ways to fill the unknown positions such that the resulting permutation contains a subsequence of length at least 3 that is an arithmetic progression.

As the number may be very large, you are only asked to calculate it modulo $10^9 + 7$.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 50$): the length of the permutation.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le n$), where $p_i = 0$ means that p_i is unknown, and all non-zero elements are distinct.

It is guaranteed that the sum of n in all test cases does not exceed 50.

Output

For each test case, output an integer denoting the answer.

Example

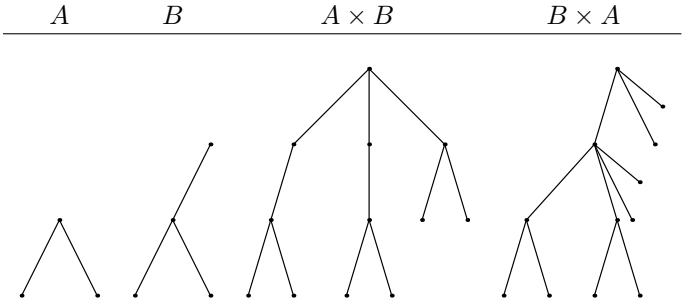
standard input	standard output
2 3 0 0 0 7 1 0 3 0 0 6 0	2 21

Problem B. Tree Product

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given n rooted trees $T_1, T_2, \ldots, T_n$, find two permutations $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_n$ such that the diameter of $T_{p_1} \times T_{p_2} \times \ldots \times T_{p_n}$ is maximum and the diameter of $T_{q_1} \times T_{q_2} \times \ldots \times T_{q_n}$ is minimum.

For two rooted trees A and B, their *tree product* $T = A \times B$ is defined as follows: copy tree A, and then for each vertex x in it, make a copy of B and merge its root with vertex x. See the table below for an example:



It can be shown that tree product is associative: $(A \times B) \times C = A \times (B \times C)$. So the parentheses in a product of three or more trees can be omitted.

Recall that:

- A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. The parent of a vertex v is the last vertex different from v on the path from the root to v.
- The diameter of a rooted tree is the length of the longest simple path in the tree, where the length of a path is the number of edges in the path.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^6$), indicating the number of rooted trees.

Each of the next n lines starts from an integer m_i ($1 \le m_i \le 10^5$), indicating the number of vertices in the i-th rooted tree. It is followed by m_i integers $p_{i,1}, p_{i,2}, \ldots, p_{i,m_i}$ ($0 \le p_{i,j} \le m_i$) on the same line, where the j-th of them denotes the parent of the j-th vertex. The root of the tree has 0 as parent.

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

Output

For each test case, output two integers: the maximum and the minimum diameter, in that order.

Example

standard input	standard output
2 3 5 0 1 2 1 4 3 2 0 2 2 2 0 2 1 0 1 0	8 7 0 0

Note

For the first sample test case, $T_1 \times T_2 \times T_3$ will provide the maximum diameter, while $T_3 \times T_2 \times T_1$ will provide the minimum diameter.

Problem C. Distinct Number

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

Given n intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$ and an integer x, you should find the size of the set $S = \{y \mid y = i \operatorname{AND} x,\ i \in [l_1, r_1] \cup [l_2, r_2] \cup \ldots \cup [l_n, r_n]\}$, where $i \operatorname{AND} x$ is the bitwise *and* of integers i and x.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and x ($1 \le n \le 5 \cdot 10^3$, $0 \le x \le 10^{18}$).

Each of the next n lines contains two integers l_i and r_i ($0 \le l_i \le r_i \le 10^{18}$).

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^3$.

Output

For each test case, output an integer denoting the size of the set S.

Example

standard input	standard output
3 2 1 1 2 343 34345 1 3 1 3 1 123242343 1 1000000000000000000	2 3 32768

Note

For the first sample test case, we have $S = \{0, 1\}$, so the answer is 2.

For the second sample test case, we have $S = \{1, 2, 3\}$, so the answer is 3.

Problem D. Fibonacci Partition

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

The sequence of Fibonacci numbers is defined as:

$$F_n = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

The first few elements of the sequence are $1, 2, 3, 5, 8, 13, 21, 34, \ldots$

For a given positive integer n, let $partition(n)$ be the maximum value of m such that n can be expressed as a sum of m distinct Fibonacci numbers. For example, $partition(1) = partition(2) = 1$, $partition(3) = partition(4) = partition(5) = partition(7) = 2$, $partition(6) = partition(8) = 3$.

Chiaki has an integer X which initially equals to 0. She will perform some operations on X: the i-th operation will add $a_i \cdot F_{b_i}$ to X.

After each operation, Chiaki would like to know the value of $partition(X)$. It is guaranteed that, after each operation, X will be a positive integer.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 5 \cdot 10^4$): the number of operations.

Each of the next n lines contains two integers a_i and b_i ($1 \le |a_i|, b_i \le 10^9$).

It is guaranteed that the sum of n for all test cases will not exceed $5 \cdot 10^4$.

Output

For each test case, output n integers: the i-th integer denotes the value of $partition(X)$ after the i-th operation.

Example

standard input	standard output
1 10 1 1 1 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 -2 5	1 1 2 2 3 3 4 4 5 6

Note

The value of X after each operation in the sample: $1, 2, 4, 7, 12, 20, 33, 54, 88, 72$.

Problem E. Longest Common Subsequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Chiaki has two sequences $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$. She would like to find their longest common subsequence $c_1, c_2, \ldots, c_k$ such that $c_1 \leq c_2 \leq \ldots \leq c_k$.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \leq n, m \leq 10^6$): the lengths of two sequences.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 3$).

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 3$).

It is guaranteed that the sum of $\max\{n, m\}$ in all test cases does not exceed 10^6.

Output

For each test case, output a single integer k: the length of the longest common subsequence $c_1, c_2, \ldots, c_k$ such that $c_1 \leq c_2 \leq \ldots \leq c_k$.

Example

standard input	standard output
3 3 3 1 2 3 1 2 3 3 3 1 1 1 1 1 2 3 3 1 3 2 1 2 3	3 2 2

Problem F. Necklace

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

Chiaki has n beautiful gems. The color of the i-th gem is c_i and the value is v_i.

Chiaki would like to choose at least 3 gems and make a necklace such that the adjacent gems must have different color. Formally, let the indices of gems used in the necklace be $a_1, a_2, \ldots, a_m$ ($m \geq 3$) in clockwise order. For each i ($1 \leq i \leq m$), c_{a_i} should be different from $c_{a_{i \bmod m+1}}$.

Chiaki would like to find a necklace with the maximum possible sum of values: that is, to maximize $\sum_{i=1}^{m} v_{a_i}$.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 2 \cdot 10^5$): the number of gems.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq n$) denoting the color of each gem.

The third line contains n integers $v_1, v_2, \ldots, v_n$ ($-10^9 \leq v_i \leq 10^9$) denoting the value of each gem.

It is guaranteed that the sum of n in all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, the first line contains an integer m ($m \geq 3$): the number of gems in the necklace (note that you don't need to maximize it). The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \leq a_i \leq n$): the indices of gems used in the necklace in clockwise order. If there are several possible answers, print any one of them.

If Chiaki could not find such a necklace, just output an integer -1 on a single line.

Example

standard input	standard output
4 4 1 1 1 1 1 2 3 4 4 1 1 2 2 1 2 3 4 8 2 6 5 4 3 1 7 7 -1 4 -1 2 10 -1 3 0 6 5 5 3 3 4 6 5 8 0 -1 -2 -7	-1 4 1 3 2 4 4 5 2 4 7 4 3 1 4 2

Problem G. Paper-cutting

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

Paper-cutting is one of the oldest and most popular folk arts in China. It can be geographically divided into a southern and a northern style. The southern style, represented by works from Yangzhou in Jiangsu Province and Yueqing in Zhejiang Province, features ingenious and beautiful designs, exquisite carving and interesting shapes. The northern style, mainly from Yuxian and Fengning in Hebei Province and best represented by works from northern Shaanxi, features exaggerated shapes, vigorousness, vivid depictions, and diverse patterns.

There are basic cut-outs, consisting of a single image, and symmetrical designs, that are usually created by some folding over a proportioned crease, and then cutting a shape, so that when unfolded, it forms a symmetrical design. Chinese paper cuttings are usually symmetrical. The paper cutouts are usually in an even number series of 2, 4, 24, etc.

You are given a piece of paper in the shape of a matrix of size $n \times m$. It is partitioned into $n \times m$ blocks of size 1×1. The piece of paper can be folded in the following way:

1. You choose a vertical line between two of its columns or a horizontal line between two of its rows. This line splits the paper into two sides.
2. You use the line as the folding axis and fold the smaller side of the paper onto the larger one going over the axis. If both sides of the paper are of equal size, you may fold from either side.

You would like to make a paper-cutting masterpiece from this paper. At first, you fold the paper using the method above several times (including zero times). Then you use scissors to cut the paper. Each time you cut, you can cut out a connected component from the folded paper (even if the component is not reachable from outside) and throw it away. Note that two 1×1 blocks are connected if they share an edge.

Given the final look of the paper, which is a matrix of size $n \times m$ containing 0s and 1s, you would like to know the minimum number of cuts needed when using the scissors.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n \times m \le 10^6$): the size of the paper.

Each of the next n lines contains a binary string of length m, where "0" means the 1×1 block is cut out and "1" means the 1×1 block remains on the final paper-cutting.

It is guaranteed that the sum of $n \times m$ over all test cases does not exceed 10^6.

Output

For each test case output one line containing one integer, indicating the minimum number of cuts needed.

Example

standard input	standard output
3 2 5 11001 11001 5 7 1001100 0110011 0101101 0010010 1000000 3 2 11 11 11	1 4 0

Note

For the first sample test case, you can fold in the following way and cut the only 0 out:

$$\begin{array}{cccc|cc} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{array} \rightarrow \begin{array}{ccc} 1 & 1 & 0 \\ \hline 1 & 1 & 0 \end{array} \rightarrow \begin{array}{ccc} 1 & 1 & 0 \end{array}$$

For the second sample test case, you can fold in the following way and cut the 4 connected components of 0s out:

$$\begin{array}{cccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \rightarrow \begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{array}$$

Problem H. Partition Number

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

You are given an integer set $A = \{a_1, a_2, \ldots, a_n\}$. Please calculate the number of solutions for equation $x_1 + x_2 + \ldots + x_k = m$, where x_i are positive integers, $x_1 \le x_2 \le \ldots \le x_k$ and $x_i \notin A$.

As the answer may be very large, you are only asked to calculate it modulo $(10^9 + 7)$.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n \le 500$, $n \le m \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$, $a_i \ne a_j$ for all $i \ne j$).

It is guaranteed that the sum of n over all test cases does not exceed 500.

Output

For each test cases, output an integer denoting the answer.

Example

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

Note

There are 5 solutions for $m = 4$ if the constraints set A is empty. They are:

$$\begin{aligned} 4 &= 1+1+1+1 \\ &= 1+1+2 \\ &= 1+3 \\ &= 2+2 \\ &= 4 \end{aligned}$$

Problem I. Stirling Number

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

The Stirling number of the first kind $\begin{bmatrix} n \\ k \end{bmatrix}$ is the number of permutations of n elements with exactly k disjoint cycles. The well-known recurrence relation is defined as follows:

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}$$

for $k > 0$, with the initial conditions

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 1 \quad \text{and} \quad \begin{bmatrix} n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} = 0$$

for $n > 0$.

Given four integers, n, l, r, and p, find the value of

$$\left(\sum_{k=l}^{r} \begin{bmatrix} n \\ k \end{bmatrix} \right) \bmod p$$

where p is prime.

Input

The first line contains four integers, n, l, r, and p ($1 \le n \le 10^{18}$, $0 \le l \le r \le n$, $2 \le p \le 10^6$, p is prime).

Output

Output an integer denoting the answer.

Examples

standard input	standard output
4 1 4 5	4
6 5 5 29	15
1000 685 975 999983	482808

Problem J. Ternary String Counting

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

Chiaki studies ternary strings s of lentgh n. A ternary string is a string consisting of characters "0", "1", and "2".

Chiaki has made m restrictions, and the i-th restriction is: the number of distinct characters of the substring of s from the l_i-th position to the r_i-th position (both inclusive) is exactly x_i.

Chiaki would like to know the number of strings which meet the m restrictions. As the number may be very large, you are only asked to calculate it modulo $10^9 + 7$.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n \le 5000$, $0 \le m \le 10^6$): the length of the string and the number of restrictions.

Each of the next m lines contains three integers, l_i, r_i, and x_i ($1 \le l_i \le r_i \le n$, $1 \le x_i \le 3$).

It is guaranteed that the sum of n over all test cases does not exceed 5000, and the sum of m over all test cases does not exceed 10^6.

Output

For each test case, output an integer denoting the number of such strings modulo $10^9 + 7$.

Example

standard input	standard output
4 1 0 2 0 3 0 5 2 1 3 3 4 5 1	3 9 27 18

Note

In the fourth sample, all possible strings are: 21000, 12000, 20100, 02100, 10200, 01200, 21011, 12011, 20111, 02111, 10211, 01211, 21022, 12022, 20122, 02122, 10222, 01222.

Problem K. Anti-hash Test

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

It is well-known that the following string $s(n) = s_0 s_1 \dots s_{2^n-1}$ can challenge almost every solution that uses polynomial hashes modulo 2^{64}:

$$s_i = \begin{cases} \text{“a”}, & \text{popcount}(i) \bmod 2 = 0 \\ \text{“b”}, & \text{popcount}(i) \bmod 2 = 1 \end{cases}$$

where popcount(i) means the number of ones in binary representation of number i.

Given a string u and an integer n, find the number of occurrences of u in string $s(n)$ and the number of distinct strings v which have the same number of occurrences in string $s(n)$. As both the numbers may be very large, you are only asked to calculate them modulo $10^9 + 7$.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^{18}$).

The second line contains a string u ($1 \le |u| \le \min(10^6, 2^n)$) consisting only of letters “a” and “b”.

It is guaranteed that the sum of $|u|$ over all test cases does not exceed 10^6.

Output

For each test case, if the string u does not appear in string $s(n)$, you should simply output -1. Otherwise, output two integers denoting the the number of occurrences of u in string $s(n)$ modulo $10^9 + 7$ and the number of distinct strings v which have the same number of occurrences in string $s(n)$ modulo $10^9 + 7$.

Example

standard input	standard output
4 10 a 10 abba 5 abbabaabbaababbabaababbaabbabaab 20 ababab	512 2 171 4 1 344 -1

Problem L. Tokens on the Tree

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

Chiaki has a tree with n vertices, labeled by integers from 1 to n. For each vertex in the tree, there is a white token or a black token or no token at all. There are exactly w white tokens and exactly b black tokens. Also, for each pair of vertices with the same color of tokens, there exists a path between them such that every vertex on the path contains a token of the same color.

Chiaki would like to perform the following operations:

1. Choose a vertex u with a token.
2. Choose a path $p_1, p_2, \ldots, p_k$ such that $p_1 = u$, all vertices $p_1, p_2, \ldots, p_{k-1}$ contain a token of the same color, and there's no token in p_k.
3. Move the token in p_1 to p_k. Now there's no token in p_1 and p_k contains a token.

For two initial configurations of tokens S and T, if Chiaki could perform the above operations several (zero or more) times to make S become T, then S and T are considered equivalent.

Let $f(w, b)$ be the number of equivalence classes (that is, the maximum number of configurations that no two are equivalent). Chiaki would like to know the value of

$$\left(\sum_{w=1}^{n-1}\sum_{b=1}^{n-w} w \cdot b \cdot f(w, b)\right) \bmod (10^9 + 7).$$

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains an integer n ($2 \le n \le 2 \cdot 10^5$): the number of vertices in the tree.

The second line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i means there is an edge between vertex i and vertex p_i.

It is guaranteed that the sum of n of all test cases will not exceed $2 \cdot 10^5$.

Output

For each test case, output an integer denoting the value of

$$\left(\sum_{w=1}^{n-1}\sum_{b=1}^{n-w} w \cdot b \cdot f(w, b)\right) \bmod (10^9 + 7).$$

Example

standard input	standard output
2 5 1 2 3 3 10 1 2 3 4 3 6 3 8 2	71 989

Note

For the first sample, the values of $f(w, b)$ for each w and b are:
$f(1, 1) = 1$, $f(1, 2) = 2$, $f(1, 3) = 3$, $f(1, 4) = 3$,

$f(2,1) = 2,\ f(2,2) = 2,\ f(2,3) = 1,$
$f(3,1) = 3,\ f(3,2) = 1,$
$f(4,1) = 3.$

Problem M. A + B Problem

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

A binary string is a string consisting only of digits "0" and "1". Given a binary string s of length $(n+m)$, please divide it into two subsequences $A = a_1a_2\ldots a_n$ of length n and $B = b_1b_2\ldots b_m$ of length m such that each digit in s belongs to exactly one subsequence.

Let f be the function that transforms a sequence of "0" and "1" into a binary integer. For example, $f(\{1,0,1,0\}) = 1010_2$ and $f(\{0,0,1,0\}) = 10_2$. Your task is to find such A and B that maximize $(f(A) + f(B))$.

Recall that a subsequence of a string is a sequence that can be derived by deleting some characters (possibly none) from the string without changing the order of the remaining characters.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ indicating the lengths of the desired subsequences.

The second line contains a binary string s $(|s| = n + m,\ s_i \in \{\text{"0"}, \text{"1"}\})$.

It is guaranteed that the sum of $(n + m)$ of all test cases will not exceed $2 \cdot 10^6$.

Output

For each test case, output one line containing a binary integer indicating the largest possible result of $(f(A)+f(B))$. Note that $(f(A)+f(B))$ should be printed as a binary integer with no leading zeroes, while A and B are sequences, and leading zeros are allowed in the sequences.

Example

standard input	standard output
3 4 3 1000101 2 2 1111 1 1 00	1101 110 0

Note

We now use underline to indicate subsequence A in the binary string.

For the first sample test case, a valid solution is to divide the binary string into $\underline{1000}\underline{101}$ such that $f(\{1,1,0,1\}) + f(\{0,0,0\}) = 1101_2 + 0_2 = 1101_2$.

For the second sample test case, a valid solution is to divide the binary string into $\underline{11}\underline{1}1$ such that $f(\{1,1\}) + f(\{1,1\}) = 11_2 + 11_2 = 110_2$.