Problem A. A Tree Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Little I and Little J are playing a game again.

Little J brings a tree with n vertices. Each edge of the tree has two states: open and closed. Initially, all edges of the tree are open.

There is a chip initially placed at vertex 1. Little I can move the chip, and the goal is to move the chip to a vertex with degree **exactly equal to** 1. Little J can close edges of the tree with the goal of preventing Little I from moving the chip to a vertex with degree exactly 1. The degree of a vertex is the number of edges connected to it, regardless of whether they are open or closed.

The game consists of several rounds, each round having the following steps:

1. Little I Task Determination: If the chip is at a vertex with degree exactly 1, Little I wins. Otherwise, proceed to step 2.
2. Little J Action: Little J closes one currently open edge permanently. If there are no open edges at the moment, skip the action and proceed to step 3.
3. Little I Action: Little I chooses an open edge connected to the vertex currently containing the chip, and moves the chip to the other end of this edge. If there is no such edge, Little J wins. Otherwise, a new round begins, going back to step 1.

Little J wants to know who will win if Little I and Little J know the structure of this tree and are extremely smart.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) representing the number of vertices in the tree.

Then follow $n-1$ lines, each containing two integers u and v ($1 \le u, v \le n$) representing two vertices connected by an edge of the tree.

Output

If Little I wins, print `1`. Otherwise, print `0`.

Examples

standard input	standard output

6
1 2
2 3
2 4
1 5
5 6
	0

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

Problem B. Binary String

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

You are given a string $s_1 s_2 \ldots s_n$ of length n with elements from the character set "`01?`".

For every $k \in [1, n]$, consider the string $T_k = t_1 t_2 \ldots t_n$ where, for $1 \le i \le n$:

- If $s_i \neq$ `?`, then $t_i = s_i$.
- Otherwise, if $i \le k$, then $t_i =$ `0`.
- Otherwise, $t_i = t_{i-k}$, and you can recursively compute t_{i-k} to obtain t_i.

It is easy to see that the character set of T_k is "`01`". You need to calculate the number of `1` in T_k for all $k \in [1, n]$.

Input

The first line of input contains an integer n ($1 \le n \le 10^5$) representing the length of the string.

The second line contains the string $s_1 s_2 \ldots s_n$ of length n with elements from the character set "`01?`".

Output

Output n lines, where the k-th line contains an integer representing the number of `1` in T_k.

Example

standard input	standard output
5 10?1?	3 4 2 3 2

Problem C. Colored Slime Balls

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

There are n slime balls arranged in a row, where the i-th slime ball has color c_i and mass m_i.

You can perform any number of operations to increase the mass of a slime ball by 1, and it costs w per operation.

However, once the mass of a slime ball reaches k or more, it becomes unstable, and must be sold **before** the next operation. You can only sell slime balls with mass greater than or equal to k. According to the market price, selling a slime ball with mass i earns you income p_i.

It is guaranteed that $p_i - p_{i-1} < w$, but p_i is not necessarily monotonically non-decreasing.

After you sell a slime ball, all the slime balls on its two sides move to close the gap. Moreover, if the ball you sold had two neighbors of the same color, they will merge into one slime ball whose mass is the sum of the two. This new slime ball may also need to be sold, continuing the process.

You want to know the maximum net profit after you sell all the slime balls.

Input

The first line contains three positive integers n, k, w ($1 \le n \le 150$, $2 \le k \le 10$, $1 \le w \le 10^6$).

The second line contains n positive integers, where the i-th integer is the color c_i of the i-th slime ball ($1 \le c_i \le n$). It is guaranteed that $c_i \ne c_{i-1}$. In other words, initially, there are no neighboring balls that have the same color.

The third line contains n positive integers, where the i-th integer represents the initial mass m_i of the i-th slime ball ($1 \le m_i < k$).

The fourth line contains $k-1$ integers representing the income from selling slime balls with masses k to $2k-2$. Specifically, the numbers are $p_k, p_{k+1}, \ldots, p_{2k-2}$ ($0 \le p_i \le 10^9$, and $p_i - p_{i-1} < w$).

Output

Print a line with a single integer: the maximum net profit from selling all the slime balls.

Example

standard input	standard output

4 5 6
2 1 2 3
3 3 3 4
5 7 9 11
	-1

Note

In the example, first, increase the mass of the slime ball with color 3. Then, it is sold, earning income 5.

Then, increase the mass of the slime ball with color 1 twice. Then, it is sold, earning income 5. Next, the two slime balls with color 2 merge and are sold, earning income 7.

After three operations incurring a total cost of 18, the net profit is -1. It can be proved that there is no better solution.

Problem D. Daisies on a Grid

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

You are given a grid consisting of n rows and m columns where each cell is a place for a magical daisy. Each daisy can have one of the three possible colors, represented by 0, 1, and 2.

Each daisy gazes at its surrounding flowers and desires to transform into the appearance of other daisies. If at the start of some second, a daisy of color c has at least one neighboring flower above, below, to the left, or to the right with color $c - 1$, then in the next second, this daisy will turn into color $c - 1$; otherwise, its color remains c in the next second (we consider the colors modulo 3).

Consider an initial arrangement of daisies on the grid. The arrangement is *beautiful* if, after a finite number of seconds, all daisies become the same color.

It is easy to see that, for a beautiful daisy arrangement, each flower has an earliest second t such that its color remains unchanged after second t. We call this second the flower's *stable second.* We start counting from second 0, so, if a flower never changes color, its stable second is 0.

Now, some daisies are already placed in some cells of the grid, and the other cells are empty. How many ways are there to place daisies in the remaining cells so that the arrangement is beautiful? Also, for all these beautiful arrangements, what is the total sum of stable seconds for the daisy in the top left cell (the cell on the intersection of the first row and the first column)?

As both numbers may be very large, find them modulo 998 244 353.

Input

The first line of the input contains two integers n and m ($2 \le n \le 5$, $2 \le m \le 50$).

Then the description of the initial state of the grid follows: the following n lines contain m integers each. The j-th integer in the i-th line, $a_{i,j} \in \{0, 1, 2, 3\}$, represents the state of the corresponding cell. Here, $a_{i,j} \in \{0, 1, 2\}$ indicates a daisy of the respective color, and $a_{i,j} = 3$ indicates that the cell contains no daisy.

Output

Print a line with two integers: the number of beautiful arrangements and the total sum of stable seconds for the daisies in the top left cell in those arrangements, both modulo 998 244 353.

Examples

standard input	standard output

2 2
1 0
3 2
	1 2

5 5
3 3 3 3 2
2 3 3 3 1
1 3 3 3 3
3 3 3 3 3
3 3 3 3 3
	50830224 170059345

Problem E. Easily Broadcastable Tensors

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 mebibytes

Little I is learning to use PyTorch. It is a very popular Python library for machine learning training.

Little I noticed that PyTorch has a mechanism for tensor operations called "broadcasting". You can think of tensors as multidimensional arrays. For a k-dimensional tensor A, we use a sequence of length k denoted as $(a_1, a_2, \ldots, a_k)$ to represent the lengths of its dimensions, meaning A is a tensor of size $a_1 \times a_2 \times \ldots \times a_k$.

For two tensors A and B, with dimensions $(a_1, a_2, \ldots, a_m)$ and $(b_1, b_2, \ldots, b_n)$, respectively, A and B are *easily broadcastable* if and only if the following property holds:

For any integer $0 \le i \le \min(n, m) - 1$, either $a_{m-i} = b_{n-i}$ or at least one of a_{m-i} and b_{n-i} is 1.

Now, Little I has two tensors with dimensions $(p_1, p_2, \ldots, p_m)$ and $(q_1, q_2, \ldots, q_n)$, and they may not be easily broadcastable.

To make them easily broadcastable, Little I can use several operations (or none), where each operation modifies the sequence p or q as follows:

Choose p or q, and insert a 1 at any position in the chosen sequence.

Little I wants to know the minimum number of operations required to make the two tensors easily broadcastable.

Input

The first line contains two integers m and n ($1 \le m, n \le 2000$) representing the dimensions of the two tensors.

The second line contains m integers $p_1, p_2, \ldots, p_m$ ($1 \le p_i \le 2000$) describing the length of each dimension for the first tensor.

The third line contains n integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le 2000$) describing the length of each dimension for the second tensor.

Output

Print a line with a single integer: the minimum number of insertions of 1 required to make the two tensors easily broadcastable.

Example

standard input	standard output

4 2
2 1 3 2
4 2
	1

Note

In the example, inserting a 1 before the second position in sequence q (resulting in `4 1 2`) makes the two tensors easily broadcastable.

Problem F. Fast Algorithm

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Little I has invented an algorithm for finding the minimum cycle in a directed graph with the time complexity of $O(n+m)$, and now he wants to test your skills.

You are given a directed graph with n vertices and m edges. Each edge has a positive integer weight. Your task is to find a cycle in the graph such that the sum of edge weights on the cycle is minimized. Output the minimum value of this sum or report if there is no cycle.

Of course, since you may not know the $O(n+m)$ algorithm for finding the minimum cycle, Little I has relaxed the conditions: the input graph is guaranteed to be weakly connected, and $m-n$ won't be very large. A graph is weakly connected if and only if it becomes a connected undirected graph after replacing directed edges with undirected edges.

Input

The first line contains two integers n and m ($1 \le n \le 3 \cdot 10^5$, $-1 \le m-n \le 1500$) representing the number of vertices and edges in the graph.

Each of the next m lines contains three integers u_i, v_i, w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^9$) representing a directed edge from u_i to v_i with weight w_i. The graph is guaranteed to be weakly connected.

Output

Print a line with a single integer: the length of the minimum cycle in the graph, or -1 if there is no cycle.

Examples

standard input	standard output

4 6
1 2 1
4 3 3
4 1 9
2 4 1
3 1 2
3 2 6
	7

1 0	-1
1 1 1 1 1	1

Note

In the first example, the minimum cycle is $1 \to 2 \to 4 \to 3 \to 1$.

Problem G. Generating the Sequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	1024 mebibytes

There is a sequence $a_1, a_2, \ldots, a_n$ of length n, and it is guaranteed that each a_i is an **odd** number.

There are two types of operations:

1. Given ℓ, r, and x, add an **even** number x to each of the elements $a_\ell, a_{\ell+1}, \ldots, a_r$.
2. Given ℓ and r, find the product of $a_\ell, a_{\ell+1}, \ldots, a_r$, and output the answer modulo 2^{20}.

Given the initial sequence and the operations, perform them efficiently.

Input

The first line of the input contains two positive integers n and q ($1 \le n, q \le 2 \cdot 10^5$) representing the length of the sequence and the number of queries.

The second line contains n odd numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i < 2^{20}$).

The following q lines represent queries, each in one of the following two formats:

- "1 ℓ r x": perform the first type of operation ($1 \le \ell \le r \le n$, the value x is even and $0 \le x < 2^{20}$).
- "2 ℓ r": perform the second type of operation ($1 \le \ell \le r \le n$).

All the values in the queries are integers.

Output

For each query of type 2, output one line with an integer representing the answer.

Example

standard input

```
10 10
969575 741825 24903 1047319 450475 256145 1045323 479255 810659 768323
1 5 6 3034
2 1 10
2 1 9
2 1 4
1 3 6 126904
2 5 5
2 9 9
1 7 7 853094
1 4 9 1025178
2 5 8
```

standard output

```
1045541
1012343
558151
580413
810659
527353
```

Problem H. Hamiltonian Circuit

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given n pairs of integers (a_i, b_i).

Consider a weighted directed complete graph G with n vertices, where the weight of the edge from i $(1 \le i \le n)$ to j $(1 \le j \le n)$ is $|a_i - b_j|$.

Find a Hamiltonian circuit in G such that the sum of weights of the edges it traverses is maximized, and output this maximum value.

Input

The first line of the input contains an integer n $(2 \le n \le 10^5)$ representing the number of pairs.

Each of the next n lines contains two integers a_i and b_i $(0 \le a_i, b_i \le 10^9)$ representing a single pair.

You may assume that **all** $2n$ integers a_i and b_i are pairwise distinct.

Output

Print a line with a single integer: the maximum sum of weights of the Hamiltonian circuit.

Example

standard input	standard output

3
1 10
8 2
4 5
	10

Note

In the example, consider the Hamiltonian circuit $1 \to 2 \to 3 \to 1$, with edge weights $|1-2| + |8-5| + |4-10| = 10$. It can be proven that there is no Hamiltonian circuit with sum of weights exceeding 10, so the answer is 10.

Problem I. Interesting Words

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Ene loves palindromes.

Ene has a list of interesting words. She wants to select some interesting words and concatenate them to form a palindrome of length exactly L. Each interesting word can be chosen multiple times or not chosen at all.

Ene wants to know the number of ways to do this. Two ways are considered different when the sequences of the concatenated interesting words in them are different. Note that multiple different ways may result in the same palindrome. Since the answer may be very large, you need to find the result modulo $10^9 + 7$.

Input

The first line of the input contains two positive integers N and L representing the number of the interesting words and the length of the palindrome to be formed ($1 \le N \le 333$, $1 \le L \le 1000$).

Each of the following N lines contains an interesting word s_i ($1 \le |s_i| \le L$, $\sum_{i=1}^{N} |s_i| \le 600$; the interesting words only contain lowercase English letters and are pairwise distinct).

Output

Output a line with a single integer: the number of ways to form a palindrome modulo $10^9 + 7$.

Examples

standard input	standard output
7 9 cats eel eve evil lee olive stack	5
2 2 a aa	2
6 12 aa aab no on pets step	43

Note

In the first example, the five different ways are the following:

`stack cats`; `evil olive`; `eel eve lee`; `lee eve eel`; `eve eve eve`.

Problem J. Junctions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

The streets and junctions of the Martian City can be represented as a weighted bidirectional complete graph where the n junctions are the vertices and the streets are the edges. The weight of an edge is the length of the corresponding street.

For each edge (a, b), determine whether there exists a pair of vertices (x, y) such that all shortest paths from x to y pass through the edge (a, b).

Input

The first line contains a positive integer n ($1 \le n \le 500$) representing the number of junctions in the city.

Each of the next n lines contains n space-separated integers. Together, they form an $n \times n$ matrix. The number $a_{i,j}$ ($1 \le a_{i,j} \le 10^6$) in the i-th row and j-th column represents the length of the bidirectional street between junctions i and j. Specifically, $a_{i,i} = 0$ and $a_{i,j} = a_{j,i}$.

Output

Output a binary matrix of size $n \times n$ without spaces. The entry in the i-th row and j-th column must be 1 if the edge (i, j) satisfies the conditions described in the problem, and 0 otherwise.

In particular, output 0 when $i = j$.

Example

standard input	standard output

4
0 3 2 100
3 0 8 100
2 8 0 10
100 100 10 0
	0110
	1000
	1001
	0010

Problem K. Kids and Integers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

This is the fourth time that Little I and Little J have played the counting game, so they have decided not to create lengthy problem statements anymore. You only need to know that they have come up with another strange rule for selecting the numbers to count, and want to figure out how many such numbers are there.

For any positive integer n, we define the function $f(n)$ as the sum of its decimal digits, for example, $f(114514) = 1 + 1 + 4 + 5 + 1 + 4 = 16$. Obviously, $f(n)$ is also a positive integer, so the function can be applied repeatedly as $f(f(n))$, $f(f(f(n)))$, and so on. For positive integers n and k, we define the function $g(n, k)$ as $f(f(\ldots f(n)\ldots))$ with k applications of f.

To make the counting game different each time, the kids decided to set two positive integers k and m for each round, and then to specify the rule: in this round, the numbers to count are all positive integers n satisfying $g(n, k) = m$.

As both of them are game experts, they can find arbitrarily many such integers. To prevent the game from going on forever, they also pick a positive integer N in each round as the upper bound for counting. They want to know: among positive integers not exceeding N, how many numbers satisfy $g(n, k) = m$? Since the answer may be very large, find it modulo $10^9 + 7$.

Input

The first line of the input contains one integer T, representing the number of rounds Little I and Little J will play ($1 \le T \le 5$).

Each of the next T lines contains three positive integers N, k, m describing a single round of the game ($1 \le N \le 10^{1000}$, $1 \le k, m \le 10^9$).

Output

For each of the T rounds, print a line with a single integer: the number of numbers that cannot be counted in this round of the game, modulo $10^9 + 7$.

Example

standard input	standard output

2
114 1 5
514 2 10
	8
	10