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 $s_1 s_2 \dots s_n$ of length n with elements from the character set "`01?`".

For every $k \in [1, n]$, consider the string $T_k = t_1 t_2 \dots t_n$ 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 $s_1 s_2 \dots s_n$ 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. Crimson ant Turquoise

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

You have been tasked with painting a fence made of 2024 planks. You have a shades of crimson paint and b shades of turquoise paint. Additional customer requirement:

- Among any 5 consecutive planks, the first, third, and fourth planks must be painted with the same shade of crimson.
- The fence must contain at least one crimson and at least one turquoise plank.

How many different ways are there to paint the fence, satisfying the customer's requirement?

Input

The first line of input contains one integer a ($1 \le a \le 10^4$) — the number of shades of crimson paint. The second line of input contains one integer b ($1 \le b \le 10^4$) — the number of shades of turquoise paint.

Output

Output a single number — the number of different ways to paint the fence, satisfying the customer's requirement.

Example

standard input	standard output
2 2	4

Problem D. Digital Coolness

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

Among the clients of the CoolStar mobile operator, the "cool" phone numbers are considered particularly prestigious.

The phone number has 11 digits. The first four digits of the phone number are the prefix (fixed for the entire number pool).

The *coolness* of a phone number is determined as the maximum of the following three numbers:

- the maximum length of a sequence of consecutive digits, each of which is 1 greater than the previous one,
- the maximum length of a sequence of consecutive digits, each of which is 1 less than the previous one,
- the maximum length of a sequence of consecutive identical digits.

For example, let's say we have the number 380165444567. This number has 3 consecutive identical digits (444), three consecutive decreasing digits (654), and 4 consecutive increasing digits (4567), the maximum is 4, so the coolness of the number is 4. The coolness of the number 38010987654 is 6 (since 0 and 9 are not consecutive: the set of digits is not cyclical).

You are given four digits: the prefix of the phone number. Find the number of numbers with the given prefix that have the maximum possible coolness.

Input

The input contains a string of four decimal digits: the prefix of the phone number.

Output

Output a single integer: the number of numbers with the given prefix that have the maximum possible coolness.

Examples

standard input	standard output
3802	2
1987	1

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:	8 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 868475 731715 23183 1012379 432335 333145 1022253 414255 212349 212123 1 5 6 3032 2 1 10 2 1 9 2 1 4 1 3 6 116304 2 5 5 2 9 9 1 7 7 823184 1 4 9 1015028 2 5 8
standard output
715383 823637 74885 551671 212349 1047685

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:	4 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*
2 2 o oo	2
6 12 pp ppq ex xe cuts stuc	43

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