Problem A. Car Show

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

There are n cities and m car styles in NIO Kingdom. According to the investigations, for the i-th city, cars of style T_i are the most popular.

Now there will be a car show in NIO Kingdom, and the person in charge wants to choose an integer interval $[l, r]$ $(1 \le l \le r \le n)$ and let the cities whose indices are in this interval hold the show jointly. And to manifest the variety of the cars, every style should occur at least once among the most popular styles in the host cities. Determine the number of integer intervals satisfying the constraint mentioned before.

Input

The first line contains two integers n and m $(1 \le m \le n \le 100\,000)$, denoting the number of cities and the number of car styles respectively.

The second line contains n integers $T_1, T_2, \cdots, T_n$ $(1 \le T_i \le m)$, denoting the most popular styles.

Output

Output one line containing one integer, denoting the answer.

Example

standard input	standard output
5 3 1 2 3 2 1	5

Note

For the sample case, the 5 intervals are $[1, 3], [1, 4], [1, 5], [2, 5], [3, 5]$ respectively.

Problem B. Two Frogs

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

In the Lonely Mountain, there are a lot of treasure well protected by dwarfs. Later, one day, the last dragon Smaug came and sensed the treasure being. As known to all, dragons are always much too greedy for treasure. So definitely, the war between dwarfs and Smaug begins.

During the war, two goblins Alice and Bob are turned into frogs by Gandalf, The Grey. In front of them, there are n lotus leaves in a line. In order to break the spell, they must jump from the 1-st lotus leaf to the n-th lotus leaf. If the frog is now on the i-th lotus leaf instead of the n-th lotus leaf, it can jump to a lotus leaf in range $(i, i + a_i]$.

Goblins are lack of intelligence and it's also true even after turned into frogs. So Alice and Bob will jump randomly, which means, they will separately pick an available lotus leaf in every jump uniformly at random.

Since Alice and Bob have already being playing games for decades, so they want to know the probability that they jump to the n-th lotus leaf with the same count of jumps.

Input

The first line contains an integer n ($2 \le n \le 8\,000$), denoting the number of lotus leaf.

The second line contains $n-1$ integers $a_1, a_2, \ldots, a_{n-1}$, where the i-th integer a_i ($1 \le a_i \le n-i$) indicates the range of lotus leaves that can be reached from the i-th lotus leaf.

Output

Output a line containing a single integer, indicating the probability that Alive and Bob jump to n-th lotus leaf with the same count of jumps, taken modulo 998 244 353.

Formally speaking, let the result, which is a rational number, be $\frac{x}{y}$ as an irreducible fraction, you need to output $x \cdot y^{-1} \bmod 998\,244\,353$, where y^{-1} is a number such that $y \cdot y^{-1} \equiv 1 \pmod{998\,244\,353}$. You may safely assume that such y^{-1} always exists.

Examples

standard input	standard output
5 1 1 1 1	1
5 4 3 2 1	440198031

Problem C. Global Positioning System

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Global Positioning System (GPS) is based on n artificial satellites, where each satellite has a 3D-coordinates (x, y, z). Since the coordinates are always changing, we can hardly know the exact value of the coordinates, even if the satellites are relative static to each other.

To get more information about their coordinate, we still try every effort to do m measurements. For each measurement, we choose the u-th and the v-th satellites, and measure the coordinate difference between them. Specifically, we will get $(x_v - x_u, y_v - y_u, z_v - z_u)$ after the measurement of them. With these m measurement results, we can now know the coordinate difference between any two satellites.

However, `lzr010506` has the impression that he has mistaken exactly one measurement result, so there may exist no possible coordinates to match all the measurements for the satellites for now. You need to find out all the measurements results that might be mistaken, so he can try to fix it as soon as possible.

Here a measurement result might be mistaken iff there exist possible coordinates to match all the measurements after modifying the mistaken measurement result to the one other than the original one.

Input

The first line contains two integers n ($2 \le n \le 500\,000$) and m ($n - 1 \le m \le 500\,000$), denoting the number of the satellites that form the GPS and the number of the measurements that we have made.

The following m lines decribe the m measurements. Each of the m lines contains five integers u, v ($1 \le u, v \le n, u \ne v$), x, y and z ($-2 \times 10^9 \le x, y, z \le 2 \times 10^9$), which means that the measurement result between the u-th and the v-th satellites is (x, y, z).

It is guaranteed that any (unordered) pair of satellites will not be measured more than once, and at least one measurement result might be mistaken.

Output

The first line contains a single integer k, denoting the number of the measurements results that might be mistaken.

The second line contains k integers, denoting the indices of the measurement results that might be mistaken in increasing order.

Examples

standard input	standard output
3 3 1 2 3 3 3 1 3 1 1 1 2 3 -3 -2 -3	3 1 2 3
5 6 1 2 1 1 1 2 3 1 1 1 3 1 -2 -2 -2 1 5 4 4 4 4 5 1 1 1 3 4 2 2 2	3 4 5 6
3 2 1 2 1 1 1 2 3 1 1 1	2 1 2

Note

In the first sample test case, you can modify the first measurement result to $(4, 3, 4)$, or modify the second measurement result to $(0, 1, 0)$, or modify the third measurement result to $(-2, -2, -2)$.

In the third sample test case, even if possible coordinates of the satellites already exist, you can still modify any measurement results to keep the existence of possible coordinates.

Problem D. Half Turns

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given two even integers n and m, construct a matrix M satisfying following constraints:

- The number of rows and columns of M are n and m respectively.
- For each integer i ($1 \le i \le nm$), i appears exactly once in M.
- For each two adjacent integers i and $i+1$ ($1 \le i < nm$), denoting $M_{x_1,y_1} = i$ and $M_{x_2,y_2} = i+1$, then $|x_1 - x_2| + |y_1 - y_2| = 1$ holds.
- The number of turning integers is exactly $\frac{nm}{2}$, where i ($1 < i < nm$) is a turning integer iff the three integers $i-1$, i and $i+1$ are not in the same row or the same column.

If multiple solution exist, print any one of them. If no solution exist, report it.

Input

The only line contains two even integers n and m ($2 \le n, m \le 1\,000$).

Output

If no solution, print "No" (without quotes) in one line.

If solution exists, print "Yes" (without quotes) in the first line. Then print n lines each containing m integers $M_{i,1}, M_{i,2}, \cdots, M_{i,m}$, denoting the answer matrix.

Example

standard input	standard output
4 4	Yes 2 3 4 5 1 8 7 6 16 9 10 11 15 14 13 12

Note

In the sample case, the number of turning integers is 8, which equals $\frac{4\times4}{2}$, and the 8 turning integers are $2, 5, 6, 8, 9, 11, 12, 15$.

Problem E. Longest Increasing Subsequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given an integer m, you need to construct a permutation p whose length does not exceed 100 such that there are exactly m longest increasing subsequences in p.

If multiple solutions exist, print any of them. If no solution exists, report it.

Input

The first line contains an integer T ($1 \le T \le 1\,000$), denoting the number of test cases.

Each test case contains an integer m ($1 \le m \le 10^9$) in one line.

Output

For each test case:

If no solution exists, print "`-1`" (without quotes) in one line.

Otherwise, print one integer n ($1 \le n \le 100$) in the first line denoting the length of the permutation you construct. Then print n integers $p_1, p_2, \cdots, p_n$, ($1 \le p_i \le n, \forall 1 \le i < j \le n, p_i \ne p_j$) in the second line denoting the permutation you construct.

Example

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

Note

In the first test case, the length of the LIS (longest increasing subsequence) is 2, and the 3 LISs are $\{2, 4\}, \{2, 3\}, \{1, 3\}$.

In the second test case, the length of the LIS is 2, and the 5 LISs are $\{3, 5\}, \{3, 4\}, \{2, 5\}, \{2, 4\}, \{1, 4\}$.

Problem F. Matrix and GCD

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	512 megabytes

Given an $n \times m$ Matrix M where each integer i ($1 \le i \le nm$) appears exactly once in M. Let the value of a submatrix of M as the greatest common divisor (GCD) of all the entries among the submatrix, you need to determine the sum of the values for all **continuous** submatrices of M.

Input

The first line contains two integers n and m ($1 \le n, m \le 1\,000$).

Following n lines each contains m integers $M_{i,j}$ ($1 \le M_{i,j} \le nm$), denoting the given matrix.

It is guaranteed that each integer i ($1 \le i \le nm$) appears exactly once in M.

Output

Output one line containing one integer, denoting the answer.

Example

standard input	standard output
3 3 1 2 3 4 5 6 7 8 9	78

Problem G. Magic Spells

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

One day in the magic world, the young wizard RoundDog was learning the compatibility of spells. After experimenting for a long time, he reached the conclusion that the compatibility of a spell collection can be measured by the number of distinct palindromes that are substrings of all the spells in the collection. He was so excited and planned to write a program to calculate the compatibility of any input spell collection.

However, RoundDog was busy participating the NowWizard Multi-Universe Training this week. He was too struggling during the competition and feels tired now.

Since RoundDog is not in the mood to finish the program now, could you help him?

Input

The first line contains a single integer k ($1 \le k \le 5$), indicating the number of spells in a spell collection.

In the following k lines, each line contains a spell S ($1 \le |S| \le 300\,000$), which is a string containing only lowercase English letters.

It is guaranteed that the total length of the spells does not exceed 300 000.

Output

Output the compatibility of the input spell collection, which is the number of distinct palindromes that are substrings of all the spells in the collection.

Example

standard input	standard output
3 abaca abccaca acabb	4

Note

In the example, "a", "b", "c", "aca" are the four distinct palindromes that are substrings of all the input spells.

Problem H. Radar Scanner

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

A military manoeuvre is going on a two-dimension Cartesian plane and n enemy targets are hiding somewhere.

Our troops has occupied a radar station located at the origin, and they realize that the radar scanner in the radar station can instruct the exact locations of the enemy targets in an open half-plane, where the boundary passes through the radar station.

Every time when the radar scanner shows some locations of the exposed enemy targets, Little Q, the commander of our troops, would like to know the minimum area of the convex polygon such that each exposed enemy target lies either on the boundary of the polygon or inside it, so that he can estimate the price to surround all the exposed enemy targets.

Input

The first line of the input contains two integers n ($1 \le n \le 100\,000$) and q ($1 \le q \le 200\,000$), indicating the number of enemy targets and the number of times the radar scanner shows some locations of the exposed enemy targets.

Each of the following n lines contains two integers x and y ($-10^9 \le x, y \le 10^9$), indicating an enemy target located at coordinate (x, y).

Each of the following q lines contains two integers a and b ($-10^9 \le a, b \le 10^9, a^2 + b^2 > 0$), indicating that an enemy target located at coordinate (x, y) that satisfies $ax + by > 0$ will be exposed during the radar scan. Since it is a military manoeuvre, the exposed enemy targets will not be eliminated after the radar scan.

Output

Output q lines, each of which contains a single integer, indicating two times the minimum area of the desired convex polygon during the radar scan.

Example

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

Problem I. The Great Wall II

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Beacon towers are built throughout and alongside the Great Wall. There was once a time when there were n beacon towers built from west to east for defending against the invaders. The altitude of the i-th beacon tower, based on historical records, is a_i.

The defenders divide strategically all beacon towers into k parts where each part contains several, but at least one, consecutive beacon towers. To fully defend against the invaders, to each part a team of defenders should be assigned, the cost of which is given by the highest altitudes of beacon towers in that part.

As a historian, you are dying to know the minimum costs of assignments for every $k = 1, 2, \ldots, n$.

Input

The first line contains an integer n ($1 \le n \le 8\,000$), denoting the number of beacon towers alongside the Great Wall.

The second line contains n integers $a_1, a_2, \ldots, a_n$, where the i-th integer a_i ($1 \le a_i \le 100\,000$) is the altitude of the i-th beacon tower.

Output

Output n lines, the i-th of which contains an integer indicating the minimum cost for $k = i$.

Examples

standard input	standard output
5 1 2 3 4 5	5 6 8 11 15
5 1 2 1 2 1	2 3 4 6 7

Problem J. Colourful Journey

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Recently an epoch-making smart electric car developed by NIO company goes on sale. To celebrate, the chief test driver plans to drive the new car in Byteland for q days.

Byteland has n cities numbered from 1 to n, and n bi-directional roads connecting them. For each pair of cities, the driver can always arrive one from another one through these roads. There is a garage selling car paints of some colours on each road. Every time when the driver passes through a road, he will choose one colour of car paints from the garage and repaint the car if that colour is different from the car's current colour.

In each of the next q days, before the day's journey starts, the driver will paint the car with any one colour even if car paints of that colour are not on sale in any garages in Byteland. Then he will drive from the a-th city to the b-th city without visiting any cities more than once.

To make the car more colourful to attract everyone's attention, the driver wonders the maximum times that he repaints the car in each day.

Input

The first line contains two integers n ($2 \le n \le 200\,000$) and q ($1 \le q \le 200\,000$), indicating the number of cities in Byteland and the number of days of the chief test driver's journey in Byteland.

In each of the next n lines, three integers u, v ($1 \le u, v \le n, u \ne v$) and k (≥ 1) comes first, indicating that the u-th city and the v-th city is connected by a road, where the garage sells car paints of k colours. Then k distinct integers $c_1, c_2, ..., c_k$ ($1 \le c_i \le 500\,000$) follows, indicating the colours of car paints sold by the garage. It is guaranteed that the sum of k does not exceed $500\,000$.

Each of the next q lines contains two integers a and b ($1 \le a, b \le n, a \ne b$), indicating a journey from the a-th city to the b-th city.

Output

Output q lines, each of which contains a single integer, indicating the maximum times of repainting the car during the journey from the a-th city to the b-th city.

Examples

standard input	standard output
5 5 1 2 4 4 5 9 2 1 4 1 4 2 3 1 2 4 5 1 2 5 3 5 2 9 6 5 4 5 2 2 3 2 4 1 3 1 5	3 4 3 3 3
8 7 1 2 2 4 1 1 4 3 4 1 2 1 5 3 1 2 4 1 7 1 4 2 3 1 1 3 4 3 1 2 4 4 6 1 1 7 8 1 2 8 7 4 1 2 3 4 7 4 8 5 8 1 8	1 3 3 3 4 3 2

Problem K. NIO's OAuth2 Server

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

NIO owns an awesome API platform with various functionalities such as *"Graph APIs"*, *"Email APIs"* and *"Vision APIs"*. Today, there are lots of third party applications using NIO's APIs, to better manage the access permission of each API, NIO is developing his own OAuth2 service.

There are K functionalities of the API platform labeled with $1, 2, \ldots, K$. In order to call the APIs of the i-th functionality, an OAuth token with the i-th capability is needed.

NIO creates various templates for third party applications to use when doing authorization. If an application uses the j-th template to do authorization, a token with capabilities $C_j^1, C_j^2, \ldots, C_j^{M_j}$ is granted, using which, the application is able to call functionality $C_j^1, C_j^2, \ldots, C_j^{M_j}$.

Each third party application has its API requirements $R_1, R_2, \ldots, R_m$, which means it needs to call APIs of $R_1, R_2, \ldots, R_m$ functionalities. To achieve this, the application must do several authorizations using the pre-defined templates to get tokens with enough capabilities.

After defining N templates, NIO wants to know how the user experience of his templates is. Hence, he defines the degree of an application as the minimal authorizations it has to make to meet all its API requirements. If an application's API requirements cannot be satisfied even after using all the templates, just ignore it because NIO doesn't care.

As the intern of NIO, NIO wants you to tell him the number of applications with degree $1, 2, \ldots, K$ among all possible $2^K - 1$ applications. Two applications are consider different iff some of their API requirements are different.

Input

The first line contains two integers N ($1 \le N \le 100\,000$) and K ($1 \le K \le 20$).

In the i-th of the following N lines describes the i-th pre-defined template. The first integer M_i ($1 \le M_i \le K$) denotes the number of capabilities the i-th templates can grant, then M_i integers $C_i^1, C_i^2, \ldots, C_i^{M_i}$ ($1 \le C_i^1 < C_i^2 < \ldots < C_i^{M_i} \le K$) follows.

Output

Output one line containing K integers, the i-th of which is the number of different applications with degree i.

Example

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

Note

For the sample case, degrees of applications $\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}$ are all 1 while the degree of application $\{1, 2, 3\}$ is 2.