Problem A. Antiparticle Antiphysics

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

In an alternate universe, where the laws of physics are out of whack...

A new research facility has just been built. It is called the Large Antihadron Collider (LAC), the largest antiparticle collider of its kind, and antiphysicists are eager to use it to study something called "regular matter", which is similar to antimatter except with reversed charge, parity, and time.

In one of their LAC experiments, the antiphysicists successfully confined two kinds of particles, *antiprotons* and *protons*, in a container, where particles are lined up from left to right. We can represent the container's *state* as a 1-indexed string. The length of the string equals the number of particles in the container, and the i-th character of the string is `A` if the i-th particle from the left is an antiproton, or `P` if it is a proton.

Using the LAC's bizarro-energy beams, they can modify the state using any of four different types of operations:

Operation 1: Choose a particular proton, and then insert two antiprotons, one to its left and the other to its right. This has the effect of replacing the corresponding character `P` in the state string with `APA`.

Operation 2: Choose a particular antiproton, and then insert two protons, one to its left and the other to its right. This has the effect of replacing the corresponding character `A` in the state string with `PAP`.

Operation 3: Choose a contiguous subsequence of a antiprotons, and then remove them.

Operation 4: Choose a contiguous subsequence of p protons, and then remove them.

Note that the integers a in Operation 3 and p in Operation 4 are given in the input and are fixed.

These operations can be performed an arbitrary number of times in an arbitrary order, but only one operation can be performed at a time.

The *initial state* is represented by the string S. They would like to transform it into the *goal state* represented by the string E using a sequence of operations. Determine whether it is possible to do so. If it is possible, find one sequence of operations that transforms the initial state into the goal state.

Input

The first line of input contains one integer t ($1 \le t \le 10$) representing the number of test cases. After that, t test cases follow. Each of them consists of a single line containing two integers a and p ($5 \le a, p \le 20$) and two strings S and E ($1 \le |S|, |E| \le 50$, $S \ne E$). The strings S and E consist only of characters `A` and `P`.

Output

For each test case, output the following.

If the transformation is impossible, output `-1` in a single line.

Otherwise, on the first line, output an integer k representing the number of operations to transform the initial state into the goal state. On each of the next k lines, output one of the following, without the quotes, to describe one operation:

1. "`+P` i" to apply Operation 1 on the i-th particle from the left ($i \ge 1$). This particle must be a proton.

2. "`+A` i" to apply Operation 2 on the i-th particle from the left ($i \geq 1$). This particle must be an antiproton.
3. "`-A` i" to apply Operation 3 on the a consecutive particles whose leftmost particle is the i-th particle from the left ($i \geq 1$). These particles must be antiprotons.
4. "`-P` i" to apply Operation 4 on the p consecutive particles whose leftmost particle is the i-th particle from the left ($i \geq 1$). These particles must be protons.

These operations are performed in the order of the output lines and must transform the initial state into the goal state.

The number of operations k must satisfy $1 \leq k \leq 35\,000$. It can be shown that there's always a sequence of operations satisfying this bound for k if the initial state can be transformed into the goal state at all. Any valid sequence satisfying this bound for k will be accepted. In particular, you do not need to minimize the value of k.

Example

standard input	standard output
4 13 10 PP PAAAAPAAAA 10 13 AAAAAAA PPPPPPP 7 8 PPAAAAAAAAP PPAP 8 9 PAPPPPPPPPP PPAP	4 +P 2 +P 3 +P 4 +P 5 -1 1 -A 3 2 +A 2 -P 5

Note

Explanation for the sample input/output #1

In the first test case, the sequence of state string operations is
`PP` $\rightarrow$ `PAPA` $\rightarrow$ `PAAPAA` $\rightarrow$ `PAAAPAAA` $\rightarrow$ `PAAAAPAAAA`.

In the fourth test case, the sequence of state string operations is
`PAPPPPPPPPP` $\rightarrow$ `PPAPPPPPPPPPP`, then `PPAPPPPPPPPPP` $\rightarrow$ `PPAP`.

Problem B. Attraction Score

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

There are n cities, numbered from 1 to n, in the fictional country of Manteiv. We can consider these cities to be on a flat plane with a 2D coordinate system, where city i is at coordinates (x_i, y_i). No two cities are located at the same position.

There are m highways, numbered from 1 to m, each of which is a line segment with two different cities as its endpoints and has a number of attraction points alongside it. Specifically, highway j has a_j attraction points and connects cities u_j and v_j as its endpoints. Having intersections on highways causes traffic jams, and building a highway on top of another highway costs a lot of money. Therefore, it is guaranteed that

- no two highways intersect at any point other than at a city,
- no highway passes through a city other than its two endpoints, and
- there is at most one highway connecting each pair of cities.

The Manteiv Ministry of Tourism would like to choose a subset of cities as tourist attractions. Intuitively, the ministry would like many pairs of chosen cities to be connected by a highway with many attraction points. Formally, the *attraction score* of a non-empty subset of cities S is defined as follows:

- For every pair of integers (a, b) where $a < b$, cities a and b are in S, and they are connected by a highway, add the number of attraction points on the highway to the score.
- Let $f(S)$ be the number of pairs of integers (a, b) where $a < b$, cities a and b are in S, and they are **not** connected by a highway. The score incurs a penalty (negative) score of 10^6 multiplied by **the square of** $f(S)$. In other words, subtract $10^6 \times f(S)^2$ from the score.

For example, let $n = 3$, cities 1 and 2 be connected by a highway with 10 attraction points, cities 2 and 3 be connected by a highway with 20 attraction points, and cities 1 and 3 not be connected by a highway.

- The attraction score of the subset of cities $\{1\}$ is 0.
- The attraction score of the subset of cities $\{1, 2\}$ is $10 - 10^6 \times 0^2 = 10$.
- The attraction score of the subset of cities $\{2, 3\}$ is $20 - 10^6 \times 0^2 = 20$.
- The attraction score of the subset of cities $\{1, 2, 3\}$ is $10 + 20 - 10^6 \times 1^2 = -999\,970$.

As an advisor to the ministry, you would like to find the maximum attraction score among all possible non-empty subsets of cities S.

Input

The first line of input contains two integers n and m ($1 \le n \le 100\,000; 0 \le m \le 300\,000$). Each of the next n lines contains two integers. The i-th line contains x_i and y_i ($0 \le x_i, y_i \le 10^9$). Each of the next m lines contains three integers. The j-th line contains u_j, v_j, and a_j ($1 \le u_j < v_j \le n; 0 \le a_j \le 10^6$). The highways are guaranteed to satisfy the conditions in the problem statement.

Output

Output an integer representing the maximum attraction score among all possible non-empty subsets of cities S.

Examples

standard input	standard output
3 2 0 0 0 1 1 0 1 2 10 2 3 20	20
3 3 0 0 0 1 1 0 1 2 10 2 3 20 1 3 30	60

Note

Explanation for the sample input/output #1

This sample is the example given in the problem statement above. The subset of cities $\{2, 3\}$ gives the highest attraction score of 20.

Explanation for the sample input/output #2

The cities and highways are illustrated by Figure 6. By choosing cities 1, 2, and 3 in S, the attraction score would be $10 + 20 + 30 - 10^6 \times 0^2 = 60$.

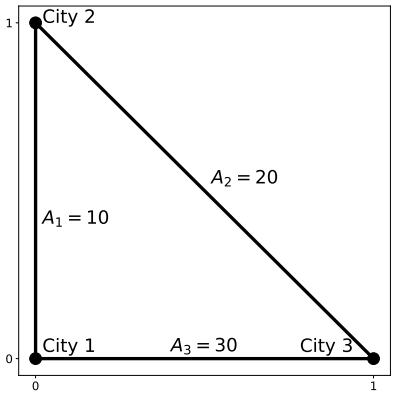

Рис. 1: Illustration of sample input #2.

Problem C. Bit Counting Sequence

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

For a non-negative integer x, let $p(x)$ be the number of ones in the binary representation of x. For example, $p(26) = 3$ because $26 = (11010)_2$.

You are given a sequence of n integers $(a_1, a_2, \ldots, a_n)$. Your task is to determine whether there exists a non-negative integer x such that $(p(x), p(x+1), ..., p(x+n-1))$ is equal to $(a_1, a_2, \ldots, a_n)$. Furthermore, if it exists, compute the smallest x satisfying the condition.

Input

The first line of input contains one integer t ($1 \le t \le 1000$) representing the number of test cases. After that, t test cases follow. Each of them is presented as follows.

The first line contains one integer n ($1 \le n \le 500\,000$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 60$ for all i).

The sum of n across all test cases in one input file does not exceed $500\,000$.

Output

For each test case, output the smallest non-negative integer x satisfying the condition above. If there is no such x, output `-1` instead.

Example

standard input	standard output
4 5 3 3 4 1 2 3 2 1 2 2 60 60 2 8 0	13 3 2305843009213693949 -1

Note

Explanation for the sample input/output #1

For the first test case, $x = 13$ satisfies the condition above since $(p(13), p(14), p(15), p(16), p(17)) = (3, 3, 4, 1, 2)$. It can be shown that there is no non-negative integer smaller than 13 that satisfies the condition above.

This page is intentionally left blank.

Problem D. Bánh Bò

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	1024 megabytes

Ever since the Earth got destroyed, Trillian has been missing some Earth delicacies. Today, she had the spaceship's food machine generate for her a Vietnamese delicacy she once enjoyed: bánh bò hấp *(steamed chewy sponge cake).*

Trillian has an unlimited number of *bánh bò hấp* pieces. Each piece of *bánh bò hấp* is either *red* or *white*. She wants to assemble rc pieces of *bánh bò hấp* into a grid with dimensions $r \times c$, where each cell contains a single piece of *bánh bò hấp*. Thus, there are exactly 2^{rc} distinct ways to assemble *bánh bò hấp* into an $r \times c$ grid, since we consider pieces of the same color to be identical.

We say an assembly of *bánh bò hấp* is *uniform* if all 6×7 subgrids have the same number of red pieces. Consequently, in a uniform *bánh bò hấp* assembly, all 6×7 subgrids have the same number of white pieces as well. Note that an $r \times c$ grid has $(r-5)(c-6)$ subgrids of dimensions 6×7.

For example, Figure 2 illustrates a uniform assembly of 7×8 pieces of *bánh bò hấp*, where shaded cells represent red *bánh bò hấp* pieces and unshaded cells represent white *bánh bò hấp* pieces. Figure 3 shows that all four 6×7 subgrids have 6 red pieces and 36 white pieces.

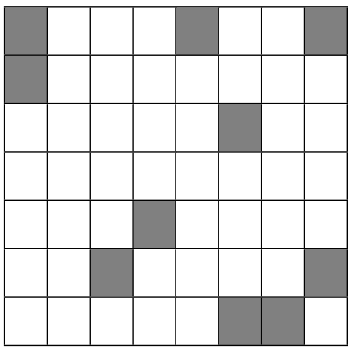

Figure 2: An example of uniform *bánh bò hấp* assembly.

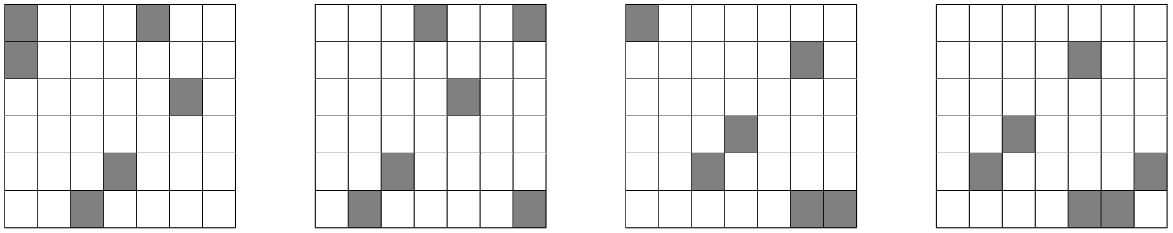

Figure 3: All four 6×7 subgrids of the uniform *bánh bò hấp* assembly illustrated in Figure 2.

Given r and c, where r **is a multiple of** 6 and c **is a multiple of** 7, Trillian would like to calculate the number of possible uniform *bánh bò hấp* assemblies modulo 998 244 353.

Input

Input consists of a single line containing two integers r and c ($6 \le r \le 66\,666$; r is a multiple of 6; $7 \le c \le 77\,777$; c is a multiple of 7).

Output

Output the number of possible uniform *bánh bò hấp* assemblies modulo 998 244 353.

Examples

standard input	standard output
6 7	780136139
12 14	22889737
12 42	96403614
42 14	94940316

Note

Explanation for the sample input/output #1

The output is 2^{42} modulo $998\,244\,353$.

Problem E. Duplicates

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

We say that a number sequence *contains duplicates* if there is an element that appears more than once in the sequence. Formally, a sequence $(a_1, \ldots, a_n)$ contains duplicates if there exist two indices i and j such that $i \neq j$ and $a_i = a_j$.

You are given an $n \times n$ matrix X. Each entry in X is an integer between 1 and n, inclusive. You can modify zero or more entries in X to arbitrary integers between 1 and n, inclusive. Different entries can be modified to different integers.

Your task is to make modifications to entries of X such that all of the following hold:

- For each row i, the sequence $(X_{i1}, X_{i2}, ..., X_{in})$ contains duplicates.
- For each column j, the sequence $(X_{1j}, X_{2j}, ..., X_{nj})$ contains duplicates.

Compute the minimum number of entries that need to be modified to achieve this. Also, find one possible set of modifications to do it. For each modification, you have to specify which entry will be modified and to what value. Note that the minimum number of entries to be modified can be zero when the given matrix X already satisfies the conditions above.

Input

The first line of input contains one integer t ($1 \leq t \leq 1000$) representing the number of test cases. After that, t test cases follow. Each of them is presented as follows.

The first line of a test case contains one integer n ($3 \leq n \leq 100$). Each of the next n lines contains n integers. The j-th integer in the i-th line denotes X_{ij} ($1 \leq X_{ij} \leq n$).

The sum of n^2 across all test cases in one input file does not exceed 10 000.

Output

For each test case, output a set of modifications in the following format.

On the first line, output an integer m representing the minimum number of entries that need to be modified. On each of the next m lines, output three integers i, j, and v. This represents a single modification where the entry X_{ij} will be modified to v. All of the three integers must be between 1 and n, inclusive.

If there are multiple solutions, you can output any of them.

Example

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

Note

Explanation for the sample input/output #1

In the first test case, the matrix after the modification is as follows.

$$\begin{bmatrix} 3 & 2 & 1 & 1 \\ \mathbf{1} & 1 & 3 & 4 \\ 1 & 3 & 3 & 1 \\ 4 & \mathbf{3} & 4 & 2 \end{bmatrix}$$

Problem F. Forming Groups

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

There are n students, numbered from 1 to n, who need to form groups for the upcoming hackathon. You are student 1, the captain of the students. Student i has *skill level* a_i.

Students 2 to n are standing in a line from left to right in order. You can choose to stand in between any two students, to the left of student 2, or to the right of student n. You cannot change the order of the $n-1$ students.

You can also choose the number of groups k ($k > 1$ and k must be a divisor of n) to participate in the hackathon. The groups will be numbered from 1 to k. After you have chosen your position and the value of k, the students will be grouped as follows:

- The first student from the left will be assigned to group 1.
- The second student from the left will be assigned to group 2.
- ...
- The k-th student from the left will be assigned to group k.
- The $(k+1)$-th student from the left will be assigned to group 1.
- The $(k+2)$-th student from the left will be assigned to group 2.
- ...
- The n-th student from the left will be assigned to group k.

Formally, for each j ($1 \le j \le k$) and for each i ($0 \le i < n/k$), the $(i \times k + j)$-th student from the left will be assigned to group j. It can be shown that each student will be assigned to exactly one group and all the groups have the same number of students.

The *skill level of a group* is the sum of the skill levels of the students inside the group. By choosing where you stand as well as the number of groups k optimally, you want to minimize the ratio $x_{\max}/x_{\min}$ where

- $x_{\max}$ is the skill level of the group with the largest skill level, and
- $x_{\min}$ is the skill level of the group with the smallest skill level.

Input

The first line of input contains one integer t ($1 \le t \le 100\,000$) representing the number of test cases. After that, t test cases follow. Each of them is presented as follows.

The first line of a test case contains two integers n and a_1 ($2 \le n \le 10^6; 1 \le a_1 \le 1000$). The next line contains $n-1$ integers $a_2, a_3, \ldots, a_n$ ($1 \le a_i \le 1000$ for all i).

The sum of n across all test cases in one input file does not exceed 10^6.

Output

For each test case, output one line containing two positive integers p and q such that the minimum ratio is p/q. The fraction p/q should be irreducible. In other words, p and q should be coprime.

Example

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

Note

Explanation for the sample input/output #1

In the first test case, by standing between students 2 and 3 (or between students 3 and 4) and choosing $k = 2$, group 1 will have the skill level $2 + 1$ and group 2 will have the skill level $1 + 2$, thus the ratio is $1/1$.

In the second test case, the only choice for the value of k is 3.

Problem G. Personality Test

Input file:	standard input
Output file:	standard output
Time limit:	5 seconds
Memory limit:	1024 megabytes

There are n students taking a personality test consisting of m questions. The students are numbered from 1 to n and the questions are numbered from 1 to m. For each question, each student can either answer it with a single uppercase Latin character (`A`–`Z`) or not answer it. Let S_i be a string of m characters representing the answers of student i, where the j-th character of S_i is an uppercase Latin character if they answered question j, or a period (`.`) if they did not.

Two students are considered *similar* if there is a set of at least k questions where both students answered all questions in the set, and for each question in the set, they answered it with the same answer.

For example, let $n = 3$, $m = 3$, $k = 2$, $S_1 =$ `BBC`, $S_2 =$ `..C`, and $S_3 =$ `.BC`. In this example, students 1 and 3 are similar since they answered questions 2 and 3 with the same answer, while students 2 and 3 are not similar since they answered only question 3 with the same answer.

You want to find a pair of integers (a, b) such that $a < b$ and students a and b are similar, or determine if there is no such pair. If there is more than one pair, find the one with the **smallest** b. If there is still more than one pair, find the one with the **largest** a.

Input

The first line of input contains three integers n, m, and k ($2 \le n \le 5000; 1 \le m \le 3000; 1 \le k \le 5$). Each of the next n lines contains a string of m characters. The i-th line contains the string S_i.

Output

Output one line containing the integers a and b representing the pair of similar students as mentioned in the problem statement, or just the integer `-1` if there is no such pair.

Examples

standard input	standard output
3 3 2 BBC ..C .BC	1 3
3 3 1 BBC ..C .BC	1 2
3 3 3 BBC ..C .BC	-1
4 12 2 GOOD.LUCK.IN WINNING.ICPC ASIA.PACIFIC CHAMPIONSHIP	2 3

Note

Explanation for the sample input/output #1

This is the example in the problem statement.

Explanation for the sample input/output #2

Students 1 and 2 are similar.

Explanation for the sample input/output #3

There is no pair of similar students.

Problem H. Pho Restaurant

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

As you may know, pho is one of the most common dishes in Hanoi. It contains a special kind of flour noodles, meat (usually beef or chicken), and green onions dipped in a tasty broth. Vietnamese people enjoy pho for breakfast, lunch, dinner, and even light meals. For tourists, trying pho is a must, especially in the cold of Hanoi.

You own a *phở bò* (beef pho) restaurant in Vietnam with n tables, numbered 1 to n. The 2024 ICPC Asia Pacific Championship contestants are currently in your restaurant. Each contestant is initially seated at one of the tables and there is at least one contestant initially seated at each table.

Each contestant would like to order one of the two most well-known kinds of pho: *phở tái* (pho with medium-rare beef) or *phở chín* (pho with well-done beef). The initial state of table i is represented by the binary string S_i. The length of S_i is the number of contestants initially seated at table i. The j-th character of S_i is `0` if the j-th contestant initially seated at the table would like to order a *phở tái*, and `1` if the contestant would like to order a *phở chín*.

To make it easier to track the orders, the restaurant wants the contestants seated at the same table to have the same order. That is, for each table, at least one of the following must be true:

- All of the contestants seated at that table would like to order a *phở tái*.
- All of the contestants seated at that table would like to order a *phở chín*.

To satisfy this requirement and the contestants' orders, you want to move zero or more contestants to a different table. The destination table must be one of the n tables. In other words, you must not add new tables. There is no limit to the number of contestants that can be seated at the same table. After moving the contestants, the following condition should be satisfied by each table: either there is no contestant seated at that table or all contestants seated at that table would like to order the same dish.

Since moving contestants takes some time, you would like to compute the minimum number of contestants you need to move.

Input

The first line of input contains one integer n ($2 \le n \le 100\,000$). Each of the next n lines contains a binary string. The i-th line contains S_i ($1 \le |S_i| \le 200\,000$). The sum of $|S_i|$ across all i does not exceed $500\,000$.

Output

Output an integer representing the minimum number of contestants you need to move.

Examples

standard input	standard output
4 11101101 00 10001 10	5
2 101010 010101	6
5 0000 11 0 00000000 1	0

Note

Explanation for the sample input/output #1

You can move

- the seventh contestant initially seated at table 1 to table 3,
- the fourth contestant initially seated at table 1 to table 4,
- the first and fifth contestants initially seated at table 3 to table 1, and
- the first contestant initially seated at table 4 to table 1.

You will then have all contestants seated at table 1 ordering *phở chín*, while the contestants seated at the other tables will be ordering *phở tái*. It can be shown that you cannot move fewer than 5 contestants to satisfy the requirements.

Problem I. Symmetric Boundary

Input file:	`standard input`
Output file:	`standard output`
Time limit:	11 seconds
Memory limit:	1024 megabytes

Symmetrical figures are beautiful—and they are the subject of this task. A region in a 2D plane is *convex* if, for every pair of points p and q in the region, the segment connecting p and q is entirely included in the region. Also, a region in a 2D plane is *point-symmetric* if, when you rotate the region by 180 degrees around a certain point, the rotated region exactly matches the original region.

You are given a convex polygon in a 2D plane with n vertices, numbered from 1 to n in counterclockwise order. Vertex i has coordinates (x_i, y_i). No three vertices are collinear. Determine whether there exists a convex, point-symmetric region containing all of the n vertices on its boundary. If one or more such regions exist, compute the minimum area among all of them.

Input

The first line of input contains one integer n ($3 \le n \le 30$). Each of the next n lines contains two integers. The i-th line contains x_i and y_i ($0 \le x_i, y_i \le 1000$).

It is guaranteed that the given polygon is convex, its vertices are given in counterclockwise order, and no three of its vertices are collinear.

Output

If one or more such regions exist, output the minimum area among all of them. The relative error of the output must be within 10^{-9}.

If such a region does not exist, output `-1` instead.

Examples

standard input	standard output
4 0 0 10 0 8 9 4 9	90.0
8 8 10 2 9 0 8 0 2 2 0 8 0 10 2 10 8	-1
6 231 77 359 20 829 124 998 461 941 735 879 825	486567.9669655848

Note

Explanation for the sample input/output

Figure 4 illustrates the vertices in the sample input as black dots. For sample inputs #1 and #3, the shaded regions represent the regions with the minimum possible area.

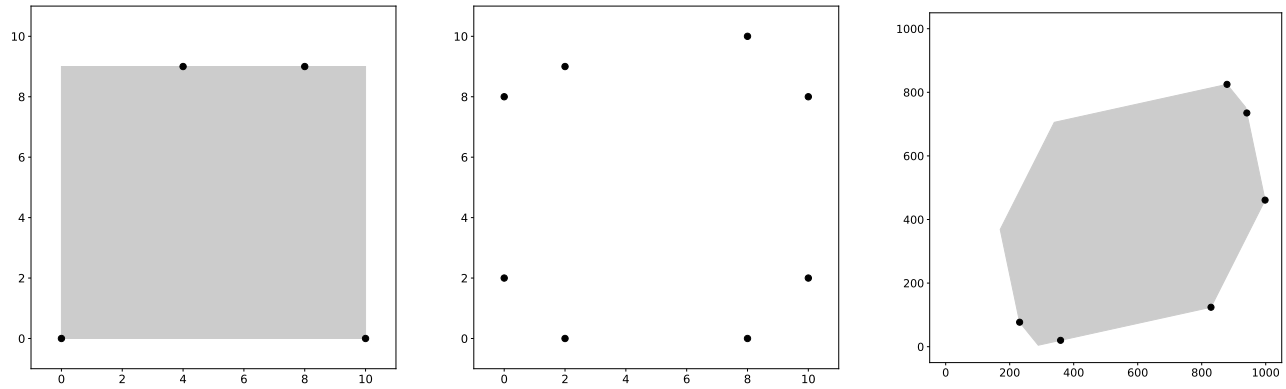

Рис. 4: Illustrations of the sample inputs (from left to right).

Problem J. There and Back Again

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

There are n cities in Asia-Pacific, numbered from 1 to n. The 2024 ICPC Asia Pacific Championship is held in Hanoi, which is city n.

There are m bidirectional roads, numbered from 1 to m, connecting some pairs of cities. Road i connects cities u_i and v_i and takes t_i units of time to travel in either direction. Each road connects different cities and different roads connect different pairs of cities.

You live in city 1. You would like to travel to city n to attend the contest through a sequence of roads, and then travel back to city 1 through a sequence of roads. Traveling through the same route is boring, so you would like the routes in both traversals to be different. Two routes are considered different if the **set of distinct** roads traversed through one route is different from the **set of distinct** roads traversed through the other route.

In each traversal, it is possible to pass through the same city or road multiple times. It is also possible to continue traversing after reaching the destination city (i.e., city 1 or city n). The *traversal time* is the sum of the travel times of the roads passed through in the traversal. If a road is passed through multiple times in the traversal, then the travel time of the road is also counted multiple times accordingly.

Determine the minimum total traversal time to do both traversals satisfying the requirements above, or indicate if the requirements cannot be satisfied.

Input

The first line of input contains two integers n and m ($2 \le n \le 100\,000; 1 \le m \le \min(\frac{n(n-1)}{2}, 300\,000)$). Each of the next m lines contains three integers. The i-th line contains u_i, v_i, and t_i ($1 \le u_i < v_i \le n; 1 \le t_i \le 1000$). Different roads connect different pairs of cities.

Output

Output an integer representing the minimum total traversal time to do both traversals satisfying the requirements above, or `-1` if the requirements cannot be satisfied.

Examples

standard input	standard output
3 2 1 2 10 1 3 5	30
4 3 1 2 10 2 3 5 3 4 2	-1
4 4 1 2 3 2 4 2 1 3 3 3 4 4	12
3 1 1 2 1000	-1

Note

Explanation for the sample input/output #1

The cities and roads are illustrated by Figure 7.

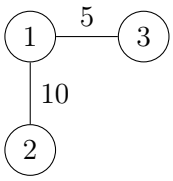

Рис. 5: Illustration of sample input #1.

One possible way to minimize the total traversal time is as follows:

- Travel from city 1 to city 3 by passing through road 2 (connecting cities 1 and 3). The traversal time is 5. The set of traversed roads is $\{2\}$.
- Travel from city 3 to city 1 by passing through road 2, and then road 1 (connecting cities 1 and 2) twice. The traversal time is $5 + 10 + 10 = 25$. The set of traversed roads is $\{1, 2\}$.

It can be shown that there is no way to do both traversals with a smaller total traversal time.

Explanation for the sample input/output #2

The cities and roads are illustrated by Figure 6.

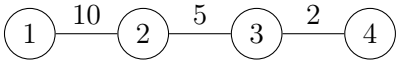

Рис. 6: Illustration of sample input #2.

To go from city 1 to city 4 and then back, both traversals have to pass through all the roads. Therefore, it is impossible to satisfy the requirements above.

Problem K. Tree Quiz

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

Your friend wants to quiz you. You are given a *rooted tree* with n nodes, numbered from 1 to n. For every node i, its parent is node p_i, except for the *root* (the node without a parent) which has $p_i = 0$. Node u is an *ancestor* of node v if either $u = v$, or node u is an ancestor of the parent of node v (if it exists).

We say that node z is a *common ancestor* of nodes x and y if node z is an ancestor of both nodes x and y. We say that node z is the *lowest common ancestor* of nodes x and y if it is a common ancestor of nodes x and y, and every common ancestor of nodes x and y is also an ancestor of node z. We denote the lowest common ancestor of nodes x and y by $\mathrm{LCA}(x, y)$. In particular, $\mathrm{LCA}(x, x) = x$.

Your friend would like to run the following pseudocode:

```
let L be an empty array
for x = 1 to n
  for y = 1 to n
    append ((x - 1) * n * n + (LCA(x, y) - 1) * n + (y - 1)) to L
sort L in non-decreasing order
```

Your friend has q questions, numbered from 1 to q. In question j, you are given an integer k_j and asked to find the k_j-th element of the array L. Note that L is 1-indexed, so the indices range from 1 to n^2, inclusive. To pass the quiz, you have to answer all of the questions.

Input

The first line of input contains two integers n and q ($1 \le n \le 100\,000; 1 \le q \le 100\,000$). The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le n$ for all i). It is guaranteed that the given values represent a rooted tree. Each of the next q lines contains an integer. The j-th line contains k_j ($1 \le k_j \le n^2$).

Output

For each question in order, output an integer representing the answer to the question.

Example

standard input	standard output
5 3 3 0 2 2 3 1 18 25	0 82 124

Note

Explanation for the sample input/output #1

The tree in the input is illustrated by Figure 7.

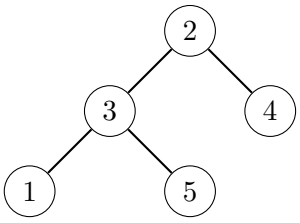

Рис. 7: Illustration of the tree in sample input #1.

The elements of L are
$(0, 6, 8, 12, 14, 30, 31, 32, 33, 34, 56, 58, 60, 62, 64, 80, 81, 82, 84, 93, 106, 108, 110, 112, 124)$.

Problem L. XOR Operations

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

You are given n integers $a_1, a_2, \ldots, a_n$. You have a sequence of n integers $B = (b_1, b_2, \ldots, b_n)$ which initially are all zeroes.

In one operation, you choose two different indices i and j, then simultaneously

- replace b_i with $b_i \oplus a_i \oplus a_j$, and
- replace b_j with $b_j \oplus a_i \oplus a_j$.

Note that $\oplus$ represents the bitwise XOR operation, which returns an integer whose binary representation has a 1 in each bit position for which the corresponding bits of either but not both operands are 1. For example, $3 \oplus 10 = 9$ because $(0011)_2 \oplus (1010)_2 = (1001)_2$.

You want to compute the number of different possible sequences B you can obtain after performing zero or more operations. Since this number might be huge, calculate this number modulo 998 244 353.

Two sequences of length n are considered different if and only if there exists an index i ($1 \le i \le n$) such that the i-th element of one sequence differs from the i-th element of the other sequence.

Input

The first line of input contains one integer n ($2 \le n \le 200\,000$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$ for all i).

Output

Output an integer representing the number of different possible sequences B you can obtain after performing zero or more operations modulo 998 244 353.

Examples

standard input	standard output
3 1 2 1	4
4 852415 852415 852415 852415	1

Note

Explanation for the sample input/output #1

Starting from $B = (0, 0, 0)$, we can obtain the following two sequences B:

- Perform the operation with $i = 1$ and $j = 2$. We will have $B = (3, 3, 0)$.
- After that, perform the operation with $i = 2$ and $j = 3$. We will have $B = (3, 0, 3)$.

Starting from $B = (0, 0, 0)$, we can also obtain the following sequence B:

- Perform the operation with $i = 2$ and $j = 3$. We will have $B = (0, 3, 3)$.

It can be shown that $(0, 0, 0)$, $(3, 3, 0)$, $(3, 0, 3)$, and $(0, 3, 3)$ are the only possible sequences B you can obtain. Therefore, the answer is 4.

This page is intentionally left blank.

Problem M. Zig-zag

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	1024 megabytes

Zack's Zergonomics Zegree has taught him that the optimal way to display items in a store is to stack them into a zig-zag pattern.

Zack needs to display n boxes lined up on the storefront, each one containing an action figure. These boxes can be stacked on top of one another, and they are identical and indistinguishable from each other. His goal is to decide the number of stacks, and then stack up the boxes such that each stack is non-empty, and the numbers of boxes in the stacks form a *zig-zag* sequence.

Formally, if there are s ($s \geq 1$) stacks numbered 1 to s from left to right, and stack i contains a_i boxes, then the following conditions must be satisfied:

- $a_i \geq 1$ for each i from 1 to s,
- $a_1 + a_2 + \ldots + a_s = n$, and
- at least one of the following is true:
  - $a_1 < a_2 > a_3 < a_4 > \ldots$, or
  - $a_1 > a_2 < a_3 > a_4 < \ldots$

For example, for $n = 6$, there are 12 ways as illustrated by Figure 8.

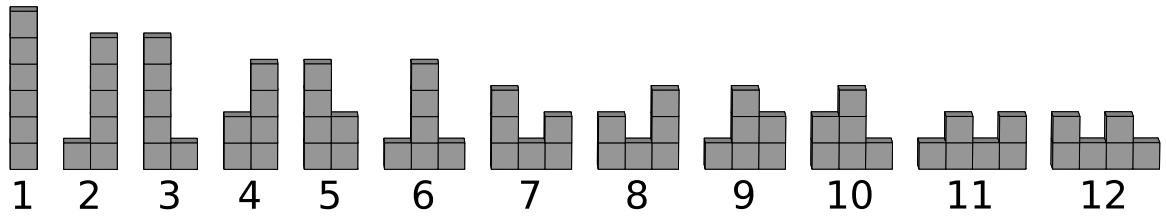

Рис. 8: All 12 possible ways for $n = 6$.

Find the number of different ways Zack can stack n boxes modulo 998 244 353.

Two ways are considered the same if and only if the number of stacks is the same, and pairs of stacks at the same positions have the same number of boxes.

Input

The first line of input contains one integer t ($1 \leq t \leq 300\,000$) representing the number of test cases. After that, t test cases follow. Each of them consists of a single line containing one integer n ($1 \leq n \leq 300\,000$).

Output

For each test case, output an integer representing the number of different ways to stack n boxes modulo 998 244 353.

Example

standard input	standard output
4 5 6 7 890	7 12 19 502674609

Note

Explanation for the sample input/output #1

The value of n on the second test case is 6, and the 12 ways are illustrated in the problem description.