Problem A. Acoustic String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A few people know this but binary strings can be *acoustic*. It's quite easy to check if a binary string is acoustic: you just have to make it resonate with itself.

In the process of resonance a binary string s of length n is changed as follows: while $n > 1$, a new string s^* of length $n - 1$ is constructed, where $s_i^* = s_i \oplus s_{i+1}$ for all i from 1 to $n - 1$, where $\oplus$ denotes the "exclusive OR" operation. In other words, each bit of a new string is a sum of two consecutive bits of s modulo 2.

The string is considered acoustic if after $n - 1$ iterations of resonance, when its length becomes 1, the only remaining bit is equal to 1. Find out if a given string is acoustic or not.

Input

A single line contains a binary string s consisting of characters '`0`' and '`1`' ($1 \le |s| \le 10^6$).

Output

Output a single line with the answer 1 if the string is acoustic; otherwise, output 0.

Examples

standard input	standard output
1010	0
1101	1

Note

In the first example, the string changes as follows: $1010 \to 111 \to 00 \to 0$.

In the second example, the string changes as follows: $1101 \to 011 \to 10 \to 1$.

Problem B. Breaking the Code

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

While playing a game you noticed that there are several easter eggs hidden around it. They definitely must lead to some great item!

You've managed to crack most of the codes in those easter eggs and only the last one remains. You are left with a string s and a password for a hidden room may be obtained from it by performing any of the following actions several times:

- Remove the first letter of the current string;
- Remove the second letter of the current string;
- Remove the second-to-last letter of the current string;
- Remove the last letter of the current string.

While communicating with the in-game AI assistant, you've learned that the password has a length of k and is also the lexicographically smallest word of length k among all that can be obtained from s in the described manner.

Find the password: output the lexicographically minimal string of length k that can be obtained from s by those four actions.

Input

The first line contains the string s, consisting of lowercase Latin letters ($1 \le |s| \le 500\,000$).

The second line contains a natural number k — the length of the password ($1 \le k \le |s|$).

Output

Output the string — the required password.

Examples

standard input	standard output
abacaba 3	aaa
qwerty 2	er

Problem C. Catch the Animals

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

Let's consider a square area of the plane containing all points with coordinates that do not exceed 10^9 in absolute value. There are n points in this area with coordinates (x_i, y_i) representing highly dangerous animals that escaped the zoo.

They can't escape the area but they can harm each other. To prevent this, you need to separate them from each other using laser walls, which we will represent by infinite straight lines intersecting the area.

The sizes of the animals and the thickness of the walls can be ignored. The only restriction is that a laser may not pass exactly through an animal. The goal is to use the minimal number of walls to split the whole area into polygons such that there is no more than one animal in each of them.

Find an optimal way to achieve that.

Input

The first line contains one integer n — the number of animals ($1 \le n \le 12$).

The next n lines contain two integers x_i and y_i each — the coordinates of the point where the i-th animal is located ($|x_i|, |y_i| \le 1\,000$). It is guaranteed that no two points coincide.

Output

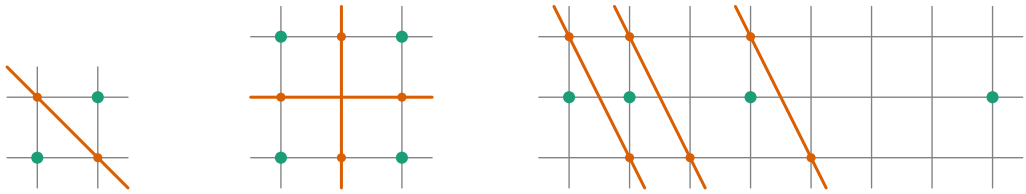
In the first line, output one integer m — the minimum number of walls that you have to create.

In each of the next m lines, output four integers $x_{j,1}$, $y_{j,1}$, $x_{j,2}$, and $y_{j,2}$ — the coordinates of two distinct points $(x_{j,1}, y_{j,1})$ and $(x_{j,2}, y_{j,2})$ through which the j-th wall should pass ($|x_{j,1}|, |y_{j,1}|, |x_{j,2}|, |y_{j,2}| \le 10^9$). The lines corresponding to the walls must not contain points corresponding to the animals. You may output any answer that contains the minimum possible number of walls.

Examples

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

Note



(a) First test (b) Second test (c) Third test

Рис. 1: Explanation of example tests

The positions of the animals are marked in green, the walls and the points they pass through are marked in red.

Problem D. Devilish Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You play a number game that your suspicious opponent recently learned about. The problem is: if you lose, you may also lose your soul, so it's quite important to win!

The rules of the game are quite simple: there are two positive integers A and B and a number k. Players take turns, starting with you. On their turn, a player can perform exactly one of two actions:

- Subtract an integer from 1 to k from the number A;
- Add an integer from 1 to k to the number B.

At the same time, during the game, the numbers A and B must remain positive; if a player makes a move that causes one of the numbers to become non-positive, they lose. The player who makes a move such that the sum of the numbers A and B becomes a prime number wins the game.

Can you win regardless of your opponent's actions?

Input

Each test contains several sets of input data. The first line of input contains a single integer t — the number of sets of input data ($1 \le t \le 10$).

The following t lines describe the sets of input data: each consists of one line containing three integers A, B, k — the numbers A and B at the start of the game and the number k ($1 \le A, B, k \le 10^9$).

It is guaranteed that $A + B$ is initially not a prime number.

Output

For each set of input data, output on a separate line "`Yes`" if you can win with the given initial data, and "`No`" otherwise.

Example

standard input	standard output
4 1 7 2 3 5 2 3 6 1 253309356 182963670 154154154	No Yes No Yes

Problem E. Equinox

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A nameless country consists of n cities connected by m bidirectional roads, such that it is possible to reach any city from any other.

There are three factions controlling this country: i-th faction occupies some connected set of cities $\{a_{i,j}\}$. That is, if we leave only the cities controlled by a specific faction, along with the roads between them, there will be a path between any two of them. It is also known that no city is controlled by more than one faction, meaning $a_{i,p} \neq a_{j,q}$ for $i \neq j$.

To celebrate the Equinox you need to mark some cities as main tourist attractions for the celebration. For all the factions so be satisfied, these cities along with factions' cities should form a connected set. Formally, it is necessary to choose a set of cities B such that any two cities from the set $S = B \cup \{a_{i,j}\}$ are connected by a path that passes only through cities from S.

Of course, the most obvious way is to mark all the cities in the country, but the government will not be satisfied with such a solution. Find a suitable set B of minimal size.

Input

The first line contains two integers n and m — the number of cities and roads in the country ($1 \leq n, m \leq 2 \cdot 10^5$).

The next 6 lines describe the factions: two lines for each. The first line of the description of the i-th faction contains integer k_i — the number of cities that it controls ($1 \leq k_i \leq n-2$). The second line lists k_i distinct integers $a_{i,j}$ — the cities controlled by the i-th faction ($1 \leq a_{i,j} \leq n$).

It is guaranteed that each number from 1 to n appears among $a_{i,j}$ no more than once.

In the next m lines, the i-th line contains two integers u_i and v_i — the cities connected by the i-th road ($1 \leq u_i, v_i \leq n$).

It is guaranteed that it is possible to reach any city from any other using the given roads.

Output

In the first line, output an integer b — the minimum number of cities that you need to mark with for celebration. If the factions are already connected and no cities need to be marked, output 0.

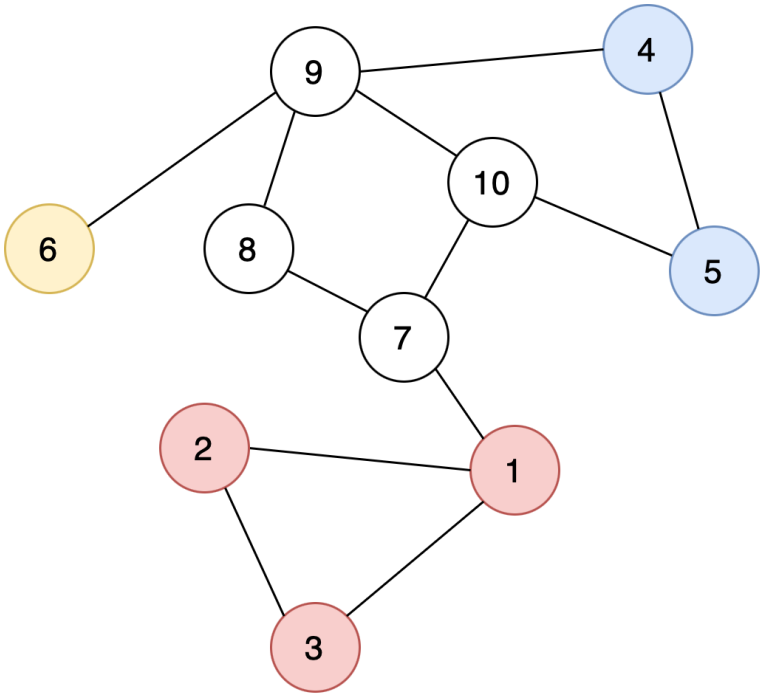
In the second line, output b distinct integers from 1 to n — the numbers of these cities. No city controlled by a faction should be marked.

Examples

standard input	standard output
4 3 1 1 1 2 1 3 4 1 4 2 4 3	1 4
10 12 3 1 2 3 2 4 5 1 6 1 2 2 3 1 3 4 5 1 7 7 8 8 9 9 6 9 10 7 10 10 5 4 9	3 9 7 10

Note

An illustration for the second example from the statement is provided in the figure below. Suitable answers include cities 7, 8, 9 or 7, 9, 10.


9
4
10
6
8
5
7
2
1
3

Problem F. Famous Smoothie

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Making the smoothie is a huge quest, during which you select n ingredients, represented by numbers from 0 to 10^9, and arrange them in a sequence a_i. Repeating ingredients is allowed, meaning there could be i and j such that $a_i = a_j$.

Then you need to find the secret ingredient, the number of which is calculated as follows:

1. For each i from 1 to n, for the sequence $a_1, \ldots, a_{i-1}, a_i$, determine b_i — the smallest number that is not present in this sequence (denoted as $\text{mex}(a_1, \ldots, a_i)$). For example, $\text{mex}(1, 7, 2, 2, 0, 5, 4) = 3$ and $\text{mex}(1, 4, 5, 2, 3) = 0$.
2. From the resulting sequence $b_1, \ldots, b_n$, compute the mex again.
3. The final value $\text{mex}(b_1, \ldots, b_n)$ is taken as the number of the secret ingredient.

Obviously the order of these ingredients can change the number of the secret ingredient. It is known that the higher this number is, the more delicious the smoothie made with it will be.

Arrange the selected ingredients so that the corresponding number of the secret ingredient for the resulting sequence is as large as possible.

Input

The first line of input contains a single integer n — the number of ingredients you have ($1 \le n \le 2 \cdot 10^5$).

The second line lists n integers from 0 to 10^9 — the numbers of these ingredients.

Output

Output a single integer — the maximum possible number of the secret ingredient.

Examples

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

Note

In the first example, one way to arrange the ingredients looks like $a = [5, 0, 1, 2, 1]$. In this case, we get $b = [0, 1, 2, 3, 3]$, the mex of which is 4.

Problem G. Great Ascent

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Have you ever played Skyrim? As any self-respecting Dragonborn, you cannot afford to embark on the main quest line. Therefore, by the time you decide to ascend to the monastery of High Hrothgar at the top of a mountain with a height of n meters, you have probably already managed to learn several dragon shouts and accumulate an impressive set of side quests.

Among these quests, there are also $m + q$ quests that you can complete on your way to Hrothgar and back: for the i-th of them, you need to ascend to a height of a_i meters from the foot of the mountain, pick up an item from one NPC, and deliver it to another NPC at a height of b_i from the foot of the mountain. At the same time,

- you spend t_u time to ascend one meter;
- you spend t_d time to descend one meter;
- no time is spent to ~~listen to~~ skip the dialogue when you pick up or hand over a quest item.

Of these $m + q$ quests, you definitely intend to complete the first m, while the remaining q are optional. Before starting your ascent, you can choose any i from 0 to q and activate the first i of the remaining quests in addition to the m already active ones. Then you can begin your ascent.

During the ascent, you can change your direction of movement as many times as you want and hold any number of quest items in your inventory at the same time. As soon as all $m + i$ quests are completed, you can immediately stop, call a passing dragon, and fly away on it to explore the ruins.

Note that it is **not required** to visit Hrothgar at a height of n at least once during the completion of the quests — we understand that while completing such a number of side quests, it is quite easy to forget about the main plot again. It is also not required to descend back to a height of 0 after completing the quests.

For each i from 0 to q, output the minimum time required to complete the m active quests and the first i of the additional quests, starting from the foot of the mountain at a height of 0.

Input

The first line of input contains an integer n — the height at which the top of the mountain is located ($1 \le n \le 10^9$).

The second line contains two integers t_u and t_d — the time to ascend one meter and the time to descend one meter ($1 \le t_u, t_d \le 10^9$).

The third line contains an integer m — the number of active quests ($1 \le m \le 2 \cdot 10^5$). In the i-th of the following m lines, two integers a_i and b_i are given — the heights of the starting and ending points of the i-th quest ($0 \le a_i, b_i \le n$).

Then an integer q is given — the number of additional quests ($0 \le q \le 2 \cdot 10^5$), followed by q lines describing these quests in the same format.

Output

In the first line, output the minimum time required to complete the m active quests. Then output q more lines, in the i-th of which output the answer to the problem if the first i quests from the remaining q are activated before the ascent.

Example

standard input	standard output
20 1 1 4 1 10 11 20 20 16 15 11 2 5 1 10 6	29 37 39

Problem H. Harmony of Skills

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3.5 seconds
Memory limit:	512 megabytes

In any strategy game, including Civilization, there are many skills that a player can learn and acquire.

Each of the n skills in Civilization has a *type* — military, cultural, construction, and so on. We denote the type of the i-th skill with the integer c_i. The skills are arranged in a rooted tree: each skill has one predecessor that must be learned to gain access to it. The process of learning skills starts from the root of the tree (the initial skill), the learning of which is necessary to unlock all other skills.

We say that two skills u and v are *similar* if the multisets of skill types in their subtrees are the same. In other words, for two skills to be similar, there must be an equal number of skills of each type in their subtrees.

"Civilization VII" has not yet been released, and the developers are testing various hypotheses about how the skill tree should look. To do this, they perform the re-suspension of the tree from another vertex as root q times and calculate the number of pairs of similar skills. When re-suspending the tree at skill x, it becomes the root (the first to be learned), while all other edges of the tree remain, but the direction of dependency for some of them changes accordingly.

For each of the q re-suspension queries, calculate the number of unordered pairs of similar skills after the query is executed. Unordered pairs means that the pair (u, v) and the pair (v, u) are considered the same. Since the answer may be large, output it modulo $10^9 + 7$.

Input

Each test contains several sets of input data. The first line of the input contains one integer t — the number of sets of input data ($1 \le t \le 2 \cdot 10^5$). The description of the input data sets follows.

In the first description of the input data set, an integer n is given — the number of skills in the game ($2 \le n \le 2 \cdot 10^5$).

In the second line, n integers c_i are listed — the types of skills ($1 \le c_i \le 10^9$).

In each of the next $n - 1$ lines, two integers u_i and v_i are given — the numbers of skills, one of which directly depends on the other ($1 \le u_i, v_i \le n$). The dependency structure is determined by which skill is chosen as the "root". It is guaranteed that the dependency structure forms a tree.

In the next line, an integer q is given — the number of re-suspension queries for the skill tree ($1 \le q \le n$).

The last line of the input contains q integers x_i — the numbers of the vertices from which the tree is re-suspended ($1 \le x_i \le n$).

It is guaranteed that the sum of n across all input data sets does not exceed $2 \cdot 10^5$.

Output

For each query, output the number of pairs of similar skills in a separate line, modulo $10^9 + 7$.

Examples

standard input	standard output
1 7 1 1 2 4 3 3 4 1 2 1 3 2 4 4 5 3 6 3 7 7 1 2 3 4 5 6 7	1 1 1 1 0 0 1
1 7 1 2 2 3 3 3 3 1 2 1 3 2 4 4 5 3 6 3 7 7 1 2 3 4 5 6 7	4 3 3 3 1 1 1

Problem I. Intelligent Amulet

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

This is an interactive problem.

Recently, you've heard a legend in a tavern about an abandoned shrine of an old cult, where the Intelligent Amulet may be hidden. It's an artifact that grants its owner the ability to use two unique spells. Naturally, the search for the Amulet immediately became the next priority quest.

Through overgrown paths, you reached the ruins of the shrine. Behind a secret passage, revealed after disarming a clever trap, there was an altar where a mysterious voice informed you that you must now prove your sharp wit by solving a riddle to obtain the Amulet. Specifically, you must guess two numbers l and r that this voice has in mind (it is known that $l \le r$).

Since guessing random numbers without any additional information is more of a test of luck than wisdom, the voice allowed you to make m queries of the form "replace l and r with $l + d_l$ and $r + d_r$", where d_l and d_r are arbitrary numbers you choose. After each query, the voice will inform you of the number of prime numbers in the segment from $\min(|l|, |r|)$ to $\max(|l|, |r|)$ inclusive.

At any moment, you can state the current values of l and r. If you guessed correctly, you will receive the Intelligent Amulet. Otherwise, the path to it will be closed to you forever. Obtain the Amulet!

Input

Each test contains several sets of input data. The only line of input contains a single integer t — the number of sets of input data ($1 \le t \le 500$). For each set of input data, an interaction process with the interactor begins.

Interaction Protocol

Interaction with the interactor occurs in the form of queries from your program and responses from the interactor. You can perform the action described in the statement no more than 26 times.

To make a query, output the string "`?` d_l d_r" ($-2 \cdot 10^6 \le d_l, d_r \le 2 \cdot 10^6$), after which l and r will change to $l + d_l$ and $r + d_r$, respectively. The interactor will respond with a separate line containing the number of prime numbers in the segment $[\min(|l|, |r|), \max(|l|, |r|)]$ for the new values of l and r.

To output the answer to the problem, output the string "`!` l r", where l and r must be the **current** boundaries. This output does not count towards the number of queries. In response, the interactor will give a verdict: `1` if your guess is correct and `0` otherwise. If your answer is incorrect, your solution will receive a verdict of `WA` (Wrong Answer), and the interactor will terminate. To avoid receiving incorrect verdicts, your solution must also terminate upon receiving '`0`'.

It is guaranteed that the original l and r satisfy the condition $1 \le l \le r \le 10^6$.

If at any point your program exceeds the limit of 26 queries, or the values of l or r after the next query exceed $2 \cdot 10^6$ in absolute value, your program will terminate with a verdict of `WA`.

Important: Do not forget to flush the output buffer after each output line so that the interactor receives your query. This can be done using `std::cout.flush()` in C++, `System.out.flush()` in Java, and `sys.stdout.flush()` in Python, as well as similar commands in other languages. If your program does not flush the output buffer, it will receive a verdict of `TL` (Time Limit Exceeded) or `IL` (Idleness Limit Exceeded).

Example

standard input	standard output
1 2 1 1 2 1	 ? 0 0 ? 0 -1 ? 1 1 ? 2 5 ! 5 8

Problem J. Jester's Scroll

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

This is a run-twice problem. Your solution will be executed twice.

At the gaming table, there are four players: the game master, you (the paladin), and two of your friends — the wizard and the rogue. Your party is currently taking inventory; it turns out that dozens of unnecessary items you picked up along the way are cluttering your backpacks, making it difficult to move efficiently through the forest.

In your backpack, among the loot, the wizard found a scroll with the squad's expense report that he kept for the last n days (you suspected that the rogue secretly puts a couple of extra coins in his pocket). It is known that

- your squad spends gold very responsibly, so the same number of coins was spent each of the n days;
- all expenses are divided into three categories: food, equipment, and tavern parties.

In other words, the scroll recorded three arrays of non-negative integers a (food expenses), b (equipment expenses), and c (party expenses), each of length n, and it is known that $a_i + b_i + c_i = a_j + b_j + c_j$ for all i and j.

Upon closer inspection of the scroll, the wizard noticed that something was wrong: the arrays were of different sizes, and the condition for maintaining the sum of the corresponding elements of the three arrays was no longer satisfied. By casting the spell "Detect Magic" and rolling a die, the wizard determined that this was a "Jester's Compression Scroll", which removes consecutive identical elements from the arrays. For example, if the scroll recorded $a = [2, 2, 2, 1, 5]$, $b = [3, 3, 1, 2, 3]$, and $c = [4, 4, 6, 6, 1]$, then after some time they would turn into $a' = [2, 1, 5]$, $b' = [3, 1, 2, 3]$, and $c' = [4, 6, 1]$.

Your character also does not fully trust the wizard. Therefore, you randomly found a note with reduced information about the expenses S, consisting of no more than k integers from 0 to $2^{30} - 1$. Surprisingly, this information was enough to reconstruct what the original a, b, and c could have looked like.

What information should S contain so that given n, a', b', and c', it would be possible to reconstruct some a, b, and c that satisfy the condition?

Input

In the first line of input, there is a single integer 1 or 2 — the run number.

First Run

In the second line of input, two integers n and k are given — the number of days in the report and the limit on the length of S ($1 \le n \le 30,000$; $1 \le k \le 1000$).

In the third line, n integers a_i are listed — the squad's food expenses for each day ($0 \le a_i < 2^{20}$). In the fourth and fifth lines, the same format lists b_i and c_i — the expenses for equipment and parties ($0 \le b_i, c_i < 2^{20}$). It is guaranteed that $a_i + b_i + c_i$ is a constant value.

Second Run

In the second line of input, five integers n, m_s, $m_{a'}$, $m_{b'}$, and $m_{c'}$ are given — the original n from the first run, the length of the sequence S output by your solution in the first run, and the lengths of a', b', and c' ($1 \le m_{a'}, m_{b'}, m_{c'} \le n \le 30,000$; $1 \le m_s \le k$).

In the next four lines, the elements of S, a', b', and c' are listed respectively. It is guaranteed that S was obtained in the first run of your solution, and that a', b', and c' were obtained from a, b, and c for the first run by removing consecutive identical elements.

Output

First Run

In the first line, output an integer m_s from 1 to k — the length of the sequence S that you will write down and put in your pocket.

In the second line, output m_s integers from 0 to $2^{30} - 1$ — the elements of S separated by spaces.

Second Run

Output three lines, each containing n integers from 0 to $2^{20} - 1$ — the elements of arrays a, b, and c, respectively.

If there are multiple possible suitable answers, output any of them — it is not necessary to restore the exact a, b, and c that were given in the input during the first run.

Interaction Protocol

To avoid receiving incorrect verdicts such as `Idleness Limit Exceeded` or `Security Violation`, end the output of each line with a newline character ('`\n`') and flush the output buffer (`cout.flush()`, `sys.stdout.flush()`, `System.out.flush()`).

Scoring

It's guaranteed that it's always possible to restore fitting a, b and c with S of length not exceeding k without considering special constraints and corner cases. Your solution will be judged as correct if its $|S|$ doesn't exceed k on every test.

Tests are split into several groups with following constraints:

Tests	Additional Constraints
3 – 5	$n \leq 3$; $k = 1$
6 – 9	$n \leq 10$; $k = 1$
10 – 15	$n \leq 100$; $k = 4$
16 – 21	$n \leq 1000$; $k = 1000$
22 – 28	$a_i = 0$ for all i; $k = 1$
29 – 48	$k = 1000$

Examples

standard input	standard output
1 5 15 2 2 2 1 5 3 3 1 2 3 4 4 6 6 1 2 5 3 3 4 3 3 2 4 2 1 5 3 1 2 3 4 6 1	 3 3 2 4 2 2 1 5 5 3 1 2 3 3 4 6 6 1 1
1 8 24 4 5 7 5 7 9 7 7 3 3 2 4 4 3 5 6 7 6 5 5 3 2 2 1 2 8 1 7 6 6 11347 4 5 7 5 7 9 7 3 2 4 3 5 6 7 6 5 3 2 1	 1 11347 4 5 7 5 7 9 7 7 3 3 2 4 4 3 5 6 7 6 5 5 3 2 2 1

Problem K. Killer Tetris

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

The game of Tetris can be represented as coloring cells black on an initially white table of size $n \times m$. Each game block is a "stamp", which takes the form of an inverted histogram of width 3, where each column does not exceed a height of 3. In other words, a stamp consists of three columns of cells with heights ranging from 1 to 3, arranged consecutively and having a common upper boundary.

These stamps can be printed on the table according to the following rules:

- The bottom boundary of the block must touch either the upper boundary of another block or the bottom boundary of the table;
- At the moment of adding a new block, there must not be any colored cells above the cells of the added block in the same columns;
- If at any moment a row becomes completely filled, it is immediately discarded; all cells in that row are removed, and all colored cells above that row fall down one row.

In other words, if a new block needs to be placed in columns x_i, x_i+1, x_i+2, one must look at the highest colored cell in each of these columns and position the block so that it touches at least one of them and does not overlap with the others.

The process stops as soon as the next block, when placed on the field according to the described rules, goes beyond the upper boundary of the field.

You need to simulate the game: you are given a sequence of actions, and you are required to output the number of the block at which the process will stop.

Input

The first line of input contains three integers n, m, and q — the number of rows and columns on the field, and the number of blocks to be placed on it ($1 \le n, m, q \le 2 \cdot 10^5$; $3 \le m$).

Each of the following q lines contains four integers x_i, y_i^1, y_i^2, y_i^3 — the number of the leftmost of the three columns where the block should be placed, and the description of its shape — the three heights of the columns of the inverted histogram that define the block ($1 \le x_i \le m-2$; $1 \le y_i^j \le 3$).

Output

Output a single integer — the number of the first block from 1 to q that cannot be placed in the desired location because it will go beyond the boundaries of the field.

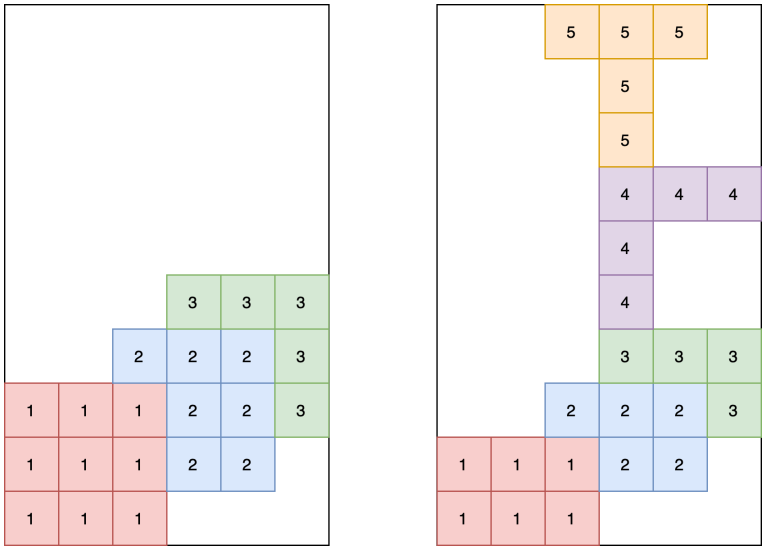
If all blocks can be placed, output -1.

Examples

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

Note

In the first example, the field changes as follows:



Problem L. Large Event

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

Less than a month remains until the main celebration of the year, yet invitations aren't sent yet. Fortunately, there are n organizations that can do this job for you!

Exactly a_i helpers work in the i-th organization. There are also m royal courts that should be invited, and according to the rules, a delegation consisting of exactly b_i helpers must be sent to the i-th court.

So far, you have only managed to prepare two official royal invitations to the event, which means that sending more than two delegations today is not possible, so you will have to send delegations to exactly two royal courts out of m.

Both delegations must be sent by the same organization. That is, when choosing courts i and j, it is necessary to select an organization of size $a_k \geq b_i + b_j$, after which an unordered set of helpers of size b_i will be chosen to send to court i, and from the remaining $a_k - b_i$ helpers another unordered set of size b_j will be chosen to send to court j.

Calculate the number of ways to choose helpers for two delegations. Since the answer can be very large, output the remainder of its division by 998 244 353.

Input

The first line of input contains two integers n and m — the number of organizations and the number of royal courts ($1 \leq n \leq 2 \cdot 10^5$; $2 \leq m \leq 2 \cdot 10^5$).

The second line lists n integers a_i — the size of each organization ($1 \leq a_i \leq 2 \cdot 10^5$).

The third line lists m integers b_i — the required number of delegates for each royal court ($1 \leq b_i \leq 2 \cdot 10^5$).

Output

Output a single integer — the number of ways to choose an organization, two royal courts, and two sets of helpers from that organization that will go to these courts, modulo 998 244 353.

Examples

standard input	standard output
2 2 1 2 1 1	2
2 3 5 3 3 2 1	63

Note

In the first example, there is only one way to choose an organization and two royal courts, for which there are two ways to distribute the helpers among the delegations.

In the second example, there are

- $C_3^2 \cdot C_1^1 = 3$ ways to send a delegation of three helpers to royal courts numbered 2 and 3;
- $C_5^3 \cdot C_2^1 = 20$ ways to send a delegation of five helpers to royal courts 1 and 3;
- $C_5^3 \cdot C_2^2 = 10$ ways to send the same delegation to royal courts 1 and 2;

- $C_5^2 \cdot C_3^1 = 30$ ways to send the same delegation to royal courts 2 and 3.

Problem M. Macro Tree

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

You have to transport a large tree. The tree is rooted and consists of n vertices with the root at vertex 1, and each vertex has its own weight a_i. The weight of a tree is a total weight of its vertices.

Sadly the tree exceeds the capacity of your backpack by w units of weight.

So in one action, you can choose a certain vertex v and cut it off along with its subtree (all vertices u such that a path between u and root contains v). Your goal is to cut vertices with a total weight of exactly w.

Determine if it's possible.

Input

The first line of input contains two integers n and w — the number of vertices in the tree and the required weight of the vertices to be removed ($1 \le n, w \le 1000$).

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ — the weights of the vertices ($1 \le a_i \le 100$).

In each of the next $n - 1$ lines, there are two integers u_i and v_i — the numbers of the vertices connected by the i-th edge. It is guaranteed that the given graph is a tree ($1 \le u, v \le n$; $u \ne v$).

Output

If it is impossible to cut off vertices with a total weight of w using the described actions, output -1.

Otherwise, first output an integer k from 0 to n — the number of cuts. Then output k distinct integers from 1 to n — the numbers of the vertices that need to be cut off. No two printed vertices should be a child and an ancestor.

If there are multiple suitable sequences of actions, output any.

Examples

standard input	standard output
5 7 3 4 3 2 2 1 2 2 3 2 4 4 5	2 4 3
5 8 3 4 3 2 2 1 2 2 3 2 4 4 5	-1
5 14 3 4 3 2 2 1 2 2 3 2 4 4 5	1 1

Note

In the first example, you can cut off vertices 3 and 4, and along with vertex 4, you will also cut off vertex 5, since vertex 5 is in a subtree of vertex 4. The total weight of the removed vertices 3, 4, 5 equals $3 + 2 + 2 = 7$.