Problem A
Abstract Painting
Problem ID: abstractpainting

Gon is currently training to become a modern artist.

Everyday, Gon practices his painting skill on a rectangular canvas, divided into $R \cdot C$ unit squares, with R rows and C columns. Gon wants to paint all the edges of all unit squares.

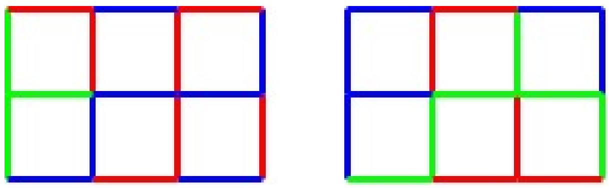
Contrary to popular belief, creating a good modern painting is not an easy task. A good modern painting should use a limited number of colors, simple yet elegant. Thus, when creating his painting, Gon strictly adheres to the following rules:

Abstract Painting.

- Only 3 colors are used: Red, Green and Blue.
- All edges of all unit squares must be painted. Each edge must be painted with exactly one color.
- For each unit square, exactly 2 colors must be used to paint its 4 edges. Furthermore, each color must be used to paint exactly 2 edges.

In the following figure:

- The painting on the left is a good painting.
- The painting on the right is **not** a good painting, because the top-left unit square has 3 blue edges.



Now Gon is wondering, how many different good paintings are there? Two paintings, both with R rows and C columns, are considered different, if there exists one edge painted with different colors in the two paintings. Please help Gon!

Input

The first line contains exactly one integer T — the number of test cases $(1 \leq T \leq 5)$.

T lines follow, each line contains exactly two integers R and C $(1 \leq R \leq 14, 1 \leq C \leq 2\,000)$.

Output

Output exactly T lines, each line contains a single integer — the number of different good paintings, modulo $10^9 + 7$.

Sample Input 1	Sample Output 1

3
1 1
1 2
2 1
	18
	108
	108

Problem B
Banana Problem
Problem ID: bananaproblem

Gon is playing the game 'Greedy Monkeys'.
The game is setup as follows:

- There are N ladders, numbered from 1 to N. Each ladder has height H, and has $H + 1$ steps at height $0, 1 \ldots, H$. Let (ℓ, h) represent the step at height h of the ℓ-th ladder.
- There are R horizontal ropes. Each rope connects two steps at the same height of two different ladders. The ropes are numbered from 1 to R, and the i-th rope's length is b_i. No rope is connected to any 0-th or H-th step of any ladders. **A step in a ladder has at most one rope connected to it**.
- There are B special positions, numbered from 1 to B. The i-th position is parameterized by 4 integers ℓ_i, h_i, x_i and y_i, where $1 \le \ell_i \le N$ and $1 \le h_i \le H$. For every non-negative integer t:
  - At time $t \cdot (x_i + y_i) + 0.5$, a new banana appears at position (ℓ_i, h_i).
  - At time $t \cdot (x_i + y_i) + x_i + 0.5$, this banana disappears from position (ℓ_i, h_i).

  For example, if $x_i = 2$ and $y_i = 3$: a banana appears at 0.5 and disappears at 2.5. Then a **different** banana appears at 5.5 and disappears at 7.5, and so on. It is guaranteed that **no rope is connected to a special location**. All special locations are distinct.

At the beginning of the game, there are N monkeys, numbered from 1 to N. The i-th monkey stands at $(i, 0)$. It takes the i-th monkey c_i seconds to climb up one step of any ladders, and d_i seconds to move one unit along any ropes.

A monkey can eat a banana if and only if the monkey is at the same position as the banana, and the banana has not yet disappeared. It takes a monkey a negligible amount of time to eat a banana. Naturally, a banana can only be eaten once. If multiple monkeys is at the same position as a banana at the same time, only one monkey can eat that banana. However, if monkeys arrive at a special location multiple times, they may eat more than one banana in this location.

The player needs to control the monkeys. As Gon is a simple boy, he controls the monkeys using a very simple strategy:

- If a monkey is at a step of a ladder connected to a rope, and **the monkey has not moved along this rope in either directions**, it will immediately start moving along the rope towards the other end.
- Otherwise, the monkey will move up the ladder.
- The monkeys always move according to the above two rules, unless they can no longer moves (which happens only when they reach the H-th step of some ladders).

- The monkeys can freely pass through each other.

How many bananas will the monkeys eat, if Gon follows this strategy?

Input

The first line of input contains 4 integers N, H, R and B $(1 \le N \le 3 \cdot 10^5, 0 \le R, B \le 3 \cdot 10^5, 1 \le H \le 10^9)$ — the number of ladders, the height of each ladder, the number of ropes and the number of special locations.

In the next R lines, the i-th line contains 4 integers b_i, $\ell_{1,i}$, $\ell_{2,i}$ and s_i $(1 \le b_i \le 10^4, 1 \le \ell_{1,i}, \ell_{2,i} \le N, \ell_{1,i} \ne \ell_{2,i}, 1 \le s_i < H)$, meaning that the i-th rope has length b_i and connects two positions $(\ell_{1,i}, s_i)$ and $(\ell_{2,i}, s_i)$.

In the next N lines, the i-th line contains 2 integers c_i and d_i $(1 \le c_i, d_i \le 10^4)$ — the number of seconds it takes for a monkey to move up one step of some ladder, and move one unit along some rope.

In the next B lines, the i-th line contains 4 integers ℓ_i, h_i, x_i and y_i $(1 \le \ell_i \le N, 1 \le h_i \le H, 1 \le x_i, y_i \le 10^4)$ — where (ℓ_i, h_i) is the position of the i-th special location. x_i and y_i are its parameters.

It is guaranteed that:

- Each position is connected to at most one rope.
- All special positions, where bananas appear, are distinct.
- Special positions are not connected to any ropes.

Output

Output a single integer — the number of bananas that the monkeys eat, following Gon's strategy.

Explanation of Sample input

In the first example:

- There is only 1 ladder and no ropes.
- The monkey initially stands at $(1, 0)$. It takes 2 seconds for the monkey to move up one step of a ladder.
- There are 2 special locations:
  - The first special location is at $(1, 1)$. A banana appears at time 0.5, disappears at time 1.5. Another banana appears at time 3.5, disappears at time 4.5, and so on.
  - The second special location is at $(1, 2)$. A banana appears at time 0.5, disappears at time 2.5. Another banana appears at time 3.5, disappears at time 5.5, and so on.
- The monkey arrives at $(1, 1)$ at time 2. There is no banana at $(1, 2)$ at this time.

- The monkey arrives at $(1, 2)$ at time 4. The monkey eats 1 banana here.
- Thus, the answer is 1.

In the second example:

- There are 2 ladders, and 1 rope connecting $(1, 1)$ and $(2, 1)$ with length 1.
- The first monkey is initially at $(1, 0)$. It will be at:
  - $(1, 1)$ at time 1,
  - $(2, 1)$ at time 3,
  - $(2, 2)$ at time 4.
- The second monkey is initially at $(2, 0)$. It will be at:
  - $(2, 1)$ at time 2,
  - $(1, 1)$ at time 3,
  - $(1, 2)$ at time 5.
- There are 2 special locations:
  - The first special location is at $(1, 2)$. There is a banana from 0.5 to 3.5, another banana from 4.5 to 7.5 and so on.
  - The second special location is at $(2, 2)$. There is a banana from 0.5 to 2.5, another banana from 5.5 to 7.5 and so on.
- The second monkey can eat the banana at the first special location, and the first monkey can not eat any banana.
- Thus, the answer is 1.

Sample Input 1

```
1 2 0 2
2 1000
1 1 1 2
1 2 2 1
```

Sample Output 1

```
1
```

Sample Input 2

```
2 2 1 2
1 1 2 1
1 2
2 1
1 2 3 1
2 2 2 3
```

Sample Output 2

```
1
```

Problem C
Counting Palindromes
Problem ID: countingpalindromes

A palindrome number is a non-negative number without leading zeroes, that reads the same forward or backward. For example, 12321, 44 and 9 are palindrome numbers, while 010, 123 and 100 are not.

Given a positive integer n, a prime number p and a non-negative integer k less than p. A palindrome number x is called a good palindrome, iff x is equal to k modulo p. In other words, the remainder when x is divided by p equals k.

Please count the number of good palindromes with exactly n digits. As this number can be very large, please calculate the result modulo $10^9 + 7$.

Input

The input contains 3 integers n, p and k $(1 \leq n \leq 10^{18}, 2 \leq p \leq 1\,000, 0 \leq k < p)$. It is guaranteed that p is a prime.

Output

Output a single integer — the number of good palindromes with exactly n digits, modulo $10^9 + 7$.

Explanation of Sample input

The good palindromes are 0, 2, 4, 6 and 8.

Sample Input 1	Sample Output 1
`1 2 0`	`5`

Problem D
Dating time
Problem ID: datingtime

Our two lovebirds, Banmuon and Henho want to go on a date. However, they are afraid that their friends or families may find out. Thus, they have decided to setup their dating time using the following secret methods:

- Banmuon will send Henho a string with format `h1:m1 h2:m2 alpha` where `h1:m1` and `h2:m2` represent some time in 24-hour format. $alpha$ is an integer between 0 and 180 and **is divisible by** 90.
- Their dating time will be some time between `h1:m1` and `h2:m2`, where the two clock hands, hour and minute, forms an angle equals to $alpha$ degree.

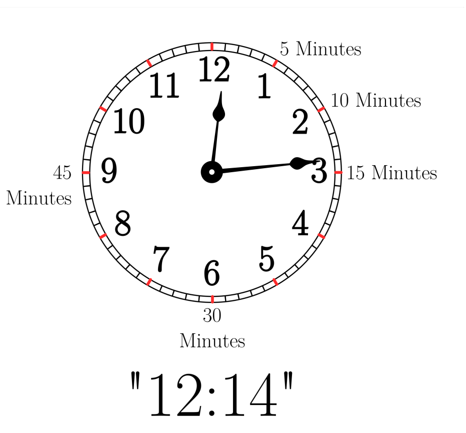


Photo by Phrontis, via Wikipedia.

However, there is a problem: their dating time may not be unique! Please help Banmuon and Henho figure out how many possible valid dating times there are.

Note

In this problem, we use a normal analog clock:

- The minute hand rotates continuously. It takes 60 minutes to make one complete rotation (from 12 to 12).
- The hour hand rotates continuously. It takes 12 hours to make one complete rotation (from 12 to 12).

Input

The first line of input contains a single integer t $(1 \le t \le 100)$ — the number of test cases.

t lines follow, each line has format: `h1:m1 h2:m2 alpha` $(0 \le h_1, h_2 \le 23, 0 \le m_1, m_2 \le 59, 0 \le alpha \le 180)$. h_1, m_1, h_2 and m_2 will have exactly two digits, with leading zero if they are less than 10.

It is guaranteed that `h1:m1` is less than or equal to `h2:m2` and $alpha$ **is divisible by** 90.

Output

For each test case, print the number of instants when the two clock hands form an angle equals to $alpha$ degrees, between `h1:m1` and `h2:m2`, inclusive.

Explanation of Sample input

In the first test case, there are 22 instants when the two clock hands form a 0-degree angle. Below are the first 11 instants.

1. `00:00`
2. `01:05` and $\frac{5}{11}$ minute.
3. `02:10` and $\frac{10}{11}$ minute.
4. `03:16` and $\frac{4}{11}$ minute.
5. `04:21` and $\frac{9}{11}$ minute.
6. `05:27` and $\frac{3}{11}$ minute.
7. `06:32` and $\frac{8}{11}$ minute.
8. `07:38` and $\frac{2}{11}$ minute.
9. `08:43` and $\frac{7}{11}$ minute.
10. `09:49` and $\frac{1}{11}$ minute.
11. `10:54` and $\frac{6}{11}$ minute.

Sample Input 1

```
3
00:00 23:59 0
00:00 23:59 90
18:00 18:01 180
```

Sample Output 1

```
22
44
1
```

Problem E
Easy Query
Problem ID: easyquery

You are given an array a with n integers $a_1, a_2, \ldots, a_n$.

You need to answer q queries. In each query, you are given 4 integers ℓ, r, u and v ($1 \le \ell \le r \le n, 1 \le u \le v \le r - \ell + 1$):

- Consider the sub-array of a from index ℓ to index r, inclusive: $a_\ell, a_{\ell+1}, \ldots, a_r$.
- Let $s = [s_1, s_2, \ldots, s_{r-\ell+1}]$ be the array obtained by sorting the above sub-array in non-decreasing order. Please note that we do not change the order of any elements in the original array a, i.e. array a remains the same after every query.
- Let $t^{(i)}$ be the **set** of values which appears at least i times in $s_u, s_{u+1}, \ldots, s_v$.
- Let $f(t^{(i)})$ be the bitwise OR of all elements in $t^{(i)}$.
- The result of the query is $f(t^{(1)}) + f(t^{(2)}) + f(t^{(3)})$.

For example, let $a = [123, 3, 2, 2, 7, 2, 1, 5, 5, 7, 456]$. Consider the query with $\ell = 2, r = 10, u = 2, v = 8$:

- $s = [1, 2, 2, 2, 3, 5, 5, 7, 7]$.
- $t^{(1)} = \{2, 3, 5, 7\}$.
- $t^{(2)} = \{2, 5\}$.
- $t^{(3)} = \{2\}$.
- $f(t^{(1)}) = 2 \,|\, 3 \,|\, 5 \,|\, 7 = 7$.
- $f(t^{(2)}) = 2 \,|\, 5 = 7$.
- $f(t^{(3)}) = 2$.
- Result of the query is $7 + 7 + 2 = 16$.

Input

The first line of input contains T ($1 \le T \le 2 \cdot 10^5$) — the number of test cases.

T test cases follow, each consists of:

- The first line contains 2 integers n and q ($1 \le n, q \le 2 \cdot 10^5$).
- The second line contains exactly n integers, representing array a ($1 \le a_i \le 10^9$).

- In the next q lines, each line contains exactly 4 integers ℓ, r, u and v $(1 \le \ell \le r \le n, 1 \le u \le v \le r - \ell + 1)$.

The sum of n over all T test cases does not exceed $2 \cdot 10^5$.
The sum of q over all T test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output exactly q lines, the i-th line contains exactly one integer — the answer to the i-th query.

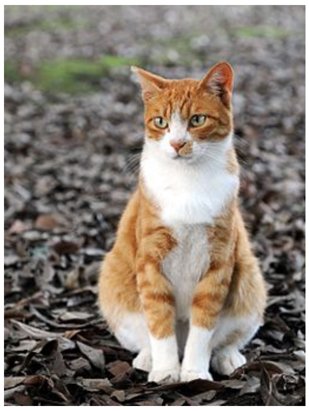
Sample Input 1

```
1
11 3
123 3 2 2 7 2 1 5 5 7 456
2 10 2 8
1 1 1 1
1 4 1 3
```

Sample Output 1

```
16
123
5
```

Problem F
Fair Bandwidth Sharing
Problem ID: fairbandwidthsharing


Orange tabby cat.

Dreaming of revolutionizing the tech industry and making the world a better place, Ging has recently moved to Silicon Valley to found his own startup, Katbook.

Katbook allows users to download cat pictures. Katbook is capable of transferring t bits per second.

There are n species of cats, numbered from 1 to n. There is a demand ratio d_i for the pictures of the i-th species. This means, if there is no further restrictions, the i-th cat species should have a 'fair share' of $\frac{d_i}{\sum_{j=1}^{n} d_j}$ fraction of the total bandwidth t.

However, because of infrastructure limitations, the bandwidth for the i-th species must be between a_i and b_i.

You are newly hired by Ging as a network engineer at Katbook. Your first task is to find the most 'fair' bandwidth allocation satisfying the above constraints. More formally, let x_i be the bandwidth allocated for downloading pictures of the i-th species. You must find x_i such that:

- $a_i \le x_i \le b_i$,
- $\sum_{i=1}^{n} x_i = t$,
- Let $y_i = t \cdot \frac{d_i}{\sum_{j=1}^{n} d_j}$ (y_i is the 'fair share' bandwidth, if there was no constraints regarding a_i and b_i), the value $\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{y_i}$ should be as small as possible.

Input

The first line of input contains 2 integers n and t ($1 \le n \le 10^5, 1 \le t \le 10^6$).

In the next n lines, the i-th line contains 3 integers a_i, b_i and d_i ($0 \le a_i \le b_i \le 10^6, 0 < b_i, 1 \le d_i \le 10^6$).

It is guaranteed that there exists at least one valid solution.

It can be shown that under these constraints, the solution is unique.

Output

Output n lines, each line contains the allocation for downloading pictures of the i-th species. The answer will be accepted if and only if its relative or absolute error to the optimal solution is at most 10^{-6}.

Formally, for each line, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Explanation of Sample input

In the first sample, each cat species has its 'fair share' of $\frac{1}{3}$ of the whole bandwidth.

In the second sample, note that the cat species 1 has much more demand and could get almost all of the available bandwidth if not for its maximum to be capped at 1. The rest is divided with ratio $2:1$, hence the cat species 2 gets 6 bits per second and the cat species 3 gets 3 bits per second.

Sample Input 1

```
3 10
0 10 1
0 10 1
0 10 1
```

Sample Output 1

```
3.33333333
3.33333333
3.33333333
```

Sample Input 2

```
3 10
0 1 1000
2 8 2
2 8 1
```

Sample Output 2

```
1.00000000
6.00000000
3.00000000
```

Problem G
Generating Numbers
Problem ID: generatingnumbers

Ging is working on the next generation of computers – 'Tanquum computers'.

'Tanquum computers' take advantages of advanced Physics, allowing them to rearrange the digits of every positive integer in one instruction.

The application is limitless. For example, from number 1, 'Tanquum computers' are able to generate every positive integer x, using only two types of instructions: increment and rearrangement.

More precisely, only the following two operations are allowed on an integer a:

1. Increase a by 1, i.e. assigning

   ```
   a := a + 1
   ```

2. Rearrange the digits of a. The result must not have any leading zeroes. For example, from 102, using one instruction of this type, you can get any of the following numbers: $102, 120, 201, 210$.

As 'Tanquum computers' is very advanced, it only uses a **minimum** number of instructions.

You have recently been hired by Ging to help with the development of 'Tanquum computers'. Your first task is to verify the correctness of 'Tanquum computers'. Given some number x, you need to find the minimum number of instructions it would take to transform 1 into x.

Input

The first line of input contains a single integer t $(1 \leq t \leq 1\,000)$ — the number of test cases.

In the next t lines, each line contains a single integer x $(1 \leq x \leq 10^9)$.

Output

For each test case, print a single line containing the minimum number of instructions.

Explanation of Sample data

- In the first example, it takes 4 increments to generate the number 5.
- In the second example, you can generate 26 by applying the following 17 instructions:
  - 11 increments to get 12,
  - 1 rearrangement to get 21,
  - 5 increments to get 26.

Sample Input 1

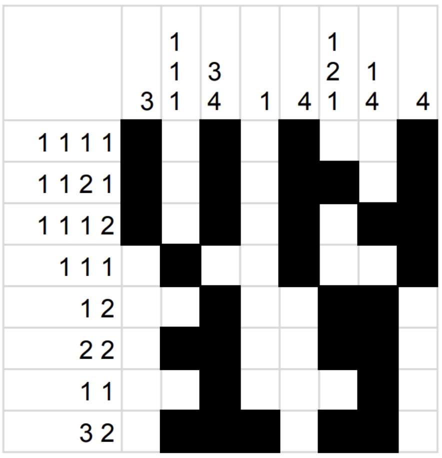
```
3
5
26
843
```

Sample Output 1

```
4
17
44
```

Problem H
Hanjie
Problem ID: hanjie

Hanjie, also known as Nonograms or Picross, are picture logic puzzles, where cells in a grid must be colored with black or white, according to numbers at the side of the grid:



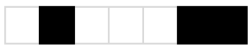
Formally, you are given a board with r rows and c columns. On each row and each column, you are also given a **clue** — a sequence $a_1, a_2, \ldots, a_k$, meaning: If you iterate through the cells in this row (or column) from left to right (or from top to bottom), you will see:

- **zero or more** consecutive white cells,
- followed by **exactly** a_1 consecutive black cells,
- followed by **at least one** consecutive white cells,
- followed by **exactly** a_2 consecutive black cells,
- followed by **at least one** consecutive white cells,
- $\ldots$
- followed by **exactly** a_k consecutive black cells,
- followed by **zero or more** consecutive white cells.

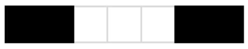
Note that it is possible for a to be empty. In that case, all cells in that row (or column) should be white.

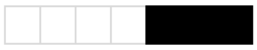
For example, consider the following row with 7 columns and $a = 1, 2$:

The following row is valid:



The following two rows are **not** valid:





Given r, c and the clues for each row and column, please count the number of different solutions satisfying all the clues. Two solutions are considered different, if there exists at least one cell with different colors.

Input

The first line of input contains 2 integers r and c $(1 \leq r, c \leq 6)$.

The next r lines, the i-th line contains the clue for the i-th row: a non-negative integer k followed by k positive integers $a_1, a_2, \ldots a_k$, which satisfy $\sum_{i=1}^{k} a_i + k - 1 \leq c$.

The next c lines, the i-th line contains the clue for the i-th column: a non-negative integer k followed by k positive integers $a_1, a_2, \ldots a_k$, which satisfy $\sum_{i=1}^{k} a_i + k - 1 \leq r$.

Output

Output a single integer — the number of different solutions.

Sample Input 1

```
2 2
1 1
1 1
1 1
1 1
```

Sample Output 1

```
2
```

Problem I
Inspecting Illumination
Problem ID: inspectingillumination

This is an interactive problem

The Zoldyck mansion is enormous, with dozens of rooms and hundreds of illuminating sources — light bulbs, chandeliers, lamps, etc.

You are recently hired by the Zoldyck family as a butler. Your daily job involves sitting in the illumination control room and control all the illuminating sources in the mansion.

In the control room, there are n switches. Each switch controls exactly one illuminating source. Switches and illuminating sources are numbered from 1 to n, inclusive. However, there is no documentation, so you **do not know** which switch controls which illuminating source.

Thus, your only choice is to repeat the following operation:

- Toggle some of the switches (at least one).
- Go around the entire mansion, check all the state of all n illuminating sources.

As the mansion is enormous, you want to go around it at most 32 times.

Interaction

First your program reads the integer n $(1 \leq n \leq 1\,000)$.

Then the following process repeats:

- Your program writes to the standard output one of the following:
  - ASK k a_1 a_2 $\ldots$ a_k ($1 \leq k \leq n, 1 \leq a_i \leq n$ and all a_i are unique) — you toggle k switches $a_1, a_2, \ldots, a_k$, and want to know what are the k illuminating sources which are toggled.
  - ANSWER b_1 b_2 $\ldots$ b_n $(1 \leq b_i \leq n)$ — you want to answer that the illuminating source i is controlled by the switch b_i.
- If your program asks a query, k integers will be available in the standard input, representing the illuminating sources which were toggled, in any order. Your program should then read them.
- If your program prints an answer, it should then terminate. You are allowed to print an answer exactly once.

Note that you are allowed to interact at most $32 + 1 = 33$ times, 32 interactions for asking queries and 1 interaction for answering.

Note

After printing a query do not forget to output end of line and flush the output. Otherwise, your submission may be rejected. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python.

Sample Interaction 1

Read	Write
	`5`
`ASK 1 1`	
	`1`
`ASK 2 1 2`	
	`1 3`
`ASK 3 1 2 3`	
	`1 3 4`
`ASK 4 1 2 3 4`	
	`1 2 3 4`
`ASK 5 1 2 3 4 5`	
	`1 2 3 4 5`
`ANSWER 1 4 2 3 5`	

Problem J
Justice for Ants
Problem ID: justiceforants

Chimera Ant is an intelligent and hard-working species.

There is a colony of Chimera Ants living in Danang, Vietnam. With careful infrastructure planning and hard work, they have built their kingdom with n towns, numbered from 1 to n. The towns are connected by $n-1$ two-way roads, such that, there is exactly one simple path between any pair of towns.

Winter is coming! The Chimera Ant Queen is planning to distribute food to the n towns: the i-th town will receive a_i units of food.

However, the ants feel that the plan is not fair. The ants demand justice! They submit q requests. In each request, 4 towns ℓ, r, u and v are given, such that the distance from ℓ to r and from u to v are equal. Here the distance from town a to town b is the number of edges on the only simple path from a to b.

Let $p_1 = u, p_2, \ldots, p_\ell = v$ denotes the simple path between town u and town v; $q_1 = x, q_2, \ldots, q_\ell = y$ denotes the simple path between town x and town y. The request says that for every i between 1 and ℓ, town p_i and town q_i receive the same amount of food.

Changing the food distribution plan turns out to be a difficult problem. The Chimera Ant Queen has a non-negative integer k, and each second, she can change the plan by applying the following steps:

- Select an integer t.
- Select a town i.
- Either decrease or increase the amount of food distributed to town i by $t \cdot k + 1$. In other words, the Chimera Ant Queen can either set $a_i := a_i + (t \cdot k + 1)$ or $a_i := a_i - (t \cdot k + 1)$. After this step, a_i **must be a non-negative integer**, as we cannot distribute a negative amount of food.

What is the shortest time it would take for the Chimera Ant Queen to change the food distribution plan to satisfy all the requests?

Input

The first line of input contains 3 integers n, q and k — the number of towns, the number of requests and the integer the Chimera Ant Queen has $(1 \le n \le 5 \cdot 10^5, 1 \le q \le 2 \cdot 10^5, 0 \le k \le 10^6)$.

The second line contains n integers, $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$, where a_i is the initial amount of food distributed to town i.

The next $n-1$ lines contain $n-1$ integers $p_2, p_3, \ldots p_n$ $(1 \le p_i < i)$, one on each line, meaning that there is a two-way road connecting town p_i and town i.

Each of the next q lines contains 4 integers u, v, x and y, describing one request $(1 \leq u, v, x, y \leq n)$. It is guaranteed that the two simple paths from u to v and from x to y have equal length.

The input guarantees that there is exactly one simple path between any pair of towns, and it is possible to satisfy all q requests.

Output

Output a single integer — the minimum number of seconds it would take for the Chimera Ant Queen to change the food distribution plan.

Explanation of Sample input

There are 4 roads: $(1, 2)$, $(1, 3)$, $(3, 4)$ and $(3, 5)$.

Since $k = 0$, in each second, the Chimera Ant Queen can either increase or decrease a_i by 1.

An optimal solution would be to reduce a_1 to 1 and a_4 to 1, which would take 3 seconds.

Sample Input 1

```
5 3 0
3 1 1 2 1
1
1
3
3
2 3 4 5
1 3 3 4
4 4 5 5
```

Sample Output 1

```
3
```

Problem K
Keep It Sorted
Problem ID: keepitsorted

Gon has a permutation $a = (a_1, a_2, \ldots, a_n)$ of integers between 1 and n (inclusive), and wants to sort it in increasing order. However, sorting is trivial — anyone can sort an array!

Thus, Gon decides to only use the following operation to sort:

- Select 2 indices ℓ and r $(1 \leq \ell \leq r \leq n)$, such that the sub-array $(a_\ell, a_{\ell+1}, \ldots, a_r)$ is sorted, in either increasing or decreasing order.
- Reverse the sub-array from ℓ to r.

For example, given a permutation $a = (3, 2, 1)$, Gon can sort it with 1 operation as follow:

- Select $\ell = 1, r = 3$.
- Reverse the sub-array (a_1, a_2, a_3) to get $a = (1, 2, 3)$.

Note that if $a = (3, 1, 2)$, Gon **cannot** select $\ell = 1, r = 3$, as the sub-array from 1 to 3 is not sorted.

As Gon's birthday is Jan 19th, Gon wants to use at most 191 operations. This turns out to be non-trivial. Please help Gon!

Input

The first line of input contains a single integer n $(1 \leq n \leq 100)$.

The second line of input contains n integer $a_1, a_2, \ldots, a_n$. It is guaranteed that a is a permutation of integers between 1 and n, inclusive.

Output

Print a single number k $(0 \leq k \leq 191)$ on the first line — the number of operations you want to use.

In each of the next k lines, print two integers ℓ and r $(1 \leq \ell \leq r \leq n)$ describing the operations. The sub-array between ℓ and r must be sorted before this operation.

After k operations, the permutation a must be sorted in increasing order. In other words, $a_i = i$ for every valid index i. Note that you **do not** need to minimize the number of operations.

It is guaranteed that a solution exists. If there are multiple solutions, you can print any of them.

Sample Input 1

```
3
3 1 2
```

Sample Output 1

```
7
1 1
2 2
3 3
1 2
2 3
1 3
1 3
```

Problem L
Latin Square
Problem ID: latinsquare

A Latin Square is a $n \cdot n$ array filled with n integers from 1 to n, such that each number appears **exactly once** in each row and in each column.

For example, below is a Latin Square:

1	2	3
2	3	1
3	1	2

Gon really likes Latin Squares. He is currently creating one.

Gon first creates an empty matrix with n rows and n columns. Gon then writes $k \cdot n$ numbers in the matrix, using k unique numbers, each appears n times; so that in each row and in each column, no number appears more than once.

However, Gon does not know how to proceed. Please help Gon fills the rest of the Latin square. Of course, you cannot remove any number that Gon wrote. In other words, you can only write on **empty cells**.

Input

The first line of input contains 2 integers, n and k $(1 \le n \le 100, 0 \le k \le n)$.

Each of the next n lines contains n integers representing the Latin square. The numbers are between 0 and n, with 0 representing an empty cell.

It is guaranteed that in each row and in each column, there are no two cells having the same value, and there are exactly k different positive numbers used in the matrix.

Output

If there is no solution, print exactly one line containing 'NO'.

Otherwise, print 'YES', followed by n lines, each containing n numbers — the Latin square.

If there are more than one solution, you can print any one.

Sample Input 1

```
3 1
2 0 0
0 2 0
0 0 2
```

Sample Output 1

```
YES
2 1 3
3 2 1
1 3 2
```

Problem M
Moscow Dream
Problem ID: moscowdream

For many students, getting a ticket to The ICPC World Finals is a huge achievement, and *The 2019 ICPC Asia Danang Regional Contest* is a chance to make their dream come true. For some others, they just like to stretch their brains and solve interesting problems.

We – *the scientific committee* understand that, and we tried our best to set up a problemset that is interesting and diverse in both topics and difficulty. For a few months, we called for problem proposals from many people and received a easy problems, b medium problems and c hard problems. Using these proposals, we want to create a problemset which:

- Consists of **exactly** n problems,
- Has at least 1 easy problem,
- Has at least 1 medium problem,
- Has at least 1 hard problem.

Your task is to check whether it is possible to create such a problemset using the available problems.

Input

The input contains 4 integers a, b, c and n $(0 \le a, b, c \le 10, 1 \le n \le 20)$.

Output

Print 'YES' if it is possible to create a problemset satisfying above requirements, and 'NO' otherwise.

Explanation of sample data

- In the first sample, the committee do not have any easy problem. Thus, they cannot create a problemset with at least 1 easy problem.
- In the second sample, the committee can use 3 easy problems, 7 medium problems and 3 hard problems to create a problemset with exactly 13 problems.

Sample Input 1	Sample Output 1
`0 3 3 5`	`NO`

Sample Input 2	Sample Output 2
`4 10 6 13`	`YES`