TASK	KARTE	AKCIJA	BALONI	TOPOVI	RELATIVNOST	UZASTOPNI

source code	`karte.pas`
	`karte.c`
	`karte.cpp`
	`karte.py`
	`karte.java`
		`akcija.pas`
		`akcija.c`
		`akcija.cpp`
		`akcija.py`
		`akcija.java`
			`baloni.pas`
			`baloni.c`
			`baloni.cpp`
			`baloni.py`
			`baloni.java`
				`topovi.pas`
				`topovi.c`
				`topovi.cpp`
				`topovi.py`
				`topovi.java`
					`relativnost.pas`
					`relativnost.c`
					`relativnost.cpp`
					`relativnost.py`
					`relativnost.java`
						`uzastopni.pas`
						`uzastopni.c`
						`uzastopni.cpp`
						`uzastopni.py`
						`uzastopni.java`

input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	2 seconds	2 seconds	4 seconds	0.5 seconds
memory limit	32 MB	32 MB	128 MB	64 MB	32 MB	32 MB
point value	**50**	**80**	**100**	**120**	**140**	**160**
	650					

Problems translated from Croatian by: **Paula Gombar**

Recently, Pero has been into robotics, so he decided to make a robot that checks whether a deck of poker cards is complete.

He's already done a fair share of work - he wrote a programme that recognizes the suits of the cards. For simplicity's sake, we can assume that all cards have a **suit** and a **number**.

The suit of the card is one of the characters **P**, **K**, **H**, **T**, and the number of the card is an integer between 1 and 13. The robot labels each card in the format TXY where T is the suit and XY is the number. If the card's number consists of one digit, then X = 0. For example, the card of suit P and number 9 is labelled P09.
A complete deck has 52 cards in total - for each of the four suits there is exactly one card with a number between 1 and 13.

The robot has read the labels of all the cards in the deck and combined them into the string **S**.
Help Pero finish the robot by writing a programme that reads the string made out of card labels and outputs how many cards are missing for each suit.
If there are two exact same cards in the deck, output GRESKA (Croatian for ERROR).

INPUT

The first and only line of input contains the string **S** ($1 \leq |\mathbf{S}| \leq 1000$), containing all the card labels.

OUTPUT

If there are two exact same cards in the deck, output "GRESKA".
Otherwise, the first and only line of output must consist of 4 space-separated numbers: how many cards of the suit P K, H, T are missing, respectively.

SAMPLE TESTS

input	**input**	**input**
P01K02H03H04	H02H10P11H02	P10K10H10T01
output	**output**	**output**
12 12 11 13	GRESKA	12 12 12 12

Clarification of the first example: The robot has read one card of the suit P, 1 card of the suit K, 2 cards of the suit H.

Clarification of the second example: There were two cards of the suit H with number 2, so the robot reports an error.

There is a promotional offer in a bookstore "Take 3, pay for the 2 more expensive ones". So, each customer who picks 3 books gets **the cheapest one** for free. Of course, the customer can take even more books and, depending on the way the books are arranged into groups of three, get the cheapest one in each group for free.

For example, let the prices of the books taken by the customer be: 10 3 2 4 6 4 9. If he arranges them into groups: (10, 3, 2), (4, 6, 4) and (9), he will get the books priced 2 from the first group for free and the book priced 4 from the second group. We can see that he won't get anything for free from the third group because it contains only one book.

The lady working in the bookstore is well-intentioned and she always wants to lower the price for each customer as much as possible. For given book prices, help the lady arrange the books into groups in the best way possible, so that the total price the customer has to pay is **minimal**.

Please note: The lady doesn't have to arrange the books into groups so that each group contains exactly 3 books, but the number of books in a group needs to be between 1 and 3, inclusively.

INPUT

The first line of input contains the integer **N** ($1 \leq \mathbf{N} \leq 100\,000$), the number of books the customer bought.
Each of the following **N** lines contains a single integer $\mathbf{C}_i$ ($1 \leq \mathbf{C}_i \leq 100\,000$), the price of each book.

OUTPUT

The first and only line of output must contain the required minimal price.

SCORING

In test cases worth 50% of points, it will hold that $\mathbf{N} \leq 2000$.

SAMPLE TESTS

input	**input**
4 3 2 3 2	6 6 4 5 5 5 5
output	**output**
8	21

Clarification of the first example: The lady can put the books priced 3, 2, 2 in one group, and only the book priced 3 in the other group, which is also the cheapest combination.

Clarification of the second example: The lady puts books priced 6, 4, 5 in one group, and 5, 5, 5 in the other, which gives us the cheapest combination.

There are **N** balloons floating in the air in a large room, lined up from left to right. Young Perica likes to play with arrows and practice his hunting abilities. He shoots an arrow from the left to the right side of the room from an arbitrary height he chooses. The arrow moves from left to right, at a chosen height **H** until it finds a balloon. The moment when an arrow touches a balloon, the balloon pops and disappears and the arrow continues its way from left to right at a height decreased by 1. Therefore, if the arrow was moving at height **H**, after popping the balloon it travels on height **H-1**.

Our hero's goal is to pop all the balloons using as little arrows as possible.

INPUT

The first line of input contains the integer **N** ($1 \leq \mathbf{N} \leq 1\,000\,000$).
The second line of input contains an array of **N** integers $\mathbf{H_i}$.

Each integer $\mathbf{H_i}$ ($1 \leq \mathbf{H_i} \leq 1\,000\,000$) is the height at which the $\mathbf{i^{th}}$ balloon floats, respectively from left to right.

OUTPUT

The first and only line of output must contain the minimal number of times Pero needs to shoot an arrow so that all balloons are popped.

SCORING

In test cases worth 40%, it will hold $\mathbf{N} \leq 5\,000$.

SAMPLE TESTS

input	input	input
5 2 1 5 4 3	5 1 2 3 4 5	5 4 5 2 1 4
output	**output**	**output**
2	5	3

Clarification of the first example: Our hero shoots the arrow at height 5 - which destroys [5, 4, 3], and shoots an arrow at height 2 - which destroys [2, 1].

Mirko is a huge fan of chess and programming, but typical chess soon became boring for him, so he started having fun with rook pieces.

He found a chessboard with **N** rows and **N** columns and placed **K** rooks on it.
Mirko's game is made of the following rules:

1. Each rook's power is determined by an integer.
2. A rook **sees** all the fields that are in his row or column **except** its own field.
3. We say that a field is attacked if the binary **XOR** of all the powers of the rooks that **see** the field is greater than 0.

Notice that the field a rook is at can be attacked or not.

Initially, Mirko placed the rooks in a certain layout on the board and will make **P** moves.
After each move, determine how many fields are attacked.
Every rook can be moved to any free field on the whole board (not only across column and row).

INPUT

The first line of input contains integers **N**, **K, P** ($1 \leq \mathbf{N} \leq 1\,000\,000\,000$, $1 \leq \mathbf{K} \leq 100\,000$, $1 \leq \mathbf{P} \leq 100\,000$).
Each of the following **K** lines contains three integers **R**, **C**, **X** ($1 \leq \mathbf{R}, \mathbf{C} \leq \mathbf{N}$, $1 \leq \mathbf{X} \leq 1\,000\,000\,000$) which denote that initially there is a rook of power **X** on the field (**R**, **C**).
Each of the following **P** lines contains four integers $\mathbf{R_1}$, $\mathbf{C_1}$, $\mathbf{R_2}$, $\mathbf{C_2}$ ($1 \leq \mathbf{R_1}, \mathbf{C_1}, \mathbf{R_2}, \mathbf{C_2} \leq \mathbf{N}$) which denote that the rook has moved from field ($\mathbf{R_1}$, $\mathbf{C_1}$) to field ($\mathbf{R_2}$, $\mathbf{C_2}$).
It is guaranteed that there will not be two rooks on one field at any point.

OUTPUT

The output must consist of **P** lines, the $\mathbf{k^{th}}$ line containing the total number of attacked fields after **k** moves.

SCORING

In test cases worth 25% of total points, **N** and **K** will not exceed 100.

SAMPLE TESTS

input	input	input
2 2 2 1 1 1 2 2 1 2 2 2 1 1 1 1 2	2 2 2 1 1 1 2 2 2 2 2 2 1 1 1 1 2	3 3 4 1 1 1 2 2 2 2 3 3 2 3 3 3 3 3 3 1 1 1 1 2 3 1 3 2
output	**output**	**output**
4 0	4 2	6 7 7 9

Clarification of the first example: After the first move, every field on the board is attacked.
For example, field (1, 1) is seen by only one rook so the total XOR for that field is 1. After the second move none of the fields are attacked. For example, field (1,1) is seen by both rooks so the total XOR for that field is 0.

Young Luka is an art dealer. He has **N** clients and sells artistic paintings to each client.
Each client can purchase either colored paintings or black and white paintings, but not both.
The client denoted with **i** wants to purchase at most $\mathbf{a_i}$ colored paintings and at most $\mathbf{b_i}$ black and white paintings.
The client will always purchase **at least one** paintings. Luka has an almost unlimited amount of paintings, so the number of paintings required from the clients is never a problem.
Luka doesn't like selling black and white paintings and knows that if less than **C** people get colored paintings, it will make him feel sad.

His clients constantly keep changing their requests or, in other words, the number of paintings they want to purchase. Because of this, Luka is often troubled by the question: "How many different purchases are there, so that at least **C** clients get at least one colored painting?" Help Luka and save him from his worries.

INPUT

The first line of input contains two integers **N, C** ($1 \leq \mathbf{N} \leq 100\,000$, $1 \leq \mathbf{C} \leq 20$).
The second line of input contains **N** integers $\mathbf{a_i}$ ($1 \leq \mathbf{a_i} \leq 1\,000\,000\,000$).
The third line of input contains **N** integers $\mathbf{b_i}$ ($1 \leq \mathbf{b_i} \leq 1\,000\,000\,000$).

The fourth line of input contains the number of requirement changes **Q** ($1 \leq \mathbf{Q} \leq 100\,000$).
Each of the following **Q** lines contains three integers, the label of the person changing the requirements **P** ($1 \leq \mathbf{P} \leq \mathbf{N}$), the maximal number of colored paintings they want to purchase $\mathbf{a_P}$ ($1 \leq \mathbf{a_P} \leq 1\,000\,000\,000$) and the maximal number of black and white paintings they want to purchase $\mathbf{b_P}$ ($1 \leq \mathbf{b_P} \leq 1\,000\,000\,000$).

OUTPUT

The output must consist of **Q** lines where each line contains the number of different purchases modulo **10 007**.

SCORING

In test cases worth 30% of total points, it will hold that **N** and **Q** are smaller than 1000.

SAMPLE TESTS

input	input	input
2 2 1 1 1 1 1 1 1 1	2 2 1 2 2 3 2 1 2 2 2 2 2	4 2 1 2 3 4 1 2 3 4 1 4 1 1
output	**output**	**output**
1	4 4	66

Clarification of the first example: After the first client changes his request from (1, 1) to (1, 1) - nothing really changes, the number of ways to sell paintings is 1. The one and only way to sell paintings is to sell the first client one colored painting, and the second client should be sold one colored painting as well. Every client is required to get at least one colored painting because **C=2**, which means that there should be at least 2 clients with colored paintings.

Petar is throwing a birthday party and he decided to invite some of the employees of his company where he is the CEO.
Each employee, including Petar, has a unique label from 1 to **N**, and an accompanying type of jokes they tell $\mathbf{V_i}$. Also, each employee of the company except Petar has exactly one supervisor.
Since Petar is the CEO of the company, he has the label 1 and is directly or indirectly superordinate to all the employees.
At the birthday party, there are certain rules that all people present (including Petar) must follow.

- At the party, there shouldn't be two people that tell the same type of jokes.
- Person X cannot be invited if their direct supervisor is not invited.
- Person X cannot be invited if the set of jokes the invitees that person X is superior to (directly or indirectly) tell and person X don't form a set of consecutive numbers.

The numbers in the set are consecutive if the difference between adjacent elements is exactly 1 when the set is sorted ascendingly. For example, (3, 1, 2) and (5, 1, 2, 4, 3).
Petar wants to know how many different sets of jokes he can see at his party with the listed constraints.

INPUT

The first line of input contains the integer **N**, ($1 \leq \mathbf{N} \leq 10\,000$).
The second line of input contains **N** integers, the types of jokes person **i** tells, $\mathbf{V_i}$, ($1 \leq \mathbf{V_i} \leq 100$).
Each of the following **N-1** lines contains two integers **A** and **B,** ($1 \leq \mathbf{A, B} \leq \mathbf{N}$), denoting that person **A** is directly superior to person **B**.

OUTPUT

The first and only line of output must contain the number of different sets of jokes that comply to the previously listed constraints.

SCORING

In test cases worth 50% of total points, it will hold that **N** does not exceed 100.

SAMPLE TESTS

input	input	input
4 2 1 3 4 1 2 1 3 3 4	4 3 4 5 6 1 2 1 3 2 4	6 5 3 6 4 2 1 1 2 1 3 1 4 2 5 5 6
output	**output**	**output**
6	3	10

Clarification of the first example: It is possible to have the following sets of jokes at the party: {2}, {2, 3}, {2, 3, 4}, {1, 2, 3, 4}, {1, 2}, {1, 2, 3}.
Clarification of the second example: The only possible sets of jokes are: {3}, {3, 4}, {3, 4, 5}. Notice that the person telling joke 6 cannot be at the party because, in that case, the set of jokes {4, 6} is not a set of consecutive numbers.