Problem E. Crazy old lady

Input file:	`crazy.in`
Output file:	`crazy.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

The following problem is a variation of a well known puzzle in the Probability theory. There are N seats on a plane and all of the seats are assigned to passengers. Passengers are asked to enter the plane, one at a time, in order of their seats, from 1 to N (the passenger that is assigned seat number 1 enters first, next enters the passenger that is assigned seat number 2, and so on...).

As you might have guessed, queues are too mainstream for a crazy old lady, so she usually goes ahead of everyone and enters first (despite the fact that she might not be assigned the first seat), furthermore she takes some seat that she likes (by chance, it may happen to be her own seat).

After she has taken a seat, the next passengers enter with respect to their seat number (as described above) and take seat as follows:

- if his/her seat is vacant, then he/she takes this seat.
- but if his/her seat is already taken, he/she takes any of the remaining seats.

Which seat takes the last passenger? The answer is simple: the last passenger takes either his own seat or the seat of the crazy old lady!

Assume that crazy old lady occupy seat j, which is not her own. When passenger that was assigned the seat j will enter, he/she will have to take some other seat from a set of free seats. This situation then may repeat with next passengers. But when some passenger take the seat of old crazy lady, obviously (!) all of the following passengers take their own seats. In this problem we would like to know the original seat of old crazy lady.

Input

Input file may contain several tests. First line of Input contains integer T — the number of tests ($1 \leq T \leq 10$). Each of the next T lines contains $N + 1$ numbers, separated by space, first is N ($1 \leq N \leq 10^3$), then N numbers follow: the seats taken by passengers in order of their entrance ($p_1, p_2, ..., p_N$ - where p_i means that the passenger that was i-th to enter took seat p_i). Input is guaranteed to be correct, i.e. to satisfy conditions of the problem statement.

Output

Output the seat of old crazy lady, if it can be determined uniquely, otherwise, output 0.

Examples

crazy.in	crazy.out
2 2 2 1 4 2 3 1 4	0 1

Note

In test case 2 2 1 we cannot uniquelly determine answer, as soon as passenger 1 is old crazy lady that take different seat, and passenger 2 may be also old crazy lady that take her own seat.

Problem F. Luxury burrow

Input file:	`burrow.in`
Output file:	`burrow.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Hobbit Bilbo is very tired from his adventures and decided to build a new burrow. The hill, where he plans to buy some land, has a shape of a rectangle and it can be represented as a table consisting of N rows and M columns (rows and columns are numbered starting from 1, where 1 corresponds to the topmost row and the leftmost column). Each cell of the table corresponds to a 1×1 square piece. Hobbits like simple shapes, that's why Bilbo intends to buy a plot of land of a rectangular shape, with sides parallel to the sides of the hill. In other words he chooses x_1, x_2, y_1, y_2 ($x_1 \leq x_2$, $y_1 \leq y_2$) and buys all cells (x, y), such that ($x_1 \leq x \leq x_2$) and ($y_1 \leq y \leq y_2$).

After all adventures he took part in, Bilbo is very rich and is not worried about the total price. However, he thinks a lot about his reputation and thus the price of the cheapest cell he will buy must be as high as possible. He also wants his new burrow to be enough to fit all of his furniture, that means the square of a rectangular plot he will buy has to have an area greater than or equal to K. If there are multiple possible plots, Bilbo chooses the one with the largest area.

Bilbo asks you to find the best plot of land for him.

Input

The first line of input file contains three numbers N, M and K, divided by exactly one space — the number of rows, the number of columns and the minimal possible square of land plot Bilbo can live on, respectively. Each of the following N lines contains M numbers — the prices of the corresponding cells. Hence, the number j in the line $i+1$, corresponds to the cell (i, j). N and M are positive integers and do not exceed 1000, K is positive integer and does not exceed the total number of cells in the table, all prices belong to range from 1 to 10^9, inclusive.

Output

As the answer, output two numbers separated by space. First goes the price of the cheapest cell in the resulting rectangular plot. Second is the total square of this plot.

Examples

burrow.in	burrow.out
3 3 3 1 1 1 1 2 2 1 2 2	2 4
1 10 5 4 3 2 5 10 7 6 5 1 100	5 5
3 5 2 5 7 5 5 5 8 5 5 7 5 8 5 8 8 8	8 3

Note

In 40% testcases M,$N \leq 200$.

In 64% testcases M,$N \leq 400$.

Problem G. Round words

Input file:	`rowords.in`
Output file:	`rowords.out`
Time limit:	2 seconds
Memory limit:	128 megabytes

After the recent apocalypse, Azamat, finally, learned about the largest common subsequences of two strings and now he is interested, what will be the largest common subsequence of two round words?

In a round word there is no difference from which symbol the word starts and in which direction it is read.

For instance, the round word “algorithm” can be read as “rithmalgo” and as “oglamhtir”.

For usual words “algorithm” and “grammar” the longest common subsequence length is 3 (the word “grm”), and for round variant of the same word the length of the longest common subsequence is 4 (the word “grma”).

Azamat quickly found out that the standard algorithm cannot get right answer for round words. Write the program which will do that for him.

Input

Two lines contain one word each. Words are non-empty and the length of each words doesn't exceed 2000 characters.

Output

The single line must contain one integer number — the length of the longest common subsequence of the given round words.

Examples

rowords.in	rowords.out
algorithm grammar	4

Problem H. Trading

Input file:	`trading.in`
Output file:	`trading.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

There are N small villages close to the highway between Almaty and Taraz numbered from 1 to N. At the beginning of the winter M unknown traders began trading knitted hats in these villages. They have only two rules: never trade in one place more than once (one day) and increase the price on hats each day.

More formally, each i-th trader:

1. begins trading in village L_i with starting price X_i.
2. each day he moves to the next adjacent village, i.e. if he was trading in village j yesterday, then today he is trading in village $j+1$.
3. each day he increases the price by 1, so if yesterday's price was x, then today's price is $x+1$.
4. stops trading at village R_i (after he traded his knitted hats in village R_i).

The problem is for each village to determine the maximal price that was there during the whole trading history.

Input

Each line contains two integer number N ($1 \le N \le 300000$) and M ($1 \le M \le 300000$) — number of villages and traders accordingly.

Next M lines contains 3 numbers each: L_i, R_i ($1 \le L_i \le R_i \le N$) and X_i ($1 \le X_i \le 10^9$) — numbers of first and last village and starting price for i-th trader.

Output

Output N integer numbers separating them with spaces — i-th number being the maximal price for the trading history of i-th village. If there was no trading in some village, output 0 for it.

Examples

trading.in	trading.out
5 2 1 3 2 2 4 6	2 6 7 8 0
6 4 4 4 3 1 2 5 5 6 1 6 6 1	5 6 0 3 1 2