E. Distributing Parts

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are an assistant director in a new musical play. The play consists of n musical parts, each part must be performed by exactly one actor. After the casting the director chose m actors who can take part in the play. Your task is to assign the parts to actors. However, there are several limitations.

First, each actor has a certain voice range and there are some parts that he cannot sing. Formally, there are two integers for each actor, c_i and d_i ($c_i \le d_i$) — the pitch of the lowest and the highest note that the actor can sing. There also are two integers for each part — a_j and b_j ($a_j \le b_j$) — the pitch of the lowest and the highest notes that are present in the part. The i-th actor can perform the j-th part if and only if $c_i \le a_j \le b_j \le d_i$, i.e. each note of the part is in the actor's voice range.

According to the contract, the i-th actor can perform at most k_i parts. Besides, you are allowed not to give any part to some actors (then they take part in crowd scenes).

The rehearsal starts in two hours and you need to do the assignment quickly!

Input

The first line contains a single integer n — the number of parts in the play ($1 \le n \le 10^5$).

Next n lines contain two space-separated integers each, a_j and b_j — the range of notes for the j-th part ($1 \le a_j \le b_j \le 10^9$).

The next line contains a single integer m — the number of actors ($1 \le m \le 10^5$).

Next m lines contain three space-separated integers each, c_i, d_i and k_i — the range of the i-th actor and the number of parts that he can perform ($1 \le c_i \le d_i \le 10^9$, $1 \le k_i \le 10^9$).

Output

If there is an assignment that meets all the criteria aboce, print a single word "YES" (without the quotes) in the first line.

In the next line print n space-separated integers. The i-th integer should be the number of the actor who should perform the i-th part. If there are multiple correct assignments, print any of them.

If there is no correct assignment, print a single word "NO" (without the quotes).

Examples

input

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

output

```
YES
1 1 2
```

input

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

output

```
NO
```

A permutation is a sequence of integers p_1, p_2, $\ldots$, p_n, consisting of n distinct positive integers and each of them does not exceed n. Assume that $r(S)$ of sequence S denotes the number of inversions in sequence S (if $i < j$ and $S_i > S_j$, then the pair of (i, j) is called an inversion of S), $l(S)$ of sequence S denotes the length of sequence S. Given a permutation P of length n, it's your task to find a subsequence S of P with maximum $\frac{r(S)}{l(S)}$. A subsequence of P is a sequence $(p_{i_1}, p_{i_2}, \ldots, p_{i_t})$ which satisfies that $0 < i_1 < i_2 < \ldots < i_t \leq n$.

Input

The first line of the input gives the number of test cases, T. T test cases follow.

For each test case, the first line contains an integer n ($1 \leq n \leq 100$), the length of the permutation P. The second line contains n integers p_1, p_2, $\ldots$, p_n, which represents the permutation P.

Output

For each test case, output one line containing '`Case #`x`:` y', where x is the test case number (starting from 1) and y is the maximum $\frac{r(S)}{l(S)}$.

Your answer will be considered correct if it is within an absolute error of 10^{-6} of the correct answer.

Sample Input

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

Sample Output

```
Case #1: 1.250000000000
```

C. Buns

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Lavrenty, a baker, is going to make several buns with stuffings and sell them.

Lavrenty has n grams of dough as well as m different stuffing types. The stuffing types are numerated from 1 to m. Lavrenty knows that he has a_i grams left of the i-th stuffing. It takes exactly b_i grams of stuffing i and c_i grams of dough to cook a bun with the i-th stuffing. Such bun can be sold for d_i tugriks.

Also he can make buns *without stuffings*. Each of such buns requires c_0 grams of dough and it can be sold for d_0 tugriks. So Lavrenty can cook any number of buns with different stuffings or without it unless he runs out of dough and the stuffings. Lavrenty throws away all excess material left after baking.

Find the maximum number of tugriks Lavrenty can earn.

Input

The first line contains 4 integers n, m, c_0 and d_0 $(1 \le n \le 1000, 1 \le m \le 10, 1 \le c_0, d_0 \le 100)$. Each of the following m lines contains 4 integers. The i-th line contains numbers a_i, b_i, c_i and d_i $(1 \le a_i, b_i, c_i, d_i \le 100)$.

Output

Print the only number — the maximum number of tugriks Lavrenty can earn.

Examples

input

```
10 2 2 1
7 3 2 100
12 3 1 10
```

output

```
241
```

input

```
100 1 25 50
15 5 20 10
```

output

```
200
```

Note

To get the maximum number of tugriks in the first sample, you need to cook 2 buns with stuffing 1, 4 buns with stuffing 2 and a bun without any stuffing.

In the second sample Lavrenty should cook 4 buns without stuffings.

- Announcement
- Tutorial

Chess inspired problems are a common source of exercises in algorithms classes. Starting with the well known 8-queens problem, several generalizations and variations were made. One of them is the N-rooks problem, which consists of placing N rooks in an N by N chessboard in such a way that they do not attack each other.

Professor Anand presented the N-rooks problem to his students. Since rooks only attack each other when they share a row or column, they soon discovered that the problem can be easily solved by placing the rooks along a main diagonal of the board. So, the professor decided to complicate the problem by adding some pawns to the board. In a board with pawns, two rooks attack each other if and only if they share a row or column and there is no pawn placed between them. Besides, pawns occupy some squares, which gives an additional restriction on which squares the rooks may be placed on.

Given the size of the board and the location of the pawns, tell Professor Anand the maximum number of rooks that can be placed on empty squares such that no two of them attack each other.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N ($1 \leq N \leq 100$) representing the number of rows and columns of the board. Each of the next N lines contains a string of N characters. In the i-th of these strings, the j-th character represents the square in the i-th row and j-th column of the board. The character is either '`.`' (dot) or the uppercase letter '`X`', indicating respectively an empty square or a square containing a pawn.

Output

For each test case, output a line with an integer representing the maximum number of rooks that can be placed on the empty squares of the board without attacking each other.

Sample Input

```
5
X....
X....
..X..
.X...
....X
4
....
.X..
....
....
1
X
```

Sample Output

```
7
5
0
```

B. Inventory

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Companies always have a lot of equipment, furniture and other things. All of them should be tracked. To do this, there is an inventory number assigned with each item. It is much easier to create a database by using those numbers and keep the track of everything.

During an audit, you were surprised to find out that the items are not numbered sequentially, and some items even share the same inventory number! There is an urgent need to fix it. You have chosen to make the numbers of the items sequential, starting with 1. Changing a number is quite a time-consuming process, and you would like to make maximum use of the current numbering.

You have been given information on current inventory numbers for n items in the company. Renumber items so that their inventory numbers form a *permutation* of numbers from 1 to n by changing the number of as few items as possible. Let us remind you that a set of n numbers forms a *permutation* if all the numbers are in the range from 1 to n, and no two numbers are equal.

Input

The first line contains a single integer n — the number of items ($1 \le n \le 10^5$).

The second line contains n numbers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^5$) — the initial inventory numbers of the items.

Output

Print n numbers — the final inventory numbers of the items in the order they occur in the input. If there are multiple possible answers, you may print any of them.

Examples

input
```
3
1 3 2
```
output
```
1 3 2
```

input
```
4
2 2 3 3
```
output
```
2 1 3 4
```

input
```
1
2
```
output
```
1
```

Note

In the first test the numeration is already a permutation, so there is no need to change anything.

In the second test there are two pairs of equal numbers, in each pair you need to replace one number.

In the third test you need to replace 2 by 1, as the numbering should start from one.

In the popular board game *One Night Werewolf*, players are distributed randomly in the roles of villagers and Werewolves. The goal of the villagers is to decide together on one person to kill during the night — hopefully they will kill a Werewolf. Werewolves pose as villagers in the hope that the person killed is a villager, not a Werewolf.

In the variation *Uncertain Werewolf*, only one Werewolf exists and the game consists of two phases. During the first phase the players are still uncertain about who they should vote to kill, so each of them chooses two other players as possible victims. After the first phase the Werewolf reveals himself, and then in the second phase each player has to decide which one of their two initial choices they will vote to kill. The Werewolf is the last one to decide between his two initial choices, doing so after all the other players have decided already.

The Werewolf then loses the game if he has more votes than anyone else. If there is a draw, the Werewolf wins.

You are given the votes of N players after the first phase of the game. You should answer how many players could reveal themselves at this point as the Werewolf and still win the game if the other players chose their votes optimally to kill the Werewolf.

Input

The input file contains several test cases, each of them as described below.

The first line contains an integer N ($3 \leq N \leq 50$), the number of players in the game. Each of the following N lines contains two integers, a_i and b_i ($1 \leq a_i, b_i \leq N$, $a_i \neq b_i$), the index of the players the i-th player decided to kill in the first voting phase. No player will try to kill himself.

Output

For each test case, output a line with one integer representing the number of players that could win the game if they were the Werewolf and everyone played optimally.

Sample Input

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

Sample Output

```
4
2
```

There are four possible blood groups for humans: **AB**, **A**, **B** and **O**, meaning that the red blood cells have antigens of types, respectively, A and B, only A, only B, and no antigen at all. Our blood group is determined by two alleles in our DNA. Each allele is of type either A, B or O. The following table lists the possible allele combinations someone may have for each blood group:

Blood group	**AB**	**A**	**B**	**O**
Possible alleles	AB	OA,AA	OB,BB	OO

We inherit exactly one allele from each of our two parents. So, given the blood groups of the two parents, we can say for sure if some blood group is possible, or not, in their offspring. For example, if the blood groups of the two parents are **AB** and **B**, then the possible allele combinations for them are, respectively, {AB} and {OB,BB}. Since the order of the alleles does not matter, the possible allele combinations for the offspring are {OA,AB,OB,BB}. That means the blood groups **AB**, **A** and **B** are possible in their offspring, but the blood group **O** is not. Very nice indeed! But what if life on Earth had evolved so that a person had three parents, three alleles, and three different antigen types? The allele combinations would look like this:

Blood group	**ABC**	**AB**	**AC**	**BC**	**A**	**B**	**C**	**O**
Possible alleles	ABC	OAB,AAB ABB	OAC,AAC ACC	OBC,BBC BCC	OOA,OAA AAA	OOB,OBB BBB	OOC,OCC CCC	OOO

If the blood groups of the three parents are **A**, **BC** and **O**, then all blood groups are possible in their offspring, except groups **BC** and **ABC**.

The universe is vast! There may be, out there in space, some form of life whose individuals have N parents, N alleles, and N different antigen types. Given the blood groups for the N parents, and a list of Q blood groups to test, your program has to determine which ones are possible, and which ones are not, in the offspring of the given parents.

Input

The input contains several test cases; each test case is formatted as follows. The first line contains two integers N and Q, representing respectively the number of parents (and alleles, and antigen types) and the number of queries ($1 \leq N \leq 100$ and $1 \leq Q \leq 40$). Each of the next N lines describes the blood group of a parent. After that, each of the next Q lines describes a blood group to test. Antigen types are identified with distinct integers from 1 to N, not letters. Each line describing a blood group contains an integer B indicating the number of antigen types in the blood group ($0 \leq B \leq N$), followed by B different integers $C_1, C_2, \ldots, C_B$ representing the antigen types present in the blood group ($1 \leq C_i \leq N$ for $i = 1, 2, \ldots, B$).

Output

For each of the Q queries, output a line with the uppercase letter 'Y' if the corresponding blood group is possible in the offspring of the given parents; otherwise output the uppercase letter 'N'. Write the results in the same order that the queries appear in the input.

Sample Input

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

Sample Output

G. Petya and Graph

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya has a simple graph (that is, a graph without loops or multiple edges) consisting of n vertices and m edges.

The weight of the i-th vertex is a_i.

The weight of the i-th edge is w_i.

A subgraph of a graph is some set of the graph vertices and some set of the graph edges. The set of edges must meet the condition: both ends of each edge from the set must belong to the chosen set of vertices.

The weight of a subgraph is the sum of the weights of its edges, minus the sum of the weights of its vertices. You need to find the maximum weight of subgraph of given graph. **The given graph does not contain loops and multiple edges**.

Input

The first line contains two numbers n and m ($1 \le n \le 10^3, 0 \le m \le 10^3$) - the number of vertices and edges in the graph, respectively.

The next line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) - the weights of the vertices of the graph.

The following m lines contain edges: the i-e edge is defined by a triple of integers v_i, u_i, w_i ($1 \le v_i, u_i \le n, 1 \le w_i \le 10^9, v_i \neq u_i$). This triple means that between the vertices v_i and u_i there is an edge of weight w_i. It is guaranteed that the graph does not contain loops and multiple edges.

Output

Print one integer — the maximum weight of the subgraph of the given graph.

Examples

input Copy
```
4 5
1 5 2 2
1 3 4
1 4 4
3 4 5
3 2 2
4 2 2
```
output Copy
```
8
```

input Copy
```
3 3
9 7 8
1 2 1
2 3 2
1 3 3
```
output Copy

Educational Codeforces Round 55 (Rated for Div. 2)

Finished

Practice

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ACM-ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++14 6.4.0

Choose file: Browse... No file selected.

Submit

→ Last submissions

Submission	Time	Verdict
46340637	Nov/28/2018 19:33	Accepted

→ Problem tags

flows graphs *2400

No tag edit access

→ Contest materials

- Announcement
- Tutorial

```
0
```

Note

In the first test example, the optimal subgraph consists of the vertices $1, 3, 4$ and has weight $4 + 4 + 5 - (1 + 2 + 2) = 8$ In the second test case, the optimal subgraph is empty.

D. Delivery Bears

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Niwel is a little golden bear. As everyone knows, bears live in forests, but Niwel got tired of seeing all the trees so he decided to move to the city.

In the city, Niwel took on a job managing bears to deliver goods. The city that he lives in can be represented as a directed graph with n nodes and m edges. Each edge has a weight capacity. A delivery consists of a bear carrying weights with their bear hands on a simple path from node 1 to node n. The total weight that travels across a particular edge must not exceed the weight capacity of that edge.

Niwel has **exactly** x bears. In the interest of fairness, no bear can rest, and the weight that each bear carries must be exactly the same. However, each bear may take different paths if they like.

Niwel would like to determine, what is the maximum amount of weight he can deliver (it's the sum of weights carried by bears). Find the maximum weight.

Input

The first line contains three integers n, m and x ($2 \le n \le 50$, $1 \le m \le 500$, $1 \le x \le 100\,000$) — the number of nodes, the number of directed edges and the number of bears, respectively.

Each of the following m lines contains three integers a_i, b_i and c_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le c_i \le 1\,000\,000$). This represents a directed edge from node a_i to b_i with weight capacity c_i. There are no self loops and no multiple edges from one city to the other city. More formally, for each i and j that $i \ne j$ it's guaranteed that $a_i \ne a_j$ or $b_i \ne b_j$. It is also guaranteed that there is at least one path from node 1 to node n.

Output

Print one real value on a single line — the maximum amount of weight Niwel can deliver if he uses exactly x bears. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Namely: let's assume that your answer is a, and the answer of the jury is b. The checker program will consider your answer correct if $\frac{|a-b|}{max(1,b)} \le 10^{-6}$.

Examples

input Copy
```
4 4 3
1 2 2
2 4 1
1 3 1
3 4 2
```
output Copy
```
1.5000000000
```

input Copy
```
5 11 23
1 2 3
2 3 4
3 4 5
4 5 6
```

```
1 3 4
2 4 5
3 5 6
1 4 2
2 5 3
1 5 2
3 2 30
```

output

```
10.2222222222
```

Note

In the first sample, Niwel has three bears. Two bears can choose the path $1 \rightarrow 3 \rightarrow 4$, while one bear can choose the path $1 \rightarrow 2 \rightarrow 4$. Even though the bear that goes on the path $1 \rightarrow 2 \rightarrow 4$ can carry one unit of weight, in the interest of fairness, he is restricted to carry 0.5 units of weight. Thus, the total weight is 1.5 units overall. Note that even though Niwel can deliver more weight with just 2 bears, he must use exactly 3 bears on this day.

A. Crazy Town

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Crazy Town is a plane on which there are n infinite line roads. Each road is defined by the equation $a_i x + b_i y + c_i = 0$, where a_i and b_i are not both equal to the zero. The roads divide the plane into connected regions, possibly of infinite space. Let's call each such region a block. We define an intersection as the point where at least two different roads intersect.

Your home is located in one of the blocks. Today you need to get to the University, also located in some block. In one step you can move from one block to another, if the length of their common border is nonzero (in particular, this means that if the blocks are adjacent to one intersection, but have no shared nonzero boundary segment, then it are not allowed to move from one to another one in one step).

Determine what is the minimum number of steps you have to perform to get to the block containing the university. It is guaranteed that neither your home nor the university is located on the road.

Input

The first line contains two space-separated integers x_1, y_1 ($-10^6 \le x_1, y_1 \le 10^6$) — the coordinates of your home.

The second line contains two integers separated by a space x_2, y_2 ($-10^6 \le x_2, y_2 \le 10^6$) — the coordinates of the university you are studying at.

The third line contains an integer n ($1 \le n \le 300$) — the number of roads in the city. The following n lines contain 3 space-separated integers ($-10^6 \le a_i, b_i, c_i \le 10^6$; $|a_i| + |b_i| > 0$) — the coefficients of the line $a_i x + b_i y + c_i = 0$, defining the i-th road. It is guaranteed that no two roads are the same. In addition, neither your home nor the university lie on the road (i.e. they do not belong to any one of the lines).

Output

Output the answer to the problem.

Examples

input Copy

```
1 1
-1 -1
2
0 1 0
1 0 0
```

output Copy

```
2
```

input Copy

```
1 1
-1 -1
3
1 0 0
0 1 0
1 1 -3
```

output Copy

```
2
```

Note

Pictures to the samples are presented below (A is the point representing the house; B is the point representing the university, different blocks are filled with different colors):