Problem A

Winning ICPC

There are N teams (numbered from 1 to N) and M problems (numbered from 1 to M) in this year's ICPC. The j-th problem has T_j testcases. Surprisingly, every team submitted exactly one solution to every problem. The i-th team managed to solve $S_{i,j}$ testcases on the j-th problem.

A team solved a problem only if the team managed to solve ALL testcases on that problem. The winning team is the team with the most number of problems solved. If there are more than one team with the most number of problems solved, then the winning team is the team with the smallest index among those teams.

Determine the index of the winning team.

Input

The first line contains two integers: N M ($1 \le N, M \le 100$) in a line denoting the number of teams and the number of problems. The second line contains M integers: $T_1\ T_2\ \cdots\ T_M$ ($0 \le T_i \le 100$) in a line denoting the number of testcases. The next N following lines, each contains M integers; the j-th integer on the i-th line is $S_{i,j}$ ($0 \le S_{i,j} \le T_j$) denoting the number of solved testcases by the i-th team for the j-th problem.

Output

The output contains the index of the winning team, in a line.

Sample Input	Output for Sample Input

`3 2`
`10 20`
`0 19`
`10 0`
`9 19`
	`2`

`3 2`
`10 20`
`0 20`
`10 0`
`9 19`
	`1`

`1 1`
`1`
`0`
	`1`

Explanation for the 1st sample case

On the first sample, the first and the third team did not solve any problem, and the second team solved the first problem. Therefore, the second team is the winner.

Explanation for the 2nd sample case

On the second sample, the first team solved the second problem, the second team solved the first problem, and the third team did not solve any problem. Since the first team has a smaller index than the second team, the first team is the winner.

Explanation for the 3rd sample case

On the third sample, there is only one team thus the winner is obvious.

Problem B

Travelling Businessmen Problem

There are N cities (numbered from 1 to N) connected by M bidirectional roads such that from any city, it is possible to reach any other cities with one or more roads. The i-th city has an economic value of S_i, and each road directly connects two different cities.

There are Q queries, which should be executed one by one, each represented by a tuple (A_i, B_i, C_i).

1. If A_i = 0, then you need to change the economic value of city B_i to C_i.
2. If A_i = 1, then you need to output the answer to the following question: Suppose there are two businessmen, one of them is in city B_i, while the other is in city C_i. They both agree to each other on a non-negative integer X, where X is the number of days in which they are moving around. Each day, both businessmen move to any city adjacent to the current city they are in, and they repeat this routine for X days. They cannot stay in the same city for two consecutive days, but they may revisit cities which have been visited before. After X days, they should be in the cities such that the difference of the economic value between those two cities are minimum. Output the minimum difference. Note that the two cities can be the same city.

Input

The first line contains two integers: N M ($1 \leq N \leq 100{,}000$; $1 \leq M \leq 200{,}000$) in a line denoting the number of cities and roads. The second line contains N integers: S_1 S_2 $\cdots$ S_N ($0 \leq S_i \leq 1{,}000{,}000{,}000$) in a line denoting the economic value of each city. The next M following lines, each contains two integers: u_i v_i ($1 \leq u_i$, $v_i \leq N$; $u_i \neq v_i$) in a line, which means the i-th road connects city u_i and city v_i. It is guaranteed that from any city, it is possible to reach any other cities by using one or more roads. The next line contains an integer: Q ($1 \leq Q \leq 100{,}000$) denoting the number of queries. The next Q lines, each contains three integers: A_i B_i C_i ($0 \leq A_i \leq 1$) in a line denoting the queries. For each query, if A_i = 0, then $1 \leq B_i \leq N$ and $0 \leq C_i \leq 1{,}000{,}000{,}000$; otherwise $1 \leq B_i$, $C_i \leq N$. There will be at least one query where A_i = 1.

Output

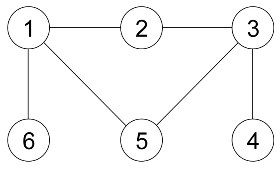
For each query where A_i = 1, output the minimum difference of the economic value of the two cities that can be reached by the businessmen after X days, in a line. Note that the value of X is any non-negative integer and independent between different queries.

Sample Input	Output for Sample Input

```
6 6
0 0 0 0 0 0
1 2
1 6
5 1
2 3
3 4
3 5
7
1 1 2
0 1 10
0 3 20
1 1 2
0 4 11
1 1 3
1 1 6
```
	```
	0
	10
	0
	1
	```

Explanation for the 1st sample case

The following are the road configuration and the queries for the first sample:



- Originally all cities have an economic value of 0.
- 1st query: 1 1 2. As the economic value of all cities is the same (i.e. 0), then they can decide on any X and move to any cities they want; the difference will be 0.
- 4th query: 1 1 2. This query is the same as the 1st query, but now city 1 has an economic value of 10 (due to the 2nd query), while city 3 has an economic value of 20 (due to the 3rd query). The minimum difference of 10 can be achieved if they choose X to be 0. Follows are some examples where the difference is NOT minimum:
  - X = 1, the first businessman moves from city 1 to city 5 (with an economic value of 0), while the second businessman moves from city 2 to city 3 (with an economic value of 20). The difference of the economic value of the two final cities is 20.
  - X = 2, the first businessman moves from city 1, to city 5, to city 3 (with an economic value of 20), while the second businessman moves from city 2, to city 3, to city 4 (with an economic value of 0). The difference of the economic value of the two final cities is 20.
  - It is possible to achieve a minimum difference of 10 with X = 1, e.g., the first businessman moves from city 1 to city 2 (with an economic value of 0), while the second businessman moves from city 2 to city 1 (with an economic value of 10). It is also possible to achieve this with X = 2, e.g., the first businessman moves from city 1, to city 5, to city 1 (with an economic value of 10), while the second businessman moves from city 2, to city 3, to city 2 (with an economic value of 0). Among all possible moves, the minimum difference which can be achieved in this case is 10.
- 6th query: 1 1 3. The two businessmen can choose X = 1 and move to city 2 (or city 5) altogether.
- 7th query: 1 1 6. They can choose X = 3 and move such that the final cities are city 4 (with an economic value of 11) and city 1 (with an economic value of 10).

Problem C
Non-Interactive Guessing Number

Romanos: "Can I submit an interactive problem to ACM-ICPC 2017 Jakarta?"
Theodora: "No."
Romanos: "Aww, man.. That's not fun."
Then Romanos decided to submit a non-interactive version of his problem to the contest; and here it is, based on his playing with Theodora.

Romanos and Theodora are playing a game. Initially, Theodora picks a number between 1 to N inclusive. Romanos' goal is to determine that number. He can make up to K guesses. For each guess, he will say a number out loud as his guess. Theodora will then say one of the following answers based on the exact condition:

- "My number is smaller than your guess (<)",
- "My number is greater than your guess (>)", or
- "Your guess is correct (=)".

The game will end right after one of the following:

- Romanos guesses the correct number (he wins), or
- All K guesses are wrong (he loses).

Sadly, Romanos is not playing the game seriously. He knows the best strategy to win this game, but he does not always use it. Nevertheless, he pretends that he plays seriously, hence his guess will never be a dumb one. In other words, his guess will always be a number between 1 to N inclusive and will always be consistent with all of Theodora's previous answers.

For example, suppose N is 10 and Romanos' first guess is 4. If Theodora answers "My number is smaller than your guess (<)" then the next Romanos's guess will always be between 1 to 3, inclusive.

Contrast to Romanos, Theodora is playing the game too seriously. She wants to win (by making Romanos lose) and "cheats" as follows.

Adaptively, Theodora might change her number as far as it is consistent with all of her previous answers. To do that, whenever possible, Theodora will always answer either "My number is smaller than your guess (<)" or "My number is greater than your guess (>)" that maximizes the possible answer range. In the case of tie, she will always answer the first one (<).

For example, suppose N is 10 and Romanos' first guess is 4. Theodora will always answer "My number is greater than your guess (>)" because the possible answer range will be between 5 to 10 inclusive; it is larger than 1 to 3 inclusive.

Now, this is the actual problem proposed by Romanos. You are given N, K, and Theodora's answer for each guess. Can you show one of the possible scenario for Romanos' guesses or state that it is impossible?

Input

The first line contains two integers: N K ($1 \le N \le 10^{18}$; $1 \le K \le 50{,}000$) in a line denoting the number range and the number of guesses. The second line contains a string in a line denoting Theodora's answers. Each character of the string is either '<', '>', or '=', and either:

- The last character of the string is '=' and each character of the string other than the last character is either '<' or '>', and the length of the string is not more than K, or
- The length of the string is exactly K and each character of the string is either '<' or '>'.

Output

- If there is a possible scenario for Romanos' guesses, the output contains M integers: A_1 A_2 $\cdots$ A_M in a line denoting Romanos' guesses where M is the length of the string and A_i is Romanos' i-th guess. If there are more than one possible scenario for Romanos' guesses, you may output any of them.
- If there is no possible scenario for Romanos' guesses, just output -1 in a line.

Sample Input	Output for Sample Input

`10 5`
`><>=`
	`5 8 6 7`

`10 5`
`><<><`
	`4 10 8 5 7`

`10 5`
`<>=`
	`-1`

`10 9`
`>>>>>>>>>`
	`1 2 3 4 5 6 7 8 9`

`10 10`
`>>>>>>>>>>`
	`-1`

Problem D

Make a Forest

In 1736, Leonhard Euler wrote a paper on the Seven Bridges of Königsberg which is regarded as the first paper in the history of graph theory. Nowadays, the study of graph theory is considered very important as indicated by the fact that most textbooks in discrete mathematics have a chapter on graph theory.

This problem is related to graph theory, especially on tree and forest. Given N tuples (u_i, v_i, w_i), your task is to construct a forest with a minimum number of trees which satisfies the following seven requirements:

1. Each tree in the forest is a rooted tree;
2. Each node x in the forest has a value x.A;
3. Each edge (x, y) in the forest has a value (x, y).B;
4. Each tuple (u_i, v_i, w_i) appears exactly once in the forest as two nodes with a parent-child relationship (parent node p and child node c) where: u_i = p.A, v_i = c.A, and w_i = (p, c).B;
5. For any non-root and non-leaf node x in the forest, (p, x).B is smaller than any (x, c).B, where p is x's parent and c is x's child;
6. All nodes in the forest have at most M children.
7. The forest should contain exactly N edges.

To simplify the problem, it is guaranteed that w_i in any tuple is unique, i.e. no two tuples with the same w_i.

Output the number of trees in such forest (the forest should have the minimum number of trees).

Input

The first line contains two integers: N M ($1 \le N, M \le 100{,}000$) in a line denoting the number of tuples and the maximum number of children for each node in the forest. The next N following lines, each contains three integers: u_i v_i w_i ($1 \le u_i, v_i \le 2{,}000{,}000{,}000$; $1 \le w_i \le N$) in a line denoting the tuple (u_i, v_i, w_i). It is guaranteed that there will be no two tuples with the same w_i.

Output

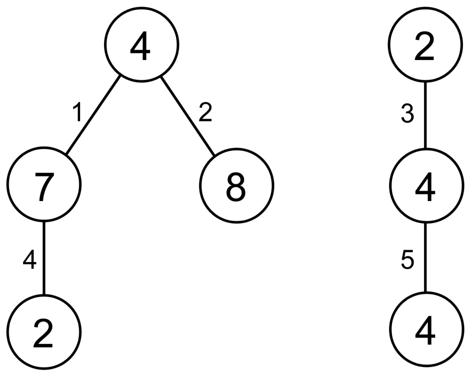
The output contains an integer denoting the number of trees in a forest with a minimum number of trees which satisfies the given requirements, in a line.

Sample Input	Output for Sample Input
5 2	2
2 4 3	
4 4 5	
4 7 1	
7 2 4	
4 8 2	

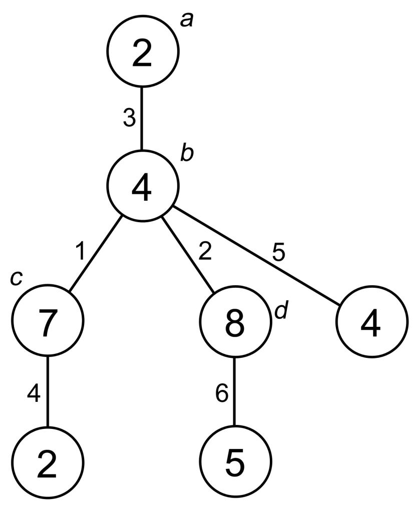
5 1
2 4 3
4 4 5
4 7 1
7 2 4
4 8 2
	3

5 10
1000 3000 3
2000 4000 5
1000 2000 1
3000 2000 4
2000 3000 2
	1

Explanation for the 1st sample case



For the first sample, this forest is the only forest which satisfies all the requirements. There are 2 trees in this forest.



On the other hand, this forest does not satisfy the requirements due to:

1. Node b violates requirement #5 as (a,b).B = 3 is larger than (b,c).B = 1 and (b,d).B = 2.
2. Node b violates requirement #6 as it has 3 children (note that M is 2).
3. There are 6 edges in the forest while N = 5 (violates requirement #7).

Note that violating even one requirement already makes the forest invalid.

Problem E
Parks of Jakarta

There are three parks in Jakarta, called Park 1, Park 2, and Park 3. The new governor of Jakarta wants to add decorations to these parks by adding bricks. These bricks are going to be stacked on top of another on these three parks. There are N bricks in Jakarta, indexed from 1 to N. A brick with index i is smaller than a brick with index $i + 1$. We do not want to put a bigger brick on top of a smaller brick. Therefore, we can put a brick with index i on top of a brick with index j if and only if $i < j$.

We define a brick configuration as a distribution of these N bricks on the three parks. Note that there must only be one stack of bricks in each park, and the bricks must be stacked in the order as explained on the previous paragraph. For example, if $N = 3$, then a possible brick configuration is: brick 1 (on the top) and brick 2 (on the bottom) are in park 1, no brick is in park 2, and brick 3 is in park 3.

We can change a brick configuration to another by doing an operation, which we will:

- Pick two different parks i and j. Take the topmost brick on the stack in park i and put it on the top of the stack in park j. If there is no brick in park i, then this operation is not valid. If this operation causes the stack in park j to violate the stack order condition, then this operation is also not valid.
- Since we need to rent a truck to transport the brick from park i to park j, we need to pay $R_{i,j}$ units of money for this operation. Note that the cost of transporting a brick from park i to park j might be different with the cost of transporting a brick from park j to park i.

Initially, the bricks are in some brick configuration, which we will call the initial brick configuration. The new governor of Jakarta wants to try M brick configurations. Therefore, from the initial brick configuration, we must do several operations such that each of the M brick configurations (in any order) wanted by the new governor is seen at least once. In the end, the new governor also wants us to put all the bricks in (any) one park. We want to minimise the total cost to satisfy his demand.

Formally, let $G_1, G_2, \cdots, G_M$ be the brick configurations wanted by the new governor. We want to construct a sequence of brick configurations $C_0, C_1, \cdots, C_k$ ($k \geq 0$) with the minimum cost where:

- C_0 is the initial brick configuration
- There exists a sequence of integers $x_1, x_2, \cdots, x_M$ ($0 \leq x_i \leq k$) where $C_{x_i} = G_i$. Note that the sequence is not necessarily increasing.
- C_k is a brick configuration where all the bricks are stacked in (any) one of the parks.
- For all $0 \leq i < k$, we can achieve C_{i+1} from C_i by doing a single operation (described in paragraph three above).
- The cost of this sequence is the sum of cost needed to change C_i to C_{i+1} for all $0 \leq i < k$.

Let us help the new governor of Jakarta!

Input

The first line contains two integers: N M ($1 \leq N \leq 40$; $0 \leq M \leq 16$) in a line denoting the number of bricks and the number of configurations. The next three lines, each contains three integers. The j-th integer on the i-th line is $R_{i,j}$ ($0 \le R_{i,j} \le 1000$; $R_{i,i} = 0$) denoting the cost of moving a brick from the i-th city to the j-th city. The next three lines denote the initial brick configuration, with the format explained in the next paragraph. The next M blocks describe the configurations wanted by the new governor, where each block contains three lines, with the format of each configuration explained in the next paragraph.

Each configuration is denoted by three lines. The i-th line begins with an integer K ($0 \le K \le N$) denoting the number of bricks in park i, followed by K integers: $A_1\ A_2\ \cdots\ A_K$ ($1 \le A_i \le N$) denoting the brick indices in park i. It is guaranteed that $A_i < A_j$ for all $1 \le i < j \le K$. It is also guaranteed that each of the integers between 1 and N, inclusive, appears exactly once on the union of the array A on the three lines. For example, the sample brick configuration given on the second paragraph will be illustrated in the first sample input.

Output

The output contains the minimum total cost (in units of money) to satisfy the new governor's demand, in a line.

Sample Input	Output for Sample Input	Sample Input	Output for Sample Input

`3 0`
`0 1 1000`
`1 0 1`
`1000 1 0`
`2 1 2`
`0`
`1 3`
	`5`	`3 2`
		`0 2 2`
		`2 0 2`
		`2 2 0`
		`2 1 2`
		`1 3`
		`0`
		`3 1 2 3`
		`0`
		`0`
		`0`
		`0`
		`3 1 2 3`
			`22`

Explanation for the 1st sample case

In the first sample, we initially have brick 1 and brick 2 in park 1, and brick 3 in park 3. Since $M = 0$, our goal is to only reach a brick configuration where all bricks are in the same (any) park. Therefore, we can do the following operations:

- Move brick 3 from park 3 to park 2. This costs 1 unit of money.
- Move brick 1 from park 1 to park 2. This costs 1 unit of money.
- Move brick 1 from park 2 to park 3. This costs 1 unit of money.
- Move brick 2 from park 1 to park 2. This costs 1 unit of money.
- Move brick 1 from park 3 to park 2. This costs 1 unit of money.

Therefore, all of the bricks are now in park 2 and we spend 5 units of money. There is no solution that costs less than 5 units of money. Note that we do not necessarily minimise the number of operations.

Explanation for the 2nd sample case

In the second sample, the optimal solution can be achieved if we satisfy the second configuration first before the first one. From the initial configuration, we can get to the second configuration with 4 operations with the total cost of 8 units. From the second configuration, we can get to the first configuration with 7 operations with the total cost of 14 units. Since the first configuration already has all the bricks stacked in one park, we don't need any additional operation. Therefore, the total cost is 22 units.

Problem F

Random Number Generator

Aristides has a random number generator. Each time he asks the random number generator, it will return a random integer between 1 to N inclusive uniformly.

Aristides' friend, Iorgos, will note the returned number one by one. Aristides would keep asking a random number until Iorgos stop him. To analyse the randomness of the random number generator, he would stop Aristides immediately after each number from 1 to N is returned at least twice.

Aristides has already asked K numbers to the random number generator. The i-th number returned by the random number generator is A_i. Since Iorgos does not stop him yet, he knows that at least one of the integer between 1 to N inclusive has not been returned at least twice.

Aristides wants to know the expected number of additional numbers to be asked until Iorgos stopped him.

Input

The first line contains an integer: T ($1 \le T \le 100{,}000$) denoting the number of testcases. The first line of each testcase contains two integers: N K ($1 \le N \le 3{,}000$; $0 \le K \le 100{,}000$) in a line denoting the range of the returned integers and the number of integers that is already asked. The second line of each testcase contains K integers: A_1 A_2 $\cdots$ A_K ($1 \le A_i \le N$) in a line denoting the first K returned integers. It is guaranteed that at least one of the integer between 1 to N inclusive has not been returned at least twice. It is also guaranteed that the sum of the value of K on all testcases is not more than 100,000.

Output

For each testcase, the output contains the expected number of additional numbers to be asked by Aristides until Iorgos stopped him, in a line. Your answer will be considered correct if the relative or absolute difference between your answer and judge's answer is not more than 10^{-6}.

Sample Input	Output for Sample Input

`4`
`1 0`
 	
`1 1`
`1`
`2 10`
`2 2 2 2 2 2 2 2 2 2`
`3 0`
	`2.000000000`
	`1.000000000`
	`4.000000000`
	`9.638888889`

Explanation for the 1st sample case

For the first sample, the first two numbers returned by the random number generator is always 1. Therefore, by asking the random number generator twice, each number from 1 to N will be returned at least twice.

Explanation for the 2nd sample case

For the second sample, Aristides already asked one number from the random number generator. Therefore, he only need to ask for one additional number.

Problem G
National Disaster: Two Towers

Despite its efforts, Indinesia is still regularly hit by uncontrolled forest fires, which causes problems to the local citizens as well as neighbouring countries. Many solutions have been proposed and executed (e.g., dropping water bombs on the burning area), but new hotspots always emerge over time. Nonetheless, Indra, the president of Indinesia, is not yet giving up in solving this problem.

Indinesia has two fire lookout towers located at (x_1, y_1) and (x_2, y_2) where $x_1 < x_2$ and $y_1 < y_2$. Moreover, there are N hotspots scattered across the country. The i-th hotspot is a circle with radius r_i and centered at (fx_i, fy_i). A point (x, y) is considered **safe** if it satisfies all the following properties:

1. x_1 ≤ x ≤ x_2,
2. y_1 ≤ y ≤ y_2,
3. It must not lie strictly inside any burning area; in other words, for all 1 ≤ i ≤ N, the distance of (x, y) to (fx_i, fy_i) should be at least r_i.

The locations of the two towers are guaranteed to be safe. The two towers can communicate properly if and only if there exists a safe path connecting the two towers. A path is considered as safe if and only if all points on the path are safe. Note that "path" as defined in this problem is any continuous and unnecessarily-straight line.

Your task in this problem is to determine whether the two towers are able to communicate properly.

Input

The first line of input contains five integers: x_1 y_1 x_2 y_2 N (-1,000,000 ≤ $x_1 < x_2$ ≤ 1,000,000; -1,000,000 ≤ $y_1 < y_2$ ≤ 1,000,000; 0 ≤ N ≤ 1000) in a line denoting the location of the two towers ((x_1, y_1) and (x_2, y_2)) and the number of hotspots. The next N lines, each contains three integers: fx_i fy_i r_i (-1,000,000 ≤ fx_i, fy_i ≤ 1,000,000; 1 ≤ r_i ≤ 2,000,000) in a line denoting the location of the center of the i-th hotspot ((fx_i, fy_i)) and its radius of burning area. It is guaranteed that there are no two hotspots at the same (fx_i, fy_i) location.

Output

The output contains either "`YES`" or "`NO`" (without quotes) whether the two towers can communicate properly, in a line.

Sample Input	Output for Sample Input

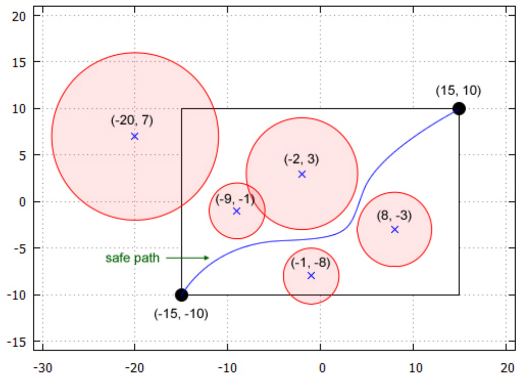
`-15 -10 15 10 5`
`-20 7 9`
`-2 3 6`
`8 -3 4`
`-1 -8 3`
`-9 -1 3`
	`YES`

`2 10 18 30 3`
`10 20 5`
`10 29 5`
`10 11 5`
	`NO`

`2 10 18 30 3`
`10 20 5`
`10 25 5`
`10 10 5`
	`YES`

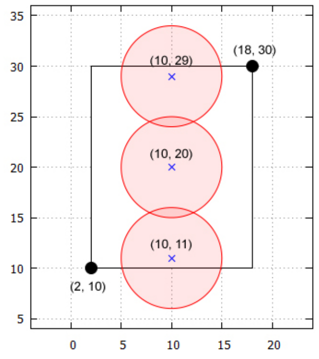
Explanation for the 1st sample case

For the first sample, the following figure shows one of the safe path connecting the two towers.



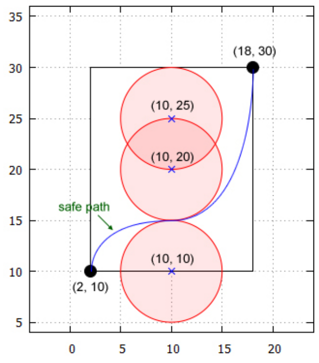
Explanation for the 2nd sample case

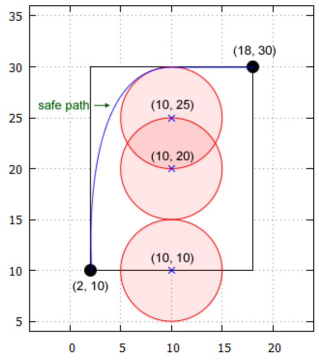
For the second sample, there is no possible safe path.



Explanation for the 3rd sample case

For the third sample, the following figures show two examples of safe paths connecting the two towers.





Notice that the points at (10, 15) in the top figure and (10, 30) in the bottom one are safe.

Problem H

ANTS

ANTS (Agency for Nonsensical Technology Storage) is a leading company specialising in storage for any nonsensical technology. These technologies cannot just be thrown away as it can be hazardous or dangerous, thus they needed to be stored and managed with care.

ANTS owns N warehouses (numbered from 1 to N) which are operated by advanced robots and connected by special rails. One special rail connects exactly two different warehouses and can be used by the robot to move between those two connected warehouses. As the cost to build the special rail at the moment is very expensive, ANTS only built a minimum number of special rails such that it is sufficient for the robots to move between any two warehouses.

Once in a while, the warehouses' manager needs to recalibrate some robots. In order to do this, all the affected robots need to assemble at any one warehouse altogether (the manager decided which warehouse he wants to use as the assembly point). As the cost to build one special rail is very expensive, the manager needs to minimise any unnecessary use of the special rails. In other words, the manager should choose a warehouse as the assembly point for the affected robots such that the number of special rail usage is as minimum as possible.

Consider the following example. Let there be 10 warehouses with the special rails are arranged in the following way (Figure 1).

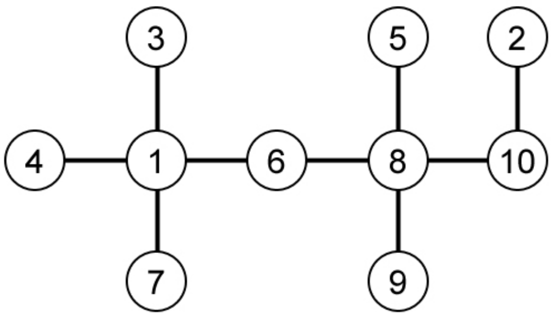


Figure 1.

Supposed the manager wants to recalibrate four robots which are located at warehouse number 4, 5, 7, and 9. If the manager chooses warehouse number 3 as the assembly point, then the special rails will be used 12 times (Figure 2). In this case, the special rail connecting warehouse 1 and 3 (1-3) in this case is used four times; 1-6 and 6-8 are used twice; 1-4, 1-7, 5-8, and 8-9 are used once. Thus, the total number of special rails usage is 1 * 4 + 2 * 2 + 4 * 1 = 12.

On the other hand, if the manager chooses warehouse number 6, then the special rails are used only 8 times (Figure 3), and this is the minimum possible for this example. Alternatively, the manager can choose warehouse number 1 or 8 as the assembly point as these choices will cause the special rails to be used 8 times as well.

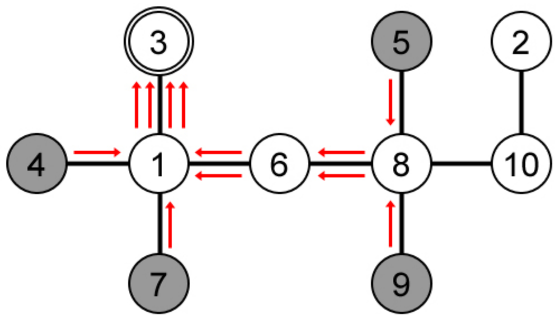


Figure 2.

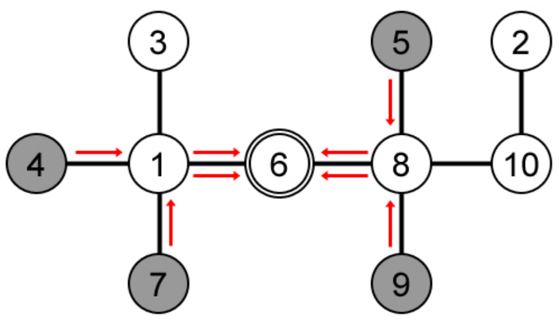


Figure 3.

Given the arrangement of N warehouses and Q queries where each query contains the location of K affected robots which are needed to be recalibrated altogether. For each query, determine the optimum assembly point for the affected robots such that the number of special rail usage is minimised (you only need to output the usage number).

Input

The first line contains an integer: N ($1 \le N \le 100{,}000$) denoting the number of warehouses. The following N - 1 lines, each contains two integers: a b ($1 \le a, b \le N$) in a line denoting a special rail connecting warehouse number a and b. You may safely assume that all the warehouses are connected to each other, i.e. there is a path from one warehouse to any other warehouse. The next line contains an integer: Q ($1 \le Q \le 5000$) denoting the number of queries. The next Q lines, each begins with an integer K ($1 \le K \le 50$) denoting the number of affected robots, followed by K integers: $A_1\ A_2 \cdots A_K$ ($1 \le A_i \le N$) in a line denoting the query (the warehouse location of the affected K robots).

Output

For each query, output in a line, the minimum number of special rail usage for that query.

Sample Input	Output for Sample Input

10
1 4
10 2
7 1
6 8
9 8
1 6
8 5
2 8
1 3
5
1 8
2 7 10
3 3 3 4
4 4 7 5 9
5 4 5 9 10 3
	0
	5
	2
	8
	10

Problem I
XEN 3166

There are N countries in this world, numbered from 1 to N. Each country has a country name and a country code. Both of them can be represented as a string. No two countries share the same country name. For simplicity, in this problem we assume that a string consists only uppercase English alphabets (A-Z).

One of the widely recognised country code standard is published by ISO (ISO 3166). However, Xenia has noticed something strange about ISO 3166. The resulting code is no longer alphabetical! For example, "INDIA" has “IN” as its country code while "INDONESIA" is “ID”. "INDONESIA" comes after "INDIA" in alphabetical order, but “IN" comes after “ID”.

Unhappy about this indecent property of the ISO 3166 country codes, Xenia seeks for a more consistent coding. She develops her own standard, called XEN 3166. The rules of XEN 3166 are:

- For each country, its country code must be a subsequence of its country name.
- For each country, its first letter of the country code must be the same as its first letter of the country name.
- For each country, the length of its country code must be exactly K.
- If a country with a country name S has a country code S' and a country with a country name T has a country code T', then S is lexicographically smaller than T if and only if S' is lexicographically smaller than T'.

A string $T = T_1T_2 \cdots T_{|T|}$ is a subsequence of a string $S = S_1S_2 \cdots S_{|S|}$ if there exists a sequence of integers $u_1, u_2, \cdots, u_{|T|}$ where $1 \le u_1 < u_2 < \cdots < u_{|T|} \le |S|$ and $S_{u_i} = T_i$ for all $1 \le i \le |T|$. For example, "ID", "IN", and "IND" are subsequences of "INDIA", but "IDN", "INN", "Z" are not subsequences of "INDIA".

String $S = S_1S_2 \cdots S_{|S|}$ is lexicographically smaller than string $T = T_1T_2 \cdots T_{|T|}$ if at least one of the following is true:

- $|S| < |T|$ and $S_i = T_i$ for all $1 \le i \le |S|$, or
- There exists an index i where $S_i < T_i$ and $S_j = T_j$ for all $1 \le j < i$.

For example, suppose there are only two countries and K = 2. The name of the first country is "INDIA" and the name of the second country is "INDONESIA". Therefore, several possible country codes assignment are:

- "ID" for "INDIA" and "IN" for “INDONESIA"
- "IA" for "INDIA" and "ID" for “INDONESIA"
- "II" for "INDIA" and "IO" for "INDONESIA"

Given N, K, and the list of country names. Help Xenia to assign the country codes for each country that follows the XEN 3166 rule, or state that it is impossible to do so.

Input

The first line contains two integers: N K ($1 \le N \le 1000$; $1 \le K \le 200$) in a line denoting the number of countries and the length of country codes. The next N following lines, each contains a string which length between 1 and 200,000 consisting of uppercase English alphabets (A-Z). The string on the i-th line denotes the country name of the i-th country. The sum of the length of all country names are guaranteed to be not more than 200,000. It is also guaranteed that no two countries share the same country name.

Output

- If it is possible for Xenia to assign the country codes for each country that follows the XEN 3166 rule, output "`YES`" (quotes only for clarity) on the first line. The next N following lines, each contains a string. The string on the i-th line denotes the country code of the i-th country. If there are more than one possible country codes assignment, you may output any of them.
- If it is not possible for Xenia to assign the country codes for each country that follows the XEN 3166 rule, just output "`NO`" (quotes only for clarity) in a single line.

Sample Input	Output for Sample Input

`2 2`
`INDIA`
`INDONESIA`
	`YES`
	`ID`
	`IN`

`3 2`
`IBAA`
`IAAA`
`IAAC`
	`NO`

Explanation for the 1st sample case

The scenario in the first sample is the same as the scenario given in the problem description.

Problem J
Meeting

You are the boss of company X and have N subordinates. Today, the i-th subordinate will come to the office A_i seconds later than you.

You will have a team meeting today. Due to the capacity of the meeting room, there must be exactly K people (excluding you) attending the team meeting. You can start the meeting S seconds after you come to the office. You can choose the value of S whatever you like, but it must be a positive real non-integer number. Everyone who is already present at the office at that start time will attend the meeting.

You can adjust the arrival time of your subordinates. By paying \$1 (one dollar) and choosing a subordinate, you can change the subordinate's arrival time by one second earlier or one second later. However, a subordinate must not arrive at the office strictly before you—that would be shameful for you. Also, a subordinate must not arrive strictly later than T seconds after you—the subordinate could get fired. You can adjust the arrival time of as many subordinates as you want. You can also adjust the arrival time of the same subordinate more than once.

Determine the minimum amount of dollars needed such that you can have a meeting of exactly K people (excluding you). If it is impossible to do so, output -1.

Input

The first line contains three integers: N K T ($1 \le K \le N \le 100{,}000$; $0 \le T \le 1{,}000{,}000{,}000$) in a line denoting the number of subordinates, the number of subordinates attending the meeting, and the maximum arrival time. The second line contains N integers: A_1 A_2 $\cdots$ A_N ($0 \le A_i \le T$) in a line denoting the arrival time of each subordinate.

Output

The output contains the minimum amount of dollars needed such that you can have a meeting of exactly K people (excluding you), in a line. If it is impossible to do, the output contains -1 instead.

Sample Input	Output for Sample Input
4 2 4 1 2 3 4	0
4 2 4 1 2 2 4	1
2 1 1 0 0	1

2 1 0 0 0	-1

Explanation for the 1st sample case

For the first sample, if you start the meeting 2.5 seconds after you arrived, the meeting will be attended by the first two subordinates.

Explanation for the 2nd sample case

For the second sample, you can adjust the arrival time of the third subordinate to become one second later.

Explanation for the 3rd sample case

For the third sample, you can adjust the arrival time of the second subordinate to become one second later and start the meeting 0.71863781 seconds after you arrived.

Explanation for the 4th sample case

For the fourth sample, you cannot adjust any of the arrival time to satisfy your needs.

Problem K
Permutation

A permutation P of size N is defined as an array $[P_1, P_2, \cdots, P_N]$ where $1 \le P_i \le N$ and $P_i \neq P_j$ for $i \neq j$.

We also define an order of a permutation. If A and B are permutations of size N, then A is less than B if and only if there exists an index i ($1 \le i \le N$) where:

- $A_i < B_i$, and
- $A_j = B_j$ for all $1 \le j < i$

We also define the multiplication of two permutations. If A and B are permutations of size N, then $A \times B$ is a permutation of size N, where the i-th element is A_{B_i}.

We also define the exponentiation of a permutation and a positive integer. If P is permutation and z is a positive integer, then P^z is defined as follow:

- $P^z = P$, for $z = 1$
- $P^z = P^{z-1} \times P$, for $z > 1$

You are given a permutation P of size N. Let M be the smallest integer greater than 1 such that $P = P^M$. We define A (index starts from 1) as an array consisting of P^i for all $1 \le i < M$ sorted in the increasing order (of permutation). In other words, $A_i < A_j$ for all $1 \le i < j < M$.

For example, suppose $P = [2,3,1,5,4]$. Therefore:

- $P^1 = [2,3,1,5,4]$,
- $P^2 = [3,1,2,4,5]$,
- $P^3 = [1,2,3,5,4]$,
- $P^4 = [2,3,1,4,5]$,
- $P^5 = [3,1,2,5,4]$,
- $P^6 = [1,2,3,4,5]$,
- $P^7 = [2,3,1,5,4]$,

Thus, the value of M in this case is 7, and $A = [P^6, P^3, P^4, P^1, P^2, P^5]$.

You are also given Q queries. The i-th query contains an integer K_i. The answer for the i-th query is an integer T_i such that $1 \le T_i < M$ and $P^{T_i} = A_{K_i}$. Can you answer all of the queries?

Input

The first line contains two integers: N Q ($1 \le N \le 100$; $1 \le Q \le 300,000$) in a line denoting the size of the permutation and the number of queries. The second line contains N integers: P_1 P_2 $\cdots$ P_N ($1 \le P_i \le N$) in a line denoting the permutation. It is guaranteed that $P_i \neq P_j$ for all $i \neq j$. The next Q lines, each contains an integer; the integer on the i-th line is K_i ($1 \le K_i < M$, where M is the smallest integer greater than 1 such that $P = P^M$ as explained above. Note that M is not explicitly given in this problem) denoting the query.

Output

Q lines, each contains an integer: T_i in a line denoting the answer of the i-th query.

Sample Input	Output for Sample Input

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

Explanation for the 1st sample case

The permutation given in the first sample is the same as the permutation given in the problem description.

Problem L

Sacred Scarecrows

Nerissa owns a rectangular-shaped paddy field which is divided into $R \times C$ small squares (R rows and C columns) of the same size. Each square is either an empty soil (empty square, which can be used for any purpose) or a rice-planted soil (specifically to plant rice).

To prevent birds like crows or sparrows from disturbing the crops, Nerissa decides to put some scarecrows on the field. Each scarecrow can only be placed on an empty square. Furthermore, to prevent other humans from stealing the crops, the scarecrow arrangement must be **sacred**. An arrangement is said to be sacred if all the following conditions are satisfied:

- Each row contains at least one scarecrow,
- Each consecutive two columns contain at least one scarecrow.

Now Nerissa wonders, how many different sacred arrangements are there? Two arrangement are different if there is a square that contains a scarecrow in one arrangement but not in the other arrangement. Help Nerissa to compute this number.

Input

The first line contains two integers: R C ($1 \le R \le 14$; $1 \le C \le 1000$) in a line denoting the size of paddy field in term of the number of squares (number of rows and columns, respectively). The paddy field is given in the next R lines where each line contains a string of length C. Each square is represented by either '`.`' which denotes an empty soil, or '`v`' which denotes a rice-planted soil.

Output

The output contains an integer representing the number of different sacred arrangement, in a line. As the output can be very large, modulo the output by 1,000,000,007.

Sample Input	Output for Sample Input

`2 2`
`v.`
`..`
	`3`

`1 3`
`...`
	`5`

`2 3`
`vvv`
`...`
	`0`

`3 3`
`...`
`.v.`
`...`
	`145`

2 4 .vv. .v.v	5

Explanation for the 2nd sample case

For the second sample, the following are all 5 sacred arrangements (the scarecrows are denoted by character '*'):

```
***      *.*      **.      .**      .*.
```

Explanation for the 3rd sample case

Notice that, for the third sample, we cannot put any scarecrow in the first row. As a sacred arrangement requires at least one scarecrow in each row, thus, in this case, there is no possible sacred arrangement.

Explanation for the 5th sample case

For the fifth sample, the following are all 5 sacred arrangements (the scarecrows are denoted by character '*'):

```
*vv*     *vv*     *vv.     *vv.     .vv*
*v*v     .v*v     .v*v     *v*v     *v*v
```