Czech ACM Student Chapter
Charles University in Prague
Masaryk University
Slovak University of Technology
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2025

CTU Open 2025 is proud to feature the unbelievably long and complex span of history of Czech and Slovak lands and the nations living here, beginning in neolithic, up to the current turbulent times. Programming opportunities in the past were ubiquitous, the tasks that the prominent figures had often to solve would easily match even today's advanced programming and contest problems. We managed to uncover a host of unique unexpected problems from the past and we present them to you for the first time in the long contests history. With open mind, strong determination and some luck, you can advance to the level of the very best minds operating on our lands in previous centuries and millennia. We wish your results in the Contest to be even more excellent and outstanding.

Your programs can be written in C, C++, Java, Python or Kotlin programming languages. The choice is yours but you will be fully responsible for the correctness and efficiency of your solutions. All we need is the correct answer produced by your code in some appropriate time.

All programs will read text from the standard input only. The results must be written to the standard output. You are not allowed to use any other files, communicate over network, or create processes. Input and output formats are described in problem statements and must be strictly followed. Values given on one line are separated by one space, if not specified otherwise. Do not print anything more than required. Each printed text line (including the last one) should be terminated by a newline character ("`\n`"), which is not considered to be a part of that line.

Good luck, thank you for coming to our contest, and see you again in 2026!

Archery

`archery.(c|cpp|py)`, `Archery.(java|kt)`

Charles IV (known in Czech as Karel IV.), the founder of the first university in the Czech lands, enjoyed a game once popular among archers in the Middle Ages — a game that, unfortunately, gradually disappeared at the dawn of the modern era. It was played during grand outdoor tournaments.

A flat rectangular field was divided into a grid of squares. The referee of the game, at his discretion, marked some of the squares by placing a boar head at their center. Each square then contained either one boar head or none.

Next, the host of the tournament chose one marked square, covering it with purple cloth to mark the starting square, and then another marked square, covering it with golden cloth to mark the final square.

The competing archer always started from the purple square and his goal was to reach the golden square. He could shoot only from the marked squares. When he shot an arrow and hit a boar head in another square, he was allowed to move to that square and continue shooting from there. If he missed, his game ended unsuccessfully. His arrow could fly only straight across a single row or column of squares, that is, parallel to the edges of the field. It did not matter how far the arrow did fly, the archer could reach any square in his current row or column.

It often happened that even the finest archers could not win, unable to reach the golden square, because the placement of the marked squares made it impossible.

Charles IV, when he was present at a tournament, used to introduce an additional preparatory phase, performed just before the purple and golden squares were chosen. In this phase, the referee's assistant sent a small number of men onto the field with the task of relocating some of the boar heads into other, previously unmarked squares. This meant that some marked squares effectively changed position.

Each man had to move exactly one boar head, and, just like an archer's shot, he could move it only within the same row or column, i.e., parallel to the edges of the field. Also, the original principle had to remain valid — no square could ever contain more than one boar head.

Note that the men were sent one after the other, in particular, it is possible that one boar head moved several times.

The purpose of this phase was obvious: Upon its completion, it was guaranteed that victory in the game would always be possible, no matter where the host subsequently placed the purple and golden cloths.

The assistant referee had to decide for each man from which square to which square that man would move the boar head. He did this by experience and intuition, but today we can solve his task precisely, by contemporary algorithmic means.

For a given initial arrangement of boar heads, determine the minimum number of men who must move some of the boar heads so that Charles IV's requirement is fulfilled — that is, it will always be possible to win the game, regardless of how the host chooses the starting and target squares.

Input Specification

The first line contains an integer N ($1 \le N \le 2 \cdot 10^5$), the number of boar heads placed on the field.

Each of the next N lines contains two integers x and y ($-10^9 \le x, y \le 10^9$), representing the coordinates of one boar head in the plane. Assume that each square of the field is one unit wide and that the borders of the field are parallel to the coordinate axes. No two boar heads occupy the same square.

Output Specification

Output a single line with a single integer — the minimum number of men required to rearrange the boar heads so that Charles IV's requirement is fulfilled.

Sample Input 1

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

Output for Sample Input 1

```
2
```

Book Burning

`book.(c|cpp|py)`, `Book.(java|kt)`

Book burning — the destruction of volumes deemed undesirable by the ruling cliques of the time — remained a favorite pastime of certain partially educated scholars in the Czech lands well into the modern era, when differential calculus was already known, Newton's law of gravitation was established, and most of the globe had been mapped.

We have discovered records indicating that, for greater dramatic effect, such burnings were sometimes performed by several burners — one senior and a few juniors. The senior burner arranged the books in a row and marked each of them with a small Latin letter. Each junior burner then received a so-called sacred incantation — a Latin word, or more often just a short sequence of Latin letters written in succession.

Each junior burner then made a single pass along the row of books from its beginning to its end. Whenever he encountered several consecutive books whose markings matched his incantation, he spoke the incantation aloud once, pulled those books from the row, and cast them onto the blazing pyre. The sequence of letters on the books had to match exactly the order of letters in the burner's incantation. After removing such a group of books, the burner continued further along the row, removing and burning every other occurrence of his incantation in the same way.

During his pass, the burner was not allowed to move backward, nor was he allowed to utter the incantation except at the moment of removing books. When his turn ended, the remaining books in the row were closed up so that their order was preserved but no gaps remained.

As books gradually disappeared, it could happen that a burner found no sequence matching his incantation at all — this, however, did not affect the progress of the ritual.

What mattered most was how many times each junior burner spoke his incantation, which depended on the order in which the burners took their turns at the prepared row of books.

Historical records provide us with the sequence of letters marking the books arranged for burning, the incantations of the junior burners, and the order in which they approached the row. To illustrate the atmosphere of those times, we wish to determine how many times each junior burner uttered his incantation.

Input Specification

The first input line contains two integers N, Q ($1 \le N \le 10^5, 1 \le Q \le 4 \cdot 10^5$), the number of books in the row and the number of junior burners.

The second line contains a string s consisting of N lowercase letters, representing the markings on the books, in the order in which they appear in the row of the books.

Next $2Q$ lines contain Q queries. Each query occupies two lines. The first line of a query contains an integer N_i ($1 \le N_i \le 5$), the number of letters in the incantation of the i-th junior burner. The second line of the query contains a string s_i consisting of N_i lowercase letters, which is the incantation of the i-th junior burner.

Output Specification

Output Q lines, each answering the corresponding query — the i-th line specifies how many incantations were uttered by the i-th junior burner.

Sample Input 1

```
6 5
banana
3
ana
3
ban
2
na
1
b
5
apple
```

Output for Sample Input 1

```
1
0
1
1
0
```

Sample Input 2

```
12 5
ccccatatatat
3
cat
3
cat
3
cat
3
cat
3
cat
```

Output for Sample Input 2

```
1
1
1
1
0
```

Constantine of Thessaloniki

`constantine.(c|cpp|py)`, `Constantine.(java|kt)`

Long before the year 863, when Constantine of Thessaloniki — later known as Cyril, the brother of Methodius — began shaping the Old Slavonic script for his missionary work in Central Europe, a rare parchment record was created in Macedonia.

The document reveals that Constantine had already been experimenting not only with letters but also with a system of numerical symbols. Influenced by ancient ideas about infinity, he proposed a novel concept: each positive integer should have its own unique symbol. To generate such a symbol, he described a precise procedure — what we would now call an algorithm.

He started with the number 1, represented simply by the Greek letter Δ. For numbers greater than 1, the process began the same way, by writing Δ, and then continued step by step:

1. If the current number was odd, a new letter Δ was written one position to the right and above the previous letter.
2. If the current number was even, a new letter Δ was written one position to the left and above the previous letter.
3. The number was then divided by two, with any fractional part discarded. If the result became 1, the process ended; otherwise, it was repeated with the new value.

Next, Constantine arranged the resulting pattern into what we would today describe as a rectangular matrix. Each letter Δ occupied one row, and each letter Δ was shifted one column left or right relative to the letter on the previous row. The matrix was made as narrow as possible, and all empty positions were filled with dots. There were no columns filled with dots only.

For example, the symbol for the number 10 occupied a matrix with four rows and two columns. The letter Δ appeared in the first and third rows of the first column, and in the second and fourth rows of the second column.

The parchment even contains traces of Constantine's attempts to perform addition of numbers written in this specific form — an operation we shall now try to reconstruct. In our reconstruction, we will replace Constantine's original letter Δ with the hash symbol #.

Input Specification

The input contains two numbers written in Constantine's notation.

Each number is presented on a few lines as follows: the first line contains an integer N ($1 \leq N \leq 30$), the number of rows in the matrix representing that number. Each of the next N lines represents one row of the matrix. All lines contain the same number of characters, each character is either a dot ('.') or a hash ('#').

The first number is immediately followed by the second number, without any blank lines.

Output Specification

Output the sum of the two input numbers, using a single symbol constructed according to Constantine's rules, in the same format as the input numbers.

Sample Input 1

```
3
#..
.#.
..#
3
.#
#.
.#
```

Output for Sample Input 1

```
4
#.
.#
#.
.#
```

Sample Input 2

```
2
#.
.#
4
..#
.#.
#..
.#.
```

Output for Sample Input 2

```
5
#....
.#...
..#..
...#.
....#
```

Ducks

`ducks.(c|cpp|py)`, `Ducks.(java|kt)`

The founder and ruler of the Samo Empire, the Frankish merchant Samo, who lived in the first half of the seventh century, was known for serving roasted ducks and hares as delicacies to foreign envoys and visiting rulers at his ceremonial feasts.

In his kitchen, a special collection of N long finely forged grillsticks was reserved for those grand occasions. Each grillstick had its own ceremonial name and was used to roast exactly 4 animals, each either a duck or a hare.

No matter how many guests attended, Samo upheld three strict rules:

- all grillsticks must be used, that is, the total number of animals roasted was always $4 \cdot N$,
- in the whole feast, the total number of ducks has to be equal to the total number of hares,
- at least one grillstick must contain only ducks.

Before serving, Samo would lay out all the grillsticks in a fixed, ritual order upon the banquet table and recite an invocation to the powerful spirits.

A sharp-eyed young cook once noticed that the head chef arranged the animals on the grillsticks slightly differently for each feast. No two feasts were ever identical — for any pair of feasts, there was always at least one grillstick that differed, either in the number of ducks and hares or, if those numbers matched, in the order in which they were arranged.

"How many different feasts can be prepared under such a rule," the young cook once asked the senior, "if every feast must have its grillsticks arranged differently?" But the older cook did not know the answer. He only laughed and said, "Ask the magicians." Of course, had he known what the distant future would bring, he might also have said, "Ask the programmers."

Input Specification

Input contains one integer N ($1 \leq N \leq 10^5$), the number of grillsticks in Samo's kitchen.

Output Specification

Print the number of distinct feasts which can be served while maintaining the conditions imposed by Samo. As the number may be huge, output the result modulo $10^9 + 7$.

Sample Input 1	Output for Sample Input 1
2	2

Sample Input 2

```
4
```

Output for Sample Input 2

```
1974
```

Emigrants

`emigrants.(c|cpp|py)`, `Emigrants.(java|kt)`

The number of Slovaks who emigrated to North America around the turn of the 19th and 20th centuries in search of better living conditions is commonly estimated at about half a million. At that time, the Slovak nation numbered a little over two million people, which means that nearly one quarter of the population left their homeland.

The journey to America took several weeks and was often efficiently organized by large steamship companies, for whom the steady stream of emigrants represented a major source of income. Agents operating in Hamburg, who placed large families in emigrant hostels ("Auswandererhallen"), soon noticed an interesting pattern: two families would get along well in the common dining hall if they were seated at a fixed group of tables arranged according to a certain rule.

A pair of families was called a *compatible pair* by the agents if the families could be seated so that

- at each used table was the same number of members of the first family,
- similarly, at each used table was the same number of members of the second family.

The agent responsible for assigning cabins to families on individual ships received a higher commission from the company whenever the group of families boarding a ship contained a sufficient number of compatible pairs of families. The agents informally called such a group of families a smart group.

Each agent had a list of traveling families, recorded in the order in which they had arrived at the hostel. To form a group of families boarding a particular ship, the agent would select an entry from this list and then include several immediately following entries, without skipping any. If the selected sequence of families formed a smart group, the agent became entitled to an increased commission.

When several ships were available at once, the agent often needed to determine quickly, based on the list, which smart groups of families could be formed.

The exact procedure used by the agents is no longer known, but the original lists of traveling families have been preserved. To assess the difficulty of the agents' work, historians wish to determine how many different smart groups could have been selected from a given list of families.

Input Specification

The first input line contains two integers N, K ($1 \leq N \leq 2 \cdot 10^5$, $1 \leq K \leq 10^9$), the number of families on the list and the minimum number of compatible pairs in a smart group. The second input line contains a sequence of N integers $A_1, A_2, \ldots, A_N$ such that $1 \leq A_i \leq 5 \cdot 10^5$, the integers correspond to the number of members of each family on the list, and preserve the order of families.

Output Specification

Print the number of smart groups that can be formed from the families on the given list.

Sample Input 1

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

Output for Sample Input 1

```
5
```

Sample Input 2

```
7 4
2 4 6 8 3 81 12
```

Output for Sample Input 2

```
10
```

Grooves

`grooves.(c|cpp|py), Grooves.(java|kt)`

The age of archaeological finds in Dolní Věstonice in Moravia is estimated at about 28,000 years. We may therefore say that the roots of culture on our territory reach at least that far into the past. Among the unexpected and seldom discussed discoveries at this site are also systems of peculiar, barely visible parallel grooves in the ground. At the ends of some of these grooves, ritual stones were probably placed, and it may be assumed that from such formations later developed the famous Neolithic monuments — such as the stone alignments in Carnac in France or the Kounov Rows in Bohemia, west of Kladno.

In prehistoric times, purposeful human activity was inseparable from ritual and from participation of individuals and groups in an imagined cosmic order. This must always be kept in mind when interpreting ancient remains. The same held true for hunting. Near Dolní Věstonice, great herds of migrating animals — horses, deer, and others — used to pass regularly. Before the hunt, the task of the hunters was to confuse the spirits that moved with the herd and could influence the outcome of the hunt. The hunters therefore altered their movement according to ritual prescriptions.

The grooves always start and end on integer points, and always run parallel to the X-axis, that is, a groove starts and ends on some points (x_p, y_p) and (x_q, y_q) such that $y_p = y_q$. The N hunters are initially positioned on points $(1, 0), (2, 0), \ldots, (N, 0)$. In each turn, they move by a distance of one along the Y-axis in the positive direction. Whenever a hunter's position coincides with an **endpoint** of a groove, he moves to the other endpoint of the groove in one quick jump. This happens without a problem even when there are hunters on both endpoints of the groove. If the hunter enters any other point of the groove, he simply ignores the groove. Each hunter stops when there are no grooves in front of him.

As the hunters changed their relative positions in the described manner, they confused the spirits of the herd and that helped to ensure a successful hunt. It may be assumed that among those hunters there were also individuals who, if they lived today, would be able to determine directly, for each hunter, his final position — and who would be able to write a computer program solving the task.

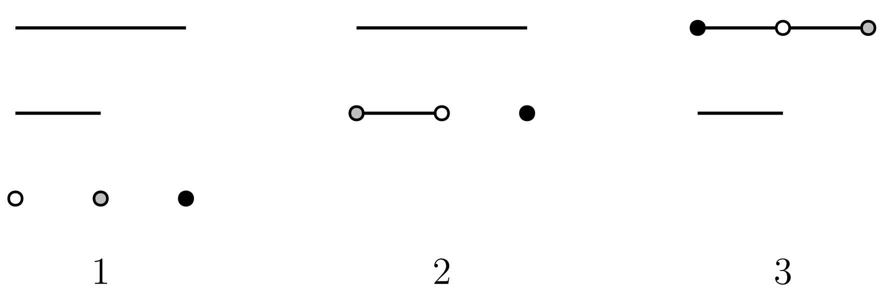

Figure 1: Example of three hunters going across an area with grooves, with hunters represented as colored circles and grooves represented as lines. From left to right: Initial position of the hunters, position after the first turn and the final position.

Input Specification

The first input line contains two integers N, M ($1 \le N \le 2 \cdot 10^5$, $0 \le M \le 2 \cdot 10^5$), the number of hunters and the number of grooves. Next, there are M lines, each contains 4 integers x_p, y_p, x_q, y_q specifying the endpoints of one groove as (x_p, y_p) and (x_q, y_q). It is guaranteed that no two endpoints (even from different grooves) coincide, and that for every groove, its endpoint coordinates satisfy $y_p = y_q$. It is guaranteed that $1 \le y_p, y_q \le 10^9$ and $1 \le x_p, x_q \le N$.

Output Specification

Output exactly N lines. On the i-th line (indexed from 1) of output indicate, for the hunter originally located at $(i, 0)$, at which x coordinate he ended before the start of the hunt.

Sample Input 1

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

Output for Sample Input 1

```
3
1
2
```

Hussites

`hussites.(c|cpp|py)`, `Hussites.(java|kt)`

In some of the battles fought by the Hussites on open plains, they employed — in addition to their famous wagon fortifications — another unexpected tactic that made use of the spare lighter or smaller wagons, which could not be incorporated into the main defensive formation.

For their enemies, they prepared what was called a false escape route. Among those spare wagons, they deliberately left various gaps, suggesting an unbending path, the false escape route, as if leading out of the battlefield along an absolutely straight line. Hussite fighters remained hidden and ready behind these wagons, from where they would charge at demoralized individuals or groups attempting to flee the engagement.

From experience, they knew how crucial the position of each wagon was relative to the false escape route. A wagon was said to be powerful if, within the group of these wagons, there was another wagon positioned so that the escape route was at the same distance from both wagons, and moreover, the straight line connecting the two wagons was perpendicular to that route.

The tactical advantage of such a configuration was clear: attackers could rush simultaneously from opposite sides at any opponent caught on the false route, leaving little chance of effective resistance. It was therefore desirable to arrange the wagons so that as many of them as possible were powerful.

Several sketches showing wagon arrangements have been preserved in regional archives, though it is uncertain whether they depict genuine battle layouts involving a false escape route. What is known — in modern terminology — is that the tactic of the false escape route was employed whenever, in a given arrangement, at least P percent of the wagons in the arrangement were powerful, where P was a specified value for that situation. Sometimes a wagon could even stand directly on the false escape route, but such a wagon was never counted among the powerful ones.

For a given wagon arrangement, determine whether it would have allowed the use of the false escape route tactic. We suppose that the positions of the wagons are fixed and that no wagon can be moved elsewhere.

Input Specification

The first line contains two integers N, P ($1 \leq N \leq 1500$, $0 \leq P \leq 100$), the number of wagons in the configuration and the required percentage of wagons that must be powerful for the false escape route tactic to be applicable.

Each of the next N lines contains two integers x and y ($-10^6 \leq x, y \leq 10^6$), the Cartesian coordinates of a point representing a wagon position. All points are pairwise distinct.

Output Specification

Print `YES` if the configuration of wagons given by the input points allows employing the false escape route tactic; otherwise, print `NO`.

Sample Input 1

```
5 80
0 0
1 0
-1 0
2 0
-2 0
```

Output for Sample Input 1

```
YES
```

Sample Input 2

```
4 75
0 0
1 1
2 2
8 10
```

Output for Sample Input 2

```
NO
```

Sample Input 3

```
6 100
-1 0
3 0
-1 1
3 1
-1 2
3 2
```

Output for Sample Input 3

```
YES
```

Sample Input 4

```
2 100
-1 0
3 0
```

Output for Sample Input 4

```
YES
```

Sample Input 5

```
6 100
1 0
0 1
2 0
0 2
3 0
0 3
```

Output for Sample Input 5

```
YES
```

JJ the Almighty

`jj.(c|cpp|py)`, `Jj.(java|kt)`

Juraj Jánošík (1688–1713) was a legendary Slovak outlaw, said to take from the rich and give to the poor. According to our sources, he did so on a countryside of a grid shape. The grid consists of $(N+1)\times(M+1)$ intersections, with each pair of neighboring intersections connected by a road. Along each road, there is a house, which is either rich or poor. Each region enclosed by 4 roads is called a field.

Now, Jánošík works in a very particular way. Each day, he picks a path along the roads and visits all houses on those roads. If the house is rich, then he of course takes from them, making them poor. On the other hand, if the house is poor, he will give to them, making them rich. Jánošík carries with him enough money, to make any number of houses rich. Similarly, Jánošík has big enough pockets to steal from any number of houses.

Furthermore, the paths chosen by Jánošík always have a particular property. Jánošík

- either drives in a straight line, going from one border of the country to the opposite one, visiting each house exactly once along this straight line,
- or he picks a connected area of fields, and walks along the roads on the perimeter of this area, visiting each house exactly once.

Recently, the historians found records, showing which houses were rich and which were poor in year 1687, and another record from year 1714, both from the same country. They want you to find out whether the redistribution of wealth can be only due to Jánošík. Jánošík might have performed his redistributions of wealth **any number of times**.

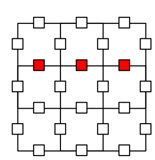

Figure 1: Example of Jánošík riding along a straight line. Visited houses are shaded.

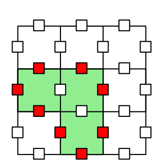

Figure 2: Example of Jánošík walking along a perimeter of a connected area of fields. The area of fields is lightly shaded, the visited houses are shaded.

Input Specification

The first line contains two numbers N, M ($1 \leq N \cdot M \leq 100$), the number of fields in each column and each row, respectively. Next follow $2N + 1$ lines describing the initial states of houses in the corresponding rows of the grid. Each line consists of a sequence of 0s and 1s, 0 indicating a poor house and 1 indicating a rich house. The first line contains M numbers, the second line contains $M + 1$ numbers, the third line contains M numbers, the fourth line contains $M + 1$ numbers, and so on.

The next $2N + 1$ lines describe the final states of the houses in the same format as before.

Output Specification

Output a single line with `Yes` or `No`, indicating whether the final state of houses can be a result of Janošík's operations on the initial state.

Sample Input 1

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

Output for Sample Input 1

```
Yes
```

Sample Input 2

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

Output for Sample Input 2

```
Yes
```

Sample Input 3

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

Output for Sample Input 3

```
No
```

Meetings

`meetings.(c|cpp|py)`, `Meetings.(java|kt)`

The political situation at the founding of the Czechoslovak state in 1918 was, in principle, favorable — but extremely chaotic. Most of the high-ranking officials were burdened with additional duties abroad, and could be present in the government only for limited periods of time. Fortunately, the presidential office managed to compile a schedule in which each official was assigned a unique time interval during which he would be surely present in Prague.

The Czechoslovak Minister of War, Milan Rastislav Štefánik, who was also an astronomer, stood out among the cabinet members thanks to his scientific background. It enabled him to seek more systematic ways of organizing the government's schedule so that the cabinet could personally meet with as many officials as possible.

The schedule spanned a long sequence of days, indexed $1, 2, 3, 4, \ldots, 100000$.

For each official, the first and the last day of his presence in Prague were recorded. The government was allotted a fixed number of meeting days. These days could be chosen freely, not necessarily in consecutive order, and meetings with officials were to be held during these days. Štefánik began by randomly selecting and marking the prescribed number of meeting days in the schedule. After that he determined the number of unavailable officials, that is, the number of those officials who would miss all the meeting days.

Next, he would select one of the chosen meeting days and shift it to another date, making sure the new date did not overlap with any of the other chosen days. After each adjustment, he again recorded the number of unavailable officials. He repeated this procedure several times. During the repeated procedure the date of any particular meeting day could be shifted more times, or not shifted at all.

To calculate the numbers, Štefánik relied on a rather imperfect mechanical calculator of the period that he had brought from his observations on Mont Blanc in the Alps.

Today, we can repeat the Minister's procedure, this time using an efficient computer program.

Input Specification

The first input line contains three numbers N, C, Q ($1 \leq N \leq 10^5, 1 \leq C \leq 10^5, 1 \leq Q \leq 10^5$), the number of officials, the number of prescribed meeting days, and the number of queries. Each query represents a shift of the date of one planned meeting day to another date.

Next, there are N lines, each represents the interval $[x, y]$ in which an official was in Prague. Here, x and y are the indices in the schedule of the first and of the last day of the interval. It is guaranteed that $x \leq y$ and $1 \leq x, y \leq 10^5$.

Next, there is a line containing the C distinct integers C_i with $1 \leq C_i \leq 10^5$, describing the meeting days initially selected by the Minister.

Next, there are Q lines, each describing one query consisting of two integers, f and t with $1 \leq f, t \leq 10^5$. Here, f and t represent a move of a meeting day from index f to index t in the schedule. It is guaranteed, that at the time of query, there is a meeting day on f, and there is not a meeting day on t.

Output Specification

First, output a single line with the number of unavailable officials immediately after the initial choice of the meeting days. Next, for each query output a single line with the number of unavailable officials after the schedule change corresponding to the query. In total, the output must contain $Q + 1$ lines.

Sample Input 1

```
2 1 3
1 4
2 7
3
3 5
5 1
1 10
```

Output for Sample Input 1

```
0
1
1
2
```

Ornithology

`ornithology.(c|cpp|py)`, `Ornithology.(java|kt)`

Vojtěch Jarník (1897–1970) is regarded as one of the most influential Czech mathematicians. He made great contributions to the topics of mathematical analysis and number theory, but he is perhaps best known for his algorithm for finding a minimum spanning tree. This was in response to the publication of Borůvka's algorithm by another great Czech mathematician, Otakar Borůvka.

Much less known is their shared interest in *ornithology*, particularly in higher dimensions. High-dimensional crows are known for their tendency to align themselves along a straight line in one dimension. Moreover, they like to maintain their respective distances from each other.

Now, given the positions of several crows, Vojtěch and Otakar were curious what the minimum total number of steps would be for the crows to form a single line in which all their original pairwise Manhattan distances would be preserved. In a single step, a crow can move by a distance of 1 along a single coordinate axis.

Formally, a crow is represented by a vector of D integer coordinates. In a single step, one crow is chosen and one of her coordinates is increased or decreased by 1. Any number of crows can share the same space.

We ask for the minimum number of steps required so that the crows reach a position satisfying the following conditions:

- For all crows, their coordinate vectors are equal in all but one coordinate.
- For each pair of crows, their Manhattan distance is the same as it was before the start of the process. The Manhattan distance between crows $a = (a_1, \ldots, a_D)$ and $b = (b_1, \ldots, b_D)$ is $\sum_{i=1}^{D} |a_i - b_i|$.

Input Specification

The first line contains two integers N and D ($1 \leq N \cdot D \leq 10^5$), the number of crows and the number of dimensions. Each of the next N lines contains D integers, each between -10^9 and 10^9 inclusive, representing the coordinates of one crow.

Output Specification

Output a single integer — the minimum number of steps required for the crows to reach a configuration satisfying the required properties. If it is not possible, output -1.

Sample Input 1

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

Output for Sample Input 1

```
12
```

Sample Input 2

```
3 3
10 5 6
20 3 4
30 1 2
```

Output for Sample Input 2

```
16
```

Snow

`snow.(c|cpp|py)`, `Snow.(java|kt)`

Since 1906, a cross-country ski race has been held in the lands of the Czech Crown, in the Krkonoše Mountains. In the eighth year of this championship, the initial conditions seemed favorable, and the racers set off wearing only light clothing. Unfortunately, the weather soon turned for the worse, and most of the competitors were forced to abandon the race. Only Jaroslav Hanč continued, but due to the increasingly dangerous conditions, his friend Václav Vrbata went out to find him. Tragically, neither of them managed to reach safety in time, and both lost their lives.

To prevent such dangers in the ninth year of the championship, a new system for weather estimation was introduced. As a result, we began measuring atmospheric conditions more precisely to better predict race conditions.

The currently observed snow in the atmosphere is a rectangular grid, with each tile containing snow or empty space, represented as `'*'` and `'.'` respectively. The snow tends to naturally fall downwards, unless there is ground or there is a pile of snow touching ground below it. A pile of snow consists of the snow flakes in one column that can no longer fall down. More precisely, if the snow flake is on the very last row or there are only other snow flakes (no empty space) below it, it stays put. Otherwise, it moves one position down.

At each important time of the race, we need to determine the total number of snow flakes in the piles that are touching the ground, and thus preventing unexpected difficulties.

Input Specification

The first line contains three integers N, M, and Q ($1 \leq N \cdot M, Q \leq 10^5$), the number of rows and columns of the grid and the number of important time points.

The next N lines each contain M characters, forming the aforementioned grid. Each character is either `'.'` (empty space) or `'*'` (snow flake).

Finally, Q lines follow, each containing one integer T_i ($0 \leq T_i \leq 10^5$), the time for which we need to know how many snow flakes are piled on the ground.

Output Specification

For each query integer T_i, print on a separate line — in the order of the queries — the total number of snow flakes in piles that have fallen to the ground at time T_i.

Sample Input 1

```
5 4 3
****
.*..
*..*
...*
*.*.
0
2
5
```

Output for Sample Input 1

```
2
7
10
```

Sample Input 2

```
4 5 5
*****
.*...
..*..
...*.
0
1
2
3
10
```

Output for Sample Input 2

```
1
2
6
8
8
```

Theatre

`theatre.(c|cpp|py), Theatre.(java|kt)`

The National Theater in Prague began its activities in June 1881, and soon afterward — in August of the same year — it burned down before its completion. Along with the building, unique decorations of great artistic value were also destroyed.

Artists therefore began preparing new designs for the future paintings and frescoes in the restored building. During that time, a transitional period began, known as the era of numbered painting, which later became the forerunner of much more famous styles like Cubism and Cubofuturism and also Neoclassical Pissoffism.

The numbered painting worked as follows: the artist prepared a large sketch of the intended work, mainly emphasizing the contours of the figures and the surrounding scenery. The assistants then produced a number of copies of the sketch, all of the same size. In each copy, the individual regions, bounded by the contours, were marked with different natural numbers. In all copies, the same regions were labeled with the same numbers.

Thus, the artist obtained many identical sketches with numbered regions. To estimate the overall visual effect of the finished painting, the artist instructed the assistants to fill in each sketch with colors. Each bounded region was painted with a single color and no additional details were added to keep the process as simple as possible.

Different sketches were colored in different ways, and no two sketches were ever painted exactly in the same manner, meaning that they differed in the color of at least one region. It has to be noted that the assistants had only one palette of available colors.

The colored sketches were then temporarily placed in various spots whose lighting and atmosphere resembled the intended final location of the painting. The artist would choose the sketch whose coloring pleased him the most and used it as a model for the completed work.

The numbering of the regions in the sketches had a specific purpose. The artists were concerned with what they called contrasting regions. In order to achieve a strong contrast effect, certain regions that were close to each other were not allowed to be painted with the same color.

To enforce this, the artist gave the assistants a list of forbidden pairs of regions, represented as pairs of numbers corresponding to the numbering in the sketches. The regions in a forbidden pair had to be painted with different colors.

The artists working on the decoration of the restored National Theater were true radicals, and they always demanded that their assistants prepare sketches covering all possible combinations of colors that obeyed the given restrictions. It was quite a large number — one that is difficult to estimate today, since most of the sketches were destroyed after the final works were completed. Only a few have survived, to the great disappointment of modern art historians.

Recently, several complete lists of forbidden pairs have been discovered, relating to some of the famous paintings inside the theater. It is now believed that, from each such list of forbidden pairs, it is possible to deduce the number of sketches that must have been used in the creation of the corresponding painting. Try to determine this number for different sizes of color palettes.

Input Specification

The first input line contains three integers N, M, T ($1 \le N \le 21, 0 \le M \le 21, 0 < T \le 10^5$), the number of regions in the painting, the number of forbidden pairs of the regions, and the number of color palettes considered. The second input line consists of integers A_i ($1 \le i \le T, 1 \le A_i < 10^6$), each representing the number of colors in the color palette used in creating the sketches. We suppose that the regions are marked by numbers $0, 1, \ldots, N-1$. Then follow M lines, each contains a pair of integers u, v ($0 \le u, v < N$), representing one forbidden pair of regions. It is guaranteed that $u \ne v$, and that all pairs of u, v are unique.

Output Specification

Print T numbers, where i-th number is the total number of sketches which would be created if the color palette contained exactly A_i different colors. Output the result modulo 1000000007.

Sample Input 1

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

Output for Sample Input 1

```
0
0
12
```

Sample Input 2

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

Output for Sample Input 2

```
0
2
18
84
```