Problem A. Villages: Landlines

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The game development team *NIO Tech* releases its debut game, *Villages: Landlines*. In this game, players develop their villages and enjoy their peaceful country life.

The game *Villages: Landlines* contains a one-dimensional power system. Initially, there are several power stations and buildings. The goal of the game is to make every power stations and buildings connected together in the power system, forming a connected graph. Players can build some power towers and wires in the game, which are the tools to connect power stations and buildings.

Power stations or buildings can connect to power towers without wires, while power towers must connect to each other by wires. What's more, power stations and buildings can't connect to each other directly, they must connect through power towers.

Assuming that the coordinate of a building is x_i and its receiving radius is r_i, all the power towers whose distance from the building is no greater than r_i are directly connected to it without wires. That is to say, for the power tower located at x, it is directly connected to the building at x_i without wires if and only if $|x - x_i| \le r_i$. Similarly, assuming that the power supply radius of a power station is r_s, all the power towers whose distance from the power station is no more than r_s are directly connected to it without wires. Supposing that the coordinates of two power towers are x_A and x_B, players can connect them with the wire, and the length of wire required is $|x_A - x_B|$. A power tower can be connected to any number (possibly zero) of power towers, buildings and power stations.

In order to make the game more friendly to the rookies, Nostalgia, a member of *NIO Tech*, decides to develop the power distribution recommendation function. In the case of using any number of power towers, players can choose **exactly one** power station and several buildings to get an optimal power supply scheme, which uses the shortest total length of wires to complete the power system. However, Nostalgia is not sure whether her scheme is correct, so she needs your help to calculate the total length of wires used in the optimal scheme to determine the correctness of her scheme.

Note that the players can build a new power tower at coordinate x even if there exists a power station or a building at x. There might be more than one power station or building at the same coordinate.

Input

The first line contains a single integer n ($1 \le n \le 2 \times 10^5$).

The second line contains two integers x_s, r_s ($-10^9 \le x_s \le 10^9, 1 \le r_s \le 10^9$) – the coordinate of the power station and its power supply radius.

The i-th line of the next $n - 1$ lines contains two integers x_i, r_i ($-10^9 \le x_i \le 10^9, 1 \le r_i \le 10^9$) – the coordinate of i-th building and its receiving radius.

Output

Print an integer in a single line, denoting the total length of wires used in the optimal scheme.

Examples

standard input	standard output
5 0 1 0 3 5 1 6 1 9 2	1
2 -1000000000 1000000000 1000000000 1000000000	0

Note

For the first sample, one possible optimal scheme is building power towers at $1, 3, 4, 6, 7$, and connect the power towers at 3 and 4 with the wire of length 1.

Problem B. Spirit Circle Observation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Potassium, a genius who masters soul conversion, is observing the world line of *Potas*. The world line can be divided into n parts and represented as an n-digit decimal number s (may have leading zeros) that reveals the property of time in the past and future of *Potas*.

For convenience, let $\overline{abc}$ denote the concatenation of digits a, b, c, so the world line can be expressed as $s = \overline{s_1 s_2 \cdots s_n}$. When observing, Potassium focuses on the era, which consists of continuous parts in the world line. An era beginning from part l to r can be denoted as $\overline{s_l s_{l+1} \cdots s_r}$ $(1 \le l \le r \le n)$, and may also have leading zeros.

When two eras are quite similar, Potassium can observe one from another by Spirit Circle. More specifically, an era $B = \overline{s_{l_2} s_{l_2+1} \cdots s_{r_2}}$ can be observed from another era $A = \overline{s_{l_1} s_{l_1+1} \cdots s_{r_1}}$ by Spirit Circle if and only if the following two conditions are satisfied, regardless of where A and B locate in s originally.

1. A and B have the same length, i.e. $r_1 - l_1 + 1 = r_2 - l_2 + 1$;
2. B is greater than A by exactly 1, i.e. $\overline{s_{l_1} s_{l_1+1} \cdots s_{r_1}} + 1 = \overline{s_{l_2} s_{l_2+1} \cdots s_{r_2}}$.

Potassium is wondering: how many different pairs (A, B) can be found in s that he can observe era B from era A. Note that two eras having the same representations but differing in the original location in s are considered as different eras.

Input

The first line contains one integer n $(1 \le n \le 4 \times 10^5)$, denoting the number of parts in the world line of *Potas*.

The second line contains an n-digit decimal number s, denoting the representation of the world line of *Potas*.

Output

Print an integer in a single line, denoting the number of different pairs (A, B) in s that Potassium can observe era B from era A by Spirit Circle.

Examples

standard input	standard output
5 10998	5
6 001100	11

Note

In the first example, Potassium can choose 5 different (A, B) pairs:

1. Choose $\overline{s_2} = 0,\ \overline{s_1} = 1$;
2. Choose $\overline{s_5} = 8,\ \overline{s_3} = 9$;

3. Choose $\overline{s_5} = 8,\ \overline{s_4} = 9$;
4. Choose $\overline{s_2 s_3} = 09,\ \overline{s_1 s_2} = 10$;
5. Choose $\overline{s_4 s_5} = 98,\ \overline{s_3 s_4} = 99$.

Problem C. Grab the Seat!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

Oshwiciqwq studies Physics in a shabby classroom in his sophomore year. The classroom has n rows and m columns of seats, which are narrowly set like a matrix. There is a huge screen in the front of the classroom.

The whole classroom can be seen as a plane, and every seat can be seen as a point on it. The set of all seats' coordinates is $\{(x, y) \mid 1 \le x \le n, 1 \le y \le m, x \in \mathbb{Z}^+, y \in \mathbb{Z}^+\}$. Additionally, the screen can be seen as a segment, with two endpoints at $(0, 1)$ and $(0, m)$.

As a student who loves studying, he always wants to seat in the front row, so that he can see the whole screen clearly. He thinks a seat is **good** if and only if there are no other people blocking him from seeing the whole screen. Formally, the seat at (a, b) is **good** if and only if there is no other person sitting on the seat that is on the segment between (a, b) and any point on the segment $(0, 1) - (0, m)$ (including endpoints).

However, today he arrived at the classroom as early as usual, only to find k people had already taken their seats. What's worse, while he was busy searching for a good seat, some people changed their seats one by one. Totally there are q events of changing seats. He is extremely anxious now, so could you please help him count how many seats are **good** after each of the q changing events?

Obviously, a seat can only be occupied by **at most one** person at the same time.

Input

The first line contains four integers n, m, k, q ($2 \le n, m \le 2 \times 10^5, 1 \le k \le 2 \times 10^5, 1 \le q \le 200$), denoting the number of rows, the number of columns, the number of people already taken seats and the number of changing events.

For the i-th line in the following k lines, there are two integers x_i, y_i ($1 \le x_i \le n, 1 \le y_i \le m$), denoting the coordinates of an occupied seat.

For the i-th line in the following q lines, there are three integers p_i, x_i, y_i ($1 \le p_i \le k, 1 \le x_i \le n, 1 \le y_i \le m$), denoting the id of person changing seats and the coordinates of a seat that the person will change to.

It is guaranteed that any coordinate appears only once in the input.

Output

The output contains q lines.

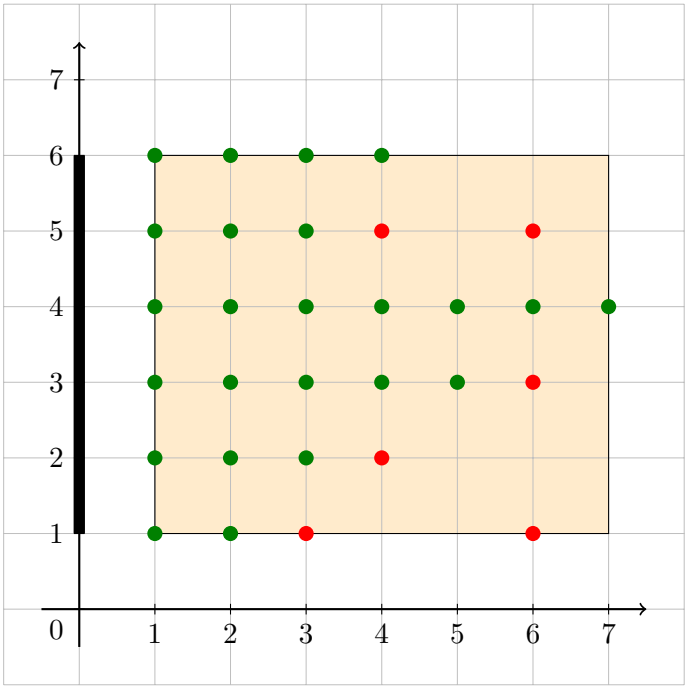
For each of the q changing events, print an integer in a single line, denoting the number of **good** seats at that time.

Examples

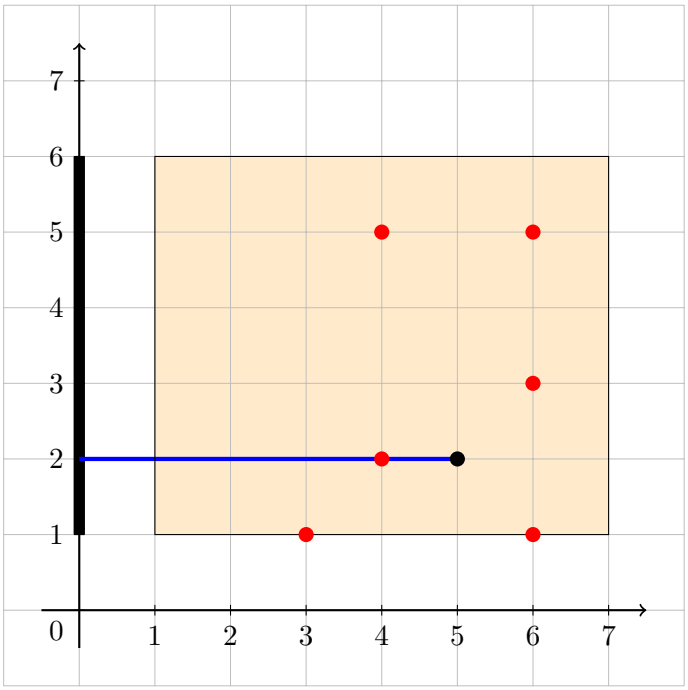
standard input	standard output
7 6 6 1 3 1 4 2 4 5 6 3 6 5 6 2 6 6 1	24
10 10 5 5 4 10 5 9 4 2 6 5 4 7 3 3 5 2 4 1 3 3 6 2 10 6 5 7 4	35 34 36 39 39

Note

The first sample is shown in the graph. The segment on the y-axis denotes the screen. The rectangular area denotes all the seats. After the changing event, the red points are the occupied seats while the green points are **good** seats.



This is an example of a non-good seat. For the seat at $(5, 2)$, the segment between it and $(0, 2)$ on the screen contains a occupied seat at $(4, 2)$, so it is not a **good** seat.


0
1
2
3
4
5
6
7
1
2
3
4
5
6
7

Problem D. Mocha and Railgun

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There is a candy store near Mocha's school. It's said that the storekeeper, Dagashiya, can cast the railgun spell. To be the most powerful Mahou Shoujo, Mocha asks Dagashiya to teach her the railgun spell.

To test her mastery of this spell, Mocha creates a circular wall C whose center is $(0, 0)$. The railgun spell has a **base segment** AB with length $2d$. When it is cast, a point P on the circular wall will be destroyed if and only if it meets the following conditions:

- The projection of point P on line AB lies on the segment AB (inclusive).
- A, B, P make a counterclockwise turn, that is to say $\overrightarrow{AB} \times \overrightarrow{AP} > 0$.

Mocha chooses a point Q which strictly lies in the circle and sets Q as the center of the base segment. Then Mocha will choose an angle α arbitrarily and rotate the base segment around Q by α. Because Mocha is a rookie of the spell, the endpoints of the base segment will always lie in C strictly, which means the distance from Q to the wall C is greater than d.

Mocha wants to know the maximum length of the wall that can be destroyed in one cast.

You can see the note for better understanding.

Input

The first line is an integer T ($1 \le T \le 10^5$) – the number of test cases.

In each test case, the first line is an integer r ($1 \le r \le 10^9$) – the radius of the circular wall C.

The next line contains three integers x_Q, y_Q, d – the coordinates of the point Q and a half of the length of the base segment.

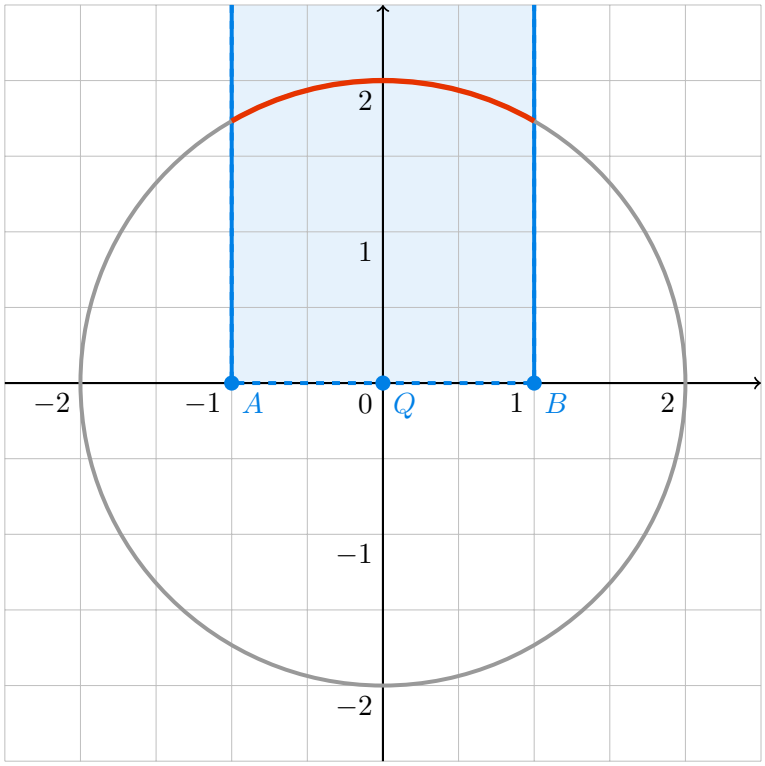
Output

For each test case, print a real number in a single line, denoting the maximum length of the wall that Mocha can destroy in one cast. Your answer will be considered equal to the correct answer when their absolute or relative error doesn't exceed 10^{-6}.

Example

standard input	standard output
1 2 0 0 1	2.094395102393

Note



In the picture above, the blue lines are the boundaries of the railgun spell. The red arc is the wall being destroyed.

Problem E. LTCS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Tcs has just learned the subsequence of a sequence. Now Tcs wants to define the subsequence of a rooted tree. A rooted tree T_a which satisfies the following constraints is called a subsequence of another rooted tree T_b:

- Suppose that there are N vertices in T_a numbered as $1, 2, \ldots, N$ and there are M vertices in T_b numbered as $1, 2, \ldots, M$. There is a function $F : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, M\}$, which satisfies $\forall x, y \in \{1, 2, \ldots, N\}, x \neq y \Rightarrow F(x) \neq F(y)$.
- For all vertices u in T_a, $c_{a,u} = c_{b,F(u)}$ holds, where $c_{i,u}$ is the value of vertex u in T_i.
- If vertex u is an ancestor of vertex v in T_a, then vertex $F(u)$ is an ancestor of vertex $F(v)$ in T_b.
- For distinct vertices u, v, w in T_a, if vertex u is the father of both vertex v and vertex w, then vertex $F(u)$ is on the simple path between vertex $F(v)$ and vertex $F(w)$ in T_b.

A rooted tree T_3 is called a **TCS** (Tree Common Subsequence) of T_1 and T_2 when T_3 is a subsequence of both rooted trees T_1 and T_2. Given the rooted trees T_1 and T_2 whose roots are both vertex 1, Tcs wants to find the largest size of the TCS of T_1 and T_2.

Input

The first line contains two integers n, m ($2 \leq n, m \leq 500$) – the number of vertices in T_1 and T_2, respectively.

The second line contains n integers $c_{1,1}, c_{1,2}, \ldots, c_{1,n}$ ($1 \leq c_{1,i} \leq 500$) – the value of each vertex in T_1.

The third line contains m integers $c_{2,1}, c_{2,2}, \ldots, c_{2,m}$ ($1 \leq c_{2,i} \leq 500$) – the value of each vertex in T_2.

Each of the next $n - 1$ lines contains two integers u and v ($1 \leq u, v \leq n$) – an edge connecting vertices u and v in T_1.

Each of the next $m - 1$ lines contains two integers u and v ($1 \leq u, v \leq m$) – an edge connecting vertices u and v in T_2.

Output

Print an integer in a single line, denoting the largest size of the TCS of T_1 and T_2.

Examples

standard input	standard output
3 3 1 1 1 1 1 1 1 2 1 3 1 2 2 3	2
3 3 1 2 3 2 2 3 1 2 1 3 1 2 2 3	1

Problem F. Cut

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Greninja is helping Zygarde cut trees.

There are n trees in the forest of Kalos. The height of the i-th tree is a_i. Since no two trees are exactly same, $a_1, a_2, \ldots, a_n$ is a permutation of $1, 2, \ldots, n$. Zygarde wants Greninja to do m operations of three following types:

1. Sort $a_l, a_{l+1}, \ldots, a_r$ in increasing order.
2. Sort $a_l, a_{l+1}, \ldots, a_r$ in decreasing order.
3. Find the maximum positive integer k that exists k integers $l \leq b_1 < b_2 < \cdots b_k \leq r$ satisfy $a_{b_i} \not\equiv a_{b_{i+1}} \pmod 2$ for $1 \leq i < k$.

Without Battle Bond, Greninja cannot do these operations by itself. Can you help him?

Input

The first line contains two integers n, m ($1 \leq n, m \leq 10^5$), denoting the number of trees and the number of operations.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$), denoting the initial height of trees. It's guaranteed that $a_1, a_2, \ldots, a_n$ is a permutation of $1, 2, \ldots, n$.

For the i-th line in the following m lines, there are three integers o_i, l_i, r_i ($1 \leq o_i \leq 3, 1 \leq l_i \leq r_i \leq n$), denoting the type of the i-th operation and the range of the i-th operation.

Output

For each operation in type 3, print an integer in a single line, denoting the answer of this operation.

Examples

standard input	standard output
3 3 1 3 2 3 1 3 1 2 3 3 1 3	2 3
10 2 1 4 3 5 6 2 9 8 10 7 2 1 5 3 1 6	5
10 6 1 4 3 5 6 2 9 8 10 7 2 1 5 3 1 6 1 2 6 3 1 7 2 1 10 3 1 10	5 6 10
10 4 1 4 3 5 6 2 9 8 10 7 2 1 5 3 1 6 1 3 4 3 1 6	5 5
10 4 6 4 3 2 10 7 8 1 5 9 1 6 8 1 8 8 3 5 9 3 6 7	4 1

Note

In the first example, the initial height of trees is $[1, 3, 2]$. In the first operation, the maximum k is 2, since $a_1 \not\equiv a_3 \pmod 2$ and $a_1 \equiv a_2 \pmod 2$ (so k cannot be 3). After the second operation, the height of trees becomes $[1, 2, 3]$. In the third operation, the maximum k is 3, since $a_1 \not\equiv a_2 \pmod 2$ and $a_2 \not\equiv a_3 \pmod 2$.

Problem G. Lexicographical Maximum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Eibwen is a newbie in Python.

You might know that when you input a number in the command line, your Python program will receive a string containing that number instead of a number that can be used to calculate. This is an interesting feature that a newbie might not know.

Eibwen wants to find the maximum of some given numbers, so he writes a program to sort the list of the numbers and print the last element in the sorted list. However, as a newbie, Eibwen doesn't know the feature. He actually sorts the list of number strings in lexicographical order and prints the last string in the list.

Now Eibwen runs his program, inputs all the integers from 1 to n, and finds his program really slow. Could you help him find out the expected output of his program?

Input

The only line contains an integer n ($1 \leq n \leq 10^{1000000}$) – the size of Eibwen's input.

Output

Print the expected output of Eibwen's program in a single line, which actually is the lexicographically maximum from 1 to n.

Example

standard input	standard output
616	99

Problem H. Fly

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Groudon wanna fly. Kyogre will tell Groudon how to fly if and only if Groudon solves the following problem.

There are n integers $a_1, a_2, \ldots, a_n$, one integer m and k integer pairs $(b_1, c_1), (b_2, c_2), \ldots, (b_k, c_k)$. You need to find the number of non-negative n-tuples $(x_1, x_2, \ldots, x_n)$ which satisfies:

$$\begin{cases} \sum_{i=1}^{n} a_i x_i \le m \\ x_{b_i} \,\&\, 2^{c_i} = 0 \text{ for } i = 1, 2, \ldots, k \end{cases}$$

Here, & denotes the bitwise operation AND.

Two non-negative n-tuples $(x_1, x_2, \ldots, x_n)$ and $(x'_1, x'_2, \ldots, x'_n)$ are considered different if and only if there exists one integer $i \in [1, n]$ which satisfies $x_i \neq x'_i$.

Since the answer is too large, Kyogre only wants to know it modulo 998 244 353.

You, as a programmer of Hoenn, please help Groudon!

Input

The first line contains three integers n, m, k ($1 \le n \le 4 \times 10^4, 0 \le m \le 10^{18}, 1 \le k \le 5 \times 10^3$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 4 \times 10^4, \sum_{i=1}^{n} a_i \le 4 \times 10^4$).

For the i-th line in the following k lines, there are two integers b_i, c_i ($1 \le b_i \le n, 0 \le c_i < 60$).

All numbers in the input denote the parameter of Kyogre's problem.

Output

Print a single integer, denoting the answer to Kyogre's problem, modulo 998 244 353.

Examples

standard input	standard output
3 5 1 2 3 3 1 0	4
6 10 2 1 1 4 5 1 4 3 1 4 2	539

Note

In the first example, there are 4 non-negative 3-tuples satisfy the condition in Kyogre's problem:

- $(0, 0, 0)$
- $(2, 0, 0)$
- $(0, 1, 0)$

- $(0, 0, 1)$

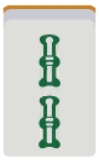
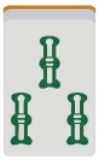
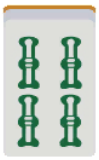
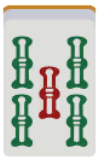
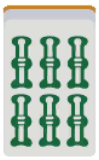
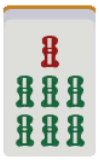
Problem I. Chiitoitsu

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Serval is learning Japanese Mahjong recently in order to play Mahjong Soul with his friends, HomuraCat, RanshiinSaikou, and Mocha.

Serval learned that in Japanese Mahjong, there are 34 different **types** of **tiles**, and there are 4 tiles of each type. According to the suits and the tile types, the tiles can be divided into four groups. Three of them form the numbered suits, each with sequences from one to nine. They are the **manzu**, the **pinzu**, and the **souzu**. The other group is the **jihai**, which can be further divided into the kazehai and the sangenpai.

Serval uses two characters to describe a tile. The former is a number and the latter is one of '`m`', '`p`', '`s`' and '`z`'. Specifically, the manzu, the pinzu and the souzu with the number k are described as '`km`', '`kp`' and '`ks`', respectively. For example, *San man* (the manzu tile with the number 3) is described as '`3m`', *Rou pin* (the pinzu tile with the number 6) is discribed as '`6p`', *Kyuu sou* (the souzu tile with the number 9) is discribed as '`9s`'. As for jihai, *Ton*, *Nan*, *Shaa*, *Pei*, *Haku*, *Hatsu*, *Chun* are described as '`1z`' to '`7z`', respectively. All types of tiles are listed below.

Manzu									
	Ii man	Ryan man	San man	Suu man	Uu man	Rou man	Chii man	Paa man	Kyuu man
	`1m`	`2m`	`3m`	`4m`	`5m`	`6m`	`7m`	`8m`	`9m`
Pinzu									
	Ii pin	Ryan pin	San pin	Suu pin	Uu pin	Rou pin	Chii pin	Paa pin	Kyuu pin
	`1p`	`2p`	`3p`	`4p`	`5p`	`6p`	`7p`	`8p`	`9p`
Souzu									
	Ii sou	Ryan sou	San sou	Suu sou	Uu sou	Rou sou	Chii sou	Paa sou	Kyuu sou
	`1s`	`2s`	`3s`	`4s`	`5s`	`6s`	`7s`	`8s`	`9s`
Jihai	Kazehai					Sangenpai			
		Ton	Nan	Shaa	Pei		Haku	Hatsu	Chun
		`1z`	`2z`	`3z`	`4z`		`5z`	`6z`	`7z`

You may know that in Japanese Mahjong, winning a hand requires at least one **yaku**, where **hand** denotes the tiles a player holds. There are a large number of yaku types, and among them, Serval loves **chiitoitsu** most. Chiitoitsu is the hand that consists of seven distinct pairs. In other words, chiitoitsu consists of seven distinct types of tiles and two tiles of each type. In this problem, there is no need for

you to learn what yaku exactly is. You only need to focus on chiitoitsu.

Serval loves chiitoitsu so much that he wants to practice winning hands with chiitoitsu. Note that the rules that Serval practices seeking chiitoitsu are **different** from those of Japanese Mahjong. At first, all the tiles are shuffled and placed face-down on the board as the stockpile. Then the **only** player, Serval, is dealt 13 tiles, which means that he will get 13 tiles from the stockpile. These 13 tiles are called the **start hand**. In each turn, Serval can draw and discard in the following order:

- Draw a tile from the stockpile.

  Serval gets a random tile from the stockpile. After that, he will hold 14 tiles.

- Tsumo if the hand reaches chiitoitsu.

  Serval declares a win immediately if his hand reaches chiitoitsu, and the game is over at the same time.

- Discard a tile from the hand.

  Serval chooses a tile and removes it from his hand. After that, he will hold 13 tiles. The discarded tile will be placed face-up on the board, which means it will not return to the stockpile.

Now Serval gets his start hand, and he wonders the expected number of turns to reach chiitoitsu if he carries out the optimal strategy. Surprisingly, Serval finds that there are **no more than** two tiles of the same type in his start hand. Serval simply concatenates the notation of each tile in the start hand, forming a string of 26 characters to describe the start hand. However, Serval cannot find out the answer, so he tells you his start hand and asks you to help him. You only need to tell him the answer modulo $10^9 + 7$.

It can be shown that the answer can be represented as a fraction p/q, where p and q are both positive integers and coprime. If there exists a non-negative integer r less than $10^9 + 7$ satisfying $q \cdot r \bmod (10^9 + 7) = p$, we call the integer r the result of p/q modulo $10^9 + 7$.

Input

Each test consists of multiple test cases.

The first line contains a single integer T ($1 \le T \le 10^5$) – the number of test cases.

Each line of the following T lines describes a test case and contains a string of 26 characters – the start hand.

It is guaranteed that there are no more than two tiles of the same type in the start hand.

Output

For the i-th test case, print a line containing "`Case #i: x`" (without quotes), where i is the number of the test case starting from 1 and x is the answer to the test case.

Example

standard input	standard output
2 1m9m1p9p1s9s1z2z3z4z5z6z7z 1m1m4m5m1p4m2m2m2p2p2s2s2z	Case #1: 927105416 Case #2: 100000041

Problem J. Serval and Essay

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Serval is a new student in Japari Kindergarten.

There is an English course in the kindergarten. The teacher sets some writing tasks for students to improve their writing skills. Therefore, Serval has to complete an essay, in English.

You might know that in an essay, the author has to convince readers of several **arguments** by showing them evidence.

Serval has collected n available arguments numbered from 1 to n that can be written in his essay. For the i-th argument, Serval can conclude that i-th argument is true when the arguments numbered $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$ are all true. Specially, the i-th argument cannot be proven true by making conclusion when $k_i = 0$. It is guaranteed that $a_{i,j} \neq i$ for all i $(1 \leq i \leq n)$, and $a_{i,j} \neq a_{i,k}$ when $j \neq k$.

At the beginning of his essay, Serval will set exactly one argument from all the arguments as the **argument basis**, which is regarded as true. Starting with the argument basis, Serval will claim that some other arguments are true by making conclusions to complete his essay. It can be shown that for the i-th argument with $k_i = 0$, it can be true if and only if it is the argument basis.

Serval wants to maximize the number of true arguments in the essay, so he needs to set the argument basis optimally. However, as a kindergarten student, he cannot even find out the number of true arguments he can obtain.

Could you help him find out the answer?

Input

Each test contains multiple test cases.

The first line contains an integer T $(1 \leq T \leq 10^5)$ – the number of test cases.

For each test case:

The first line contains a single integer n $(1 \leq n \leq 2 \times 10^5)$ – the number of the available arguments.

For the i-th line in the following n lines, there is an integer k_i $(0 \leq k_i < n)$ followed by k_i integers $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$ $(1 \leq a_{i,j} \leq n, a_{i,j} \neq i)$ – the arguments required to make the conclusion for the i-th argument. It is guaranteed that $a_{i,j} \neq i$ for all i $(1 \leq i \leq n)$, and $a_{i,j} \neq a_{i,k}$ when $j \neq k$.

It is guaranteed that the sum of n in all the test cases does not exceed 2×10^5 and the sum of k_i in all the test cases does not exceed 5×10^5.

Output

For the i-th test case, print a line containing "`Case #i: x`" (without quotes), where i is the number of the test case starting from 1 and x is the answer to the test case, which is the maximum number of true arguments when setting the argument basis optimally.

Example

standard input	standard output
3 4 0 1 1 2 1 2 2 2 3 5 1 3 1 1 1 2 1 5 4 1 2 3 4 7 0 2 1 4 1 2 1 3 2 3 4 1 1 2 5 6	Case #1: 4 Case #2: 3 Case #3: 4

Note

For the first sample, Serval can set the 1-st argument as the argument basis to obtain 4 true arguments in the essay. The following process shows how Serval makes conclusions in his essay.

- Set the 1-st argument as the argument basis, which is regarded as true.
- Conclude that 2-nd argument is true because 1-st argument is true.
- Conclude that 3-rd argument is true because both 1-st and 2-nd arguments are true.
- Conclude that 4-th argument is true because both 2-nd and 3-rd arguments are true.

Problem K. Villages: Landcircles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

Because of the great success of *Villages: Landlines*, its development team, *NIO Tech*, decides to develop its sequel. They call it *Villages: Landcircles*. In the sequel, the development team decides to use a new power system.

They change the power system from one dimension to two dimensions. In the new power system, initially there are several power stations and buildings. The goal of the game is to make every power stations and buildings connected together in the power system, forming a connected graph. Players can build some power towers and wires in the game, which are the tools to connect power stations and buildings.

Power stations or buildings can connect to power towers without wires, while power towers must connect to each other by wires. What's more, power stations and buildings can't connect to each other directly, they must connect through power towers.

Assuming that the coordinates of a building are (x_i, y_i) and its receiving radius is r_i, all the power towers whose distance from the building is no greater than r_i are directly connected to it without wires. That is to say, for the power tower located at (x, y), it is directly connected to the building at (x_i, y_i) without wires if and only if $\sqrt{(x - x_i)^2 + (y - y_i)^2} \leq r_i$. Similarly, assuming that the power supply radius of a power station is r_s, all the power towers whose distance from the power station is no more than r_s are directly connected to it without wires. Supposing that the coordinates of two power towers are (x_A, y_A) and (x_B, y_B), players can connect them with the wire, and the length of wire required is $\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$. A power tower can be connected to any number (possibly zero) of power towers, buildings and power stations.

In order to make the game more friendly to the rookies, Mocha, a member of *NIO Tech*, decides to develop the power distribution recommendation function. In the case of using any number of power towers, players can choose **exactly one** power station and several buildings to get an optimal power supply scheme, which uses the shortest total length of wires to complete the power system. However, Mocha is not sure whether her scheme is correct, so she needs your help to calculate the total length of wires used in the optimal scheme to determine the correctness of her scheme.

Input

The first line contains a single integer T ($1 \leq T \leq 10^3$) – the number of the test cases. For each test case:

The first line contains a single integer n ($2 \leq n \leq 9$).

The second line contains three integers x_s, y_s, r_s ($-2000 \leq x_s, y_s \leq 2000, 1 \leq r_s \leq 2000$) – the coordinates of the power station and its power supply radius.

The i-th line of the next $n-1$ lines contains three integers x_i, y_i, r_i ($-2000 \leq x_i, y_i \leq 2000, 1 \leq r_i \leq 2000$) – the coordinates of the i-th building and its receiving radius.

It is guaranteed that for any two distinct buildings i and j, $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} > r_i + r_j$ holds. Similarly, it is guaranteed that for any building i, $\sqrt{(x_i - x_s)^2 + (y_i - y_s)^2} > r_i + r_s$ holds.

It is guaranteed that there are no more than 10 test cases with $n \geq 8$ in a test.

Output

For each test cases, print a real number in a single line, denoting the total length of wires used in the optimal scheme. Your answer will be considered correct if its relative or absolute error does not exceed 10^{-6}.

Example

standard input	standard output
3 2 0 0 1 2 2 1 3 0 0 10 20 -1 1 -20 -1 1 3 0 0 1 1000 -1000 1 -1000 -1000 1	0.828427124746190098 18.049968789001571454 2729.050807568877293541