//MIUP'18

Maratona Inter-Universitária de Programação

Universidade da Beira Interior
Departamento de Informática

13 | Outubro | 2018
Covilhã

Maratona Inter-Universitária de Programação

October 13^{th}, 2018 - 10:00 to 15:00

Readiness IT

Tezos

Universidade da Beira Interior

Águas da Covilhã

Câmara Municipal da Covilhã

Fruition Partners

Outsystems

WEB Eindhoven

Contents

Scientific Committee

- André Restivo — Faculdade de Engenharia, Universidade do Porto
- Fábio Marques — Escola Superior de Tecnologia e Gestão de Águeda, Universidade de Aveiro
- Filipe Araújo — Faculdade de Ciências e Tecnologia, Universidade de Coimbra
- Luís Alexandre — Faculdade de Engenharia, Universidade da Beira Interior
- Margarida Mamede — Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
- Mário Pereira — Laboratoire de Recherche en Informatique, Universite Paris-Sud
- Paul Crocker — Faculdade de Engenharia, Universidade da Beira Interior
- Pedro Guerreiro — Faculdade de Ciências e Tecnologia, Universidade do Algarve
- Pedro Mariano — Faculdade de Ciências, Universidade de Lisboa
- Pedro Ribeiro — Faculdade de Ciências, Universidade do Porto
- Rui Mendes — Escola de Engenharia, Universidade do Minho
- Simão Melo de Sousa — Faculdade de Engenharia, Universidade da Beira Interior
- Vasco Pedro — Escola de Ciências e Tecnologia, Universidade de Évora

Local Organization Committee

- David Carvalho — Faculdade de Engenharia, Universidade da Beira Interior
- Eliana Mendes — Faculdade de Artes e Letras, Universidade da Beira Interior
- João Reis — Faculdade de Engenharia, Universidade da Beira Interior
- José Costa — Faculdade de Engenharia, Universidade da Beira Interior
- Luís Horta — Faculdade de Engenharia, Universidade da Beira Interior
- Nuno Pereira — Faculdade de Engenharia, Universidade da Beira Interior
- Paul Crocker — Faculdade de Engenharia, Universidade da Beira Interior
- Sebastião Pais — Faculdade de Engenharia, Universidade da Beira Interior
- Simão Melo de Sousa — Faculdade de Engenharia, Universidade da Beira Interior
- NINF — Núcleo de Informática da Universidade da Beira Interior

Compilers

language	compiler version	compile cmd	*execute cmd*
C:	gcc $\geq$ 8.1	gcc -std=gnu11 -Wall *name.c* -lm	*a.out*
C++:	g++ $\geq$ 8.1	g++ -std=gnu++11 -Wall *name.cpp* -lm	*a.out*
Java:	javac $\geq$ 1.8	javac -encoding utf8 *name.java*	*java -Xmx256M -Xss256M* name

Compilation Constraints

- **Maximum compilation time:** 60 seconds
- **Maximum source code size:** 100 KB
- Every source code must be submitted in a single file
- In case of Java submissions, the `.java` file has to have the same name as the class that contains the main method. There is no limit for the number of classes to be contained in that file.

Runtime Constraints

These limits apply to all problems.

- **Maximum CPU time:** 2 seconds

Available Editors

- Codeblocks
- Eclipse
- Emacs
- Geany
- Gedit
- Kate
- Netbeans
- Sublime Text
- Vim

About the Input/Output

- All lines (both in the input and output) should be ended by the newline character (`'\n'`)
- Except when explicitly stated, single space is used as a separator.
- No line starts or ends with any kind of whitespace.

Documentation

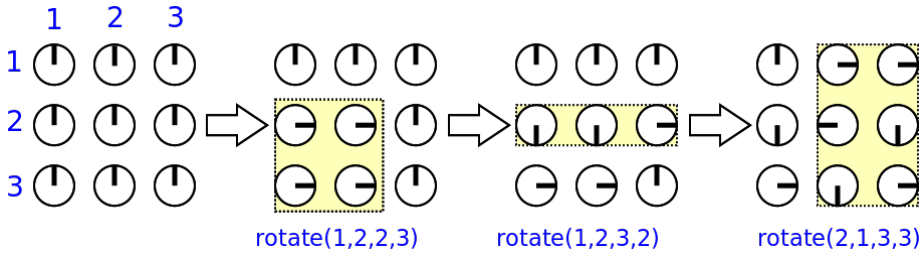
Language documentation is available, namely:

- man pages for C/C++ function *(using the command line)*
- C++ STL documentation *https://miup2018.di.ubi.pt/~mooshak/doc/*
- Java SE Documentation *https://miup2018.di.ubi.pt/~mooshak/doc/*

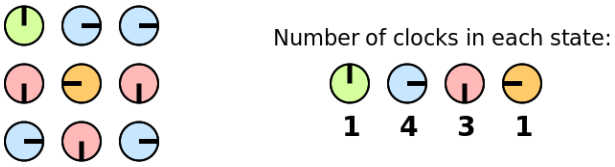
Problem A: Clocks

Mary has a new game consisting of a square of $N \times N$ clocks, each with a single pointer indicating the current hour. Initially, all the clocks are set to 12h. She can modify the current configuration by rotating the pointer in all the clocks of any subrectangle (x_1, y_1, x_2, y_2) by 90 degrees.

The following figure illustrates a square of 3×3 clocks in its initial position, followed by three example rotations (indicated by the yellow subrectangles).



After applying the rotations, Mary wants to know the final state of the clocks. Specifically, she wants to know how many clocks are in each of the possible 4 states (12h, 3h, 6h and 9h). For the example given in the previous figure, after the three rotation operations, she would end up respectively with 1, 4, 3 and 1 clocks in each of the given states, as explained in the following figure:



Mary would really like to know the final state of the clocks without actually having to manually rotate all the clocks for each operation. Can you help her?

Task

Given an $N \times N$ square of clocks and R rotation operations, each one identifying a subrectangle (x_1, y_1, x_2, y_2) of clocks that should be rotated by 90 degrees, your task is to indicate how many clocks will be in each of the possible 4 final states (12h, 3h, 6h and 9h). Note that after four rotations, a clock's pointer will return to its original 12h position.

Input

The first line contains **N**, indicating you should consider an $N \times N$ square of clocks. The second line contains **R**, the number of rotation operations. R lines follow, each one with four integers x_1 y_1 x_2 y_2 indicating that the clocks in the subrectangle (x_1, y_1, x_2, y_2) are to be rotated by 90 degrees.

Constraints

$1 \leq \mathbf{N} \leq 1000$	dimension of the square
$0 \leq \mathbf{R} \leq 50\,000$	number of rotation operations
$1 \leq x_1 \leq x_2 \leq \mathbf{N}$	x limits defining the subrectangle of one operation
$1 \leq y_1 \leq y_2 \leq \mathbf{N}$	y limits defining the subrectangle of one operation

Output

The output should have 4 integers, separated by a single space, indicating respectively the number of clocks in the 12h, 3h, 6h and 9h positions, after applying all the rotation operations.

Sample Input

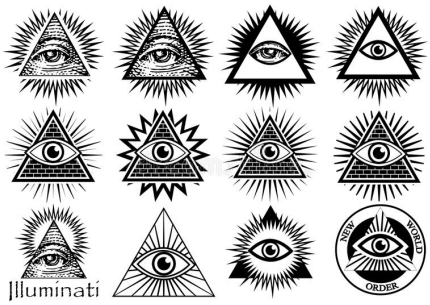
```
3
3
1 2 2 3
1 2 3 2
2 1 3 3
```

Sample Output

```
1 4 3 1
```

Note that the sample input corresponds to the figure in the problem statement.

Problem B: The Conspiracy



You suspect the Big Four, the Majestic 12 and the Illuminati exist. They are part of a monstruous conspiracy that involves the aliens to take over the world and turn us into slaves! Someone sent you a large number of files infected with secret messages. We know that all the messages are about the same thing and that only what is repeated in all the messages is in fact the conspiracy. We must find out! The truth will set us free!

In order to decrypt the secret message, you need to perform a transformation to the files. The transformation involves [redacted]. Well, of course you can't know this information because of the conspiracy! What are you thinking?

After applying this transformation to the files, we want to find the largest portion of text that exists on all the files. This will be the secret message. I'll give you the files after the transformation. You must find the common message!

Task

Your task is, given several lines of text, to find the largest substring that exists on all of them.

Input

You are given N lines of input, each with a maximum of S characters. The input is already preprocessed and consists only of lowercase letters. Each line contains one string.

Constraints

$1 \leq N \leq 100$ Number of strings.
$1 \leq S \leq 400$ Maximum length of each string.

Output

One line, printing the size of the largest common substring found.

Sample Input 1

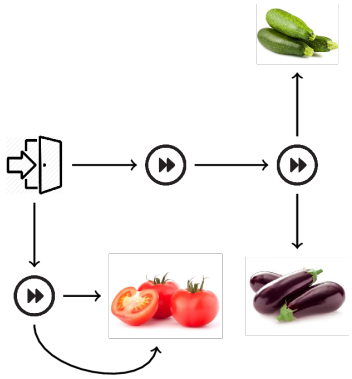
```
ponderation
respondant
despondant
ponder
```

Sample Output 1

```
4
```

Problem C: Let's cook

Welcome to the Inner Border, the first step to Paris! As our most distinctive guests, tonight you will have the pleasure to taste a meal prepared by our most famous *chef*, the little mouse António. You will be presented with many local tasty food, such as *cherovia* from Covilhã, the cherries from Fundão or the soup. But there is a little problem: António's fat-greedy brother, Bernardo, has stolen all the needed ingredients. Bernardo ran into the sewers of Covilhã and hid the fresh and tasty food in his secret rooms, so that he can eat them later that night. Moreover, being so desperately hungry, he spreads the different ingredients in two different zones of Covilhã, hoping to further confuse António. Here is the map of a part of the sewers:



The door marks the entry point and the double arrow marks a connection point. A dead-end corresponds to Bernardo's secret rooms, where he stocked some food. As you can notice, the *Covilhanenses* sewers are very old, having at most two connections for each connection point. Being always very afraid to lose himself, once Bernardo enters the sewer he decides upon a direction and then he always takes that direction to continue hiding food. Moreover, if he gets back to a room he already visited he leaves another piece of the same ingredient already stocked.

With the help of his valuable mice friends, António knows which direction his brother took for each sewer. He now possesses every piece of information he needs in order to retrieve the ingredients. However, before beginning his quest for the stolen food, António asks himself a question: if he exactly follows the same path as his brother, would he retrieve the same sequence of ingredients in both sewers? For the previous map, if Bernardo had decided to go left then António would find the sequence Zucchini, Eggplant, Tomato, and Tomato. On the other hand, had Bernardo chosen to go right, the little *chef* would find Tomato, Tomato, Eggplant, and Zucchini.

Task

Your task is to write a program to help António discover if he is going to find the same sequence of ingredients in both sewers, following the traversal of this brother.

Input

The input is divided in two sections, each one describing a sewer and starting with

$k\ t\ j$

where k represents the number of points in this sewer, t is the direction Bernardo took, and j is the index of the entry point for this sewer. t is either the character `R` (going right) or `L` (going left). After each $k\ t\ j$ line you will have k lines either of the form

i `C` $n\ m$

or of the form

i `D` f

where $0 \leq i < k$, $0 \leq n < k$, and $-1 \leq m < k$, and i is the index of this point. All i from 0 to $k-1$ are presented, without repetitions. If after i there is the character `C`, then you know it is a connection point, where n and m are the indexes of the left and right connections, respectively (if a point has only one connection we mark the second one with -1). If after i there is a `D`, then you know it is a dead-end and f is the number that represents a certain ingredient.

Constraints

$0 \leq k < 60$
$0 \leq f < 100$

Output

`YES` if António finds the exact same sequence of ingredients in both sewers, `NO` otherwise.

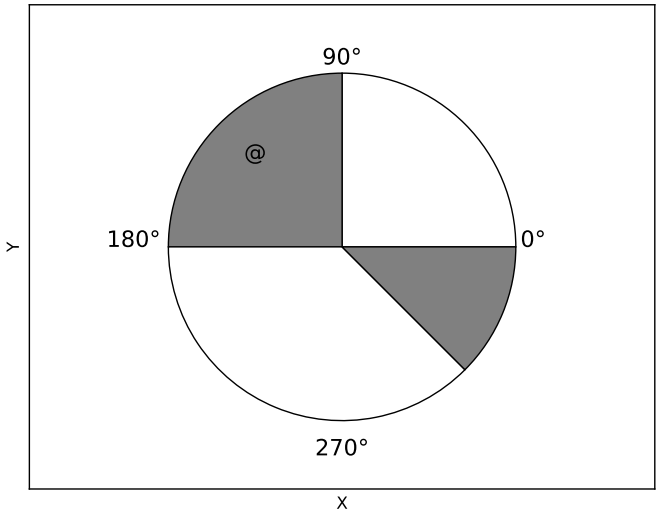
Sample Input

```
7 L 0
2 C 4 4
3 C 5 6
4 D 2
0 C 1 2
1 C 3 -1
5 D 0
6 D 1
7 R 0
3 D 1
2 D 0
1 C 3 2
4 C 6 5
0 C 4 1
5 D 2
6 D 2
```

Sample Output

```
YES
```

Problem D: The Falling Drops



Consider a flat white circular dish, that spins around an axis that crosses its center perpendicularly.

This dish was painted gray in certain circle sectors (or pie slices).

There is a tap that is leaking water drops on the dish as it spins counterclockwise. The at symbol (@) represents the position of the tap, which in the figure above is at an angle of 135 degrees. The tap cannot, for mysterious reasons, be placed over the centre of the dish.

The tap is so close to the dish that we consider that the drop falls instantly onto the dish.

If a drop falls onto the border of a slice we consider it as falling onto the slice.

Task

We want to know how many drops fall onto gray areas before the first drop falls onto a white area.

Input

All angles are measured with respect to the X-axis, from right to left, counterclockwise.

The dish rotates with an angular speed ω that is given in degrees per second. Time $t \in \mathbb{N}_0$ is measured in seconds.

The tap is placed over the dish at an angle z (an integer).

The number of slices is n.

The period of the tap is an integer, p, that represents the number of seconds until the tap behaviour repeats itself.

Each input consists of seven lines:

- the first five lines contain ω, z, t_{max}, n and p, one value per line, in this order;
- the sixth line contains the direction of the slices' edges: $s_1, e_1, s_2, e_2, \ldots, s_n, e_n$. Slice i covers the area between the start angle s_i and the end angle e_i. The slices do not touch nor overlap each other (except at the centre of the dish).
- the seventh line contains a sequence of p whitespace separated 0s and 1s that represents the behaviour of the tap during one period, where a 0 stands for no drop and a 1 for a drop falling. Each of these symbols corresponds to the behaviour of the tap in one second, and the first symbol corresponds to time 0.

Constraints

$1 \leq \omega \leq 1000$	Angular speed in degrees per second (integer value).
$0 \leq z < 360$	Angle of the tap.
$1 \leq n \leq 100$	Number of slices.
$0 \leq s_i < e_i \leq 360$	Start and end angles of slice i, in degrees (integer values).
$1 \leq t_{max} \leq 100000$	Maximum number of seconds to consider (integer value).
$1 \leq p \leq t_{max}$	The period of the tap.

Output

The output is an integer number representing the number of drops that fell on gray areas before the first drop fell on a white area, or until the end of the experiment, whichever happens first.

Sample Input 1

```
45
135
100
2
2
90 180 315 360
1 0
```

Sample Output 1

```
1
```

Sample Input 2

```
45
35
100
3
5
30 36 137 180 215 360
1 1 1 0 0
```

Sample Output 2

```
4
```

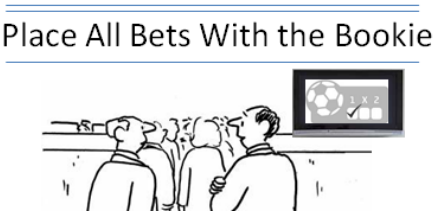
Sample Input 3

```
45
36
100
3
8
30 36 137 180 215 360
1 1 1 0 0 0 0 0
```

Sample Output 3

```
39
```

Problem E: Fair Bet



It's a well known fact that people will bet on anything ! This problem concerns betting, in fact making sure that the punters (as those in the business call those who place the bets) make fair bets.

Well, what's a fair bet you may ask ? Well a fair bet is one that the punter actually has a chance of winning and for which it's therefore valid for a betting shop (the so called book keepers or bookies as they are known in the trade) to give the poor punter some odds (odds represent the ratio between the amounts staked by parties to a wager or bet). The poor punter might only have a small negligible probability of winning, but he can not have a zero probability of winning - that would not be playing fair. In this problem your job is to help the book keepers understand when they can accept a valid bet from a punter, it's up to the bookkeeper of course to make the odds and hope he makes a profit and does not go bankrupt, but that's another story.

Anyway the punters in this case want to place a bet on the results of a tournament of events, such as a football league, where all the teams play each other and are awarded points based on the result of each game. At the end of the season, or group phase in the world cup, each team will therefore have a total score of points, in other words we shall have a set or sequence of scores. Punters want to place a bet on a particular set of scores, the problem is of course not all possibilities are fair bets. Your job is to help the betting shop and bookies to determine if a given sequence of scores is valid or not.

Task

For the sake of argument we shall say that we have N teams and each team only plays another team once and that the point system is the one that the author grew up with, that is 2 points for a win and 1 for a draw.

Lets imagine a tournament where only the Big three play each other i.e Sporting Covilhã, Evorense and Farense.

Obviously the bet (4,1,1) 4 points for Sporting Covilhã 1 for Evorense and one for Farense is a fair bet, here Covilhã won both its games but Evora and Faro could only manage a draw. The sequences (4,2,2) and (6,0,0) are clearly both not fair bets, I'm sure that you can see why.

As an other example let's have Braga join the tournament as the second (and highest pointing) team in the sequence, then the sequence(2,6,2,1) Sporting Covilhã 2 points, Braga

6 points, 2 for Evorense and 1 for Farense also can't be a fair a bet.

Input

The input starts with one line that contains and integer : the number of teams, N, in the tournament ($N \leq 70$).

This is then followed by N lines, each line represents the i'th team and each line contains T ($0 \leq t \leq N$) the total number of points that the punter thinks that the team will have.

Output

The output is either the word `fair` or `reject` followed by the new line character.

Sample Input 1

```
6
8
6
10
1
1
4
```

Sample Output 1

```
fair
```

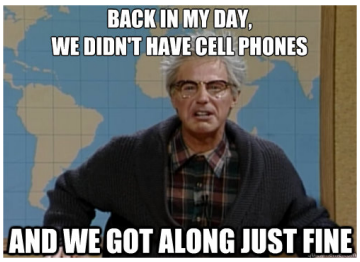
Sample Input 2

```
6
8
2
10
8
2
0
```

Sample Output 2

```
reject
```

Problem F: Old School



I am an old school programmer. I do not trust resizable arrays. Thus, whenever I declare an array, I make sure its capacity is sufficient for all computations in the program. The other day, I was passing time programming for the N^{th} time a function to list the divisors of a positive integer. I wanted to store the divisors in an array. How large must that array be?

In general, if the largest number whose divisors I want is X, what is the minimum capacity for the array of divisors?

Task

Write a program that, given a positive integer X, computes the minimum capacity necessary for an array that will store the divisors of any positive integer less or equal to X.

Constraints

$1 < X \leq 10^{15}$

Input

The number X used in the problem description is to be read from the console.

Output

The value of the computed capacity, certainly a positive integer, shall be written to the console.

Sample Input

```
30
```

Sample Output

```
8
```

Problem G: Luigi's Pizzeria

Luigi's new pizzeria has been the talk of the town in the past few weeks. Not only because it has the best pizzas you can find for miles, but also because of its crazy *all you can eat* policy.

You see, Luigi's pizzas are enormous, and they are cut into very thin slices. And that's not even the crazy part! Each slice has different ingredients and you can eat as many slices as you want. But there is one small caveat. You can only select adjacent slices and you have to eat them all! It is therefore very tricky for each client to select the best part of the pizza according to his own taste.

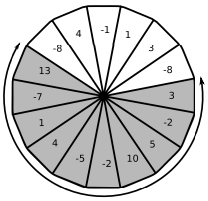


Figure 1: Selected slices must be adjacent. The section in grey has a score of 20.

Task

You enter the restaurant and see that today's special pizza has been cut into N slices. After attributing scores $(S_1, ..., S_N)$ to each one of the slices, you must devise an algorithm that selects the section of the pizza that yields the best value according to those scores. The value of a section of pizza is the sum of the scores of its slices. Notice that slice scores can be negative.

Input

The first line of the input contains a single integer N representing the number of slices in the pizza. The second line contains N integers (S_i) that represent the score you attributed to each slice.

Constraints

$1 \leq N \leq 50\,000$	Number of slices.
$-100 \leq S_i \leq 100$	Value of each slice.

Output

A single line with one integer that is equal to the value of the best possible section of adjacent pizza slices. The smallest possible section would be a single slice.

Sample Input 1

```
4
2 -2 3 -1
```

Sample Output 1

```
4
```

Sample Input 2

```
16
-1 1 3 -8 3 -2 5 10 -2 -5 4 1 -7 13 -8 4
```

Sample Output 2

```
20
```

Sample Input 3

```
4
-1 -2 -3 -4
```

Sample Output 3

```
-1
```

Problem H: How many will get to safety

A pack of hungry wolves.
A flock of fluffy sheeps.
A sheepdog pack.
How many sheeps will escape?

Task

Knowing the location of the shelters and the starting position of each of the animals, determine how many sheeps can hide themselves knowing that:

- All animals move horizontally and vertically. Horizontal movement is a priority.
- A sheep goes to the closest shelter 1 house per shift and its movement has priority over that of the remaining animals.
- A wolf tries to intercept the nearest sheep, without changing sheep, and moves to 2 houses per turn. If the wolf catches the sheep, he gets satiated and the 2 animals get out of the problem. A wolf runs away (gives up pursuing sheep and disappears from the problem) whenever he has a dog 3 houses or less away. A wolf would rather flee than to eat a sheep. A wolf also gets out of the problem if another wolf eats the sheep he was chasing or if the sheep manages to get to the shelter. The movement of the wolf takes precedence over that of the dog.
- A dog runs after the nearest wolf which is pursuing sheeps, 2 houses per turn, and pursues another wolf which in the meantime gets closer. In the event that the dog is at the same distance from two or more wolves, then the dog chases the wolf that was read first.
- There are no ambiguous beginnings, that is, at the beginning each sheep has only one nearest shelter, each wolf has only one nearest sheep, and each dog has only one nearest wolf.

Input

The first line is composed by two integers values, separated by a space, which define the size of the terrain in columns c and lines l. The number w of wolves is given in the second line, while the number s of sheeps and the number d of sheepdogs are given in the third and the fourth line, respectfully. In the fifth line the number t of shelters is given. The remaining $w+s+d+t$ lines define the coordinates of wolves, sheeps, sheepdogs and shelters, respectfully, and with the following format: x y. Where x represents the column and y the line.

Constraints

The range of each variable used in the Input section and its meaning.

$1 \leq c, l \leq 10\,000$	Dimension of the terrain.
$1 \leq w \leq 100$	Number of wolves.
$1 \leq s \leq 100$	Number of sheeps.
$1 \leq d \leq 100$	Number of sheepdogs.
$1 \leq t \leq 100$	Number of shelters.
$0 \leq x < c$, $0 \leq y < l$	Starting location of each wolf, sheep, sheepdog and shelter.

Output

The output consists of a single line with the number of the sheeps that managed to escape the pack of wolves.

Sample Input 1

```
5 6
4
4
4
2
0 0
0 2
0 4
1 4
0 3
1 5
4 1
4 0
2 0
2 4
3 0
```

```
3 4
3 3
4 2
```

Sample Output 1

```
4
```

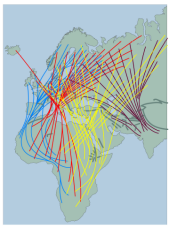
Sample Input 2

```
16 15
4
4
4
5
5 0
1 12
8 8
14 3
2 1
7 6
11 4
7 11
3 9
11 9
9 3
14 11
2 5
7 10
10 2
14 6
12 13
```

Sample Output 2

```
3
```

Problem I: Goin' south



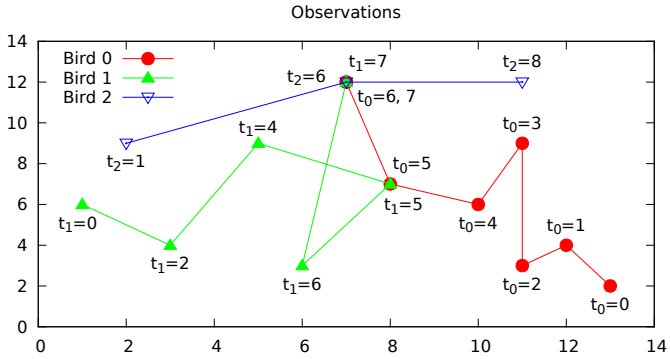
Scientists love studying, and have been studying the migration patterns of birds for decades. Things are much easier now than at the beginning, of course, as they can fit a few thousand birds with GPS trackers and leave the old binoculars at home.

As scientists started looking for the signs of climate changes in the migration patterns, they also wondered whether birds, coming from or going to different parts of the world, might meet and travel together for part of the way, just crossed paths every once in a while, or avoided each other altogether.

Task

An observation of a bird consists of the identification of a time instant and of the coordinates of the location of the bird at that time.

Given a series of observations of several birds, your task is to build a program to compute the number of times the birds were observed at the same location at the same time.



In the figure, which plots the observations of the three birds from the **Sample Input** below, t_0, t_1 and t_2 are the times of the observations of, respectively, birds 0, 1 and 2. Birds 0 and 1 were observed at the same location at time instants $t_0 = t_1 = 5$ and $t_0 = t_1 = 7$, but at no time were all birds observed at the same location at the same time.

Input

The first line of input contains the number **B** of birds tracked and the total number **O** of observations of birds. Individual birds are identified by an integer between 0 and $\mathbf{B} - 1$.

The following **O** lines contain the observations of birds, consisting of four integers separated by single spaces: the identification of the bird; the time **t**; and the longitude **x** and the latitude **y** of the observation. There is at least one observation of every tracked bird and, for each bird, the observations are listed in strictly ascending order by **t**.

The following line contains the number **T** of test cases, which are contained in the next **T** lines. Each of the lines describing a test case contains the number **n** of birds in the test case, followed by **n** distinct bird identifiers, all separated by single spaces.

Constraints

$2 \leq \mathbf{B} \leq 100\,000$	Number of birds.
$2 \leq \mathbf{O} \leq 200\,000$	Number of observations.
$1 \leq \mathbf{T} \leq 25$	Number of test cases.
$0 \leq \mathbf{t} < 2^{31}$	Time.
$-179\,999 \leq \mathbf{x} \leq 180\,000$	Longitude.
$-66\,000 \leq \mathbf{y} \leq 66\,000$	Latitude.
$2 \leq \mathbf{n} \leq \mathbf{B}$	Number of birds in a test case.

Output

The output consists of **T** lines. The i^{th} line contains an integer with the number of times all the birds in the i^{th} test case were observed at the same location at the same time.

Sample Input

```
3 17
0 0 13 2
0 1 12 4
0 2 11 3
0 3 11 9
2 1 2 9
0 4 10 6
0 5 8 7
0 6 7 12
0 7 7 12
1 0 1 6
2 6 7 12
2 8 11 12
1 2 3 4
1 4 5 9
1 5 8 7
1 6 6 3
1 7 7 12
3
2 0 1
2 2 0
3 1 0 2
```

Sample Output

```
2
1
0
```

Problem J: A tale never loses in the telling

When we tell a piece of news to someone and that person tells it to someone else, there are always differences in the transmitted information. The content can change a lot or very slightly, depending on the accuracy of the intermediate individual. That is why someone can hear quite different versions of the same facts.

Let the *interference degree* of a person be the degree of inexactness with which the person reproduces information. Now, suppose that a piece of information travels across $m+1$ distinct individuals, p_0, p_1, ..., p_m, whose interference degrees are i_0, i_1, ..., i_m, respectively. So, the source of the information is person p_0, who tells it to p_1, then p_1 tells it to p_2, then p_2 tells it to p_3, and so on. The *noise* in the information, which somehow quantifies its extent of inexactness, is defined as follows (where N is a positive number, called the *noise parameter*). When the information reaches p_1, the noise is 0. When the information reaches p_2, the noise is i_1. When the information reaches p_3, the noise is $i_1 N + i_2$. In the general case, when the information reaches p_m (for every $m \geq 2$), the noise is:

$$i_1 \, N^{m-2} + i_2 \, N^{m-3} + \cdots + i_{m-2} \, N^1 + i_{m-1} \, N^0 \, .$$

Task

Given a set of people, their interference degrees and their friendship relations, two distinct individuals (s and t) and a noise parameter, the goal is to compute the minimum possible noise (W) in a piece of information whose source is s when it reaches target t. Since noise values can be very large numbers, your program should write the value of W modulo 2^{45}.

You may assume that all pieces of information whose source is s always reach target t.

Input

The first line has four integers: P, which is the number of people; s, which is the source of the piece of information; t, which is the target person; and N, which is the noise parameter. Individuals are identified by integers, ranging from 0 to $P - 1$. Notice that $s \neq t$.

The following P lines contain each a single integer. The first of these lines has the interference degree i_0 of person 0, the second one has the interference degree i_1 of person 1, and so on. So, the last of these lines has the interference degree i_{P-1} of person $P-1$.

The next line contains a single integer, R, which is the number of friendship pairs. Each of the following R lines specifies a different friendship pair. It has two distinct integers, x and y, representing that persons x and y tell news to each other.

Constraints

$2 \leq P < 50\,000$	Number of people.
$P < N \leq 50\,000$	Noise parameter.
$1 \leq i_k \leq P$	Interference degree of person k (for $k = 0, 1, \ldots, P-1$).
$1 \leq R \leq 100\,000$	Number of friendship pairs.

Output

A single line with the minimum possible noise of a piece of information whose source is s when it reaches target t, modulo 2^{45}.

Sample Input

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

Sample Output

```
37
```