A. Nate and Integer Coefficient

time limit per test: 1.5 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nate's math teacher thinks he watches too much anime and not enough time studying for their algebra test. Nate insists that he's already prepared for the test, but in order to prove it, he has to solve the following problem that his teacher gave him.

Given a, n, and that $x^2 - ax + 1 = 0$, find b such that $x^{2n} - bx^n + 1 = 0$.

Input

The first line of input contains a single integer T, the number of test cases. The next T lines of input each contain two space-separated integers a and n, respectively, the values in the equation.

Constraints

$1 \leq T \leq 10^5$

$|a|, |n| \leq 10^{18}$

Output

For each of the T test cases, output in its own line a single integer, b, that satisfies the equation. Since the answer can get quite large, output only the positive remainder when b is divided by $10^9 + 7$, a prime number. It can be proven that b is always an integer.

Example

input

```
3
2 1231
2 10000000
5 0
```

output

```
2
2
2
```

B. Combination

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

Ilya plays a card game by the following rules.

A player has several cards. Each card contains two non-negative integers inscribed, one at the top of the card and one at the bottom. At the beginning of the round the player chooses one of his cards to play it. If the top of the card contains number a_i, and the bottom contains number b_i, then when the player is playing the card, he gets a_i points and also gets the opportunity to play additional b_i cards. After the playing the card is discarded.

More formally: let's say that there is a counter of the cards that can be played. At the beginning of the round the counter equals one. When a card is played, the counter decreases by one for the played card and increases by the number b_i, which is written at the bottom of the card. Then the played card is discarded. If after that the counter is not equal to zero, the player gets the opportunity to play another card from the remaining cards. The round ends when the counter reaches zero or the player runs out of cards.

Of course, Ilya wants to get as many points as possible. Can you determine the maximum number of points he can score provided that you know his cards?

Input

The first line contains a single integer n $(1 \le n \le 1000)$ — the number of cards Ilya has.

Each of the next n lines contains two non-negative space-separated integers — a_i and b_i $(0 \le a_i, b_i \le 10^4)$ — the numbers, written at the top and the bottom of the i-th card correspondingly.

Output

Print the single number — the maximum number of points you can score in one round by the described rules.

Examples

input

```
2
1 0
2 0
```

output

```
2
```

input

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

output

```
3
```

Note

In the first sample none of two cards brings extra moves, so you should play the one that will bring more points.

In the second sample you should first play the third card that doesn't bring any points but lets you play both remaining cards.

C. Choosing Two Paths

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

You are given an undirected unweighted tree consisting of n vertices.

An undirected tree is a connected undirected graph with $n-1$ edges.

Your task is to choose two pairs of vertices of this tree (all the chosen vertices **should be distinct**) (x_1, y_1) and (x_2, y_2) in such a way that neither x_1 nor y_1 belong to the simple path from x_2 to y_2 and vice versa (neither x_2 nor y_2 should not belong to the simple path from x_1 to y_1).

It is guaranteed that it is possible to choose such pairs for the given tree.

Among all possible ways to choose such pairs you have to choose one with the **maximum number of common vertices** between paths from x_1 to y_1 and from x_2 to y_2. And among all such pairs you have to choose one with the **maximum total length** of these two paths.

It is guaranteed that the answer with at least two common vertices exists for the given tree.

The length of the path is the number of edges in it.

The simple path is the path that visits each vertex at most once.

Input

The first line contains an integer n — the number of vertices in the tree ($6 \le n \le 2 \cdot 10^5$).

Each of the next $n-1$ lines describes the edges of the tree.

Edge i is denoted by two integers u_i and v_i, the labels of vertices it connects ($1 \le u_i, v_i \le n$, $u_i \ne v_i$).

It is guaranteed that the given edges form a tree.

It is guaranteed that the answer with at least two common vertices exists for the given tree.

Output

Print **any** two pairs of vertices satisfying the conditions described in the problem statement.

It is guaranteed that it is possible to choose such pairs for the given tree.

Examples

input

```
7
1 4
1 5
1 6
2 3
2 4
4 7
```

output

```
3 6
7 5
```

input

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

output

```
2 9
6 8
```

input

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

output

```
10 6
4 5
```

input

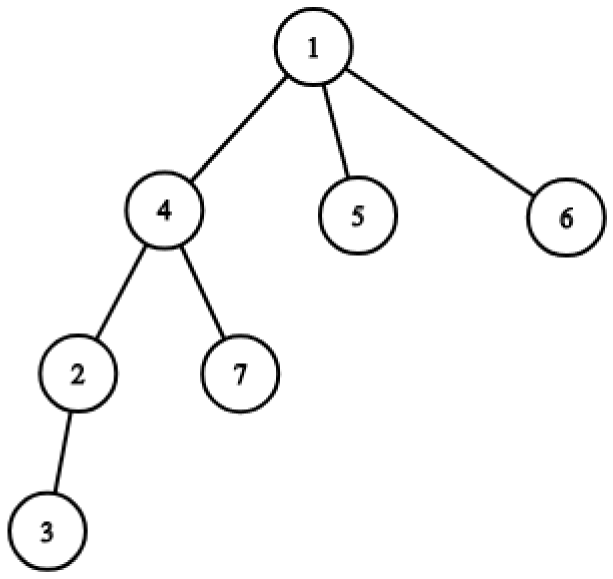
```
11
1 2
2 3
3 4
1 5
1 6
6 7
5 8
5 9
4 10
4 11
```

output

```
9 11
8 10
```

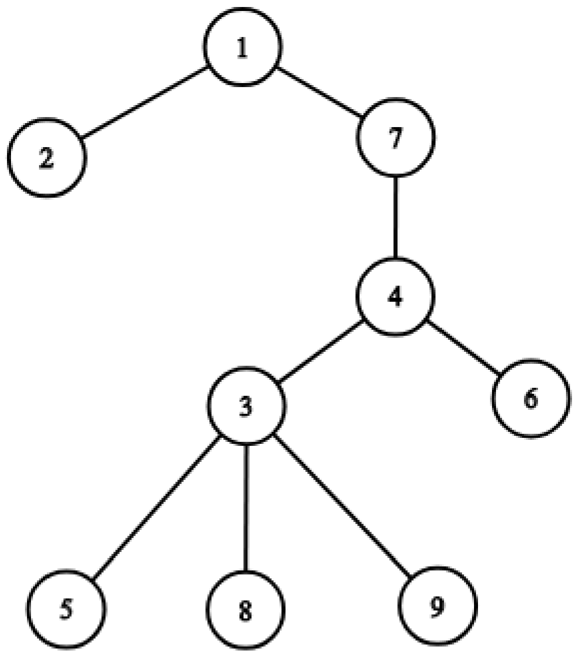
Note

The picture corresponding to the first example:



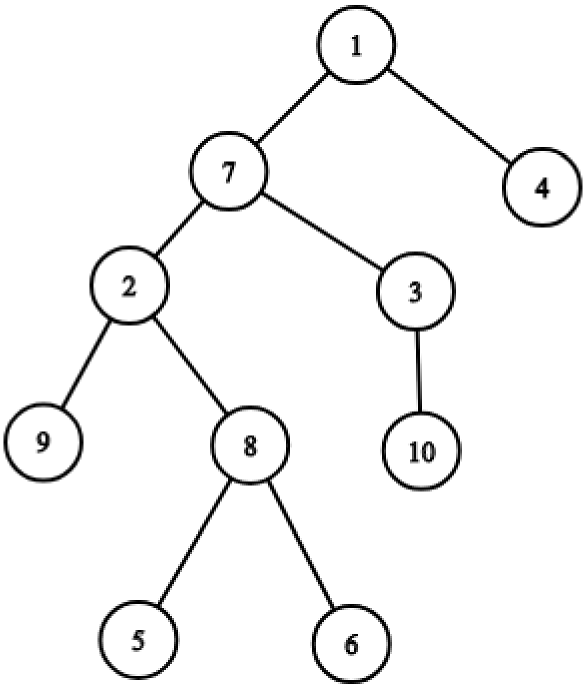
The intersection of two paths is 2 (vertices 1 and 4) and the total length is $4 + 3 = 7$.

The picture corresponding to the second example:



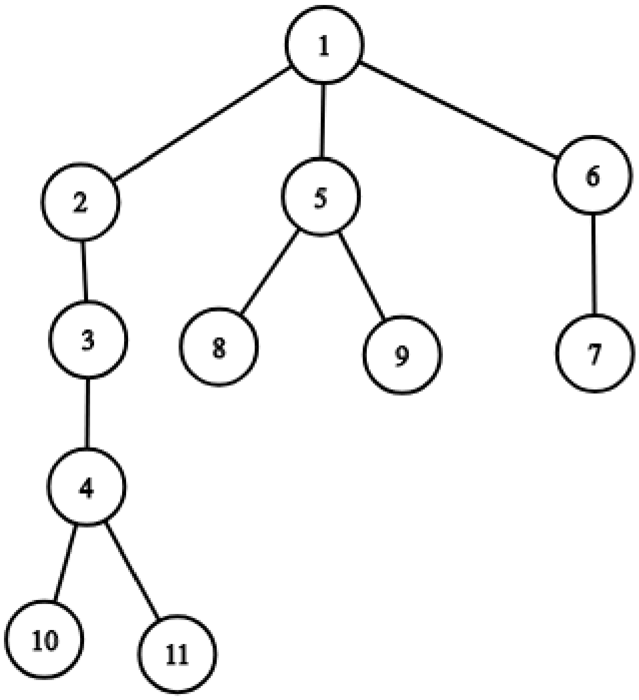
The intersection of two paths is 2 (vertices 3 and 4) and the total length is $5 + 3 = 8$.

The picture corresponding to the third example:



The intersection of two paths is 3 (vertices 2, 7 and 8) and the total length is $5+5=10$.

The picture corresponding to the fourth example:



The intersection of two paths is 5 (vertices 1, 2, 3, 4 and 5) and the total length is $6+6=12$.

D. Functional Palindromes

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's define a function, f, on a string, p, of length l as follows:

$$f(p) = (p_1 \cdot a^{l-1} + p_2 \cdot a^{l-2} + \ldots + p_l \cdot a^0) \mod (10^9 + 7)$$

where p_i denotes the ASCII value of the i-th character ($\mathtt{a} = 97$, $\mathtt{b} = 98$, ..., $\mathtt{z} = 122$) in string p and $a = 10^5 + 1$.

Nikita has a string, s, consisting of n lowercase letters that she wants to perform queries on. Each query consists of an integer, k, and you have to find the value of $f(w_k)$ where w_k is the k-th alphabetically smallest palindromic substring of s. If w_k doesn't exist, print -1 instead.

Input

The first line contains 2 space-separated integers describing the respective values of n (the length of string s; $1 \leq n \leq 10^5$) and q (the number of queries; $1 \leq q \leq 10^5$).

The second line contains a single string denoting s.

Each of the q subsequent lines contains a single integer denoting the value of k ($1 \leq k \leq \frac{n \cdot (n-1)}{2}$) for a query.

It is guaranteed that string s consists of lowercase English alphabetic letters only (i.e., a to z).

Output

For each query, print the value of function $f(w_k)$ where w_k is the k-th alphabetically smallest palindromic substring of s; if w_k doesn't exist, print -1 instead.

Example

input

```
5 7
abcba
1
2
3
4
6
7
8
```

output

```
97
97
696207567
98
29493435
99
-1
```

Note

There are 7 palindromic substrings of abcba. Let's list them in lexicographical order and find value of w_k:

1. $w_1 = \mathtt{a}$, $f(w_1) = 97$

2. $w_2 = $ `a`, $f(w_2) = 97$
3. $w_3 = $ `abcba`, $f(w_3) = 696207567$
4. $w_4 = $ `b`, $f(w_4) = 98$
5. $w_5 = $ `b`, $f(w_5) = 98$
6. $w_6 = $ `bcb`, $f(w_6) = 29493435$
7. $w_7 = $ `c`, $f(w_7) = 99$
8. $w_8 = $ doesn't exist, so we print -1 for $k = 8$

E. Sign Posts

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

One Khanate had a lot of roads and very little wood. Riding along the roads was inconvenient, because the roads did not have road signs indicating the direction to important cities.

The Han decided that it's time to fix the issue, and ordered to put signs on every road. The Minister of Transport has to do that, but he has only k signs. Help the minister to solve his problem, otherwise the poor guy can lose not only his position, but also his head.

More formally, every road in the Khanate is a line on the Oxy plane, given by an equation of the form $Ax + By + C = 0$ (A and B are not equal to 0 at the same time). You are required to determine whether you can put signs in at most k points so that each road had at least one sign installed.

Input

The input starts with two positive integers n, k ($1 \le n \le 10^5$, $1 \le k \le 5$)

Next n lines contain three integers each, A_i, B_i, C_i, the coefficients of the equation that determines the road ($|A_i|$, $|B_i|$, $|C_i| \le 10^5$, $A_i^2 + B_i^2 \ne 0$).

It is guaranteed that no two roads coincide.

Output

If there is no solution, print `"NO"` in the single line (without the quotes).

Otherwise, print in the first line `"YES"` (without the quotes).

In the second line print a single number m ($m \le k$) — the number of used signs. In the next m lines print the descriptions of their locations.

Description of a location of one sign is two integers v, u. If u and v are two distinct integers between 1 and n, we assume that sign is at the point of intersection of roads number v and u. If $u = -1$, and v is an integer between 1 and n, then the sign is on the road number v in the point not lying on any other road. In any other case the description of a sign will be assumed invalid and your answer will be considered incorrect. In case if $v = u$, or if v and u are the numbers of two non-intersecting roads, your answer will also be considered incorrect.

The roads are numbered starting from 1 in the order in which they follow in the input.

Examples

input

```
3 1
1 0 0
0 -1 0
7 -93 0
```

output

```
YES
1
1 2
```

input

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

output

```
NO
```

input

```
2 3
3 4 5
5 6 7
```

output

```
YES
2
1 -1
2 -1
```

Note

Note that you do not have to minimize m, but it shouldn't be more than k.

In the first test all three roads intersect at point (0,0).

In the second test all three roads form a triangle and there is no way to place one sign so that it would stand on all three roads at once.

Problem F. Fygon

Input file:	`fygon.in`
Output file:	`fygon.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Frederick is a young programmer. He participates in all programming contests he can find and always uses his favorite programming language Fygon. Unfortunately, he often receives Time Limit Exceeded outcome, even when his algorithm is asymptotically optimal. That's because the Fygon interpreter is very slow. Nevertheless, Frederick likes Fygon so much, that he uses non-asymptotical optimizations to fit the solution into time limit. To make it easier, he asks you to write a program, which will be able to estimate the exact number of operations that his Fygon program makes.

For simplicity, we will assume that Fygon has only two statements. The first statement is `lag`. It substitutes almost any other statement. The second statement is a `for` loop:

```
for <variable> in range(<limit>):
    <body>
```

This means that `<variable>` iterates over values from 0 to `<limit>`-1. In Fygon `<variable>` is a lowercase letter from `a` to `z`, and `<limit>` is either already defined `<variable>` or a positive integer constant. The `<body>` of the loop is indented by four spaces and contains at least one statement.

The program receives the input in the variable `n`. This variable has special meaning and cannot be used as a loop variable.

Your task is to find the formula that calculates the number of performed `lag` operations by the given Fygon program, depending on the value of the variable `n`.

Input

The input file contains the Fygon program. No two loops use the same variable as iterators. Each variable used inside a `range` is either `n` or declared in some outer loop.

The program has at most 20 statements and at most 6 of them are loops. All integer constants are from 1 to 9.

Output

Output the formula for the number of performed `lag` operations depending on n. The length of the formula should be at most 100 characters (excluding spaces). The formula should correspond to the following grammar:

$$
\begin{aligned}
\langle\text{Expression}\rangle &::= \langle\text{Product}\rangle\ ((\text{'+'} \mid \text{'-'})\ \langle\text{Product}\rangle)^{*} \\
\langle\text{Product}\rangle &::= \langle\text{Value}\rangle\ (\text{'*'}\ \langle\text{Value}\rangle)^{*} \\
\langle\text{Value}\rangle &::= \text{'n'} \mid \langle\text{Number}\rangle \mid \text{'-'}\,\langle\text{Value}\rangle \mid \text{'('}\,\langle\text{Expression}\rangle\,\text{')'} \\
\langle\text{Number}\rangle &::= [\text{'0'}..\text{'9'}]^{+}\ \left(\text{'/'}\ [\text{'0'}..\text{'9'}]^{+}\right)^{?}
\end{aligned}
$$

Example

fygon.in	fygon.out

```
for i in range(n):
    for j in range(i):
        lag
for x in range(5):
    for y in range(n):
        for z in range(n):
            lag
    lag
```
	`1/2 * n * (n-1) + 5 * (n*n + 1)`

Problem F. Fygon 2.0

Input file:	`fygon20.in`	Time limit:	3 seconds
Output file:	`fygon20.out`	Memory limit:	512 megabytes

The new version of beloved programming language Fygon has been released! The brand new Fygon 2.0 still has only two statements. The first statement is `lag`. It substitutes almost any other statement. Second statement is a `for` loop:

```
for <variable> in range(<from>, <to>):
    <body>
```

- The `for` loop makes `<variable>` iterate from `<from>` to `<to>`, **both inclusive**.
- If `<from>` is greater than `<to>`, `<body>` is not executed at all.
- `<variable>` is a lowercase letter from `a` to `z`, except for `n`, which is a variable that is defined prior to the given code snippet.
- `<from>` and `<to>` can be equal to any variable defined in outer loop. In addition to that, `<from>` can be `1` and `<to>` can be `n`.
- The `<body>` of the loop is indented by four spaces and contains at least one statement.

If you are familiar with Fygon 1.0, you can notice that, in the spirit of the best programming practices, Fygon 2.0 is not backwards compatible, since the `range` function now requires two parameters.

The performance of the new version is significantly improved, so you can write more nested `for` loops. That is why we are no longer interested in the exact number of operations, but in the *asymptotic complexity* of the program instead. For simplicity, all `for` loops are nested in a single chain and there is exactly one `lag` statement that is inside all `for` loops. All loop variables are different and are not equal to `n`.

Let's define $f(n)$ as the number of `lag` operations exectuted by a given Fygon program as the function of n. For non-negative integer k and positive rational number C we say that $C \cdot n^k$ is the *asymptotic complexity* of the program if

$$\lim_{n\to\infty} \frac{f(n)}{C \cdot n^k} = 1.$$

Given a Fygon 2.0 program, find its asymptotic complexity.

Input

The first line of the input contains single integer m — the number of lines in Fygon 2.0 program. Next m lines contain the program itself. The program has at least 1 and at most 20 `for` statements. Each `for` statement contains either single nested `for` statement or `lag` statement.

Output

Output numbers k and C. C should be output in the form of irreducible fraction p/q, where p and q are coprime.

Example

fygon20.in	fygon20.out

```
4
for i in range(1, n):
    for j in range(1, i):
        for k in range(j, n):
            lag
```
	`3 1/3`

G. Segments

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

You are given n segments on the Ox-axis. You can drive a nail in any integer point on the Ox-axis line nail so, that all segments containing this point, are considered nailed down. If the nail passes through endpoint of some segment, this segment is considered to be nailed too. What is the smallest number of nails needed to nail all the segments down?

Input

The first line of the input contains single integer number n $(1 \le n \le 1000)$ — amount of segments. Following n lines contain descriptions of the segments. Each description is a pair of integer numbers — endpoints coordinates. All the coordinates don't exceed 10000 by absolute value. Segments can degenarate to points.

Output

The first line should contain one integer number — the smallest number of nails needed to nail all the segments down. The second line should contain coordinates of driven nails separated by space in any order. If the answer is not unique, output any.

Examples

input

```
2
0 2
2 5
```

output

```
1
2
```

input

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

output

```
3
7 10 3
```

H. Thor

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

Thor is getting used to the Earth. As a gift Loki gave him a smartphone. There are n applications on this phone. Thor is fascinated by this phone. He has only one minor issue: he can't count the number of unread notifications generated by those applications (maybe Loki put a curse on it so he can't).

q events are about to happen (in chronological order). They are of three types:

1. Application x generates a notification (this new notification is unread).
2. Thor reads all notifications generated so far by application x (he may re-read some notifications).
3. Thor reads the first t notifications generated by phone applications (notifications generated in first t events of the first type). It's guaranteed that there were at least t events of the first type before this event. Please note that he doesn't read first t unread notifications, he just reads the very first t notifications generated on his phone and he may re-read some of them in this operation.

Please help Thor and tell him the number of unread notifications after each event. You may assume that initially there are no notifications in the phone.

Input

The first line of input contains two integers n and q $(1 \le n, q \le 300\,000)$ — the number of applications and the number of events to happen.

The next q lines contain the events. The i-th of these lines starts with an integer $type_i$ — type of the i-th event. If $type_i = 1$ or $type_i = 2$ then it is followed by an integer x_i. Otherwise it is followed by an integer t_i $(1 \le type_i \le 3, 1 \le x_i \le n, 1 \le t_i \le q)$.

Output

Print the number of unread notifications after each event.

Examples

input

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

output

```
1
2
3
2
```

input

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

output

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

Note

In the first sample:

1. Application 3 generates a notification (there is 1 unread notification).
2. Application 1 generates a notification (there are 2 unread notifications).
3. Application 2 generates a notification (there are 3 unread notifications).
4. Thor reads the notification generated by application 3, there are 2 unread notifications left.

In the second sample test:

1. Application 2 generates a notification (there is 1 unread notification).
2. Application 4 generates a notification (there are 2 unread notifications).
3. Application 2 generates a notification (there are 3 unread notifications).
4. Thor reads first three notifications and since there are only three of them so far, there will be no unread notification left.
5. Application 3 generates a notification (there is 1 unread notification).
6. Application 3 generates a notification (there are 2 unread notifications).

I. Lucky Days

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

Bob and Alice are often participating in various programming competitions. Like many competitive programmers, Alice and Bob have good and bad days. They noticed, that their lucky and unlucky days are repeating with some period. For example, for Alice days $[l_a; r_a]$ are lucky, then there are some unlucky days: $[r_a + 1; l_a + t_a - 1]$, and then there are lucky days again: $[l_a + t_a; r_a + t_a]$ and so on. In other words, the day is lucky for Alice if it lies in the segment $[l_a + kt_a; r_a + kt_a]$ for some non-negative integer k.

The Bob's lucky day have similar structure, however the parameters of his sequence are different: l_b, r_b, t_b. So a day is a lucky for Bob if it lies in a segment $[l_b + kt_b; r_b + kt_b]$, for some non-negative integer k.

Alice and Bob want to participate in team competitions together and so they want to find out what is the largest possible number of consecutive days, which are lucky for both Alice and Bob.

Input

The first line contains three integers l_a, r_a, t_a ($0 \le l_a \le r_a \le t_a - 1, 2 \le t_a \le 10^9$) and describes Alice's lucky days.

The second line contains three integers l_b, r_b, t_b ($0 \le l_b \le r_b \le t_b - 1, 2 \le t_b \le 10^9$) and describes Bob's lucky days.

It is guaranteed that both Alice and Bob have some unlucky days.

Output

Print one integer: the maximum number of days in the row that are lucky for both Alice and Bob.

Examples

input

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

output

```
2
```

input

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

output

```
1
```

Note

The graphs below correspond to the two sample tests and show the lucky and unlucky days of Alice and Bob as well as the possible solutions for these tests.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Alice	+	+	+	−	−	+	+	+	−	−	+	+	+	−	−	···
Bob	−	+	+	+	−	−	+	+	+	−	−	+	+	+	−	···

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Alice	+	+	−	+	+	−	+	+	−	+	+	−	+	+	−	···
Bob	−	−	+	+	−	−	−	−	+	+	−	−	−	−	+	···

J. Messenger

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Each employee of the "Blake Techologies" company uses a special messaging app "Blake Messenger". All the stuff likes this app and uses it constantly. However, some important futures are missing. For example, many users want to be able to search through the message history. It was already announced that the new feature will appear in the nearest update, when developers faced some troubles that only you may help them to solve.

All the messages are represented as a strings consisting of only lowercase English letters. In order to reduce the network load strings are represented in the special compressed form. Compression algorithm works as follows: string is represented as a concatenation of n blocks, each block containing only equal characters. One block may be described as a pair (l_i, c_i), where l_i is the length of the i-th block and c_i is the corresponding letter. Thus, the string s may be written as the sequence of pairs $\langle (l_1, c_1), (l_2, c_2), \ldots, (l_n, c_n) \rangle$.

Your task is to write the program, that given two compressed string t and s finds all occurrences of s in t. Developers know that there may be many such occurrences, so they only ask you to find the **number** of them. Note that p is the starting position of some occurrence of s in t if and only if $t_p t_{p+1} \ldots t_{p+|s|-1} = s$, where t_i is the i-th character of string t.

Note that the way to represent the string in compressed form may not be unique. For example string "aaaa" may be given as $\langle (4, a) \rangle$, $\langle (3, a), (1, a) \rangle$, $\langle (2, a), (2, a) \rangle$...

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 200\,000$) — the number of blocks in the strings t and s, respectively.

The second line contains the descriptions of n parts of string t in the format "l_i-c_i" ($1 \le l_i \le 1\,000\,000$) — the length of the i-th part and the corresponding lowercase English letter.

The second line contains the descriptions of m parts of string s in the format "l_i-c_i" ($1 \le l_i \le 1\,000\,000$) — the length of the i-th part and the corresponding lowercase English letter.

Output

Print a single integer — the number of occurrences of s in t.

Examples

input

```
5 3
3-a 2-b 4-c 3-a 2-c
2-a 2-b 1-c
```

output

```
1
```

input

```
6 1
3-a 6-b 7-a 4-c 8-e 2-a
3-a
```

output

```
6
```

input

```
5 5
1-h 1-e 1-l 1-l 1-o
1-w 1-o 1-r 1-l 1-d
```

output

```
0
```

Note

In the first sample, t = "aaabbccccaaacc", and string s = "aabbc". The only occurrence of string s in string t starts at position $p = 2$.

In the second sample, t = "aaabbbbbbaaaaaaacccceeeeeeeeaa", and s = "aaa". The occurrences of s in t start at positions $p = 1$, $p = 10$, $p = 11$, $p = 12$, $p = 13$ and $p = 14$.

Problem K — limit 10 seconds

Klingon Warfare

War has broken out between two Klingon clans. But war between Klingons is not just any war; it must be both honorable and glorious. For honor, the two sides must be exactly matched. For glory, there must be as many participants as possible.

Each clan obeys a strict hierarchy: there is one leader of the entire clan. This leader may have zero or more direct subordinates who are ordered from eldest to youngest. Each subordinate in turn may have zero or more direct subordinates of his or her own, also ordered from eldest to youngest (and so on and so forth). By tradition, every clan warrior is younger than his or her superior. Furthermore, each individual clan warrior specializes in one of several distinct fighting styles.

The subclan commanded by a clan warrior consists of the warrior himself and all direct or indirect reports (i.e., subordinates, subordinates of subordinates, etc.). A pair of subclans are said to match exactly if two conditions are met. First, the leaders of each subclan must have the same fighting style and number of subordinates. Second, assuming that the direct subordinates of each leader are ordered by decreasing age, then the subclans commanded by the first direct subordinates must match exactly, the subclans commanded by the second direct subordinates must match exactly, and so on.

Each clan will select its participants in the war, consisting of a single warrior and his or her subclan. The two subclans chosen must match exactly, and must be as large as possible. How many warriors fight for each clan?

Input

Input begins with a line with one integer T ($1 \leq T \leq 50$) denoting the number of test cases. Each test case begins with a line with two integers M and N ($1 \leq M$, $N \leq 10000$) denoting the size of the two clans. Next follow M lines with an uppercase letter f_i and an integer s_i ($s_0 = -1$, $0 \leq s_i < i$, zero-indexed) denoting the fighting style and the superior respectively of warrior i of the first clan. Warrior 0 is always the clan leader (and has a "superior" of -1 to indicate this). Warriors are ordered from eldest to youngest. Next follow N lines with an uppercase letter f_j and an integer s_j (same bounds) denoting the fighting style and the superior respectively of warrior j of the second clan.

Output

For each test case, print out a line with a single integer equal to the maximum number of warriors that each clan may send to fight.

Sample Input	**Sample Output**

```
1
3 4
A -1
B 0
C 0
Z -1
A 0
B 1
C 1
```
	`3`

L. Palisection

time limit per test: 2 seconds

memory limit per test: 512 MB

input: standard input

output: standard output

In an English class Nick had nothing to do at all, and remembered about wonderful strings called palindromes. We should remind you that a string is called a palindrome if it can be read the same way both from left to right and from right to left. Here are examples of such strings: «`eye`», «`pop`», «`level`», «`aba`», «`deed`», «`racecar`», «`rotor`», «`madam`».

Nick started to look carefully for all palindromes in the text that they were reading in the class. For each occurrence of each palindrome in the text he wrote a pair — the position of the beginning and the position of the ending of this occurrence in the text. Nick called each occurrence of each palindrome he found in the text subpalindrome. When he found all the subpalindromes, he decided to find out how many different pairs among these subpalindromes cross. Two subpalindromes cross if they cover common positions in the text. No palindrome can cross itself.

Let's look at the actions, performed by Nick, by the example of text «`babb`». At first he wrote out all subpalindromes:

- «`b`» — $1..1$
- «`bab`» — $1..3$
- «`a`» — $2..2$
- «`b`» — $3..3$
- «`bb`» — $3..4$
- «`b`» — $4..4$

Then Nick counted the amount of different pairs among these subpalindromes that cross. These pairs were six:

1. $1..1$ cross with $1..3$
2. $1..3$ cross with $2..2$
3. $1..3$ cross with $3..3$
4. $1..3$ cross with $3..4$
5. $3..3$ cross with $3..4$
6. $3..4$ cross with $4..4$

Since it's very exhausting to perform all the described actions manually, Nick asked you to help him and write a program that can find out the amount of different subpalindrome pairs that cross. Two subpalindrome pairs are regarded as different if one of the pairs contains a subpalindrome that the other does not.

Input

The first input line contains integer n ($1 \le n \le 2 \cdot 10^6$) — length of the text. The following line contains n lower-case Latin letters (from `a` to `z`).

Output

In the only line output the amount of different pairs of two subpalindromes that cross each other. Output the answer modulo 51123987.

Examples

input

```
4
babb
```

output

```
6
```

input

```
2
aa
```

output

```
2
```