A. Removing Columns

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

You are given an $n \times m$ rectangular table consisting of lower case English letters. In one operation you can completely remove one column from the table. The remaining parts are combined forming a new table. For example, after removing the second column from the table

```
abcd
edfg
hijk
```

we obtain the table:

```
acd
efg
hjk
```

A table is called *good* if its rows are ordered from top to bottom lexicographically, i.e. each row is lexicographically no larger than the following one. Determine the minimum number of operations of removing a column needed to make a given table good.

Input

The first line contains two integers — n and m $(1 \leq n, m \leq 100)$.

Next n lines contain m small English letters each — the characters of the table.

Output

Print a single number — the minimum number of columns that you need to remove in order to make the table good.

Sample test(s)

input

```
1 10
codeforces
```

output

```
0
```

input

```
4 4
case
care
test
code
```

output

```
2
```

input

```
5 4
code
forc
esco
defo
rces
```

output

```
4
```

Note

In the first sample the table is already good.

In the second sample you may remove the first and third column.

In the third sample you have to remove all the columns (note that the table where all rows are empty is considered good by definition).

Let strings s and t have equal length. Then, s is *lexicographically larger* than t if they are not equal and the character following the largest common prefix of s and t (the prefix may be empty) in s is alphabetically larger than the corresponding character of t.

B. Tennis Game

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

Petya and Gena love playing table tennis. A single match is played according to the following rules: a match consists of multiple sets, each set consists of multiple serves. Each serve is won by one of the players, this player scores one point. As soon as one of the players scores t points, he wins the set; then the next set starts and scores of both players are being set to 0. As soon as one of the players wins the total of s sets, he wins the match and the match is over. Here s and t are some positive integer numbers.

To spice it up, Petya and Gena choose new numbers s and t before every match. Besides, for the sake of history they keep a record of each match: that is, for each serve they write down the winner. Serve winners are recorded in the chronological order. In a record the set is over as soon as one of the players scores t points and the match is over as soon as one of the players wins s sets.

Petya and Gena have found a record of an old match. Unfortunately, the sequence of serves in the record isn't divided into sets and numbers s and t for the given match are also lost. The players now wonder what values of s and t might be. Can you determine all the possible options?

Input

The first line contains a single integer n — the length of the sequence of games ($1 \le n \le 10^5$).

The second line contains n space-separated integers a_i. If $a_i = 1$, then the i-th serve was won by Petya, if $a_i = 2$, then the i-th serve was won by Gena.

It is not guaranteed that at least one option for numbers s and t corresponds to the given record.

Output

In the first line print a single number k — the number of options for numbers s and t.

In each of the following k lines print two integers s_i and t_i — the option for numbers s and t. Print the options in the order of increasing s_i, and for equal s_i — in the order of increasing t_i.

Sample test(s)

input
```
5
1 2 1 2 1
```
output
```
2
1 3
3 1
```

input
```
4
1 1 1 1
```
output
```
3
1 4
2 2
4 1
```

input
```
4
1 2 1 2
```
output
```
0
```

input
```
8
2 1 2 1 1 1 1 1
```
output
```
3
1 6
2 3
6 1
```

C. Distributing Parts

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

You are an assistant director in a new musical play. The play consists of n musical parts, each part must be performed by exactly one actor. After the casting the director chose m actors who can take part in the play. Your task is to assign the parts to actors. However, there are several limitations.

First, each actor has a certain voice range and there are some parts that he cannot sing. Formally, there are two integers for each actor, c_i and d_i ($c_i \le d_i$) — the pitch of the lowest and the highest note that the actor can sing. There also are two integers for each part — a_j and b_j ($a_j \le b_j$) — the pitch of the lowest and the highest notes that are present in the part. The i-th actor can perform the j-th part if and only if $c_i \le a_j \le b_j \le d_i$, i.e. each note of the part is in the actor's voice range.

According to the contract, the i-th actor can perform at most k_i parts. Besides, you are allowed not to give any part to some actors (then they take part in crowd scenes).

The rehearsal starts in two hours and you need to do the assignment quickly!

Input

The first line contains a single integer n — the number of parts in the play ($1 \le n \le 10^5$).

Next n lines contain two space-separated integers each, a_j and b_j — the range of notes for the j-th part ($1 \le a_j \le b_j \le 10^9$).

The next line contains a single integer m — the number of actors ($1 \le m \le 10^5$).

Next m lines contain three space-separated integers each, c_i, d_i and k_i — the range of the i-th actor and the number of parts that he can perform ($1 \le c_i \le d_i \le 10^9$, $1 \le k_i \le 10^9$).

Output

If there is an assignment that meets all the criteria aboce, print a single word "`YES`" (without the quotes) in the first line.

In the next line print n space-separated integers. The i-th integer should be the number of the actor who should perform the i-th part. If there are multiple correct assignments, print any of them.

If there is no correct assignment, print a single word "`NO`" (without the quotes).

Sample test(s)

input

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

output

```
YES
1 1 2
```

input

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

output

```
NO
```

D. Gears

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

There are two polygons on the plane, A and B. Polygon A rotates around point P, and polygon B rotates around point Q. Each polygon rotates with the constant rotational speed in the clockwise direction around its point, the rotational speed values of the polygons' rotation are equal.

Your task is to determine if there will be a *collision* between polygons. A *collision* is a situation when the polygons have at least one common point.

It is guaranteed that at the moment 0 the polygons A and B do not intersect and no polygon is fully contained inside another one.

Note that:

- the polygons are not necessarily convex;
- points P and Q can be located on the border of or outside their polygons.

Input

The first line contains space-separated coordinates of point P.

The second line contains a single integer n $(3 \le n \le 1000)$ — the number of vertices of polygon A.

Each of the next n lines contains two space-separated integers — the coordinates of the corresponding vertex of polygon A.

The next line is empty.

Then follow space-separated coordinates of point Q.

The next line contains a single integer m $(3 \le m \le 1000)$ — the number of vertices of polygon B. Next m lines contain the coordinates of the vertices of the polygon B.

The vertices of both polygons are listed in the counterclockwise order. Coordinates of all points are integers, their absolute values don't exceed 10^4.

Output

Print "`YES`", if the collision takes place and "`NO`" otherwise (don't print the quotes).

Sample test(s)

input

```
1 0
4
0 0
1 0
1 5
0 5

9 0
4
9 0
9 -5
10 -5
10 0
```

output

```
YES
```

input

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

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

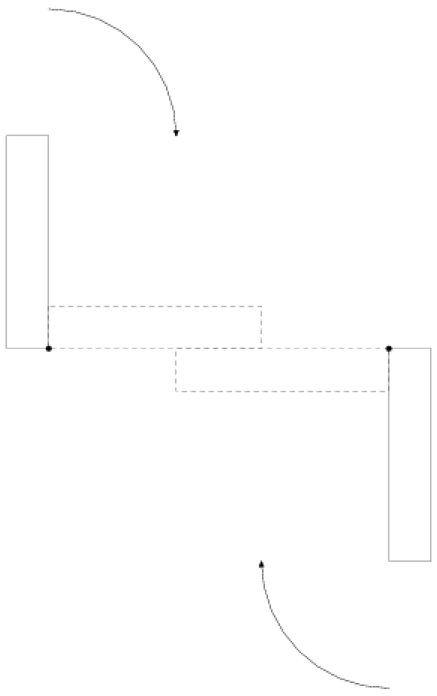
output

```
NO
```

Note

A *polygon* is a closed polyline that doesn't intersect itself and doesn't touch itself.

Picture to the first sample:



E. Subsequences Return

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

Assume that $s_k(n)$ equals the sum of digits of number n in the k-based notation. For example, $s_2(5) = s_2(101_2) = 1 + 0 + 1 = 2$, $s_3(14) = s_3(112_3) = 1 + 1 + 2 = 4$.

The sequence of integers $a_0, ..., a_{n-1}$ is defined as $a_j = s_k(j) \bmod k$. Your task is to calculate the number of distinct *subsequences* of sequence $a_0, ..., a_{n-1}$. Calculate the answer modulo $10^9 + 7$.

Sequence $a_1, ..., a_k$ is called to be a *subsequence* of sequence $b_1, ..., b_l$, if there is a sequence of indices $1 \le i_1 < ... < i_k \le l$, such that $a_1 = b_{i_1}$, ..., $a_k = b_{i_k}$. In particular, an empty sequence (i.e. the sequence consisting of zero elements) is a subsequence of any sequence.

Input

The first line contains two space-separated numbers n and k ($1 \le n \le 10^{18}$, $2 \le k \le 30$).

Output

In a single line print the answer to the problem modulo $10^9 + 7$.

Sample test(s)

input

```
4 2
```

output

```
11
```

input

```
7 7
```

output

```
128
```

Note

In the first sample the sequence a_i looks as follows: $(0, 1, 1, 0)$. All the possible subsequences are:

$$(), (0), (0, 0), (0, 1), (0, 1, 0), (0, 1, 1), (0, 1, 1, 0), (1), (1, 0), (1, 1), (1, 1, 0).$$

In the second sample the sequence a_i looks as follows: $(0, 1, 2, 3, 4, 5, 6)$. The subsequences of this sequence are exactly all increasing sequences formed from numbers from 0 to 6. It is easy to see that there are $2^7 = 128$ such sequences.