A. Bark to Unlock

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

As technologies develop, manufacturers are making the process of unlocking a phone as user-friendly as possible. To unlock its new phone, Arkady's pet dog Mu-mu has to bark the password once. The phone represents a password as a string of two lowercase English letters.

Mu-mu's enemy Kashtanka wants to unlock Mu-mu's phone to steal some sensible information, but it can only bark n distinct words, each of which can be represented as a string of two lowercase English letters. Kashtanka wants to bark several words (not necessarily distinct) one after another to pronounce a string containing the password as a substring. Tell if it's possible to unlock the phone in this way, or not.

Input

The first line contains two lowercase English letters — the password on the phone.

The second line contains single integer n $(1 \le n \le 100)$ — the number of words Kashtanka knows.

The next n lines contain two lowercase English letters each, representing the words Kashtanka knows. The words are guaranteed to be distinct.

Output

Print "`YES`" if Kashtanka can bark several words in a line forming a string containing the password, and "`NO`" otherwise.

You can print each letter in arbitrary case (upper or lower).

Examples

input

```
ya
4
ah
oy
to
ha
```

output

```
YES
```

input

```
hp
2
ht
tp
```

output

```
NO
```

input

```
ah
1
ha
```

output

```
YES
```

Note

In the first example the password is "`ya`", and Kashtanka can bark "`oy`" and then "`ah`", and then "`ha`" to form the string "`oyahha`" which contains the password. So, the answer is "`YES`".

In the second example Kashtanka can't produce a string containing password as a substring. Note that it can bark "`ht`" and then "`tp`" producing "`http`", but it doesn't contain the password "`hp`" as a substring.

In the third example the string "`hahahaha`" contains "`ah`" as a substring.

B. Race Against Time

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

Have you ever tried to explain to the coordinator, why it is eight hours to the contest and not a single problem has been prepared yet? Misha had. And this time he has a *really* strong excuse: he faced a space-time paradox! Space and time replaced each other.

The entire universe turned into an enormous clock face with three hands — hour, minute, and second. Time froze, and clocks now show the time h hours, m minutes, s seconds.

Last time Misha talked with the coordinator at t_1 o'clock, so now he stands on the number t_1 on the clock face. The contest should be ready by t_2 o'clock. In the terms of paradox it means that Misha has to go to number t_2 somehow. Note that he doesn't have to move forward only: in these circumstances time has no direction.

Clock hands are very long, and Misha cannot get round them. He also cannot step over as it leads to the collapse of space-time. That is, if hour clock points 12 and Misha stands at 11 then he cannot move to 1 along the top arc. He has to follow all the way round the clock center (of course, if there are no other hands on his way).

Given the hands' positions, t_1, and t_2, find if Misha can prepare the contest on time (or should we say *on space*?). That is, find if he can move from t_1 to t_2 by the clock face.

Input

Five integers h, m, s, t_1, t_2 $(1 \le h \le 12,\ 0 \le m, s \le 59,\ 1 \le t_1, t_2 \le 12,\ t_1 \ne t_2)$.

Misha's position and the target time do not coincide with the position of any hand.

Output

Print "YES" (quotes for clarity), if Misha can prepare the contest on time, and "NO" otherwise.

You can print each character either upper- or lowercase ("YeS" and "yes" are valid when the answer is "YES").

Examples

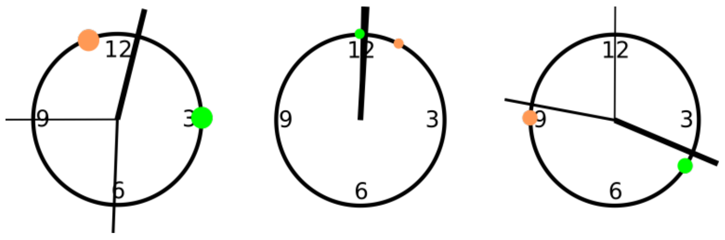
input
```
12 30 45 3 11
```
output
```
NO
```

input
```
12 0 1 12 1
```
output
```
YES
```

input
```
3 47 0 4 9
```
output
```
YES
```

Note

The three examples are shown on the pictures below from left to right. The starting position of Misha is shown with green, the ending position is shown with pink. Note that the positions of the hands on the pictures are not exact, but are close to the exact and the answer is the same.



C. Qualification Rounds

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

Snark and Philip are preparing the problemset for the upcoming pre-qualification round for semi-quarter-finals. They have a bank of n problems, and they want to select any non-empty subset of it as a problemset.

k experienced teams are participating in the contest. Some of these teams already know some of the problems. To make the contest interesting for them, each of the teams should know at most half of the selected problems.

Determine if Snark and Philip can make an interesting problemset!

Input

The first line contains two integers n, k $(1 \le n \le 10^5, 1 \le k \le 4)$ — the number of problems and the number of experienced teams.

Each of the next n lines contains k integers, each equal to 0 or 1. The j-th number in the i-th line is 1 if j-th team knows i-th problem and 0 otherwise.

Output

Print "YES" (quotes for clarity), if it is possible to make an interesting problemset, and "NO" otherwise.

You can print each character either upper- or lowercase ("YeS" and "yes" are valid when the answer is "YES").

Examples

input

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

output

```
NO
```

input

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

output

```
YES
```

Note

In the first example you can't make any interesting problemset, because the first team knows all problems.

In the second example you can choose the first and the third problems.

D. Huge Strings

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

You are given n strings $s_1, s_2, \ldots, s_n$ consisting of characters 0 and 1. m operations are performed, on each of them you concatenate two existing strings into a new one. On the i-th operation the concatenation $s_{a_i}s_{b_i}$ is saved into a new string s_{n+i} (the operations are numbered starting from 1). After each operation you need to find the maximum positive integer k such that all possible strings consisting of 0 and 1 of length k (there are 2^k such strings) are substrings of the new string. If there is no such k, print 0.

Input

The first line contains single integer n $(1 \le n \le 100)$ — the number of strings. The next n lines contain strings $s_1, s_2, \ldots, s_n$ $(1 \le |s_i| \le 100)$, one per line. The total length of strings is not greater than 100.

The next line contains single integer m $(1 \le m \le 100)$ — the number of operations. m lines follow, each of them contains two integers a_i abd b_i $(1 \le a_i, b_i \le n + i - 1)$ — the number of strings that are concatenated to form s_{n+i}.

Output

Print m lines, each should contain one integer — the answer to the question after the corresponding operation.

Example

input

```
5
01
10
101
11111
0
3
1 2
6 5
4 4
```

output

```
1
2
0
```

Note

On the first operation, a new string "0110" is created. For $k = 1$ the two possible binary strings of length k are "0" and "1", they are substrings of the new string. For $k = 2$ and greater there exist strings of length k that do not appear in this string (for $k = 2$ such string is "00"). So the answer is 1.

On the second operation the string "01100" is created. Now all strings of length $k = 2$ are present.

On the third operation the string "1111111111" is created. There is no zero, so the answer is 0.

E. Policeman and a Tree

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a tree (a connected non-oriented graph without cycles) with vertices numbered from 1 to n, and the length of the i-th edge is w_i. In the vertex s there is a policeman, in the vertices $x_1, x_2, \ldots, x_m$ ($x_j \neq s$) m criminals are located.

The policeman can walk along the edges with speed 1, the criminals can move with arbitrary large speed. If a criminal at some moment is at the same point as the policeman, he instantly gets caught by the policeman. Determine the time needed for the policeman to catch all criminals, assuming everybody behaves optimally (i.e. the criminals maximize that time, the policeman minimizes that time). Everybody knows positions of everybody else at any moment of time.

Input

The first line contains single integer n ($1 \leq n \leq 50$) — the number of vertices in the tree. The next n - 1 lines contain three integers each: u_i, v_i, w_i ($1 \leq u_i, v_i \leq n$, $1 \leq w_i \leq 50$) denoting edges and their lengths. It is guaranteed that the given graph is a tree.

The next line contains single integer s ($1 \leq s \leq n$) — the number of vertex where the policeman starts.

The next line contains single integer m ($1 \leq m \leq 50$) — the number of criminals. The next line contains m integers $x_1, x_2, \ldots, x_m$ ($1 \leq x_j \leq n$, $x_j \neq s$) — the number of vertices where the criminals are located. x_j are not necessarily distinct.

Output

If the policeman can't catch criminals, print single line "`Terrorists win`" (without quotes).

Otherwise, print single integer — the time needed to catch all criminals.

Examples

input

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

output

```
8
```

input

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

output

```
21
```

Note

In the first example one of the optimal scenarios is the following. The criminal number 2 moves to vertex 3, the criminal 4 — to vertex 4. The policeman goes to vertex 4 and catches two criminals. After that the criminal number 1 moves to the vertex 2. The policeman goes to vertex 3 and catches criminal 2, then goes to the vertex 2 and catches the remaining criminal.

F. Yet Another Minimization Problem

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

You are given an array of n integers $a_1 \dots a_n$. The cost of a subsegment is the number of unordered pairs of distinct indices within the subsegment that contain equal elements. Split the given array into k non-intersecting non-empty subsegments so that the sum of their costs is minimum possible. Each element should be present in exactly one subsegment.

Input

The first line contains two integers n and k $(2 \le n \le 10^5, 2 \le k \le min\ (n, 20))$ — the length of the array and the number of segments you need to split the array into.

The next line contains n integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le n)$ — the elements of the array.

Output

Print single integer: the minimum possible total cost of resulting subsegments.

Examples

input

```
7 3
1 1 3 3 3 2 1
```

output

```
1
```

input

```
10 2
1 2 1 2 1 2 1 2 1 2
```

output

```
8
```

input

```
13 3
1 2 2 2 1 2 1 1 1 2 2 1 1
```

output

```
9
```

Note

In the first example it's optimal to split the sequence into the following three subsegments: $[1]$, $[1, 3]$, $[3, 3, 2, 1]$. The costs are 0, 0 and 1, thus the answer is 1.

In the second example it's optimal to split the sequence in two equal halves. The cost for each half is 4.

In the third example it's optimal to split the sequence in the following way: $[1, 2, 2, 2, 1]$, $[2, 1, 1, 1, 2]$, $[2, 1, 1]$. The costs are 4, 4, 1.

G. El Toll Caves

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

The prehistoric caves of El Toll are located in Moià (Barcelona). You have heard that there is a treasure hidden in one of n possible spots in the caves. You assume that each of the spots has probability $1 / n$ to contain a treasure.

You cannot get into the caves yourself, so you have constructed a robot that can search the caves for treasure. Each day you can instruct the robot to visit exactly k distinct spots in the caves. If none of these spots contain treasure, then the robot will obviously return with empty hands. However, the caves are dark, and the robot may miss the treasure even when visiting the right spot. Formally, if one of the visited spots does contain a treasure, the robot will obtain it with probability $1 / 2$, otherwise it will return empty. Each time the robot searches the spot with the treasure, his success probability is independent of all previous tries (that is, the probability to miss the treasure after searching the right spot x times is $1 / 2^x$).

What is the expected number of days it will take to obtain the treasure if you choose optimal scheduling for the robot? Output the answer as a rational number modulo $10^9 + 7$. Formally, let the answer be an irreducible fraction P / Q, then you have to output $P \cdot Q^{-1} \bmod (10^9 + 7)$. It is guaranteed that Q is not divisible by $10^9 + 7$.

Input

The first line contains the number of test cases T ($1 \le T \le 1000$).

Each of the next T lines contains two integers n and k ($1 \le k \le n \le 5 \cdot 10^8$).

Output

For each test case output the answer in a separate line.

Example

input

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

output

```
2
500000007
777777786
```

Note

In the first case the robot will repeatedly search in the only spot. The expected number of days in this case is 2. Note that in spite of the fact that we know the treasure spot from the start, the robot still has to search there until he succesfully recovers the treasure.

In the second case the answer can be shown to be equal to $7 / 2$ if we search the two spots alternatively. In the third case the answer is $25 / 9$.