Codeforces Round #212 (Div. 2)

A. Two Semiknights Meet

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

A boy Petya loves chess very much. He even came up with a chess piece of his own, a semiknight. The semiknight can move in any of these four directions: 2 squares forward and 2 squares to the right, 2 squares forward and 2 squares to the left, 2 squares backward and 2 to the right and 2 squares backward and 2 to the left. Naturally, the semiknight cannot move beyond the limits of the chessboard.

Petya put two semiknights on a standard chessboard. Petya simultaneously moves with both semiknights. The squares are rather large, so after some move the semiknights can meet, that is, they can end up in the same square. After the meeting the semiknights can move on, so it is possible that they meet again. Petya wonders if there is such sequence of moves when the semiknights meet. Petya considers some squares bad. That is, they do not suit for the meeting. The semiknights can move through these squares but their meetings in these squares don't count.

Petya prepared multiple chess boards. Help Petya find out whether the semiknights can meet on some good square for each board.

Please see the test case analysis.

Input

The first line contains number t ($1 \leq t \leq 50$) — the number of boards. Each board is described by a matrix of characters, consisting of 8 rows and 8 columns. The matrix consists of characters "`.`", "`#`", "`K`", representing an empty good square, a bad square and the semiknight's position, correspondingly. It is guaranteed that matrix contains exactly 2 semiknights. The semiknight's squares are considered good for the meeting. The tests are separated by empty line.

Output

For each test, print on a single line the answer to the problem: "`YES`", if the semiknights can meet and "`NO`" otherwise.

Sample test(s)

input

```
2
........
........
......#.
K..##..#
.......#
...##..#
......#.
K.......

........
........
..#.....
..#..#..
..####..
...##...
........
....K#K#
```

output

```
YES
NO
```

Note

Consider the first board from the sample. We will assume the rows and columns of the matrix to be numbered 1 through 8 from top to bottom and from left to right, correspondingly. The knights can meet, for example, in square $(2, 7)$. The semiknight from square $(4, 1)$ goes to square $(2, 3)$ and the semiknight goes from square $(8, 1)$ to square $(6, 3)$. Then both semiknights go to $(4, 5)$ but this square is bad, so they move together to square $(2, 7)$.

On the second board the semiknights will never meet.

B. Petya and Staircases

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

Little boy Petya loves stairs very much. But he is bored from simple going up and down them — he loves jumping over several stairs at a time. As he stands on some stair, he can either jump to the next one or jump over one or two stairs at a time. But some stairs are too dirty and Petya doesn't want to step on them.

Now Petya is on the first stair of the staircase, consisting of n stairs. He also knows the numbers of the dirty stairs of this staircase. Help Petya find out if he can jump through the entire staircase and reach the last stair number n without touching a dirty stair once.

One has to note that anyway Petya should step on the first and last stairs, so if the first or the last stair is dirty, then Petya cannot choose a path with clean steps only.

Input

The first line contains two integers n and m ($1 \le n \le 10^9$, $0 \le m \le 3000$) — the number of stairs in the staircase and the number of dirty stairs, correspondingly. The second line contains m **different** space-separated integers $d_1, d_2, ..., d_m$ ($1 \le d_i \le n$) — the numbers of the dirty stairs (in an arbitrary order).

Output

Print "YES" if Petya can reach stair number n, stepping only on the clean stairs. Otherwise print "NO".

Sample test(s)

input
10 5 2 4 8 3 6
output
NO

input
10 5 2 4 5 7 9
output
YES

C. Insertion Sort

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

Petya is a beginner programmer. He has already mastered the basics of the C++ language and moved on to learning algorithms. The first algorithm he encountered was insertion sort. Petya has already written the code that implements this algorithm and sorts the given integer zero-indexed array a of size n in the non-decreasing order.

```
for (int i = 1; i < n; i = i + 1)
{
   int j = i;
   while (j > 0 && a[j] < a[j - 1])
   {
      swap(a[j], a[j - 1]); // swap elements a[j] and a[j - 1]
      j = j - 1;
   }
}
```

Petya uses this algorithm only for sorting of arrays that are permutations of numbers from 0 to n - 1. He has already chosen the permutation he wants to sort but he first decided to swap some two of its elements. Petya wants to choose these elements in such a way that the number of times the sorting executes function `swap`, was minimum. Help Petya find out the number of ways in which he can make the swap and fulfill this requirement.

It is guaranteed that it's always possible to swap two elements of the input permutation in such a way that the number of `swap` function calls decreases.

Input

The first line contains a single integer n ($2 \le n \le 5000$) — the length of the permutation. The second line contains n different integers from 0 to n - 1, inclusive — the actual permutation.

Output

Print two integers: the minimum number of times the `swap` function is executed and the number of such pairs (i, j) that swapping the elements of the input permutation with indexes i and j leads to the minimum number of the executions.

Sample test(s)

input
5 4 0 3 1 2
output
3 2

input
5 1 2 3 4 0
output
3 4

Note

In the first sample the appropriate pairs are $(0, 3)$ and $(0, 4)$.

In the second sample the appropriate pairs are $(0, 4)$, $(1, 4)$, $(2, 4)$ and $(3, 4)$.

D. Fools and Foolproof Roads

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

You must have heard all about the Foolland on your Geography lessons. Specifically, you must know that federal structure of this country has been the same for many centuries. The country consists of n cities, some pairs of cities are connected by bidirectional roads, each road is described by its length l_i.

The fools lived in their land joyfully, but a recent revolution changed the king. Now the king is Vasily the Bear. Vasily divided the country cities into regions, so that any two cities of the same region have a path along the roads between them and any two cities of different regions don't have such path. Then Vasily decided to upgrade the road network and construct exactly p new roads in the country. Constructing a road goes like this:

1. We choose a pair of **distinct** cities u, v that will be connected by a new road (at that, it is possible that there already is a road between these cities).
2. We define the length of the new road: if cities u, v belong to distinct regions, then the length is calculated as $min(10^9, S+1)$ (S — the total length of all roads that exist in the linked regions), otherwise we assume that the length equals 1000.
3. We build a road of the specified length between the chosen cities. If the new road connects two distinct regions, after construction of the road these regions are combined into one new region.

Vasily wants the road constructing process to result in the country that consists exactly of q regions. Your task is to come up with such road constructing plan for Vasily that it meets the requirement and minimizes the total length of the built roads.

Input

The first line contains four integers n ($1 \le n \le 10^5$), m ($0 \le m \le 10^5$), p ($0 \le p \le 10^5$), q ($1 \le q \le n$) — the number of cities in the Foolland, the number of existing roads, the number of roads that are planned to construct and the required number of regions.

Next m lines describe the roads that exist by the moment upgrading of the roads begun. Each of these lines contains three integers x_i, y_i, l_i: x_i, y_i — the numbers of the cities connected by this road ($1 \le x_i, y_i \le n, x_i \ne y_i$), l_i — length of the road ($1 \le l_i \le 10^9$). Note that one pair of cities can be connected with multiple roads.

Output

If constructing the roads in the required way is impossible, print a single string "`NO`" (without the quotes). Otherwise, in the first line print word "`YES`" (without the quotes), and in the next p lines print the road construction plan. Each line of the plan must consist of two distinct integers, giving the numbers of the cities connected by a road. The road must occur in the plan in the order they need to be constructed. If there are multiple optimal solutions, you can print any of them.

Sample test(s)

input
9 6 2 2 1 2 2 3 2 1 4 6 20 1 3 8 7 8 3 5 7 2
output
YES 9 5 1 9

input
2 0 1 2
output
NO

input
2 0 0 2
output
YES

Note

Consider the first sample. Before the reform the Foolland consists of four regions. The first region includes cities $1, 2, 3$, the second region has cities 4 and 6, the third region has cities $5, 7, 8$, the fourth region has city 9. The total length of the roads in these cities is $11, 20, 5$ and 0, correspondingly. According to the plan, we first build the road of length 6 between cities 5 and 9, then the road of length 23 between cities 1 and 9. Thus, the total length of the built roads equals 29.

E. Petya and Pipes

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

A little boy Petya dreams of growing up and becoming the Head Berland Plumber. He is thinking of the problems he will have to solve in the future. Unfortunately, Petya is too inexperienced, so you are about to solve one of such problems for Petya, the one he's the most interested in.

The Berland capital has n water tanks numbered from 1 to n. These tanks are connected by unidirectional pipes in some manner. Any pair of water tanks is connected by at most one pipe in each direction. Each pipe has a strictly positive integer width. Width determines the number of liters of water per a unit of time this pipe can transport. The water goes to the city from the main water tank (its number is 1). The water must go through some pipe path and get to the sewer tank with cleaning system (its number is n).

Petya wants to increase the width of some subset of pipes by at most k units in total so that the width of each pipe remains integer. Help him determine the maximum amount of water that can be transmitted per a unit of time from the main tank to the sewer tank after such operation is completed.

Input

The first line contains two space-separated integers n and k ($2 \le n \le 50$, $0 \le k \le 1000$). Then follow n lines, each line contains n integers separated by single spaces. The $i + 1$-th row and j-th column contain number c_{ij} — the width of the pipe that goes from tank i to tank j ($0 \le c_{ij} \le 10^6$, $c_{ii} = 0$). If $c_{ij} = 0$, then there is no pipe from tank i to tank j.

Output

Print a single integer — the maximum amount of water that can be transmitted from the main tank to the sewer tank per a unit of time.

Sample test(s)

input

```
5 7
0 1 0 2 0
0 0 4 10 0
0 0 0 0 5
0 0 0 0 10
0 0 0 0 0
```

output

```
10
```

input

```
5 10
0 1 0 0 0
0 0 2 0 0
0 0 0 3 0
0 0 0 0 4
100 0 0 0 0
```

output

```
5
```

Note

In the first test Petya can increase width of the pipe that goes from the 1st to the 2nd water tank by 7 units.

In the second test Petya can increase width of the pipe that goes from the 1st to the 2nd water tank by 4 units, from the 2nd to the 3rd water tank by 3 units, from the 3rd to the 4th water tank by 2 units and from the 4th to 5th water tank by 1 unit.

Codeforces (c) Copyright 2010-2015 Mike Mirzayanov
The only programming contests Web 2.0 platform