Growing Trees

Egon takes care of a garden with N apple trees. His responsibilities include two types of tasks: fertilizing the trees and computing some statistics about them.

For fertilizing the trees he has several bottles of *MegaBoostFertilizer*, which, when treated on trees, causes them to grow one centimeter up instantly. Every bottle has a limited capacity c_i, which determines the number of trees it can be applied to. Moreover, for each bottle there is a minimal height h_i of trees, which can be treated with it. Since Egon wants to have all his trees as big as possible, he always applies the fertilizer to the c_i smallest trees chosen from the trees that are at least h_i centimeters high.

When Egon computes statistics about trees he has to determine the number of trees whose height is in some given interval. Egon is quite busy working in the garden, so he asked you to write a program, that given the list of his tasks, computes the statistics for him.

Input

The first line of the standard input contains two integers N and M denoting the number of trees in Egon's garden and the number of his tasks. The second line contains a sequence of N integers from the range $[1, N]$ describing the initial heights of the trees in centimeters. The following M lines describe the tasks in chronological order. Each of those lines begins with a character t_i ($t_i =$ `F` or $t_i =$ `C`), which describes the type of the task.

If $t_i =$ `F` then two integers c_i and h_i follow. Such a line means that Egon applies a bottle of *MegaBoostFertilizer* to the c_i smallest trees among those trees that are at least h_i centimeters high. When there are less than c_i trees of sufficient height, he applies the fertilizer to all such trees and discards the bottle with some fertilizer remaining.

If $t_i =$ `C` then two integers min_i and max_i follow. They denote that Egon has to compute the number of trees whose height H is between min_i and max_i centimeters ($min_i \leq H \leq max_i$).

Output

For every task of type `C`, output one line containing the number of apple trees that have the required height. The order of the results should conform to the order of type `C` tasks in the input.

Constraints

$1 \leq N, M \leq 100\,000$;
$1 \leq c_i \leq N$, $0 \leq h_i \leq 1\,000\,000\,000$;
$1 \leq min_i \leq max_i \leq 1\,000\,000\,000$.
In test cases worth 40 points $1 \leq N \leq 7\,000$ and the number of type `F` tasks is at most $7\,000$.

Example

Input	Output
5 7 1 3 2 5 2 F 2 1 C 3 6 F 2 3 C 6 8 F 2 1 F 2 2 C 3 5	3 0 5

Ice Cream

Rasmus and his friends are on vacation in Italy. Since they are suffering from the heat, they decide to buy some ice cream. There are N flavors of ice cream available; flavors are numbered from 1 to N. However, some pairings of flavors should be avoided; otherwise, the taste would be unpleasant. Rasmus wants to know how many ways there are to choose three *different* flavors without any such *impossible pairing*. The order of flavors is not taken into account.

Input

The first line of input contains two non-negative integers N and M, the number of flavors and the number of impossible pairings. Each of the M following lines describes an impossible pairing and contains two different flavor numbers. No impossible pairing will appear twice.

Output

The first and only line of output must contain a single integer: the number of possible choices.

Constraints

$1 \leq N \leq 200$.
$0 \leq M \leq 10\,000$.

Example

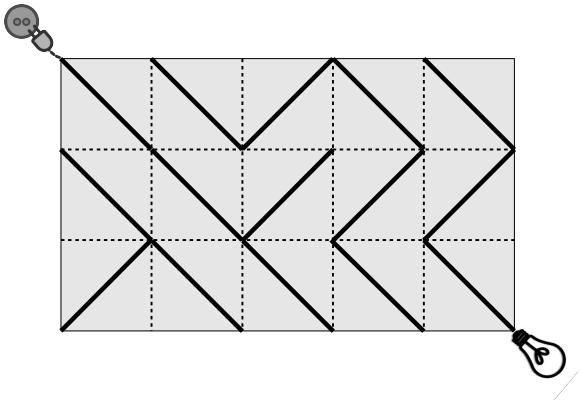
Input	Output
5 3 1 2 3 4 1 3	3

There are 5 flavors and 3 impossible pairings. Flavor 1 should be combined with neither flavor 2 nor flavor 3, and flavor 3 also should not be chosen together with flavor 4. Only 3 ways to choose three different flavors remain: (1 4 5), (2 3 5), and (2 4 5).

Switch the Lamp On

Casper is designing an electronic circuit on a $N \times M$ rectangular grid plate. There are $N \times M$ square tiles that are aligned to the grid on the plate. Two (out of four) opposite corners of each tile are connected by a wire.

A power source is connected to the top left corner of the plate. A lamp is connected to the bottom right corner of the plate. The lamp is on only if there is a path of wires connecting power source to lamp. In order to switch the lamp on, any number of tiles can be turned by 90° (in both directions).



In the picture above the lamp is off. If any one of the tiles in the second column from the right is turned by 90°, power source and lamp get connected, and the lamp is on.

Write a program to find out the minimal number of tiles that have to be turned by 90° to switch the lamp on.

Input

The first line of input contains two integer numbers N and M, the dimensions of the plate. In each of the following N lines there are M symbols – either `\` or `/` – which indicate the direction of the wire connecting the opposite vertices of the corresponding tile.

Output

There must be exactly one line of output. If it is possible to switch the lamp on, this line must contain only one integer number: the minimal number of tiles that have to be turned to switch on the lamp. If it is not possible, output the string: `NO SOLUTION`

Constraints

$1 \leq N, M \leq 500$.
In test cases worth 40 points, $1 \leq N \leq 4$ and $1 \leq M \leq 5$.

Example

Input	Output
3 5 \\/\\ \\/// /\\\\	1

The example input corresponds to the picture.

Treasures and Vikings

You have a treasure map that is arranged into a $N \times M$ grid. A grid square may be either sea or part of an island. In addition, the map shows the treasure and an enemy Viking ship that occupies one (sea) square. Finally, for convenience you have also drawn your own position.

Now you must set up a fixed route to get the treasure. The route must start at your position, end at the treasure, and consist of a sequence of moves. In each move, you can go only to an (horizontally or vertically) adjacent square that is not part of an island. But beware: The Viking ship might follow you, using the same kind of moves! After each of your moves according to your route, the Viking ship may move or not. Your move and the Vikings' reaction together is called a *round*.

After every round, the following checks are made:

- If you are in line with the Viking ship (you are in the same vertical or horizontal line as the Viking ship with only sea between the Viking ship and you), you are dead.
- If you aren't dead and at the treasure-spot, you get the treasure.

Write a program that decides whether it is possible to set up a **fixed** route **in advance** such that you can get the treasure by following this route and will not get killed by the Vikings – no matter how the Viking ship moves.

Input

The first line of input contains two integers N and M, the dimensions of the map. Each of the following N lines contain M characters. Each character describes a square in the map, and is either `.` (sea), `I` (part of an island), `V` (the Viking ship), `Y` (your position), or `T` (the treasure). Each of `V`, `Y`, and `T` will occur exactly once.

Output

The only line of the output must contain the string `YES`, if it is possible to set up a route to get the treasure, or `NO` otherwise.

Constraints

$1 \leq N, M \leq 700$.
In test cases worth 50 points, $1 \leq N, M \leq 200$.

Example

Input	Output
5 7 Y.....V ..I.... ..IIIII ....T...	YES
5 7 Y....V. ..I.... ..IIIII ....T...	NO
2 3 .YT VII	NO

In the first example, the following route will let you get the treasure:
Down, Down, Down, Right, Right, Right, Down.
In the second and third examples, there is no route to the treasure that you will survive.