Problem A. Goldberg Machine 2

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

Rube is back in business with a new mind-blowing contraption! The contraption consists of a grid with n rows and m columns, with some cells of the grid containing an arrow pointing right or down, and other cells being empty. One can place a token in the top left corner cell of the grid, and then the magic happens: the token starts moving through the cells according to the arrow directions. Further, each arrow the token passes through switches direction when the token leaves its cell. Formally, if a token is in a cell with an arrow pointing right (resp. down), it then moves to the cell immediately to the right (resp. down), and the arrow in the original cell switches to pointing down (resp. right). The movement stops once the token arrives at an empty cell, or leaves the grid altogether, at which point the token is discarded.

As an example, consider a 3×3 grid shown below on the left ("`>`" and "`v`" describe arrows pointing right and down respectively, and "`.`" describes an empty cell). Two more grids show grid states after placing a token in the top left corner of the grid, and after placing another token after that.

```
>v>    v>>    >>>
vv> -> v>> -> >>>
v.>    v.>    >.v
```

Rube has manufactured many copies of his machine and sells them for a good price. However, he has been confronted by two very sophisticated customers who argue that their machines look the same, which offends their exquisite taste for truly unique things. Indeed, their machines have the same *shape*, that is, the sets of cells containing the arrows are equal for both machines, although the arrow configurations themselves may be different.

Rube agreed to modify the machines so that, despite looking similar, machines will never end up with the same arrow configurations after feeding any number of tokens in any, or both, of them. He is now bombarded with requests from the customers on how exactly machines should be modified: each request is to change a single arrow state in one of the machines. After each request Rube needs to determine if machines can eventually arrive to the same arrow configuration, and if so, what is the smallest total number of times you need to place a token in either of the machines before that happens, distributing the tokens arbitrarily between the two machines. Note that the changes **persist**, that is, no modification is undone after giving an answer to the request.

Once again, Rube is lamenting the complexity of his inventions since the task looks incredibly difficult. Help him answer all of customers' requests and avoid being sued out of existence.

Input

The first line contains three integers n, m, q — dimensions of the grid, and the number of requests ($1 \le n, m \le 100$, $1 \le q \le 10^5$).

The following n lines describe the state of the first machine. Each of these lines contains m characters. Each of the characters is one of "`>`", "`v`", or "`.`", which describe an arrow pointing right or down, or an empty cell respectively.

The following n lines after that describe the state of the second machine in the same format. It is guaranteed that the two grids have the same shape, that is, they contains arrows (regardless of directions) in the same set of cells. Further, it is guaranteed that the top left corner of each grid contains an arrow, and that a token can potentially reach each arrow cell after placing some number of tokens in each of the machines.

The following q lines describe requests in order they are received. Each request is described by three integers t, r, c — the number of the machine to be operated, and the row and the column indices of the cell

to be switched ($t \in \{1, 2\}$, $1 \le r \le n$, $1 \le c \le m$). Rows are numbered top to bottom, and columns are numbered left to right, both starting from 1. It is guaranteed that the cell to be switched in each request contains an arrow.

Extra empty lines may be present between sections for visual clarity.

Output

Print $q + 1$ lines, each containing answers after processing the first $0, 1, 2, \ldots, q$ requests respectively, in order. If the machines in their current states can not arrive to the same arrow configuration after any number of tokens placed in either of them, print -1 as the answer. Otherwise, print the smallest total number of tokens that need to be placed in the machines before they arrive to the same configuration.

Do not print leading zeros.

Example

standard input	standard output
3 3 9 >v> vv> v.> v>> v>> v.> 2 3 1 2 2 1 2 1 1 1 3 3 1 2 2 2 3 3 2 3 1 1 1 2 1 2 1	1 -1 -1 2 14 -1 -1 -1 -1 0