Code Jam 2015 - Round 2

Pegman

Problem

While using Google Street View, you may have picked up and dropped the character Pegman before. Today, a mischievous user is going to place Pegman in some cell of a rectangular grid of unit cells with **R** rows and **C** columns. Each of the cells in this grid might be blank, or it might be labeled with an arrow pointing in one of four possible directions: up, right, down, or left.

When Pegman is placed on a grid cell, if that cell is blank, Pegman stands still forever. However, if that cell has an arrow, Pegman starts to walk in that direction. As he walks, whenever he encounters a blank cell, he just keeps walking in his current direction, but whenever he encounters another arrow, he changes to the direction of that arrow and then keeps walking.

You know that it is possible that Pegman might keep happily walking around and around the grid forever, but it is also possible that Pegman's walk will take him over the edge of the grid! You may be able to prevent this and save him by changing the direction of one or more arrows. (Each arrow's direction can only be changed to one of the other three possible directions; arrows can only be changed, not added or removed.)

What is the smallest number of arrows you will need to change to ensure that Pegman will not walk off the edge, no matter where on the grid he is initially placed?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each begins with one line with two space-separated integers **R**, **C**. This line is followed by **R** lines, each of which has **C** characters, each of which describes a grid cell and is one of the following:

```
. period = no arrow
^ caret = up arrow
> greater than = right arrow
v lowercase v = down arrow
< less than = left arrow
```

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of arrows that must be changed to ensure that Pegman will not leave the grid no matter where he is initially placed, or the text `IMPOSSIBLE` if it is not possible to ensure this, no matter how many arrows you change.

Limits

Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.

Small dataset

Time limit: 240 seconds.
1 ≤ **R, C** ≤ 4.

Large dataset

Time limit: 480 seconds.
1 ≤ **R, C** ≤ 100.

Sample

Sample Input

```
4
2 1
^
^
2 2
>v
^<
3 3
...
.^.
...
1 1
.
```

Sample Output

```
Case #1: 1
Case #2: 0
Case #3: IMPOSSIBLE
Case #4: 0
```

In Case #1, Pegman is guaranteed to walk off the top edge of the grid, no matter where he is placed. You can prevent that by changing the topmost arrow to point down, which will cause him to walk back and forth between those two arrows forever.

In Case #2, no matter where Pegman is placed, he will walk around and around the board clockwise in a circle. No arrows need to be changed.

In Case #3, the mischievous user might place Pegman on the up arrow in the middle of the grid, in which case he will start walking and then walk off the top edge of the grid. Changing the direction of this arrow won't help: it would just make him walk off a different edge.

In Case #4, the only possible starting cell is blank, so Pegman will stand still forever and is in no danger.