Problem B. Puzzle: Patrick's Parabox

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Patrick's Parabox is a *Sokoban*-like game. A Sokoban puzzle is a grid, and each cell is a wall or a floor. There are several boxes and a player in some distinct floor cells, and they can not move to the wall cells or coincide. You can control the player to move in one of four directions, left, right, up, and down. When the player touches a box, it can push the box. The target is to move all boxes to some target cells.

Please read the following rules carefully. They may be different from the usual rules.

In this problem, there is **only one** box, and the box is the grid itself. That means if the player moves out of the grid, they may be "teleported" to a cell adjacent to the box; if the player moves to the box, they may be "teleported" to a cell on the boundary of the grid. Besides, there is also a target cell for the player. The player needs to move to the target cell at the end too.

Given a puzzle, you need to find the **minimum number of times to push** the box, such that the box and the player can arrive to their respective target cells.

The following are the detailed and formal rules.

Consider an $n \times m$ grid. Denote (i, j) as the cell in i-th row and j-th column. The rows are numbered $1, 2, \ldots, n$ from top to bottom, and the columns are numbered $1, 2, \ldots, m$ from left to right.

Denote W, S, A, and D as the control commands, which mean to move up, down, left, and right respectively.

Define that $v_{\mathrm{W}} = (-1, 0)$, $v_{\mathrm{S}} = (1, 0)$, $v_{\mathrm{A}} = (0, -1)$, $v_{\mathrm{D}} = (0, 1)$.

Define that $w_{\mathrm{W}} = (n, \left\lceil \frac{m}{2} \right\rceil)$, $w_{\mathrm{S}} = (1, \left\lceil \frac{m}{2} \right\rceil)$, $w_{\mathrm{A}} = (\left\lceil \frac{n}{2} \right\rceil, m)$, $w_{\mathrm{D}} = (\left\lceil \frac{n}{2} \right\rceil, 1)$.

In each operation, you can choose one of the control commands c, one of W, S, A, and D. Denote p as the cell which contains the player and b as the cell which contains the box before the operation:

- If $p + v_c = b$ and $b + v_c$ is a floor cell, the player moves to $p + v_c$ and the box moves to $b + v_c$. **Only this case** counts towards the answer, and so has to happen the minimum possible number of times.
- If $p + v_c$ is a wall cell, nothing happens.
- If $p + v_c$ is a floor cell and $p + v_c \neq b$, the player moves to $p + v_c$.
- If $p + v_c = b$ and $b + v_c$ is outside the grid, nothing happens.
- If $p + v_c = b$, $b + v_c$ is a wall cell and w_c is a wall cell, nothing happens.
- If $p + v_c = b$, $b + v_c$ is a wall cell and w_c is a floor cell, the player moves to w_c.
- if $p + v_c$ is outside the grid and $b + v_c$ is a wall cell, nothing happens.
- if $p + v_c$ is outside the grid and $b + v_c$ is outside the grid, nothing happens.
- if $p + v_c$ is outside the grid and $b + v_c$ is a floor cell, the player moves to $b + v_c$.

Note that the above are listed for covering all possibilities, but the operations are valid in only four of them (in other cases, nothing happens).

Input

There are multiple test cases. The first line of input contains an integer T ($1 \leq T \leq 10^5$), the number of test cases.

For each test case:

The first line contains two integers n and m ($2 \leq n, m \leq 10^5$), the size of the parabox.

Each of the following n lines contains a string of length m. Each character is one of "`#`", "`.`", "`p`", "`b`", "`=`" and "`-`". Here, "`#`" denotes a wall cell, "`p`" denotes the floor cell which contains the player, "`b`" denotes the

floor cell which contains the box, "=" denotes the target floor cell of the player, "-" denotes the target floor cell of the box, and "." denotes other floor cells.

It is guaranteed that each of "p", "b", "=", "-" occurs exactly once.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case:

If it is impossible to move the box and the player to their respective target cells, output -1. Otherwise, output an integer: the **minimum number of times to push** the box.

Examples

standard input	standard output
3 9 9 ######### #####..-# #..=##.## #.p.##.## ....##.## #...b..## #...##.## #....#### ####.#### 9 9 ######### #.......# #.#####.# #.#=....# ..#....-# ###.p.#.# #.....#b# #.....#.# ####.#### 9 9 ####.#### #....#### #.####.## #.......# #.......# ###.##### #=.b#..## #-..p..## #########	7 4 19
1 2 2 pb -=	-1

Note

The three puzzles in the first example are real levels in the game.