909. Snakes and Ladders   ⋯

Medium   ⊘   👍 2.1K   👎 586   ☆   ⟳

🔒 Companies

You are given an `n x n` integer matrix `board` where the cells are labeled from `1` to `n²` in a **Boustrophedon style** starting from the bottom left of the board (i.e. `board[n - 1][0]`) and alternating direction each row.

You start on square `1` of the board. In each move, starting from square `curr`, do the following:

- Choose a destination square `next` with a label in the range `[curr + 1, min(curr + 6, n²)]`.
  - This choice simulates the result of a standard **6-sided die roll**: i.e., there are always at most 6 destinations, regardless of the size of the board.

- If `next` has a snake or ladder, you **must** move to the destination of that snake or ladder. Otherwise, you move to `next`.

- The game ends when you reach the square `n²`.

A board square on row `r` and column `c` has a snake or ladder if `board[r][c] != -1`. The destination of that snake or ladder is `board[r][c]`. Squares `1` and `n²` do not have a snake or ladder.

Note that you only take a snake or ladder at most once per move. If the destination to a snake or ladder is the start of another snake or ladder, you do **not** follow the subsequent snake or ladder.

- For example, suppose the board is `[[-1,4],[-1,3]]`, and on the first move, your destination square is `2`. You follow the ladder to square `3`, but do **not** follow the subsequent ladder to `4`.

Return *the least number of moves required to reach the square* `n²`. *If it is not possible to reach the square, return* `-1`.

Example 1:
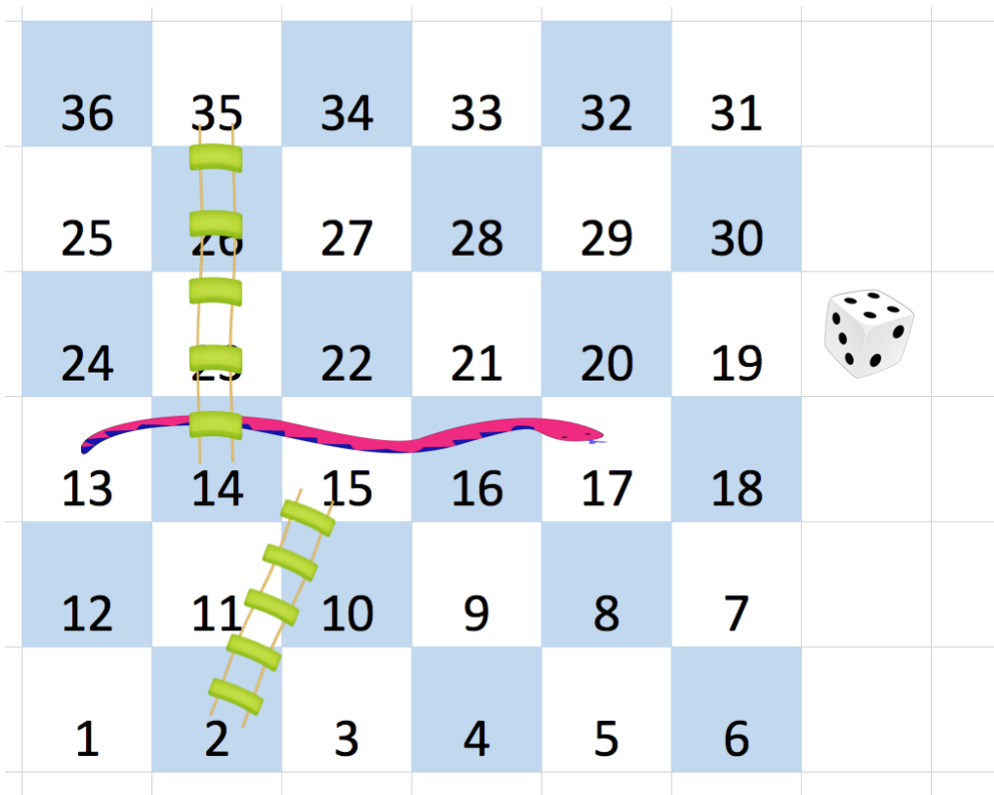
Input: board = [[-1,-1,-1,-1,-1,-1],[-1,-1,-1,-1,-1,-1],
[-1,-1,-1,-1,-1,-1],[-1,35,-1,-1,13,-1],[-1,-1,-1,-1,-1,-1],
[-1,15,-1,-1,-1,-1]]
Output: 4
Explanation:
In the beginning, you start at square 1 (at row 5, column 0).
You decide to move to square 2 and must take the ladder to square 15.
You then decide to move to square 17 and must take the snake to square 13.
You then decide to move to square 14 and must take the ladder to square 35.
You then decide to move to square 36, ending the game.
This is the lowest possible number of moves to reach the last square, so return 4.

Example 2:

Input: board = [[-1,-1],[-1,3]]
Output: 1

Constraints:

- `n == board.length == board[i].length`

- `2 <= n <= 20`

- `grid[i][j]` is either -1 or in the range $[1, n^2]$.

- The squares labeled `1` and n^2 do not have any ladders or snakes.