Problem 353: Design Snake Game

Problem Information

Difficulty: Medium
Acceptance Rate: 39.89%
Paid Only: Yes
Tags: Array, Hash Table, Design, Queue, Simulation

Problem Description

Design a [Snake game](https://en.wikipedia.org/wiki/Snake_\(video_game\)) that is played on a device with screen size `height x width`. [Play the game online](http://patorjk.com/games/snake/) if you are not familiar with the game.

The snake is initially positioned at the top left corner `(0, 0)` with a length of `1` unit.

You are given an array `food` where `food[i] = (ri, ci)` is the row and column position of a piece of food that the snake can eat. When a snake eats a piece of food, its length and the game's score both increase by `1`.

Each piece of food appears one by one on the screen, meaning the second piece of food will not appear until the snake eats the first piece of food.

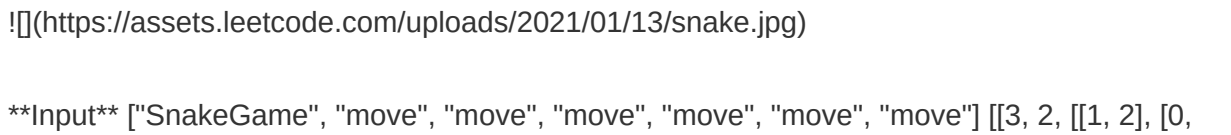
When a piece of food appears on the screen, it is **guaranteed** that it will not appear on a block occupied by the snake.

The game is over if the snake goes out of bounds (hits a wall) or if its head occupies a space that its body occupies **after** moving (i.e. a snake of length 4 cannot run into itself).

Implement the `SnakeGame` class:

* `SnakeGame(int width, int height, int[][] food)` Initializes the object with a screen of size `height x width` and the positions of the `food`. * `int move(String direction)` Returns the score of the game after applying one `direction` move by the snake. If the game is over, return `-1`.

Example 1:



Input ["SnakeGame", "move", "move", "move", "move", "move", "move"] [[3, 2, [[1, 2], [0, 1]]], ["R"], ["D"], ["R"], ["U"], ["L"], ["U"]] **Output** [null, 0, 0, 1, 1, 2, -1] **Explanation** SnakeGame snakeGame = new SnakeGame(3, 2, [[1, 2], [0, 1]]); snakeGame.move("R"); // return 0 snakeGame.move("D"); // return 0 snakeGame.move("R"); // return 1, snake eats the first piece of food. The second piece of food appears at (0, 1). snakeGame.move("U"); // return 1 snakeGame.move("L"); // return 2, snake eats the second food. No more food appears. snakeGame.move("U"); // return -1, game over because snake collides with border

Constraints:

* `1 <= width, height <= 104` * `1 <= food.length <= 50` * `food[i].length == 2` * `0 <= ri < height` * `0 <= ci < width` * `direction.length == 1` * `direction` is `'U'`, `'D'`, `'L'`, or `'R'`. * At most `104` calls will be made to `move`.

Code Snippets

C++:

```
class SnakeGame {
public:
SnakeGame(int width, int height, vector<vector<int>>& food) {

}

int move(string direction) {

}
};

/**
* Your SnakeGame object will be instantiated and called as such:
* SnakeGame* obj = new SnakeGame(width, height, food);
* int param_1 = obj->move(direction);
*/
```

Java:

```
class SnakeGame {

public SnakeGame(int width, int height, int[][] food) {

}

public int move(String direction) {

}
}

/**
* Your SnakeGame object will be instantiated and called as such:
* SnakeGame obj = new SnakeGame(width, height, food);
* int param_1 = obj.move(direction);
*/
```

Python3:

```
class SnakeGame:

def __init__(self, width: int, height: int, food: List[List[int]]):


def move(self, direction: str) -> int:



# Your SnakeGame object will be instantiated and called as such:
# obj = SnakeGame(width, height, food)
# param_1 = obj.move(direction)
```