DESCRIPTION
RULES
README
CODEWRITING
Score: 300/300

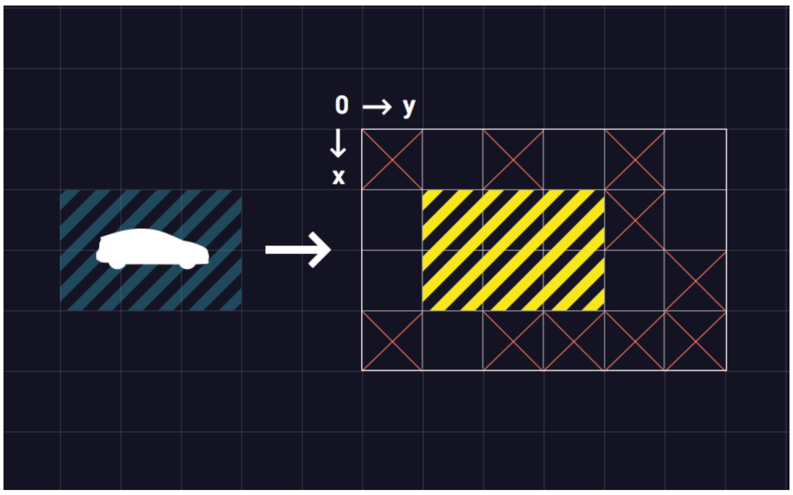
This time you are an Uber driver and you are trying your best to park your car in a parking lot.

Your car has length `carDimensions[0]` and width `carDimensions[1]`. You have already picked your lucky spot (rectangle of the same size as the car with upper left corner at `(luckySpot[0], luckySpot[1])`) and bottom right corner at `(luckySpot[2], luckySpot[3])` and you need to find out if it's possible to park there or not.
It is possible to park your car at a given spot if and only if you can drive through the parking lot without any turns to your lucky spot (for safety reasons, the car can only move in two directions inside the parking lot - forwards or backwards along the long side of the car) starting from some side of the lot (all four sides are valid options).

Naturally, you can't park your car if the lucky sport is already occupied. The car can't drive through or park at any of the occupied spots.

Example

- For `carDimensions = [3, 2]`,
```
parkingLot = [[1, 0, 1, 0, 1, 0],
              [0, 0, 0, 0, 1, 0],
              [0, 0, 0, 0, 0, 1],
              [1, 0, 1, 1, 1, 1]]
```
  and
  `luckySpot = [1, 1, 2, 3]`, the output should be
  `parkingSpot(carDimensions, parkingLot, luckySpot) = true`.



- For `carDimensions = [3, 2]`,
```
parkingLot = [[1, 0, 1, 0, 1, 0],
              [1, 0, 0, 0, 1, 0],
              [0, 0, 0, 0, 0, 1],
```

- `              [1, 0, 0, 0, 1, 1]]`
  and
  `luckySpot = [1, 1, 2, 3]`, the output should be
  `parkingSpot(carDimensions, parkingLot, luckySpot) = false`;
- For `carDimensions = [4, 1]`,
  the same `parkingLot` as in the previous example and
  `luckySpot = [0, 3, 3, 3]`, the output should be
  `parkingSpot(carDimensions, parkingLot, luckySpot) = true`.

Input/Output

- **[time limit] 4000ms (py)**

- **[input] array.integer carDimensions**

  Array of two positive integers. It is guaranteed that `carDimensions[0] > carDimensions[1]`.
  Constraints:
  `1 ≤ carDimensions[i] ≤ 10`.

- **[input] array.array.integer parkingLot**

  2-dimensional array, where `parkingLot[x][y]` is `1` if cell `(x, y)` is occupied and is `0`otherwise.
  Constraints:
  `3 ≤ parkingLot.length ≤ 10`,
  `3 ≤ parkingLot[0].length ≤ 10`.

- **[input] array.integer luckySpot**

  Array of four integers - coordinates of your lucky spot at the parking lot.
  It is guaranteed that one of the following statements is true:

  - `luckySpot[2] - luckySpot[0] + 1 = carDimensions[0]` and `luckySpot[3] - luckySpot[1] + 1 = carDimensions[1]`;
  - `luckySpot[2] - luckySpot[0] + 1 = carDimensions[1]` and `luckySpot[3] - luckySpot[1] + 1 = carDimensions[0]`.

  Constraints:
  `0 ≤ luckySpot[0] ≤ luckySpot[2] < parkingLot.length`,
  `0 ≤ luckySpot[1] ≤ luckySpot[3] < parkingLot[i].length`.

- **[output] boolean**

  `true` if it is possible to park your car, `false` otherwise.