Master algorithms together on Binary Search! Create a room, invite your friends, and race to finish the problems.

Daily Coding Problem #65

Problem

This problem was asked by Amazon.

Given a N by M matrix of numbers, print out the matrix in a clockwise spiral.

For example, given the following matrix:

```
[[1,  2,  3,  4,  5],
 [6,  7,  8,  9,  10],
 [11, 12, 13, 14, 15],
 [16, 17, 18, 19, 20]]
```

You should print out the following:

```
1
2
3
4
5
10
15
20
19
18
17
16
11
6
7
8
9
14
13
12
```

Solution

As you might imagine, there are many possible solutions for this problem. Ours involves keeping track of our current position and direction. As we move along and print each value, we set it to None. Then once we've either hit the edge or another None value (indicating we've seen it before), we change directions counterclockwise and keep on going.

We use an enum to define the directions, and some helper functions `next_direction`, `next_position`, and `should_change_direction` to help us lay out the code cleanly.

```
UP = 0
RIGHT = 1
DOWN = 2
LEFT = 3

DIRECTIONS = [RIGHT, DOWN, LEFT, UP]

def next_direction(direction):
    if direction == RIGHT:
        return DOWN
    elif direction == DOWN:
        return LEFT
    elif direction == LEFT:
        return UP
    elif direction == UP:
        return RIGHT

def next_position(position, direction):
    if direction == RIGHT:
        return (position[0], position[1] + 1)
    elif direction == DOWN:
        return (position[0] + 1, position[1])
    elif direction == LEFT:
        return (position[0], position[1] - 1)
    elif direction == UP:
        return (position[0] - 1, position[1])

def should_change_direction(M, r, c):
    in_bounds_r = 0 <= r < len(M)
    in_bounds_c = 0 <= c < len(M[0])
    return not in_bounds_r or not in_bounds_c or M[r][c] is None
```

```
def matrix_spiral_print(M):
    remaining = len(M) * len(M[0])
    current_direction = RIGHT
    current_position = (0, 0)
    while remaining > 0:
        r, c = current_position
        print(M[r][c])
        M[r][c] = None
        remaining -= 1

        possible_next_position = next_position(current_position, current_direction)
        if should_change_direction(M, possible_next_position[0], possible_next_position[1]):
            current_direction = next_direction(current_direction)
            current_position = next_position(current_position, current_direction)
        else:
            current_position = possible_next_position
```

This takes O(M * N) time.

© Daily Coding Problem 2019 Privacy Policy Terms of Service Press