Problem 2061: Number of Spaces Cleaning Robot Cleaned

Problem Information

Difficulty: Medium
Acceptance Rate: 62.42%
Paid Only: Yes
Tags: Array, Matrix, Simulation

Problem Description

A room is represented by a **0-indexed** 2D binary matrix `room` where a `0` represents an **empty** space and a `1` represents a space with an **object**. The top left corner of the room will be empty in all test cases.

A cleaning robot starts at the top left corner of the room and is facing right. The robot will continue heading straight until it reaches the edge of the room or it hits an object, after which it will turn 90 degrees **clockwise** and repeat this process. The starting space and all spaces that the robot visits are **cleaned** by it.

Return _the number of**clean** spaces in the room if the robot runs indefinitely._

Example 1:

Input: room = [[0,0,0],[1,1,0],[0,0,0]]

Output: 7

Explanation:

1. ■■■■■■■The robot cleans the spaces at (0, 0), (0, 1), and (0, 2). 2. The robot is at the edge of the room, so it turns 90 degrees clockwise and now faces down. 3. The robot cleans the spaces at (1, 2), and (2, 2). 4. The robot is at the edge of the room, so it turns 90 degrees clockwise and now faces left. 5. The robot cleans the spaces at (2, 1), and (2, 0). 6. The robot

has cleaned all 7 empty spaces, so return 7.

Example 2:

Input: room = [[0,1,0],[1,0,0],[0,0,0]]

Output: 1

Explanation:

1. The robot cleans the space at (0, 0). 2. The robot hits an object, so it turns 90 degrees clockwise and now faces down. 3. The robot hits an object, so it turns 90 degrees clockwise and now faces left. 4. The robot is at the edge of the room, so it turns 90 degrees clockwise and now faces up. 5. The robot is at the edge of the room, so it turns 90 degrees clockwise and now faces right. 6. The robot is back at its starting position. 7. The robot has cleaned 1 space, so return 1.

Example 3:

Input: room = [[0,0,0],[0,0,0],[0,0,0]]

Output: 8

Constraints:

* `m == room.length` * `n == room[r].length` * `1 <= m, n <= 300` * `room[r][c]` is either `0` or `1`. * `room[0][0] == 0`

Code Snippets

C++:

```
class Solution {
public:
int numberOfCleanRooms(vector<vector<int>>& room) {

}
```

```
};
```

Java:

```
class Solution {
public int numberOfCleanRooms(int[][] room) {

}
}
```

Python3:

```
class Solution:
def numberOfCleanRooms(self, room: List[List[int]]) -> int:
```