0221. 最大正方形

ITCharge 大约 2 分钟

- 标签：数组、动态规划、矩阵
- 难度：中等

题目链接

- 0221. 最大正方形 - 力扣

题目大意

描述：给定一个由 `'0'` 和 `'1'` 组成的二维矩阵 $matrix$。

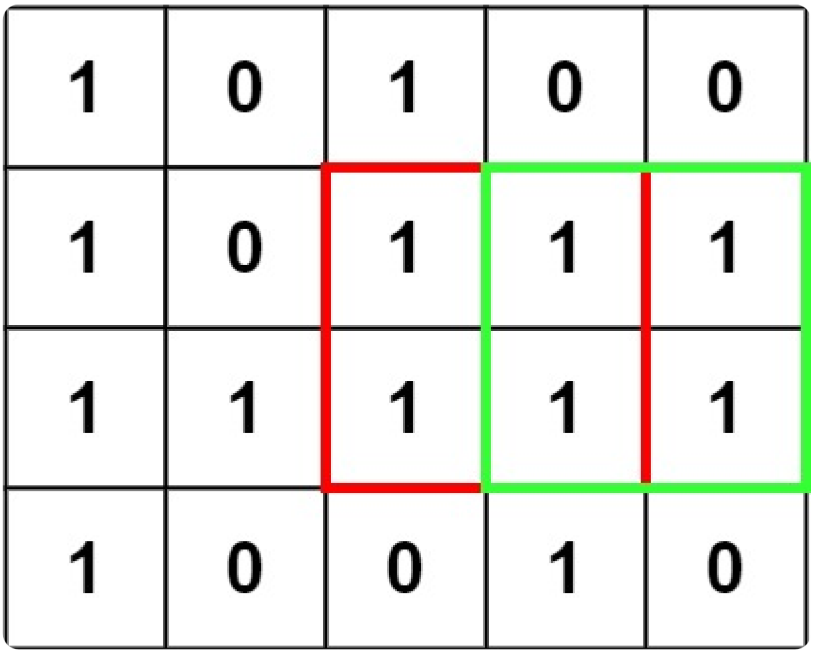
要求：找到只包含 `'1'` 的最大正方形，并返回其面积。

说明：

- $m == matrix.length$。
- $n == matrix[i].length$。
- $1 \le m, n \le 300$。
- $matrix[i][j]$ 为 `'0'` 或 `'1'`。

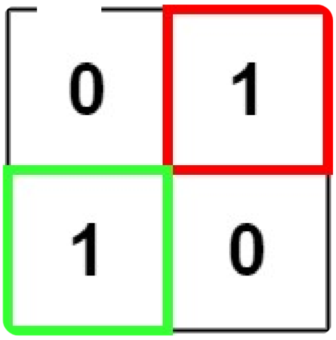
示例：

- 示例 1：




```py
输入：matrix = [["1","0","1","0","0"],["1","0","1","1","1"],
["1","1","1","1","1"],["1","0","0","1","0"]]
输出：4
```

- 示例 2：




```py
输入：matrix = [["0","1"],["1","0"]]
输出：1
```

解题思路

思路 1：动态规划

1. 划分阶段

按照正方形的右下角坐标进行阶段划分。

2. 定义状态

定义状态 $dp[i][j]$ 表示为：以矩阵位置 (i, j) 为右下角，且值包含 1 的正方形的最大边长。

3. 状态转移方程

只有当矩阵位置 (i, j) 值为 1 时，才有可能存在正方形。

- 如果矩阵位置 (i, j) 上值为 0，则 $dp[i][j] = 0$。
- 如果矩阵位置 (i, j) 上值为 1，则 $dp[i][j]$ 的值由该位置上方、左侧、左上方三者共同约束的，为三者中最小值加 1。即：$dp[i][j] = min(dp[i-1][j-1], dp[i-1][j], dp[i][j-1]) + 1$。

4. 初始条件

- 默认所有以矩阵位置 (i, j) 为右下角，且值包含 1 的正方形的最大边长都为 0，即 $dp[i][j] = 0$。

5. 最终结果

根据我们之前定义的状态， $dp[i][j]$ 表示为：以矩阵位置 (i, j) 为右下角，且值包含 1 的正方形的最大边长。则最终结果为所有 $dp[i][j]$ 中的最大值。

思路 1：代码

```py
class Solution:
    def maximalSquare(self, matrix: List[List[str]]) -> int:
        rows, cols = len(matrix), len(matrix[0])
        max_size = 0
        dp = [[0 for _ in range(cols + 1)] for _ in range(rows + 1)]
        for i in range(rows):
            for j in range(cols):
                if matrix[i][j] == '1':
                    if i == 0 or j == 0:
                        dp[i][j] = 1
                    else:
                        dp[i][j] = min(dp[i - 1][j - 1], dp[i - 1][j], dp[i][j - 1]) + 1
                    max_size = max(max_size, dp[i][j])
        return max_size * max_size
```

思路 1：复杂度分析

- **时间复杂度**：$O(m \times n)$，其中 m、n 分别为二维矩阵 $matrix$ 的行数和列数。
- **空间复杂度**：$O(m \times n)$。