

0048. 旋转图像

👤 ITCharge ⌚ 大约 2 分钟

- 标签：数组、数学、矩阵
- 难度：中等

题目链接

- [0048. 旋转图像 - 力扣](#)

题目大意

描述： 给定一个 $n \times n$ 大小的二维矩阵（代表图像） $matrix$ 。

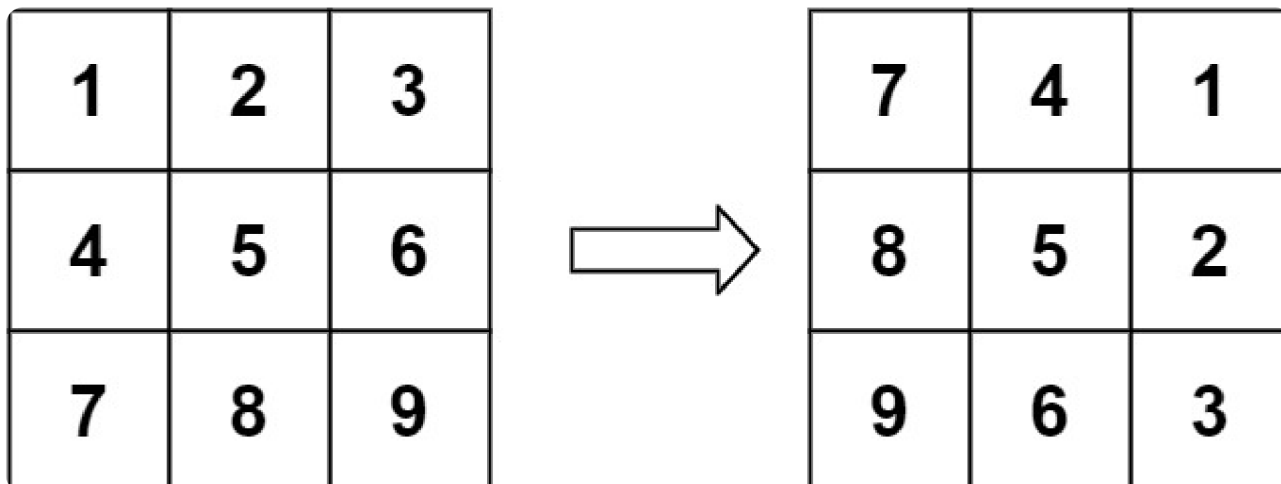
要求： 将二维矩阵 $matrix$ 顺时针旋转 90° 。

说明：

- 不能使用额外的数组空间。
- $n == matrix.length == matrix[i].length$ 。
- $1 \leq n \leq 20$ 。
- $-1000 \leq matrix[i][j] \leq 1000$ 。

示例：

- 示例 1：

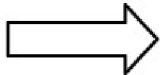


输入: `matrix = [[1,2,3],[4,5,6],[7,8,9]]`

输出: `[[7,4,1],[8,5,2],[9,6,3]]`

• 示例 2:

5	1	9	11
2	4	8	10
13	3	6	7
15	14	12	16



15	13	2	5
14	3	4	1
12	6	8	9
16	7	10	11

输入: `matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]`

输出: `[[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]`

解题思路

思路 1: 原地旋转

如果使用额外数组空间的话, 将对应元素存放到对应位置即可。如果不使用额外的数组空间, 则需要观察每一个位置上的点最初位置和最终位置有什么规律。

对于矩阵中第 i 行的第 j 个元素, 在旋转后, 它出现在倒数第 i 列的第 j 个位置。即 $matrix_{new}[j][n-i-1] = matrix[i][j]$ 。

而 $matrix_{new}[j][n-i-1]$ 的点经过旋转移动到了 $matrix[n-i-1][n-j-1]$ 的位置。

$matrix[n-i-1][n-j-1]$ 位置上的点经过旋转移动到了 $matrix[n-j-1][i]$ 的位置。

$matrix[n-j-1][i]$ 位置上的点经过旋转移动到了最初的 $matrix[i][j]$ 的位置。

这样就形成了一个循环, 我们只需要通过一个临时变量 `temp` 就可以将循环中的元素逐一进行交换。Python 中则可以直接使用语法直接交换。

思路 1：代码

```
class Solution:
    def rotate(self, matrix: List[List[int]]) -> None:
        n = len(matrix)

        for i in range(n // 2):
            for j in range((n + 1) // 2):
                matrix[i][j], matrix[n - j - 1][i], matrix[n - i - 1][n - j - 1], matrix[j][n - i - 1] = matrix[n - j - 1][i], matrix[n - i - 1][n - j - 1], matrix[j][n - i - 1], matrix[i][j]
```

py

思路 1：复杂度分析

- 时间复杂度： $O(n^2)$ 。
- 空间复杂度： $O(1)$ 。

思路 2：原地翻转

通过观察可以得出：原矩阵可以通过一次「水平翻转」+「主对角线翻转」得到旋转后的二维矩阵。

思路 2：代码

```
def rotate(self, matrix: List[List[int]]) -> None:
    n = len(matrix)

    for i in range(n // 2):
        for j in range(n):
            matrix[i][j], matrix[n - i - 1][j] = matrix[n - i - 1][j], matrix[i]
```

py

[j]

```
for i in range(n):  
    for j in range(i):  
        matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]
```

思路 2：复杂度分析

- 时间复杂度： $O(n^2)$ 。
- 空间复杂度： $O(1)$ 。