LeetCode 图解 | 48. 旋转图像

题目描述

给定一个 n × n 的二维矩阵表示一个图像。

将图像顺时针旋转 90 度。

说明：

你必须在原地旋转图像，这意味着你需要直接修改输入的二维矩阵。**请不要**使用另一个矩阵来旋转图像。

示例 1:

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

原地旋转输入矩阵，使其变为:
[
  [7,4,1],
  [8,5,2],
  [9,6,3]
]
```

示例 2:

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

原地旋转输入矩阵，使其变为:
[
  [15,13, 2, 5],
  [14, 3, 4, 1],
  [12, 6, 8, 9],
  [16, 7,10,11]
]
```

题目解析

这道题的主要难点在于如何**原地**旋转矩阵。

我们发现，矩阵中的一个元素旋转四次之后会回到原先的位置。也就是说，这四个元素在旋转时位置互相交换了。例如元素 `(i, j)` 对应的四个位置分别是：

- `(i, j)`
- `(N-1-j, i)`
- `(N-1-i, N-1-j)`
- `(j, N-1-i)`

为了旋转这四个元素，我们可以用一个临时变量保存其中一个元素，然后让几个元素依次赋值。

那么，一共有多少个这样的四元素组呢？这要分情况来看。如果 n 是偶数的话，这相当于把矩阵均分成四块，每块的元素个数是 $ (n/2) \times (n/2)$。如果 n 是奇数，矩阵的中心元素是不随旋转移动的，而剩下的元素均分成四块，每块的元素个数是 $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$。我们对一块中的所有元素做一次四元素旋转即可。

动画理解

参考代码

```
class Solution {
    public void rotate(int[][] matrix) {
        int N = matrix.length;
        for (int i = 0; i < N/2; i++) {
            for (int j = 0; j < (N+1)/2; j++) {
                int t = matrix[i][j];
                matrix[i][j] = matrix[N-1-j][i];
                matrix[N-1-j][i] = matrix[N-1-i][N-1-j];
                matrix[N-1-i][N-1-j] = matrix[j][N-1-i];
                matrix[j][N-1-i] = t;
            }
        }
    }
}
```

复杂度分析

- 时间复杂度：$O(n^2)$。
- 空间复杂度：$O(1)$。