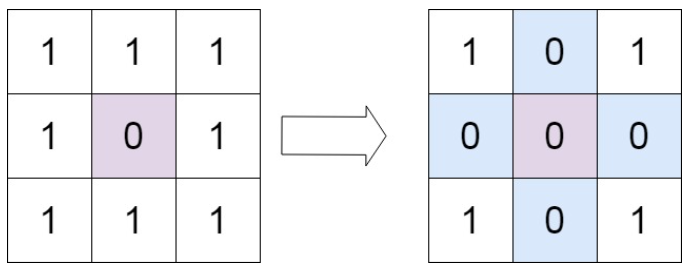
# 题目

给定一个m x n的矩阵，如果一个元素为0，则将其所在行和列的所有元素都设为0。请使用原地算法。

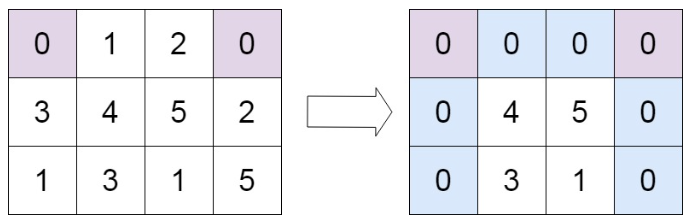
示例 1：



输入：matrix = [[1,1,1],[1,0,1],[1,1,1]]

输出：[[1,0,1],[0,0,0],[1,0,1]]

示例 2：



输入：matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]

输出：[[0,0,0,0],[0,4,5,0],[0,3,1,0]]

提示：

m == matrix.length

n == matrix[0].length

1 <= m, n <= 200

-231 <= matrix[i][j] <= 231 - 1

进阶：

一个直观的解决方案是使用O(mn)的额外空间，但这并不是一个好的解决方案。

一个简单的改进方案是使用O(m + n)的额外空间，但这仍然不是最好的解决方案。

你能想出一个仅使用常量空间的解决方案吗？

# 分析

## 方法一：模拟/使用标记数组

思路和算法：

我们可以用两个标记数组分别记录每一行和每一列是否有零出现。

具体地，我们首先遍历该数组一次，如果某个元素为0，那么就将该元素所在的行和列所对应标记数组的位置置为true。最后我们再次遍历该数组，用标记数组更新原数组即可。

代码：

class Solution {

public:

void setZeroes(vector<vector<int>>& matrix) {

int m = matrix.size();

int n = matrix[0].size();

vector<int> row(m), col(n);

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

if (matrix[i][j] == 0) {

row[i] = col[j] = true;

}

}

}

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

if (row[i] || col[j]) {

matrix[i][j] = 0;

}

}

}

}

};

复杂度分析：

时间复杂度：O(mn)，其中m是矩阵的行数，n是矩阵的列数。我们至多只需要遍历该矩阵两次。

空间复杂度：O(m+n)，其中m是矩阵的行数，n是矩阵的列数。我们需要分别记录每一行或每一列是否有零出现。