LeetCode 73. Set Matrix Zeroes

1. Problem Title & Link

- **73. Set Matrix Zeroes**
- https://leetcode.com/problems/set-matrix-zeroes/

2. Problem Statement (Short Summary)

Given an m x n matrix, if any element is 0, set **its entire row and column** to 0.
Do this **in-place**, meaning modify the matrix without using extra space for another matrix.

3. Examples (Input → Output)

Input:
[[1,1,1],
[1,0,1],
[1,1,1]]

Output:
[[1,0,1],
[0,0,0],
[1,0,1]]
Input:
[[0,1,2,0],
[3,4,5,2],
[1,3,1,5]]

Output:
[[0,0,0,0],
[0,4,5,0],
[0,3,1,0]]

4. Constraints

- m == matrix.length
- n == matrix[0].length
- 1 <= m, n <= 200
- -2^{31} <= matrix[i][j] <= $2^{31} - 1$

5. Thought Process (Step by Step)

This problem checks if students can mark what to zero **without corrupting** the matrix too early.

We can approach it in three ways ❤️

Brute Force (O(m×n) Space)

- Create two arrays: row[] and col[] to mark which rows and columns must be zeroed.
- Then make another pass and zero them out.

  ❌ Works but uses O(m + n) extra space.

Optimized (In-Place using First Row & Column as Markers)

💡 Key idea: use the first row and column *as flag storage* instead of extra arrays.

Steps:

1. Check if first row or column has any zeros → store booleans firstRowZero, firstColZero.
2. Traverse matrix (excluding first row/col):
   - If matrix[i][j] == 0, mark:
     - matrix[i][0] = 0
     - matrix[0][j] = 0
3. Traverse again (excluding first row/col):
   - If matrix[i][0] == 0 or matrix[0][j] == 0 → set matrix[i][j] = 0
4. Finally, zero out the first row and column if needed.

This preserves the marking information inside the matrix itself. 🧩

6. Pseudocode

```
firstRowZero = any(matrix[0][j] == 0)
firstColZero = any(matrix[i][0] == 0)

# Mark zeros
for i in 1..m-1:
  for j in 1..n-1:
    if matrix[i][j] == 0:
      matrix[i][0] = 0
      matrix[0][j] = 0

# Apply zeros
for i in 1..m-1:
  for j in 1..n-1:
    if matrix[i][0] == 0 or matrix[0][j] == 0:
      matrix[i][j] = 0

# Zero first row/column if needed
if firstRowZero: zero out row 0
if firstColZero: zero out col 0
```

7. Code Implementation

✅ **Python**

```python
class Solution:
  def setZeroes(self, matrix: List[List[int]]) -> None:
    rows, cols = len(matrix), len(matrix[0])
    first_row_zero = any(matrix[0][j] == 0 for j in range(cols))
    first_col_zero = any(matrix[i][0] == 0 for i in range(rows))

    # Step 1: mark zeros in first row/col
    for i in range(1, rows):
      for j in range(1, cols):
        if matrix[i][j] == 0:
```

```
                matrix[i][0] = 0
                matrix[0][j] = 0

        # Step 2: use marks to set cells
        for i in range(1, rows):
            for j in range(1, cols):
                if matrix[i][0] == 0 or matrix[0][j] == 0:
                    matrix[i][j] = 0

        # Step 3: handle first row
        if first_row_zero:
            for j in range(cols):
                matrix[0][j] = 0

        # Step 4: handle first col
        if first_col_zero:
            for i in range(rows):
                matrix[i][0] = 0
```

✅ **Java**

```
class Solution {
    public void setZeroes(int[][] matrix) {
        int rows = matrix.length, cols = matrix[0].length;
        boolean firstRowZero = false, firstColZero = false;

        for (int j = 0; j < cols; j++)
            if (matrix[0][j] == 0) firstRowZero = true;

        for (int i = 0; i < rows; i++)
            if (matrix[i][0] == 0) firstColZero = true;

        for (int i = 1; i < rows; i++)
            for (int j = 1; j < cols; j++)
                if (matrix[i][j] == 0) {
                    matrix[i][0] = 0;
                    matrix[0][j] = 0;
                }

        for (int i = 1; i < rows; i++)
            for (int j = 1; j < cols; j++)
                if (matrix[i][0] == 0 || matrix[0][j] == 0)
                    matrix[i][j] = 0;

        if (firstRowZero)
            for (int j = 0; j < cols; j++) matrix[0][j] = 0;

        if (firstColZero)
            for (int i = 0; i < rows; i++) matrix[i][0] = 0;
```

```
    }
}
```

8. Time & Space Complexity

- **Time:** O(m × n) — each cell visited twice
- **Space:** O(1) — in-place, using matrix as markers

9. Dry Run (Step-by-Step Execution)

👉 Input:

[[1,1,1],
[1,0,1],
[1,1,1]]

Step	Action	Matrix
Initial	firstRowZero=False, firstColZero=False	unchanged
Mark phase	mark (1,0)=0, (0,1)=0	[[1,0,1],[0,0,1],[1,1,1]]
Apply zeros	row 1 and col 1 → zeroed	[[1,0,1],[0,0,0],[1,0,1]]

✅ Final:

[[1,0,1],
[0,0,0],
[1,0,1]]

10. Concept Insight Table

Core Concept	Common Use Cases	Common Traps	Builds / Next Steps
In-Place Matrix Marking — use existing matrix as storage to reduce extra space.	- Matrix transformation problems - Space-optimized in-place algorithms - Flag propagation in grids	- Forgetting to store first row/col zero status - Overwriting markers too early - Incorrect order of operations	🔷 Builds to **LeetCode 289 (Game of Life)** 🔷 Leads into **matrix mutation with O(1) extra space** 🔷 Reinforces “marker-cell” optimization for 2D arrays

11. Common Mistakes / Edge Cases

- Modifying first row/column before using them as markers.
- Forgetting that (0,0) belongs to both — special handling needed.
- Failing when entire first row/col is already zero.

12. Variations / Follow-Ups

- Similar problem: **Game of Life (LC 289)** — evolve matrix using states in place.
- “Set Matrix Ones” or “Flip Matrix Zones” variants — identical logic.
- Extend to 3D matrices (practice space reasoning).