# **Leetcode 73 – Set Matrix Zeroes**

## Problem Understanding

You are given an m x n integer matrix.  
If an element is 0, **set its entire row and column to 0**.

Do it **in-place** (without using extra space proportional to the matrix size).

### Example:

**Input:**

[

[1,1,1],

[1,0,1],

[1,1,1]

]

**Output:**

[

[1,0,1],

[0,0,0],

[1,0,1]

]

## Optimized Java Solution (In-Place using First Row & Column)

class Solution {

public void setZeroes(int[][] matrix) {

int m = matrix.length, n = matrix[0].length;

boolean firstRowZero = false, firstColZero = false;

// Step 1: Mark rows and cols to be zeroed

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

if (matrix[i][0] == 0) firstColZero = true;

}

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

if (matrix[0][j] == 0) firstRowZero = true;

}

for (int i = 1; i < m; i++) {

for (int j = 1; j < n; j++) {

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

matrix[i][0] = 0; // mark row

matrix[0][j] = 0; // mark col

}

}

}

// Step 2: Zero out cells based on markers

for (int i = 1; i < m; i++) {

for (int j = 1; j < n; j++) {

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

matrix[i][j] = 0;

}

}

// Step 3: Handle first row and column

if (firstRowZero) {

for (int j = 0; j < n; j++) matrix[0][j] = 0;

}

if (firstColZero) {

for (int i = 0; i < m; i++) matrix[i][0] = 0;

}

}

}

## Dry Run Using Table

### Input Matrix:

[ [1, 1, 1],

[1, 0, 1],

[1, 1, 1] ]

**Step 1: Detect Zeros**

* firstRowZero = false, firstColZero = false
* Mark row and column of cell (1,1) = 0:
  + Set matrix[1][0] = 0, matrix[0][1] = 0

**Step 2: Apply Markers**

* For every cell not in first row/col, if marker is 0 → set to 0

**Step 3: First Row/Col**

* First row/col are unchanged (no initial zeros)

### Final Matrix:

[ [1, 0, 1],

[0, 0, 0],

[1, 0, 1] ]

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(m × n) |
| Space | O(1) ✅ |

We use **matrix itself as marker space**, achieving constant space.

## Alternate Approaches

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Time | Space | Notes |
| ✅ In-Place (marker) | O(m×n) | O(1) | Optimal |
| 🔁 Extra arrays | O(m×n) | O(m+n) | Simple but not in-place |
| Brute Force (extra copy) | O(m×n) | O(m×n) | Unacceptable space |