MATRIX

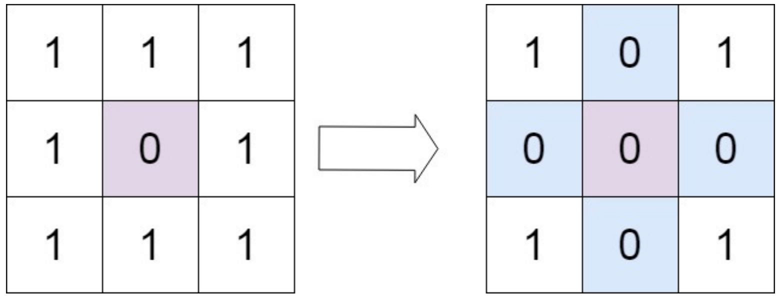
Problem – 73. Set Matrix Zeroes

Medium

LeetCode leetcode.com/problems/set-matrix-zeroes

Problem

- You are given a **matrix m x n**
- When an element in the matrix is **0**, set the whole column and row to **zero**
- **You must do it in place**



Solution – 73. Set Matrix Zeroes

Medium

LeetCode leetcode.com/problems/set-matrix-zeroes

Solution

- Iterate through the matrix top-down
- For each column (from col = 1 to n - 1), when you **find 0**, set:

  matrix[row][0] = 0 → marks that the row should be zeroed

  matrix[0][col] = 0 → marks that the col should be zeroed
- If **matrix[row][0] == 0**, set a variable **col0 = true** to remember if column 0 must be zeroed later
- Iterate bottom-top, right-left
- For each cell, if **matrix[row][0] == 0** or **matrix[0][col] == 0** set **matrix[row][col] = 0**
- It must be bottom-top, right-left to not set the first row to zero first
- Also check if col0 is true. If true, set the first column to zero:

  matrix[row][0] = 0

Code – 73. Set Matrix Zeroes

Medium

LeetCode leetcode.com/problems/set-matrix-zeroes

Code

Time: **O(m × n)** Space: **O(1)** where m is the number of rows and n the number of columns. As this is an in-place implementation, there is no additional space being allocated, therefore space complexity is O(1)

```
void setZeroes(vector<vector<int>>& matrix) {
    int m = matrix.size();
    int n = matrix[0].size();
    bool col0 = false;

    for (int row = 0; row < m; ++row) {
        if (matrix[row][0] == 0) col0 = true;
        for (int col = 1; col < n; ++col) {
            if (matrix[row][col] == 0) {
                matrix[row][0] = 0;
                matrix[0][col] = 0;
            }
        }
    }

    for (int row = m - 1; row >= 0; --row) {
        for (int col = n - 1; col >= 1; --col) {
            if (matrix[row][0] == 0 || matrix[0][col] == 0) {
                matrix[row][col] = 0;
            }
        }
        if (col0) {
            matrix[row][0] = 0;
        }
    }
}
```

Problem – 54. Spiral Matrix

Medium

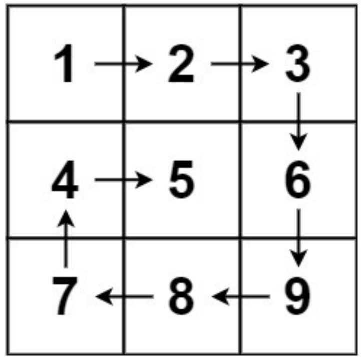
LeetCode leetcode.com/problems/spiral-matrix

Problem

- You are given a matrix **m x n**
- Return the elements in a flat array, the same order as the image
- **Example**

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

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



Solution – 54. Spiral Matrix

Medium

LeetCode leetcode.com/problems/spiral-matrix

Solution

- **Traverse the matrix in spiral order following four directions**: right across top row, down the right column, left across bottom row, and up the left column, then repeat inward
- **Maintain four boundary variables that shrink after each direction**: rowStart, rowEnd, colStart, and colEnd get updated after completing each side of the spiral to move the boundaries inward
- **Boundary handling is the main challenge**: It's easy to introduce bugs when determining when to stop traversal or when to skip certain directions, requiring careful condition checks
- **Non-square matrices require special boundary checks**: The conditional statements if (`rowStart <= rowEnd`) and if (`colStart <= colEnd`) prevent adding duplicate elements when dealing with rectangular matrices or edge cases like single rows/columns

Code – 54. Spiral Matrix

Medium

LeetCode leetcode.com/problems/spiral-matrix

Code

Time: **O(m × n)** Space: **O(1)**

```
vector<int> spiralOrder(vector<vector<int>>& matrix) {
    vector<int> result;
    int rowStart = 0;
    int colStart = 0;
    int colEnd = matrix[0].size() - 1;
    int rowEnd = matrix.size() - 1;

    while(rowStart <= rowEnd && colStart <= colEnd) {
        for (int col = colStart; col <= colEnd; ++col) {
            result.push_back(matrix[rowStart][col]);
        }
        ++rowStart;

        for (int row = rowStart; row <= rowEnd; ++row) {
            result.push_back(matrix[row][colEnd]);
        }
        --colEnd;

        // if matrix is not square condition
        if (rowStart <= rowEnd) {
            for (int col = colEnd; col >= colStart; --col) {
                result.push_back(matrix[rowEnd][col]);
            }
            --rowEnd;
        }


        if (colStart <= colEnd) {
            for (int row = rowEnd; row >= rowStart; --row) {
                result.push_back(matrix[row][colStart]);
            }
            colStart++;
        }
    }
    return result;
}
```