SET MATRIX ZEROES

The question seems to be pretty simple but the trick here is that we need to modify the given matrix in place i.e. our space complexity needs to $O(1)$.

We will go through two different approaches to the question. The first approach makes use of additional memory while the other does not.

Approach 1: Additional Memory Approach

Intuition

If any cell of the matrix has a zero we can record its row and column number. All the cells of this recorded row and column can be marked zero in the next iteration.

Algorithm

1. We make a pass over our original array and look for zero entries.
2. If we find that an entry at `[i, j]` is 0, then we need to record somewhere the row `i` and column `j`.
3. So, we use two `sets`, one for the rows and one for the columns.

```
4.  if cell[i][j] == 0 {
5.      row_set.add(i)
6.      column_set.add(j)
    }
```

7. Finally, we iterate over the original matrix. For every cell we check if the row `r` or column `c` had been marked earlier. If any of them was marked, we set the value in the cell to 0.

```
8.  if r in row_set or c in column_set {
9.      cell[r][c] = 0
    }
```

Complexity Analysis

- Time Complexity: $O(M \times N)$ where M and N are the number of rows and columns respectively.
- Space Complexity: $O(M+N)$.

Approach 2: O(1) Space, Efficient Solution

Intuition

Rather than using additional variables to keep track of rows and columns to be reset, we use the matrix itself as the *indicators*.

The idea is that we can use the **first cell** of every row and column as a **flag**. This flag would determine whether a row or column has been set to zero. This means for every cell instead of going to $\mathrm{M}+\mathrm{N}M+N$ cells and setting it to zero we just set the flag in two cells.

```
if cell[i][j] == 0 {

    cell[i][0] = 0

    cell[0][j] = 0

}
```

These flags are used later to update the matrix. If the first cell of a row is set to zero this means the row should be marked zero. If the first cell of a column is set to zero this means the column should be marked zero.

Algorithm

1. We iterate over the matrix and we mark the first cell of a row `i` and first cell of a column `j`, if the condition in the pseudo code above is satisfied. i.e. if `cell[i][j] == 0`.

2. The first cell of row and column for the first row and first column is the same i.e. `cell[0][0]`. Hence, we use an additional variable to tell us if the first column had been marked or not and the `cell[0][0]` would be used to tell the same for the first row.

3. Now, we iterate over the original matrix starting from second row and second column i.e. `matrix[1][1]` onwards. For every cell we check if the row `r` or column `c` had been marked earlier by checking the respective first row cell or first column cell. If any of them was marked, we set the value in the cell to 0. Note the first row and first column serve as the `row_set` and `column_set` that we used in the first approach.

4. We then check if `cell[0][0] == 0`, if this is the case, we mark the first row as zero.

5. And finally, we check if the first column was marked, we make all entries in it as zeros.

0	0	0	0
0	0	1	1
0	1	0	0
0	0	0	1

We iterate the matrix we got from the above steps and mark respective cells zeroes.

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Complexity Analysis

- Time Complexity : $O(M \times N)$
- Space Complexity : $O(1)$

```
class Solution {

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

```
Boolean isCol = false;
int R = matrix.length;
int C = matrix[0].length;

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

  // Since first cell for both first row and first column is the same i.e. matrix[0][0]
  // We can use an additional variable for either the first row/column.
  // For this solution we are using an additional variable for the first column
  // and using matrix[0][0] for the first row.
  if (matrix[i][0] == 0) {
    isCol = true;
  }

  for (int j = 1; j < C; j++) {
    // If an element is zero, we set the first element of the corresponding row and column to 0
    if (matrix[i][j] == 0) {
      matrix[0][j] = 0;
      matrix[i][0] = 0;
    }
  }
}

// Iterate over the array once again and using the first row and first column, update the elements.
for (int i = 1; i < R; i++) {
  for (int j = 1; j < C; j++) {
    if (matrix[i][0] == 0 || matrix[0][j] == 0) {
      matrix[i][j] = 0;
    }
  }
```

```
    }

    // See if the first row needs to be set to zero as well
    if (matrix[0][0] == 0) {
      for (int j = 0; j < C; j++) {
        matrix[0][j] = 0;
      }
    }

    // See if the first column needs to be set to zero as well
    if (isCol) {
      for (int i = 0; i < R; i++) {
        matrix[i][0] = 0;
      }
    }
  }
}
```