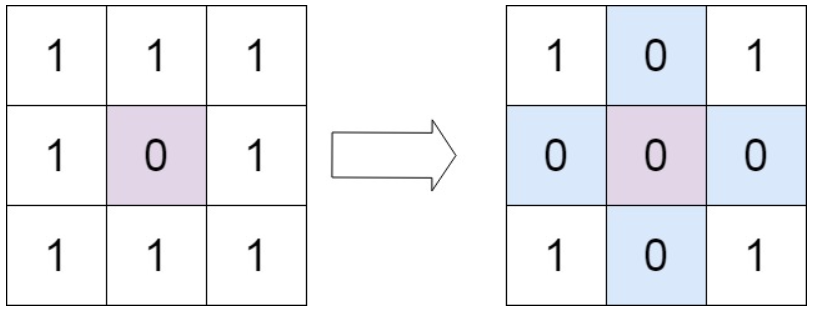
<https://leetcode.com/problems/set-matrix-zeroes/>

Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's.

You must do it [in place](https://en.wikipedia.org/wiki/In-place_algorithm).

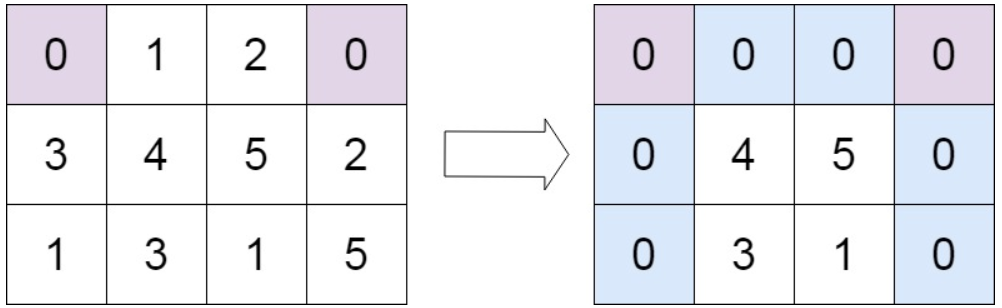
**Example 1:**



Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]

Output: [[1,0,1],[0,0,0],[1,0,1]]

**Example 2:**



Input: matrix = [[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]]

**Constraints:**

* m == matrix.length
* n == matrix[0].length
* 1 <= m, n <= 200
* -231 <= matrix[i][j] <= 231 - 1

**Follow up:**

* A straightforward solution using O(mn) space is probably a bad idea.
* A simple improvement uses O(m + n) space, but still not the best solution.
* Could you devise a constant space solution?

**Attempt 1: 2023-09-10**

**Solution 1: Brute Force, require m \* n extra space (10 min)**

class Solution {

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

int m = matrix.length;

int n = matrix[0].length;

int[][] copy = new int[m][n];

// Copy all the elements of given 'matrix' to 'copy'

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

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

copy[i][j] = matrix[i][j];

}

}

// While traversing given 'matrix' whenever we encounter 0,

// we will make the entire row and column of the 'copy' to 0

// Note: in this step, no change on original 'matrix',

// because we don't want on the fly change overwrite non-zero

// cells to zero on original 'matrix', which will result

// wrong consequence

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

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

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

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

copy[k][j] = 0;

}

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

copy[i][k] = 0;

}

}

}

}

// Finally we can again copy all the elements of 'copy' to

// given matrix

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

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

matrix[i][j] = copy[i][j];

}

}

}

}

Time Complexity: O((mn)∗(m+n))

Space Complexity: O(mn)

**Refer to**

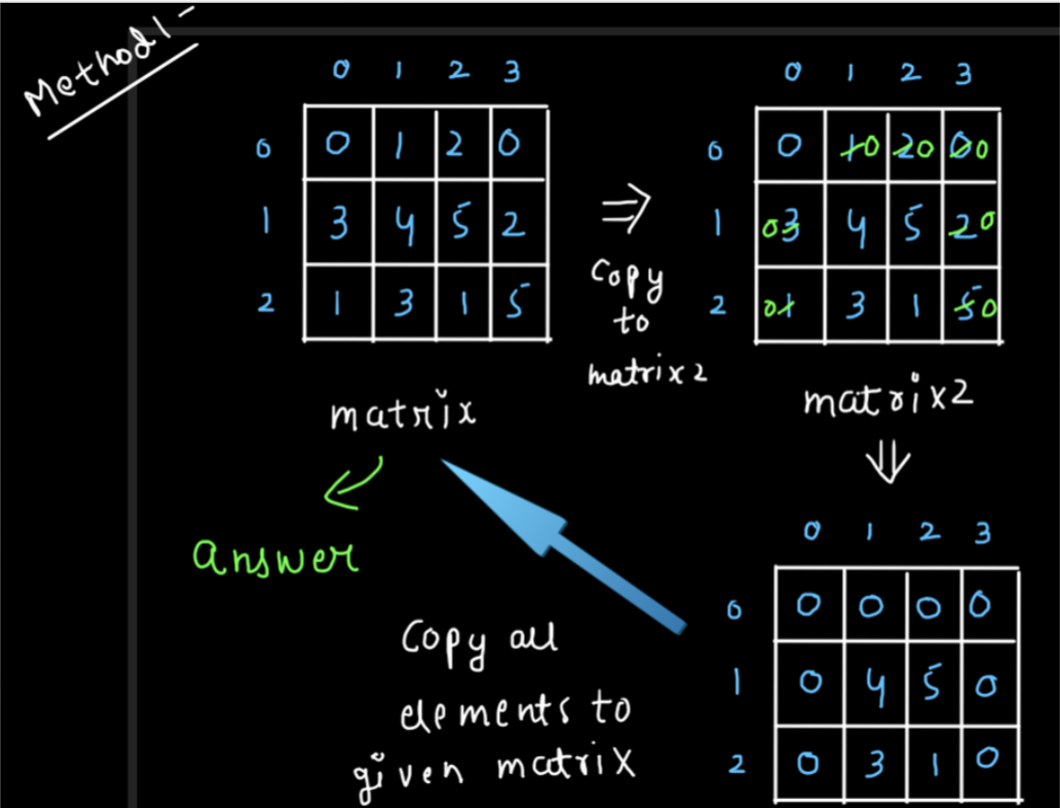
<https://leetcode.com/problems/set-matrix-zeroes/solutions/2525398/all-approaches-from-brute-force-to-optimal-with-easy-explanation/>

**Method 1:**(Brute force)

-using another matrix (let's say it matrix2)

* we can copy all the elements of given matrix to matrix2
* while traversing given matrix whenever we encounter 0, we will make the entire row and column of the matrix2 to 0
* finally we can again copy all the elements of matrix2 to given matrix

-**Time:**O((mn)∗(m+n)), **Space:**O(mn)



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

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

int matrix2[][]= new int[m][n];

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

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

matrix2[i][j]=matrix[i][j];

}

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

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

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

for(int k=0;k<n;k++)

matrix2[i][k]=0;

for(int k=0;k<m;k++)

matrix2[k][j]=0;

}

}

}

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

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

matrix[i][j]=matrix2[i][j];

}

}

**Solution 2: Optimize to (m + n) extra space (10 min)**

class Solution {

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

int m = matrix.length;

int n = matrix[0].length;

// We can use two separate arrays one for rows (zero\_rows)

// and one for columns (zero\_cols) and initialize them to 1

int[] zero\_rows = new int[m];

int[] zero\_cols = new int[n];

Arrays.fill(zero\_rows, 1);

Arrays.fill(zero\_cols, 1);

// While traversing the given matrix whenever we encounter

// 0 at (i,j), we will set zero\_rows[i]=0 and zero\_cols[j]=0

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

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

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

zero\_rows[i] = 0;

zero\_cols[j] = 0;

}

}

}

// After completion of step 2, again iterate through the

// matrix and for any (i,j), if zero\_rows[i] or zero\_cols[j]

// is 0 then update matrix[i][j] to 0

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

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

if(zero\_rows[i] == 0 || zero\_cols[j] == 0) {

matrix[i][j] = 0;

}

}

}

}

}

Time Complexity: O(mn)

Space Complexity: O(m+n)

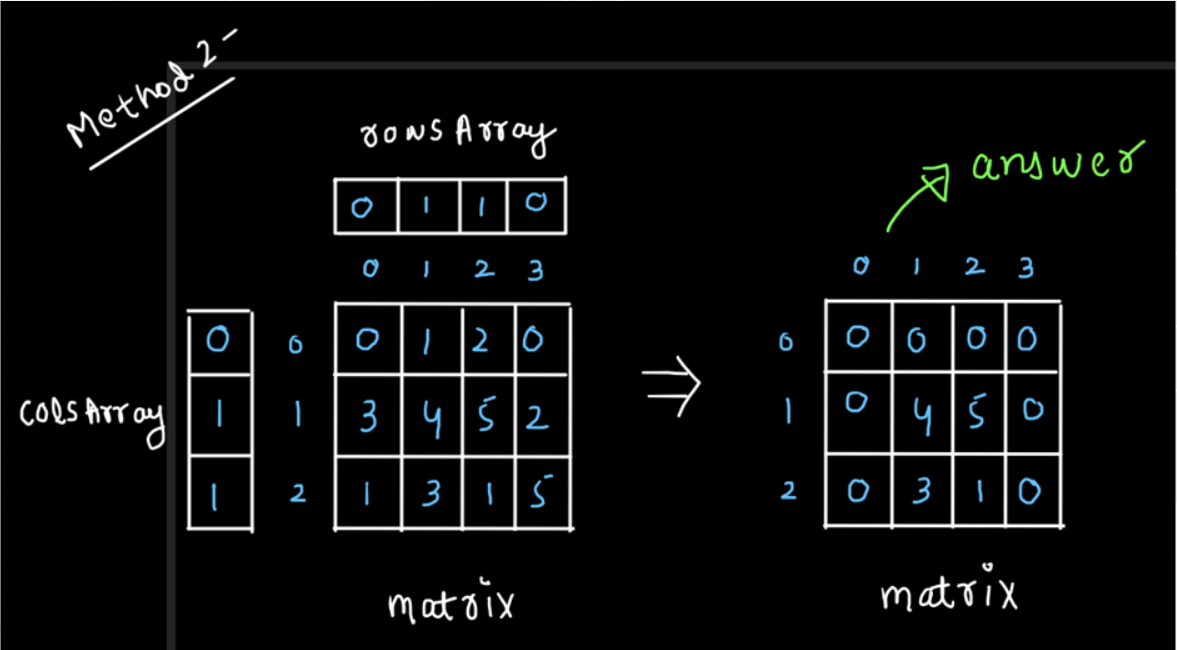
**Refer to**

<https://leetcode.com/problems/set-matrix-zeroes/solutions/2525398/all-approaches-from-brute-force-to-optimal-with-easy-explanation/>

**Method 2:**(Better)

* we can use two separate arrays one for rows (rowsArray) and one for columns (colsArray) and initialize them to 1
* while traversing the given matrix whenever we encounter 0 at (i,j), we will set rowsArray[i]=0 and colsArray[j]=0
* After completion of step 2, again iterate through the matrix and for any (i,j), if rowsArray[i] or colsArray[j] is 0 then update matrix[i][j] to 0.

-**Time:**O(mn), **Space:**O(m+n)



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

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

int rowsArray[]= new int[m];

int colsArray[]= new int[n];

Arrays.fill(rowsArray,1);

Arrays.fill(colsArray,1);

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

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

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

rowsArray[i]=0;

colsArray[j]=0;

}

}

}

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

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

if(rowsArray[i]==0 || colsArray[j]==0)

matrix[i][j]=0;

}

}

}

**Solution 3: Optimize to inplace, no extra space needed (10 min)**

class Solution {

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

int m = matrix.length;

int n = matrix[0].length;

// First we will traverse the 0th row and 0th column

// of the given matrix and if we encounter any 0 then

// we will set the first\_row\_zero/first\_col\_zero

// variable to true which indicates that the 0th

// row/0th column of the given matrix will become 0

boolean first\_row\_zero = false;

boolean first\_col\_zero = false;

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

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

first\_col\_zero = true;

}

}

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

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

first\_row\_zero = true;

}

}

// Next we will traverse the remaining matrix except 0th

// row and 0th column and if we encounter any 0, we will

// make the corresponding row no. and column no. equal

// to 0 in the 0th column and 0th row respectively

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

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

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

matrix[i][0] = 0;

matrix[0][j] = 0;

}

}

}

// Now we will update the values of the matrix except

// first row and first column to 0 if matrix[i][0] = 0

// or matrix[0][j] = 0 for any (i,j).

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

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

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

matrix[i][j] = 0;

}

}

}

// Finally we will traverse the 0th row and 0th column and

// if we find any 0, we will make the whole row and whole

// column equal to 0

if(first\_row\_zero) {

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

matrix[0][j] = 0;

}

}

if(first\_col\_zero) {

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

matrix[i][0] = 0;

}

}

}

}

Time Complexity: O(mn)

Space Complexity: O(1)

**Refer to**

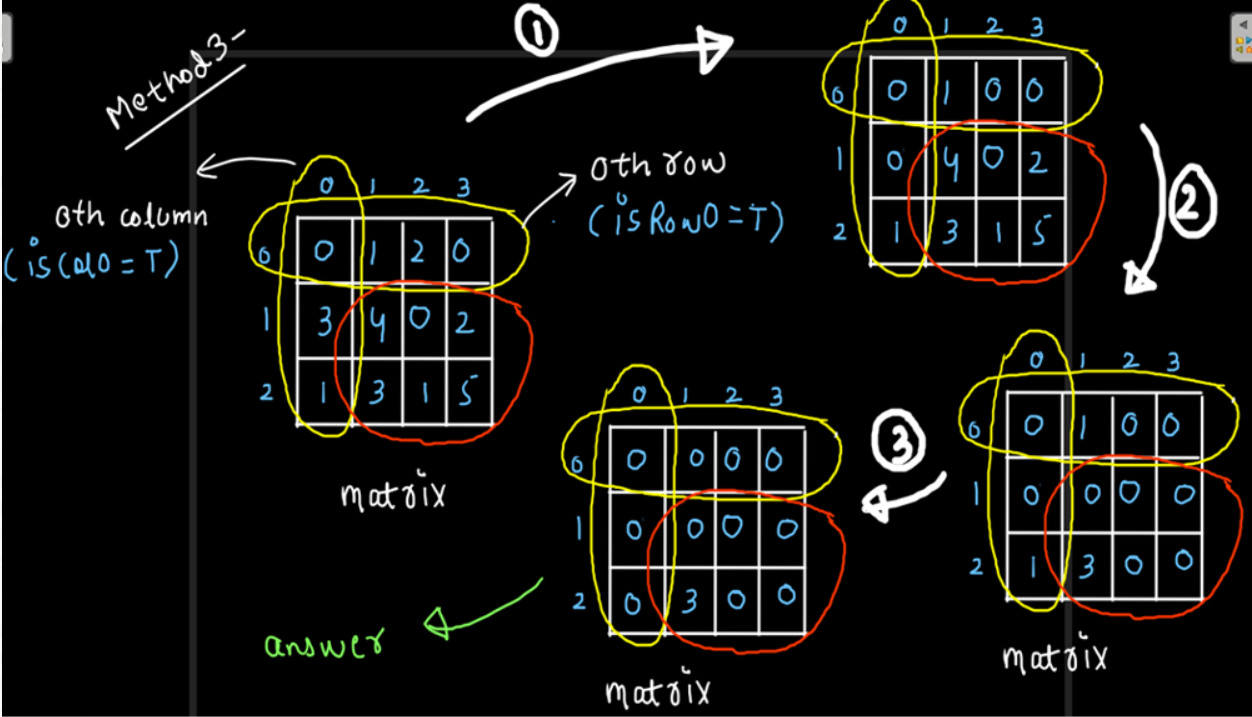
<https://leetcode.com/problems/set-matrix-zeroes/solutions/2525398/all-approaches-from-brute-force-to-optimal-with-easy-explanation/>

**Method 3:**(Optimal)

-we can use the 0th row and 0th column of the given matrix itself instead of using two separate arrays

* first we will traverse the 0th row and 0th column of the given matrix and if we encounter any 0 then we will set the isRow0/isCol0 variable to true which indicates that the 0th row/0th column of the given matrix will become 0
* next we will traverse the remaining matrix except 0th row and 0th column and if we encounter any 0, we will make the corresponding row no. and column no. equal to 0 in the 0th column and 0th row respectively
* Now we will update the values of the matrix except first row and first column to 0 if matrix[i][0]=0 or matrix[0][j]=0 for any (i,j).
* finally we will traverse the 0th row and 0th column and if we find any 0, we will make the whole row and whole column equal to 0

-**Time:**O(mn), **Space:**O(1)



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

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

boolean isRow0=false, isCol0=false;

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

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

isRow0=true;

}

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

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

isCol0=true;

}

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

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

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

matrix[i][0]=0;

matrix[0][j]=0;

}

}

}

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

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

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

matrix[i][j]=0;

}

}

if(isRow0){

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

matrix[0][j]=0;

}

if(isCol0){

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

matrix[i][0]=0;

}

}