# 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, and return *the matrix*.

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?

Logic 1:

* Use flags for rows and columns where there is 0;

Logic 2:

* Take a bool variable by name ‘firstColumn’. Traverse the matrix from 0th row and 1st column. While traversing…
* If There any 0’s in the 1st column, make this ‘firstColumn’ variable ‘true’.
* While traversing if any cell has the value 0, then set the 0th row element to 0 and 0th column element to 0.
* Now traverse the matrix from bottom. That is, row-1 and column-1 till 1st column(not the 0th column)
* While traversing, if the cell’s corresponding 0th row element is 0 or 0th column element is 0, then make that cell element 0.
* Also if the firstColumn variable is true, then set all the first column cells to 0.

-------------------------------------------------------------Solution 1-------------------------------------------------------------

#include <iostream>

#include <iomanip>

#include <vector>

using namespace *std*;

void display2DMatrix(*vector*<*vector*<int>>& matrix)

{

for (auto i : matrix)

{

for (auto j : i)

{

*cout* << *setw*(2) << j << " ";

}

*cout* << *endl*;

}

*cout* << *endl*;

}

void displayVector(const *vector*<int>& i)

{

for (auto j : i)

{

*cout* << j << " ";

}

*cout* << *endl*;

}

void setZeroes(*vector*<*vector*<int>>& matrix) {

int rows = matrix.*size*();

int cols = matrix[0].*size*();

*vector*<bool> rowFlag(rows);

*vector*<bool> colFlag(cols);

for (auto i = 0, n = 0; i < rows; i++)

{

for (auto j = 0, m = 0; j < cols; j++)

{

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

{

rowFlag[i] = true;

colFlag[j] = true;

}

}

}

for (auto i = 0; i < rows; i++)

if (rowFlag[i] == true)

for (auto j = 0; j < cols; j++)

matrix[i][j] = 0;

for (auto j = 0; j < cols; j++)

if (colFlag[j] == true)

for (auto i = 0; i < rows; i++)

matrix[i][j] = 0;

}

int main(void)

{

*vector*<*vector*<int>> matrix{{0, 1, 2, 0}, {3, 4, 5, 2}, {1, 3, 1, 5}};

setZeroes(matrix);

display2DMatrix(matrix);

return 0;

}

-------------------------------------------------------------Solution 2-------------------------------------------------------------

#include <iostream>

#include <iomanip>

#include <vector>

using namespace *std*;

void display2DMatrix(*vector*<*vector*<int>>& matrix)

{

for (auto i : matrix)

{

for (auto j : i)

{

*cout* << *setw*(2) << j << " ";

}

*cout* << *endl*;

}

*cout* << *endl*;

}

void displayVector(const *vector*<int>& i)

{

for (auto j : i)

{

*cout* << j << " ";

}

*cout* << *endl*;

}

void setZeroes(*vector*<*vector*<int>>& matrix) {

int rows = matrix.*size*();

int cols = matrix[0].*size*();

bool firstCol = false;

for (auto i = 0; i < rows; i++)

{

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

for (auto j = 1; j < cols; j++)

{

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

{

matrix[0][j] = 0;

matrix[i][0] = 0;

}

}

}

for (auto i = rows - 1; i >= 0; i--)

{

for (auto j = cols - 1; j >= 1; j--)

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

matrix[i][j] = 0;

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

}

}

int main(void)

{

*vector*<*vector*<int>> matrix{{0, 1, 2, 0}, {3, 4, 5, 2}, {1, 3, 1, 5}};

setZeroes(matrix);

display2DMatrix(matrix);

return 0;

}