$m = 5$
$n = 4$

	0	1	2	3
0	4	5	7	2
1	3	6	0	4
2	0	8	4	2
3	6	7	0	0
4	0	8	9	4

⇒

	0	1	2	3
0	-1	5	-1	-1
1	-1	-1	0	-1
2	0	-1	-1	-1
3	-1	-1	0	0
4	0	-1	-1	-1

Replace -1 with 0

(I) Brute Force

($O(mn)$) 1) Traverse linearly through each elements of the matrix.

($O(m+n)$) 2) Whenever you get 0 at a[i][j], then make every element of ith row and jth column -1 except zeroes that come in the way.

($O(mn)$) 3) After that iterate over the matrix and replace -1 in the way with 0. And leave zeroes as it is.

$T = O(mn)\,O(m+n)$

- $O(mn)$ → Total no. of elements
- $O(m+n)$ → Time taken for traversing rows and columns for an element

(II) Better Approach

1) r: | 0 | -1 | 0 | 0 | auxiliary array 1 (Initialize with -1)

c:

0	0
1	0
2	0
3	0
4	0

	0	1	2	3
0	4	5	7	2
1	3	6	0	4
2	0	8	4	2
3	6	7	0	0
4	0	8	9	4

auxiliary array 2 (Initialize with -1)

$S = O(m+n)$

$T = O(mn) + O(mn) = O(mn)$

2) Now, iterate over matrix and check for every a[i][j] if either c[i] or r[j] is zero then a[i][j] will be made zero.

III) Best Approach

We will enhance our II approach by reducing space complexity.

1) We will consider 1st row and 1st column as our auxiliary array.

2) We will iterate, if any element in first row is found 0 then we will make a[0][0] as zero.

- If any element in first column is found 0 then we will make the variable "colTop" as 0.
- If any element a[i][j] is 0, then a[i][0] should be made 0 and a[0][j] should also be made zero.

colTop [1]

(Initialize with 1)

1 means there is no 0 in the first column

	0	1	2	3
0	4	5	0	0
1	0	6	0	4
2	0	8	4	2
3	0	7	0	0
4	0	8	9	4

3) Now, iterate from bottom and for a[i][j] if either a[i][0] or a[0][j] is 0 then make a[i][j] = 0

4) For any element in 1st row a[0][j], if

a[0][0] is 0 then make a[0][1] = 0.

And for any element in 1st column, a[i][0], if colTop = 0 then make a[i][0] = 0.

	0	1	2	3
0	0	5	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

5) a[0][0] will be checked at last. If colTop = 0 then make a[0][0] = 0