Naive Solution,

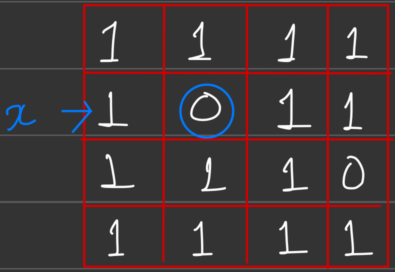
In a matrix, if you find a 0,

- make a pointer, and check value at the pointer should not be 0
- This pointer first moves left of curr_elem, then right, then top, then bottom
- At every point check if the value at pointer is not 0, replace it by -1
- At the end replace all -1's by 0

a =

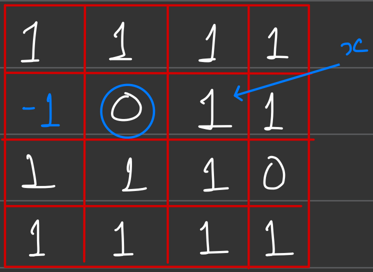
1	1	1	1
1	0	1	1
1	1	1	0
1	1	1	1

```
for (int i=0; i<row_size; i++)
  for (int j=0; j<col-size; j++)
    if( a[i][j] == 0) {
        ∞— Logic here —∞
    }
```



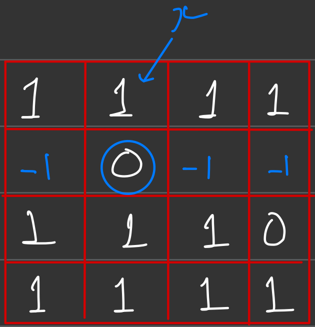
1	1	1	1
1	0	1	1
1	1	1	0
1	1	1	1

```
x = i - 1;
while (x >= 0) {
  if(a[x][j] != 0)
      a[x][j] = -1;
  x--; }
```



1	1	1	1
-1	0	1	1
1	1	1	0
1	1	1	1

```
x = i+1 ;
while (x < row_size) {
   if (a[x][j] != 0)
      a[x][j] = -1;
   x++; }
```



1	1	1	1
-1	0	-1	-1
1	1	1	0
1	1	1	1

```
x = j - 1 ;
while ( x >= 0) {
   if ( a[i][x] != 0)
      a[i][x] = -1;
   x--; }
```

1	-1	1	1
-1	0	-1	-1
1	1	1	0
1	1	1	1

x

```
x = j+1 ;
while ( x < col_size) {
  if (a[i][x] != 0)
    a[i][x] = -1;
x++; }
```

1	-1	1	1
-1	0	-1	-1
1	-1	1	0
1	-1	1	1

→

1	-1	1	1
-1	0	-1	-1
1	-1	1	0
1	-1	1	1

1	-1	1	1
-1	0	-1	-1
-1	-1	-1	0
1	-1	1	1

1	-1	1	-1
-1	0	-1	-1
-1	-1	-1	0
1	-1	1	1

1	-1	1	-1
-1	0	-1	-1
-1	-1	-1	0
1	-1	1	-1

1	-1	1	-1
-1	0	-1	-1
-1	-1	-1	0
1	-1	1	-1

convert all

-1

→

to

0

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

Output :

$$a = \begin{array}{|c|c|c|c|}\hline 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}$$

Time complexity : $O\left(\underbrace{(N \times M)}_{\text{Traversal for each element}} * \underbrace{(N+M)}_{\text{Traversal for } x \text{ pointer.}}\right)$

Space complexity : $O(1)$

Optimised Solution

Create two sets, to add every i, j with value 0

traverse again, and check if index is either in row_set or col_set, if yes, make that element 0

$$a = \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

row_set = { }

col_set = { }

$$\begin{array}{c|c|c|c|c|} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 & 1 \\ \hline 2 & 1 & 1 & 1 & 0 \\ \hline 3 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

row_set = { 1 }

col_set = { 1 }

```
if (a[i][j] == 0) {
    row_set.insert(i);
    col_set.insert(j);
}
```

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 1 & 1 & 1 & \boxed{0} \\ 3 & 1 & 1 & 1 & 1 \end{array} \longrightarrow \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

row_set = { 1,2 } row_set = { 1,2 }

col_set = { 1,3 } col_set = { 1,3 }

- Traverse the matrix , if curr-elem != 0 check if it's index is in either of the sets

```
if (a[i][j] != 0)
   if (row_set.find(i) != row_set.end() ||
           col_set.find(j) != col_set.end())
        a[i][j] = 0
```

$$\begin{array}{c|cccc} & 0 & \boxed{1} & 2 & \boxed{3} \\ \hline 0 & 1 & 0 & 1 & 0 \\ \boxed{1} & 0 & 0 & 0 & 0 \\ \boxed{2} & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 1 & 0 \end{array}$$

Output $a = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$

Time complexity : $O(n \times m)$

Space complexit: $O(n + m)$