Analyzing the Contraints

9:05pm

$\boxed{10^9} \rightarrow$ 1 sec

10^{10}

Constraint	T.C
$n <= 10^6$	$O(N)$, $O(N \log N)$
$n <= 20$	$O(N!)$, $O(2^N)$ →
$n <= 10^{10}$	$O(\log N)$, $O(\sqrt{N})$

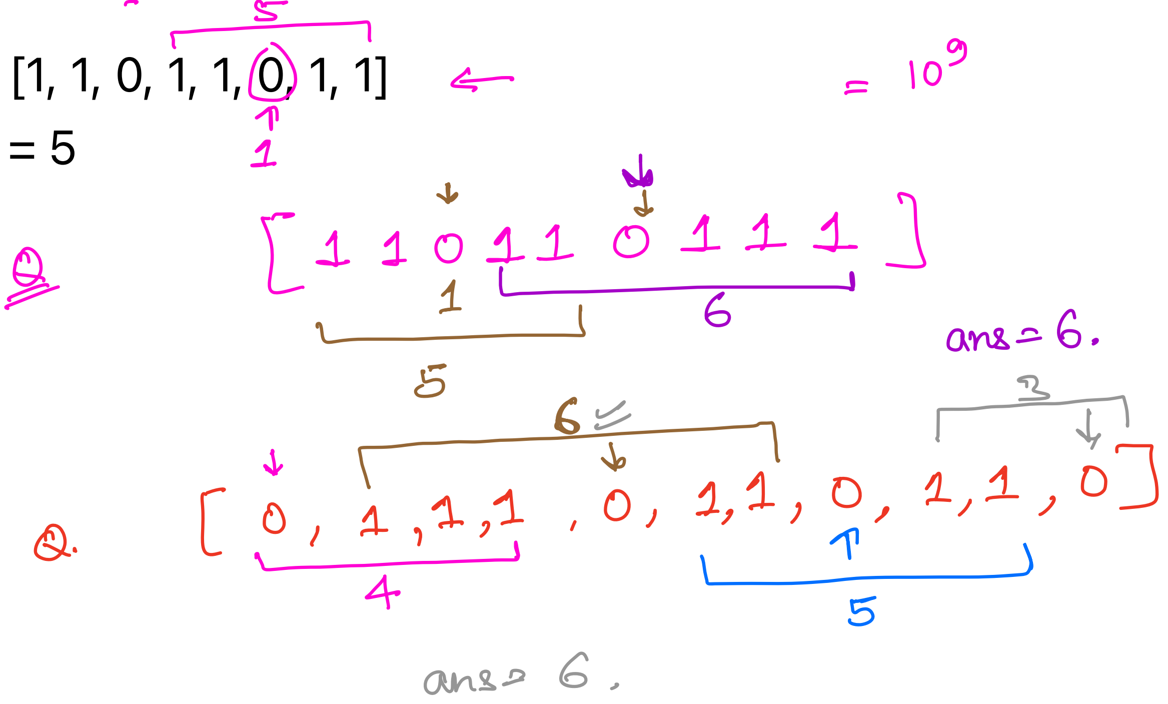
Generic Guildnes.

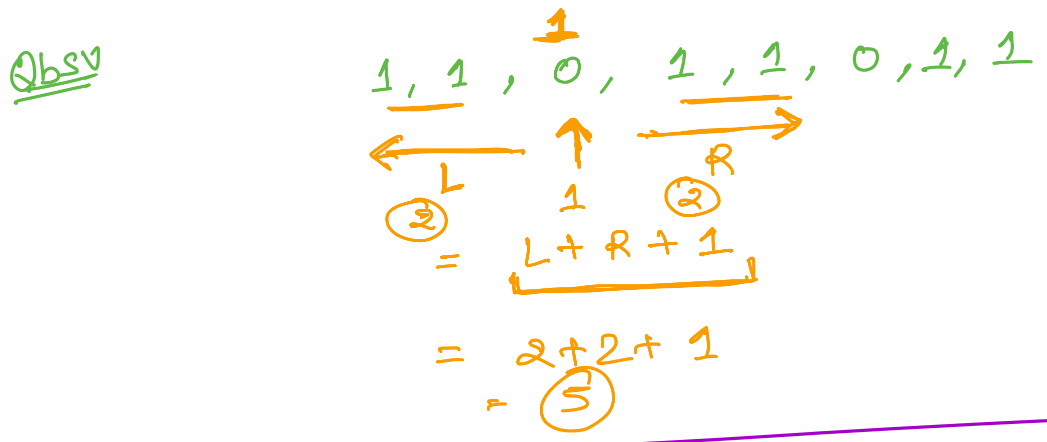
Amazon, Adobe, Microsoft.

Q Given an array of 1's and 0's, you are allowed to replace only one 0 with 1. Find the maximum number of consecutive 1's that can be obtained after making the replacement.

Input = [1, 1, 0, 1, 1, 0, 1, 1] $= 10^9$

Output = 5

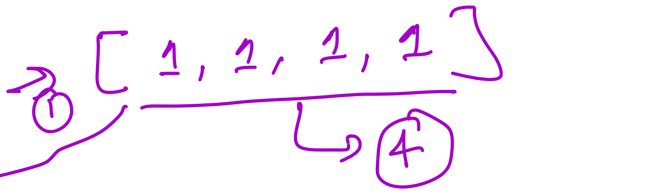


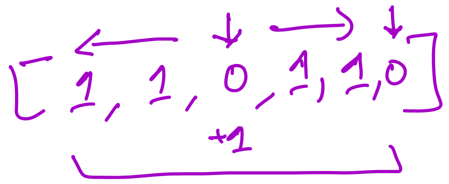


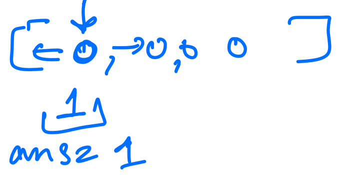
```
int arr[];
int n;
int max_ones = 0;

for (i=0; i<n; i++)      O(N)
{
    if (arr[i] == 0)  ✓
    {
        int left=0, right=0;
```

Count the no. of one if == length array return the count.

checking 0 in array





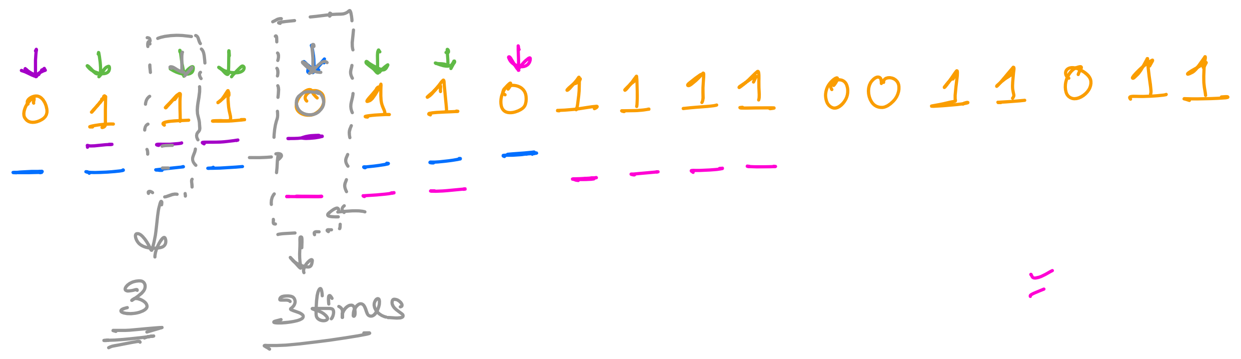


```
k = i-1
j = i+1

while (j < n && arr[j] == 1) {
    right ++;
    j ++;
}
while ( k >= 0 && arr(k) == 1) {
    left ++;
    k --;
}
max_one = max (max_one,
                left + right + 1);
}

}
return max_one;
```

iteration toward righ and countings. 1

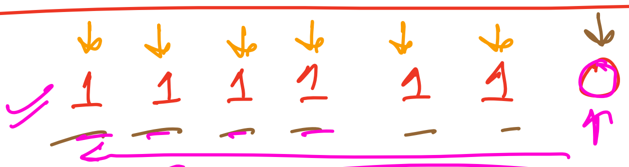
iterating toward left and count

max



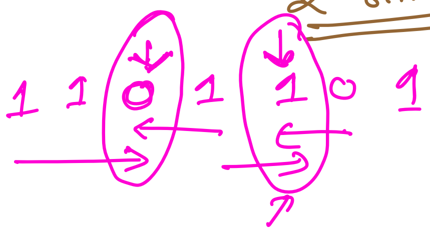
→ iterating 3 time. $O(3N)$
$= O(N)$

S.C. $O(1)$



2 times

$O(2N)$
$= O(N)$

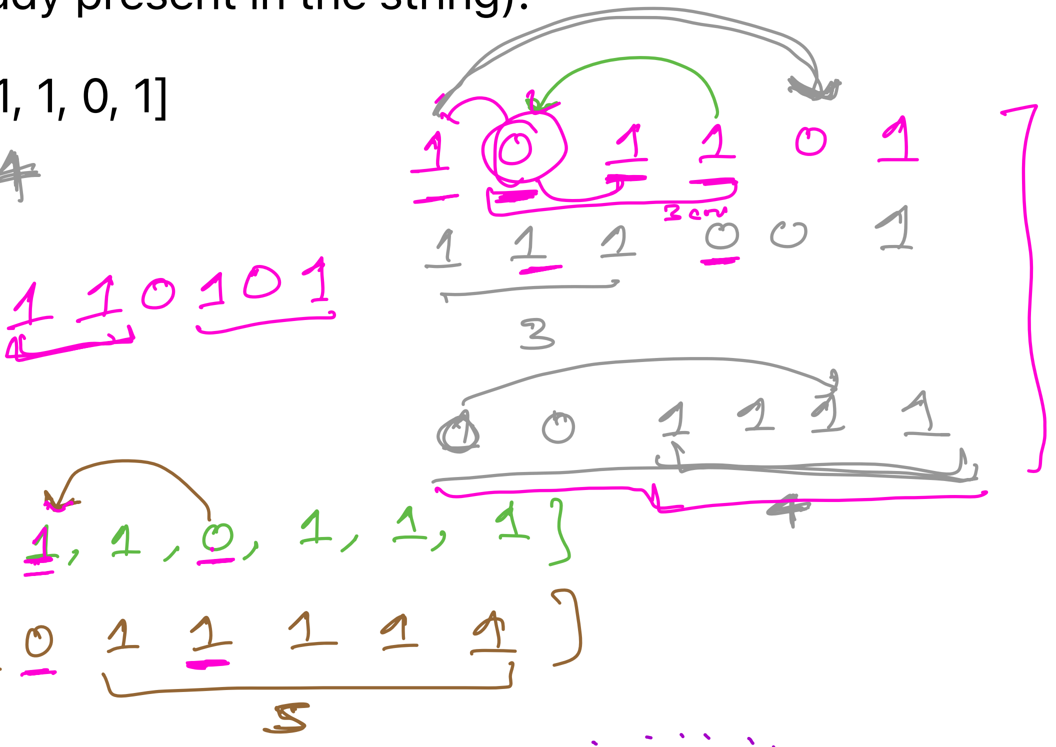


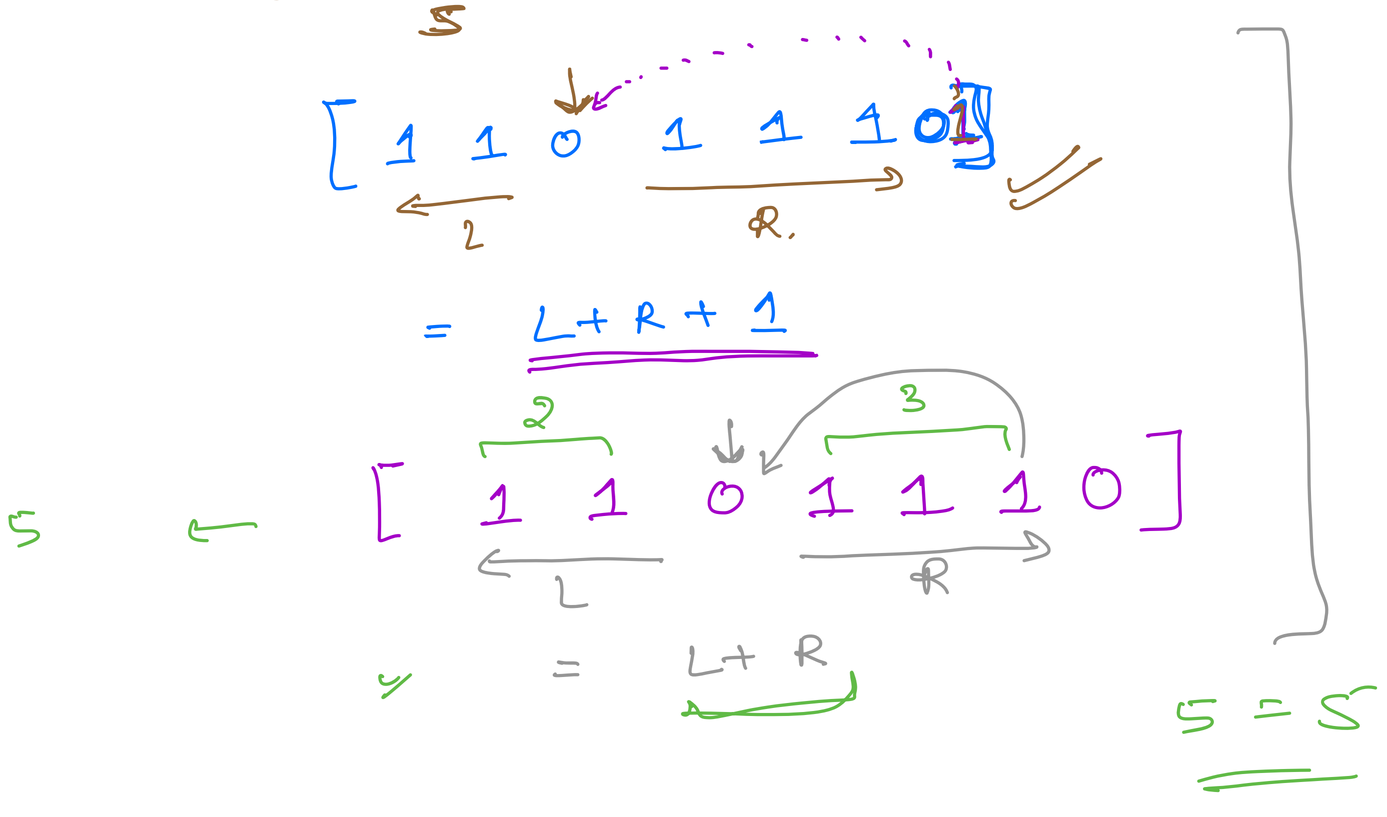
Variation Amazon, Direct-i, Microsoft, Adobe.

Given an array of 1's and 0's, find the maximum number of consecutive 1's that can be obtained by SWAPPING at most one 0 with 1(already present in the string).

Input: [1, 0, 1, 1, 0, 1]

Output: ~~5~~ 4





int max_one = 0

```
int  arr[]
int  n
int  totalOne = 0
for ( i=0;  i<n;  i++ )
     if  arr[i] == 1 :
          totalone++;          [1, 1, 1, 1]
```

count of one.

```
if ( total_one == n )
     return n;

for ( i=0;  i<n;  i++ )
{
     if ( arr[i] == 0 )
     {
          int  left = 0
               right = 0
          k = i-1
          j = i+1

          while (j<n && arr[j]==1) {
               right++;
               j++;
          }
          while ( k>=0 && arr[k]==1) {
               left++;
               k--;
          }
          if ( left+right == no.of one )
               {  max_one = max( max_one, (l+r) )
               }
          else
               max_one  max (max_one , l+r+1)
               }
}
```

T.C	O(N)
S.C	O(1)

10:10pm

Q Given an array of N integers, find the majority element.

The majority element is the element that occurs more than n/2 times where n is size of the array.

A[ ] = { 2, 1, 4 } $\longrightarrow$ $2 \to 1$ $1 \to 1$ $4 \to 1$

Ans = No Majority element ✓ - 1

A[ ] = { 3, 4, 3, 2, 4, 4, 4, 4} $\longrightarrow$ $3 \to 2$ $\boxed{4 = 5}$ $2 \to 1$

$8/2 = 4$

Ans = 4

3, 4, 3, 6, 1, 3, 2, 5, 3, 3, 3

$\boxed{3 \to 6}$ ✓

$4 \to 1$

$6 \to 1$

$2 \to 1$

$5 \to 1$

$11/2 = 5$

4, 6, 5, 3, 4, 5, 6, 4, 4, 4

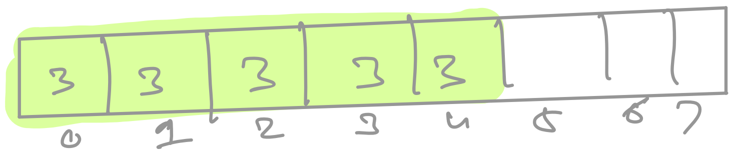
$4 \to 5$

$6 \to 2$ $\Rightarrow$ No Majority element.

$5 \to 2$

$3 - 1$

$\frac{10}{2} = 5$



C_1 C_2

$> n/2$ $> n/2$

only 1 majority element.

$\underbrace{C_1 + C_2}$ $\geq n/2 + n/2$

$\boxed{C_1 + C_2}$ $\underbrace{\geq n}$

$\cancel{2}$ $\cancel{1}$ 1 1 1 $\cancel{8}$ $\cancel{2}$

$\boxed{\begin{array}{l} \text{maj} = 1 \\ \text{count} = 1 \end{array}}$

Brute Force

① Sorting - calculate count

T.C = O(nlogn)

S.C = O(1)

two loop and count

T.C : O(N²)

S.C : O(1)

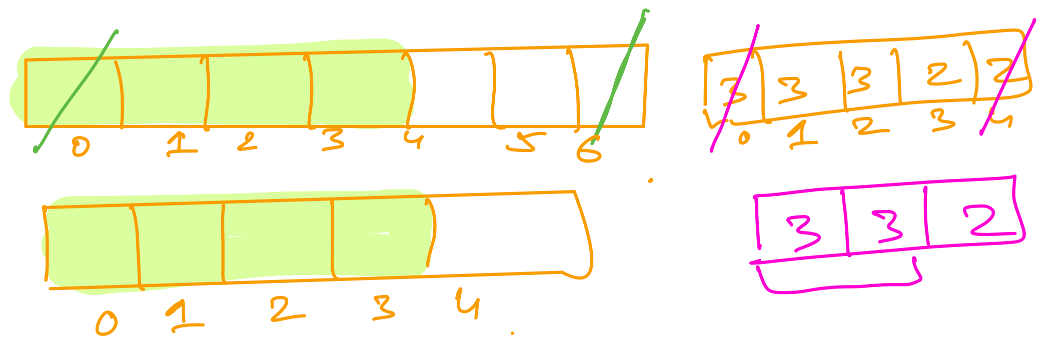
③ Hash Map freq check for > n/2

T.C = O(n)

S.C = O(n)

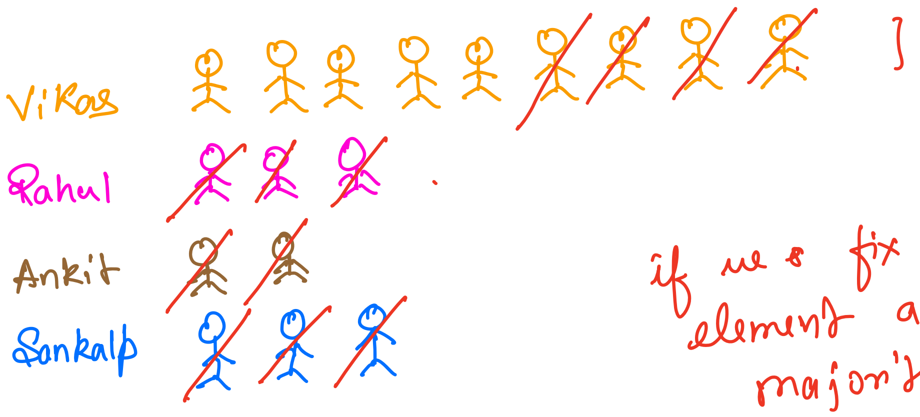
Optimize it

③



Obs 2

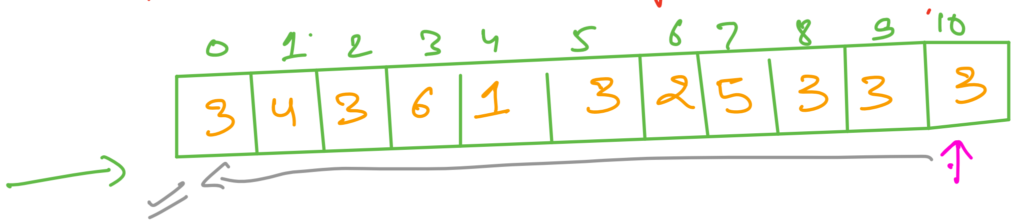
Elections



if we fix two distinct element and remove majority dosent.

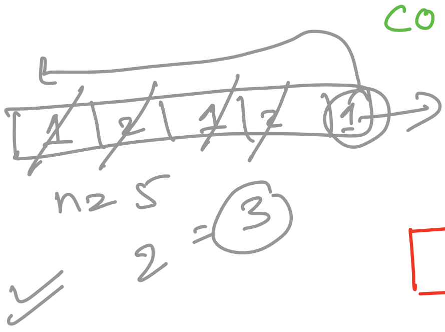
Remove	Orange	Pink	Brown	Blue	Winner
1 Vikas & 1 Rahul	8	2	2	3	orange
1 Vikas & 1 Sankalp	7	2	2	2	orange
1 Vikas & 1 Ankit	6	2	1	2	orange.
1 Rahul & 1 Ankit	6	1	0	2	orange.
1 Rahul & 1 Sankalp	6	0	0	1	orange.
1 Vikas & 1 Sankalp	5	0	0	0	orange

Moore's Voting algorithm



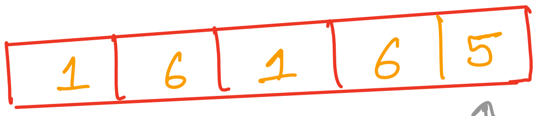
majority element = ~~3~~ ~~3~~ ~~1~~ ~~2~~ 3

count = 0 ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~2~~ 3



ans = 3

→ NO Match

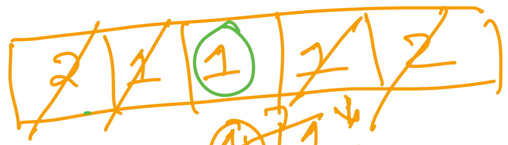


majority element : ~~1~~ ~~1~~ 5

Count : ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

$1 > n/2 = 1$

frequency of the majority
if it is $> n/2$
then ans =
else -1



majority 1 1

Count = 1

majority 1

count = 3

$n/2$

```
int arr[]
int n
int major_index = 0
int count = 1
for ( i = 1 ; i < n ; i++ )
{
        if (count == 0 )
        {
            maj_index = i;
            count = 1
        }
        else
            if ( arr[maj_index] == a[i])
            {
                count ++;
            }
```

T.C. O(N)

S.C O(1)

```
            else {
                    count--;
            }
    }
}

int count = 0
for ( i=0; i<n; i++ )
{
    if ( arr[i] == arr[majority_index] )
            count++
}
    if (count > n/2)
            return majority index.
    else
            return -1

}
```

Que

You are given a 2D integer matrix A, make all the elements in a row or column zero if the A[i][j] = 0. Specifically, make entire ith row

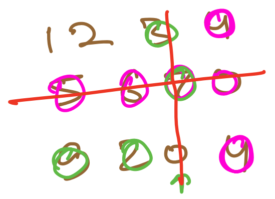
⇒ [Non of the element is negative]

Input:

[1,2,3,4]

[5,6,7,0]

[9,2,0,4]



$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Output:

[1,2,0,0]

[0,0,0,0]

[0,0,0,0]

$$\Rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 9 & 2 & 0 & 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 2 & 0 & 0 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 2 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 3 \\ 0 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$\boxed{\text{INT\_MIN}}$

T.C. $O(N^2)$

S.C : $O(1)$

$$\Rightarrow \begin{bmatrix} 1 & 2 & -3 & -4 \\ -5 & -6 & -7 & 0 \\ -9 & -2 & 0 & -4 \end{bmatrix}$$

H.W

$$2^{nd}\text{ it} \Rightarrow \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \checkmark$$

Algo $\longrightarrow$

① find zero and make every element in the row and colum of that zero negative.

$O(N^2)$

$+$

② traverse the matrix and convert all the negative to zero.

$O(N^2)$