# 2-D Arrays

**Question:-** Rows & column are sorted in Matrix, Find out whether element K is present or not.

Ex:-
$$\begin{bmatrix} -5 & -2 & 1 & 13 \\ -4 & 0 & 3 & 14 \\ -3 & 2 & 6 & 18 \end{bmatrix}$$

K = 13 → True
K = 2 → True
K = 15 → False

**Quiz!:-** what is the brute force approach & the time complexity of it?

**Brute force**

↳ Iterate over Matrix & Search

TC - O(N*M)
SC - O(1)

↳ **OPTIMIZATION**

**Observation:-**

① Somehow we need to use Sorted rows & columns.

② Start from a cell from where you can decide where to move.

Ex:-
$$\begin{bmatrix} -5 & -2 & 1 & 13 \\ -4 & 0 & 3 & 14 \\ -3 & 2 & 6 & 18 \end{bmatrix}$$

K=9

K=2

③ Base on $2^N$ obs. we can conclude that we will start from top right.

Quiz 2:- Say we are at 1 & want to find 0, where should we move?

Ans

Ex:-

| | | | |
|---|---|---|---|
| -5 | -2 | 1 | 13 |
| -4 | 0 | 3 | 14 |
| -3 | 2 | 6 | 18 |

## PSEUDO CODE

```
i = 0 , j = m-1   // Top right.
while ( j >= 0 && i < n ) {
    if ( arr[i][j] == k ) {
        return True
    } else if ( arr[i][j] < k ) {
        i++;
    } else {
        j--;
    }
}
```

TC → O(N+M)

SC → O(1)

**Question:** Given a binary sorted matrix A of size N×N. Find the row with the maximum no. of 1.

Note:- ① If two rows have the maximum no. of 1 then return the row which has a lower index.

② Assume each row to sorted by values.

Ex1:-

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \rightarrow \text{Ans} = 0$$

Ex2:-

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

**Brute force**

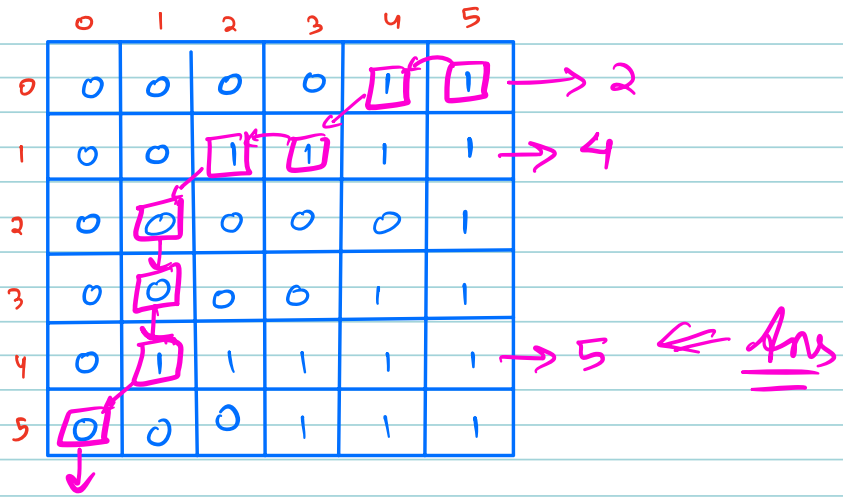
↳ Travel each row & count 1's

TC → $O(N^2)$

SC → $O(1)$

↳ **OPTIMIZED**

**IDEA:-** Since rows are sorted. So, we need use this property.




## Observations

① The earliest I get more ones the better it is. So I need to start from first row.

② what if I am on 1?
- → If there is ① on left, move left.
- → If there is ⓪ on left, move (i+1, j-1) call

③ what if I'm on 0?
- → I have to move downward.

## PSEUDO CODE

```
i = 0 , j = M-1 , ans = 0;
while ( i<n && j>=0 ){
    while ( j>=0 && arr[i][j] == 1){
        j--;
        ans = i;
    }
    i++;
```

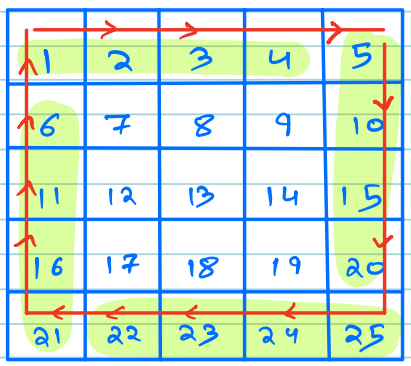
}

return ans;

TC - O(N)
SC - O(1)

**Question :- Print Boundary Elements**

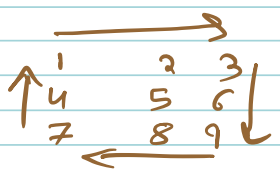
Given an Matrix of N x N i.e. Mat[N][N], print boundary elements in clockwise direction.




output :- 1, 2, 3, 4, 5, 10, 15, 20, 25, 24, 23, 22, 21, 16, 11, 6

Quiz3 Given N & matrix mat, select the correct order of boundary elements traversed in clockwise direction.

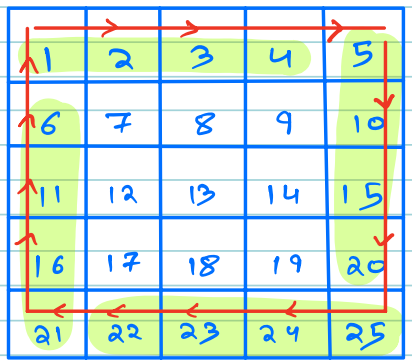
Mat :




output :- 1 2 3 6 9 8 7 4

IDEA :- ① I need to use 4 loops.

② Each loop will iterate n-1 time.




PSEUDO CODE

```
i=0 , j=0
// Print first row.
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    j++;
}
// i=0 , j=N-1
// Print last column.
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    i++;
}
// i=N-1 , j=N-1
```

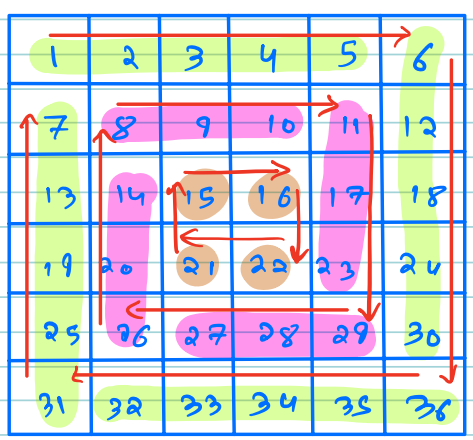
```
// Print last row.
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    j--;
}

// i = N-1 , j = 0

// Print first column
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    i--;
}

// i = 0 , j = 0
```

TC → O(N)

SC → O(1)

**Question 4** :- Given an matrix of N×N i.e. Mat[N][N]. Print elements in spiral order in clockwise direction

Ex :-




output :- 1, 2, 3, 4, 5, 6, 12, 18, 24, 30, 36, 35, 34, 33, 32, 31, 25, 19, 13, 7, 8, 9, 10, 11, 17, 23, 29, 28, 27, 26, 20, 14, 15, 16, 22, 21

## observations

① It is similar to previous question with one addition of multiple boundries.

② The starting point is following this pattern

(0,0) → (1,1) → (2,2) .....

③ The no of iterations is decreasing by 2

OR → (6×6) → (4×4) → (2×2)

④ We can keep a check N>1

**EDGE Case :-** If the matrix is 5×5, then

(5×5) → (3×3) → (1×1)

So, if we run for N=1, No loop will run, hence we can handle seperately.

Code :- if (N==1) { print (A[i][j]) }

## PSEUDO CODE

```
i:0 , j=0;
while (N>1) {
    // Print first row.
    for (k=1 ; k<N ; k++) {
        Print (A[i][j]);
        j++;
    }
```

```
// i=0 , j = N-1
// Print last column.
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    i++;
}

// i= N-1 , j=N-1
// Print last row.
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    j--;
}

// i = N-1 , j=0

// Print first column
for (k=1 ; k<N ; k++) {
    Print (A[i][j]);
    i--;
}

  i++;
  j++;
  N -= 2;
}

if (N==1) { print (A[i][j]); }
```

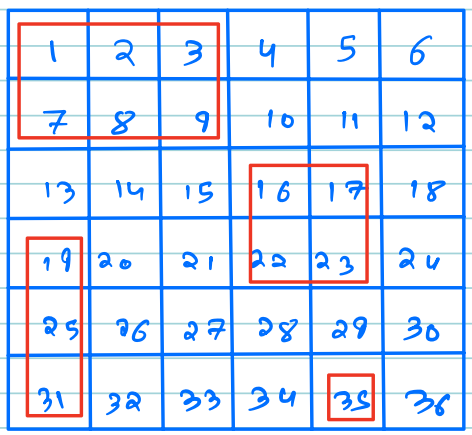
TC → O(N * N)

SC → O(1)

# SUBMATRIX

↳ Same as how a subarray is continuous part of an array, a Submatrix is continuous sub-matrix of a matrix.

→ Sub-matrix is nothing its just a rectangle.




Question :- To identify Rectangle / Sub-matrix how many coordinates I need?

Ans : 2 → Top Left && bottom right

OR

Top Right && bottom Left.

**Question :-** Given a matrix of N rows & M columns determine the sum of all the possible submatrices.

Ex :-

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 4 | 9 | 6 |
| 1 | 5 | -1 | 2 |

All the Submatrices

| | | | | | |
|---|---|---|---|---|---|
| [4] → 4 | [4,9] → 13 | [4,9,6] → 19 | [4 / 5] → 9 | [4 9 / 5 -1] → 17 | [4 9 6 / 5 -1 2] ↓ 25 |
| [9] → 9 | [9,6] → 15 | [5,-1,2] → 6 | [9 / -1] → 8 | [9 6 / -1 2] → 16 | |
| [6] → 6 | [5,-1] → 4 | | [6 / 2] → 8 | | |
| [5] → 5 | [-1,2] → 1 | | | | |
| [-1] → -1 | | | | | |
| [2] → 2 | | | | | |

Total = 166 ← Ans

Brute force

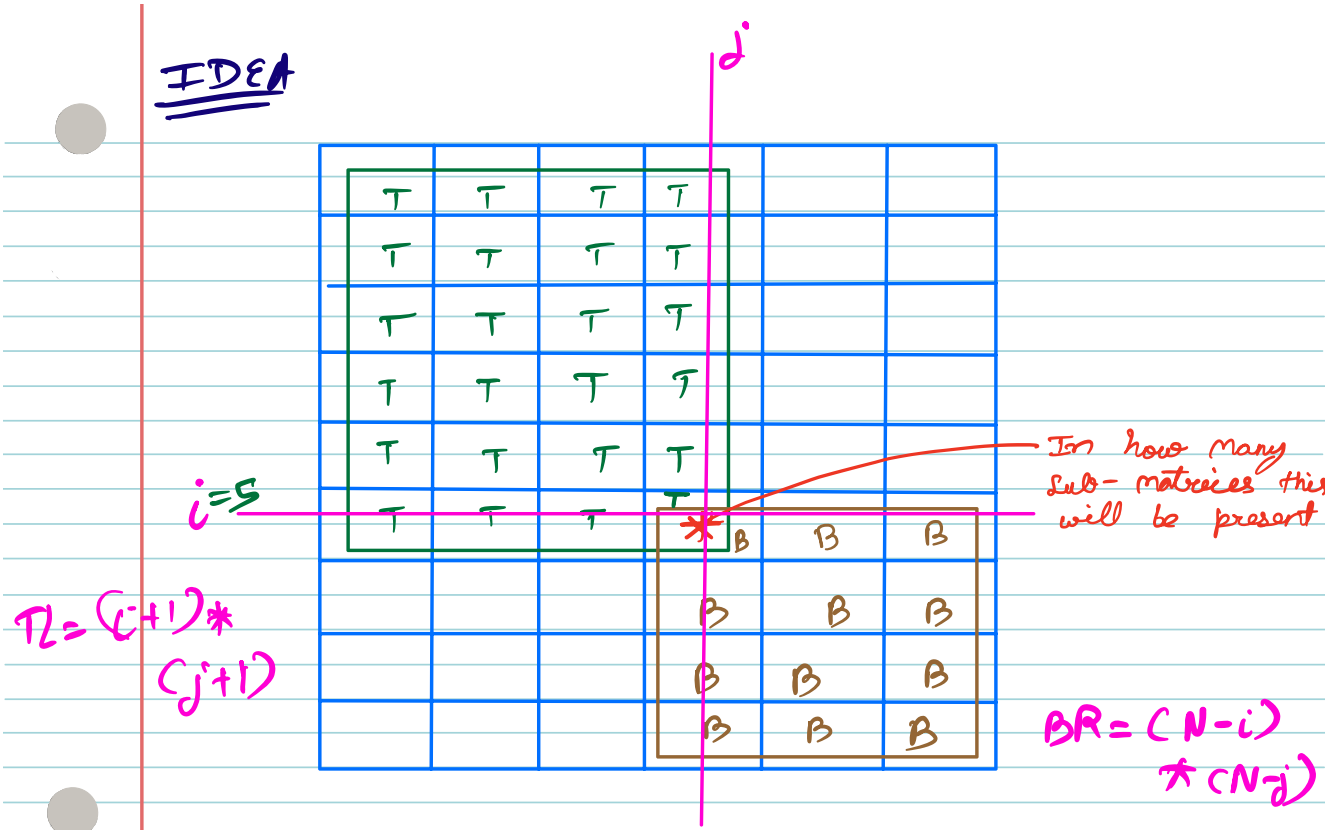
↳ Find all the Sub-Matrices & calculate Sum

↳ BAD IDEA

↳ OPTIMIZATION

↳ Similar to Sub-array questions, we can find contribution of each element.

## IDEA



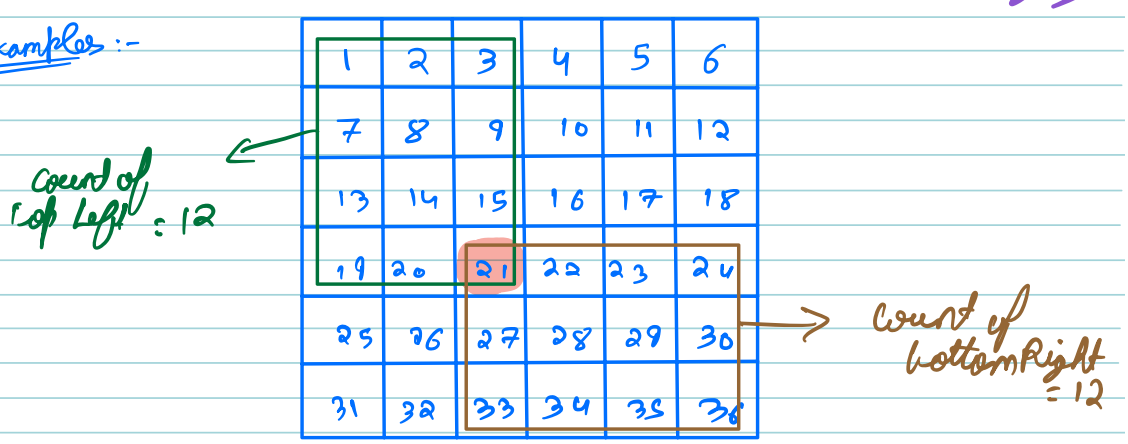

## Observations

① We Need to Count Top Left & bottom Right.

② Total No. of Sub-matrices where (*) will be there is:-

(Count of Top Left) * (Count of Bottom right)

Examples :-



Total Submatrices = 12 x 12
= 144

Generically

Given (i, j)

Count of Top Left = (i+1) * (j+1)

Count of Bottom right = (N-i) * (N-j)

Submatrices where (i, j) will be here = TL x BR
= (i+1)(j+1) * (N-i)(N-j)

Quiz 4 :- In a Matrix of 4*5, in how many sub-matrices (1, 2) is part of?

N = 4
M = 5

i = 1
j = 2

Ans = (i+1)(j+1) * (N-i)(M-j)
= (2 * 3) * (3)(3)
= 6 * 9
= 54

PSEUDO CODE :-

```
total = 0
for (i = 0; i < N; i++){
    for (j=0; j < M; j++) {
        top_left = (i+1) * (j+1)
        bottom_right = (N-i)*(M-j);
        contri = A[i][j] * top_left * bottom_right
        total += contri
    }
}
return total;
```

TC — O( N*M )

SC — O( 1 )