SCALER

2D Matrices

TABLE OF CONTENTS

Search in row-wise & column-wise sorted 2D array

each row ⇒ sorted asc
each column ⇒ sorted asc

$$\begin{bmatrix} -5 & -2 & 1 & 13 \\ -4 & 0 & 3 & 14 \\ -3 & 2 & 5 & 18 \\ 2 & 6 & 10 & 20 \end{bmatrix}_{N * M}$$

Search (6) → true

Search (15) → false

BF Idea

Iterate on whole matrix & check.

TC: O(NM)

Observation

(0, m-1)

search for 6

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} -5 & -2 & 1 & 13 \\ -4 & 0 & 3 & 14 \\ -3 & 2 & 5 & 18 \\ 2 & 6 & 10 & 20 \end{bmatrix} \end{array}$$

if cur_elem > k
move left

if cur_elem < k
move down

if cur_elem = k
found it!!!

< / > Code

```
i = 0          j = m-1
while ( i < N && j >= 0) {
    if (arr[i][j] == k)
        return true
    else if ( arr[i][j] > k )
        j--
    else
        i++
}
return false
```

- If there are multiple K's, return smallest value of i+1009*j

arr[i][j]=k

0, m-1

	0	1	2	3	4
0	-5	-2	0	7	8
1	0	2	5	10	20
2	3	15	15	15	21
3	4	15	20	30	40

Row with maximum number of 1's

Given a binary sorted matrix A of size N*N. Find the row with the maximum number of 1's [Only rows are sorted]

ans = 0th row

```
          0   1   2
A = [  0 [ 0 , 1 , 1 ]
       1 [ 0 , 0 , 1 ]
       2 [ 0 , 1 , 1 ]  ]
```

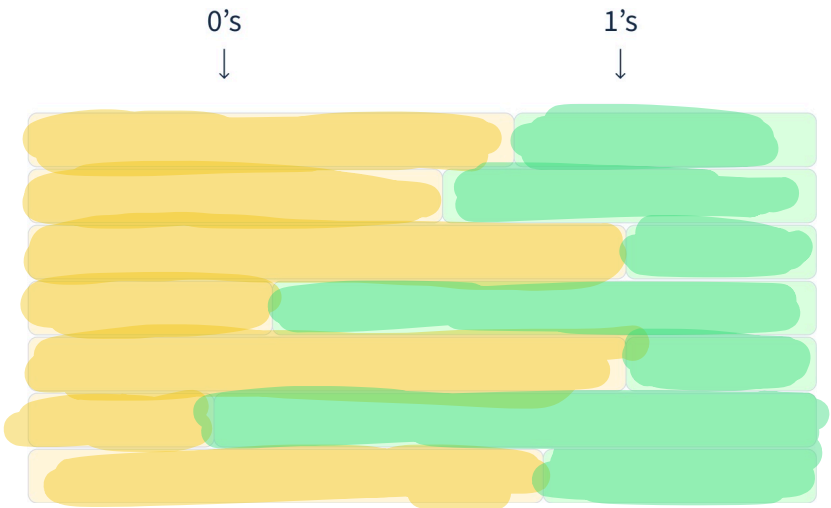
```
          0   1   2   3
A = [  0 [ 0 , 0 , 0 , 0 ]
       1 [ 0 , 0 , 0 , 1 ]
       2 [ 0 , 0 , 1 , 1 ]
       3 [ 0 , 1 , 1 , 1 ]  ]
```

Idea

Count 1's in each row

TC: O(NM)

Observation



m → left

n → down

n + m

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

</ > Code

```
i = 0          j = m-1          ans = 0
while( i < N && j >= 0 ){
    while( j >= 0 && ar[i][j] == 1 ){
        j--
        ans = i
    }
    i++
}
return ans
```

TC: $(N+M)$

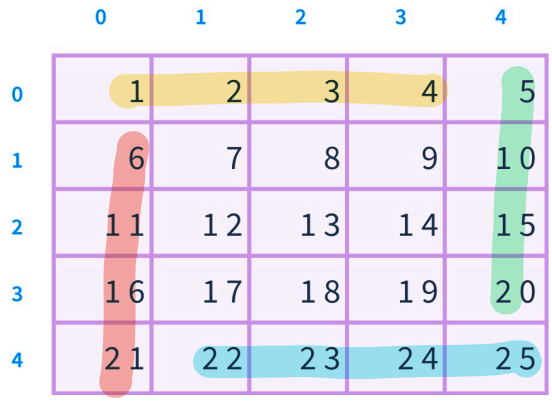
0 0 1 1

1 0 1 1

Print Boundary Elements in clockwise direction

mat[N][N]

mat[5][5]



15

o/p → [1 , 2 , 3 , 4 , 5 , 10 , 20 , 25 , 24 , 23 , 22 , 21 , 16 , 11 , 6]

1

Idea

1) Print N-1 elems from row 0
2) Print N-1 elem from last col
3) Print n-1 elem from last row
4) Print n-1 elem from first col

< / > Code

```
Void printBoundary( arr[N][N]) {
    i = 0                          j = 0
    for ( k : 0 → n-2 ) {
        |       print (arr[i][j])
        |       j++
    }
    for ( k : 0 → n-2 ) {
        |       print (arr[i][j])
        |       i++
    }
    for ( k : 0 → n-2 ) {
        |       print (arr[i][j])
        |       j--
}   }
    for ( k : 0 → n-2 ) {
        |       print (arr[i][j])
        |       i--
```

TC : O(N)

Spiral Matrix

mat[N][N]

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

o/p → [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]

Idea

Quiz :

$$\begin{bmatrix} 13 & 14 & 12 & 8 \\ 9 & 1 & 2 & 7 \\ 0 & 4 & 3 & 0 \\ 10 & 5 & 6 & 11 \end{bmatrix}$$

o/p → 13 14 12 8 7 0 11 6 5 10

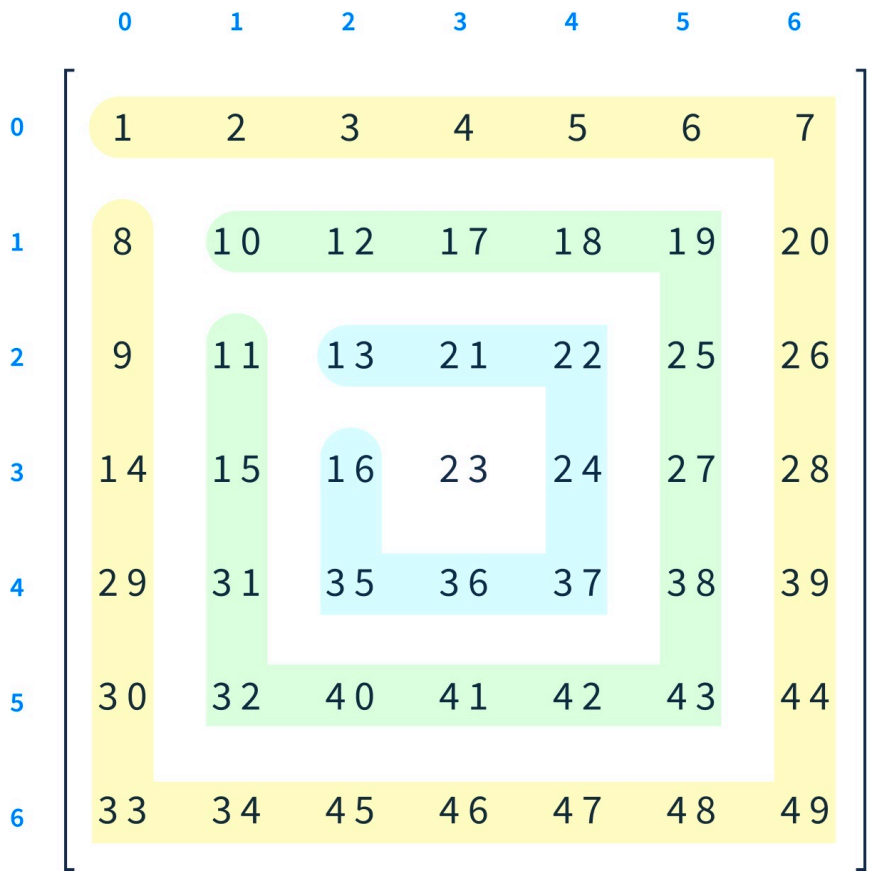
0 9 1 2 3 4

1x1 . 10

```
i=0   j=0
while (N>1) {
    for (k : 0 → n-2) {
        print (arr[i][j])
        j++
    }
    for (k : 0 → n-2) {
        print (arr[i][j])
        i++
    }
    for (k : 0 → n-2) {
        print (arr[i][j])
        j--
    }
    for (k : 0 → n-2) {
        print arr[i][j]
        i--
    }
    i++   j++   n-=2
}
```

```
if (N == 1)
    print (ele[i][j])
```

TC: N^2

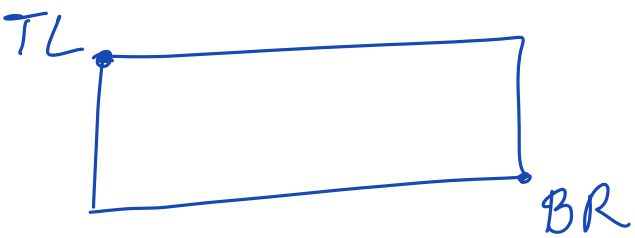


Sub - Matrices

- Contiguous part of a matrix

1, 1 2, 3

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

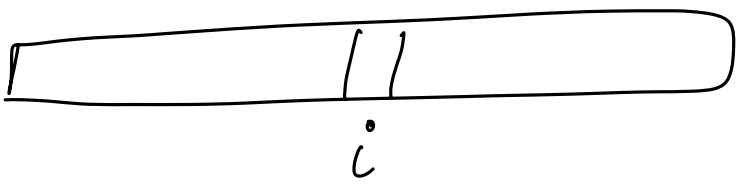


< *Question* > : Given mat[N] [M]. Find sum of all sub-matrix sums.

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

[Brut Force Approach] →

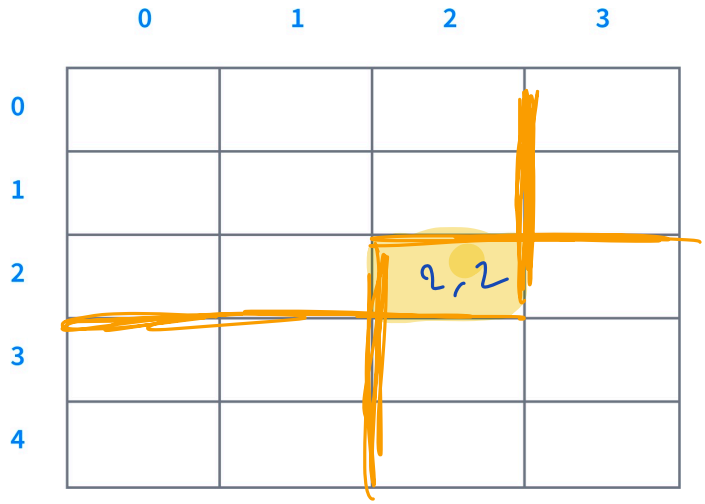
Iterate over all submatrices & calculate sum for each.



Idea

Contribution Technique

N = 5 M = 4



TL = 9

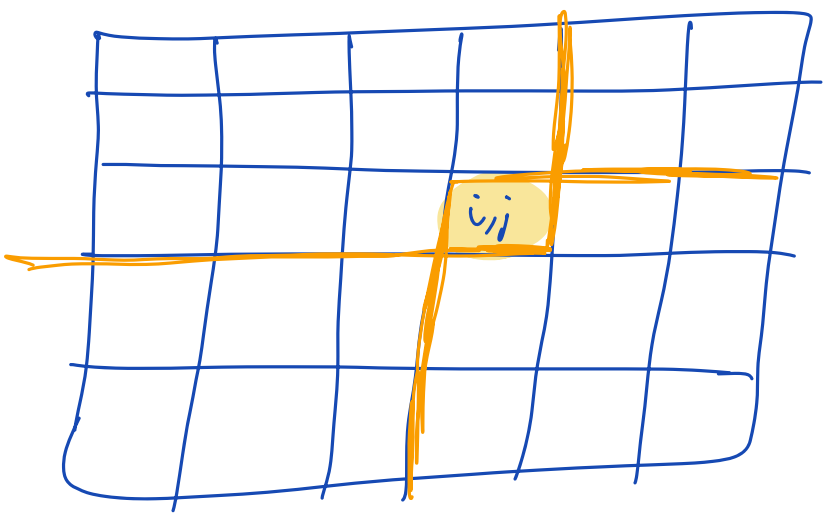
BR = 6

TL × BR = 9 × 6
= 54

- In how many sub - matrices (2 , 2) will be present?

TL = $(i+1)(j+1)$

n × m



BR = $(n-i)(m-j)$

< / > Final - Code

```
ans = 0
for ( i : 0 → n-1 ) {
    for ( j : 0 → m-1 ) {
        top_left = (i+1)(j+1)
        bottom_right = (n-i)(m-j)
        num_sub = top_left * bottom_right
        ans += a[i][j] * num_sub
    }
}
return ans.
```

TC : O(NM)

Number of submatrices = $\frac{n(n+1)}{2} \times \frac{m(m+1)}{2}$

a → 97

b = 98

String Builder

c = 99

"Omansh" + 11 = "Omansh 11"

int x = 'c' + 1 x → 100

99

char mychar = 'c' + 1

↳ 'd'