Q Find in row wise & col wise sorted matrix

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

13 → true
2 → true
15 → false

Brute: Iterate through the whole matrix
TC: O(NM)

Idea: Start from top right

-5	-2	1	13
-4	0	3	14
-3	2 ←	6	18

k = 2

Code

i = 0   j = m - 1

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

return false
```

TC: O(N)   linear

Q2 Binary matrix - each row sorted. Find row with max number of 1's. If multiple rows are answer, return smallest row no.

0	0	1	1
1	0	0	1
2	0	1	1

→ 0

0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1

→ 3

Brute: Count the 1's in each row & find which row has max
$O(N^2)$

Idea Since rows are sorted, we can use that

left movement till you have 1's.

Code

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

TC:

	0	1	2	3
0	0	0	1	1
1	1	1	1	1
2	0	1	1	1

ans = ~~0~~ 1

0, 3
0, 2
0, 1
1, 1
1, 0
1, -1
2, -1
3, -1

TC: same as above

Q3 Boundary elements Print NXN matrix

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

ans → 1 2 3 4 5 10 15 20 25
24 23 22 21 16 11 6

Idea: 1) Print N-1 of first row
2) Print N-1 of last col
3) Print N-1 of last row
4) Print N-1 of first col

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

$i=0$ $j=0$

0, 4

1 2 3 4

Code

i=0 j=0

```
// N-1 of first row
for ( k=0 ; k< n-1 ; k++ ) {
        print ( ar[i][j] )
        j++
}
```

i= 0 j = n-1

```
// N-1 of last col
for ( k=0 ; k< n-1 ; k++ ) {
        print ( ar[i][j] )
        i++
}
```

i = n-1 j = n-1

```
// N-1 of last row
for ( k=0 ; k< n-1 ; k++ ) {
        print ( ar[i][j] )
        j--
}
```

i = n-1 j = 0

```
// N-1 of first col
for ( k=0 ; k< n-1 ; k++ ) {
        print ( ar[i][j] )
        i--
}
```

i=0 j=0

Q4 Spiral print $N \times N$

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

$N = 5$
$\hookrightarrow N = 3$

ans → 1 2 3 4 5 10 15 20 25
24 23 22 21 16 11 6
7 8 9 14 19 18 17 12 13

Idea: Is this not equivalent to border printing multiple times ? Yes

i	j	N
0	0	5
1	1	3
2	2	1

0,0 N ⇒ border
1,1 N-2 border
2,2 N-4 border
⋮

N = 0 or N = 1

Code

```
i=0        j=0
while (N>1) {

    // N-1 of first row
    for ( k=0 ; k< n-1 ; k++) {
        print ( ar[i][j] )
        j++
    }

    // N-1 of last col
    for ( k=0 ; k< n-1 ; k++) {
        print ( ar[i][j] )
        i++
    }

    // N-1 of last row
    for ( k=0 ; k< n-1 ; k++) {
        print ( ar[i][j] )
        j--
    }

    // N-1 of first col
    for ( k=0 ; k< n-1 ; k++) {
        print ( ar[i][j] )
        i--
    }
```

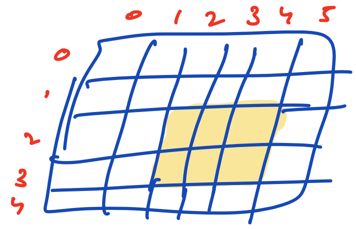
```
        N = N - 2
        i++
        j++
}

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

Q5 All submatrix sum

Submatrix like subarray

subarray → s, e
submatrix → TL, BR



2, 2

3, 4

TL

BR

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \Rightarrow$$

1 2 3 4

(1) (2) (3) (4)

(1 2) 3 (3 4) 7

$\begin{pmatrix} 1 \\ 3 \end{pmatrix}$ $\begin{pmatrix} 2 \\ 4 \end{pmatrix}$ $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

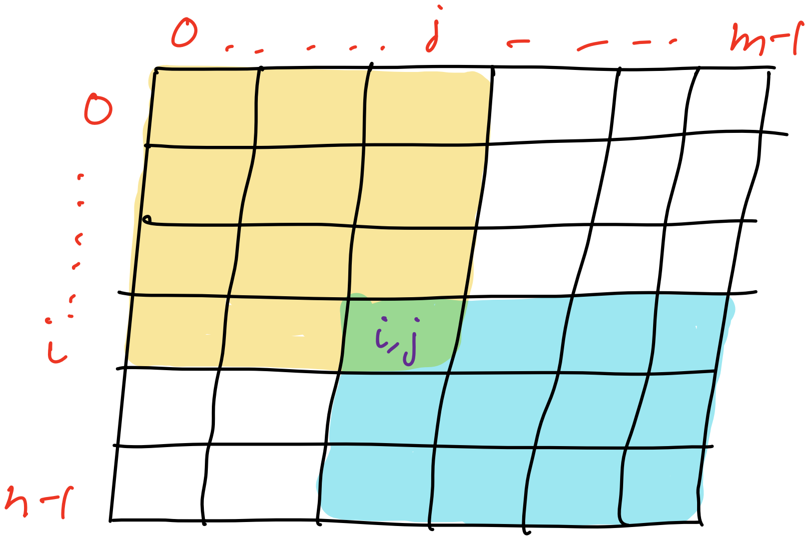
4 6 10

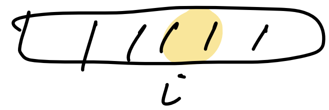
$\Leftarrow$ 40

Idea: Have you solved this ques for subarray? yes

How → Contribution Technique

Use same idea





$[i, n-1]$

$TL \to (i+1)(j+1)$

$BR \to (n-i)(m-j)$

Like in subarray $\Rightarrow$ contribution $= \overset{s}{(i+1)} \times \overset{e}{n-i}$

submatrix $\quad TL \times BR$

$$(i+1)(j+1)(n-i)(m-j)$$

1	2
3	4

$N = 2$

$N = 0$

Code

```
for(i=0 ; i<n ; i++) {
    for( j=0 ; j<m ; j++ ) {

        TL = (i+1)(j+1)
        BR = (n-i)(m-j)

        contri = TL x BR

        ans += ar[i][j] x contri
    }
}
```

TC: $O(NM)$

SC: $O(1)$