2d Arrays → int arr[N][M];

Ques

Given a row wise and column wise sorted matrix, find out whether element **k** is present or not.

$$A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & -5 & -2 & 1 & 13 \\ 1 & -4 & 0 & 3 & 14 \\ 2 & -3 & 2 & 6 & 18 \end{array}$$

k = 13 → True
k = 2 → True
k = 15 → false.

idea 1:- Brute force idea :-

Travel the whole matrix.

T.C → $O(N*M)$
S.C → $O(1)$.

idea 2 :- Binary Search:-

$O(N*\log M)$ or $O(M \log N)$.

idea 3:-

$$A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & -5 & -2 & 1 & 13 \\ 1 & -4 & 0 & 3 & 14 \\ 2 & -3 & 2 & 6 & 18 \end{array}$$

k = 0

$$A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & -5 & -2 & 1 & 13 \\ 1 & -4 & 0 & 3 & 14 \\ 2 & -3 & 2 & 6 & 18 \end{array}$$

$\underline{k = 0}$

Say we are at 1 and want to find 0, where should we move ?

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

$$A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & -5 & -2 & 1 & 13 \\ 1 & -4 & 0 & 3 & 14 \\ 2 & -3 & 2 & 6 & 18 \end{array}$$

$\underline{k = 2}$

$$A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & -5 & -2 & 1 & 13 \\ 1 & -4 & 0 & 3 & 14 \\ 2 & -3 & 2 & 6 & 18 \end{array}$$

$k = 15$

incr decr

incr incr

decr decr

incr dec

$$
A = \begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & -5 & -2 & 1 & 13 \\
1 & -4 & 0 & 3 & 14 \\
2 & -3 & 2 & 6 & 18
\end{array}
\qquad k = 15
$$

S.C → O(1).

→ O(N+M) (every time you skip one row or col).

```
i = 0, j = M-1
while (i<N && j>=0)
    if (arr[i][j] == k) {
        return True
    }
    else if (arr[i][j] < k) {
        i++;    // move down
    }
    else {
        j--;
    }
return false;
```

Q2)

Given a binary sorted matrix A of size N x N. Find the row with the maximum number of 1.

row wise

NOTE:

- If two rows have the maximum number of 1 then return the row which has a lower index.
- Assume each row to be sorted by values.

Example 1:

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

Ans → 0.

Example 2:

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

Ans → 3

Brute force

Iterate over each row & count no of 1s.

Optimized soln :-

	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

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

T.C -> O(N + M)   (M => N)

T.C -> O(N)

S.C -> O(1).

Break   8:10 Am - 8:20 Am.

Ques 3)

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

$n = 5$

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

mat[][] = $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$ (1, 2, 3, 6, 9, 8, 7, 4)

(0,0)

N = 5 x 5

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

// i=0, j= N-1

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

// i= n-1 , j= N-1

// print n-1 elements of last row

```
for (k = 1; k < n; k++) {     // n-1
        print(arr[i][j]);
        j--;
}
```

// i = n-1 , j = 0

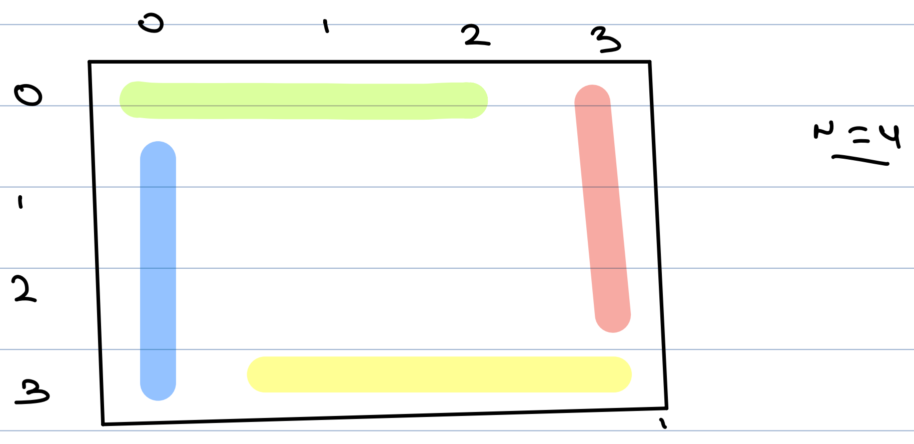
// print n-1 elements of first col

```
for (k = 1; k < n; k++) {     // n-1
        print(arr[i][j]);
        i--;
}
```

// i = 0 , j = 0

T.C → O(n)

S.C → O(1)

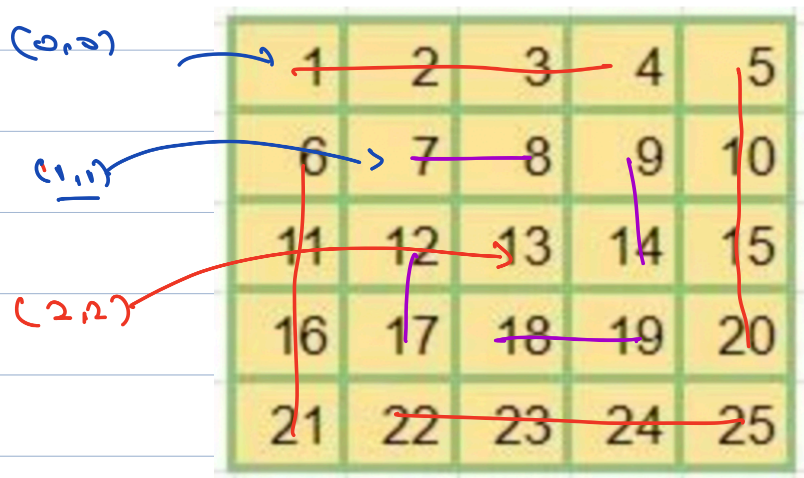


Ques)

Given an matrix of N X N i.e. Mat[N][N]. Print elements in spiral order in clockwise direction.

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

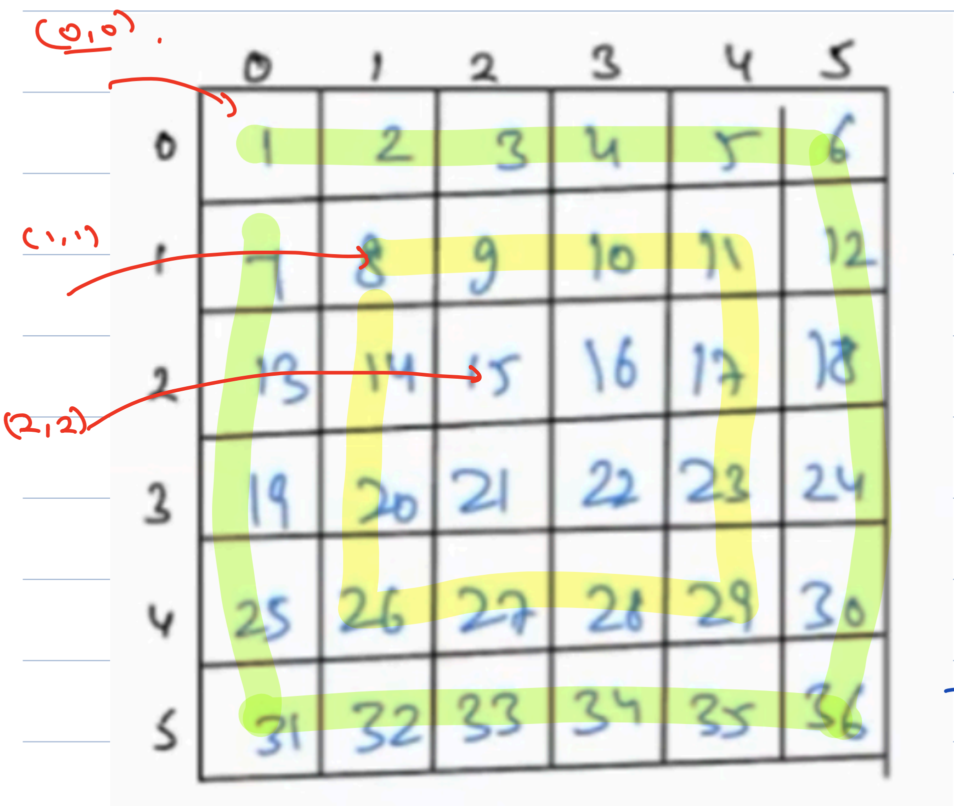
O/P → 1 2 3 4 5 10 15 20 25 24
23 22 21 16 11 6 7 8 9 14
19 18 17 12 13



N= 5x5.

N= 3x3

N= 1



N= 6x6

N= 4x4

N= 2x2.

T.C -> O(N*M)

n, i=0, j=0,

while (n > 1) {

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

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

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

// print n-1 elements of first col
for (k=1; k<n; k++) {      //n-1
    print (arr[i][j]);
    i--;
}

i++, j++;
```

N-- = 2;

}

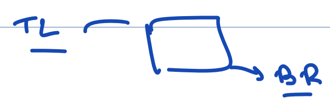
```
if (N==1) {
    print(Mat[i][j]);
}
```

Ques)

Given a matrix of N rows and M columns determine the sum of all the possible submatrices.

$$\begin{array}{c|c} & \begin{matrix} 0 & \ \ 1 & \ \ 2 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{bmatrix} 4 & 9 & 6 \\ 5 & -1 & 2 \end{bmatrix} \end{array}$$

TL, BR



$[4] \rightarrow 4$

$[9] \rightarrow 9$

$[6] \rightarrow 6$

$[5] \rightarrow 5$

$[-1] \rightarrow -1$

$[2] \rightarrow 2$

$[4\ \ 9] \rightarrow 13$

$[9\ \ 6] \rightarrow 17$

$[5\ \ -1] \rightarrow 4$

$[-1\ \ 2] \rightarrow 1$

$[4\ \ 9\ \ 6] \rightarrow 19$

$[5\ \ -1\ \ 2] \rightarrow 6$

$\begin{bmatrix} 4 \\ 5 \end{bmatrix} \rightarrow 9$

$\begin{bmatrix} 9 \\ -1 \end{bmatrix} \rightarrow 8$

$\begin{bmatrix} 6 \\ 2 \end{bmatrix} \rightarrow 8$

$\begin{bmatrix} 4 & 9 \\ 5 & -1 \end{bmatrix} \rightarrow 17$

$\begin{bmatrix} 9 & 6 \\ -1 & 2 \end{bmatrix} \rightarrow 16$

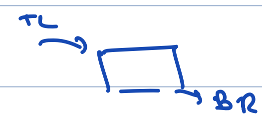
$\begin{bmatrix} 4 & 9 & 6 \\ 5 & -1 & 2 \end{bmatrix} \rightarrow 25$

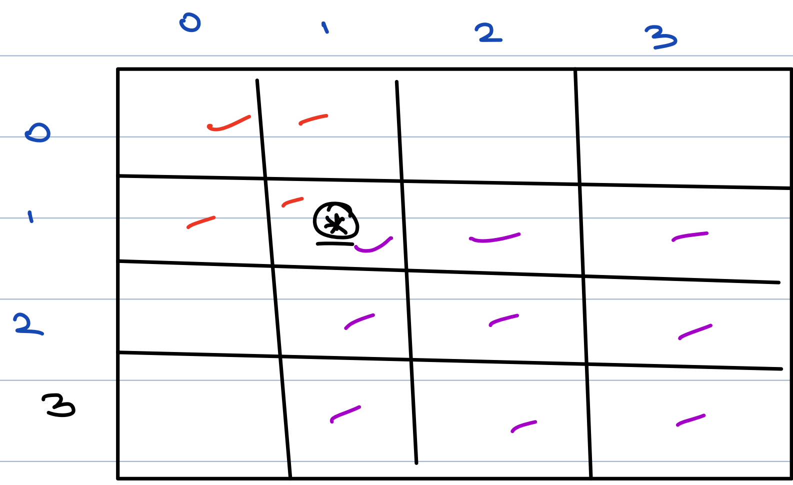
Sum of all subarray Sum:

arr[] = {1, 2, 3}

{1} {2} {3},

{1,2} {2,3} {1,2,3}

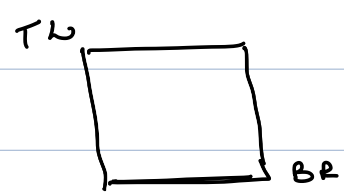


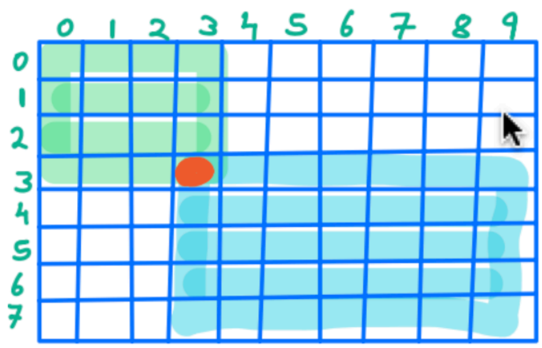


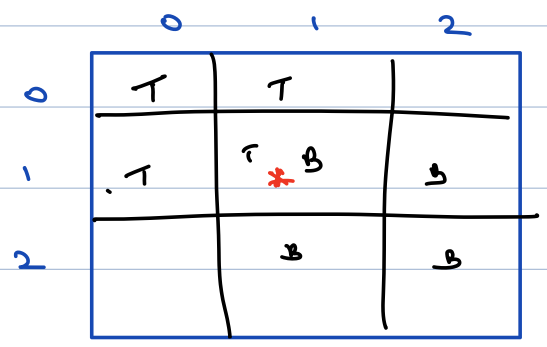
TL BR

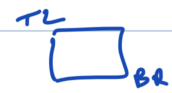
4 9 = 36

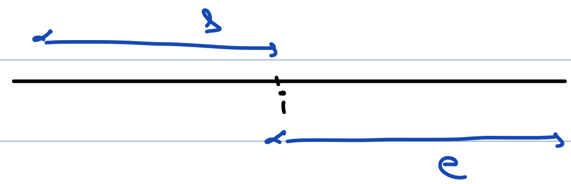
BR => (N−i)(M−j)





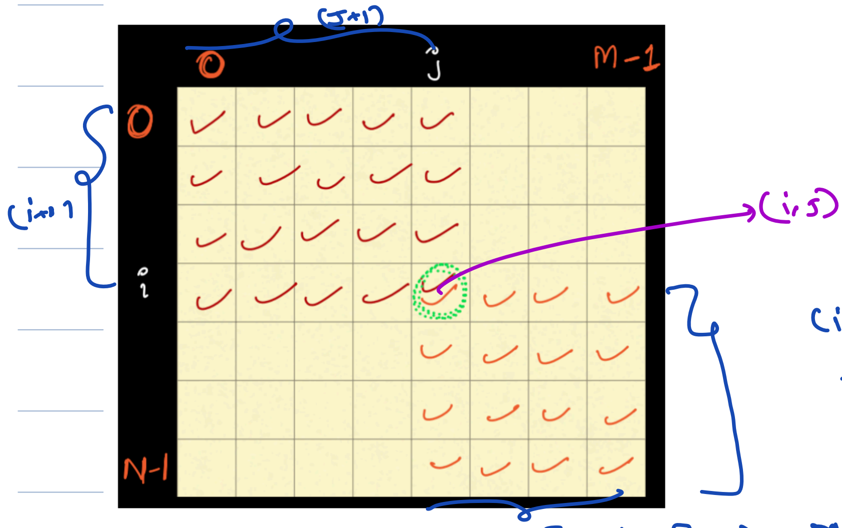






BR => $(n-i)(m-j)$

How to find the number of TL cells and BR cells in which (i,j) is part of.



(i to N-1)

$\Rightarrow N - x - i + x$

$\Rightarrow \underline{N-i}$

[j M-1] $\Rightarrow M - x - j + x \Rightarrow (M-j)$

TL $\rightarrow$ $(i+1) * (j+1)$

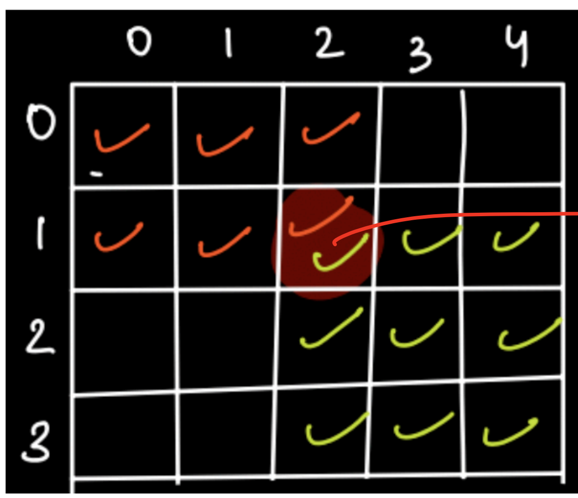
BR $\Rightarrow$ $(N-i)(M-j)$

Total submatrices in which this cell will be present =

$(i+1) * (j+1)(N-i) * \underline{(M-j)}$.

$\Rightarrow (1+1)(2+1)(4-1) * (5-2)$

$\Rightarrow 2 * 3 * 3 * 3 \Rightarrow \underline{54}$.



$N * M = 4 * 5$

(1, 2)

```
ans = 0;
for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
        TL = (i+1)*(j+1)
        BR = (n-i)(m-j)
        Contribution = TL*BR* arr[i][j];
        ans += Contribution;
    }
}
return ans;
```

T.C → O(N*M)
S.C → O(1).