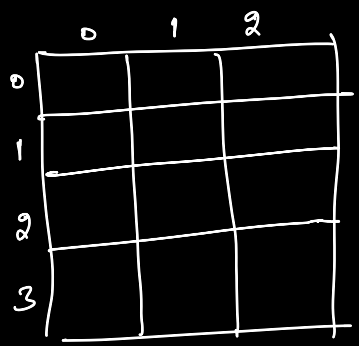
Today's content →
- → 2 D Arrays
- → Problems on 2-D Arrays.

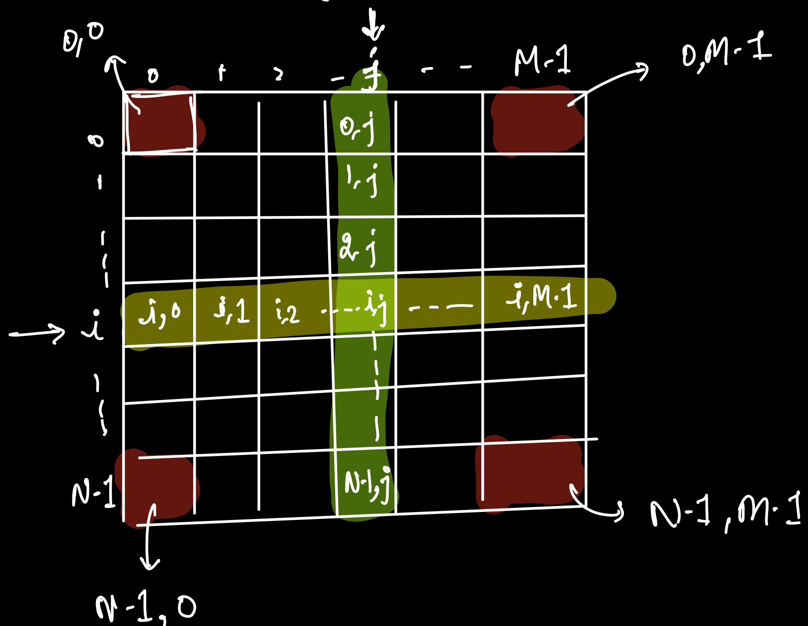
✓ How to declare 2-D array?

int mat[4][3] → no. of columns.

no. of rows



int mat[N][M]



observation 1. : If we move in j^{th} – row
col will change $[0, M-1]$

observation 2. : If we move in j^{th} – col
row will change $[0, N-1]$

Q:) Given mat[N][M]. Print row-wise sum.

	0	1	2	3
0	4	3	1	7
1	2	5	8	6
2	9	1	3	4

o/p.
15
21
17

Pseudo code.

```
void printSum(arr, N, M){
    for( i=0 ; i<N ; i++){
        sum = 0
        for( j=0; j<M; j++){
            sum += arr[i][j]
        }
        // print(sum)
    }
}
```

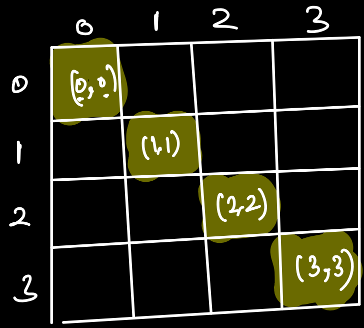
T.C → $O(N*M)$
S.C → $O(1)$

Q:) Print column-wise sum.
[To-do].

	0	1	2	3
0	4	3	1	7
1	2	5	8	6
2	9	1	3	4

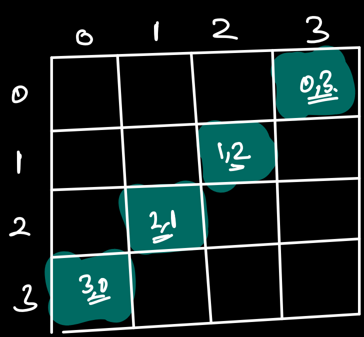
o/p → 15, 9, 12, 17

Q2) Given arr[N][N]. Print Diagonals.
- from l to r
- from r to l




```
i=0, j=0
while (i<N && j<N) {
    print(arr[i][j]);
    i++
    j++
}
```

$\boxed{T.C \to O(N),\ S.C \to O(1)}$



0,3
↓
1,2
↓
2,1
↓
3,0
↓
4,-1

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

$\boxed{T.C \to O(N),\ S.C \to O(1)}$

Q: Given a mat[N][M]. Print all diagonals from R to L.

⇓

diagonals starting from 0^{th} row or $M-1^{th}$ column.

	0	1	2	3	4
0		0,1		0,3	0,4
1	1,0		1,2	1,3	1,4
2		2,1	2,2	2,3	
3	3,0	3,1	3,2		

r, c

0, 3

↓

1, 2

↓

2, 1

↓

3, 0

↓

4, -1 → stop.

r, c

1, 4.

↓

2, 3

↓

3, 2

↓

4, 1 → stop.

arr[4][5].

	0	1	2	3	4
0	5	7	3	2	6
1	11	9	20	21	13
2	14	-2	3	11	17
3	27	37	47	50	60

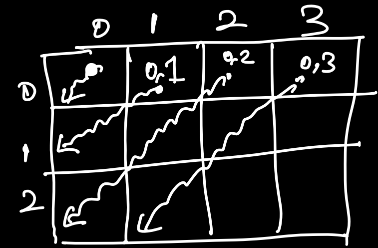
o/p.

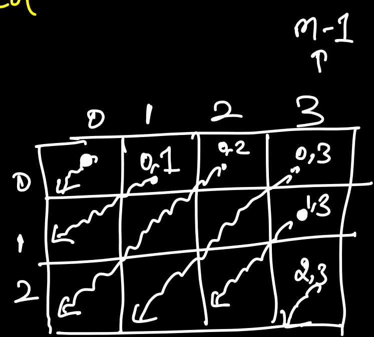
5
7 11
3 9 14
2 20 -2 27
6 21 3 37
13 11 47
17 50
60

Final obj. for every s.p, print diagonal from l to r.

pseudo-code.

```
void  printAllDiagonals ( arr, N, M) {
    // print all Diagonals starting from 0th row
    for( j = 0 ; j < M ; j++) {
        r = 0 , c = j
        while (r<N && c>=0) {
            print (arr[r][c])
            r++ , c--
        }
    }
    // print all Diagonals starting from last col.
    for( i = 1 ; i < N ; i++) {
        r = i , c = M-1
        while (r<N && c>=0) {
            print (arr[r][c])
            r++ , c--
        }
    }
}
```





$\boxed{T.C \rightarrow O(N*M) \quad , \quad S.C \rightarrow O(1)}$

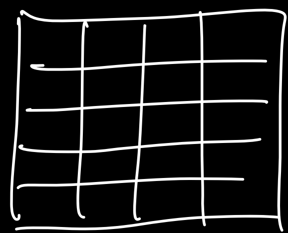
⇓

[we are touching all the elements once.]

arr[N] →

→ O(N).

mat [N][M]



T.C → O(N*M)

[printing all elements in a fancy way doesn't impact T.C.]

[Break → 10:10 → 10:20]

Q: Given mat[N][N]. Calculate the transpose of the given matrix. [S.C → O(1)]

Note → take transpose in given matrix only.

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

row0 → col0
row1 → col1
row2 → col2
row3 → col3
row4 → col4

	0	1	2	3	4
0	1	6	11	16	21
1	2	7	12	17	22
2	3	8	13	18	23
3	4	9	14	19	24
4	5	10	15	20	25

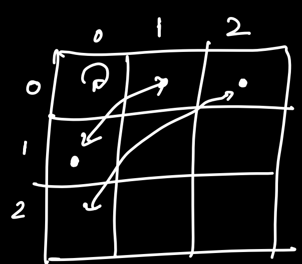
i=0 → j → [1,4]
i=1 → j → [2,4]
i=2 → j → [3,4]
i → j → [i+1, N-1]

idea: Swap elements of upper half with lower half

```
void takeTranspose( arr, N) {
    for( i = 0 ; i < N ; i++){
        for( j = i+1 ; j < N ; j++){
            //swap arr[i][j] with arr[j][i]
            temp = arr[i][j]
            arr[i][j] = arr[j][i]
            arr[j][i] = temp
        }
    }
}
```

$\boxed{\begin{array}{l} T.C \to O(N^2) \\ S.C \to O(1) \end{array}}$

```
void takeTranspose( arr, N) {
    for( i = 0 ; i < N ; i++){
        for ( j = 0 ; j < N ; j++){
            //swap arr[i][j] with arr[j][i]
            temp = arr[i][j]
            arr[i][j] = arr[j][i]
            arr[j][i] = temp
        }
    }
}
```



arr[0][0] ⟷ arr[0][0]

arr[0][1] ⟷ arr[1][0]

arr[1][0] ⟷ arr[0][1]

arr[0][2] ⟵ arr[2][0]

∴ Matrix is going to remain as it is.

// Rectangular matrix $\Rightarrow$ We need to have extra space.

mat[2][5]

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \end{array} \quad (2*5) \quad \Rightarrow \quad \begin{bmatrix} 1 & 6 \\ 2 & 7 \\ 3 & 8 \\ 4 & 9 \\ 5 & 10 \end{bmatrix} \; \underline{\underline{5*2}}$$

// Now questn is simplified.

Q.) Given mat[N][N]. Rotate matrix by 90° in clockwise direction. { S.C → O(1) }

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \\ 2 & 11 & 12 & 13 & 14 & 15 \\ 3 & 16 & 17 & 18 & 19 & 20 \\ 4 & 21 & 22 & 23 & 24 & 25 \end{array}$$

0th row → 4th col
1st row → 3rd col
2nd row → 2nd col
3rd row → 1st col
4th row → 0th col

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 21 & 16 & 11 & 6 & 1 \\ 1 & 22 & 17 & 12 & 7 & 2 \\ 2 & 23 & 18 & 13 & 8 & 3 \\ 3 & 24 & 19 & 14 & 9 & 4 \\ 4 & 25 & 20 & 15 & 10 & 5 \end{array}$$

⇓ transpose.

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 6 & 11 & 16 & 21 \\ 1 & 2 & 7 & 12 & 17 & 22 \\ 2 & 3 & 8 & 13 & 18 & 23 \\ 3 & 4 & 9 & 14 & 19 & 24 \\ 4 & 5 & 10 & 15 & 20 & 25 \end{array}$$

reverse 0th row
" 1st row
reverse 2nd row
reverse 3rd row
reverse 4th row

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 21 & 16 & 11 & 6 & 1 \\ 1 & 22 & 17 & 12 & 7 & 2 \\ 2 & 23 & 18 & 13 & 8 & 3 \\ 3 & 24 & 19 & 14 & 9 & 4 \\ 4 & 25 & 20 & 15 & 10 & 5 \end{array}$$

left ↓ ... right ↓

1	6	11	16	21
0	1	2	3	4

[Reverse an array] row.

// 1. take transpose of the given matrix.] → N^2

// 2. Reverse every row.

```
for( i = 0 ; i < N ; i++){
    left = 0 , right = N-1
    while ( left < right) {
        //swap arr[i][left] with arr[i][right].
        temp = arr[i][left]
        arr[i][left] = arr[i][right]
        arr[i][right] = temp.
        left++ , right --
    }
}
```

] → N^2

$\boxed{\begin{array}{l} T.C \rightarrow O(N^2) \\ S.C \rightarrow O(1) \end{array}}$

for 360°

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

(Remain same)

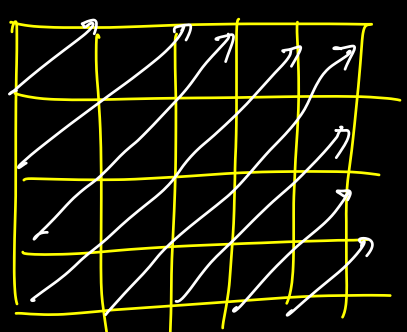
→ [Rotate 90° anti-clockwise.] [180°, 270°, - - -]

// For Rectangular Matrix. // We have to take extra space.

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \end{array} \quad (2 \times 5) \longrightarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 6 & 1 \\ 1 & 7 & 2 \\ 2 & 8 & 3 \\ 3 & 9 & 4 \\ 4 & 10 & 5 \end{array} \quad \{5 \times 2\}$$

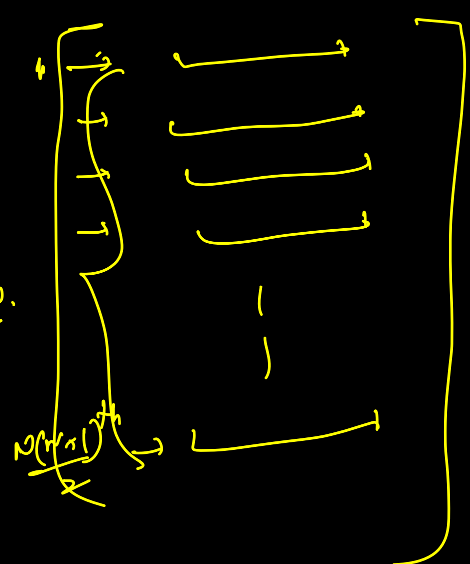
$$\boxed{T.C \rightarrow O(N*M), \quad S.C \rightarrow O(N*M)}$$

Extra Todo → Print Diagonals from l to r



total no. of subarrays → $\boxed{\frac{N(N+1)}{2}}$

length of every subarray → n.



T.C $\rightarrow O(N^2 * N)$

→ Sliding Window + Spiral matrix ? → next session.