Applications

1) Image Processing
2) Chess → 8x8
3) Sudoku
4) Ludo
5) Tic Tac Toe

X		O
	O	
		X

suppose we want to store 5 subjects marks for a single student

A1[ ] =

5	1	2	5	4

storing 5 subjects marks for 10 students

— Having 10 different 1D arrays doesn't makes sense

Rather let's have array of arrays → 2D arrays

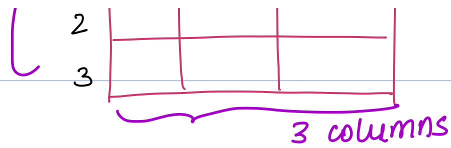
int mat[10][5] = { {5, 1, 2, 4, 5}, {4, 19, 10, 25, 17},
{4, 1, 11, 12, 15}, ... }

2D arrays are better to visualize in form of matrix having rows & columns.

int mat[4][3];

	0	1	2
0			
1			

4 rows

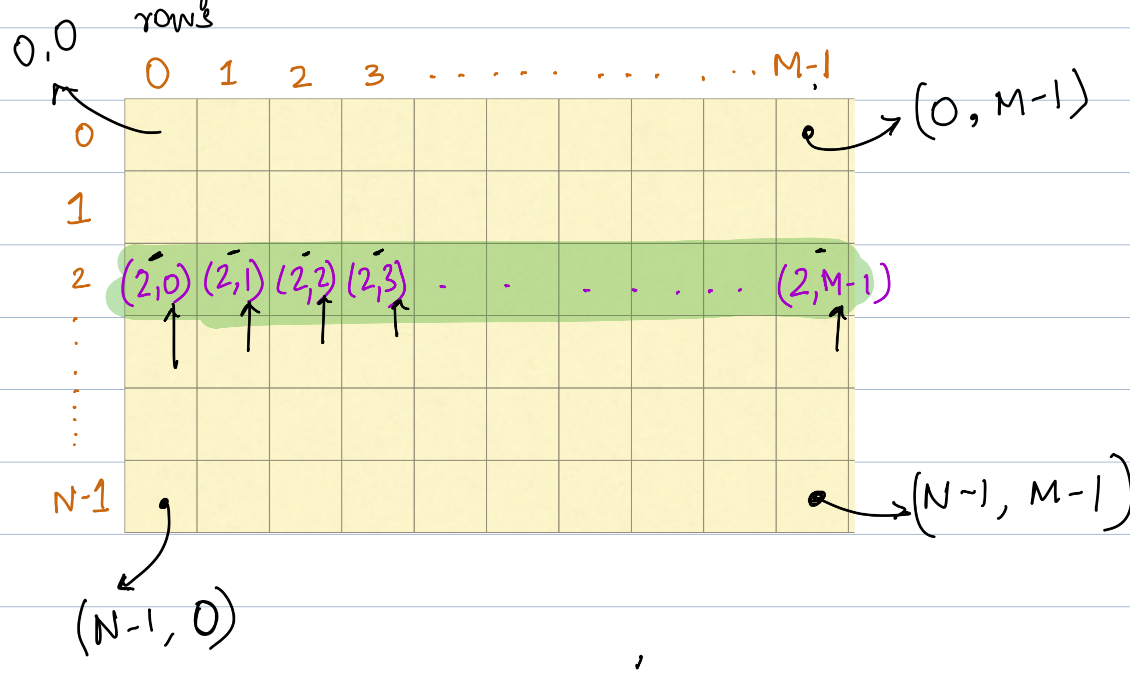


int mat [N][M] ;

no. of rows

no. of columns.

N → rows
M → columns



Que. Given a row no., find the sum of the elements of that row.

Eg ⇒ row no. ⇒ 2

i \ j	0	1	2	3
0	4	1	5	2
1	9	6	1	7
2	10	14	1	17

→ 10 + 14 + 1 + 17
= 32

Along a row

constant - row no.

constant → row no.
Change → col No.
[0 M-1]

```
i=2
sum = 0
for(j=0;j<M; j++){
  sum+ = A[i][j]
}
```

Dry Run

i	j	sum = 0
2	0	+ A[2][0] = 10
2	1	+ A[2][1] = 14
2	2	+ A[2][2] = 1
2	3	+ A[2][3] = 17

Q. Print sum of all the rows one by one

```
for ( i = 0 ; i< N ; i++) {
    sum = 0
    for(j=0;j<M; j++){
      sum+ = A[i][j]
    }
    print(sum)
}
```

	0	1	2	3
0	4	1	5	2
1	9	6	1	7
2	10	14	1	17

TC → O(N×M)
SC → O(1)

i = 0 j = 0 A[0][0] +
1 A[0][1] +
2 A[0][2] +
3 A[0][3]

i = 1 j = 0 —
1 —
2 —
3 —

i = 2 j = 0 —

1
2
3

Que Print sum of ele of a given column no.

	0	1	2	3	4	.	.	.	..	M-1
0				(0,3)						
1				(1,3)						
2				(2,3)						
3				(3,3)						
⋮				⋮						
N-1				(N-1, 3)						

Along column

row no → changing [0 N-1]
col no. → constant

```
j
sum = 0
for ( i = 0 ; i < N ; i++ ) {
        sum += A[i][j]
}
```

Dry Run

say j = 1

i = 0	A[0][1]
i = 1	A[1][1]
i = 2	A[2][1]

Que.

Given a matrix, print sum of column by column

eg

$$N=3 \quad M=3$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 4 & 1 & 5 \\ 1 & 6 & 7 & 9 \\ 2 & 2 & 5 & 9 \end{array}$$

```
for(j = 0; j < M; j++){
    sum = 0
    for ( i = 0 ; i < N ; i++){
        sum + = A[i][j]
    }
    print( sum)
}
```

j	i	
j = 0	i = 0	A[0][0]
	1	A[1][0]
	2	A[2][0]
j = 1	i = 0	A[0][1]
	1	A[1][1]
	2	A[2][1]
j = 2	i = 0	A[0][2]
	1	A[1][2]
	2	A[2][2]

TC : $O(N \times M)$

SC : $O(1)$

Que. Given a square matrix [no. of rows = no. of cols}

int mat[N][N]

eg 5x5

0 1 2 3 4 5

$$\begin{array}{c|ccccc} 0 & 3 & 2 & 14 & 15 & 19 \\ 1 & 7 & 13 & 15 & 26 & 27 \\ 2 & 22 & 20 & 5 & 28 & 22 \\ 3 & 30 & 10 & 16 & 11 & 21 \\ 4 & 32 & 29 & 8 & 29 & 7 \end{array}$$

Right to left

→ 19 + 26 + 5 + 10 + 32

Left to Right

→ 3 + 13 + 5 + 11 + 7

	0	1	2	3	4	
0	3	2	14	15	19	0, 0
1	7	13	15	26	27	1, 1
2	22	20	5	28	22	2, 2
3	30	10	16	11	21	3, 3
4	32	29	8	29	7	4, 4

let's consider left to right diagonal first.

I

```
sum = 0
for ( i = 0; i < N ; i ++) {
        for (j = 0; j < N; j ++){
            if(i == j){
                sum += A[i][j]
            }
        }
}
print(sum)
```

TC = O(N²)

SC = O(1)

II

Optimized Code

```
sum = 0
for ( i = 0; i < N; i++) {
    sum += A[i][i]
}
print(sum)
```

TC → O(N)

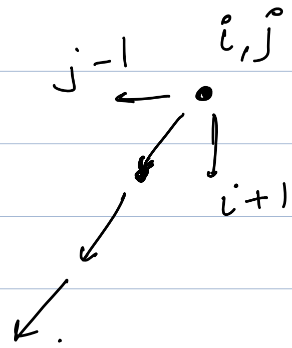
SC → O(1)

Right to left

	0	1	2	3	4
0	3	2	14	15	19
1	7	13	15	26	27
2	22	20	5	28	22
3	30	10	16	11	21
4	32	29	8	29	7

i j
0, 4
1, 3
2, 2
3, 1
4, 0

$\boxed{i<N}$ $\boxed{j>=0}$



I

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

A single condition will do

TC → O(N)
SC → O(1)

0 1 2 . . . N-1

		0	1	2	3	4	
0	0	3	2	14	15	19	
1	1	7	13	15	26	27	
2	2	22	20	5	28	22	
:	3	30	10	16	11	21	
N-1	4	32	29	8	29	7	

i j
0, N-1 = 0 + (N-1) = N-1
1, N-2 = 1 + (N-2) = N-1
2, N-3 = 2 + (N-3) = N-1
N-1, 0 = N-1 + 0 = N-1

$i + j = N-1$

$j = N-1-i$

II

```
sum=0
for(i=0; i<N; i++){
    j = N-1-i
    sum += A[i][j]
}
```

TC → O(N)

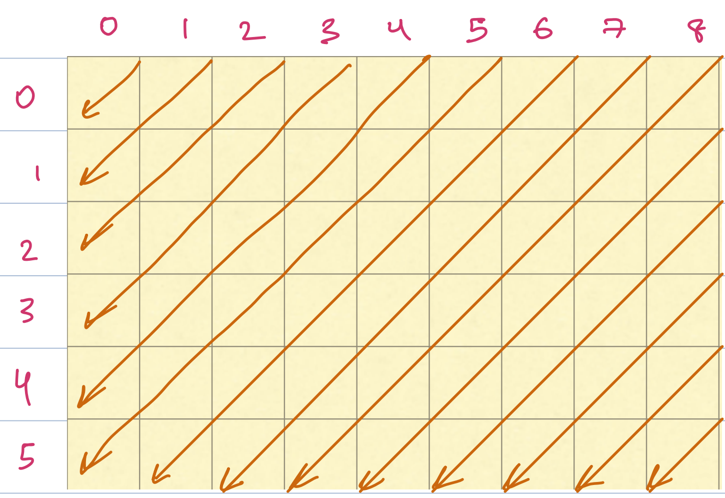
SC → O(1)

Break! → 8 min.

$\boxed{9:30}$

Que: Given rectangular matrix, print elements diagonal by diagonal in the given direction.

Ans.



4x5

	0	1	2	3	4
0	11	9	3	5	10
1	18	13	100	9	-1
2	6	2	5	10	7
3	14	13	26	8	18

11
9 18
3 13 6
5 100 2 14
10 9 5 13
-1 10 26
7 8
18

From each cell of first row, a diagonal is originating.

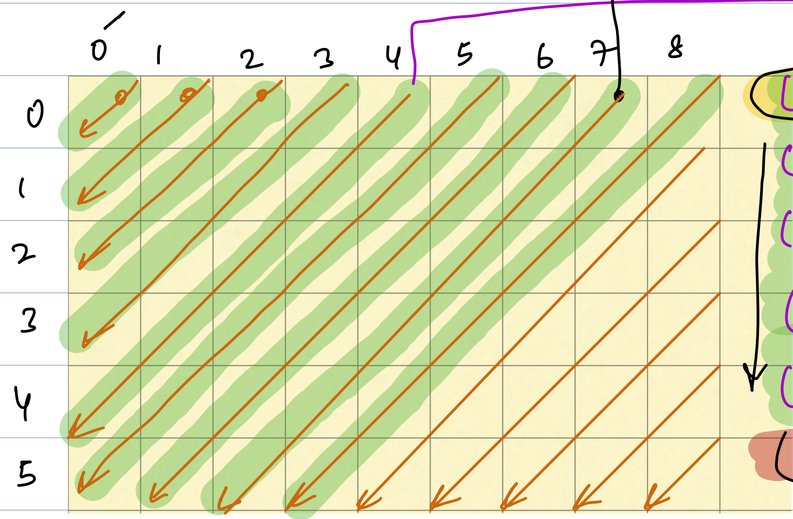
No. of cells in first row → M

From each cell of last col, a diag. is originating.

No. of cells in last col → N

M + N ?

$\boxed{N + M - 1}$ [Since we are calculating top right diagonal twice]



i j	i j	i j
(0,4)	(0,7)	(1,8)
(1,3)	(1,6)	(2,7)
(2,2)	(2,5)	(3,6)
(3,1)	(3,4)	(4,5)
(4,0)	(4,3)	(5,4)
(5,-1)	(5,2)	(6,3)
	(6,1)	

(i<N && j>=0)

Observations

1. Along a diagonal, i ↑, j ↓
2. If a starting coordinates of an diagonal are known, we can iterate over ele of that diagonal.
   i ↑
   j ↓

given starting coordinates of any diagonal, we can iterate over ele of that diagonal { i, j

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

starting indices of all the diagonals

M-1

⇒ 0,0 0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8

1,8
2,8
3,8
4,8
5,8

i=[1 N-1] j const

```
i=0
for( j = 0 ; j < M ; j++){ //M
    I = i     J = j
    while ( I<N  && J >= 0){
        print( A[I][J] )
        I++
        J--
    }
    print("\n")
}
```

	0	1	2	3	4
0	4	5	1	7	8
1	1	9	7	5	10
2	2	5	11	10	9
3	6	4	5	3	11

N+M-1

```
j = M-1
for ( i = 1 ; i<N; i++){
    I = i     J = j
    while (I<N  && J >= 0 ) {
        print ( A[I][J] )
        I++
        J--
    }
```

i=0 j=0
I=0 J=0 4
1 -1

i=0 j=1
I=0 J=1 5 1
1 0

i=0 j=2 1 9 2
I=0 J=2
1 1
2 0
3 -1

i=0 j=3

print("\n")

}

TC

$O(N * M)$

Que. Transpose

square matrix.

int mat [N] [N]

INPLACE - means no extra space

(0,1)

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \\ 3 & 13 & 14 & 15 & 16 \end{array}$$

(1,3)

(3,1)

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 5 & 9 & 13 \\ 1 & 2 & 6 & 10 & 14 \\ 2 & 3 & 7 & 11 & 15 \\ 3 & 4 & 8 & 12 & 16 \end{array}$$

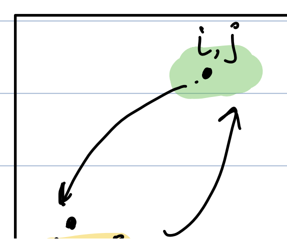
(1,3)

(1,0) (3,1)

Observation

The big diagonal from left to right is same.

```
for( i = 0; i<N ; i++) {
    for(j = 0; j<N; j ++) {
        ( A[i][j]   A[j][i])
```



swap(A[i][j], A[j][i])   (j, i

}

}

```
swap(a, b){
    c = a
    a = b
    b = c
}
```

$$
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & 1 & 2 & 3 & 4 \\
1 & 5 & 6 & 7 & 8 \\
2 & 9 & 10 & 11 & 12 \\
3 & 13 & 14 & 15 & 16
\end{array}
$$

i j

0 1   0 2   0 3

1 2   1 3

2 3

N=4

i	j
0	[1 3]
1	[2 3]
2	[3 3]
	i+1 N-1

```
for (i=0; i<N-1; i++){
    for(j = i+1; j< N; j++) {
        swap (A[i][j], A[j][i])
    }
}
```

Iterations $\approx \frac{N^2}{2}$

TC $\Rightarrow O(N^2)$

SC $\Rightarrow O(1)$

N=4

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

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

transposed

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

i=0 j=1

swap(A[0][1], A[1][0])

i=0 j=2

A[0][2], A[2][0]

i=0 j=3

A[0][3], A[3][0]

i=1 j=2

A[1][2], A[2][1]

i=1 j=3

A[1][3] A[3][1]

i=2 j=3

A[2][3], A[3][2]

i=3 j=4