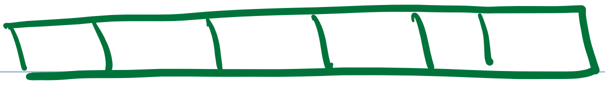
2D Matrix

Print row wise/ col wise sum

Principal Diagonals

Row to Column Zero

1D arrays : list of elements



2D matrix : 2D array which has a rectangular grid of nos. → element

Store elements arranged in row and cols

1. int mat[N][M] (C++)

- int ↓ data type
- mat ↓ name of 2D mat
- [N] ↓ rows
- [M] ↓ cols

2. int[][] mat = new int[N][M] (Java)

3. mat = [[0]*M for _ in range(N)] (Python)

		cols	
rows	0	1	2
0	0,0	0,1	0,2
1	1,0	1,1	1,2
2	2,0	2,1	2,2
3	3,0	3,1	3,2

r,c

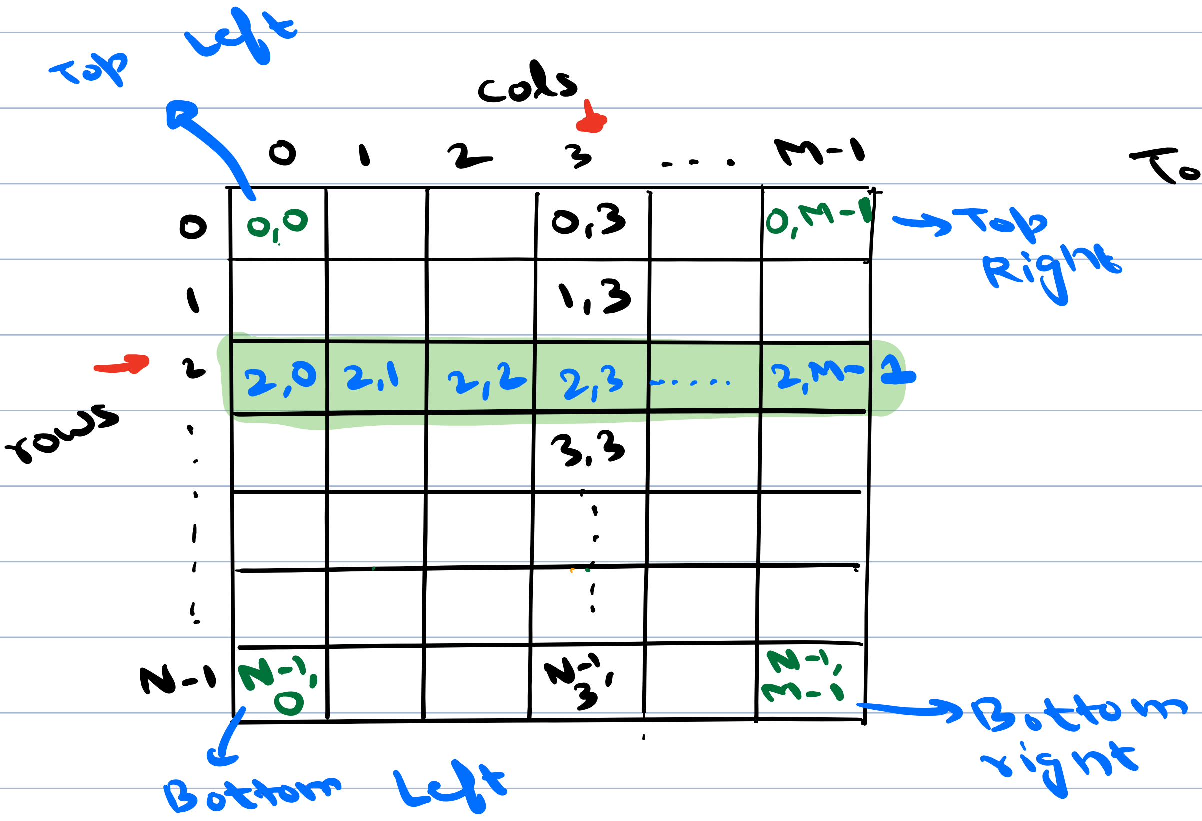
int mat[4][3]
- [4] ↓ rows
- [3] ↓ cols

cell
↓
mat[r][c]

Total ele = 4×3 = 12 ele

int mat[N][M]
rows cols

Total ele
= N×M



mat[N][M]
rows col

★ Iterate in a row
1. Row no. is fixed
2. Col no. → [0 M-1]   M → cols

★ Iterate in a col
1. Col no. is fixed
2. Row no. → [0 N-1]   N → rows

Given 2D Matrix [N][M], print row wise sum.

mat[3][4]

$$\begin{array}{c|cccc|c} & 0 & 1 & 2 & 3 & O/P \\ \hline 0 & 1 & 2 & 3 & 4 & 10 \\ 1 & 5 & 6 & 7 & 8 & 26 \\ 2 & 9 & 10 & 11 & 12 & 42 \end{array}$$

Traverse each row and while traversing take sum of elements present in that row

mat [N][M]

```
for (r=0; r<n; r++){
    // iterate r^th row to get sum
    int sum = 0
    for (c=0; c<M; c++){
        sum = sum + mat[r][c]
    print (sum)
    print ("\n")
```

TC: O(N*M)
SC: O(1)

Given 2D Matrix [N][M], print col wise sum.

mat[3][4]

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \\ \hline O/P & 15 & 18 & 21 & 24 \end{array}$$

Traverse each col, while traversing take some of elements present in that col.

```
for (c=0 ; c < M ; c++) {
    // iterate on cth column ; get sum
        int sum=0
        for (r=0 ; r < N ; r++) {
            sum = sum + mat[r][c]
        }
        print(sum)
        print("\n")
}
```

TC: O(NxM)
SC: O(1)

rows = cols

3. Given a 2D square matrix mat[N][N], print diagonals

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array}$$

mat[3][3]

2 diagonals in square matrix

① Principal Dia onal

Top Left → Bottom Right

O/P : 1 5 9

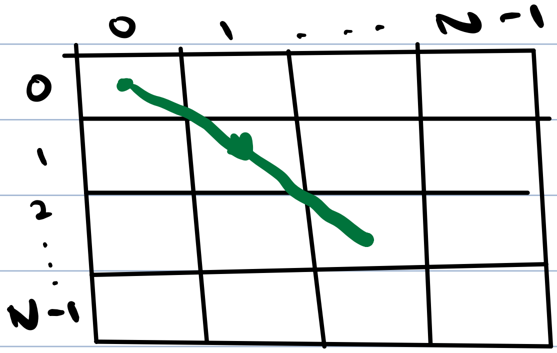
0,0 → 1,1 → 2,2

O/P : 3 5 7

② Anti - Diagonal

Top Right → Bottom Left

mat [N] [N]



① r = c

②

0,0
↓
1,1
↓ +1
2,2
↓
3,3
↓
⋮
N-1, N-1

```
void printDiagonal (int mat[N][N]) {
    int i = 0
    while( i < N ) {
        print (mat[i][i])
        i++
    }
}
```

OR

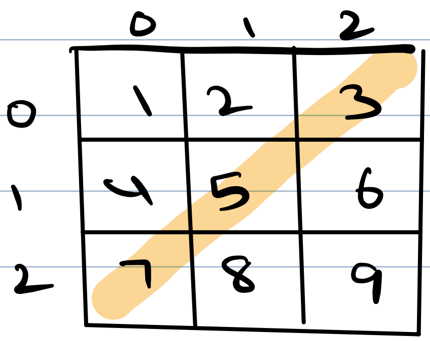
```
for (i=0 ; i<N ; i++)
    print (mat [i][i]
```

i → [0 N-1]

TC : O(N)

SC : O(1)

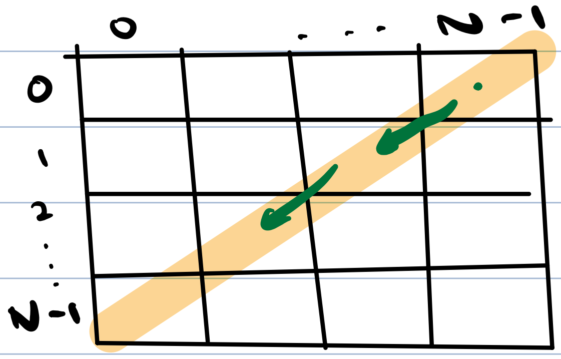
Anti- Diagonal



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

3 → 5 → 7

0,2 → 1,1 → 2,0



r , c

TR 0,N-1

↓

1, N-2

↓

2,N-3

↓

⋮

BL N-1, 0

+1 ↓ ↓ -1

STOP N, -1

r++ c--

```
void printAntiDiagonal (int mat[N][N]) {
        int r=0, c= N-1
        while( r<N && c>=0)
                print (mat[r][c])
                r++ c--
}
```

TC : O(N)
SC : O(1)

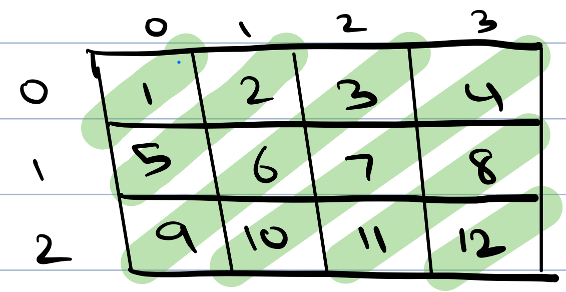
No need for 2 conditions, either of 1 conditions is sufficient

(R→L)

Print all anti diagonals of a non-square matrix.

↓
rectangle
rows != cols

mat [N] [M]



	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

mat[3][4]

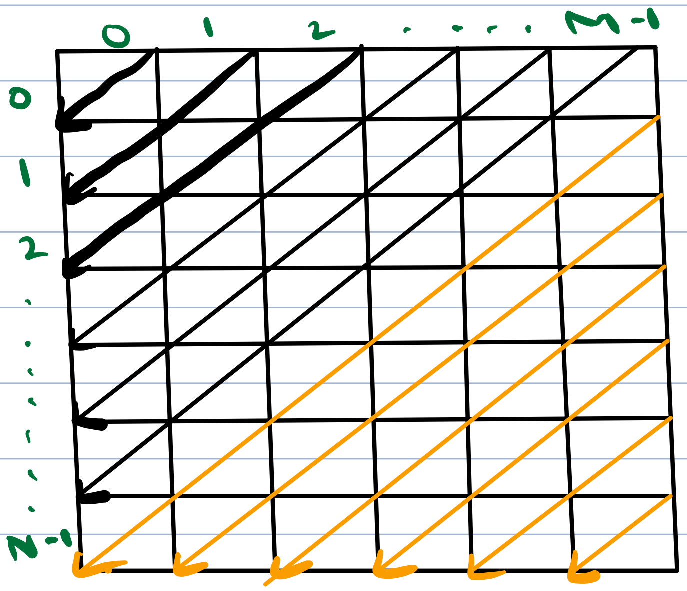
cnt = 6

O/P

1
2 5
3 6 9
4 7 10
8 11
12

① Every cell in 0th row is starting pt.

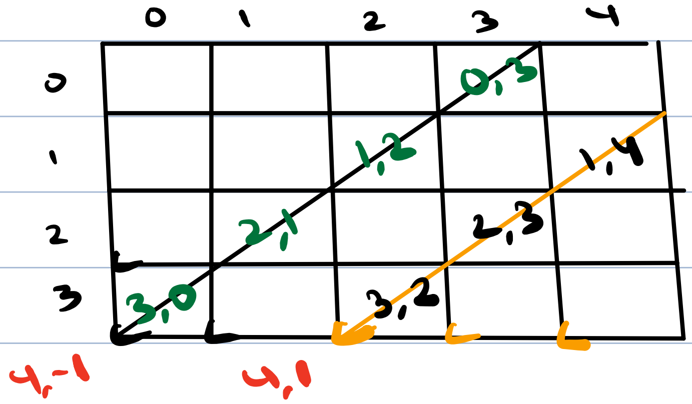
② Every cell in last col is starting pt.



mat [N][M]

cnt = M + N − 1

Q. Given mat $\overset{N}{[4]}\overset{M}{[5]}$, cnt of right to left (anti) diagonals?



cnt = M + N - 1
= 5 + 4 - 1 = 8

r, c
0,3
+1 ↓ -1
1,2
↓
2,1
↓
3,0
↓
$\boxed{4,-1}$ STOP

r++, c--

Stop if cell invalid
if r or c are invalid

continue when both are valid

```
void printAntiDiagonal(int i, int j, mat[N][M]) {
        int r=i, c=j
        while(r<N && c≥0){
            print(mat[r][c])
            r++  c--
        }
        print("\n")
}
```

// print all diagonals starting at 0th row

```
int row=0
for (col = 0 ; col < M ; col++ ){
    printAntiDiagonal (row, col, mat)
}
```

// print all diagonals starting at last col

```
    col = M-1
for (row = ~~0~~ 1 ; row < N ; row++) {
    printAntiDiagonal (row, col, mat)
}
```

TC : O(N×M)

SC : O(1)

10:30

5.

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 0 \\ 2 & 9 & 2 & 0 & 4 \end{array}$$

if A[i][j] = 0

make entire

ith row → 0

jth col → 0

↓

O/P

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

1	0	6	7
8	9	10	12
13	14	15	2

→

O/P

0	0	0	0
8	0	10	12
13	0	15	2

~~1~~ 0	0	~~6~~ 0	~~7~~ 0
8	~~9~~ 0	10	12
13	~~14~~ 0	15	2

0 → originally in mat

0 → your changes

~~1~~ -1	0	~~6~~ -1	~~7~~ -1
8	~~9~~ -1	10	12
13	~~14~~ -1	15	2

→

0	0	0	0
8	0	10	12
13	0	15	2

```
int N = mat.size()
    ↓
    rows

int M = mat[0].size()
    ↓
    cols
```

① Iterate in all rows, make elements of row as -1, if any ele in that row is 0.

② Repeat process for cols

③ Finally traverse matrix, make all -1 into 0.

```
void rowToColZero(int mat[N][M]){

    for(r=0; r<N; r++){
        bool flag=F
        for(c=0; c<M; c++){
            if(mat[r][c]==0) flag=T
        }
        if(flag==T){
            for(c=0; c<M; c++){
                if(mat[r][c]!=0)
                                mat[r][c]=-1
            }
        }
    }
for (c=0 ; c<M ; c++){
    bool flag =F
    for (r=0 ; r<N ; r++ ){
        if (mat[r][c] ==0) flag=T
    }
    if(flag==T){
        for (r=0; r<N; r++){
            if (mat[r][c]!=0)
                mat[r][c]=-1
        }
    }
}

for (r=0 ; r<N ; r++)
    for (c=0 ; c<M ; c++)
        if (mat[r][c] ==-1)
            mat[r][c]=0

}
```

TC: O(NM)
SC: O(1)