2D Matrix

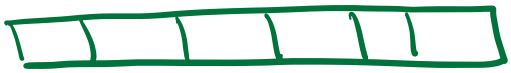
Print row wise / col wise sum

Principal Diagonals

Matrix Transpose

Matrix Inverse

1D arrays : list of elements



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

Store elements arranged in row and cols

Total dc
= N * M

```
int mat [N] [M]
```

- int → data type
- mat → matrix name
- N → rows
- M → cols

	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

```
int mat [4] [3]
```

- 4 → rows → horizontal
- 3 → cols → vertical

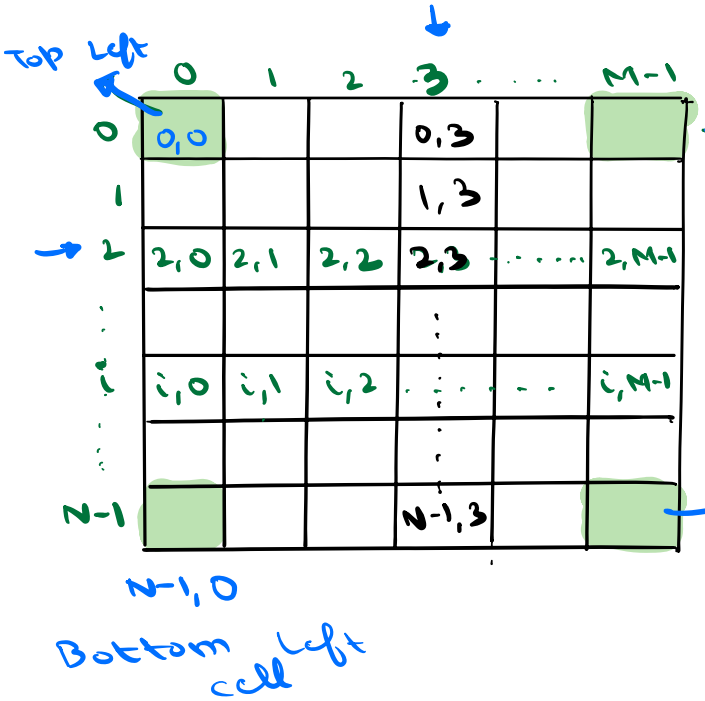
Total dc = 12
= 4 * 3

```
mat [i] [j]
```

- i → row no
- j → col no

int mat[N][M]

rows cols



mat[N][M]
rows col

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

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

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

mat[3][4]

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

output

10
26
42

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

mat [N][M]

```
for (row = 0 ; row < N ; row++) {
    sum=0
    for ( col = 0 ; col < M ; col++) {
        sum += mat [row][col]
    }
    print (sum)
}
```

fixed changing

TC: O(N*M)

SC: O(1)

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

mat[3][4]

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

Output-> 15 18 21 24

Traverse each col, while traversing, calculate sum of all elements present in that column

```
for (col = 0 ; col < M ; col ++ ) <  N x M
    sum = 0
    for (row = 0 ; row < N ; row ++ ) <
        sum += mat[row][col]
    >
    print (sum)
>
```

TC: O(N * M)

SC: O(1)

Given a 2D square matrix[N][N], print diagnol elements from left to right.

rows = cols

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

2 diagnols in square mat

① Principal Diagonal

Top Left → Bottom Right

1, 5, 9

1 → 5 → 9

0,0 → 1,1 → 2,2

mat[i][j]

i == j

② Anti Diagonal

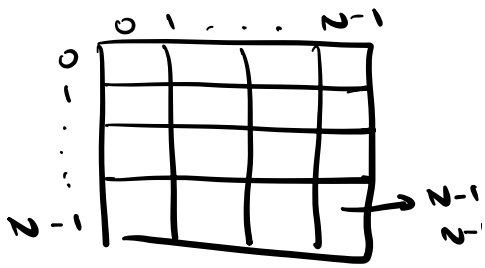
Top Right → Bottom Left

3, 5, 7

Print principal diagonal

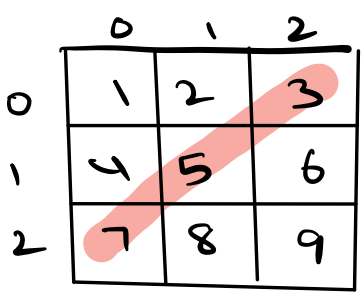
TL $\xrightarrow{+1}$ 0,0 $\xrightarrow{+1}$ 1,1 $\xrightarrow{+1}$ 2,2 → ..... → BR N-1, N-1

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



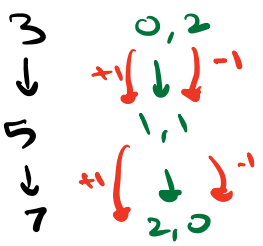

```
for (i=0 ; i < N ; i++) {
    print (mat[i][i])
}
```

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



Print anti-diagonal

row ++ , col --

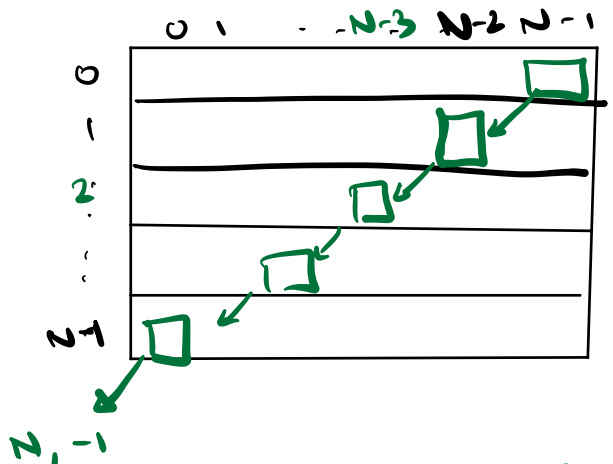


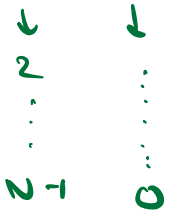
mat[N][N]

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

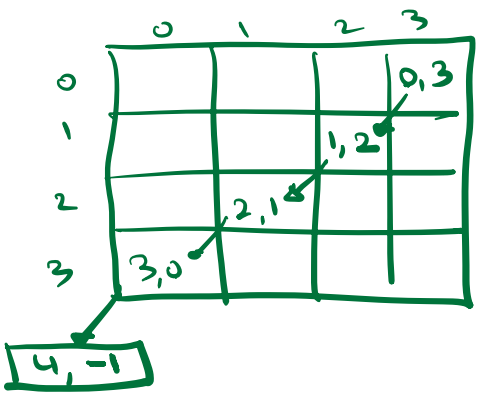
```
int i=0, j=N-1
while( i<N && j>=0 ) {
    print (mat[i][j])
    i++   j--
}
```

i	j
0	N-1
↓	↓
1	N-2





10:17

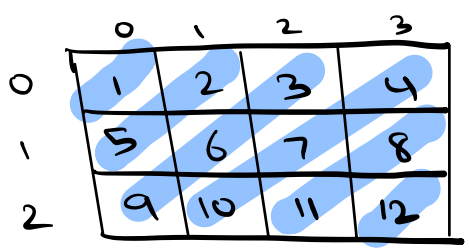


4 x 4
N = 4
N-1 = 3

Print all anti diagonals of a non-square matrix. R → L

rows != cols

N x M
mat[N][M]



mat[3][4]

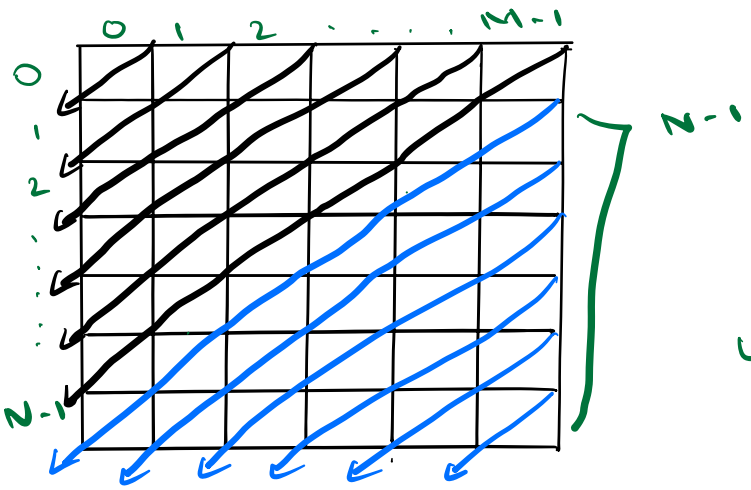
cnt = 6

Output

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

Every cell in 0th row is starting a diagonal

Every cell in last col is starting a diagon



Total diagonals
= M + N - 1

mat[N][M]

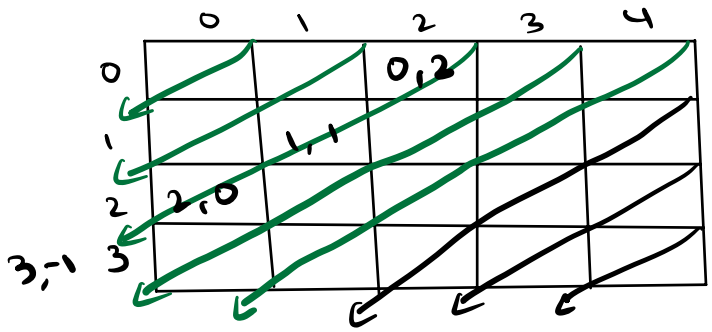
0th row → M cells

Last col → N cells

M + N - 1
↓ ↓
0th row last col

Q. Given mat[4][5], cnt of right to left (anti) diagonals?

0th row → 5 cells → 5
Last col → 4 cells → 3 (+)
8 diagonals



① 0th row

② Last col

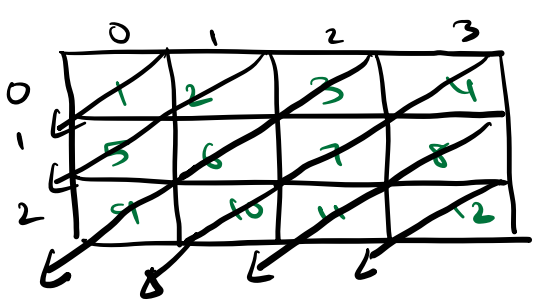
(0,0)

(0,1) → (1,0)

(0,2) → (1,1) → (2,0) → 3,-1 ✓ ✗

```
for (col=0 ; col < m ; col++) {
        i=0     j = col
    while(i<N && j>=0 ) {
        print(mat[i][j])
        i++   j--
    }
    print("\n")
}
```

```
for (row=0 ; row < N; row++) {
        i=row   j= M-1
    while(i<N && j>=0 ) {
        print(mat[i][j])
        i++   j--
    }
    print("\n")
}
```



i=0 N = 3 M = 4

i j

(0,0) → (1, -1)

(0,1) → (1,0) → (2,-1)

(0,2) → (1,1) → (2,0)

↓

(3,-1)

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

(0,3) → (1,2) → (2,1)
↓
(3,0)

$j = M-1$ $\quad j = 3$

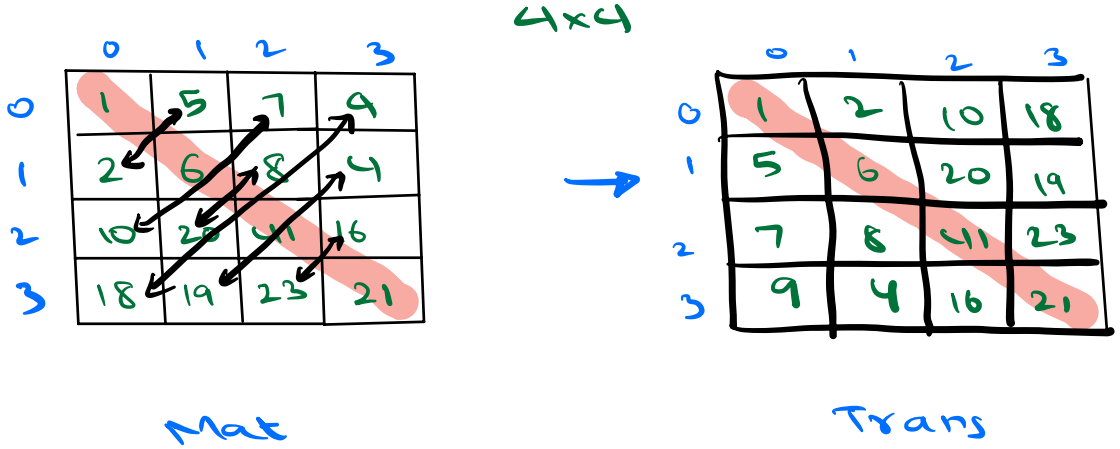
TC: O(N*M)
SC: O(1)

i j
(1, 3) → (2, 2) → (3,1)

(2,3) → (3,2)

Q. Given a square mat[N][N], find transpose

4×4



9
(0,3) ⟷ (3,0)

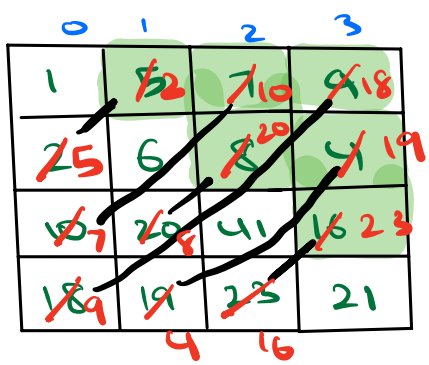
5 2
(0,1) ⇄ (1,0)

7 10
(0,2) ⟷ (2,0)

(i, j) ⟷ (j, i)

Just swap elements of upper Δ or lower Δ

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



N = 4

→ 0
→ 1
→ 2
→ 3

i j
0,1 0,2 0,3

1,2 1,3

2,3

3,4

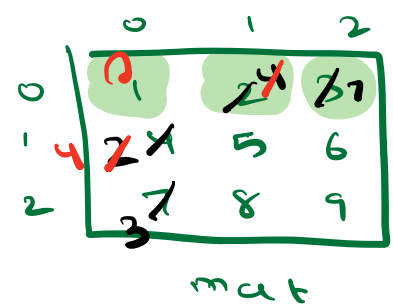
i = N-1 , j = N

	a	b
	7	10
1 swap	10	7
2 swap	7	10

j cannot start from 0

Trans

1	4	7
2	5	8
3	6	9



mat

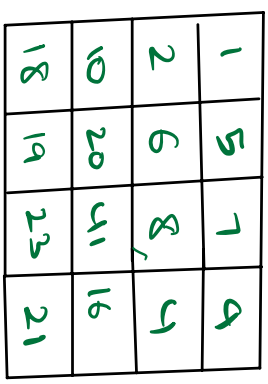
Total elements $= N^2$

No. of iterations $\simeq N^2/2$

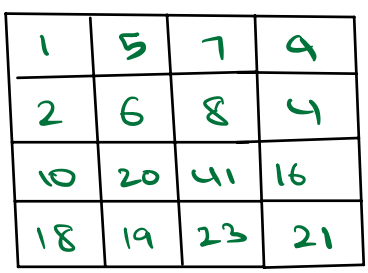
TC : $O(N^2)$

SC : $O(1)$

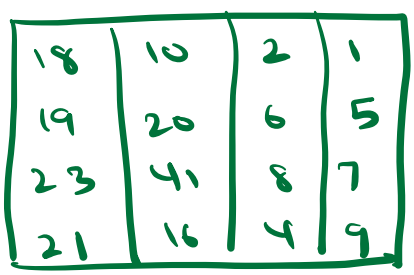
6. Given mat[N][N], rotate by 90° clockwise



Rotated mat



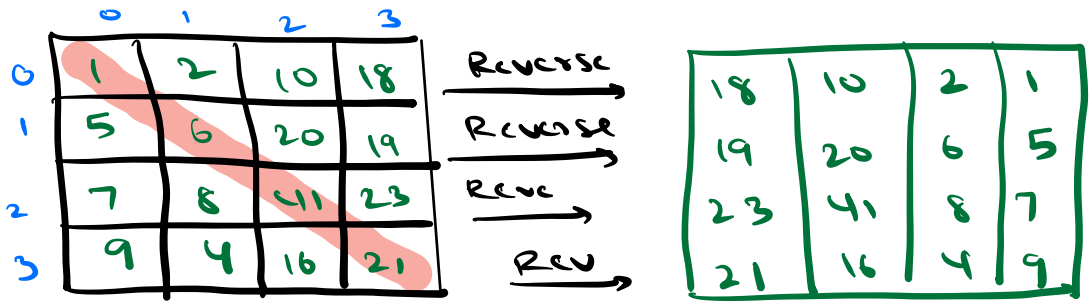
Mat



Rotated Mat

① Transpose

② Reverse each row



Rotated Mat

① Transpose → $N^2/2$

② Go to every row and reverse it. → $N^2/2$

TC : $O(N^2)$

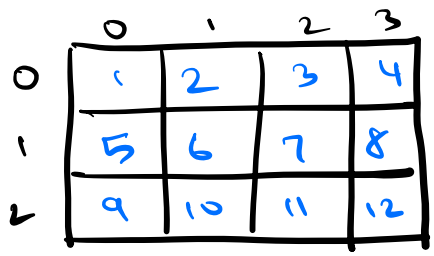
SC : $O(1)$

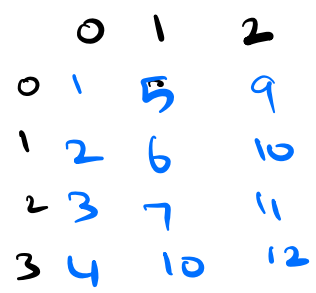
```
transpose (mat)
for (i=0 ; i<N ; i++)<
    reverse (mat[i])
|
```

Reverse 1 row → $\frac{\text{Row size}}{2}$ Iterations

1 row = $\frac{N}{2}$ itr

N row → $\frac{N^2}{2}$ itr



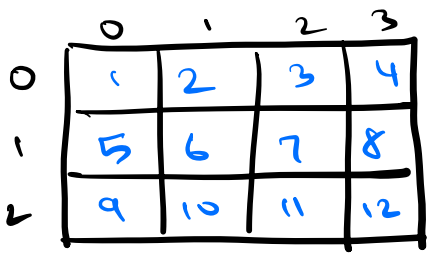


DOUBTS

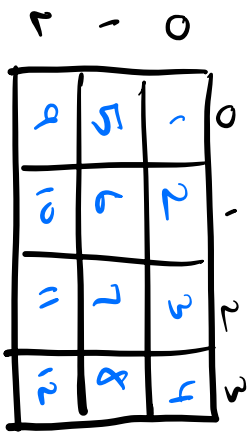
3 x 4

4 x 3

int mat[N][M] → trans[M][N]



Mat 3x4



Rot 4x3

int mat[N][M] → rot[M][N]

[1, 2, 3]

[[1], [1,2] [1,2,3]]

list of list

list of subarrays

for i s ⟶

for e in arr[i] ⟶

s e

for idx ⟶

res.insert(arr[idx])

output.insert(l)

[1, 2, 3]

[[1]
[1, 2]
[1, 2, 3]
[2]
[2, 3]
[3]]

rows: $\frac{N*(N+1)}{2}$

cols: N

mat[6][3]

	0	1	2
0			
1			
2			
3			
4			
5			

Ordered - HM

HM

0.01 → 2

0.03 → 2

0.06 → 1

0.7 → 1

0.9 → 1

(x)