20 Matrices

content

- Definition
- print row wise & col wise sum
- print diagonal & anti diagonal
- Diagonal traversal
- Transpose of a square matrix
- Rotate Matrix

2D Matrices Definition

An array of array is a 2D matrix.

A matrix of size N * M represents

N ⟶ no. of rows

M ⟶ no. of 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

4 rows and 3 cols

4 X 3

How to initialize ⟶ int[4][3]

rows

cols

Index of top right corner ⟶ [0, M-1]

Index of bottom right corner ⟶ [N-1, M-1]

Q> Given a 2D matrix $N \times M$.

Print the row wise sum

mat[4][3]

	0	1	2	output
0	1	2	3	→ 6
1	4	5	6	→ 15
2	7	8	9	→ 24
3	10	11	12	→ 33

```
for i ⟶ 0 to n-1 {
    total = 0
    for j ⟶ 0 to M-1 {   // iterate all cols
        total += mat[i][j]
    }
    print(total)
}
```

TC : O(N*M)

SC : O(1)

Q> Given a 2D matrix $N \times M$.
Print the col wise sum

mat[4][3]

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

output 22 26 30

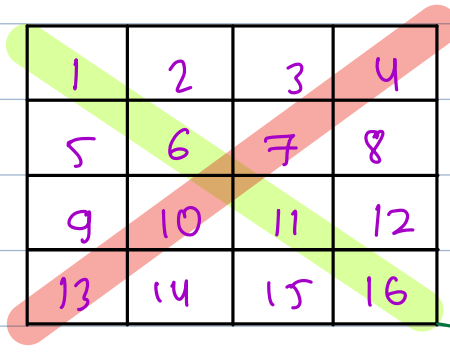
```
for c ⟶ 0 to M-1 {
    total = 0
    for r ⟶ 0 to N-1 {
        total += mat[r][c]
    }
    print(total)
}
```

TC : O(N*M)

SC : O(1)

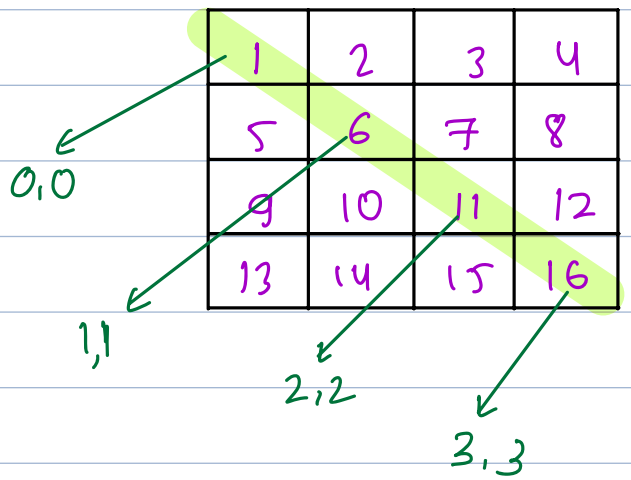
Q> Given 2D matrix mat[N][N] { square matrix }

Print the digonal and anti diagonal



anti diagonal → 4 7 10 13

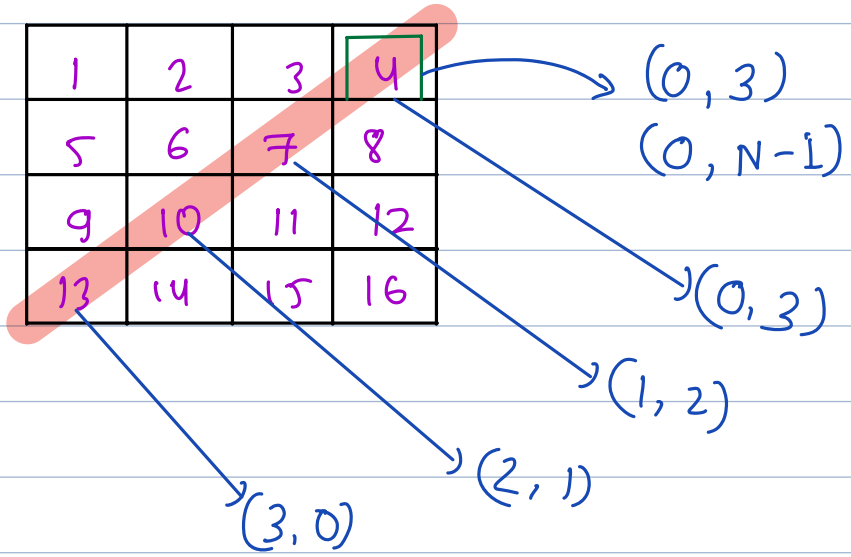
diagonal → 1 6 11 16



Observation r == c for the diagonal

```
void printDiagonal (int[][] mat) {
    N = mat.length      // no. of rows
    M = mat[0].length   // no. of cols

    r = 0   c = 0
    while ( r < N && c < N ) {      // square matrix N == M
        print (mat[r][c])
        r++
        c++
    }
}
```

TC : O(N)

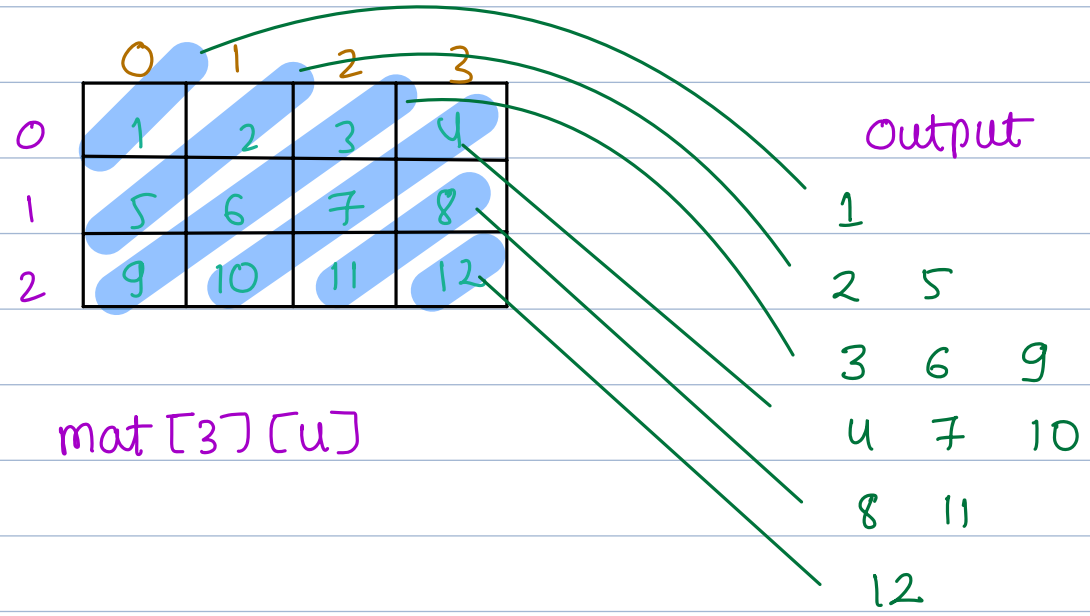


Observation ⟶ row gets incremented and column gets decremented.

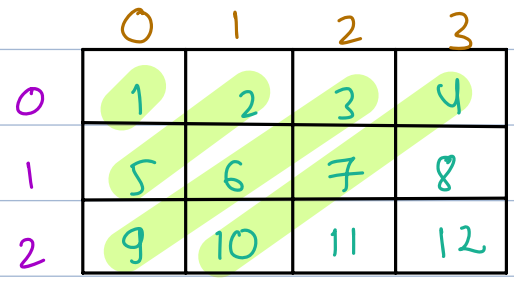
```
mat[N][N]
r = 0    c = n-1

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

TC : O(N)

Given a 2D matrix mat[N][M].
Print all the anti diagonals from right to left.

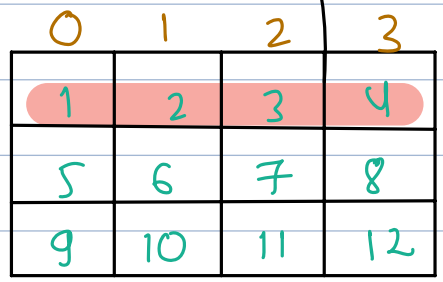


No. of diagonals of matrix[N][M] = N+M-1

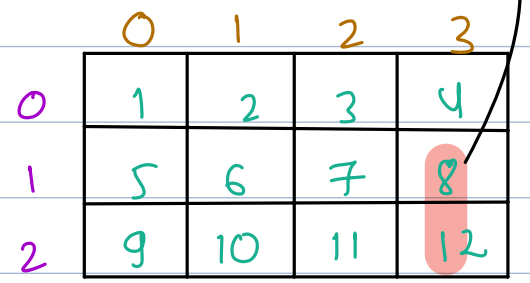


output
1
2 5
3 6 9
4 7 10

```
for j ⟶ 0 to M-1 {
    r = 0    c = j
    while ( r < N && c >= 0 ) {
        print ( mat[r][c] )
        r++
        c--
    }
    print ("\n")
}
```



```
for i ——→ 1 to N-1 {
    r = i     c = M-1
    while ( r < N && c >= 0 ) {
        print ( mat[r][c] )
        r++
        c--
    }
    print ("\n")
}
```



Output 8 11

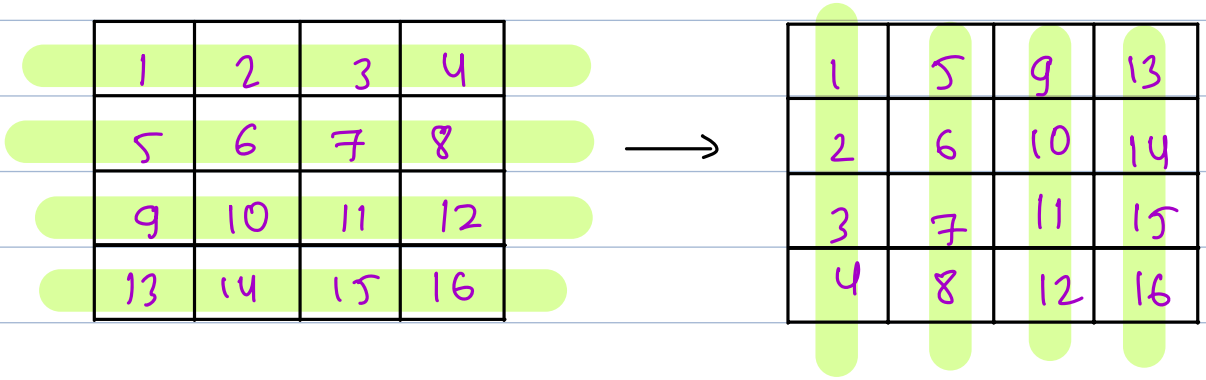
12

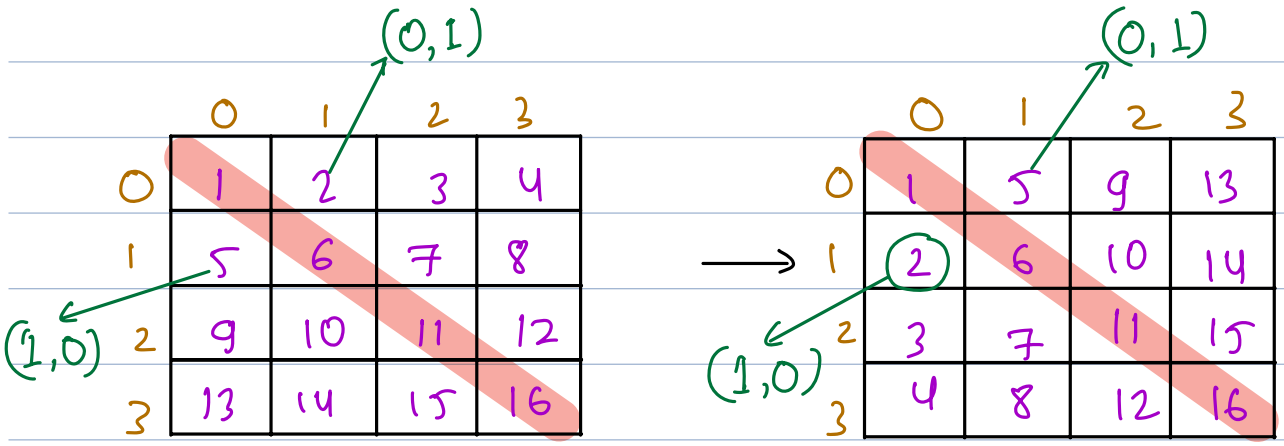
TC : O(N*M) ∵ we print all values exactly once

SC : O(1)

Break : 22:46

Q> Given a 2D matrix N x N {square matrix}
Find the transpose



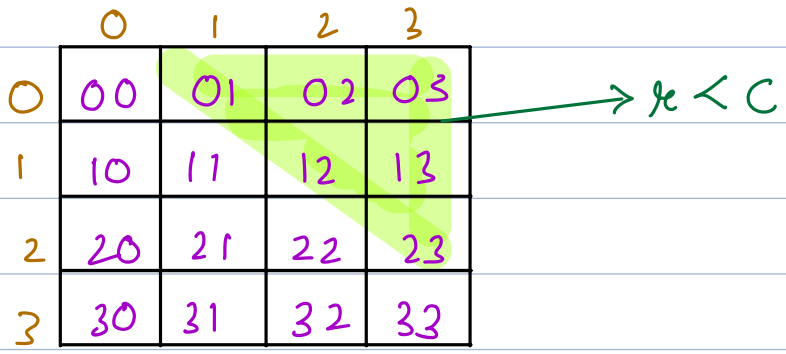


The i, j pairs have been swapped across diagonal
A[i][j] ⟶ A[j][i]

Pseudocode

```
for i ⟶ 0 to N-1 {
    for j ⟶ 0 to M-1 {
        swap (mat, i, j)
    }
}
```

∵ this will finally return original matrix.



alternate.
[r+1, N-1]

```
for r ⟶ 0 to N-1 {
    for c ⟶ 0 to N-1 {
        if (r < c) {
            swap (mat, r, c)
        }
    }
}
```

```
void swap (int[][] A, int r, int c) {
    temp = A[r][c]
    A[r][c] = A[c][r]
    A[c][r] = temp
}
```

TC : $O(N^2)$

SC : $O(1)$

Given a 2D matrix N x N { square matrix }

Rotate by 90° ↻ clockwise

input

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

transpose ↓

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

output

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

// step 1 transpose the matrix

```
for r ⟶ 0 to N-1 {
    reverse ( mat[r] )
}
```

TC : O(N²)

SC : O(1)

```
void reverse (int[] A) {
    l = 0      r = A.length - 1
    while (l < r) {
        swap (A, l, r)
        l++
        r--
    }
}
```

write the actual code for this.

}

Lets say rotate 90 (int[][] mat) rotates the matrix by 90°

180 ⟶ call rotate 90° twice

270 ⟶ call rotate 90 thrice

360 ⟶ Don't call

TC ⟶ $O(N^2 + N^2)$ ⟶ $O(N^2)$

TC ⟶ $O(N^2 + N^2 + N^2)$ ⟶ $O(N^2)$

Doubt Session

mat = [[1, 2], 0
[3, 4], 1
[5, 6] 2
]

mat [0] = [1, 2]