Arrays - 2D Matrices

Java

Syntax

```
int [][] a =    {
        { 10, 20, 30 },
        { 40, 50, 60 }
    };
```

Python

Syntax

```
a = [
    [10, 20, 30],
    [40, 50, 60]
]
```

	0	1	2
0	10	20	30
1	40	50	60

a[1][1] → 50

a[1][2] → 60

a[0][1] → 20

```
int [][] a = new int [5][6]
```

Rows

Columns

List Comprehension

```
a = [ [0] * 6  for i in range(5)]
```

Columns

Rows

OR

```
a = []
for i in range(5):
    a.append([0] * 6)
```

6 columns

5 rows

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	.	.	-	-
2	.	.	-	.		
3						
4						

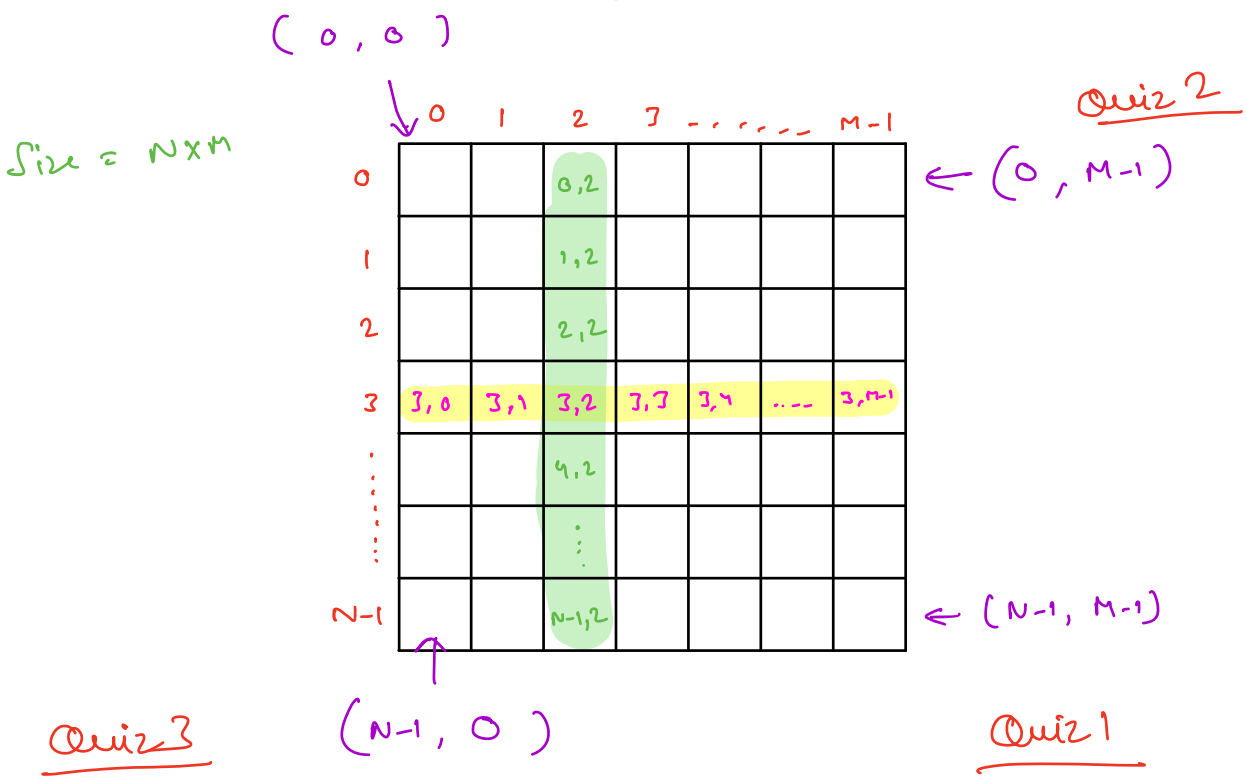
How to declare a matrix of size N * M ?

```
int [][] a = new int[N][M];
```

Rows — Columns

```
a = [[0] * M for i in range(N)]
```

Columns — Rows



When we iterate over a row, column no changes from (0 to M-1)

When we iterate over a col, row no changes from (0 to N-1)

Q1. Given a mat[N][M], print row-wise sum.

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

mat [3] [4]

	0	1	2	3	
0	3	8	9	2	→ 22
1	1	2	3	6	→ 12
2	4	10	11	17	→ 42

Row - i

Column - j

Iterate over each row & compute its sum

```
for (i=0; i<n; i++) {
    // Sum of ith row
    sum = 0
    for (j=0; j<m; j++) {
        sum += arr[i][j]
    }
    print(sum)
}
```

Q2. Given a mat[N][M], find max column-wise sum.

mat [3] [4]

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

	0	1	2	3
0	-3	-8	-9	-2
1	-1	-2	-3	-6
2	-4	-10	-11	-8
	-8	-20	-23	-16

↑ Ans

1) Iterate over each column & compute its sum.
2) Store the max sum.

```
ans = - ∞
for ( j=0 ; j < M ; j++) {
    // Sum of jth column
    sum=0
    for ( i=0 ; i<N ; i++) {
        sum += mat[i][j]
    }
    ans = max( sum, ans)
}
return ans
```

ans = $-\infty$

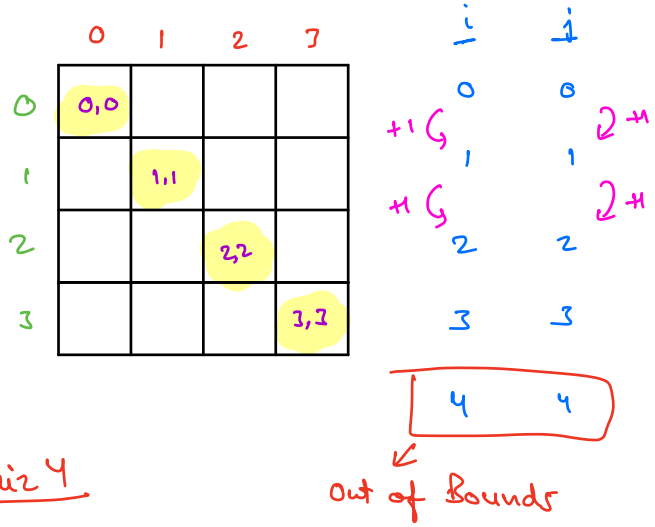
ans = - float ("inf") or float ("-inf")

Q3 Given a mat[N][N], print diagonal elements.

Square matrix

→ Left diagonal

→ Right diagonal

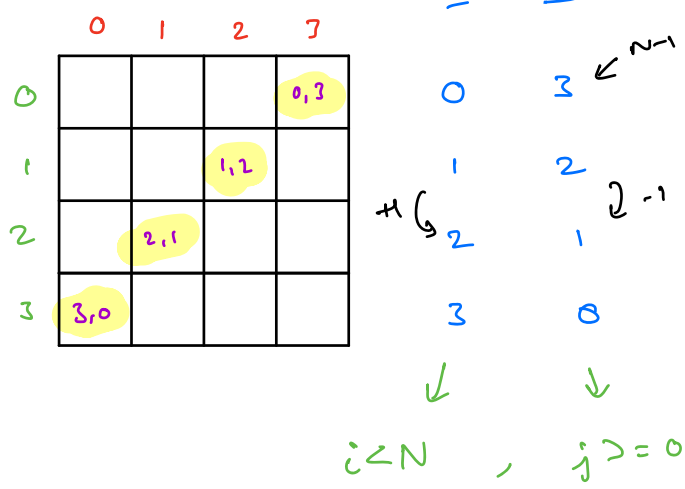


Quiz 4

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

TC: O(N)

SC: O(1)



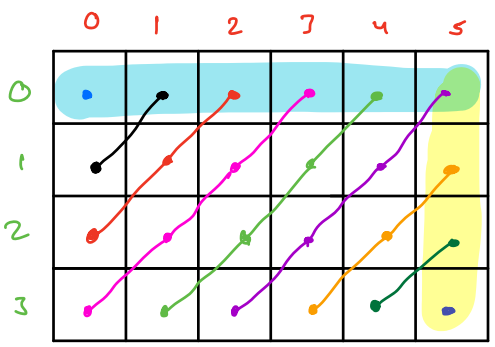

```
i, j = 0, N-1
while( j>=0 ) {
    print(mat[i][j])
    i++
    j--
}
```

TC: O(N)

SC: O(1)

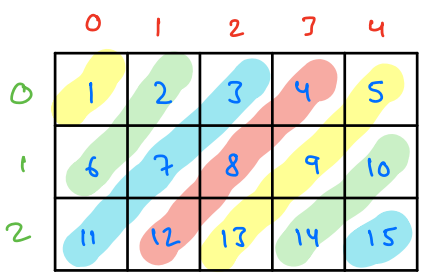
Q4 Given a mat[N][M], print diagonal elements going R-L.

mat [4] [6]



Print all diagonals going R to L, starting from 0^{th} row & $(M-1)^{th}$ col

mat [3] [5]



Output

1

2 6

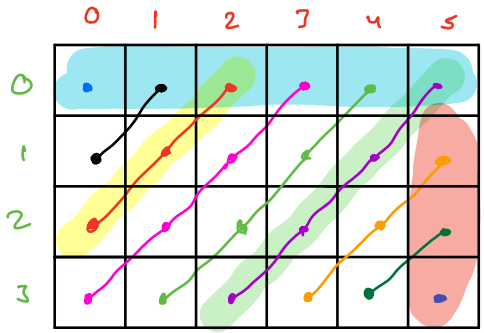
3 7 11

4 8 12

5 9 13

10 14

15



Row No. x	Col No. y
0	5
1	4
2	3
3	2
4	1

Out of bounds

Row No. x	Col No. y
0	2
1	1
2	0
3	-1

Out of bounds

Row No. $x < N$ and Col no. $y >= 0$

Pseudocode

```
// Iterate over 0th row
for (j = 0; j < M; j++)
    // Cell [0, j]
    x = 0, y = j
    while (x < N and y >= 0) {
        print( A[x][y])
        x = x+1
        y = y-1
    }
}
```

```
// Iterate over last col / (M-1)th col

for ( i = 1 , i<N ; i++ ) {

    // Cell [i, M-1]
    x = i ,  y = M-1

    while (x<N and y>=0) {
        print ( A[x][y])
        x = x+1
        y = y-1
    }

}
```

To prevent repetition of [0, M-1]

TC : O (N * M)

SC : O (1)

Break till 10: 20 PM

Q5 Given a mat[N][N], find the transpose inplace.

Square matrix

↓ Change the given matrix itself

Expected SC: O(1)

Row ↔ Cols

mat [5] [5]

Quiz 5

	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

	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

mat [0] [1] ⟷ mat [1] [0]

mat [3] [4] ⟷ mat [4] [3]

mat [i] [j] ⟷ mat [j] [i]

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

Work ??

Not work

N=5

i, j

0,0 ↔ 0,0

0,1 ↔ 1,0

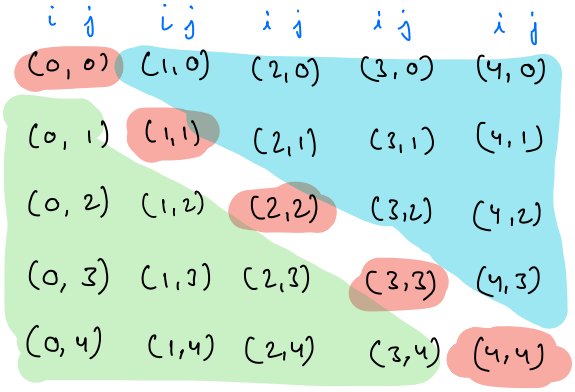
0,2 ↔ 2,0

0,3 ↔ 3,0

0,4 ↔ 4,0

1,0 ↔ 0,1

1,1 ↔ 1,1



2 swaps will nullify each other

→ TODO: write correct code on your own

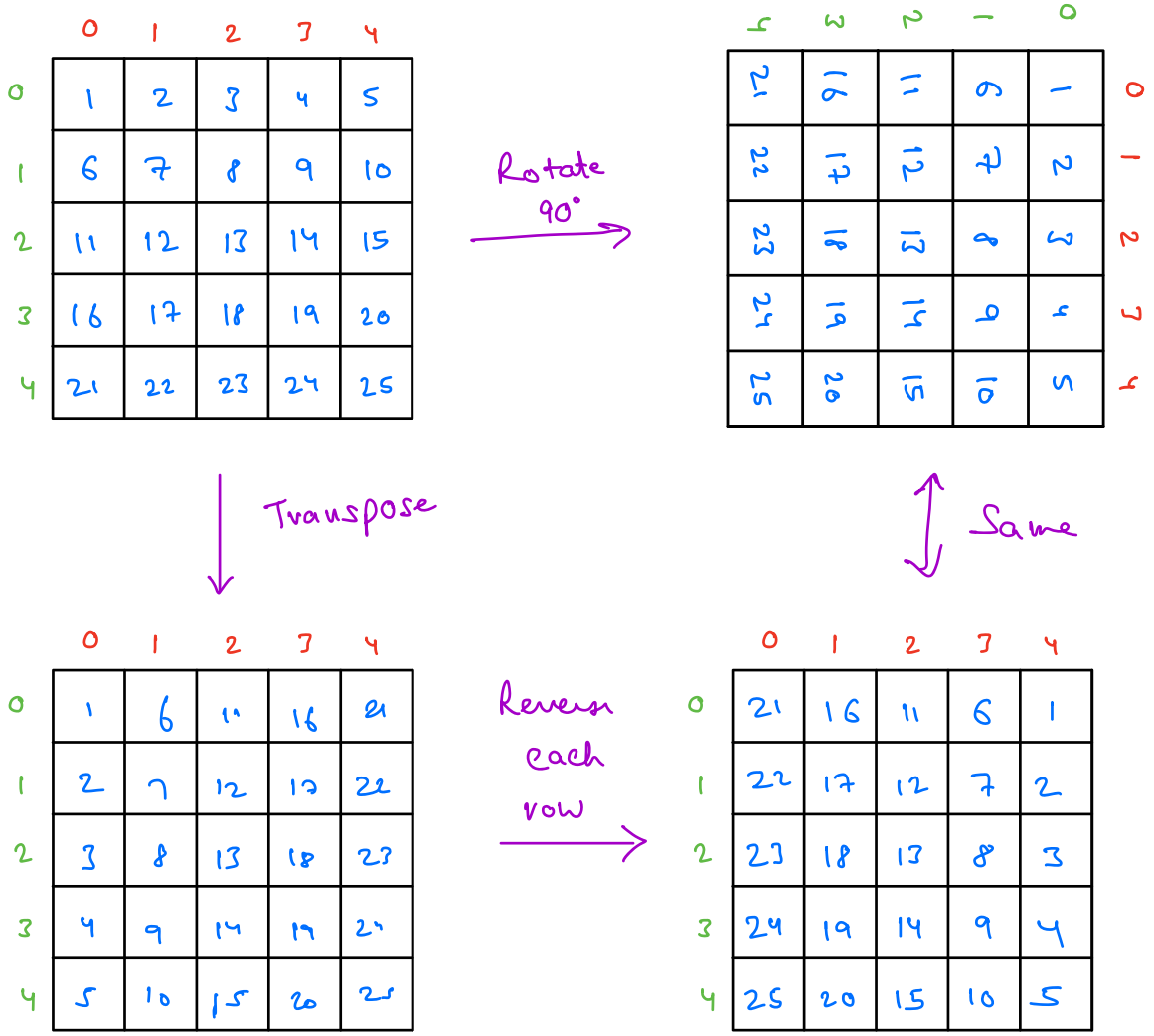
TC : $O(N^2)$

SC : $O(1)$

Q6 Given a mat[N][N], rotate it by 90 degrees, in clockwise direction.

Expected SC: O(1)

mat [5] [5]



1. Take the transpose of the given matrix

2. Reverse each row.

Quiz 6

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

Rotate 90° →

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

↕ Same

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

Code - Try it out.

Doubts

Thank You

Problem Solving Sessions on weekends

KPMG Report

Good Night

Thank You

Wednesday