

Array: 2D Matrices

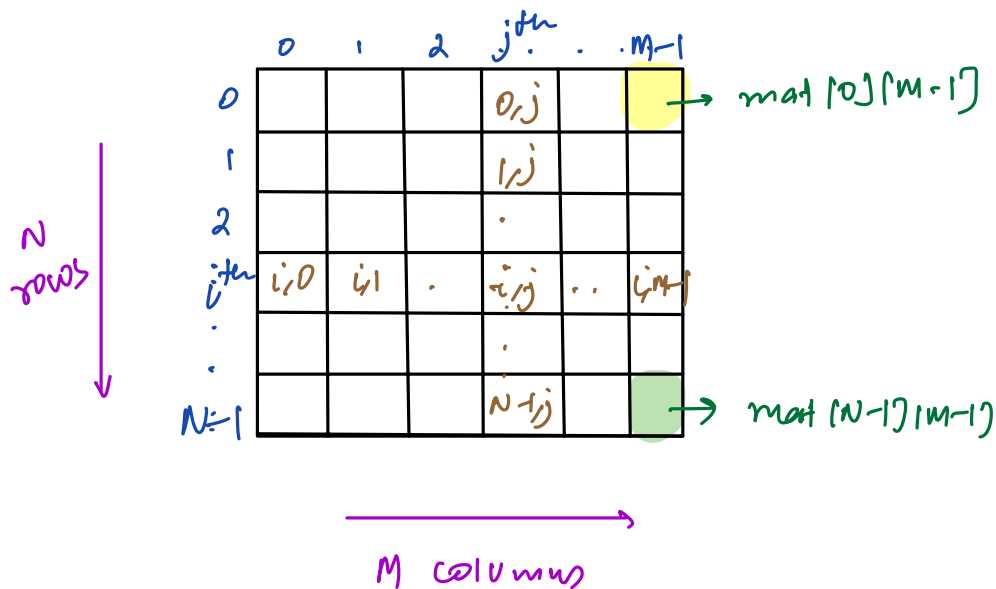
2D Matrix \rightarrow 2D Array

rectangular grid of values

Declare

`int mat[N][M]`
 \swarrow \searrow
 rows columns

`int (**) mat = new int[N][M]`



Observation

\rightarrow If we move in i^{th} row, row no. is constant but column will change $\rightarrow [0, M-1]$

\rightarrow If we move in j^{th} column, column no. is constant but row will change $\rightarrow [0, N-1]$

Question-1

Given $\text{mat}[N][M]$, print row-wise sum.

	0	1	2	
0	1	5	7	$\rightarrow 1+5+7 = 13$
1	2	8	9	$\rightarrow 19$
2	7	6	2	$\rightarrow 15$

```
def sumRow (mat [N][M]) {
```

```
    for (i = 0 to N-1) {
```

```
        // ith row
```

```
        sum = 0
```

```
        for (j = 0 to M-1) {
```

```
            sum += mat[i][j]
```

```
        }
```

```
        print (sum)
```

```
    }
```

$TC = O(N \times M)$

$SC = O(1)$

Question 2

Given $\text{mat}(N \times M)$, print column wise sum.

	0	1	2
0	1	5	7
1	2	8	9
2	7	6	2
	↓	↓	↓
	10	19	18

```
for (j=0 to M-1) {
```

```
    sum = 0
```

```
    for (i=0 to N-1) {
```

```
        sum += mat[i][j]
```

```
    }
```

```
    print(sum)
```

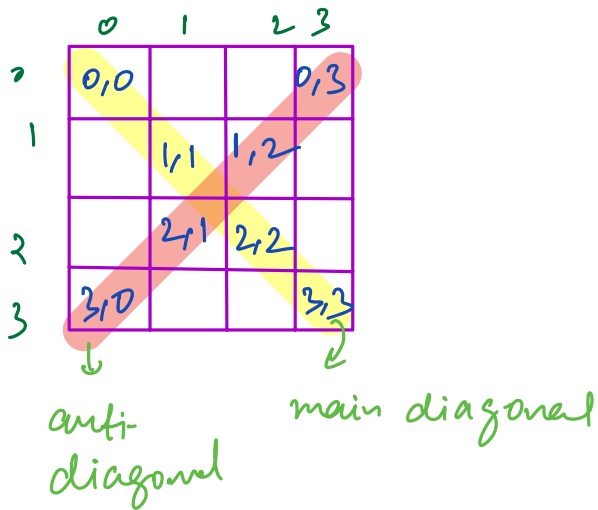
```
}
```

$TC = O(N \times M)$

$SC = O(1)$

Question 3

Given a square matrix $mat[N][N]$, print diagonals.



```
for (i=0 to n-1) {  
    for (j=0 to n-1) {  
        if (i==j)  
            print(mat[i][j])  
    }  
}
```

TC = $O(N^2)$

Main Diagonal :

```
for (i=0 to N-1) {  
    print(mat[i][i])  
}
```

TC = $O(N)$
SL = $O(1)$

Anti Diagonal

$$i + j = n - 1 \Rightarrow j = n - 1 - i$$

```
for (i=0 to N-1) {  
    print(mat[i][n-1-i])  
}
```

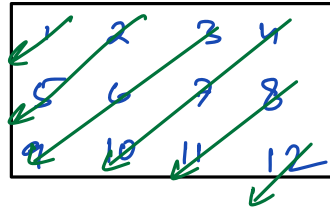
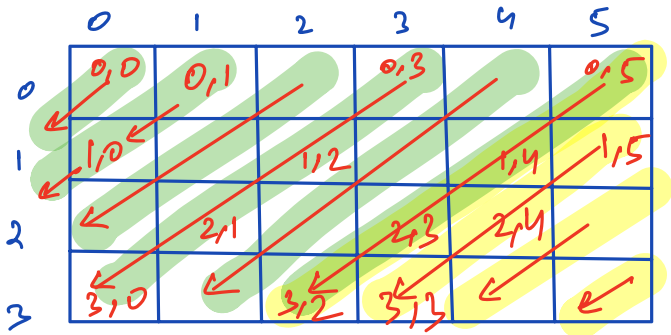
```
i=0, j=N-1  
while (i<N) {  
    print(mat[i][j])  
    i++, j--  
}
```

TC = $O(N)$ SL = $O(1)$

Question 4

Given $\text{mat}(N \times M)$, print all diagonals. going from right to left.

Diagonals should start from 0^{th} row OR $m-1^{\text{th}}$ col.



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

M diagonals from 0^{th} row

N diagonals from $M-1^{\text{th}}$ col.

1 overlap

$$\Rightarrow M + N - 1$$

```
printDiagonals( mat[N][M] ) {
```

// print all diagonals from 0th row

```
for( j = 0 to M-1 ) {
```

```
    r = 0, c = j
```

```
    while( r < N & c >= 0 ) {
```

```
        print( mat[r][c] )
```

```
        r++, c--
```

```
    }
```

```
    print( newline )
```

```
}
```

// print all diagonals from m-1th col.

```
for( i = 0 to N-1 ) {
```

```
    r = i, c = M-1
```

```
    while( r < N & c >= 0 ) {
```

```
        print( mat[r][c] )
```

```
        r++, c--
```

```
    }
```

```
    print( newline )
```

```
}
```

j	iterations
0	1
1	2
2	3
⋮	⋮
⋮	⋮
m-1	m
	$\frac{m(m+1)}{2}$

i	iterations
1	N-1
2	N-2
⋮	⋮
⋮	⋮
N-1	1
	$\frac{N(N-1)}{2}$

$$TC = O(N \times M)$$

$$SC = O(1)$$

Question 5

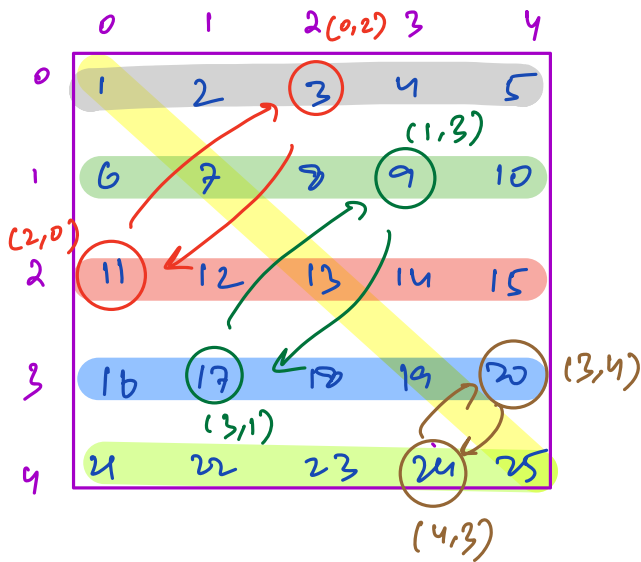
Given $\text{mat}[N][N]$, calculate transpose of the matrix w/o extra space

Transpose:

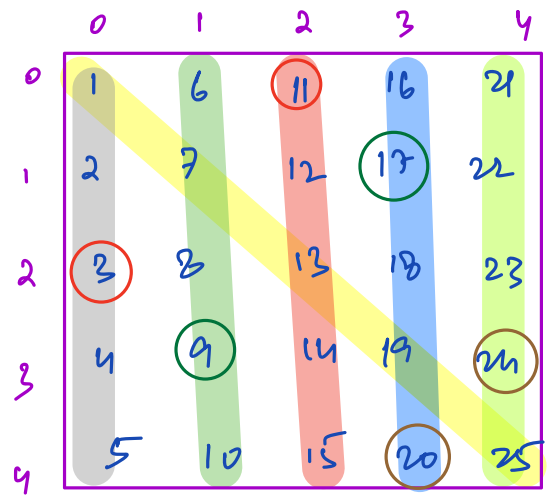
0th row \rightarrow 0th col

1st row \rightarrow 1st col

\vdots



transpose



$$[i, j] \longleftrightarrow [j, i]$$

$$[i, i] \longleftrightarrow [i, i]$$

```
for (i=0 to n-1) {
```

X will do double swap

```
  for (j=0 to n-1) {
```

```
    swap(mat[i][j], mat[j][i])
```

```
  }
```

```
}
```

i=0, j=1

[0,1] ↔ [1,0]

i=1, j=0

[1,0] ↔ [0,1]

```
for (i=0 to n-1) {
```

```
  for (j=i+1 to n-1) {
```

```
    swap(mat[i][j], mat[j][i])
```

```
  }
```

```
}
```

TC = $O(N^2)$

SC = $O(1)$

Question 6

Given $mat[N][N]$, rotate 90° clockwise from top-right.
SC = $O(1)$

	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

rotate
 \rightarrow
 90° clockwise

	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
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

90°
clockwise

0	1	2	3	4
21	16	11	6	1
22	17	12	7	2
23	18	13	8	3
24	19	14	9	4
25	20	15	10	5

transpose

0	1	2	3	4
1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

reverse
every row

rotate (mat[N][N]) {

mat = transpose (mat) → TODO → O(N²)

for (i = 0 to n-1) {

reverse (mat[i]) → TODO
↓
O(N)

TC = O(N²)

SC = O(1)

like reversing
an array