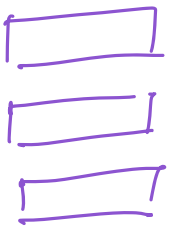
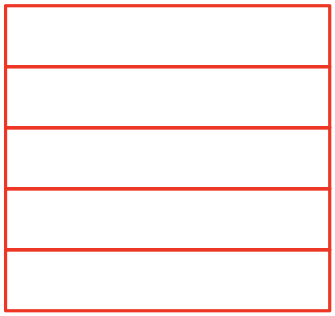
Arrays: 2D Matrices

2-D Matrix : Array of arrays

Array



Arrays



2D - Matrix

all arrays are of same size

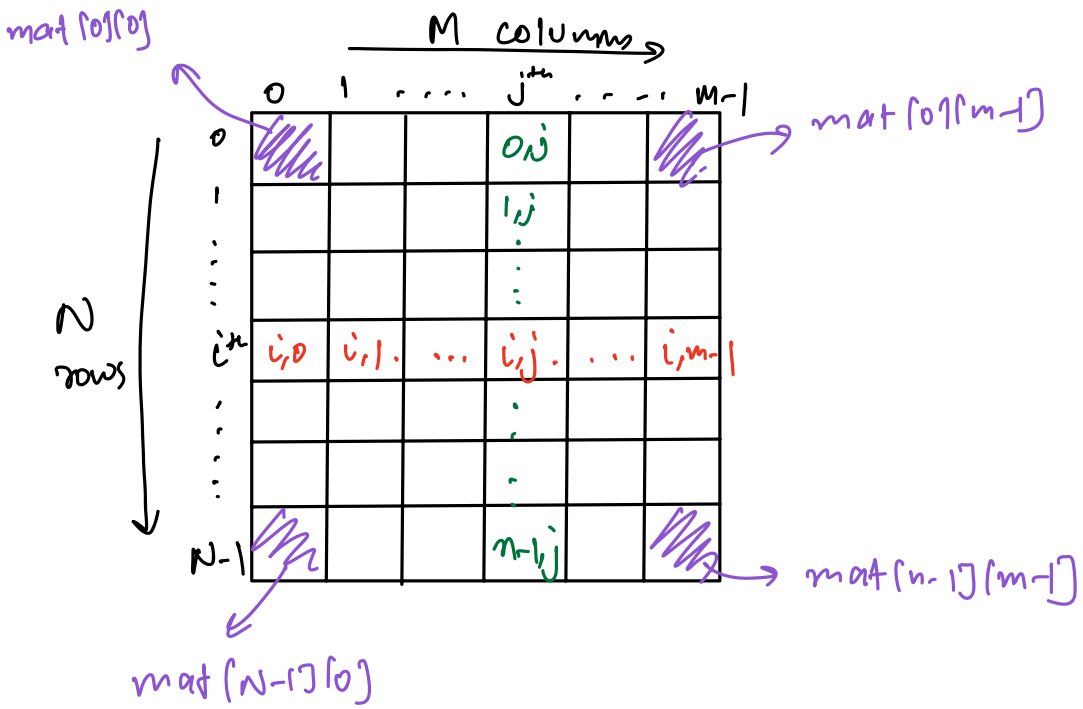
Say, size of each array in a matrix → M (no. of columns)

Total no. of arrays → N (no. of rows)

Declare : int mat[N][M] (TODO → learn in your own language)

N → no. of rows

M → no. of columns



Observation

→ If we move in i^{th} row, column index will change from $[0, m-1]$.

→ If we move in j^{th} column, row index will change from $[0, n-1]$.

Question 1

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

eg

	0	1	2	
0	1	5	9	→ 15
1	2	8	2	→ 12
2	7	6	2	→ 15

Code

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

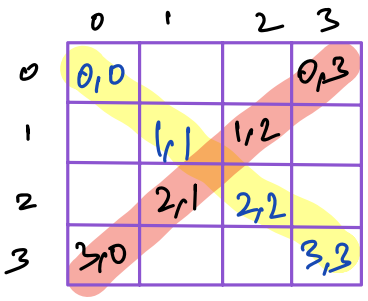
TC: $O(N \times M)$

SC: $O(1)$

Question 2

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

square matrix : (# of rows) = (# of columns)



Left → right diagonals

right → left

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

TC: $O(N^2)$

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

optional

OR

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

left → right

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

for right → left diagonal, $i+j = n-1$

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

optional

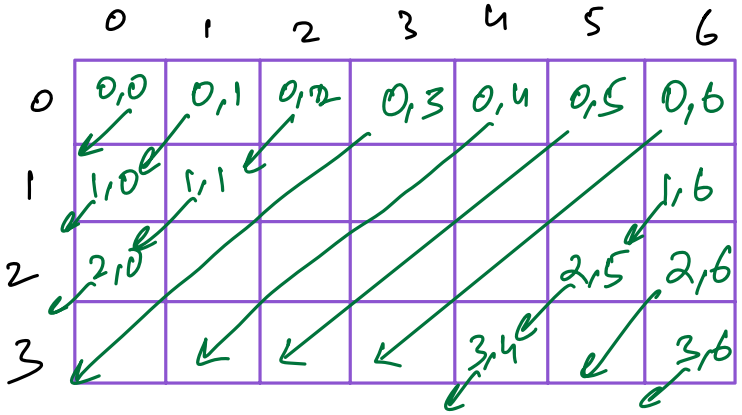
right → left

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

Question 3

Given mat[N][M], print all diagonals going from R → L.

Note: Diagonals will start from 0th row OR m-1th column.



mat[4][7]

Diagonal starting point:

(0,0) (0,1) (0,2) . ~ ~ . (0,6)

(1,6) (2,6) (3,6)

1. Print all diagonals from 0th row

```
for(k=0; k<m; ++k) {
    // starting: (0,k)
    i=0, j=k
    while(i<n && j>=0)
        print(mat[i][j])
        i++, j--
    }
    print(newline)
}
```

k = 0 - 6

k=0	i=0, j=0	(0,0)
1	i=0, j=1	(0,1) (1,0)
2	i=0, j=2	
3	;	
4	(	
5	(	
6		

2. Print all diognals from $m-1^{th}$ column

```
for ( K=1; K<n; ++K ) {
    i=K, j=m-1
    while(i<n && j>=0) {
        print(mat[i][j])
        i++, j--
    }
    print(newline)
}
```

K=1 i=1, j=6 (1,6) (2,5) (3,4)

2 :

3 :

total TC : O(N*M)

total SC : O(1)

BREAK: 8:14 - 8:24

Question 4

Given a mat[N][N], calculate transpose of a matrix without extra space.

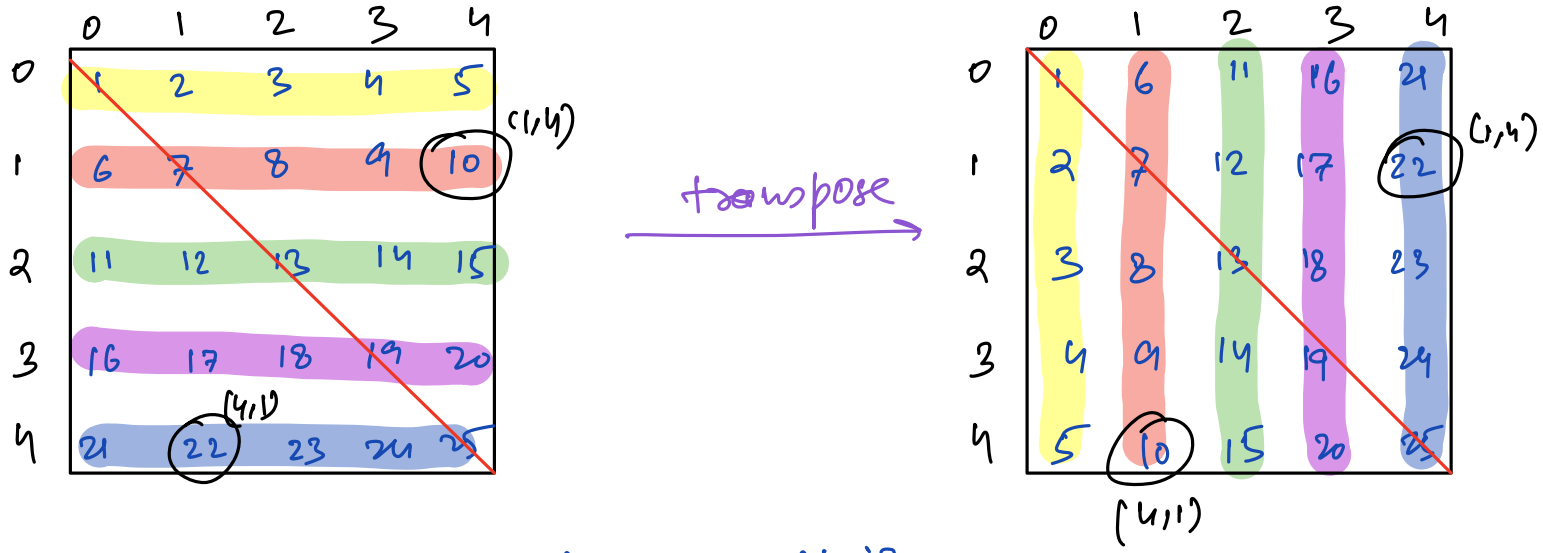
Transpose is:

0th row → 0th column

1st row → 1st column

⋮

n-1th row → n-1th column



mat[5][5]

[i,j] → [j,i]

Code

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

✗ Not work

}
}

i=1, j=3

swap(mat[1][3], mat[3][1])

⋮

i=3, j=1

swap(mat[3][1], mat[1][3])

double swapping

solution

either swap when i<j OR i>j

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

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

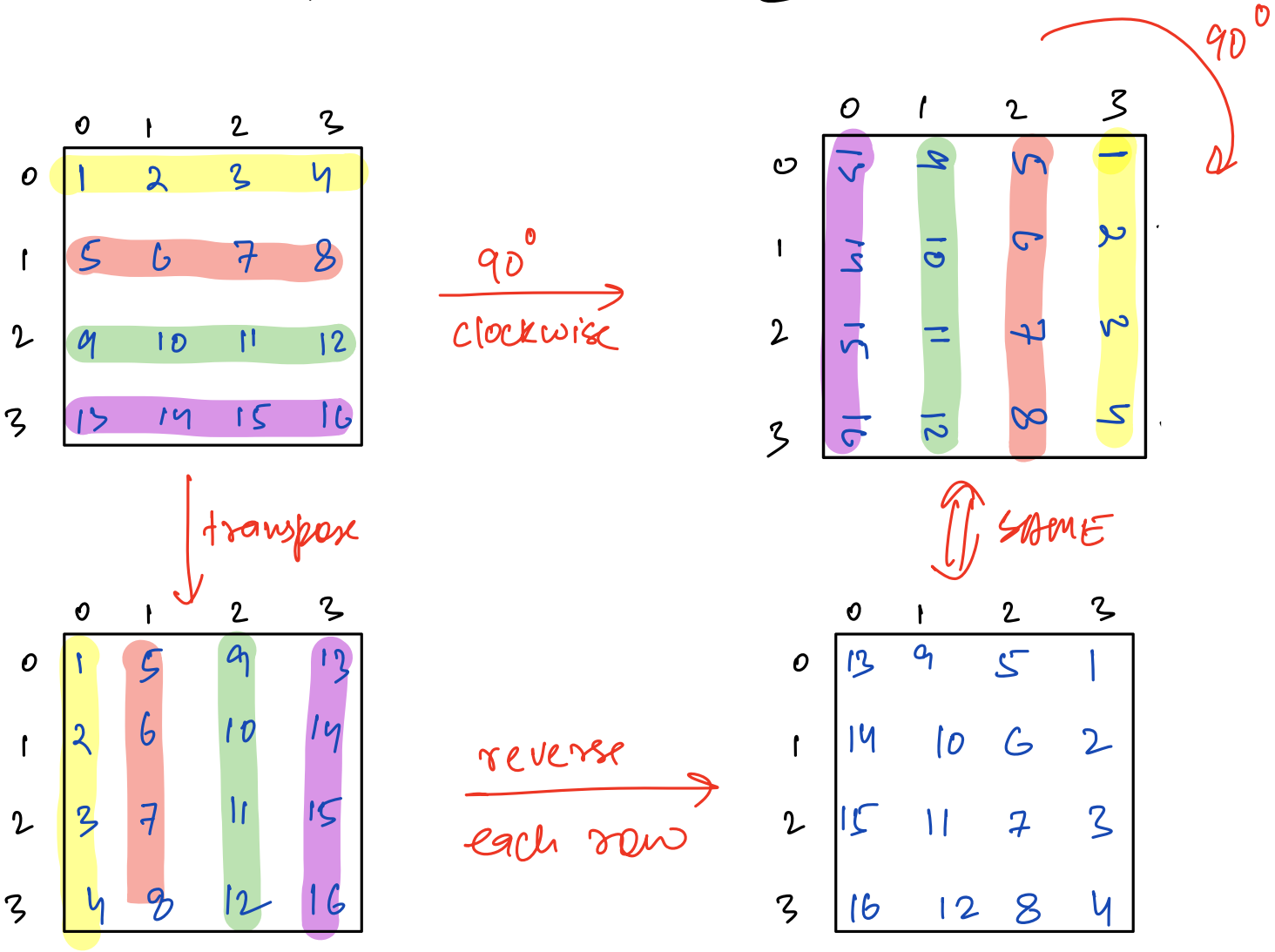
✓ WORK

TC: $O(N^2)$

SC: $O(1)$

Question 5

Given square matrix mat[N][N], rotate 90° clockwise from top left. SC: O(1)



90° clockwise rotation = transpose + reverse

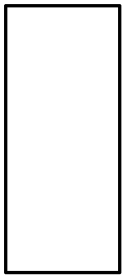
transpose: O(N²) TC, O(1) SC

reverse: O(N²) TC, O(1) SC

total TC: O(N²)

SC: O(1)

mat



mat X

Doubt

$10^5 \times 10^5$ ..... 1 times

10^5 times

$10^5 \times 10^5 = 10^{5+5} = 10^{10}$

$10^{5 \times 10^5}$

$\therefore 10^9 + 7$