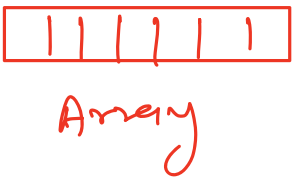
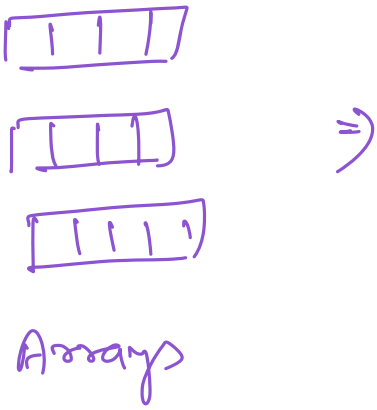
Array: 2D Matrices

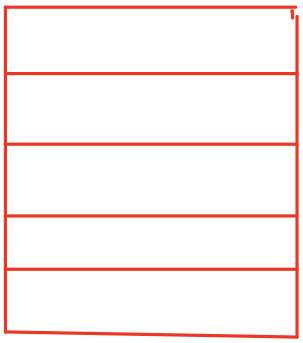
2-D Matrix : Array of arrays



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)

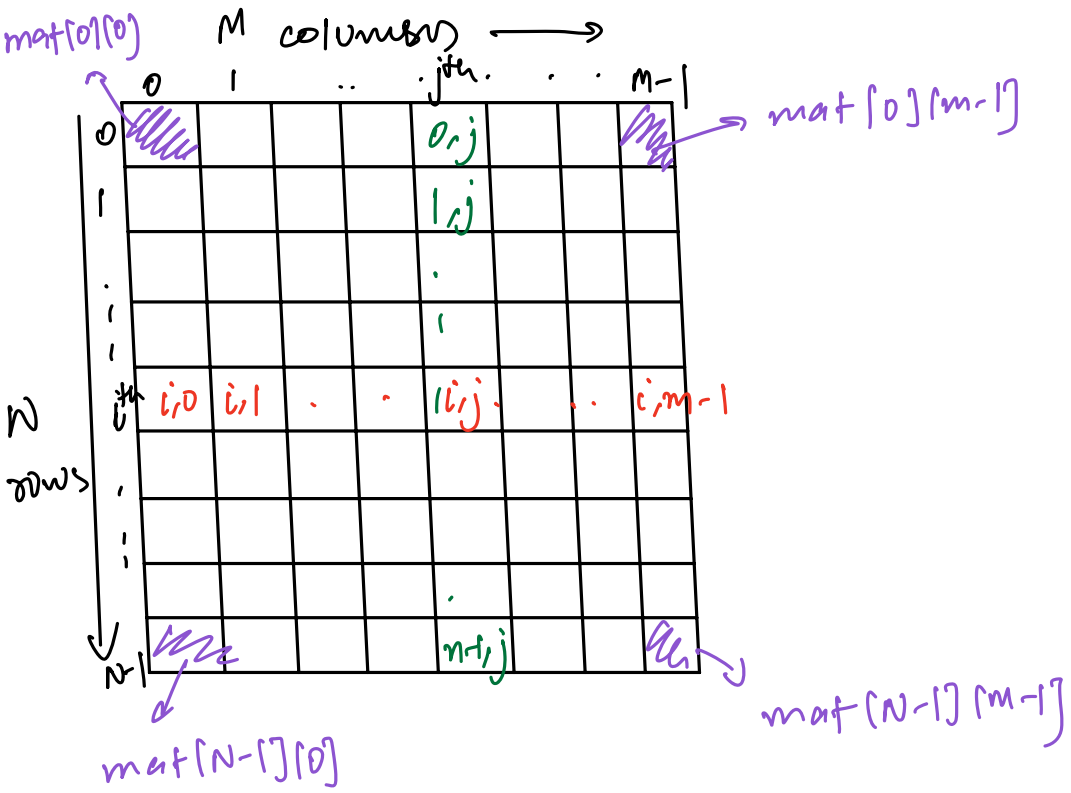
N x M matrix

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

3×3

Code

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

TC: O(N×M)

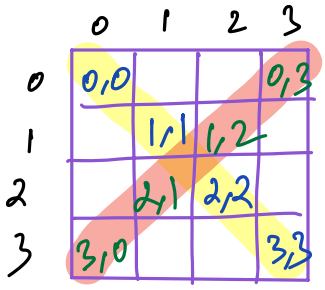
SC: O(1)

Question 2

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

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

N = M



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++
}
```

⇒

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

left → right

TC: $O(N)$

SC: $O(1)$

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

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

right → left ⇒

```
for(i=0; i<n; ++i) {
    print (mat[i][n-i-1])
}
```

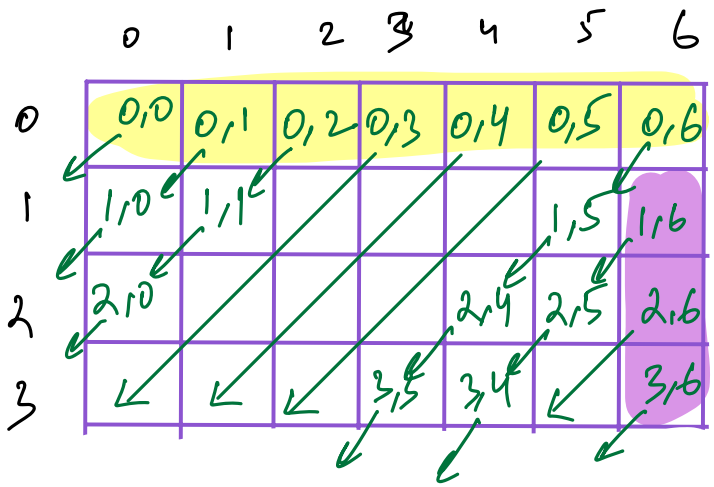
TC : O(N)

SC : O(1)

Question 3

Given mat[N][M], print all diagonals going from right → left.

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 0^{th} 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 diagonals from $m-1^{th}$ columns

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

total TC : O(N*M)

total SC : O(1)

BREAK: 10:07 - 10:17

Question 4

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

Transpose is:

0^{th} row → 0^{th} column

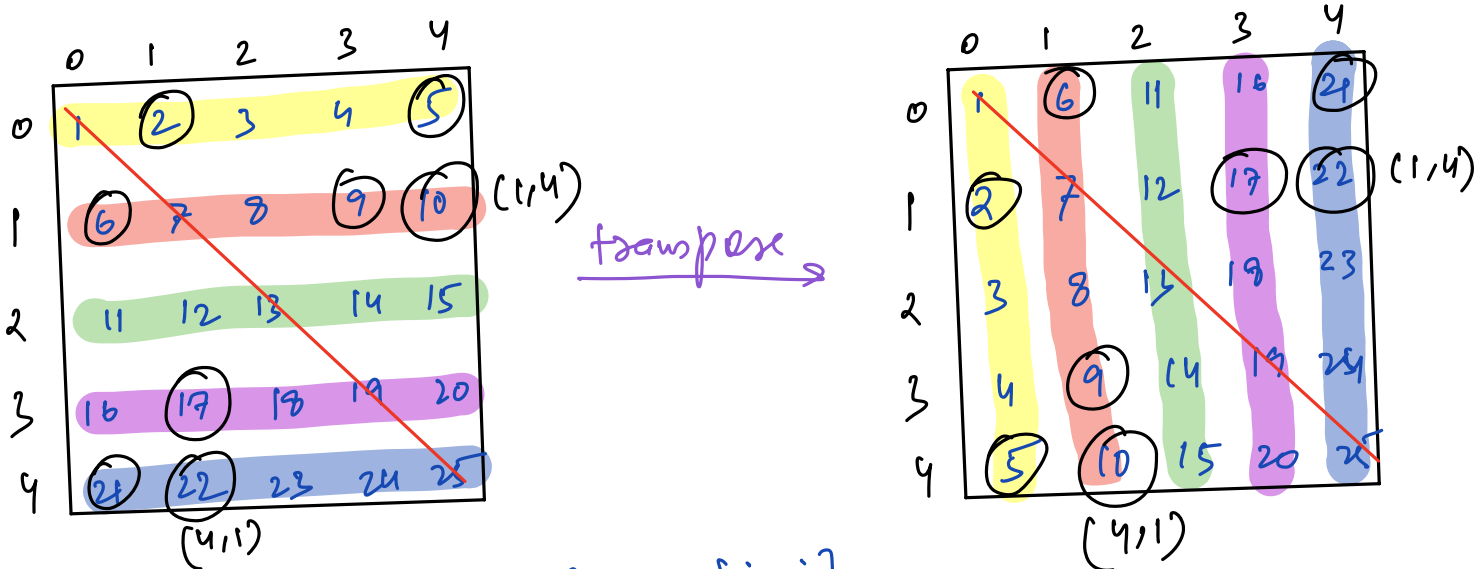
1^{st} row → 1^{st} column

.

.

.

$n-1^{th}$ row → $n-1^{th}$ column



[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])

double swapping

i=3, j=1

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

solution

either swap when j >= i or i >= j

optional

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

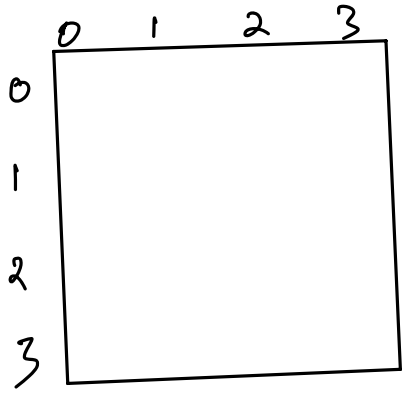
```
for(i=0; i<n; ++i){
    for(j=i+1; j<n; ++j){
        swap(mat[i][j], mat[j][i])
    }
}
```

Work

TC: $O(N^2)$

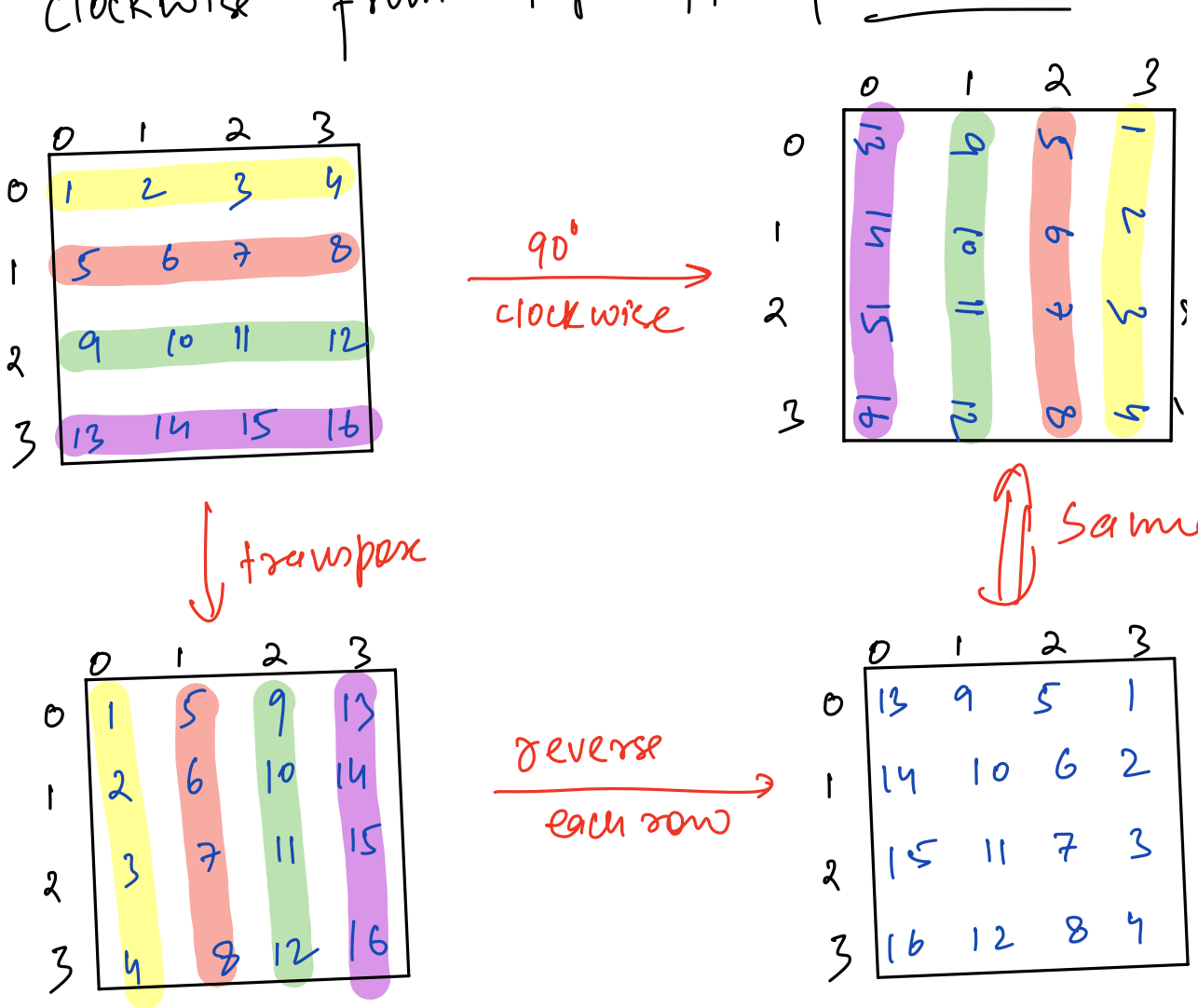
total iterations $= \frac{N^2 - N}{2}$

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 → TC: O(N²), SC: O(1)

reverse → TC: O(N²), SC: O(1)

total TC: O(N²)

SC: O(1)