2D Matrix

> Act
> as if what
> you do makes
> a difference.
> It does.
>
> —WILLIAM JAMES

Good

Evening ☺

Today's content

01. Matrix basics

02. Print row wise sum

03. Print col wise sum

04. Print Diagonal of square matrix

05. Print all diagonal of Rectangular matrix

06. Transpose a matrix

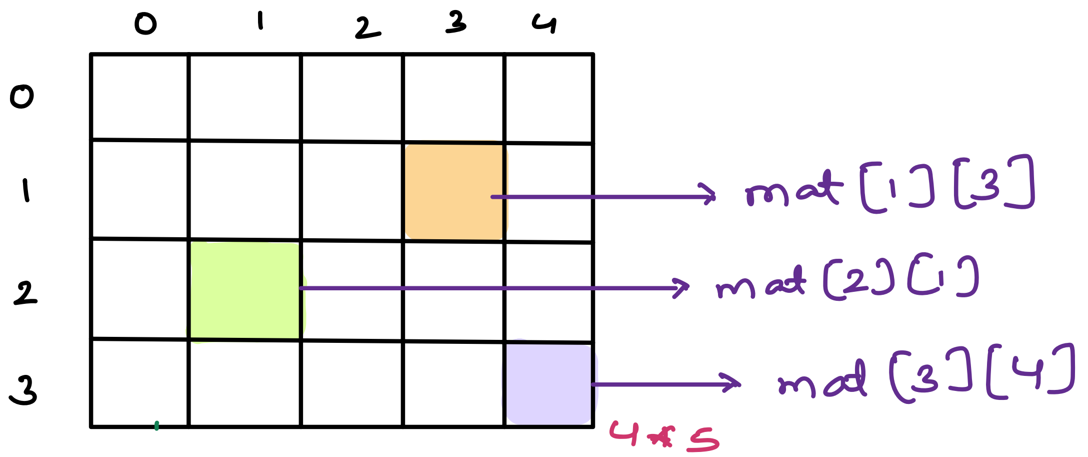
07. Rotate matrix by 90°

Matrix → 2D Array → Array of Arrays

Declaration :- int mat [4] [5];

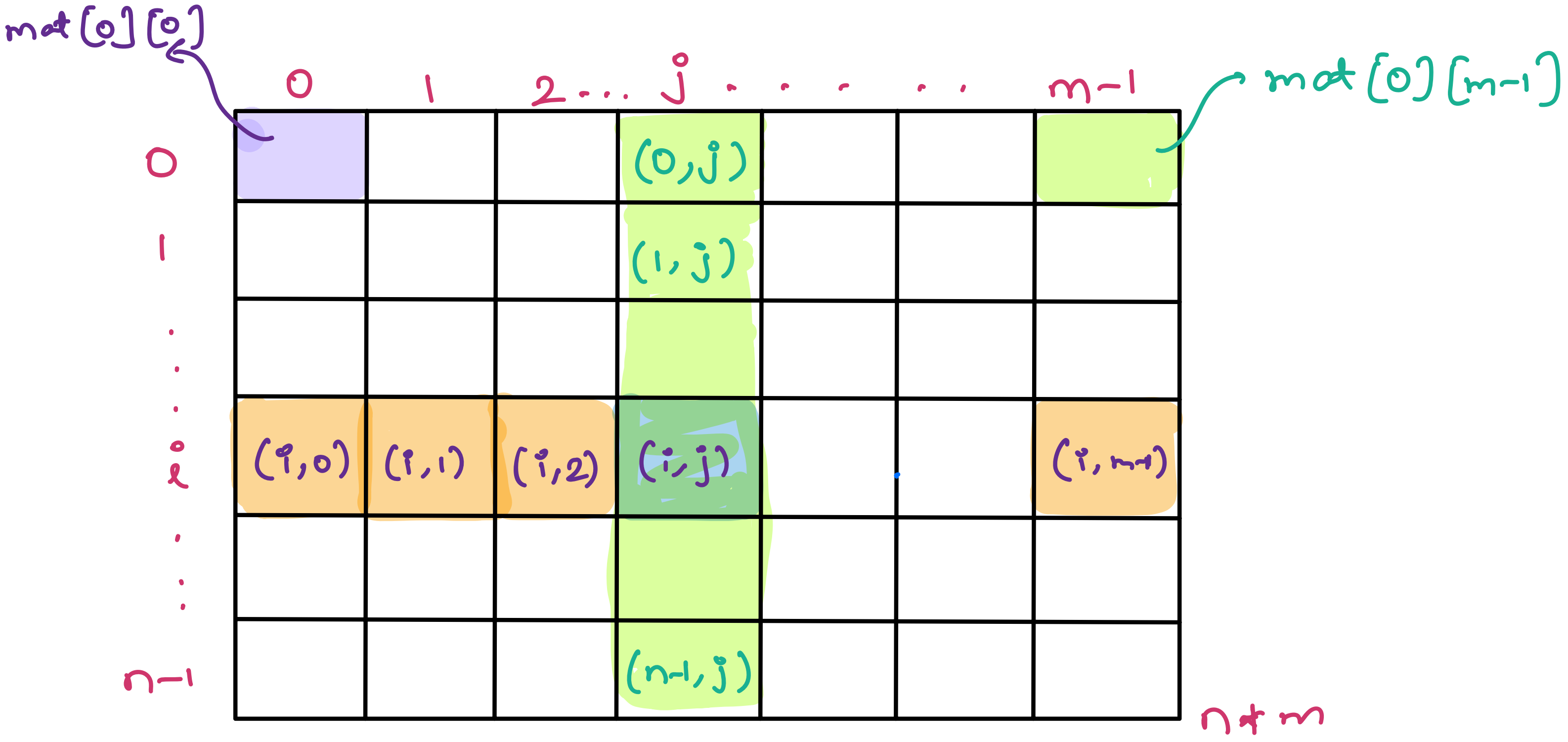
cols / vertical

rows / horizontal



Generalise

int [n] [m] mat



Obs → If we move on i^{th} row

↳ col will change [0 m-1]

Obs → If we move on j^{th} col

↳ row will change [0 n-1]

—x—x—x—x—x—

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

	0	1	2	3	sum
0	4	3	1	7	15
1	6	2	3	4	15
2	5	3	2	7	17

```
int n = mat.length → rows
int m = mat[0].length → coloumns
```

Idea → Iterate on each row & sum all the col values

```
void  rowsum ( int [][] mat )

        int  n= mat.length          // rows
        int  m = mat[0].length      // cols

        for ( r=0 ; r<n ; r++)
                sum=0             // sum of rth row
            for (c=0; c<m; c++) {
                sum = sum+ ar[r][c] ;
            }
            println ( sum );
        }
}
```

TC = O(n*m)

SC = O(1)

02. Given mat [n][m], print col wise sum.

	0	1	2	3
0	4	3	1	7
1	6	2	3	4
2	5	3	2	7
	15	8	6	18

{ TODO }

Q3. Given square matrix mat[N][N],

print diagonal → Left to right
→ Right to left

	0	1	2	3
0	(0,0)			
1		(1,1)		
2			(2,2)	
3				(3,3)

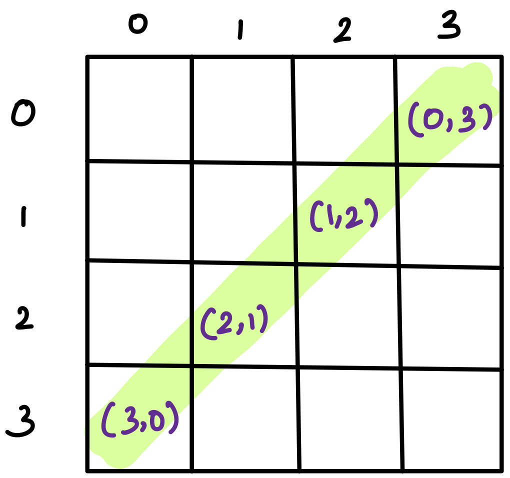
Observation → i & j are exactly same & both are moving by 1

i	j
0	0
1	1
2	2
3	3
4	4 *

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

TC= O(n)
SC= O(1)

Diagonal → Right to left



	i	j	
	0	3	
+1	1	2	-1
+1	2	1	-1
+1	3	0	-1
	4	-1	*

Obs → i=0 , j=n-1

& i will increase by 1 &

j will decrease by 1

```
i = 0 , j = n-1

while ( i<n && j≥0 ){
    println ( mat[i][j]);
    i = i+1
    j = j-1
}
```

TC : O(n)

SC : O(1)

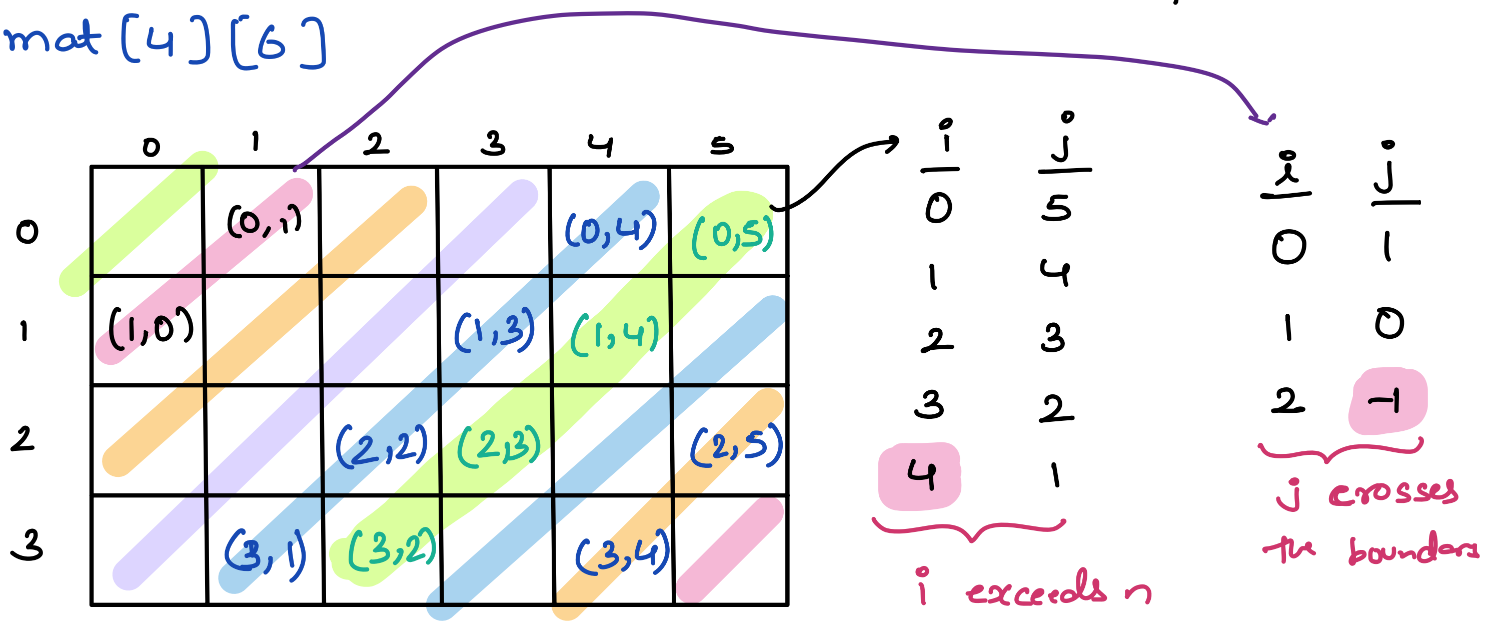
TODO → For loop instead of while loop

→ rectangular matrix

03. Given a matrix mat[N][M], print all the diagonals from Right to left

Note :- Diagonals should start from 0th row & can also start from last col

mat[4][6]



	0	1	2	3	4	5
0		(0,1)			(0,4)	(0,5)
1	(1,0)			(1,3)	(1,4)	
2			(2,2)	(2,3)		(2,5)
3		(3,1)	(3,2)		(3,4)	

i	j
0	5
1	4
2	3
3	2
4	1

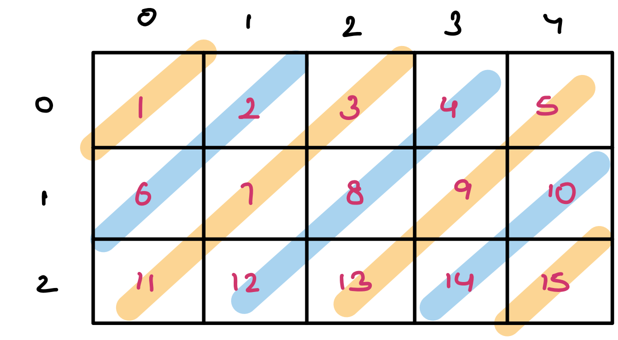
i exceeds n

i	j
0	1
1	0
2	-1

j crosses the boundary

Obs → $i < n$ && $j \geq 0$

arr[3][5]



	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

Output

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

Idea → First print all the diagonals of 0th row

→ Print all the diagonals of last col

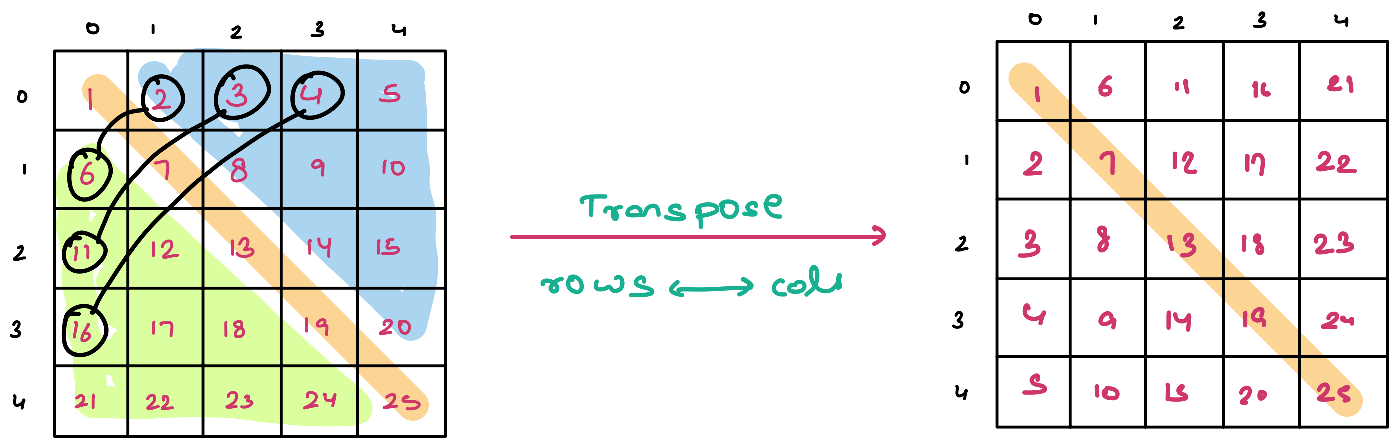
```
void   printrectagle diagonal (int [][] mat)
        int  n = mat.length;
        int  m = mat[0].length;
      // print diagonals of 0' row
      for ( j=0 ; j<m ; j++)
          → int r=0, c=j   ←
            while (r<n && c≥0){
                print (mat[r][c]);
                r++;  c--;
            }
            println();
      }

   // print diagonals of last col
      for ( i=1 ; i<n; i++ )          careful on this
            r= i  ,  c=m-1
            while (r<n && c≥0){
                print (mat[r][c]);
                r++;  c--;
            }
            println()'
      }
}
```

TC = O(n*m)
SC = O(1)

04. Given a mat[N][N], calculate transpose of matrix

Interchanging rows with cols

Expected 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

	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] $\xrightarrow{\text{Transpose}}$ mat[1][0]

mat[0][2] $\xrightarrow{\text{Transpose}}$ mat[2][0]

mat[0][3] $\xrightarrow{\text{Transpose}}$ mat[3][0]

mat[2][3] $\xrightarrow{\text{Transpose}}$ mat[3][2]

Generalisation - mat[i][j] $\xleftrightarrow{\text{swap}}$ mat[j][i]

i = = j ⟶ No need to swap

```
void transpose (int [][] mat)
        int n= mat.length;

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

$TC = O(n^2)$

$SC = O(1)$

Note :- Swap either in lower triangle or in the upper triangle

Q5. Given a mat[n][n], Rotate the matrix by 90° from top right   sc: O(1)

Transpose

0th row — 0th col

1st row — 1st col

2nd row — 2nd col

⋮

Rotate

0th row → 4th col

1st row → 3rd col

2nd row → 2nd col

⋮

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

Transpose

	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

Reverse each & every row

r = 3

i = 0   j = 4

```
void  rotateby 90° ( int [][] mat)

    transpose (mat)

    for ( r=0 ; r<n ; r++)

        int  i=0    j= n-1

        while ( i<j )

            int  temp = mat [r] [i]
            mat [r][i] = mat [r][j]        swap
            mat [r] [j]  =  temp

            i = i+1
            j = j-1
        }
    }
}
```

mat [3][0] with

mat [3] [4]

TC: $O(n^2)$

SC: $O(1)$