Today's Quote ⇒

Today's content

→ Basics
→ Problems] 2-D array or matrices.

How to declare?

rows : horizontal lines

int mat [4] [5]

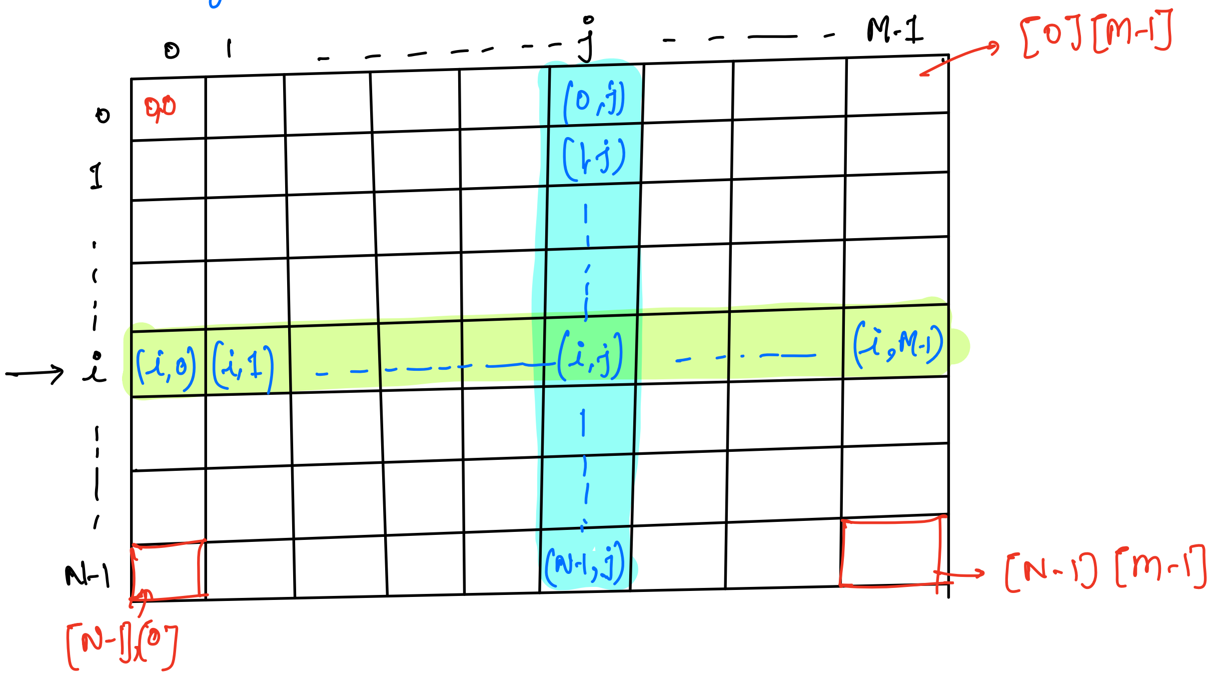
colums. : vertical lines

	0	1	2	3	4
0					
1					
2					
3					

int mat [N] [M]

N → No. of rows

M → M no. of colums



observation 1 : If we move in i^{th} row
col changes $[0 \rightarrow M-1]$

observation 2 : If we move in j^{th}-col
row changes $[0 \rightarrow N-1]$

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

Eg→ mat[3][4]

	0	1	2	3	o/p.
0	4	3	1	7	: 15
1	6	2	3	4	: 15
2	5	3	2	7	: 17

total no. of elements = N*M

T.C → O(N*M)
S.C → O(1)

```
void printSum( arr, N, M){

    for( i → 0 to N-1) {
        sum = 0
        for( j → 0 to M-1) {
            sum += arr[i][j]
        }
        // print sum.
    }
}
```

Given mat[N][M], print col wise sum {To-do}

↓

{Code in today's doubt session}

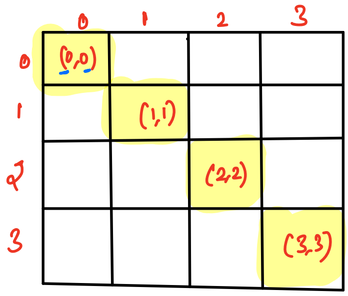
Eg→ mat[3][4]

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

o/p. → 15 8 6 18.

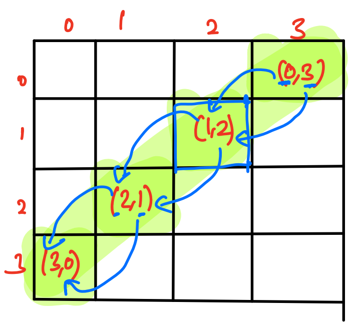
Q) Given square mat[N][N]. print diagonals → left to right / right to left

Eg: mat[4][4]




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

$\boxed{T.C \to O(N) \;,\; S.C \to O(1)}$



0, 3
+1 ⌈ ⌉ -1
1 2
+1 ⌈ ⌉ -1
2 1
+1 ⌈ ⌉ -1
3 0
+1 ⌈ ⌉ -1
4 -1

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

$\boxed{T.C \to O(N) \;,\; S.C \to O(1)}$

→ All squares are rectangles. ✓
→ All rectangles are square. ✗

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

diagonals starting from 0th row or M-1th column.

mat[4][6]

	0	1	2	3	4	5
0			[0,2]		[0,4]	[0,5]
1		[1,1]		[1,3]	[1,4]	[1,5]
2	[2,0]		[2,2]	[2,3]	[2,4]	
3		[3,1]	[3,2]	[3,3]		

i, j: 0,4 → 1,3 → 2,2 → 3,1 → 4,0 stop.

i, j: 0,2 → 1,1 → [2,0] → [3,-1] stop

i, j: 1,5 → 2,4 → 3,3 → [4,2] stop.

mat[3][5]

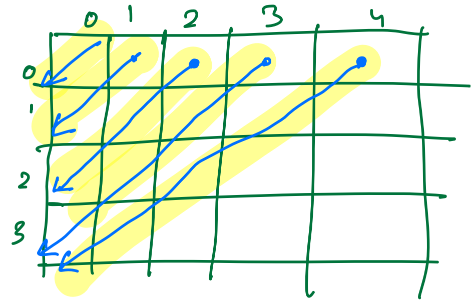
	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15

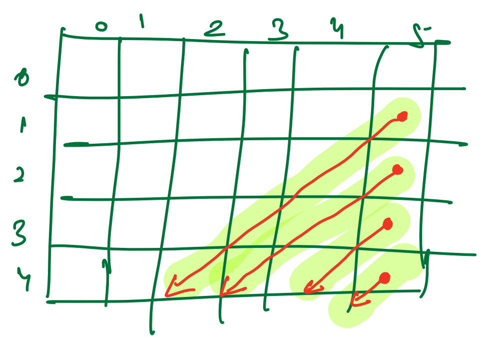
Output:

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

pseudo code

```
void printDiagonals( mat[][] , N , M){
    //print all Diagonals starting from 0th row.
    for( j -> 0 to M-1){
        r=0 , c=j
        while( r<N && c>=0){
                print( mat[r][c])
                r+=1
                c-=1
        }
    }
    //print all Diagonals starting from M-1th col
    for( i -> 1 to N-1){
        r = i , c = M-1
        while( r<N && c>=0){
                print( mat[r][c])
                r+=1
                c-=1
        }
    }
}
```





$\boxed{T.C \rightarrow O(N*M) \;,\; S.C \rightarrow O(1)}$

$\Downarrow$

{we are touching all elements once}

Q. Given matrix [N][N]. Calculate transpose of mat[] with S.C → O(1).

Note → get transpose in the given matrix itself.

mat[5][5]:

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \\ 2 & 11 & 12 & 13 & 14 & 15 \\ 3 & 16 & 17 & 18 & 19 & 20 \\ 4 & 21 & 22 & 23 & 24 & 25 \end{array}$$

row 0 → col 0
row 1 → col 1
row 2 → col 2
row 3 → col 3
row 4 → col 4

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 6 & 11 & 16 & 21 \\ 1 & 2 & 7 & 12 & 17 & 22 \\ 2 & 3 & 8 & 13 & 18 & 23 \\ 3 & 4 & 9 & 14 & 19 & 24 \\ 4 & 5 & 10 & 15 & 20 & 25 \end{array}$$

idea: Swap upper-half elements with lower half.

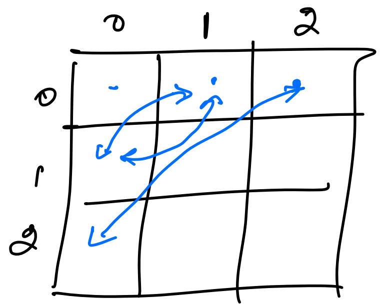
```
void takeTranspose ( arr, N, M) {

    for (i → 0 to N-1) {
        for ( j → i+1 to N-1 ) {
            //swap arr[i][j] with arr[j][i]
            temp = arr[i][j]
            arr[i][j] = arr[j][i]
            arr[j][i] = temp
        }
    }
}
```

$\boxed{T.C \rightarrow O(N^2), \quad S.C \rightarrow O(1)}$

```
void takeTranspose ( arr, N, M) {
    for (i → 0 to N-1) {
        for ( j → 0 to N-1 ) {
            //swap arr[i][j] with arr[j][i]
            temp = arr[i][j]
            arr[i][j] = arr[j][i]
            arr[j][i] = temp
        }
    }
}
```

3×3

Q: what will be o/p.



arr[0][0] ⟷ arr[0][0]

arr[0][1] ⟷ arr[1][0]

arr[1][0] ⟵ arr[0][1]

3×3

3×5

⇒ Matrix is going to remain as it is.

// If rectangle → We need to have extra space.

$$mat[2][5] \xrightarrow{\text{transpose}}$$

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \end{array} \longrightarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1 & 6 \\ 1 & 2 & 7 \\ 2 & 3 & 8 \\ 3 & 4 & 9 \\ 4 & 5 & 10 \end{array}$$

$\underline{2 \times 5}$ $\qquad$ $\underline{5 \times 2.}$

Q) Given a square matrix. Rotate 90' clockwise.

$\boxed{SC \to O(1)}$

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \\ 2 & 11 & 12 & 13 & 14 & 15 \\ 3 & 16 & 17 & 18 & 19 & 20 \\ 4 & 21 & 22 & 23 & 24 & 25 \end{array}$$

0^{th} row ⟷ 4^{th} col

1^{st} row ⟵ 3^{rd} col

2^{nd} row ⟵ 2^{nd} col

3^{rd} row ⟵ 1^{st} col

4^{th} row ⟵ 0^{th} col

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 21 & 16 & 11 & 6 & 1 \\ 1 & 22 & 17 & 12 & 7 & 2 \\ 2 & 23 & 18 & 13 & 8 & 3 \\ 3 & 24 & 19 & 14 & 9 & 4 \\ 4 & 25 & 20 & 15 & 10 & 5 \end{array}$$

transpose

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 6 & 11 & 16 & 21 \\ 1 & 2 & 7 & 12 & 17 & 22 \\ 2 & 3 & 8 & 13 & 18 & 23 \\ 3 & 4 & 9 & 14 & 19 & 24 \\ 4 & 5 & 10 & 15 & 20 & 25 \end{array}$$

reverse 0^{th} row

reverse 1^{st} row

reverse 2^{nd} row

reverse 3^{rd} row

reverse 4^{th} row

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 21 & 16 & 11 & 6 & 1 \\ 1 & 22 & 17 & 12 & 7 & 2 \\ 2 & 23 & 18 & 13 & 8 & 3 \\ 3 & 24 & 19 & 14 & 9 & 4 \\ 4 & 25 & 20 & 15 & 10 & 5 \end{array}$$

1	6	11	16	21

// step-1 take transpose of the given matrix.

// step-2. Reverse every row.

```
for ( i → 0 to N-1) {
    left = 0 , right = N-1
    while ( left < right) {
        // swap left element with right element
        temp = arr[i][left]
        arr[i][left] = arr[i][right]
        arr[i][right] = temp.
        left += 1 , right -= 1
    }
}
```

$\boxed{T.C \rightarrow O(N^2) \quad S.C \rightarrow O(1)}$

Rotate Rectangular matrix. {We need to have extra space}

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 6 & 7 & 8 & 9 & 10 \end{array} \Rightarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 6 & 1 \\ 1 & 7 & 2 \\ 2 & 8 & 3 \\ 3 & 9 & 4 \\ 4 & 10 & 5 \end{array}$$

2x5

{ToDo}.

Doubts :

$\boxed{9 \text{ to } 11:30}$ ⇒ least solved problems till sub-arrays.

Timings → 9 pm to 11:30 pm.

→ { Revision on weekly basis. [concepts]
↳ Questions that were not solved in 1st attempt. }

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & \boxed{13} & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{matrix}$$

while (———) {

top-boundary

right-boundary

bottom-boundary

left-boundary

}

0 90 180 270.

n.

intermediate. → basic idea nearly all D.S.

arrays n strings.