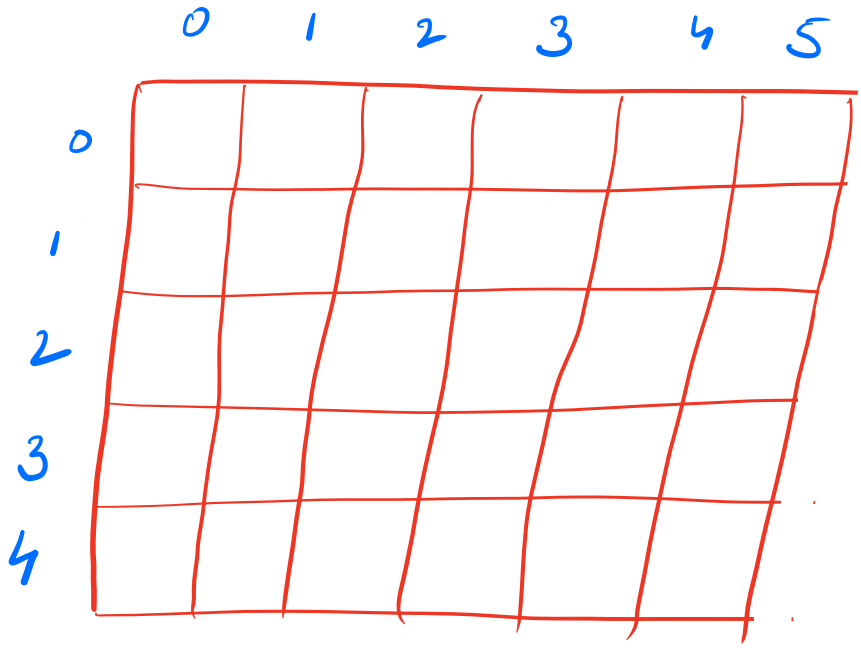
int arr[5][6]

rows columns

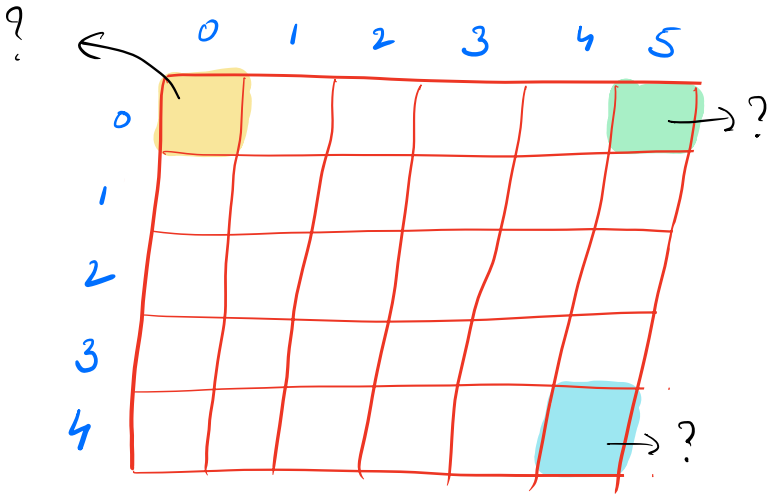
5x6 matrix



How to access?

arr [i] (1D case)

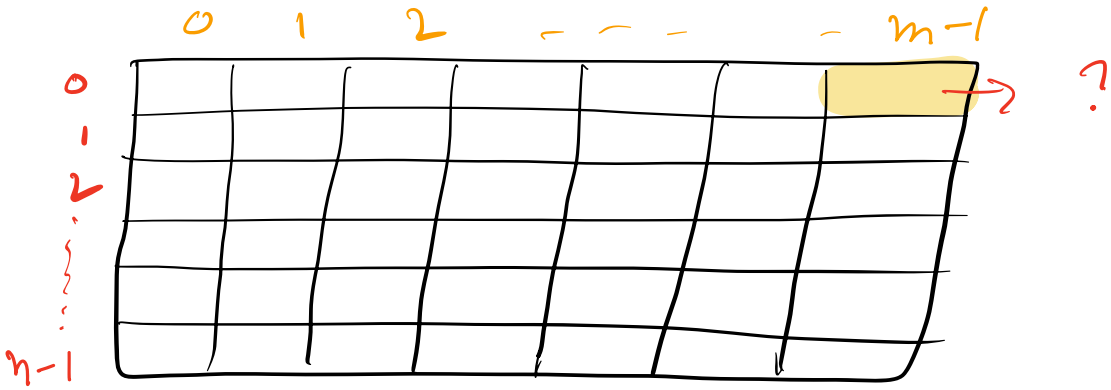
arr [row_no][col_no] (2D)



arr [N]

arr [N][M]

N x M matrix



Q1 Given arr[N][M], print row-wise sum

Eg

	0	1	2	3	
0	1	2	3	4	→ 10
1	5	6	7	8	→ 26
2	9	1	1	2	→ 13

row → i

col → j

```
for( i=0; i<N ; i++ ) {
    // ith row
    int sum=0
    for(j=0; j<M ; j++) {
        sum += arr[i][j]
    }
    print(sum)
}
```

TC: O(NM) SC: O(1)

Q2 Given arr[N][M], find maximum column sum

Eg

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

ans = 15

```
for ( j=0 ; j<m ; j++ ) {
        // jth col
        int sum = 0
        for ( i=0 ; i<n ; i++ ) {
            sum += arr[i][j]
        }
        max_col_sum = max ( sum, max_col_sum )
}
return max_col_sum
```

TC: O(NM) SC: O(1)

Why a[j][i] is not used generally?

i → row_no

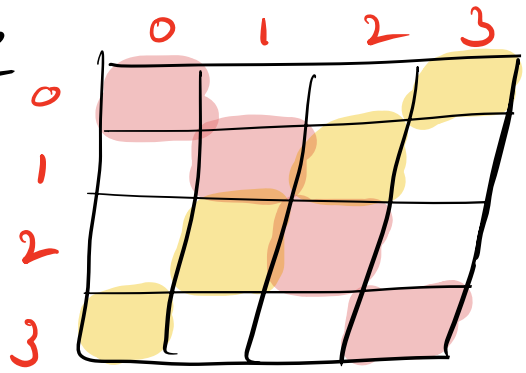
j → col_no

ar[i][j]

Q3 Given arr[N][N], print diagnols

↳ square matrix

Eg



→ L-R diagnol

→ R-L diagnol

L-R

a[0][0]
a[1][1]
a[2][2]
a[3][3]

Obs: Row no = Col no

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

TC: O(N)

OR

```
int i=0
while ( i<N ) {
    print ( arr[i][i] )
    i++
}
```

R-L

arr[0][3]

arr[1][2]

arr[2][1]

arr[3][0]

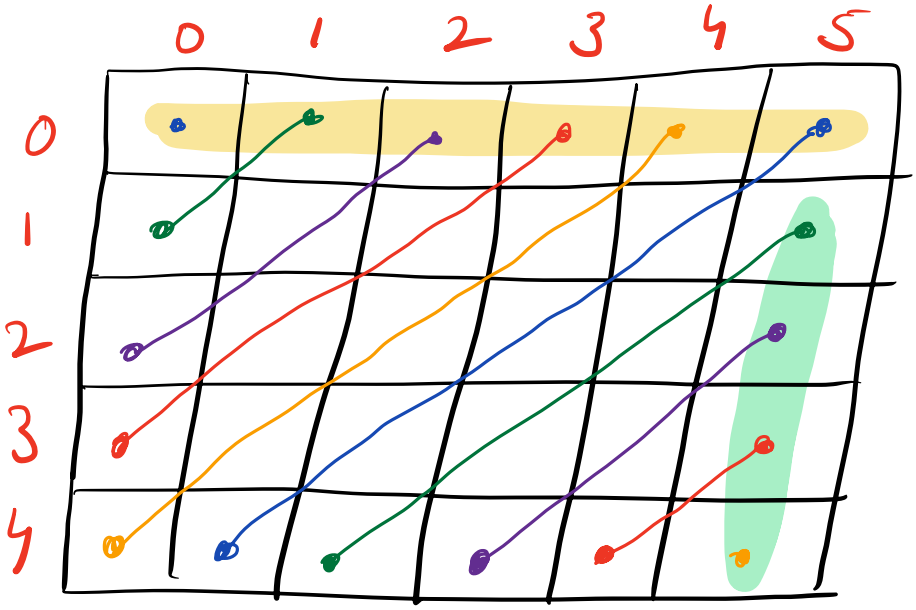
	0	3	
+1	1	2	-1
	2	1	-1
	3	0	-1
	4	-1	

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

TC: $O(n)$

SC: $O(1)$

Q4 Given arr[N][M], print all R-L diagnols



5x6

⇒ 10 diag

1) Diagnols starting row 0 → m

2) Diagnols starting col m-1 → n-1

Total = m+n-1

2, 5

3 4

4 3

1, 5

2, 5

3, 5

4, 5

```
for (j=0 ; j<M ; j++) {
    int x = 0                    //row
    int y = j                    // col
    while (x<N && y>=0) {
        print (a[x][y])
        x++
        y--
    }
}
```

$j=0$

$x=0 \quad y=0$

arr [0] [0]

$x=1 \quad y=-1$

break

$j=1$

$x=0 \quad y=1$

arr [0] [1]

$x=1 \quad y=0$

arr [1] [0]

$x=2 \quad y=-1$

break

$j = 2$

$x = 0 \quad y = 2$

$a[0][2]$

$x = 1 \quad y = 1$

$a[1][1]$

$x = 2 \quad y = 0$

$a[2][0]$

$x = 3 \quad y = -1$

break

```
for (i=1 ; i<N; i++) {
    int x= i        y= m-1
    while (x<N && y>=0 ) {
        print (a[x][y])
        x++
        y--
    }
}
```

i=1

x= 1 y=5

1,5

2,4

3,3

4,2

5,1 break

$i = 2$

$x = 2$ $y = 5$

2	5	
3	4	
4	3	
5	2	} break

TC: $O(N \times M)$ SC:

Q5 Given arr[N][N], find the transpose of this matrix

inplace → SC: O(1)

arr[N][N] needs to be updated

Transpose ⇒ i^{th} row → i^{th} col

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

→

	0	1	2	3
0	1	5	9	13
1	2	6	10	14
2	3	7	11	15
3	4	8	12	16

Pattern ⇒ 0,1 1,0 1,2 2,1

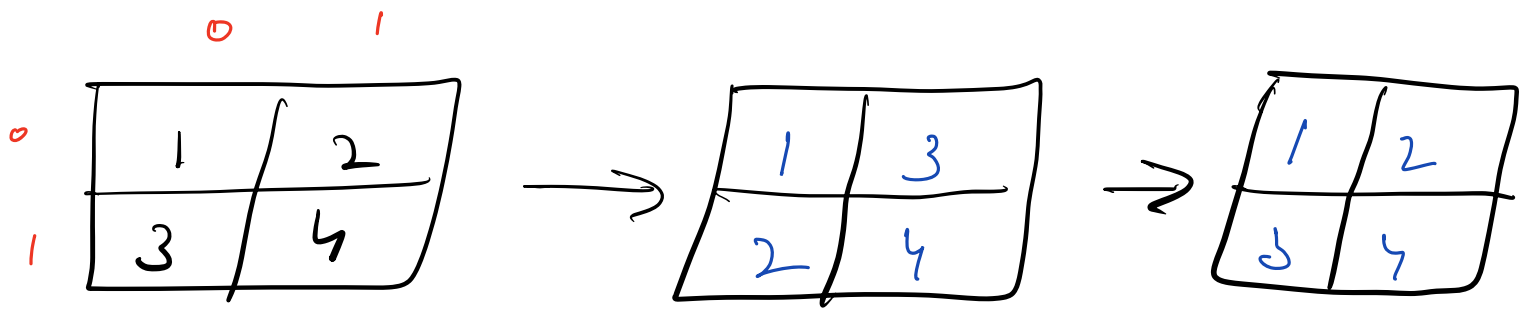
0,2 2,0

arr[i][j] ⟺ arr[j][i]

Solution ⇒

Swap arr[i][j], arr[j][i]

```
for( i=0 ; i< N ; i++ ){
    for( j=0 ; j<N ; j++ ){
        swap(arr[i][j], arr[j][i])
    }
}
```



How to avoid? only run on upper half.

```
for( i=0 ; i< N ; i++ ){
    for( j=i ; j<N ; j++ ){
        swap(arr[i][j], arr[j][i])
    }
}
```

1	2	3
4	5	6
7	8	9

$\Longrightarrow$

TC:

SC:

06 Rotate by 90° clockwise

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

→

	0	1	2	3
0	13	9	5	1
1	14	10	6	2
2	15	11	7	3
3	16	12	8	4

Hint: first take transpose

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

→

	0	1	2	3
0	1	5	9	13
1	2	6	10	14
2	3	7	11	15
3	4	8	12	16

What is the pattern now?

reverse each row

Ans: reverse every row.

1) Transpose

2) Reverse each row

```
for ( i=0 ; i<n ; i++ ){
        start = 0          end = m-1
            while (start < end) {
                swap (a[i][start], a[i][end])
                start ++
                end --
            }
}
```

10 Nov → 9 Nov
Fri Thurs

{done}

$[prev \quad , \quad i]$

$i - prev + 1$

(a, b)

$b - a + 1$