2D Array

MATRIX →

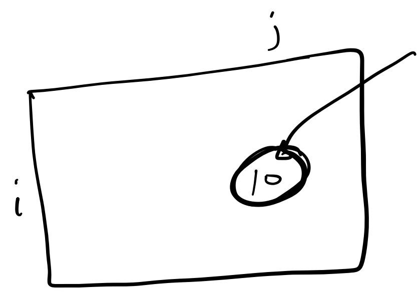
$$
M \quad
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & 2 & 3 & 5 & 1 \\
1 & 2 & 1 & 2 & 2 \\
2 & 4 & 5 & 6 & 3
\end{array}
\quad \underline{3 \times 4}
$$

(i, j) → row, col

$(1, 2)$

`int M[3][4];`

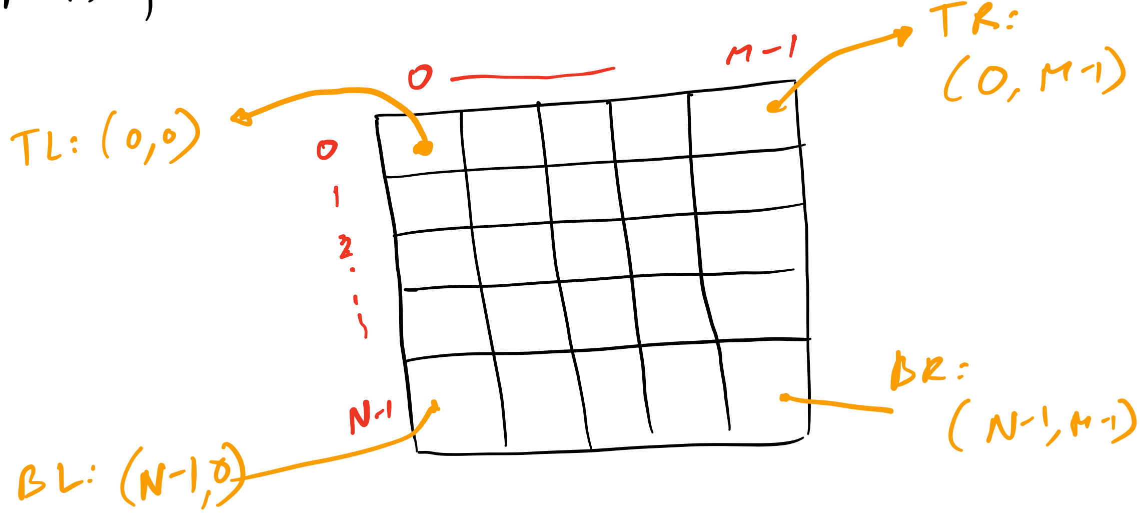
Access time?



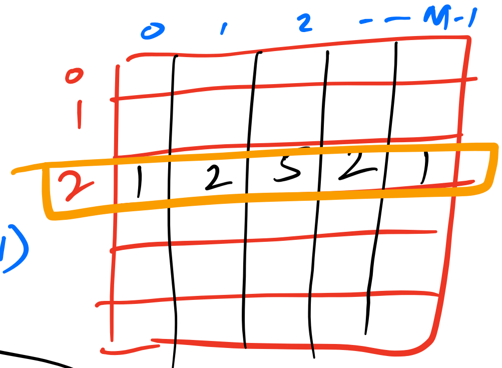
$M[i][j] \rightarrow O(1)$

→ b/c it is stored linearly in memory!

$M[i][j] = 10 \rightarrow O(1)$



Q. Given a 2D array!
Find the sum of elements of the 2nd row!



2nd row →

(2,0), (2,1), (2,2)...(2,M-1)

```
sum = 0
for(j=0; j<M; j++) {
    sum += Mat[2][j];
}
// sum → ANS
```

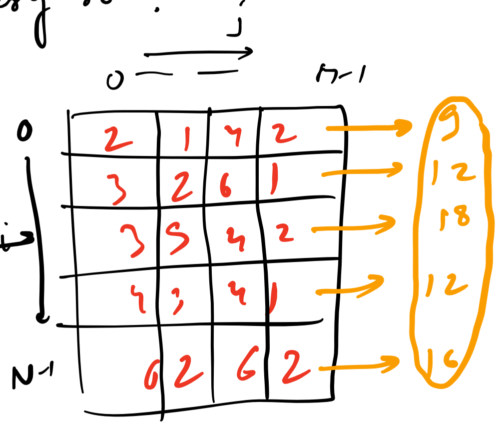
$\boxed{TC: O(M)}$

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

Q. Given a 2D array.
Find the sum of every row!

i: 0

```
for( i=0; i<N; i++) {
    sum = 0;
    for( j=0; j<M; j++) {
        sum += Mat[i][j];
    }
    print(sum);
}
```



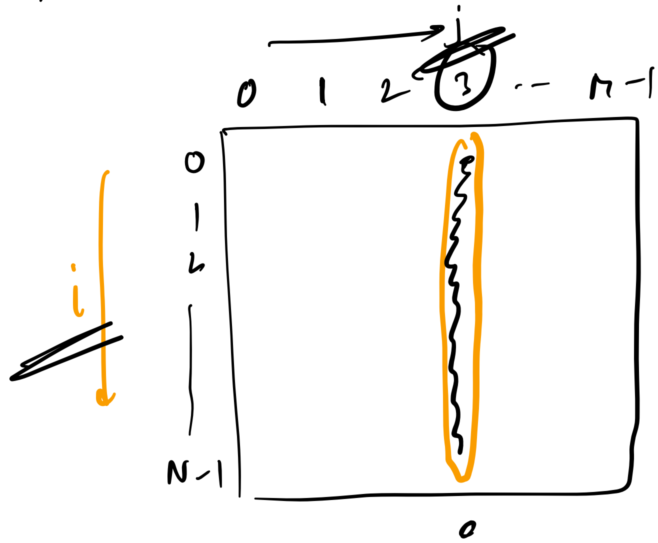
$\boxed{TC: O(NM)}$

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

Q. Given a Matrix A[N][M].
Find the MAXIMUM column Sum!

1	2	4
2	1	2
3	1	5
4	1	1

$\underbrace{10 \quad 5 \quad 12}_{\text{MAX : 12}}$



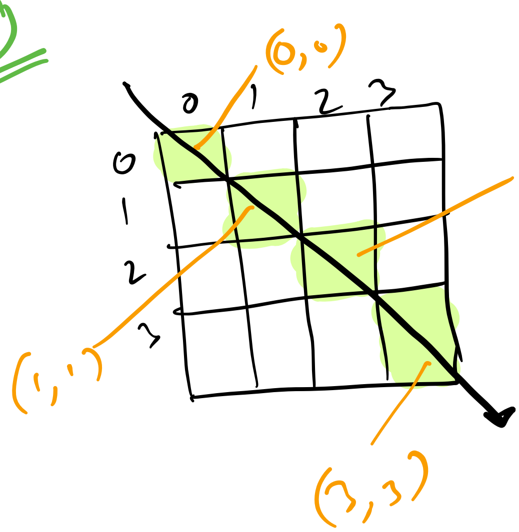

```
maxSum = -∞
for(j=0; j<M; j++) {
    sum = 0;
    for(i=0; i<N; i++) {
        sum += A[i][j];
    }
    maxSum = max(maxSum, sum);
}
ret maxSum
```

$\boxed{TC : O(NM)}$

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

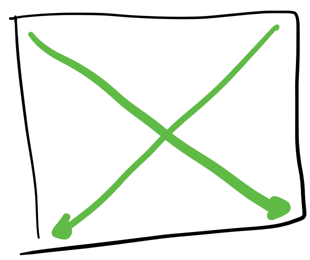
Q Given a 2D arr of size **N × N**
print the diagonal values

1



1

2



(i,i)

$i : 0 \longrightarrow N-1$

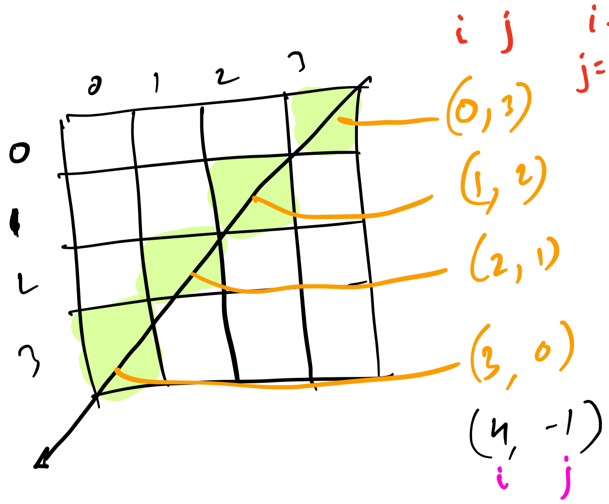
(i,i)

```
f( i=0; i<N; i++) {
    print( A[i][i])
}
```

$\boxed{TC : O(N)}$

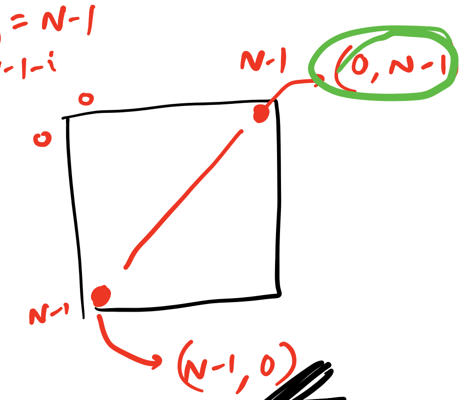
$\boxed{SC : O(1)}$

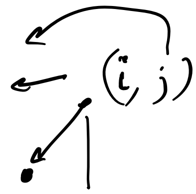
2



$i + j = N-1$

$j = N-1-i$





O(N)

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

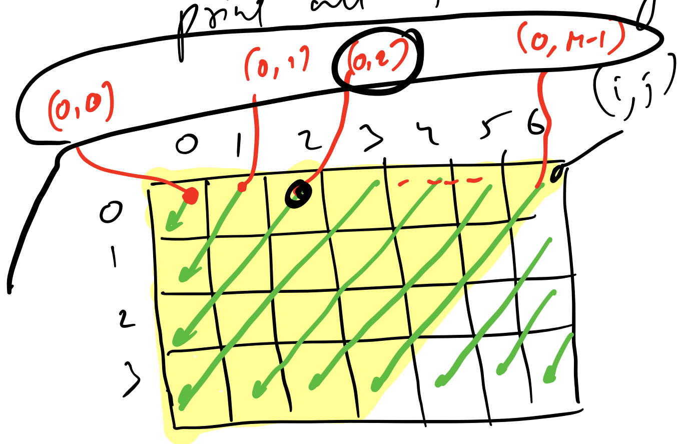
NOT NEEDED !

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

O(N)

```
for (i = 0; i < N; i++) {
    j = N-1-i;
    print(A[i][j])
}
```

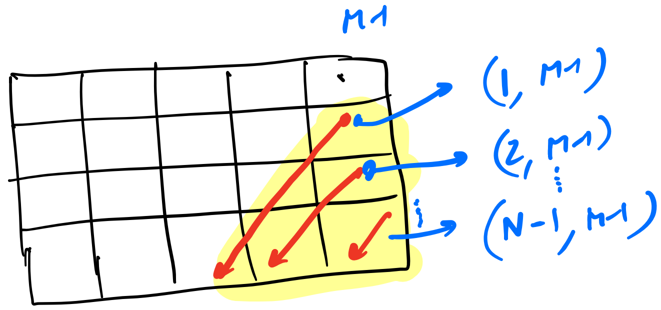
Q Given a 2D array $A[N][M]$.
print all the diagonals ($R \to L$, $T \to B$)



(0,0)
(0,1), (1,0)
(0,2), (1,1) (2,0)
(0,3), ———— (3,6)
|
|

print diagonals in yellow region

```
f( j=0; j<M; j++) {
    I=0, J=j;
    while( I<N && J>=0) {
        print( A[I][J]);
        I++;
        J--;
    }
}
```




```
f( i= 1; i< N; i++) {
    I = i, J= M-1;
    while( I < N && J>=0) {
        print( A[I][J]);
        I++;
        J--;
    }
}
```

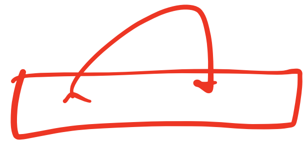
$\boxed{TC: O(NM)}$ $\boxed{SC: O(1)}$

Q Given a SQUARE MATRIX → **N×N**

Find the transpose of it!

$N^2 - N$

$$A \quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array} \qquad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 4 & 7 \\ 1 & 2 & 5 & 8 \\ 2 & 3 & 6 & 9 \end{array}$$



I

$\boxed{TC = O(N^2)}$

$\boxed{SC = O(N^2)}$

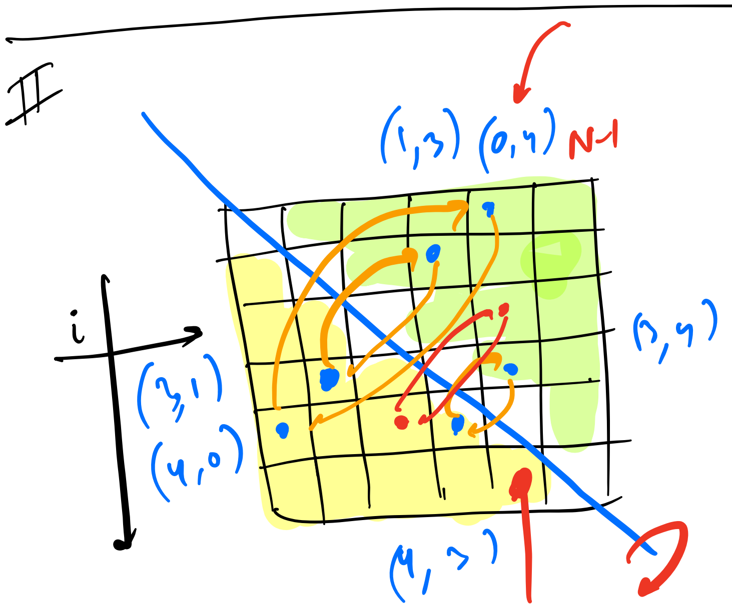
$f(i: 0 \longrightarrow N-1)$

$\quad f(j = 0 \longrightarrow M-1)$

$\quad\quad T[j][i] = A[i][j]$

Copy $T \longrightarrow A$

II



j →

(1,0)

(2,0) (2,1)

i=3 → (3,0) (3,1) (3,2) -

(N-1,0) (N-1,1) - - - (N-1,N-2)

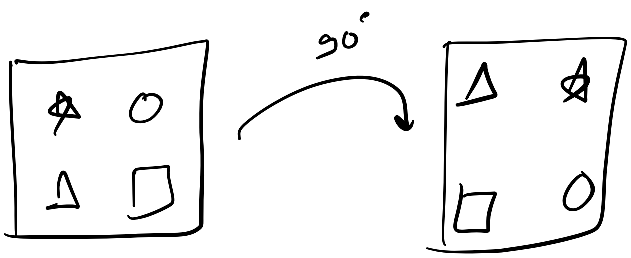
```
for(i=1; i<N; i++){
    for(j=0; j<i; j++){
        swap(A[i][j], A[j][i]);
    }
}
```

$\boxed{TC: O(N^2)}$

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

INPLACE

Q. Given a square matrix.
Rotate it by 90° clockwise!



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

90°

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

<u>Transpose</u> $O(N^2)$

reverse all rows!

$O(N^2)$

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

$\boxed{TC: O(N^2)}$

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

Idea:

$TC: O(N^2)$

$SC: O(1)$