Rectangular grid ⇒ Element / cell

Matrix ⇒ Rows × Columns ⇒ Size

Declare ⇒ int mat[N][N]

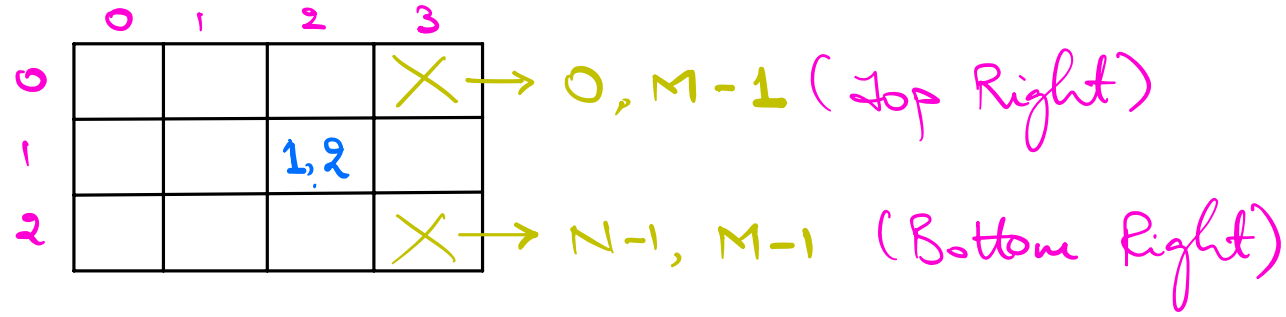
Rows Column

Access element in ith row & jth column

mat[i][j]

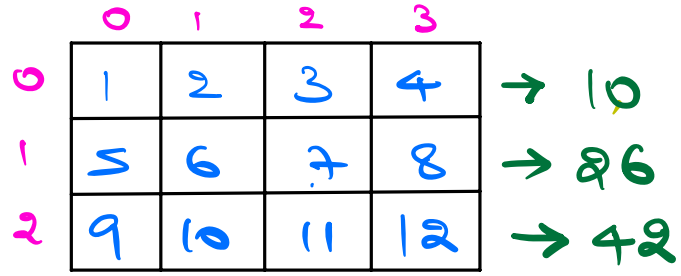
3 × 4 matrix

N M



Given a 2D matrix of size N×M. Print the row-wise sum of the matrix

3 × 4 matrix



Sol^{n} Iterate all the rows one by one.

Code

```
void sumRow ( int mat[N][M] ) {

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

T.C. = $O(N \times M)$

S.C. = $O(1)$

Q Given a 2D matrix of size N×M. Print the column-wise sum of the matrix

3×4 matrix

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
	15	18	21	24

```
void sumRow ( int mat[N][M] ){
    for (j = 0; j < M; j++) {          M
        sum = 0;
        for (i = 0; i < N; i++) {      N
            sum = sum + mat[i][j];
        }
        print (sum);
    }
}
```

T.C. = $O(N \times M)$

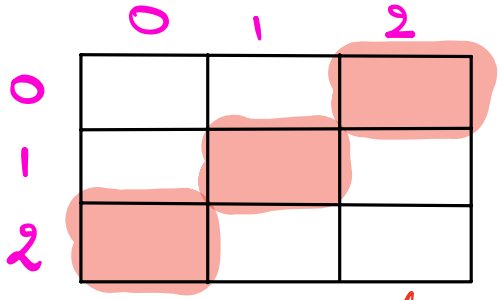
S.C. = $O(1)$

Given a square matrix of size N×N

Print all the diagonal elements from left to right.

Principal diagonal

(i == j)



Anti-diagonal

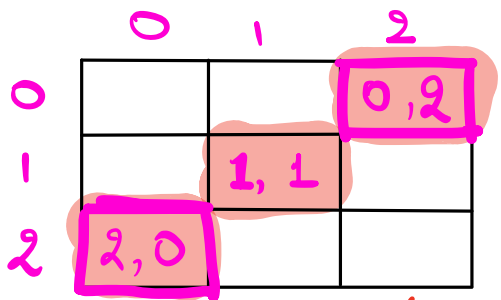
1) Principal diagonal

```
void printdiagonal(mat){
    int i = 0;
    int j = 0;

    while(i<N && j<N){
        print(mat[i][j]);
        i++;
        j++;
    }
}
```

```
void printdiagonal(mat){
    int i = 0;
    while(i<N){
        print mat[i][i];
        i++;
    }
}
```

T.C. = O(N)

2) Anti Diagonal

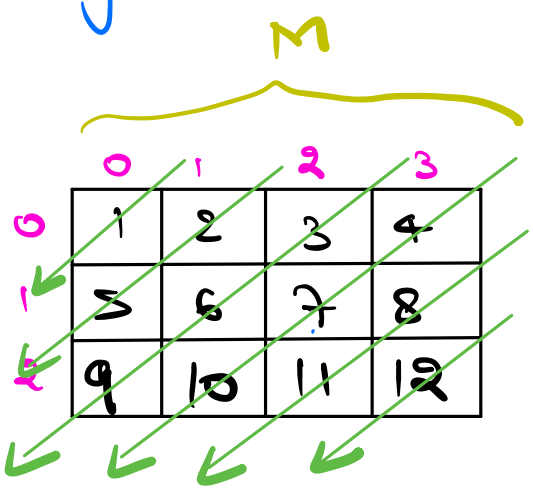


i--, j++

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

T.C. = O(N)

Rectangular matrix ⇒ N x M



1

2, 5

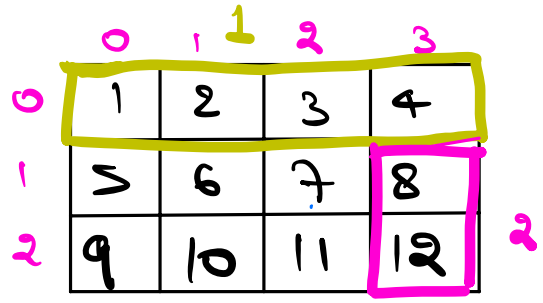
3, 6, 9

4, 7, 10

8, 11

12

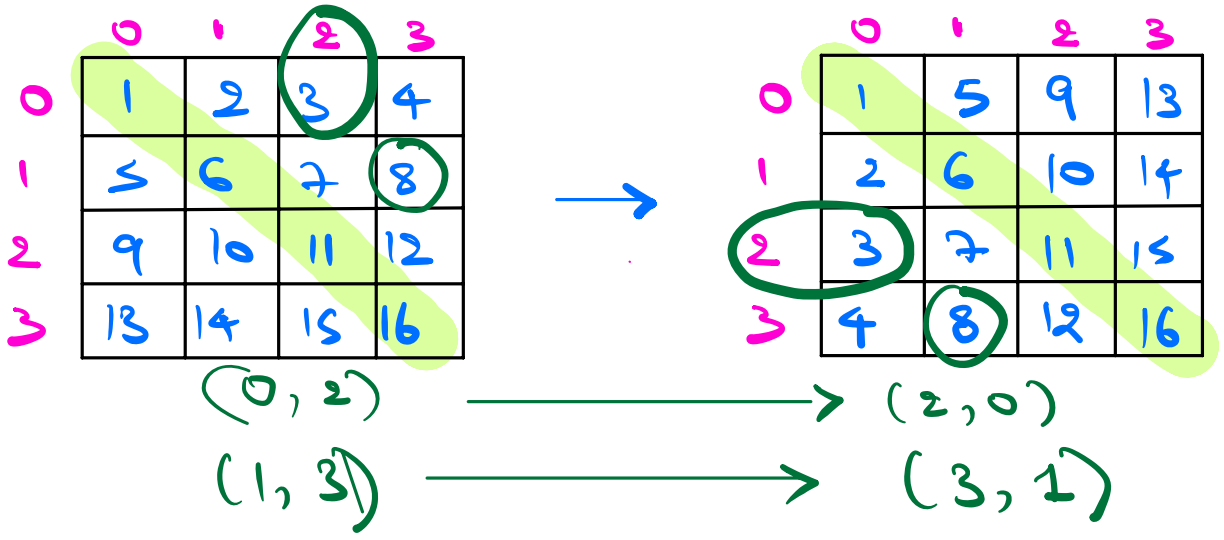
Diagonals = N+M-1




```
for (i=0; i<M; i++) {
    r = 0
    c = i
    while (r<N && c>=0) {
        print (mat[r][c]);

        r++;
        c--;
    }
    print ("\n")
}

for (i=1; i<N; i++) {
    c = M-1
    r = i;
    while (r<N && c>=0) {
        print (mat[r][c]);

        r++;
        c--;
    }
```

```
        printf("\n");
    }
```

Given a square matrix of N x N.
Convert the matrix to its transpose.




```
for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
        transpose[j][i] = mat[i][j];
    }
}
```

Code

```
for (i=0; i<N; i++) {
    for (j=i+1; j<N; j++) {
```

```
        temp = mat[i][j];
        mat[i][j] = mat[j][i];
        mat[j][i] = temp;
    }
}
```

T.C. = $O(N^2)$

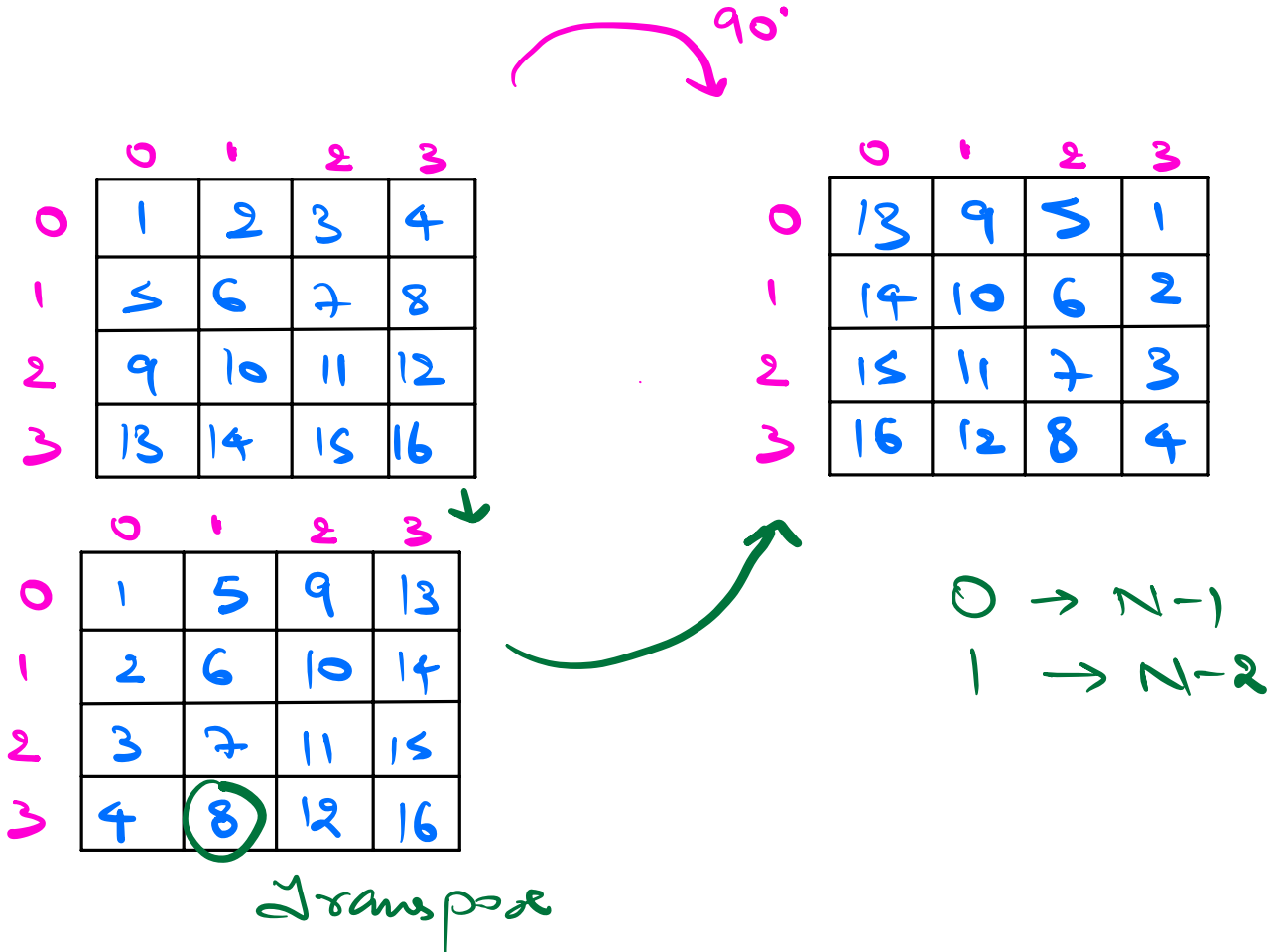
S.C. = $O(1)$

Rectangular matrix

⇒

3 × 4 ⇒ 4 × 3

Given a square matrix of N×N
Rotate the matrix 90° clockwise



0 → 0
1 → 1

1) Take transpose of matrix N^2

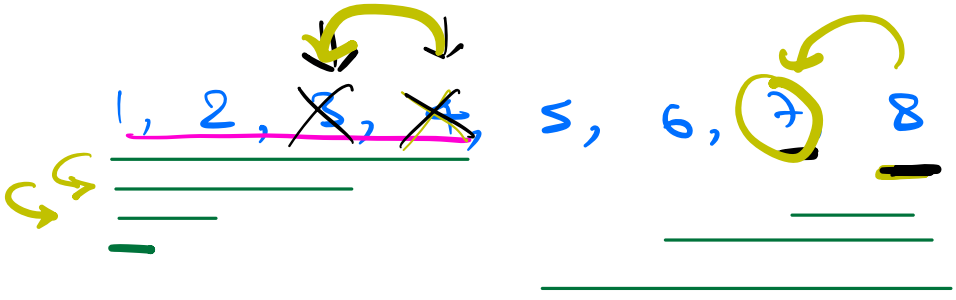
2) Reverse each row. N^2

T.C. = $O(N^2)$
S.C. = $O(1)$

Interview Bit
Leetcode
Geeksforgeeks

---

100 pep 100 leet



B = 4

Sum

```
i = B-1
j = N-1
for (k=0; k<B; k++)
```

Sum = Sum - A[i] + A[j];

i++

j--

}