2D Matrix

→ 2D array

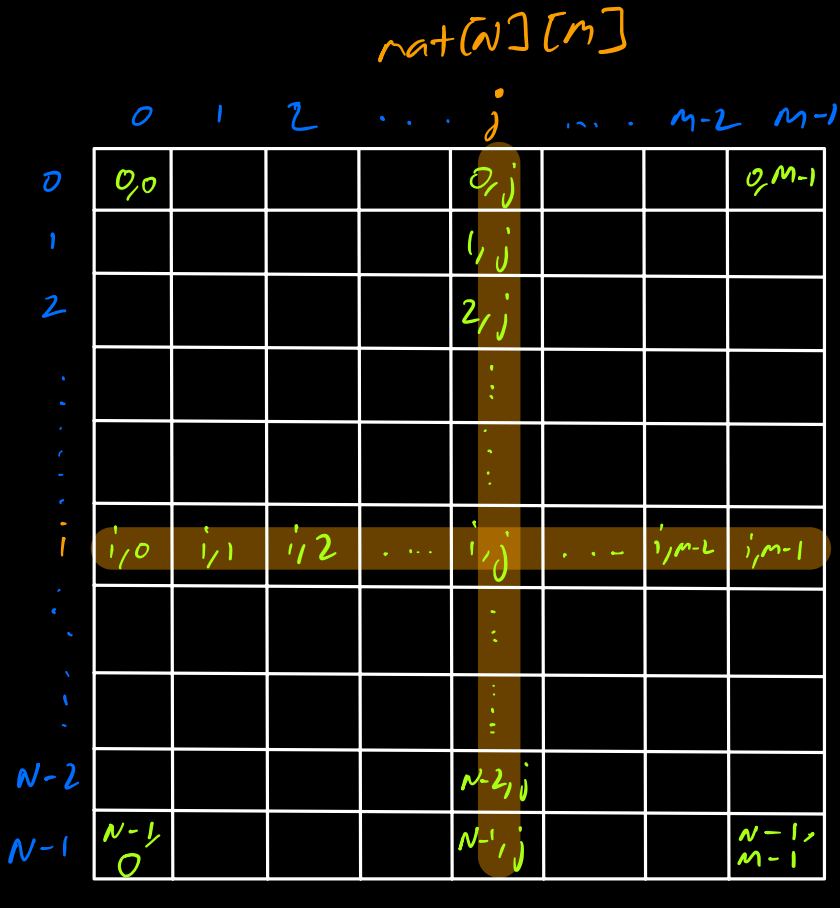
→ Rectangular of numbers, where each number is called an element.

int mat[N][M]

- M → Columns (Vertical)
- N → Rows (Horizontal)

	0	1	2	3	4
0	0,0	0,1	0,2	0,3	0,4
1	1,0	1,1	1,2	1,3	1,4
2	2,0	2,1	2,2	2,3	2,4
3	3,0	3,1	3,2	3,3	3,4
4	4,0	4,1	4,2	4,3	4,4

⇒ mat[5][5]



Obs: When iterating in a row

→ Row number is fixed

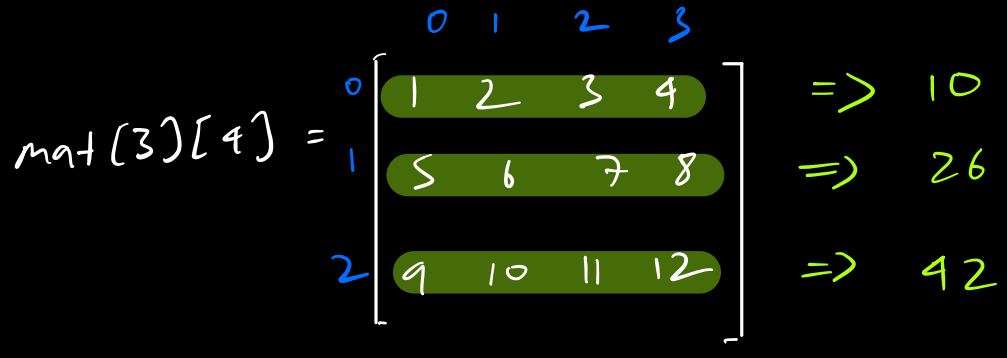
→ Column number goes from 0 to M-1

When iterating in a column

→ Column number is fixed

→ Row number goes from 0 to N-1

Q Given 2D Matrix, mat[N][M].
Print the row-wise sum.



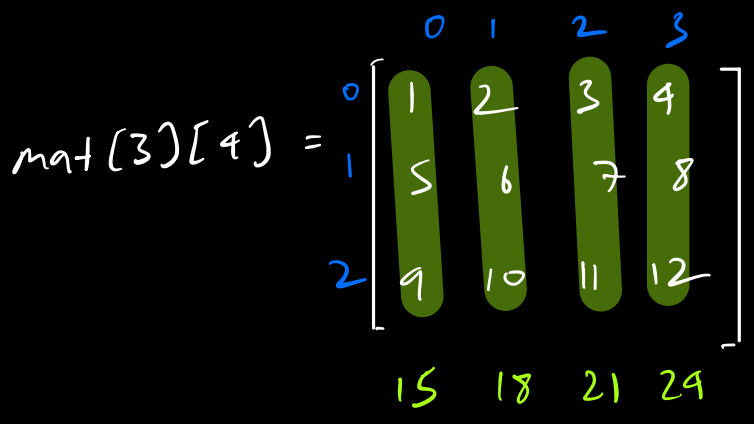

```
void sumRow (int mat[N][M]){
    int N = mat.size()
    int M = mat[0].size()

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

TC: O(N*M)

SC: O(1)

Q Given 2D Matrix, mat[N][M].
Print the row-wise Sum.



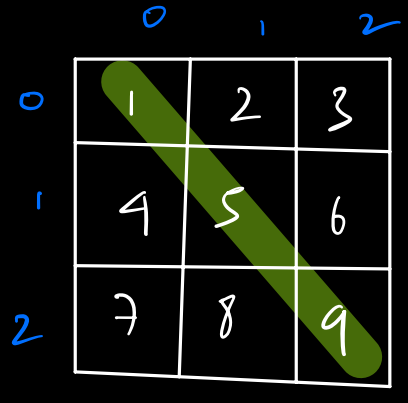

```
void colSum(int mat[N][M]){
    int N = mat.size()
    int M = mat[0].size()

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

TC: O(N*M)

SC: O(1)

Q Given a 2D Square matrix.
Print all the diagonals from left to right.



Principal (right) Diagonal

=> 1 5 9

How many diagonals in a Square matrix ? => 2

=> Only Printing Principal Diagonal Row number == Col number

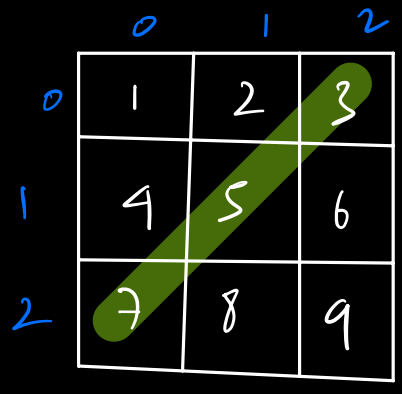
```
Void printDiagonal( int mat[N][N] ){
    int i = 0
    while(i < N){
        print(mat[i][i]);
        i = i+1;
    }
}
```

(0 0)
↓
(1,1)
↓
(2,2)
↓
(3,3)
↓
(N-1, N-1)

TC: O(N)

SC: O(1)

⇒ Print Anti-Diagonal



⇒ 3 5 7

Anti (Left) Diagonal

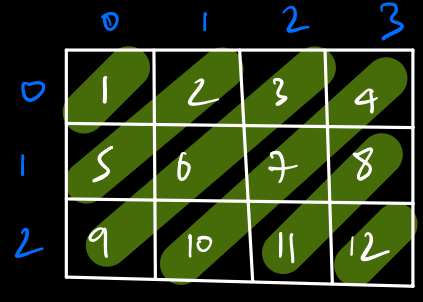
(0, 2)
↓
(1, 1)
↓
(2, 0)

Row += 1
Col -= 1

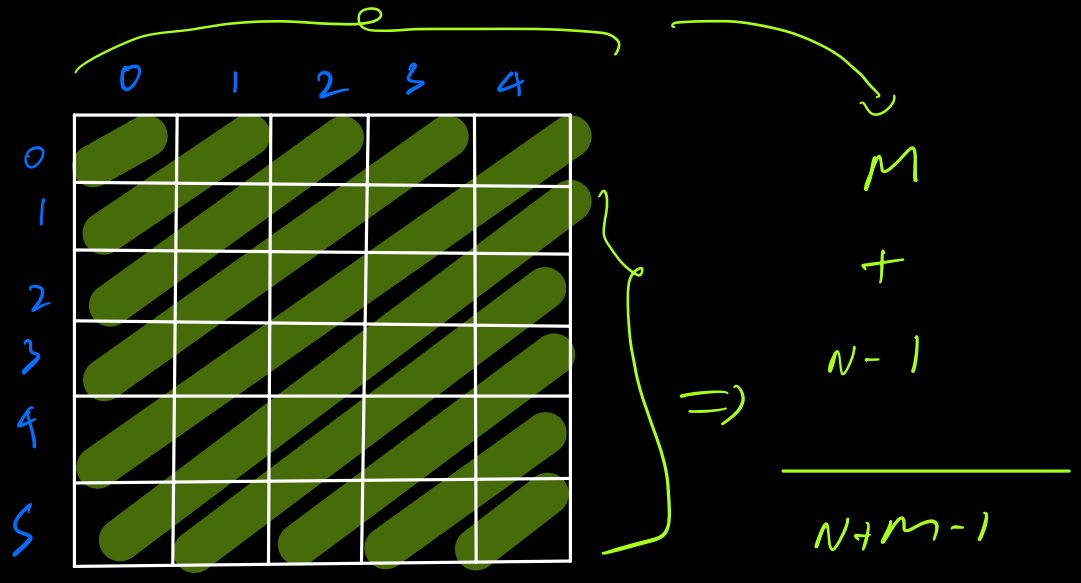
```
void printAntiDiagonal( int mat[N][N]){
    int i=0 , j= N-1
    while(i< N){ // we can also use j>=0
        print(mat[i][j]);
        i++ ;
        j-- ;
    }
}
```

TC: O(N)
SC: O(1)

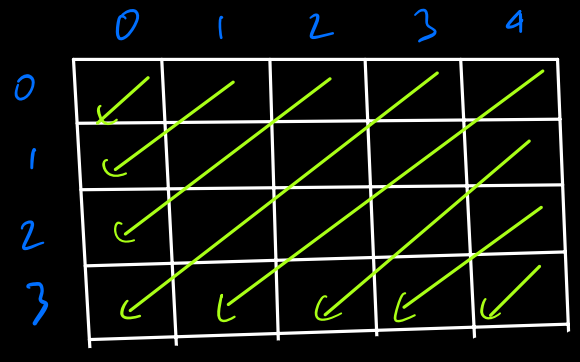
Q Print all anti-diagonals in a non-square matrix?



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



Q Given 4 x 5 matrix. How many right to left diagonals will be there?



5 => first Row

4-1=3 => last Column

8

(0,0)

(0,1) → (1,0)

(0,2) → (1,1) → (2,0)

(0,3) → (1,2) → (2,1) → (3,0)

row ++

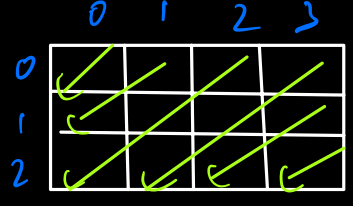
col --

```
void printAllAntiDiagonal(int mat[N][M]) {
  # First Row
  for(j=0; j<M; j++) {
      r = 0
      c = j

      while(r<N && c>=0) {
          print(mat[r][c]);
          r++;
          c--;
      }
      print('\n')
  }

  # Last Col
      1 as first row is done
  for(i=0; i<N; i++) {
      r = i
      c = M-1

      while(r<N && c>=0) {
          print(mat[r][c]);
          r++;
          c--;
      }
      print('\n');
  }
```



(0,0) → (1,-1)
(0,1) → (1,0) → (2,-1)
(0,2) → (1,1) → (2,0) → (3,-1)
(0,3) → (1,2) → (2,1) → (3,0)

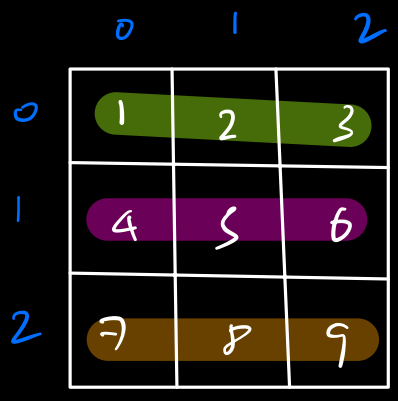
(1,3) → (2,2) → (3,1)

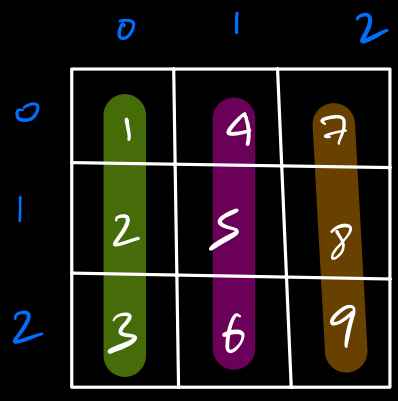
(2,3) → (3,2)

TC: O(N * M) SC: O(1)

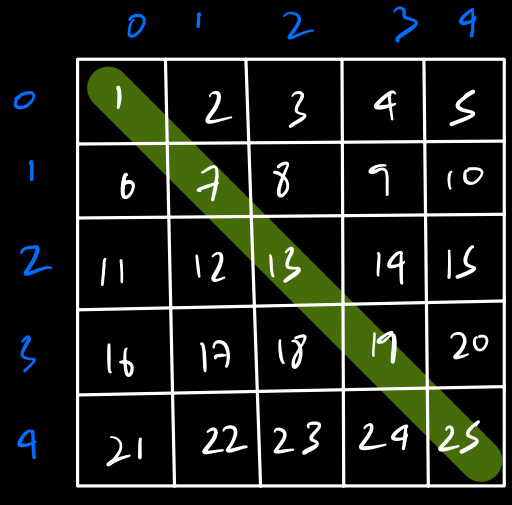
Q Given a square 2D matrix mat[N][N], Find transpose.

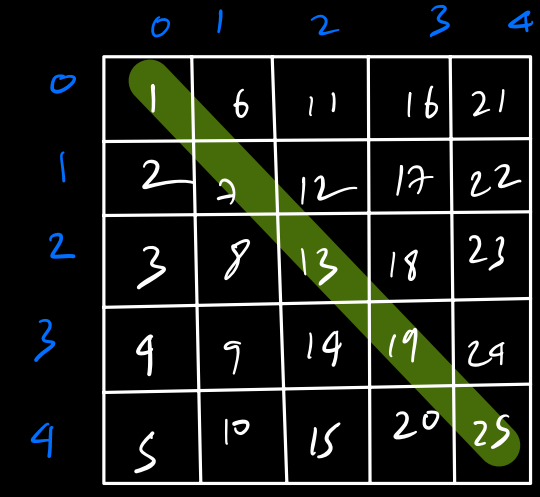
!! only valid for square matrix.





AS



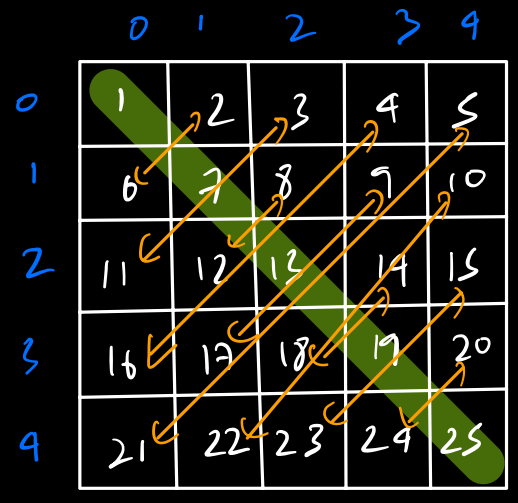


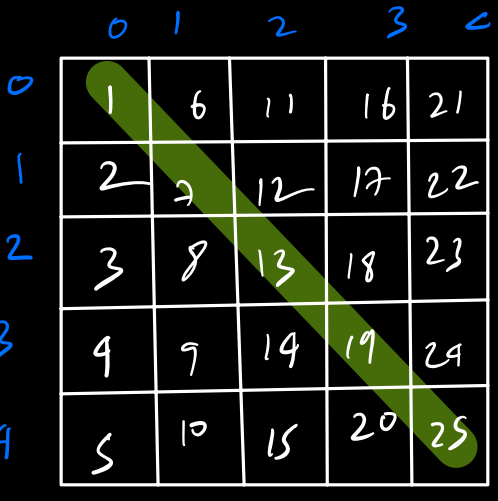
Brute Force: Create a new matrix and fill each column suffering new given matrix

TC: O(N * N) SC: O(N^2)

Can you do in SC: O(1), change the given matrix.

Obs: → Diagonals are same

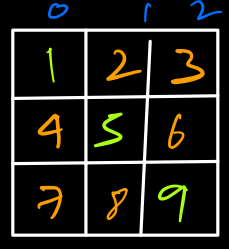




→ Considering diagonal as mirror rest of things are swapped

mat[i][j] => mat[j][i]

```
void transpose( matrix [N][m] ) {
    for(int i = 0; i < N; i++) {
        for(int j=0; j < N; j++) {
            int temp = mat[i][j]
            mat[i][j] = mat[j][i]
            mat[j][i] = temp
        }
    }
}
```



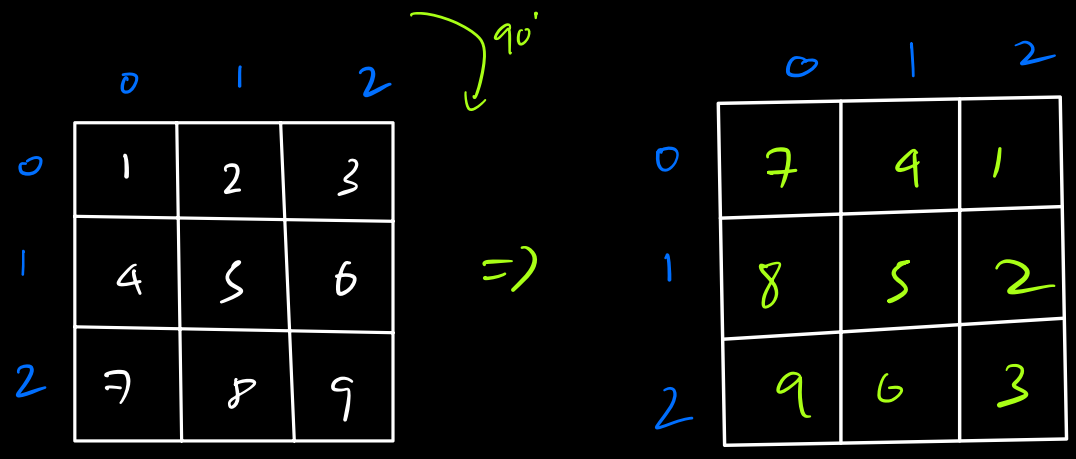
→ This does not works as swapping all cells will undo the first swaps.

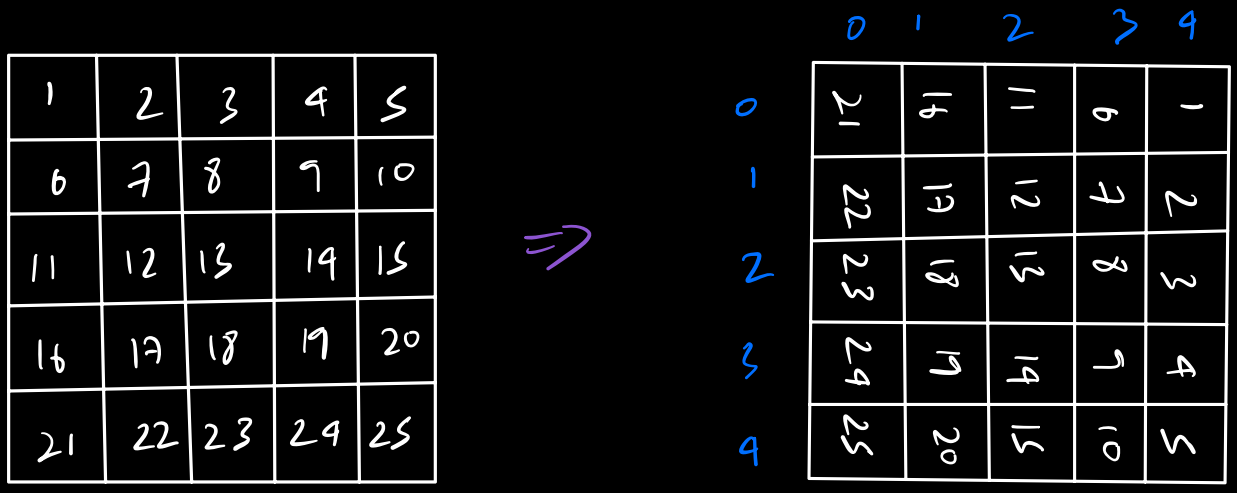
Sol: → Consider only upper triagle or lower triangle

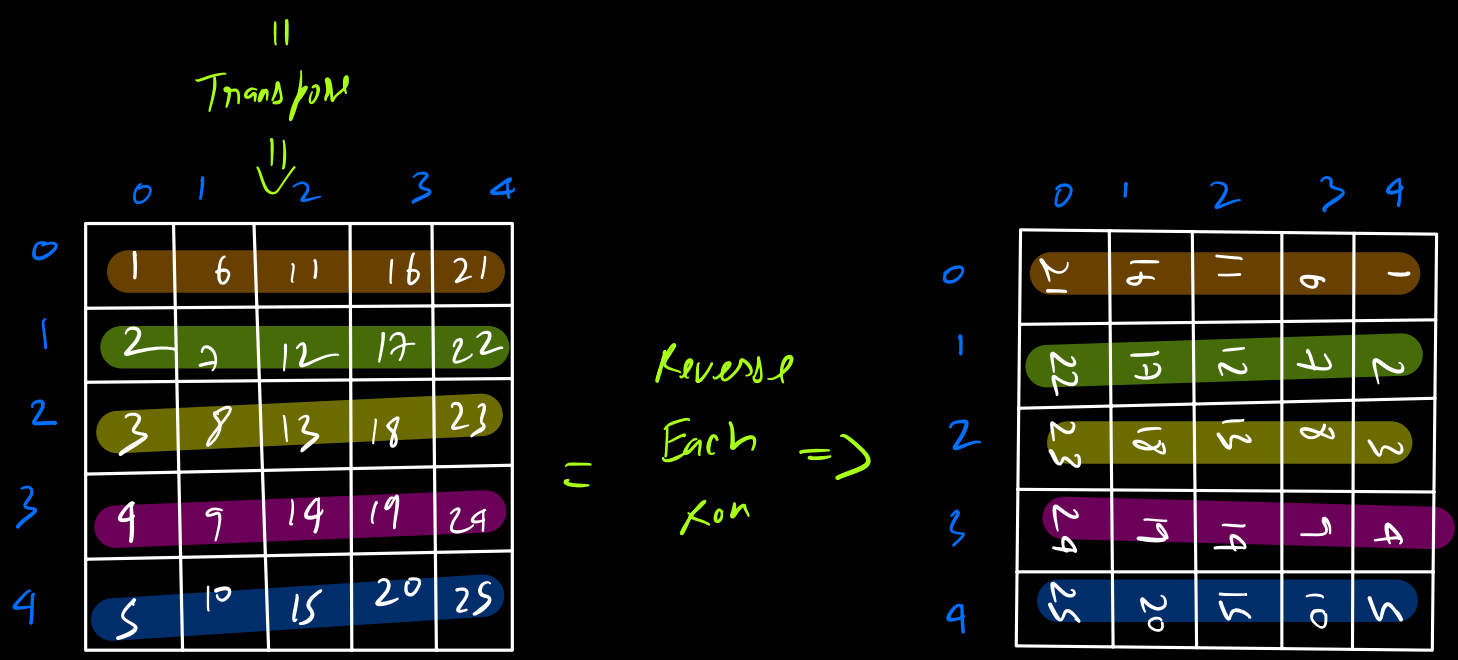
```
void transpose( matrix [N] [m] ) {
    for(int i = 0; i < N; i++) {
        for(int j = i+1; j < N; j++) {
            int temp = mat[i][j]
            mat[i][j] = mat[j][i]
            mat[j][i] = temp
        }
    }
}
```

TC: $O(N^2)$ SC: $O(1)$

Q Given a matrix [N] [N]. Rotate it to 90 degree clockwise





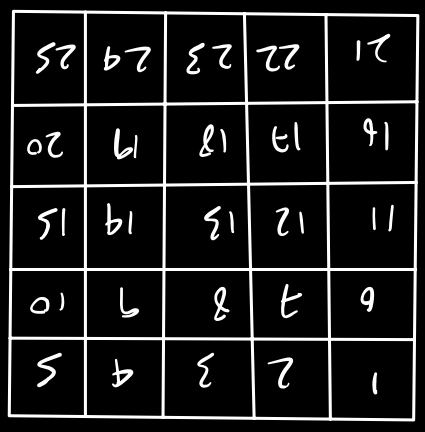


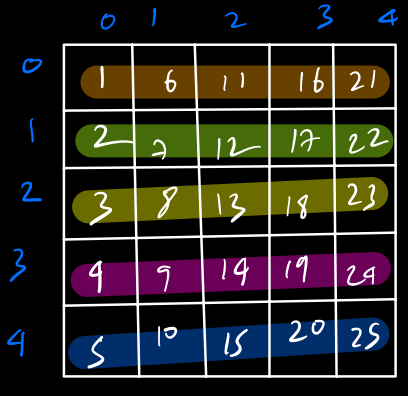
```
rotate(mat[N][M]){
mat = transpose(mat);
for(i=0; i<N; i++){
    reverse(mat[i]);
}
return matrix
}
```

Doubts

180° rotating

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



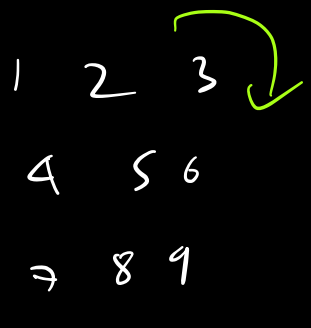


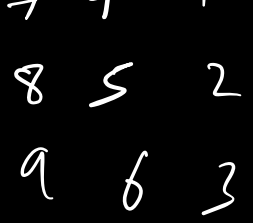
↳ Rows swapped

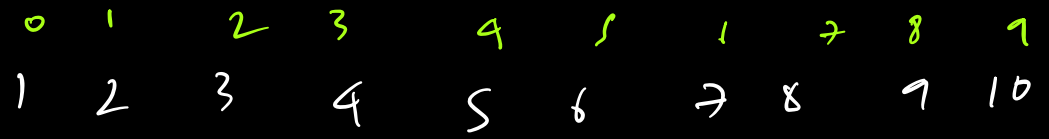
0 (=) N-1

1 (=) N-2

↳ Rivers each row







$K = 5$

$\Delta t =$ 0 ⇒ 5
1 ⇒ 5
2 ⇒ 5
3 ⇒ 5
4 ⇒ 5
5 ⇒ 5

30

$10 \times 10 = 100$

$\frac{10 \times 10}{4}$

$N = 10^3$

$\frac{10^3 \times 10^3}{4} = O(N^2)$

$\simeq 10^5$

$O(N^2)$ not $O(N)$