→ traversal in 2D array

arr

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

size = 4 * 3

```
int rows = arr.length;
int cols = arr[0].length;
for(int i=0; i< rows ; i++){
    for(int j=0; j< cols ; j++){
        Syso( arr[i][j] + " ");
    }
    Sysoln();
}
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int rows = scn.nextInt();
    int cols = scn.nextInt();
    int[][] arr = new int[rows][cols];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            arr[i][j] = scn.nextInt();
        }
    }

    printMatrix(arr, rows, cols);
}
public static void printMatrix(int[][] arr, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            System.out.print( arr[i][j] + " " );
        }
        System.out.println();
    }
}
```

arr

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

i=0, j=0 (0,0)
i=0 j=1 (0,1)
j=2 (0,2)
i=1, j=0 (1,0)
j=1 (1,1)
j=2 (1,2)
i=2, j=0 (2,0)
j=1 (2,1)
j=2 (2,2)
i=3, j=0 (3,0)
j=1 (3,1)
j=2 (3,2)

1_2_3_
4_5_6_
7_8_9_
10_11_12_

$T.C = O(\text{rows} * \text{cols})$

$O(m * n)$

Note :- linear

Print Alternate Rows

arr

	0	1	2	3
0	1	2	3	20
1	4	5	6	19
2	7	8	9	18
3	10	11	12	17
4	13	14	15	16

O/P

1 _ 2 _ 3 _ 20 _
7 _ 8 _ 9 _ 18 _
13 _ 14 _ 15 _ 16 _

size = 5 * 4

no. of rows

no. of cols

```
int rows = arr.length;
int cols = arr[0].length;
for(int i=0; i< rows ; i+=2){
    for(int j=0; j< cols ; j++){
        Syso( arr[i][j] + " ");
    }
    Sysoln();
}
```

T.C = O(rows * cols)

Print Upper triangular matrix 1

arr

	j = 0	j = 1	j = 2	j = 3	j = 4
i = 0	1	2	3	4	5
i = 1	6	7	8	9	10
i = 2	11	12	13	14	15
i = 3	16	17	18	19	20
i = 4	21	22	23	24	25

(i, j)

(indexes)

(0,0) ✓
(0,1) ✓
(0,2) ✓
(0,3) ✓
(0,4) ✓
(1,0)
(1,1) ✓
(1,2) ✓
(1,3) ✓
(1,4) ✓
(2,0)
(2,1)
(2,2) ✓
(2,3) ✓
(2,4) ✓
(3,0)
(3,1)
(3,2)
(3,3) ✓
(3,4) ✓
(4,0)
(4,1)
(4,2)
(4,3)
(4,4) ✓

pattern

(j >= i)

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int m = scn.nextInt();
    int n = scn.nextInt();
    int[][] arr = new int[m][n];
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            arr[i][j] = scn.nextInt();
        }
    }
    printUpperTraingle(arr);
}
public static void printUpperTraingle(int[][] arr) {
    int rows = arr.length;
    int cols = arr[0].length;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if ( j >= i ) {
                System.out.print(arr[i][j] + " ");
            } else {
                System.out.print("0" + " ");
            }
        }
        System.out.println();
    }
}
```

T.C = O(m*n)

S.C = O(1)

Print the matrix left-diagonal wise

(left diagonal
right diagonal ← (i == j))

arr

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

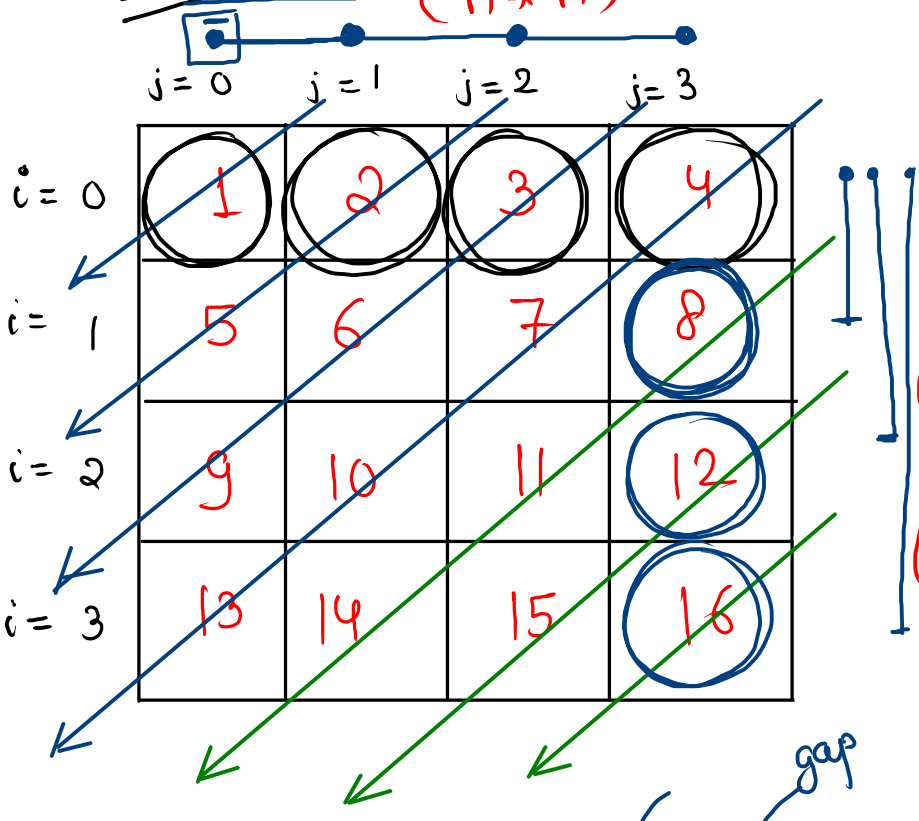
output

1 2 5 3 6 9 4 7 10

13 8 11 14 12 15 16

logic

(n*n)



loop :-

- from where to start
- when to stop
- how to move

starting from (0, 0), (0, 1), (0, 2), (0, 3)

gap gap gap gap

how to move i++ , j--

when to stop j >= 0

---

first half

```
for(int gap=0; gap<n ; gap++){
    for(int i=0, j=gap ; j>=0 ; i++, j--){
        Syso( arr[i][j] + " " );
    }
}
```

second half

$\boxed{n = 4}$

starting

(1, 3)

(2, 3)

(3, 3)

$i = gap$ $j = n-1$

when to stop

$i < n$

how to move

$i{+}{+}$

$j{-}{-}$

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[][] arr = new int[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            arr[i][j] = scn.nextInt();
        }
    }
    leftDiagonal(arr, n);
}
public static void leftDiagonal(int[][] arr, int n) {

    for (int gap = 0; gap < n; gap++) {
        for (int i = 0, j = gap; j >= 0; i++, j--) {
            System.out.print(arr[i][j] + " ");
        }
    }

    for (int gap = 1; gap < n; gap++) {
        for (int i = gap, j = n - 1; i < n; i++, j--) {
            System.out.print(arr[i][j] + " ");
        }
    }

}
```

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

$S.C = O(1)$

h.w

code

arr

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