

⇒ 2D array

0 1 2 3 4 m

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

finding
length:

$n = \text{arr.length};$ // no. of rows

$m = \text{arr}[0].\text{length};$ // no. of cols.

\boxed{n} $\text{int}[\] \text{arr} = \text{new int}[n]$

$n=5, m=5$ ↓

$\text{int}[\][\] \text{arr} = \text{new int}[n][m];$

access:-

↳ $\text{arr}[2][2]$

Print the Matrix Row-wise

```
public static void main(String[] args) {  
    Scanner scn = new Scanner(System.in);  
    → int m = scn.nextInt(); // row no  
    int n = scn.nextInt(); // col no  
    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();  
        }  
    }  
  
    m → for (int i = 0; i < m; i++) { // rows  
        n → for (int j = 0; j < n; j++) { // cols  
            System.out.print(arr[i][j] + " ");  
        }  
        System.out.println();  
    }  
}
```

Operations = $m * n$

T.C = $O(m * n)$

S.C = $O(1)$

Print Alternate Row

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

code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int m = scn.nextInt(); // row no
    int n = scn.nextInt(); // col no
    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();
        }
    }

    printAlternateRows(arr, m, n);
}

public static void printAlternateRows(int[][] arr, int m, int n) {
    for (int i = 0; i < m; i += 2) { // rows
        for (int j = 0; j < n; j++) { // cols
            System.out.print(arr[i][j] + " ");
        }
        System.out.println();
    }
}
```

also valid

```
public static void printAlternateRows(int[][] arr, int m, int n) {
    for (int i = 0; i < m; i++) { // rows
        if (i % 2 == 0) {
            for (int j = 0; j < n; j++) { // cols
                System.out.print(arr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Print Upper triangular matrix 1

	$j = 0$	$j = 1$	$j = 2$	$j = 3$	$j = 4$
$i = 0$	1	2	3	4	5
<u>$i = 1$</u>	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

which values to
print

$i = 0, j = 0, 1, 2, 3, 4$

$i = 1, j = 1, 2, 3, 4$

$i = 2, j = 2, 3, 4$

$i = 3, j = 3, 4$

$i = 4, j = 4$

condition $i \leq j$

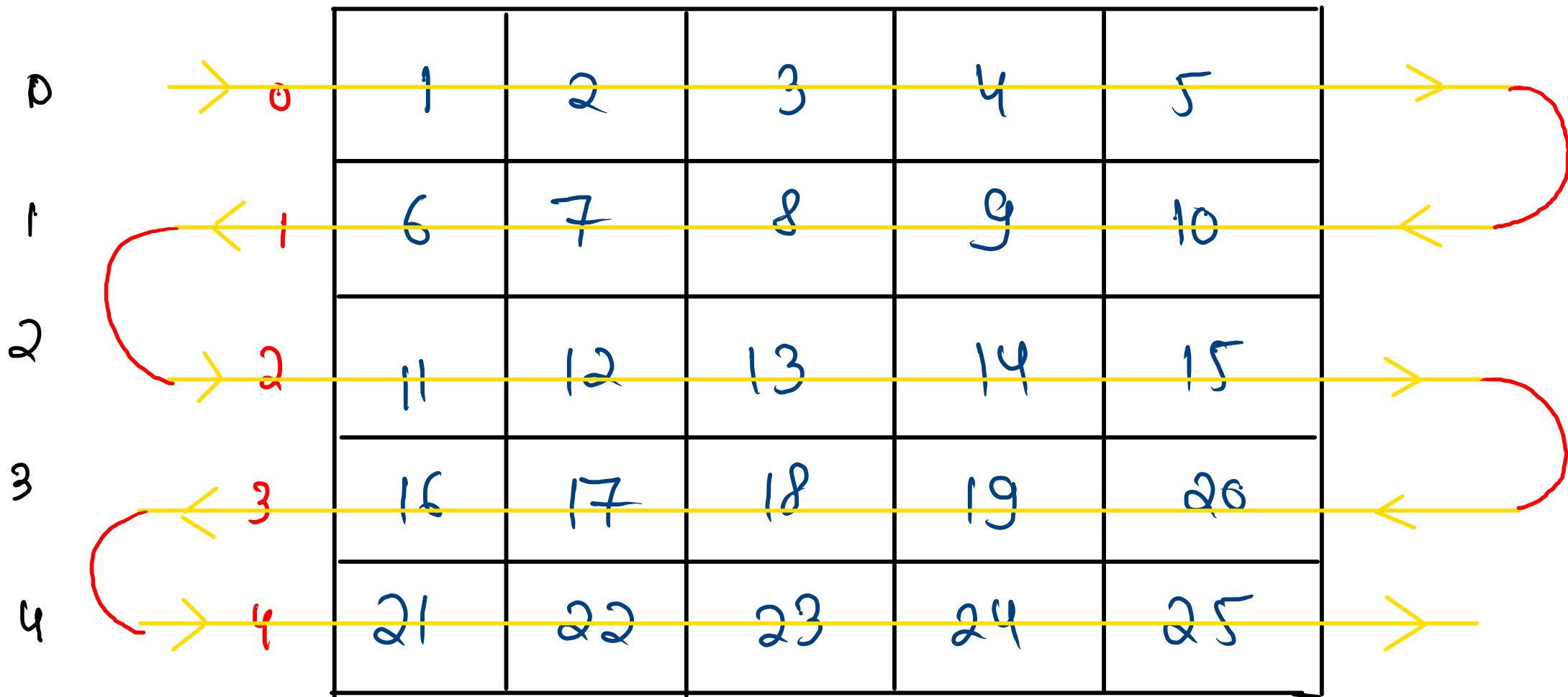
code

```
public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN.
    Scanner scn = new Scanner(System.in);
    int m = scn.nextInt(); // row no
    int n = scn.nextInt(); // col no
    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();
        }
    }
    upperMatrix(arr);
}

public static void upperMatrix(int[][] arr) {
    int m = arr.length;
    int n = arr[0].length;

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (i <= j)
                System.out.print(arr[i][j] + " ");
            else
                System.out.print("0 ");
        }
        System.out.println();
    }
}
```

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



Code

```
public static void main(String[] args) {  
    int[][] arr = {  
        {1, 2, 3, 4},  
        {1, 5, 3, 5},  
        {9, 8, 3, 0},  
        {2, 4, 3, 9}  
    };  
    // 1 2 3 4 5 3 5 1 9 8 3 0 9 3 4 2  
    for (int i = 0; i < arr.length; i++) {  
        if ( i % 2 == 0 ) {  
            for (int j = 0; j < arr[0].length; j++) {  
                System.out.print(arr[i][j] + " ");  
            }  
        } else {  
            for (int j = arr[0].length - 1; j >= 0; j--) {  
                System.out.print(arr[i][j] + " ");  
            }  
        }  
    }  
}
```


One)

A 5x5 grid of numbers from 0 to 25, arranged in rows and columns. The numbers are written in blue. The grid is surrounded by yellow arrows indicating a path that visits every cell exactly once. The path starts at the top-left cell (0) and proceeds in a boustrophedon pattern, alternating between left-to-right and right-to-left rows. The path ends at the bottom-right cell (25). The yellow arrows are labeled with red numbers 0 through 4, indicating the sequence of the path.

0	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	

Ques print :- 1, 7, 13, 19, 25, 2, 8, 14, 20, 3, 9, 15, 4, 10, 5

gap = 0 g = 1 g = 2 g = 3 g = 4 j

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

Ques

what is the
starting point,
when to stop,
how to move

```

public static void main(String[] args) {
    int[][] arr = {
        {1, 2, 3, 4},
        {1, 5, 3, 5},
        {9, 8, 3, 0},
        {2, 4, 3, 9}
    };
    // 1 5 3 9 2 3 0 3 5 4
    int n = arr.length;
    for (int gap = 0; gap < n; gap++) {
        for (int i = 0, j = gap; j < n; i++, j++) {
            System.out.print(arr[i][j] + " ");
        }
    }
}

```

gap \rightarrow 0 \rightarrow 4

for ($i=0, j=0,1,2,3,4$, $j < n$, $i++, j++$)

i	j
0	0
0	1
0	2
0	3
0	4