

Today's Questions:

- ① State of Wakanda I { zig-zag order }
- ② State of Wakanda II { Diagonal order }
- ③ Special Order
- ④ Exit Point

① State of Wakanda I

	0	1	2	3
0	11	12	13	14
1	21	22	23	24
2	31	32	33	34
3	41	42	43	44

order \rightarrow 11, 21, 31, 41, 42, 32, 22, 12,
 13, 23, 33, 43, 44, 31, 24, 14

start from column zero upto $m-1$

& alternatively traverse from (0 to $n-1$) or ($n-1$ to 0)

print the corresponding element

update the column

CODE

```

void waveTraversals (int[][] mat) {
    int rows = mat.length, col = mat[0].length;
    for (int j=0; j< col; j++) {
        if (j%2 == 0)
            for (int i=0; i< rows; i++)
                System.out.println(mat[i][j]);
        else
            for (int i=rows-1; i>=0; i--)
                System.out.println(mat[i][j]);
    }
}
    
```

To print matrix column by column

```

for (int j=0; j<m; j++){
    for (int i=0; i<n; i++){
        System.out.print(mat[i][j] + "\t");
    }
    System.out.println();
}

```

(2) State of Wakanda II

for matrix



traverse as: 1 → 5 → 9 → 2 → 6 → 3.

observation, diagonal starts always from $i=0$ / $row=0$
and column always increases by 1 for
every diagonal start.

after the starting point,
print every element
 $stRow++$, $stCol++$
upto when $col < n$.

CODE:

```
int j=0;
```

```
while (j < n) {
```

```
    int row = 0, col = j;
```

```
    while (col < n) {
```

```
        System.out.println(mat[row][col]);
```

```
        row++;
```

```
        col--;
```

```
}
```

```
y  
}++;
```

$\overbrace{\quad \quad \quad}^{\text{j}} \text{columns.}$

0	1	2	3	4
---	---	---	---	---

0	11	12	13	14	15	11	22	33	44	\rightarrow	12	23	34	\rightarrow	13	24	\rightarrow	14
---	----	----	----	----	----	----	----	----	----	---------------	----	----	----	---------------	----	----	---------------	----

1	21	22	23	24	25	\downarrow					\downarrow				\downarrow			
---	----	----	----	----	----	--------------	--	--	--	--	--------------	--	--	--	--------------	--	--	--

2	31	32	33	34	35	$i = j$					$j = i + 1$				$j = i + 2$			
---	----	----	----	----	----	---------	--	--	--	--	-------------	--	--	--	-------------	--	--	--

3	41	42	43	44	45	$row = col$					$row + 1 = col$				$row = col - 2$			
---	----	----	----	----	----	-------------	--	--	--	--	-----------------	--	--	--	-----------------	--	--	--

4	51	52	53	54	55	$(j - i) = 0$					$j - i = 1$				$j - i = 2$			
---	----	----	----	----	----	---------------	--	--	--	--	-------------	--	--	--	-------------	--	--	--

keep variable

gap \rightarrow increases by 1

every time from 0 to $n-1$

or gap can go from 0 to number of columns in rectangle matrix

```
for (gap = 0; gap < n; gap++) {
```

```
    for (int i = 0; i + gap < n; i++)
```

```
        System.out.println(mat[i][i + gap]);
```

or

```
for (int gap = 0; gap < m; gap++) { // gap = no. of columns.
```

// gap = $j - i \rightarrow j = i + gap$.

```
for (int i = 0; i + gap < n; i++)
```

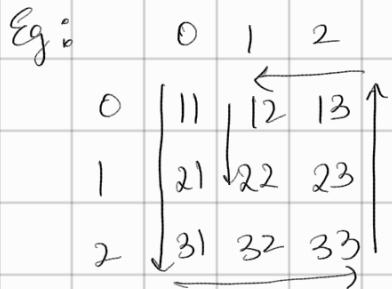
```
    System.out.println(mat[i][i + gap]);
```

```
}.
```

or

```
for (int gap=0; gap < m; gap++) {  
    for (int i=0, j=gap; j < m; i++, j++)  
        System.out.println(mat[i][j]);  
}
```

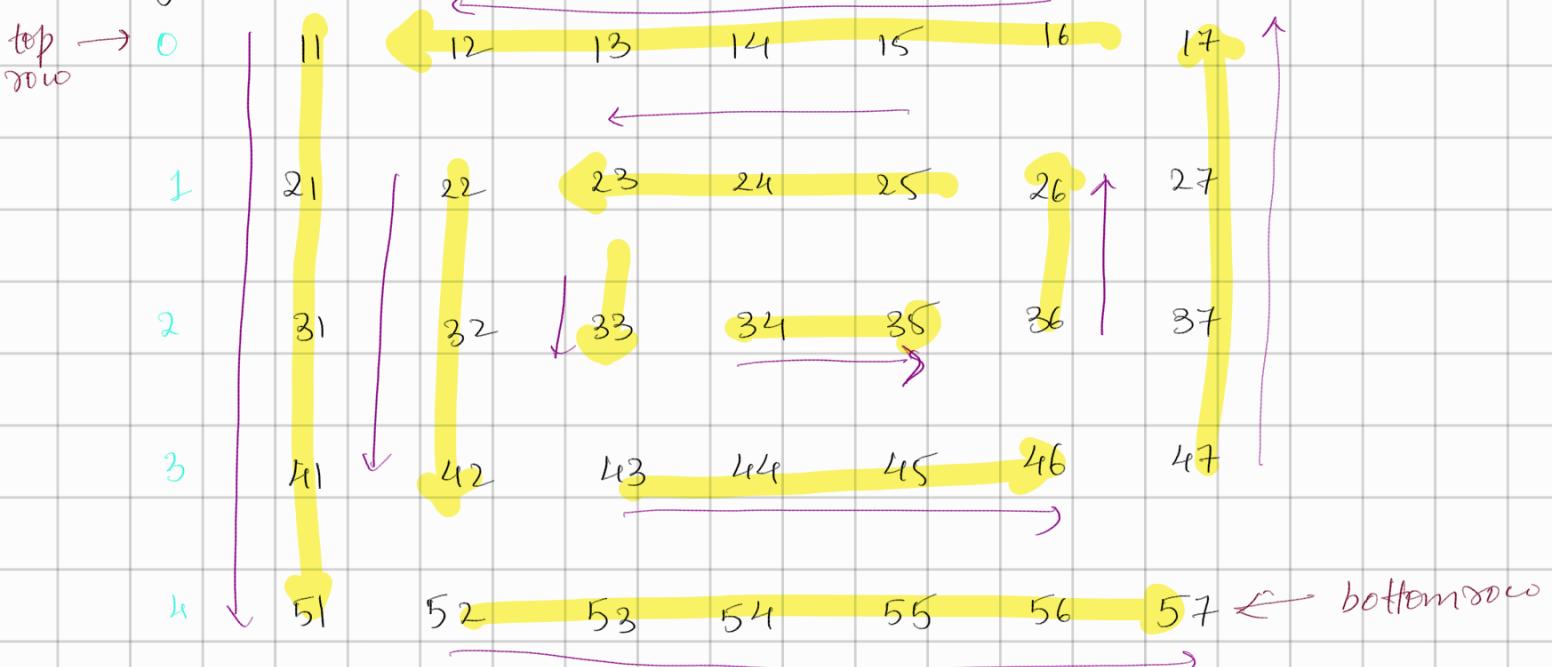
③ Spiral Order.



11 21 31 32 33 23 13 12 22

↑ top column

Eg:



{ Left wall: Top to Bottom
Bottom wall: Left to Right
Right wall: Bottom to Top
Top wall: Right to left }

bottom column.

for outermost spiral

Print corner elements only once

for outermost loop:

Left : loop from $i=0$ to $n-1$ print $\text{mat}[i][0]$

Bottom: loop from $j=1$ to $m-1$ print $\text{mat}[n-1][j]$

Right: loop from $i=n-2$ to 0 print $\text{mat}[i][m-1]$

Top : loop from $j= m-2$ to 1 print $\text{mat}[0][j]$

Logic

Use Row & Column for corners.

→ Left wall print from first Row to last Row in first col

Bottom wall print from first col+1 to last col in last Row

Right wall print from last Row -1 to first Row in last col

Top wall print from last col-1 to first col+1 in first Row ✓

update first Row++, first col++, last col--, last Row--;

for middle elements to avoid duplicate elements

always check & return if completed all elements

CODE

```
int fRow=0, fCol=0, lRow= n-1, lCol= m-1;
```

```
int count = 0;
```

```
while (count < n*m)
```

// left Wall from top to bottom

```
for( int i= fRow ; i <= lRow ; i++ ) {
```

```
System.out.println( mat [i][fCol] );
```

```
count++;
```

```
if (count == n*m) return;
```

```
}
```

```
fCol++;
```

// Bottom wall from left to right

```
for( int j= fCol ; j <= lCol ; j++ ) {
```

```

for( int j= fRow; j <= lRow; j++ ){
    System.out.println( mat[ lRow ][ j ] );
    count++;
    if( count == n * m ) return;
}

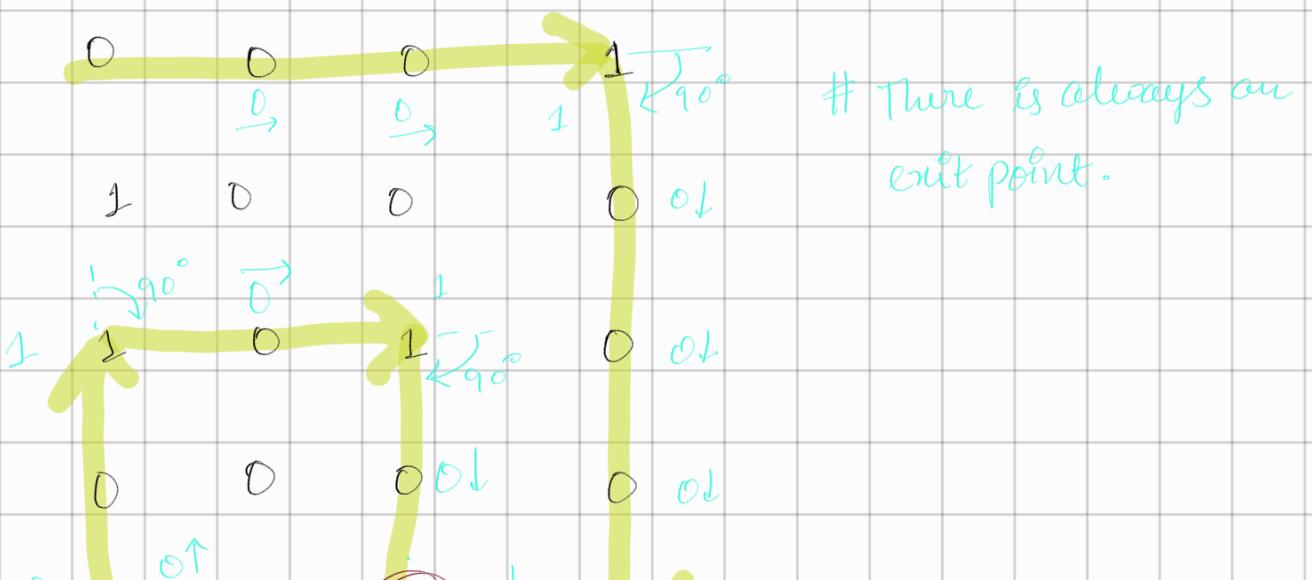
lRow--;
// Right wall from bottom to top
for( int i= lRow; i >= fRow; i-- ){
    System.out.println( mat[ i ][ lCol ] );
    count++;
    if( count == n * m ) return;
}

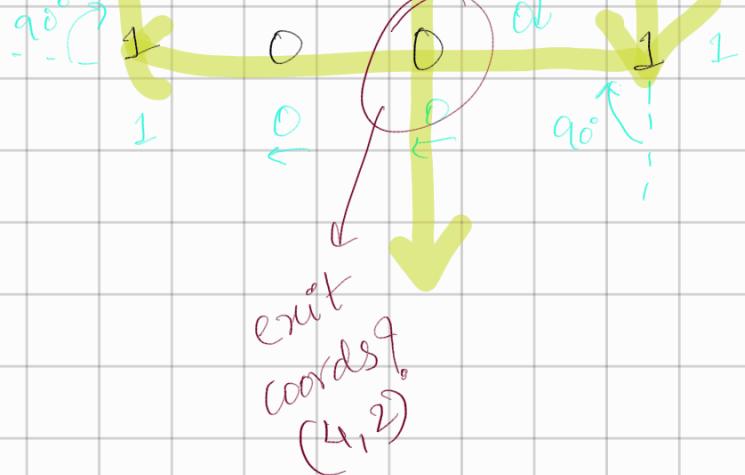
lCol--;
for( int j= lCol; j >= fCol; j-- ){
    System.out.println( mat[ fRow ][ j ] );
    count++;
    if( count == n * m ) return;
}

fRow++;
}

```

(4) Exit Point: On 0, move same way, on 1, move 90° right.
 Constructive algorithm.





$i-1$	$i-1$	$i-1$
$j-1$	j	$j+1$
i	(i, j)	$j+1$
$j-1$	j	$j+1$
$i+1$	$i+1$	$i+1$
$j-1$	j	$j+1$

to go down \rightarrow row + 1, col
 to go left \rightarrow row, col - 1
 to go right \rightarrow row, col + 1
 to go up \rightarrow row - 1, col

top left diagonal: row - 1, col - 1

top right diagonal: row - 1, col + 1

bottom left diagonal: row + 1, col - 1

bottom right diagonal: row + 1, col + 1.

Top wall exit: currRow >= 0

Left wall exit: currCol >= 0

Bottom wall exit: currRow < n

Right wall exit: currCol < m.

If printed directly, out of bounds value will be printed.

So maintain 2 variables for prevRow & prevCol.

OR

use last direction to go back & print exit point.

