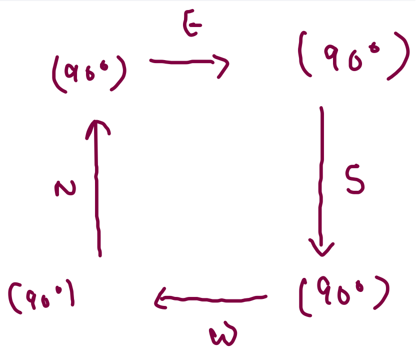
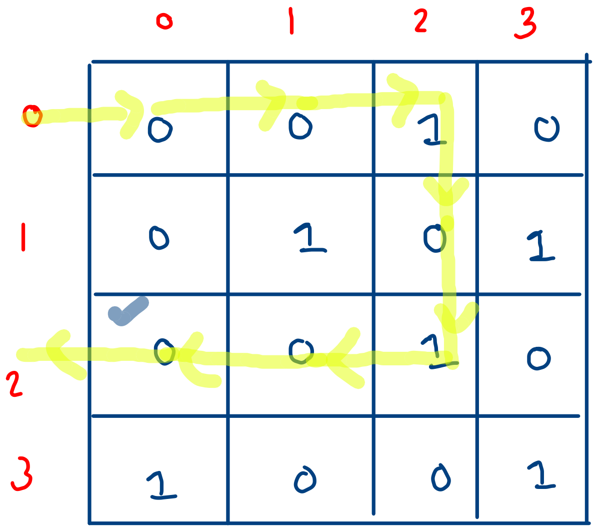
1. You are given a number n, representing the number of rows.
2. You are given a number m, representing the number of columns.
3. You are given n*m numbers (1's and 0's), representing elements of 2d array a.
4. Consider this array a maze and a player enters from top-left corner in east direction.
5. The player moves in the same direction as long as he meets '0'. On seeing a 1, he takes a 90 deg right turn.
6. You are required to print the indices in (row, col) format of the point from where you exit the matrix.

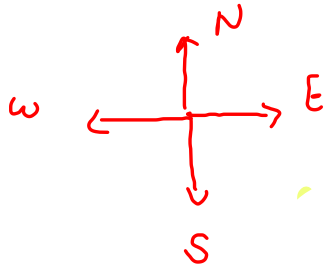
exit point

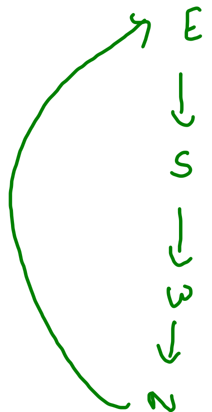
(2, 0)

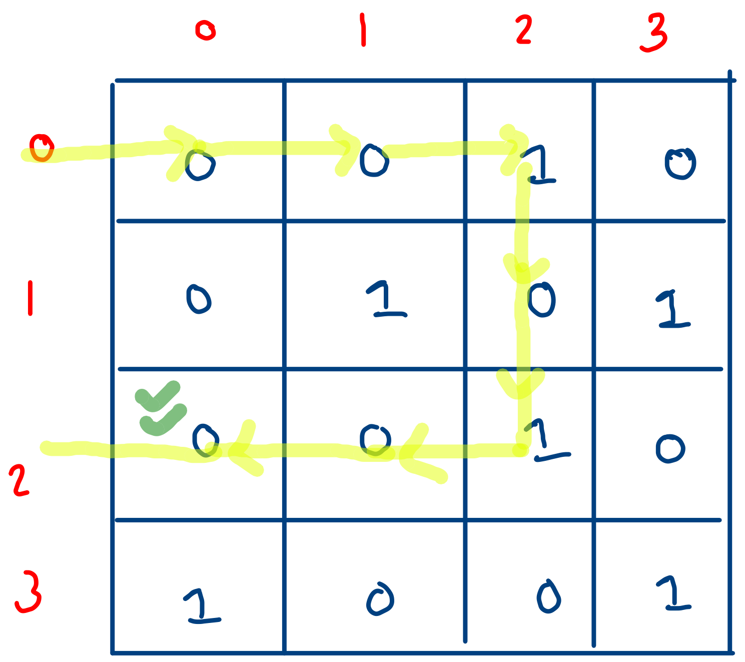




$$\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
2 & 0 & 0 & 1 & 0 \\
3 & 1 & 0 & 0 & 1
\end{array}$$



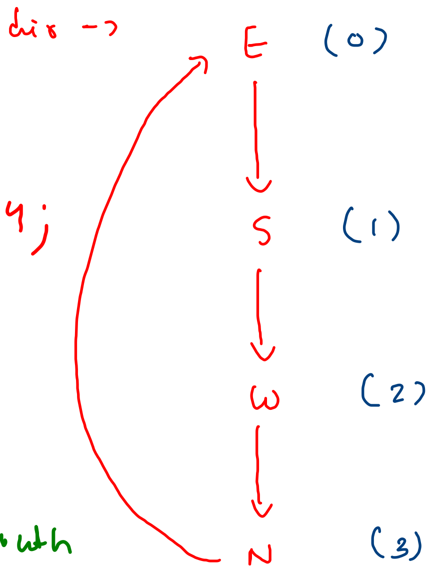




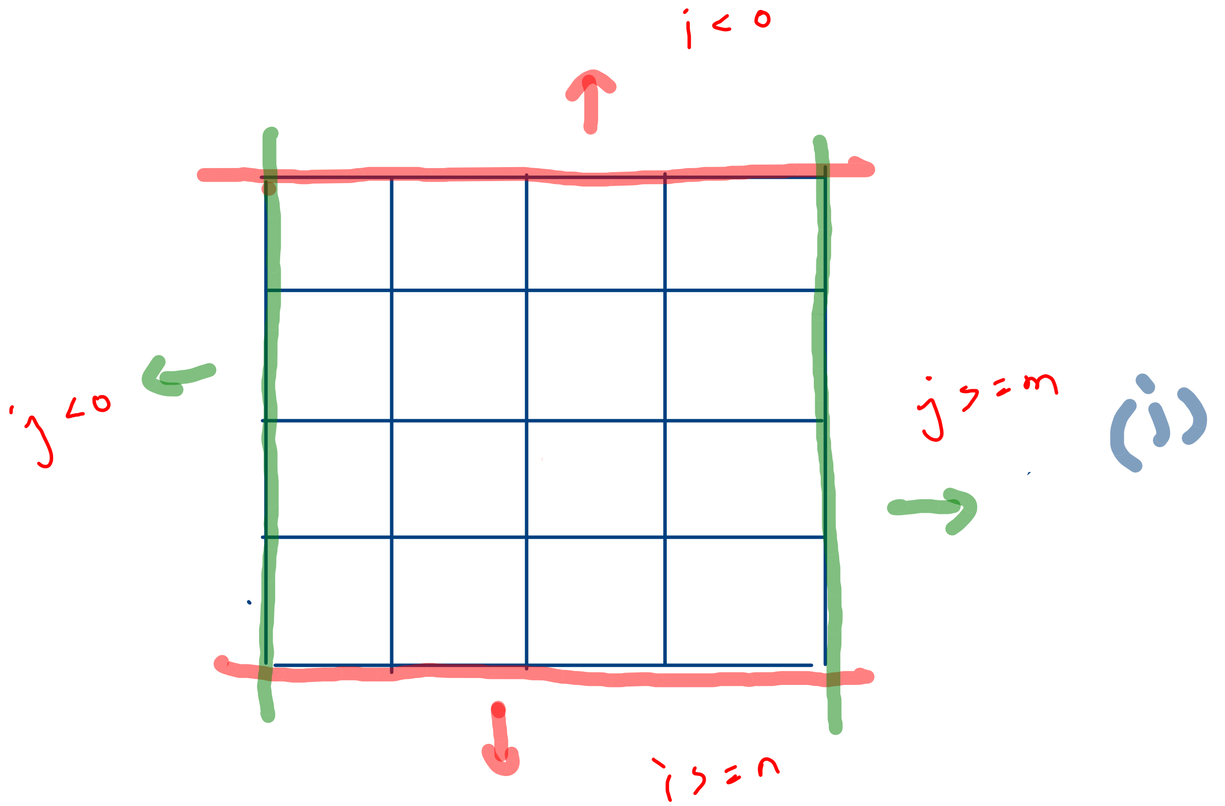
$\underline{(i, j)}$

```
dir = (dir + mat[i][j]) % 4;

if(dir == 0){ //east
        j++;
}
else if(dir == 1){ //south
        i++;
}
else if(dir == 2){ //west
        j--;
}
else { //north
        i--;
}
```

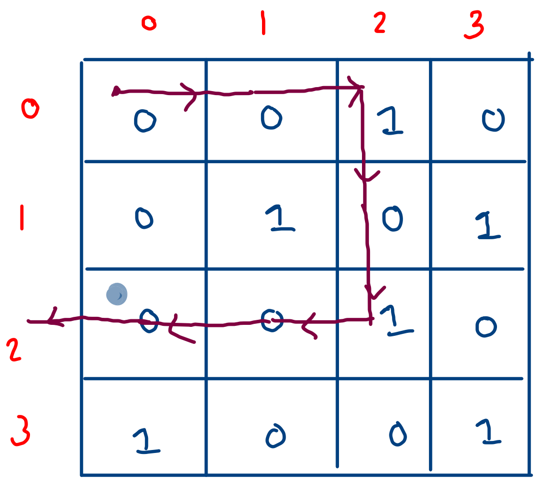


$\boxed{\begin{array}{l} dir = 0 \\ i = 0,\ j = 0 \end{array}}$



$i >= 0$ && $i < n$ &&

$j >= 0$ && $j < m$

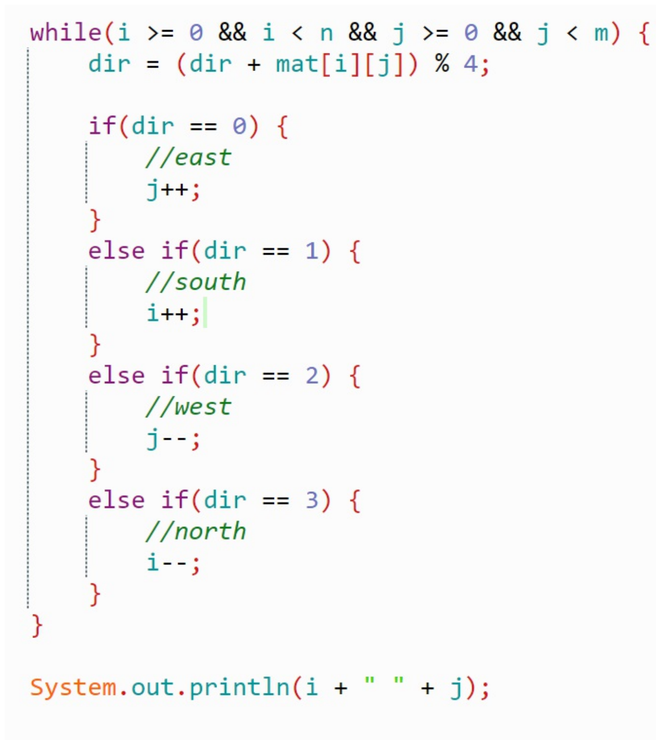


$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 \end{array}$$

$i = \cancel{0}\ \cancel{1}\ 2$

$j = \cancel{0}\ \cancel{1}\ \cancel{2}\ \cancel{1}\ \cancel{0}\ -1$

$dir = 2 + 0 = 2$

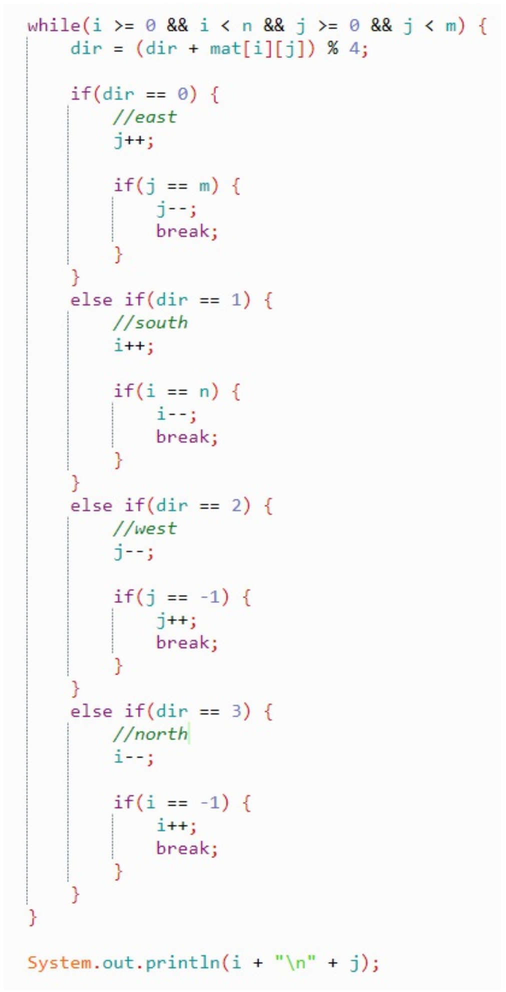


```
while(i >= 0 && i < n && j >= 0 && j < m) {
    dir = (dir + mat[i][j]) % 4;

    if(dir == 0) {
        //east
        j++;
    }
    else if(dir == 1) {
        //south
        i++;
    }
    else if(dir == 2) {
        //west
        j--;
    }
    else if(dir == 3) {
        //north
        i--;
    }
}

System.out.println(i + " " + j);
```

$(2, -1)$



```
while(i >= 0 && i < n && j >= 0 && j < m) {
    dir = (dir + mat[i][j]) % 4;

    if(dir == 0) {
        //east
        j++;

        if(j == m) {
            j--;
            break;
        }
    }
    else if(dir == 1) {
        //south
        i++;

        if(i == n) {
            i--;
            break;
        }
    }
    else if(dir == 2) {
        //west
        j--;

        if(j == -1) {
            j++;
            break;
        }
    }
    else if(dir == 3) {
        //north
        i--;

        if(i == -1) {
            i++;
            break;
        }
    }
}

System.out.println(i + "\n" + j);
```

$n = 5$

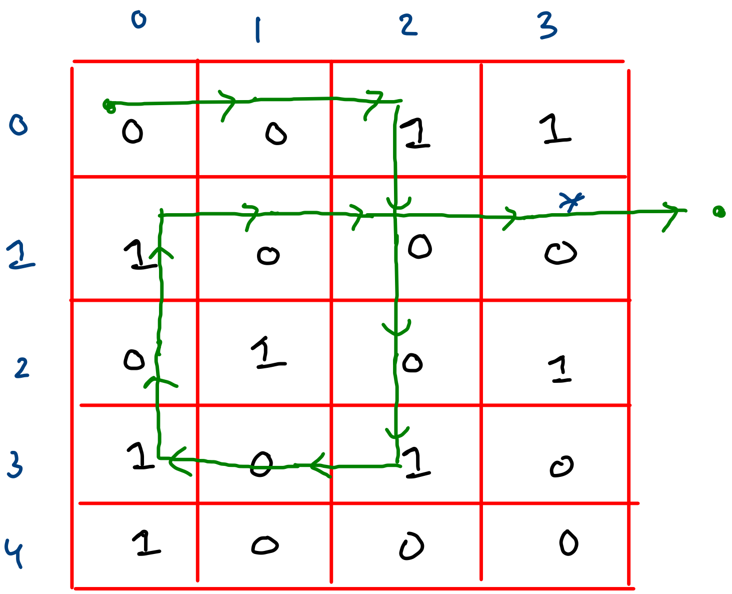
$m = 4$

$i =$ ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ 1

$j =$ ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 3

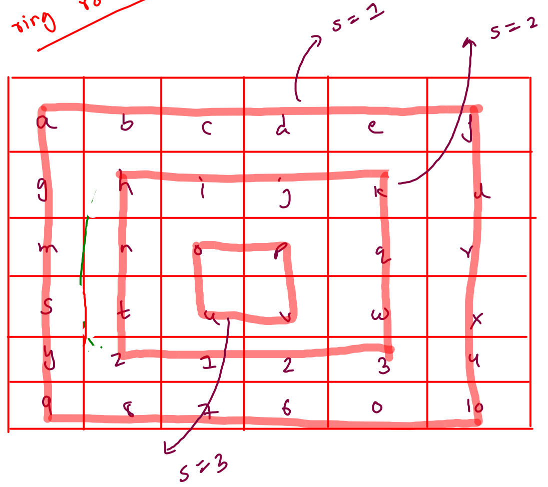
$dir = 0 + 0 = 0$

$(1, 3)$



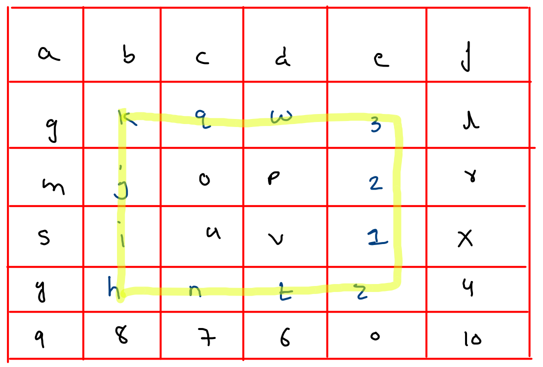
$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 0 \\ 4 & 1 & 0 & 0 & 0 \end{array}$$

ring rotate

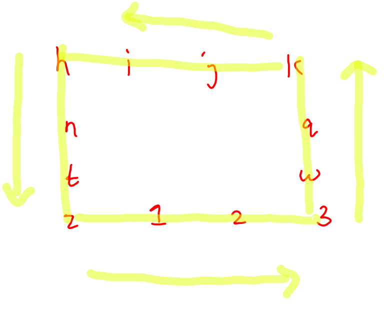


a	b	c	d	e	f
g	h	i	j	k	l
m	n	o	p	q	r
s	t	u	v	w	x
y	z	1	2	3	4
9	8	7	6	0	10

s = 2 , r = 3



a	b	c	d	e	f
g	k	q	w	3	l
m	j	o	p	2	r
s	i	u	v	1	x
y	h	n	t	z	4
9	8	7	6	0	10

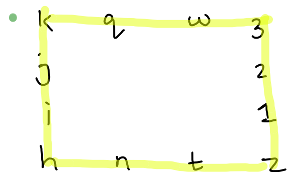


original shell

h n t z 1 2 3 w q k j i

⇓

k j i h n t z 1 2 3 w q



rotated shell

(i) 's' shell to 1d array.

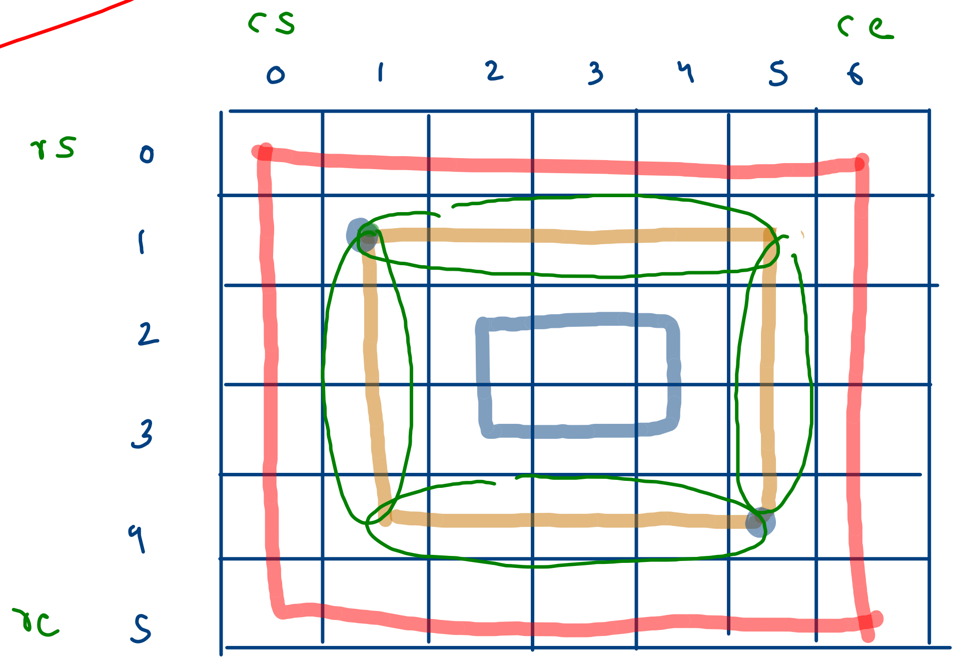
(ii) rotated this 1d anay by 'r'.

(iii) fill 's' shell using this rotated array.

Fill 1d array

$n = 6$

$m = 7$



$rs = s-1$

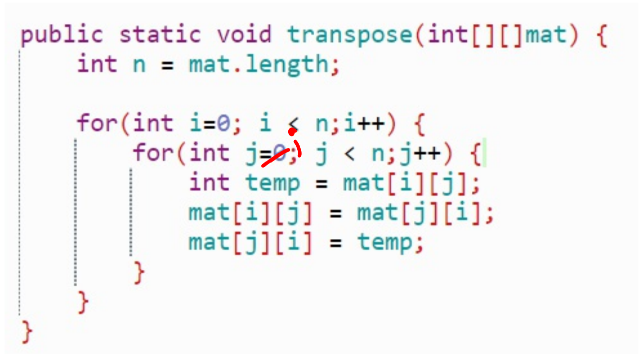
$cs = s-1$

$re = n-s$

$ce = m-s$

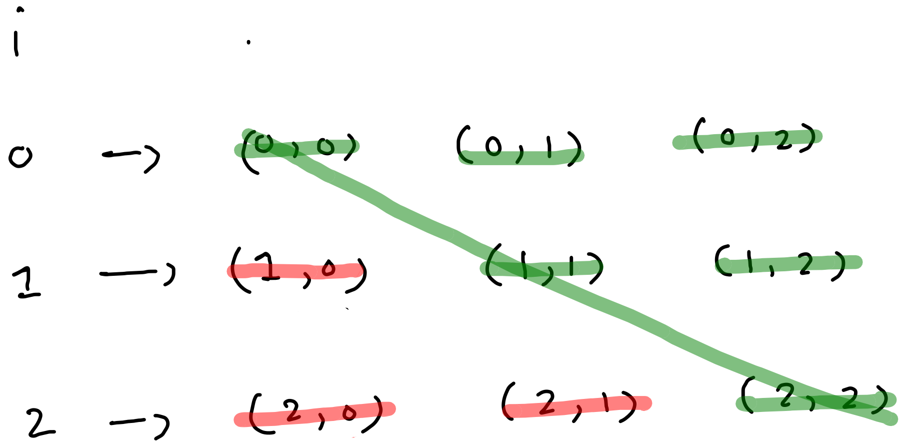
$count = 2 * (re - rs + 1) + 2 * ((e - (s+1)) - 4$

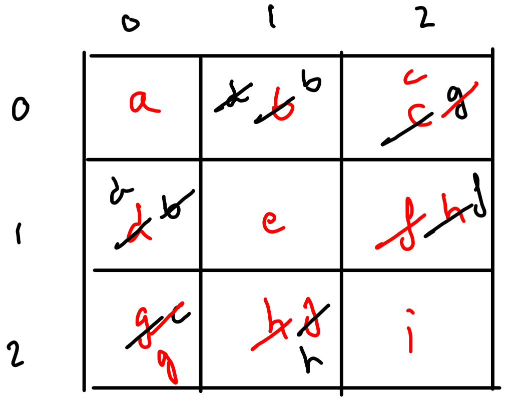
$$\begin{tikzcd}
\begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array} \arrow[rr, "90^\circ \text{ deg}", "\text{clockwise}"'] \arrow[dr, "\text{transpose}"] & & \begin{array}{|c|c|c|c|}\hline m & i & e & a \\ \hline n & j & f & b \\ \hline o & k & g & c \\ \hline p & l & h & d \\ \hline \end{array} \\
& \begin{array}{|c|c|c|c|}\hline a & e & i & m \\ \hline b & f & j & n \\ \hline c & g & k & o \\ \hline d & h & l & p \\ \hline \end{array} \arrow[ur, bend right, "\text{column reversal}"'] &
\end{tikzcd}$$

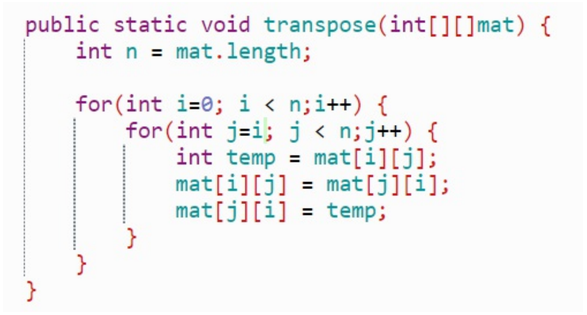


```
public static void transpose(int[][]mat) {
    int n = mat.length;

    for(int i=0; i < n;i++) {
        for(int j=~~0~~i; j < n;j++) {
            int temp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = temp;
        }
    }
}
```







```java
public static void transpose(int[][]mat) {
    int n = mat.length;

    for(int i=0; i < n;i++) {
        for(int j=i; j < n;j++) {
            int temp = mat[i][j];
            mat[i][j] = mat[j][i];
            mat[j][i] = temp;
        }
    }
}
```

$i = 2$

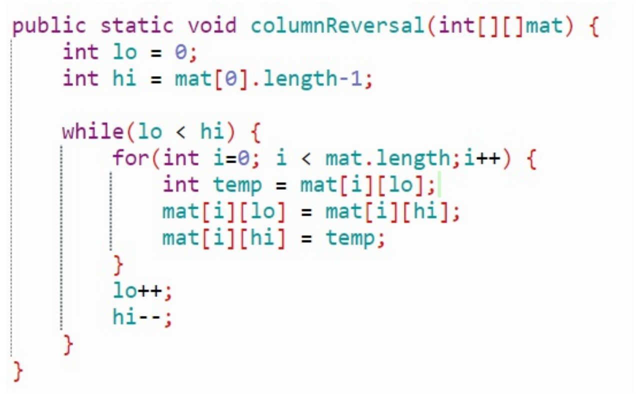
$j = 2$

$$\begin{array}{c|c|c|c}
 & 0 & 1 & 2 \\
\hline
0 & a & d\ \cancel{b} & \cancel{c}\ g \\
\hline
1 & \cancel{d}\ b & e & \cancel{f}\ h \\
\hline
2 & \cancel{g}\ c & \cancel{h}\ j & i
\end{array}$$

	~~lo~~ hi	~~hi~~ lo	
0	1	2	3
~~a~~ m	~~e~~ i	~~i~~ e	~~m~~ a
~~b~~ n	~~f~~ j	~~j~~ f	~~n~~ b
~~c~~ o	~~g~~ k	~~k~~ g	~~o~~ c
~~d~~ p	~~h~~ l	~~l~~ h	~~p~~ d

for [mat [i] [lo],

mat [i] [hi]



```
public static void columnReversal(int[][]mat) {
    int lo = 0;
    int hi = mat[0].length-1;

    while(lo < hi) {
        for(int i=0; i < mat.length;i++) {
            int temp = mat[i][lo];
            mat[i][lo] = mat[i][hi];
            mat[i][hi] = temp;
        }
        lo++;
        hi--;
    }
}
```

		lo	hi	
	0	1	2	3
0	~~a~~ m	~~e~~ i	~~i~~ e	~~m~~ a
1	~~b~~ n	~~f~~ j	~~j~~ f	~~n~~ b
2	~~c~~ o	~~g~~ k	~~k~~ g	~~o~~ c
3	~~d~~ p	~~h~~ l	~~l~~ h	~~p~~ d