int[] b = → [1, 4, 5, 6, 8] — array.

→ 1D horizontal

→ iterating col

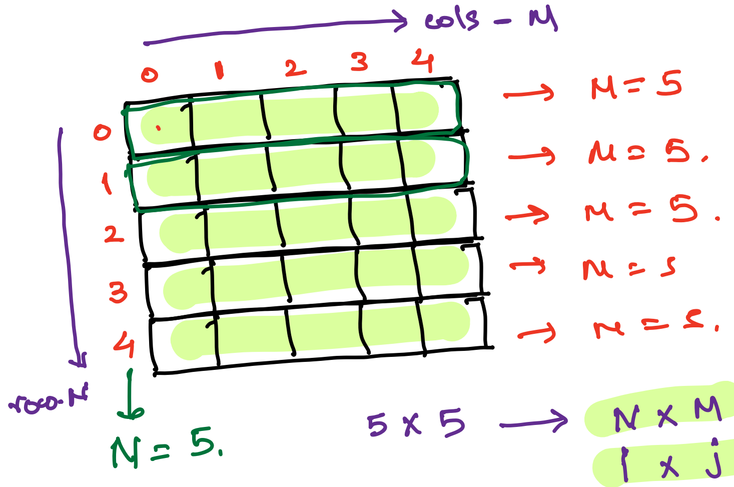
int [] a = new int[];

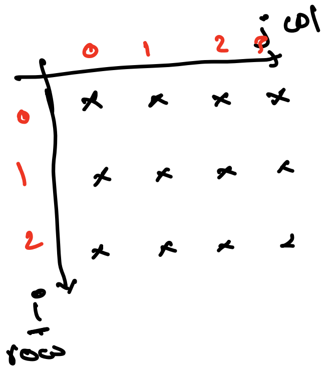
a1 → [1 , 4 , 5 , 6 , 8] → row1

a2 → [10 , 20 , 30 , 40 , 50] — row 2

a3 → [1, 3, 5, 7, 9] — row3 .

mat [a1, a2, a3] → 2D array





5 x 5 → N x M

i x j

row x col

Matrix
↓
2D Array

integer → int num = 5;

1D array → int[] arr = new int[N]

Syntax

2D array → int [][] mat = new int[N] [M];

5 x 5

user defined

row

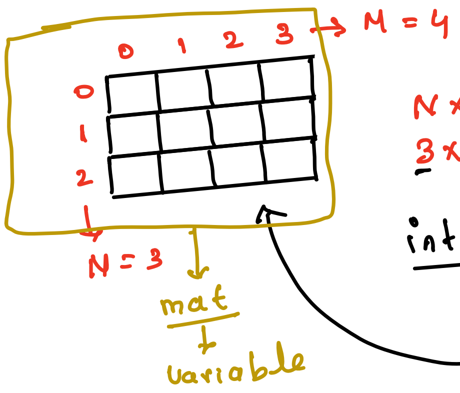
col

N → row size

M → col size

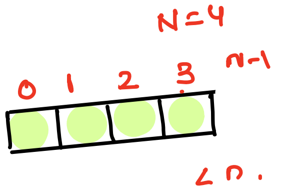
Q → mat 3 × 4

3 rows
4 colums

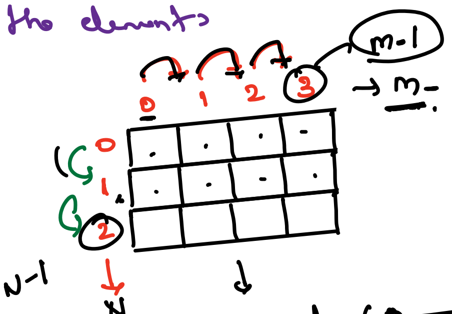


N × M
3 × 4

int[][] mat = new int[3][4]

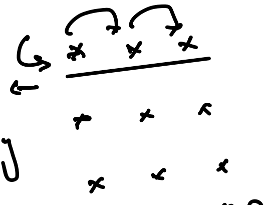


→ Access the elements



access

nested forloop



for(0 —— N-1)
  for (0 → m-1)

for(i —— n-1) → rows
  for (j —— m-1) → cols

Q) Take N and M. after that create N × M matrix and put the elements int matrix of N×M. And finally print the matrix.

1D
for (i=0 → N)
  arr[i] = sc.nextInt()

```
int N = sc.nextInt();
int M = sc.nextInt();
int[][] mat = new int[N][M];  N×M
row ← for(int i=0; i<N; i++){
col   ← for(int j=0; j<M; j++ {
            mat[i][j] = sc.nextInt();
      }
}
```

2 3 | 10 20 30 40 12 16

	0	1	2
0	10	20	30
1	40	12	16

N×m

sc.nextInt();

i	i<N (2)	j	j<M (3)	mat[i][j]=val	j++	i++
0	T	0	T	mat[0][0]=10	1	
		1	T	mat[0][1]=20	2	
		2	T	mat[0][2]=30	3	
		3	F	out of j loop →		1
1	T	0	T	mat[1][0]=40	1	
		1	T	mat[1][1]=12	2	
		2	T	mat[1][2]=16	3	
		3	F	Come out of j loop →		2
2	F			Come out of i loop →		break

```
// Printing row by row
for(int i=0; i<N; i++){
    for(int j=0; j<M; j++){
        sop(mat[i][j]);
    }
    sopln();
}
```

o/p single line

Print the matrix col by col

$$\begin{array}{c|ccc} & 0 & 1 & 2 \to j \text{ col} \\ \hline 0 & 10 & 20 & 30 \\ 1 & 40 & 12 & 16 \end{array}$$

i = row ✓

o/p
10 , 40
20 12
30 16

row → col

i → j → 1st
j → i → 2nd

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

$$\begin{array}{c|ccc} & 0 & 1 & 2 \to j \text{ col} \\ \hline 0 & 10 & 20 & 30 \\ 1 & 40 & 12 & 16 \end{array}$$

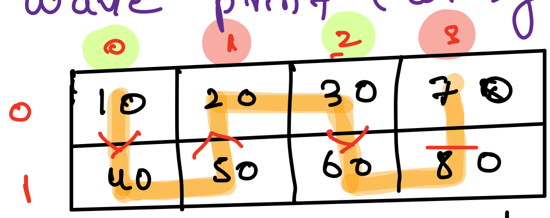
i = row ✓

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

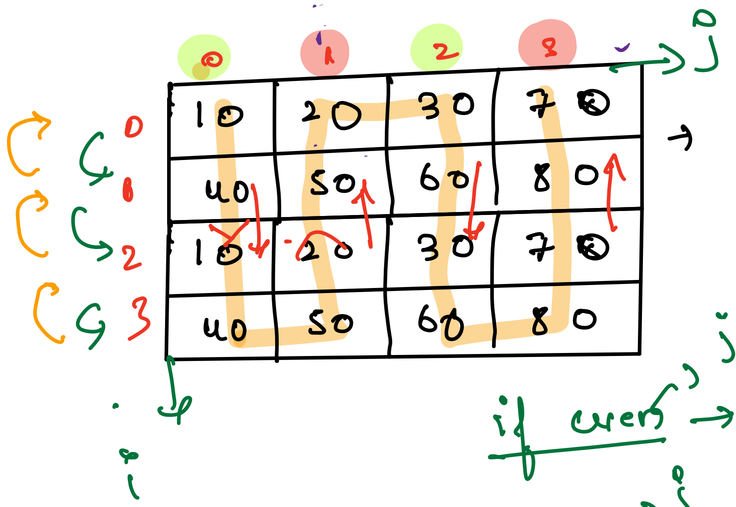
10_40
20_12.
30_16_
_

j	j<m (3)	i	i<N (2)	mat[i][j]	i++	j++
0	T	0	T	mat[0][0]	1	
		1	T	mat[1][0]	2	
		2	F	come out of i loop → sopln()		1
1	T	0	T	mat[0][1]	1	
		1	T	mat[1][1]	2	
		2	f	iloop → sopln		2
2	T	0	T	mat[0][2]	1	
		1	T	mat[1][2]	2	
		2	f	illoop → sopln		3
3	f			→ come out of j loop		

Q wave print (col by col)



↳ 10 40 50 20 30 60 80 70



→ 10 40 10 40 50 20 50 20
30 60 30 60 80 70 80 70

if even j → top to bottom.
i 0 → N-1

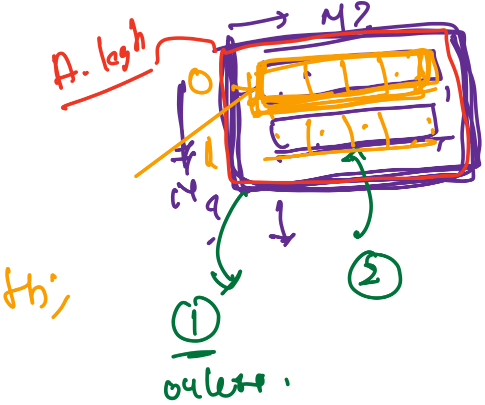
if odd j → bottom to top
i N-1 → 0

inc 1 → i++
dec 1 → i--
inc n → i+=n
$\frac{n}{2}$ i+=2

```
for(int j=0; j<m ;j++){
    if(j%2==0){
        for(int i=0; i<n; i++){
            sop(mat[i][j]);
        }
    }
    else{
        for(int i=N-1; i>=0; i--){
            sop(mat[i][j]);
        }
    }
}
```

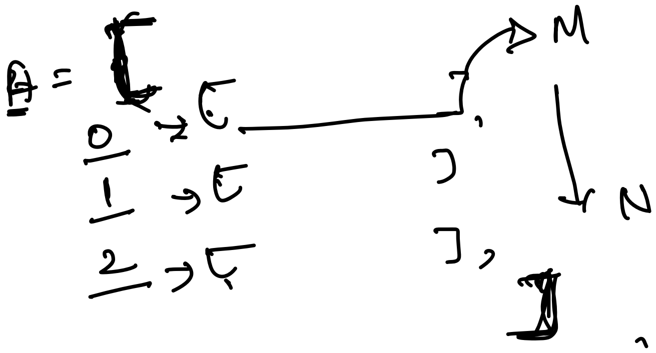
public static int[][] solve(int[][] A) {

returning 2Darray



N = A.length;

M = A[0].length;



cols = M

A.length.   A[0].length.

Doubt

5 → N

[2, 3, 1, 4, 2] →

1	2	3	4	5
2	3	~~1~~ 5	4	2

③ — position x

5 → value — y

a[3]=5.

1	2	3	4	5	6
2	3	5	1	4	2

0 1 2 3 4

[0, 2, 4, 1, 3]

for (0 → N-1)

i=0 arr[i] = arr[(arr[i]+3)%. arr.length]

arr[0] = arr[3]

i=1 arr[1] = arr[0]

i=2 arr[2] = arr(2)

[1 , 2 , 4 , 1 , 3]

→

x = 2

ls = [5 , 2 , 3 , 4 , 5] 4

[1 , 3 , 5 , 7] -1

2x1 = 2

A[i] % x == 0

```
{1, 5, 5, 5, 5, 1}

max = 1;
index = 0;
 for(int i = 1; i < ...; i++) {
      if (myList[i] > max) {
        ↳ max = myList[i];
          index = i;
      }
 }
```

5 > 1

max =

https://www.interviewbit.com/snippet/a82494b90e6ec85968b3/