Introduction to 2D Arrays

A 2D-array is an **array of 1D arrays**, referred by a single name, which is used to store a **sequence** of values of the **same type** and can be visualized as a **matrix**.

int arr[][]

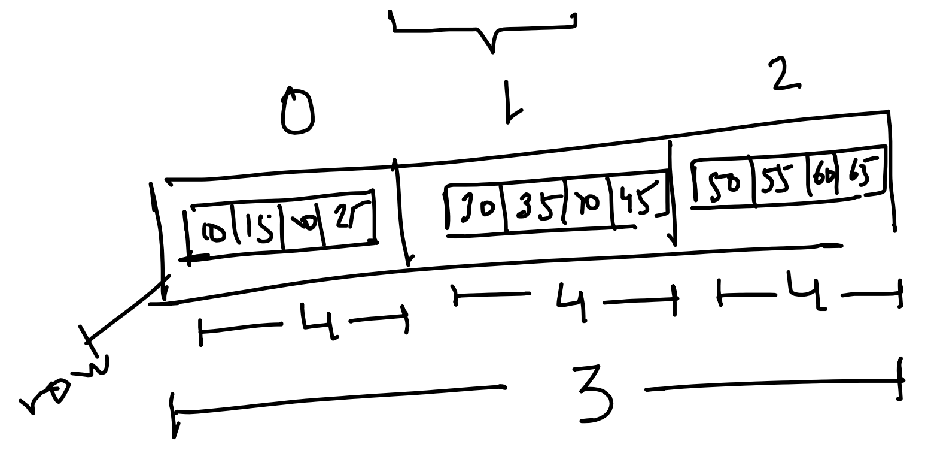
	0	1	2	3
0	**10**	**15**	**20**	**25**
1	**30**	**35**	**40**	**45**
2	**50**	**55**	**60**	**65**

3 x 4

$m \times n$ → #rows, → #coln

0 to m-1

0 to n-1



How to **declare** a 2D-array in C++ ?

```
// syntax for a 2D-array declaration

type name[rows][cols];
```

4B

int arr[3][4];

12

4B×12

= 48B (contiguous)

element access is O(1) op.

48B

A 2D-array is allocated a **contiguous** block of memory to store its elements.

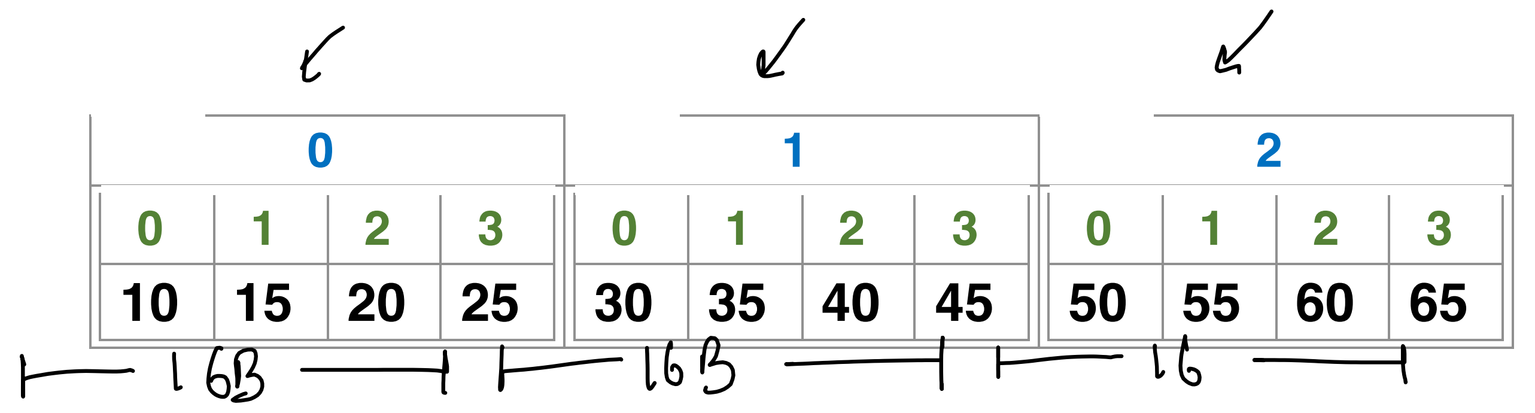
	0	1	2	3
0	**10**	**15**	**20**	**25**
1	**30**	**35**	**40**	**45**
2	**50**	**55**	**60**	**65**

3×4

This allocation of contiguous memory is done either in the **row-wise** manner a.k.a row-major order or in a **column-wise** manner a.k.a column-major order.

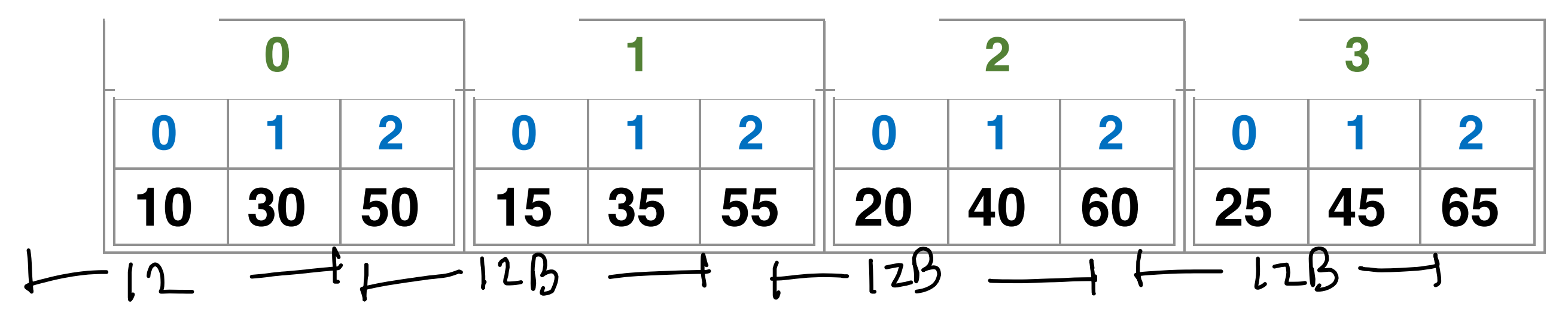
- Row-Major Order (default)

#



	0	1	2	3
0	10	15	20	25
1	30	35	40	45
2	50	55	60	65

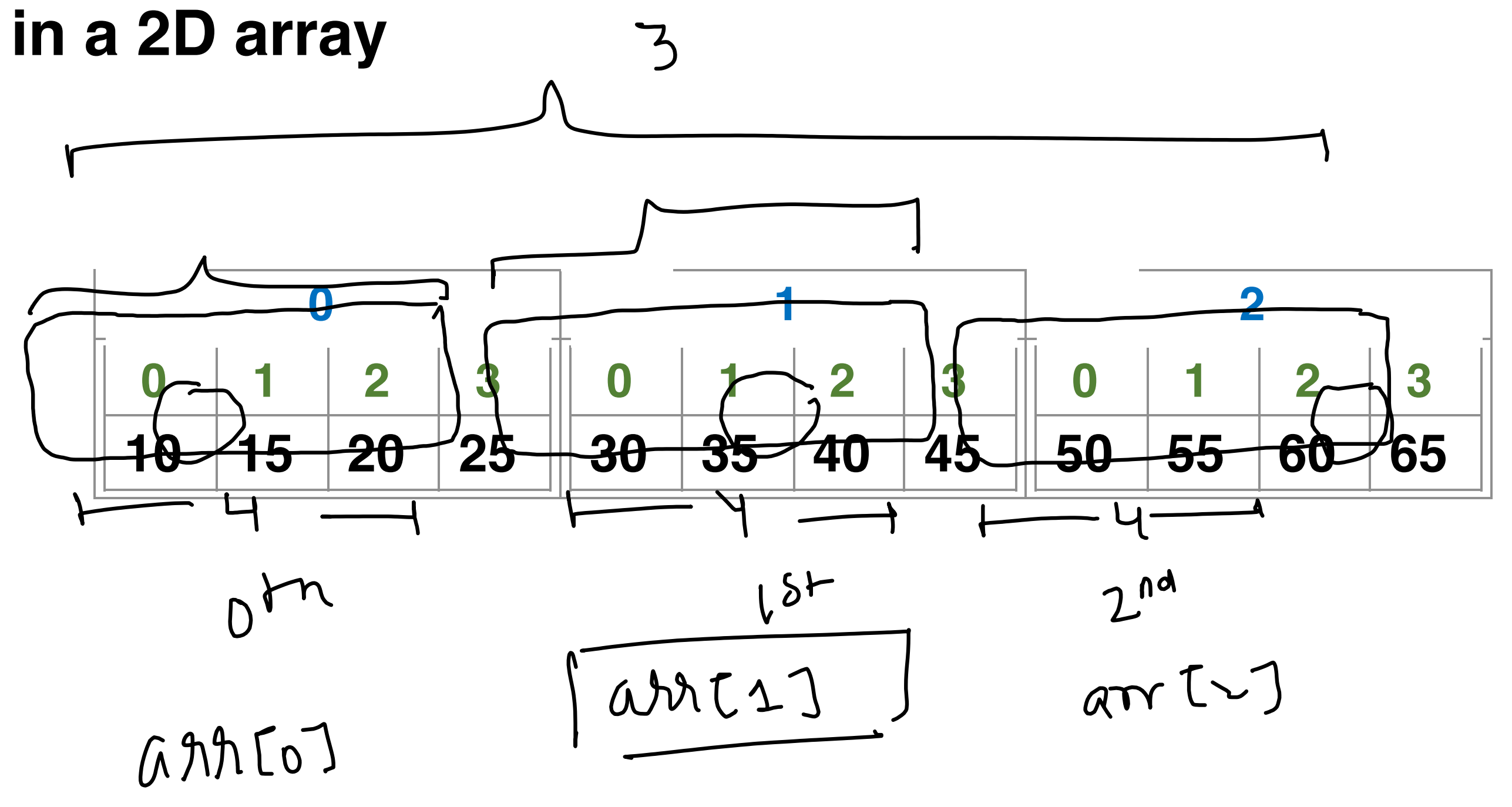
- Column-Major Order



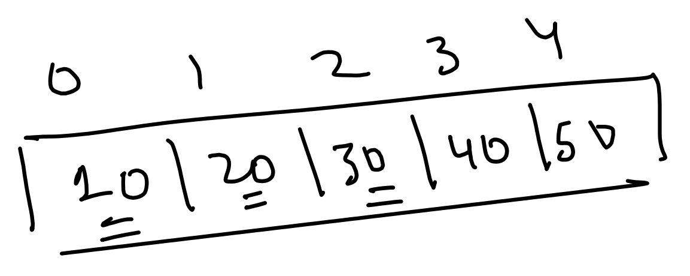
Accessing an element in a 2D array

	0	1	2	3
0	10	15	20	25
1	30	35	40	45
2	50	55	60	65

3x4



int arr[] = {10, 20, 30, 40, 50}



arr[0]

arr[i]

arr[2]

arr[i]

$0 \le i \le n-1$

arr[i] #rows

$0 \le i \le m-1$

arr[0][1] → 15

arr[1][2] → 40

arr[2][3] → 65

Since 2D-arrays are allocated memory in a **contiguous** manner, accessing an element in a 2D-array is a **constant time** i.e. **O (1)** operation.

$\boxed{arr[i][j]}$

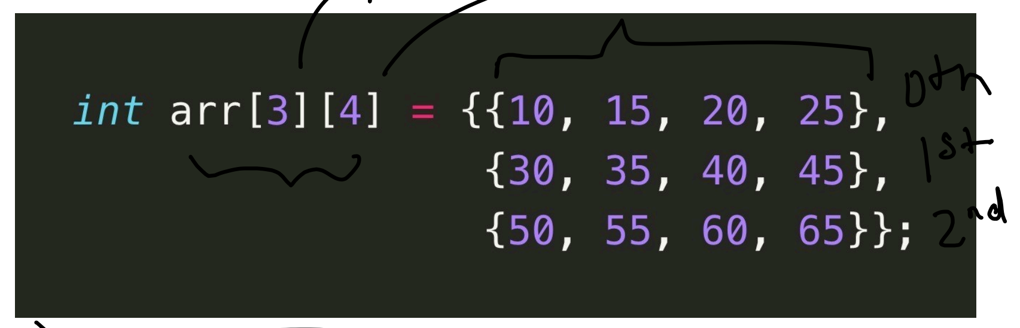
$0 \le j \le n-1$ → #coln

2D-Array Initialization

By default, when we declare a 2D-array in the local-scope, it contains **garbage** value.

optional

#coln : Mandatory



```
int arr[3][4] = {{10, 15, 20, 25},
                 {30, 35, 40, 45},
                 {50, 55, 60, 65}};
```

0th
1st
2nd

arr

	0	1	2	3
0	10	15	20	25
1	30	35	40	45
2	50	55	60	65

During the initialization of a 2D-array, specifying the no. of rows is **optional**.

While initializing a row in a 2D-array, the size of the **initializer list** must not exceed the no. of columns in the 2D-array.

```
int arr[][2] = {{10, 20, 30},
                {40, 50}};
```

error

What if the size of the row initializer list is less than the no. of columns ?

```
int arr[][3] = {{10, 20, 30},
                {40, 50}};
```

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 10 & 20 & 30 \\ 1 & 40 & 50 & \underset{\text{default}}{0} \end{array}$$

Wave Print

Given an integer matrix of dimensions **m x n**, write a program that prints the matrix in the **wave** form.

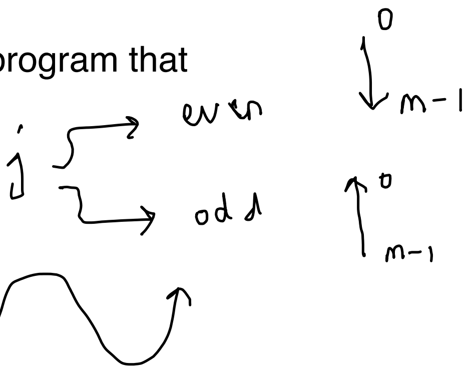
Example

Input

	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

Output

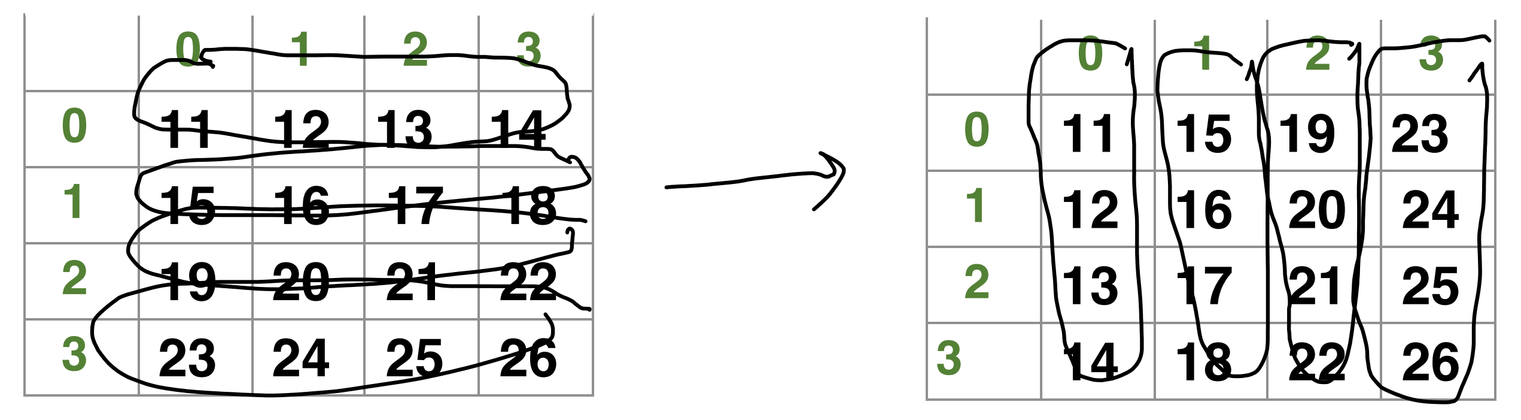
10	40	70	80	50	20	30	60	90



Transpose a Square Matrix

$m=n$

Given an square matrix of integers of dimensions **m x n**, write a program to **transpose** it.



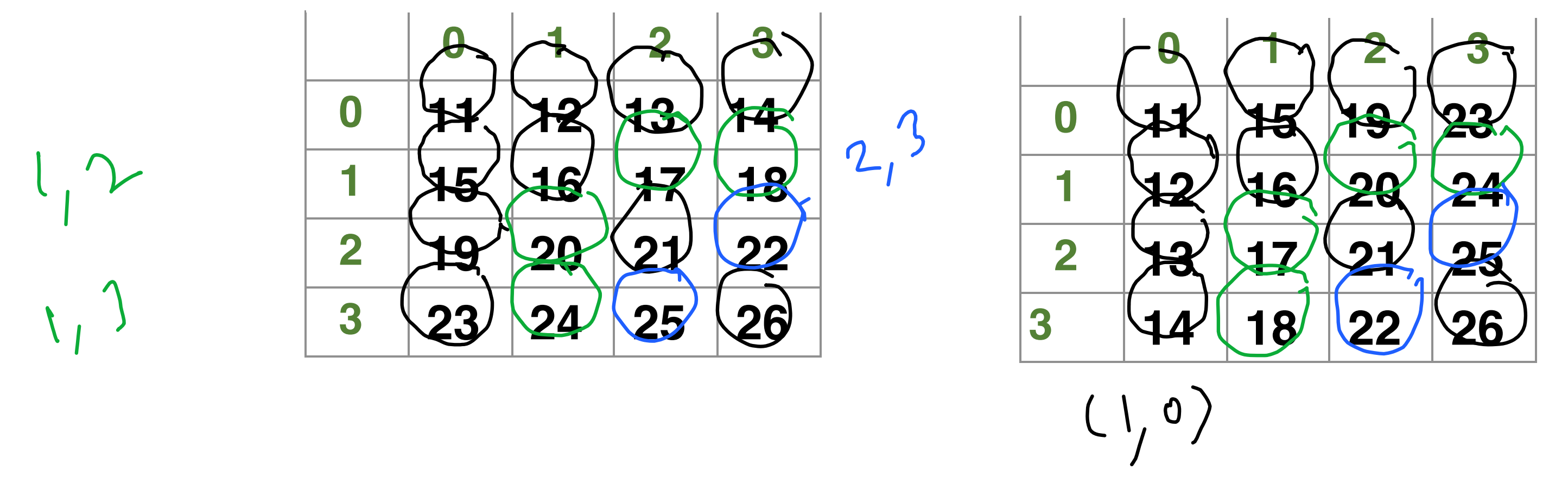
	0	1	2	3
0	11	12	13	14
1	15	16	17	18
2	19	20	21	22
3	23	24	25	26

Input

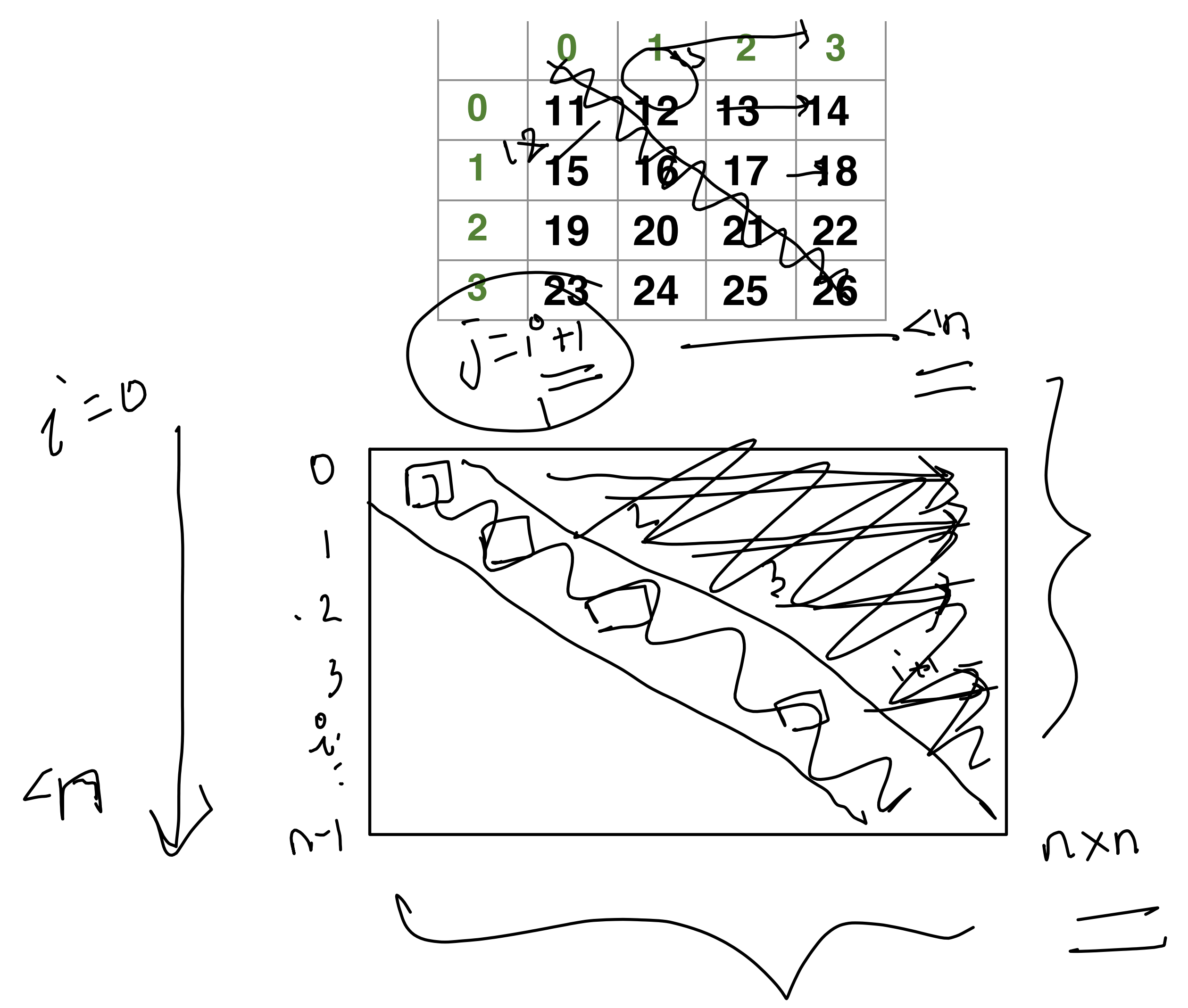
	0	1	2	3
0	11	15	19	23
1	12	16	20	24
2	13	17	21	25
3	14	18	22	26

Output

(0,1) (0,2) (0,3)



$$(i,j) \longrightarrow (j,i)$$



$$\frac{n(n-1)}{2} \approx \frac{n^2}{2}$$