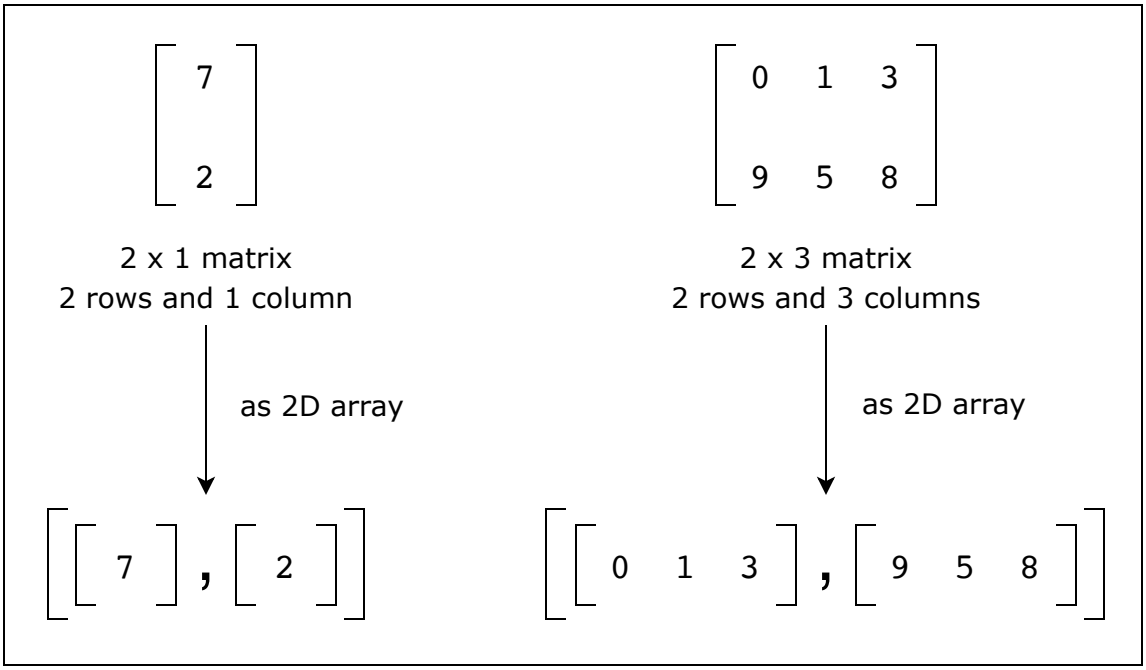
Introduction to Matrices

Let’s go over the Matrices pattern, its real-world applications, and some problems we can solve with it.

We'll cover the following...

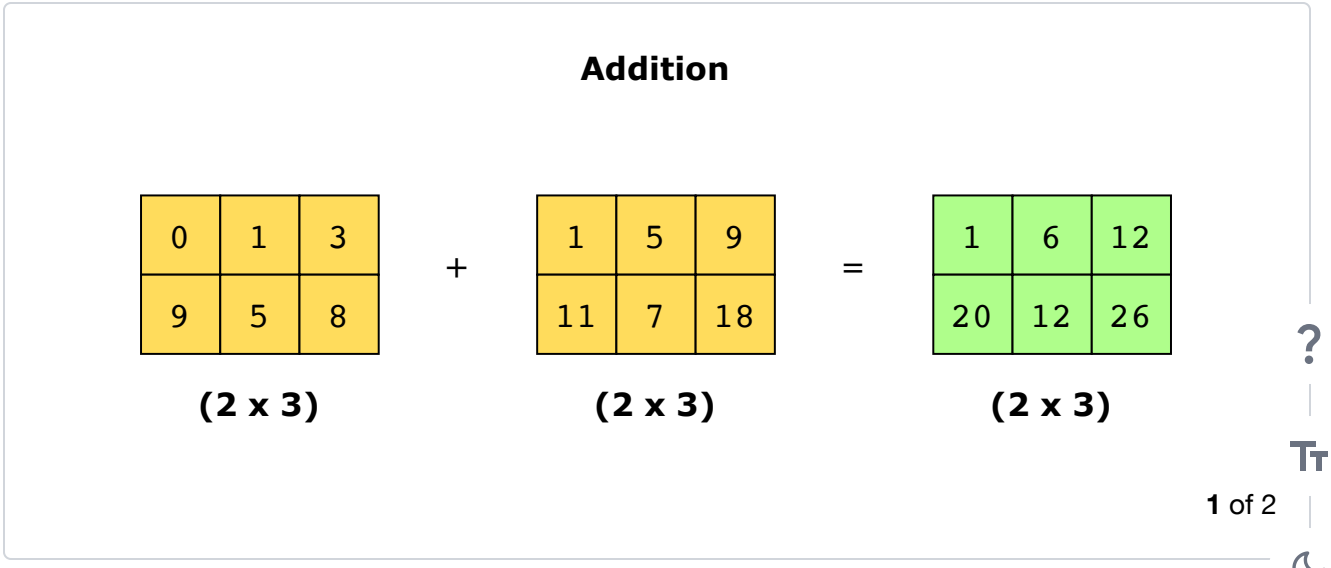
About the pattern

A **matrix** is a group of numbers arranged in rows and columns in a rectangular pattern. In computer science, matrixes are represented by 2D arrays with dimensions $m \times n$, where m is the number of rows, and n is the number of columns. Each element in the matrix can be accessed using the array indexes. The first index represents the row, and the second index represents the column. For example, $matrix[i][j]$ is an element from the i^{th} row and the j^{th} column of the matrix.



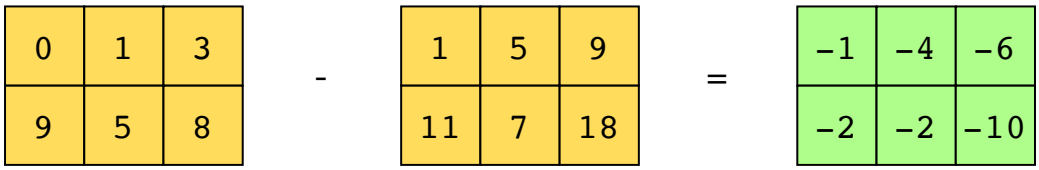
Matrix transformations are operations performed on matrices that result in a new matrix. There are several types of matrix transformations commonly used in mathematics and computer science. Here is a list of some of the most fundamental matrix transformations:

1. **Addition and subtraction:** Element-wise addition or subtraction of two matrices of the same dimensions. Here, the dimensions of both matrices must be identical.



1 of 2

Subtraction

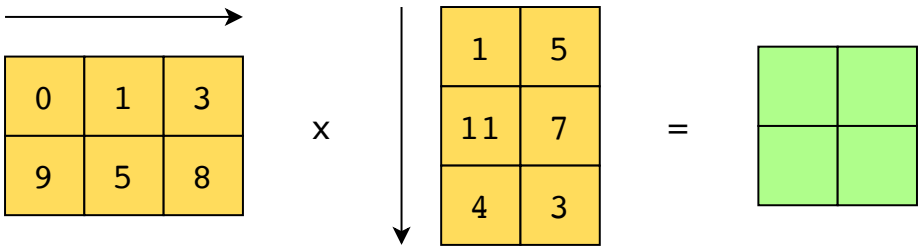


(2 x 3) **(2 x 3)** **(2 x 3)**

2 of 2

2. **Multiplication:** We perform the dot product between the rows of the first matrix and the columns of the second matrix. The dot product between a row and a column is calculated by multiplying the corresponding elements of the row of the first matrix by the corresponding elements of the column of the second matrix and summing the results. Here, the number of columns in the first matrix must be equal to the number of rows in the second matrix.

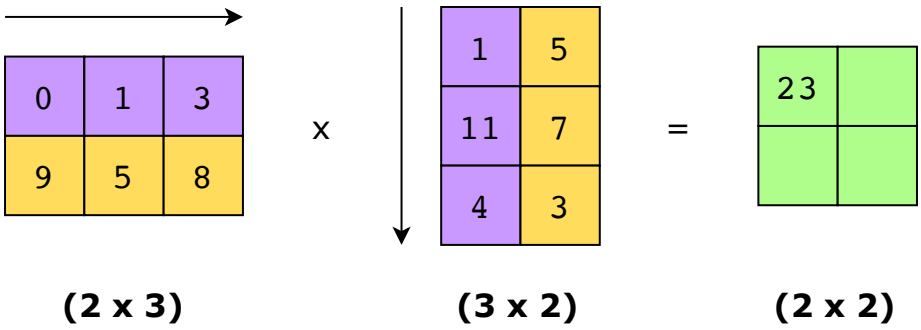
Product



(2 x 3) **(3 x 2)** **(2 x 2)**

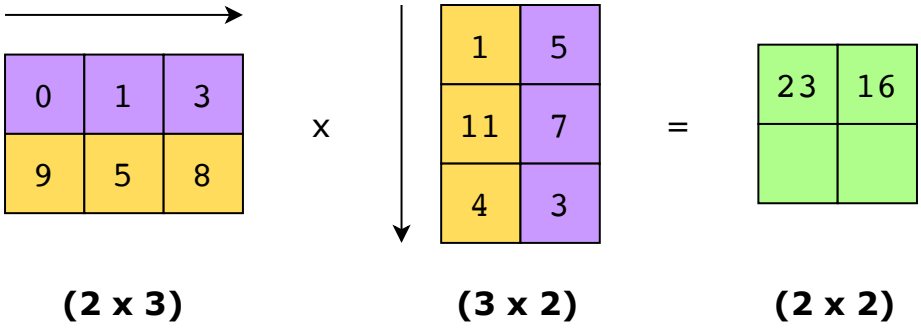
1 of 5

Product



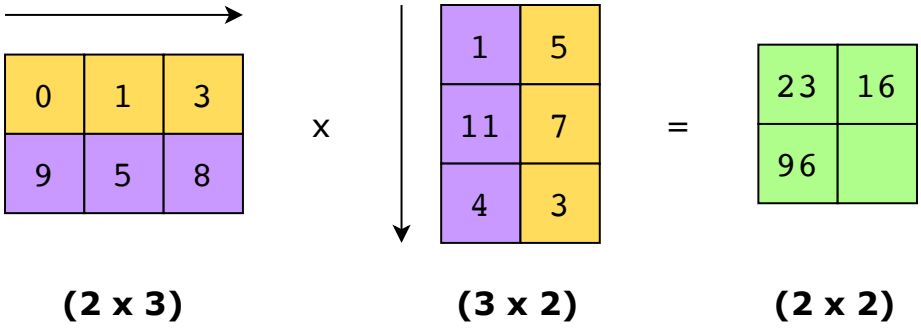
2 of 5

Product

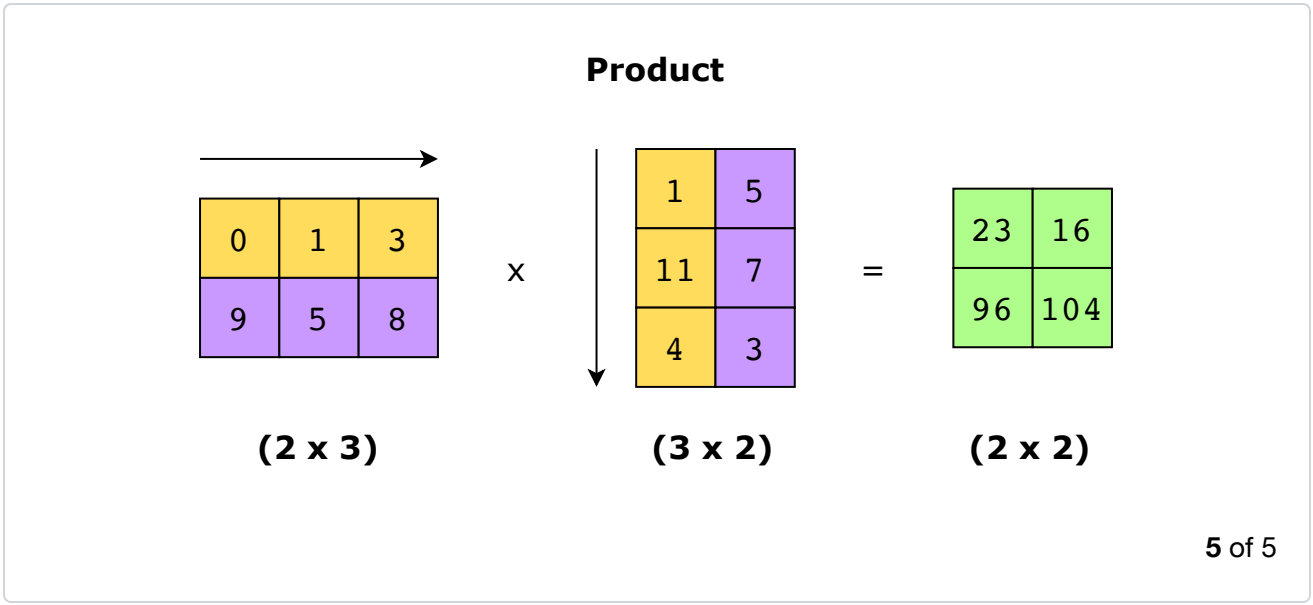


3 of 5

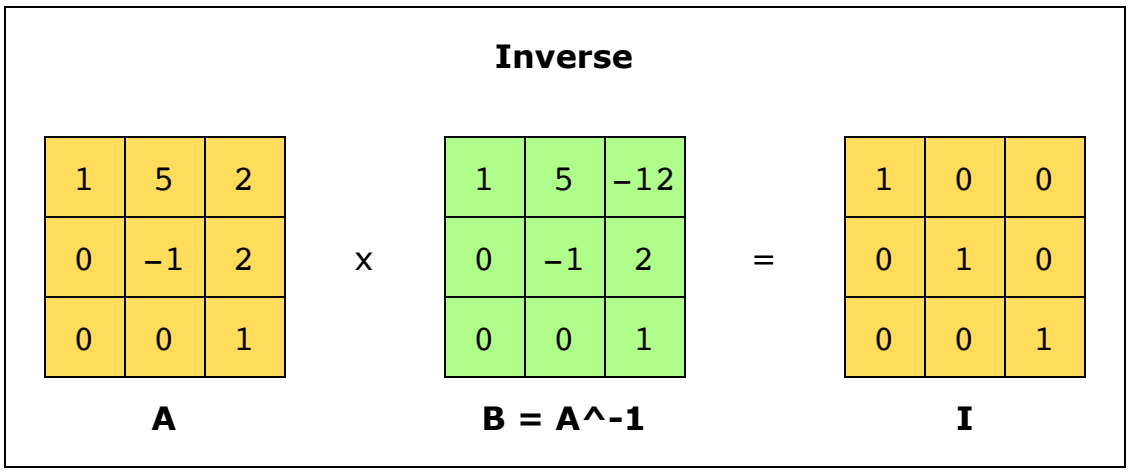
Product



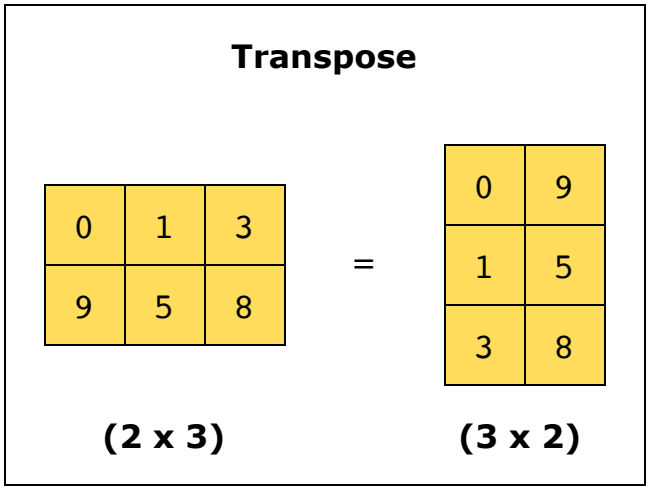
4 of 5



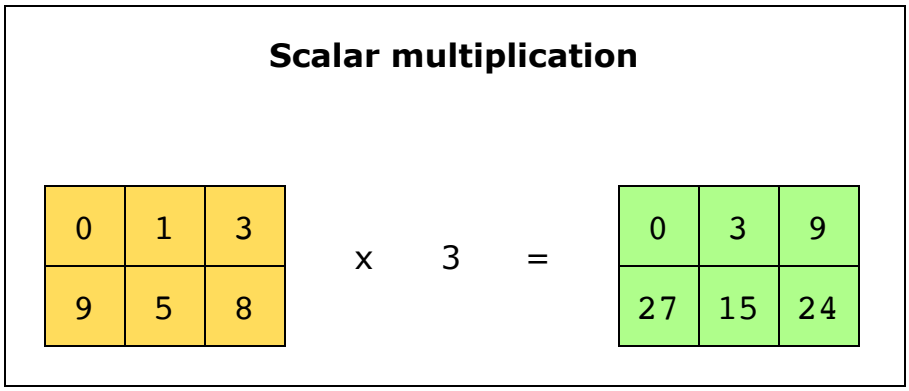
3. **Inverse:** For a square matrix A, if there exists another square matrix B such that $AB = BA = I$ (where I is the identity matrix), then B is called the inverse of A, denoted as A^{-1}.



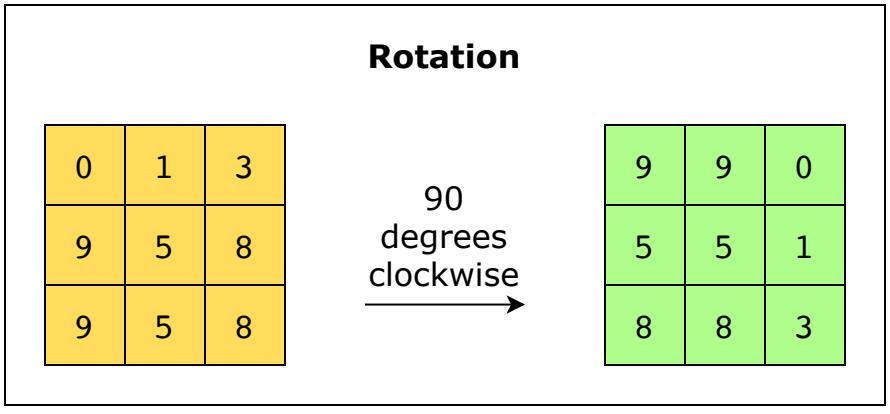
4. **Transpose:** Swapping the rows and columns of a matrix.



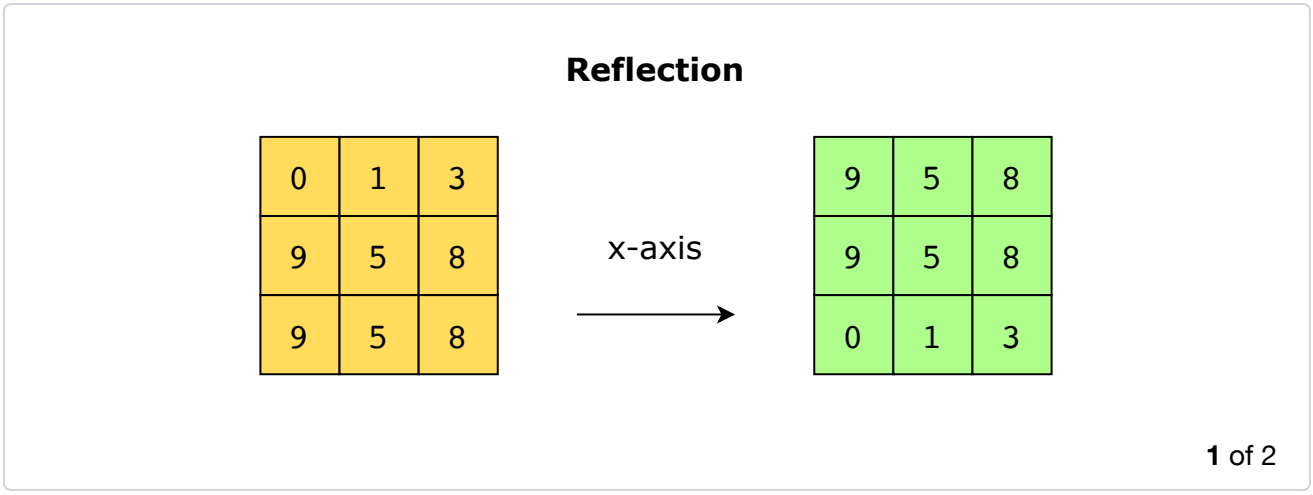
5. **Scalar multiplication:** Each element of the matrix is multiplied by a scalar value.



6. **Rotation:** The elements of a square matrix are rotated at an angle.

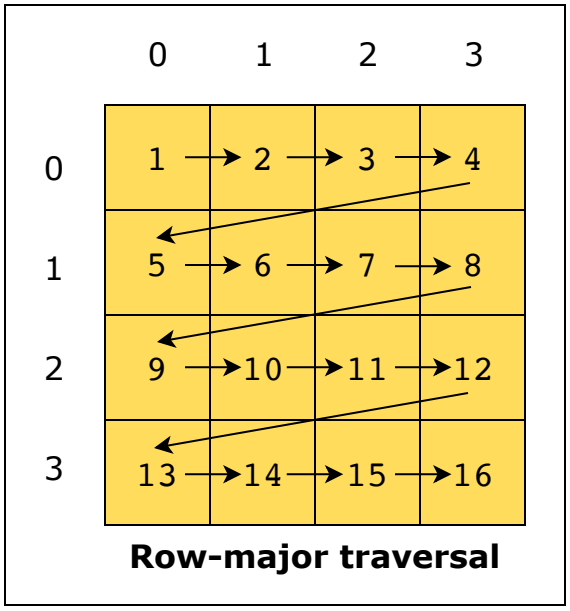


7. **Reflection:** The elements of a matrix are flipped over an axis.

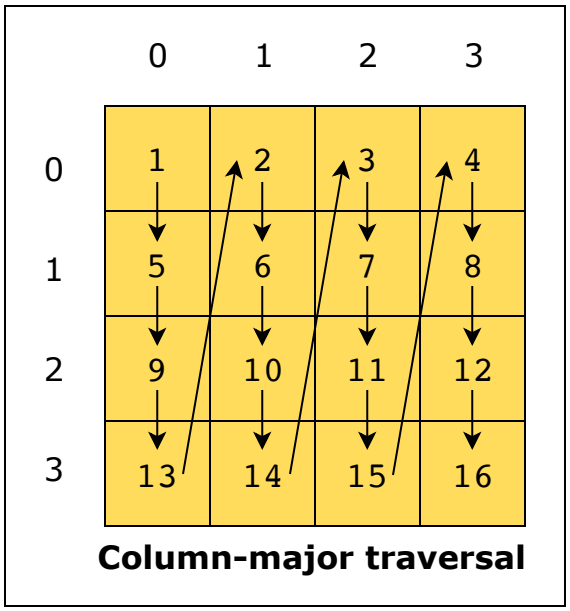


Matrix traversal refers to the process of systematically visiting each element in a matrix exactly once. This process is often used in computer science and mathematics for various applications such as searching, sorting, pathfinding, and data manipulation. There are several common methods for traversing a matrix:

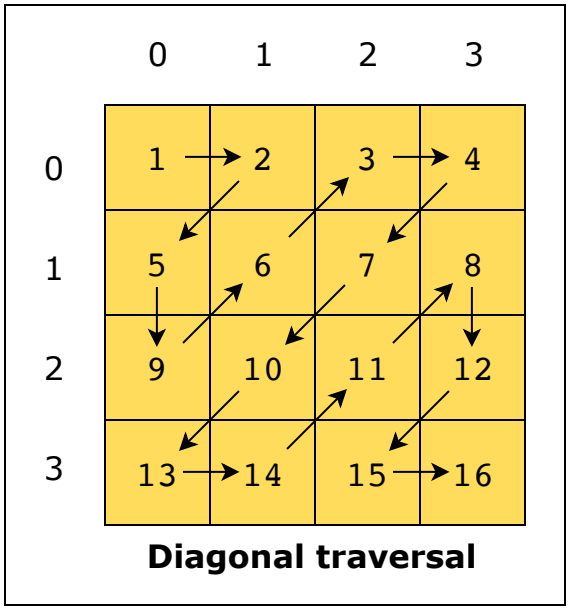
1. **Row-major traversal:** In this method, we traverse the matrix row by row, moving horizontally first and then vertically. For example, if we have a matrix with dimensions m rows and n columns, we would start at element $(0, 0)$, then move to $(0, 1), (0, 2), ..., (0, n - 1)$, then proceed to the next row and repeat until you reach $(M - 1, N - 1)$.



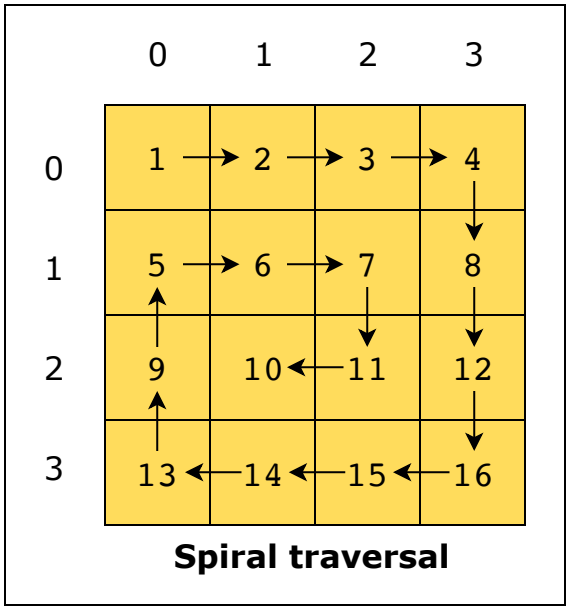
2. **Column-major traversal:** This is the opposite of row-major traversal. We traverse the matrix column by column, moving vertically first and then horizontally. We start at element $(0, 0)$, then move to $(1, 0), (2, 0), ..., (m - 1, 0)$, then proceed to the next column and repeat until we reach $(m - 1, n - 1)$.



3. **Diagonal traversal:** In this method, we traverse the matrix along its diagonal elements. There are different variations of diagonal traversal, such as traversing from the top-left corner to the bottom-right corner (main diagonal) or from the top-right corner to the bottom-left corner (secondary diagonal).



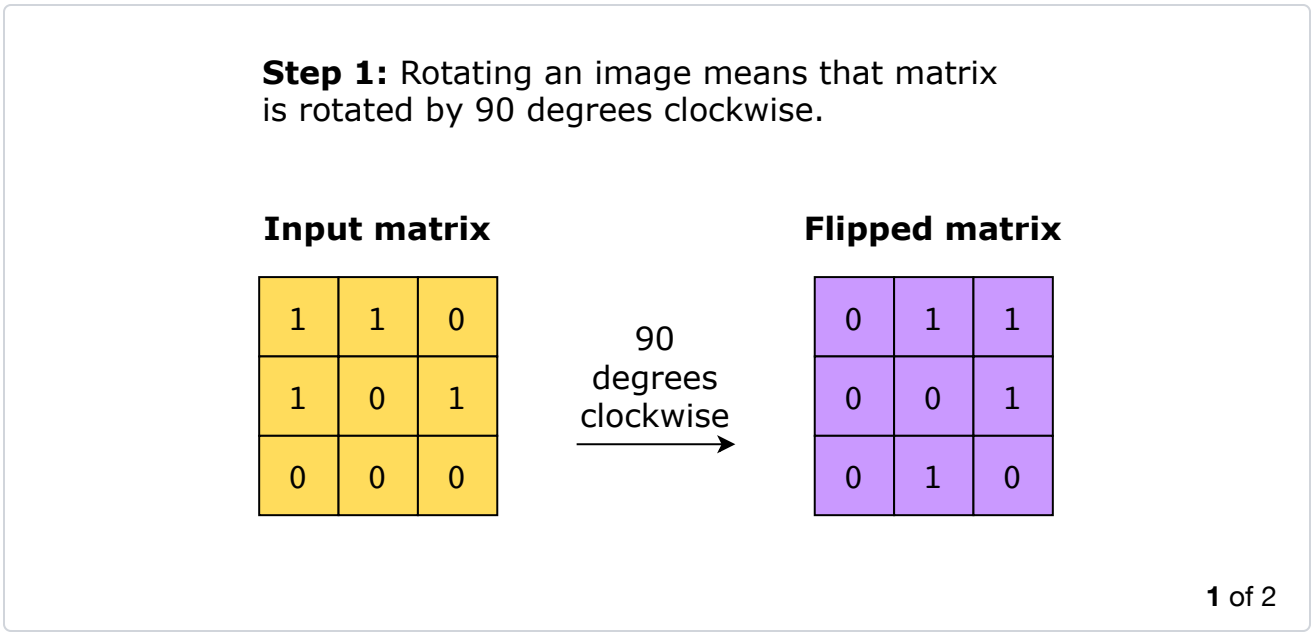
4. **Spiral traversal:** This method involves traversing the matrix in a spiral pattern, starting from the outermost elements and moving towards the center. The traversal direction alternates between right, down, left, and up until all elements are visited.



Examples

The following examples illustrate some problems that can be solved with these approaches:

1. **Rotate and invert an image:** Given an $n \times n$ binary image, rotate it clockwise, invert it,
   and return the final image.



2. **Toeplitz matrix:** Given an $m \times n$ matrix, return TRUE if the matrix is Toeplitz, otherwise, return FALSE. A matrix is Toeplitz if every descending diagonal from left to right (or basically every left diagonal) has the same elements.

We'll perform row-major traversal from the second row and second column of the matrix. For each element, **matrix[i][j]**, we'll check if it is equal to the element on its top left, i.e., **matrix[i - 1][j - 1]**. If it is, we continue the traversal. Otherwise, we return FALSE.

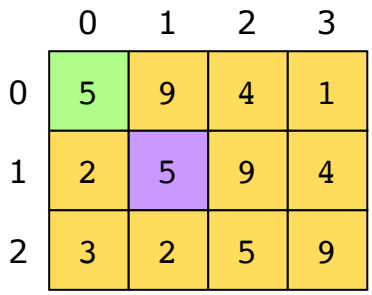
	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Value: matrix[1][1]

We compare the value at **matrix[0][0]**.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

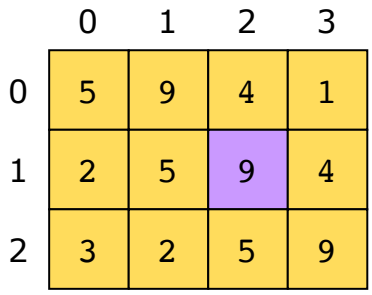
Because the values are identical, we continue the traversal.



3 of 14

Value: matrix[1][2]

We compare the value at **matrix[0][1]**.



4 of 14

Because the values are identical, we continue the traversal.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Value: matrix[1][3]

We compare the value at **matrix[0][2]**.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Because the values are identical, we continue the traversal.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Value: matrix[2][1]

We compare the value at **matrix[1][0]**.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Because the values are identical, we continue the traversal.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Value: matrix[2][2]

We compare the value at **matrix[1][1]**.

	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Because the values are identical, we continue the traversal.

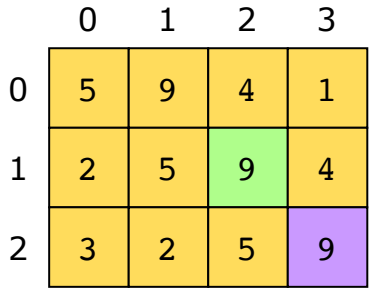
	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Value: matrix[2][3]

We compare the value at **matrix[1][2]**.

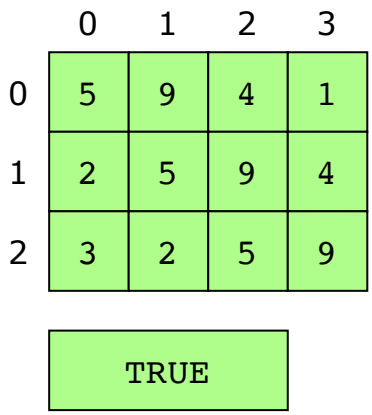
	0	1	2	3
0	5	9	4	1
1	2	5	9	4
2	3	2	5	9

Because the values are identical, we continue the traversal.



13 of 14

Because we have successfully traversed the entire matrix, we return TRUE, since all the left diagonals contain identical elements.



14 of 14

Does your problem match this pattern?

Yes, if the following condition is fulfilled:

- **2D array input:** The input data is given as a 2D array. However, some exceptions to this could be problems that have a 2D array as an input, but are solved with some other pattern, e.g., graphs, dynamic programming etc.

Real-world problems

Many problems in the real world share the matrix transformation pattern. Let's look at some examples:

- **Image processing:** Matrices are used to represent images, where each pixel's color values are stored in a matrix. Matrix transformations such as scaling, rotation, translation, and affine transformations are applied to manipulate images in graphics