Day 17

Prepared By Azan Imtiaz

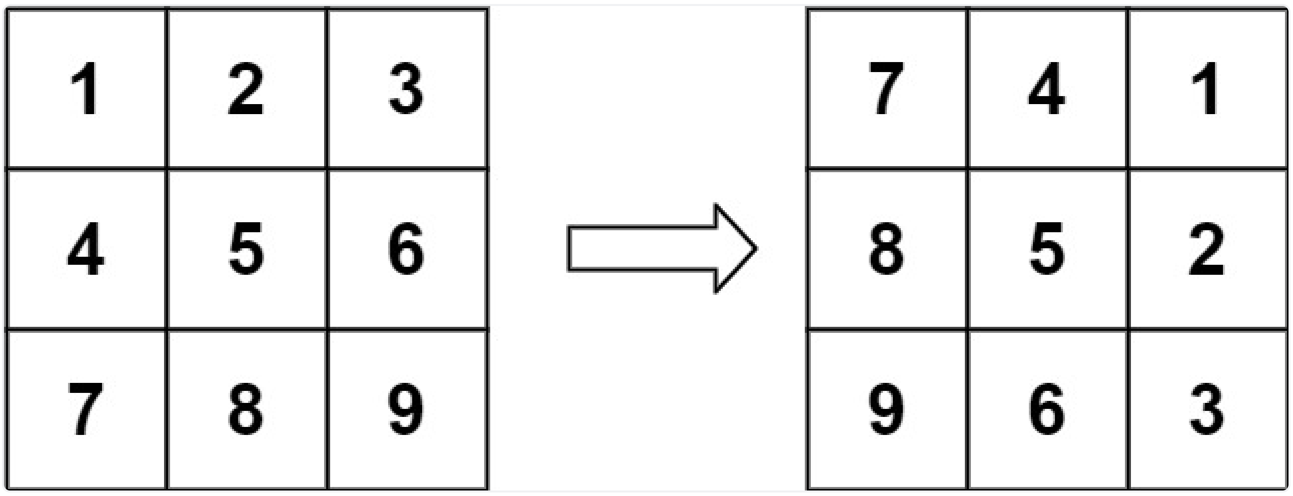
Rotate Matrix (Leatcode)

Description:

You are given an `n x n` 2D `matrix` representing an image, rotate the image by **90** degrees (clockwise).

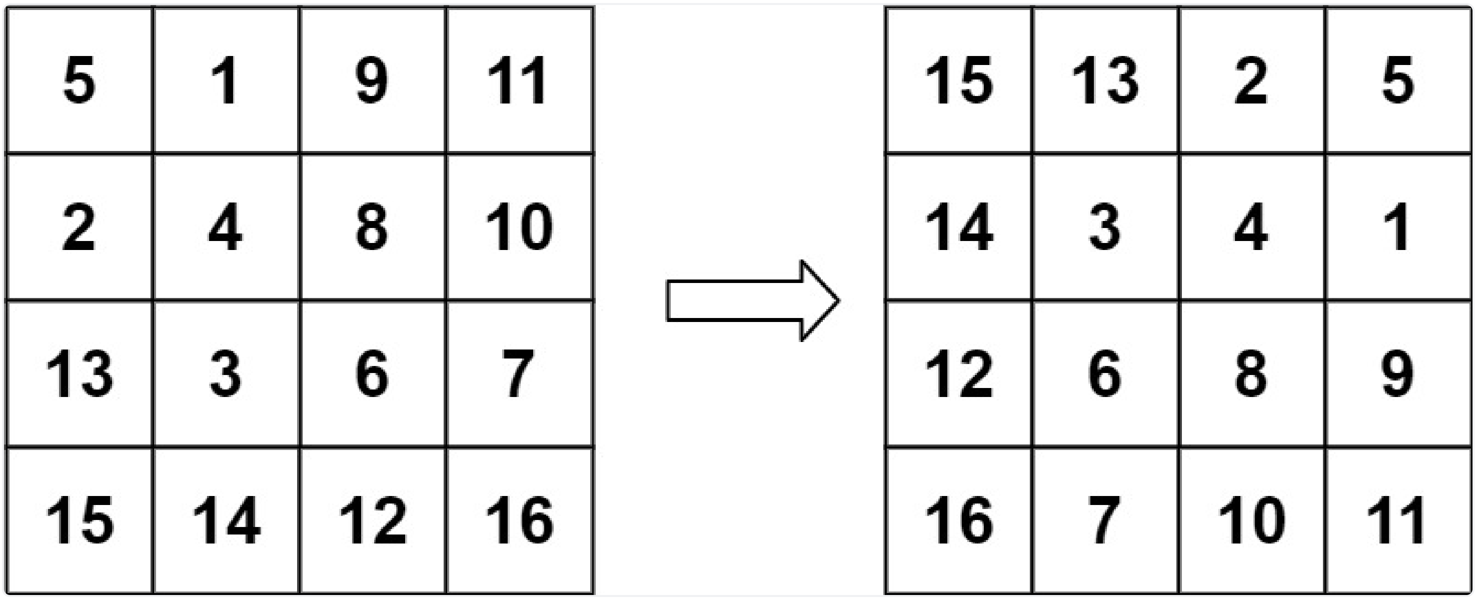
You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:



```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]
```

Example 2:



```
Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]
Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]
```

Constraints:

- `n == matrix.length == matrix[i].length`
- `1 <= n <= 20`
- `-1000 <= matrix[i][j] <= 1000`

Solution Intuition

The goal is to rotate the matrix by 90 degrees to the right. The trick is to perform this in place, meaning without using extra space for another matrix. We can achieve this by rotating four pixels at a time, one from each corner of the current layer of the matrix we are inspecting.

Approach

1. Loop through layers: Since we're rotating the matrix in layers (like an onion), we only need to go halfway through the matrix for rows (`(n+1)/2`) and halfway for columns (`n/2`).
2. Four-way swap: In each iteration, perform a four-way swap. This moves the pixels from one corner to the next, going clockwise.

Solution

```
class Solution {
    public void rotate(int[][] matrix) {
        int n = matrix.length;
        for (int i = 0; i < (n + 1) / 2; i++) {
            for (int j = 0; j < n / 2; j++) {
                int temp = matrix[n - 1 - j][i]; // Save bottom left
                matrix[n - 1 - j][i] = matrix[n - 1 - i][n - j - 1]; // Bottom left <-
Bottom right
                matrix[n - 1 - i][n - j - 1] = matrix[j][n - 1 - i]; // Bottom right <-
Top right
                matrix[j][n - 1 - i] = matrix[i][j]; // Top right <- Top left
                matrix[i][j] = temp; // Top left <- Saved bottom left
            }
        }
    }
}
```

Description of Solution

We start with the outermost layer and move inward. For each pixel, we save the bottom-left value in `temp`. Then we move the bottom-right pixel to the bottom-left position, the top-right pixel to the bottom-right position, the top-left pixel to the top-right position, and finally put the saved `temp` value (original bottom-left) into the top-left position. This effectively rotates the pixels 90 degrees clockwise within their current layer.

Dry Run on Test Cases

Test Case 1:

Matrix:

```
1 2
3 4
```

Iterations:

- Layer 1 (Outer Layer):
  - Swap index `[1][0]` with `[1][1]`, `[1][1]` with `[0][1]`, `[0][1]` with `[0][0]`, and `[0][0]` with saved `temp`.

- Result:

```
3 1
4 2
```

Test Case 2:

Matrix:

```
1 2 3
4 5 6
7 8 9
```

Iterations:

- Layer 1 (Outer Layer):
  - Swap index `[2][0]` with `[2][1]`, `[2][1]` with `[0][2]`, `[0][2]` with `[0][0]`, and `[0][0]` with saved `temp`.
  - Swap index `[2][1]` with `[2][2]`, `[2][2]` with `[1][2]`, `[1][2]` with `[0][1]`, and `[0][1]` with saved `temp`.
  - Result after Layer 1:

```
7 4 1
8 5 2
9 6 3
```

Test Case 3:

Matrix:

```
 5  1  9 11
 2  4  8 10
13  3  6  7
15 14 12 16
```

Iterations:

- Layer 1 (Outer Layer):

    - Swaps occur for indices `[3][0]` with `[3][3]`, `[3][3]` with `[0][3]`, `[0][3]` with `[0][0]`, and `[0][0]` with saved `temp`.
    - Continue swapping around the outer layer.
- Layer 2 (Inner Layer):
    - Swaps occur for indices `[2][1]` with `[2][2]`, `[2][2]` with `[1][2]`, `[1][2]` with `[1][1]`, and `[1][1]` with saved `temp`.
    - Result after all layers:

```
15 13  2  5
14  3  4  1
12  6  8  9
16  7 10 11
```