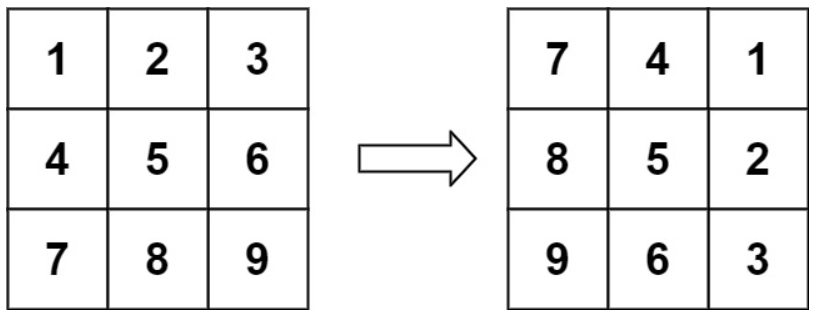
<https://leetcode.com/problems/rotate-image/>

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](https://en.wikipedia.org/wiki/In-place_algorithm)**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

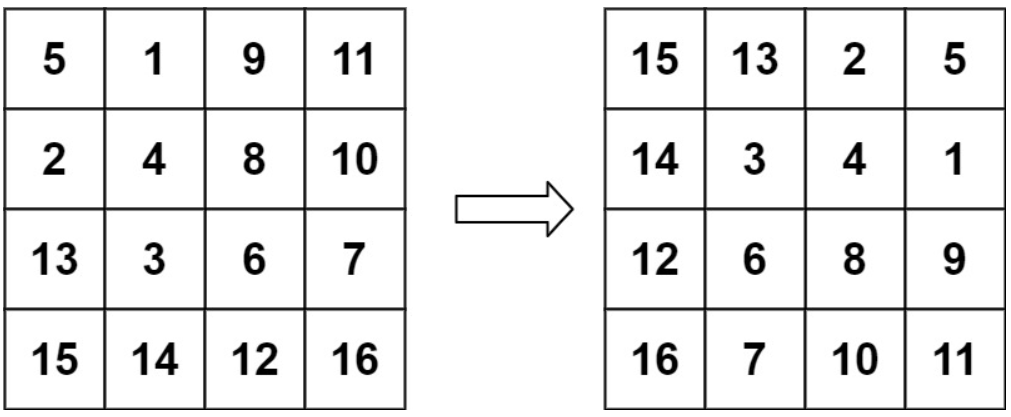
**Example 1:**



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

**Attempt 1: 2023-06-26**

**Wrong Solution: The swap function not actually work because matrix[i][j] is an object, but not change at all since the swap function only swap passed in integer value, not swap the original matrix cell**

class Solution {

public void rotate(int[][] matrix) {

int n = matrix.length;

for(int i = 0; i < n; i++) {

for(int j = i; j < n; j++) {

swap(matrix[i][j], matrix[j][i]);

}

}

for(int i = 0; i < n; i++) {

int a = 0;

int b = n - 1;

while(a < b) {

swap(matrix[i][a], matrix[i][b]);

a++;

b--;

}

}

}

private void swap(int x, int y) {

int t = x;

x = y;

y = t;

}

}

**Solution 1: Transpose the matrix + Swap the columns (10 min)**

class Solution {

/\*\*

1 2 3 1 4 7 7 4 1

4 5 6 => 2 5 8 => 8 5 2

7 8 9 3 6 9 9 6 3

5 1 9 11 5 2 13 15 15 13 2 5

2 4 8 10 1 4 3 14 14 3 4 1

13 3 6 7 => 9 8 6 12 => 12 6 8 9

15 14 12 16 11 10 7 16 16 7 10 11

\*/

public void rotate(int[][] matrix) {

int n = matrix.length;

for(int i = 0; i < n; i++) {

for(int j = i; j < n; j++) {

//swap(matrix[i][j], matrix[j][i]);

int t = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = t;

}

}

for(int i = 0; i < n; i++) {

int a = 0;

int b = n - 1;

while(a < b) {

//swap(matrix[i][a], matrix[i][b]);

int t = matrix[i][a];

matrix[i][a] = matrix[i][b];

matrix[i][b] = t;

a++;

b--;

}

}

}

private void swap(int x, int y) {

int t = x;

x = y;

y = t;

}

}

**Refer to**

<https://leetcode.com/problems/rotate-image/solutions/3440564/animation-understand-in-30-seconds/>

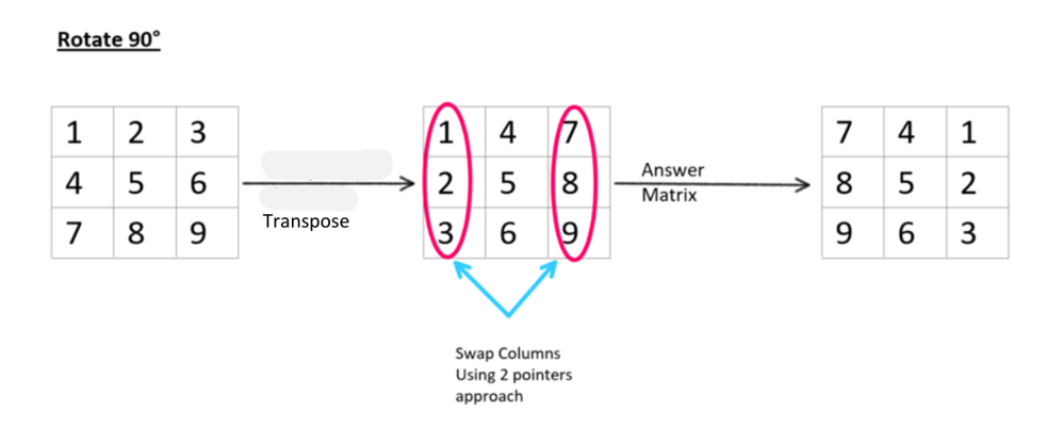
2 Steps to rotate image

* Transpose the matrix
* Swap the columns

This is a GIF



<https://leetcode.com/problems/rotate-image/solutions/1449737/rotation-90-code-180-270-360-explanation-inplace-o-1-space/>



void rotate(vector<vector<int>>& matrix) {

// complement of matrix

int n = matrix.size();

for(int i=0; i<n; ++i) {

for(int j=i; j<n; ++j) {

swap(matrix[i][j], matrix[j][i]);

}

}

for(int i=0; i<n; ++i) {

// 2 Pointer approach : just like we do in 1D array we take left and right pointers

// and swap the values and then make those pointers intersect at some point.

int left = 0, right = n-1;

while(left < right) {

swap(matrix[i][left], matrix[i][right]);

++left;

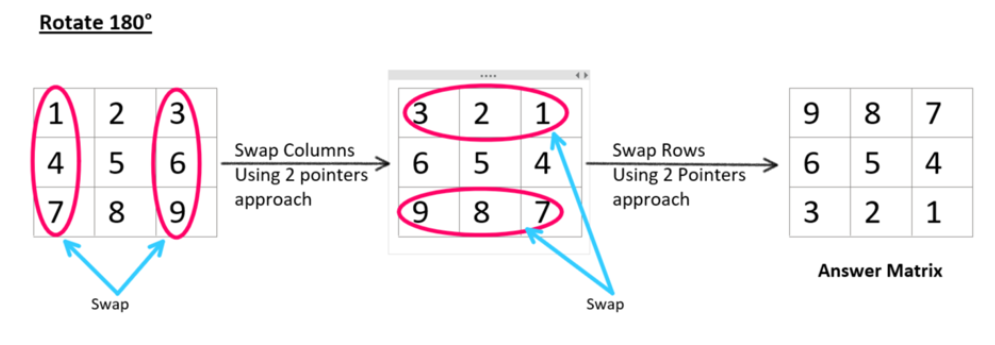
--right;

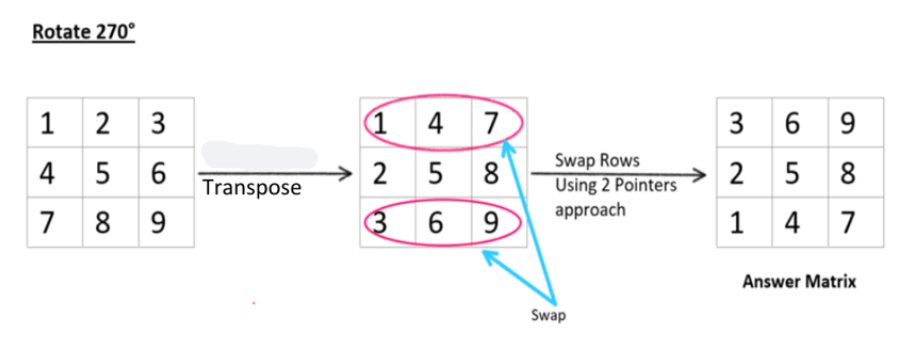
}

}

}

**For 180°, 270° and 360°**





**Rotating 360° is same given matrix so no need to do anything**

***Code will be similar for all the approaches, applying transpose and swapping(using 2 Pointers) in different order, gives us different results, This is how we can do it inplace.***