Problem 48: Rotate Image

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem 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

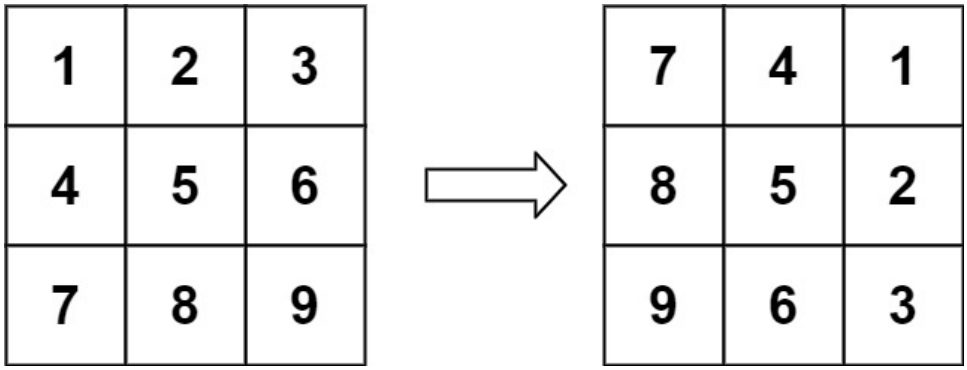
in-place

, which means you have to modify the input 2D matrix directly.

DO NOT

allocate another 2D matrix and do the rotation.

Example 1:



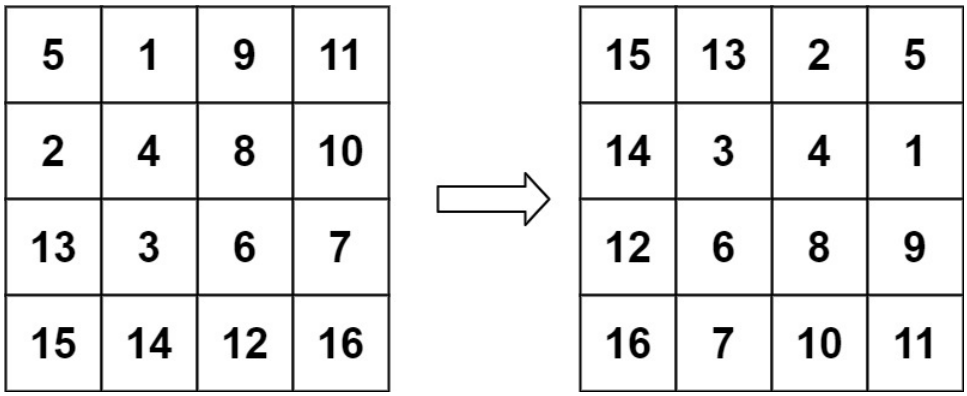
Input:

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

Code Snippets

C++:

```
class Solution {
public:
void rotate(vector<vector<int>>& matrix) {

}
};
```

Java:

```
class Solution {
public void rotate(int[][] matrix) {

}
}
```

Python3:

```
class Solution:
def rotate(self, matrix: List[List[int]]) -> None:
"""
Do not return anything, modify matrix in-place instead.
"""
```

Python:

```
class Solution(object):
def rotate(self, matrix):
"""
:type matrix: List[List[int]]
:rtype: None Do not return anything, modify matrix in-place instead.
"""
```

JavaScript:

```
/**
* @param {number[][]} matrix
* @return {void} Do not return anything, modify matrix in-place instead.
*/
var rotate = function(matrix) {

};
```

TypeScript:

```
/**
Do not return anything, modify matrix in-place instead.
*/
function rotate(matrix: number[][]): void {

};
```

C#:

```
public class Solution {
public void Rotate(int[][] matrix) {

}
}
```

C:

```
void rotate(int** matrix, int matrixSize, int* matrixColSize) {

}
```

Go:

```
func rotate(matrix [][]int) {

}
```

Kotlin:

```
class Solution {
fun rotate(matrix: Array<IntArray>): Unit {
```

```
}
}
```

Swift:

```
class Solution {
func rotate(_ matrix: inout [[Int]]) {

}
}
```

Rust:

```
impl Solution {
pub fn rotate(matrix: &mut Vec<Vec<i32>>) {

}
}
```

Ruby:

```
# @param {Integer[][]} matrix
# @return {Void} Do not return anything, modify matrix in-place instead.
def rotate(matrix)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $matrix
* @return NULL
*/
function rotate(&$matrix) {

}
}
```

Dart:

```
class Solution {
void rotate(List<List<int>> matrix) {

}
}
```

Scala:

```
object Solution {
def rotate(matrix: Array[Array[Int]]): Unit = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Rotate Image
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
void rotate(vector<vector<int>>& matrix) {

}
};
```

Java Solution:

```
/**
* Problem: Rotate Image
* Difficulty: Medium
* Tags: array, math
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Rotate Image
Difficulty: Medium
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def rotate(self, matrix: List[List[int]]) -> None:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def rotate(self, matrix):
"""
:type matrix: List[List[int]]
:rtype: None Do not return anything, modify matrix in-place instead.
"""
```

JavaScript Solution:

```
/**
* Problem: Rotate Image
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[][]} matrix
* @return {void} Do not return anything, modify matrix in-place instead.
*/
var rotate = function(matrix) {

};
```

TypeScript Solution:

```
/**
* Problem: Rotate Image
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
Do not return anything, modify matrix in-place instead.
*/
function rotate(matrix: number[][]): void {

};
```

C# Solution:

```
/*
* Problem: Rotate Image
* Difficulty: Medium
```

```
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public void Rotate(int[][] matrix) {

}
}
```

C Solution:

```
/*
* Problem: Rotate Image
* Difficulty: Medium
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

void rotate(int** matrix, int matrixSize, int* matrixColSize) {

}
```

Go Solution:

```
// Problem: Rotate Image
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func rotate(matrix [][]int) {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun rotate(matrix: Array<IntArray>): Unit {

}
}
```

Swift Solution:

```
class Solution {
func rotate(_ matrix: inout [[Int]]) {

}
}
```

Rust Solution:

```
// Problem: Rotate Image
// Difficulty: Medium
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn rotate(matrix: &mut Vec<Vec<i32>>) {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} matrix
# @return {Void} Do not return anything, modify matrix in-place instead.
def rotate(matrix)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $matrix
* @return NULL
*/
function rotate(&$matrix) {

}
}
```

Dart Solution:

```
class Solution {
void rotate(List<List<int>> matrix) {

}
}
```

Scala Solution:

```
object Solution {
def rotate(matrix: Array[Array[Int]]): Unit = {

}
}
```