leetcode problem #48 Rotate Image (Medium)

Rotate image 90 degrees in-place (means without creating new 2d array)

1 2 3   7 4 1

4 5 6 -> 8 5 2

7 8 9   9 6 3

Approach 1 :

Topic: Matrix

reverse rows

    7 8 9

    4 5 6

    1 2 3

swap symmetry (01 - 10) (02 - 20) (12 - 21) turn rows to cols

    7 4 1

    8 5 2

    9 6 3

(or)

reverse columns

    3 2 1

    6 5 4

    9 8 7

swap (00 - 22) (10 - 21) (01 - 12)

    7 4 1

    8 5 2

9 6 3

Code :

```
public void rotate(int[][] matrix) {
    for (int col = 0; col < matrix[0].length; col++) {
      int row1 = 0;
      int row2 = matrix.length - 1;

      while (row1 < row2) {
        int tmp = matrix[row1][col];
        matrix[row1++][col] = matrix[row2][col];
        matrix[row2--][col] = tmp;
      }
    }

    for (int i = 0; i < matrix.length; i++) {
      for (int j = i + 1; j < matrix[i].length; j++) {
        int tmp = matrix[i][j];
        matrix[i][j] = matrix[j][i];
        matrix[j][i] = tmp;
      }
    }
}
```

Time Complexity: O(n^2)

Space Complexity: O(1)

Approach 2:

Eg:

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

boundary = 3

Step 1: i = 0, j = 0

```
13 .   .  1
.   .   .   .
.   .   .   .
16 .   .  4
```

Step 2: i = 0, j =1

01 - 20, 20 - 31, 31 - 13, 13 - 01

```
13  9   .  1
 .   .   .  2
15   .   .  .
16  12  .   4
```

Step 3: i = 0, j = 2

02 - 10, 10 - 32, 32 - 23, 12 - 02

```
13   9    5    1
14    .    .    2
15    .    .    3
16  12   8    4
```

---- First cycle complete

boundary = 2

Step4: i = 1, j = 1

```
13   9    5    1

14  10   6    2

15  11   7    3

16  12   8    4
```

----Second cycle complete

Code:

```
public void rotate(int[][] matrix) {
        int n = matrix.length;
        int span = n - 1;
        int boundary = span;
        for (int i = 0; i < boundary; i++) {
            for (int j = i; j < boundary; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[span - j][i];
                matrix[span - j][i] = matrix[span - i][span - j];
                matrix[span - i][span - j] = matrix[j][span - i];
                matrix[j][span - i] = temp;

            }
            boundary--;
        }
    }
```

Analysis of time complexity:

Outer Loop: The outer loop runs for $n-1$ iterations, where

n is the size of the matrix.

Inner Loop: The number of iterations of the inner loop depends on the current value of boundary, which decreases by 1 in each iteration of the outer loop. In the worst case, the inner loop runs for n - 1

Therefore, the total number of iterations of the inner loop over all iterations of the outer loop is:

(n - 1) + (n - 2) + (n - 3) + ... + 1 = n * (n - 1) /2

So, the overall time complexity of the provided code is O(n^2) where

n is the size of the matrix