Determine Whether Matrix Can Be Obtained By Rotation LeetCode

You are given a 2D square matrix of integers. Your task is to determine whether a given target matrix can be obtained by rotating the input matrix clockwise in 90-degree increments.

Examples:

Input Matrix:

1 2 3

4 5 6

7 8 9

Target Matrix:

7 4 1

8 5 2

9 6 3

Output: The Target matrix can be obtained by rotating the input Matrix.

Input Matrix:

0 0 0

0 1 0

1 1 1

Target Matrix:

1 1 1

0 1 0

0 0 0

Output: Output: The Target matrix can be obtained by rotating the input Matrix.

Input Matrix:

1 1

0 1

Target Matrix:

1 0

0 1

Output: The Target matrix cannot be obtained by rotating the input Matrix.

Approach 1: Function to rotate the matrix by creating a temporary matrix

- Create a function **rotateImage** to rotate the input matrix by creating a temporary matrix.
- Iterate over the columns and rows, and reverse each column to get the rotated matrix.
- Compare the rotated matrix with the target matrix. If they are the same, return true; otherwise, rotate the input matrix again and repeat for a total of 4 rotations. If no match is found, return false.
- **Time Complexity: O(N^2), where N is the number of rows or columns in the matrix.**
- **Space Complexity: O(N^2) for the temporary matrix.**

Approach 2: Function to rotate the matrix in-place

- Create a function **rotateImageInPlace** to rotate the input matrix in-place.
- Transpose the matrix by swapping each element (i, j) with (j, i).
- Reverse each row of the transposed matrix.
- Compare the rotated matrix with the target matrix. If they are the same, return true; otherwise, rotate the input matrix again and repeat for a total of 4 rotations. If no match is found, return false.
- **Time Complexity: O(N^2), where N is the number of rows or columns in the matrix.**
- **Space Complexity: O(1) for the in-place rotation.**

Approach 3: Function to rotate the matrix in-place by using divide and conquer approach.

- Create a function **rotateImageDivideAndConquer** to rotate the input matrix in-place using the divide and conquer approach.
- Divide the matrix into layers from the outer layer to the inner layer.
- For each layer, perform four swaps to rotate the elements.
- Compare the rotated matrix with the target matrix. If they are the same, return true; otherwise, rotate the input matrix again and repeat for a total of 4 rotations. If no match is found, return false.

- **Time Complexity: O(N^2), where N is the number of rows or columns in the matrix.**
- **Space Complexity: O(1) for the in-place rotation.**