Rotate a Matrix by 90 Degrees

Question

Question Statement:

Given a 2D matrix of size N x N, rotate the matrix 90 degrees clockwise in-place (i.e., without using any extra space for another matrix).

Example:

Input:

```
[
 [1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]
]
```

Output:

```
[
 [7, 4, 1],
 [8, 5, 2],
 [9, 6, 3]
]
```

Approach 1: Transpose + Reverse Rows

Approach 1: Transpose + Reverse Rows (Optimal In-Place)

Rotate a Matrix by 90 Degrees

Steps:

1. Transpose the matrix - Swap elements across the diagonal.

2. Reverse each row - Reverse all rows to get the final rotated matrix.

Code (C++ Style):

```
void rotate(vector<vector<int>>& matrix) {
    int n = matrix.size();

    // Step 1: Transpose
    for(int i = 0; i < n; i++) {
        for(int j = i+1; j < n; j++) {
            swap(matrix[i][j], matrix[j][i]);
        }
    }

    // Step 2: Reverse each row
    for(int i = 0; i < n; i++) {
        reverse(matrix[i].begin(), matrix[i].end());
    }
}
```

Approach 2: Layer by Layer Rotation

Approach 2: Rotate Layer by Layer (In-Place)

Rotate a Matrix by 90 Degrees

Steps:

- Move four cells at a time in groups, layer by layer.

- Use a temp variable to perform cyclic swaps between top, right, bottom, and left cells.

Note: This method is slightly more complex and error-prone compared to the transpose + reverse method.

Edge Cases

Edge Cases to Consider:

1. 1x1 Matrix:

   Input: [[1]] -> Output: [[1]]

2. Empty Matrix:

   Should handle gracefully (no operation or return same).

3. Non-square Matrix:

   Not applicable here unless explicitly asked to rotate MxN matrix. This code works only for square matrices.

4. Matrix with Negative or Zero Elements:

   Rotation should preserve all values regardless of sign.