Brute Force Approach:

1. **Approach**:
    - You could create a new matrix and copy the elements from the original matrix to the new matrix in a rotated order.
    - For example, to rotate the matrix by 90 degrees clockwise, the element at position (i, j) in the original matrix will move to (j, n-1-i) in the new matrix.

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

    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            rotatedMatrix[j][n-1-i] = matrix[i][j];
        }
    }

    return rotatedMatrix;
}
```

3. **Complexity**:
    - **Time Complexity**: O(n^2), where n is the size of the matrix.
    - **Space Complexity**: O(n^2), due to the additional matrix used for storing the result.

Better Approach:

1. **Approach**:
    - The better approach involves rotating the matrix in two steps in-place:
        1. Transpose the matrix: Convert all rows to columns and columns to rows.
        2. Reverse each row: After transposition, reverse each row to get the final rotated matrix.

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

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

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

3. **Complexity**:
    - **Time Complexity**: O(n^2) for transposing the matrix and reversing each row.
    - **Space Complexity**: O(1), since we're modifying the matrix in place.

Best Approach:

1. **Approach**:
    - The best approach is the same as the better approach because it already achieves the desired result with the most optimal time and space complexities.
2. **Code**:
    - Same as the better approach.
3. **Complexity**:
    - **Time Complexity**: O(n^2).
    - **Space Complexity**: O(1).

Explanation of the Process:

1. **Transpose**:
    - Transposing involves swapping elements (i, j) with (j, i).
    - For example, given the matrix:

After transposition, it becomes:

Reverse Rows:

- Reversing each row of the transposed matrix gives:

1.
    - This is the matrix rotated 90 degrees clockwise.

This approach efficiently rotates the matrix in-place with O(1) extra space, which is optimal for this problem.