1. Power of an Element

class Solution:

def myPow(self, x: float, n: int) -> float:

if n == 0:

return 1

elif n < 0:

x = 1/x

n = -n

return self.myPow(x, n)

elif n == 1:

return x

else:

mid = n//2

b = self.myPow(x, mid)

result = b \* b

if n % 2 == 0:

return result

else:

return result \* x

2. Rotation of Image

// time complexity: O(n ^ 2)

// space complexity: O(1)

class Solution {

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

transpose(matrix);

swapCols(matrix);

}

public void transpose(int[][] matrix){

int n = matrix.length;

for(int i=0; i<n; i++){

for(int j=i+1; j<n; j++){

int temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

}

public void swapCols(int[][] matrix){

int n = matrix.length;

for(int i=0; i<n; i++){

for(int j=0; j<n/2; j++){

int temp = matrix[i][j];

matrix[i][j] = matrix[i][n - j - 1];

matrix[i][n - j - 1] = temp;

}

}

}

}

3. Matrix Diagonal Sum

// time complexity: O(n)

// space complexity: O(1)

class Solution {

public int diagonalSum(int[][] mat) {

int n = mat.length;

int result = 0;

for(int i=0; i<n; i++){

result += mat[i][i];

result += mat[n - i - 1][i];

}

if(n % 2 != 0){

result -= mat[n/2][n/2];

}

return result;

}

}