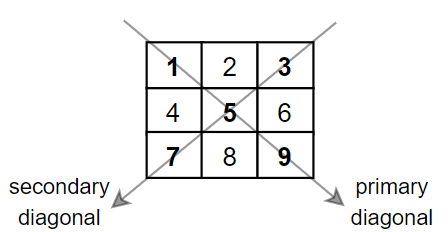
<https://leetcode.com/problems/matrix-diagonal-sum/description/>

Given a square matrix mat, return the sum of the matrix diagonals.

Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

**Example 1:**



**Input:** mat = [[**1**,2,**3**],

[4,**5**,6],

[**7**,8,**9**]]

**Output:** 25

**Explanation:** Diagonals sum: 1 + 5 + 9 + 3 + 7 = 25

Notice that element mat[1][1] = 5 is counted only once.

**Example 2:**

**Input:** mat = [[**1**,1,1,**1**],

[1,**1**,**1**,1],

[1,**1**,**1**,1],

[**1**,1,1,**1**]]

**Output:** 8

**Example 3:**

**Input:** mat = [[**5**]]

**Output:** 5

**Constraints:**

n == mat.length == mat[i].length

1 <= n <= 100

1 <= mat[i][j] <= 100

**Attempt 1: 2024-06-16**

**Solution 1: Math (10 min)**

class Solution {

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

        int n = mat.length;

        int sum = 0;

        for(int i = 0; i < n; i++) {

            sum += mat[i][i];

            sum += mat[n - 1 - i][i];

        }

        if(n % 2 == 1) {

            sum -= mat[(n - 1) / 2][(n - 1) / 2];

        }

        return sum;

    }

}

Time Complexity: O(n)

Space Complexity: O(1)

**Refer to Deepseek**

**Approach**

To solve this problem, we need to calculate the sum of the primary diagonal (top-left to bottom-right) and the secondary diagonal (top-right to bottom-left) of a square matrix. The key observation is that if the matrix has an odd size, the center element is counted twice (once in each diagonal), so we need to subtract it once from the total sum.

**Solution Code**

class Solution {

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

int n = mat.length;

int sum = 0;

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

// Add primary diagonal element

sum += mat[i][i];

// Add secondary diagonal element

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

}

// If matrix size is odd, subtract the center element (counted twice)

if (n % 2 == 1) {

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

}

return sum;

}

}

**Explanation**

**Initialization**: We start by getting the size of the matrix n and initialize sum to 0.

**Iterate through the matrix**: For each row i:

Add the element from the primary diagonal (mat[i][i]).

Add the element from the secondary diagonal (mat[i][n-1-i]).

**Adjust for odd-sized matrices**: If the matrix size is odd, subtract the center element (mat[n/2][n/2]) since it was added twice in the previous step.

**Return the result**: The final sum represents the sum of all diagonal elements without double-counting the center in odd-sized matrices.

This approach efficiently calculates the diagonal sum in O(n) time complexity (where n is the matrix size) and O(1) space complexity.

**Refer to**

[L2133.Check if Every Row and Column Contains All Numbers (Ref.L36,L37)](note://WEBfdf31b1322c8a4b1fb7c2cf200177f7b)

[L2319.Check if Matrix Is X-Matrix (Ref.L1572)](note://WEB4a9281f2dab95496c818809d49db3681)