<https://leetcode.com/problems/check-if-matrix-is-x-matrix/description/>

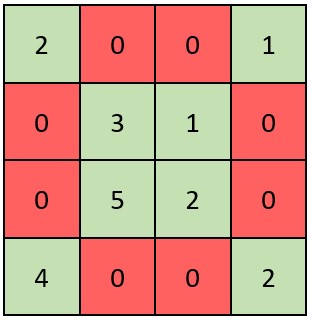
A square matrix is said to be an **X-Matrix** if **both** of the following conditions hold:

All the elements in the diagonals of the matrix are **non-zero**.

All other elements are 0.

Given a 2D integer array grid of size n x n representing a square matrix, return true*if*grid*is an X-Matrix*. Otherwise, return false.

**Example 1:**

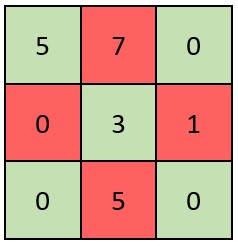


**Input:** grid = [[2,0,0,1],[0,3,1,0],[0,5,2,0],[4,0,0,2]]

**Output:** true

**Explanation:** Refer to the diagram above. An X-Matrix should have the green elements (diagonals) be non-zero and the red elements be 0.Thus, grid is an X-Matrix.

**Example 2:**



**Input:** grid = [[5,7,0],[0,3,1],[0,5,0]]

**Output:** false

**Explanation:** Refer to the diagram above.An X-Matrix should have the green elements (diagonals) be non-zero and the red elements be 0.Thus, grid is not an X-Matrix.

**Constraints:**

n == grid.length == grid[i].length

3 <= n <= 100

0 <= grid[i][j] <= 105

**Attempt 1: 2025-08-05**

**Solution 1: Math (10 min)**

class Solution {

    public boolean checkXMatrix(int[][] grid) {

        int n = grid.length;

        for(int i = 0; i < n; i++) {

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

                if((i == j || i == n - 1 - j) && grid[i][j] == 0) {

                    return false;

                }

                if(i != j && i != n - 1 - j && grid[i][j] != 0) {

                    return false;

                }

            }

        }

        return true;

    }

}

Time Complexity: O(n)

Space Complexity: O(1)

**Refer to Deepseek**

**Approach**

To determine if a matrix is an X-Matrix:

**Check Diagonal Elements**: All elements on the main diagonal (where row index equals column index) and anti-diagonal (where row index + column index equals n-1) must be non-zero.

**Check Non-Diagonal Elements**: All other elements must be zero.

**Efficient Checking**: Iterate through each element in the matrix and verify the above conditions.

**Solution Code**

class Solution {

public boolean checkXMatrix(int[][] grid) {

int n = grid.length;

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

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

// Check if current element is on either diagonal

if (i == j || i + j == n - 1) {

// Diagonal elements must be non-zero

if (grid[i][j] == 0) {

return false;

}

} else {

// Non-diagonal elements must be zero

if (grid[i][j] != 0) {

return false;

}

}

}

}

return true;

}

}

**Explanation**

**Matrix Size**: Determine the size of the square matrix n.

**Iterate Through Elements**: For each element at position (i, j):

**Diagonal Check**: If the element is on the main diagonal (i == j) or anti-diagonal (i + j == n - 1), verify it is non-zero. If zero, return false.

**Non-Diagonal Check**: For elements not on either diagonal, verify they are zero. If non-zero, return false.

**Success Case**: If all elements satisfy the conditions, return true.

This solution efficiently checks the X-Matrix conditions in O(n^2) time complexity (where n is the matrix dimension) with O(1) space complexity.

**Refer to**

[L1572.Matrix Diagonal Sum (Ref.L2133,L2319)](note://WEB4f166767aa668b346d883f3d30593f17)