Problem 2319: Check if Matrix Is X-Matrix

Problem Information

Difficulty: Easy
Acceptance Rate: 0.00%
Paid Only: No

Problem 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

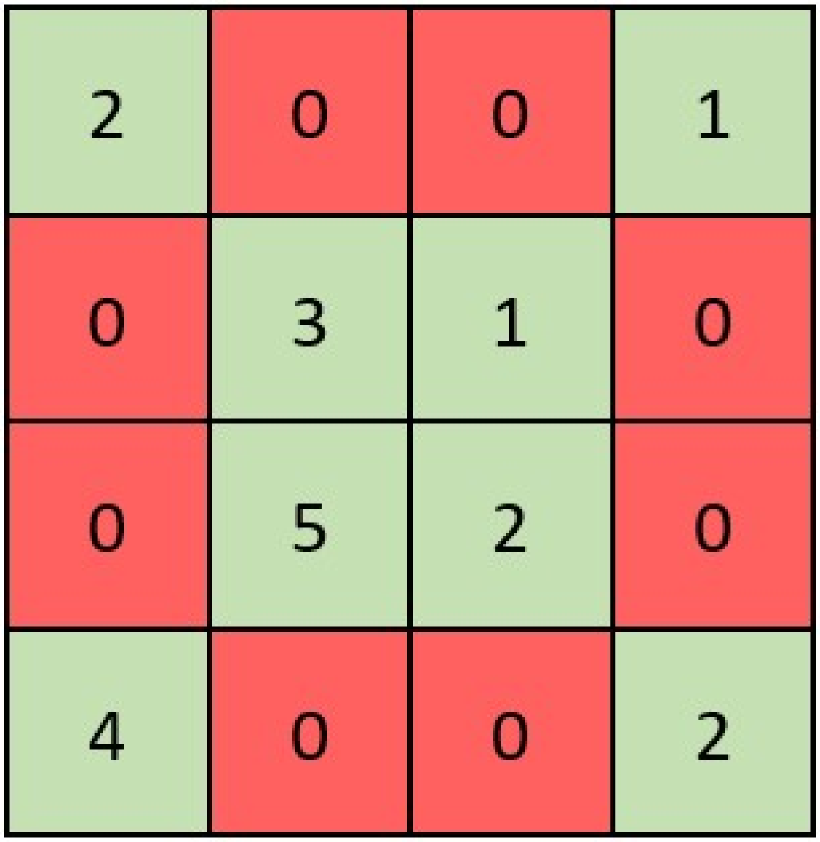
is an X-Matrix

. Otherwise, return

false

.

Example 1:



Input:

grid = [[2,0,0,1],[0,3,1,0],[0,5,2,0],[4,0,0,2]]

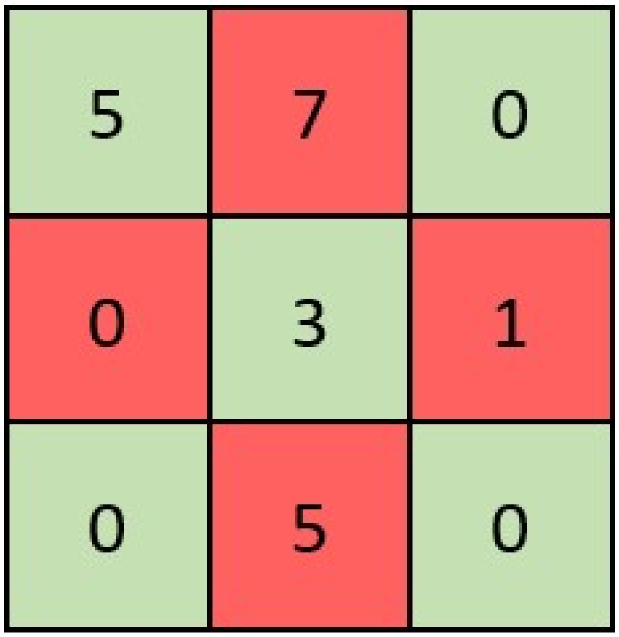
Output:

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] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
bool checkXMatrix(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public boolean checkXMatrix(int[][] grid) {

}
}
```

Python3:

```
class Solution:
def checkXMatrix(self, grid: List[List[int]]) -> bool:
```

Python:

```
class Solution(object):
def checkXMatrix(self, grid):
"""
:type grid: List[List[int]]
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number[][]} grid
* @return {boolean}
*/
var checkXMatrix = function(grid) {

};
```

TypeScript:

```
function checkXMatrix(grid: number[][]): boolean {

};
```

C#:

```
public class Solution {
public bool CheckXMatrix(int[][] grid) {

}
}
```

C:

```
bool checkXMatrix(int** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func checkXMatrix(grid [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun checkXMatrix(grid: Array<IntArray>): Boolean {

}
}
```

Swift:

```
class Solution {
func checkXMatrix(_ grid: [[Int]]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn check_x_matrix(grid: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[][]} grid
# @return {Boolean}
def check_x_matrix(grid)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Boolean
*/
function checkXMatrix($grid) {

}
}
```

Dart:

```
class Solution {
bool checkXMatrix(List<List<int>> grid) {

}
}
```

Scala:

```
object Solution {
def checkXMatrix(grid: Array[Array[Int]]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_x_matrix(grid :: [[integer]]) :: boolean
def check_x_matrix(grid) do

end
end
```

Erlang:

```
-spec check_x_matrix(Grid :: [[integer()]]) -> boolean().
check_x_matrix(Grid) ->
.
```

Racket:

```
(define/contract (check-x-matrix grid)
(-> (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if Matrix Is X-Matrix
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
bool checkXMatrix(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Matrix Is X-Matrix
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Check if Matrix Is X-Matrix
Difficulty: Easy
Tags: array
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def checkXMatrix(self, grid: List[List[int]]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def checkXMatrix(self, grid):
"""
:type grid: List[List[int]]
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Check if Matrix Is X-Matrix
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[][]} grid
* @return {boolean}
*/
var checkXMatrix = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if Matrix Is X-Matrix
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function checkXMatrix(grid: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if Matrix Is X-Matrix
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool CheckXMatrix(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Check if Matrix Is X-Matrix
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

bool checkXMatrix(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Check if Matrix Is X-Matrix
// Difficulty: Easy
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func checkXMatrix(grid [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkXMatrix(grid: Array<IntArray>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func checkXMatrix(_ grid: [[Int]]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if Matrix Is X-Matrix
// Difficulty: Easy
// Tags: array
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn check_x_matrix(grid: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} grid
# @return {Boolean}
def check_x_matrix(grid)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $grid
* @return Boolean
*/
function checkXMatrix($grid) {

}
}
```

Dart Solution:

```
class Solution {
bool checkXMatrix(List<List<int>> grid) {

}
}
```

Scala Solution:

```
object Solution {
def checkXMatrix(grid: Array[Array[Int]]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_x_matrix(grid :: [[integer]]) :: boolean
def check_x_matrix(grid) do

end
end
```

Erlang Solution:

```
-spec check_x_matrix(Grid :: [[integer()]]) -> boolean().
check_x_matrix(Grid) ->
.
```

Racket Solution:

```
(define/contract (check-x-matrix grid)
(-> (listof (listof exact-integer?)) boolean?)
)
```