Problem 766: Toeplitz Matrix

Problem Information

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

Problem Description

Given an

m x n

matrix

, return

true

if the matrix is Toeplitz. Otherwise, return

false

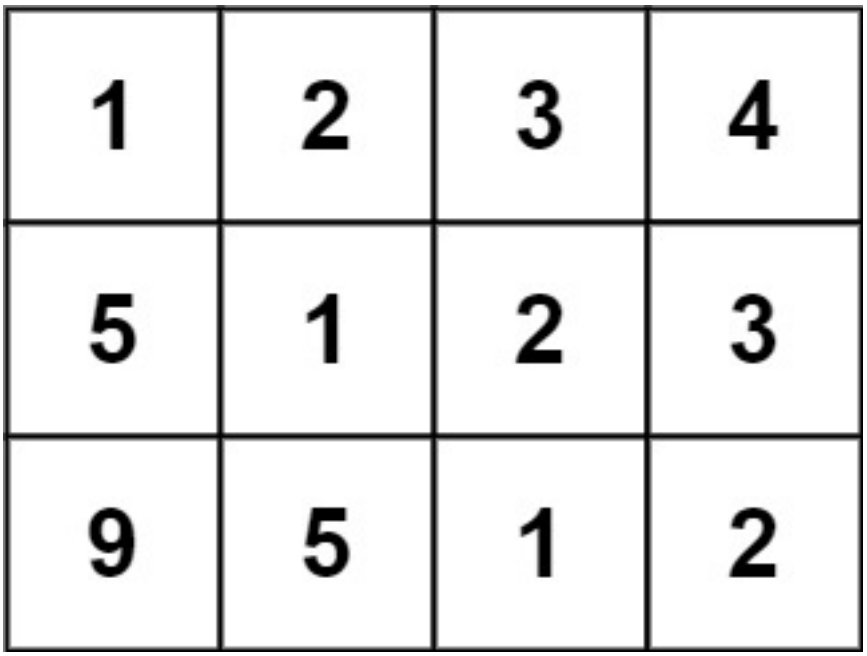
.

A matrix is

Toeplitz

if every diagonal from top-left to bottom-right has the same elements.

Example 1:



Input:

matrix = [[1,2,3,4],[5,1,2,3],[9,5,1,2]]

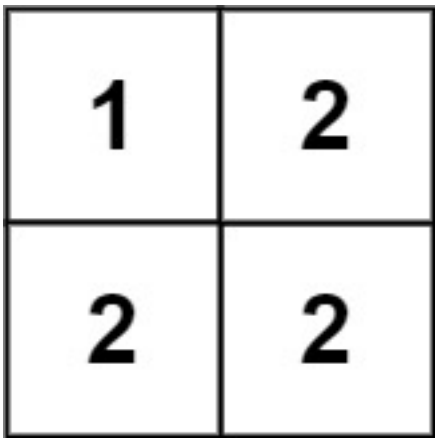
Output:

true

Explanation:

In the above grid, the diagonals are: "[9]", "[5, 5]", "[1, 1, 1]", "[2, 2, 2]", "[3, 3]", "[4]". In each diagonal all elements are the same, so the answer is True.

Example 2:



Input:

matrix = [[1,2],[2,2]]

Output:

false

Explanation:

The diagonal "[1, 2]" has different elements.

Constraints:

m == matrix.length

n == matrix[i].length

1 <= m, n <= 20

0 <= matrix[i][j] <= 99

Follow up:

What if the

matrix

is stored on disk, and the memory is limited such that you can only load at most one row of the matrix into the memory at once?

What if the

matrix

is so large that you can only load up a partial row into the memory at once?

Code Snippets

C++:

```
class Solution {
public:
bool isToeplitzMatrix(vector<vector<int>>& matrix) {

}
};
```

Java:

```
class Solution {
public boolean isToeplitzMatrix(int[][] matrix) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function isToeplitzMatrix(matrix: number[][]): boolean {

};
```

C#:

```
public class Solution {
public bool IsToeplitzMatrix(int[][] matrix) {

}
}
```

C:

```
bool isToeplitzMatrix(int** matrix, int matrixSize, int* matrixColSize) {

}
```

Go:

```
func isToeplitzMatrix(matrix [][]int) bool {

}
```

Kotlin:

```
class Solution {
fun isToeplitzMatrix(matrix: Array<IntArray>): Boolean {

}
}
```

Swift:

```
class Solution {
func isToeplitzMatrix(_ matrix: [[Int]]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_toeplitz_matrix(matrix: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[][]} matrix
# @return {Boolean}
def is_toeplitz_matrix(matrix)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $matrix
* @return Boolean
*/
function isToeplitzMatrix($matrix) {

}
}
```

Dart:

```
class Solution {
bool isToeplitzMatrix(List<List<int>> matrix) {

}
}
```

Scala:

```
object Solution {
def isToeplitzMatrix(matrix: Array[Array[Int]]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_toeplitz_matrix(matrix :: [[integer]]) :: boolean
def is_toeplitz_matrix(matrix) do

end
end
```

Erlang:

```
-spec is_toeplitz_matrix(Matrix :: [[integer()]]) -> boolean().
is_toeplitz_matrix(Matrix) ->
.
```

Racket:

```
(define/contract (is-toeplitz-matrix matrix)
(-> (listof (listof exact-integer?)) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Toeplitz 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 isToeplitzMatrix(vector<vector<int>>& matrix) {

}
};
```

Java Solution:

```
/**
* Problem: Toeplitz 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 isToeplitzMatrix(int[][] matrix) {

}
}
```

Python3 Solution:

```
"""
Problem: Toeplitz 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 isToeplitzMatrix(self, matrix: List[List[int]]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Toeplitz 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[][]} matrix
* @return {boolean}
*/
var isToeplitzMatrix = function(matrix) {

};
```

TypeScript Solution:

```
/**
* Problem: Toeplitz 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 isToeplitzMatrix(matrix: number[][]): boolean {

};
```

C# Solution:

```
/*
* Problem: Toeplitz 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 IsToeplitzMatrix(int[][] matrix) {

}
}
```

C Solution:

```
/*
* Problem: Toeplitz 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 isToeplitzMatrix(int** matrix, int matrixSize, int* matrixColSize) {

}
```

Go Solution:

```
// Problem: Toeplitz 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 isToeplitzMatrix(matrix [][]int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isToeplitzMatrix(matrix: Array<IntArray>): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isToeplitzMatrix(_ matrix: [[Int]]) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Toeplitz 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 is_toeplitz_matrix(matrix: Vec<Vec<i32>>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} matrix
# @return {Boolean}
def is_toeplitz_matrix(matrix)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $matrix
* @return Boolean
*/
function isToeplitzMatrix($matrix) {

}
}
```

Dart Solution:

```
class Solution {
bool isToeplitzMatrix(List<List<int>> matrix) {

}
}
```

Scala Solution:

```
object Solution {
def isToeplitzMatrix(matrix: Array[Array[Int]]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_toeplitz_matrix(matrix :: [[integer]]) :: boolean
def is_toeplitz_matrix(matrix) do

end
end
```

Erlang Solution:

```
-spec is_toeplitz_matrix(Matrix :: [[integer()]]) -> boolean().
is_toeplitz_matrix(Matrix) ->
.
```

Racket Solution:

```
(define/contract (is-toeplitz-matrix matrix)
(-> (listof (listof exact-integer?)) boolean?)
)
```