Problem 2387: Median of a Row Wise Sorted Matrix

Problem Information

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

Problem Description

Given an

m x n

matrix

grid

containing an

odd

number of integers where each row is sorted in

non-decreasing

order, return

the

median

of the matrix

.

You must solve the problem in less than

O(m * n)

time complexity.

Example 1:

Input:

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

Output:

2

Explanation:

The elements of the matrix in sorted order are 1,1,1,2,

2

,3,3,3,4. The median is 2.

Example 2:

Input:

grid = [[1,1,3,3,4]]

Output:

3

Explanation:

The elements of the matrix in sorted order are 1,1,

3

,3,4. The median is 3.

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 500

m

and

n

are both odd.

1 <= grid[i][j] <= 10

6

grid[i]

is sorted in non-decreasing order.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int matrixMedian(int[][] grid) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function matrixMedian(grid: number[][]): number {

};
```

C#:

```
public class Solution {
public int MatrixMedian(int[][] grid) {
```

```
}
}
```

C:

```
int matrixMedian(int** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func matrixMedian(grid [][]int) int {

}
```

Kotlin:

```
class Solution {
fun matrixMedian(grid: Array<IntArray>): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int matrixMedian(List<List<int>> grid) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (matrix-median grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Median of a Row Wise Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search
*
* 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:
int matrixMedian(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Median of a Row Wise Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 int matrixMedian(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Median of a Row Wise Sorted Matrix
Difficulty: Medium
Tags: array, sort, search

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 matrixMedian(self, grid: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Median of a Row Wise Sorted Matrix
* Difficulty: Medium
```

```
* Tags: array, sort, search
*
* 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 {number}
*/
var matrixMedian = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Median of a Row Wise Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 matrixMedian(grid: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Median of a Row Wise Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search
*
* 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 int MatrixMedian(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Median of a Row Wise Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search
*
* 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
*/

int matrixMedian(int** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Median of a Row Wise Sorted Matrix
// Difficulty: Medium
// Tags: array, sort, search
//
// 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 matrixMedian(grid [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun matrixMedian(grid: Array<IntArray>): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Median of a Row Wise Sorted Matrix
// Difficulty: Medium
// Tags: array, sort, search
//
// 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 matrix_median(grid: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[][] $grid
* @return Integer
*/
function matrixMedian($grid) {

}
}
```

Dart Solution:

```
class Solution {
int matrixMedian(List<List<int>> grid) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (matrix-median grid)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```