Problem 1428: Leftmost Column with at Least a One

Problem Information

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

Problem Description

A

row-sorted binary matrix

means that all elements are

0

or

1

and each row of the matrix is sorted in non-decreasing order.

Given a

row-sorted binary matrix

binaryMatrix

, return

the index (0-indexed) of the

leftmost column

with a 1 in it

. If such an index does not exist, return

-1

.

You can't access the Binary Matrix directly.

You may only access the matrix using a

BinaryMatrix

interface:

BinaryMatrix.get(row, col)

returns the element of the matrix at index

(row, col)

(0-indexed).

BinaryMatrix.dimensions()

returns the dimensions of the matrix as a list of 2 elements

[rows, cols]

, which means the matrix is

rows x cols

.

Submissions making more than

1000

calls to

BinaryMatrix.get

will be judged

Wrong Answer

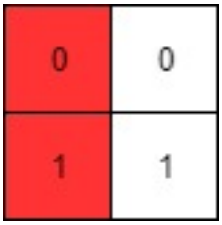
. Also, any solutions that attempt to circumvent the judge will result in disqualification.

For custom testing purposes, the input will be the entire binary matrix

mat

. You will not have access to the binary matrix directly.

Example 1:



Input:

mat = [[0,0],[1,1]]

Output:

0

Example 2:

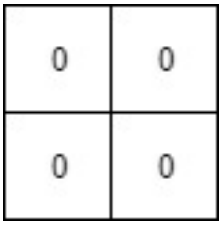
Input:

mat = [[0,0],[0,1]]

Output:

1

Example 3:



Input:

mat = [[0,0],[0,0]]

Output:

-1

Constraints:

rows == mat.length

cols == mat[i].length

1 <= rows, cols <= 100

mat[i][j]

is either

0

or

1

.

mat[i]

is sorted in non-decreasing order.

Code Snippets

C++:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* public:
* int get(int row, int col);
* vector<int> dimensions();
* };
*/

class Solution {
public:
int leftMostColumnWithOne(BinaryMatrix &binaryMatrix) {

}
};
```

Java:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
```

```
* interface BinaryMatrix {
* public int get(int row, int col) {}
* public List<Integer> dimensions {}
* };
*/

class Solution {
public int leftMostColumnWithOne(BinaryMatrix binaryMatrix) {

}
}
```

Python3:

```
# """
# This is BinaryMatrix's API interface.
# You should not implement it, or speculate about its implementation
# """
#class BinaryMatrix(object):
# def get(self, row: int, col: int) -> int:
# def dimensions(self) -> list[]:

class Solution:
def leftMostColumnWithOne(self, binaryMatrix: 'BinaryMatrix') -> int:
```

Python:

```
# """
# This is BinaryMatrix's API interface.
# You should not implement it, or speculate about its implementation
# """
#class BinaryMatrix(object):
# def get(self, row, col):
# """
# :type row : int, col : int
# :rtype int
# """
#
# def dimensions:
# """
# :rtype list[]
# """
```

```
class Solution(object):
def leftMostColumnWithOne(self, binaryMatrix):
"""
:type binaryMatrix: BinaryMatrix
:rtype: int
"""
```

JavaScript:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* function BinaryMatrix() {
* @param {integer} row, col
* @return {integer}
* this.get = function(row, col) {
* ...
* };
*
* @return {[integer, integer]}
* this.dimensions = function() {
* ...
* };
* };
*/

/**
* @param {BinaryMatrix} binaryMatrix
* @return {number}
*/
var leftMostColumnWithOne = function(binaryMatrix) {

};
```

TypeScript:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* get(row: number, col: number): number {}
```

```
*
* dimensions(): number[] {}
* }
*/

function leftMostColumnWithOne(binaryMatrix: BinaryMatrix) {

};
```

C#:

```
/**
* // This is BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* public int Get(int row, int col) {}
* public IList<int> Dimensions() {}
* }
*/

class Solution {
public int LeftMostColumnWithOne(BinaryMatrix binaryMatrix) {

}
}
```

C:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* struct BinaryMatrix {
* int (*get)(struct BinaryMatrix*, int, int);
* int* (*dimensions)(struct BinaryMatrix*);
* };
*/

int leftMostColumnWithOne(struct BinaryMatrix* matrix) {

}
```

Go:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* type BinaryMatrix struct {
* Get func(int, int) int
* Dimensions func() []int
* }
*/

func leftMostColumnWithOne(binaryMatrix BinaryMatrix) int {

}
```

Kotlin:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* fun get(row:Int, col:Int):Int {}
* fun dimensions():List<Int> {}
* }
*/

class Solution {
fun leftMostColumnWithOne(binaryMatrix:BinaryMatrix):Int {

}
}
```

Swift:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* public class BinaryMatrix {
* public func get(_ row: Int, _ col: Int) -> Int {}
* public func dimensions() -> [Int] {}
* };
*/

class Solution {
func leftMostColumnWithOne(_ binaryMatrix: BinaryMatrix) -> Int {
```

```
}
}
```

Rust:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* struct BinaryMatrix;
* impl BinaryMatrix {
* fn get(&self, row: i32, col: i32) -> i32;
* fn dimensions() -> Vec<i32>;
* };
*/

impl Solution {
pub fn left_most_column_with_one(binaryMatrix: &BinaryMatrix) -> i32 {

}
}
```

Ruby:

```
# """
# This is BinaryMatrix's API interface.
# You should not implement it, or speculate about its implementation
# """
# class BinaryMatrix
# def get(row, col)
# @return {Integer}
# end
#
# def dimensions()
# @return {List[Integer]}
# end
# end

# @param {BinaryMatrix} binaryMatrix
# @return {Integer}
def leftMostColumnWithOne(binaryMatrix)
```

```
end
```

PHP:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* public function get($row, $col) {} @return Integer
* public function dimensions() {} @return Integer[]
* }
*/

class Solution {
/**
* @param BinaryMatrix $binaryMatrix
* @return Integer
*/
public function leftMostColumnWithOne($binaryMatrix) {

}
}
```

Scala:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* def get(row: Int, col: Int): Int = {}
* def dimensions(): Array[Int] = {}
* }
*/

object Solution {
def leftMostColumnWithOne(binaryMatrix: BinaryMatrix): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Leftmost Column with at Least a One
* 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
*/

/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* public:
* int get(int row, int col);
* vector<int> dimensions();
* };
*/

class Solution {
public:
int leftMostColumnWithOne(BinaryMatrix &binaryMatrix) {

}
};
```

Java Solution:

```
/**
* Problem: Leftmost Column with at Least a One
* 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
*/
```

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* interface BinaryMatrix {
* public int get(int row, int col) {
// TODO: Implement optimized solution
return 0;
}
* public List<Integer> dimensions {
// TODO: Implement optimized solution
return 0;
}
* };
*/

class Solution {
public int leftMostColumnWithOne(BinaryMatrix binaryMatrix) {

}
}
```

Python3 Solution:

```
"""
Problem: Leftmost Column with at Least a One
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
"""

# """
# This is BinaryMatrix's API interface.
# You should not implement it, or speculate about its implementation
# """
#class BinaryMatrix(object):
# def get(self, row: int, col: int) -> int:
# def dimensions(self) -> list[]:
```

```
class Solution:
def leftMostColumnWithOne(self, binaryMatrix: 'BinaryMatrix') -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# """
# This is BinaryMatrix's API interface.
# You should not implement it, or speculate about its implementation
# """
#class BinaryMatrix(object):
# def get(self, row, col):
# """
# :type row : int, col : int
# :rtype int
# """
#
# def dimensions:
# """
# :rtype list[]
# """

class Solution(object):
def leftMostColumnWithOne(self, binaryMatrix):
"""
:type binaryMatrix: BinaryMatrix
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Leftmost Column with at Least a One
* 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
*/
```

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* function BinaryMatrix() {
* @param {integer} row, col
* @return {integer}
* this.get = function(row, col) {
* ...
* };
*
* @return {[integer, integer]}
* this.dimensions = function() {
* ...
* };
* };
*/

/**
* @param {BinaryMatrix} binaryMatrix
* @return {number}
*/
var leftMostColumnWithOne = function(binaryMatrix) {

};
```

TypeScript Solution:

```
/**
* Problem: Leftmost Column with at Least a One
* 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
*/

/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
```

```
* class BinaryMatrix {
* get(row: number, col: number): number {}
*
* dimensions(): number[] {}
* }
*/

function leftMostColumnWithOne(binaryMatrix: BinaryMatrix) {

};
```

C# Solution:

```
/*
* Problem: Leftmost Column with at Least a One
* 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
*/

/**
* // This is BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* public int Get(int row, int col) {}
* public IList<int> Dimensions() {}
* }
*/

class Solution {
public int LeftMostColumnWithOne(BinaryMatrix binaryMatrix) {

}
}
```

C Solution:

```
/*
* Problem: Leftmost Column with at Least a One
* 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
*/

/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* struct BinaryMatrix {
* int (*get)(struct BinaryMatrix*, int, int);
* int* (*dimensions)(struct BinaryMatrix*);
* };
*/

int leftMostColumnWithOne(struct BinaryMatrix* matrix) {

}
```

Go Solution:

```
// Problem: Leftmost Column with at Least a One
// 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

/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* type BinaryMatrix struct {
* Get func(int, int) int
* Dimensions func() []int
* }
*/
```

```
func leftMostColumnWithOne(binaryMatrix BinaryMatrix) int {

}
```

Kotlin Solution:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* fun get(row:Int, col:Int):Int {}
* fun dimensions():List<Int> {}
* }
*/

class Solution {
fun leftMostColumnWithOne(binaryMatrix:BinaryMatrix):Int {

}
}
```

Swift Solution:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* public class BinaryMatrix {
* public func get(_ row: Int, _ col: Int) -> Int {}
* public func dimensions() -> [Int] {}
* };
*/

class Solution {
func leftMostColumnWithOne(_ binaryMatrix: BinaryMatrix) -> Int {

}
}
```

Rust Solution:

```
// Problem: Leftmost Column with at Least a One
// 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

/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* struct BinaryMatrix;
* impl BinaryMatrix {
* fn get(&self, row: i32, col: i32) -> i32;
* fn dimensions() -> Vec<i32>;
* };
*/

impl Solution {
pub fn left_most_column_with_one(binaryMatrix: &BinaryMatrix) -> i32 {

}
}
```

Ruby Solution:

```
# """
# This is BinaryMatrix's API interface.
# You should not implement it, or speculate about its implementation
# """
# class BinaryMatrix
# def get(row, col)
# @return {Integer}
# end
#
# def dimensions()
# @return {List[Integer]}
# end
# end

# @param {BinaryMatrix} binaryMatrix
# @return {Integer}
```

```
def leftMostColumnWithOne(binaryMatrix)

end
```

PHP Solution:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* public function get($row, $col) {} @return Integer
* public function dimensions() {} @return Integer[]
* }
*/

class Solution {
/**
* @param BinaryMatrix $binaryMatrix
* @return Integer
*/
public function leftMostColumnWithOne($binaryMatrix) {

}
}
```

Scala Solution:

```
/**
* // This is the BinaryMatrix's API interface.
* // You should not implement it, or speculate about its implementation
* class BinaryMatrix {
* def get(row: Int, col: Int): Int = {}
* def dimensions(): Array[Int] = {}
* }
*/

object Solution {
def leftMostColumnWithOne(binaryMatrix: BinaryMatrix): Int = {

}
}
```