Problem 2639: Find the Width of Columns of a Grid

Problem Information

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

Problem Description

You are given a

0-indexed

m x n

integer matrix

grid

. The width of a column is the maximum

length

of its integers.

For example, if

grid = [[-10], [3], [12]]

, the width of the only column is

3

since

-10

is of length

3

.

Return

an integer array

ans

of size

n

where

ans[i]

is the width of the

i

th

column

.

The

length

of an integer

x

with

len

digits is equal to

len

if

x

is non-negative, and

len + 1

otherwise.

Example 1:

Input:

grid = [[1],[22],[333]]

Output:

[3]

Explanation:

In the 0

th

column, 333 is of length 3.

Example 2:

Input:

grid = [[-15,1,3],[15,7,12],[5,6,-2]]

Output:

[3,1,2]

Explanation:

In the 0

th

column, only -15 is of length 3. In the 1

st

column, all integers are of length 1. In the 2

nd

column, both 12 and -2 are of length 2.

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 100

-10

9

<= grid[r][c] <= 10

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[][]} grid
* @return {number[]}
*/
```

```
var findColumnWidth = function(grid) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findColumnWidth(int** grid, int gridSize, int* gridColSize, int*
returnSize) {

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
```

```
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-column-width grid)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Width of Columns of a Grid
* 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:
vector<int> findColumnWidth(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Width of Columns of a Grid
* 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 int[] findColumnWidth(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Width of Columns of a Grid
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 findColumnWidth(self, grid: List[List[int]]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Width of Columns of a Grid
* 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 {number[]}
*/
var findColumnWidth = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Width of Columns of a Grid
* 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 findColumnWidth(grid: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find the Width of Columns of a Grid
* 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 int[] FindColumnWidth(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Find the Width of Columns of a Grid
* 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
*/

/**
```

```
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findColumnWidth(int** grid, int gridSize, int* gridColSize, int*
returnSize) {

}
```

Go Solution:

```
// Problem: Find the Width of Columns of a Grid
// 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 findColumnWidth(grid [][]int) []int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Width of Columns of a Grid
// 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 find_column_width(grid: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-column-width grid)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```