Problem 2732: Find a Good Subset of the Matrix

Problem Information

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

Problem Description

You are given a

0-indexed

m x n

binary matrix

grid

.

Let us call a

non-empty

subset of rows

good

if the sum of each column of the subset is at most half of the length of the subset.

More formally, if the length of the chosen subset of rows is

k

, then the sum of each column should be at most

floor(k / 2)

.

Return

an integer array that contains row indices of a good subset sorted in

ascending

order.

If there are multiple good subsets, you can return any of them. If there are no good subsets, return an empty array.

A

subset

of rows of the matrix

grid

is any matrix that can be obtained by deleting some (possibly none or all) rows from

grid

.

Example 1:

Input:

grid = [[0,1,1,0],[0,0,0,1],[1,1,1,1]]

Output:

[0,1]

Explanation:

We can choose the 0

th

and 1

st

rows to create a good subset of rows. The length of the chosen subset is 2. - The sum of the 0

th

column is 0 + 0 = 0, which is at most half of the length of the subset. - The sum of the 1

st

column is 1 + 0 = 1, which is at most half of the length of the subset. - The sum of the 2

nd

column is 1 + 0 = 1, which is at most half of the length of the subset. - The sum of the 3

rd

column is 0 + 1 = 1, which is at most half of the length of the subset.

Example 2:

Input:

grid = [[0]]

Output:

[0]

Explanation:

We can choose the 0

th

row to create a good subset of rows. The length of the chosen subset is 1. - The sum of the 0

th

column is 0, which is at most half of the length of the subset.

Example 3:

Input:

grid = [[1,1,1],[1,1,1]]

Output:

[]

Explanation:

It is impossible to choose any subset of rows to create a good subset.

Constraints:

m == grid.length

n == grid[i].length

1 <= m <= 10

4

1 <= n <= 5

grid[i][j]

is either

0

or

1

.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public List<Integer> goodSubsetofBinaryMatrix(int[][] grid) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def goodSubsetofBinaryMatrix(self, grid):
```

```
"""
:type grid: List[List[int]]
:rtype: List[int]
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public IList<int> GoodSubsetofBinaryMatrix(int[][] grid) {

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (good-subsetof-binary-matrix grid)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find a Good Subset of the Matrix
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Find a Good Subset of the Matrix
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public List<Integer> goodSubsetofBinaryMatrix(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Find a Good Subset of the Matrix
Difficulty: Hard
Tags: array, hash, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find a Good Subset of the Matrix
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

```
var goodSubsetofBinaryMatrix = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Find a Good Subset of the Matrix
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function goodSubsetofBinaryMatrix(grid: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find a Good Subset of the Matrix
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public IList<int> GoodSubsetofBinaryMatrix(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Find a Good Subset of the Matrix
* Difficulty: Hard
* Tags: array, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

Go Solution:

```
// Problem: Find a Good Subset of the Matrix
// Difficulty: Hard
// Tags: array, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func goodSubsetofBinaryMatrix(grid [][]int) []int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find a Good Subset of the Matrix
// Difficulty: Hard
// Tags: array, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (good-subsetof-binary-matrix grid)
(-> (listof (listof exact-integer?)) (listof exact-integer?))

)
```