Problem 2482: Difference Between Ones and Zeros in Row and Column

Problem Information

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

Problem Description

You are given a

0-indexed

m x n

binary matrix

grid

.

A

0-indexed

m x n

difference matrix

diff

is created with the following procedure:

Let the number of ones in the

i

th

row be

onesRow

i

.

Let the number of ones in the

j

th

column be

onesCol

j

.

Let the number of zeros in the

i

th

row be

zerosRow

i

.

Let the number of zeros in the

j

th

column be

zerosCol

j

.

diff[i][j] = onesRow

i

+ onesCol

j

- zerosRow

i

- zerosCol

j

Return

the difference matrix

diff

.

Example 1:

grid

0	1	1
1	0	1
0	0	1

diff

0	0	4
0	0	4
-2	-2	2

Input:

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

Output:

[[0,0,4],[0,0,4],[-2,-2,2]]

Explanation:

- diff[0][0] =

onesRow

0

+ onesCol

0

- zerosRow

0

- zerosCol

0

= 2 + 1 - 1 - 2 = 0 - diff[0][1] =

onesRow

0

+ onesCol

1

- zerosRow

0

- zerosCol

1

= 2 + 1 - 1 - 2 = 0 - diff[0][2] =

onesRow

0

+ onesCol

2

- zerosRow

0

- zerosCol

2

= 2 + 3 - 1 - 0 = 4 - diff[1][0] =

onesRow

1

+ onesCol

0

- zerosRow

1

- zerosCol

0

= 2 + 1 - 1 - 2 = 0 - diff[1][1] =

onesRow

1

+ onesCol

1

- zerosRow

1

- zerosCol

1

= 2 + 1 - 1 - 2 = 0 - diff[1][2] =

onesRow

1

+ onesCol

2

- zerosRow

1

- zerosCol

2

= 2 + 3 - 1 - 0 = 4 - diff[2][0] =

onesRow

2

+ onesCol

0

- zerosRow

2

- zerosCol

0

= 1 + 1 - 2 - 2 = -2 - diff[2][1] =

onesRow

2

+ onesCol

1

- zerosRow

2

- zerosCol

1

= 1 + 1 - 2 - 2 = -2 - diff[2][2] =

onesRow

2

+ onesCol

2

- zerosRow

2

- zerosCol

2

= 1 + 3 - 2 - 0 = 2

Example 2:

grid

1	1	1
1	1	1

diff

5	5	5
5	5	5

Input:

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

Output:

[[5,5,5],[5,5,5]]

Explanation:

- diff[0][0] = onesRow

0

+ onesCol

0

- zerosRow

0

- zerosCol

0

= 3 + 2 - 0 - 0 = 5 - diff[0][1] = onesRow

0

+ onesCol

1

- zerosRow

0

- zerosCol

1

= 3 + 2 - 0 - 0 = 5 - diff[0][2] = onesRow

0

+ onesCol

2

- zerosRow

0

- zerosCol

2

= 3 + 2 - 0 - 0 = 5 - diff[1][0] = onesRow

1

+ onesCol

0

- zerosRow

1

- zerosCol

0

= 3 + 2 - 0 - 0 = 5 - diff[1][1] = onesRow

1

+ onesCol

1

- zerosRow

1

- zerosCol

1

= 3 + 2 - 0 - 0 = 5 - diff[1][2] = onesRow

1

+ onesCol

2

- zerosRow

1

- zerosCol

2

= 3 + 2 - 0 - 0 = 5

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 10

5

1 <= m * n <= 10

5

grid[i][j]

is either

0

or

1

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
/**
* Return an array of arrays of size *returnSize.
* The sizes of the arrays are returned as *returnColumnSizes array.
* Note: Both returned array and *columnSizes array must be malloced, assume
caller calls free().
*/
int** onesMinusZeros(int** grid, int gridSize, int* gridColSize, int*
returnSize, int** returnColumnSizes) {

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

```
(define/contract (ones-minus-zeros grid)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Difference Between Ones and Zeros in Row and Column
* Difficulty: Medium
* 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<vector<int>> onesMinusZeros(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Difference Between Ones and Zeros in Row and Column
```

```
* Difficulty: Medium
* 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[][] onesMinusZeros(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Difference Between Ones and Zeros in Row and Column
Difficulty: Medium
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 onesMinusZeros(self, grid: List[List[int]]) -> List[List[int]]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```javascript
/**
* Problem: Difference Between Ones and Zeros in Row and Column
* Difficulty: Medium
* 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 onesMinusZeros = function(grid) {

};
```

TypeScript Solution:

```typescript
/**
* Problem: Difference Between Ones and Zeros in Row and Column
* Difficulty: Medium
* 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 onesMinusZeros(grid: number[][]): number[][] {

};
```

C# Solution:

```csharp
/*
* Problem: Difference Between Ones and Zeros in Row and Column
* Difficulty: Medium
* 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[][] OnesMinusZeros(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Difference Between Ones and Zeros in Row and Column
* Difficulty: Medium
* 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
*/

/**
* Return an array of arrays of size *returnSize.
* The sizes of the arrays are returned as *returnColumnSizes array.
* Note: Both returned array and *columnSizes array must be malloced, assume
caller calls free().
*/
int** onesMinusZeros(int** grid, int gridSize, int* gridColSize, int*
returnSize, int** returnColumnSizes) {

}
```

Go Solution:

```
// Problem: Difference Between Ones and Zeros in Row and Column
// Difficulty: Medium
// 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 onesMinusZeros(grid [][]int) [][]int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Difference Between Ones and Zeros in Row and Column
// Difficulty: Medium
// 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 ones_minus_zeros(grid: Vec<Vec<i32>>) -> Vec<Vec<i32>> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
```

```
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (ones-minus-zeros grid)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)))
)
```