Problem 2965: Find Missing and Repeated Values

Problem Information

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

Problem Description

You are given a

0-indexed

2D integer matrix

grid

of size

n * n

with values in the range

[1, n

2

]

. Each integer appears

exactly once

except

a

which appears

twice

and

b

which is

missing

. The task is to find the repeating and missing numbers

a

and

b

.

Return

a

0-indexed

integer array

ans

of size

2

where

ans[0]

equals to

a

and

ans[1]

equals to

b

.

Example 1:

Input:

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

Output:

[2,4]

Explanation:

Number 2 is repeated and number 4 is missing so the answer is [2,4].

Example 2:

Input:

grid = [[9,1,7],[8,9,2],[3,4,6]]

Output:

[9,5]

Explanation:

Number 9 is repeated and number 5 is missing so the answer is [9,5].

Constraints:

2 <= n == grid.length == grid[i].length <= 50

1 <= grid[i][j] <= n * n

For all

x

that

1 <= x <= n * n

there is exactly one

x

that is not equal to any of the grid members.

For all

x

that

1 <= x <= n * n

there is exactly one

x

that is equal to exactly two of the grid members.

For all

x

that

1 <= x <= n * n

except two of them there is exactly one pair of

i, j

that

0 <= i, j <= n - 1

and

grid[i][j] == x

.

Code Snippets

C++:

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

}
};
```

Java:

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

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-missing-and-repeated-values grid)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Missing and Repeated Values
* Difficulty: Easy
* Tags: array, math, hash
*
* 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> findMissingAndRepeatedValues(vector<vector<int>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Find Missing and Repeated Values
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 int[] findMissingAndRepeatedValues(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Missing and Repeated Values
Difficulty: Easy
Tags: array, math, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Missing and Repeated Values
* Difficulty: Easy
```

```
* Tags: array, math, hash
*
* 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 findMissingAndRepeatedValues = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Missing and Repeated Values
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 findMissingAndRepeatedValues(grid: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find Missing and Repeated Values
* Difficulty: Easy
* Tags: array, math, hash
*
* 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 int[] FindMissingAndRepeatedValues(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Find Missing and Repeated Values
* Difficulty: Easy
* Tags: array, math, hash
*
* 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* findMissingAndRepeatedValues(int** grid, int gridSize, int* gridColSize,
int* returnSize) {

}
```

Go Solution:

```
// Problem: Find Missing and Repeated Values
// Difficulty: Easy
// Tags: array, math, hash
//
// 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 findMissingAndRepeatedValues(grid [][]int) []int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Missing and Repeated Values
// Difficulty: Easy
// Tags: array, math, hash
//
// 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 find_missing_and_repeated_values(grid: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-missing-and-repeated-values grid)
(-> (listof (listof exact-integer?)) (listof exact-integer?))
)
```