Problem 2215: Find the Difference of Two Arrays

Problem Information

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

Problem Description

Given two

0-indexed

integer arrays

nums1

and

nums2

, return

a list

answer

of size

2

where:

answer[0]

is a list of all

distinct

integers in

nums1

which are

not

present in

nums2

.

answer[1]

is a list of all

distinct

integers in

nums2

which are

not

present in

nums1

.

Note

that the integers in the lists may be returned in

any

order.

Example 1:

Input:

nums1 = [1,2,3], nums2 = [2,4,6]

Output:

[[1,3],[4,6]]

Explanation:

For nums1, nums1[1] = 2 is present at index 0 of nums2, whereas nums1[0] = 1 and nums1[2] = 3 are not present in nums2. Therefore, answer[0] = [1,3]. For nums2, nums2[0] = 2 is present at index 1 of nums1, whereas nums2[1] = 4 and nums2[2] = 6 are not present in nums1. Therefore, answer[1] = [4,6].

Example 2:

Input:

nums1 = [1,2,3,3], nums2 = [1,1,2,2]

Output:

[[3],[]]

Explanation:

For nums1, nums1[2] and nums1[3] are not present in nums2. Since nums1[2] == nums1[3], their value is only included once and answer[0] = [3]. Every integer in nums2 is present in

nums1. Therefore, answer[1] = [].

Constraints:

1 <= nums1.length, nums2.length <= 1000

-1000 <= nums1[i], nums2[i] <= 1000

Code Snippets

C++:

```
class Solution {
public:
vector<vector<int>> findDifference(vector<int>& nums1, vector<int>& nums2) {

}
};
```

Java:

```
class Solution {
public List<List<Integer>> findDifference(int[] nums1, int[] nums2) {

}
}
```

Python3:

```
class Solution:
def findDifference(self, nums1: List[int], nums2: List[int]) ->
List[List[int]]:
```

Python:

```
class Solution(object):
def findDifference(self, nums1, nums2):
"""
:type nums1: List[int]
:type nums2: List[int]
:rtype: List[List[int]]
```

```
"""
```

JavaScript:

```
/**
* @param {number[]} nums1
* @param {number[]} nums2
* @return {number[][]}
*/
var findDifference = function(nums1, nums2) {

};
```

TypeScript:

```
function findDifference(nums1: number[], nums2: number[]): number[][] {

};
```

C#:

```
public class Solution {
public IList<IList<int>> FindDifference(int[] nums1, int[] nums2) {

}
}
```

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** findDifference(int* nums1, int nums1Size, int* nums2, int nums2Size,
int* returnSize, int** returnColumnSizes) {

}
```

Go:

```
func findDifference(nums1 []int, nums2 []int) [][]int {

}
```

Kotlin:

```
class Solution {
fun findDifference(nums1: IntArray, nums2: IntArray): List<List<Int>> {

}
}
```

Swift:

```
class Solution {
func findDifference(_ nums1: [Int], _ nums2: [Int]) -> [[Int]] {

}
}
```

Rust:

```
impl Solution {
pub fn find_difference(nums1: Vec<i32>, nums2: Vec<i32>) -> Vec<Vec<i32>> {

}
}
```

Ruby:

```
# @param {Integer[]} nums1
# @param {Integer[]} nums2
# @return {Integer[][]}
def find_difference(nums1, nums2)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums1
```

```
* @param Integer[] $nums2
* @return Integer[][]
*/
function findDifference($nums1, $nums2) {

}
}
```

Dart:

```
class Solution {
List<List<int>> findDifference(List<int> nums1, List<int> nums2) {

}
}
```

Scala:

```
object Solution {
def findDifference(nums1: Array[Int], nums2: Array[Int]): List[List[Int]] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_difference(nums1 :: [integer], nums2 :: [integer]) :: [[integer]]
def find_difference(nums1, nums2) do

end
end
```

Erlang:

```
-spec find_difference(Nums1 :: [integer()], Nums2 :: [integer()]) ->
[[integer()]].
find_difference(Nums1, Nums2) ->
.
```

Racket:

```
(define/contract (find-difference nums1 nums2)
(-> (listof exact-integer?) (listof exact-integer?) (listof (listof
exact-integer?)))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Difference of Two Arrays
* Difficulty: Easy
* Tags: array, 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<vector<int>> findDifference(vector<int>& nums1, vector<int>& nums2) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Difference of Two Arrays
* Difficulty: Easy
* Tags: array, 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 List<List<Integer>> findDifference(int[] nums1, int[] nums2) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Find the Difference of Two Arrays
Difficulty: Easy
Tags: array, 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 findDifference(self, nums1: List[int], nums2: List[int]) ->
List[List[int]]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findDifference(self, nums1, nums2):
"""
:type nums1: List[int]
:type nums2: List[int]
:rtype: List[List[int]]
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Difference of Two Arrays
* Difficulty: Easy
* Tags: array, 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[]} nums1
* @param {number[]} nums2
* @return {number[][]}
*/
var findDifference = function(nums1, nums2) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Difference of Two Arrays
* Difficulty: Easy
* Tags: array, 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 findDifference(nums1: number[], nums2: number[]): number[][] {

};
```

C# Solution:

```
/*
* Problem: Find the Difference of Two Arrays
* Difficulty: Easy
* Tags: array, 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 IList<IList<int>> FindDifference(int[] nums1, int[] nums2) {
```

```
}
}
```

C Solution:

```
/*
* Problem: Find the Difference of Two Arrays
* Difficulty: Easy
* Tags: array, 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
*/

/**
* 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** findDifference(int* nums1, int nums1Size, int* nums2, int nums2Size,
int* returnSize, int** returnColumnSizes) {

}
```

Go Solution:

```
// Problem: Find the Difference of Two Arrays
// Difficulty: Easy
// Tags: array, 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 findDifference(nums1 []int, nums2 []int) [][]int {

}
```

Kotlin Solution:

```
class Solution {
fun findDifference(nums1: IntArray, nums2: IntArray): List<List<Int>> {

}
}
```

Swift Solution:

```
class Solution {
func findDifference(_ nums1: [Int], _ nums2: [Int]) -> [[Int]] {

}
}
```

Rust Solution:

```
// Problem: Find the Difference of Two Arrays
// Difficulty: Easy
// Tags: array, 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_difference(nums1: Vec<i32>, nums2: Vec<i32>) -> Vec<Vec<i32>> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums1
# @param {Integer[]} nums2
# @return {Integer[][]}
def find_difference(nums1, nums2)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums1
* @param Integer[] $nums2
* @return Integer[][]
*/
function findDifference($nums1, $nums2) {

}
}
```

Dart Solution:

```
class Solution {
List<List<int>> findDifference(List<int> nums1, List<int> nums2) {

}
}
```

Scala Solution:

```
object Solution {
def findDifference(nums1: Array[Int], nums2: Array[Int]): List[List[Int]] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_difference(nums1 :: [integer], nums2 :: [integer]) :: [[integer]]
def find_difference(nums1, nums2) do

end
end
```

Erlang Solution:

```
-spec find_difference(Nums1 :: [integer()], Nums2 :: [integer()]) ->
[[integer()]].
find_difference(Nums1, Nums2) ->
```

```
.
```

Racket Solution:

```
(define/contract (find-difference nums1 nums2)
(-> (listof exact-integer?) (listof exact-integer?) (listof (listof
exact-integer?)))
)
```