Problem 1365: How Many Numbers Are Smaller Than the Current Number

Problem Information

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

Problem Description

Given the array

nums

, for each

nums[i]

find out how many numbers in the array are smaller than it. That is, for each

nums[i]

you have to count the number of valid

j's

such that

j != i

and

nums[j] < nums[i]

.

Return the answer in an array.

Example 1:

Input:

nums = [8,1,2,2,3]

Output:

[4,0,1,1,3]

Explanation:

For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3). For nums[1]=1 does not exist any smaller number than it. For nums[2]=2 there exist one smaller number than it (1). For nums[3]=2 there exist one smaller number than it (1). For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).

Example 2:

Input:

nums = [6,5,4,8]

Output:

[2,1,0,3]

Example 3:

Input:

nums = [7,7,7,7]

Output:

[0,0,0,0]

Constraints:

2 <= nums.length <= 500

0 <= nums[i] <= 100

Code Snippets

C++:

```
class Solution {
public:
vector<int> smallerNumbersThanCurrent(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int[] smallerNumbersThanCurrent(int[] nums) {

}
}
```

Python3:

```
class Solution:
def smallerNumbersThanCurrent(self, nums: List[int]) -> List[int]:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function smallerNumbersThanCurrent(nums: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] SmallerNumbersThanCurrent(int[] nums) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* smallerNumbersThanCurrent(int* nums, int numsSize, int* returnSize) {

}
```

Go:

```
func smallerNumbersThanCurrent(nums []int) []int {

}
```

Kotlin:

```
class Solution {
fun smallerNumbersThanCurrent(nums: IntArray): IntArray {

}
```

```
}
```

Swift:

```
class Solution {
func smallerNumbersThanCurrent(_ nums: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn smaller_numbers_than_current(nums: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer[]}
def smaller_numbers_than_current(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer[]
*/
function smallerNumbersThanCurrent($nums) {

}
}
```

Dart:

```
class Solution {
List<int> smallerNumbersThanCurrent(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def smallerNumbersThanCurrent(nums: Array[Int]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smaller_numbers_than_current(nums :: [integer]) :: [integer]
def smaller_numbers_than_current(nums) do

end
end
```

Erlang:

```
-spec smaller_numbers_than_current(Nums :: [integer()]) -> [integer()].
smaller_numbers_than_current(Nums) ->
.
```

Racket:

```
(define/contract (smaller-numbers-than-current nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: How Many Numbers Are Smaller Than the Current Number
```

```
* Difficulty: Easy
* 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> smallerNumbersThanCurrent(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: How Many Numbers Are Smaller Than the Current Number
* Difficulty: Easy
* 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 int[] smallerNumbersThanCurrent(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: How Many Numbers Are Smaller Than the Current Number
Difficulty: Easy
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 smallerNumbersThanCurrent(self, nums: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: How Many Numbers Are Smaller Than the Current Number
* Difficulty: Easy
* 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[]} nums
* @return {number[]}
*/
var smallerNumbersThanCurrent = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: How Many Numbers Are Smaller Than the Current Number
* Difficulty: Easy
* 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 smallerNumbersThanCurrent(nums: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: How Many Numbers Are Smaller Than the Current Number
* Difficulty: Easy
* 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 int[] SmallerNumbersThanCurrent(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: How Many Numbers Are Smaller Than the Current Number
* Difficulty: Easy
* 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* smallerNumbersThanCurrent(int* nums, int numsSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: How Many Numbers Are Smaller Than the Current Number
// Difficulty: Easy
// 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 smallerNumbersThanCurrent(nums []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun smallerNumbersThanCurrent(nums: IntArray): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func smallerNumbersThanCurrent(_ nums: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: How Many Numbers Are Smaller Than the Current Number
// Difficulty: Easy
// 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 smaller_numbers_than_current(nums: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer[]}
def smaller_numbers_than_current(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer[]
*/
function smallerNumbersThanCurrent($nums) {

}
}
```

Dart Solution:

```
class Solution {
List<int> smallerNumbersThanCurrent(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def smallerNumbersThanCurrent(nums: Array[Int]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smaller_numbers_than_current(nums :: [integer]) :: [integer]
def smaller_numbers_than_current(nums) do

end
end
```

Erlang Solution:

```
-spec smaller_numbers_than_current(Nums :: [integer()]) -> [integer()].
smaller_numbers_than_current(Nums) ->
.
```

Racket Solution:

```
(define/contract (smaller-numbers-than-current nums)
(-> (listof exact-integer?) (listof exact-integer?))
)
```