Problem 1608: Special Array With X Elements Greater Than or Equal X

Problem Information

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

Problem Description

You are given an array

nums

of non-negative integers.

nums

is considered

special

if there exists a number

x

such that there are

exactly

x

numbers in

nums

that are

greater than or equal to

x

.

Notice that

x

does not

have to be an element in

nums

.

Return

x

if the array is

special

, otherwise, return

-1

. It can be proven that if

nums

is special, the value for

x

is

unique

.

Example 1:

Input:

nums = [3,5]

Output:

2

Explanation:

There are 2 values (3 and 5) that are greater than or equal to 2.

Example 2:

Input:

nums = [0,0]

Output:

-1

Explanation:

No numbers fit the criteria for x. If x = 0, there should be 0 numbers >= x, but there are 2. If x = 1, there should be 1 number >= x, but there are 0. If x = 2, there should be 2 numbers >= x, but there are 0. x cannot be greater since there are only 2 numbers in nums.

Example 3:

Input:

nums = [0,4,3,0,4]

Output:

3

Explanation:

There are 3 values that are greater than or equal to 3.

Constraints:

1 <= nums.length <= 100

0 <= nums[i] <= 1000

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int specialArray(int* nums, int numsSize) {

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun specialArray(nums: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
```

```
*/
function specialArray($nums) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (special-array nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Special Array With X Elements Greater Than or Equal X
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 specialArray(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Special Array With X Elements Greater Than or Equal X
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 specialArray(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Special Array With X Elements Greater Than or Equal X
Difficulty: Easy
Tags: array, sort, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Special Array With X Elements Greater Than or Equal X
* Difficulty: Easy
* Tags: array, sort, search
*
* 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[]} nums
* @return {number}
*/
var specialArray = function(nums) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Special Array With X Elements Greater Than or Equal X
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 specialArray(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Special Array With X Elements Greater Than or Equal X
* Difficulty: Easy
* Tags: array, sort, search
*
* 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 SpecialArray(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Special Array With X Elements Greater Than or Equal X
* Difficulty: Easy
```

```
* Tags: array, sort, search
*
* 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
*/

int specialArray(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Special Array With X Elements Greater Than or Equal X
// Difficulty: Easy
// Tags: array, sort, search
//
// 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 specialArray(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun specialArray(nums: IntArray): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Special Array With X Elements Greater Than or Equal X
// Difficulty: Easy
// Tags: array, sort, search
//
// 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 special_array(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (special-array nums)
(-> (listof exact-integer?) exact-integer?)
)
```