Problem 2529: Maximum Count of Positive Integer and Negative Integer

Problem Information

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

Problem Description

Given an array

nums

sorted in

non-decreasing

order, return

the maximum between the number of positive integers and the number of negative integers.

In other words, if the number of positive integers in

nums

is

pos

and the number of negative integers is

neg

, then return the maximum of

pos

and

neg

.

Note

that

0

is neither positive nor negative.

Example 1:

Input:

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

Output:

3

Explanation:

There are 3 positive integers and 3 negative integers. The maximum count among them is 3.

Example 2:

Input:

nums = [-3,-2,-1,0,0,1,2]

Output:

3

Explanation:

There are 2 positive integers and 3 negative integers. The maximum count among them is 3.

Example 3:

Input:

nums = [5,20,66,1314]

Output:

4

Explanation:

There are 4 positive integers and 0 negative integers. The maximum count among them is 4.

Constraints:

1 <= nums.length <= 2000

-2000 <= nums[i] <= 2000

nums

is sorted in a

non-decreasing order

.

Follow up:

Can you solve the problem in

O(log(n))

time complexity?

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
```

```
var maximumCount = function(nums) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

```
}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (maximum-count nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Count of Positive Integer and Negative Integer
* 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 maximumCount(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Count of Positive Integer and Negative Integer
* 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 maximumCount(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Count of Positive Integer and Negative Integer
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 maximumCount(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Count of Positive Integer and Negative Integer
* 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 maximumCount = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Count of Positive Integer and Negative Integer
* 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 maximumCount(nums: number[]): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Maximum Count of Positive Integer and Negative Integer
* 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 MaximumCount(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Count of Positive Integer and Negative Integer
* 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 maximumCount(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Maximum Count of Positive Integer and Negative Integer
// 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 maximumCount(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Count of Positive Integer and Negative Integer
// 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 maximum_count(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (maximum-count nums)
(-> (listof exact-integer?) exact-integer?)
)
```