Problem 2587: Rearrange Array to Maximize Prefix Score

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

. You can rearrange the elements of

nums

to

any order

(including the given order).

Let

prefix

be the array containing the prefix sums of

nums

after rearranging it. In other words,

prefix[i]

is the sum of the elements from

0

to

i

in

nums

after rearranging it. The

score

of

nums

is the number of positive integers in the array

prefix

.

Return

the maximum score you can achieve

.

Example 1:

Input:

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

Output:

6

Explanation:

We can rearrange the array into nums = [2,3,1,-1,-3,0,-3]. prefix = [2,5,6,5,2,2,-1], so the score is 6. It can be shown that 6 is the maximum score we can obtain.

Example 2:

Input:

nums = [-2,-3,0]

Output:

0

Explanation:

Any rearrangement of the array will result in a score of 0.

Constraints:

1 <= nums.length <= 10

5

-10

6

<= nums[i] <= 10

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

```
var maxScore = function(nums) {

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

```
}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-score nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Rearrange Array to Maximize Prefix Score
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 maxScore(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Rearrange Array to Maximize Prefix Score
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 maxScore(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Rearrange Array to Maximize Prefix Score
Difficulty: Medium
Tags: array, greedy, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Rearrange Array to Maximize Prefix Score
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 maxScore = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Rearrange Array to Maximize Prefix Score
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 maxScore(nums: number[]): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Rearrange Array to Maximize Prefix Score
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 MaxScore(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Rearrange Array to Maximize Prefix Score
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 maxScore(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Rearrange Array to Maximize Prefix Score
// Difficulty: Medium
```

```
// Tags: array, greedy, sort
//
// 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 maxScore(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Rearrange Array to Maximize Prefix Score
// Difficulty: Medium
// Tags: array, greedy, sort
//
// 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 max_score(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-score nums)
(-> (listof exact-integer?) exact-integer?)
)
```