Problem 2219: Maximum Sum Score of Array

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

of length

n

.

The

sum

score

of

nums

at an index

i

where

0 <= i < n

is the

maximum

of:

The sum of the

first

i + 1

elements of

nums

.

The sum of the

last

n - i

elements of

nums

.

Return

the

maximum

sum

score

of

nums

at any index.

Example 1:

Input:

nums = [4,3,-2,5]

Output:

10

Explanation:

The sum score at index 0 is max(4, 4 + 3 + -2 + 5) = max(4, 10) = 10. The sum score at index 1 is max(4 + 3, 3 + -2 + 5) = max(7, 6) = 7. The sum score at index 2 is max(4 + 3 + -2, -2 + 5) = max(5, 3) = 5. The sum score at index 3 is max(4 + 3 + -2 + 5, 5) = max(10, 5) = 10. The maximum sum score of nums is 10.

Example 2:

Input:

nums = [-3,-5]

Output:

-3

Explanation:

The sum score at index 0 is max(-3, -3 + -5) = max(-3, -8) = -3. The sum score at index 1 is max(-3 + -5, -5) = max(-8, -5) = -5. The maximum sum score of nums is -3.

Constraints:

n == nums.length

1 <= n <= 10

5

-10

5

<= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public long maximumSumScore(int[] nums) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long MaximumSumScore(int[] nums) {

}
}
```

C:

```
long long maximumSumScore(int* nums, int numsSize) {

}
```

Go:

```
func maximumSumScore(nums []int) int64 {

}
```

Kotlin:

```
class Solution {
fun maximumSumScore(nums: IntArray): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn maximum_sum_score(nums: Vec<i32>) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Maximum Sum Score of Array
* Difficulty: Medium
* Tags: array
*
* 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:
long long maximumSumScore(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Sum Score of Array
* Difficulty: Medium
* Tags: array
*
* 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 long maximumSumScore(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Sum Score of Array
Difficulty: Medium
Tags: array

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Sum Score of Array
* Difficulty: Medium
* Tags: array
*
* 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 maximumSumScore = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Sum Score of Array
* Difficulty: Medium
* Tags: array
*
* 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 maximumSumScore(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Sum Score of Array
* Difficulty: Medium
* Tags: array
*
* 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 long MaximumSumScore(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Sum Score of Array
* Difficulty: Medium
* Tags: array
*
* 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
*/

long long maximumSumScore(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Maximum Sum Score of Array
// Difficulty: Medium
// Tags: array
//
// 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 maximumSumScore(nums []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun maximumSumScore(nums: IntArray): Long {

}
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Maximum Sum Score of Array
// Difficulty: Medium
// Tags: array
//
// 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_sum_score(nums: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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