Problem 2593: Find Score of an Array After Marking All Elements

Problem Information

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

Problem Description

You are given an array

nums

consisting of positive integers.

Starting with

score = 0

, apply the following algorithm:

Choose the smallest integer of the array that is not marked. If there is a tie, choose the one with the smallest index.

Add the value of the chosen integer to

score

.

Mark

the chosen element and its two adjacent elements if they exist

.

Repeat until all the array elements are marked.

Return

the score you get after applying the above algorithm

.

Example 1:

Input:

nums = [2,1,3,4,5,2]

Output:

7

Explanation:

We mark the elements as follows: - 1 is the smallest unmarked element, so we mark it and its two adjacent elements: [

2

,

1

,

3

,4,5,2]. - 2 is the smallest unmarked element, so we mark it and its left adjacent element: [

2

,

1

,

3

,4,

5

,

2

]. - 4 is the only remaining unmarked element, so we mark it: [

2

,

1

,

3

,

4

,

5

,

2

]. Our score is 1 + 2 + 4 = 7.

Example 2:

Input:

nums = [2,3,5,1,3,2]

Output:

5

Explanation:

We mark the elements as follows: - 1 is the smallest unmarked element, so we mark it and its two adjacent elements: [2,3,

5

,

1

,

3

,2]. - 2 is the smallest unmarked element, since there are two of them, we choose the left-most one, so we mark the one at index 0 and its right adjacent element: [

2

,

3

,

5

,

1

,

3

,2]. - 2 is the only remaining unmarked element, so we mark it: [

2

,

3

,

5

,

1

,

3

,

2

]. Our score is 1 + 2 + 2 = 5.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

6

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find Score of an Array After Marking All Elements
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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:
long long findScore(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find Score of an Array After Marking All Elements
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 long findScore(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Score of an Array After Marking All Elements
Difficulty: Medium
Tags: array, hash, sort, queue, heap

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Score of an Array After Marking All Elements
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 findScore = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Score of an Array After Marking All Elements
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 findScore(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find Score of an Array After Marking All Elements
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* 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 long FindScore(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find Score of an Array After Marking All Elements
* Difficulty: Medium
* Tags: array, hash, sort, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
```

Go Solution:

```
// Problem: Find Score of an Array After Marking All Elements
// Difficulty: Medium
// Tags: array, hash, sort, queue, heap
//
// 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 findScore(nums []int) int64 {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Score of an Array After Marking All Elements
// Difficulty: Medium
// Tags: array, hash, sort, queue, heap
//
// 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 find_score(nums: Vec<i32>) -> i64 {

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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