Problem 2640: Find the Score of All Prefixes of an Array

Problem Information

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

Problem Description

We define the

conversion array

conver

of an array

arr

as follows:

conver[i] = arr[i] + max(arr[0..i])

where

max(arr[0..i])

is the maximum value of

arr[j]

over

0 <= j <= i

.

We also define the

score

of an array

arr

as the sum of the values of the conversion array of

arr

.

Given a

0-indexed

integer array

nums

of length

n

, return

an array

ans

of length

n

where

ans[i]

is the score of the prefix

nums[0..i]

.

Example 1:

Input:

nums = [2,3,7,5,10]

Output:

[4,10,24,36,56]

Explanation:

For the prefix [2], the conversion array is [4] hence the score is 4 For the prefix [2, 3], the conversion array is [4, 6] hence the score is 10 For the prefix [2, 3, 7], the conversion array is [4, 6, 14] hence the score is 24 For the prefix [2, 3, 7, 5], the conversion array is [4, 6, 14, 12] hence the score is 36 For the prefix [2, 3, 7, 5, 10], the conversion array is [4, 6, 14, 12, 20] hence the score is 56

Example 2:

Input:

nums = [1,1,2,4,8,16]

Output:

[2,4,8,16,32,64]

Explanation:

For the prefix [1], the conversion array is [2] hence the score is 2 For the prefix [1, 1], the conversion array is [2, 2] hence the score is 4 For the prefix [1, 1, 2], the conversion array is [2, 2, 4] hence the score is 8 For the prefix [1, 1, 2, 4], the conversion array is [2, 2, 4, 8] hence the score is 16 For the prefix [1, 1, 2, 4, 8], the conversion array is [2, 2, 4, 8, 16] hence the score is 32 For the prefix [1, 1, 2, 4, 8, 16], the conversion array is [2, 2, 4, 8, 16, 32] hence the score is 64

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
long long* findPrefixScore(int* nums, int numsSize, int* returnSize) {

}
```

Go:

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

}
```

Kotlin:

```
class Solution {
fun findPrefixScore(nums: IntArray): LongArray {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find the Score of All Prefixes of an 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:
vector<long long> findPrefixScore(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Score of All Prefixes of an 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[] findPrefixScore(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Score of All Prefixes of an 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 findPrefixScore(self, nums: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Score of All Prefixes of an 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 findPrefixScore = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Score of All Prefixes of an 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 findPrefixScore(nums: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Find the Score of All Prefixes of an 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[] FindPrefixScore(int[] nums) {

}
}
```

C Solution:

```c
/*
* Problem: Find the Score of All Prefixes of an 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
long long* findPrefixScore(int* nums, int numsSize, int* returnSize) {

}
```

Go Solution:

```go
// Problem: Find the Score of All Prefixes of an 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 findPrefixScore(nums []int) []int64 {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun findPrefixScore(nums: IntArray): LongArray {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Score of All Prefixes of an 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 find_prefix_score(nums: Vec<i32>) -> Vec<i64> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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