Problem 3221: Maximum Array Hopping Score II

Problem Information

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

Problem Description

Given an array

nums

, you have to get the

maximum

score starting from index 0 and

hopping

until you reach the last element of the array.

In each

hop

, you can jump from index

i

to an index

j > i

, and you get a

score

of

(j - i) * nums[j]

.

Return the

maximum score

you can get.

Example 1:

Input:

nums = [1,5,8]

Output:

16

Explanation:

There are two possible ways to reach the last element:

0 -> 1 -> 2

with a score of

(1 - 0) * 5 + (2 - 1) * 8 = 13

.

0 -> 2

with a score of

(2 - 0) * 8 = 16

.

Example 2:

Input:

nums = [4,5,2,8,9,1,3]

Output:

42

Explanation:

We can do the hopping

0 -> 4 -> 6

with a score of

(4 - 0) * 9 + (6 - 4) * 3 = 42

.

Constraints:

2 <= nums.length <= 10

5

1 <= nums[i] <= 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

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]): Long = {

}
}
```

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: Maximum Array Hopping Score II
* Difficulty: Medium
* Tags: array, greedy, stack
*
* 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 maxScore(vector<int>& nums) {
}
};
```

Java Solution:

```
/**
* Problem: Maximum Array Hopping Score II
* Difficulty: Medium
* Tags: array, greedy, stack
*
* 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 maxScore(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Array Hopping Score II
Difficulty: Medium
Tags: array, greedy, stack

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: Maximum Array Hopping Score II
* Difficulty: Medium
* Tags: array, greedy, stack
*
* 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: Maximum Array Hopping Score II
* Difficulty: Medium
* Tags: array, greedy, stack
*
* 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: Maximum Array Hopping Score II
* Difficulty: Medium
* Tags: array, greedy, stack
*
* 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 MaxScore(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Array Hopping Score II
* Difficulty: Medium
* Tags: array, greedy, stack
*
* 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 maxScore(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Maximum Array Hopping Score II
// Difficulty: Medium
// Tags: array, greedy, stack
//
// 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) int64 {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Array Hopping Score II
// Difficulty: Medium
// Tags: array, greedy, stack
//
// 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>) -> i64 {

}
}
```

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]): Long = {

}
}
```

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?)
)
```