Problem 2874: Maximum Value of an Ordered Triplet II

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

.

Return

the maximum value over all triplets of indices

(i, j, k)

such that

i < j < k

.

If all such triplets have a negative value, return

0

.

The

value of a triplet of indices

(i, j, k)

is equal to

(nums[i] - nums[j]) * nums[k]

.

Example 1:

Input:

nums = [12,6,1,2,7]

Output:

77

Explanation:

The value of the triplet (0, 2, 4) is (nums[0] - nums[2]) * nums[4] = 77. It can be shown that there are no ordered triplets of indices with a value greater than 77.

Example 2:

Input:

nums = [1,10,3,4,19]

Output:

133

Explanation:

The value of the triplet (1, 2, 4) is (nums[1] - nums[2]) * nums[4] = 133. It can be shown that there are no ordered triplets of indices with a value greater than 133.

Example 3:

Input:

nums = [1,2,3]

Output:

0

Explanation:

The only ordered triplet of indices (0, 1, 2) has a negative value of (nums[0] - nums[1]) * nums[2] = -3. Hence, the answer would be 0.

Constraints:

3 <= nums.length <= 10

5

1 <= nums[i] <= 10

6

Code Snippets

C++:

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

```
}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (maximum-triplet-value nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Value of an Ordered Triplet II
* 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 maximumTripletValue(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Value of an Ordered Triplet II
* 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 maximumTripletValue(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Value of an Ordered Triplet II
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 maximumTripletValue(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Value of an Ordered Triplet II
* 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 maximumTripletValue = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Value of an Ordered Triplet II
* 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 maximumTripletValue(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Value of an Ordered Triplet II
* 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 MaximumTripletValue(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Value of an Ordered Triplet II
* 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 maximumTripletValue(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Maximum Value of an Ordered Triplet II
// 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 maximumTripletValue(nums []int) int64 {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Value of an Ordered Triplet II
// 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_triplet_value(nums: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (maximum-triplet-value nums)
(-> (listof exact-integer?) exact-integer?)
)
```