Problem 3334: Find the Maximum Factor Score of Array

Problem Information

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

Problem Description

You are given an integer array

nums

.

The

factor score

of an array is defined as the

product

of the LCM and GCD of all elements of that array.

Return the

maximum factor score

of

nums

after removing

at most

one element from it.

Note

that

both

the

LCM

and

GCD

of a single number are the number itself, and the

factor score

of an

empty

array is 0.

Example 1:

Input:

nums = [2,4,8,16]

Output:

64

Explanation:

On removing 2, the GCD of the rest of the elements is 4 while the LCM is 16, which gives a maximum factor score of

4 * 16 = 64

.

Example 2:

Input:

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

Output:

60

Explanation:

The maximum factor score of 60 can be obtained without removing any elements.

Example 3:

Input:

nums = [3]

Output:

9

Constraints:

1 <= nums.length <= 100

1 <= nums[i] <= 30

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: Find the Maximum Factor Score of Array
* Difficulty: Medium
* Tags: array, math
*
* 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: Find the Maximum Factor Score of Array
* Difficulty: Medium
* Tags: array, math
*
* 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: Find the Maximum Factor Score of Array
Difficulty: Medium
Tags: array, math

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: Find the Maximum Factor Score of Array
* Difficulty: Medium
* Tags: array, math
*
* 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: Find the Maximum Factor Score of Array
* Difficulty: Medium
* Tags: array, math
*
* 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: Find the Maximum Factor Score of Array
* Difficulty: Medium
* Tags: array, math
*
* 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: Find the Maximum Factor Score of Array
* Difficulty: Medium
* Tags: array, math
*
* 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: Find the Maximum Factor Score of Array
// Difficulty: Medium
// Tags: array, math
//
// 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: Find the Maximum Factor Score of Array
// Difficulty: Medium
// Tags: array, math
//
// 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?)
)
```