Problem 1283: Find the Smallest Divisor Given a Threshold

Problem Information

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

Problem Description

Given an array of integers

nums

and an integer

threshold

, we will choose a positive integer

divisor

, divide all the array by it, and sum the division's result. Find the

smallest

divisor

such that the result mentioned above is less than or equal to

threshold

.

Each result of the division is rounded to the nearest integer greater than or equal to that element. (For example:

7/3 = 3

and

10/2 = 5

).

The test cases are generated so that there will be an answer.

Example 1:

Input:

nums = [1,2,5,9], threshold = 6

Output:

5

Explanation:

We can get a sum to 17 (1+2+5+9) if the divisor is 1. If the divisor is 4 we can get a sum of 7 (1+1+2+3) and if the divisor is 5 the sum will be 5 (1+1+1+2).

Example 2:

Input:

nums = [44,22,33,11,1], threshold = 5

Output:

44

Constraints:

1 <= nums.length <= 5 * 10

4

1 <= nums[i] <= 10

6

nums.length <= threshold <= 10

6

Code Snippets

C++:

```
class Solution {
public:
int smallestDivisor(vector<int>& nums, int threshold) {

}
};
```

Java:

```
class Solution {
public int smallestDivisor(int[] nums, int threshold) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def smallestDivisor(self, nums, threshold):
```

```
"""
:type nums: List[int]
:type threshold: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function smallestDivisor(nums: number[], threshold: number): number {

};
```

C#:

```
public class Solution {
public int SmallestDivisor(int[] nums, int threshold) {

}
}
```

C:

```
int smallestDivisor(int* nums, int numsSize, int threshold) {

}
```

Go:

```
func smallestDivisor(nums []int, threshold int) int {

}
```

Kotlin:

```
class Solution {
fun smallestDivisor(nums: IntArray, threshold: Int): Int {

}
}
```

Swift:

```
class Solution {
func smallestDivisor(_ nums: [Int], _ threshold: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn smallest_divisor(nums: Vec<i32>, threshold: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} threshold
# @return {Integer}
def smallest_divisor(nums, threshold)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $threshold
* @return Integer
*/
function smallestDivisor($nums, $threshold) {
```

```
}
}
```

Dart:

```
class Solution {
int smallestDivisor(List<int> nums, int threshold) {

}
}
```

Scala:

```
object Solution {
def smallestDivisor(nums: Array[Int], threshold: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smallest_divisor(nums :: [integer], threshold :: integer) :: integer
def smallest_divisor(nums, threshold) do

end
end
```

Erlang:

```
-spec smallest_divisor(Nums :: [integer()], Threshold :: integer()) ->
integer().
smallest_divisor(Nums, Threshold) ->
.
```

Racket:

```
(define/contract (smallest-divisor nums threshold)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Smallest Divisor Given a Threshold
* Difficulty: Medium
* Tags: array, search
*
* 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:
int smallestDivisor(vector<int>& nums, int threshold) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Smallest Divisor Given a Threshold
* Difficulty: Medium
* Tags: array, search
*
* 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 int smallestDivisor(int[] nums, int threshold) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Smallest Divisor Given a Threshold
Difficulty: Medium
Tags: array, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Smallest Divisor Given a Threshold
* Difficulty: Medium
* Tags: array, search
*
* 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
* @param {number} threshold
* @return {number}
*/
```

```
var smallestDivisor = function(nums, threshold) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Smallest Divisor Given a Threshold
* Difficulty: Medium
* Tags: array, search
*
* 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 smallestDivisor(nums: number[], threshold: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Smallest Divisor Given a Threshold
* Difficulty: Medium
* Tags: array, search
*
* 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 int SmallestDivisor(int[] nums, int threshold) {

}
}
```

C Solution:

```c
/*
* Problem: Find the Smallest Divisor Given a Threshold
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

int smallestDivisor(int* nums, int numsSize, int threshold) {

}
```

Go Solution:

```go
// Problem: Find the Smallest Divisor Given a Threshold
// Difficulty: Medium
// Tags: array, search
//
// 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 smallestDivisor(nums []int, threshold int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun smallestDivisor(nums: IntArray, threshold: Int): Int {

}
}
```

Swift Solution:

```swift
class Solution {
func smallestDivisor(_ nums: [Int], _ threshold: Int) -> Int {

}
```

```
}
```

Rust Solution:

```
// Problem: Find the Smallest Divisor Given a Threshold
// Difficulty: Medium
// Tags: array, search
//
// 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 smallest_divisor(nums: Vec<i32>, threshold: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} threshold
# @return {Integer}
def smallest_divisor(nums, threshold)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $threshold
* @return Integer
*/
function smallestDivisor($nums, $threshold) {

}
}
```

Dart Solution:

```
class Solution {
int smallestDivisor(List<int> nums, int threshold) {

}
}
```

Scala Solution:

```
object Solution {
def smallestDivisor(nums: Array[Int], threshold: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smallest_divisor(nums :: [integer], threshold :: integer) :: integer
def smallest_divisor(nums, threshold) do

end
end
```

Erlang Solution:

```
-spec smallest_divisor(Nums :: [integer()], Threshold :: integer()) ->
integer().
smallest_divisor(Nums, Threshold) ->
.
```

Racket Solution:

```
(define/contract (smallest-divisor nums threshold)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```