Problem 2971: Find Polygon With the Largest Perimeter

Problem Information

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

Problem Description

You are given an array of

positive

integers

nums

of length

n

.

A

polygon

is a closed plane figure that has at least

3

sides. The

longest side

of a polygon is

smaller

than the sum of its other sides.

Conversely, if you have

k

(

k >= 3

)

positive

real numbers

a

1

,

a

2

,

a

3

, ...,

a

k

where

a

1

<= a

2

<= a

3

<= ... <= a

k

and

a

1

+ a

2

+ a

3

+ ... + a

k-1

> a

k

, then there

always

exists a polygon with

k

sides whose lengths are

a

1

,

a

2

,

a

3

, ...,

a

k

.

The

perimeter

of a polygon is the sum of lengths of its sides.

Return

the

largest

possible

perimeter

of a

polygon

whose sides can be formed from

nums

,

or

-1

if it is not possible to create a polygon

.

Example 1:

Input:

nums = [5,5,5]

Output:

15

Explanation:

The only possible polygon that can be made from nums has 3 sides: 5, 5, and 5. The perimeter is 5 + 5 + 5 = 15.

Example 2:

Input:

nums = [1,12,1,2,5,50,3]

Output:

12

Explanation:

The polygon with the largest perimeter which can be made from nums has 5 sides: 1, 1, 2, 3, and 5. The perimeter is 1 + 1 + 2 + 3 + 5 = 12. We cannot have a polygon with either 12 or 50 as the longest side because it is not possible to include 2 or more smaller sides that have a greater sum than either of them. It can be shown that the largest possible perimeter is 12.

Example 3:

Input:

nums = [5,5,50]

Output:

-1

Explanation:

There is no possible way to form a polygon from nums, as a polygon has at least 3 sides and 50 > 5 + 5.

Constraints:

3 <= n <= 10

5

1 <= nums[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (largest-perimeter nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Polygon With the Largest Perimeter
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 largestPerimeter(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Find Polygon With the Largest Perimeter
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 largestPerimeter(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Polygon With the Largest Perimeter
Difficulty: Medium
Tags: array, greedy, sort
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find Polygon With the Largest Perimeter
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 largestPerimeter = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Polygon With the Largest Perimeter
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 largestPerimeter(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find Polygon With the Largest Perimeter
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 LargestPerimeter(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Find Polygon With the Largest Perimeter
* Difficulty: Medium
* Tags: array, greedy, sort
*
* 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 largestPerimeter(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Find Polygon With the Largest Perimeter
// Difficulty: Medium
// Tags: array, greedy, sort
//
// 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 largestPerimeter(nums []int) int64 {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find Polygon With the Largest Perimeter
// Difficulty: Medium
// Tags: array, greedy, sort
```

```
//
// 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 largest_perimeter(nums: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (largest-perimeter nums)
(-> (listof exact-integer?) exact-integer?)
)
```