Problem 976: Largest Perimeter Triangle

Problem Information

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

Problem Description

Given an integer array

nums

, return

the largest perimeter of a triangle with a non-zero area, formed from three of these lengths

. If it is impossible to form any triangle of a non-zero area, return

0

.

Example 1:

Input:

nums = [2,1,2]

Output:

5

Explanation:

You can form a triangle with three side lengths: 1, 2, and 2.

Example 2:

Input:

nums = [1,2,1,10]

Output:

0

Explanation:

You cannot use the side lengths 1, 1, and 2 to form a triangle. You cannot use the side lengths 1, 1, and 10 to form a triangle. You cannot use the side lengths 1, 2, and 10 to form a triangle. As we cannot use any three side lengths to form a triangle of non-zero area, we return 0.

Constraints:

3 <= nums.length <= 10

4

1 <= nums[i] <= 10

6

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

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]): Int = {

}
}
```

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: Largest Perimeter Triangle
* Difficulty: Easy
* Tags: array, greedy, math, 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:
int largestPerimeter(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Perimeter Triangle
* Difficulty: Easy
* Tags: array, greedy, math, 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 int largestPerimeter(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Perimeter Triangle
Difficulty: Easy
Tags: array, greedy, math, 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: Largest Perimeter Triangle
* Difficulty: Easy
* Tags: array, greedy, math, 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: Largest Perimeter Triangle
* Difficulty: Easy
* Tags: array, greedy, math, 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: Largest Perimeter Triangle
* Difficulty: Easy
* Tags: array, greedy, math, 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 int LargestPerimeter(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Largest Perimeter Triangle
* Difficulty: Easy
* Tags: array, greedy, math, 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
*/

int largestPerimeter(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Largest Perimeter Triangle
// Difficulty: Easy
// Tags: array, greedy, math, 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) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Largest Perimeter Triangle
// Difficulty: Easy
// Tags: array, greedy, math, 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>) -> i32 {

}
}
```

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]): Int = {

}
}
```

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?)
)
```