Problem 1732: Find the Highest Altitude

Problem Information

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

Problem Description

There is a biker going on a road trip. The road trip consists of

n + 1

points at different altitudes. The biker starts his trip on point

0

with altitude equal

0

.

You are given an integer array

gain

of length

n

where

gain[i]

is the

net gain in altitude

between points

i

and

i + 1

for all (

0 <= i < n)

. Return

the

highest altitude

of a point.

Example 1:

Input:

gain = [-5,1,5,0,-7]

Output:

1

Explanation:

The altitudes are [0,-5,-4,1,1,-6]. The highest is 1.

Example 2:

Input:

gain = [-4,-3,-2,-1,4,3,2]

Output:

0

Explanation:

The altitudes are [0,-4,-7,-9,-10,-6,-3,-1]. The highest is 0.

Constraints:

n == gain.length

1 <= n <= 100

-100 <= gain[i] <= 100

Code Snippets

C++:

```
class Solution {
public:
int largestAltitude(vector<int>& gain) {

}
};
```

Java:

```
class Solution {
public int largestAltitude(int[] gain) {

}
}
```

Python3:

```
class Solution:
def largestAltitude(self, gain: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function largestAltitude(gain: number[]): number {

};
```

C#:

```
public class Solution {
public int LargestAltitude(int[] gain) {

}
}
```

C:

```
int largestAltitude(int* gain, int gainSize) {

}
```

Go:

```
func largestAltitude(gain []int) int {

}
```

Kotlin:

```
class Solution {
fun largestAltitude(gain: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func largestAltitude(_ gain: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn largest_altitude(gain: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} gain
# @return {Integer}
def largest_altitude(gain)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $gain
* @return Integer
*/
function largestAltitude($gain) {

}
}
```

Dart:

```
class Solution {
int largestAltitude(List<int> gain) {

}
}
```

Scala:

```
object Solution {
def largestAltitude(gain: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_altitude(gain :: [integer]) :: integer
def largest_altitude(gain) do

end
end
```

Erlang:

```
-spec largest_altitude(Gain :: [integer()]) -> integer().
largest_altitude(Gain) ->
.
```

Racket:

```
(define/contract (largest-altitude gain)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Highest Altitude
* Difficulty: Easy
* 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:
int largestAltitude(vector<int>& gain) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Highest Altitude
* Difficulty: Easy
* 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 int largestAltitude(int[] gain) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Highest Altitude
Difficulty: Easy
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 largestAltitude(self, gain: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Highest Altitude
* Difficulty: Easy
* 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[]} gain
* @return {number}
*/
var largestAltitude = function(gain) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Highest Altitude
* Difficulty: Easy
* 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 largestAltitude(gain: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Find the Highest Altitude
* Difficulty: Easy
* 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 int LargestAltitude(int[] gain) {

}
}
```

C Solution:

```
/*
* Problem: Find the Highest Altitude
* Difficulty: Easy
* 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
*/

int largestAltitude(int* gain, int gainSize) {

}
```

Go Solution:

```
// Problem: Find the Highest Altitude
// Difficulty: Easy
// 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 largestAltitude(gain []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largestAltitude(gain: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func largestAltitude(_ gain: [Int]) -> Int {
}
}
```

Rust Solution:

```
// Problem: Find the Highest Altitude
// Difficulty: Easy
// 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 largest_altitude(gain: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} gain
# @return {Integer}
def largest_altitude(gain)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $gain
* @return Integer
*/
function largestAltitude($gain) {

}
}
```

Dart Solution:

```
class Solution {
int largestAltitude(List<int> gain) {

}
}
```

Scala Solution:

```
object Solution {
def largestAltitude(gain: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_altitude(gain :: [integer]) :: integer
def largest_altitude(gain) do

end
end
```

Erlang Solution:

```
-spec largest_altitude(Gain :: [integer()]) -> integer().
largest_altitude(Gain) ->
.
```

Racket Solution:

```
(define/contract (largest-altitude gain)
(-> (listof exact-integer?) exact-integer?)
)
```