Problem 3730: Maximum Calories Burnt from Jumps

Problem Information

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

Problem Description

You are given an integer array

heights

of size

n

, where

heights[i]

represents the height of the

i

th

block in an exercise routine.

You start on the ground (height 0) and

must

jump onto each block

exactly once

in any order.

The

calories burned

for a jump from a block of height

a

to a block of height

b

is

(a - b)

2

.

The

calories burned

for the first jump from the ground to the chosen first block

heights[i]

is

(0 - heights[i])

2

.

Return the

maximum

total calories you can burn by selecting an optimal jumping sequence.

Note:

Once you jump onto the first block, you cannot return to the ground.

Example 1:

Input:

heights = [1,7,9]

Output:

181

Explanation:

The optimal sequence is

[9, 1, 7]

.

Initial jump from the ground to

heights[2] = 9

:

(0 - 9)

2

= 81

.

Next jump to

heights[0] = 1

:

(9 - 1)

2

= 64

.

Final jump to

heights[1] = 7

:

(1 - 7)

2

= 36

.

Total calories burned =

81 + 64 + 36 = 181

.

Example 2:

Input:

heights = [5,2,4]

Output:

38

Explanation:

The optimal sequence is

[5, 2, 4]

.

Initial jump from the ground to

heights[0] = 5

:

(0 - 5)

2

= 25

.

Next jump to

heights[1] = 2

:

(5 - 2)

2

= 9

.

Final jump to

heights[2] = 4

:

(2 - 4)

2

= 4

.

Total calories burned =

25 + 9 + 4 = 38

.

Example 3:

Input:

heights = [3,3]

Output:

9

Explanation:

The optimal sequence is

[3, 3]

.

Initial jump from the ground to

heights[0] = 3

:

(0 - 3)

2

= 9

.

Next jump to

heights[1] = 3

:

(3 - 3)

2

= 0

.

Total calories burned =

9 + 0 = 9

.

Constraints:

1 <= n == heights.length <= 10

5

1 <= heights[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
long long maxCaloriesBurnt(vector<int>& heights) {

}
};
```

Java:

```
class Solution {
public long maxCaloriesBurnt(int[] heights) {

}
}
```

Python3:

```
class Solution:
def maxCaloriesBurnt(self, heights: List[int]) -> int:
```

Python:

```
class Solution(object):
def maxCaloriesBurnt(self, heights):
```

```
"""
:type heights: List[int]
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function maxCaloriesBurnt(heights: number[]): number {

};
```

C#:

```
public class Solution {
public long MaxCaloriesBurnt(int[] heights) {

}
}
```

C:

```
long long maxCaloriesBurnt(int* heights, int heightsSize) {

}
```

Go:

```
func maxCaloriesBurnt(heights []int) int64 {

}
```

Kotlin:

```
class Solution {
fun maxCaloriesBurnt(heights: IntArray): Long {

}
}
```

Swift:

```
class Solution {
func maxCaloriesBurnt(_ heights: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_calories_burnt(heights: Vec<i32>) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} heights
# @return {Integer}
def max_calories_burnt(heights)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $heights
* @return Integer
*/
function maxCaloriesBurnt($heights) {

}
}
```

Dart:

```
class Solution {
int maxCaloriesBurnt(List<int> heights) {

}
}
```

Scala:

```
object Solution {
def maxCaloriesBurnt(heights: Array[Int]): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_calories_burnt(heights :: [integer]) :: integer
def max_calories_burnt(heights) do

end
end
```

Erlang:

```
-spec max_calories_burnt(Heights :: [integer()]) -> integer().
max_calories_burnt(Heights) ->
.
```

Racket:

```
(define/contract (max-calories-burnt heights)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Calories Burnt from Jumps
* 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 maxCaloriesBurnt(vector<int>& heights) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Calories Burnt from Jumps
* 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 maxCaloriesBurnt(int[] heights) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Calories Burnt from Jumps
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 maxCaloriesBurnt(self, heights: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Calories Burnt from Jumps
* 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[]} heights
* @return {number}
*/
var maxCaloriesBurnt = function(heights) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Calories Burnt from Jumps
* 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 maxCaloriesBurnt(heights: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Calories Burnt from Jumps
* 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 MaxCaloriesBurnt(int[] heights) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Calories Burnt from Jumps
* 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 maxCaloriesBurnt(int* heights, int heightsSize) {

}
```

Go Solution:

```
// Problem: Maximum Calories Burnt from Jumps
// 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 maxCaloriesBurnt(heights []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun maxCaloriesBurnt(heights: IntArray): Long {

}
}
```

Swift Solution:

```
class Solution {
func maxCaloriesBurnt(_ heights: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Calories Burnt from Jumps
// 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 max_calories_burnt(heights: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} heights
# @return {Integer}
def max_calories_burnt(heights)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $heights
* @return Integer
*/
function maxCaloriesBurnt($heights) {

}
}
```

Dart Solution:

```
class Solution {
int maxCaloriesBurnt(List<int> heights) {

}
}
```

Scala Solution:

```
object Solution {
def maxCaloriesBurnt(heights: Array[Int]): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_calories_burnt(heights :: [integer]) :: integer
def max_calories_burnt(heights) do

end
end
```

Erlang Solution:

```
-spec max_calories_burnt(Heights :: [integer()]) -> integer().
max_calories_burnt(Heights) ->
.
```

Racket Solution:

```
(define/contract (max-calories-burnt heights)
(-> (listof exact-integer?) exact-integer?)
)
```