Problem 3296: Minimum Number of Seconds to Make Mountain Height Zero

Problem Information

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

Problem Description

You are given an integer

mountainHeight

denoting the height of a mountain.

You are also given an integer array

workerTimes

representing the work time of workers in

seconds

.

The workers work

simultaneously

to

reduce

the height of the mountain. For worker

i

:

To decrease the mountain's height by

x

, it takes

workerTimes[i] + workerTimes[i] * 2 + ... + workerTimes[i] * x

seconds. For example:

To reduce the height of the mountain by 1, it takes

workerTimes[i]

seconds.

To reduce the height of the mountain by 2, it takes

workerTimes[i] + workerTimes[i] * 2

seconds, and so on.

Return an integer representing the

minimum

number of seconds required for the workers to make the height of the mountain 0.

Example 1:

Input:

mountainHeight = 4, workerTimes = [2,1,1]

Output:

3

Explanation:

One way the height of the mountain can be reduced to 0 is:

Worker 0 reduces the height by 1, taking

workerTimes[0] = 2

seconds.

Worker 1 reduces the height by 2, taking

workerTimes[1] + workerTimes[1] * 2 = 3

seconds.

Worker 2 reduces the height by 1, taking

workerTimes[2] = 1

second.

Since they work simultaneously, the minimum time needed is

max(2, 3, 1) = 3

seconds.

Example 2:

Input:

mountainHeight = 10, workerTimes = [3,2,2,4]

Output:

12

Explanation:

Worker 0 reduces the height by 2, taking

workerTimes[0] + workerTimes[0] * 2 = 9

seconds.

Worker 1 reduces the height by 3, taking

workerTimes[1] + workerTimes[1] * 2 + workerTimes[1] * 3 = 12

seconds.

Worker 2 reduces the height by 3, taking

workerTimes[2] + workerTimes[2] * 2 + workerTimes[2] * 3 = 12

seconds.

Worker 3 reduces the height by 2, taking

workerTimes[3] + workerTimes[3] * 2 = 12

seconds.

The number of seconds needed is

max(9, 12, 12, 12) = 12

seconds.

Example 3:

Input:

mountainHeight = 5, workerTimes = [1]

Output:

15

Explanation:

There is only one worker in this example, so the answer is

workerTimes[0] + workerTimes[0] * 2 + workerTimes[0] * 3 + workerTimes[0] * 4 + workerTimes[0] * 5 = 15

.

Constraints:

1 <= mountainHeight <= 10

5

1 <= workerTimes.length <= 10

4

1 <= workerTimes[i] <= 10

6

Code Snippets

C++:

```
class Solution {
public:
long long minNumberOfSeconds(int mountainHeight, vector<int>& workerTimes) {

}
};
```

Java:

```
class Solution {
public long minNumberOfSeconds(int mountainHeight, int[] workerTimes) {

}
}
```

Python3:

```
class Solution:
def minNumberOfSeconds(self, mountainHeight: int, workerTimes: List[int]) ->
int:
```

Python:

```
class Solution(object):
def minNumberOfSeconds(self, mountainHeight, workerTimes):
"""
:type mountainHeight: int
:type workerTimes: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} mountainHeight
* @param {number[]} workerTimes
* @return {number}
*/
var minNumberOfSeconds = function(mountainHeight, workerTimes) {

};
```

TypeScript:

```
function minNumberOfSeconds(mountainHeight: number, workerTimes: number[]):
number {

};
```

C#:

```
public class Solution {
public long MinNumberOfSeconds(int mountainHeight, int[] workerTimes) {

}
}
```

C:

```
long long minNumberOfSeconds(int mountainHeight, int* workerTimes, int
workerTimesSize) {

}
```

Go:

```
func minNumberOfSeconds(mountainHeight int, workerTimes []int) int64 {

}
```

Kotlin:

```
class Solution {
fun minNumberOfSeconds(mountainHeight: Int, workerTimes: IntArray): Long {

}
}
```

Swift:

```
class Solution {
func minNumberOfSeconds(_ mountainHeight: Int, _ workerTimes: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_number_of_seconds(mountain_height: i32, worker_times: Vec<i32>) ->
i64 {
```

```
}
}
```

Ruby:

```
# @param {Integer} mountain_height
# @param {Integer[]} worker_times
# @return {Integer}
def min_number_of_seconds(mountain_height, worker_times)

end
```

PHP:

```
class Solution {

/**
* @param Integer $mountainHeight
* @param Integer[] $workerTimes
* @return Integer
*/
function minNumberOfSeconds($mountainHeight, $workerTimes) {

}
}
```

Dart:

```
class Solution {
int minNumberOfSeconds(int mountainHeight, List<int> workerTimes) {

}
}
```

Scala:

```
object Solution {
def minNumberOfSeconds(mountainHeight: Int, workerTimes: Array[Int]): Long =
{

}
}
```

Elixir:

```
defmodule Solution do
@spec min_number_of_seconds(mountain_height :: integer, worker_times ::
[integer]) :: integer
def min_number_of_seconds(mountain_height, worker_times) do

end
end
```

Erlang:

```
-spec min_number_of_seconds(MountainHeight :: integer(), WorkerTimes ::
[integer()]) -> integer().
min_number_of_seconds(MountainHeight, WorkerTimes) ->
.
```

Racket:

```
(define/contract (min-number-of-seconds mountainHeight workerTimes)
(-> exact-integer? (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Seconds to Make Mountain Height Zero
* Difficulty: Medium
* Tags: array, greedy, math, search, queue, heap
*
* 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 minNumberOfSeconds(int mountainHeight, vector<int>& workerTimes) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Seconds to Make Mountain Height Zero
* Difficulty: Medium
* Tags: array, greedy, math, search, queue, heap
*
* 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 minNumberOfSeconds(int mountainHeight, int[] workerTimes) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Seconds to Make Mountain Height Zero
Difficulty: Medium
Tags: array, greedy, math, search, queue, heap

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 minNumberOfSeconds(self, mountainHeight: int, workerTimes: List[int]) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minNumberOfSeconds(self, mountainHeight, workerTimes):
"""
:type mountainHeight: int
:type workerTimes: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Number of Seconds to Make Mountain Height Zero
* Difficulty: Medium
* Tags: array, greedy, math, search, queue, heap
*
* 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} mountainHeight
* @param {number[]} workerTimes
* @return {number}
*/
var minNumberOfSeconds = function(mountainHeight, workerTimes) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Seconds to Make Mountain Height Zero
* Difficulty: Medium
* Tags: array, greedy, math, search, queue, heap
*
* 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 minNumberOfSeconds(mountainHeight: number, workerTimes: number[]):
```

```
number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Seconds to Make Mountain Height Zero
* Difficulty: Medium
* Tags: array, greedy, math, search, queue, heap
*
* 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 MinNumberOfSeconds(int mountainHeight, int[] workerTimes) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Seconds to Make Mountain Height Zero
* Difficulty: Medium
* Tags: array, greedy, math, search, queue, heap
*
* 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 minNumberOfSeconds(int mountainHeight, int* workerTimes, int
workerTimesSize) {

}
```

Go Solution:

```
// Problem: Minimum Number of Seconds to Make Mountain Height Zero
// Difficulty: Medium
// Tags: array, greedy, math, search, queue, heap
//
// 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 minNumberOfSeconds(mountainHeight int, workerTimes []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun minNumberOfSeconds(mountainHeight: Int, workerTimes: IntArray): Long {

}
}
```

Swift Solution:

```
class Solution {
func minNumberOfSeconds(_ mountainHeight: Int, _ workerTimes: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Seconds to Make Mountain Height Zero
// Difficulty: Medium
// Tags: array, greedy, math, search, queue, heap
//
// 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 min_number_of_seconds(mountain_height: i32, worker_times: Vec<i32>) ->
i64 {
```

```
}
}
```

Ruby Solution:

```
# @param {Integer} mountain_height
# @param {Integer[]} worker_times
# @return {Integer}
def min_number_of_seconds(mountain_height, worker_times)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $mountainHeight
* @param Integer[] $workerTimes
* @return Integer
*/
function minNumberOfSeconds($mountainHeight, $workerTimes) {

}
}
```

Dart Solution:

```
class Solution {
int minNumberOfSeconds(int mountainHeight, List<int> workerTimes) {

}
}
```

Scala Solution:

```
object Solution {
def minNumberOfSeconds(mountainHeight: Int, workerTimes: Array[Int]): Long =
{

}
```

```
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_number_of_seconds(mountain_height :: integer, worker_times ::
[integer]) :: integer
def min_number_of_seconds(mountain_height, worker_times) do

end
end
```

Erlang Solution:

```
-spec min_number_of_seconds(MountainHeight :: integer(), WorkerTimes ::
[integer()]) -> integer().
min_number_of_seconds(MountainHeight, WorkerTimes) ->
.
```

Racket Solution:

```
(define/contract (min-number-of-seconds mountainHeight workerTimes)
(-> exact-integer? (listof exact-integer?) exact-integer?)
)
```