Problem 2798: Number of Employees Who Met the Target

Problem Information

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

Problem Description

There are

n

employees in a company, numbered from

0

to

n - 1

. Each employee

i

has worked for

hours[i]

hours in the company.

The company requires each employee to work for

at least

target

hours.

You are given a

0-indexed

array of non-negative integers

hours

of length

n

and a non-negative integer

target

.

Return

the integer denoting the number of employees who worked at least

target

hours

.

Example 1:

Input:

hours = [0,1,2,3,4], target = 2

Output:

3

Explanation:

The company wants each employee to work for at least 2 hours. - Employee 0 worked for 0 hours and didn't meet the target. - Employee 1 worked for 1 hours and didn't meet the target. - Employee 2 worked for 2 hours and met the target. - Employee 3 worked for 3 hours and met the target. - Employee 4 worked for 4 hours and met the target. There are 3 employees who met the target.

Example 2:

Input:

hours = [5,1,4,2,2], target = 6

Output:

0

Explanation:

The company wants each employee to work for at least 6 hours. There are 0 employees who met the target.

Constraints:

1 <= n == hours.length <= 50

0 <= hours[i], target <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int numberOfEmployeesWhoMetTarget(vector<int>& hours, int target) {

}
};
```

Java:

```
class Solution {
public int numberOfEmployeesWhoMetTarget(int[] hours, int target) {

}
}
```

Python3:

```
class Solution:
def numberOfEmployeesWhoMetTarget(self, hours: List[int], target: int) ->
int:
```

Python:

```
class Solution(object):
def numberOfEmployeesWhoMetTarget(self, hours, target):
"""
:type hours: List[int]
:type target: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} hours
* @param {number} target
* @return {number}
*/
var numberOfEmployeesWhoMetTarget = function(hours, target) {

};
```

TypeScript:

```
function numberOfEmployeesWhoMetTarget(hours: number[], target: number):
number {

};
```

C#:

```
public class Solution {
public int NumberOfEmployeesWhoMetTarget(int[] hours, int target) {

}
}
```

C:

```
int numberOfEmployeesWhoMetTarget(int* hours, int hoursSize, int target) {

}
```

Go:

```
func numberOfEmployeesWhoMetTarget(hours []int, target int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfEmployeesWhoMetTarget(hours: IntArray, target: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfEmployeesWhoMetTarget(_ hours: [Int], _ target: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_employees_who_met_target(hours: Vec<i32>, target: i32) ->
i32 {

}
}
```

Ruby:

```
# @param {Integer[]} hours
# @param {Integer} target
# @return {Integer}
def number_of_employees_who_met_target(hours, target)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $hours
* @param Integer $target
* @return Integer
*/
function numberOfEmployeesWhoMetTarget($hours, $target) {

}
}
```

Dart:

```
class Solution {
int numberOfEmployeesWhoMetTarget(List<int> hours, int target) {

}
}
```

Scala:

```
object Solution {
def numberOfEmployeesWhoMetTarget(hours: Array[Int], target: Int): Int = {
}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_employees_who_met_target(hours :: [integer], target ::
integer) :: integer
def number_of_employees_who_met_target(hours, target) do

end
end
```

Erlang:

```
-spec number_of_employees_who_met_target(Hours :: [integer()], Target ::
integer()) -> integer().
number_of_employees_who_met_target(Hours, Target) ->
.
```

Racket:

```
(define/contract (number-of-employees-who-met-target hours target)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Employees Who Met the Target
* 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 numberOfEmployeesWhoMetTarget(vector<int>& hours, int target) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Employees Who Met the Target
* 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 numberOfEmployeesWhoMetTarget(int[] hours, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Employees Who Met the Target
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 numberOfEmployeesWhoMetTarget(self, hours: List[int], target: int) ->
int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def numberOfEmployeesWhoMetTarget(self, hours, target):
"""
:type hours: List[int]
:type target: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Employees Who Met the Target
* 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[]} hours
* @param {number} target
* @return {number}
*/
var numberOfEmployeesWhoMetTarget = function(hours, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Employees Who Met the Target
* 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 numberOfEmployeesWhoMetTarget(hours: number[], target: number):
number {

};
```

C# Solution:

```
/*
* Problem: Number of Employees Who Met the Target
* 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 NumberOfEmployeesWhoMetTarget(int[] hours, int target) {

}
}
```

C Solution:

```
/*
* Problem: Number of Employees Who Met the Target
* 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 numberOfEmployeesWhoMetTarget(int* hours, int hoursSize, int target) {

}
```

Go Solution:

```go
// Problem: Number of Employees Who Met the Target
// 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 numberOfEmployeesWhoMetTarget(hours []int, target int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun numberOfEmployeesWhoMetTarget(hours: IntArray, target: Int): Int {

}
}
```

Swift Solution:

```swift
class Solution {
func numberOfEmployeesWhoMetTarget(_ hours: [Int], _ target: Int) -> Int {

}
}
```

Rust Solution:

```rust
// Problem: Number of Employees Who Met the Target
// 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 number_of_employees_who_met_target(hours: Vec<i32>, target: i32) ->
i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} hours
# @param {Integer} target
# @return {Integer}
def number_of_employees_who_met_target(hours, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $hours
* @param Integer $target
* @return Integer
*/
function numberOfEmployeesWhoMetTarget($hours, $target) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfEmployeesWhoMetTarget(List<int> hours, int target) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfEmployeesWhoMetTarget(hours: Array[Int], target: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_employees_who_met_target(hours :: [integer], target ::
integer) :: integer
def number_of_employees_who_met_target(hours, target) do

end
end
```

Erlang Solution:

```
-spec number_of_employees_who_met_target(Hours :: [integer()], Target ::
integer()) -> integer().
number_of_employees_who_met_target(Hours, Target) ->
.
```

Racket Solution:

```
(define/contract (number-of-employees-who-met-target hours target)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```