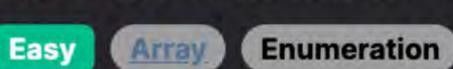
2798. Number of Employees Who Met the Target



Problem Description

In this problem, there is a company with n employees, each of whom is assigned a unique number from 0 to n - 1. We are provided with an array called hours, which is indexed from 0. The value at each index i in this array represents the number of hours employee i has worked. The company has also set a minimum required number of hours that each employee must meet or exceed, which is specified by a target variable.

Leetcode Link

The objective is to determine how many employees have worked for at least target hours. This is an example of a counting problem where we need to count the number of elements in an array that meet a particular condition.

Intuition

The intuition behind the solution is very straightforward. Since we need to count the number of employees who have met or exceeded the target hours, we can iterate through the hours array and compare each employee's hours with the target. Every time we find an employee whose hours are greater than or equal to target, we increment our count.

We arrive at the solution approach by recognizing that it's a direct application of the concept of iteration and comparison. Each employee's hours worked are independently compared to target, and if they satisfy the condition x >= target, they are included in our count.

In Python, this can be neatly expressed in a single line using a generator expression inside the sum() function, which iterates through each element x in hours and evaluates the condition x >= target. The sum() function effectively counts the number of True evaluations, which corresponds to the number of employees who have met the target hours.

Solution Approach

The implementation of the solution employs a simple approach that does not necessarily rely on complex algorithms, data structures, or patterns. It uses Python's built-in features to perform the operation efficiently and with clarity.

The solution is to use a generator expression inside the sum() function. A generator expression is a concise way to create a generator on the fly without the overhead of loops or the need to define a generator function. In this case, the expression (x >= target for x in hours) generates a sequence of boolean values (True or False). Each boolean value corresponds to whether an individual employee's hours worked is greater than or equal to the target.

The sum() function then takes this sequence of booleans and counts the number of True values. In Python, True is equivalent to 1 and False is equivalent to 0 when performing arithmetic operations, so summing a sequence of booleans effectively counts the number of True values.

To step through the code:

- hours is a list of integers, where each integer represents the hours worked by an employee.
- 2. target is an integer representing the target number of hours that employees should meet or exceed.

 3. The generator expression (x >= target for x in hours) iterates over each element x in hours.
- 4. For each x, it checks whether x >= target. This returns True if the condition is met, and False otherwise.
- 5. The sum() function adds up all the True values, effectively counting them.
- 6. The final returned value is the total count of employees who have worked at least target hours.
- or the final retained value to the total country of employees who have worked at least target hear

No additional data structures are needed for this approach, and its time complexity is O(n), where n is the number of employees, since it involves a single pass through the hours list.

Example Walkthrough

Let's use a small example to depict how the solution approach is applied to determine the number of employees who have met or exceeded the target number of hours.

has set a minimum required number of hours (target) to be 5.

Suppose we have a company with 5 employees and the following hours they have worked: hours = [6, 2, 9, 4, 7]. The company

1. We have our hours list: [6, 2, 9, 4, 7].

Now, let's walk through the solution step by step:

- 2. Our target is 5.
- 3. We construct a generator expression: (x >= target for x in hours) which we will use to iterate over the hours.
- 4. As we iterate through the hours, we compare each value (each employee's hours) with the target:
 6 >= 5 → True
- o 2 >= 5 → False

```
4 >= 5 → False
7 >= 5 → True

5. We then pass the generator expression to the sum() function, which will process the boolean values:

sum([True, False, True, False, True]) is the same as sum([1, 0, 1, 0, 1])
```

 \circ 9 >= 5 \rightarrow True

6. The sum() function counts the number of 1s (or True values), which is 3 in this case.

followed are not only intuitive but also concise and efficient, resulting in an elegant solution to our counting problem.

// Method to count the number of employees who have met or exceeded the target number of hours

public int numberOfEmployeesWhoMetTarget(int[] hoursWorked, int targetHours) {

int numberOfEmployeesWhoMetTarget(vector<int>& hours, int target) {

for (int workedHours : hours) {

// Iterate over the array of work hours.

if (hoursWorked >= targetHours) {

for (const hoursWorked of workHours) {

int count = 0; // Initialize the count of employees meeting the target

// Iterate over the vector containing hours worked by each employee

return count; // Return the total count of employees meeting the target.

sum. As a result, the space complexity is 0(1) using a constant amount of extra space.

1 class Solution: 2 def number_of_employees_who_met_target(self, hours: List[int], target: int) -> int: 3 # Initialize the count of employees who meet or exceed the target hours 4 count_met_target = 0

Python Solution

```
for employee_hours in hours:
    # If an employee meets or exceeds the target, increment the counter
    if employee_hours >= target:
        count_met_target += 1

# Return the total count of employees who met or exceeded the target hours
return count_met_target

Java Solution

1 class Solution {
```

The final output of this operation is 3, which means 3 employees have worked at least the target number of hours 5. The steps we

// Initialize a counter to keep track of the employees meeting the target int numberOfEmployeesMeetingTarget = 0;

```
// Iterate through the hours worked by each employee
           for (int hours : hoursWorked) {
               // If the employee has worked hours greater than or equal to the target, increment the counter
               if (hours >= targetHours) {
10
                   numberOfEmployeesMeetingTarget++;
11
13
14
15
           // Return the total count of employees who have met or exceeded the target hours
           return numberOfEmployeesMeetingTarget;
16
17
18 }
19
C++ Solution
1 class Solution {
2 public:
       // This function counts and returns the number of employees who have worked at least 'target' number of hours.
```


// If this employee's hours are greater than or equal to the target, increment the counter.

// If the current employee's hours are greater than or equal to the target, increment the count

13 } 14 } 15

count++;

Time and Space Complexity

Time Complexity:

10

16

18

17 }

The given code has a single list comprehension that iterates over the list of hours once and checks if each element is greater than or equal to the given target. This operation is O(n) where n is the number of elements in the hours list. Hence, the time complexity of the code is O(n).

the code is O(n).

Space Complexity:

This code does not use any additional space that scales with the size of the input except for a few variables to keep track of the