Problem 495: Teemo Attacking

Problem Information

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

Problem Description

Our hero Teemo is attacking an enemy Ashe with poison attacks! When Teemo attacks Ashe, Ashe gets poisoned for a exactly

duration

seconds. More formally, an attack at second

t

will mean Ashe is poisoned during the

inclusive

time interval

[t, t + duration - 1]

. If Teemo attacks again

before

the poison effect ends, the timer for it is

reset

, and the poison effect will end

duration

seconds after the new attack.

You are given a

non-decreasing

integer array

timeSeries

, where

timeSeries[i]

denotes that Teemo attacks Ashe at second

timeSeries[i]

, and an integer

duration

.

Return

the

total

number of seconds that Ashe is poisoned

.

Example 1:

Input:

timeSeries = [1,4], duration = 2

Output:

4

Explanation:

Teemo's attacks on Ashe go as follows: - At second 1, Teemo attacks, and Ashe is poisoned for seconds 1 and 2. - At second 4, Teemo attacks, and Ashe is poisoned for seconds 4 and 5. Ashe is poisoned for seconds 1, 2, 4, and 5, which is 4 seconds in total.

Example 2:

Input:

timeSeries = [1,2], duration = 2

Output:

3

Explanation:

Teemo's attacks on Ashe go as follows: - At second 1, Teemo attacks, and Ashe is poisoned for seconds 1 and 2. - At second 2 however, Teemo attacks again and resets the poison timer. Ashe is poisoned for seconds 2 and 3. Ashe is poisoned for seconds 1, 2, and 3, which is 3 seconds in total.

Constraints:

1 <= timeSeries.length <= 10

4

0 <= timeSeries[i], duration <= 10

7

timeSeries

is sorted in

non-decreasing

order.

Code Snippets

C++:

```
class Solution {
public:
int findPoisonedDuration(vector<int>& timeSeries, int duration) {

}
};
```

Java:

```
class Solution {
public int findPoisonedDuration(int[] timeSeries, int duration) {

}
}
```

Python3:

```
class Solution:
def findPoisonedDuration(self, timeSeries: List[int], duration: int) -> int:
```

Python:

```
class Solution(object):
def findPoisonedDuration(self, timeSeries, duration):
"""
:type timeSeries: List[int]
```

```
:type duration: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} timeSeries
* @param {number} duration
* @return {number}
*/
var findPoisonedDuration = function(timeSeries, duration) {

};
```

TypeScript:

```
function findPoisonedDuration(timeSeries: number[], duration: number): number
{

};
```

C#:

```
public class Solution {
public int FindPoisonedDuration(int[] timeSeries, int duration) {

}
}
```

C:

```
int findPoisonedDuration(int* timeSeries, int timeSeriesSize, int duration) {

}
```

Go:

```
func findPoisonedDuration(timeSeries []int, duration int) int {

}
```

Kotlin:

```
class Solution {
fun findPoisonedDuration(timeSeries: IntArray, duration: Int): Int {

}
}
```

Swift:

```
class Solution {
func findPoisonedDuration(_ timeSeries: [Int], _ duration: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn find_poisoned_duration(time_series: Vec<i32>, duration: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} time_series
# @param {Integer} duration
# @return {Integer}
def find_poisoned_duration(time_series, duration)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $timeSeries
* @param Integer $duration
* @return Integer
*/
function findPoisonedDuration($timeSeries, $duration) {



}
}
```

Dart:

```
class Solution {
int findPoisonedDuration(List<int> timeSeries, int duration) {

}
}
```

Scala:

```
object Solution {
def findPoisonedDuration(timeSeries: Array[Int], duration: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec find_poisoned_duration(time_series :: [integer], duration :: integer)
:: integer
def find_poisoned_duration(time_series, duration) do

end
end
```

Erlang:

```
-spec find_poisoned_duration(TimeSeries :: [integer()], Duration ::
integer()) -> integer().
find_poisoned_duration(TimeSeries, Duration) ->
.
```

Racket:

```
(define/contract (find-poisoned-duration timeSeries duration)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Teemo Attacking
* Difficulty: Easy
* Tags: array, 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:
int findPoisonedDuration(vector<int>& timeSeries, int duration) {

}
};
```

Java Solution:

```
/**
* Problem: Teemo Attacking
* Difficulty: Easy
* Tags: array, 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 int findPoisonedDuration(int[] timeSeries, int duration) {

}
}
```

Python3 Solution:

```
"""
Problem: Teemo Attacking
Difficulty: Easy
Tags: array, 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 findPoisonedDuration(self, timeSeries: List[int], duration: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def findPoisonedDuration(self, timeSeries, duration):
"""
:type timeSeries: List[int]
:type duration: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Teemo Attacking
* Difficulty: Easy
* Tags: array, 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[]} timeSeries
* @param {number} duration
* @return {number}
*/
```

```
var findPoisonedDuration = function(timeSeries, duration) {

};
```

TypeScript Solution:

```
/**
* Problem: Teemo Attacking
* Difficulty: Easy
* Tags: array, 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 findPoisonedDuration(timeSeries: number[], duration: number): number
{

};
```

C# Solution:

```
/*
* Problem: Teemo Attacking
* Difficulty: Easy
* Tags: array, 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 int FindPoisonedDuration(int[] timeSeries, int duration) {

}
}
```

C Solution:

```
/*
* Problem: Teemo Attacking
* Difficulty: Easy
* Tags: array, 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
*/

int findPoisonedDuration(int* timeSeries, int timeSeriesSize, int duration) {

}
```

Go Solution:

```
// Problem: Teemo Attacking
// Difficulty: Easy
// Tags: array, 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 findPoisonedDuration(timeSeries []int, duration int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findPoisonedDuration(timeSeries: IntArray, duration: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func findPoisonedDuration(_ timeSeries: [Int], _ duration: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Teemo Attacking
// Difficulty: Easy
// Tags: array, 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 find_poisoned_duration(time_series: Vec<i32>, duration: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} time_series
# @param {Integer} duration
# @return {Integer}
def find_poisoned_duration(time_series, duration)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $timeSeries
* @param Integer $duration
* @return Integer
*/
function findPoisonedDuration($timeSeries, $duration) {

}
}
```

Dart Solution:

```
class Solution {
int findPoisonedDuration(List<int> timeSeries, int duration) {

}
}
```

Scala Solution:

```
object Solution {
def findPoisonedDuration(timeSeries: Array[Int], duration: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec find_poisoned_duration(time_series :: [integer], duration :: integer)
:: integer
def find_poisoned_duration(time_series, duration) do

end
end
```

Erlang Solution:

```
-spec find_poisoned_duration(TimeSeries :: [integer()], Duration ::
integer()) -> integer().
find_poisoned_duration(TimeSeries, Duration) ->
.
```

Racket Solution:

```
(define/contract (find-poisoned-duration timeSeries duration)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```