Problem 1450: Number of Students Doing Homework at a Given Time

Problem Information

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

Problem Description

Given two integer arrays

startTime

and

endTime

and given an integer

queryTime

.

The

ith

student started doing their homework at the time

startTime[i]

and finished it at time

endTime[i]

.

Return

the number of students

doing their homework at time

queryTime

. More formally, return the number of students where

queryTime

lays in the interval

[startTime[i], endTime[i]]

inclusive.

Example 1:

Input:

startTime = [1,2,3], endTime = [3,2,7], queryTime = 4

Output:

1

Explanation:

We have 3 students where: The first student started doing homework at time 1 and finished at time 3 and wasn't doing anything at time 4. The second student started doing homework at time 2 and finished at time 2 and also wasn't doing anything at time 4. The third student started doing homework at time 3 and finished at time 7 and was the only student doing homework at time 4.

Example 2:

Input:

startTime = [4], endTime = [4], queryTime = 4

Output:

1

Explanation:

The only student was doing their homework at the queryTime.

Constraints:

startTime.length == endTime.length

1 <= startTime.length <= 100

1 <= startTime[i] <= endTime[i] <= 1000

1 <= queryTime <= 1000

Code Snippets

C++:

```
class Solution {
public:
int busyStudent(vector<int>& startTime, vector<int>& endTime, int queryTime)
{

}
};
```

Java:

```
class Solution {
public int busyStudent(int[] startTime, int[] endTime, int queryTime) {
```

```
}
}
```

Python3:

```
class Solution:
def busyStudent(self, startTime: List[int], endTime: List[int], queryTime:
int) -> int:
```

Python:

```
class Solution(object):
def busyStudent(self, startTime, endTime, queryTime):
"""
:type startTime: List[int]
:type endTime: List[int]
:type queryTime: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} startTime
* @param {number[]} endTime
* @param {number} queryTime
* @return {number}
*/
var busyStudent = function(startTime, endTime, queryTime) {

};
```

TypeScript:

```
function busyStudent(startTime: number[], endTime: number[], queryTime:
number): number {

};
```

C#:

```
public class Solution {
public int BusyStudent(int[] startTime, int[] endTime, int queryTime) {

}
}
```

C:

```
int busyStudent(int* startTime, int startTimeSize, int* endTime, int
endTimeSize, int queryTime) {

}
```

Go:

```
func busyStudent(startTime []int, endTime []int, queryTime int) int {

}
```

Kotlin:

```
class Solution {
fun busyStudent(startTime: IntArray, endTime: IntArray, queryTime: Int): Int
{

}
}
```

Swift:

```
class Solution {
func busyStudent(_ startTime: [Int], _ endTime: [Int], _ queryTime: Int) ->
Int {

}
}
```

Rust:

```
impl Solution {
pub fn busy_student(start_time: Vec<i32>, end_time: Vec<i32>, query_time:
i32) -> i32 {
```

```
}
}
```

Ruby:

```
# @param {Integer[]} start_time
# @param {Integer[]} end_time
# @param {Integer} query_time
# @return {Integer}
def busy_student(start_time, end_time, query_time)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $startTime
* @param Integer[] $endTime
* @param Integer $queryTime
* @return Integer
*/
function busyStudent($startTime, $endTime, $queryTime) {

}
}
```

Dart:

```
class Solution {
int busyStudent(List<int> startTime, List<int> endTime, int queryTime) {

}
}
```

Scala:

```
object Solution {
def busyStudent(startTime: Array[Int], endTime: Array[Int], queryTime: Int):
Int = {
```

```
}
}
```

Elixir:

```
defmodule Solution do
@spec busy_student(start_time :: [integer], end_time :: [integer], query_time
:: integer) :: integer
def busy_student(start_time, end_time, query_time) do

end
end
```

Erlang:

```
-spec busy_student(StartTime :: [integer()], EndTime :: [integer()],
QueryTime :: integer()) -> integer().
busy_student(StartTime, EndTime, QueryTime) ->
.
```

Racket:

```
(define/contract (busy-student startTime endTime queryTime)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?
exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Students Doing Homework at a Given Time
* 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 busyStudent(vector<int>& startTime, vector<int>& endTime, int queryTime)
{

}
};
```

Java Solution:

```
/**
* Problem: Number of Students Doing Homework at a Given Time
* 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 busyStudent(int[] startTime, int[] endTime, int queryTime) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Students Doing Homework at a Given Time
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 busyStudent(self, startTime: List[int], endTime: List[int], queryTime:
```

```
int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def busyStudent(self, startTime, endTime, queryTime):
"""
:type startTime: List[int]
:type endTime: List[int]
:type queryTime: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Students Doing Homework at a Given Time
* 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[]} startTime
* @param {number[]} endTime
* @param {number} queryTime
* @return {number}
*/
var busyStudent = function(startTime, endTime, queryTime) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Students Doing Homework at a Given Time
* 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 busyStudent(startTime: number[], endTime: number[], queryTime:
number): number {

};
```

C# Solution:

```
/*
* Problem: Number of Students Doing Homework at a Given Time
* 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 BusyStudent(int[] startTime, int[] endTime, int queryTime) {

}
}
```

C Solution:

```
/*
* Problem: Number of Students Doing Homework at a Given Time
* 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 busyStudent(int* startTime, int startTimeSize, int* endTime, int
endTimeSize, int queryTime) {

}
```

Go Solution:

```
// Problem: Number of Students Doing Homework at a Given Time
// 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 busyStudent(startTime []int, endTime []int, queryTime int) int {

}
```

Kotlin Solution:

```
class Solution {
fun busyStudent(startTime: IntArray, endTime: IntArray, queryTime: Int): Int
{

}
}
```

Swift Solution:

```
class Solution {
func busyStudent(_ startTime: [Int], _ endTime: [Int], _ queryTime: Int) ->
Int {

}
}
```

Rust Solution:

```
// Problem: Number of Students Doing Homework at a Given Time
// 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 busy_student(start_time: Vec<i32>, end_time: Vec<i32>, query_time:
i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} start_time
# @param {Integer[]} end_time
# @param {Integer} query_time
# @return {Integer}
def busy_student(start_time, end_time, query_time)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $startTime
* @param Integer[] $endTime
* @param Integer $queryTime
* @return Integer
*/
function busyStudent($startTime, $endTime, $queryTime) {

}
}
```

Dart Solution:

```
class Solution {
int busyStudent(List<int> startTime, List<int> endTime, int queryTime) {

}
}
```

Scala Solution:

```
object Solution {
def busyStudent(startTime: Array[Int], endTime: Array[Int], queryTime: Int):
Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec busy_student(start_time :: [integer], end_time :: [integer], query_time
:: integer) :: integer
def busy_student(start_time, end_time, query_time) do

end
end
```

Erlang Solution:

```
-spec busy_student(StartTime :: [integer()], EndTime :: [integer()],
QueryTime :: integer()) -> integer().
busy_student(StartTime, EndTime, QueryTime) ->
.
```

Racket Solution:

```
(define/contract (busy-student startTime endTime queryTime)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?
exact-integer?)
)
```