Problem 1010: Pairs of Songs With Total Durations Divisible by 60

Problem Information

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

Problem Description

You are given a list of songs where the

i

th

song has a duration of

time[i]

seconds.

Return

the number of pairs of songs for which their total duration in seconds is divisible by

60

. Formally, we want the number of indices

i

,

j

such that

i < j

with

(time[i] + time[j]) % 60 == 0

.

Example 1:

Input:

time = [30,20,150,100,40]

Output:

3

Explanation:

Three pairs have a total duration divisible by 60: (time[0] = 30, time[2] = 150): total duration 180 (time[1] = 20, time[3] = 100): total duration 120 (time[1] = 20, time[4] = 40): total duration 60

Example 2:

Input:

time = [60,60,60]

Output:

3

Explanation:

All three pairs have a total duration of 120, which is divisible by 60.

Constraints:

1 <= time.length <= 6 * 10

4

1 <= time[i] <= 500

Code Snippets

C++:

```
class Solution {
public:
int numPairsDivisibleBy60(vector<int>& time) {

}
};
```

Java:

```
class Solution {
public int numPairsDivisibleBy60(int[] time) {

}
}
```

Python3:

```
class Solution:
def numPairsDivisibleBy60(self, time: List[int]) -> int:
```

Python:

```
class Solution(object):
def numPairsDivisibleBy60(self, time):
"""
:type time: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} time
* @return {number}
*/
var numPairsDivisibleBy60 = function(time) {

};
```

TypeScript:

```
function numPairsDivisibleBy60(time: number[]): number {

};
```

C#:

```
public class Solution {
public int NumPairsDivisibleBy60(int[] time) {

}
}
```

C:

```
int numPairsDivisibleBy60(int* time, int timeSize) {

}
```

Go:

```
func numPairsDivisibleBy60(time []int) int {

}
```

Kotlin:

```
class Solution {
fun numPairsDivisibleBy60(time: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func numPairsDivisibleBy60(_ time: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_pairs_divisible_by60(time: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} time
# @return {Integer}
def num_pairs_divisible_by60(time)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $time
* @return Integer
*/
function numPairsDivisibleBy60($time) {

}
}
```

Dart:

```
class Solution {
int numPairsDivisibleBy60(List<int> time) {

}
```

```
}
```

Scala:

```
object Solution {
def numPairsDivisibleBy60(time: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_pairs_divisible_by60(time :: [integer]) :: integer
def num_pairs_divisible_by60(time) do

end
end
```

Erlang:

```
-spec num_pairs_divisible_by60(Time :: [integer()]) -> integer().
num_pairs_divisible_by60(Time) ->
.
```

Racket:

```
(define/contract (num-pairs-divisible-by60 time)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Pairs of Songs With Total Durations Divisible by 60
* Difficulty: Medium
* Tags: array, hash
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int numPairsDivisibleBy60(vector<int>& time) {

}
};
```

Java Solution:

```
/**
* Problem: Pairs of Songs With Total Durations Divisible by 60
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int numPairsDivisibleBy60(int[] time) {

}
}
```

Python3 Solution:

```
"""
Problem: Pairs of Songs With Total Durations Divisible by 60
Difficulty: Medium
Tags: array, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""
```

```
class Solution:
def numPairsDivisibleBy60(self, time: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numPairsDivisibleBy60(self, time):
"""
:type time: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Pairs of Songs With Total Durations Divisible by 60
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number[]} time
* @return {number}
*/
var numPairsDivisibleBy60 = function(time) {

};
```

TypeScript Solution:

```
/**
* Problem: Pairs of Songs With Total Durations Divisible by 60
* Difficulty: Medium
* Tags: array, hash
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function numPairsDivisibleBy60(time: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Pairs of Songs With Total Durations Divisible by 60
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int NumPairsDivisibleBy60(int[] time) {

}
}
```

C Solution:

```
/*
* Problem: Pairs of Songs With Total Durations Divisible by 60
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int numPairsDivisibleBy60(int* time, int timeSize) {
```

```
}
```

Go Solution:

```
// Problem: Pairs of Songs With Total Durations Divisible by 60
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func numPairsDivisibleBy60(time []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numPairsDivisibleBy60(time: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func numPairsDivisibleBy60(_ time: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Pairs of Songs With Total Durations Divisible by 60
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
```

```
// Space Complexity: O(n) for hash map

impl Solution {
pub fn num_pairs_divisible_by60(time: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} time
# @return {Integer}
def num_pairs_divisible_by60(time)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $time
* @return Integer
*/
function numPairsDivisibleBy60($time) {

}
}
```

Dart Solution:

```
class Solution {
int numPairsDivisibleBy60(List<int> time) {

}
}
```

Scala Solution:

```
object Solution {
def numPairsDivisibleBy60(time: Array[Int]): Int = {
```

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_pairs_divisible_by60(time :: [integer]) :: integer
def num_pairs_divisible_by60(time) do

end
end
```

Erlang Solution:

```
-spec num_pairs_divisible_by60(Time :: [integer()]) -> integer().
num_pairs_divisible_by60(Time) ->
.
```

Racket Solution:

```
(define/contract (num-pairs-divisible-by60 time)
(-> (listof exact-integer?) exact-integer?)
)
```