Problem 2437: Number of Valid Clock Times

Problem Information

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

Problem Description

You are given a string of length

5

called

time

, representing the current time on a digital clock in the format

"hh:mm"

. The

earliest

possible time is

"00:00"

and the

latest

possible time is

"23:59"

.

In the string

time

, the digits represented by the

?

symbol are

unknown

, and must be

replaced

with a digit from

0

to

9

.

Return

an integer

answer

, the number of valid clock times that can be created by replacing every

?

with a digit from

0

to

9

.

Example 1:

Input:

time = "?5:00"

Output:

2

Explanation:

We can replace the ? with either a 0 or 1, producing "05:00" or "15:00". Note that we cannot replace it with a 2, since the time "25:00" is invalid. In total, we have two choices.

Example 2:

Input:

time = "0?:0?"

Output:

100

Explanation:

Each ? can be replaced by any digit from 0 to 9, so we have 100 total choices.

Example 3:

Input:

time = "??:??"

Output:

1440

Explanation:

There are 24 possible choices for the hours, and 60 possible choices for the minutes. In total, we have 24 * 60 = 1440 choices.

Constraints:

time

is a valid string of length

5

in the format

"hh:mm"

.

"00" <= hh <= "23"

"00" <= mm <= "59"

Some of the digits might be replaced with

'?'

and need to be replaced with digits from

0

to

9

.

Code Snippets

C++:

```
class Solution {
public:
int countTime(string time) {

}
};
```

Java:

```
class Solution {
public int countTime(String time) {

}
}
```

Python3:

```
class Solution:
def countTime(self, time: str) -> int:
```

Python:

```
class Solution(object):
def countTime(self, time):
"""
:type time: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} time
* @return {number}
*/
var countTime = function(time) {

};
```

TypeScript:

```
function countTime(time: string): number {

};
```

C#:

```
public class Solution {
public int CountTime(string time) {

}
}
```

C:

```
int countTime(char* time) {

}
```

Go:

```
func countTime(time string) int {

}
```

Kotlin:

```
class Solution {
fun countTime(time: String): Int {

}
}
```

Swift:

```
class Solution {
func countTime(_ time: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_time(time: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} time
# @return {Integer}
def count_time(time)

end
```

PHP:

```
class Solution {

/**
* @param String $time
* @return Integer
*/
function countTime($time) {

}
}
```

Dart:

```
class Solution {
int countTime(String time) {

}
```

```
}
```

Scala:

```
object Solution {
def countTime(time: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_time(time :: String.t) :: integer
def count_time(time) do

end
end
```

Erlang:

```
-spec count_time(Time :: unicode:unicode_binary()) -> integer().
count_time(Time) ->
.
```

Racket:

```
(define/contract (count-time time)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Valid Clock Times
* Difficulty: Easy
* Tags: string
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int countTime(string time) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Valid Clock Times
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int countTime(String time) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Valid Clock Times
Difficulty: Easy
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""
```

```
class Solution:
def countTime(self, time: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countTime(self, time):
"""
:type time: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Valid Clock Times
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} time
* @return {number}
*/
var countTime = function(time) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Valid Clock Times
* Difficulty: Easy
* Tags: string
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countTime(time: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Valid Clock Times
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int CountTime(string time) {

}
}
```

C Solution:

```
/*
* Problem: Number of Valid Clock Times
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int countTime(char* time) {
```

```
}
```

Go Solution:

```
// Problem: Number of Valid Clock Times
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countTime(time string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countTime(time: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func countTime(_ time: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Valid Clock Times
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
```

```
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_time(time: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} time
# @return {Integer}
def count_time(time)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $time
* @return Integer
*/
function countTime($time) {

}
}
```

Dart Solution:

```
class Solution {
int countTime(String time) {

}
}
```

Scala Solution:

```
object Solution {
def countTime(time: String): Int = {
```

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_time(time :: String.t) :: integer
def count_time(time) do

end
end
```

Erlang Solution:

```
-spec count_time(Time :: unicode:unicode_binary()) -> integer().
count_time(Time) ->
.
```

Racket Solution:

```
(define/contract (count-time time)
(-> string? exact-integer?)
)
```