Problem 681: Next Closest Time

Problem Information

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

Problem Description

Given a

time

represented in the format

"HH:MM"

, form the next closest time by reusing the current digits. There is no limit on how many times a digit can be reused.

You may assume the given input string is always valid. For example,

"01:34"

,

"12:09"

are all valid.

"1:34"

,

"12:9"

are all invalid.

Example 1:

Input:

time = "19:34"

Output:

"19:39"

Explanation:

The next closest time choosing from digits

1

,

9

,

3

,

4

, is

19:39

, which occurs 5 minutes later. It is not

19:33

, because this occurs 23 hours and 59 minutes later.

Example 2:

Input:

time = "23:59"

Output:

"22:22"

Explanation:

The next closest time choosing from digits

2

,

3

,

5

,

9

, is

22:22

. It may be assumed that the returned time is next day's time since it is smaller than the input time numerically.

Constraints:

time.length == 5

time

is a valid time in the form

"HH:MM"

.

0 <= HH < 24

0 <= MM < 60

Code Snippets

C++:

```
class Solution {
public:
string nextClosestTime(string time) {

}
};
```

Java:

```
class Solution {
public String nextClosestTime(String time) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function nextClosestTime(time: string): string {

};
```

C#:

```
public class Solution {
public string NextClosestTime(string time) {

}
}
```

C:

```
char* nextClosestTime(char* time) {

}
```

Go:

```
func nextClosestTime(time string) string {

}
```

Kotlin:

```
class Solution {
fun nextClosestTime(time: String): String {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}

}
```

Dart:

```
class Solution {
String nextClosestTime(String time) {

}
}
```

Scala:

```
object Solution {
def nextClosestTime(time: String): String = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec next_closest_time(Time :: unicode:unicode_binary()) ->
unicode:unicode_binary().
next_closest_time(Time) ->
.
```

Racket:

```
(define/contract (next-closest-time time)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Next Closest Time
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string nextClosestTime(string time) {

}
};
```

Java Solution:

```
/**
* Problem: Next Closest Time
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String nextClosestTime(String time) {

}
}
```

Python3 Solution:

```
"""
Problem: Next Closest Time
```

```
Difficulty: Medium
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Next Closest Time
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Next Closest Time
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function nextClosestTime(time: string): string {

};
```

C# Solution:

```
/*
* Problem: Next Closest Time
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string NextClosestTime(string time) {

}
}
```

C Solution:

```
/*
* Problem: Next Closest Time
* Difficulty: Medium
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* nextClosestTime(char* time) {

}
```

Go Solution:

```
// Problem: Next Closest Time
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func nextClosestTime(time string) string {

}
```

Kotlin Solution:

```
class Solution {
fun nextClosestTime(time: String): String {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Next Closest Time
// Difficulty: Medium
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
String nextClosestTime(String time) {

}
}
```

Scala Solution:

```
object Solution {
def nextClosestTime(time: String): String = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec next_closest_time(Time :: unicode:unicode_binary()) ->
unicode:unicode_binary().
next_closest_time(Time) ->
.
```

Racket Solution:

```
(define/contract (next-closest-time time)
(-> string? string?)
)
```