Problem 949: Largest Time for Given Digits

Problem Information

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

Problem Description

Given an array

arr

of 4 digits, find the latest 24-hour time that can be made using each digit

exactly once

.

24-hour times are formatted as

"HH:MM"

, where

HH

is between

00

and

23

, and

MM

is between

00

and

59

. The earliest 24-hour time is

00:00

, and the latest is

23:59

.

Return

the latest 24-hour time in

"HH:MM"

format

. If no valid time can be made, return an empty string.

Example 1:

Input:

arr = [1,2,3,4]

Output:

"23:41"

Explanation:

The valid 24-hour times are "12:34", "12:43", "13:24", "13:42", "14:23", "14:32", "21:34", "21:43", "23:14", and "23:41". Of these times, "23:41" is the latest.

Example 2:

Input:

arr = [5,5,5,5]

Output:

""

Explanation:

There are no valid 24-hour times as "55:55" is not valid.

Constraints:

arr.length == 4

0 <= arr[i] <= 9

Code Snippets

C++:

```
class Solution {
public:
string largestTimeFromDigits(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public String largestTimeFromDigits(int[] arr) {

}
}
```

Python3:

```
class Solution:
def largestTimeFromDigits(self, arr: List[int]) -> str:
```

Python:

```
class Solution(object):
def largestTimeFromDigits(self, arr):
"""
:type arr: List[int]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number[]} arr
* @return {string}
*/
var largestTimeFromDigits = function(arr) {

};
```

TypeScript:

```
function largestTimeFromDigits(arr: number[]): string {

};
```

C#:

```
public class Solution {
public string LargestTimeFromDigits(int[] arr) {


}
}
```

C:

```
char* largestTimeFromDigits(int* arr, int arrSize) {

}
```

Go:

```
func largestTimeFromDigits(arr []int) string {

}
```

Kotlin:

```
class Solution {
fun largestTimeFromDigits(arr: IntArray): String {

}
}
```

Swift:

```
class Solution {
func largestTimeFromDigits(_ arr: [Int]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn largest_time_from_digits(arr: Vec<i32>) -> String {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {String}
def largest_time_from_digits(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return String
*/
function largestTimeFromDigits($arr) {

}
}
```

Dart:

```
class Solution {
String largestTimeFromDigits(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def largestTimeFromDigits(arr: Array[Int]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_time_from_digits(arr :: [integer]) :: String.t
def largest_time_from_digits(arr) do

end
end
```

Erlang:

```
-spec largest_time_from_digits(Arr :: [integer()]) ->
unicode:unicode_binary().
largest_time_from_digits(Arr) ->
.
```

Racket:

```
(define/contract (largest-time-from-digits arr)
(-> (listof exact-integer?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Time for Given Digits
* Difficulty: Medium
* Tags: array, string
*
* 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:
string largestTimeFromDigits(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Time for Given Digits
* Difficulty: Medium
* Tags: array, string
*
```

```
* 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 String largestTimeFromDigits(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Time for Given Digits
Difficulty: Medium
Tags: array, string

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 largestTimeFromDigits(self, arr: List[int]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def largestTimeFromDigits(self, arr):
"""
:type arr: List[int]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Largest Time for Given Digits
* Difficulty: Medium
* Tags: array, string
*
* 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[]} arr
* @return {string}
*/
var largestTimeFromDigits = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Time for Given Digits
* Difficulty: Medium
* Tags: array, string
*
* 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 largestTimeFromDigits(arr: number[]): string {

};
```

C# Solution:

```
/*
* Problem: Largest Time for Given Digits
* Difficulty: Medium
* Tags: array, string
*
* 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 string LargestTimeFromDigits(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Largest Time for Given Digits
* Difficulty: Medium
* Tags: array, string
*
* 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
*/

char* largestTimeFromDigits(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Largest Time for Given Digits
// Difficulty: Medium
// Tags: array, string
//
// 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 largestTimeFromDigits(arr []int) string {

}
```

Kotlin Solution:

```
class Solution {
fun largestTimeFromDigits(arr: IntArray): String {

}
}
```

Swift Solution:

```
class Solution {
func largestTimeFromDigits(_ arr: [Int]) -> String {

}
}
```

Rust Solution:

```
// Problem: Largest Time for Given Digits
// Difficulty: Medium
// Tags: array, string
//
// 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 largest_time_from_digits(arr: Vec<i32>) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {String}
def largest_time_from_digits(arr)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[] $arr
* @return String
*/
function largestTimeFromDigits($arr) {

}
}
```

Dart Solution:

```
class Solution {
String largestTimeFromDigits(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def largestTimeFromDigits(arr: Array[Int]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_time_from_digits(arr :: [integer]) :: String.t
def largest_time_from_digits(arr) do

end
end
```

Erlang Solution:

```
-spec largest_time_from_digits(Arr :: [integer()]) ->
unicode:unicode_binary().
largest_time_from_digits(Arr) ->
.
```

Racket Solution:

```
(define/contract (largest-time-from-digits arr)
(-> (listof exact-integer?) string?)
)
```