Problem 556: Next Greater Element III

Problem Information

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

Problem Description

Given a positive integer

n

, find

the smallest integer which has exactly the same digits existing in the integer

n

and is greater in value than

n

. If no such positive integer exists, return

-1

.

Note

that the returned integer should fit in

32-bit integer

, if there is a valid answer but it does not fit in

32-bit integer

, return

-1

.

Example 1:

Input:

n = 12

Output:

21

Example 2:

Input:

n = 21

Output:

-1

Constraints:

1 <= n <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
int nextGreaterElement(int n) {

}
};
```

Java:

```
class Solution {
public int nextGreaterElement(int n) {

}
}
```

Python3:

```
class Solution:
def nextGreaterElement(self, n: int) -> int:
```

Python:

```
class Solution(object):
def nextGreaterElement(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var nextGreaterElement = function(n) {

};
```

}

TypeScript:

```
function nextGreaterElement(n: number): number {

};
```

C#:

```
public class Solution {
public int NextGreaterElement(int n) {

}
}
```

C:

```
int nextGreaterElement(int n) {

}
```

Go:

```
func nextGreaterElement(n int) int {

}
```

Kotlin:

```
class Solution {
fun nextGreaterElement(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func nextGreaterElement(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn next_greater_element(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def next_greater_element(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function nextGreaterElement($n) {

}
}
```

Dart:

```
class Solution {
int nextGreaterElement(int n) {

}
}
```

Scala:

```
object Solution {
def nextGreaterElement(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec next_greater_element(n :: integer) :: integer
def next_greater_element(n) do

end
end
```

Erlang:

```
-spec next_greater_element(N :: integer()) -> integer().
next_greater_element(N) ->
.
```

Racket:

```
(define/contract (next-greater-element n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Next Greater Element III
* Difficulty: Medium
* Tags: array, string, math
*
* 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 nextGreaterElement(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Next Greater Element III
* Difficulty: Medium
* Tags: array, string, math
*
* 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 nextGreaterElement(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Next Greater Element III
Difficulty: Medium
Tags: array, string, math

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 nextGreaterElement(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def nextGreaterElement(self, n):
"""
:type n: int
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Next Greater Element III
* Difficulty: Medium
* Tags: array, string, math
*
* 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} n
* @return {number}
*/
var nextGreaterElement = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Next Greater Element III
* Difficulty: Medium
* Tags: array, string, math
*
* 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 nextGreaterElement(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Next Greater Element III
* Difficulty: Medium
* Tags: array, string, math
*
* 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 NextGreaterElement(int n) {

}
}
```

C Solution:

```
/*
* Problem: Next Greater Element III
* Difficulty: Medium
* Tags: array, string, math
*
* 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 nextGreaterElement(int n) {

}
```

Go Solution:

```
// Problem: Next Greater Element III
// Difficulty: Medium
// Tags: array, string, math
//
// 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 nextGreaterElement(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun nextGreaterElement(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func nextGreaterElement(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Next Greater Element III
// Difficulty: Medium
// Tags: array, string, math
//
// 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 next_greater_element(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def next_greater_element(n)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function nextGreaterElement($n) {

}
}
```

Dart Solution:

```
class Solution {
int nextGreaterElement(int n) {

}
}
```

Scala Solution:

```
object Solution {
def nextGreaterElement(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec next_greater_element(n :: integer) :: integer
def next_greater_element(n) do

end
end
```

Erlang Solution:

```
-spec next_greater_element(N :: integer()) -> integer().
next_greater_element(N) ->
.
```

Racket Solution:

```
(define/contract (next-greater-element n)
(-> exact-integer? exact-integer?)
)
```