Problem 738: Monotone Increasing Digits

Problem Information

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

Problem Description

An integer has

monotone increasing digits

if and only if each pair of adjacent digits

x

and

y

satisfy

x <= y

.

Given an integer

n

, return

the largest number that is less than or equal to n

with

monotone increasing digits

.

Example 1:

Input:

n = 10

Output:

9

Example 2:

Input:

n = 1234

Output:

1234

Example 3:

Input:

n = 332

Output:

299

Constraints:

0 <= n <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int monotoneIncreasingDigits(int n) {

}
```

Go:

```
func monotoneIncreasingDigits(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (monotone-increasing-digits n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Monotone Increasing Digits
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/
```

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

}
};
```

Java Solution:

```
/**
* Problem: Monotone Increasing Digits
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int monotoneIncreasingDigits(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Monotone Increasing Digits
Difficulty: Medium
Tags: greedy, math

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def monotoneIncreasingDigits(self, n: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Monotone Increasing Digits
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Monotone Increasing Digits
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

function monotoneIncreasingDigits(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Monotone Increasing Digits
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MonotoneIncreasingDigits(int n) {

}
}
```

C Solution:

```
/*
* Problem: Monotone Increasing Digits
* Difficulty: Medium
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int monotoneIncreasingDigits(int n) {

}
```

Go Solution:

```
// Problem: Monotone Increasing Digits
// Difficulty: Medium
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func monotoneIncreasingDigits(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Monotone Increasing Digits
// Difficulty: Medium
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (monotone-increasing-digits n)
(-> exact-integer? exact-integer?)
)
```