Problem 2165: Smallest Value of the Rearranged Number

Problem Information

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

Problem Description

You are given an integer

num.

Rearrange

the digits of

num

such that its value is

minimized

and it does not contain

any

leading zeros.

Return

the rearranged number with minimal value

.

Note that the sign of the number does not change after rearranging the digits.

Example 1:

Input:

num = 310

Output:

103

Explanation:

The possible arrangements for the digits of 310 are 013, 031, 103, 130, 301, 310. The arrangement with the smallest value that does not contain any leading zeros is 103.

Example 2:

Input:

num = -7605

Output:

-7650

Explanation:

Some possible arrangements for the digits of -7605 are -7650, -6705, -5076, -0567. The arrangement with the smallest value that does not contain any leading zeros is -7650.

Constraints:

-10

15

<= num <= 10

Code Snippets

C++:

```
class Solution {
public:
long long smallestNumber(long long num) {

}
};
```

Java:

```
class Solution {
public long smallestNumber(long num) {

}
}
```

Python3:

```
class Solution:
def smallestNumber(self, num: int) -> int:
```

Python:

```
class Solution(object):
def smallestNumber(self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {number}
*/
```

```
var smallestNumber = function(num) {

};
```

TypeScript:

```
function smallestNumber(num: number): number {

};
```

C#:

```
public class Solution {
public long SmallestNumber(long num) {

}
}
```

C:

```
long long smallestNumber(long long num) {

}
```

Go:

```
func smallestNumber(num int64) int64 {

}
```

Kotlin:

```
class Solution {
fun smallestNumber(num: Long): Long {

}
}
```

Swift:

```
class Solution {
func smallestNumber(_ num: Int) -> Int {
```

```
}
}
```

Rust:

```
impl Solution {
pub fn smallest_number(num: i64) -> i64 {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Integer}
def smallest_number(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function smallestNumber($num) {

}
}
```

Dart:

```
class Solution {
int smallestNumber(int num) {

}
}
```

Scala:

```
object Solution {
def smallestNumber(num: Long): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smallest_number(num :: integer) :: integer
def smallest_number(num) do

end
end
```

Erlang:

```
-spec smallest_number(Num :: integer()) -> integer().
smallest_number(Num) ->
.
```

Racket:

```
(define/contract (smallest-number num)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Value of the Rearranged Number
* Difficulty: Medium
* Tags: math, sort
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public:
long long smallestNumber(long long num) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Value of the Rearranged Number
* Difficulty: Medium
* Tags: math, sort
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public long smallestNumber(long num) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Value of the Rearranged Number
Difficulty: Medium
Tags: math, sort

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def smallestNumber(self, num: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def smallestNumber(self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Smallest Value of the Rearranged Number
* Difficulty: Medium
* Tags: math, sort
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} num
* @return {number}
*/
var smallestNumber = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Value of the Rearranged Number
* Difficulty: Medium
* Tags: math, sort
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function smallestNumber(num: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Smallest Value of the Rearranged Number
* Difficulty: Medium
* Tags: math, sort
*
* Approach: Optimized algorithm based on problem constraints
* 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 long SmallestNumber(long num) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Value of the Rearranged Number
* Difficulty: Medium
* Tags: math, sort
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

long long smallestNumber(long long num) {

}
```

Go Solution:

```
// Problem: Smallest Value of the Rearranged Number
// Difficulty: Medium
```

```
// Tags: math, sort
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func smallestNumber(num int64) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun smallestNumber(num: Long): Long {

}
}
```

Swift Solution:

```
class Solution {
func smallestNumber(_ num: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Smallest Value of the Rearranged Number
// Difficulty: Medium
// Tags: math, sort
//
// Approach: Optimized algorithm based on problem constraints
// 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 smallest_number(num: i64) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Integer}
def smallest_number(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function smallestNumber($num) {

}
}
```

Dart Solution:

```
class Solution {
int smallestNumber(int num) {

}
}
```

Scala Solution:

```
object Solution {
def smallestNumber(num: Long): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smallest_number(num :: integer) :: integer
def smallest_number(num) do
```

```
end
end
```

Erlang Solution:

```
-spec smallest_number(Num :: integer()) -> integer().
smallest_number(Num) ->
.
```

Racket Solution:

```
(define/contract (smallest-number num)
(-> exact-integer? exact-integer?)
)
```