Problem 483: Smallest Good Base

Problem Information

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

Problem Description

Given an integer

n

represented as a string, return

the smallest

good base

of

n

.

We call

k >= 2

a

good base

of

n

, if all digits of

n

base

k

are

1

's.

Example 1:

Input:

n = "13"

Output:

"3"

Explanation:

13 base 3 is 111.

Example 2:

Input:

n = "4681"

Output:

"8"

Explanation:

4681 base 8 is 11111.

Example 3:

Input:

n = "1000000000000000000"

Output:

"999999999999999999"

Explanation:

1000000000000000000 base 999999999999999999 is 11.

Constraints:

n

is an integer in the range

[3, 10

18

]

.

n

does not contain any leading zeros.

Code Snippets

C++:

```
class Solution {
public:
string smallestGoodBase(string n) {

}
};
```

Java:

```
class Solution {
public String smallestGoodBase(String n) {

}
}
```

Python3:

```
class Solution:
def smallestGoodBase(self, n: str) -> str:
```

Python:

```
class Solution(object):
def smallestGoodBase(self, n):
"""
:type n: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} n
* @return {string}
*/
var smallestGoodBase = function(n) {

};
```

TypeScript:

```
function smallestGoodBase(n: string): string {

};
```

C#:

```
public class Solution {
public string SmallestGoodBase(string n) {

}
}
```

C:

```
char* smallestGoodBase(char* n) {

}
```

Go:

```
func smallestGoodBase(n string) string {

}
```

Kotlin:

```
class Solution {
fun smallestGoodBase(n: String): String {

}
}
```

Swift:

```
class Solution {
func smallestGoodBase(_ n: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn smallest_good_base(n: String) -> String {

}
}
```

Ruby:

```
# @param {String} n
# @return {String}
def smallest_good_base(n)

end
```

PHP:

```
class Solution {

/**
* @param String $n
* @return String
*/
function smallestGoodBase($n) {

}
}
```

Dart:

```
class Solution {
String smallestGoodBase(String n) {

}
}
```

Scala:

```
object Solution {
def smallestGoodBase(n: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smallest_good_base(n :: String.t) :: String.t
def smallest_good_base(n) do

end
end
```

Erlang:

```
-spec smallest_good_base(N :: unicode:unicode_binary()) ->
unicode:unicode_binary().
smallest_good_base(N) ->
.
```

Racket:

```
(define/contract (smallest-good-base n)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Good Base
* Difficulty: Hard
* Tags: string, math, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string smallestGoodBase(string n) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Good Base
* Difficulty: Hard
* Tags: string, math, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String smallestGoodBase(String n) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Good Base
Difficulty: Hard
Tags: string, math, search

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

```
class Solution(object):
def smallestGoodBase(self, n):
"""
:type n: str
:rtype: str
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Smallest Good Base
* Difficulty: Hard
* Tags: string, math, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} n
* @return {string}
*/
var smallestGoodBase = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Good Base
* Difficulty: Hard
* Tags: string, math, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function smallestGoodBase(n: string): string {

};
```

C# Solution:

```
/*
* Problem: Smallest Good Base
* Difficulty: Hard
* Tags: string, math, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string SmallestGoodBase(string n) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Good Base
* Difficulty: Hard
* Tags: string, math, search
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* smallestGoodBase(char* n) {

}
```

Go Solution:

```
// Problem: Smallest Good Base
// Difficulty: Hard
// Tags: string, math, search
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach
```

```
func smallestGoodBase(n string) string {

}
```

Kotlin Solution:

```
class Solution {
fun smallestGoodBase(n: String): String {

}
}
```

Swift Solution:

```
class Solution {
func smallestGoodBase(_ n: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Smallest Good Base
// Difficulty: Hard
// Tags: string, math, search
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn smallest_good_base(n: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} n
# @return {String}
def smallest_good_base(n)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $n
* @return String
*/
function smallestGoodBase($n) {

}
}
```

Dart Solution:

```
class Solution {
String smallestGoodBase(String n) {

}
}
```

Scala Solution:

```
object Solution {
def smallestGoodBase(n: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smallest_good_base(n :: String.t) :: String.t
def smallest_good_base(n) do

end
end
```

Erlang Solution:

```
-spec smallest_good_base(N :: unicode:unicode_binary()) ->
unicode:unicode_binary().
smallest_good_base(N) ->
.
```

Racket Solution:

```
(define/contract (smallest-good-base n)
(-> string? string?)
)
```