Problem 2847: Smallest Number With Given Digit Product

Problem Information

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

Problem Description

Given a

positive

integer

n

, return

a string representing the

smallest positive

integer such that the product of its digits is equal to

n

, or

"-1"

if no such number exists

.

Example 1:

Input:

n = 105

Output:

"357"

Explanation:

3 * 5 * 7 = 105. It can be shown that 357 is the smallest number with a product of digits equal to 105. So the answer would be "357".

Example 2:

Input:

n = 7

Output:

"7"

Explanation:

Since 7 has only one digit, its product of digits would be 7. We will show that 7 is the smallest number with a product of digits equal to 7. Since the product of numbers 1 to 6 is 1 to 6 respectively, so "7" would be the answer.

Example 3:

Input:

n = 44

Output:

"-1"

Explanation:

It can be shown that there is no number such that its product of digits is equal to 44. So the answer would be "-1".

Constraints:

1 <= n <= 10

18

Code Snippets

C++:

```
class Solution {
public:
string smallestNumber(long long n) {

}
};
```

Java:

```
class Solution {
public String smallestNumber(long n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function smallestNumber(n: number): string {

};
```

C#:

```
public class Solution {
public string SmallestNumber(long n) {

}
}
```

C:

```
char* smallestNumber(long long n) {

}
```

Go:

```
func smallestNumber(n int64) string {

}
```

Kotlin:

```
class Solution {
fun smallestNumber(n: Long): String {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

```
class Solution {
String smallestNumber(int n) {

}
}
```

Scala:

```
object Solution {
def smallestNumber(n: Long): String = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec smallest_number(N :: integer()) -> unicode:unicode_binary().
smallest_number(N) ->
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Smallest Number With Given Digit Product
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 smallestNumber(long long n) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Number With Given Digit Product
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 smallestNumber(long n) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Number With Given Digit Product
Difficulty: Medium
Tags: string, greedy, math
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Number With Given Digit Product
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 {number} n
* @return {string}
*/
var smallestNumber = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Number With Given Digit Product
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 smallestNumber(n: number): string {

};
```

C# Solution:

```
/*
* Problem: Smallest Number With Given Digit Product
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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 SmallestNumber(long n) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Number With Given Digit Product
* Difficulty: Medium
* Tags: string, greedy, math
*
* 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* smallestNumber(long long n) {

}
```

Go Solution:

```
// Problem: Smallest Number With Given Digit Product
// Difficulty: Medium
// Tags: string, greedy, math
//
// 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 smallestNumber(n int64) string {

}
```

Kotlin Solution:

```
class Solution {
fun smallestNumber(n: Long): String {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Number With Given Digit Product
// Difficulty: Medium
// Tags: string, greedy, math
```

```
//
// 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_number(n: i64) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
String smallestNumber(int n) {

}
}
```

Scala Solution:

```
object Solution {
def smallestNumber(n: Long): String = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec smallest_number(N :: integer()) -> unicode:unicode_binary().
smallest_number(N) ->
.
```

Racket Solution:

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