Problem 2375: Construct Smallest Number From DI String

Problem Information

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

Problem Description

You are given a

0-indexed

string

pattern

of length

n

consisting of the characters

'I'

meaning

increasing

and

'D'

meaning

decreasing

.

A

0-indexed

string

num

of length

n + 1

is created using the following conditions:

num

consists of the digits

'1'

to

'9'

, where each digit is used

at most

once.

If

pattern[i] == 'I'

, then

num[i] < num[i + 1]

.

If

pattern[i] == 'D'

, then

num[i] > num[i + 1]

.

Return

the lexicographically

smallest

possible string

num

that meets the conditions.

Example 1:

Input:

pattern = "IIIDIDDD"

Output:

"123549876"

Explanation:

At indices 0, 1, 2, and 4 we must have that num[i] < num[i+1]. At indices 3, 5, 6, and 7 we must have that num[i] > num[i+1]. Some possible values of num are "245639871", "135749862", and "123849765". It can be proven that "123549876" is the smallest possible num that meets the conditions. Note that "123414321" is not possible because the digit '1' is used more than once.

Example 2:

Input:

pattern = "DDD"

Output:

"4321"

Explanation:

Some possible values of num are "9876", "7321", and "8742". It can be proven that "4321" is the smallest possible num that meets the conditions.

Constraints:

1 <= pattern.length <= 8

pattern

consists of only the letters

'I'

and

'D'

.

Code Snippets

C++:

```
class Solution {
public:
string smallestNumber(string pattern) {

}
};
```

Java:

```
class Solution {
public String smallestNumber(String pattern) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function smallestNumber(pattern: string): string {

};
```

C#:

```
public class Solution {
public string SmallestNumber(string pattern) {

}
}
```

C:

```
char* smallestNumber(char* pattern) {

}
```

Go:

```
func smallestNumber(pattern string) string {

}
```

Kotlin:

```
class Solution {
fun smallestNumber(pattern: String): String {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
String smallestNumber(String pattern) {

}
}
```

Scala:

```
object Solution {
def smallestNumber(pattern: String): String = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec smallest_number(Pattern :: unicode:unicode_binary()) ->
unicode:unicode_binary().
smallest_number(Pattern) ->
.
```

Racket:

```
(define/contract (smallest-number pattern)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Construct Smallest Number From DI String
* Difficulty: Medium
* Tags: string, graph, greedy, stack
*
* 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(string pattern) {

}
};
```

Java Solution:

```
/**
* Problem: Construct Smallest Number From DI String
* Difficulty: Medium
* Tags: string, graph, greedy, stack
*
* 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(String pattern) {

}
}
```

Python3 Solution:

```
"""
Problem: Construct Smallest Number From DI String
Difficulty: Medium
Tags: string, graph, greedy, stack

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, pattern: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Construct Smallest Number From DI String
* Difficulty: Medium
* Tags: string, graph, greedy, stack
*
* 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} pattern
* @return {string}
*/
var smallestNumber = function(pattern) {

};
```

TypeScript Solution:

```
/**
* Problem: Construct Smallest Number From DI String
* Difficulty: Medium
* Tags: string, graph, greedy, stack
*
* 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(pattern: string): string {

};
```

C# Solution:

```
/*
* Problem: Construct Smallest Number From DI String
* Difficulty: Medium
* Tags: string, graph, greedy, stack
*
* 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(string pattern) {

}
}
```

C Solution:

```
/*
* Problem: Construct Smallest Number From DI String
* Difficulty: Medium
* Tags: string, graph, greedy, stack
*
* 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(char* pattern) {

}
```

Go Solution:

```
// Problem: Construct Smallest Number From DI String
// Difficulty: Medium
// Tags: string, graph, greedy, stack
//
// 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(pattern string) string {

}
```

Kotlin Solution:

```
class Solution {
fun smallestNumber(pattern: String): String {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Construct Smallest Number From DI String
// Difficulty: Medium
// Tags: string, graph, greedy, stack
//
// 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(pattern: String) -> String {

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
String smallestNumber(String pattern) {

}
}
```

Scala Solution:

```
object Solution {
def smallestNumber(pattern: String): String = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec smallest_number(Pattern :: unicode:unicode_binary()) ->
unicode:unicode_binary().
smallest_number(Pattern) ->
.
```

Racket Solution:

```
(define/contract (smallest-number pattern)
(-> string? string?)
)
```