Problem 423: Reconstruct Original Digits from English

Problem Information

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

Problem Description

Given a string

s

containing an out-of-order English representation of digits

0-9

, return

the digits in

ascending

order

.

Example 1:

Input:

s = "owoztneoer"

Output:

"012"

Example 2:

Input:

s = "fviefuro"

Output:

"45"

Constraints:

1 <= s.length <= 10

5

s[i]

is one of the characters

["e","g","f","i","h","o","n","s","r","u","t","w","v","x","z"]

.

s

is

guaranteed

to be valid.

Code Snippets

C++:

```
class Solution {
public:
string originalDigits(string s) {

}
};
```

Java:

```
class Solution {
public String originalDigits(String s) {

}
}
```

Python3:

```
class Solution:
def originalDigits(self, s: str) -> str:
```

Python:

```
class Solution(object):
def originalDigits(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var originalDigits = function(s) {

};
```

TypeScript:

```
function originalDigits(s: string): string {
```

```
};
```

C#:

```
public class Solution {
public string OriginalDigits(string s) {

}
}
```

C:

```
char* originalDigits(char* s) {

}
```

Go:

```
func originalDigits(s string) string {

}
```

Kotlin:

```
class Solution {
fun originalDigits(s: String): String {

}
}
```

Swift:

```
class Solution {
func originalDigits(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn original_digits(s: String) -> String {
```

```
}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def original_digits(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function originalDigits($s) {

}
}
```

Dart:

```
class Solution {
String originalDigits(String s) {

}
}
```

Scala:

```
object Solution {
def originalDigits(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec original_digits(s :: String.t) :: String.t
def original_digits(s) do

end
end
```

Erlang:

```
-spec original_digits(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
original_digits(S) ->
.
```

Racket:

```
(define/contract (original-digits s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reconstruct Original Digits from English
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string originalDigits(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Reconstruct Original Digits from English
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String originalDigits(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Reconstruct Original Digits from English
Difficulty: Medium
Tags: string, math, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

```
class Solution(object):
def originalDigits(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Reconstruct Original Digits from English
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @return {string}
*/
var originalDigits = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Reconstruct Original Digits from English
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function originalDigits(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Reconstruct Original Digits from English
* Difficulty: Medium
* Tags: string, math, hash
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string OriginalDigits(string s) {

}
}
```

C Solution:

```
/*
* Problem: Reconstruct Original Digits from English
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* originalDigits(char* s) {

}
```

Go Solution:

```
// Problem: Reconstruct Original Digits from English
// Difficulty: Medium
// Tags: string, math, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func originalDigits(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun originalDigits(s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func originalDigits(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Reconstruct Original Digits from English
// Difficulty: Medium
// Tags: string, math, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn original_digits(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def original_digits(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
*/
function originalDigits($s) {

}
}
```

Dart Solution:

```
class Solution {
String originalDigits(String s) {

}
}
```

Scala Solution:

```
object Solution {
def originalDigits(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec original_digits(s :: String.t) :: String.t
def original_digits(s) do

end
end
```

Erlang Solution:

```
-spec original_digits(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
original_digits(S) ->
.
```

Racket Solution:

```
(define/contract (original-digits s)
(-> string? string?)
)
```