Problem 1419: Minimum Number of Frogs Croaking

Problem Information

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

Problem Description

You are given the string

croakOfFrogs

, which represents a combination of the string

"croak"

from different frogs, that is, multiple frogs can croak at the same time, so multiple

"croak"

are mixed.

Return the minimum number of

different

frogs to finish all the croaks in the given string.

A valid

"croak"

means a frog is printing five letters

'c'

,

'r'

,

'o'

,

'a'

, and

'k'

sequentially

. The frogs have to print all five letters to finish a croak. If the given string is not a combination of a valid

"croak"

return

-1

.

Example 1:

Input:

croakOfFrogs = "croakcroak"

Output:

1

Explanation:

One frog yelling "croak

"

twice.

Example 2:

Input:

croakOfFrogs = "crcoakroak"

Output:

2

Explanation:

The minimum number of frogs is two. The first frog could yell "

cr

c

oak

roak". The second frog could yell later "cr

c

oak

roak

".

Example 3:

Input:

croakOfFrogs = "croakcrook"

Output:

-1

Explanation:

The given string is an invalid combination of "croak

"

from different frogs.

Constraints:

1 <= croakOfFrogs.length <= 10

5

croakOfFrogs

is either

'c'

,

'r'

,

'o'

,

'a'

, or

'k'

.

Code Snippets

C++:

```
class Solution {
public:
int minNumberOfFrogs(string croakOfFrogs) {

}
};
```

Java:

```
class Solution {
public int minNumberOfFrogs(String croakOfFrogs) {

}
}
```

Python3:

```
class Solution:
def minNumberOfFrogs(self, croakOfFrogs: str) -> int:
```

Python:

```
class Solution(object):
def minNumberOfFrogs(self, croakOfFrogs):
"""
:type croakOfFrogs: str
```

```
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} croakOfFrogs
* @return {number}
*/
var minNumberOfFrogs = function(croakOfFrogs) {

};
```

TypeScript:

```
function minNumberOfFrogs(croakOfFrogs: string): number {

};
```

C#:

```
public class Solution {
public int MinNumberOfFrogs(string croakOfFrogs) {

}
}
```

C:

```
int minNumberOfFrogs(char* croakOfFrogs) {

}
```

Go:

```
func minNumberOfFrogs(croakOfFrogs string) int {

}
```

Kotlin:

```
class Solution {
fun minNumberOfFrogs(croakOfFrogs: String): Int {

}
}
```

Swift:

```
class Solution {
func minNumberOfFrogs(_ croakOfFrogs: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_number_of_frogs(croak_of_frogs: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} croak_of_frogs
# @return {Integer}
def min_number_of_frogs(croak_of_frogs)

end
```

PHP:

```
class Solution {

/**
* @param String $croakOfFrogs
* @return Integer
*/
function minNumberOfFrogs($croakOfFrogs) {

}
}
```

Dart:

```
class Solution {
int minNumberOfFrogs(String croakOfFrogs) {

}
}
```

Scala:

```
object Solution {
def minNumberOfFrogs(croakOfFrogs: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_number_of_frogs(croak_of_frogs :: String.t) :: integer
def min_number_of_frogs(croak_of_frogs) do

end
end
```

Erlang:

```
-spec min_number_of_frogs(CroakOfFrogs :: unicode:unicode_binary()) ->
integer().
min_number_of_frogs(CroakOfFrogs) ->
.
```

Racket:

```
(define/contract (min-number-of-frogs croakOfFrogs)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Frogs Croaking
* Difficulty: Medium
* Tags: string
*
* 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:
int minNumberOfFrogs(string croakOfFrogs) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Frogs Croaking
* Difficulty: Medium
* Tags: string
*
* 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 int minNumberOfFrogs(String croakOfFrogs) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Frogs Croaking
Difficulty: Medium
Tags: string
```

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

Python Solution:

```
class Solution(object):
def minNumberOfFrogs(self, croakOfFrogs):
"""
:type croakOfFrogs: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Number of Frogs Croaking
* Difficulty: Medium
* Tags: string
*
* 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} croakOfFrogs
* @return {number}
*/
var minNumberOfFrogs = function(croakOfFrogs) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Frogs Croaking
* Difficulty: Medium
* Tags: string
*
* 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 minNumberOfFrogs(croakOfFrogs: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Frogs Croaking
* Difficulty: Medium
* Tags: string
*
* 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 int MinNumberOfFrogs(string croakOfFrogs) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Frogs Croaking
* Difficulty: Medium
* Tags: string
*
* 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
```

```
*/

int minNumberOfFrogs(char* croakOfFrogs) {

}
```

Go Solution:

```
// Problem: Minimum Number of Frogs Croaking
// Difficulty: Medium
// Tags: string
//
// 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 minNumberOfFrogs(croakOfFrogs string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minNumberOfFrogs(croakOfFrogs: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minNumberOfFrogs(_ croakOfFrogs: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Frogs Croaking
// Difficulty: Medium
// Tags: string
```

```
//
// 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 min_number_of_frogs(croak_of_frogs: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} croak_of_frogs
# @return {Integer}
def min_number_of_frogs(croak_of_frogs)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $croakOfFrogs
* @return Integer
*/
function minNumberOfFrogs($croakOfFrogs) {

}
}
```

Dart Solution:

```
class Solution {
int minNumberOfFrogs(String croakOfFrogs) {

}
}
```

Scala Solution:

```
object Solution {
def minNumberOfFrogs(croakOfFrogs: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_number_of_frogs(croak_of_frogs :: String.t) :: integer
def min_number_of_frogs(croak_of_frogs) do

end
end
```

Erlang Solution:

```
-spec min_number_of_frogs(CroakOfFrogs :: unicode:unicode_binary()) ->
integer().
min_number_of_frogs(CroakOfFrogs) ->
.
```

Racket Solution:

```
(define/contract (min-number-of-frogs croakOfFrogs)
(-> string? exact-integer?)
)
```