Problem 2047: Number of Valid Words in a Sentence

Problem Information

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

Problem Description

A sentence consists of lowercase letters (

'a'

to

'z'

), digits (

'0'

to

'9'

), hyphens (

'-'

), punctuation marks (

'!'

,

'.'

, and

','

), and spaces (

' '

) only. Each sentence can be broken down into

one or more tokens

separated by one or more spaces

' '

.

A token is a valid word if

all three

of the following are true:

It only contains lowercase letters, hyphens, and/or punctuation (

no

digits).

There is

at most one

hyphen

'-'

. If present, it

must

be surrounded by lowercase characters (

"a-b"

is valid, but

"-ab"

and

"ab-"

are not valid).

There is

at most one

punctuation mark. If present, it

must

be at the

end

of the token (

"ab,"

,

"cd!"

, and

"."

are valid, but

"a!b"

and

"c.,"

are not valid).

Examples of valid words include

"a-b."

,

"afad"

,

"ba-c"

,

"a!"

, and

"!"

.

Given a string

sentence

, return

the

number

of valid words in

sentence

.

Example 1:

Input:

sentence = "

cat

and

dog

"

Output:

3

Explanation:

The valid words in the sentence are "cat", "and", and "dog".

Example 2:

Input:

sentence = "!this 1-s b8d!"

Output:

0

Explanation:

There are no valid words in the sentence. "!this" is invalid because it starts with a punctuation mark. "1-s" and "b8d" are invalid because they contain digits.

Example 3:

Input:

sentence = "

alice

and

bob

are

playing

stone-game10"

Output:

5

Explanation:

The valid words in the sentence are "alice", "and", "bob", "are", and "playing". "stone-game10" is invalid because it contains digits.

Constraints:

1 <= sentence.length <= 1000

sentence

only contains lowercase English letters, digits,

' '

,

'-'

,

'!'

,

'.'

, and

','

.

There will be at least

1

token.

Code Snippets

C++:

```
class Solution {
public:
```

```
int countValidWords(string sentence) {

}
};
```

Java:

```
class Solution {
public int countValidWords(String sentence) {

}
}
```

Python3:

```
class Solution:
def countValidWords(self, sentence: str) -> int:
```

Python:

```
class Solution(object):
def countValidWords(self, sentence):
"""
:type sentence: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} sentence
* @return {number}
*/
var countValidWords = function(sentence) {

};
```

TypeScript:

```
function countValidWords(sentence: string): number {

};
```

C#:

```
public class Solution {
public int CountValidWords(string sentence) {

}
}
```

C:

```
int countValidWords(char* sentence) {

}
```

Go:

```
func countValidWords(sentence string) int {

}
```

Kotlin:

```
class Solution {
fun countValidWords(sentence: String): Int {

}
}
```

Swift:

```
class Solution {
func countValidWords(_ sentence: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_valid_words(sentence: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} sentence
# @return {Integer}
def count_valid_words(sentence)

end
```

PHP:

```
class Solution {

/**
* @param String $sentence
* @return Integer
*/
function countValidWords($sentence) {

}
}
```

Dart:

```
class Solution {
int countValidWords(String sentence) {

}
}
```

Scala:

```
object Solution {
def countValidWords(sentence: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_valid_words(sentence :: String.t) :: integer
def count_valid_words(sentence) do
```

```
end
end
```

Erlang:

```
-spec count_valid_words(Sentence :: unicode:unicode_binary()) -> integer().
count_valid_words(Sentence) ->
.
```

Racket:

```
(define/contract (count-valid-words sentence)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Valid Words in a Sentence
* Difficulty: Easy
* 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 countValidWords(string sentence) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Valid Words in a Sentence
```

```
* Difficulty: Easy
* 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 countValidWords(String sentence) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Valid Words in a Sentence
Difficulty: Easy
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 countValidWords(self, sentence: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countValidWords(self, sentence):
"""
:type sentence: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Valid Words in a Sentence
* Difficulty: Easy
* 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} sentence
* @return {number}
*/
var countValidWords = function(sentence) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Valid Words in a Sentence
* Difficulty: Easy
* 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 countValidWords(sentence: string): number {

};
```

C# Solution:

```
/*
* Problem: Number of Valid Words in a Sentence
* Difficulty: Easy
* 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 CountValidWords(string sentence) {

}
}
```

C Solution:

```
/*
* Problem: Number of Valid Words in a Sentence
* Difficulty: Easy
* 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 countValidWords(char* sentence) {

}
```

Go Solution:

```
// Problem: Number of Valid Words in a Sentence
// Difficulty: Easy
// 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 countValidWords(sentence string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countValidWords(sentence: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func countValidWords(_ sentence: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Valid Words in a Sentence
// Difficulty: Easy
// 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 count_valid_words(sentence: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} sentence
# @return {Integer}
def count_valid_words(sentence)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $sentence
* @return Integer
*/
function countValidWords($sentence) {

}
}
```

Dart Solution:

```
class Solution {
int countValidWords(String sentence) {

}
}
```

Scala Solution:

```
object Solution {
def countValidWords(sentence: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_valid_words(sentence :: String.t) :: integer
def count_valid_words(sentence) do

end
end
```

Erlang Solution:

```
-spec count_valid_words(Sentence :: unicode:unicode_binary()) -> integer().
count_valid_words(Sentence) ->
.
```

Racket Solution:

```
(define/contract (count-valid-words sentence)
(-> string? exact-integer?)
)
```