Problem 2586: Count the Number of Vowel Strings in Range

Problem Information

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

Problem Description

You are given a

0-indexed

array of string

words

and two integers

left

and

right

.

A string is called a

vowel string

if it starts with a vowel character and ends with a vowel character where vowel characters are

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

.

Return

the number of vowel strings

words[i]

where

i

belongs to the inclusive range

[left, right]

.

Example 1:

Input:

words = ["are","amy","u"], left = 0, right = 2

Output:

2

Explanation:

- "are" is a vowel string because it starts with 'a' and ends with 'e'. - "amy" is not a vowel string because it does not end with a vowel. - "u" is a vowel string because it starts with 'u' and ends with 'u'. The number of vowel strings in the mentioned range is 2.

Example 2:

Input:

words = ["hey","aeo","mu","ooo","artro"], left = 1, right = 4

Output:

3

Explanation:

- "aeo" is a vowel string because it starts with 'a' and ends with 'o'. - "mu" is not a vowel string because it does not start with a vowel. - "ooo" is a vowel string because it starts with 'o' and ends with 'o'. - "artro" is a vowel string because it starts with 'a' and ends with 'o'. The number of vowel strings in the mentioned range is 3.

Constraints:

1 <= words.length <= 1000

1 <= words[i].length <= 10

words[i]

consists of only lowercase English letters.

0 <= left <= right < words.length

Code Snippets

C++:

```
class Solution {
public:
int vowelStrings(vector<string>& words, int left, int right) {

}
};
```

Java:

```
class Solution {
public int vowelStrings(String[] words, int left, int right) {

}
}
```

Python3:

```
class Solution:
def vowelStrings(self, words: List[str], left: int, right: int) -> int:
```

Python:

```
class Solution(object):
def vowelStrings(self, words, left, right):
"""
:type words: List[str]
:type left: int
:type right: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string[]} words
```

```
 * @param {number} left
 * @param {number} right
 * @return {number}
 */
var vowelStrings = function(words, left, right) {

};
```

TypeScript:

```
function vowelStrings(words: string[], left: number, right: number): number {

};
```

C#:

```
public class Solution {
public int VowelStrings(string[] words, int left, int right) {

}
}
```

C:

```
int vowelStrings(char** words, int wordsSize, int left, int right) {

}
```

Go:

```
func vowelStrings(words []string, left int, right int) int {

}
```

Kotlin:

```
class Solution {
fun vowelStrings(words: Array<String>, left: Int, right: Int): Int {

}
}
```

Swift:

```
class Solution {
func vowelStrings(_ words: [String], _ left: Int, _ right: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn vowel_strings(words: Vec<String>, left: i32, right: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String[]} words
# @param {Integer} left
# @param {Integer} right
# @return {Integer}
def vowel_strings(words, left, right)

end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @param Integer $left
* @param Integer $right
* @return Integer
*/
function vowelStrings($words, $left, $right) {

}
}
```

Dart:

```
class Solution {
int vowelStrings(List<String> words, int left, int right) {

}
}
```

Scala:

```
object Solution {
def vowelStrings(words: Array[String], left: Int, right: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec vowel_strings(words :: [String.t], left :: integer, right :: integer)
:: integer
def vowel_strings(words, left, right) do

end
end
```

Erlang:

```
-spec vowel_strings(Words :: [unicode:unicode_binary()], Left :: integer(),
Right :: integer()) -> integer().
vowel_strings(Words, Left, Right) ->
.
```

Racket:

```
(define/contract (vowel-strings words left right)
(-> (listof string?) exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Number of Vowel Strings in Range
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int vowelStrings(vector<string>& words, int left, int right) {

}
};
```

Java Solution:

```
/**
* Problem: Count the Number of Vowel Strings in Range
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int vowelStrings(String[] words, int left, int right) {

}
}
```

Python3 Solution:

```
"""
Problem: Count the Number of Vowel Strings in Range
Difficulty: Easy
Tags: array, string
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def vowelStrings(self, words: List[str], left: int, right: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def vowelStrings(self, words, left, right):
"""
:type words: List[str]
:type left: int
:type right: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count the Number of Vowel Strings in Range
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string[]} words
* @param {number} left
* @param {number} right
* @return {number}
*/
var vowelStrings = function(words, left, right) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Count the Number of Vowel Strings in Range
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function vowelStrings(words: string[], left: number, right: number): number {

};
```

C# Solution:

```
/*
* Problem: Count the Number of Vowel Strings in Range
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int VowelStrings(string[] words, int left, int right) {

}
}
```

C Solution:

```
/*
* Problem: Count the Number of Vowel Strings in Range
* Difficulty: Easy
```

```
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int vowelStrings(char** words, int wordsSize, int left, int right) {

}
```

Go Solution:

```
// Problem: Count the Number of Vowel Strings in Range
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func vowelStrings(words []string, left int, right int) int {

}
```

Kotlin Solution:

```
class Solution {
fun vowelStrings(words: Array<String>, left: Int, right: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func vowelStrings(_ words: [String], _ left: Int, _ right: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count the Number of Vowel Strings in Range
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn vowel_strings(words: Vec<String>, left: i32, right: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @param {Integer} left
# @param {Integer} right
# @return {Integer}
def vowel_strings(words, left, right)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @param Integer $left
* @param Integer $right
* @return Integer
*/
function vowelStrings($words, $left, $right) {

}
}
```

Dart Solution:

```
class Solution {
int vowelStrings(List<String> words, int left, int right) {

}
}
```

Scala Solution:

```
object Solution {
def vowelStrings(words: Array[String], left: Int, right: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec vowel_strings(words :: [String.t], left :: integer, right :: integer)
:: integer
def vowel_strings(words, left, right) do

end
end
```

Erlang Solution:

```
-spec vowel_strings(Words :: [unicode:unicode_binary()], Left :: integer(),
Right :: integer()) -> integer().
vowel_strings(Words, Left, Right) ->
.
```

Racket Solution:

```
(define/contract (vowel-strings words left right)
(-> (listof string?) exact-integer? exact-integer? exact-integer?)
)
```