Problem 2063: Vowels of All Substrings

Problem Information

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

Problem Description

Given a string

word

, return

the

sum of the number of vowels

(

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

)

in every substring of

word

.

A

substring

is a contiguous (non-empty) sequence of characters within a string.

Note:

Due to the large constraints, the answer may not fit in a signed 32-bit integer. Please be careful during the calculations.

Example 1:

Input:

word = "aba"

Output:

6

Explanation:

All possible substrings are: "a", "ab", "aba", "b", "ba", and "a". - "b" has 0 vowels in it - "a", "ab", "ba", and "a" have 1 vowel each - "aba" has 2 vowels in it Hence, the total sum of vowels = 0 + 1 + 1 + 1 + 1 + 2 = 6.

Example 2:

Input:

word = "abc"

Output:

3

Explanation:

All possible substrings are: "a", "ab", "abc", "b", "bc", and "c". - "a", "ab", and "abc" have 1 vowel each - "b", "bc", and "c" have 0 vowels each Hence, the total sum of vowels = 1 + 1 + 1 + 0 + 0 + 0 = 3.

Example 3:

Input:

word = "ltcd"

Output:

0

Explanation:

There are no vowels in any substring of "ltcd".

Constraints:

1 <= word.length <= 10

5

word

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
long long countVowels(string word) {

}
};
```

Java:

```
class Solution {
public long countVowels(String word) {

}
}
```

Python3:

```
class Solution:
def countVowels(self, word: str) -> int:
```

Python:

```
class Solution(object):
def countVowels(self, word):
"""
:type word: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} word
* @return {number}
*/
var countVowels = function(word) {
};
```

TypeScript:

```
function countVowels(word: string): number {

};
```

C#:

```
public class Solution {
public long CountVowels(string word) {

}
}
```

C:

```
long long countVowels(char* word) {

}
```

Go:

```
func countVowels(word string) int64 {

}
```

Kotlin:

```
class Solution {
fun countVowels(word: String): Long {

}
}
```

Swift:

```
class Solution {
func countVowels(_ word: String) -> Int {

}

}
```

Rust:

```
impl Solution {
pub fn count_vowels(word: String) -> i64 {

}
}
```

Ruby:

```
# @param {String} word
# @return {Integer}
def count_vowels(word)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @return Integer
*/
function countVowels($word) {

}
}
```

Dart:

```
class Solution {
int countVowels(String word) {

}
}
```

Scala:

```
object Solution {
def countVowels(word: String): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_vowels(word :: String.t) :: integer
def count_vowels(word) do

end
end
```

Erlang:

```
-spec count_vowels(Word :: unicode:unicode_binary()) -> integer().
count_vowels(Word) ->
.
```

Racket:

```
(define/contract (count-vowels word)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Vowels of All Substrings
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
long long countVowels(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Vowels of All Substrings
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public long countVowels(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Vowels of All Substrings
Difficulty: Medium
Tags: string, tree, dp, math

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

class Solution:
def countVowels(self, word: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countVowels(self, word):
"""
:type word: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Vowels of All Substrings
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} word
* @return {number}
*/
var countVowels = function(word) {

};
```

TypeScript Solution:

```
/**
* Problem: Vowels of All Substrings
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function countVowels(word: string): number {
};
```

C# Solution:

```
/*
* Problem: Vowels of All Substrings
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public long CountVowels(string word) {

}
}
```

C Solution:

```
/*
* Problem: Vowels of All Substrings
* Difficulty: Medium
* Tags: string, tree, dp, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

long long countVowels(char* word) {

}
```

Go Solution:

```
// Problem: Vowels of All Substrings
// Difficulty: Medium
// Tags: string, tree, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func countVowels(word string) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun countVowels(word: String): Long {

}
}
```

Swift Solution:

```
class Solution {
func countVowels(_ word: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Vowels of All Substrings
// Difficulty: Medium
// Tags: string, tree, dp, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn count_vowels(word: String) -> i64 {

}
}
```

Ruby Solution:

```
# @param {String} word
# @return {Integer}
def count_vowels(word)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @return Integer
*/
function countVowels($word) {

}
}
```

Dart Solution:

```
class Solution {
int countVowels(String word) {

}
}
```

Scala Solution:

```
object Solution {
def countVowels(word: String): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_vowels(word :: String.t) :: integer
def count_vowels(word) do
```

```



end
end
```

Erlang Solution:

```
-spec count_vowels(Word :: unicode:unicode_binary()) -> integer().
count_vowels(Word) ->
.
```

Racket Solution:

```
(define/contract (count-vowels word)
(-> string? exact-integer?)
)
```