Problem 1839: Longest Substring Of All Vowels in Order

Problem Information

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

Problem Description

A string is considered

beautiful

if it satisfies the following conditions:

Each of the 5 English vowels (

'a'

,

'e'

,

'i'

,

'o'

,

'u'

) must appear

at least once

in it.

The letters must be sorted in

alphabetical order

(i.e. all

'a'

s before

'e'

s, all

'e'

s before

'i'

s, etc.).

For example, strings

"aeiou"

and

"aaaaaaeiiiioou"

are considered

beautiful

, but

"uaeio"

,

"aeoiu"

, and

"aaaeeeooo"

are

not beautiful

.

Given a string

word

consisting of English vowels, return

the

length of the longest beautiful substring

of

word

. If no such substring exists, return

0

.

A

substring

is a contiguous sequence of characters in a string.

Example 1:

Input:

word = "aeiaaio

aaaaeiiiiouuu

ooaauuaeiu"

Output:

13

Explanation:

The longest beautiful substring in word is "aaaaeiiiiouuu" of length 13.

Example 2:

Input:

word = "aeeeiiiioooauuu

aeiou

"

Output:

5

Explanation:

The longest beautiful substring in word is "aeiou" of length 5.

Example 3:

Input:

word = "a"

Output:

0

Explanation:

There is no beautiful substring, so return 0.

Constraints:

1 <= word.length <= 5 * 10

5

word

consists of characters

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

.

Code Snippets

C++:

```
class Solution {
public:
int longestBeautifulSubstring(string word) {

}
};
```

Java:

```
class Solution {
public int longestBeautifulSubstring(String word) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int LongestBeautifulSubstring(string word) {

}
}
```

C:

```
int longestBeautifulSubstring(char* word) {

}
```

Go:

```
func longestBeautifulSubstring(word string) int {

}
```

Kotlin:

```
class Solution {
fun longestBeautifulSubstring(word: String): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn longest_beautiful_substring(word: String) -> i32 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
```

```
}
```

Scala:

```
object Solution {
def longestBeautifulSubstring(word: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (longest-beautiful-substring word)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Substring Of All Vowels in Order
* Difficulty: Medium
* Tags: array, string, tree, sort
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
int longestBeautifulSubstring(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Substring Of All Vowels in Order
* Difficulty: Medium
* Tags: array, string, tree, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int longestBeautifulSubstring(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Substring Of All Vowels in Order
Difficulty: Medium
Tags: array, string, tree, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
```

```
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Substring Of All Vowels in Order
* Difficulty: Medium
* Tags: array, string, tree, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Longest Substring Of All Vowels in Order
* Difficulty: Medium
```

```
* Tags: array, string, tree, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function longestBeautifulSubstring(word: string): number {

};
```

C# Solution:

```
/*
* Problem: Longest Substring Of All Vowels in Order
* Difficulty: Medium
* Tags: array, string, tree, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int LongestBeautifulSubstring(string word) {

}
}
```

C Solution:

```
/*
* Problem: Longest Substring Of All Vowels in Order
* Difficulty: Medium
* Tags: array, string, tree, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
int longestBeautifulSubstring(char* word) {

}
```

Go Solution:

```
// Problem: Longest Substring Of All Vowels in Order
// Difficulty: Medium
// Tags: array, string, tree, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func longestBeautifulSubstring(word string) int {

}
```

Kotlin Solution:

```
class Solution {
fun longestBeautifulSubstring(word: String): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Substring Of All Vowels in Order
// Difficulty: Medium
// Tags: array, string, tree, sort
//
// Approach: Use two pointers or sliding window technique
```

```
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn longest_beautiful_substring(word: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def longestBeautifulSubstring(word: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (longest-beautiful-substring word)
(-> string? exact-integer?)
)
```