Problem 3541: Find Most Frequent Vowel and Consonant

Problem Information

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

Problem Description

You are given a string

s

consisting of lowercase English letters (

'a'

to

'z'

).

Your task is to:

Find the vowel (one of

'a'

,

'e'

,

'i'

,

'o'

, or

'u'

) with the

maximum

frequency.

Find the consonant (all other letters excluding vowels) with the

maximum

frequency.

Return the sum of the two frequencies.

Note

: If multiple vowels or consonants have the same maximum frequency, you may choose any one of them. If there are no vowels or no consonants in the string, consider their frequency as 0.

The

frequency

of a letter

x

is the number of times it occurs in the string.

Example 1:

Input:

s = "successes"

Output:

6

Explanation:

The vowels are:

'u'

(frequency 1),

'e'

(frequency 2). The maximum frequency is 2.

The consonants are:

's'

(frequency 4),

'c'

(frequency 2). The maximum frequency is 4.

The output is

2 + 4 = 6

.

Example 2:

Input:

s = "aeiaeia"

Output:

3

Explanation:

The vowels are:

'a'

(frequency 3),

'e'

(frequency 2),

'i'

(frequency 2). The maximum frequency is 3.

There are no consonants in

s

. Hence, maximum consonant frequency = 0.

The output is

3 + 0 = 3

.

Constraints:

1 <= s.length <= 100

s

consists of lowercase English letters only.

Code Snippets

C++:

```
class Solution {
public:
int maxFreqSum(string s) {

}
};
```

Java:

```
class Solution {
public int maxFreqSum(String s) {

}
}
```

Python3:

```
class Solution:
def maxFreqSum(self, s: str) -> int:
```

Python:

```
class Solution(object):
def maxFreqSum(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var maxFreqSum = function(s) {

};
```

TypeScript:

```
function maxFreqSum(s: string): number {

};
```

C#:

```
public class Solution {
public int MaxFreqSum(string s) {

}
}
```

C:

```
int maxFreqSum(char* s) {

}
```

Go:

```
func maxFreqSum(s string) int {

}
```

Kotlin:

```
class Solution {
fun maxFreqSum(s: String): Int {

}
}
```

Swift:

```
class Solution {
func maxFreqSum(_ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_freq_sum(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def max_freq_sum(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function maxFreqSum($s) {

}
}
```

Dart:

```
class Solution {
int maxFreqSum(String s) {

}
}
```

Scala:

```
object Solution {
def maxFreqSum(s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_freq_sum(s :: String.t) :: integer
def max_freq_sum(s) do

end
end
```

Erlang:

```
-spec max_freq_sum(S :: unicode:unicode_binary()) -> integer().
max_freq_sum(S) ->
.
```

Racket:

```
(define/contract (max-freq-sum s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Most Frequent Vowel and Consonant
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public:
int maxFreqSum(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Find Most Frequent Vowel and Consonant
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int maxFreqSum(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Most Frequent Vowel and Consonant
Difficulty: Easy
Tags: string, hash

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

class Solution:
def maxFreqSum(self, s: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def maxFreqSum(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find Most Frequent Vowel and Consonant
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @return {number}
*/
var maxFreqSum = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Most Frequent Vowel and Consonant
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

function maxFreqSum(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Find Most Frequent Vowel and Consonant
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MaxFreqSum(string s) {

}
}
```

C Solution:

```
/*
* Problem: Find Most Frequent Vowel and Consonant
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int maxFreqSum(char* s) {

}
```

Go Solution:

```
// Problem: Find Most Frequent Vowel and Consonant
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func maxFreqSum(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxFreqSum(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxFreqSum(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find Most Frequent Vowel and Consonant
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn max_freq_sum(s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def max_freq_sum(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function maxFreqSum($s) {

}
}
```

Dart Solution:

```
class Solution {
int maxFreqSum(String s) {

}
}
```

Scala Solution:

```
object Solution {
def maxFreqSum(s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_freq_sum(s :: String.t) :: integer
def max_freq_sum(s) do

end
end
```

Erlang Solution:

```
-spec max_freq_sum(S :: unicode:unicode_binary()) -> integer().
max_freq_sum(S) ->
.
```

Racket Solution:

```
(define/contract (max-freq-sum s)
(-> string? exact-integer?)
)
```