Problem 2272: Substring With Largest Variance

Problem Information

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

Problem Description

The

variance

of a string is defined as the largest difference between the number of occurrences of

any

2

characters present in the string. Note the two characters may or may not be the same.

Given a string

s

consisting of lowercase English letters only, return

the

largest variance

possible among all

substrings

of

s

.

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

s = "aababbb"

Output:

3

Explanation:

All possible variances along with their respective substrings are listed below: - Variance 0 for substrings "a", "aa", "ab", "abab", "aababb", "ba", "b", "bb", and "bbb". - Variance 1 for substrings "aab", "aba", "abb", "aabab", "ababb", "aababbb", and "bab". - Variance 2 for substrings "aaba", "ababbb", "abbb", and "babb". - Variance 3 for substring "babbb". Since the largest possible variance is 3, we return it.

Example 2:

Input:

s = "abcde"

Output:

0

Explanation:

No letter occurs more than once in s, so the variance of every substring is 0.

Constraints:

1 <= s.length <= 10

4

s

consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int largestVariance(char* s) {

}
```

Go:

```
func largestVariance(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
```

```
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (largest-variance s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Substring With Largest Variance
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int largestVariance(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Substring With Largest Variance
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int largestVariance(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Substring With Largest Variance
Difficulty: Hard
Tags: array, string, tree, dp
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Substring With Largest Variance
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Substring With Largest Variance
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function largestVariance(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Substring With Largest Variance
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
}
```

C Solution:

```
/*
* Problem: Substring With Largest Variance
* Difficulty: Hard
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

int largestVariance(char* s) {

}
```

Go Solution:

```
// Problem: Substring With Largest Variance
// Difficulty: Hard
// Tags: array, string, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func largestVariance(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Substring With Largest Variance
// Difficulty: Hard
// Tags: array, string, tree, dp
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (largest-variance s)
(-> string? exact-integer?)
)
```