# Problem 3557: Find Maximum Number of Non Intersecting Substrings

## Problem Information

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

## Problem Description

You are given a string

word

.

Return the

maximum

number of non-intersecting

substrings

of word that are at

least

four characters long and start and end with the same letter.

Example 1:

Input:

word = "abcdeafdef"

Output:

2

Explanation:

The two substrings are

"abcdea"

and

"fdef"

.

Example 2:

Input:

word = "bcdaaaab"

Output:

1

Explanation:

The only substring is

"aaaa"

. Note that we cannot

also

choose

"bcdaaaab"

since it intersects with the other substring.

Constraints:

1 <= word.length <= 2 * 10

5

word

consists only of lowercase English letters.

## Code Snippets

**C++:**

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

}
};
```

**Java:**

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

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
int maxSubstrings(char* word) {

}
```

**Go:**

```
func maxSubstrings(word string) int {

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
```

```
}
```

**Dart:**

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

}
}
```

**Scala:**

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

}
}
```

**Elixir:**

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

end
end
```

**Erlang:**

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

**Racket:**

```
(define/contract (max-substrings word)
(-> string? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Find Maximum Number of Non Intersecting Substrings
* Difficulty: Medium
* Tags: string, tree, dp, greedy, hash
*
* 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:
int maxSubstrings(string word) {

}
};
```

### Java Solution:

```
/**
* Problem: Find Maximum Number of Non Intersecting Substrings
* Difficulty: Medium
* Tags: string, tree, dp, greedy, hash
*
* 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 int maxSubstrings(String word) {

}
}
```

### Python3 Solution:

```
"""
Problem: Find Maximum Number of Non Intersecting Substrings
Difficulty: Medium
Tags: string, tree, dp, greedy, hash
```

```
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 maxSubstrings(self, word: str) -> int:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

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

**JavaScript Solution:**

```
/**
* Problem: Find Maximum Number of Non Intersecting Substrings
* Difficulty: Medium
* Tags: string, tree, dp, greedy, hash
*
* 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 maxSubstrings = function(word) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Find Maximum Number of Non Intersecting Substrings
* Difficulty: Medium
* Tags: string, tree, dp, greedy, hash
*
* 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 maxSubstrings(word: string): number {

};
```

### C# Solution:

```
/*
* Problem: Find Maximum Number of Non Intersecting Substrings
* Difficulty: Medium
* Tags: string, tree, dp, greedy, hash
*
* 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 int MaxSubstrings(string word) {

}
}
```

### C Solution:

```
/*
* Problem: Find Maximum Number of Non Intersecting Substrings
* Difficulty: Medium
* Tags: string, tree, dp, greedy, hash
*
* 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
```

```
*/

int maxSubstrings(char* word) {

}
```

### Go Solution:

```
// Problem: Find Maximum Number of Non Intersecting Substrings
// Difficulty: Medium
// Tags: string, tree, dp, greedy, hash
//
// 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 maxSubstrings(word string) int {

}
```

### Kotlin Solution:

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

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Find Maximum Number of Non Intersecting Substrings
// Difficulty: Medium
// Tags: string, tree, dp, greedy, hash
```

```
//
// 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 max_substrings(word: String) -> i32 {

}
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

}
}
```

### Dart Solution:

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

}
}
```

### Scala Solution:

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

}
}
```

### Elixir Solution:

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

end
end
```

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (max-substrings word)
(-> string? exact-integer?)
)
```