Problem 1316: Distinct Echo Substrings

Problem Information

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

Problem Description

Return the number of

distinct

non-empty substrings of

text

that can be written as the concatenation of some string with itself (i.e. it can be written as

a + a

where

a

is some string).

Example 1:

Input:

text = "abcabcabc"

Output:

3

Explanation:

The 3 substrings are "abcabc", "bcabca" and "cabcab".

Example 2:

Input:

text = "leetcodeleetcode"

Output:

2

Explanation:

The 2 substrings are "ee" and "leetcodeleetcode".

Constraints:

1 <= text.length <= 2000

text

has only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int distinctEchoSubstrings(string text) {

}
};
```

Java:

```
class Solution {
public int distinctEchoSubstrings(String text) {

}
}
```

Python3:

```
class Solution:
def distinctEchoSubstrings(self, text: str) -> int:
```

Python:

```
class Solution(object):
def distinctEchoSubstrings(self, text):
"""
:type text: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} text
* @return {number}
*/
var distinctEchoSubstrings = function(text) {

};
```

TypeScript:

```
function distinctEchoSubstrings(text: string): number {

};
```

C#:

```
public class Solution {
public int DistinctEchoSubstrings(string text) {
```

```
}
}
```

C:

```
int distinctEchoSubstrings(char* text) {

}
```

Go:

```
func distinctEchoSubstrings(text string) int {

}
```

Kotlin:

```
class Solution {
fun distinctEchoSubstrings(text: String): Int {

}
}
```

Swift:

```
class Solution {
func distinctEchoSubstrings(_ text: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn distinct_echo_substrings(text: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} text
# @return {Integer}
def distinct_echo_substrings(text)

end
```

PHP:

```
class Solution {

/**
* @param String $text
* @return Integer
*/
function distinctEchoSubstrings($text) {

}
}
```

Dart:

```
class Solution {
int distinctEchoSubstrings(String text) {

}
}
```

Scala:

```
object Solution {
def distinctEchoSubstrings(text: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec distinct_echo_substrings(text :: String.t) :: integer
def distinct_echo_substrings(text) do

end
end
```

Erlang:

```
-spec distinct_echo_substrings(Text :: unicode:unicode_binary()) ->
integer().
distinct_echo_substrings(Text) ->
.
```

Racket:

```
(define/contract (distinct-echo-substrings text)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Distinct Echo Substrings
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
int distinctEchoSubstrings(string text) {

}
};
```

Java Solution:

```
/**
* Problem: Distinct Echo Substrings
* Difficulty: Hard
* Tags: string, tree, hash
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int distinctEchoSubstrings(String text) {

}
}
```

Python3 Solution:

```
"""
Problem: Distinct Echo Substrings
Difficulty: Hard
Tags: string, tree, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

```
class Solution(object):
def distinctEchoSubstrings(self, text):
"""
:type text: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Distinct Echo Substrings
```

```
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {string} text
* @return {number}
*/
var distinctEchoSubstrings = function(text) {

};
```

TypeScript Solution:

```
/**
* Problem: Distinct Echo Substrings
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function distinctEchoSubstrings(text: string): number {

};
```

C# Solution:

```
/*
* Problem: Distinct Echo Substrings
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* 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 DistinctEchoSubstrings(string text) {

}
}
```

C Solution:

```
/*
* Problem: Distinct Echo Substrings
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int distinctEchoSubstrings(char* text) {

}
```

Go Solution:

```
// Problem: Distinct Echo Substrings
// Difficulty: Hard
// Tags: string, tree, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func distinctEchoSubstrings(text string) int {

}
```

Kotlin Solution:

```
class Solution {
fun distinctEchoSubstrings(text: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func distinctEchoSubstrings(_ text: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Distinct Echo Substrings
// Difficulty: Hard
// Tags: string, tree, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn distinct_echo_substrings(text: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} text
# @return {Integer}
def distinct_echo_substrings(text)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $text
* @return Integer
*/
function distinctEchoSubstrings($text) {

}
}
```

Dart Solution:

```
class Solution {
int distinctEchoSubstrings(String text) {

}
}
```

Scala Solution:

```
object Solution {
def distinctEchoSubstrings(text: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec distinct_echo_substrings(text :: String.t) :: integer
def distinct_echo_substrings(text) do

end
end
```

Erlang Solution:

```
-spec distinct_echo_substrings(Text :: unicode:unicode_binary()) ->
integer().
distinct_echo_substrings(Text) ->
.
```

Racket Solution:

```
(define/contract (distinct-echo-substrings text)
(-> string? exact-integer?)
)
```