

Problem 3234: Count the Number of Substrings With Dominant Ones

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a binary string

s

.

Return the number of

substrings

with

dominant

ones.

A string has

dominant

ones if the number of ones in the string is

greater than or equal to

the

square

of the number of zeros in the string.

Example 1:

Input:

s = "00011"

Output:

5

Explanation:

The substrings with dominant ones are shown in the table below.

i

j

s[i..j]

Number of Zeros

Number of Ones

3

3

1

0

1

4

4

1

0

1

2

3

01

1

1

3

4

11

0

2

2

4

011

1

2

Example 2:

Input:

s = "101101"

Output:

16

Explanation:

The substrings with

non-dominant

ones are shown in the table below.

Since there are 21 substrings total and 5 of them have non-dominant ones, it follows that there are 16 substrings with dominant ones.

i

j

s[i..j]

Number of Zeros

Number of Ones

1

1

0

1

0

4

4

0

1

0

1

4

0110

2

2

0

4

10110

2

3

1

5

01101

2

3

Constraints:

$1 \leq s.length \leq 4 * 10$

4

s

consists only of characters

'0'

and

'1'

.

Code Snippets

C++:

```
class Solution {  
public:  
    int numberOfSubstrings(string s) {  
  
    }  
};
```

Java:

```
class Solution {  
    public int numberOfSubstrings(String s) {  
  
    }  
}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

    }
}
```

C:

```
int numberOfSubstrings(char* s) {  
  
}
```

Go:

```
func numberOfSubstrings(s string) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun numberOfSubstrings(s: String): Int {  
  
    }  
}
```

Swift:

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

Rust:

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

Ruby:

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

PHP:


```

class Solution {

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

  }

}

```

Dart:

```

class Solution {
  int numberOfSubstrings(String s) {

  }

}

```

Scala:

```

object Solution {
  def numberOfSubstrings(s: String): Int = {

  }

}

```

Elixir:

```

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

  end

end

```

Erlang:

```

-spec number_of_substrings(S :: unicode:unicode_binary()) -> integer().
number_of_substrings(S) ->
.

```

Racket:

```
(define/contract (number-of-substrings s)
  (-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Count the Number of Substrings With Dominant Ones
 * Difficulty: Medium
 * Tags: string, tree
 *
 * 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 numberOfSubstrings(string s) {

    }
};
```

Java Solution:

```
/**
 * Problem: Count the Number of Substrings With Dominant Ones
 * Difficulty: Medium
 * Tags: string, tree
 *
 * 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 numberOfSubstrings(String s) {

    }
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Count the Number of Substrings With Dominant Ones
Difficulty: Medium
Tags: string, tree

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

Python Solution:

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

JavaScript Solution:

```
/**
 * Problem: Count the Number of Substrings With Dominant Ones
 * Difficulty: Medium
 * Tags: string, tree
 *
 * 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} s
* @return {number}
*/
var numberOfSubstrings = function(s) {

};

```

TypeScript Solution:

```

/**
 * Problem: Count the Number of Substrings With Dominant Ones
 * Difficulty: Medium
 * Tags: string, tree
 *
 * 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 numberOfSubstrings(s: string): number {

};

```

C# Solution:

```

/*
 * Problem: Count the Number of Substrings With Dominant Ones
 * Difficulty: Medium
 * Tags: string, tree
 *
 * 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 NumberOfSubstrings(string s) {

    }
}

```

C Solution:

```
/*
 * Problem: Count the Number of Substrings With Dominant Ones
 * Difficulty: Medium
 * Tags: string, tree
 *
 * 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 numberOfSubstrings(char* s) {

}
```

Go Solution:

```
// Problem: Count the Number of Substrings With Dominant Ones
// Difficulty: Medium
// Tags: string, tree
//
// 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 numberOfSubstrings(s string) int {

}
```

Kotlin Solution:

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

    }
}
```

Swift Solution:

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

```
}  
}
```

Rust Solution:

```
// Problem: Count the Number of Substrings With Dominant Ones  
// Difficulty: Medium  
// Tags: string, tree  
//  
// 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 number_of_substrings(s: String) -> i32 {  
  
    }  
}
```

Ruby Solution:

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

PHP Solution:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @return Integer  
     */  
    function numberOfSubstrings($s) {  
  
    }  
}
```

Dart Solution:

```
class Solution {  
  int numberOfSubstrings(String s) {  
  
  }  
}
```

Scala Solution:

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

Elixir Solution:

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

Erlang Solution:

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

Racket Solution:

```
(define/contract (number-of-substrings s)  
  (-> string? exact-integer?)  
)
```