Problem 3228: Maximum Number of Operations to Move Ones to the End

Problem Information

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

Problem Description

You are given a

binary string

s

.

You can perform the following operation on the string

any

number of times:

Choose

any

index

i

from the string where

i + 1 < s.length

such that

s[i] == '1'

and

s[i + 1] == '0'

.

Move the character

s[i]

to the

right

until it reaches the end of the string or another

'1'

. For example, for

s = "010010"

, if we choose

i = 1

, the resulting string will be

s = "0

001

10"

.

Return the

maximum

number of operations that you can perform.

Example 1:

Input:

s = "1001101"

Output:

4

Explanation:

We can perform the following operations:

Choose index

i = 0

. The resulting string is

s = "

001

1101"

.

Choose index

i = 4

. The resulting string is

s = "0011

01

1"

.

Choose index

i = 3

. The resulting string is

s = "001

01

11"

.

Choose index

i = 2

. The resulting string is

s = "00

01

111"

.

Example 2:

Input:

s = "00111"

Output:

0

Constraints:

1 <= s.length <= 10

5

s[i]

is either

'0'

or

'1'

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int MaxOperations(string s) {
}
}
```

C:

```
int maxOperations(char* s) {

}
```

Go:

```
func maxOperations(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Operations to Move Ones to the End
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Operations to Move Ones to the End
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int maxOperations(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Operations to Move Ones to the End
Difficulty: Medium
Tags: string, greedy

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Number of Operations to Move Ones to the End
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Operations to Move Ones to the End
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function maxOperations(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Operations to Move Ones to the End
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

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

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Operations to Move Ones to the End
* Difficulty: Medium
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int maxOperations(char* s) {

}
```

Go Solution:

```
// Problem: Maximum Number of Operations to Move Ones to the End
// Difficulty: Medium
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func maxOperations(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Operations to Move Ones to the End
// Difficulty: Medium
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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