

# Problem 1573: Number of Ways to Split a String

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given a binary string

$s$

, you can split

$s$

into 3

non-empty

strings

$s_1$

,

$s_2$

, and

$s_3$

where

$$s_1 + s_2 + s_3 = s$$

.

Return the number of ways

$s$

can be split such that the number of ones is the same in

$s_1$

,

$s_2$

, and

$s_3$

. Since the answer may be too large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

$s = "10101"$

Output:

4

Explanation:

There are four ways to split s in 3 parts where each part contain the same number of letters '1'. "1|010|1" "1|01|01" "10|10|1" "10|1|01"

Example 2:

Input:

s = "1001"

Output:

0

Example 3:

Input:

s = "0000"

Output:

3

Explanation:

There are three ways to split s in 3 parts. "0|0|00" "0|00|0" "00|0|0"

Constraints:

$3 \leq s.length \leq 10$

5

s[i]

is either

'0'

or

'1'

.

## Code Snippets

### C++:

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

### Java:

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

### Python3:

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

### Python:

```
class Solution(object):  
    def numWays(self, s):  
        """  
        :type s: str
```

```
:rtype: int
"""
```

### JavaScript:

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

};
```

### TypeScript:

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

};
```

### C#:

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

    }
}
```

### C:

```
int numWays(char* s) {

}
```

### Go:

```
func numWays(s string) int {

}
```

### Kotlin:

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

### Swift:

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

### Rust:

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

### Ruby:

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

### PHP:

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

### Dart:

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

### Scala:

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

### Elixir:

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

### Erlang:

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

### Racket:

```
(define/contract (num-ways s)  
  (-> string? exact-integer?)  
)
```

## Solutions

### C++ Solution:

```

/*
 * Problem: Number of Ways to Split a String
 * Difficulty: Medium
 * Tags: string, math
 *
 * 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 numWays(string s) {

    }
};

```

### Java Solution:

```

/**
 * Problem: Number of Ways to Split a String
 * Difficulty: Medium
 * Tags: string, math
 *
 * 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 numWays(String s) {

    }
}

```

### Python3 Solution:

```

"""
Problem: Number of Ways to Split a String
Difficulty: Medium
Tags: string, math

```

```

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

```

### Python Solution:

```

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

```

### JavaScript Solution:

```

/**
 * Problem: Number of Ways to Split a String
 * Difficulty: Medium
 * Tags: string, math
 *
 * 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 numWays = function(s) {

};

```

### TypeScript Solution:

```

/**
 * Problem: Number of Ways to Split a String
 * Difficulty: Medium
 * Tags: string, math
 *
 * 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 numWays(s: string): number {

};

```

### C# Solution:

```

/*
 * Problem: Number of Ways to Split a String
 * Difficulty: Medium
 * Tags: string, math
 *
 * 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 NumWays(string s) {

    }
}

```

### C Solution:

```

/*
 * Problem: Number of Ways to Split a String
 * Difficulty: Medium
 * Tags: string, math
 *
 * 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 numWays(char* s) {

}

```

### Go Solution:

```

// Problem: Number of Ways to Split a String
// Difficulty: Medium
// Tags: string, math
//
// 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 numWays(s string) int {

}

```

### Kotlin Solution:

```

class Solution {
    fun numWays(s: String): Int {

    }
}

```

### Swift Solution:

```

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

    }
}

```

### Rust Solution:

```

// Problem: Number of Ways to Split a String
// Difficulty: Medium
// Tags: string, math

```

```
//
// 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 num_ways(s: String) -> i32 {

    }
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

    }

}
```

### Dart Solution:

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

    }

}
```

### Scala Solution:

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

### Elixir Solution:

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

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (num-ways s)  
  (-> string? exact-integer?)  
)
```