

Problem 1784: Check if Binary String Has at Most One Segment of Ones

Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given a binary string

`s`

without leading zeros

, return

`true`

if

`s`

contains

at most one contiguous segment of ones

. Otherwise, return

`false`

.

Example 1:

Input:

`s = "1001"`

Output:

`false`

Explanation:

The ones do not form a contiguous segment.

Example 2:

Input:

`s = "110"`

Output:

`true`

Constraints:

$1 \leq s.length \leq 100$

`s[i]`

is either

`'0'`

or

`'1'`

.

s[0]

is

'1'

.

Code Snippets

C++:

```
class Solution {  
public:  
    bool checkOnesSegment(string s) {  
  
    }  
};
```

Java:

```
class Solution {  
    public boolean checkOnesSegment(String s) {  
  
    }  
}
```

Python3:

```
class Solution:  
    def checkOnesSegment(self, s: str) -> bool:
```

Python:

```
class Solution(object):  
    def checkOnesSegment(self, s):  
        """  
        :type s: str  
        :rtype: bool  
        """
```

JavaScript:

```
/**
 * @param {string} s
 * @return {boolean}
 */
var checkOnesSegment = function(s) {

};
```

TypeScript:

```
function checkOnesSegment(s: string): boolean {

};
```

C#:

```
public class Solution {
    public bool CheckOnesSegment(string s) {

    }
}
```

C:

```
bool checkOnesSegment(char* s) {

}
```

Go:

```
func checkOnesSegment(s string) bool {

}
```

Kotlin:

```
class Solution {
    fun checkOnesSegment(s: String): Boolean {

    }
}
```

Swift:

```
class Solution {  
    func checkOnesSegment(_ s: String) -> Bool {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn check_ones_segment(s: String) -> bool {  
  
    }  
}
```

Ruby:

```
# @param {String} s  
# @return {Boolean}  
def check_ones_segment(s)  
  
end
```

PHP:

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

Dart:

```
class Solution {  
    bool checkOnesSegment(String s) {  
  
    }  
}
```

```
}
```

Scala:

```
object Solution {  
  def checkOnesSegment(s: String): Boolean = {  
  
  }  
}
```

Elixir:

```
defmodule Solution do  
  @spec check_ones_segment(s :: String.t) :: boolean  
  def check_ones_segment(s) do  
  
  end  
end
```

Erlang:

```
-spec check_ones_segment(S :: unicode:unicode_binary()) -> boolean().  
check_ones_segment(S) ->  
.
```

Racket:

```
(define/contract (check-ones-segment s)  
  (-> string? boolean?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Check if Binary String Has at Most One Segment of Ones  
 * Difficulty: Easy  
 * Tags: string  
 */
```

```

* 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:
    bool checkOnesSegment(string s) {

    }
};

```

Java Solution:

```

/**
 * Problem: Check if Binary String Has at Most One Segment of Ones
 * Difficulty: Easy
 * Tags: string
 *
 * 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 boolean checkOnesSegment(String s) {

    }
}

```

Python3 Solution:

```

"""
Problem: Check if Binary String Has at Most One Segment of Ones
Difficulty: Easy
Tags: string

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 checkOnesSegment(self, s: str) -> bool:
# TODO: Implement optimized solution
pass

```

Python Solution:

```

class Solution(object):
def checkOnesSegment(self, s):
"""
:type s: str
:rtype: bool
"""

```

JavaScript Solution:

```

/**
 * Problem: Check if Binary String Has at Most One Segment of Ones
 * Difficulty: Easy
 * Tags: string
 *
 * 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 {boolean}
 */
var checkOnesSegment = function(s) {

};

```

TypeScript Solution:

```

/**
 * Problem: Check if Binary String Has at Most One Segment of Ones
 * Difficulty: Easy
 * Tags: string

```

```

*
* 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 checkOnesSegment(s: string): boolean {

};

```

C# Solution:

```

/*
* Problem: Check if Binary String Has at Most One Segment of Ones
* Difficulty: Easy
* Tags: string
*
* 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 bool CheckOnesSegment(string s) {

    }
}

```

C Solution:

```

/*
* Problem: Check if Binary String Has at Most One Segment of Ones
* Difficulty: Easy
* Tags: string
*
* 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
*/

bool checkOnesSegment(char* s) {

```

```
}
```

Go Solution:

```
// Problem: Check if Binary String Has at Most One Segment of Ones
// Difficulty: Easy
// Tags: string
//
// 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 checkOnesSegment(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
    fun checkOnesSegment(s: String): Boolean {

    }
}
```

Swift Solution:

```
class Solution {
    func checkOnesSegment(_ s: String) -> Bool {

    }
}
```

Rust Solution:

```
// Problem: Check if Binary String Has at Most One Segment of Ones
// Difficulty: Easy
// Tags: string
//
// 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 check_ones_segment(s: String) -> bool {

    }
}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def check_ones_segment(s)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param String $s
     * @return Boolean
     */
    function checkOnesSegment($s) {

    }
}
```

Dart Solution:

```
class Solution {
    bool checkOnesSegment(String s) {

    }
}
```

Scala Solution:

```
object Solution {
    def checkOnesSegment(s: String): Boolean = {
```

```
}  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec check_ones_segment(s :: String.t) :: boolean  
  def check_ones_segment(s) do  
  
  end  
end
```

Erlang Solution:

```
-spec check_ones_segment(S :: unicode:unicode_binary()) -> boolean().  
check_ones_segment(S) ->  
.
```

Racket Solution:

```
(define/contract (check-ones-segment s)  
  (-> string? boolean?)  
)
```