

Problem 2027: Minimum Moves to Convert String

Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a string

s

consisting of

n

characters which are either

'X'

or

'O'

.

A

move

is defined as selecting

three

consecutive characters

of

s

and converting them to

'O'

. Note that if a move is applied to the character

'O'

, it will stay the

same

.

Return

the

minimum

number of moves required so that all the characters of

s

are converted to

'O'

.

Example 1:

Input:

s = "XXX"

Output:

1

Explanation:

XXX

-> OOO We select all the 3 characters and convert them in one move.

Example 2:

Input:

s = "XXOX"

Output:

2

Explanation:

XXO

X -> O

OOX

-> OOOO We select the first 3 characters in the first move, and convert them to

'O'

. Then we select the last 3 characters and convert them so that the final string contains all

'O'

s.

Example 3:

Input:

s = "OOOO"

Output:

0

Explanation:

There are no

'X's

in

s

to convert.

Constraints:

$3 \leq s.length \leq 1000$

s[i]

is either

'X'

or

'O'

Code Snippets

C++:

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

    }
};
```

Java:

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

    }
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
 * @param {string} s
 * @return {number}
 */
```

```
var minimumMoves = function(s) {  
  
};
```

TypeScript:

```
function minimumMoves(s: string): number {  
  
};
```

C#:

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

C:

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

Go:

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

Kotlin:

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

Swift:

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

```
}  
}
```

Rust:

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

Ruby:

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

PHP:

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

Dart:

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

Scala:

```

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

  }
}

```

Elixir:

```

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

  end
end

```

Erlang:

```

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

```

Racket:

```

(define/contract (minimum-moves s)
  (-> string? exact-integer?)
)

```

Solutions

C++ Solution:

```

/*
 * Problem: Minimum Moves to Convert String
 * Difficulty: Easy
 * 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 minimumMoves(string s) {

    }

};

```

Java Solution:

```

/**
 * Problem: Minimum Moves to Convert String
 * Difficulty: Easy
 * 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 minimumMoves(String s) {

    }

}

```

Python3 Solution:

```

"""
Problem: Minimum Moves to Convert String
Difficulty: Easy
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 minimumMoves(self, s: str) -> int:
        # TODO: Implement optimized solution
        pass

```

Python Solution:

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

JavaScript Solution:

```
/**
 * Problem: Minimum Moves to Convert String
 * Difficulty: Easy
 * 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 minimumMoves = function(s) {

};
```

TypeScript Solution:

```
/**
 * Problem: Minimum Moves to Convert String
 * Difficulty: Easy
 * 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 minimumMoves(s: string): number {
```

```
};
```

C# Solution:

```
/*
 * Problem: Minimum Moves to Convert String
 * Difficulty: Easy
 * 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 MinimumMoves(string s) {

    }
}
```

C Solution:

```
/*
 * Problem: Minimum Moves to Convert String
 * Difficulty: Easy
 * 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 minimumMoves(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Moves to Convert String
// Difficulty: Easy
```

```
// 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 minimumMoves(s string) int {

}
```

Kotlin Solution:

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

    }
}
```

Swift Solution:

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

    }
}
```

Rust Solution:

```
// Problem: Minimum Moves to Convert String
// Difficulty: Easy
// 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 minimum_moves(s: String) -> i32 {

    }
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

    }

}
```

Dart Solution:

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

  }

}
```

Scala Solution:

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

  }

}
```

Elixir Solution:

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

```
end  
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (minimum-moves s)  
  (-> string? exact-integer?)  
)
```