Problem 3227: Vowels Game in a String

Problem Information

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

Problem Description

Alice and Bob are playing a game on a string.

You are given a string

s

, Alice and Bob will take turns playing the following game where Alice starts

first

:

On Alice's turn, she has to remove any

non-empty

substring

from

s

that contains an

odd

number of vowels.

On Bob's turn, he has to remove any

non-empty

substring

from

s

that contains an

even

number of vowels.

The first player who cannot make a move on their turn loses the game. We assume that both Alice and Bob play

optimally

.

Return

true

if Alice wins the game, and

false

otherwise.

The English vowels are:

a

,

e

,

i

,

o

, and

u

.

Example 1:

Input:

s = "leetcoder"

Output:

true

Explanation:

Alice can win the game as follows:

Alice plays first, she can delete the underlined substring in

s = "

leetco

der"

which contains 3 vowels. The resulting string is

s = "der"

.

Bob plays second, he can delete the underlined substring in

s = "

d

er"

which contains 0 vowels. The resulting string is

s = "er"

.

Alice plays third, she can delete the whole string

s = "

er

"

which contains 1 vowel.

Bob plays fourth, since the string is empty, there is no valid play for Bob. So Alice wins the game.

Example 2:

Input:

s = "bbcd"

Output:

false

Explanation:

There is no valid play for Alice in her first turn, so Alice loses the game.

Constraints:

1 <= s.length <= 10

5

s

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool doesAliceWin(string s) {

}
};
```

Java:

```
class Solution {
public boolean doesAliceWin(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool doesAliceWin(char* s) {

}
```

Go:

```
func doesAliceWin(s string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
```

```
*/
function doesAliceWin($s) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (does-alice-win s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Vowels Game in a String
* Difficulty: Medium
* Tags: string, tree, math
*
* 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:
bool doesAliceWin(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Vowels Game in a String
* Difficulty: Medium
* Tags: string, tree, math
*
* 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 boolean doesAliceWin(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Vowels Game in a String
Difficulty: Medium
Tags: string, tree, math

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Vowels Game in a String
* Difficulty: Medium
* Tags: string, tree, math
*
* 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 {boolean}
*/
var doesAliceWin = function(s) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Vowels Game in a String
* Difficulty: Medium
* Tags: string, tree, math
*
* 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 doesAliceWin(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Vowels Game in a String
* Difficulty: Medium
* Tags: string, tree, math
*
* 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 bool DoesAliceWin(string s) {

}
}
```

C Solution:

```
/*
* Problem: Vowels Game in a String
* Difficulty: Medium
```

```
* Tags: string, tree, math
*
* 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
*/

bool doesAliceWin(char* s) {

}
```

Go Solution:

```
// Problem: Vowels Game in a String
// Difficulty: Medium
// Tags: string, tree, math
//
// 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 doesAliceWin(s string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Vowels Game in a String
// Difficulty: Medium
// Tags: string, tree, math
//
// 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 does_alice_win(s: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (does-alice-win s)
(-> string? boolean?)
)
```