Problem 3707: Equal Score Substrings

Problem Information

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

Problem Description

You are given a string

s

consisting of lowercase English letters.

The

score

of a string is the sum of the positions of its characters in the alphabet, where

'a' = 1

,

'b' = 2

, ...,

'z' = 26

.

Determine whether there exists an index

i

such that the string can be split into two

non-empty

substrings

s[0..i]

and

s[(i + 1)..(n - 1)]

that have

equal

scores.

Return

true

if such a split exists, otherwise return

false

.

Example 1:

Input:

s = "adcb"

Output:

true

Explanation:

Split at index

i = 1

:

Left substring =

s[0..1] = "ad"

with

score = 1 + 4 = 5

Right substring =

s[2..3] = "cb"

with

score = 3 + 2 = 5

Both substrings have equal scores, so the output is

true

.

Example 2:

Input:

s = "bace"

Output:

false

Explanation:

No split produces equal scores, so the output is

false

.

Constraints:

2 <= s.length <= 100

s

consists of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool scoreBalance(char* s) {

}
```

Go:

```
func scoreBalance(s string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (score-balance s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Equal Score Substrings
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
bool scoreBalance(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Equal Score Substrings
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public boolean scoreBalance(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Equal Score Substrings
Difficulty: Easy
Tags: array, string, tree

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Equal Score Substrings
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* 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 scoreBalance = function(s) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Equal Score Substrings
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function scoreBalance(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Equal Score Substrings
* Difficulty: Easy
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* 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 ScoreBalance(string s) {

}
}
```

C Solution:

```
/*
* Problem: Equal Score Substrings
* Difficulty: Easy
```

```
* Tags: array, string, tree
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

bool scoreBalance(char* s) {

}
```

Go Solution:

```
// Problem: Equal Score Substrings
// Difficulty: Easy
// Tags: array, string, tree
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func scoreBalance(s string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Equal Score Substrings
// Difficulty: Easy
// Tags: array, string, tree
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (score-balance s)
(-> string? boolean?)
)
```