Problem 1003: Check If Word Is Valid After Substitutions

Problem Information

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

Problem Description

Given a string

s

, determine if it is

valid

.

A string

s

is

valid

if, starting with an empty string

t = ""

, you can

transform

t

into

s

after performing the following operation

any number of times

:

Insert string

"abc"

into any position in

t

. More formally,

t

becomes

t

left

+ "abc" + t

right

, where

t == t

left

+ t

right

. Note that

t

left

and

t

right

may be

empty

.

Return

true

if

s

is a

valid

string, otherwise, return

false

.

Example 1:

Input:

s = "aabcbc"

Output:

true

Explanation:

"" -> "

abc

" -> "a

abc

bc" Thus, "aabcbc" is valid.

Example 2:

Input:

s = "abcabcababcc"

Output:

true

Explanation:

"" -> "

abc

" -> "abc

abc

" -> "abcabc

abc

" -> "abcabcab

abc

c" Thus, "abcabcababcc" is valid.

Example 3:

Input:

s = "abccba"

Output:

false

Explanation:

It is impossible to get "abccba" using the operation.

Constraints:

1 <= s.length <= 2 * 10

4

s

consists of letters

'a'

,

'b'

, and

'c'

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
:rtype: bool
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isValid(char* s) {

}
```

Go:

```
func isValid(s string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-valid s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check If Word Is Valid After Substitutions
* Difficulty: Medium
* Tags: string, stack
*
* 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 isValid(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Check If Word Is Valid After Substitutions
* Difficulty: Medium
* Tags: string, stack
*
* 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 isValid(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Check If Word Is Valid After Substitutions
Difficulty: Medium
Tags: string, stack
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check If Word Is Valid After Substitutions
* Difficulty: Medium
* Tags: string, stack
*
* 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 isValid = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Check If Word Is Valid After Substitutions
* Difficulty: Medium
* Tags: string, stack
*
* 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 isValid(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Check If Word Is Valid After Substitutions
* Difficulty: Medium
* Tags: string, stack
*
* 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 IsValid(string s) {

}
}
```

C Solution:

```
/*
* Problem: Check If Word Is Valid After Substitutions
* Difficulty: Medium
* Tags: string, stack
*
* 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 isValid(char* s) {

}
```

Go Solution:

```
// Problem: Check If Word Is Valid After Substitutions
// Difficulty: Medium
// Tags: string, stack
//
// 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 isValid(s string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check If Word Is Valid After Substitutions
// Difficulty: Medium
// Tags: string, stack
```

```
//
// 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 is_valid(s: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-valid s)
(-> string? boolean?)
)
```