Problem 2645: Minimum Additions to Make Valid String

Problem Information

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

Problem Description

Given a string

word

to which you can insert letters "a", "b" or "c" anywhere and any number of times, return

the minimum number of letters that must be inserted so that

word

becomes

valid

.

A string is called

valid

if it can be formed by concatenating the string "abc" several times.

Example 1:

Input:

word = "b"

Output:

2

Explanation:

Insert the letter "a" right before "b", and the letter "c" right next to "b" to obtain the valid string "

a

b

c

".

Example 2:

Input:

word = "aaa"

Output:

6

Explanation:

Insert letters "b" and "c" next to each "a" to obtain the valid string "a

bc

a

bc

a

bc

".

Example 3:

Input:

word = "abc"

Output:

0

Explanation:

word is already valid. No modifications are needed.

Constraints:

1 <= word.length <= 50

word

consists of letters "a", "b" and "c" only.

Code Snippets

C++:

```
class Solution {
public:
int addMinimum(string word) {

}
};
```

Java:

```
class Solution {
public int addMinimum(String word) {

}
}
```

Python3:

```
class Solution:
def addMinimum(self, word: str) -> int:
```

Python:

```
class Solution(object):
def addMinimum(self, word):
"""
:type word: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} word
* @return {number}
*/
var addMinimum = function(word) {

};
```

TypeScript:

```
function addMinimum(word: string): number {

};
```

C#:

```
public class Solution {
public int AddMinimum(string word) {
}
}
```

C:

```
int addMinimum(char* word) {

}
```

Go:

```
func addMinimum(word string) int {

}
```

Kotlin:

```
class Solution {
fun addMinimum(word: String): Int {

}
}
```

Swift:

```
class Solution {
func addMinimum(_ word: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn add_minimum(word: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} word
# @return {Integer}
def add_minimum(word)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @return Integer
*/
function addMinimum($word) {

}
}
```

Dart:

```
class Solution {
int addMinimum(String word) {

}
}
```

Scala:

```
object Solution {
def addMinimum(word: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec add_minimum(word :: String.t) :: integer
def add_minimum(word) do

end
end
```

Erlang:

```
-spec add_minimum(Word :: unicode:unicode_binary()) -> integer().
add_minimum(Word) ->
.
```

Racket:

```
(define/contract (add-minimum word)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Additions to Make Valid String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int addMinimum(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Additions to Make Valid String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int addMinimum(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Additions to Make Valid String
Difficulty: Medium
Tags: string, dp, greedy, stack

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def addMinimum(self, word: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def addMinimum(self, word):
"""
:type word: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Additions to Make Valid String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} word
* @return {number}
*/
var addMinimum = function(word) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Additions to Make Valid String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function addMinimum(word: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Additions to Make Valid String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

public class Solution {
public int AddMinimum(string word) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Additions to Make Valid String
* Difficulty: Medium
* Tags: string, dp, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int addMinimum(char* word) {

}
```

Go Solution:

```
// Problem: Minimum Additions to Make Valid String
// Difficulty: Medium
// Tags: string, dp, greedy, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func addMinimum(word string) int {

}
```

Kotlin Solution:

```
class Solution {
fun addMinimum(word: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func addMinimum(_ word: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Additions to Make Valid String
// Difficulty: Medium
// Tags: string, dp, greedy, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn add_minimum(word: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} word
# @return {Integer}
def add_minimum(word)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $word
* @return Integer
*/
function addMinimum($word) {

}
}
```

Dart Solution:

```
class Solution {
int addMinimum(String word) {

}
}
```

Scala Solution:

```
object Solution {
def addMinimum(word: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec add_minimum(word :: String.t) :: integer
def add_minimum(word) do

end
end
```

Erlang Solution:

```
-spec add_minimum(Word :: unicode:unicode_binary()) -> integer().
add_minimum(Word) ->
.
```

Racket Solution:

```
(define/contract (add-minimum word)
(-> string? exact-integer?)
)
```