Problem 1653: Minimum Deletions to Make String Balanced

Problem Information

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

Problem Description

You are given a string

s

consisting only of characters

'a'

and

'b'

.

You can delete any number of characters in

s

to make

s

balanced

.

s

is

balanced

if there is no pair of indices

(i,j)

such that

i < j

and

s[i] = 'b'

and

s[j]= 'a'

.

Return

the

minimum

number of deletions needed to make

s

balanced

.

Example 1:

Input:

s = "aababbab"

Output:

2

Explanation:

You can either: Delete the characters at 0-indexed positions 2 and 6 ("aa

b

abb

a

b" -> "aaabbb"), or Delete the characters at 0-indexed positions 3 and 6 ("aab

a

bb

a

b" -> "aabbbb").

Example 2:

Input:

s = "bbaaaaabb"

Output:

2

Explanation:

The only solution is to delete the first two characters.

Constraints:

1 <= s.length <= 10

5

s[i]

is

'a'

or

'b'

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var minimumDeletions = function(s) {

};
```

TypeScript:

```
function minimumDeletions(s: string): number {

};
```

C#:

```
public class Solution {
public int MinimumDeletions(string s) {

}
}
```

C:

```
int minimumDeletions(char* s) {

}
```

Go:

```
func minimumDeletions(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Minimum Deletions to Make String Balanced
* Difficulty: Medium
* Tags: string, dp, 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 minimumDeletions(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Deletions to Make String Balanced
* Difficulty: Medium
* Tags: string, dp, 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 minimumDeletions(String s) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Minimum Deletions to Make String Balanced
Difficulty: Medium
Tags: string, dp, 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 minimumDeletions(self, s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Deletions to Make String Balanced
* Difficulty: Medium
* Tags: string, dp, 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} s
* @return {number}
*/
var minimumDeletions = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Deletions to Make String Balanced
* Difficulty: Medium
* Tags: string, dp, 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 minimumDeletions(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Deletions to Make String Balanced
* Difficulty: Medium
* Tags: string, dp, 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 MinimumDeletions(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Deletions to Make String Balanced
* Difficulty: Medium
* Tags: string, dp, 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 minimumDeletions(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Deletions to Make String Balanced
// Difficulty: Medium
// Tags: string, dp, 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 minimumDeletions(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Minimum Deletions to Make String Balanced
// Difficulty: Medium
// Tags: string, dp, 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 minimum_deletions(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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