Problem 1737: Change Minimum Characters to Satisfy One of Three Conditions

Problem Information

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

Problem Description

You are given two strings

a

and

b

that consist of lowercase letters. In one operation, you can change any character in

a

or

b

to

any lowercase letter

.

Your goal is to satisfy

one

of the following three conditions:

Every

letter in

a

is

strictly less

than

every

letter in

b

in the alphabet.

Every

letter in

b

is

strictly less

than

every

letter in

a

in the alphabet.

Both

a

and

b

consist of

only one

distinct letter.

Return

the

minimum

number of operations needed to achieve your goal.

Example 1:

Input:

a = "aba", b = "caa"

Output:

2

Explanation:

Consider the best way to make each condition true: 1) Change b to "ccc" in 2 operations, then every letter in a is less than every letter in b. 2) Change a to "bbb" and b to "aaa" in 3 operations, then every letter in b is less than every letter in a. 3) Change a to "aaa" and b to "aaa" in 2 operations, then a and b consist of one distinct letter. The best way was done in 2 operations (either condition 1 or condition 3).

Example 2:

Input:

a = "dabadd", b = "cda"

Output:

3

Explanation:

The best way is to make condition 1 true by changing b to "eee".

Constraints:

1 <= a.length, b.length <= 10

5

a

and

b

consist only of lowercase letters.

Code Snippets

C++:

```
class Solution {
public:
int minCharacters(string a, string b) {

}
};
```

Java:

```
class Solution {
public int minCharacters(String a, String b) {

}
}
```

Python3:

```
class Solution:
def minCharacters(self, a: str, b: str) -> int:
```

Python:

```
class Solution(object):
def minCharacters(self, a, b):
"""
:type a: str
:type b: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} a
* @param {string} b
* @return {number}
*/
var minCharacters = function(a, b) {

};
```

TypeScript:

```
function minCharacters(a: string, b: string): number {

};
```

C#:

```
public class Solution {
public int MinCharacters(string a, string b) {

}
}
```

C:

```
int minCharacters(char* a, char* b) {

}
```

Go:

```
func minCharacters(a string, b string) int {

}
```

Kotlin:

```
class Solution {
fun minCharacters(a: String, b: String): Int {

}
}
```

Swift:

```
class Solution {
func minCharacters(_ a: String, _ b: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_characters(a: String, b: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} a
# @param {String} b
# @return {Integer}
def min_characters(a, b)

end
```

PHP:

```
class Solution {

/**
* @param String $a
* @param String $b
* @return Integer
*/
function minCharacters($a, $b) {

}
}
```

Dart:

```
class Solution {
int minCharacters(String a, String b) {

}
}
```

Scala:

```
object Solution {
def minCharacters(a: String, b: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_characters(a :: String.t, b :: String.t) :: integer
def min_characters(a, b) do

end
end
```

Erlang:

```
-spec min_characters(A :: unicode:unicode_binary(), B ::
unicode:unicode_binary()) -> integer().
min_characters(A, B) ->
.
```

Racket:

```
(define/contract (min-characters a b)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Change Minimum Characters to Satisfy One of Three Conditions
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
```

```
int minCharacters(string a, string b) {

}
};
```

Java Solution:

```
/**
* Problem: Change Minimum Characters to Satisfy One of Three Conditions
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minCharacters(String a, String b) {

}
}
```

Python3 Solution:

```
"""
Problem: Change Minimum Characters to Satisfy One of Three Conditions
Difficulty: Medium
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def minCharacters(self, a: str, b: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minCharacters(self, a, b):
"""
:type a: str
:type b: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Change Minimum Characters to Satisfy One of Three Conditions
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} a
* @param {string} b
* @return {number}
*/
var minCharacters = function(a, b) {

};
```

TypeScript Solution:

```
/**
* Problem: Change Minimum Characters to Satisfy One of Three Conditions
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function minCharacters(a: string, b: string): number {
};
```

C# Solution:

```
/*
* Problem: Change Minimum Characters to Satisfy One of Three Conditions
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MinCharacters(string a, string b) {

}
}
```

C Solution:

```
/*
* Problem: Change Minimum Characters to Satisfy One of Three Conditions
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minCharacters(char* a, char* b) {

}
```

Go Solution:

```
// Problem: Change Minimum Characters to Satisfy One of Three Conditions
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func minCharacters(a string, b string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minCharacters(a: String, b: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minCharacters(_ a: String, _ b: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Change Minimum Characters to Satisfy One of Three Conditions
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn min_characters(a: String, b: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} a
# @param {String} b
# @return {Integer}
def min_characters(a, b)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $a
* @param String $b
* @return Integer
*/
function minCharacters($a, $b) {

}
}
```

Dart Solution:

```
class Solution {
int minCharacters(String a, String b) {

}
}
```

Scala Solution:

```
object Solution {
def minCharacters(a: String, b: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_characters(a :: String.t, b :: String.t) :: integer
def min_characters(a, b) do

end
end
```

Erlang Solution:

```
-spec min_characters(A :: unicode:unicode_binary(), B ::
unicode:unicode_binary()) -> integer().
min_characters(A, B) ->
.
```

Racket Solution:

```
(define/contract (min-characters a b)
(-> string? string? exact-integer?)
)
```