Problem 3675: Minimum Operations to Transform String

Problem Information

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

Problem Description

You are given a string

s

consisting only of lowercase English letters.

You can perform the following operation any number of times (including zero):

Choose any character

c

in the string and replace

every

occurrence of

c

with the

next

lowercase letter in the English alphabet.

Return the

minimum

number of operations required to transform

s

into a string consisting of

only

'a'

characters.

Note:

Consider the alphabet as circular, thus

'a'

comes after

'z'

.

Example 1:

Input:

s = "yz"

Output:

2

Explanation:

Change

'y'

to

'z'

to get

"zz"

.

Change

'z'

to

'a'

to get

"aa"

.

Thus, the answer is 2.

Example 2:

Input:

s = "a"

Output:

0

Explanation:

The string

"a"

only consists of

'a'

characters. Thus, the answer is 0.

Constraints:

1 <= s.length <= 5 * 10

5

s

consists only of lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int minOperations(String s) {
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int minOperations(char* s) {

}
```

Go:

```
func minOperations(s string) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (min-operations s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Operations to Transform String
* Difficulty: Medium
* Tags: string, greedy
*
* 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:
int minOperations(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Operations to Transform String
* Difficulty: Medium
* Tags: string, greedy
*
* 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 int minOperations(String s) {
}
}
```

Python3 Solution:

```
"""
Problem: Minimum Operations to Transform String
Difficulty: Medium
Tags: string, greedy

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Operations to Transform String
* Difficulty: Medium
* Tags: string, greedy
*
* 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 {number}
*/
var minOperations = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Operations to Transform String
* Difficulty: Medium
* Tags: string, greedy
*
* 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 minOperations(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Operations to Transform String
* Difficulty: Medium
* Tags: string, greedy
*
* 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 int MinOperations(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Operations to Transform String
* Difficulty: Medium
* Tags: string, greedy
*
* 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
*/

int minOperations(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Operations to Transform String
// Difficulty: Medium
// Tags: string, greedy
//
// 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 minOperations(s string) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Minimum Operations to Transform String
// Difficulty: Medium
// Tags: string, greedy
//
// 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 min_operations(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (min-operations s)
(-> string? exact-integer?)
)
```