Problem 3335: Total Characters in String After Transformations I

Problem Information

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

Problem Description

You are given a string

s

and an integer

t

, representing the number of

transformations

to perform. In one

transformation

, every character in

s

is replaced according to the following rules:

If the character is

'z'

, replace it with the string

"ab"

.

Otherwise, replace it with the

next

character in the alphabet. For example,

'a'

is replaced with

'b'

,

'b'

is replaced with

'c'

, and so on.

Return the

length

of the resulting string after

exactly

t

transformations.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

s = "abcyy", t = 2

Output:

7

Explanation:

First Transformation (t = 1)

:

'a'

becomes

'b'

'b'

becomes

'c'

'c'

becomes

'd'

'y'

becomes

'z'

'y'

becomes

'z'

String after the first transformation:

"bcdzz"

Second Transformation (t = 2)

:

'b'

becomes

'c'

'c'

becomes

'd'

'd'

becomes

'e'

'z'

becomes

"ab"

'z'

becomes

"ab"

String after the second transformation:

"cdeabab"

Final Length of the string

: The string is

"cdeabab"

, which has 7 characters.

Example 2:

Input:

s = "azbk", t = 1

Output:

5

Explanation:

First Transformation (t = 1)

:

'a'

becomes

'b'

'z'

becomes

"ab"

'b'

becomes

'c'

'k'

becomes

'l'

String after the first transformation:

"babcl"

Final Length of the string

: The string is

"babcl"

, which has 5 characters.

Constraints:

1 <= s.length <= 10

5

s

consists only of lowercase English letters.

1 <= t <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int lengthAfterTransformations(string s, int t) {

}
};
```

Java:

```
class Solution {
public int lengthAfterTransformations(String s, int t) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function lengthAfterTransformations(s: string, t: number): number {

};
```

C#:

```
public class Solution {
public int LengthAfterTransformations(string s, int t) {

}
}
```

C:

```
int lengthAfterTransformations(char* s, int t) {

}
```

Go:

```
func lengthAfterTransformations(s string, t int) int {

}
```

Kotlin:

```
class Solution {
fun lengthAfterTransformations(s: String, t: Int): Int {

}
}
```

Swift:

```
class Solution {
func lengthAfterTransformations(_ s: String, _ t: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn length_after_transformations(s: String, t: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} t
# @return {Integer}
def length_after_transformations(s, t)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $t
* @return Integer
*/
function lengthAfterTransformations($s, $t) {

}
}
```

Dart:

```
class Solution {
int lengthAfterTransformations(String s, int t) {

}
}
```

Scala:

```
object Solution {
def lengthAfterTransformations(s: String, t: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec length_after_transformations(S :: unicode:unicode_binary(), T ::
integer()) -> integer().
```

```
length_after_transformations(S, T) ->
.
```

Racket:

```
(define/contract (length-after-transformations s t)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Total Characters in String After Transformations I
* Difficulty: Medium
* Tags: string, dp, math, hash
*
* 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 lengthAfterTransformations(string s, int t) {

}
};
```

Java Solution:

```
/**
* Problem: Total Characters in String After Transformations I
* Difficulty: Medium
* Tags: string, dp, math, hash
*
* 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 lengthAfterTransformations(String s, int t) {

}
}
```

Python3 Solution:

```
"""
Problem: Total Characters in String After Transformations I
Difficulty: Medium
Tags: string, dp, math, hash

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Total Characters in String After Transformations I
* Difficulty: Medium
* Tags: string, dp, math, hash
```

```
*
* 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
* @param {number} t
* @return {number}
*/
var lengthAfterTransformations = function(s, t) {

};
```

TypeScript Solution:

```
/**
* Problem: Total Characters in String After Transformations I
* Difficulty: Medium
* Tags: string, dp, math, hash
*
* 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 lengthAfterTransformations(s: string, t: number): number {

};
```

C# Solution:

```
/*
* Problem: Total Characters in String After Transformations I
* Difficulty: Medium
* Tags: string, dp, math, hash
*
* 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 LengthAfterTransformations(string s, int t) {

}
}
```

C Solution:

```
/*
* Problem: Total Characters in String After Transformations I
* Difficulty: Medium
* Tags: string, dp, math, hash
*
* 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 lengthAfterTransformations(char* s, int t) {

}
```

Go Solution:

```
// Problem: Total Characters in String After Transformations I
// Difficulty: Medium
// Tags: string, dp, math, hash
//
// 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 lengthAfterTransformations(s string, t int) int {

}
```

Kotlin Solution:

```
class Solution {
fun lengthAfterTransformations(s: String, t: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func lengthAfterTransformations(_ s: String, _ t: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Total Characters in String After Transformations I
// Difficulty: Medium
// Tags: string, dp, math, hash
//
// 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 length_after_transformations(s: String, t: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} t
# @return {Integer}
def length_after_transformations(s, t)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $t
* @return Integer
*/
function lengthAfterTransformations($s, $t) {

}
}
```

Dart Solution:

```
class Solution {
int lengthAfterTransformations(String s, int t) {

}
}
```

Scala Solution:

```
object Solution {
def lengthAfterTransformations(s: String, t: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec length_after_transformations(S :: unicode:unicode_binary(), T ::
integer()) -> integer().
length_after_transformations(S, T) ->
```

```
.
```

Racket Solution:

```
(define/contract (length-after-transformations s t)
(-> string? exact-integer? exact-integer?)
)
```