

# Problem 3337: Total Characters in String After Transformations II

## Problem Information

Difficulty: Hard

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

You are given a string

s

consisting of lowercase English letters, an integer

t

representing the number of

transformations

to perform, and an array

nums

of size 26. In one

transformation

, every character in

s

is replaced according to the following rules:

Replace

$s[i]$

with the

next

$\text{nums}[s[i] - 'a']$

consecutive characters in the alphabet. For example, if

$s[i] = 'a'$

and

$\text{nums}[0] = 3$

, the character

'a'

transforms into the next 3 consecutive characters ahead of it, which results in

"bcd"

.

The transformation

wraps

around the alphabet if it exceeds

'z'

. For example, if

`s[i] = 'y'`

and

`nums[24] = 3`

, the character

'y'

transforms into the next 3 consecutive characters ahead of it, which results in

"zab"

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:

## Output:

7

### Explanation:

## First Transformation ( $t = 1$ ):

'a'

becomes

'b'

as

`nums[0] == 1`

'b'

becomes

'C'

as

nums[1] == 1

'C'

becomes

'd'

as

nums[2] == 1

'y'

becomes

'z'

as

nums[24] == 1

'y'

becomes

'z'

as

nums[24] == 1

String after the first transformation:

"bcdzz"

Second Transformation (t = 2):

'b'

becomes

'c'

as

nums[1] == 1

'c'

becomes

'd'

as

nums[2] == 1

'd'

becomes

'e'

as

nums[3] == 1

'z'

becomes

'ab'

as

nums[25] == 2

'z'

becomes

'ab'

as

nums[25] == 2

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, nums = [2,2]

Output:

8

Explanation:

First Transformation (t = 1):

'a'

becomes

'bc'

as

nums[0] == 2

'z'

becomes

'ab'

as

nums[25] == 2

'b'

becomes

'cd'

as

nums[1] == 2

'k'

becomes

'lm'

as

nums[10] == 2

String after the first transformation:

"bcabcdlm"

Final Length of the string:

The string is

"bcabcdlm"

, which has 8 characters.

Constraints:

$1 \leq s.length \leq 10$

5

s

consists only of lowercase English letters.

$1 \leq t \leq 10$

9

$\text{nums.length} == 26$

$1 \leq \text{nums}[i] \leq 25$

## Code Snippets

### C++:

```
class Solution {
public:
    int lengthAfterTransformations(string s, int t, vector<int>& nums) {
        }
};
```

### Java:

```
class Solution {
public int lengthAfterTransformations(String s, int t, List<Integer> nums) {
        }
}
```

### Python3:

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

### Python:

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

### JavaScript:

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

### TypeScript:

```
function lengthAfterTransformations(s: string, t: number, nums: number[]):  
    number {  
  
};
```

### C#:

```
public class Solution {  
    public int LengthAfterTransformations(string s, int t, IList<int> nums) {  
  
    }  
}
```

### C:

```
int lengthAfterTransformations(char* s, int t, int* nums, int numsSize) {  
  
}
```

### Go:

```
func lengthAfterTransformations(s string, t int, nums []int) int {  
  
}
```

### Kotlin:

```
class Solution {  
    fun lengthAfterTransformations(s: String, t: Int, nums: List<Int>): Int {  
  
    }  
}
```

### Swift:

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

### Rust:

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

### Ruby:

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

```
end
```

### PHP:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @param Integer $t  
     * @param Integer[] $nums  
     * @return Integer  
     */  
    function lengthAfterTransformations($s, $t, $nums) {  
  
    }  
}
```

### Dart:

```
class Solution {  
int lengthAfterTransformations(String s, int t, List<int> nums) {  
  
}  
}
```

### Scala:

```
object Solution {  
def lengthAfterTransformations(s: String, t: Int, nums: List[Int]): Int = {  
  
}  
}
```

### Elixir:

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

```
end
```

### Erlang:

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

### Racket:

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

## Solutions

### C++ Solution:

```
/*  
 * Problem: Total Characters in String After Transformations II  
 * Difficulty: Hard  
 * Tags: array, string, dp, math, hash  
 *  
 * Approach: Use two pointers or sliding window technique  
 * 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, vector<int>& numbs) {  
        }  
    };
```

### Java Solution:

```
/**  
 * Problem: Total Characters in String After Transformations II
```

```

* Difficulty: Hard
* Tags: array, string, dp, math, hash
*
* Approach: Use two pointers or sliding window technique
* 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, List<Integer> nums) {
        return 0;
    }
}

```

### Python3 Solution:

```

"""
Problem: Total Characters in String After Transformations II
Difficulty: Hard
Tags: array, string, dp, math, hash

Approach: Use two pointers or sliding window technique
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, nums: List[int]) -> int:
        # TODO: Implement optimized solution
        pass

```

### Python Solution:

```

class Solution(object):
    def lengthAfterTransformations(self, s, t, nums):
        """
        :type s: str
        :type t: int
        :type nums: List[int]
        :rtype: int
        """

```

### JavaScript Solution:

```
/**  
 * Problem: Total Characters in String After Transformations II  
 * Difficulty: Hard  
 * Tags: array, string, dp, math, hash  
 *  
 * Approach: Use two pointers or sliding window technique  
 * 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  
 * @param {number[]} nums  
 * @return {number}  
 */  
var lengthAfterTransformations = function(s, t, nums) {  
  
};
```

### TypeScript Solution:

```
/**  
 * Problem: Total Characters in String After Transformations II  
 * Difficulty: Hard  
 * Tags: array, string, dp, math, hash  
 *  
 * Approach: Use two pointers or sliding window technique  
 * 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, nums: number[]):  
number {  
  
};
```

### C# Solution:

```

/*
 * Problem: Total Characters in String After Transformations II
 * Difficulty: Hard
 * Tags: array, string, dp, math, hash
 *
 * Approach: Use two pointers or sliding window technique
 * 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, IList<int> nums) {

    }
}

```

## C Solution:

```

/*
 * Problem: Total Characters in String After Transformations II
 * Difficulty: Hard
 * Tags: array, string, dp, math, hash
 *
 * Approach: Use two pointers or sliding window technique
 * 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, int* nums, int numssize) {

}

```

## Go Solution:

```

// Problem: Total Characters in String After Transformations II
// Difficulty: Hard
// Tags: array, string, dp, math, hash
//
// Approach: Use two pointers or sliding window technique
// 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, nums []int) int {  
    }  
}
```

### Kotlin Solution:

```
class Solution {  
    fun lengthAfterTransformations(s: String, t: Int, nums: List<Int>): Int {  
        }  
    }
```

### Swift Solution:

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

### Rust Solution:

```
// Problem: Total Characters in String After Transformations II  
// Difficulty: Hard  
// Tags: array, string, dp, math, hash  
//  
// Approach: Use two pointers or sliding window technique  
// 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, nums: Vec<i32>) -> i32  
    {  
        }  
    }
```

### Ruby Solution:

```

# @param {String} s
# @param {Integer} t
# @param {Integer[]} nums
# @return {Integer}
def length_after_transformations(s, t, nums)

end

```

### PHP Solution:

```

class Solution {

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

    }
}

```

### Dart Solution:

```

class Solution {
  int lengthAfterTransformations(String s, int t, List<int> nums) {
    }
}

```

### Scala Solution:

```

object Solution {
  def lengthAfterTransformations(s: String, t: Int, nums: List[Int]): Int = {
    }
}

```

### Elixir Solution:

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

### Erlang Solution:

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

### Racket Solution:

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