Problem 3031: Minimum Time to Revert Word to Initial State II

Problem Information

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

Problem Description

You are given a

0-indexed

string

word

and an integer

k

.

At every second, you must perform the following operations:

Remove the first

k

characters of

word

.

Add any

k

characters to the end of

word

.

Note

that you do not necessarily need to add the same characters that you removed. However, you must perform

both

operations at every second.

Return

the

minimum

time greater than zero required for

word

to revert to its

initial

state

.

Example 1:

Input:

word = "abacaba", k = 3

Output:

2

Explanation:

At the 1st second, we remove characters "aba" from the prefix of word, and add characters "bac" to the end of word. Thus, word becomes equal to "cababac". At the 2nd second, we remove characters "cab" from the prefix of word, and add "aba" to the end of word. Thus, word becomes equal to "abacaba" and reverts to its initial state. It can be shown that 2 seconds is the minimum time greater than zero required for word to revert to its initial state.

Example 2:

Input:

word = "abacaba", k = 4

Output:

1

Explanation:

At the 1st second, we remove characters "abac" from the prefix of word, and add characters "caba" to the end of word. Thus, word becomes equal to "abacaba" and reverts to its initial state. It can be shown that 1 second is the minimum time greater than zero required for word to revert to its initial state.

Example 3:

Input:

word = "abcbabcd", k = 2

Output:

4

Explanation:

At every second, we will remove the first 2 characters of word, and add the same characters to the end of word. After 4 seconds, word becomes equal to "abcbabcd" and reverts to its initial state. It can be shown that 4 seconds is the minimum time greater than zero required for word to revert to its initial state.

Constraints:

1 <= word.length <= 10

6

1 <= k <= word.length

word

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int minimumTimeToInitialState(string word, int k) {

}
};
```

Java:

```
class Solution {
public int minimumTimeToInitialState(String word, int k) {
}
}
```

Python3:

```
class Solution:
def minimumTimeToInitialState(self, word: str, k: int) -> int:
```

Python:

```
class Solution(object):
def minimumTimeToInitialState(self, word, k):
"""
:type word: str
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} word
* @param {number} k
* @return {number}
*/
var minimumTimeToInitialState = function(word, k) {

};
```

TypeScript:

```
function minimumTimeToInitialState(word: string, k: number): number {

};
```

C#:

```
public class Solution {
public int MinimumTimeToInitialState(string word, int k) {

}
}
```

C:

```
int minimumTimeToInitialState(char* word, int k) {

}
```

Go:

```
func minimumTimeToInitialState(word string, k int) int {

}
```

Kotlin:

```
class Solution {
fun minimumTimeToInitialState(word: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func minimumTimeToInitialState(_ word: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_time_to_initial_state(word: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} word
# @param {Integer} k
# @return {Integer}
def minimum_time_to_initial_state(word, k)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String $word
* @param Integer $k
* @return Integer
*/
function minimumTimeToInitialState($word, $k) {

}
}
```

Dart:

```
class Solution {
int minimumTimeToInitialState(String word, int k) {

}
}
```

Scala:

```
object Solution {
def minimumTimeToInitialState(word: String, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_time_to_initial_state(word :: String.t, k :: integer) ::
integer
def minimum_time_to_initial_state(word, k) do

end
end
```

Erlang:

```
-spec minimum_time_to_initial_state(Word :: unicode:unicode_binary(), K ::
integer()) -> integer().
minimum_time_to_initial_state(Word, K) ->
.
```

Racket:

```
(define/contract (minimum-time-to-initial-state word k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Time to Revert Word to Initial State II
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int minimumTimeToInitialState(string word, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Time to Revert Word to Initial State II
* Difficulty: Hard
* Tags: string, hash
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minimumTimeToInitialState(String word, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Time to Revert Word to Initial State II
Difficulty: Hard
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def minimumTimeToInitialState(self, word: str, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minimumTimeToInitialState(self, word, k):
"""
:type word: str
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Time to Revert Word to Initial State II
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} word
* @param {number} k
* @return {number}
*/
var minimumTimeToInitialState = function(word, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Time to Revert Word to Initial State II
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function minimumTimeToInitialState(word: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Time to Revert Word to Initial State II
* Difficulty: Hard
* Tags: string, hash
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MinimumTimeToInitialState(string word, int k) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Time to Revert Word to Initial State II
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minimumTimeToInitialState(char* word, int k) {

}
```

Go Solution:

```
// Problem: Minimum Time to Revert Word to Initial State II
// Difficulty: Hard
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func minimumTimeToInitialState(word string, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumTimeToInitialState(word: String, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumTimeToInitialState(_ word: String, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Time to Revert Word to Initial State II
// Difficulty: Hard
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn minimum_time_to_initial_state(word: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} word
# @param {Integer} k
# @return {Integer}
def minimum_time_to_initial_state(word, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @param Integer $k
* @return Integer
*/
function minimumTimeToInitialState($word, $k) {

}
}
```

Dart Solution:

```
class Solution {
int minimumTimeToInitialState(String word, int k) {

}
}
```

Scala Solution:

```
object Solution {
def minimumTimeToInitialState(word: String, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_time_to_initial_state(word :: String.t, k :: integer) ::
integer
def minimum_time_to_initial_state(word, k) do

end
end
```

Erlang Solution:

```
-spec minimum_time_to_initial_state(Word :: unicode:unicode_binary(), K ::
integer()) -> integer().
```

```
minimum_time_to_initial_state(Word, K) ->
.
```

Racket Solution:

```
(define/contract (minimum-time-to-initial-state word k)
(-> string? exact-integer? exact-integer?)
)
```