Problem 3135: Equalize Strings by Adding or Removing Characters at Ends

Problem Information

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

Problem Description

Given two strings

initial

and

target

, your task is to modify

initial

by performing a series of operations to make it equal to

target

.

In one operation, you can add or remove

one character

only at the

beginning

or the

end

of the string

initial

.

Return the

minimum

number of operations required to

transform

initial

into

target

.

Example 1:

Input:

initial = "abcde", target = "cdef"

Output:

3

Explanation:

Remove

'a'

and

'b'

from the beginning of

initial

, then add

'f'

to the end.

Example 2:

Input:

initial = "axxy", target = "yabx"

Output:

6

Explanation:

Operation

Resulting String

Add

'y'

to the beginning

"yaxxy"

Remove from end

"yaxx"

Remove from end

"yax"

Remove from end

"ya"

Add

'b'

to the end

"yab"

Add

'x'

to the end

"yabx"

Example 3:

Input:

initial = "xyz", target = "xyz"

Output:

0

Explanation:

No operations are needed as the strings are already equal.

Constraints:

1 <= initial.length, target.length <= 1000

initial

and

target

consist only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int minOperations(string initial, string target) {

}
};
```

Java:

```
class Solution {
public int minOperations(String initial, String target) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minOperations(initial: string, target: string): number {

};
```

C#:

```
public class Solution {
public int MinOperations(string initial, string target) {

}
}
```

C:

```
int minOperations(char* initial, char* target) {

}
```

Go:

```
func minOperations(initial string, target string) int {

}
```

Kotlin:

```
class Solution {
fun minOperations(initial: String, target: String): Int {

}
}
```

Swift:

```
class Solution {
func minOperations(_ initial: String, _ target: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_operations(initial: String, target: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} initial
# @param {String} target
# @return {Integer}
def min_operations(initial, target)

end
```

PHP:

```
class Solution {
```

```
/**
* @param String $initial
* @param String $target
* @return Integer
*/
function minOperations($initial, $target) {

}
}
```

Dart:

```
class Solution {
int minOperations(String initial, String target) {

}
}
```

Scala:

```
object Solution {
def minOperations(initial: String, target: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_operations(initial :: String.t, target :: String.t) :: integer
def min_operations(initial, target) do

end
end
```

Erlang:

```
-spec min_operations(Initial :: unicode:unicode_binary(), Target ::
unicode:unicode_binary()) -> integer().
min_operations(Initial, Target) ->
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Equalize Strings by Adding or Removing Characters at Ends
* Difficulty: Medium
* Tags: array, string, dp, hash, search
*
* 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 minOperations(string initial, string target) {

}
};
```

Java Solution:

```
/**
* Problem: Equalize Strings by Adding or Removing Characters at Ends
* Difficulty: Medium
* Tags: array, string, dp, hash, search
*
* 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 minOperations(String initial, String target) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Equalize Strings by Adding or Removing Characters at Ends
Difficulty: Medium
Tags: array, string, dp, hash, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Equalize Strings by Adding or Removing Characters at Ends
* Difficulty: Medium
* Tags: array, string, dp, hash, search
*
* 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} initial
* @param {string} target
* @return {number}
*/
var minOperations = function(initial, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Equalize Strings by Adding or Removing Characters at Ends
* Difficulty: Medium
* Tags: array, string, dp, hash, search
*
* 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 minOperations(initial: string, target: string): number {

};
```

C# Solution:

```
/*
* Problem: Equalize Strings by Adding or Removing Characters at Ends
* Difficulty: Medium
* Tags: array, string, dp, hash, search
*
* 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 MinOperations(string initial, string target) {
```

```
}
}
```

C Solution:

```
/*
* Problem: Equalize Strings by Adding or Removing Characters at Ends
* Difficulty: Medium
* Tags: array, string, dp, hash, search
*
* 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 minOperations(char* initial, char* target) {

}
```

Go Solution:

```
// Problem: Equalize Strings by Adding or Removing Characters at Ends
// Difficulty: Medium
// Tags: array, string, dp, hash, search
//
// 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 minOperations(initial string, target string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minOperations(initial: String, target: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minOperations(_ initial: String, _ target: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Equalize Strings by Adding or Removing Characters at Ends
// Difficulty: Medium
// Tags: array, string, dp, hash, search
//
// 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 min_operations(initial: String, target: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} initial
# @param {String} target
# @return {Integer}
def min_operations(initial, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $initial
* @param String $target
* @return Integer
*/
```

```
function minOperations($initial, $target) {

}
}
```

Dart Solution:

```
class Solution {
int minOperations(String initial, String target) {

}
}
```

Scala Solution:

```
object Solution {
def minOperations(initial: String, target: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_operations(initial :: String.t, target :: String.t) :: integer
def min_operations(initial, target) do

end
end
```

Erlang Solution:

```
-spec min_operations(Initial :: unicode:unicode_binary(), Target ::
unicode:unicode_binary()) -> integer().
min_operations(Initial, Target) ->
.
```

Racket Solution:

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

```
)
```