Problem 1055: Shortest Way to Form String

Problem Information

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

Problem Description

A

subsequence

of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e.,

"ace"

is a subsequence of

"

a

b

c

d

e

"

while

"aec"

is not).

Given two strings

source

and

target

, return

the minimum number of

subsequences

of

source

such that their concatenation equals

target

. If the task is impossible, return

-1

.

Example 1:

Input:

source = "abc", target = "abcbc"

Output:

2

Explanation:

The target "abcbc" can be formed by "abc" and "bc", which are subsequences of source "abc".

Example 2:

Input:

source = "abc", target = "acdbc"

Output:

-1

Explanation:

The target string cannot be constructed from the subsequences of source string due to the character "d" in target string.

Example 3:

Input:

source = "xyz", target = "xzyxz"

Output:

3

Explanation:

The target string can be constructed as follows "xz" + "y" + "xz".

Constraints:

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

source

and

target

consist of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int shortestWay(string source, string target) {

}
};
```

Java:

```
class Solution {
public int shortestWay(String source, String target) {

}
}
```

Python3:

```
class Solution:
def shortestWay(self, source: str, target: str) -> int:
```

Python:

```
class Solution(object):
def shortestWay(self, source, target):
"""
:type source: str
```

```
:type target: str
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function shortestWay(source: string, target: string): number {

};
```

C#:

```
public class Solution {
public int ShortestWay(string source, string target) {

}
}
```

C:

```
int shortestWay(char* source, char* target) {

}
```

Go:

```
func shortestWay(source string, target string) int {

}
```

Kotlin:

```
class Solution {
fun shortestWay(source: String, target: String): Int {

}
}
```

Swift:

```
class Solution {
func shortestWay(_ source: String, _ target: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn shortest_way(source: String, target: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} source
# @param {String} target
# @return {Integer}
def shortest_way(source, target)

end
```

PHP:

```
class Solution {

/**
* @param String $source
* @param String $target
* @return Integer
*/
function shortestWay($source, $target) {

}
```

```
}
```

Dart:

```
class Solution {
int shortestWay(String source, String target) {

}
}
```

Scala:

```
object Solution {
def shortestWay(source: String, target: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec shortest_way(Source :: unicode:unicode_binary(), Target ::
unicode:unicode_binary()) -> integer().
shortest_way(Source, Target) ->
.
```

Racket:

```
(define/contract (shortest-way source target)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Shortest Way to Form String
* Difficulty: Medium
* Tags: array, string, greedy, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int shortestWay(string source, string target) {

}
};
```

Java Solution:

```
/**
* Problem: Shortest Way to Form String
* Difficulty: Medium
* Tags: array, string, greedy, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int shortestWay(String source, String target) {

}
}
```

Python3 Solution:

```
"""
Problem: Shortest Way to Form String
```

```
Difficulty: Medium
Tags: array, string, greedy, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def shortestWay(self, source: str, target: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Shortest Way to Form String
* Difficulty: Medium
* Tags: array, string, greedy, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

```
};
```

TypeScript Solution:

```
/**
* Problem: Shortest Way to Form String
* Difficulty: Medium
* Tags: array, string, greedy, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function shortestWay(source: string, target: string): number {

};
```

C# Solution:

```
/*
* Problem: Shortest Way to Form String
* Difficulty: Medium
* Tags: array, string, greedy, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int ShortestWay(string source, string target) {

}
}
```

C Solution:

```
/*
* Problem: Shortest Way to Form String
```

```
* Difficulty: Medium
* Tags: array, string, greedy, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int shortestWay(char* source, char* target) {

}
```

Go Solution:

```
// Problem: Shortest Way to Form String
// Difficulty: Medium
// Tags: array, string, greedy, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func shortestWay(source string, target string) int {

}
```

Kotlin Solution:

```
class Solution {
fun shortestWay(source: String, target: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func shortestWay(_ source: String, _ target: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Shortest Way to Form String
// Difficulty: Medium
// Tags: array, string, greedy, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn shortest_way(source: String, target: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} source
# @param {String} target
# @return {Integer}
def shortest_way(source, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $source
* @param String $target
* @return Integer
*/
function shortestWay($source, $target) {

}
}
```

Dart Solution:

```
class Solution {
int shortestWay(String source, String target) {

}
}
```

Scala Solution:

```
object Solution {
def shortestWay(source: String, target: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec shortest_way(Source :: unicode:unicode_binary(), Target ::
unicode:unicode_binary()) -> integer().
shortest_way(Source, Target) ->
.
```

Racket Solution:

```
(define/contract (shortest-way source target)
(-> string? string? exact-integer?)
)
```