Problem 3316: Find Maximum Removals From Source String

Problem Information

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

Problem Description

You are given a string

source

of size

n

, a string

pattern

that is a

subsequence

of

source

, and a

sorted

integer array

targetIndices

that contains

distinct

numbers in the range

[0, n - 1]

.

We define an

operation

as removing a character at an index

idx

from

source

such that:

idx

is an element of

targetIndices

.

pattern

remains a

subsequence

of

source

after removing the character.

Performing an operation

does not

change the indices of the other characters in

source

. For example, if you remove

'c'

from

"acb"

, the character at index 2 would still be

'b'

.

Return the

maximum

number of

operations

that can be performed.

Example 1:

Input:

source = "abbaa", pattern = "aba",

targetIndices

= [0,1,2]

Output:

1

Explanation:

We can't remove

source[0]

but we can do either of these two operations:

Remove

source[1]

, so that

source

becomes

"a_baa"

.

Remove

source[2]

, so that

source

becomes

"ab_aa"

.

Example 2:

Input:

source = "bcda", pattern = "d",

targetIndices

= [0,3]

Output:

2

Explanation:

We can remove

source[0]

and

source[3]

in two operations.

Example 3:

Input:

source = "dda", pattern = "dda",

targetIndices

= [0,1,2]

Output:

0

Explanation:

We can't remove any character from

source

.

Example 4:

Input:

source =

"yeyeykyded"

, pattern =

"yeyyd"

,

targetIndices

=

[0,2,3,4]

Output:

2

Explanation:

We can remove

source[2]

and

source[3]

in two operations.

Constraints:

1 <= n == source.length <= 3 * 10

3

1 <= pattern.length <= n

1 <= targetIndices.length <= n

targetIndices

is sorted in ascending order.

The input is generated such that

targetIndices

contains distinct elements in the range

[0, n - 1]

.

source

and

pattern

consist only of lowercase English letters.

The input is generated such that

pattern

appears as a subsequence in

source

.

Code Snippets

C++:

```
class Solution {
public:
int maxRemovals(string source, string pattern, vector<int>& targetIndices) {

}
};
```

Java:

```
class Solution {
public int maxRemovals(String source, String pattern, int[] targetIndices) {

}
}
```

Python3:

```
class Solution:
def maxRemovals(self, source: str, pattern: str, targetIndices: List[int]) ->
int:
```

Python:

```
class Solution(object):
def maxRemovals(self, source, pattern, targetIndices):
"""
:type source: str
:type pattern: str
:type targetIndices: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} source
* @param {string} pattern
* @param {number[]} targetIndices
* @return {number}
*/
var maxRemovals = function(source, pattern, targetIndices) {

};
```

TypeScript:

```
function maxRemovals(source: string, pattern: string, targetIndices:
number[]): number {

};
```

C#:

```
public class Solution {
public int MaxRemovals(string source, string pattern, int[] targetIndices) {

}
}
```

C:

```
int maxRemovals(char* source, char* pattern, int* targetIndices, int
targetIndicesSize) {

}
```

Go:

```
func maxRemovals(source string, pattern string, targetIndices []int) int {

}
```

Kotlin:

```
class Solution {
fun maxRemovals(source: String, pattern: String, targetIndices: IntArray):
Int {

}
}
```

Swift:

```
class Solution {
func maxRemovals(_ source: String, _ pattern: String, _ targetIndices: [Int])
-> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_removals(source: String, pattern: String, target_indices:
Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {String} source
# @param {String} pattern
# @param {Integer[]} target_indices
# @return {Integer}
def max_removals(source, pattern, target_indices)

end
```

PHP:

```
class Solution {

/**
* @param String $source
* @param String $pattern
* @param Integer[] $targetIndices
* @return Integer
*/
function maxRemovals($source, $pattern, $targetIndices) {

}
}
```

Dart:

```
class Solution {
int maxRemovals(String source, String pattern, List<int> targetIndices) {

}
}
```

Scala:

```
object Solution {
def maxRemovals(source: String, pattern: String, targetIndices: Array[Int]):
Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_removals(source :: String.t, pattern :: String.t, target_indices ::
[integer]) :: integer
def max_removals(source, pattern, target_indices) do

end
end
```

Erlang:

```
-spec max_removals(Source :: unicode:unicode_binary(), Pattern ::
unicode:unicode_binary(), TargetIndices :: [integer()]) -> integer().
max_removals(Source, Pattern, TargetIndices) ->
.
```

Racket:

```
(define/contract (max-removals source pattern targetIndices)
(-> string? string? (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Maximum Removals From Source String
* Difficulty: Medium
* Tags: array, string, dp, hash, sort
*
* 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 maxRemovals(string source, string pattern, vector<int>& targetIndices) {

}
};
```

Java Solution:

```
/**
* Problem: Find Maximum Removals From Source String
* Difficulty: Medium
* Tags: array, string, dp, hash, sort
*
* 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 maxRemovals(String source, String pattern, int[] targetIndices) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Maximum Removals From Source String
Difficulty: Medium
Tags: array, string, dp, hash, sort

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 maxRemovals(self, source: str, pattern: str, targetIndices: List[int]) ->
int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxRemovals(self, source, pattern, targetIndices):
"""
:type source: str
```

```
:type pattern: str
:type targetIndices: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find Maximum Removals From Source String
* Difficulty: Medium
* Tags: array, string, dp, hash, sort
*
* 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} source
* @param {string} pattern
* @param {number[]} targetIndices
* @return {number}
*/
var maxRemovals = function(source, pattern, targetIndices) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Maximum Removals From Source String
* Difficulty: Medium
* Tags: array, string, dp, hash, sort
*
* 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 maxRemovals(source: string, pattern: string, targetIndices:
number[]): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Find Maximum Removals From Source String
* Difficulty: Medium
* Tags: array, string, dp, hash, sort
*
* 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 MaxRemovals(string source, string pattern, int[] targetIndices) {

}
}
```

C Solution:

```
/*
* Problem: Find Maximum Removals From Source String
* Difficulty: Medium
* Tags: array, string, dp, hash, sort
*
* 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 maxRemovals(char* source, char* pattern, int* targetIndices, int
targetIndicesSize) {

}
```

Go Solution:

```
// Problem: Find Maximum Removals From Source String
// Difficulty: Medium
// Tags: array, string, dp, hash, sort
//
// 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 maxRemovals(source string, pattern string, targetIndices []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxRemovals(source: String, pattern: String, targetIndices: IntArray):
Int {

}
}
```

Swift Solution:

```
class Solution {
func maxRemovals(_ source: String, _ pattern: String, _ targetIndices: [Int])
-> Int {

}
}
```

Rust Solution:

```
// Problem: Find Maximum Removals From Source String
// Difficulty: Medium
// Tags: array, string, dp, hash, sort
//
// 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 max_removals(source: String, pattern: String, target_indices:
```

```
Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} source
# @param {String} pattern
# @param {Integer[]} target_indices
# @return {Integer}
def max_removals(source, pattern, target_indices)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $source
* @param String $pattern
* @param Integer[] $targetIndices
* @return Integer
*/
function maxRemovals($source, $pattern, $targetIndices) {

}
}
```

Dart Solution:

```
class Solution {
int maxRemovals(String source, String pattern, List<int> targetIndices) {

}
}
```

Scala Solution:

```
object Solution {
def maxRemovals(source: String, pattern: String, targetIndices: Array[Int]):
```

```
Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_removals(source :: String.t, pattern :: String.t, target_indices ::
[integer]) :: integer
def max_removals(source, pattern, target_indices) do

end
end
```

Erlang Solution:

```
-spec max_removals(Source :: unicode:unicode_binary(), Pattern ::
unicode:unicode_binary(), TargetIndices :: [integer()]) -> integer().
max_removals(Source, Pattern, TargetIndices) ->
.
```

Racket Solution:

```
(define/contract (max-removals source pattern targetIndices)
(-> string? string? (listof exact-integer?) exact-integer?)
)
```