Problem 1717: Maximum Score From Removing Substrings

Problem Information

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

Problem Description

You are given a string

s

and two integers

x

and

y

. You can perform two types of operations any number of times.

Remove substring

"ab"

and gain

x

points.

For example, when removing

"ab"

from

"c

ab

xbae"

it becomes

"cxbae"

.

Remove substring

"ba"

and gain

y

points.

For example, when removing

"ba"

from

"cabx

ba

e"

it becomes

"cabxe"

.

Return

the maximum points you can gain after applying the above operations on

s

.

Example 1:

Input:

s = "cdbcbbaaabab", x = 4, y = 5

Output:

19

Explanation:

- Remove the "ba" underlined in "cdbcbbaaa

ba

b". Now, s = "cdbcbbaaab" and 5 points are added to the score. - Remove the "ab" underlined in "cdbcbbaa

ab

". Now, s = "cdbcbbaa" and 4 points are added to the score. - Remove the "ba" underlined in "cdbcb

ba

a". Now, s = "cdbcba" and 5 points are added to the score. - Remove the "ba" underlined in "cdbc

ba

". Now, s = "cdbc" and 5 points are added to the score. Total score = 5 + 4 + 5 + 5 = 19.

Example 2:

Input:

s = "aabbaaxybbaabb", x = 5, y = 4

Output:

20

Constraints:

1 <= s.length <= 10

5

1 <= x, y <= 10

4

s

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int maximumGain(string s, int x, int y) {
```

```
}
};
```

Java:

```
class Solution {
public int maximumGain(String s, int x, int y) {

}
}
```

Python3:

```
class Solution:
def maximumGain(self, s: str, x: int, y: int) -> int:
```

Python:

```
class Solution(object):
def maximumGain(self, s, x, y):
"""
:type s: str
:type x: int
:type y: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number} x
* @param {number} y
* @return {number}
*/
var maximumGain = function(s, x, y) {

};
```

TypeScript:

```
function maximumGain(s: string, x: number, y: number): number {

};
```

C#:

```
public class Solution {
public int MaximumGain(string s, int x, int y) {

}
}
```

C:

```
int maximumGain(char* s, int x, int y) {

}
```

Go:

```
func maximumGain(s string, x int, y int) int {

}
```

Kotlin:

```
class Solution {
fun maximumGain(s: String, x: Int, y: Int): Int {

}
}
```

Swift:

```
class Solution {
func maximumGain(_ s: String, _ x: Int, _ y: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_gain(s: String, x: i32, y: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} x
# @param {Integer} y
# @return {Integer}
def maximum_gain(s, x, y)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $x
* @param Integer $y
* @return Integer
*/
function maximumGain($s, $x, $y) {

}
}
```

Dart:

```
class Solution {
int maximumGain(String s, int x, int y) {

}
}
```

Scala:

```
object Solution {
def maximumGain(s: String, x: Int, y: Int): Int = {
```

```
}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_gain(s :: String.t, x :: integer, y :: integer) :: integer
def maximum_gain(s, x, y) do

end
end
```

Erlang:

```
-spec maximum_gain(S :: unicode:unicode_binary(), X :: integer(), Y ::
integer()) -> integer().
maximum_gain(S, X, Y) ->
.
```

Racket:

```
(define/contract (maximum-gain s x y)
(-> string? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Score From Removing Substrings
* Difficulty: Medium
* Tags: string, tree, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
class Solution {
public:
int maximumGain(string s, int x, int y) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Score From Removing Substrings
* Difficulty: Medium
* Tags: string, tree, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int maximumGain(String s, int x, int y) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Score From Removing Substrings
Difficulty: Medium
Tags: string, tree, greedy, stack

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def maximumGain(self, s: str, x: int, y: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maximumGain(self, s, x, y):
"""
:type s: str
:type x: int
:type y: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Score From Removing Substrings
* Difficulty: Medium
* Tags: string, tree, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {string} s
* @param {number} x
* @param {number} y
* @return {number}
*/
var maximumGain = function(s, x, y) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Score From Removing Substrings
* Difficulty: Medium
* Tags: string, tree, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(h) for recursion stack where h is height
*/

function maximumGain(s: string, x: number, y: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Score From Removing Substrings
* Difficulty: Medium
* Tags: string, tree, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int MaximumGain(string s, int x, int y) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Score From Removing Substrings
* Difficulty: Medium
* Tags: string, tree, greedy, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int maximumGain(char* s, int x, int y) {

}
```

Go Solution:

```
// Problem: Maximum Score From Removing Substrings
// Difficulty: Medium
// Tags: string, tree, greedy, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func maximumGain(s string, x int, y int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumGain(s: String, x: Int, y: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximumGain(_ s: String, _ x: Int, _ y: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Score From Removing Substrings
// Difficulty: Medium
// Tags: string, tree, greedy, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn maximum_gain(s: String, x: i32, y: i32) -> i32 {
```

```
}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} x
# @param {Integer} y
# @return {Integer}
def maximum_gain(s, x, y)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $x
* @param Integer $y
* @return Integer
*/
function maximumGain($s, $x, $y) {

}
}
```

Dart Solution:

```
class Solution {
int maximumGain(String s, int x, int y) {

}
}
```

Scala Solution:

```
object Solution {
def maximumGain(s: String, x: Int, y: Int): Int = {
```

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_gain(s :: String.t, x :: integer, y :: integer) :: integer
def maximum_gain(s, x, y) do

end
end
```

Erlang Solution:

```
-spec maximum_gain(S :: unicode:unicode_binary(), X :: integer(), Y ::
integer()) -> integer().
maximum_gain(S, X, Y) ->
.
```

Racket Solution:

```
(define/contract (maximum-gain s x y)
(-> string? exact-integer? exact-integer? exact-integer?)
)
```