Problem 2287: Rearrange Characters to Make Target String

Problem Information

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

Problem Description

You are given two

0-indexed

strings

s

and

target

. You can take some letters from

s

and rearrange them to form new strings.

Return

the

maximum

number of copies of

target

that can be formed by taking letters from

s

and rearranging them.

Example 1:

Input:

s = "ilovecodingonleetcode", target = "code"

Output:

2

Explanation:

For the first copy of "code", take the letters at indices 4, 5, 6, and 7. For the second copy of "code", take the letters at indices 17, 18, 19, and 20. The strings that are formed are "ecod" and "code" which can both be rearranged into "code". We can make at most two copies of "code", so we return 2.

Example 2:

Input:

s = "abcba", target = "abc"

Output:

1

Explanation:

We can make one copy of "abc" by taking the letters at indices 0, 1, and 2. We can make at most one copy of "abc", so we return 1. Note that while there is an extra 'a' and 'b' at indices 3

and 4, we cannot reuse the letter 'c' at index 2, so we cannot make a second copy of "abc".

Example 3:

Input:

s = "abbaccaddaeea", target = "aaaaa"

Output:

1

Explanation:

We can make one copy of "aaaaa" by taking the letters at indices 0, 3, 6, 9, and 12. We can make at most one copy of "aaaaa", so we return 1.

Constraints:

1 <= s.length <= 100

1 <= target.length <= 10

s

and

target

consist of lowercase English letters.

Note:

This question is the same as

1189: Maximum Number of Balloons.

Code Snippets

C++:

```
class Solution {
public:
int rearrangeCharacters(string s, string target) {

}
};
```

Java:

```
class Solution {
public int rearrangeCharacters(String s, String target) {

}
}
```

Python3:

```
class Solution:
def rearrangeCharacters(self, s: str, target: str) -> int:
```

Python:

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

JavaScript:

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

```
};
```

TypeScript:

```
function rearrangeCharacters(s: string, target: string): number {

};
```

C#:

```
public class Solution {
public int RearrangeCharacters(string s, string target) {

}
}
```

C:

```
int rearrangeCharacters(char* s, char* target) {

}
```

Go:

```
func rearrangeCharacters(s string, target string) int {

}
```

Kotlin:

```
class Solution {
fun rearrangeCharacters(s: String, target: String): Int {

}
}
```

Swift:

```
class Solution {
func rearrangeCharacters(_ s: String, _ target: String) -> Int {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn rearrange_characters(s: String, target: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {String} target
# @return {Integer}
def rearrange_characters(s, target)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $target
* @return Integer
*/
function rearrangeCharacters($s, $target) {

}
}
```

Dart:

```
class Solution {
int rearrangeCharacters(String s, String target) {

}
}
```

Scala:

```
object Solution {
def rearrangeCharacters(s: String, target: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec rearrange_characters(S :: unicode:unicode_binary(), Target ::
unicode:unicode_binary()) -> integer().
rearrange_characters(S, Target) ->
.
```

Racket:

```
(define/contract (rearrange-characters s target)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Rearrange Characters to Make Target String
* Difficulty: Easy
* 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 rearrangeCharacters(string s, string target) {

}
};
```

Java Solution:

```
/**
* Problem: Rearrange Characters to Make Target String
* Difficulty: Easy
* 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 rearrangeCharacters(String s, String target) {

}
}
```

Python3 Solution:

```
"""
Problem: Rearrange Characters to Make Target String
Difficulty: Easy
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 rearrangeCharacters(self, s: str, target: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Rearrange Characters to Make Target String
* Difficulty: Easy
* 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} s
* @param {string} target
* @return {number}
*/
var rearrangeCharacters = function(s, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Rearrange Characters to Make Target String
* Difficulty: Easy
* 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 rearrangeCharacters(s: string, target: string): number {

};
```

C# Solution:

```
/*
* Problem: Rearrange Characters to Make Target String
* Difficulty: Easy
* 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 RearrangeCharacters(string s, string target) {

}
}
```

C Solution:

```
/*
* Problem: Rearrange Characters to Make Target String
* Difficulty: Easy
* 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 rearrangeCharacters(char* s, char* target) {

}
```

Go Solution:

```
// Problem: Rearrange Characters to Make Target String
// Difficulty: Easy
// 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 rearrangeCharacters(s string, target string) int {

}
```

Kotlin Solution:

```
class Solution {
fun rearrangeCharacters(s: String, target: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func rearrangeCharacters(_ s: String, _ target: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Rearrange Characters to Make Target String
// Difficulty: Easy
// 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 rearrange_characters(s: String, target: String) -> i32 {
```

```
}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} target
# @return {Integer}
def rearrange_characters(s, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $target
* @return Integer
*/
function rearrangeCharacters($s, $target) {

}
}
```

Dart Solution:

```
class Solution {
int rearrangeCharacters(String s, String target) {

}
}
```

Scala Solution:

```
object Solution {
def rearrangeCharacters(s: String, target: String): Int = {

}
```

```
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec rearrange_characters(S :: unicode:unicode_binary(), Target ::
unicode:unicode_binary()) -> integer().
rearrange_characters(S, Target) ->
.
```

Racket Solution:

```
(define/contract (rearrange-characters s target)
(-> string? string? exact-integer?)
)
```