Problem 2516: Take K of Each Character From Left and Right

Problem Information

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

Problem Description

You are given a string

s

consisting of the characters

'a'

,

'b'

, and

'c'

and a non-negative integer

k

. Each minute, you may take either the

leftmost

character of

s

, or the

rightmost

character of

s

.

Return

the

minimum

number of minutes needed for you to take

at least

k

of each character, or return

-1

if it is not possible to take

k

of each character.

Example 1:

Input:

s = "aabaaaacaabc", k = 2

Output:

8

Explanation:

Take three characters from the left of s. You now have two 'a' characters, and one 'b' character. Take five characters from the right of s. You now have four 'a' characters, two 'b' characters, and two 'c' characters. A total of 3 + 5 = 8 minutes is needed. It can be proven that 8 is the minimum number of minutes needed.

Example 2:

Input:

s = "a", k = 1

Output:

-1

Explanation:

It is not possible to take one 'b' or 'c' so return -1.

Constraints:

1 <= s.length <= 10

5

s

consists of only the letters

'a'

,

'b'

, and

'c'

.

0 <= k <= s.length

Code Snippets

C++:

```
class Solution {
public:
int takeCharacters(string s, int k) {

}
};
```

Java:

```
class Solution {
public int takeCharacters(String s, int k) {

}
}
```

Python3:

```
class Solution:
def takeCharacters(self, s: str, k: int) -> int:
```

Python:

```
class Solution(object):
def takeCharacters(self, s, k):
```

```
"""
:type s: str
:type k: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function takeCharacters(s: string, k: number): number {

};
```

C#:

```
public class Solution {
public int TakeCharacters(string s, int k) {

}
}
```

C:

```
int takeCharacters(char* s, int k) {

}
```

Go:

```
func takeCharacters(s string, k int) int {

}
```

Kotlin:

```
class Solution {
fun takeCharacters(s: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func takeCharacters(_ s: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn take_characters(s: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def take_characters(s, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function takeCharacters($s, $k) {
```

```
}
}
```

Dart:

```
class Solution {
int takeCharacters(String s, int k) {

}
}
```

Scala:

```
object Solution {
def takeCharacters(s: String, k: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec take_characters(S :: unicode:unicode_binary(), K :: integer()) ->
integer().
take_characters(S, K) ->
.
```

Racket:

```
(define/contract (take-characters s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Take K of Each Character From Left and Right
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int takeCharacters(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Take K of Each Character From Left and Right
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int takeCharacters(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Take K of Each Character From Left and Right
Difficulty: Medium
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Take K of Each Character From Left and Right
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @param {number} k
* @return {number}
*/
```

```
var takeCharacters = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Take K of Each Character From Left and Right
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function takeCharacters(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Take K of Each Character From Left and Right
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int TakeCharacters(string s, int k) {

}
}
```

C Solution:

```c
/*
* Problem: Take K of Each Character From Left and Right
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int takeCharacters(char* s, int k) {

}
```

Go Solution:

```go
// Problem: Take K of Each Character From Left and Right
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func takeCharacters(s string, k int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun takeCharacters(s: String, k: Int): Int {

}
}
```

Swift Solution:

```swift
class Solution {
func takeCharacters(_ s: String, _ k: Int) -> Int {

}
```

```
}
```

Rust Solution:

```
// Problem: Take K of Each Character From Left and Right
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn take_characters(s: String, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def take_characters(s, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function takeCharacters($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
int takeCharacters(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def takeCharacters(s: String, k: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec take_characters(S :: unicode:unicode_binary(), K :: integer()) ->
integer().
take_characters(S, K) ->
.
```

Racket Solution:

```
(define/contract (take-characters s k)
(-> string? exact-integer? exact-integer?)
)
```