Problem 1684: Count the Number of Consistent Strings

Problem Information

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

Problem Description

You are given a string

allowed

consisting of

distinct

characters and an array of strings

words

. A string is

consistent

if all characters in the string appear in the string

allowed

.

Return

the number of

consistent

strings in the array

words

.

Example 1:

Input:

allowed = "ab", words = ["ad","bd","aaab","baa","badab"]

Output:

2

Explanation:

Strings "aaab" and "baa" are consistent since they only contain characters 'a' and 'b'.

Example 2:

Input:

allowed = "abc", words = ["a","b","c","ab","ac","bc","abc"]

Output:

7

Explanation:

All strings are consistent.

Example 3:

Input:

allowed = "cad", words = ["cc","acd","b","ba","bac","bad","ac","d"]

Output:

4

Explanation:

Strings "cc", "acd", "ac", and "d" are consistent.

Constraints:

1 <= words.length <= 10

4

1 <= allowed.length <=

26

1 <= words[i].length <= 10

The characters in

allowed

are

distinct

.

words[i]

and

allowed

contain only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int countConsistentStrings(string allowed, vector<string>& words) {

}
};
```

Java:

```
class Solution {
public int countConsistentStrings(String allowed, String[] words) {

}
}
```

Python3:

```
class Solution:
def countConsistentStrings(self, allowed: str, words: List[str]) -> int:
```

Python:

```
class Solution(object):
def countConsistentStrings(self, allowed, words):
"""
:type allowed: str
:type words: List[str]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} allowed
* @param {string[]} words
```

```
 * @return {number}
 */
var countConsistentStrings = function(allowed, words) {

};
```

TypeScript:

```
function countConsistentStrings(allowed: string, words: string[]): number {

};
```

C#:

```
public class Solution {
public int CountConsistentStrings(string allowed, string[] words) {

}
}
```

C:

```


int countConsistentStrings(char * allowed, char ** words, int wordsSize){

}
```

Go:

```
func countConsistentStrings(allowed string, words []string) int {

}
```

Kotlin:

```
class Solution {
fun countConsistentStrings(allowed: String, words: Array<String>): Int {

}
}
```

Swift:

```
class Solution {
func countConsistentStrings(_ allowed: String, _ words: [String]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_consistent_strings(allowed: String, words: Vec<String>) -> i32 {

}
}
```

Ruby:

```
# @param {String} allowed
# @param {String[]} words
# @return {Integer}
def count_consistent_strings(allowed, words)

end
```

PHP:

```
class Solution {

/**
* @param String $allowed
* @param String[] $words
* @return Integer
*/
function countConsistentStrings($allowed, $words) {

}
}
```

Scala:

```
object Solution {
def countConsistentStrings(allowed: String, words: Array[String]): Int = {
```

```
}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Count the Number of Consistent Strings
* Difficulty: Easy
* 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 countConsistentStrings(string allowed, vector<string>& words) {

}
};
```

Java Solution:

```
/**
* Problem: Count the Number of Consistent Strings
* Difficulty: Easy
* 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 countConsistentStrings(String allowed, String[] words) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Count the Number of Consistent Strings
Difficulty: Easy
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 countConsistentStrings(self, allowed: str, words: List[str]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countConsistentStrings(self, allowed, words):
"""
:type allowed: str
:type words: List[str]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count the Number of Consistent Strings
* Difficulty: Easy
* 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} allowed
* @param {string[]} words
* @return {number}
*/
var countConsistentStrings = function(allowed, words) {

};
```

TypeScript Solution:

```
/**
* Problem: Count the Number of Consistent Strings
* Difficulty: Easy
* 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 countConsistentStrings(allowed: string, words: string[]): number {

};
```

C# Solution:

```
/*
* Problem: Count the Number of Consistent Strings
* Difficulty: Easy
* 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 CountConsistentStrings(string allowed, string[] words) {
```

```
}
}
```

C Solution:

```
/*
* Problem: Count the Number of Consistent Strings
* Difficulty: Easy
* 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 countConsistentStrings(char * allowed, char ** words, int wordsSize){

}
```

Go Solution:

```
// Problem: Count the Number of Consistent Strings
// Difficulty: Easy
// 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 countConsistentStrings(allowed string, words []string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countConsistentStrings(allowed: String, words: Array<String>): Int {

}
```

```
}
```

Swift Solution:

```
class Solution {
func countConsistentStrings(_ allowed: String, _ words: [String]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count the Number of Consistent Strings
// Difficulty: Easy
// 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 count_consistent_strings(allowed: String, words: Vec<String>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} allowed
# @param {String[]} words
# @return {Integer}
def count_consistent_strings(allowed, words)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $allowed
```

```
 * @param String[] $words
 * @return Integer
 */
function countConsistentStrings($allowed, $words) {

}
}
```

Scala Solution:

```
object Solution {
def countConsistentStrings(allowed: String, words: Array[String]): Int = {

}
}
```