Problem 771: Jewels and Stones

Problem Information

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

Problem Description

You're given strings

jewels

representing the types of stones that are jewels, and

stones

representing the stones you have. Each character in

stones

is a type of stone you have. You want to know how many of the stones you have are also jewels.

Letters are case sensitive, so

"a"

is considered a different type of stone from

"A"

.

Example 1:

Input:

jewels = "aA", stones = "aAAbbbb"

Output:

3

Example 2:

Input:

jewels = "z", stones = "ZZ"

Output:

0

Constraints:

1 <= jewels.length, stones.length <= 50

jewels

and

stones

consist of only English letters.

All the characters of

jewels

are

unique

.

Code Snippets

C++:

```
class Solution {
public:
int numJewelsInStones(string jewels, string stones) {

}
};
```

Java:

```
class Solution {
public int numJewelsInStones(String jewels, String stones) {

}
}
```

Python3:

```
class Solution:
def numJewelsInStones(self, jewels: str, stones: str) -> int:
```

Python:

```
class Solution(object):
def numJewelsInStones(self, jewels, stones):
"""
:type jewels: str
:type stones: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} jewels
* @param {string} stones
```

```
* @return {number}
*/
var numJewelsInStones = function(jewels, stones) {

};
```

TypeScript:

```
function numJewelsInStones(jewels: string, stones: string): number {

};
```

C#:

```
public class Solution {
public int NumJewelsInStones(string jewels, string stones) {

}
}
```

C:

```
int numJewelsInStones(char* jewels, char* stones) {

}
```

Go:

```
func numJewelsInStones(jewels string, stones string) int {

}
```

Kotlin:

```
class Solution {
fun numJewelsInStones(jewels: String, stones: String): Int {

}
}
```

Swift:

```
class Solution {
func numJewelsInStones(_ jewels: String, _ stones: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_jewels_in_stones(jewels: String, stones: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} jewels
# @param {String} stones
# @return {Integer}
def num_jewels_in_stones(jewels, stones)

end
```

PHP:

```
class Solution {

/**
* @param String $jewels
* @param String $stones
* @return Integer
*/
function numJewelsInStones($jewels, $stones) {

}
}
```

Dart:

```
class Solution {
int numJewelsInStones(String jewels, String stones) {

}
```

```
}
```

Scala:

```
object Solution {
def numJewelsInStones(jewels: String, stones: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_jewels_in_stones(jewels :: String.t, stones :: String.t) :: integer
def num_jewels_in_stones(jewels, stones) do

end
end
```

Erlang:

```
-spec num_jewels_in_stones(Jewels :: unicode:unicode_binary(), Stones ::
unicode:unicode_binary()) -> integer().
num_jewels_in_stones(Jewels, Stones) ->
.
```

Racket:

```
(define/contract (num-jewels-in-stones jewels stones)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Jewels and Stones
* 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 numJewelsInStones(string jewels, string stones) {

}
};
```

Java Solution:

```
/**
* Problem: Jewels and Stones
* 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 numJewelsInStones(String jewels, String stones) {

}
}
```

Python3 Solution:

```
"""
Problem: Jewels and Stones
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 numJewelsInStones(self, jewels: str, stones: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numJewelsInStones(self, jewels, stones):
"""
:type jewels: str
:type stones: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Jewels and Stones
* 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} jewels
* @param {string} stones
* @return {number}
*/
var numJewelsInStones = function(jewels, stones) {

};
```

TypeScript Solution:

```
/**
* Problem: Jewels and Stones
* 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 numJewelsInStones(jewels: string, stones: string): number {

};
```

C# Solution:

```
/*
* Problem: Jewels and Stones
* 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 NumJewelsInStones(string jewels, string stones) {

}
}
```

C Solution:

```
/*
* Problem: Jewels and Stones
* 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 numJewelsInStones(char* jewels, char* stones) {

}
```

Go Solution:

```
// Problem: Jewels and Stones
// 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 numJewelsInStones(jewels string, stones string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numJewelsInStones(jewels: String, stones: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func numJewelsInStones(_ jewels: String, _ stones: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Jewels and Stones
// 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 num_jewels_in_stones(jewels: String, stones: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} jewels
# @param {String} stones
# @return {Integer}
def num_jewels_in_stones(jewels, stones)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $jewels
* @param String $stones
* @return Integer
*/
function numJewelsInStones($jewels, $stones) {

}
}
```

Dart Solution:

```
class Solution {
int numJewelsInStones(String jewels, String stones) {

}
}
```

Scala Solution:

```
object Solution {
def numJewelsInStones(jewels: String, stones: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_jewels_in_stones(jewels :: String.t, stones :: String.t) :: integer
def num_jewels_in_stones(jewels, stones) do

end
end
```

Erlang Solution:

```
-spec num_jewels_in_stones(Jewels :: unicode:unicode_binary(), Stones ::
unicode:unicode_binary()) -> integer().
num_jewels_in_stones(Jewels, Stones) ->
.
```

Racket Solution:

```
(define/contract (num-jewels-in-stones jewels stones)
(-> string? string? exact-integer?)
)
```