Problem 299: Bulls and Cows

Problem Information

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

Problem Description

You are playing the

Bulls and Cows

game with your friend.

You write down a secret number and ask your friend to guess what the number is. When your friend makes a guess, you provide a hint with the following info:

The number of "bulls", which are digits in the guess that are in the correct position.

The number of "cows", which are digits in the guess that are in your secret number but are located in the wrong position. Specifically, the non-bull digits in the guess that could be rearranged such that they become bulls.

Given the secret number

secret

and your friend's guess

guess

, return

the hint for your friend's guess

.

The hint should be formatted as

"xAyB"

, where

x

is the number of bulls and

y

is the number of cows. Note that both

secret

and

guess

may contain duplicate digits.

Example 1:

Input:

secret = "1807", guess = "7810"

Output:

"1A3B"

Explanation:

Bulls are connected with a '|' and cows are underlined: "1807" | "

7

8

10

"

Example 2:

Input:

secret = "1123", guess = "0111"

Output:

"1A1B"

Explanation:

Bulls are connected with a '|' and cows are underlined: "1123" "1123" | or | "01

1

1" "011

1

" Note that only one of the two unmatched 1s is counted as a cow since the non-bull digits can only be rearranged to allow one 1 to be a bull.

Constraints:

1 <= secret.length, guess.length <= 1000

secret.length == guess.length

secret

and

guess

consist of digits only.

Code Snippets

C++:

```
class Solution {
public:
string getHint(string secret, string guess) {

}
};
```

Java:

```
class Solution {
public String getHint(String secret, String guess) {

}
}
```

Python3:

```
class Solution:
def getHint(self, secret: str, guess: str) -> str:
```

Python:

```
class Solution(object):
def getHint(self, secret, guess):
"""
:type secret: str
:type guess: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} secret
* @param {string} guess
* @return {string}
*/
var getHint = function(secret, guess) {

};
```

TypeScript:

```
function getHint(secret: string, guess: string): string {

};
```

C#:

```
public class Solution {
public string GetHint(string secret, string guess) {

}
}
```

C:

```
char* getHint(char* secret, char* guess) {

}
```

Go:

```
func getHint(secret string, guess string) string {

}
```

Kotlin:

```
class Solution {
fun getHint(secret: String, guess: String): String {

}
}
```

Swift:

```
class Solution {
func getHint(_ secret: String, _ guess: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn get_hint(secret: String, guess: String) -> String {

}
}
```

Ruby:

```
# @param {String} secret
# @param {String} guess
# @return {String}
def get_hint(secret, guess)

end
```

PHP:

```
class Solution {

/**
* @param String $secret
* @param String $guess
* @return String
*/
function getHint($secret, $guess) {

}
}
```

Dart:

```
class Solution {
String getHint(String secret, String guess) {



}
}
```

Scala:

```
object Solution {
def getHint(secret: String, guess: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_hint(secret :: String.t, guess :: String.t) :: String.t
def get_hint(secret, guess) do

end
end
```

Erlang:

```
-spec get_hint(Secret :: unicode:unicode_binary(), Guess ::
unicode:unicode_binary()) -> unicode:unicode_binary().
get_hint(Secret, Guess) ->
.
```

Racket:

```
(define/contract (get-hint secret guess)
(-> string? string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Bulls and Cows
* Difficulty: Medium
* 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:
string getHint(string secret, string guess) {

}
};
```

Java Solution:

```
/**
* Problem: Bulls and Cows
* Difficulty: Medium
* 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 String getHint(String secret, String guess) {

}
}
```

Python3 Solution:

```
"""
Problem: Bulls and Cows
Difficulty: Medium
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 getHint(self, secret: str, guess: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getHint(self, secret, guess):
"""
:type secret: str
:type guess: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Bulls and Cows
* Difficulty: Medium
* 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} secret
* @param {string} guess
* @return {string}
*/
var getHint = function(secret, guess) {

};
```

TypeScript Solution:

```
/**
* Problem: Bulls and Cows
* Difficulty: Medium
* 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 getHint(secret: string, guess: string): string {

};
```

C# Solution:

```
/*
* Problem: Bulls and Cows
* Difficulty: Medium
* 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 string GetHint(string secret, string guess) {

}
}
```

C Solution:

```
/*
* Problem: Bulls and Cows
* Difficulty: Medium
* 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
```

```
*/

char* getHint(char* secret, char* guess) {

}
```

Go Solution:

```
// Problem: Bulls and Cows
// Difficulty: Medium
// 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 getHint(secret string, guess string) string {

}
```

Kotlin Solution:

```
class Solution {
fun getHint(secret: String, guess: String): String {

}
}
```

Swift Solution:

```
class Solution {
func getHint(_ secret: String, _ guess: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Bulls and Cows
// Difficulty: Medium
// 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 get_hint(secret: String, guess: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} secret
# @param {String} guess
# @return {String}
def get_hint(secret, guess)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $secret
* @param String $guess
* @return String
*/
function getHint($secret, $guess) {

}
}
```

Dart Solution:

```
class Solution {
String getHint(String secret, String guess) {

}
}
```

Scala Solution:

```
object Solution {
def getHint(secret: String, guess: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_hint(secret :: String.t, guess :: String.t) :: String.t
def get_hint(secret, guess) do

end
end
```

Erlang Solution:

```
-spec get_hint(Secret :: unicode:unicode_binary(), Guess ::
unicode:unicode_binary()) -> unicode:unicode_binary().
get_hint(Secret, Guess) ->
.
```

Racket Solution:

```
(define/contract (get-hint secret guess)
(-> string? string? string?)
)
```