Problem 383: Ransom Note

Problem Information

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

Problem Description

Given two strings

ransomNote

and

magazine

, return

true

if

ransomNote

can be constructed by using the letters from

magazine

and

false

otherwise

.

Each letter in

magazine

can only be used once in

ransomNote

.

Example 1:

Input:

ransomNote = "a", magazine = "b"

Output:

false

Example 2:

Input:

ransomNote = "aa", magazine = "ab"

Output:

false

Example 3:

Input:

ransomNote = "aa", magazine = "aab"

Output:

true

Constraints:

1 <= ransomNote.length, magazine.length <= 10

5

ransomNote

and

magazine

consist of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool canConstruct(string ransomNote, string magazine) {

}
};
```

Java:

```
class Solution {
public boolean canConstruct(String ransomNote, String magazine) {

}
}
```

Python3:

```
class Solution:
def canConstruct(self, ransomNote: str, magazine: str) -> bool:
```

Python:

```
class Solution(object):
def canConstruct(self, ransomNote, magazine):
"""
:type ransomNote: str
:type magazine: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} ransomNote
* @param {string} magazine
* @return {boolean}
*/
var canConstruct = function(ransomNote, magazine) {

};
```

TypeScript:

```
function canConstruct(ransomNote: string, magazine: string): boolean {

};
```

C#:

```
public class Solution {
public bool CanConstruct(string ransomNote, string magazine) {

}
}
```

C:

```
bool canConstruct(char* ransomNote, char* magazine) {

}
```

Go:

```
func canConstruct(ransomNote string, magazine string) bool {

}
```

Kotlin:

```
class Solution {
fun canConstruct(ransomNote: String, magazine: String): Boolean {

}
}
```

Swift:

```
class Solution {
func canConstruct(_ ransomNote: String, _ magazine: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn can_construct(ransom_note: String, magazine: String) -> bool {

}
}
```

Ruby:

```
# @param {String} ransom_note
# @param {String} magazine
# @return {Boolean}
def can_construct(ransom_note, magazine)

end
```

PHP:

```
class Solution {

/**
* @param String $ransomNote
```

```
* @param String $magazine
* @return Boolean
*/
function canConstruct($ransomNote, $magazine) {

}
}
```

Dart:

```
class Solution {
bool canConstruct(String ransomNote, String magazine) {

}
}
```

Scala:

```
object Solution {
def canConstruct(ransomNote: String, magazine: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec can_construct(ransom_note :: String.t, magazine :: String.t) :: boolean
def can_construct(ransom_note, magazine) do

end
end
```

Erlang:

```
-spec can_construct(RansomNote :: unicode:unicode_binary(), Magazine ::
unicode:unicode_binary()) -> boolean().
can_construct(RansomNote, Magazine) ->
.
```

Racket:

```
(define/contract (can-construct ransomNote magazine)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Ransom Note
* 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:
bool canConstruct(string ransomNote, string magazine) {

}
};
```

Java Solution:

```
/**
* Problem: Ransom Note
* 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 boolean canConstruct(String ransomNote, String magazine) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Ransom Note
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 canConstruct(self, ransomNote: str, magazine: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def canConstruct(self, ransomNote, magazine):
"""
:type ransomNote: str
:type magazine: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Ransom Note
* 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} ransomNote
* @param {string} magazine
* @return {boolean}
*/
var canConstruct = function(ransomNote, magazine) {

};
```

TypeScript Solution:

```
/**
* Problem: Ransom Note
* 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 canConstruct(ransomNote: string, magazine: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Ransom Note
* 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 bool CanConstruct(string ransomNote, string magazine) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Ransom Note
* 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
*/

bool canConstruct(char* ransomNote, char* magazine) {

}
```

Go Solution:

```
// Problem: Ransom Note
// 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 canConstruct(ransomNote string, magazine string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun canConstruct(ransomNote: String, magazine: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func canConstruct(_ ransomNote: String, _ magazine: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Ransom Note
// 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 can_construct(ransom_note: String, magazine: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} ransom_note
# @param {String} magazine
# @return {Boolean}
def can_construct(ransom_note, magazine)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $ransomNote
* @param String $magazine
* @return Boolean
*/
function canConstruct($ransomNote, $magazine) {
```

```
}
}
```

Dart Solution:

```
class Solution {
bool canConstruct(String ransomNote, String magazine) {

}
}
```

Scala Solution:

```
object Solution {
def canConstruct(ransomNote: String, magazine: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec can_construct(ransom_note :: String.t, magazine :: String.t) :: boolean
def can_construct(ransom_note, magazine) do

end
end
```

Erlang Solution:

```
-spec can_construct(RansomNote :: unicode:unicode_binary(), Magazine ::
unicode:unicode_binary()) -> boolean().
can_construct(RansomNote, Magazine) ->
.
```

Racket Solution:

```
(define/contract (can-construct ransomNote magazine)
(-> string? string? boolean?)
)
```