Problem 758: Bold Words in String

Problem Information

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

Problem Description

Given an array of keywords

words

and a string

s

, make all appearances of all keywords

words[i]

in

s

bold. Any letters between

<b>

and

</b>

tags become bold.

Return

s

after adding the bold tags

. The returned string should use the least number of tags possible, and the tags should form a valid combination.

Example 1:

Input:

words = ["ab","bc"], s = "aabcd"

Output:

"a<b>abc</b>d"

Explanation:

Note that returning

"a<b>a<b>b</b>c</b>d"

would use more tags, so it is incorrect.

Example 2:

Input:

words = ["ab","cb"], s = "aabcd"

Output:

"a<b>ab</b>cd"

Constraints:

1 <= s.length <= 500

0 <= words.length <= 50

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

s

and

words[i]

consist of lowercase English letters.

Note:

This question is the same as

616. Add Bold Tag in String

.

Code Snippets

C++:

```
class Solution {
public:
string boldWords(vector<string>& words, string s) {

}
};
```

Java:

```
class Solution {
public String boldWords(String[] words, String s) {

}
}
```

Python3:

```
class Solution:
def boldWords(self, words: List[str], s: str) -> str:
```

Python:

```
class Solution(object):
def boldWords(self, words, s):
"""
:type words: List[str]
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string[]} words
* @param {string} s
* @return {string}
*/
var boldWords = function(words, s) {

};
```

TypeScript:

```
function boldWords(words: string[], s: string): string {

};
```

C#:

```
public class Solution {
public string BoldWords(string[] words, string s) {

}
}
```

C:

```
char* boldWords(char** words, int wordsSize, char* s) {

}
```

Go:

```
func boldWords(words []string, s string) string {

}
```

Kotlin:

```
class Solution {
fun boldWords(words: Array<String>, s: String): String {

}
}
```

Swift:

```
class Solution {
func boldWords(_ words: [String], _ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn bold_words(words: Vec<String>, s: String) -> String {

}
}
```

Ruby:

```
# @param {String[]} words
# @param {String} s
# @return {String}
def bold_words(words, s)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @param String $s
* @return String
*/
function boldWords($words, $s) {

}
}
```

Dart:

```
class Solution {
String boldWords(List<String> words, String s) {

}
}
```

Scala:

```
object Solution {
def boldWords(words: Array[String], s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec bold_words(words :: [String.t], s :: String.t) :: String.t
def bold_words(words, s) do

end
end
```

Erlang:

```
-spec bold_words(Words :: [unicode:unicode_binary()], S ::
unicode:unicode_binary()) -> unicode:unicode_binary().
bold_words(Words, S) ->
.
```

Racket:

```
(define/contract (bold-words words s)
(-> (listof string?) string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Bold Words in String
* 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:
string boldWords(vector<string>& words, string s) {

}
};
```

Java Solution:

```
/**
* Problem: Bold Words in String
* 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 String boldWords(String[] words, String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Bold Words in String
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 boldWords(self, words: List[str], s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def boldWords(self, words, s):
"""
:type words: List[str]
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Bold Words in String
* 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[]} words
* @param {string} s
* @return {string}
*/
var boldWords = function(words, s) {

};
```

TypeScript Solution:

```
/**
* Problem: Bold Words in String
* 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 boldWords(words: string[], s: string): string {

};
```

C# Solution:

```
/*
* Problem: Bold Words in String
* 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 string BoldWords(string[] words, string s) {

}
}
```

C Solution:

```
/*
* Problem: Bold Words in String
* 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
*/

char* boldWords(char** words, int wordsSize, char* s) {

}
```

Go Solution:

```
// Problem: Bold Words in String
// 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 boldWords(words []string, s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun boldWords(words: Array<String>, s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func boldWords(_ words: [String], _ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Bold Words in String
// 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 bold_words(words: Vec<String>, s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @param {String} s
# @return {String}
def bold_words(words, s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @param String $s
* @return String
*/
function boldWords($words, $s) {

}
}
```

Dart Solution:

```
class Solution {
String boldWords(List<String> words, String s) {

}
}
```

Scala Solution:

```
object Solution {
def boldWords(words: Array[String], s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec bold_words(words :: [String.t], s :: String.t) :: String.t
def bold_words(words, s) do

end
end
```

Erlang Solution:

```
-spec bold_words(Words :: [unicode:unicode_binary()], S ::
unicode:unicode_binary()) -> unicode:unicode_binary().
bold_words(Words, S) ->
```

```
.
```

Racket Solution:

```
(define/contract (bold-words words s)
(-> (listof string?) string? string?)
)
```