Problem 616: Add Bold Tag in String

Problem Information

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

Problem Description

You are given a string

s

and an array of strings

words

.

You should add a closed pair of bold tag

and

to wrap the substrings in

s

that exist in

words

.

If two such substrings overlap, you should wrap them together with only one pair of closed bold-tag.

If two substrings wrapped by bold tags are consecutive, you should combine them.

Return

s

after adding the bold tags

.

Example 1:

Input:

s = "abcxyz123", words = ["abc","123"]

Output:

"abcxyz123"

Explanation:

The two strings of words are substrings of s as following: "

abc

xyz

123

". We add before each substring and after each substring.

Example 2:

Input:

s = "aaabbb", words = ["aa","b"]

Output:

"aaabbb"

Explanation:

"aa" appears as a substring two times: "

aa

abbb" and "a

aa

bbb". "b" appears as a substring three times: "aaa

b

bb", "aaab

b

b", and "aaabb

b

". We add before each substring and after each substring: "aaabbb". Since the first two 's overlap, we merge them: "aaabbb". Since now the four 's are consecutive, we merge them: "aaabbb".

Constraints:

1 <= s.length <= 1000

0 <= words.length <= 100

1 <= words[i].length <= 1000

s

and

words[i]

consist of English letters and digits.

All the values of

words

are

unique

.

Note:

This question is the same as

758. Bold Words in String

.

Code Snippets

C++:

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

```
}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
def add_bold_tag(s, words) do

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Add Bold Tag in String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Add Bold Tag in String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Add Bold Tag in String
Difficulty: Medium
Tags: array, string, tree, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def addBoldTag(self, s: str, words: List[str]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Add Bold Tag in String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Add Bold Tag in String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function addBoldTag(s: string, words: string[]): string {

};
```

C# Solution:

```
/*
* Problem: Add Bold Tag in String
* Difficulty: Medium
```

```
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

C Solution:

```
/*
* Problem: Add Bold Tag in String
* Difficulty: Medium
* Tags: array, string, tree, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
```

Go Solution:

```
// Problem: Add Bold Tag in String
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

```
}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Add Bold Tag in String
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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