Problem 2109: Adding Spaces to a String

Problem Information

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

Problem Description

You are given a

0-indexed

string

s

and a

0-indexed

integer array

spaces

that describes the indices in the original string where spaces will be added. Each space should be inserted

before

the character at the given index.

For example, given

s = "EnjoyYourCoffee"

and

spaces = [5, 9]

, we place spaces before

'Y'

and

'C'

, which are at indices

5

and

9

respectively. Thus, we obtain

"Enjoy

Y

our

C

offee"

.

Return

the modified string

after

the spaces have been added.

Example 1:

Input:

s = "LeetcodeHelpsMeLearn", spaces = [8,13,15]

Output:

"Leetcode Helps Me Learn"

Explanation:

The indices 8, 13, and 15 correspond to the underlined characters in "Leetcode

H

elps

M

e

L

earn". We then place spaces before those characters.

Example 2:

Input:

s = "icodeinpython", spaces = [1,5,7,9]

Output:

"i code in py thon"

Explanation:

The indices 1, 5, 7, and 9 correspond to the underlined characters in "i

c

ode

i

n

p

y

t

hon". We then place spaces before those characters.

Example 3:

Input:

s = "spacing", spaces = [0,1,2,3,4,5,6]

Output:

" s p a c i n g"

Explanation:

We are also able to place spaces before the first character of the string.

Constraints:

1 <= s.length <= 3 * 10

5

s

consists only of lowercase and uppercase English letters.

1 <= spaces.length <= 3 * 10

5

0 <= spaces[i] <= s.length - 1

All the values of

spaces

are

strictly increasing

.

Code Snippets

C++:

```
class Solution {
public:
string addSpaces(string s, vector<int>& spaces) {

}
};
```

Java:

```
class Solution {
public String addSpaces(String s, int[] spaces) {

}
}
```

Python3:

```
class Solution:
def addSpaces(self, s: str, spaces: List[int]) -> str:
```

Python:

```
class Solution(object):
def addSpaces(self, s, spaces):
"""
:type s: str
:type spaces: List[int]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number[]} spaces
* @return {string}
*/
var addSpaces = function(s, spaces) {

};
```

TypeScript:

```
function addSpaces(s: string, spaces: number[]): string {

};
```

C#:

```
public class Solution {
public string AddSpaces(string s, int[] spaces) {

}
}
```

C:

```
char* addSpaces(char* s, int* spaces, int spacesSize) {

}
```

Go:

```
func addSpaces(s string, spaces []int) string {

}
```

Kotlin:

```
class Solution {
fun addSpaces(s: String, spaces: IntArray): String {

}
}
```

Swift:

```
class Solution {
func addSpaces(_ s: String, _ spaces: [Int]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn add_spaces(s: String, spaces: Vec<i32>) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer[]} spaces
# @return {String}
def add_spaces(s, spaces)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer[] $spaces
* @return String
*/
function addSpaces($s, $spaces) {

}
}
```

Dart:

```
class Solution {
String addSpaces(String s, List<int> spaces) {

}
}
```

Scala:

```
object Solution {
def addSpaces(s: String, spaces: Array[Int]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec add_spaces(s :: String.t, spaces :: [integer]) :: String.t
def add_spaces(s, spaces) do

end
end
```

Erlang:

```
-spec add_spaces(S :: unicode:unicode_binary(), Spaces :: [integer()]) ->
unicode:unicode_binary().
add_spaces(S, Spaces) ->
.
```

Racket:

```
(define/contract (add-spaces s spaces)
(-> string? (listof exact-integer?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Adding Spaces to a String
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string addSpaces(string s, vector<int>& spaces) {

}
};
```

Java Solution:

```
/**
* Problem: Adding Spaces to a String
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String addSpaces(String s, int[] spaces) {

}
}
```

Python3 Solution:

```
"""
Problem: Adding Spaces to a String
Difficulty: Medium
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

```
class Solution(object):
def addSpaces(self, s, spaces):
"""
:type s: str
:type spaces: List[int]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Adding Spaces to a String
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @param {number[]} spaces
* @return {string}
*/
var addSpaces = function(s, spaces) {

};
```

TypeScript Solution:

```
/**
* Problem: Adding Spaces to a String
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function addSpaces(s: string, spaces: number[]): string {

};
```

C# Solution:

```
/*
* Problem: Adding Spaces to a String
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string AddSpaces(string s, int[] spaces) {

}
}
```

C Solution:

```
/*
* Problem: Adding Spaces to a String
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* addSpaces(char* s, int* spaces, int spacesSize) {

}
```

Go Solution:

```
// Problem: Adding Spaces to a String
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func addSpaces(s string, spaces []int) string {

}
```

Kotlin Solution:

```
class Solution {
fun addSpaces(s: String, spaces: IntArray): String {

}
}
```

Swift Solution:

```
class Solution {
func addSpaces(_ s: String, _ spaces: [Int]) -> String {

}
}
```

Rust Solution:

```
// Problem: Adding Spaces to a String
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn add_spaces(s: String, spaces: Vec<i32>) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer[]} spaces
# @return {String}
def add_spaces(s, spaces)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer[] $spaces
* @return String
*/
function addSpaces($s, $spaces) {

}
}
```

Dart Solution:

```
class Solution {
String addSpaces(String s, List<int> spaces) {

}
}
```

Scala Solution:

```
object Solution {
def addSpaces(s: String, spaces: Array[Int]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec add_spaces(s :: String.t, spaces :: [integer]) :: String.t
def add_spaces(s, spaces) do

end
end
```

Erlang Solution:

```
-spec add_spaces(S :: unicode:unicode_binary(), Spaces :: [integer()]) ->
unicode:unicode_binary().
add_spaces(S, Spaces) ->
```

```
.
```

Racket Solution:

```
(define/contract (add-spaces s spaces)
(-> string? (listof exact-integer?) string?)
)
```