Problem 806: Number of Lines To Write String

Problem Information

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

Problem Description

You are given a string

s

of lowercase English letters and an array

widths

denoting

how many pixels wide

each lowercase English letter is. Specifically,

widths[0]

is the width of

'a'

,

widths[1]

is the width of

'b'

, and so on.

You are trying to write

s

across several lines, where

each line is no longer than

100

pixels

. Starting at the beginning of

s

, write as many letters on the first line such that the total width does not exceed

100

pixels. Then, from where you stopped in

s

, continue writing as many letters as you can on the second line. Continue this process until you have written all of

s

.

Return

an array

result

of length 2 where:

result[0]

is the total number of lines.

result[1]

is the width of the last line in pixels.

Example 1:

Input:

widths = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10], s = "abcdefghijklmnopqrstuvwxyz"

Output:

[3,60]

Explanation:

You can write s as follows: abcdefghij // 100 pixels wide klmnopqrst // 100 pixels wide uvwxyz // 60 pixels wide There are a total of 3 lines, and the last line is 60 pixels wide.

Example 2:

Input:

widths = [4,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10], s = "bbbcccdddaaa"

Output:

[2,4]

Explanation:

You can write s as follows: bbbccccdddaa // 98 pixels wide a // 4 pixels wide There are a total of 2 lines, and the last line is 4 pixels wide.

Constraints:

widths.length == 26

2 <= widths[i] <= 10

1 <= s.length <= 1000

s

contains only lowercase English letters.

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int[] numberOfLines(int[] widths, String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function numberOfLines(widths: number[], s: string): number[] {

};
```

C#:

```
public class Solution {
public int[] NumberOfLines(int[] widths, string s) {

}
}
```

C:

```


/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* numberOfLines(int* widths, int widthsSize, char * s, int* returnSize){
```

```
}
```

Go:

```
func numberOfLines(widths []int, s string) []int {

}
```

Kotlin:

```
class Solution {
fun numberOfLines(widths: IntArray, s: String): IntArray {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Scala:

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

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Lines To Write String
* Difficulty: Easy
* 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:
vector<int> numberOfLines(vector<int>& widths, string s) {

}
```

```
};
```

Java Solution:

```
/**
* Problem: Number of Lines To Write String
* Difficulty: Easy
* 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 int[] numberOfLines(int[] widths, String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Lines To Write String
Difficulty: Easy
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 numberOfLines(self, widths: List[int], s: str) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfLines(self, widths, s):
```

```
"""
:type widths: List[int]
:type s: str
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Lines To Write String
* Difficulty: Easy
* 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 {number[]} widths
* @param {string} s
* @return {number[]}
*/
var numberOfLines = function(widths, s) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Lines To Write String
* Difficulty: Easy
* 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 numberOfLines(widths: number[], s: string): number[] {
```

```
};
```

C# Solution:

```
/*
* Problem: Number of Lines To Write String
* Difficulty: Easy
* 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 int[] NumberOfLines(int[] widths, string s) {

}
}
```

C Solution:

```
/*
* Problem: Number of Lines To Write String
* Difficulty: Easy
* 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
*/




/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* numberOfLines(int* widths, int widthsSize, char * s, int* returnSize){

}
```

Go Solution:

```
// Problem: Number of Lines To Write String
// Difficulty: Easy
// 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 numberOfLines(widths []int, s string) []int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfLines(widths: IntArray, s: String): IntArray {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Lines To Write String
// Difficulty: Easy
// 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 number_of_lines(widths: Vec<i32>, s: String) -> Vec<i32> {
```

```
}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Scala Solution:

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

}
}
```