Problem 1165: Single-Row Keyboard

Problem Information

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

Problem Description

There is a special keyboard with

all keys in a single row

.

Given a string

keyboard

of length

26

indicating the layout of the keyboard (indexed from

0

to

25

). Initially, your finger is at index

0

. To type a character, you have to move your finger to the index of the desired character. The time taken to move your finger from index

i

to index

j

is

|i - j|

.

You want to type a string

word

. Write a function to calculate how much time it takes to type it with one finger.

Example 1:

Input:

keyboard = "abcdefghijklmnopqrstuvwxyz", word = "cba"

Output:

4

Explanation:

The index moves from 0 to 2 to write 'c' then to 1 to write 'b' then to 0 again to write 'a'. Total time = 2 + 1 + 1 = 4.

Example 2:

Input:

keyboard = "pqrstuvwxyzabcdefghijklmno", word = "leetcode"

Output:

73

Constraints:

keyboard.length == 26

keyboard

contains each English lowercase letter exactly once in some order.

1 <= word.length <= 10

4

word[i]

is an English lowercase letter.

Code Snippets

C++:

```
class Solution {
public:
int calculateTime(string keyboard, string word) {

}
};
```

Java:

```
class Solution {
public int calculateTime(String keyboard, String word) {

}
}
```

Python3:

```
class Solution:
def calculateTime(self, keyboard: str, word: str) -> int:
```

Python:

```
class Solution(object):
def calculateTime(self, keyboard, word):
"""
:type keyboard: str
:type word: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} keyboard
* @param {string} word
* @return {number}
*/
var calculateTime = function(keyboard, word) {

};
```

TypeScript:

```
function calculateTime(keyboard: string, word: string): number {

};
```

C#:

```
public class Solution {
public int CalculateTime(string keyboard, string word) {

}
}
```

C:

```
int calculateTime(char* keyboard, char* word) {

}
```

Go:

```
func calculateTime(keyboard string, word string) int {

}
```

Kotlin:

```
class Solution {
fun calculateTime(keyboard: String, word: String): Int {

}
}
```

Swift:

```
class Solution {
func calculateTime(_ keyboard: String, _ word: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn calculate_time(keyboard: String, word: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} keyboard
# @param {String} word
# @return {Integer}
def calculate_time(keyboard, word)
```

```

end
```

PHP:

```
class Solution {

/**
* @param String $keyboard
* @param String $word
* @return Integer
*/
function calculateTime($keyboard, $word) {

}
}
```

Dart:

```
class Solution {
int calculateTime(String keyboard, String word) {

}
}
```

Scala:

```
object Solution {
def calculateTime(keyboard: String, word: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec calculate_time(keyboard :: String.t, word :: String.t) :: integer
def calculate_time(keyboard, word) do

end
end
```

Erlang:

```
-spec calculate_time(Keyboard :: unicode:unicode_binary(), Word ::
unicode:unicode_binary()) -> integer().
calculate_time(Keyboard, Word) ->
.
```

Racket:

```
(define/contract (calculate-time keyboard word)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Single-Row Keyboard
* 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:
int calculateTime(string keyboard, string word) {

}
};
```

Java Solution:

```
/**
* Problem: Single-Row Keyboard
* 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 int calculateTime(String keyboard, String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Single-Row Keyboard
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 calculateTime(self, keyboard: str, word: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def calculateTime(self, keyboard, word):
"""
:type keyboard: str
:type word: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Single-Row Keyboard
* 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} keyboard
* @param {string} word
* @return {number}
*/
var calculateTime = function(keyboard, word) {

};
```

TypeScript Solution:

```
/**
* Problem: Single-Row Keyboard
* 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 calculateTime(keyboard: string, word: string): number {

};
```

C# Solution:

```
/*
* Problem: Single-Row Keyboard
* 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 int CalculateTime(string keyboard, string word) {

}
}
```

C Solution:

```
/*
* Problem: Single-Row Keyboard
* 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
*/

int calculateTime(char* keyboard, char* word) {

}
```

Go Solution:

```
// Problem: Single-Row Keyboard
// 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 calculateTime(keyboard string, word string) int {

}
```

Kotlin Solution:

```
class Solution {
fun calculateTime(keyboard: String, word: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func calculateTime(_ keyboard: String, _ word: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Single-Row Keyboard
// 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 calculate_time(keyboard: String, word: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} keyboard
# @param {String} word
# @return {Integer}
def calculate_time(keyboard, word)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $keyboard
* @param String $word
* @return Integer
*/
function calculateTime($keyboard, $word) {

}
}
```

Dart Solution:

```
class Solution {
int calculateTime(String keyboard, String word) {

}
}
```

Scala Solution:

```
object Solution {
def calculateTime(keyboard: String, word: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec calculate_time(keyboard :: String.t, word :: String.t) :: integer
def calculate_time(keyboard, word) do

end
end
```

Erlang Solution:

```
-spec calculate_time(Keyboard :: unicode:unicode_binary(), Word ::
unicode:unicode_binary()) -> integer().
calculate_time(Keyboard, Word) ->
```

```
.
```

Racket Solution:

```
(define/contract (calculate-time keyboard word)
(-> string? string? exact-integer?)
)
```