Problem 1974: Minimum Time to Type Word Using Special Typewriter

Problem Information

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

Problem Description

There is a special typewriter with lowercase English letters

'a'

to

'z'

arranged in a

circle

with a

pointer

. A character can

only

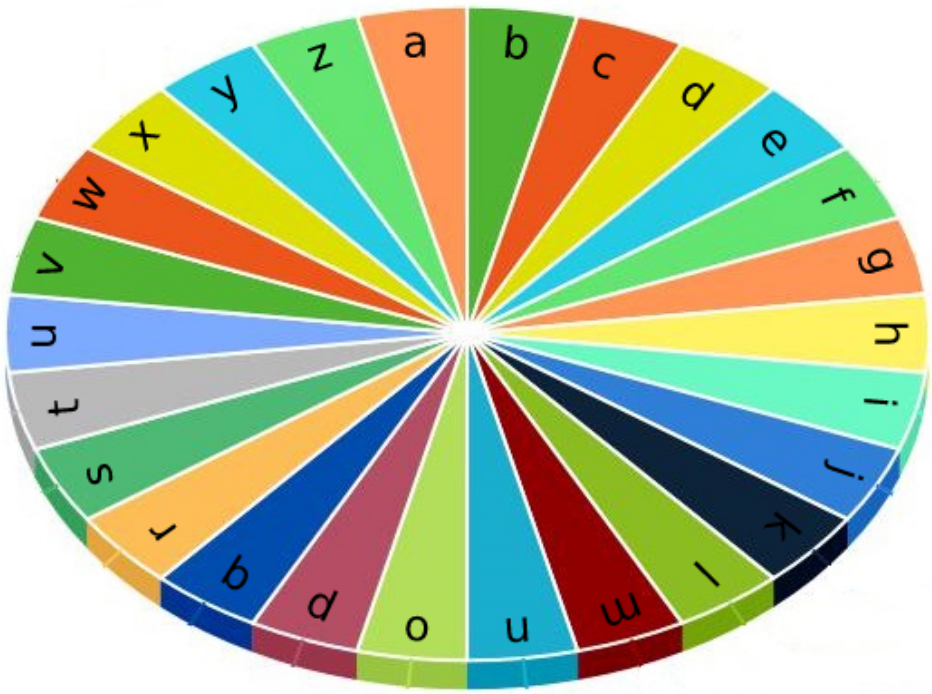
be typed if the pointer is pointing to that character. The pointer is

initially

pointing to the character

'a'

.



Each second, you may perform one of the following operations:

Move the pointer one character

counterclockwise

or

clockwise

.

Type the character the pointer is

currently

on.

Given a string

word

, return the

minimum

number of seconds to type out the characters in

word

.

Example 1:

Input:

word = "abc"

Output:

5

Explanation:

The characters are printed as follows: - Type the character 'a' in 1 second since the pointer is initially on 'a'. - Move the pointer clockwise to 'b' in 1 second. - Type the character 'b' in 1 second. - Move the pointer clockwise to 'c' in 1 second. - Type the character 'c' in 1 second.

Example 2:

Input:

word = "bza"

Output:

7

Explanation:

The characters are printed as follows: - Move the pointer clockwise to 'b' in 1 second. - Type the character 'b' in 1 second. - Move the pointer counterclockwise to 'z' in 2 seconds. - Type the character 'z' in 1 second. - Move the pointer clockwise to 'a' in 1 second. - Type the character 'a' in 1 second.

Example 3:

Input:

word = "zjpc"

Output:

34

Explanation:

The characters are printed as follows: - Move the pointer counterclockwise to 'z' in 1 second. - Type the character 'z' in 1 second. - Move the pointer clockwise to 'j' in 10 seconds. - Type the character 'j' in 1 second. - Move the pointer clockwise to 'p' in 6 seconds. - Type the character 'p' in 1 second. - Move the pointer counterclockwise to 'c' in 13 seconds. - Type the character 'c' in 1 second.

Constraints:

1 <= word.length <= 100

word

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int minTimeToType(string word) {

}
};
```

Java:

```
class Solution {
public int minTimeToType(String word) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minTimeToType(word: string): number {
```

```
};
```

C#:

```
public class Solution {
public int MinTimeToType(string word) {

}
}
```

C:

```
int minTimeToType(char* word) {

}
```

Go:

```
func minTimeToType(word string) int {

}
```

Kotlin:

```
class Solution {
fun minTimeToType(word: String): Int {

}
}
```

Swift:

```
class Solution {
func minTimeToType(_ word: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_time_to_type(word: String) -> i32 {
```

```
}
}
```

Ruby:

```
# @param {String} word
# @return {Integer}
def min_time_to_type(word)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @return Integer
*/
function minTimeToType($word) {

}
}
```

Dart:

```
class Solution {
int minTimeToType(String word) {

}
}
```

Scala:

```
object Solution {
def minTimeToType(word: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_time_to_type(word :: String.t) :: integer
def min_time_to_type(word) do

end
end
```

Erlang:

```
-spec min_time_to_type(Word :: unicode:unicode_binary()) -> integer().
min_time_to_type(Word) ->
.
```

Racket:

```
(define/contract (min-time-to-type word)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Time to Type Word Using Special Typewriter
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int minTimeToType(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Time to Type Word Using Special Typewriter
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int minTimeToType(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Time to Type Word Using Special Typewriter
Difficulty: Easy
Tags: string, greedy

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def minTimeToType(self, word: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Time to Type Word Using Special Typewriter
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} word
* @return {number}
*/
var minTimeToType = function(word) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Time to Type Word Using Special Typewriter
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function minTimeToType(word: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Time to Type Word Using Special Typewriter
* Difficulty: Easy
* Tags: string, greedy
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MinTimeToType(string word) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Time to Type Word Using Special Typewriter
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int minTimeToType(char* word) {

}
```

Go Solution:

```
// Problem: Minimum Time to Type Word Using Special Typewriter
// Difficulty: Easy
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func minTimeToType(word string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minTimeToType(word: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minTimeToType(_ word: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Time to Type Word Using Special Typewriter
// Difficulty: Easy
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn min_time_to_type(word: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} word
# @return {Integer}
def min_time_to_type(word)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @return Integer
*/
function minTimeToType($word) {

}
}
```

Dart Solution:

```
class Solution {
int minTimeToType(String word) {

}
}
```

Scala Solution:

```
object Solution {
def minTimeToType(word: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_time_to_type(word :: String.t) :: integer
def min_time_to_type(word) do

end
end
```

Erlang Solution:

```
-spec min_time_to_type(Word :: unicode:unicode_binary()) -> integer().
min_time_to_type(Word) ->
.
```

Racket Solution:

```
(define/contract (min-time-to-type word)
(-> string? exact-integer?)
)
```