Problem 1682: Longest Palindromic Subsequence II

Problem Information

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

Problem Description

A subsequence of a string

s

is considered a

good palindromic subsequence

if:

It is a subsequence of

s

.

It is a palindrome (has the same value if reversed).

It has an

even

length.

No two consecutive characters are equal, except the two middle ones.

For example, if

s = "abcabcabb"

, then

"abba"

is considered a

good palindromic subsequence

, while

"bcb"

(not even length) and

"bbbb"

(has equal consecutive characters) are not.

Given a string

s

, return

the

length

of the

longest good palindromic subsequence

in

s

.

Example 1:

Input:

s = "bbabab"

Output:

4

Explanation:

The longest good palindromic subsequence of s is "baab".

Example 2:

Input:

s = "dcbccacdb"

Output:

4

Explanation:

The longest good palindromic subsequence of s is "dccd".

Constraints:

1 <= s.length <= 250

s

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int longestPalindromeSubseq(string s) {

}
};
```

Java:

```
class Solution {
public int longestPalindromeSubseq(String s) {

}
}
```

Python3:

```
class Solution:
def longestPalindromeSubseq(self, s: str) -> int:
```

Python:

```
class Solution(object):
def longestPalindromeSubseq(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var longestPalindromeSubseq = function(s) {
```

```
};
```

TypeScript:

```
function longestPalindromeSubseq(s: string): number {

};
```

C#:

```
public class Solution {
public int LongestPalindromeSubseq(string s) {

}
}
```

C:

```
int longestPalindromeSubseq(char* s) {

}
```

Go:

```
func longestPalindromeSubseq(s string) int {

}
```

Kotlin:

```
class Solution {
fun longestPalindromeSubseq(s: String): Int {

}
}
```

Swift:

```
class Solution {
func longestPalindromeSubseq(_ s: String) -> Int {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn longest_palindrome_subseq(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def longest_palindrome_subseq(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function longestPalindromeSubseq($s) {

}
}
```

Dart:

```
class Solution {
int longestPalindromeSubseq(String s) {

}
}
```

Scala:

```
object Solution {
def longestPalindromeSubseq(s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_palindrome_subseq(s :: String.t) :: integer
def longest_palindrome_subseq(s) do

end
end
```

Erlang:

```
-spec longest_palindrome_subseq(S :: unicode:unicode_binary()) -> integer().
longest_palindrome_subseq(S) ->
.
```

Racket:

```
(define/contract (longest-palindrome-subseq s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Palindromic Subsequence II
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
int longestPalindromeSubseq(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Palindromic Subsequence II
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int longestPalindromeSubseq(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Palindromic Subsequence II
Difficulty: Medium
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

```
class Solution(object):
def longestPalindromeSubseq(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Palindromic Subsequence II
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s
* @return {number}
*/
var longestPalindromeSubseq = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Palindromic Subsequence II
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function longestPalindromeSubseq(s: string): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Longest Palindromic Subsequence II
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int LongestPalindromeSubseq(string s) {

}
}
```

C Solution:

```
/*
* Problem: Longest Palindromic Subsequence II
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int longestPalindromeSubseq(char* s) {

}
```

Go Solution:

```
// Problem: Longest Palindromic Subsequence II
// Difficulty: Medium
```

```
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func longestPalindromeSubseq(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun longestPalindromeSubseq(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func longestPalindromeSubseq(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Longest Palindromic Subsequence II
// Difficulty: Medium
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn longest_palindrome_subseq(s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def longest_palindrome_subseq(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function longestPalindromeSubseq($s) {

}
}
```

Dart Solution:

```
class Solution {
int longestPalindromeSubseq(String s) {

}
}
```

Scala Solution:

```
object Solution {
def longestPalindromeSubseq(s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_palindrome_subseq(s :: String.t) :: integer
def longest_palindrome_subseq(s) do
```

```
end
end
```

Erlang Solution:

```
-spec longest_palindrome_subseq(S :: unicode:unicode_binary()) -> integer().
longest_palindrome_subseq(S) ->
.
```

Racket Solution:

```
(define/contract (longest-palindrome-subseq s)
(-> string? exact-integer?)
)
```