# Problem 647: Palindromic Substrings

## Problem Information

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

## Problem Description

Given a string

s

, return

the number of

palindromic substrings

in it

.

A string is a

palindrome

when it reads the same backward as forward.

A

substring

is a contiguous sequence of characters within the string.

Example 1:

Input:

s = "abc"

Output:

3

Explanation:

Three palindromic strings: "a", "b", "c".

Example 2:

Input:

s = "aaa"

Output:

6

Explanation:

Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

Constraints:

1 <= s.length <= 1000

s

consists of lowercase English letters.

## Code Snippets

**C++:**

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

}
};
```

**Java:**

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

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
int countSubstrings(char* s) {

}
```

**Go:**

```
func countSubstrings(s string) int {

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

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

}
}
```

**Scala:**

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

}
}
```

### Elixir:

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

end
end
```

### Erlang:

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

### Racket:

```
(define/contract (count-substrings s)
(-> string? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Palindromic Substrings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int countSubstrings(string s) {

}
};
```

### Java Solution:

```
/**
* Problem: Palindromic Substrings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int countSubstrings(String s) {

}
}
```

### Python3 Solution:

```
"""
Problem: Palindromic Substrings
Difficulty: Medium
Tags: array, string, tree, dp

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

### Python Solution:

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

```
"""
```

### JavaScript Solution:

```
/**
* Problem: Palindromic Substrings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* 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 countSubstrings = function(s) {

};
```

### TypeScript Solution:

```
/**
* Problem: Palindromic Substrings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function countSubstrings(s: string): number {

};
```

### C# Solution:

```
/*
* Problem: Palindromic Substrings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* 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 CountSubstrings(string s) {

}
}
```

### C Solution:

```
/*
* Problem: Palindromic Substrings
* Difficulty: Medium
* Tags: array, string, tree, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int countSubstrings(char* s) {

}
```

### Go Solution:

```
// Problem: Palindromic Substrings
// Difficulty: Medium
// Tags: array, string, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table
```

```
func countSubstrings(s string) int {

}
```

**Kotlin Solution:**

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

}
}
```

**Swift Solution:**

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

}
}
```

**Rust Solution:**

```
// Problem: Palindromic Substrings
// Difficulty: Medium
// Tags: array, string, tree, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

**Ruby Solution:**

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

```
end
```

**PHP Solution:**

```
class Solution {

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

}
}
```

**Dart Solution:**

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

}
}
```

**Scala Solution:**

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

}
}
```

**Elixir Solution:**

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

end
end
```

## Erlang Solution:

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

## Racket Solution:

```
(define/contract (count-substrings s)
(-> string? exact-integer?)
)
```