# Problem 1542: Find Longest Awesome Substring

## Problem Information

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

## Problem Description

You are given a string

s

. An

awesome

substring is a non-empty substring of

s

such that we can make any number of swaps in order to make it a palindrome.

Return

the length of the maximum length

awesome substring

of

s

.

Example 1:

Input:

s = "3242415"

Output:

5

Explanation:

"24241" is the longest awesome substring, we can form the palindrome "24142" with some swaps.

Example 2:

Input:

s = "12345678"

Output:

1

Example 3:

Input:

s = "213123"

Output:

6

Explanation:

"213123" is the longest awesome substring, we can form the palindrome "231132" with some swaps.

Constraints:

1 <= s.length <= 10

5

s

consists only of digits.

## Code Snippets

**C++:**

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

}
};
```

**Java:**

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

}
}
```

**Python3:**

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

**Python:**

```
class Solution(object):
def longestAwesome(self, s):
"""
:type s: str
```

```
:rtype: int
"""
```

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
int longestAwesome(char* s) {

}
```

**Go:**

```
func longestAwesome(s string) int {

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

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

}
}
```

**Scala:**

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

}
}
```

**Elixir:**

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

end
end
```

**Erlang:**

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

**Racket:**

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

## Solutions

**C++ Solution:**

```
/*
* Problem: Find Longest Awesome Substring
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
};
```

### Java Solution:

```
/**
* Problem: Find Longest Awesome Substring
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int longestAwesome(String s) {

}
}
```

### Python3 Solution:

```
"""
Problem: Find Longest Awesome Substring
Difficulty: Hard
Tags: string, tree, hash
```

```
Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

**Python Solution:**

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

**JavaScript Solution:**

```
/**
* Problem: Find Longest Awesome Substring
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

};
```

**TypeScript Solution:**

```
/**
* Problem: Find Longest Awesome Substring
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function longestAwesome(s: string): number {

};
```

### C# Solution:

```
/*
* Problem: Find Longest Awesome Substring
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

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

}
}
```

### C Solution:

```
/*
* Problem: Find Longest Awesome Substring
* Difficulty: Hard
* Tags: string, tree, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
```

```
*/

int longestAwesome(char* s) {

}
```

### Go Solution:

```
// Problem: Find Longest Awesome Substring
// Difficulty: Hard
// Tags: string, tree, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func longestAwesome(s string) int {

}
```

### Kotlin Solution:

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

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Find Longest Awesome Substring
// Difficulty: Hard
// Tags: string, tree, hash
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

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

}
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

}
}
```

### Dart Solution:

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

}
}
```

### Scala Solution:

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

}
}
```

### Elixir Solution:

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

end
end
```

### Erlang Solution:

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

### Racket Solution:

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