

# Problem 1156: Swap For Longest Repeated Character Substring

## Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

You are given a string

text

. You can swap two of the characters in the

text

Return

the length of the longest substring with repeated characters

Example 1:

Input:

text = "ababa"

Output:

Explanation:

We can swap the first 'b' with the last 'a', or the last 'b' with the first 'a'. Then, the longest repeated character substring is "aaa" with length 3.

Example 2:

Input:

```
text = "aaabaaa"
```

Output:

6

Explanation:

Swap 'b' with the last 'a' (or the first 'a'), and we get longest repeated character substring "aaaaaa" with length 6.

Example 3:

Input:

```
text = "aaaaa"
```

Output:

5

Explanation:

No need to swap, longest repeated character substring is "aaaaa" with length is 5.

Constraints:

$1 \leq \text{text.length} \leq 2 * 10$

text

consist of lowercase English characters only.

## Code Snippets

### C++:

```
class Solution {  
public:  
    int maxRepOpt1(string text) {  
  
    }  
};
```

### Java:

```
class Solution {  
public int maxRepOpt1(String text) {  
  
}  
}
```

### Python3:

```
class Solution:  
    def maxRepOpt1(self, text: str) -> int:
```

### Python:

```
class Solution(object):  
    def maxRepOpt1(self, text):  
        """  
        :type text: str  
        :rtype: int  
        """
```

### JavaScript:

```
/**  
 * @param {string} text
```

```
* @return {number}
*/
var maxRepOpt1 = function(text) {

};
```

### TypeScript:

```
function maxRepOpt1(text: string): number {

};
```

### C#:

```
public class Solution {
public int MaxRepOpt1(string text) {

}
```

### C:

```
int maxRepOpt1(char* text) {

}
```

### Go:

```
func maxRepOpt1(text string) int {

}
```

### Kotlin:

```
class Solution {
fun maxRepOpt1(text: String): Int {

}
```

### Swift:

```
class Solution {  
    func maxRepOpt1(_ text: String) -> Int {  
        }  
        }  
}
```

### Rust:

```
impl Solution {  
    pub fn max_rep_opt1(text: String) -> i32 {  
        }  
        }  
}
```

### Ruby:

```
# @param {String} text  
# @return {Integer}  
def max_rep_opt1(text)  
  
end
```

### PHP:

```
class Solution {  
  
    /**  
     * @param String $text  
     * @return Integer  
     */  
    function maxRepOpt1($text) {  
  
    }  
}
```

### Dart:

```
class Solution {  
    int maxRepOpt1(String text) {  
        }  
        }
```

### **Scala:**

```
object Solution {  
    def maxRepOpt1(text: String): Int = {  
  
    }  
}
```

### **Elixir:**

```
defmodule Solution do  
  @spec max_rep_opt1(text :: String.t) :: integer  
  def max_rep_opt1(text) do  
  
  end  
end
```

### **Erlang:**

```
-spec max_rep_opt1(Text :: unicode:unicode_binary()) -> integer().  
max_rep_opt1(Text) ->  
.
```

### **Racket:**

```
(define/contract (max-rep-opt1 text)  
  (-> string? exact-integer?)  
)
```

## **Solutions**

### **C++ Solution:**

```
/*  
 * Problem: Swap For Longest Repeated Character Substring  
 * Difficulty: Medium  
 * Tags: array, string, tree, hash  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(h) for recursion stack where h is height  
 */
```

```

class Solution {
public:
    int maxRepOpt1(string text) {
        }
    };

```

### Java Solution:

```

/**
 * Problem: Swap For Longest Repeated Character Substring
 * Difficulty: Medium
 * Tags: array, string, tree, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

class Solution {
public int maxRepOpt1(String text) {

}
}

```

### Python3 Solution:

```

"""
Problem: Swap For Longest Repeated Character Substring
Difficulty: Medium
Tags: array, string, tree, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
    def maxRepOpt1(self, text: str) -> int:
        # TODO: Implement optimized solution

```

```
pass
```

### Python Solution:

```
class Solution(object):
    def maxRepOpt1(self, text):
        """
        :type text: str
        :rtype: int
        """

```

### JavaScript Solution:

```
/**
 * Problem: Swap For Longest Repeated Character Substring
 * Difficulty: Medium
 * Tags: array, string, tree, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height
 */

/**
 * @param {string} text
 * @return {number}
 */
var maxRepOpt1 = function(text) {

};


```

### TypeScript Solution:

```
/**
 * Problem: Swap For Longest Repeated Character Substring
 * Difficulty: Medium
 * Tags: array, string, tree, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(h) for recursion stack where h is height

```

```
*/\n\nfunction maxRepOpt1(text: string): number {\n};
```

### C# Solution:

```
/*\n * Problem: Swap For Longest Repeated Character Substring\n * Difficulty: Medium\n * Tags: array, string, tree, hash\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(h) for recursion stack where h is height\n */\n\npublic class Solution {\n    public int MaxRepOpt1(string text) {\n\n    }\n}
```

### C Solution:

```
/*\n * Problem: Swap For Longest Repeated Character Substring\n * Difficulty: Medium\n * Tags: array, string, tree, hash\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(h) for recursion stack where h is height\n */\n\nint maxRepOpt1(char* text) {\n\n}
```

### Go Solution:

```

// Problem: Swap For Longest Repeated Character Substring
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func maxRepOpt1(text string) int {

}

```

### Kotlin Solution:

```

class Solution {
    fun maxRepOpt1(text: String): Int {
        return 0
    }
}

```

### Swift Solution:

```

class Solution {
    func maxRepOpt1(_ text: String) -> Int {
        return 0
    }
}

```

### Rust Solution:

```

// Problem: Swap For Longest Repeated Character Substring
// Difficulty: Medium
// Tags: array, string, tree, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
    pub fn max_rep_opt1(text: String) -> i32 {
        return 0
    }
}

```

```
}
```

### Ruby Solution:

```
# @param {String} text
# @return {Integer}
def max_rep_opt1(text)

end
```

### PHP Solution:

```
class Solution {

    /**
     * @param String $text
     * @return Integer
     */
    function maxRepOpt1($text) {

    }
}
```

### Dart Solution:

```
class Solution {
int maxRepOpt1(String text) {

}
```

### Scala Solution:

```
object Solution {
def maxRepOpt1(text: String): Int = {

}
```

### Elixir Solution:

```
defmodule Solution do
@spec max_rep_opt1(text :: String.t) :: integer
def max_rep_opt1(text) do

end
end
```

### Erlang Solution:

```
-spec max_rep_opt1(Text :: unicode:unicode_binary()) -> integer().
max_rep_opt1(Text) ->
.
```

### Racket Solution:

```
(define/contract (max-rep-opt1 text)
(-> string? exact-integer?))
```