Problem 424: Longest Repeating Character Replacement

Problem Information

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

Problem Description

You are given a string

s

and an integer

k

. You can choose any character of the string and change it to any other uppercase English character. You can perform this operation at most

k

times.

Return

the length of the longest substring containing the same letter you can get after performing the above operations

.

Example 1:

Input:

s = "ABAB", k = 2

Output:

4

Explanation:

Replace the two 'A's with two 'B's or vice versa.

Example 2:

Input:

s = "AABABBA", k = 1

Output:

4

Explanation:

Replace the one 'A' in the middle with 'B' and form "AABBBBA". The substring "BBBB" has the longest repeating letters, which is 4. There may exists other ways to achieve this answer too.

Constraints:

1 <= s.length <= 10

5

s

consists of only uppercase English letters.

0 <= k <= s.length

Code Snippets

C++:

```
class Solution {
public:
int characterReplacement(string s, int k) {

}
};
```

Java:

```
class Solution {
public int characterReplacement(String s, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

```
};
```

TypeScript:

```
function characterReplacement(s: string, k: number): number {

};
```

C#:

```
public class Solution {
public int CharacterReplacement(string s, int k) {

}
}
```

C:

```
int characterReplacement(char* s, int k) {

}
```

Go:

```
func characterReplacement(s string, k int) int {

}
```

Kotlin:

```
class Solution {
fun characterReplacement(s: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func characterReplacement(_ s: String, _ k: Int) -> Int {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn character_replacement(s: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def character_replacement(s, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function characterReplacement($s, $k) {

}
}
```

Dart:

```
class Solution {
int characterReplacement(String s, int k) {

}
}
```

Scala:

```
object Solution {
def characterReplacement(s: String, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec character_replacement(s :: String.t, k :: integer) :: integer
def character_replacement(s, k) do

end
end
```

Erlang:

```
-spec character_replacement(S :: unicode:unicode_binary(), K :: integer()) ->
integer().
character_replacement(S, K) ->
.
```

Racket:

```
(define/contract (character-replacement s k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Repeating Character Replacement
* 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 characterReplacement(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Repeating Character Replacement
* 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 characterReplacement(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Repeating Character Replacement
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 characterReplacement(self, s: str, k: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Repeating Character Replacement
* 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} s
* @param {number} k
* @return {number}
*/
var characterReplacement = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Repeating Character Replacement
* 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
*/

function characterReplacement(s: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Longest Repeating Character Replacement
* 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
*/

public class Solution {
public int CharacterReplacement(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: Longest Repeating Character Replacement
* 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
*/

int characterReplacement(char* s, int k) {

}
```

Go Solution:

```
// Problem: Longest Repeating Character Replacement
// 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 characterReplacement(s string, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun characterReplacement(s: String, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func characterReplacement(_ s: String, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Longest Repeating Character Replacement
// 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 character_replacement(s: String, k: i32) -> i32 {
```

```
}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {Integer}
def character_replacement(s, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return Integer
*/
function characterReplacement($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
int characterReplacement(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def characterReplacement(s: String, k: Int): Int = {

}
```

```
}
```

Elixir Solution:

```
defmodule Solution do
@spec character_replacement(s :: String.t, k :: integer) :: integer
def character_replacement(s, k) do

end
end
```

Erlang Solution:

```
-spec character_replacement(S :: unicode:unicode_binary(), K :: integer()) ->
integer().
character_replacement(S, K) ->
.
```

Racket Solution:

```
(define/contract (character-replacement s k)
(-> string? exact-integer? exact-integer?)
)
```