Problem 2024: Maximize the Confusion of an Exam

Problem Information

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

Problem Description

A teacher is writing a test with

n

true/false questions, with

'T'

denoting true and

'F'

denoting false. He wants to confuse the students by

maximizing

the number of

consecutive

questions with the

same

answer (multiple trues or multiple falses in a row).

You are given a string

answerKey

, where

answerKey[i]

is the original answer to the

i

th

question. In addition, you are given an integer

k

, the maximum number of times you may perform the following operation:

Change the answer key for any question to

'T'

or

'F'

(i.e., set

answerKey[i]

to

'T'

or

'F'

).

Return

the

maximum

number of consecutive

'T'

s or

'F'

s

in the answer key after performing the operation at most

k

times

.

Example 1:

Input:

answerKey = "TTFF", k = 2

Output:

4

Explanation:

We can replace both the 'F's with 'T's to make answerKey = "

TTTT

". There are four consecutive 'T's.

Example 2:

Input:

answerKey = "TFFT", k = 1

Output:

3

Explanation:

We can replace the first 'T' with an 'F' to make answerKey = "

FFF

T". Alternatively, we can replace the second 'T' with an 'F' to make answerKey = "T

FFF

". In both cases, there are three consecutive 'F's.

Example 3:

Input:

answerKey = "TTFTTFTT", k = 1

Output:

5

Explanation:

We can replace the first 'F' to make answerKey = "

TTTTT

FTT" Alternatively, we can replace the second 'F' to make answerKey = "TTF

TTTTT

". In both cases, there are five consecutive 'T's.

Constraints:

n == answerKey.length

1 <= n <= 5 * 10

4

answerKey[i]

is either

'T'

or

'F'

1 <= k <= n

Code Snippets

C++:

```
class Solution {
public:
```

```
int maxConsecutiveAnswers(string answerKey, int k) {

}
};
```

Java:

```
class Solution {
public int maxConsecutiveAnswers(String answerKey, int k) {

}
}
```

Python3:

```
class Solution:
def maxConsecutiveAnswers(self, answerKey: str, k: int) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maxConsecutiveAnswers(answerKey: string, k: number): number {

};
```

C#:

```
public class Solution {
public int MaxConsecutiveAnswers(string answerKey, int k) {

}
}
```

C:

```
int maxConsecutiveAnswers(char* answerKey, int k) {

}
```

Go:

```
func maxConsecutiveAnswers(answerKey string, k int) int {

}
```

Kotlin:

```
class Solution {
fun maxConsecutiveAnswers(answerKey: String, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func maxConsecutiveAnswers(_ answerKey: String, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_consecutive_answers(answer_key: String, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {String} answer_key
# @param {Integer} k
# @return {Integer}
def max_consecutive_answers(answer_key, k)

end
```

PHP:

```
class Solution {

/**
* @param String $answerKey
* @param Integer $k
* @return Integer
*/
function maxConsecutiveAnswers($answerKey, $k) {

}
}
```

Dart:

```
class Solution {
int maxConsecutiveAnswers(String answerKey, int k) {

}
}
```

Scala:

```
object Solution {
def maxConsecutiveAnswers(answerKey: String, k: Int): Int = {

}
```

```
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec max_consecutive_answers(AnswerKey :: unicode:unicode_binary(), K ::
integer()) -> integer().
max_consecutive_answers(AnswerKey, K) ->
.
```

Racket:

```
(define/contract (max-consecutive-answers answerKey k)
(-> string? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximize the Confusion of an Exam
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
```

```
public:
int maxConsecutiveAnswers(string answerKey, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Maximize the Confusion of an Exam
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int maxConsecutiveAnswers(String answerKey, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximize the Confusion of an Exam
Difficulty: Medium
Tags: array, string, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def maxConsecutiveAnswers(self, answerKey: str, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximize the Confusion of an Exam
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} answerKey
* @param {number} k
* @return {number}
*/
var maxConsecutiveAnswers = function(answerKey, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximize the Confusion of an Exam
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
function maxConsecutiveAnswers(answerKey: string, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximize the Confusion of an Exam
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int MaxConsecutiveAnswers(string answerKey, int k) {

}
}
```

C Solution:

```
/*
* Problem: Maximize the Confusion of an Exam
* Difficulty: Medium
* Tags: array, string, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int maxConsecutiveAnswers(char* answerKey, int k) {

}
```

Go Solution:

```
// Problem: Maximize the Confusion of an Exam
// Difficulty: Medium
// Tags: array, string, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func maxConsecutiveAnswers(answerKey string, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxConsecutiveAnswers(answerKey: String, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxConsecutiveAnswers(_ answerKey: String, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximize the Confusion of an Exam
// Difficulty: Medium
// Tags: array, string, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn max_consecutive_answers(answer_key: String, k: i32) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {String} answer_key
# @param {Integer} k
# @return {Integer}
def max_consecutive_answers(answer_key, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $answerKey
* @param Integer $k
* @return Integer
*/
function maxConsecutiveAnswers($answerKey, $k) {

}
}
```

Dart Solution:

```
class Solution {
int maxConsecutiveAnswers(String answerKey, int k) {

}
}
```

Scala Solution:

```
object Solution {
def maxConsecutiveAnswers(answerKey: String, k: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec max_consecutive_answers(AnswerKey :: unicode:unicode_binary(), K ::
integer()) -> integer().
max_consecutive_answers(AnswerKey, K) ->
.
```

Racket Solution:

```
(define/contract (max-consecutive-answers answerKey k)
(-> string? exact-integer? exact-integer?)
)
```