<https://leetcode.com/problems/maximize-the-confusion-of-an-exam/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 ith 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 = "FFFT".Alternatively, we can replace the second 'T' with an 'F' to make answerKey = "TFFF".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 = "TTTTTFTT"Alternatively, we can replace the second 'F' to make answerKey = "TTFTTTTT". 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

**Attempt 1: 2024-12-25**

**Solution 1: Not fixed length Sliding Window (10 min, this question is exactly same as** [L424.P2.6.Longest Repeating Character Replacement (Ref.L340)](note://1581773836A946879EB5237C85BFF6C2)**)**

**Exactly same as L424, but we only have two chars ('T', 'F'), so no need 'frequency' array, just need to handle with two variables and each time expanding or shrinking the Sliding Window just update both variables and always update the maximum repeat count, no matter its 'T' or 'F'**

class Solution {

    public int maxConsecutiveAnswers(String answerKey, int k) {

        int maxLen = 0;

        int maxRepeat = 0;

        int TCountInWindow = 0;

        int FCountInWindow = 0;

        char[] chars = answerKey.toCharArray();

        int i = 0;

        for(int j = 0; j < chars.length; j++) {

            if(chars[j] == 'T') {

                TCountInWindow++;

            } else {

                FCountInWindow++;

            }

            maxRepeat = Math.max(maxRepeat, Math.max(TCountInWindow, FCountInWindow));

            while(j - i + 1 - maxRepeat > k) {

                if(chars[i] == 'T') {

                    TCountInWindow--;

                } else {

                    FCountInWindow--;

                }

                i++;

            }

            maxLen = Math.max(maxLen, j - i + 1);

        }

        return maxLen;

    }

}

**Refer to L424, just rename the method**

class Solution {

    public int maxConsecutiveAnswers(String s, int k) {

        int maxLen = 0;

        // 'i' is left end pointer, 'j' is right end pointer

        int i = 0;

        int len = s.length();

        // s consists of only uppercase English letters

        int[] freq = new int[26];

        int maxRepeat = 0;

        for(int j = 0; j < len; j++) {

            char c = s.charAt(j);

            freq[c - 'A']++;

            maxRepeat = Math.max(maxRepeat, freq[c - 'A']);

            // When (length of substring - max frequency char in substring) > k

            // we start to shrink the left end

            if(j - i + 1 - maxRepeat > k) {

                char c1 = s.charAt(i);

                freq[c1 - 'A']--;

                // Below 'maxRepeat' update is not necessary, because when

                // the sliding window shrinks, the frequency counts array

                // won't get larger. So basically 'maxRepeat' never be updated

                // in this loop. The only potential update for 'maxRepeat' is

                // when substring expand on right end

                //for(int m = 0; m < 26; m++) {

                //    maxRepeat = Math.max(maxRepeat, freq[c1 - 'A']);

                //}

                i++;

            }

            maxLen = Math.max(maxLen, j - i + 1);

        }

        return maxLen;

    }

}

**Refer to**

[L424.P2.6.Longest Repeating Character Replacement (Ref.L340)](note://1581773836A946879EB5237C85BFF6C2)

[L1482.Minimum Number of Days to Make m Bouquets (Ref.L410,L1011)](note://WEBd5ce394160da45a79e92dbbc806111ec)

[L2379.Minimum Recolors to Get K Consecutive Black Blocks](note://WEB8ae9f88200296e35c40df40d278096aa)

[L2401.Longest Nice Subarray](note://WEB7538a2bbd5f8e4e2b3e803ba68408e0d)