<https://leetcode.com/problems/get-equal-substrings-within-budget/description/>

You are given two strings s and t of the same length and an integer maxCost.

You want to change s to t. Changing the ith character of s to ith character of t costs |s[i] - t[i]| (i.e., the absolute difference between the ASCII values of the characters).

Return the maximum length of a substring of s that can be changed to be the same as the corresponding substring of t with a cost less than or equal to maxCost. If there is no substring from s that can be changed to its corresponding substring from t, return 0.

**Example 1:**

**Input:** s = "abcd", t = "bcdf", maxCost = 3

**Output:** 3

**Explanation:** "abc" of s can change to "bcd".That costs 3, so the maximum length is 3.

**Example 2:**

**Input:** s = "abcd", t = "cdef", maxCost = 3

**Output:** 1

**Explanation:** Each character in s costs 2 to change to character in t, so the maximum length is 1.

**Example 3:**

**Input:** s = "abcd", t = "acde", maxCost = 0

**Output:** 1

**Explanation:** You cannot make any change, so the maximum length is 1.

**Constraints:**

1 <= s.length <= 10^5

t.length == s.length

0 <= maxCost <= 10^6

s and t consist of only lowercase English letters.

**Attempt 1: 2024-12-29**

**Solution 1: Not fixed length Sliding Window (10 min)**

**Style 1: while loop**

class Solution {

    public int equalSubstring(String s, String t, int maxCost) {

        int maxLen = 0;

        int cost = 0;

        int i = 0;

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

            cost += Math.abs(s.charAt(j) - t.charAt(j));

            while(cost > maxCost) {

                cost -= Math.abs(s.charAt(i) - t.charAt(i));

                i++;

            }

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

        }

        return maxLen;

    }

}

Time Complexity: O(n)

Space Complexity: O(1)

**Style 2: if condition instead of while loop**

class Solution {

    public int equalSubstring(String s, String t, int maxCost) {

        int maxLen = 0;

        int cost = 0;

        int i = 0;

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

            cost += Math.abs(s.charAt(j) - t.charAt(j));

            if(cost > maxCost) {

                cost -= Math.abs(s.charAt(i) - t.charAt(i));

                i++;

            }

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

        }

        return maxLen;

    }

}

Time Complexity: O(n)

Space Complexity: O(1)

**Why we can use 'if' instead of 'while loop' ?**

**Refer to** [L1838.Frequency of the Most Frequent Element (Ref.L2968)](note://WEBe75023aadd133f6c8d91ff09f6ec8794) **Approach 2 explain**

**Refer to**

[L209.P2.2.Minimum Size Subarray Sum (Ref.L862)](note://195893F25BDC45F2B63535FC5D9A4B26)

[L1838.Frequency of the Most Frequent Element (Ref.L2968)](note://WEBe75023aadd133f6c8d91ff09f6ec8794)

[L2401.Longest Nice Subarray (Ref.L424,L2024)](note://WEB7538a2bbd5f8e4e2b3e803ba68408e0d)