Problem# 395: Longest Substring with At Least k Repeating Characters (Medium)

<https://leetcode.com/problems/longest-substring-with-at-least-k-repeating-characters/>

My Solution: Runtime beats 29.75%.

1. Let n be length of s. If k is 1, then return n.

2. If n is less than k, return 0.

3. Construct a dictionary called my\_dict with Counter applied to s.

4. Construct a badList with a list of key(s) whose corresponding value(s) < k.

5. If length of badList is 0, return n.

6. Define a function isValidSubstring which will take as argument a string and will return True if it is a valid string and False otherwise. For this, we construct a dictionary of frequencies using Counter. Then we make sure that all the values in the dictionary are at least k, i.e. every value in the dictionary >=k.

7. Initialize length and max\_len to 0.

8. Iterate through each element in the string and get a substring starting at i. There is a pointer j pointing to the end of the substring and this moves to i. Break if the length of the substring that we are considering is less than or equal to max\_len, since this will not affect max\_len.

If the substring is valid, then update max\_len to be the maximum of max\_len and length of the substring and break the inner loop with j since we do not want to consider a smaller substring. Then go to the outer loop with I to the next element.

9. Return max\_len.

---------------------------------------------------

from collections import Counter

class Solution:

def longestSubstring(self, s: str, k: int) -> int:

n = len(s)

if k == 1:

return n

if n < k:

return 0

my\_dict = Counter(s)

badList = [key for key, value in my\_dict.items() if value < k]

if len(badList) == 0:

return n

def isValidSubstring(substr):

n = len(substr)

if n < k:

return False

the\_dict = Counter(substr)

aBadList = [key for key, value in the\_dict.items() if value < k]

if len(aBadList) == 0:

return True

return False

max\_len = 0

length = 0

for i in range(n):

if s[i] not in badList:

j = n - 1

for j in range(n-1, i, -1):

if s[j] not in badList:

length = j - i + 1

if length <= max\_len:

break

if isValidSubstring(s[i:j+1]):

max\_len = max(max\_len, length)

return max\_len