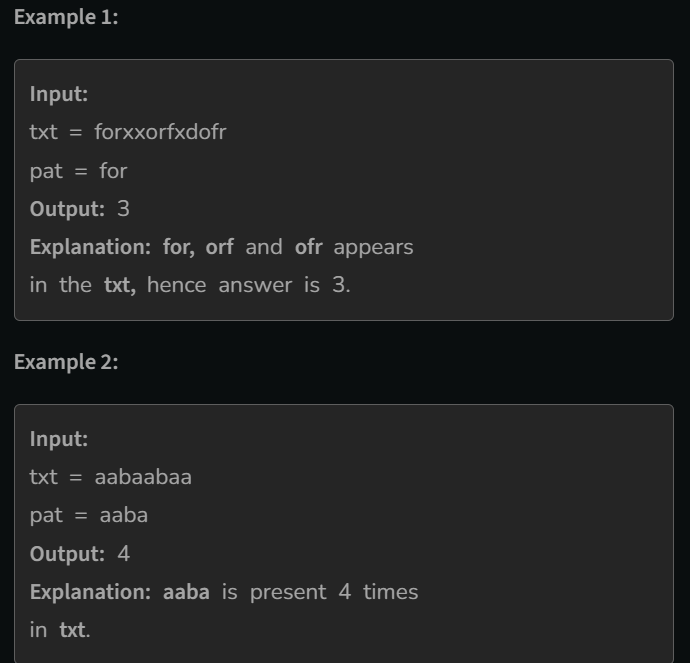
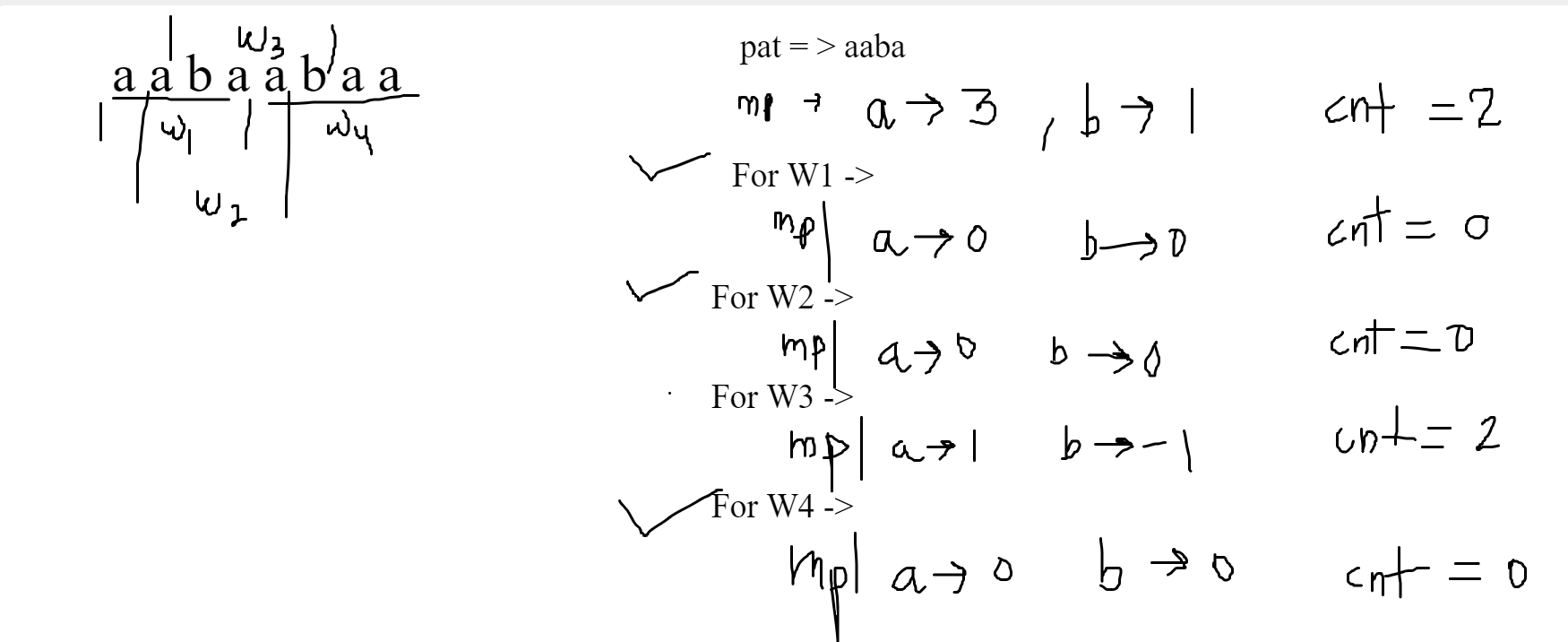
**Fixed Window size**

1. Maximum sum subarray of size K -> Fixed size ki window h, usme maximum sum kitna jaa skta h kisi bhi ek window ka,
   1. Approach -> ek window lelo, shuru left subtract krke, aur right add krke window maintain krni h
2. First Negative number in window of size K -> A black text on a white background

   Description automatically generated
   1. Approach -> Deque maintain krlo, aur hr baar left pe dhyaan rkhke deque m se nikaal do, aur right agar negative h to Deque m daal do
3. Count Occurrences of anagrams ->



Anagrams of abc => abc, acb, bca, bac, cab,cba

* 1. Approach 1-> Maintain two maps of size 26, first map will contain the character frequency in pattern, second map will maintain the character frequency for each window. Match the maps at each window TC – 26\*O(n)
  2. Approach 2-> Use a single map, store the frequency of characters in pattern in it, and count the unique characters in the pattern ex:- count = 2 for aaba. Now while creating the first window decrease the frequencies from the map of characters, as soon frequency of a character touches 0, decrease count, and if the frequency of a character moves away from 0 increases count. This way at every window you can tell if count = 0, this is an anagram. Note -> increase count as the freq of a character becomes 1 or -1. TC = O(n)

1. Maximum of all subarray of size K -> A white sheet with black text and numbers

   Description automatically generated



* 1. Approach -> Use a Deque, to keep the elements in decreasing order, socho ki agr right m hme koi bada element mil gya then it will last longer to hme picchle elements ki ab koi jrurat nahi

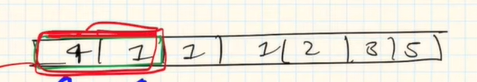
**Variable window size ->**

1. Largest subarray of sum K ->

Sab me same approach use krungaa, l=0, r=-1 se shuru krke, hrek l ke liye r ko possible rightmost le jaane ki koshish krna, bs dhyaan rahe, r ko le jaane se pehle check krungaa, aur l ko ek ek krke bdhaungaa

* 1. Variation 1 -> Only positive elements

Link-> [Code studio](https://www.codingninjas.com/studio/problems/longest-subarray-with-sum-k_6682399?utm_source=youtube&utm_medium=affiliate&utm_campaign=striver_Arrayproblems)

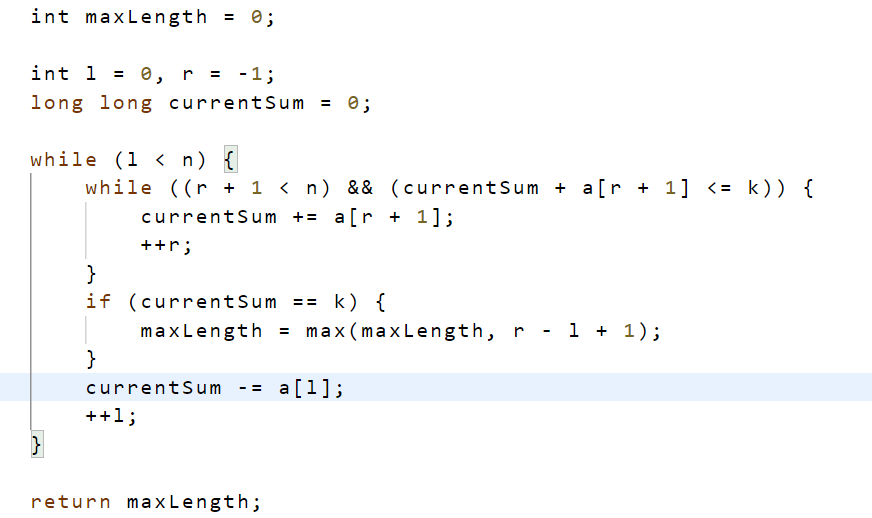
 K=5

Use simple logic, start at l=0, r=0,

If curr\_sum==k -> compare , ans=max(ans,r-l+1) and increase l (no point in increasing r further)

Else if curr\_sum<k -> increase r

Else if currsum>k -> increase l, while the same condition holds

****

Use this approach, right pe jaane se pehle check kro ki right pe jaane se nuksaan to nhi ho jaayega, mtlb hrek l ke liye hm kis possible right tk jaa skte h

* 1. Variation 2 -> positive and negative numbers

**Approach-** use priority sum approach (using maps) **Best** works on all the variations great

Using map store the last index for every priority sum

Ex -> 2 -1 3 5 2 -4 6 -2 1

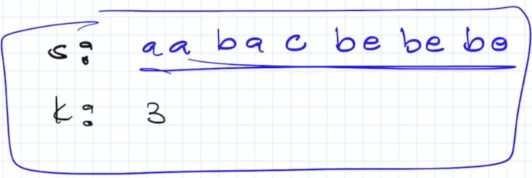
Priority sum 0 2 1 4 9 11 7 13 11 12

Let’s say k = 2, now in map(mp) store the last index of every possible priority sum

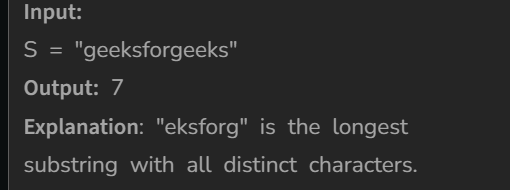
For each ps: priority\_sum , try to check if ps+k exist in mapand if it does, maintain your ans by ans = max(ans, mp[ps+k] – current\_index)

**Note-** also remember to check when priority sum is 0 initially <https://practice.geeksforgeeks.org/problems/longest-sub-array-with-sum-k0809/1>

* 1. Variation 3 -> <https://leetcode.com/problems/subarray-sum-equals-k/>

1. Longest Substring with exack K unique characters -> 

Approach - > store the frequencies in a map, and use a count to maintain the count of no of unique characters, if map hits 0 somewhere reduce count, if map increases from 0 to 1 increase the count. Now for every l, try to find out, upto where safest r we can go

1. Longest substring without repeating characters 

Approach -> store characters frequency in map, hrek l ke liye possible rightmost r pe jaane ki koshish kro, freq 2 nhi ho jaaye kisi bhi character ki

<https://www.geeksforgeeks.org/problems/longest-distinct-characters-in-string5848/1?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=bottom_sticky_on_article>

1. Minimum window substring - > given two strings s and t, find the minimum substring in s such that every character of t must present in that substring in the same frequency.

Ex-> s = “time to practice” , t = “toc”, ans = “to prac”

Use same format, count t in a different map, now traverse s making sure that (initially l=0, r=-1), for each l find the minimum r such that the condition is true. Find the answer this way

<https://leetcode.com/problems/minimum-window-substring/description/>