1 ----- Leetcode 1343 (Medium) : Number of Sub Arrays of Size K and Avg Greater than or Equal to Threshold

Given an array of integers arr and two integers k and threshold, return *the number of sub-arrays of size* k *and average greater than or equal to* threshold.

**Example 1:**

**Input:** arr = [2,2,2,2,5,5,5,8], k = 3, threshold = 4

**Output:** 3

**Explanation:** Sub-arrays [2,5,5],[5,5,5] and [5,5,8] have averages 4, 5 and 6 respectively. All other sub-arrays of size 3 have averages less than 4 (the threshold).

**Example 2:**

**Input:** arr = [11,13,17,23,29,31,7,5,2,3], k = 3, threshold = 5

**Output:** 6

**Explanation:** The first 6 sub-arrays of size 3 have averages greater than 5. Note that averages are not integers.

**Constraints:**

* 1 <= arr.length <= 105
* 1 <= arr[i] <= 104
* 1 <= k <= arr.length
* 0 <= threshold <= 104

2 ----- Leetcode 3 (Medium) : Longest Substring Without Repeating Characters

Given a string s, find the length of the **longest** **substring** without repeating characters.

**Example 1:**

**Input:** s = "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** s = "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Example 3:**

**Input:** s = "pwwkew"

**Output:** 3

**Explanation:** The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Constraints:**

* 0 <= s.length <= 5 \* 104
* s consists of English letters, digits, symbols and spaces.

3 ----- Leetcode 424 (Medium) : Longest Repeating Character Replacement

You are given a string s and an integer k. You can choose any character of the string and change it to any other uppercase English character. You can perform this operation at most k times.

Return *the length of the longest substring containing the same letter you can get after performing the above operations*.

**Example 1:**

**Input:** s = "ABAB", k = 2

**Output:** 4

**Explanation:** Replace the two 'A's with two 'B's or vice versa.

**Example 2:**

**Input:** s = "AABABBA", k = 1

**Output:** 4

**Explanation:** Replace the one 'A' in the middle with 'B' and form "AABBBBA".

The substring "BBBB" has the longest repeating letters, which is 4.

There may exists other ways to achieve this answer too.

**Constraints:**

* 1 <= s.length <= 105
* s consists of only uppercase English letters.
* 0 <= k <= s.length

4 ----- Leetcode 567 (Medium) : Permutation in String

Given two strings s1 and s2, return true *if* s2 *contains a permutation of* s1*, or* false *otherwise*.

In other words, return true if one of s1's permutations is the substring of s2.

**Example 1:**

**Input:** s1 = "ab", s2 = "eidbaooo"

**Output:** true

**Explanation:** s2 contains one permutation of s1 ("ba").

**Example 2:**

**Input:** s1 = "ab", s2 = "eidboaoo"

**Output:** false

**Constraints:**

* 1 <= s1.length, s2.length <= 104
* s1 and s2 consist of lowercase English letters.

5 ----- Leetcode 1838 (Medium) : Frequency of the Most Frequent Element

The **frequency** of an element is the number of times it occurs in an array.

You are given an integer array nums and an integer k. In one operation, you can choose an index of nums and increment the element at that index by 1.

Return *the* ***maximum possible frequency*** *of an element after performing* ***at most***k *operations*.

**Example 1:**

**Input:** nums = [1,2,4], k = 5

**Output:** 3

**Explanation:** Increment the first element three times and the second element two times to make nums = [4,4,4].

4 has a frequency of 3.

**Example 2:**

**Input:** nums = [1,4,8,13], k = 5

**Output:** 2

**Explanation:** There are multiple optimal solutions:

- Increment the first element three times to make nums = [4,4,8,13]. 4 has a frequency of 2.

- Increment the second element four times to make nums = [1,8,8,13]. 8 has a frequency of 2.

- Increment the third element five times to make nums = [1,4,13,13]. 13 has a frequency of 2.

**Example 3:**

**Input:** nums = [3,9,6], k = 2

**Output:** 1

**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i] <= 105
* 1 <= k <= 105

6 ----- Leetcode 904 (Medium) : Fruit Into Baskets

You are visiting a farm that has a single row of fruit trees arranged from left to right. The trees are represented by an integer array fruits where fruits[i] is the **type** of fruit the ith tree produces.

You want to collect as much fruit as possible. However, the owner has some strict rules that you must follow:

* You only have **two** baskets, and each basket can only hold a **single type** of fruit. There is no limit on the amount of fruit each basket can hold.
* Starting from any tree of your choice, you must pick **exactly one fruit** from **every** tree (including the start tree) while moving to the right. The picked fruits must fit in one of your baskets.
* Once you reach a tree with fruit that cannot fit in your baskets, you must stop.

Given the integer array fruits, return *the* ***maximum*** *number of fruits you can pick*.

**Example 1:**

**Input:** fruits = [1,2,1]

**Output:** 3

**Explanation:** We can pick from all 3 trees.

**Example 2:**

**Input:** fruits = [0,1,2,2]

**Output:** 3

**Explanation:** We can pick from trees [1,2,2].

If we had started at the first tree, we would only pick from trees [0,1].

**Example 3:**

**Input:** fruits = [1,2,3,2,2]

**Output:** 4

**Explanation:** We can pick from trees [2,3,2,2].

If we had started at the first tree, we would only pick from trees [1,2].

**Constraints:**

* 1 <= fruits.length <= 105
* 0 <= fruits[i] < fruits.length

7 ----- Leetcode 1456: Maximum Number of Vowels in a Substring of Given Length

Given a string s and an integer k, return *the maximum number of vowel letters in any substring of* s *with length* k.

**Vowel letters** in English are 'a', 'e', 'i', 'o', and 'u'.

**Example 1:**

**Input:** s = "abciiidef", k = 3

**Output:** 3

**Explanation:** The substring "iii" contains 3 vowel letters.

**Example 2:**

**Input:** s = "aeiou", k = 2

**Output:** 2

**Explanation:** Any substring of length 2 contains 2 vowels.

**Example 3:**

**Input:** s = "leetcode", k = 3

**Output:** 2

**Explanation:** "lee", "eet" and "ode" contain 2 vowels.

**Constraints:**

* 1 <= s.length <= 105
* s consists of lowercase English letters.
* 1 <= k <= s.length

8 ----- Leetcode 1888 (Medium) : Minimum Number of Flips to Make The Binary String Alternating

You are given a binary string s. You are allowed to perform two types of operations on the string in any sequence:

* **Type-1: Remove** the character at the start of the string s and **append** it to the end of the string.
* **Type-2: Pick** any character in s and **flip** its value, i.e., if its value is '0' it becomes '1' and vice-versa.

Return *the* ***minimum*** *number of* ***type-2*** *operations you need to perform* *such that* s *becomes* ***alternating****.*

The string is called **alternating** if no two adjacent characters are equal.

* For example, the strings "010" and "1010" are alternating, while the string "0100" is not.

**Example 1:**

**Input:** s = "111000"

**Output:** 2

**Explanation**: Use the first operation two times to make s = "100011".

Then, use the second operation on the third and sixth elements to make s = "101010".

**Example 2:**

**Input:** s = "010"

**Output:** 0

**Explanation**: The string is already alternating.

**Example 3:**

**Input:** s = "1110"

**Output:** 1

**Explanation**: Use the second operation on the second element to make s = "1010".

**Constraints:**

* 1 <= s.length <= 105
* s[i] is either '0' or '1'

9 ----- Leetcode 209 (Medium): Minimum Size Subarray Sum

Given an array of positive integers nums and a positive integer target, return *the* ***minimal length*** *of a subarray* *whose sum is greater than or equal to* target. If there is no such subarray, return 0 instead.

**Example 1:**

**Input:** target = 7, nums = [2,3,1,2,4,3]

**Output:** 2

**Explanation:** The subarray [4,3] has the minimal length under the problem constraint.

**Example 2:**

**Input:** target = 4, nums = [1,4,4]

**Output:** 1

**Example 3:**

**Input:** target = 11, nums = [1,1,1,1,1,1,1,1]

**Output:** 0

**Constraints:**

* 1 <= target <= 109
* 1 <= nums.length <= 105
* 1 <= nums[i] <= 104

10 ----- Leetcode 658 (Medium) : Find K Closest Elements

Given a **sorted** integer array arr, two integers k and x, return the k closest integers to x in the array. The result should also be sorted in ascending order.

An integer a is closer to x than an integer b if:

* |a - x| < |b - x|, or
* |a - x| == |b - x| and a < b

**Example 1:**

**Input:** arr = [1,2,3,4,5], k = 4, x = 3

**Output:** [1,2,3,4]

**Example 2:**

**Input:** arr = [1,2,3,4,5], k = 4, x = -1

**Output:** [1,2,3,4]

**Constraints:**

* 1 <= k <= arr.length
* 1 <= arr.length <= 104
* arr is sorted in **ascending** order.
* -104 <= arr[i], x <= 104

11 ----- Leetcode 1658 (Medium) : Minimum Operations to Reduce X to Zero

You are given an integer array nums and an integer x. In one operation, you can either remove the leftmost or the rightmost element from the array nums and subtract its value from x. Note that this **modifies** the array for future operations.

Return *the* ***minimum number*** *of operations to reduce* x *to* ***exactly*** 0 *if it is possible, otherwise, return* -1.

**Example 1:**

**Input:** nums = [1,1,4,2,3], x = 5

**Output:** 2

**Explanation:** The optimal solution is to remove the last two elements to reduce x to zero.

**Example 2:**

**Input:** nums = [5,6,7,8,9], x = 4

**Output:** -1

**Example 3:**

**Input:** nums = [3,2,20,1,1,3], x = 10

**Output:** 5

**Explanation:** The optimal solution is to remove the last three elements and the first two elements (5 operations in total) to reduce x to zero.

**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i] <= 104
* 1 <= x <= 109

12 ----- Leetcode 1208 (Medium) : Get Equal Substrings Within Budget

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 <= 105
* t.length == s.length
* 0 <= maxCost <= 106
* s and t consist of only lowercase English letters.

13 ----- Leetcode 930 (Medium): Binary Subarrays with Sum

Given a binary array nums and an integer goal, return *the number of non-empty* ***subarrays*** *with a sum* goal.

A **subarray** is a contiguous part of the array.

**Example 1:**

**Input:** nums = [1,0,1,0,1], goal = 2

**Output:** 4

**Explanation:** The 4 subarrays are bolded and underlined below:

[**1,0,1**,0,1]

[**1,0,1,0**,1]

[1,**0,1,0,1**]

[1,0,**1,0,1**]

**Example 2:**

**Input:** nums = [0,0,0,0,0], goal = 0

**Output:** 15

**Constraints:**

* 1 <= nums.length <= 3 \* 104
* nums[i] is either 0 or 1.
* 0 <= goal <= nums.length

14 ---- Leetcode 713: Subarray Product Less Than K

Given an array of integers nums and an integer k, return *the number of contiguous subarrays where the product of all the elements in the subarray is strictly less than* k.

**Example 1:**

**Input:** nums = [10,5,2,6], k = 100

**Output:** 8

**Explanation:** The 8 subarrays that have product less than 100 are:

[10], [5], [2], [6], [10, 5], [5, 2], [2, 6], [5, 2, 6]

Note that [10, 5, 2] is not included as the product of 100 is not strictly less than k.

**Example 2:**

**Input:** nums = [1,2,3], k = 0

**Output:** 0

**Constraints:**

* 1 <= nums.length <= 3 \* 104
* 1 <= nums[i] <= 1000
* 0 <= k <= 106

15 ----- Leetcode 2958 (Medium) : Length of Longest Subarray With at Most K Frequency

You are given an integer array nums and an integer k.The **frequency** of an element x is the number of times it occurs in an array.

An array is called **good** if the frequency of each element in this array is **less than or equal** to k.

Return *the length of the* ***longest******good*** *subarray of* nums*.*

A **subarray** is a contiguous non-empty sequence of elements within an array.

**Example 1:**

**Input:** nums = [1,2,3,1,2,3,1,2], k = 2

**Output:** 6

**Explanation:** The longest possible good subarray is [1,2,3,1,2,3] since the values 1, 2, and 3 occur at most twice in this subarray. Note that the subarrays [2,3,1,2,3,1] and [3,1,2,3,1,2] are also good.

It can be shown that there are no good subarrays with length more than 6.

**Example 2:**

**Input:** nums = [1,2,1,2,1,2,1,2], k = 1

**Output:** 2

**Explanation:** The longest possible good subarray is [1,2] since the values 1 and 2 occur at most once in this subarray. Note that the subarray [2,1] is also good.

It can be shown that there are no good subarrays with length more than 2.

**Example 3:**

**Input:** nums = [5,5,5,5,5,5,5], k = 4

**Output:** 4

**Explanation:** The longest possible good subarray is [5,5,5,5] since the value 5 occurs 4 times in this subarray.

It can be shown that there are no good subarrays with length more than 4.

**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i] <= 109
* 1 <= k <= nums.length

16 ----- Leetcode 2962 (Medium): Count Subarrays Where Max Element Appears at Least K Times

You are given an integer array nums and a **positive** integer k.

Return *the number of subarrays where the* ***maximum*** *element of* nums *appears* ***at least***k *times in that subarray.*

A **subarray** is a contiguous sequence of elements within an array.

**Example 1:**

**Input:** nums = [1,3,2,3,3], k = 2

**Output:** 6

**Explanation:** The subarrays that contain the element 3 at least 2 times are: [1,3,2,3], [1,3,2,3,3], [3,2,3], [3,2,3,3], [2,3,3] and [3,3].

**Example 2:**

**Input:** nums = [1,4,2,1], k = 3

**Output:** 0

**Explanation:** No subarray contains the element 4 at least 3 times.

**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i] <= 106
* 1 <= k <= 105

17 ----- Leetcode 76 (Hard) : Minimum Window Substring

Given two strings s and t of lengths m and n respectively, return *the* ***minimum window***

***substring***

*of* s *such that every character in* t *(****including duplicates****) is included in the window*. If there is no such substring, return *the empty string* "".

The testcases will be generated such that the answer is **unique**.

**Example 1:**

**Input:** s = "ADOBECODEBANC", t = "ABC"

**Output:** "BANC"

**Explanation:** The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.

**Example 2:**

**Input:** s = "a", t = "a"

**Output:** "a"

**Explanation:** The entire string s is the minimum window.

**Example 3:**

**Input:** s = "a", t = "aa"

**Output:** ""

**Explanation:** Both 'a's from t must be included in the window.

Since the largest window of s only has one 'a', return empty string.

**Constraints:**

* m == s.length
* n == t.length
* 1 <= m, n <= 105
* s and t consist of uppercase and lowercase English letters.

18 ----- Leetcode 239 (Hard) : Sliding Window Maximum

You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return *the max sliding window*.

**Example 1:**

**Input:** nums = [1,3,-1,-3,5,3,6,7], k = 3

**Output:** [3,3,5,5,6,7]

**Explanation:**

Window position Max

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

[1 3 -1] -3 5 3 6 7 **3**

1 [3 -1 -3] 5 3 6 7 **3**

1 3 [-1 -3 5] 3 6 7  **5**

1 3 -1 [-3 5 3] 6 7 **5**

1 3 -1 -3 [5 3 6] 7 **6**

1 3 -1 -3 5 [3 6 7] **7**

**Example 2:**

**Input:** nums = [1], k = 1

**Output:** [1]

**Constraints:**

* 1 <= nums.length <= 105
* -104 <= nums[i] <= 104
* 1 <= k <= nums.length

19 ----- GFG (Medium) : First Negative Number in every window of Size K

Given an array **A[]** of size **N** and a positive integer **K**, find the first negative integer for each and every window(contiguous subarray) of size **K**.

**Example 1:**

**Input :**

N = 5

A[] = {-8, 2, 3, -6, 10}

K = 2

**Output :**

-8 0 -6 -6

**Explanation :**

First negative integer for each window of size k

**{-8, 2}** = -8

**{2, 3}** = 0 (does not contain a negative integer)

**{3, -6}** = -6

**{-6, 10}** = -6

**Example 2:**

**Input :**

N = 8

A[] = {12, -1, -7, 8, -15, 30, 16, 28}

K = 3

**Output :**

-1 -1 -7 -15 -15 0

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **printFirstNegativeInteger()** which takes the array **A[]**, its size **N** and an integer **K** as inputs and returns the first negative number in every window of size K starting from the first till the end. If a window does not contain a negative integer , then return 0 for that window.

**Expected Time Complexity:** O(N)  
**Expected Auxiliary Space:** O(K)  
  
**Constraints:**  
1 <= N <= 105  
-105 <= A[i] <= 105  
1 <= K <= N

20 ----- GFG (Medium) : Count Occurrences of Anagrams

Given a word **pat** and a text **txt**. Return the count of the occurrences of anagrams of the word in the text.

**Example 1:**

**Input:**

txt = forxxorfxdofr

pat = for

**Output:** 3

**Explanation:** **for, orf** and **ofr** appears in the **txt,** hence answer is 3.

**Example 2:**

**Input:**

txt = aabaabaa

pat = aaba

**Output:** 4

**Explanation:** **aaba** is present 4 times in **txt**.

**Your Task:**  
Complete the function **search()** which takes two strings **pat, txt,** as input parameters and returns an integer denoting the answer.   
You don't need to print answer or take inputs.

**Expected Time Complexity:** O(N)  
**Expected Auxiliary Space:** O(26) or O(256)

**Constraints:**  
1 <= |pat| <= |txt| <= 105  
Both strings contain lowercase English letters

21 ----- GFG (Medium) : Longest Subarray With Sum K

Given an array **arr** containing **n** integers and an integer **k**. Your task is to find the length of the longest Sub-Array with the sum of the elements equal to the given value **k**.

**Examples:**

**Input :** n = 6,arr[] = {10, 5, 2, 7, 1, 9}, k = 15

**Output :** 4

**Explanation:** The sub-array is **{5, 2, 7, 1}**.

**Input :** n= 3, arr[] = {-1, 2, 3}, k = 6

**Output :** 0

**Explanation:** There is no such sub-array with sum 6.

**Expected Time Complexity:** O(n).  
**Expected Auxiliary Space:** O(n).

**Constraints:**  
1<=n<=105  
-105<=arr[i], K<=105

22 ----- Leetcode 560 (Medium) : Subarray Sum Equals K

Given an array of integers nums and an integer k, return *the total number of subarrays whose sum equals to* k.

A subarray is a contiguous **non-empty** sequence of elements within an array.

**Example 1:**

**Input:** nums = [1,1,1], k = 2

**Output:** 2

**Example 2:**

**Input:** nums = [1,2,3], k = 3

**Output:** 2

**Constraints:**

* 1 <= nums.length <= 2 \* 104
* -1000 <= nums[i] <= 1000
* -107 <= k <= 107

23 ----- GFG (Medium) : Longest Substring with K unique characters

Given a string you need to print the size of the longest possible substring that has exactly **K unique** characters. If there is no possible substring then print -1

**Example 1:**

**Input:**

S = "aabacbebebe", K = 3

**Output:**   
7

**Explanation**:   
"cbebebe" is the longest substring with 3 distinct characters.

**Example 2:**

**Input**:

S = "aaaa", K = 2

**Output:** -1

**Explanation**:   
There's no substring with 2 distinct characters.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **longestKSubstr()**which takes the string S and an integer K as input and returns the length of the longest substring with exactly K distinct characters. If there is no substring with exactly K distinct characters then return -1.

**Expected Time Complexity:** O(|S|).  
**Expected Auxiliary Space:**O(|S|).

**Constraints:**  
1 ≤ |S| ≤ 105  
1 ≤ K ≤ 26  
All characters are lowercase latin characters

24 ----- Leetcode 904 (Medium) : Fruits into Baskets

You are visiting a farm that has a single row of fruit trees arranged from left to right. The trees are represented by an integer array fruits where fruits[i] is the **type** of fruit the ith tree produces.

You want to collect as much fruit as possible. However, the owner has some strict rules that you must follow:

* You only have **two** baskets, and each basket can only hold a **single type** of fruit. There is no limit on the amount of fruit each basket can hold.
* Starting from any tree of your choice, you must pick **exactly one fruit** from **every** tree (including the start tree) while moving to the right. The picked fruits must fit in one of your baskets.
* Once you reach a tree with fruit that cannot fit in your baskets, you must stop.

Given the integer array fruits, return *the* ***maximum*** *number of fruits you can pick*

**Example 1:**

**Input:** fruits = [1,2,1]

**Output:** 3

**Explanation:** We can pick from all 3 trees.

**Example 2:**

**Input:** fruits = [0,1,2,2]

**Output:** 3

**Explanation:** We can pick from trees [1,2,2].

If we had started at the first tree, we would only pick from trees [0,1].

**Example 3:**

**Input:** fruits = [1,2,3,2,2]

**Output:** 4

**Explanation:** We can pick from trees [2,3,2,2].

If we had started at the first tree, we would only pick from trees [1,2].

**Constraints:**

* 1 <= fruits.length <= 105
* 0 <= fruits[i] < fruits.length