**ARRAYS PROBLEMS:**

**Question: Rearrange Array Element by sign:**

You are given a **0-indexed** integer array nums of **even** length consisting of an **equal** number of positive and negative integers.

You should **rearrange** the elements of nums such that the modified array follows the given conditions:

1. Every **consecutive pair** of integers have **opposite signs**.
2. For all integers with the same sign, the **order** in which they were present in nums is **preserved**.
3. The rearranged array begins with a positive integer.

Return *the modified array after rearranging the elements to satisfy the aforementioned conditions*.

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

**Output:** [3,-2,1,-5,2,-4]

**Explanation:**

The positive integers in nums are [3,1,2]. The negative integers are [-2,-5,-4].

The only possible way to rearrange them such that they satisfy all conditions is [3,-2,1,-5,2,-4].

Other ways such as [1,-2,2,-5,3,-4], [3,1,2,-2,-5,-4], [-2,3,-5,1,-4,2] are incorrect because they do not satisfy one or more conditions.

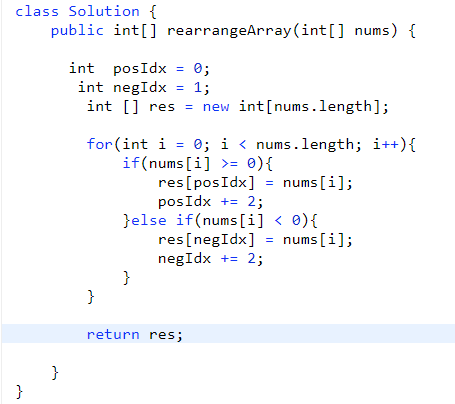
Solutions

One more solution is two make two

ArrayList<Integer> pos = new ArrayList<>();

ArrayList<> negative = new ArrayList<>();

And add the value accordingly by there signatures



**Questions : Find more than one missing numbers from sorted array:**

**Solutions:**

**Question : Longest Consecutive Sequence**

Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.

You must write an algorithm that runs in O(n) time.

**Example 1:**

**Input:** nums = [100,4,200,1,3,2]

**Output:** 4

**Explanation:** The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

**Example 2:**

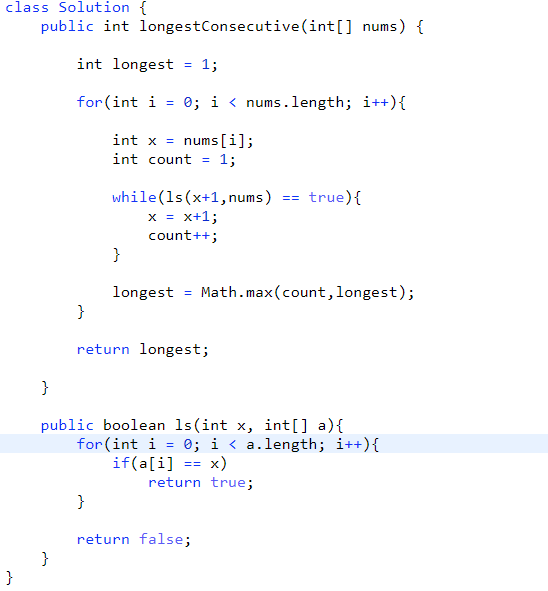
**Input:** nums = [0,3,7,2,5,8,4,6,0,1]

**Output:** 9

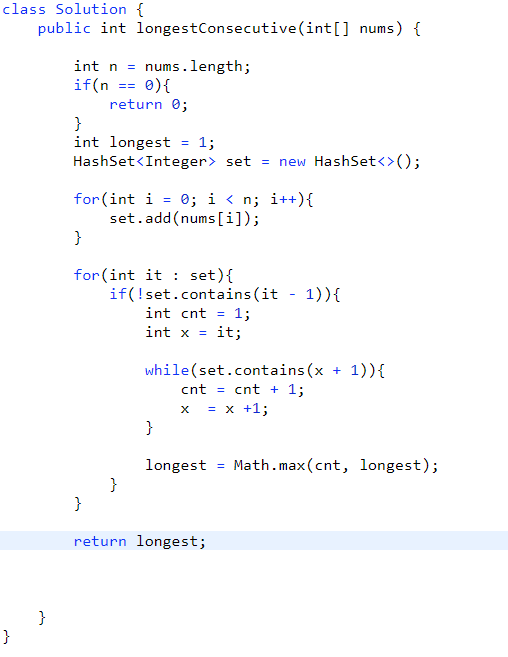
**Solutions:**

Brute – force approach take element of the arr[i] and check for the arr[i]+1 into the array if yes then increase the counter and arr[i]+1 as well and at last give the max len of the array !

Give solutions TC is = O(n^2)



**#Optimal Solution:**

****

Question: Sort 0’s 1’s 2’s in java

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

**Output:** [0,0,1,1,2,2]

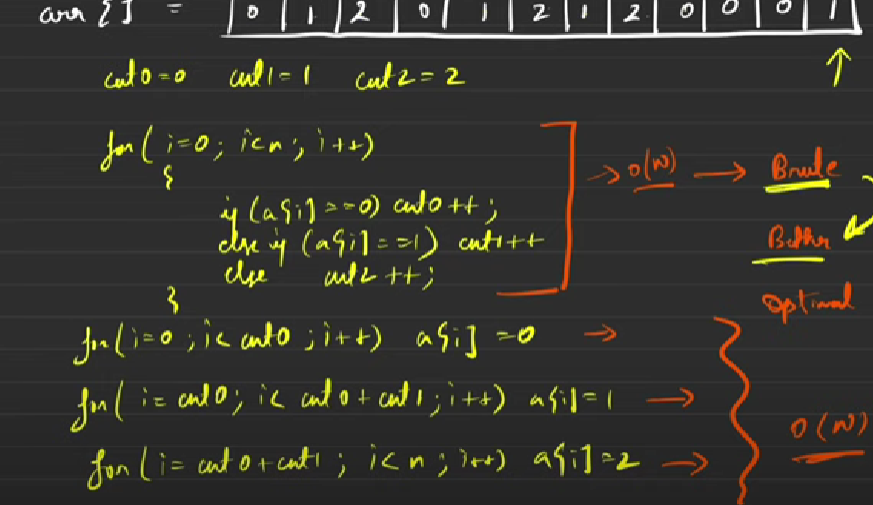
Solution:

Brute Force : merge sort which is in O(nlogn);

Arrays.sort(nums);

**Better** Solutions :

I know this in the give arrays we have 3 number only then we can hold the count of every number in the array.

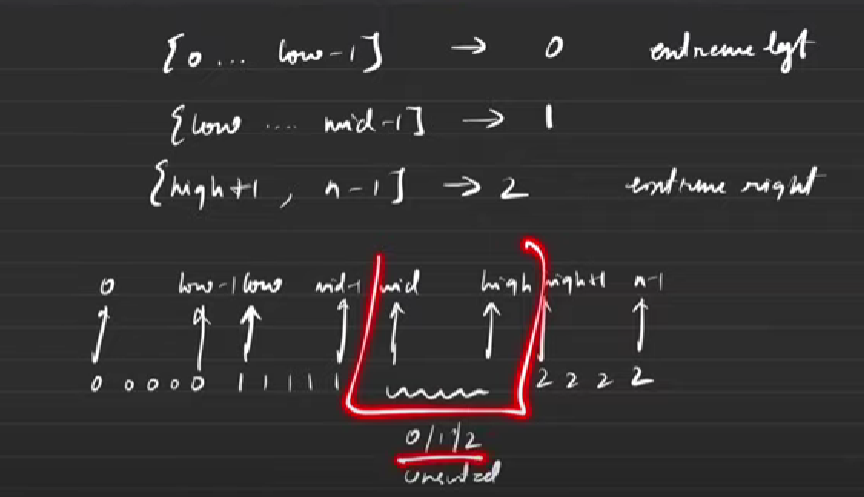


Overall time complexity is of O(2N).

But here we are taking more than one pass of iteration in java.

**Optimal Solutions:**

**Deustch National flag Algorithm to solve this problem:**

****

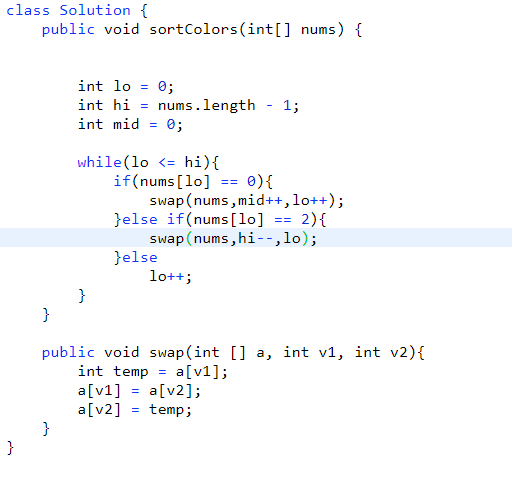
These are the three rules on which we will store the array of element

From 0 to low – 1 => 0

Form low to mid -1 => 1

From mid to high – 1 => {}anything

From high to n – 1 => 2



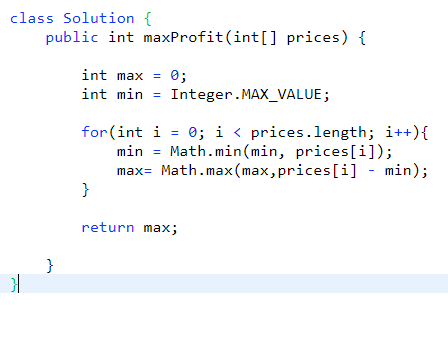
Question: Best Time to Buy and Sell Stock:

**Input:** prices = [7,1,5,3,6,4]

**Output:** 5

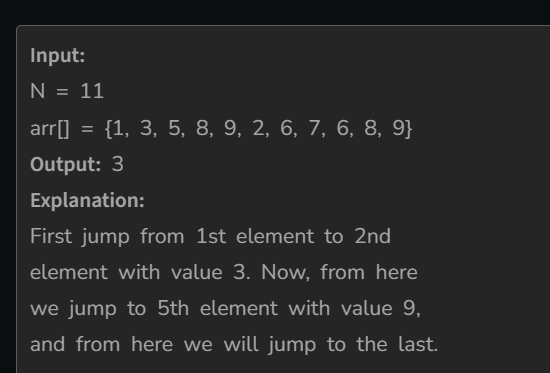
**Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell



**Question:** Minimum Number Of Jumps to reach end of the arrays

Given an array of **N** integers **arr[]** where each element represents the **maximum** length of the jump that can be made forward from that element. This means if arr[i] = x, then we can jump any distance y such that y ≤ x.  
Find the minimum number of jumps to reach the end of the array (starting from the first element). If an element is **0**, then you cannot move through that element.  
  
**Note:**Return -1 if you can't reach the end of the array.



Solutions :

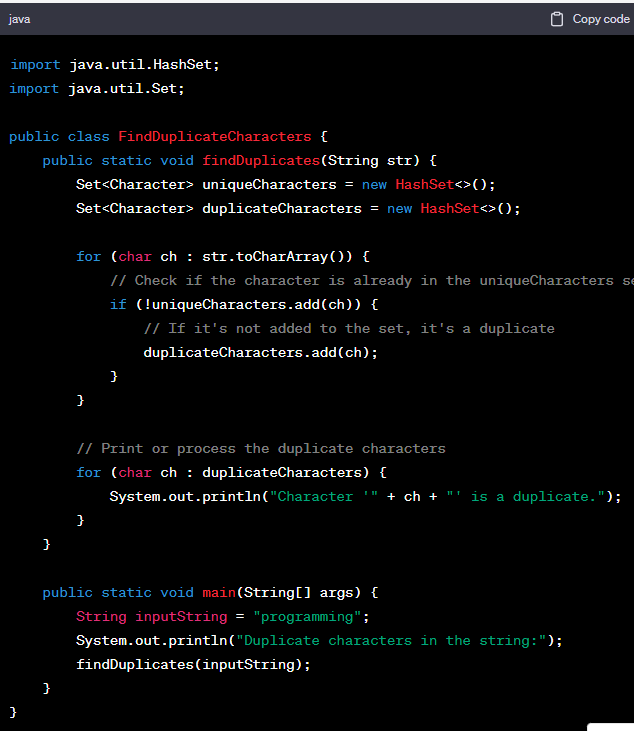
We will check the value of arr[i] to ask the array how much we can jump from this point where we are standing and update the jump as well

# ***STRING PROBLEMS***

**Q1. Find Duplicate characters in a string**

**#Brute Force**

Take hash map and store the character occurrence of the strings



* We use a single integer **checker** as a bit vector to represent the presence of characters in the input string.
* We iterate through the characters in the input string, and for each character, we calculate its bit index by subtracting 'a' from it. This ensures that each lowercase letter corresponds to a unique bit position in the integer.
* We use bitwise operations to check if the bit at the calculated index is already set in the **checker**. If it is set, the character is a duplicate; otherwise, we set the bit to indicate that we've seen the character.

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

**Question : Find Longest common prefix in String**

**Example 1:**

**Input:** strs = ["flower","flow","flight"]

**Output:** "fl"

**Example 2:**

**Input:** strs = ["dog","racecar","car"]

**Output:** ""

**Explanation:** There is no common prefix among the input strings.

****

**Question : Isomorphic String Java**

Given two strings s and t, *determine if they are isomorphic*.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

**Example 1:**

**Input:** s = "egg", t = "add"

**Output:** true

**Example 2:**

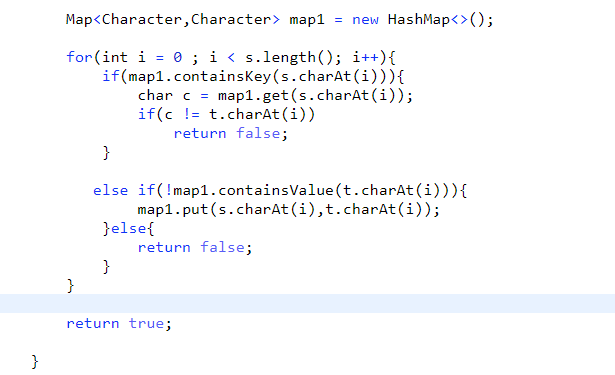
**Input:** s = "foo", t = "bar"

**Output:** false

**Solutions :**

**Make a map and store all the character of first and second string and check**

* **Key exist in map yes then check for the value s2.charAt(i) == map.get(s1.charAt(i));**
* **If not then check weather that value exist or not if not then store it into the map**
* **If value exist then it’s map to another key then return false;**



Question : Rotate String

Given two strings s and goal, return true *if and only if* s *can become* goal *after some number of****shifts****on* s.

A **shift** on s consists of moving the leftmost character of s to the rightmost position.

* For example, if s = "abcde", then it will be "bcdea" after one shift.

**Example 1:**

**Input:** s = "abcde", goal = "cdeab"

**Output:** true

**Example 2:**

**Input:** s = "abcde", goal = "abced"

**Output:** false

Question : Check if two string are anagram or not !

Given two strings s and t, return true *if* t *is an anagram of* s*, and* false *otherwise*.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

**Example 1:**

**Input:** s = "anagram", t = "nagaram"

**Output:** true

**Example 2:**

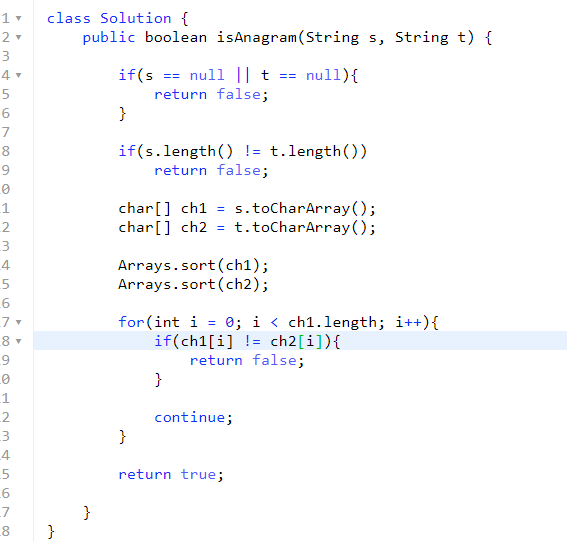
**Input:** s = "rat", t = "car"

**Output:** false

#**BruteForce :**

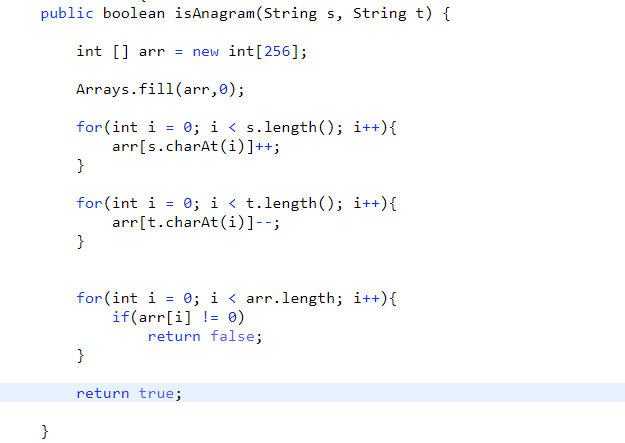
* *Sort the array*
* *Check each character value*
* *If all the character equal then return true or else return false.*

*#Time Complexity will be => O(n) + O(nlogn);*

**

*#Optimal Approach to find ANAGRAM :*

* Create count arrays of size 256 for both strings. Initialize all values in count arrays as 0.
* Iterate through every character of both strings and increment the count of characters in the corresponding count arrays.
* Compare count arrays. If both count arrays are the same, then return **true**else return **false**.

**

*Question : Largest Odd number in String*

You are given a string num, representing a large integer. Return *the****largest-valued odd****integer (as a string) that is a****non-empty substring****of*num*, or an empty string*""*if no odd integer exists*.

A **substring** is a contiguous sequence of characters within a string.

**Example 1:**

**Input:** num = "52"

**Output:** "5"

**Explanation:** The only non-empty substrings are "5", "2", and "52". "5" is the only odd number.

**Example 2:**

**Input:** num = "4206"

**Output:** ""

**Explanation:** There are no odd numbers in "4206".

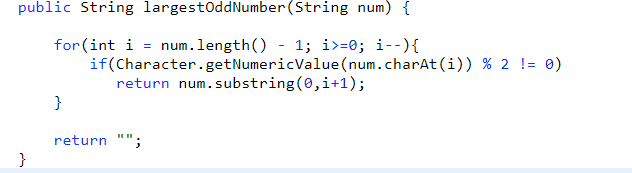
*Code :*

* *Iterate the loop from last to check for odd number get the number from string*

*And check for Character.getNumeric(s.charAt(i)) %2 != 0*

*Then return the SubString of s.substring(0,i+1);*

*What so ever the idx value is*

**

***Questions : Remove Outer Most Parenthesis in java***

A valid parentheses string is either empty "", "(" + A + ")", or A + B, where A and B are valid parentheses strings, and + represents string concatenation.

* For example, "", "()", "(())()", and "(()(()))" are all valid parentheses strings.

A valid parentheses string s is primitive if it is nonempty, and there does not exist a way to split it into s = A + B, with A and B nonempty valid parentheses strings.

Given a valid parentheses string s, consider its primitive decomposition: s = P1 + P2 + ... + Pk, where Pi are primitive valid parentheses strings.

Return s *after removing the outermost parentheses of every primitive string in the primitive decomposition of*s.

**Example 1:**

**Input:** s = "(()())(())"

**Output:** "()()()"

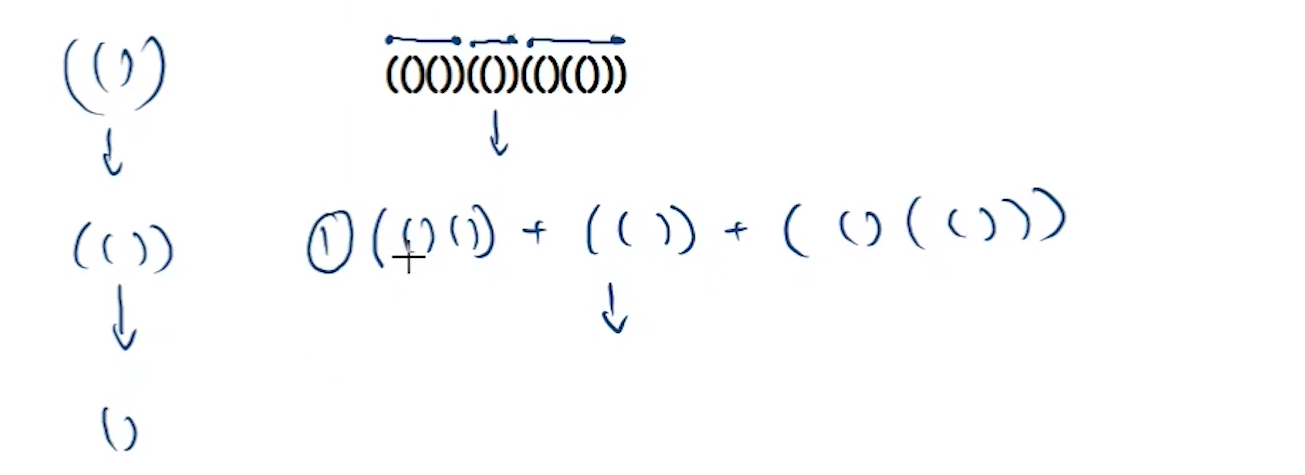
**Explanation:**

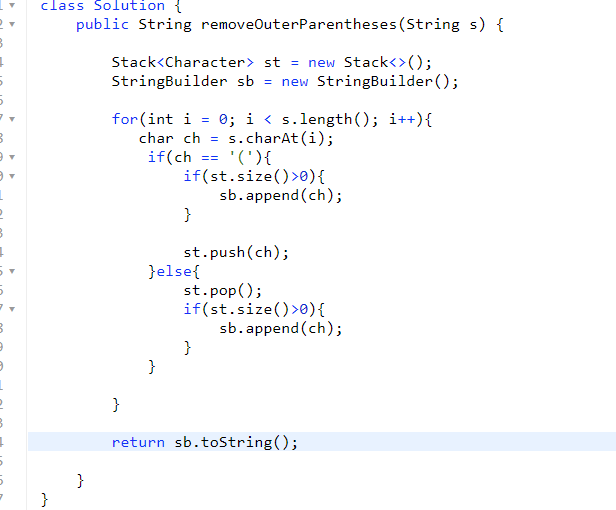
The input string is "(()())(())", with primitive decomposition "(()())" + "(())".

After removing outer parentheses of each part, this is "()()" + "()" = "()()()".

***Solutions :***

***S -> (()())() parenthesis will be balanced Split the string then remove the outer para.***

**

**

*Algorithm to solve this problem is be like :*

1. *Check for the stack if the size of the stack is > 0 then it’s valid answer if not the add the char in stack.*
2. *After this we will pop the character from stack and check the size of the ST weather it’s > 0 then it’s the valid answer append the string*
3. *If after popping the stack it will become empty then we won’t going to add the char in string and return the string as empty stack represent that the parenthesis is outer most parenthesis!!*

**451. Sort Characters By Frequency**

Medium

7269242Add to ListShare

Given a string s, sort it in **decreasing order** based on the **frequency** of the characters. The **frequency** of a character is the number of times it appears in the string.

Return *the sorted string*. If there are multiple answers, return *any of them*.

**Example 1:**

**Input:** s = "tree"

**Output:** "eert"

**Explanation:** 'e' appears twice while 'r' and 't' both appear once.

So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

**Example 2:**

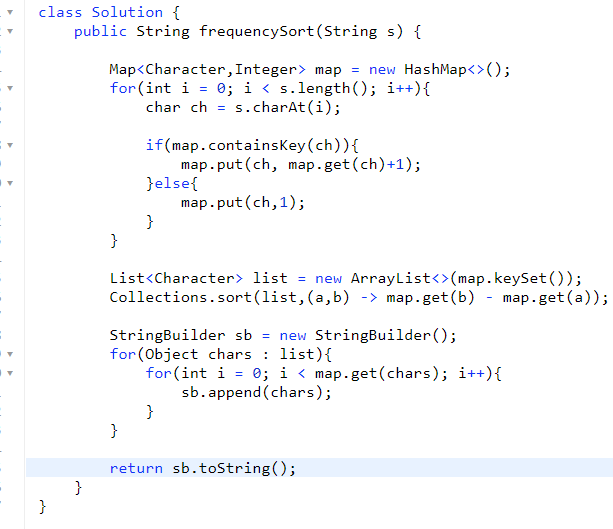
**Input:** s = "cccaaa"

**Output:** "aaaccc"

**Explanation:** Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.

Note that "cacaca" is incorrect, as the same characters must be together.

*Solution :  
Here we need to sort the character based on the frequency*

**

*Question Reverse A words in a String*

**Reverse Words in a String**

**Problem Statement:** Given a string s, reverse the words of the string.

**Examples:**

**Example 1:**

**Input:** s=”this is an amazing program”

**Output:** “program amazing an is this”

**Example 2:**

**Input:** s=”This is decent”

**Output:** “decent is This”

**Solutions: Brute Force**

**Approach**

* Use a stack to push all the words in a stack
* Now, all the words of the string are present in the stack, but in reverse order
* Pop elements of the stack one by one and add them to our answer variable. Remember to add a space between the words as well.
* Here’s a quick demonstration of the same

**Question:** Longest Palindrome in a string

Given a string s, return *the longest* *palindromic* *substring* in s.

**Input:** s = "babad"

**Output:** "bab"

**Explanation:** "aba" is also a valid answer.

**Example 2:**

**Input:** s = "cbbd"

**Output:** "bb"

**Question : Count and Say in String**

### Look and Say Pattern

Given an integer n. Return the nth row of the following look-and-say pattern.  
1  
11  
21  
1211  
111221  
**Look-and-Say Pattern:**

**Solutions** :

N = 1 => 1

N = 2 => (Check the previous value count and that particular number )

11

N = 3 => 21(for n=4 2 comes one time and number is 2 and one come one’s time and number is one

N = 4 => 1211

N=5 => 111221

N=6 => 312211

Approach to solve the problem:



**Question: Longest Substring palindrome in string**

Given a string s, return the longest palindromic substring in s.

**Example 1:**

**Input:** s = "babad"

**Output:** "bab"

**Explanation:** "aba" is also a valid answer.

**Example 2:**

**Input:** s = "cbbd"

**Output:** "bb"

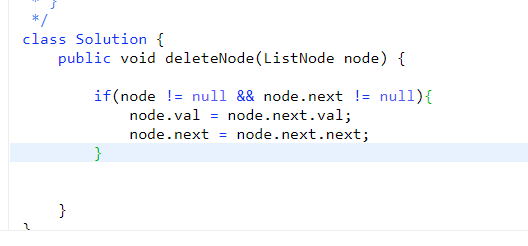
**Solutions:**

# ***LINKED LIST:***-

Questions1 : Delete a node in a linked list

Note: The give node is not a Head node of the list u r given a Node in a fuction that need to be deleted write your logic for that only

Note2: Given node is not a Tail node !!

Solutions :   


Question2:

**21. Merge Two Sorted Lists**

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

**Example 1:**



**Input:** list1 = [1,2,4], list2 = [1,3,4]

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

***Solutions:-*** *We need to combine these sorted list in such a way that a final merge list will be in Sorted array*

*Write now we have to make a entire new Linked list:*

*Two solutions – 1. Inplace Algorithms 2. Outplace (External Space)*

#### 

#### This is the simple solutions for merging two list

#### Make a node dummy and give the reference of that dummy node to one pointer

#### Check the value of l1 & l2 update the dummy node and move or increment the node value by 1 as well.

#### #Optimal Solutions for this-

#### From line 15 to 19 I have seen that by any chance if l1.val > l2.val then swap both of the val and we are maintaining the l1 as small and l2 as large val.

#### 

Step 1: Create two pointers, say l1 and l2. Compare the first node of both lists and find the small among the two. Assign pointer l1 to the smaller value node.

Step 2: Create a pointer, say res, to l1. An iteration is basically iterating through both lists till the value pointed by l1 is less than or equal to the value pointed by l2.

Step 3: Start iteration. Create a variable, say, temp. It will keep track of the last node sorted list in an iteration.

Step 4: Once an iteration is complete, link node pointed by temp to node pointed by l2. Swap l1 and l2.

Step 5: If any one of the pointers among l1 and l2 is NULL, then move the node pointed by temp to the next higher value node.

#### Question: Remove Nth node from the list from end !!

Given the head of a linked list, remove the nth node from the end of the list and return its head.

**Example 1:**



**Input:** head = [1,2,3,4,5], n = 2

**Output:** [1,2,3,5]

#### Solution 1 : take the count of the list then

#### Take value of count – n = node to be deleted;

#### We know this nth Node from the end is equal to count – n

#### Solution 2 :

#### Using two pointer approach we can solve this problem

#### We will create one empty list and slow & fast pointer pointing to empty list

#### Loop from I = 1 to I <= n i++

#### Fast = fast.next;

#### With this fast pointer will be at deleting location and the we will move slow pointer and fast pointer till fast.next != null

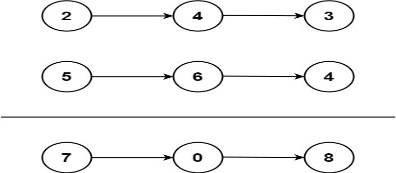
#### 

#### Question : Add two Numbers in linked List :

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**



**Input:** l1 = [2,4,3], l2 = [5,6,4]

**Output:** [7,0,8]

**Explanation:** 342 + 465 = 807.

#### Solutions :

#### *Take Two pointers to solve this problem and travers the list*

#### *While(l1 != null || l2 != null || carry == 1)*

#### *Sum = 0*

#### *Sum + =l1.value;l1 = l1.next*

#### *Sum += l2.value;l2 = l2.next;*

#### *Sum += carry;*

#### *ListNode node = new ListNode(sum % 10);*

#### *Carry = sum /10;*

#### *This is how We get the add of two numbers from the list!!*

#### 

*Question* **237. Delete Node in a Linked List**

Medium

38721087Add to ListShare

There is a singly-linked list head and we want to delete a node node in it.

You are given the node to be deleted node. You will **not be given access** to the first node of head.

All the values of the linked list are **unique**, and it is guaranteed that the given node node is not the last node in the linked list.

Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:

* The value of the given node should not exist in the linked list.
* The number of nodes in the linked list should decrease by one.
* All the values before node should be in the same order.
* All the values after node should be in the same order.

#### *Solution: We are give a node directly to delete it what we can do we can use the swap Concept for this*

#### *If(node != null && node.next != null){*

#### *Node.val = node.next.val;*

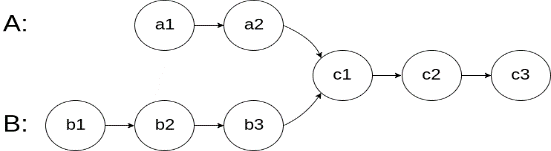
#### *Node.next = node.next.next;*

#### *}*

#### *Question : Find intersection of 2 linked List:*

Given the heads of two singly linked-lists headA and headB, return *the node at which the two lists intersect*. If the two linked lists have no intersection at all, return null.

For example, the following two linked lists begin to intersect at node c1:



The test cases are generated such that there are no cycles anywhere in the entire linked structure.

**Note** that the linked lists must **retain their original structure** after the function returns.

#### *Solutions:*

#### *Bruteforce : a. Take all the nodes find the other node I* if both the node are same then return that

#### It will take O(n^2) TC

#### Output of this linked list is c1 as both the list is meeting at C1 only !!

#### Approach 1

#### 

#### It’s like nested loop with O(n^2);

#### Approach 2 : Using External data Strucutre like HashSet

#### 

#### Approach 3: calculate length of both the linked list find the difference of the list and move pointer to that node so that both the length of linked list will be equal

public ListNode getIntersectionNode(ListNode headA, ListNode headB) {

int l1 = length(headA);

int l2 = length(headB);

int d = 0;

ListNode ptr1;

ListNode ptr2;

if (l1>l2)

{

d = l1-l2;

ptr1 = headA;

ptr2 = headB;

}

else

{

d = l2-l1;

ptr1 = headB;

ptr2 = headA;

}

while(d>0)

{

if (ptr1==null)

{

return null;

}

ptr1 = ptr1.next;

d--;

}

while (ptr1!=null && ptr2!=null)

{

if (ptr1==ptr2)

return ptr1;

ptr1 = ptr1.next;

ptr2 = ptr2.next;

}

return null;

}

public int length(ListNode head)

{

int i = 0;

while(head!=null)

{

i++;

head = head.next;

}

return i;

}

#### Approach 4 : Is two pointer approach where we will traverse the list till one of the is equal to null and assign the head of another list and viceversa after that both of them travers at equal distance

#### As soon as d1 == d2 loop will break and we can return the node of the list !!

#### Question: Detect Cycle in a linked List

#### 

#### This above solution is taking time complexity of TC = O(n) + space Complexity O(N).

#### Approach2 : using two pointer solution we can solve this

#### While (fast != null && fast.next != null){

#### Fast = fast.next.next;

#### Slow = slow.next;

#### If(slow == fast)

#### Return true; // Cycle exist in the linked List

#### TC = O(n) , SC = O(1)\

#### Question: Reverse node in k – group | reverser K – linked List:

#### To reverse the chunks of K In linked list what we can do is we can use the method “AddFirst()” in linked list

#### We will maintain four pointer to do this approach

#### orgHead orgTail tempHead, tempTail

#### all will point to null

#### step 1: take the current addFirst() into the tempHead and do this adding for k times and assign this tempHead and tempTail value to orgHead and orgTail

#### and soon do this stuff ….

#### 

#### 

#### Question: Check if a linked list is palindrome or not

Given the head of a singly linked list, return true*if it is a palindrome or*false*otherwise*.

**Example 1:**

https://assets.leetcode.com/uploads/2021/03/03/pal1linked-list.jpg

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

**Output:** true

**Example 2:**

https://assets.leetcode.com/uploads/2021/03/03/pal2linked-list.jpg

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

**Output:** false

#### Brute Force Approach is take the reverse of linked list into other linked list and iterate that both the list if both are equal return true else return false.

#### Optimal Approach : Take 2 pointers fast and slow pointer

#### 

#### 

#### Question Detect first node of the cycle in the linked List Take 2 pointer fast and slow to find out the point where they will collide

#### After that creat one more pointer entry point which will start from head and after that slow will aslo move by one

#### At one point of time slow and entry colloid that is the loop start return the count from the entry point.

#### Given a Linked List of size N, where every node represents a sub-linked-list and contains two pointers: (i) a**next**pointer to the next node, (ii) a**bottom** pointer to a linked list where this node is head. Each of the sub-linked-list is in sorted order. Flatten the Link List such that all the nodes appear in a single level while maintaining the sorted order.  **Note:** The flattened list will be printed using the bottom pointer instead of the next pointer. For more clarity have a look at the printList() function in the driver code.

5 -> 10 -> 19 -> 28

| | | |

7 20 22 35

| | |

8 50 40

| |

30 45

**Output:**  5-> 7-> 8- > 10 -> 19-> 20->

22-> 28-> 30-> 35-> 40-> 45-> 50.

**Explanation**:

The resultant linked lists has every

node in a single level.

(**Note:** | represents the bottom pointer.)

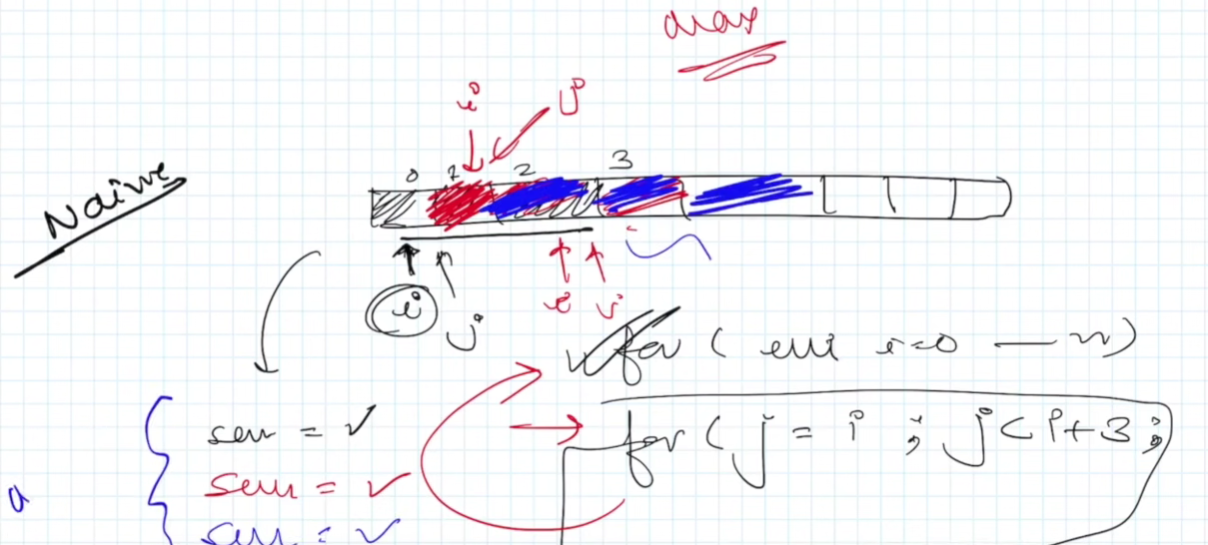
#### Solutions :

#### If there are two linked list we can make two pointer and solve this problem

# **SLIDING WINDOW PROTOCOLS:**

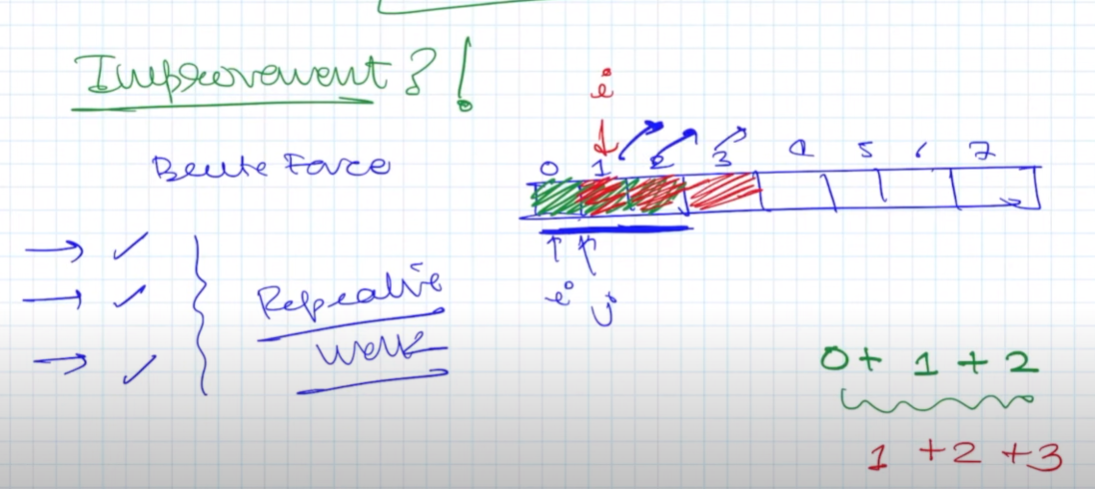
In SWP we have to take the sub arrays and we have to find the sub arrays from the array and find the sum

BRUTE FORCE : APPROACH TO FIND THE SUBARRAY IN O(n^2)is

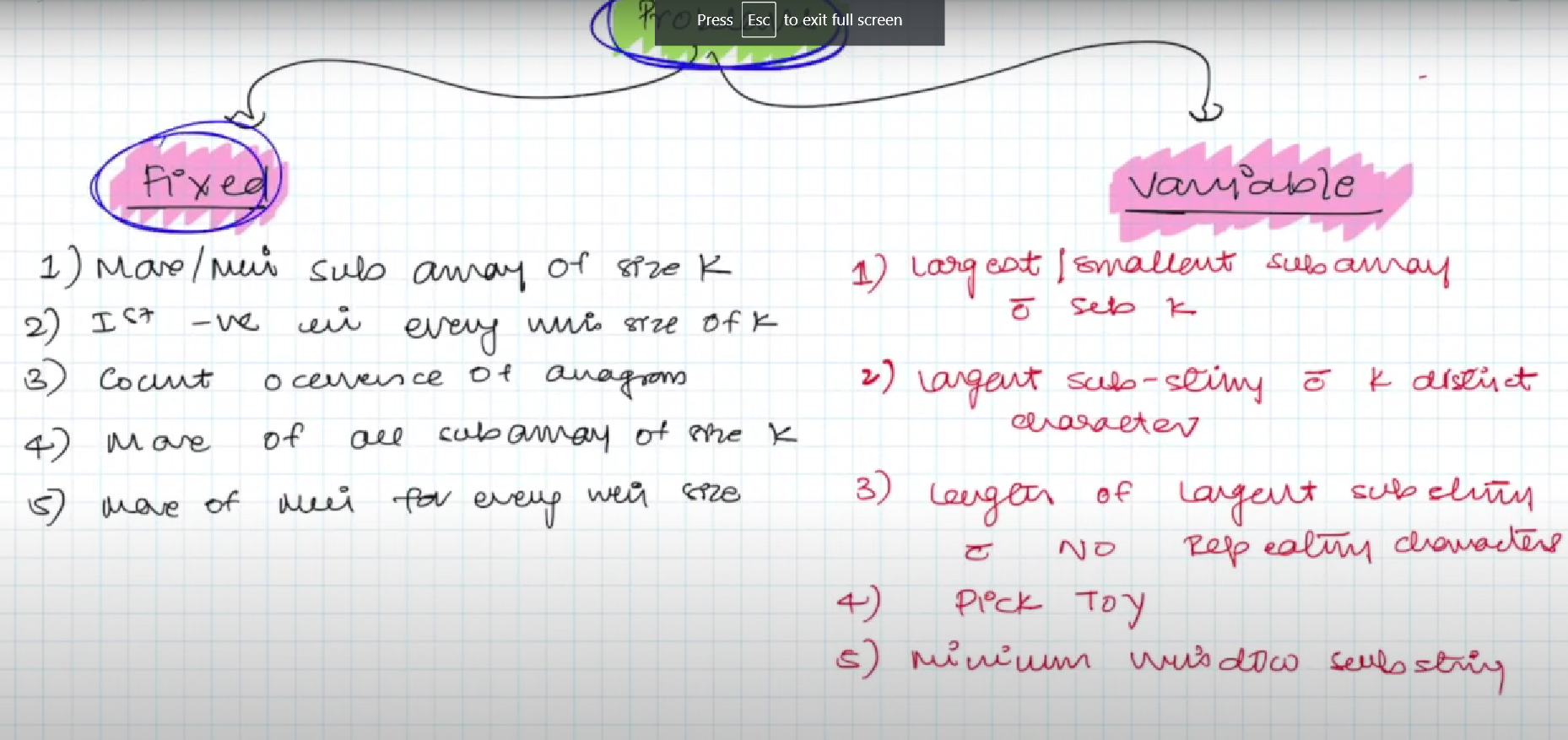


OPTIMAL APPROACH:

arr = {} with some values



Arrays and string problem are of Sliding Window problems!!



Here problems are divided into 2 category one is fixed size window and other one is variable size window!!

Question : Maximum sum of subarray of size K

Size = 7 & arr[] = { 2,5,,1,8,2,9,1};

K = 3

We need to find the MAX(sum) from this arrays of size 3

Given an integer array nums, find the subarray with the largest sum, and return its sum.

**Example 1:**

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

**Output:** 6

**Explanation:** The subarray [4,-1,2,1] has the largest sum 6.

**Example 2:**

**Input:** nums = [1]

**Output:** 1

**Explanation:** The subarray [1] has the largest sum 1.

Approach To Solve:

Question : **Longest Substring with At Least K Distinct Characters :**

**Solutions :**

Here we will have a string of characters along with value K

We need to find the longest substring with K character in that and that must be unique

Example : aabacbebebe K = 3

Output for the following Code is as: cbebebe

Approach to solve this problem we will maintain one hashmap through the the string length;

Step : 1 Take two pointer i and j init with -1

Step : 2 loop till string of length and check weather we have the character in map or not if yes then we will update the occurrence of that character

Step:3 Here we will check if the map.size() < k (we will continue the loop till the size equal to K)

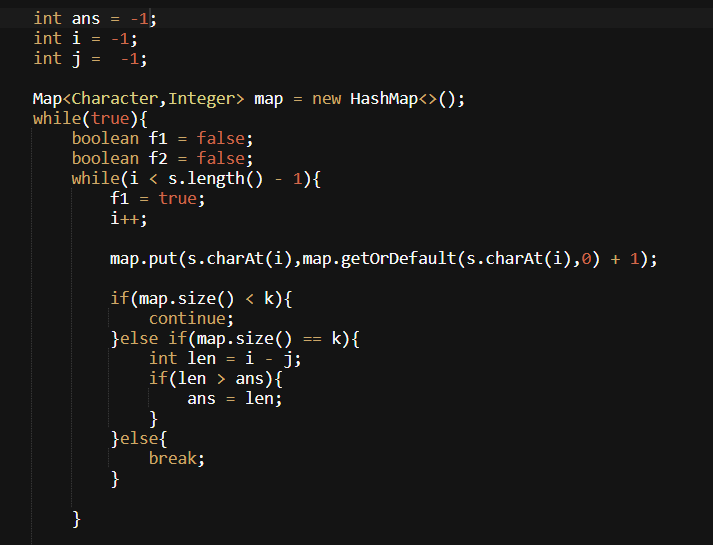
||

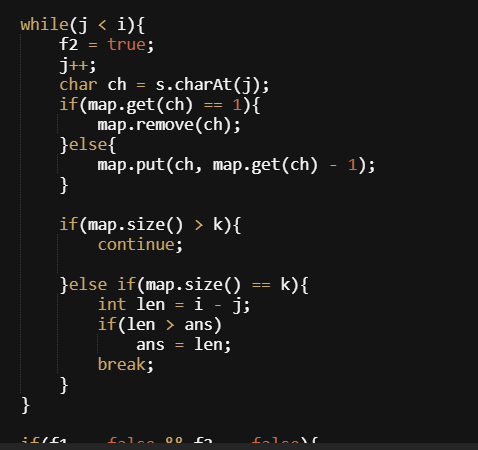
map.size() == k (then we will calculate the length int len = i – j

and check the value if (len > ans ) ans = len;

|| map.size() > k (here we will break the (inner loop and start another loop to make the map.size() == k)

Step:4 Here we will check J < I or not if yes then we will update the map value if character count is one then remove that otherwise decrease the value from the map

First face algo 1   


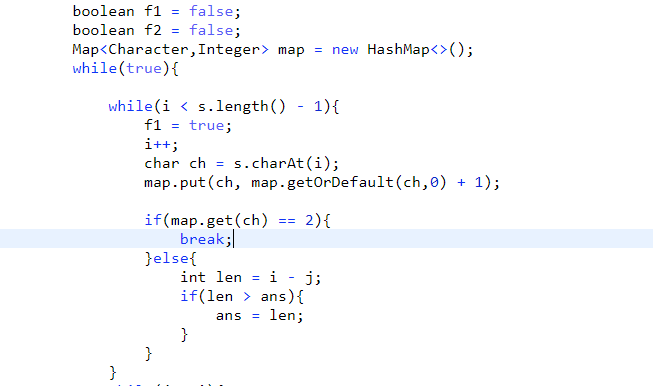


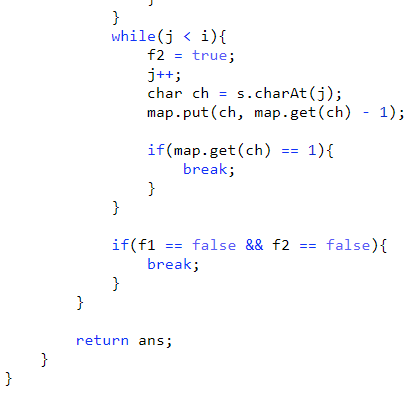
Question: Longest Substring without repeating Character :

Step1 : acquire the string

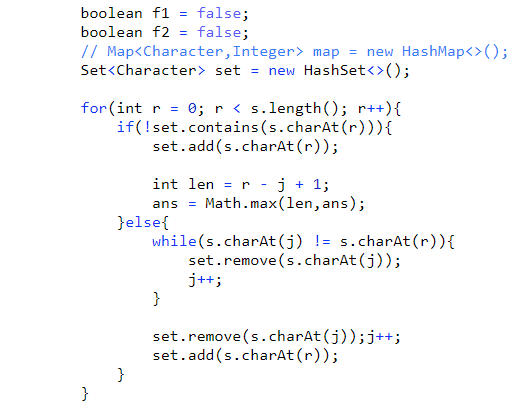
Step2 validate the length

Step3: release if it’s more than one repeating character !!





Approach2 : Using hashset with similar sliding window left and right and calculate the distance of left and right with => right – left + 1



Questions: Count Of Substrings Having All Unique Characters

Aabcbcdbca Sample Output = 24

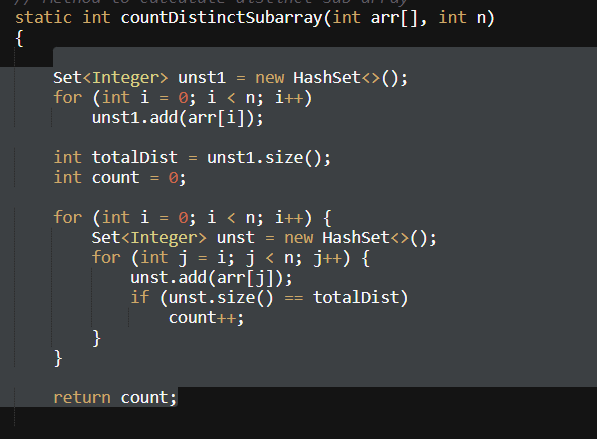
Solution:

Step1: check the string and add it into the map if value of str is == 2 then we have to release after that we will count the total number of substring in between left and right pointer



# Questions: Count subarrays having total distinct elements same as original array

The approach is to get the distinct subarrays for the given arrays



**Question: Maximum Consecutive One’s**

Given a binary array nums and an integer k, return the maximum number of consecutive 1's in the array if you can flip at most k 0's.

**Example 1:**

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

**Output:** 6

**Explanation:** [1,1,1,0,0,**1**,1,1,1,1,**1**]

Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.

**Example 2:**

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

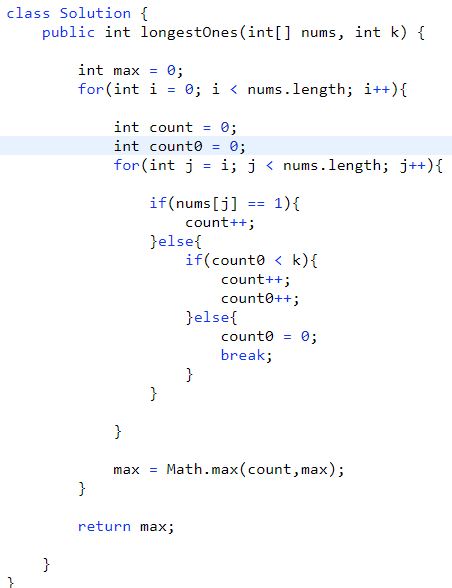
**Output:** 10

**Explanation:** [0,0,1,1,**1**,**1**,1,1,1,**1**,1,1,0,0,0,1,1,1,1]

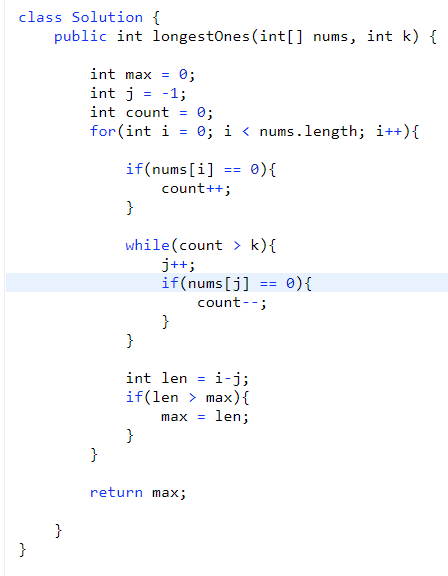
Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.

Solution : BRUTE\_FORCE

1. Take two count first is to maintain the max number of one’s other one is to count 0 if it’s less than k then we can flip the 0
2. If 0 > k then we will break and find the maximu,



OPTIMAL APPRAOCH:



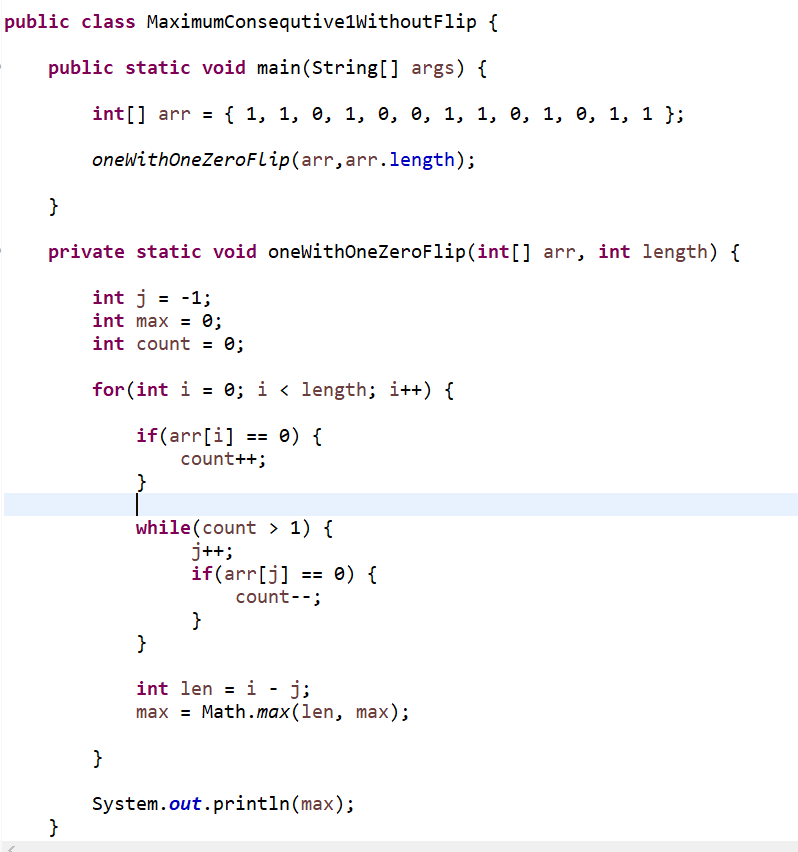
Code explain:

1. We are looping the array and maintain the count of 0
2. If count of 0 > K then we will release the o from the count
3. After that we will calculate the length from I – j
4. After this fetch the max len and return after all iterations

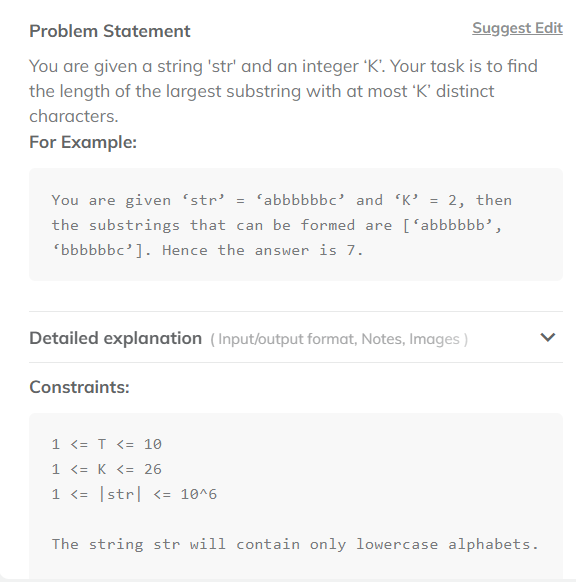
Question: Maximum Consecutive 1’s where we can flip the atmax one 0 only

Solutions : The approach in such a way is like we will check for the o’s if it’s greater than 1 then we will release the 0 and take the len value !!

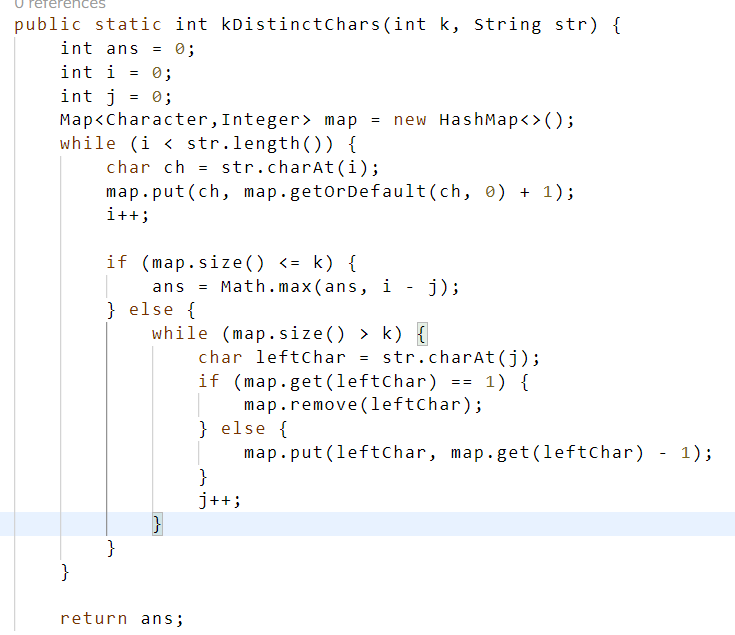
At every iteration we will get the max len if it’s present !



Question Longest Substring With Atmost K unique character



Solution :



Question: