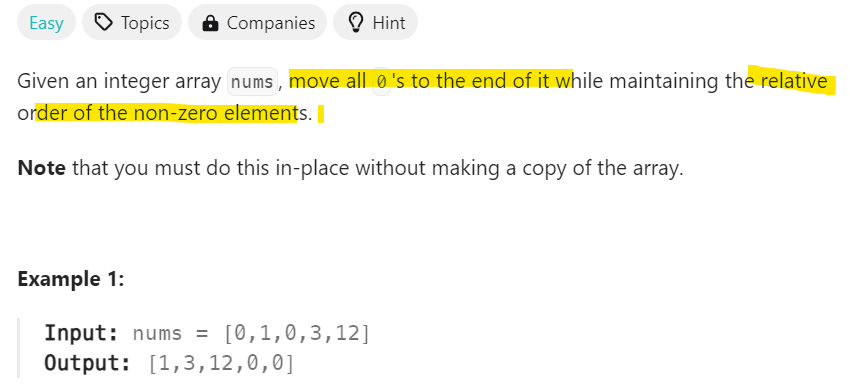
Move Zeroes : <https://leetcode.com/problems/move-zeroes/description/>  
  
My Approach :   
**We will use 3 variable temp like we sort array but here if element is 0 we will swap it .  
class Solution {**

**public int[] moveZeroes(int[] nums) {**

**int temp;**

**for(int i=0 ; i< nums.length ; i++){**

**for(int j=i+1;j<nums.length;j++){**

**if(nums[i]==0){**

**temp=nums[i];**

**nums[i]=nums[j];**

**nums[j]=temp;**

**}**

**}**

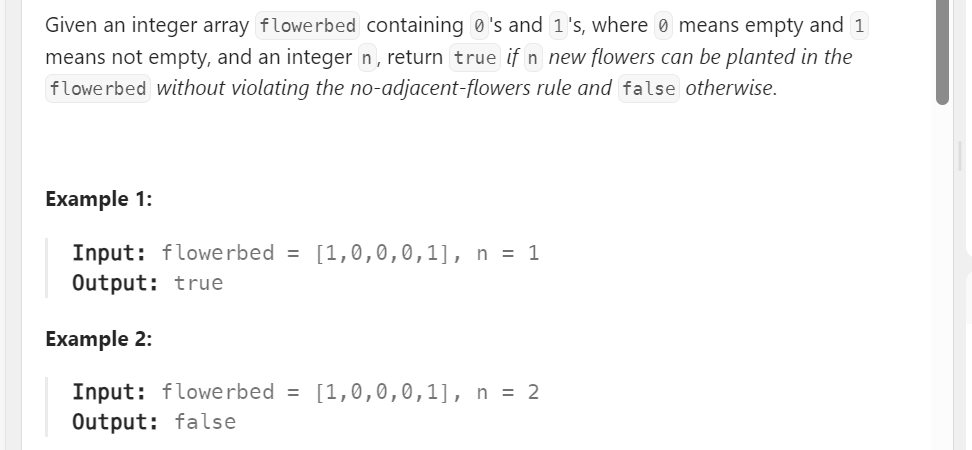
**}**

**return nums;**

**}**

**}**

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

Can Place Flowers: <https://leetcode.com/problems/can-place-flowers/description/>  
  
**Here we have to plant n trees in which previous and next element should not be 1** .  
Now we have to plant n trees keeping the above condition.  
Solution:  
class Solution {

public boolean canPlaceFlowers(int[] flowerbed, int n) {

int count=0;

for(int i=0;i<flowerbed.length;i++){

**if(flowerbed[i]==0){**

**int prev=(i==0 || flowerbed[i-1]==0)?0:1;**

**int next=(i==flowerbed.length-1 || flowerbed[i+1]==0)?0:1;**

**if(prev==0 && next ==0){**

**flowerbed[i]=1;**

**count++;**

**}**

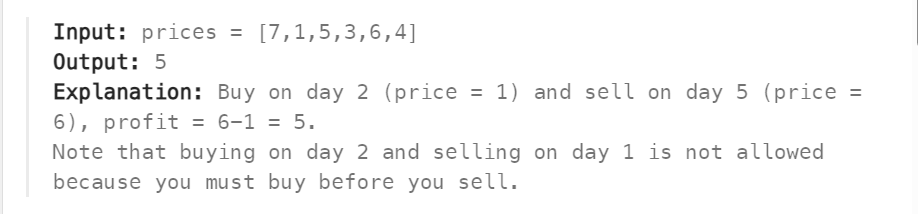
**}**

**}**

return count>=n;

}

}  
Explanation: **If it is first element we need not check the prev element and if it is last element we will not check next element and if it middle we will check both elements.**

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/>  
Leet code : Best time to buy and sell Stock  
  
  
  
So our approach is we take the **first element as min (price) min = price[0]; & profit =0**  
so if the after **elements is greater than min :** price[i]> min   
then **price[i]- min > profit then return price[i]-min  
Code:**class Solution {

public int maxProfit(int[] prices) {

int min = prices[0];

int profit =0;

for(int i=1;i<prices.length;i++){

if(min > prices [i]){

min = prices[i]; //best day to buy

}

else if(**prices[i]- min> profit**){

profit = prices[i]-min; //best day to sell

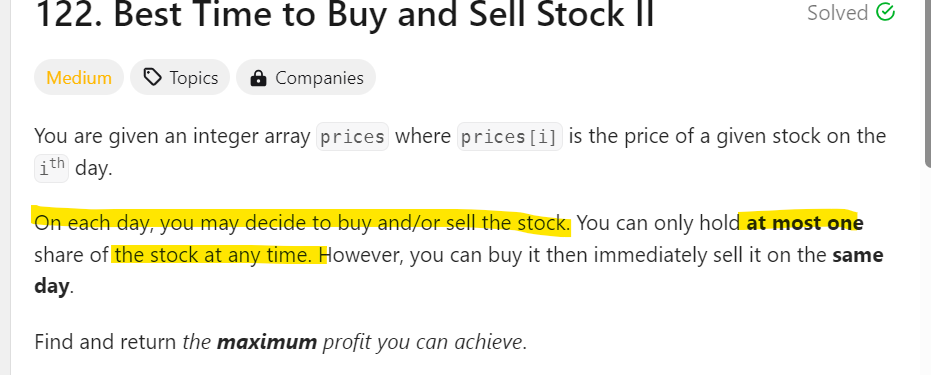
}

}

return profit;

}

}

Best Time to buy and sell stock 2  
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/description/>  
In Part 1 you can buy and sell stock at one time single day so in that way you have to   
calculate max profit  
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>  
  
Now in Part 2:You can **buy and sell multiple times   
  
So what we did whenever we will find profit ie number > min we will add it to profit  
  
class Solution {**

**public int maxProfit(int[] prices) {**

**int profit =0;**

**for(int i=1 ; i< prices.length ; i++){**

**if(prices[i] > prices[i-1]){**

**profit+= prices[i]-prices[i-1];**

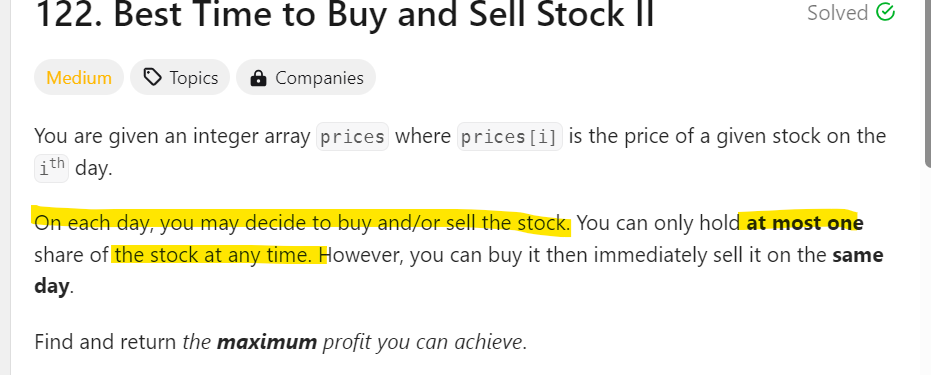
**}**

**}**

**return profit;**

**}**

**}**

------------------------------------------------------------------------------------------------------------------------------------------  
Best Time to buy and sell stock 2  
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/description/>  
In Part 1 you can buy and sell stock at one time single day so in that way you have to   
calculate max profit  
<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>  
  
Now in Part 2:You can **buy and sell multiple times   
  
So what we did whenever we will find profit ie number > min we will add it to profit  
  
class Solution {**

**public int maxProfit(int[] prices) {**

**int profit =0;**

**for(int i=1 ; i< prices.length ; i++){**

**if(prices[i] > prices[i-1]){**

**profit+= prices[i]-prices[i-1];**

**}**

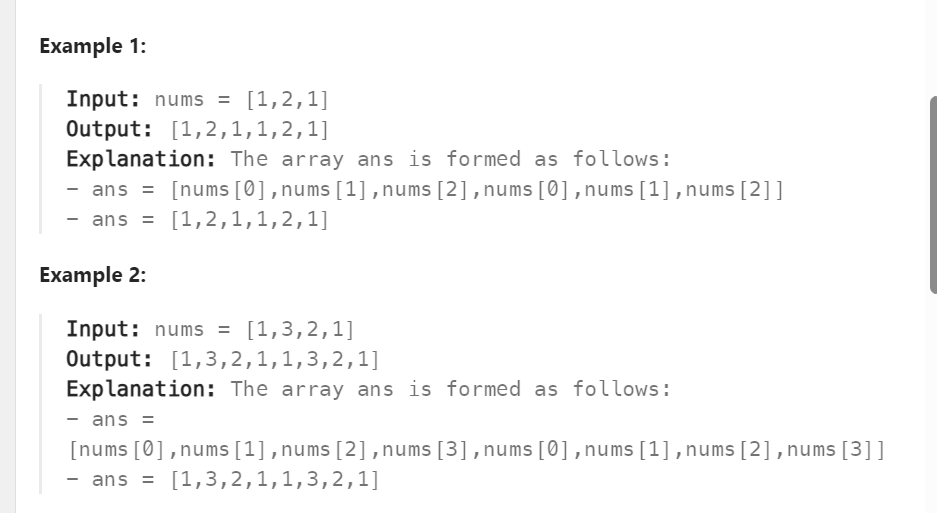
**}**

**return profit;**

**}**

**}**

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

Concatenation of Array: <https://leetcode.com/problems/concatenation-of-array/description/>  
  
-------------------------------------------------------------------------------------------------------------------------  
**Our approach: First we create array -> ans[] -> traverse through i< nums.length  
so ans[0]=1 to ans[2]=1   
Now for ans[3] i.e ans[nums.length] to i<nums.length\*2   
ans [3] = nums [0] -> ans[i -k] where k =nums.length  
--------------------------------------------------------------------------------------------------------------------  
Code:  
class Solution {**

**public int[] getConcatenation(int[] nums) {**

**int arr[]=new int[nums.length\*2];**

**int n=nums.length;**

**for(int i=0;i<nums.length;i++){**

**arr[i]=nums[i];**

**}**

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

**arr[j+n]=nums[j];**

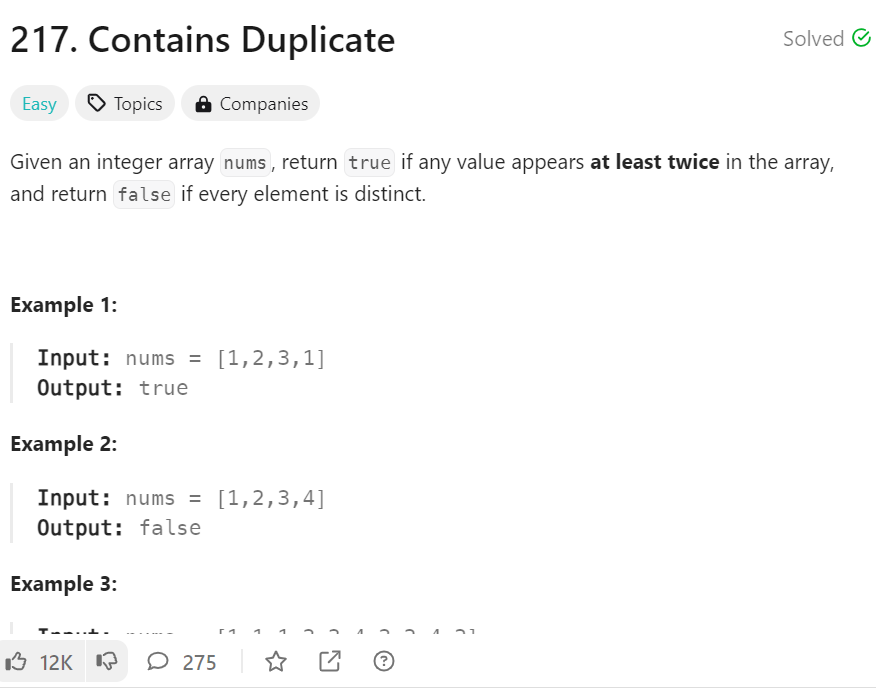
**}**

**return arr;**

**}**

**}**

**------------------------------------------------------------------------------------------------------------------------------------------** Contains Duplicate :   
<https://leetcode.com/problems/contains-duplicate>

  
  
Now we will use better appoach : **To sort array and then compare next element and if it is same then use return . After return keyword code ends didn't read below lines.**class Solution {

public boolean containsDuplicate(int[] nums) {

Arrays.sort(nums);

for(int i=0 ; i<nums.length-1;i++){

**if(nums[i]==nums[i+1]) return true;**

}

return false;

}  
----------------------------------------------------------------------------------------------------------------------------------------  
Fibonacci of a Number: <https://leetcode.com/problems/fibonacci-number/description/>  
  
**Starightforward approach a= 0 b=1 starting from n=2  
 so i can only be 1 so c = 0+1 =1;  
so when n=3 i can be 2 -> a=1 b=1 c=1+1->2  
  
Code:  
class Solution {**

**public int fib(int n) {**

**int a =0 , b=1;**

**int c =0;**

**if(n==0) return 0;**

**if(n==1) return 1;**

**for(int i=1;i<n;i++){**

**c=a+b;**

**a=b;**

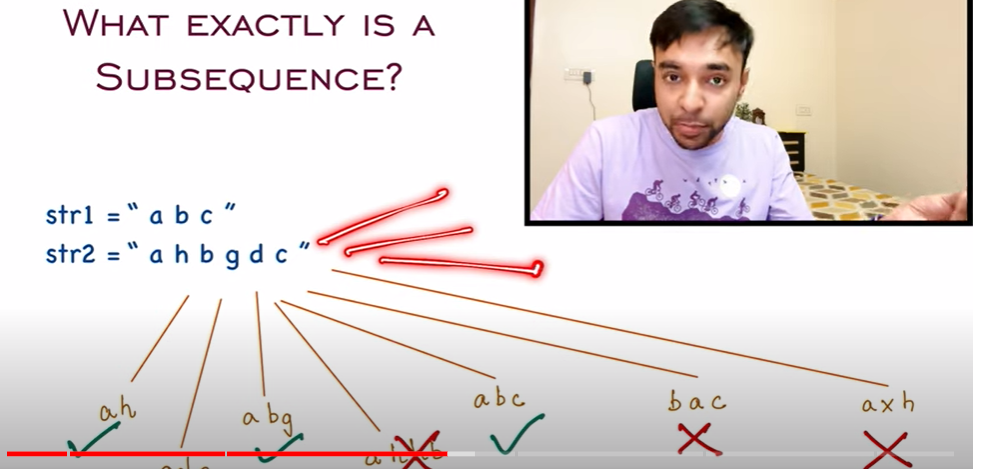
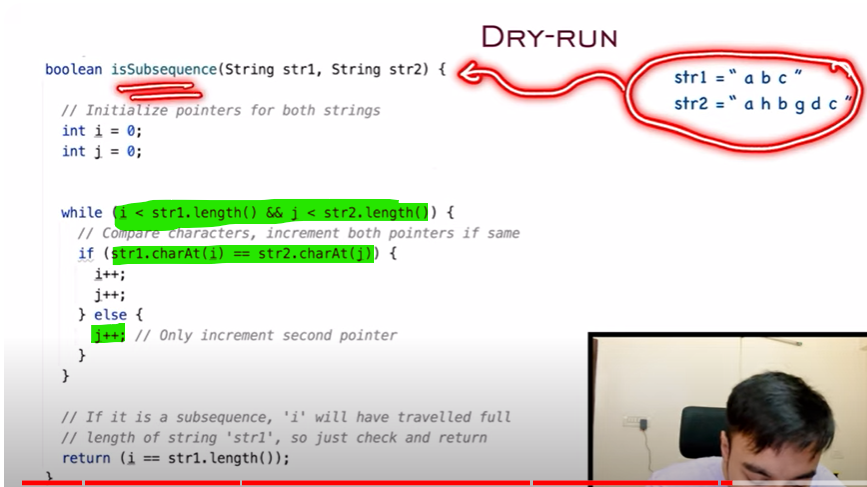
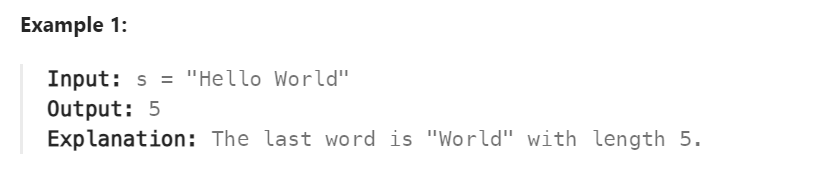
**b=c;**

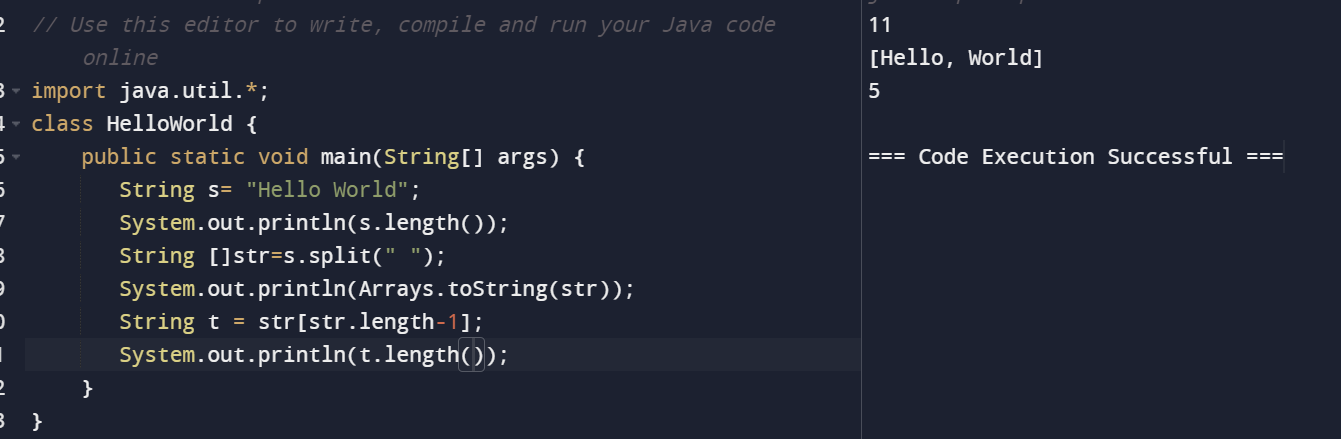
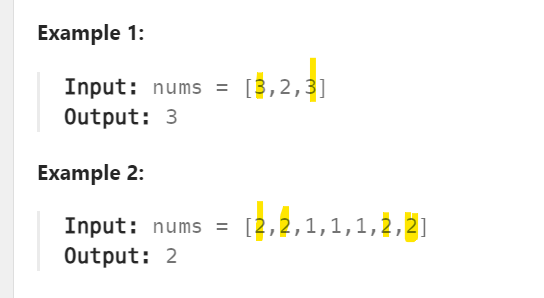
**}**

**return c;**

**}**

**}**

------------------------------------------------------------------------------------------------------------------------------  
  
https://leetcode.com/problems/is-subsequence/description/  
<https://www.youtube.com/watch?v=Gkz3SwtdSes>  
  
  
  
  
****  
  
--------------------------------------------------------------------------------------------------------------------------------------  
****

****  
---------------------------------------------------------------------------------------------------------------------------------------  
  
Majority Element :  
<https://leetcode.com/problems/majority-element/submissions/1321016628/>  
  
  
  
My approach : Voting algorithm   
Taking majority element as the first element   
if next element is same then votes is increase otherwise decrease  
if current votes == 0 then majorityElement is current element  
  
**class Solution {**

**public int majorityElement(int[] nums) {**

**int maxElement=nums[0];**

**int votes =0;**

**for(int i=0;i<nums.length;i++){**

**if(votes==0){**

**maxElement=nums[i];**

**}**

**if(maxElement==nums[i]){**

**votes++;**

**}else{**

**votes--;**

**}**

**}**

**return maxElement;**

**}**

**}**------------------------------------------------------------------------------------------------------------------------------------------  
  
Maximum Number of Balloons:   
<https://leetcode.com/problems/maximum-number-of-balloons/description/>

  
  
class Solution {

public int maxNumberOfBalloons(String text) {

int b\_count=0 ,a\_count=0 , l\_count=0 , o\_count=0 , n\_count=0;

for(char c : text.toCharArray()){

if(c=='b')b\_count++;

else if( c == 'a')a\_count++;

else if( c == 'l')l\_count++;

else if( c == 'o')o\_count++;

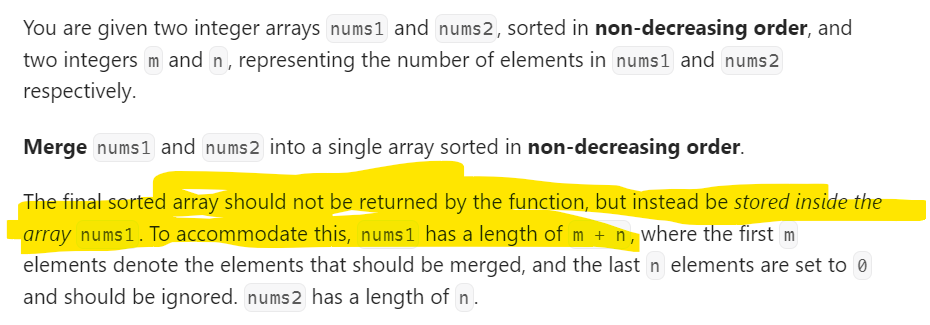
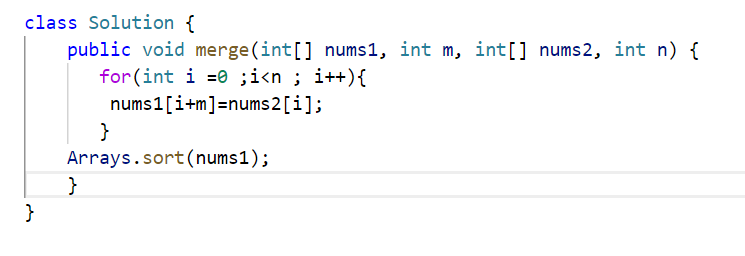
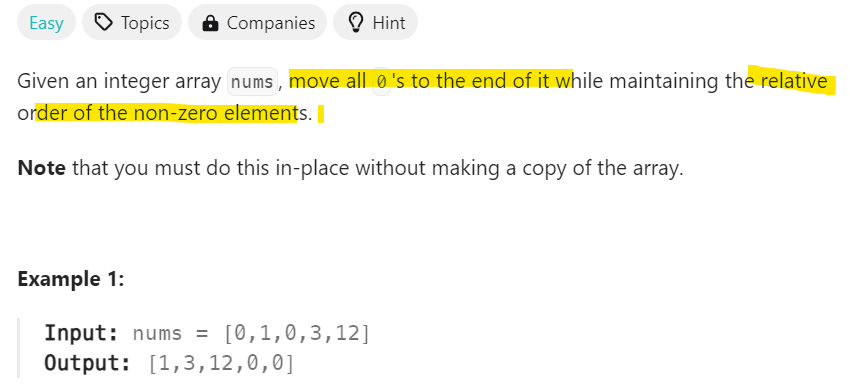
else if( c == 'n')n\_count++;

}

int max\_instances = Math.min(Math.min(Math.min(Math.min(b\_count, a\_count), n\_count), l\_count / 2), o\_count / 2);

return max\_instances;

}

-----------------------------------------------------------------------------------------------------------------------------------------  
Merge sorted Array :  
<https://leetcode.com/problems/merge-sorted-array/description/>  
  
  
  
  
****  
  
-----------------------------------------------------------------------------------------------------------------------------------------  
Move Zeroes : <https://leetcode.com/problems/move-zeroes/description/>  
  
  
**class Solution {**

**public int[] moveZeroes(int[] nums) {**

**int temp;**

**for(int i=0 ; i< nums.length ; i++){**

**for(int j=i+1;j<nums.length;j++){**

**if(nums[i]==0){**

**temp=nums[i];**

**nums[i]=nums[j];**

**nums[j]=temp;**

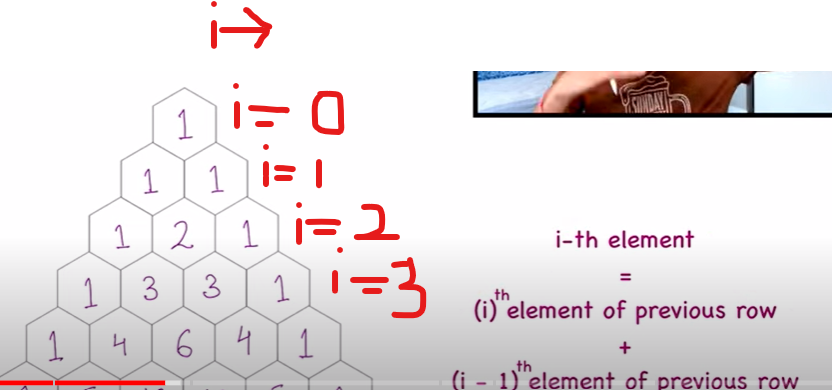
**}**

**}**

**}**

**return nums;**

**}**

-----------------------------------------------------------------------------------------------------------------------------------------  
Pascal's Triangle : <https://leetcode.com/problems/pascals-triangle/description/>: [ I](https://www.youtube.com/watch?v=nFqeCXOJn0I)

Refer link : <https://www.youtube.com/watch?v=nFqeCXOJn0I>  
**Observations:**  
Here you can see **if n=5**   
[1] **number of rows =n** ; [2**] first** & **Last element** of each row is 1   
[3] Suppose we want to find the 3 rd element of 4 th row -> i=3-> 3rd element of i=2 =>1 & 2nd element of i=2 =>2 so 2+1 =3;(**Formula given in pic)**  
[4] so now you see **number of colums** in each row if **we ignore first and last 1 values is i-1**;  
so **our loop loop will be : j=0 to j<i-1**[5] **j value = prevRow[j]+prevRow[j+1];  
  
Code:**class Solution {

public List<List<Integer>> generate(int numRows) {

List<List<Integer>> result = new ArrayList<>();

**if(numRows == 0) return result;**

List<Integer> firstRow = new ArrayList<>();

**firstRow.add(1);**

**result.add(firstRow);**

**if(numRows==1) return result;**

// start the next row

for(**int i=1;i< numRows ; i++){**

List<Integer>prevRow = result.get(i-1);

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

row.add(1);

for(int j=0 ; j<i-1;j++){

row.add(prevRow.get(j)+prevRow.get(j+1));

}

row.add(1);

//Add this row to final result

result.add(row);

}

return result;

}

}  
---------------------------------------------------------------------------------------------------------------------------------  
**Find Pivot Index :** [**https://leetcode.com/problems/find-pivot-index/description/**](https://leetcode.com/problems/find-pivot-index/description/) **My Approach: First we have taken total sum of array   
then we have subtaract start from first element and then continue   
if sum of leftSum == rightSum -> return i  
  
Code:**class Solution {

public int pivotIndex(int[] nums) {

int leftSum=0,rightSum=0, pivotIndex=-1;

**if(nums.length == 0) return -1**;

for(int i=0;i<nums.length;i++){

**rightSum+=nums[i];**

}

for(int i=0;i<nums.length;i++){

**rightSum -= nums[i];**

if(**rightSum==leftSum**){

return i;

}

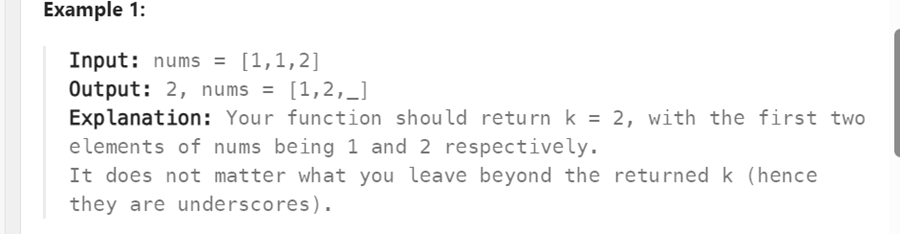
**leftSum += nums[i];**

}

return -1;

}

**}**

-------------------------------------------------------------------------------------------------------------------------------------  
  
Remove Duplicates from Sorted Array  
<https://leetcode.com/problems/remove-duplicates-from-sorted-array/description/>  
Youtube Refer Link: <https://www.youtube.com/watch?v=Fm_p9lJ4Z_8>  
  
  
  
  
  
  
  
  
class Solution {

public int removeDuplicates(int[] nums) {

**int i=0;**

**for(int j=1;j<nums.length;j++){**

**if(nums[i]!=nums[j]){**

**i++;**

**nums[i]=nums[j];**

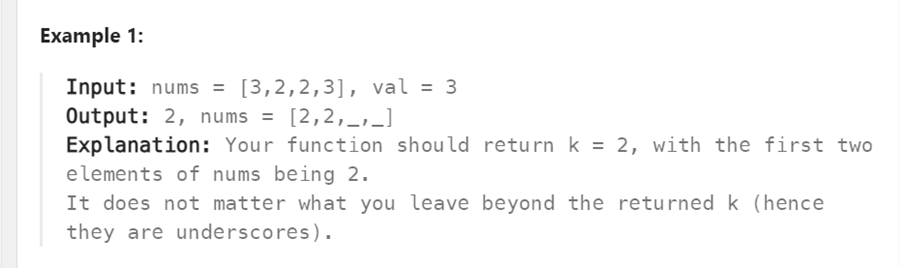
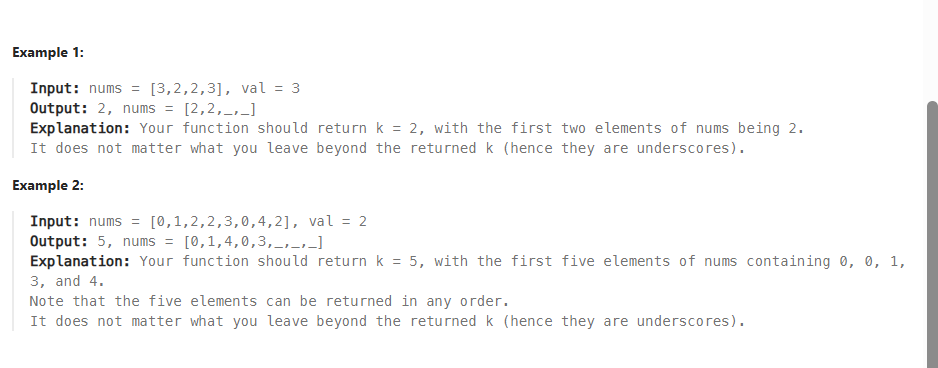
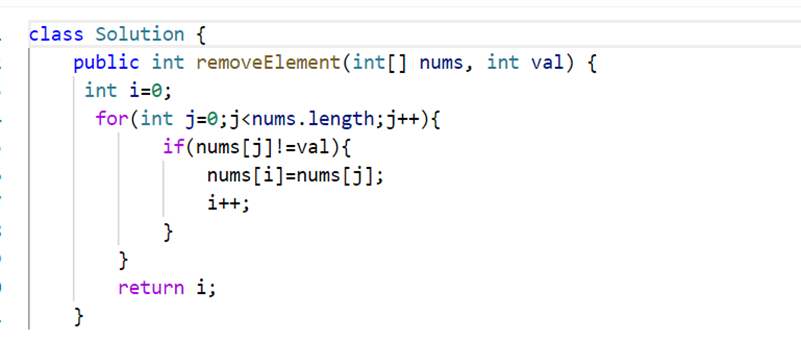
**}**

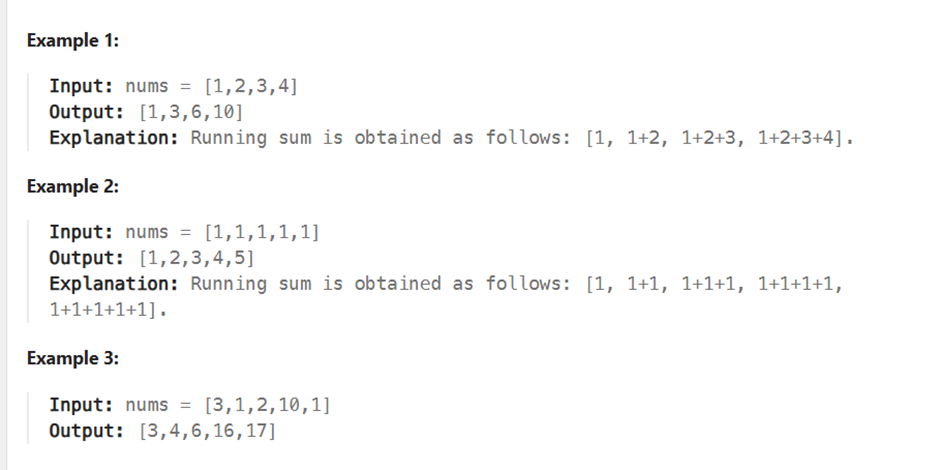
**}**

**return i+1;**

}

}

**Remove Element:** [**https://leetcode.com/problems/remove-element/description/**](https://leetcode.com/problems/remove-element/description/) ****Remove Element  **Same Concept : https://leetcode.com/problems/remove-element/description/**  
Our Approach :int i=0  
 **we will start from 0 if nums[j]!=val -> nums[0]!=val**then if condition satisfy-> nums[i]=nums[j];  
and then we have increment i.  
  
  
  
  
------------------------------------------------------------------------------------------------------------------------------------  
[**Running Sum of 1d Array**](https://leetcode.com/problems/running-sum-of-1d-array/) :  
https://leetcode.com/problems/running-sum-of-1d-array/description/

  
  
class Solution {

    public int[] runningSum(int[] nums) {

         int sum =0;

         int b[]=new int[nums.length];

      for(int i=0;i<nums.length;i++){

          for(int j=0 ;j<=i;j++){

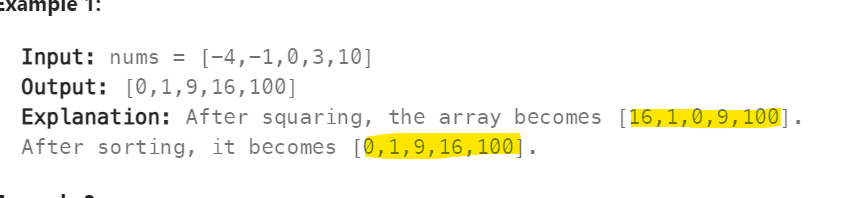
              b[i] += nums[j];

          }

      }

      return b;

    }

}  
  
-----------------------------------------------------------------------------------------------------------------------------  
Squares of a Sorted Array: <https://leetcode.com/problems/squares-of-a-sorted-array/description/>  
  
  
Code:  
class Solution {

public int[] sortedSquares(int[] nums) {

int arr[]=new int[nums.length];

for(int i=0;i<nums.length;i++){

arr[i]=nums[i]\*nums[i];

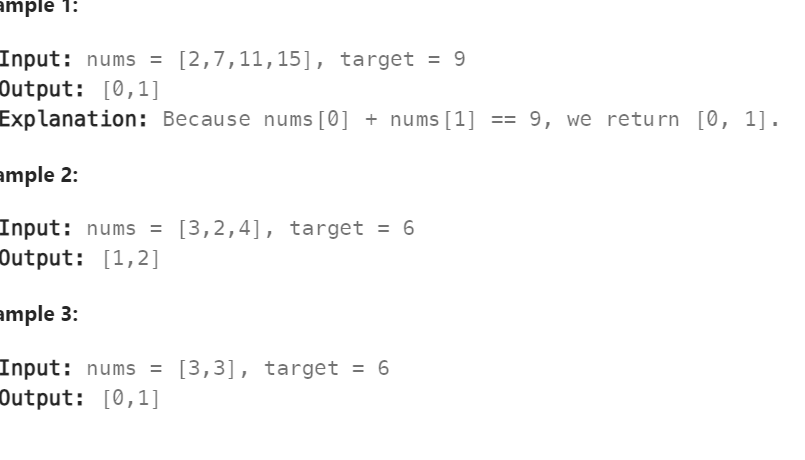
}

Arrays.sort(arr);

return arr;

}

}

-------------------------------------------------------------------------------------------------------------------------------  
[1] Two Sum  
<https://leetcode.com/problems/two-sum/description/>  
My Solution:  
class Solution {

public int[] twoSum(int[] nums, int target) {

int outArr[]=new int[2];

for(int i=0;i<nums.length;i++){

for(int j=i+1;j<nums.length;j++){

if(nums[i]+nums[j]== target){

outArr[0]= i;

outArr[1]=j;

}

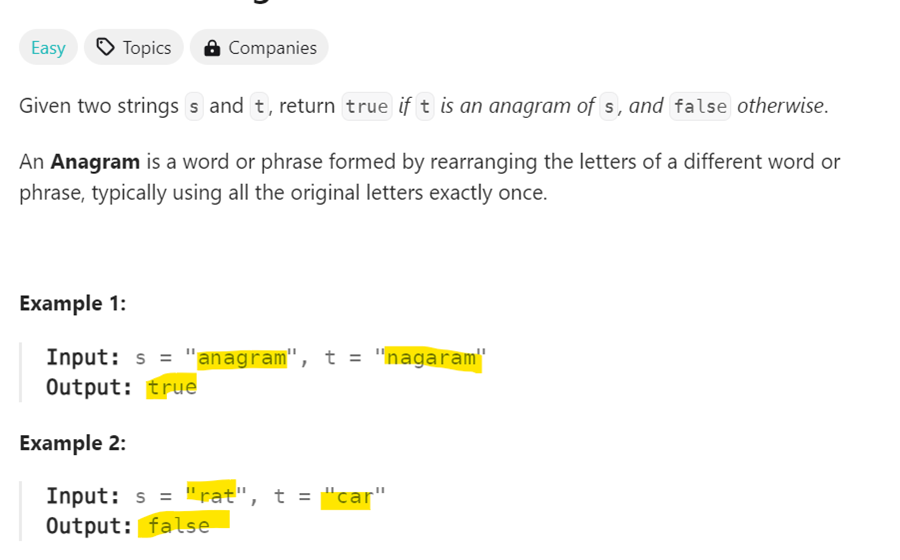
}

}

return outArr;

}

}

-----------------------------------------------------------------------------------------------------------------------  
Valid Anagram : <https://leetcode.com/problems/valid-anagram/description/>  
  
  


class Solution {

public boolean isAnagram(String s, String t) {

String [] str = s.split("");

String [] ktr = t.split("");

Arrays.sort(str);

Arrays.sort(ktr);

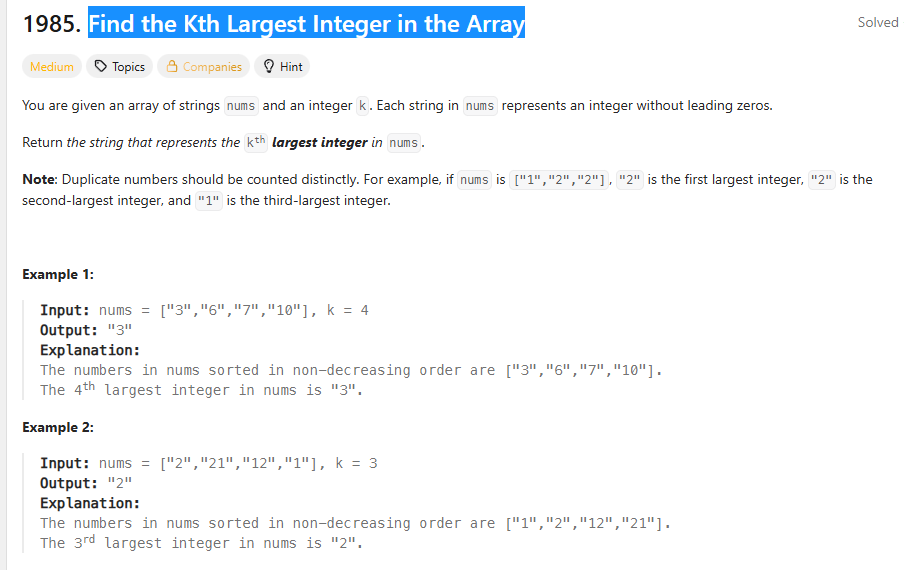
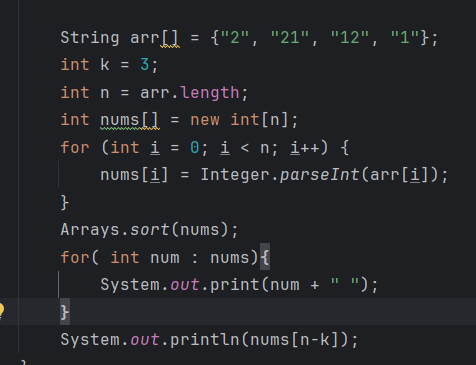
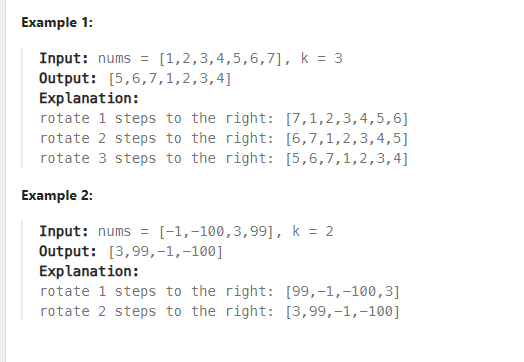
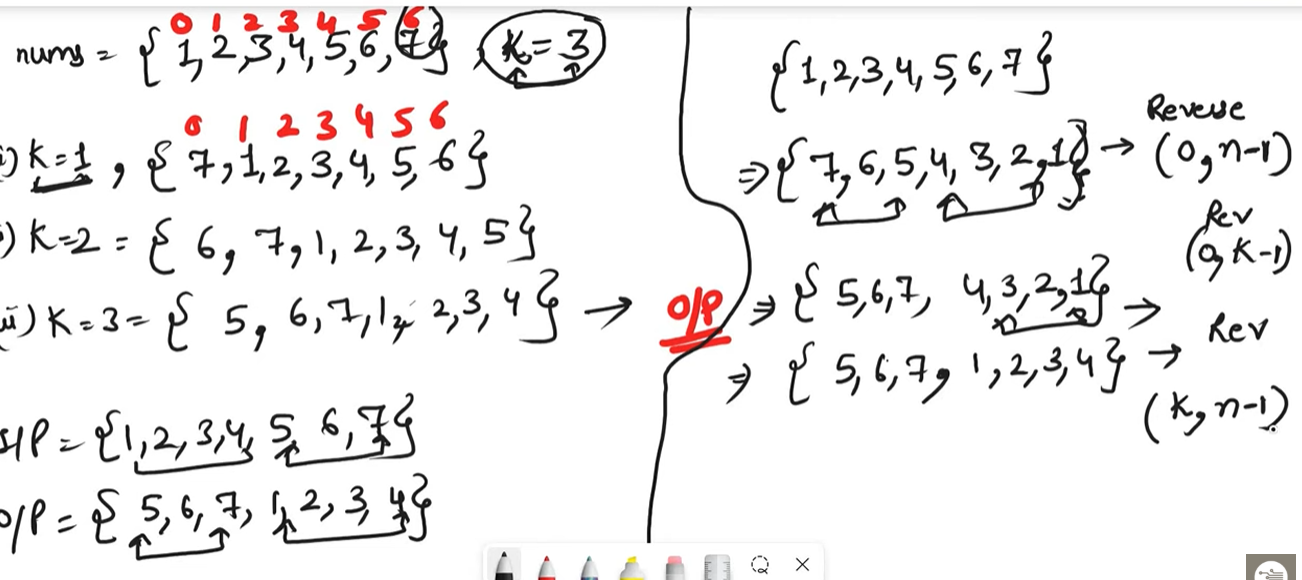
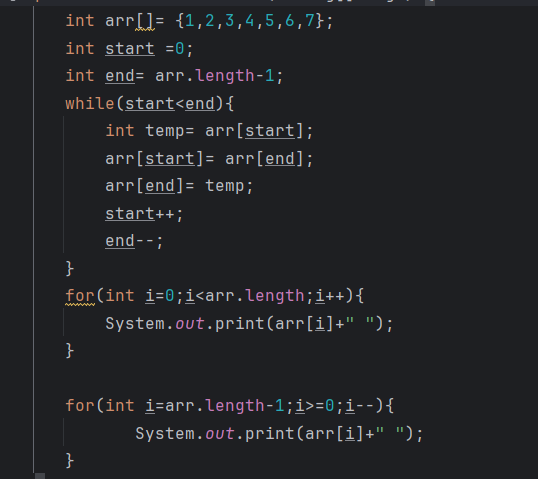
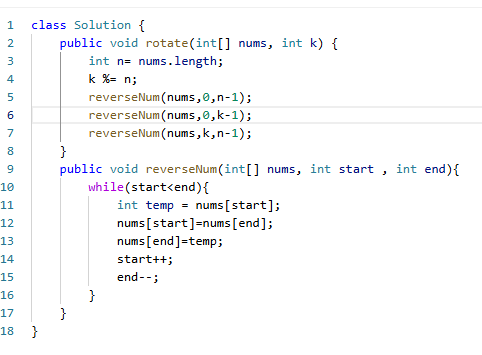
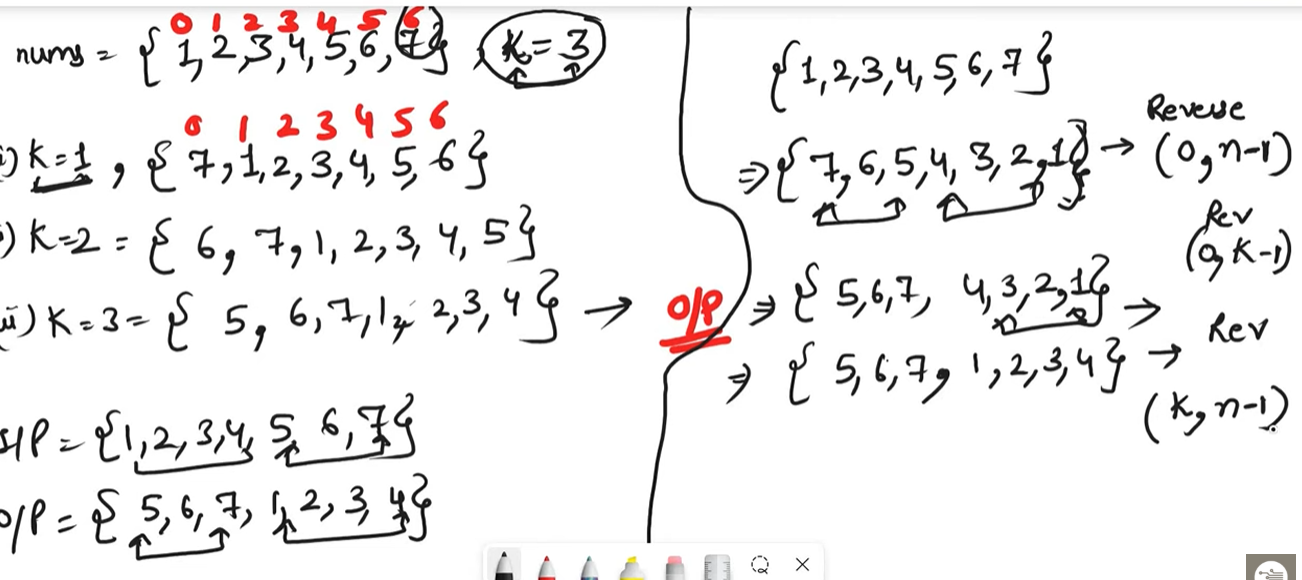
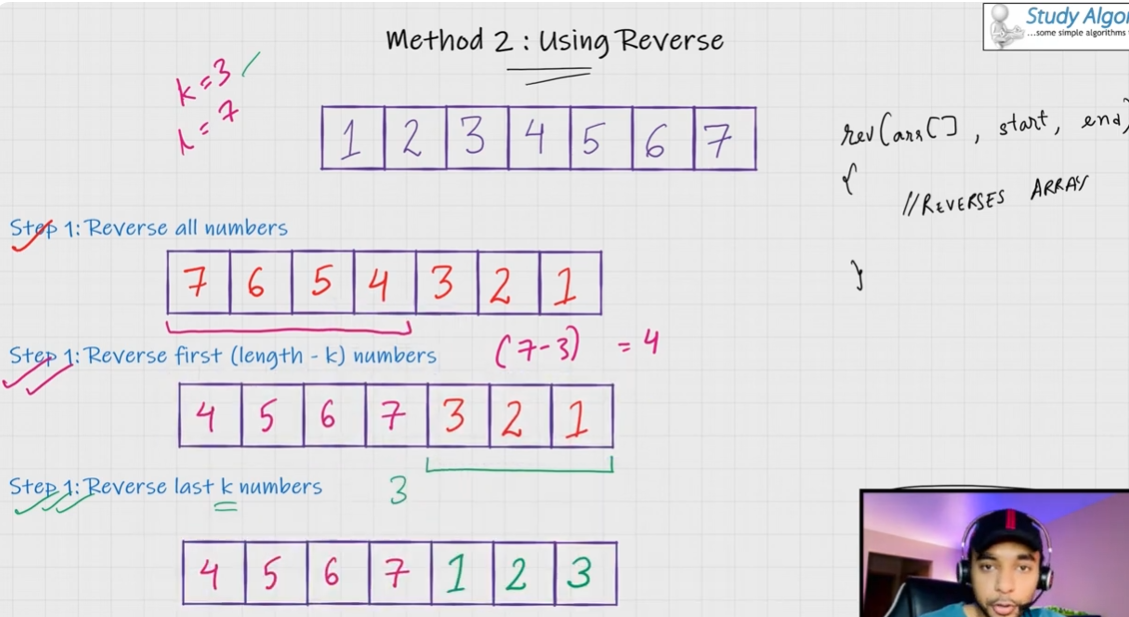
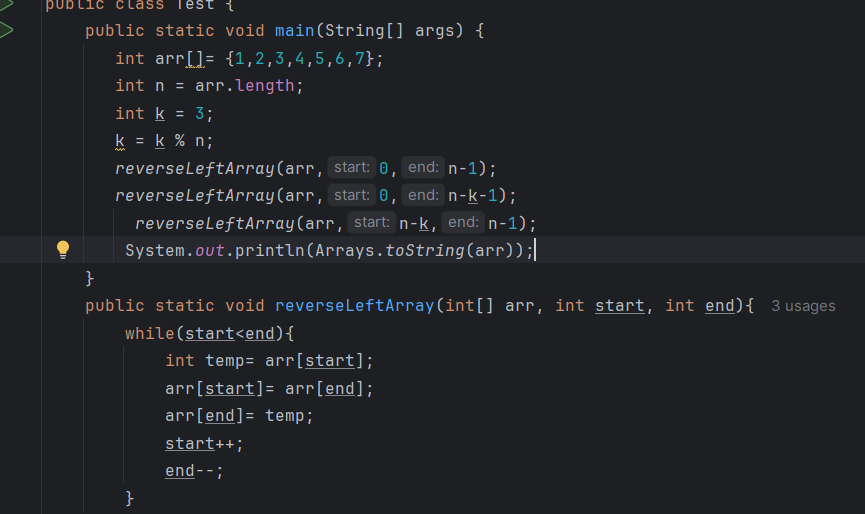
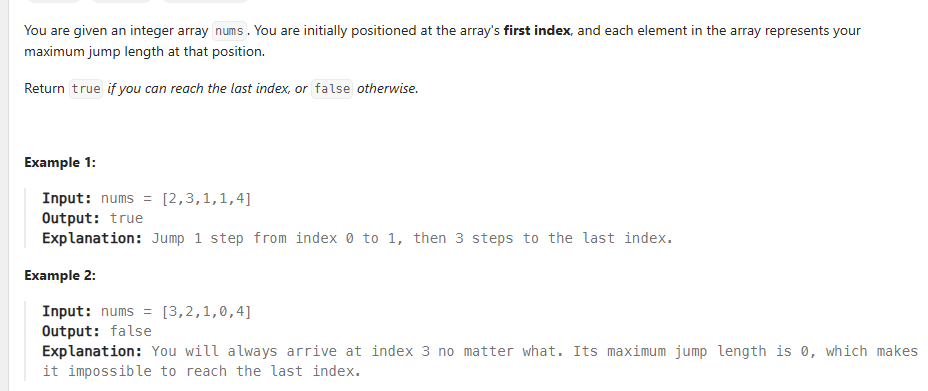
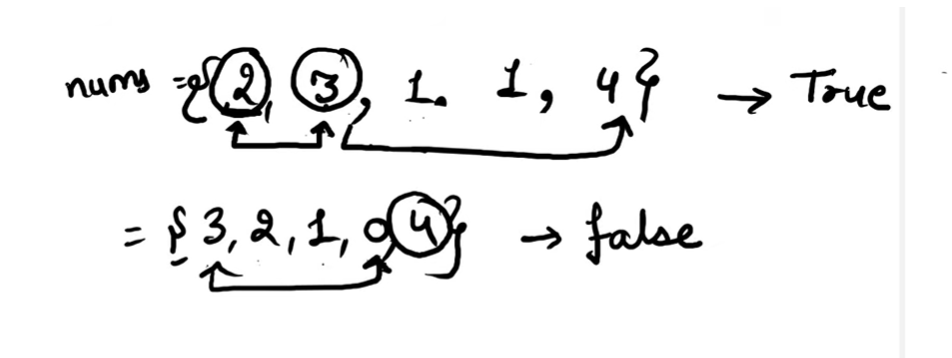
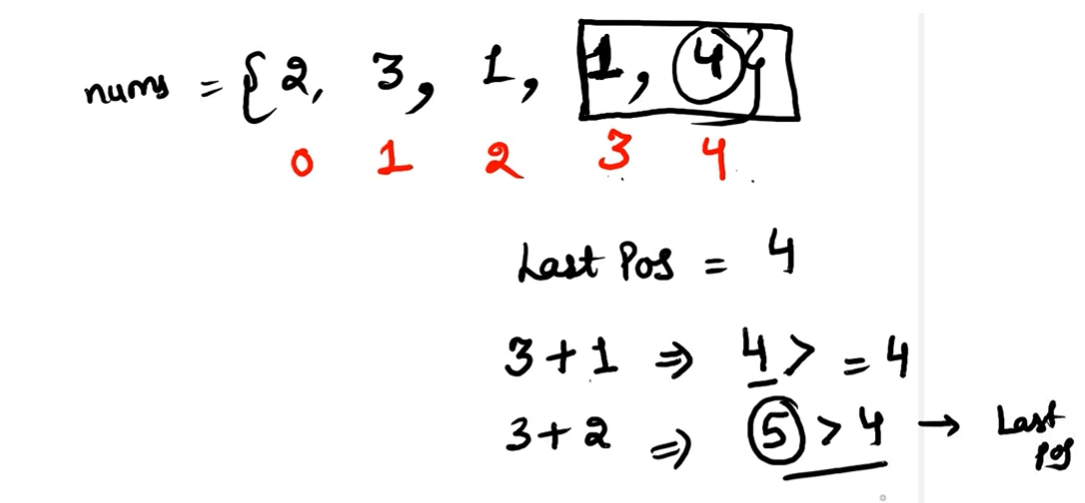
**List lt = Arrays.asList(str);**

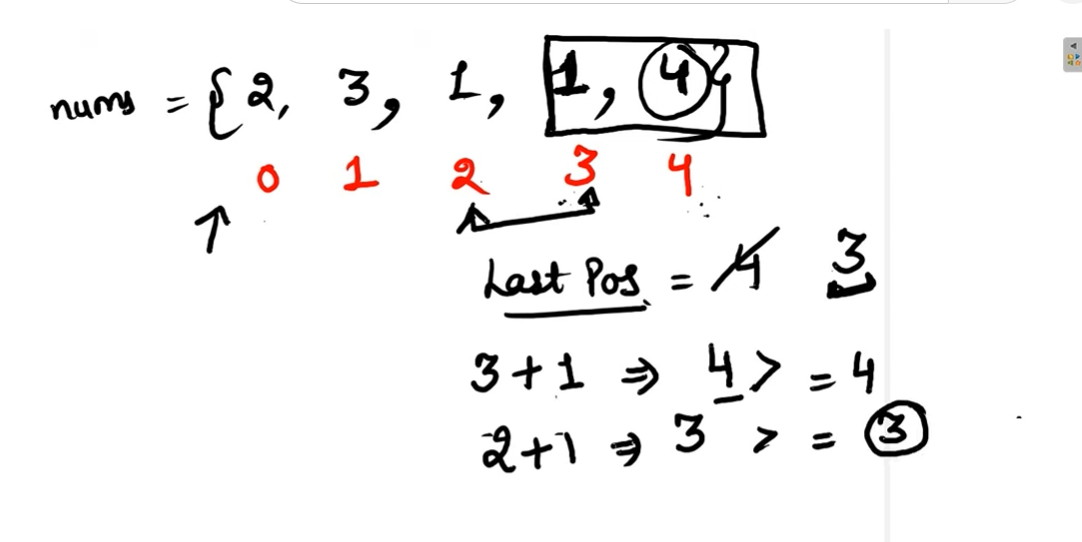
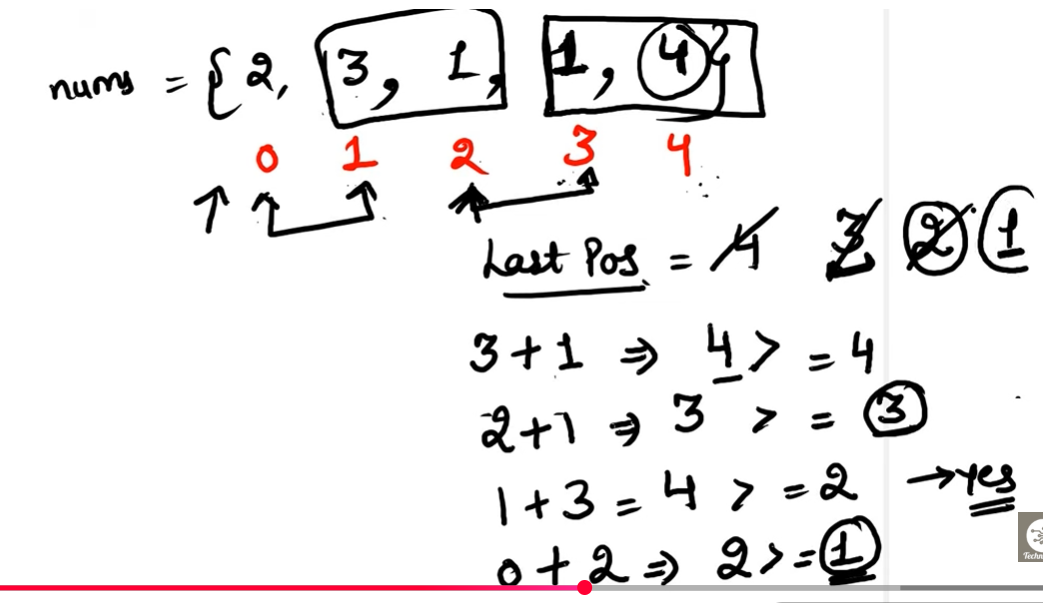
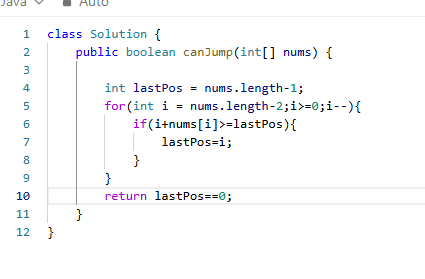
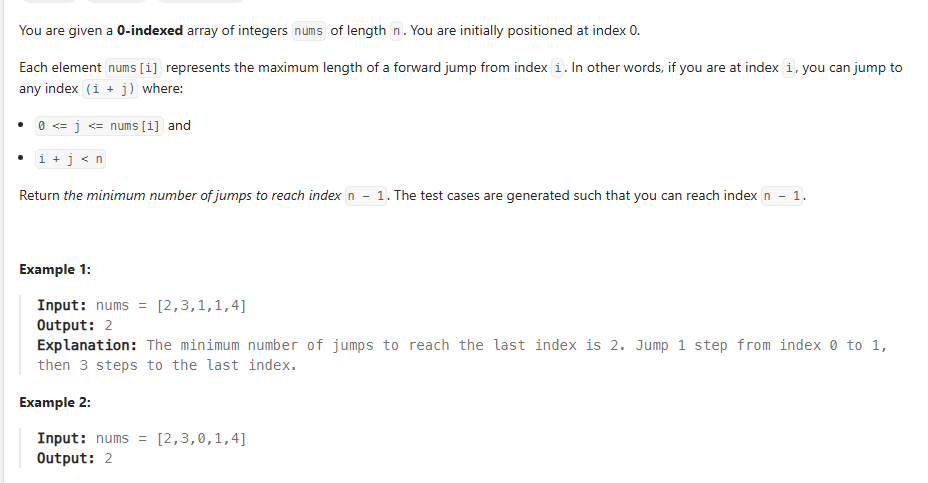
List kt = Arrays.asList(ktr);

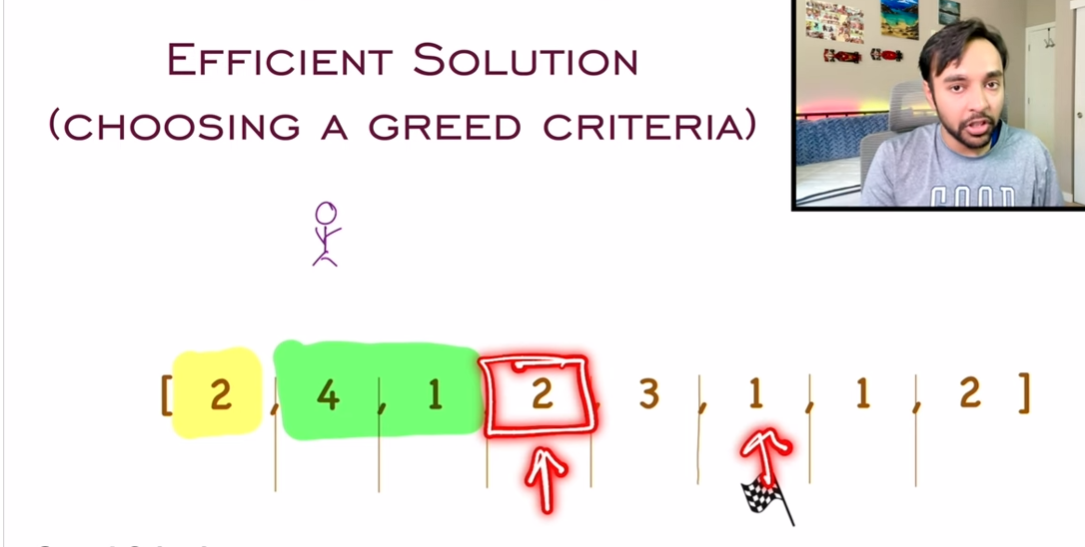
return lt.equals(kt);

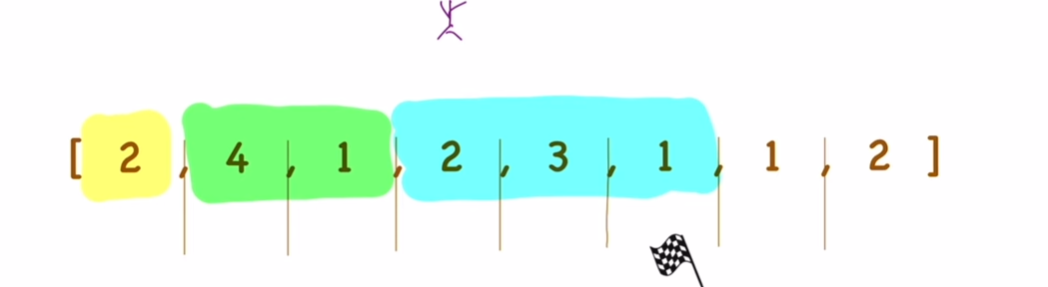
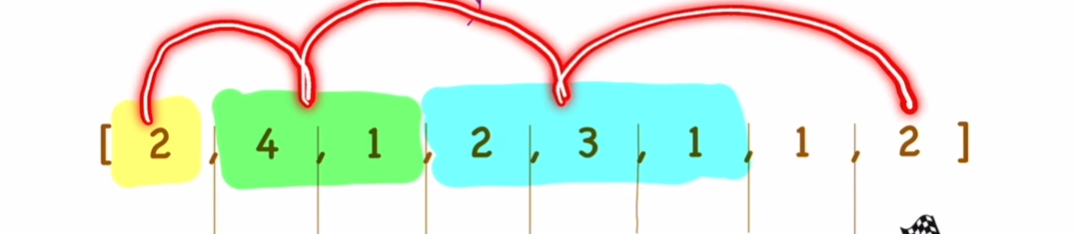
}

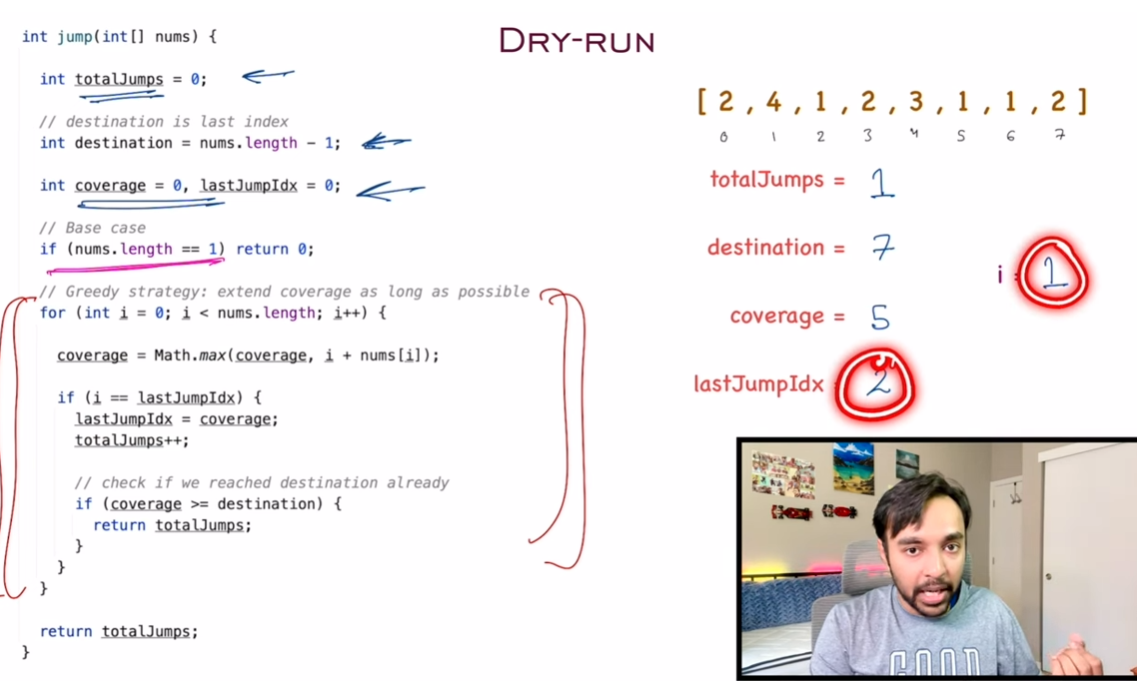
}

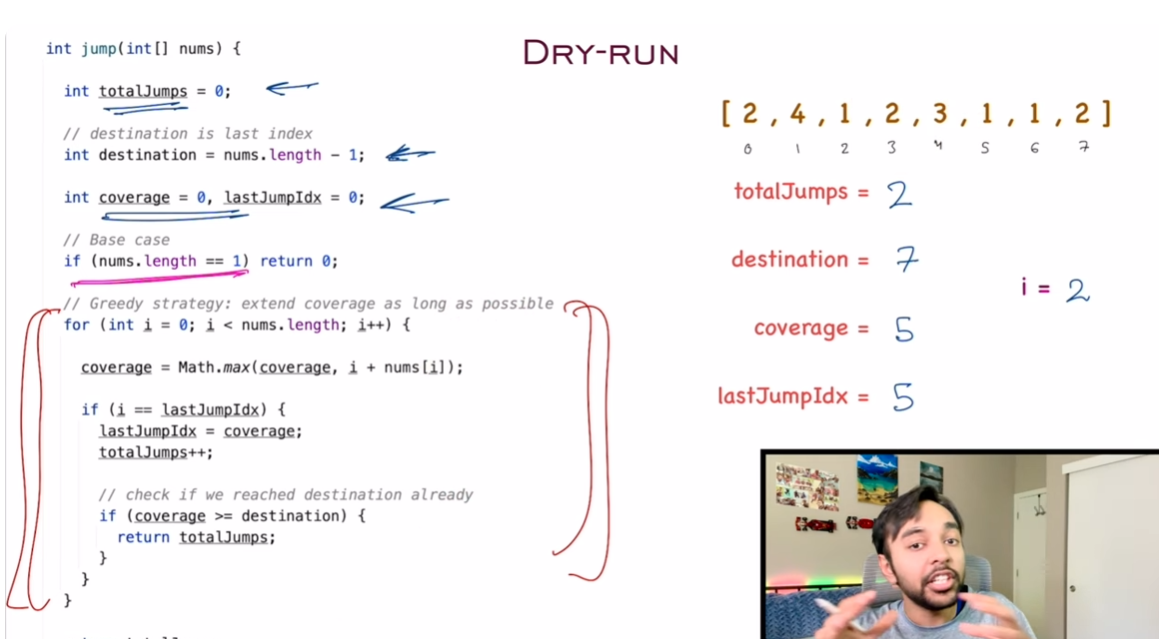
-----------------------------------------------------------------------------------------------------------------------------  
[**Find the Kth Largest Integer in the Array**](https://leetcode.com/problems/find-the-kth-largest-integer-in-the-array/)  
https://leetcode.com/problems/find-the-kth-largest-integer-in-the-array/description/  
  
  
  
  
  
  
[**Rotate Array**](https://leetcode.com/problems/rotate-array/)  
https://leetcode.com/problems/rotate-array/description/  
  
  
**At First step we reverse the whole array then in 2 step we reverse the first half 0 to k-1 then in last half k to n-1.**  
  
  
  
**So when k is greater then arr.length 🡪 it will throw out of bound exceptions   
So k = K%N;**  
  
  
  
  
  
  
Reverse Concept :  
  
  
LeetCode Solution:  
  
**FOR LEFT ROTATION : SAME REVERSE NUMBERS CONCEPT IN 2 STEP WE WILL REVERSE ARR.LENGTH-K  
SUCH AS INSTEAD OF 0 TO K-1 FOR RIGHT  
FOR LEFT 0 TO ARR.LENGTH-K-1**   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Jump Game**https://leetcode.com/problems/jump-game/description/  
  
  
Here each element in array represent the maximum jump you can take to reach last index.  
  
  
Here we do step by step we will traverse back first we will check from index 3 we can get index 4 or not so jump is max jump we can also take least jump   


So from 3 we can reach last index so we have updated our last position and check from 2 to 3  
  
Similar way we will check jumps and update the value   
  
  
  
Jump Game 2 description :  
<https://leetcode.com/problems/jump-game-ii/description>  
  
  
Now we need to find minimum number of jumps to reach last position.  
<https://www.youtube.com/watch?v=9kyHYVxL4fw>



Here we will use greedy algorithm and will see how far we can reach   
  
  
So if we choose 4 we can go to index 5 and can check that group in the above pic.  
  


So here we have taken 3 jumps to reach last index.  
  


  
  
<https://github.com/nikoo28/java-solutions/blob/master/src/main/java/leetcode/medium/JumpGameII.java>  
  
