**Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0.**

class Solution {

    public int reverse(int x) {

        long rev= 0;

        while( x !=0){

            rev= rev\*10 + x % 10;

            x=x/10;

        }

        if( rev > Math.pow(2,31) || rev < Math.pow(-2,31))

            return 0;

        return (int)rev;

    }

}

**The complement of an integer is the integer you get when you flip all the 0's to 1's and all the 1's to 0's in its binary representation.**

**For example, The integer 5 is "101" in binary and its complement is "010" which is the integer 2.**

**Given an integer n, return its complement.**

**Example 1:**

**Input: n = 5**

**Output: 2**

**Explanation: 5 is "101" in binary, with complement "010" in binary, which is 2 in base-10.**

class Solution {

    public int bitwiseComplement(int n) {

        if(n==0){

            return 1;

        }

        int mask=0;

        int temp=n;

        while(temp>0){

            mask=(mask << 1) | 1;

            temp=temp >> 1;

        }

        return (~n) & mask;

    }

}

**Power of Two**

**Example 1:**

**Input: n = 1**

**Output: true**

**Explanation: 20 = 1**

**Example 2:**

**Input: n = 16**

**Output: true**

**Explanation: 24 = 16**

**for ex: take the number, n=16**

**n (in bits) -> 1 0 0 0 0 (16 is a power of 2 and thus has only one high bit)**

**n-1 (in bits) -> 0 1 1 1 1 ( n-1 i.e. 15 will make all bits high excpet the 5th bit)**

**n & n-1 -> 0 0 0 0 0 (& operation will make all the bits to 0, thus its power of 2)**

class Solution {

    public boolean isPowerOfTwo(int n) {

        if(n<=0){

            return false;

        }

        return ((n&(n-1))==0);

    }

}

**SEARCH & SORT**

**Book Allocation problem**

Case 1: When no valid answer exists.

If the number of students is greater than the number of books (i.e, M > N), In this case at least 1 student will be left to which no book has been assigned.

Case 2: When a valid answer exists.

The maximum possible answer could be when there is only one student. So, all the book will be assigned to him and the result would be the sum of pages of all the books.

The minimum possible answer could be when number of student is equal to the number of book (i.e, M == N) , In this case all the students will get at most one book. So, the result would be the maximum number of pages among them (i.e, maximum(pages[])).

Below is the approach to solve this problem using Binary Search:

1.Calculate the mid and check if mid number of pages can be assigned to students such that all students will get at least one book.

2.If yes, then update the result and check for the previous search space (end = mid-1)

3.Otherwise, check for the next search space (start = mid+1)

import java.util.\* ;

import java.io.\*;

public class Solution {

    public static boolean isPossible(int[]time, int n, int m, long mid){

        int pageSum=0;

        int studentCount=1;

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

            if(pageSum+time[i] <= mid){

                pageSum+=time[i];

            }

            else{

                studentCount++;

                if(studentCount>n || time[i]>mid){

                    return false;

                }

                pageSum=time[i];

            }

        }

        return true;

    }

    public static long ayushGivesNinjatest(int n, int m, int[] time) {

        // Write your code here.

        //n here no of students ie:Code babbar concept

        //m length of array

        long s=0;

        long sum = 0;

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

            sum+=time[i];

        }

        long e = sum;

        long mid = s+(e-s)/2;

        long ans=-1;

        while(s<=e){

            if(isPossible(time,n,m,mid)){

                ans=mid;

                e=mid-1;

            }

            else{

                s=mid+1;

            }

            mid = s+(e-s)/2;

        }

        return ans;

    }

}

**Painter’s partition :**

import java.util.ArrayList;

public class Solution

{

    public static boolean isPossible(ArrayList<Integer> arr, int k, int mid){

        int painterCount=1;

        int painted=0;

        for(int i=0;i<arr.size();i++){

            if(painted+arr.get(i)<=mid){

                painted+=arr.get(i);

            }

            else{

                painterCount++;

                if(painterCount > k || arr.get(i)>mid){

                    return false;

                }

                painted=arr.get(i);

            }

        }

        return true;

    }

    public static int findLargestMinDistance(ArrayList<Integer> arr, int k)

    {

        int start=0;

        int sum=0;

        for(int i=0;i<arr.size();i++){

            sum+=arr.get(i);

        }

        int ans=-1;

        int end=sum;

        int mid=start+(end-start)/2;

        while(start<=end){

            if(isPossible(arr,k,mid)){

                ans=mid;

                end=mid-1;

            }

            else{

                start=mid+1;

            }

            mid=start+(end-start)/2;

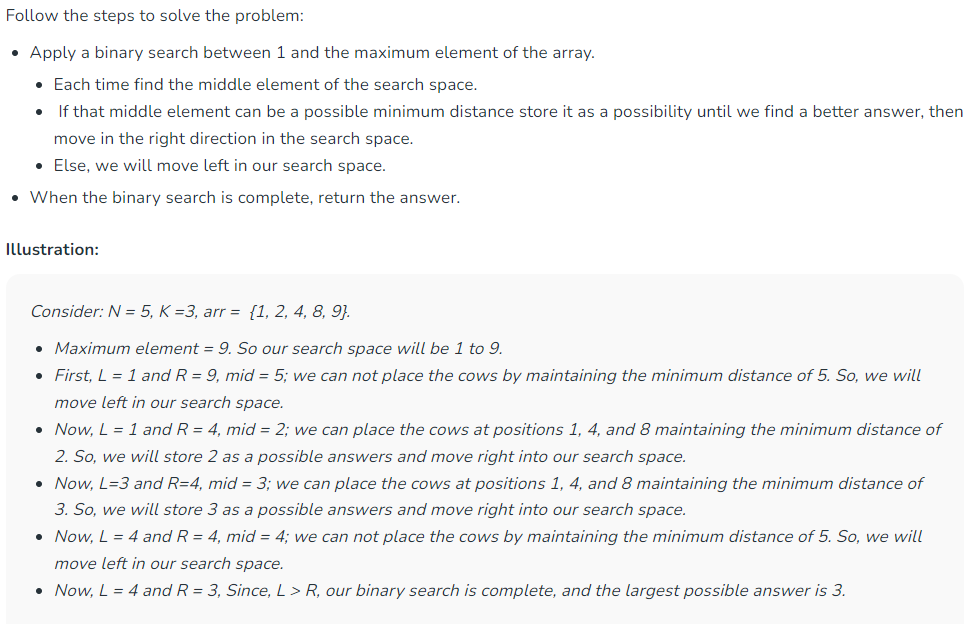
        }

        return ans;

    }

}

**Aggressive cow’s:**



import java.util.Arrays;

public class Solution {

    public static boolean isPossible(int[] stalls, int k, int mid){

        int cowCount=1;

        int lastPos=stalls[0];

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

            if(stalls[i]-lastPos >= mid){

                cowCount++;

                if(cowCount==k){

                    return true;

                }

                lastPos = stalls[i];

            }

        }

        return false;

    }

    public static int aggressiveCows(int []stalls, int k) {

        //    Write your code here.

        Arrays.sort(stalls);

        int s=0;

        int max=stalls[stalls.length-1];

        int e=max;

        int ans=-1;

        int mid = s+(e-s)/2;

        while(s<=e){

            if(isPossible(stalls,k,mid)){

                ans=mid;

                s=mid+1;

            }

            else{

                e=mid-1;

            }

            mid = s+(e-s)/2;

        }

        return ans;

    }

}

**Selection sort :**

import java.util.\*;

import java.io.\*;

public class Solution {

    public static void selectionSort(int arr[], int n) {

        // Write your code here.

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

            int minIndex = i;

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

                if (arr[j] < arr[minIndex]) {

                    minIndex = j;

                }

            }

            int t = arr[minIndex];

            arr[minIndex] = arr[i];

            arr[i] = t;

        }

    }

}

**Bubble sort:**

import java.util.\* ;

import java.io.\*;

public class Solution {

    public static void bubbleSort(int[] arr, int n) {

        // Write your code here.

        for(int i=1;i<n;i++){

            boolean swapped=false;

            //for round 1 to n-1

            for(int j=0;j<n-i;j++){

                if(arr[j]>arr[j+1]){

                    int t = arr[j+1];

                    arr[j+1]=arr[j];

                    arr[j]=t;

                    swapped=true;

                }

            }

            if(swapped==false){

                break;

            }

        }

    }

}

**Insertion sort:**

import java.util.\* ;

import java.io.\*;

public class Solution {

    public static void insertionSort(int n , int[] arr) {

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

            int temp=arr[i];

            int j=i-1;

            for(;j>=0;j--){

                if(arr[j]>temp){

                    arr[j+1]=arr[j];

                }

                else{

                    break;

                }

            }

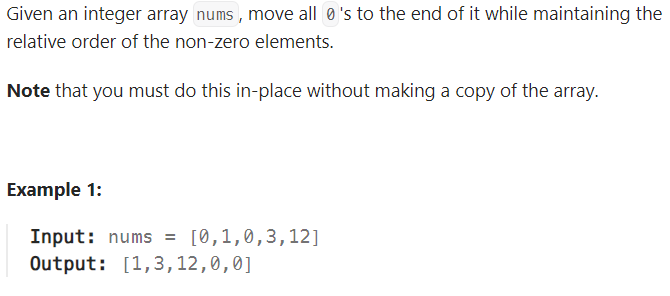
            arr[j+1]=temp;

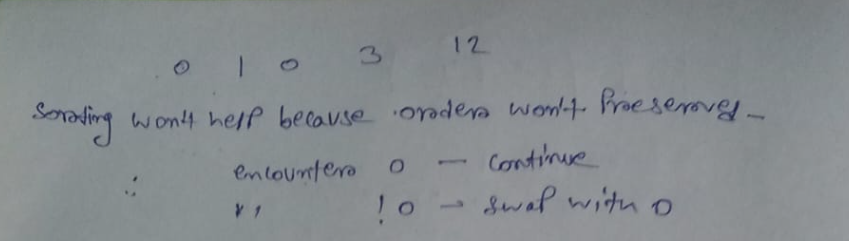
        }

    }

}

**Move zeros:**





class Solution {

    public void moveZeroes(int[] nums) {

        int i=0;

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

            if(nums[j]!=0){

                int temp=nums[j];

                nums[j]=nums[i];

                nums[i]=temp;

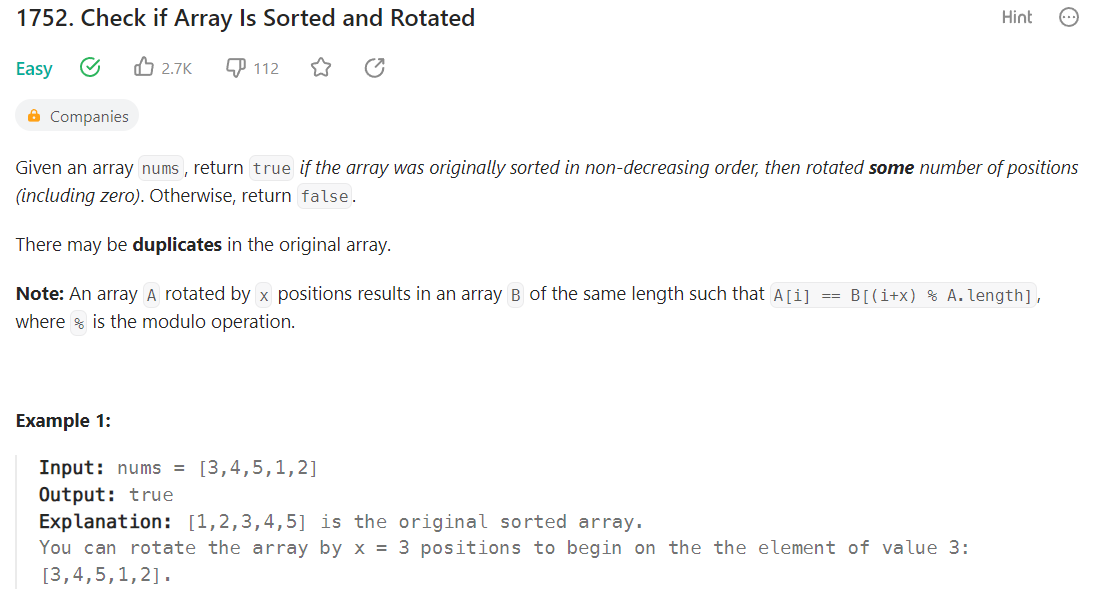
                i++;

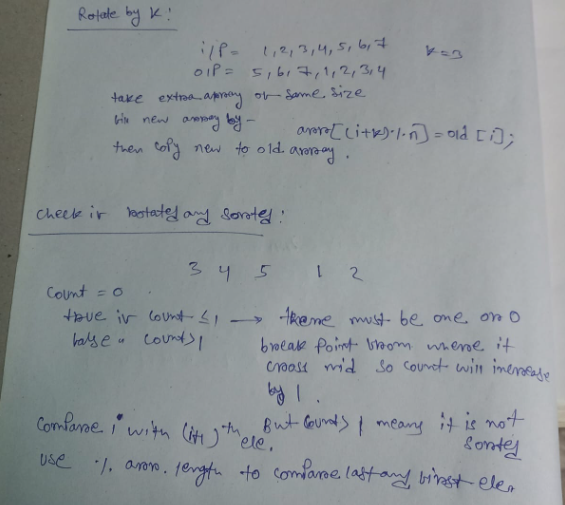
            }

        }

    }

}





class Solution {

    public boolean check(int[] nums) {

        int count=0;

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

            //using %nums.length to compare last and first element

            if(nums[i]>nums[(i+1)%nums.length]){

                count++;

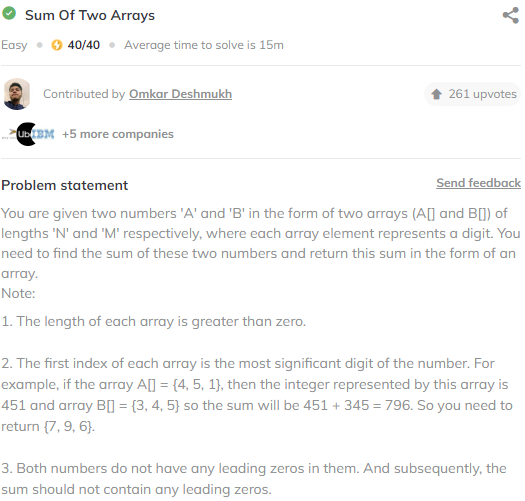
            }

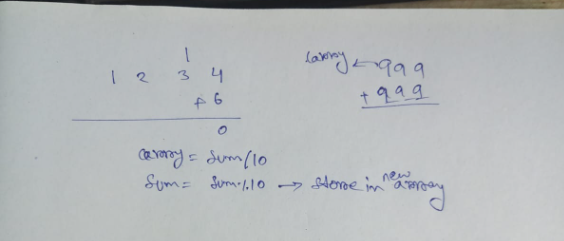
        }

        return (count<=1);

    }

}





import java.util.\* ;

import java.io.\*;

public class Solution {

    public static int[] findArraySum(int[] a, int n, int[] b, int m) {

        int i=n-1;

        int j=m-1;

        int carry=0;

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

        while(i>=0 && j>=0){

            int sum=a[i]+b[j]+carry;

            carry=sum/10;

            int val=sum%10;

            li.add(0,val);

            i--;

            j--;

        }

        //when first array is bigger

        while(i>=0){

            int sum=a[i]+carry;

            carry=sum/10;

            int val=sum%10;

            li.add(0,val);

            i--;

        }

        //2nd array is bigger

        while(j>=0){

            int sum=b[j]+carry;

            carry=sum/10;

            int val=sum%10;

            li.add(0,val);

            j--;

        }

        //two same array but carry left

        if(carry!=0){

            li.add(0,carry);

        }

        int[] arr = new int[li.size()];

        for(int k=0;k<arr.length;k++){

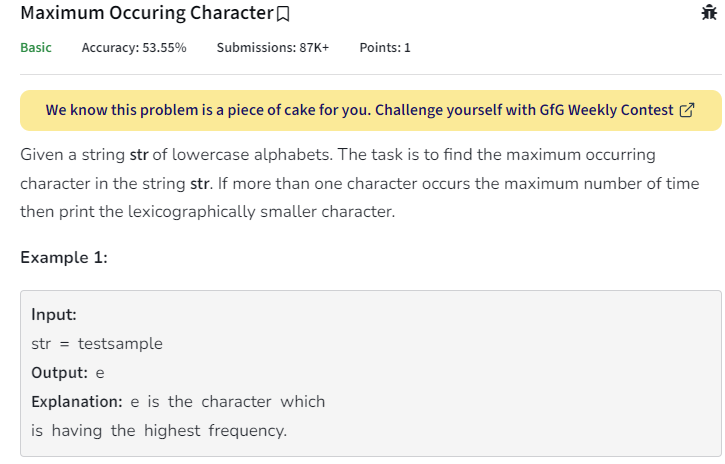
            arr[k]=li.get(k);

        }

        return arr;

    }

}



public static char getMaxOccuringChar(String line)

{

int[] arr=new int[26];

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

char ch = line.charAt(i);

int loc = (int)(ch-'a');

arr[loc]++;

}

int val=-1,maxi=-1;

for(int i=0;i<26;i++){

if(val<arr[i]){

maxi=i;

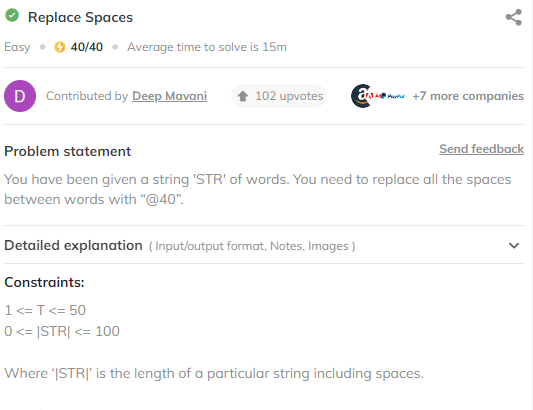
val=arr[i];

}

}

return (char)('a'+maxi);

}



import java.util.\* ;

import java.io.\*;

public class Solution {

    public static StringBuilder replaceSpaces(StringBuilder str) {

        // Write your code here.

        StringBuilder ans =new StringBuilder();

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

            //cant use "" with == as == for char only

            if(str.charAt(i)==' '){

                //as ans is stringBilder, + wont work

                //we have to use append

                ans.append("@40");

            }

            else{

                ans.append(str.charAt(i));

            }

        }

        return ans;

    }