Contains Duplicate

class Solution {

    public boolean containsDuplicate(int[] nums) {

        int n=nums.length;

        Arrays.sort(nums);

        for(int i=1;i<n;i++)

        {

            if(nums[i]==nums[i-1])

            {

                return true;

            }

        }

        return false;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Missing Number

public class Solution {

    public static int missingNumber(int []a, int N) {

        // Write your code here.

        int length=a.length;

        int sum=0;

        for(int i=0;i<length;i++)

        {

            sum+=a[i];

        }

        int total\_sum=(N\*(N+1))/2;

        int res=total\_sum-sum;

        return res;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Two Sum

import java.util.HashMap;

public class Solution {

    public static String read(int n, int []book, int target){

        // Write your code here.

        HashMap<Integer,Integer> hm=new HashMap<>();

        for (int i=0;i<n;i++) {

            int num=book[i];

            int need=target-num;

            if(hm.containsKey(need))

            {

                return "YES";

            }

            hm.put(num,i);

        }

        return "NO";

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How many number are smaller than current number

class Solution {

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

        int n=nums.length,count=0;

        int[] res=new int[n];

        for(int i=0;i<n;i++)

        {

            int ele=nums[i];

            {

                for(int j=0;j<n;j++)

                {

                    if(i!=j && nums[j]<ele)

                    {

                        count++;

                    }

                }

            }

            res[i]=count;

            count=0;

        }

        return res;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Spiral Matrix

class Solution {

    public List<Integer> spiralOrder(int[][] matrix) {

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

        int top=0,left=0;

        int bottom=matrix.length-1,right=matrix[0].length-1;

        while (top<=bottom && left<=right)

        {

            for (int i=left;i<=right;i++)

                res.add(matrix[top][i]);

            top++;

            for (int i=top;i<=bottom;i++)

                res.add(matrix[i][right]);

            right--;

            if (top<=bottom) {

                for (int i=right;i>=left;i--)

                    res.add(matrix[bottom][i]);

                bottom--;

            }

            if (left<=right) {

                for (int i=bottom;i>=top;i--)

                    res.add(matrix[i][left]);

                left++;

            }

        }

        return res;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Best time to buy and sell stock

import java.util.\* ;

import java.io.\*;

import java.util.ArrayList;

public class Solution{

    public static int maximumProfit(ArrayList<Integer> prices){

        // Write your code here.

        int min=Integer.MAX\_VALUE;

        int maxp=0;

        for(int i:prices)

        {

            min=Math.min(min,i);

            int profit=i-min;

            maxp=Math.max(maxp,profit);

        }

        return maxp;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Squares of a sorted array

class Solution {

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

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

        int p1=0;

        int p2=nums.length-1;

        for(int i=nums.length-1;i>=0;i--)

        {

            if(Math.abs(nums[p1])>Math.abs(nums[p2]))

            {

                res[i]=nums[p1]\*nums[p1];

                p1++;

            }

            else

            {

                res[i]=nums[p2]\*nums[p2];

                p2--;

            }

        }

        return res;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3Sum

import java.util.\*;

public class Solution {

    public static List< List < Integer > > triplet(int n, int []arr) {

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

        Arrays.sort(arr);

        for(int i=0;i<n-2;i++)

        {

            if(i>0 && arr[i]==arr[i-1])

            continue;

            int a=i+1,b=n-1;

            while(a<b)

            {

                int sum=arr[i]+arr[a] +arr[b];

                if(sum==0)

                {

                    res.add(Arrays.asList(arr[i],arr[a],arr[b]));

                    while(a<b && arr[a]==arr[a+1]){

                    a++;

                    }

                    while(a<b && arr[b]==arr[b-1]){

                    b--;

                    }

                    a++;

                    b--;

                }

                else if(sum<0){

                a++;

                }

                else{

                b--;

                }

            }

        }

        return res;

    }

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_