**1.Contains Duplicate**

class Solution {

    public boolean containsDuplicate(int[] nums) {

        Arrays.sort(nums);

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

        {

            if(nums[i]==nums[i-1])

            {

                return true;

            }

        }

        return false;

    }

}

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

**2.Missing Number**

public class Solution {

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

        // Write your code here.

        int n=a.length,sum=0;

        for(int i=0;i<n;i++)

        {

            sum+=a[i];

        }

        int ts=(N\*(N+1))/2;

        return ts-sum;

    }

}

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

**4.Two Sum**

public class Solution {

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

        // Write your code here.

        for (int i=0;i<n;i++)

        {

            for (int j=i+1;j<n;j++)

            {

                if (book[i]+book[j]==target)

                return "YES";

            }

        }

        return "NO";

    }

}

**5.How many number are smaller than current number**

class Solution {

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

        int count=0;

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

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

        {

            count=0;

            int num=nums[i];

            {

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

                {

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

                    {

                        count++;

                    }

                }

            }

            ans[i]=count;

        }

        return ans;

    }

}

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

**7.Spiral Matrix**

class Solution {

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

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

        int top=0,left=0;

        int n=matrix.length;

        int m=matrix[0].length;

        int a=n-1,b=m-1;

        while (top<=a && left<=b)

        {

            for (int i=left;i<=b;i++)

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

            top++;

            for (int i=top;i<=a;i++)

                ans.add(matrix[i][b]);

            b--;

            if (top<=a) {

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

                    ans.add(matrix[a][i]);

                a--;

            }

            if (left<=b) {

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

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

                left++;

            }

        }

        return ans;

    }

}

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

**9. 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 n=prices.size();

        int mp=0;

        for(int i=0;i<n-1;i++)

        {

            for(int j=i+1;j<n;j++)

            {

                int p=prices.get(j)-prices.get(i);

                mp=Math.max(mp,p);

            }

        }

        return mp;

    }

}

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

**10.Squares of a sorted array**

class Solution {

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

        int n=nums.length;

        int[] ans=new int[n];

        int a=0;

        int b=n-1;

        for(int i=n-1;i>=0;i--)

        {

            if(Math.abs(nums[a])>Math.abs(nums[b]))

            {

                ans[i]=nums[a]\*nums[a];

                a++;

            }

            else

            {

                ans[i]=nums[b]\*nums[b];

                b--;

            }

        }

        return ans;

    }

}

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

**11.3Sum**

import java.util.\*;

public class Solution {

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

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

        Arrays.sort(arr);

        for(int i=0;i<n-2;i++)

        {

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

            continue;

            int left=i+1,right=n-1;

            while(left<right)

            {

                int sum=arr[i]+arr[left] +arr[right];

                if(sum==0)

                {

                    result.add(Arrays.asList(arr[i],arr[left],arr[right]);

                    while(left<right && arr[left]==arr[left+1])

                    left++;

                    while(left<right && arr[right]==arr[right-1])

                    right--;

                    left++;

                    right--;

                }

                else if(sum<0)

                left++;

                else

                right--;

            }

        }

        return result;

    }

}