Find Peak Element : Leetcode – 162

CODE:

class Solution {

    public int findPeakElement(int[] nums) {

        int i=0;

        int j=nums.length-1;

        while(i<j){

            int mid= i+(j-i)/2;

            if(nums[mid]<nums[mid+1]){

                i=mid+1;

            }

            else{

                j=mid;

            }

        }

        return i;

    }

}

OUTPUT:

A screenshot of a computer

Description automatically generated with medium confidence

Sort Array By Parity : LC : 905

CODE:

class Solution {

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

        // I am using Two pointer method

        // One pointer is at 0 called start

        // Ather one is at the last element called end

        int start = 0;

        int end = nums.length - 1;

        while(start < end){ // Condition to stop the Loop

            // Check if the element at start is even

            // If it is even then it moves to the right

            // If it is not even then it swaps it with the element at the end pointer

            if((nums[start] % 2) == 0){ // You can also do nums[start] & 1 ==0

                start++;

            }else{

                int temp = nums[start];

                nums[start] = nums[end];

                nums[end] = temp;

            }

            // Check if the element at end is odd

            // If it is odd then it moves to the left

            // If it is not odd then it swaps it with the element at the start pointer

            if((nums[end] % 2 ) == 1){ // You can also do nums[start] & 1 !=0

                end--;

            }else{

                int temp = nums[start];

                nums[start] = nums[end];

                nums[end] = temp;

            }

            }

        return nums;

    }

}

OUTPUT :

A screenshot of a computer

Description automatically generated with medium confidence

Find subarray whose sum is equal to given sum

i/p : {10,20,30,40,50,60}, sum = 90

CODE:

class Solution {

    public int subarraySum(int[] nums, int k) {

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

        int count=0;

        int sum=0;

        map.put(0,1);

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

            sum+=nums[i];

            if(map.containsKey(sum-k)){

                count+=map.get(sum-k);

            }

            if(!map.containsKey(sum)){

                map.put(sum,1);

            }else{

                map.put(sum,map.get(sum)+1);

            }

        }

        return count;

    }

}

OUTPUT:

A screenshot of a computer

Description automatically generated with medium confidence

NBonacci Series

- if n = 2 - fibonacci Series

- if n = 3 - triboancci Series

- if n = n - NBonacci Series

i/p : n = 3, nthTerm = 15

o/p : 0,0,1,1,2,4,7,13,24,44,81,149,274,504

CODE:

// public class NBonacciSeries{

//     public static void main(String[] args) {

//         int n = 3;

//         int nthTerm = 15;

//         int[] result = NBonacci(n, nthTerm);

//         for (int num : result) {

//             System.out.print(num + " ");

//         }

//     }

//     public static int[] NBonacci(int n, int nthTerm) {

//         if (n < 2) {

//             System.out.println("Invalid value of n. n should be greater than or equal to 2.");

//             return new int[0];

//         }

//         int[] series = new int[nthTerm];

//         series[n - 1] = 1;

//         for (int i = n; i < nthTerm; i++) {

//             int sum = 0;

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

//                 sum += series[j];

//             }

//             series[i] = sum;

//         }

//         return series;

//     }

// }

public class NBonacciSeries {

    public static void main(String[] args) {

        int n = 3;

        int nthTerm = 15;

        int[] result = NBonacci(n, nthTerm);

        for (int num : result) {

            System.out.print(num + " ");

        }

    }

    public static int[] NBonacci(int n, int nthTerm) {

        if (n < 2) {

            System.out.println("Invalid value of n. n should be greater than or equal to 2.");

            return new int[0];

        }

        int[] series = new int[nthTerm];

        series[n - 1] = 1;

        int sum = 0;

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

            if (i < n) {

                sum += series[i];

            } else {

                sum += series[i] - series[i - n];

            }

            series[i] = sum;

        }

        return series;

    }

}

OUTPUT :



Count distinct element in every window of size k

i/p : arr[] = {1,2,3,4,4,6,6,7,7}, k = 4

o/p : 4,3,3,2,3,2

Hint : Window Sliding with Hashing

CODE:

import java.util.HashMap;

public class Count\_Distinct\_Elements {

    public static void main(String[] args) {

        int[] arr = {1, 2, 3, 4, 4, 6, 6, 7, 7};

        int k = 4;

        int[] result = countDistinctElements(arr, k);

        for (int count : result) {

            System.out.print(count + " ");

        }

    }

    public static int[] countDistinctElements(int[] arr, int k) {

        int n = arr.length;

        int[] result = new int[n - k + 1];

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

        int distinctCount = 0;

        // Count distinct elements in the first window

        for (int i = 0; i < k; i++) {

            countMap.put(arr[i], countMap.getOrDefault(arr[i], 0) + 1);

            if (countMap.get(arr[i]) == 1) {

                distinctCount++;

            }

        }

        result[0] = distinctCount;

        // Slide the window and update distinct count for each window

        for (int i = k; i < n; i++) {

            // Remove the leftmost element from the window

            int leftElem = arr[i - k];

            countMap.put(leftElem, countMap.get(leftElem) - 1);

            if (countMap.get(leftElem) == 0) {

                distinctCount--;

            }

            // Add the rightmost element to the window

            int rightElem = arr[i];

            countMap.put(rightElem, countMap.getOrDefault(rightElem, 0) + 1);

            if (countMap.get(rightElem) == 1) {

                distinctCount++;

            }

            result[i - k + 1] = distinctCount;

        }

        return result;

    }

}

OUTPUT :

