**717821E113**

**Array prefix sum**

**724** Q1. Find Pivot Index Given an array of integers nums, calculate the pivot index of this array.

class Solution {

    public int pivotIndex(int[] nums) {

        int[] l=new int[nums.length];

        int[] r=new int[nums.length];

        l[0]=0;

        r[nums.length-1]=0;

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

            l[i]=l[i-1]+nums[j++];

        }

        for(int i=nums.length-2,j=nums.length-1;i>=0;i--){

            r[i]=r[i+1]+nums[j--];

        }

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

            if(l[i]==r[i]){

                return i;

            }

        }

        return -1;

    }

}

**1413** Q2. Minimum Value to Get Positive Step by Step Sum Given an array of integers nums, you start with an initial positive value startValue. In each iteration, you calculate the step by step sum of startValue plus elements in nums (from left to right). Return the minimum positive value of startValue such that the step by step sum is never less than 1.

class Solution {

    public int minStartValue(int[] nums) {

        int start = 1;

        int current = start;

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

            current += nums[i];

            if (current < 1) {

                start += 1 - current;

                current = 1;

            }

        }

        return start;

    }

}

**1480** Q3.Running Sum of 1d Array

Given an array nums. We define a running sum of an arrayas runningSum[i] = sum(nums[0]…nums[i]).

Return the running sum of nums.

class Solution {

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

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

            nums[i]=nums[i-1]+nums[i];

        }

        return nums;

    }

}

**1732**.Q4 Find the Highest Altitude

There is a biker going on a road trip. The road trip consistsof n + 1 points at different altitudes. The biker starts histrip on point 0 with altitude equal 0.

You are given an integer array gain of length n wheregain[i] is the net gain in altitude between points i and i + 1for all (0 <= i < n). Return the highest altitude of a point.

class Solution {

    public int largestAltitude(int[] gain) {

        int max=0;

        int[]lsum=new int[gain.length+1];

        lsum[0]=0;

        for(int i=0,j=1;i<gain.length;j++,i++){

            lsum[j]=lsum[j-1]+gain[i];

            if(max<=lsum[j]){

                max=lsum[j];

            }

        }

        return max;

    }

}

**209**.Q5.

Minimum Size Subarray Sum

Given an array of positive integers nums and a positive integertarget, return

the

minimal length

of a subarray whose sum isgreater than or equal to target. If there is no such subarray,return 0 instead.

class Solution {

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

        int a = Integer.MAX\_VALUE;

        int left = 0;

        int sum = 0;

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

            sum += nums[right];

            while (sum >= target) {

                sum -= nums[left];

                a = Math.min(a, right - left + 1);

                left++;

            }

        }

        if (a == Integer.MAX\_VALUE) {

            return 0;

        }

        return a;

    }

}

**560.** Q6. Subarray Sum Equals K

Given an array of integers nums and an integer k, return thetotal number of subarrays whose sum equals to k.

A subarray is a contiguous non-empty sequence of elementswithin an array.