QN :

**Q. Find the count of subarray of** longest **length whose sum==k**

**A = {10,5,2,7,1,9,8,7}   k = 15**

**O/p: 1**

Eg. subarrays having sum =15 , {10,5} , {5,2,7,1} , {8,7}

Longest length is 4

And there is only 1 such subarray present in the array A. {5,2,7,1}

Step 1 : Find the maximum length of subarray of K

Step 2 : Count the subarray of sum K and length max

class Solution {

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

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

        int max = Integer.MIN\_VALUE;

        int sum = 0;

        int cnt = 0;

        //Finding subarray of sum K with smallest length

        for(int i = 0 ; i < n; i++){

            sum += nums[i];

            if(sum == k){

                min = Math.min(min , i + 1);

            }else if(map.containsKey(k - nums[i])){

                int length = i - map.get(k - nums[i]);

                min = Math.min(min , length);

            }

            if(map.containsKey(sum) == false)

                 map.put(sum , i);

        }

        //If subarray is not found return -1

         if(max == Integer.MIN\_VALUE){

            return -1;

         }

         //Reinit Map and vars

         map.clear();

         sum = 0;

         //We re getting count so init the map with {0 , 1}

         map.put(0 , 1);

         //Counting subarray of sum k with length min

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

            sum += nums[i];

            if(sum == k & i +1 == max){

                cnt++;

            }else if(map.containsKey(k - nums[i])){

                int length = i - map.get(k - nums[i]);

                if(length == max){

                    cnt++;

                }

            }

            if(map.containsKey(sum) == false)

                 map.put(sum , i);

         }

         return cnt;

    }

}