Given an integer array nums and an integer k, your task is to find the number of subarrays with the largest sum that equals k. The function should return the count of such subarrays.

Input: nums = [1, -1, 5, -2, 3], k = 3

Output: 2

Explanation:

- There are two subarrays with sum 3:

1. [1, -1, 5, -2] has a sum of 3 and length 4.

2. [5, -2] has a sum of 3 and length 2.

- The largest subarray with sum 3 is [1, -1, 5, -2] with a length of 4.

- The number of subarrays with the largest sum equal to `k` is 2 (both [1, -1, 5, -2] and [5, -2] have the same length of 4).

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

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

int n = nums.length;

int sum = 0;

int max = 0;

int cnt = 0;

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

sum += nums[i];

if(sum == k) {

if(max < i + 1){

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

cnt = 1;

}else if (max == i + 1){

cnt++;

}

}

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

if(max < i - map.get(sum - k)){

max = Math.max(max , i - map.get(sum - k));

cnt = 1;

}else if (max == i - map.get(sum - k)){

cnt++;

}

}

if(!map.containsKey(sum)) map.put(sum , i);

}

return cnt;

}

Provide description with sample testcase in LC format and explation for this code