1343. Number of Sub-arrays of Size K and Average Greater

Medium | Topics | Companies | Hint

Given an array of integers `arr` and two integers `k` and `threshold`, return *the number of sub-arrays of size* `k` *and average greater than or equal to* `threshold`.

Example 1:

```
Input: arr = [2,2,2,2,5,5,5,8], k = 3, threshold = 4
Output: 3
Explanation: Sub-arrays [2,5,5],[5,5,5] and [5,5,8] have averages 4, 5 and 6 respectively. All other sub-arrays of size 3 have averages less than 4 (the threshold).
```

Example 2:

```
Input: arr = [11,13,17,23,29,31,7,5,2,3], k = 3, threshold = 5
Output: 6
Explanation: The first 6 sub-arrays of size 3 have averages greater than 5. Note that averages are not integers.
```

Constraints:

- `1 <= arr.length <= 10^5`
- `1 <= arr[i] <= 10^4`
- `1 <= k <= arr.length`
- `0 <= threshold <= 10^4`

Accepted 71.6K | Submissions 105.2K | Acceptance Rate 68.1%

Approach 1: Brute force

$T(n) : O(n^2)$

$S(n) : O(1)$

Approach 2: Sliding window

Here subarray is fixed k so we can just slide the k sized window by one in each iteration and check.

Here we can proceed in two ways.

(1)

→ we can create the window of size k first.

```
for(i:0 to k-1)
    windowSum = windowSum + nums[i]
```

→ now start checking each window by removing the first element of window and adding the element after end of window.

```
for(i:0 to n-k)
{
    if(i != 0)
    {
        windowSum = windowSum - nums[i-1]
        windowSum = windowSum + nums[i+k-1]
    }
    avg = windowSum / k
    if(avg ≥ threshold)
        ans++
}
```

0 1 2 3 4

}

$T(n) : O(k) + O(n-k)$

$S(n) : O(1)$

(2) we can do classic sliding window implementation with increasing, shrinking, sliding all being done in same loop.

```
l = 0, r = 0, ws = 0, sum = 0

while (r < n)
{
    sum = sum + nums[r]

    while (l < n && ws > k)
    {
        sum = sum - nums[l]
        l++
        ws--
    }

    if (ws == k)
    {
        avg = sum / k
        if (avg > threshold)
            ans++
    }
```

$T(n) : O(n) + O(n)$

$S(n) : O(1)$