2302. Count Subarrays With Score Less Than K

Hard Topics Companies Hint

The **score** of an array is defined as the **product** of its sum and its length.

- For example, the score of `[1, 2, 3, 4, 5]` is `(1 + 2 + 3 + 4 + 5) * 5 = 75`.

Given a positive integer array `nums` and an integer `k`, return *the **number of non-empty subarrays** of `nums` whose score is **strictly less** than `k`*.

A **subarray** is a contiguous sequence of elements within an array.

Example 1:

```
Input: nums = [2,1,4,3,5], k = 10
Output: 6
Explanation:
The 6 subarrays having scores less than 10 are:
- [2] with score 2 * 1 = 2.
- [1] with score 1 * 1 = 1.
- [4] with score 4 * 1 = 4.
- [3] with score 3 * 1 = 3.
- [5] with score 5 * 1 = 5.
- [2,1] with score (2 + 1) * 2 = 6.
Note that subarrays such as [1,4] and [4,3,5] are not considered because their scores are 10 and 36
respectively, while we need scores strictly less than 10.
```

Example 2:

```
Input: nums = [1,1,1], k = 5
Output: 5
Explanation:
Every subarray except [1,1,1] has a score less than 5.
[1,1,1] has a score (1 + 1 + 1) * 3 = 9, which is greater than 5.
Thus, there are 5 subarrays having scores less than 5.
```

Constraints:

- `1 <= nums.length <= 10^5`
- `1 <= nums[i] <= 10^5`
- `1 <= k <= 10^15`

Approach: Sliding Window

Intuition

According to the definition of array scores in the question, and given that $nums$ is an array of positive integers, for a subarray $[i,j]$, as the right endpoint j is fixed, the sum of the subarray decreases and its length shortens with the increase of the left endpoint i, so the score of the subarray monotonically decreases. If the score of the subarray $[i,j]$ is less than k, since the score is monotonically decreasing, then the score of the subarray $[p,j], i < p \le j$ is also less than k.

Based on the above properties, we can use the sliding window method to solve the question. Starting from $j = 0$, enumerate the right endpoint of the subarray and maintain a left endpoint i (initially set to 0). For each j:

- Expand window: Add $nums[j]$ to the subarray sum corresponding to the current window $total$.
- Shrink window: If the score of the corresponding subarray in the current window, $total \times (j - i + 1)$, is greater than or equal to k, it indicates that the subarray does not meet the requirements, and therefore, the left endpoint i needs to be moved to the right until the score is less than k.
- Count the number of subarrays: At this moment, the number of subarrays with j as the right endpoint and a score less than k is $j - i + 1$, and it is accumulated into the final result res.

After the enumeration, return the final result res.

Implementation

C++ Java C C# JavaScript Go Python3 Rust TypeScript Copy

```cpp
class Solution {
public:
    long long countSubarrays(vector<int>& nums, long long k) {
        int n = nums.size();
        long long res = 0, total = 0;
        for (int i = 0, j = 0; j < n; j++) {
            total += nums[j];
            while (i <= j && total * (j - i + 1) >= k) {
                total -= nums[i];
                i++;
            }
            res += j - i + 1;
        }
        return res;
    }
};
```

Complexity Analysis

Let n be the length of the $nums$.

- Time complexity: $O(n)$.

We only need to traverse the array once.

- Space complexity: $O(1)$.

Only a few additional variables are needed.