Find The Power Of K Size Subarrays I

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	3254
tags	Sliding Window
language	C++
solved on	@16/11/2024
link	https://leetcode.com/problems/find-the-power-of-k-size-subarrays-i/description/
Completion	☑

Intuition

The goal is to process an input array `nums` and find the elements where a sequence of consecutive increasing numbers of length `k` exists. The result array contains `-1` if no such sequence ends at that position or the last element of the sequence if it does.

Approach

1. **Sliding Window Technique**:
   - Use two pointers (`l` for the left and `r` for the right) to maintain a sliding window of size `k` over the array.
   - Keep a counter (`length`) to track the current number of consecutive increasing elements in the window.
2. **Initialize the Window**:
   - Traverse the first `k` elements to check if they form a sequence of consecutive increasing numbers. Update `length` accordingly.
3. **Iterate Over the Array**:
   - Shift the window one element at a time.
   - Check if the current and previous numbers are consecutive. Adjust the `length` counter.
   - If the current `length` is greater than or equal to `k`, update the result array at the corresponding position.
4. **Output the Result**:
   - Return the result array containing the last element of the sequence or `1` where no such sequence exists.

Complexity

Time Complexity:

- **Initialization**: Traverses the first `k` elements → **O(k)**.
- **Sliding Window**: Traverses the rest of the array → **O(n - k)**.

- **Total**: **O(n)**, where **n** is the size of the array.

Space Complexity:

- Uses a result array of size **O(n - k + 1)**.
- Total space complexity: **O(n - k + 1)**.

Code

```
class Solution {
public:
    vector<int> resultsArray(vector<int>& nums, int k) {
        int n = nums.size();
        if (n == 1) return nums;

        vector<int> answer(n - k + 1, -1);
        int length = 1;

        for (int r = 1; r < k; r++) {
            if (nums[r] == nums[r - 1] + 1) length++;
            else length = 1;
        }
        if (length == k) answer[0] = nums[k - 1];

        for (int l = 1, r = k; r < n; l++, r++) {
            if (nums[r] == nums[r - 1] + 1) length++;
            else length = 1;
            if (length >= k) answer[l] = nums[r];
        }

        return answer;
    }
};
```