Problem 2261: K Divisible Elements Subarrays

Problem Information

Difficulty: Medium
Acceptance Rate: 54.51%
Paid Only: No
Tags: Array, Hash Table, Trie, Rolling Hash, Hash Function, Enumeration

Problem Description

Given an integer array `nums` and two integers `k` and `p`, return _the number of**distinct subarrays,** which have **at most**_ `k` _elements_ that are _divisible by_ `p`.

Two arrays `nums1` and `nums2` are said to be **distinct** if:

* They are of **different** lengths, or * There exists **at least** one index `i` where `nums1[i] != nums2[i]`.

A **subarray** is defined as a **non-empty** contiguous sequence of elements in an array.

Example 1:

Input: nums = [_**2**_ ,3,3,_**2**_ ,_**2**_], k = 2, p = 2 **Output:** 11 **Explanation:** The elements at indices 0, 3, and 4 are divisible by p = 2. The 11 distinct subarrays which have at most k = 2 elements divisible by 2 are: [2], [2,3], [2,3,3], [2,3,3,2], [3], [3,3], [3,3,2], [3,3,2,2], [3,2], [3,2,2], and [2,2]. Note that the subarrays [2] and [3] occur more than once in nums, but they should each be counted only once. The subarray [2,3,3,2,2] should not be counted because it has 3 elements that are divisible by 2.

Example 2:

Input: nums = [1,2,3,4], k = 4, p = 1 **Output:** 10 **Explanation:** All element of nums are divisible by p = 1. Also, every subarray of nums will have at most 4 elements that are divisible by 1. Since all subarrays are distinct, the total number of subarrays satisfying all the constraints is 10.

Constraints:

* `1 <= nums.length <= 200` * `1 <= nums[i], p <= 200` * `1 <= k <= nums.length`

Follow up:

Can you solve this problem in O(n2) time complexity?

Code Snippets

C++:

```
class Solution {
public:
int countDistinct(vector<int>& nums, int k, int p) {

}
};
```

Java:

```
class Solution {
public int countDistinct(int[] nums, int k, int p) {

}
}
```

Python3:

```
class Solution:
def countDistinct(self, nums: List[int], k: int, p: int) -> int:
```