Subarray Sums Divisible by K :-

974. Subarray Sums Divisible by K

Solved

Medium Topics Companies

Given an integer array `nums` and an integer `k`, return *the number of non-empty **subarrays** that have a sum divisible by* `k`.

A **subarray** is a **contiguous** part of an array.

Example 1:

```
Input: nums = [4,5,0,-2,-3,1], k = 5
Output: 7
Explanation: There are 7 subarrays with a sum divisible by k = 5:
[4, 5, 0, -2, -3, 1], [5], [5, 0], [5, 0, -2, -3], [0], [0, -2, -3], [-2, -3]
```

Example 2:

```
Input: nums = [5], k = 9
Output: 0
```

Constraints:

- $1 <= nums.length <= 3 * 10^4$
- $-10^4 <= nums[i] <= 10^4$
- $2 <= k <= 10^4$

Seen this question in a real interview before? 1/5

Yes No

Accepted 393.5K Submissions 708.6K Acceptance Rate 55.5%

eg. I/p : nums = [4,5,0,-2,-3,1], k = 5

[4, 9, 9, 7, 4, 5)

[4, 0, 0, 3, 2, 1)

[4, 4, 4, 2, 4, 0)

a+n %k - a %k %k

4 → 0
4-1
4-2
4-k

1+2+3+1

```cpp
class Solution {
public:
    int subarraysDivByK(vector<int>& a, int k) {
        int n=a.size();
        int ans=0;
        int cur=0;
        unordered_map<int, int> mp;
        for(int i=0;i<n;i++){
            cur+=a[i];
            int p = (cur % k + k) % k;
            if(mp.count(p)){
                ans+=mp[p];
            }
            if(p==0){ans++;}
            mp[p]++;
        }
        return ans;
    }
};
```

T.C. O(n)
S.C. O(n)

C++ Java Copy

```cpp
class Solution {
public:
    int subarraysDivByK(vector<int>& nums, int k) {
        int n = nums.size();
        int prefixMod = 0, result = 0;

        // There are k mod groups 0...k-1.
        vector<int> modGroups(k);
        modGroups[0] = 1;

        for (int num : nums) {
            // Take modulo twice to avoid negative remainders.
            prefixMod = (prefixMod + num % k + k) % k;
            // Add the count of subarrays that have the same remainder as the current
            // one to cancel out the remainders.
            result += modGroups[prefixMod];
            modGroups[prefixMod]++;
        }

        return result;
    }
};
```

Complexity Analysis

Here, n is the length of `nums` and k is the given integer.

- Time complexity: $O(n + k)$
  - We require $O(k)$ time to initialize the `modGroups` array.
  - We also require $O(n)$ time to iterate over all the elements of the `nums` array. The computation of the `prefixSum` and the calculation of the subarrays divisible by `k` take $O(1)$ time for each index of the array.
- Space complexity: $O(k)$
  - We require $O(k)$ space for the `modGroups` array.

Comments (72) Sort by: Best