# 题目

给定一个整数数组 nums 和一个整数 k ，返回其中元素之和可被 k 整除的非空 子数组 的数目。

子数组 是数组中 连续 的部分。

示例 1：

输入：nums = [4,5,0,-2,-3,1], k = 5

输出：7

解释：

有 7 个子数组满足其元素之和可被 k = 5 整除：

[4, 5, 0, -2, -3, 1], [5], [5, 0], [5, 0, -2, -3], [0], [0, -2, -3], [-2, -3]

示例 2:

输入: nums = [5], k = 9

输出: 0

提示:

1 <= nums.length <= 3 \* 104

-104 <= nums[i] <= 104

2 <= k <= 104

# 分析

要解决“找到元素之和可被k整除的非空子数组数目”的问题，核心思路是利用前缀和与同余定理，通过统计前缀和除以k的余数出现次数，高效计算符合条件的子数组数量。

解题思路

1、前缀和与同余定理：

设子数组nums[i..j]的和为sum(i,j)，其值等于前缀和prefix[j] - prefix[i-1]（prefix[0] = 0，prefix[j]表示前j个元素的和）。

若sum(i,j)能被k整除，则(prefix[j] - prefix[i-1]) % k == 0，根据同余定理，等价于prefix[j] % k == prefix[i-1] % k。

因此，问题转化为：统计所有前缀和中，余数相同的对数（每对相同余数对应一个符合条件的子数组）。

2、处理负余数：

由于数组元素可正可负，前缀和可能为负数，导致余数为负。需将负余数转换为正余数（如-1 % 5 = 4），确保余数在[0, k-1]范围内，统一统计标准。

3、哈希表统计余数频次：

- 用哈希表count存储每个余数出现的次数，初始时count[0] = 1（对应prefix[0] = 0的情况）。

- 遍历数组，计算当前前缀和的余数rem，累加count[rem]（之前出现过的相同余数的次数，即当前可形成的符合条件的子数组数），再将rem的频次加1。

代码：

class Solution {

public:

int subarraysDivByK(vector<int>& nums, int k) {

unordered\_map<int, int> count;

count[0] = 1; // 初始化：前缀和为0的余数0出现1次（对应prefix[0]）

int prefix\_sum = 0; // 前缀和

int result = 0; // 符合条件的子数组数目

for (int num : nums) {

// 累加当前元素，更新前缀和

prefix\_sum += num;

// 计算当前前缀和的余数（处理负余数，转换为非负）

int rem = (prefix\_sum % k + k) % k;

// 累加之前出现过的相同余数的次数（这些次数对应可形成的有效子数组）

result += count[rem];

// 更新当前余数的频次

count[rem]++;

}

return result;

}

};

代码解释

1、初始化哈希表：

count[0] = 1是因为前缀和prefix[0] = 0（虚拟的前缀和，用于处理从数组起始位置开始的子数组，如nums[0..j]的和若能被k整除，则prefix[j] % k == 0，此时count[0]提供初始计数）。

2、前缀和与余数计算：

- 遍历数组时，prefix\_sum累加当前元素，得到前i个元素的和（prefix[i]）。

- 计算余数rem时，通过(prefix\_sum % k + k) % k确保结果为非负（例如，当prefix\_sum为-1，k=5时，(-1 % 5) = -1，加k后为4，再模k仍为4，即正确的正余数）。

3、统计有效子数组：

- 每次得到余数rem后，count[rem]表示之前有多少个前缀和的余数也是rem，这些前缀和对应的下标i-1与当前下标i形成的子数组nums[i..j]的和可被k整除，因此将count[rem]累加到result。

- 最后更新count[rem]，将当前余数的频次加1，供后续元素统计使用。

复杂度分析

- 时间复杂度：O(n)，其中n是数组长度。仅遍历数组一次，哈希表的插入和查询操作均为O(1)（平均情况）。

- 空间复杂度：O(min(n, k))。哈希表存储的余数种类最多为k（余数范围是0~k-1），因此空间复杂度受k和n的较小值限制。