# 题目

给你一个下标从 0 开始长度为 n 的整数数组 nums 和一个整数 k ，请你返回满足 0 <= i < j < n ，nums[i] == nums[j] 且 (i \* j) 能被 k 整除的数对 (i, j) 的数目。

示例 1：

输入：nums = [3,1,2,2,2,1,3], k = 2

输出：4

解释：

总共有 4 对数符合所有要求：

- nums[0] == nums[6] 且 0 \* 6 == 0 ，能被 2 整除。

- nums[2] == nums[3] 且 2 \* 3 == 6 ，能被 2 整除。

- nums[2] == nums[4] 且 2 \* 4 == 8 ，能被 2 整除。

- nums[3] == nums[4] 且 3 \* 4 == 12 ，能被 2 整除。

示例 2：

输入：nums = [1,2,3,4], k = 1

输出：0

解释：由于数组中没有重复数值，所以没有数对 (i,j) 符合所有要求。

提示：

1 <= nums.length <= 100

1 <= nums[i], k <= 100

# 分析

## 方法一：遍历

思路：

我们用n表示数组nums 的长度。为了统计符合要求数对数量，我们可以使用两层循环遍历所有满足 0≤i<j<n的数对 (i,j)，并逐个检查i×jmodk 是否等于 0，且nums[i] 是否等于nums[j]。

与此同时，我们用res 统计符合要求的数对数量，如果某个数对 (i,j) 符合要求，则我们将res 加上 1。最终，我们返回 res 作为符合要求的数对个数。

代码：

class Solution {

public:

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

int n = nums.size();

int res = 0; // 符合要求数对个数

for (int i = 0; i < n - 1; ++i) {

for (int j = i + 1; j < n; ++j) {

if ((i \* j) % k == 0 && nums[i] == nums[j]) {

++res;

}

}

}

return res;

}

};

复杂度分析

时间复杂度：O(n^2)，其中n为nums数组的长度。即为遍历数对并统计符合要求个数的时间复杂度。

空间复杂度：O(1)。