<https://leetcode.com/problems/count-array-pairs-divisible-by-k/description/>

Given a **0-indexed** integer array nums of length n and an integer k, return **the number of pairs** (i, j) **such that:**

0 <= i < j <= n - 1 **and**

nums[i] \* nums[j] **is divisible by** k

**Example 1:**

**Input:** nums = [1,2,3,4,5], k = 2

**Output:** 7

**Explanation:**

The 7 pairs of indices whose corresponding products are divisible by 2 are(0, 1), (0, 3), (1, 2), (1, 3), (1, 4), (2, 3), and (3, 4).

Their products are 2, 4, 6, 8, 10, 12, and 20 respectively.

Other pairs such as (0, 2) and (2, 4) have products 3 and 15 respectively, which are not divisible by 2.

**Example 2:**

**Input:** nums = [1,2,3,4], k = 5

**Output:** 0

**Explanation:** There does not exist any pair of indices whose corresponding product is divisible by 5.

**Constraints:**

1 <= nums.length <= 10^5

1 <= nums[i], k <= 10^5

**Attempt 1: 2024-01-17**

**Solution 1: Brute Force (10 min, TLE 93/115)**

class Solution {

    public long countPairs(int[] nums, int k) {

        int n = nums.length;

        int count = 0;

        for(int i = 0; i < n; i++) {

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

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

                    count++;

                }

            }

        }

        return count;

    }

}

**Solution 2: Greatest Common Dividsor - GCD (10 min)**

**Note: L2176.Count Equal and Divisible Pairs in an Array do the same job**

class Solution {

    public long countPairs(int[] nums, int k) {

        Map<Long, Long> map = new HashMap<>();

        long result = 0;

        for(int num : nums) {

            long gcd\_a = gcd(num, (long)k);

            for(long gcd\_b : map.keySet()) {

                if(gcd\_a \* gcd\_b % k == 0) {

                    result += map.get(gcd\_b);

                }

            }

            map.put(gcd\_a, (long)map.getOrDefault(gcd\_a, (long)0) + 1);

        }

        return result;

    }

    private long gcd(long x, long y) {

        if(y == 0) {

            return x;

        }

        return gcd(y, x % y);

    }

}

Time Complexity: O(N\*sqrt(k))

Space Complexity: O(N)

**Refer to**

<https://leetcode.com/problems/count-array-pairs-divisible-by-k/solutions/1785906/how-gcd-a-k-gcd-b-k-k-0-explained-with-example/>

**Intuition**

Let's first understand this property:

**If (a\*b)%k == 0, then gcd(a,k) \* gcd(b,k) % k is also 0**

Let’s assume two numbers 504 and 819. Their prime factorization can be written as:

504 = (2^3) \* (3^2) \* 7

819 = (3^2) \* 7 \* 13

Now gcd(504,819) = 63 and 63 = (3^2) \* 7

**gcd(a,b) is the multiplication of common prime factors of a and b.**

Coming back to the statement

How gcd(a,k) \* gcd(b,k) % k is 0 ?

For any number to be divisble by k it need to have atleast all the prime factors of k.

gcd(a,k) = Multiplication of all prime factors of k available in a.

and

gcd(b,k) = Multiplication of all prime factors of k available in b.

If gcd(a,k) \* gcd(b,k) % k is 0, it means some of the prime factors of k are contributed by a and some of the prime factors of k are contributed by b and thier multiplication has all the prime factors of k which means a\*b is divisble by k.

We dont care about prime factors of a or b which are not prime factors of k because they will not help us in making a\*b divisible by k.

Example:

Let k=84, a=24, b=273

k = 84 = (2^2) \* 3 \* 7

a = 24 = (2^3) \* 3

b = 273 = 3 \* 7 \* 13

gcd(a,k) = (2^2) \* 3 (Common prime factors of a and k)

gcd(b,k) = 7 \* 3 (Common prime factors of b and k)

gcd(a,k) \* gcd(b,k) = (2^2) \* (3^2) \* 7

which has all prime factors of k thus a\*b is divisble by k.

**Now the solution:**

As compared to checking for every pair, if we check for gcd of every number with k then the operations will be less because the number of prime factors of a number will be less.

Code:

typedef long long ll;

class Solution {

public:

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

unordered\_map<ll, ll> gcdCount;

ll ans = 0;

for (ll i = 0; i < nums.size(); ++i)

{

ll currgcd = \_\_gcd(nums[i], k);

for (auto &[gc\_d, count] : gcdCount)

if ((currgcd \* gc\_d) % k == 0)

ans += count;

gcdCount[currgcd]++;

}

return ans;

}

};

**Java version (convert by chatGPT)**

class Solution {

    public long countPairs(int[] nums, int k) {

        HashMap<Long, Long> gcdCount = new HashMap<>();

        long ans = 0;

        for (int i = 0; i < nums.length; ++i) {

            long currgcd = gcd(nums[i], k);

            for (HashMap.Entry<Long, Long> entry : gcdCount.entrySet()) {

                long gc\_d = entry.getKey();

                long count = entry.getValue();

                if ((currgcd \* gc\_d) % k == 0) {

                    ans += count;

                }

            }

            gcdCount.put(currgcd, gcdCount.getOrDefault(currgcd, 0L) + 1);

        }

        return ans;

    }

    private long gcd(long a, long b) {

        return b == 0 ? a : gcd(b, a % b);

    }

}