2183. Count Array Pairs Divisible by K

Leetcode Link

Problem Description

Given an array `nums` and an integer `k`, we have to find the number of pairs `(i,j)` such that the product of `nums[i]` and `nums[j]` is divisible by `k`. Note that `k` can be as large as 1e5.

Here's a high-level approach to solving this problem:

1. Calculate the greatest common divisor (gcd) of each value in `nums` with `k`.
2. Then, for each pair `(nums[i], nums[j])`, check if their gcd values multiplied together are divisible by `k`. If yes, add them to the count.

Let's walk through an example to better understand the approach.

Example

Suppose nums = [1, 2, 3, 4, 5] and k = 6. We first find the gcd values for each element in the array `nums` with k.

1 × 6 = 6, gcd(1, 6) = 1 2 × 3 = 6, gcd(2, 6) = 2 3 × 2 = 6, gcd(3, 6) = 3 4 × 1.5 = 6, gcd(4, 6) = 2 5 × 1.2 = 6, gcd(5, 6) = 1

Now the gcd values are [1, 2, 3, 2, 1].

We go through each pair of gcd values and check if their product is divisible by `k`. In our example:

(1, 2) - No (1, 3) - No (1, 2) - No (1, 1) - No (2, 3) - Yes (2, 2) - No (2, 1) - No (3, 2) - Yes (3, 1) - No (2, 1) - No

We find two pairs that are divisible by `k`: (2, 3) and (3, 2). So the final answer is 2.

Solution

Python

```python
python
from math import gcd
from typing import List


class Solution:
    def countPairs(self, nums: List[int], k: int) -> int:
        ans = 0
        gcds = {}

        for num in nums:
            gcd_num = gcd(num, k)
            for gcd_val, count in gcds.items():
                if gcd_num * gcd_val % k == 0:
                    ans += count
            gcds[gcd_num] = gcds.get(gcd_num, 0) + 1

        return ans
```

Java

```java
java
import java.util.HashMap;
import java.util.Map;

class Solution {
    public long countPairs(int[] nums, int k) {
        long ans = 0;
        Map<Integer, Integer> gcds = new HashMap<>();

        for (int num : nums) {
            int gcdNum = gcd(num, k);
            for (Map.Entry<Integer, Integer> entry : gcds.entrySet()) {
                if (gcdNum * entry.getKey() % k == 0) {
                    ans += entry.getValue();
                }
            }
            gcds.put(gcdNum, gcds.getOrDefault(gcdNum, 0) + 1);
        }

        return ans;
    }

    private int gcd(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return gcd(b, a % b);
        }
    }
}
```

JavaScript

```javascript
javascript
class Solution {
    countPairs(nums, k) {
        let ans = 0;
        const gcds = new Map();

        for (const num of nums) {
            const gcdNum = this.gcd(num, k);
            for (const [gcdVal, count] of gcds.entries()) {
                if (gcdNum * gcdVal % k === 0) {
                    ans += count;
                }
            }
            gcds.set(gcdNum, (gcds.get(gcdNum) || 0) + 1);
        }

        return ans;
    }

    gcd(a, b) {
        if (b === 0) {
            return a;
        } else {
            return this.gcd(b, a % b);
        }
    }
}
```

C++

```cpp
cpp
#include <unordered_map>

class Solution {
public:
    long long countPairs(vector<int>& nums, int k) {
        long ans = 0;
        unordered_map<int, int> gcds;

        for (const int num : nums) {
            const int gcdNum = gcd(num, k);
            for (const auto& [gcdVal, count] : gcds) {
                if (gcdNum * gcdVal % k == 0) {
                    ans += count;
                }
            }
            ++gcds[gcdNum];
        }

        return ans;
    }

private:
    int gcd(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return gcd(b, a % b);
        }
    }
};
```

C#

```csharp
csharp
using System.Collections.Generic;

public class Solution {
    public long CountPairs(int[] nums, int k) {
        long ans = 0;
        Dictionary<int, int> gcds = new Dictionary<int, int>();

        foreach (int num in nums) {
            int gcdNum = GCD(num, k);
            foreach (KeyValuePair<int, int> entry in gcds) {
                if (gcdNum * entry.Key % k == 0) {
                    ans += entry.Value;
                }
            }
            if (gcds.ContainsKey(gcdNum)) {
                gcds[gcdNum]++;
            } else {
                gcds[gcdNum] = 1;
            }
        }

        return ans;
    }

    private int GCD(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return GCD(b, a % b);
        }
    }
}
```

In conclusion, the above solutions follow the high-level approach explained earlier and find the count of pairs such that their product is divisible by `k`. Each solution utilizes the greatest common divisor (gcd) function to calculate the gcd values, iterate through each pair of gcd values, and check if their product is divisible by `k`. The implementation is provided in Python, Java, JavaScript, C++, and C#.