SURYA KIRAN'S POSTS

A Dance with Mobius Function

Surya Kiran

In this post I am going to share my little knowledge on how to solve some problems regarding Mobius Inversion Formula. I chose this topic because it has a lot of varieties of problems (mostly categorized as medium or hard), but has very few blogs explaining the theory behind. I have tried to present a generalized approach in solving the problems in this topic.

Prerequisites:
1. High-School mathematics
2. Basics of computer programming like loops, arrays
3. Basic Sieve techniques

Beware! This post contains a lot of things which are not proved in here, but are left as an exercise. So wherever it is mentioned as to prove by yourself, I recommend you to stop there and try to prove those things and then only move forward, so that there won't be any confusion.

I am going to explain how to solve the problems SPOJ.com - Problem GCDEX   and http://www.codechef.com/LTIME13/...   below. But, before going into the actual solutions, first, we need to know a little bit about what the mobius inversion formula is, what a mobius function is, how to manipulate the summations, etc. This topic has a lot of fascinating theory behind it. If you are well equipped with the theory part, then solving any algorithmic problem of this topic will be a cake-walk with less than 5 to 6 steps of reduction, which might look obvious.

I strongly recommend reading this Page on berkeley.edu   before proceeding as it will make things extremely easy to understand. It has awesome theory on multiplicative functions and many theorems with proofs. There are also many problems with hints and have solutions for a few. Some definitions and literature from the article is explained below for better understanding.

Throughout this topic we are only going to deal with integers and multiplicative functions.

An arithmetic function f(n) : N → C is multiplicative if for any relatively prime n, m ∈ N: $f(mn) = f(m)f(n)$.

From now on we shall write the prime decomposition of n ∈ N as
$n = p_1^{\alpha_1} * p_2^{\alpha_2} * p_3^{\alpha_3} * \ldots * p_r^{\alpha_r}$

Let us define some functions :

1) **e(n)**, the Dirichlet identity function
$e(n) = 1$ for n = 1,
$e(n) = 0$ for n > 1;

2. **I(n)** = 1 for all n ∈ N;

3. **id(n)** = n for all n ∈ N;

4. **μ(n)**, the **Mobius function**;
$\mu(n) = 1$ if $n = 1$,
$\mu(n) = 0$ if n is not square-free, i.e $\alpha_i > 1$ for some prime factor p_i.

5. For any function $f(n)$, lets define the "**sum function**", $S_f(n)$ as the sum of f(d) for all factors d of n, i.e,

$S_f(n) = \sum_{d|n} f(d)$, $(a|b)$ means that a is a factor of b, or simply a divides b.

For example $S_f(6) = f(1) + f(2) + f(3) + f(6)$

6. Euler totient function, φ(n), is the number of the integers **d** between 1 and **n,** which are relatively prime to **n**, gcd(d, n) = 1. (set φ(1) = 1)

Dirichlet Convolution:

For any two functions f(x) and g(x), the dirichlet convolution is defined as

$fOg(n) = \sum_{d1*d2=n} f(d1) * g(d2)$
or
$fOg(n) = \sum_{d|n} f(d) * g(n/d)$

The Dirichlet product, O, is commutative, and associative:

$fOg(n) = gOf(n)$

$(fOg)Oh(n) = fO(gOh)(n)$

You can prove these by yourself easily.

Now some interesting observations:

$S_f(n) = \sum_{d|n} f(d) = \sum_{d|n} f(d) * I(n/d)$

Or simply,

$S_f(n) = fOI(n)$, i.e, Dirichlet convolution of the functions f(x) and I(x) (the function I(n) is defined above)

$S_\phi(n) = \sum_{d|n} \phi(d) = id(n)$ Proof left as an exercise.

$S_\mu(n) = \sum_{d|n} \mu(d) = e(n)$ Proof left as an exercise.

The above formula can be re-written as $IO\mu(n) = \mu OI(n) = e(n)$.

$fOe(n) = \sum_{d|n} f(n/d) * e(d) = f(n)$, because e(d) is equal to 1 only when d = 1.

Now, $S_f O\mu(n) = (fOI)O\mu(n) = fO(IO\mu)(n) = fOe(n) = f(n)$.

Or Simply

$f(n) = \sum_{d|n} S_f(d) * \mu(n/d)$. This is called the famous Möbius inversion formula .

Now lets see how this concept is useful in solving problems.

Problem :

Find $G = \sum_{i=1}^{n} \sum_{j=i+1}^{n} gcd(i, j)$ -- (Eq.1), gcd(i, j) is the Greatest common divisor of i and j.

Solution:

The brute force approach: Iterate through all i and j, and find the sum of their gcds. This will take $O(N^2 * k)$ time, where k is the maximum number of steps needed to calculate gcd of two numbers.

Using Mobius Inversion Formula:

It is easy to see that the gcd of any pair (i, j) : (1 <= i < j <= n) will be between 1 to n.

So the summation $G = \sum_{i=1}^{n} \sum_{j=i+1}^{n} gcd(i, j)$ can be re-written as

gcd(i, j) = g implies, gcd(i/g, j/g) = 1. The relation between i and j is that i/g and j/g should be co-prime.

Lets try something else here.

The equation 2 can be re-written as

$$G = \sum_{g=1}^{n} h(g) * cnt[g]$$

Where $h(g) = g$ in this question.

Now lets try to write h(n) as sum function of some function f(n), i.e,
$$S_f(n) = h(n) = \sum_{d|n} f(d)$$

f(n) can be found by using mobius inversion formula, $f(n) = \sum_{d|n} h(d) * \mu(n/d)$.

The Mobius function μ(n) can be found using a similar sieve which we use to find the prime numbers, see this Sieve Methods : Prime, Divisor, Euler Phi etc. - Codeforces for some help.

If we know h(n) and μ(n), f(n) can be found by using a sieve like this:

```
1  for (int i = 1; i <= N; i++) {
2      for (int j = i; j <= N; j += i) {
3          f[j] += h[i] * μ[j/i];
4      }
5  }
```

The above code runs in O(N * logN) time. The reason is, for each i, the inner for loop runs N/i times (i.e, number of multiples of i), and as $\sum_{i=1}^{n} N/i = O(N * logN)$.

In this problem, as $h(g) = g$, $f(n) = \sum_{d|n} d * \mu(n/d)$. simplifying this we get $f(n) = \phi(n)$. This is left as an exercise.

Now G in equation 2 will become

$$G = \sum_{g=1}^{n} \left(\sum_{d|g} f(d)\right) * cnt[g] \text{--(Eq.3)}$$

which can be rewritten as

$$G = \sum_{d=1}^{n} f(d) * \left(\sum_{g:d|g} cnt[g]\right) \text{--(Eq.4)}$$

Let us define cnt2[d] as $cnt2[d] = \sum_{g:d|g} cnt[g]$

What exactly does cnt2[d] represent?

It is the count of number of pairs (i, j) such that gcd(i, j) is a multiple of d. Here the relationship between i and j is simple. There is no such relation between i and j that (i/d and j/d) should be coprime. It is only required that gcd(i, j) is a multiple of d, which means both i and j should be multiples of d. This is necessary and sufficient condition. Number of multiples of d between 1 and n is equal to n/d. So number of pairs (i, j) such that (i < j) and both i and j are multiples of d, will be equal to (n/d) * (n/d - 1) / 2 (Why?)

So our equation 4 becomes

$$G = \sum_{d=1}^{n} f(d) * (n/d) * (n/d - 1)/2.$$

One important observation we need to make here is that the value of (n/d) will take only O(sqrt(n)) distinct values. It is not very hard to prove this.

So we can iterate through all distinct values of (n/d) and add the term $(n/d) * (n/d - 1)/2 * \sum_{k:n/k=n/d} f(k)$ to the final answer.

In the term $\sum_{k:n/k=n/d} f(k)$ all the k's will be consecutive, so if we maintain prefix sums of f(i), we can find this term in O(1).

Complexity of this solution is $O(N * log(n) + sqrt(n) * T)$ T = Number of Test Cases.

1. Let the problem be to find $G = \sum_{i=1} \sum_{j=i+1} h(gcd(i,j))$, here h(n) should be a multiplicative function.

For example if the problem was to find $G = \sum_{i=1}^{n} \sum_{j=i+1}^{n} gcd(i,j)^3$, then the function h() will be $h(n) = n^3$.

2. Re-write the equation like this: $G = \sum_{g=1}^{n} h(g) * cnt[g]$
Where cnt[g] = number of pairs (i, j) such that gcd(i, j) = g, (1 <= i < j <= n).

3. Find the function f(n), such that $h(n) = \sum_{d|n} f(d)$. This can be done using mobius inversion formula and sieve.

4. Rewrite the equation in second step like this,
$G = \sum_{d=1}^{n} f(d) * cnt2[d]$.

5. Iterate through the O(sqrt(n)) distinct values of cnt2[d] and find the answer in O(sqrt(n)) time.

Using the knowledge of above method lets solve the problem
http://www.codechef.com/LTIME13/...

1. Here the question is to find the number of triplets (i, j, k) such that
1 <= i < j < k <= n, gcd(a[i], a[j], a[k]) = 1.

Or simply,

Find $G = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} e(gcd(a[i], a[j], a[k]))$.

Here the function e(n) is the dirichlet identity function, i.e, e(n) = 1 if n = 1, e(n) = 0, if n > 1, which means that 1 is added to the final answer if the gcd(a[i],a[j],a[k]) is equal to 1.

2. Now rewrite the equation like this $G = \sum_{g=1}^{maxg} h(g) * cnt[g]$.

Here cnt[g] = Number of triplets (i, j, k) such that gcd(a[i], a[j], a[k]) = g, maxg will be equal to 10^6 in this question.

3. Find the function f(n), such that $h(n) = \sum_{d|n} f(d)$. This can be done using mobius inversion formula and sieve. In this problem f(n) will be simply equal to μ(n).

Then the problem becomes to find $G = \sum_{g=1}^{maxg} (\sum_{d:d|g} f(d)) * cnt[g]$.

4. We can rewrite the equation in second step like this,

$G = \sum_{d=1}^{n} f(d) * cnt2[d]$.

Here cnt2[d] is number of triplets (i, j, k) such that gcd(a[i], a[j], a[k]) is a multiple of d. If dp[x] is the number of i such that a[i] is a multiple of x, then cnt2[d] will be simply Choose(dp[x], 3). Choose(n, r) = number of ways to choose a combination of r items among n distinct items.

The values of dp[x] can be found using a sieve.

5. Iterate through all d and find the $G = \sum_{d=1}^{n} f(d) * cnt2[d]$.

Complexity: O(N * logN) for the sieve.

Finally, Here are some exercises for you to try

http://www.codechef.com/JULY10/p...
http://www.codechef.com/COOK29/p...
https://www.hackerrank.com/chall...
http://www.spoj.com/problems/LCM...
http://www.spoj.com/problems/GCD...