Competitive Programming

From Problem 2 Solution in O(1)

Number Theory

Totient and Möbius Functions

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Euler's totient function

- φ(n), the Phi Function
  - Count integers i < n such that gcd(i, n) = 1
  - gcd(a, b) = 1 => then coprimes: gcd(5, 7), gcd(4, 9)
  - gcd(prime, i) = 1 for i < prime
- φ(10) = 4 => 1, 3, 7, 9
- φ(5) = 4 => 1, 2, 3, 4 …. φ(prime) = prime-1
- If a, b, c are pairwise coprimes, then
  - φ(a*b*c) = φ(a) * φ(b) * φ(c)
- if k >= 1

$$\varphi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1) = p^k\left(1 - \frac{1}{p}\right).$$

Euler's totient numbers

- Online Sequence
- $\varphi(n)$ = 1, 1, 2, 2, 4, 2, 6, 4, 6, 4, 10, 4, 12, 6, 8, 8, 16, 6, 18, 8, 12, 10, 22, 8, 20, 12, 18, 12, 28, 8, 30, 16, 20, 16, 24, 12, 36, 18, 24, 16, 40, 12
  - $\varphi(1) = \varphi(2) = 1$. $\varphi(5) = 4$
  - $\varphi(n)$ is even for $n > 2$
  - $\mathrm{sqrt}(n) <= \varphi(n) <= n - \mathrm{sqrt}(n)$: Except 2, 6
  - $\varphi(n^k) = n^{k-1} * \varphi(n)$
  - $n = \sum_i \varphi(di)$ where d are the divisors of n

Menon's identity

$$\sum_{\substack{1 \le k \le n \\ \gcd(k,n)=1}} \gcd(k-1, n) = \varphi(n) d(n),$$

d(n): # of n divisors

The identity is not important in competitive :)

Phi Code: Brute Force

```
int phi(int n) {
    int ret = 0;
    for (int i = 1; i <= n; i++)
        if (gcd(i, n) == 1)
            ret++;
    return ret;
}
```

Phi Code: Prime Factorization

```
// Factorize and use fact p^(n-1)*(p-1)
int phi(int n)
{
    int p_to_k, relative_primes = 1;

    for (int i = 2, d = 1; i*i <= n ; i += d, d = 2) {
        if(!(n % i)) {
            p_to_k = 1;
            while (!(n % i))
                p_to_k *= i, n /= i;
            relative_primes *= (p_to_k/i) * (i-1);
        }
    }

    if (n != 1)
        relative_primes *= (n-1);
    return relative_primes;
}
```

Phi Code: Range Generator

```
void phi_generator()        // using seive
{
    const int MAX = 1000000;
    char primes[MAX];
    int phi[MAX];

    memset(primes, 1, sizeof(primes));

    for (int k = 0; k < MAX; ++k)
        phi[k] = 1;

    for (int i = 2; i <= MAX; ++i) {
        if (primes[i]) {
            phi[i] = i-1;   // ph(prime) = p-1

            for (int j = i * 2; j <= MAX; j += i) {
                primes[j] = 0;
                int n = j, pow = 1;
                while (!(n % i))
                    pow *= i, n /= i;
                phi[j] *= (pow/i) * (i-1);
            }
        }
    }
}
```

Phi Factorial Code

- Can you prove?

```
// phi(N!) = (N is prime ? N-1 : N ) * phi((N-1)!)
ll phi_factn(int n)
{
    ll ret = 1;
    for (int i = 2; i <= n; ++i)
        ret = ret * (isprime(i) ? i-1 : i);
    return ret;
}
```

Square-free integer

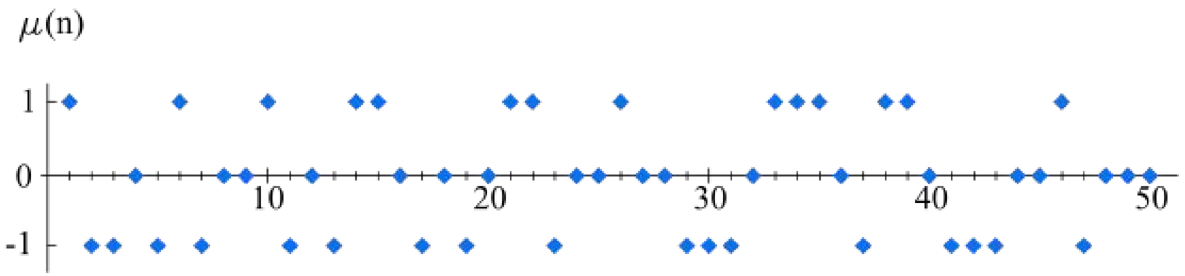
- Is not divisible by perfect square (except 1)
  - perfect square: sqrt(n) = is integer. sqrt(16) = 4
  - SQ: e.g. not divisible by 16=4x4…or 49=7x7…etc
- In other words, no prime number occurs more than once: e.g. n = 2*5*11 is square free, but n = 2*3*3*3*7 is not (divisible by 9 = 3x3)
- I-th square free: 1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 33, 34
  - F(13) = 19

Möbius function

- $\mu(1) = 1$
- $\mu(n) = 1$ if n is a **square-free** positive integer with an **even number** of prime factors.
  - E.g. $\mu(2*3*5*7) = 1$
- $\mu(n) = -1$ if n is a square-free positive integer with an odd number of prime factors.
  - E.g. $\mu(2*3*5) = -1$
- $\mu(n) = 0$ if n has a squared prime factor.
  - E.g. $\mu(2*\mathbf{3}*\mathbf{3}*7) = 0$

Möbius sequence

- $\mu(n)$ = 1, -1, -1, 0, -1, 1, -1, 0, 0, 1, -1, 0, -1, 1, 1, 0, -1, 0, -1, 0, 1, 1, -1, 0, 0, 1, 0, 0, -1, -1, -1, 0, 1, 1, 1, 0, -1, 1, 1, 0, -1, -1, -1, 0, 0, 1, -1, 0
- $\mu(n)$ +1 = 2, 0, 0, 1, 0, 2, 0, 1, 1, 2, 0, 1, 0, 2



Möbius function code

```
int moebius(int n)
{
    int mebVal = -1;

    for(int i = 2; i * i <= n; i++)
        if(n % i == 0)
        {
            if(n %(i*i)) == 0)
                return 0;
            n /= i, mebVal = -mebVal;
        }

    if(n)
        mebVal = - mebVal;
    return mebVal;
}
```

Möbius generator code

```
void moebius_generator() {
    const int MAX = 1000000;
    int moebius[MAX+1];
    char prime[MAX+1];

    for (ll i=2; i<=MAX; i++)
        moebius[i] = -1, prime[i] = 1;

    for (ll i=2; i<=MAX; ++i) if(prime[i]) {
        moebius[i] = 1;
        for (ll j=2*i; j<=MAX; j+=i)
            prime[j] = 0, moebius[j] = j%(i*i) == 0 ? 0 : -moebius[j];
    }
}
```

Möbius and Inclusion Exclusion

- Recall, in IE we compute all subsets, and add odd subsets and remove negative subsets
- Assume generating all subsets of primes but in implicit way (e.g. iterate on numbers), Möbius can tell you if number is odd subset or even
- Typically, ignoring numbers with repeated prime factors is target
- Then Möbius(n) plays perfect role in that

Möbius and Inclusion Exclusion

- Given square free number, find its index?
  - E.g. F_reverse(n = 19) = 13
- Reverse thinking: Can we remove non SFree?
- In range n, remove who divides by 2x2, 3x3, 4x4, 5x5, 6x6...etc
  - 4x4 and 6x6 already computed by previous ones
  - Ignore duplicate primes (4x4)...use IE for others F(2)+F(3) +F(5)-F(6)

```
ll val = 19, idx = val;
for (ll i = 2; i*i<=val; i++)
    idx -= moebius[i]*(val/(i*i));
```

Möbius and Inclusion Exclusion

- Count the triples (a,b,c) such a, b, c <= n, and gcd(a, b, c) = 1
  - Reverse thinking, total - (# triples gcd > 1)
  - How many triples with gcd multiple of 2: $(n/2)^3$
  - How many triples with gcd multiple of 3: $(n/3)^3$
  - and 4? Ignore any numbers of internal duplicate primes
  - and 6? already computed in 2, 3. Remove it: $-(n/6)^3$
  - This is Inclusion Exclusion

```
int n = 4;
ll sum = n*n*n;
for (ll i = 2; i <= n; ++i)
    sum -= moebius[i]*(n/i)*(n/i)*(n/i);
```

Totient and Möbius connection

- Sum over divisors d of n

$$\sum_{d} d\,\mu\left(\frac{n}{d}\right) = \phi\,(n),$$

- Read Totient functions proves

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً

Problems

- UVA 417, 11417, 10179, 10820, 10990, 11426, 10299, 11327
- TIMUS: 1673
- http://www.geeksforgeeks.org/eulers-totient-function/