Number Theory - Factors, GCD, and Primes

1 Introduction

Number theory is the branch of mathematics that deal with properties of integers. However, it has a very close tie with computer science. The most notable applications of number theory is in the field of cryptography. For now, we will discuss the basic aspects of number theory. First are some definitions:

Definition 1. *An integer b is **divisible** by an integer a if there exists and integer x such that $b = ax$. In this case, we say that a is a **factor** of b and b is a **multiple** of a. We write this as $a \mid b$.*

Definition 2. *A positive integer $p > 1$ is **prime** if there is no factor a such that $1 < a < p$. Otherwise, p is a composite number.*

Definition 3. *The **prime factorization** of a number n is the product $p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k} = n$ where each $p_1, p_2, ..., p_k$ are distinct primes and $\alpha_1, \alpha_2, ..., \alpha_k$ are positive integers.*

Definition 4. *A positive integer d is said to be a **common divisor** of two integers a and b if $d \mid a$ and $d \mid b$.*

Definition 5. *The **greatest common divisor** or **gcd** of two integers a and b is the largest positive integer g such that g is a common divisor of a and b.*

Definition 6. *The **least common multiple** or **lcm** of two integer a and b is the smallest positive integer m such that $a \mid m$ and $b \mid m$.*

2 Some Basic Theorems

Here are some basic theorems in number theory. The proofs are omitted.

Theorem 1. *The gcd g of two integers a and b can be characterized by two ways:*

- $g = min\ \{ax + by > 0 \mid x, y \in Z\}$
- *g is the positive common divisor of a and b that is divisible by all other common divisors.*

Corollary. $(a, b) = (b, a) = (a, b + ax)$ *for any integer x.*

Fundamental Theorem of Arithmetic
Every integer n can be prime factorized. Furthermore, this factorization is unique up to the order of the prime factors.

Theorem 2. *There are an infinite number of prime. Furthermore, for any integer k, there exists two consecutive primes p_1 and p_2 such that $|p_1 - p_2| > k$.*

3 Factoring and Primality Testing

One of the most fundamental problem in number theory is to find a all factors of an integer n. The simplest approach is to iterate through all numbers less than n and check whether n is divisible. The runtime of this is $O(n)$. However, we can speed this up significantly by noting that if d is a factor of n, then so is n/d. Furthermore, one of d and n/d is less than or equal to $\sqrt{n}$ and the other one is larger than or equal to $\sqrt{n}$. This gives the following simple $O(\sqrt{n})$ algorithm:

```
vector<int> findFactors(int N) {
  vector<int> ret;
  for (int i = 1; i * i <= N; ++i) {
    if (N % i == 0) {
      ret.push_back(i);
      if (N/i != i) ret.push_back(N/i);
    }
  }
  return ret;
}
```

Note that faster algorithm for factoring do exists, but no polynomial time algorithm is known. In fact, many encryption algorithm depend the fact that factoring is hard!.

A closely related problem is to determine whether an integer n is prime or not. This has very important applications in cryptography since many algorithms require the use of large primes. We can use a similar algorithm above to test whether an integer is prime:

```
bool isPrime(int N) {
  for (int i = 2; i * i <= N; ++i)
    if (N % i == 0) return false;
  return true;
}
```

Note that it is possible to test for primality in polynomial time, making the above algorithm rather slow. In practice, testing whether a number is prime is done probabilistically, using algorithm such as the Rabin-Miller Primality Test. More on this next class.

4 Finding Multiples in a Range

Another common problems in number theory asks: how many numbers in the range $[a, b]$ are divisible by a number k? Note that this problem is reducible to asking how many number are in the range $[1, n]$. To find the answer for the range $[a, b]$, we can find the answer for $[1, b]$ and then subtract from it the answer for $[1, a-1]$. Now, the number of multiples of k in the range $[1, n]$ can be given by the formula $\lfloor \frac{n}{k} \rfloor$. Thus, the problem is solvable in constant time.

Let us now extend the problem to find how many numbers in the range $[1, n]$ that are divisible by any one of the following numbers $k_1, k_2, ..., k_n$. Let us denote the set A_i to be all number in the range $[1, n]$ that is divisible by k_i. We showed how to calculate $|A_i|$ above, but now we need to find out $|\bigcup_{i=1}^{n} A_i|$. How can we do this? The answer lies in the **Inclusion Exclusion Principle**, which states that:

$$|\bigcup_{i=1}^{n} A_i| = \sum_{i=1}^{n} |A_i| - \sum_{i \neq j}^{n} |A_i \cap A_j| + \sum_{i \neq j \neq k}^{n} |A_i \cap A_j \cap A_k| + ... + (-1)^{n+1} |\bigcap_{i=1}^{n} A_i|$$

Now we just have to note that $|A_i \cap A_j|$ is equal to the number of multiple of m in the range $[1, n]$, where m is the LCM of k_i and k_j. We will discuss how to find LCM in section 6.

5 Sieve of Eratosthenes

Note that the primality testing algorithm in section 3 can be made faster if we have a list of primes less than n. This is because instead of iterating through all numbers from 2 to $\sqrt{n}$, we can just iterate all the prime in that range. More generally, we can ask the question: how do we generate all primes from the range $[1, n]$? The naive algorithm iterate through all numbers in the range and call `isPrime()` on each of them. However, this is slow since we are doing a lot of redundant work here. Suppose we just determined 2 is a prime. Is there a point in calling the function `isPrime(4)`? It's clear that 4 is a composite number since it's divisible by 2. Actually, any multiple of 2 won't be a prime! The Sieve of Eratosthenes (or sieve for short) is an efficient algorithm to generate all primes from in the range $[1, n]$ using the above observation. This algorithm runs in time $O(n * logn)$.

```
bool prime[1024];
void sieve() {
  memset(prime, 1, sizeof(prime));
  prime[0] = prime[1] = false;
  for (int i = 0; i * i <= 1024; ++i) {
    if (prime[i]) {
      // mark all multiples of prime[i] as composite
      for (int j = i * i; j <= N; j += i)
        prime[j] = false;
    }
  }
}
```

6 Finding GCD

Another basic problem in number theory is to find the GCD and LCM of two integers a and b. Note that we can easily find the LCM m easily if we know the GCD g via the formula $m = \frac{ab}{g}$. To most common method to find the GCD is the Euclidan Algorithm:

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

Why does the above algorithm work? It is clear from the definition of GCD that $(a, 0) = a$, for any integer a. To show that the recursive call is correct, note that $a \% b = a - \lfloor \frac{a}{b} \rfloor * b$. Using the corollary from section 2, it follows that $(a, b) = (b, a) = (b, a - bx), x = \lfloor \frac{a}{b} \rfloor$. It can be shows that

after every two recursive calls, the parameter a must reduce by more than half. Thus, the algorithm runs in $O(log(\max\{a, b\})$ time.

6.1 Extended GCD

Let $g = (a, b)$. Based on the theorem in section 2, we know that there exists two integers x, y such that $ax + by = g$. We can easily modify the Euclidean Algorithm to return the number x, y as well. First of all, if $b = 0$, then we can set $x = 1, y = 0$. How do we handle the recursive call. Suppose that we have s, t such that $bs + (a \% b)t = g$ from the recursion, we need to derive x and y. Note that:

$$\begin{aligned} g &= bs + (a \% b)t \\ &= bs + (a - \lfloor \frac{a}{b} \rfloor * b)t \\ &= at + b(s - \lfloor \frac{a}{b} \rfloor t) \end{aligned}$$

So after finding s and t, we simply set $x = t$ and $y = s - \lfloor \frac{a}{b} \rfloor t$. Implementation as follow:

```
int egcd(int a, int b, int& x, int& y) {
  if (b == 0) {
    x = 1; y = 0;
    return a;
  }
  int s, t;
  int g = egcd(b, a%b, s, t);
  x = t;
  y = s - (a/b) * t;
  return g;
}
```

7 Application of Extended GCD: Diophantine Equation

One of the most basic application of extended gcd is to find integer solutions to the equation $Ax + By = C$, where A, B, C are given. There can be zero solution or infinite solutions to the equation. To check for the zero solution case, take note that the left hand side of the equation is divisible by $g = (A, B)$. Thus, the right hand side must also be divisible by g. Thus, unless $g \mid C$, there is no solution to the equation.

Now let us assume $g \mid C$ and let $C = gz$ for some integer z. Then we can find one solution as by calling `egcd(A, B, s, t)`. After the function terminate, we know $As + Bt = g$. Multiply both side by z yield one solution: $A(sz) + B(tz) = gz = C$. Now, note that for any integer k, setting $x = sz + k\frac{B}{g}, y = tz - k\frac{A}{g}$ is also a solution:

$$A(sz + k\frac{B}{g}) + B(tz - k\frac{A}{g}) = A(sz) + B(tz) + k\frac{AB}{g} - k\frac{AB}{g} = C$$

Thus, there are infinite number of solutions in this case.