Chapter 11

Number-Theoretic Algorithms

Preview

- Overview
- Lecture Notes
  - Number Theory Review
    - Composite and Prime Numbers
    - Greatest Common Divisor
    - Prime Factorization
    - Least Common Multiple
  - Computing the Greatest Common Divisor
    - Euclid's Algorithm
    - An Extension to Euclid's Algorithm
  - Modular Arithmetic Review
    - Group Theory
    - Congruency Modulo n
    - Subgroups
- Quick Check
  - Solving Modular Linear Equations
  - Computing Modular Powers
  - Finding Large Prime Numbers
    - Searching for a Large Prime
    - Checking if a Number Is Prime
  - The RSA Public-Key Cryptosystem
    - Public-Key Cryptosystems

- The RSA Cryptosystem

- Quick Check
- Classroom Discussion
- Homework
- Keywords
- Important Links

Overview

Number theory is the branch of mathematics concerned with the properties of integers. Number-theoretic algorithms are algorithms that solve problems involving the integers. A number-theoretic algorithm might find the greatest common divisor of two integers.

An important application of number-theoretic algorithms is **cryptography**, which is the discipline concerned with encrypting a message sent from one party to another, so that someone who intercepts the message will not be able to decode it.

Unlike the methods presented for solving algorithms in previous chapters, number-theoretic algorithms are concerned with solving a certain type of problem; they are not algorithms that share a common technique.

Lecture Notes

Number Theory Review

This section will review some basic elements of number theory.

Composite and Prime Numbers

The set Z = {. . . - 2,-1, 0, 1, 2, …} is called the set of **integers**. For any two integers n, $h \in Z$, we say h **divides** n, denoted $h|n$, if there is some integer k such that $n = kh$. If $h|n$, we also say n is **divisible** by h, n is a **multiple** of h, and h is a **divisor** or **factor** of n.

An integer $n > 1$ whose only positive divisors are 1 and n is a called a **prime number** or simply a **prime**. A prime number has no factors. An integer n > 1 that is not prime is called a composite number. A **composite** number has at least one factor.

Greatest Common Divisor

If $h|n$ and $h|m$, then h is called a **common divisor** of n and m. If n and m are not both zero, the **greatest common divisor** (gcd) of n and m, denoted gcd(n,m) is the largest integer that divides both of them. Refer to the text for examples.

Theorem 11.1: If $h|n$ and $h|$m, then for any integers i and j

$$h| (in + jm).$$

For any two integers n and m, where $m \neq 0$, the **quotient** q of m is given by

$$q = [n/m],$$

and the **remainder** r of dividing n by m is given by

$$r = n - qm.$$

The **division algorithm** says not only that the integers if q and r in Equality 10.1 exist but also that they are unique.

Prime Factorization

Every integer greater than one can be written as a unique product of primes. Two integers *n* and *h*, not both zero, are called **relatively prime** if gcd(*n*, *h*) = 1.

Theorem 11.5 is called the **unique factorization theorem** and the **fundamental theory of arithmetic**.

Every integer *n* > 1 has a unique factorization of prime numbers.

Least Common Multiple

A concept similar to that of the greatest common divisor is that of the **least common multiple** (lcm). If *n* and *m* are both nonzero, the least common multiple of *n* and *m*, denoted lcm(n,m), is the smallest positive integer that they both divide.

Computing the Greatest Common Divisor

Theorem 11.6 gives us a straightforward way to compute the greatest common divisor of two integers. We simply find the unique factorizations for the two integers, determine which primes they have in common, and determine the greatest common divisor to be a product whose terms are these common primes, where the power of each prime in the product is the smaller of its orders in the two integers. Refer to example 11.13 for an example.

The problem with the previous technique is that it is not easy to find the unique factorization of an integer. Imagine the difficulty if the integers had 25 digits instead of 7. No one has ever found a polynomial-time algorithm for determining the factorization of an integer.

Euclid's Algorithm

Theorem 11.1 gives us a straightforward method for determining the greatest common divisor of two integers. Example 11.9 illustrates this method. To find the gcd (*n,m*), we recursively apply the equality in the theorem until *m* = 0, and then we return *n*. This method is called **Euclid's Algorithm** after the person who developed it in around 300 B.C. Refer to the text for a more detailed discussion.

```
public static int gcd( int n , int m)
{
          i f (m == 0)
       return n ;
          else
            return gcd(m, n mod m) ; // The C++ would be n % m.
}
```

For example:

gcd(7,276,500, 3,185,325) = gcd(3,185,325, 905,850)
= gcd(90,5850, 467,775)
= gcd(467,775, 438,075)
= gcd(438,075, 29,700)
= gcd(29,700, 22,275)
= gcd(22,275, 7,425)
= gcd(7,425, 0)
= 7,425.

An Extension to Euclid's Algorithm

Theorem 11.1 entails that there are integers i and j such that

gcd (*n*,*m*) = *in* +j*m*.

Knowledge of these integers will be important to our algorithm for solving modular linear equations. We modify 11.1 to also produce these integers. In this version (Algorithm 11.2) we make gcd a variable because so doing makes the proof of the correctness of the algorithm more transparent.

```
public static void Euclid ( int n , int m, int)
{
 if (m == 0) {
          gcd = n ; i = 1; j = 0;
}
else {
           int i' , j'; gcd';
           Euclid (m, n mod m, gcd', i', j'); // The C++ code would be n % m
          gcd = gcd';
           i = j';
           j = i' − (n/m) j';

          }
}
```

This table illustrates the flow of Euclid's Algorithm when the top-level call is Euclid(42, 30, **gcd**, *i*, *j*); The values returned at the top level are *gcd* = 6, *i* = -2, and *j* = 3.

Call	*n*	*m*	*gcd*	*i*	*j*
0	40	30	6	-2	3
1	30	12	6	1	-2
2	12	6	6	0	1
3	6	0	6	1	0
	→	→	î	î	î

Modular Arithmetic Review

We develop modular arithmetic within the context of group theory. So first we review that theory.

Group Theory

A closed binary operation * on a set *S* is a rule for combining two elements of *S* to yield another element of *S*. We have the following definition:

A **group** *G* = (*S*, *) is a set *S* together with a closed binary operation * on S satisfying the following:

1. * is associative. That is, for all *a*, *b*, *c* Є *S*
   a * (*b* * *c*) = (*a* * *b*) * *c*.
2. There is an identity element *e* in *S*. That is, for each *a* Є S
   e * *a* = *a* * *e* = *a*.
3. For each a Є S there exists an inverse element a' such that
   a * *a*' = *a*' * *a* = *e*

Congruency Modulo n

Let's begin with a definition:

Let m and k be integers and n be a positive integer. If $n|\ (m - k)$ we say m is congruent to k modulo n, and we write

$m = k \bmod n$.

The set of all integers congruent to m modulo n is called the **equivalence class modulo *n* containing *m***. Since congruency is an equivalence relation, any integer in a given class determines the class. For a given integer m, it is the set

$\{m + in$ such that $i \in Z\}$

We denote the equivalence class modulo n containing m by $[m]_n$. The set of all equivalence classes modulo n is denoted Z_n. That is,

$Z_n = \{[0]_n , [1]_n , \ldots , [n-1]_n\}$.

--For two members of Z_n we define addition as follows:

$[m]_n + [k]_n = [m + k]_n$

Subgroups

If $G = (S, *)$ is a group, $S' \subseteq S$, and $G' = (S', *)$ is a group, then G' is said to be a subgroup of G. It is called a proper subgroup if $S' \neq S$.

Given a group $G = (S, *)$, and $a \in S$ with inverse a', for $k > 0$ we define

$a^k = a * a * \ldots *a$ k times

$a^0 = 1$

$a^{-k} = (a')^k$

Suppose we have a group $G = (S, *)$ and a subset $S' \subseteq S$ such that for every $a, b \in S'$, $a * b \in S'$. Then S' is said to be **closed** relative to *.

Quick Check

1. Number-theoretic algorithms are algorithms that solve problems involving the ____.
   Answer: integers
2. If $h | n$, we say n is ____ by h, and n is a ___ of h
   Answer: divisible, multiple
3. Two integers n and h, both not zero, are called ____ if gcd $(n,h)=1$.
   Answer: relatively prime
4. For all $a, b, c \in S$ if $a * (b * c) = (a * b) * c$. Then * is said to be ____.
   Answer: associative

Solving Modular Linear Equations

Consider this modular equation:

$$[m]_n x = [k]_n$$

for x, where x is an equivalence class modulo n, and m, $n > 0$.

Let $(|m|_n)$ be the subgroup generated by $[m]_n$ relative to the group (Z_n+).

Since $(|m|_n) = \{[0]_n , [m]_n , [2m]_n , \ldots \}$, We can find a solution to the above solution only if:

$$[k]_n \in ([m]_n)$$

The input size of Algorithm 11.3 is the number of bits it takes to encode the input, which is given by

$$s = [\lg n] + 1 \qquad t = [\lg m] + 1 \qquad u = [\lg k] + 1$$

The time complexity includes the time complexity for Algorithm 11.2 (Euclid's Algorithm 2), which we already knoe is O (*st*), plus the time complexity of the for-*I* loop. Since d can be as large as *m* or *n,* this time complexity is worst-case in terms of input size. However, we can do nothing about this since the problem requires that we compute and display an exponential number of values in the worst case.

Computing Modular Powers

In Example 11.51, we determined the power by simply taking 7 to the 11^{th} power. This method has exponential time complexity in terms of the input size. We will develop a more efficient algorithm. This algorithm uses the method of repeated squaring.

The table below, shows the state of a after each iteration of the for-*i* loop in Algorithm 11.4 given this input. The final value of a, namely $[147]_{257}$,is the value of $([5]_{257})^{45}$

i	5	4	3	2	1	0
b_i	1	0	1	1	0	1
k_i	1	2	5	11	22	45
a	$[5]_{257}$	$[25]_{257}$	$[41]_{257}$	$[181]_{257}$	$[122]_{257}$	$[147]_{257}$

Finding Large Prime Numbers

Finding large prime numbers is necessary to the success of the RSA public-key cryptosystem, discussed in section 11.7. . After discussing searching for a large prime number, we want to test whether a number is prime.

Searching for a Large Prime

To find a large prime number we randomly select integers of the appropriate size and test whether each selected integer is prime until one is found to be prime. When we use this method, we should consider the probability of finding a prime when an integer is chosen at random.

The prime distribution theorem enables us to approximate this probability.

The **prime distribution function π(*n*)** is the number of primes that are less than or equal to n. For example, π(12) = 5 since there are five primes, namely 2, 3, 5, 7, and11, that are less than or equal to 12. The **prime number theorem** (11.27) gives an approximation of π (*n*).

For large values of *n* we can use *n*/ ln *n* as an estimate of the number of primes less than or equal to *n*. Therefore, if we randomly choose an integer between 1 and *n* according to the uniform distribution, the probability of it being prime is about

$$n/ \ln n) / n = 1 / \ln n$$

To find a large prime number we randomly select numbers in the appropriate range, and we then check them for being prime until one is found to be prime. We can see from the examples in the text that it should not take long to find a prime.

Checking if a Number Is Prime

Until recently no one had ever found a polynomial-time algorithm for the prime-checking problem, and many people thought it to be *NP*-complete. The standard efficient algorithm for prime-checking was a Monte Carlo algorithm called the **Miller-Rabin Randomized Primality Test**, which appeared in Rabin (1980). This algorithm says that if a number is prime, then the algorithm says the number is always prime. This happens so rarely, that the algorithm can be counted upon for accuracy. The Miller-Rabin algorithm does not find a factor of the number if it determines the number is composite. So, it cannot be used for factoring.

A polynomial time algorithm

Let $f(x)$ and $g(x)$ be polynomials with integral coefficients. If the coefficients of each power of x are congruent modulo n, we say that $f(x)$ and $g(x)$ are congruent modulo n, and we write

$$f(x) \equiv g(x) \bmod n$$

Refer to the text for a more detailed discussion of the algorithm.

The RSA Public-Key Cryptosystem

Recall the scenario at the beginning of the chapter, in which Bob wants to send Alice a secrete love note over the Internet. In order to do this, he needs to **encrypt** the message and make sure only Alice knows how to **decrypt** it.

Public-Key Cryptosystems

A **public-key cryptosystem** consists of a set of permissible messages, a set of participants such that each participant has a **public key** and a **secret key**, and a network for sending messages among the participants. The set of permissible messages might include all character sequences of some length or less. We let

M = set of permissible messages,

then each participant *x*'s public key *pkeyx* and secret key *skeyx* determine functions *pubx* and *secx* from *M* to *M*, which are inverses of each other.

The public keys are known to all the participants, but the secret key of *x* is known only to *x*. This is how Bob can send a secret love message to Alice.

That is, for each $b \in M$

- $b = pub_X (sec_X(b))$
- $b = sec_X (pub_X(b))$

The public keys of all participants are known to all the participants, but the secret key of x is known only to x.

Example: if Bob wants to send Alice a secret love note *b*, he and Alice do the following:

1. Bob computes $c = pub_{Alice}(b)$ using Alice's public key, $pkey_{Alice}$. The message *c* is called ciphertext. It is unreadable.
2. Bob sends ciphertext *c* to Alice.
3. Alice computes $b = sec_{Alice}(c)$ using her secret key $skey_{Alice}$.

The RSA Cryptosystem

The RSA cryptosystem relies on the facts that we can find large primes fairly readily, but we have no efficient method for factoring a large number.

In the **RSA public-key cryptosystem**, each participant creates his public key and secret key according to the following steps:

1. Select two very large prime number *p* and *q*. The number of bits needed to represent *p* and *q* might be 1024.
2. Compute $n = pq$
   $\varphi(n) = (p-1)(q-1)$.
3. Select a small prime number g that is relatively prime to $\varphi(n)$.
4. Compute the multiplicative inverse $[h]_{\varphi(n)}$ of $[g]_{\varphi(n)}$.
   That is, $[g]_{\varphi(n)} [h]_{\varphi(n)} = [1]_{\varphi(n)}$
5. Let *pkey* = (*n*, *g*) by the public key, and *skey* = (*n*, *h*) be the secret key. The set of permissible messages is Z_n. The function corresponding to the public key *pkey* = (*n*, *g*) is $pub(b) = b^g$, and the function corresponding to the secret key skey = (n, h) is $sec(b) = b^h$

The values of these functions can be computes by using Algorithm 11.5.

Quick Check

1. If *m* and *n* are relatively prime, then *n* ____ is if and only if $(x - m)^n = (x^n - m) \bmod n$
   Answer: prime
2. In addition to other things, a public-key cryptosystem consists of a set of participants such that each participant has a(n) ____and a(n) ____.
   Answer: public key, secret key
3. The ____ returns the remainder when one polynomial is divided by the other.
   Answer: mod function

Classroom Discussion

- Prove that are infinitely many prime numbers.
- Can you think of a new way to check if a number is prime? Discuss

Homework

Assign Exercises 3, 5, 19, and 36.

Keywords

- **Cipher-text –** message $c = pub_X$ (b).
- **Closed binary operation -** * on a set *S* is a rule for combining two elements of *S* to yield another element *S*.
- **Common divisor –** if $h|n$ and $h|m$ then *h* is common divisor of *n* and *m*.
- **Composite number –** not prime; has at least one factor.
- **Congruent to *k* modulo *n* –** let *m* and *k* be integers and *n* be a positive integer. If $n|(m - k)$ we say *m* is congruent to *k* modulo *n* and we write $m = k \bmod m$.
- **Cryptography –** the discipline concerned with encrypting a message sent from one party to another.
- **Decryption –** the process of making encrypted information readable again.
- **Divides –** integer *h* divides *n* ($h|n$) if there is some integer *k* such that $n = kh$.
- **Division algorithm –** the theorem in mathematics that expresses the outcome of the division of integers.
- **Encryption –** the process of making information unreadable to anyone who does not have a code for deciphering it.
- **Euclid's Algorithm –** an algorithm that computes the greatest common divisor of two positive integers.
- **Greatest common divisor –** of two integers is the largest integer that divides both of them.
- **Group -** a group $G = (S, *)$ is a set *S* together with the closed operation * on *S* satisfying some rules.
- **If *h* | *n***
  n is **divisible** by *h*
  n is a **multiple** of *h*
  h is a **divisor** or **factor** of *n*

- **Integer –** numbers that are written without a fractional or decimal component.
- **Least common multiple -** of two integers is the smallest integer that divides both of them.
- **Miller Rabin Randomized Primality test –** an algorithm that determines whether a given number is prime.
- **Mod-function –** returns the remainder when one polynomial is divided by the other.
- **Number Theory –** branch of mathematics concerned with the properties of integers.
- **Prime distribution function π (*n*) –** the number of primes that are less than or equal to *n*.
- **Prime number –** an integer $n > 1$ whose only positive divisors are 1 and *n.*
- **Prime number theorem –** gives a rough description of how primes are distributed.
- **Public-key cryptosystem –** consists of a set of permissible messages a set of participants such that each is a **public key** and a **secret key**, and a network for sending messages among the participants
- **Relatively prime –** two integers *n* and *h* not both zero are called relatively prime if $\gcd(n,h) = 1$
- **RSA public key cryptosystem –** a system where each participant creates his public key and secret key according to some steps.
- **Unique factorization theorem or fundamental theorem of arithmetic –** every integer $n > 1$ has a unique factorization as a product of prime numbers.

Important Links

- http://www.nist.gov/dads/ (Dictionary of Algorithms and Data Structures)
- http://yacas.sourceforge.net/Algochapter2.html (Number Theory Algorithms)
- http://userpages.umbc.edu/~rcampbel/NumbThy/Class/BasicNumbThy.html (Basics of Computational Number Theory)