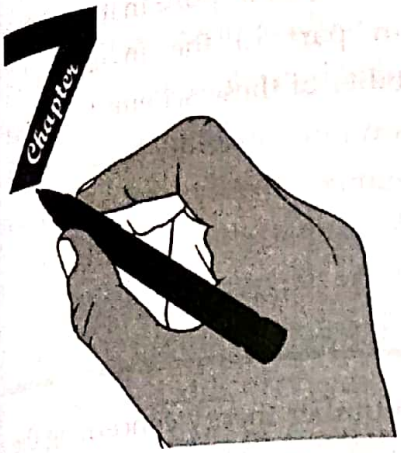

NUMBER THEORETIC ALGORITHMS

CHAPTER OUTLINE

After studying this chapter, the reader will be able to understand the

- Number Theoretic Notations, Euclid's and Extended Euclid's Algorithms and their Analysis.
- Solving Modular Linear Equations, Chinese Remainder Theorem, Primility Testing: Miller-Rabin Randomized Primility Test and their Analysis

Number theory was once viewed as a beautiful but largely useless subject in pure mathematics. Today number-theoretic algorithms are used widely, due in part to the invention of cryptographic schemes based on large prime numbers. The feasibility of these schemes rests on our ability to find large primes easily, while their security rests on our inability to factor the product of large primes. This chapter presents some of the number theory and associated algorithms that underlie such applications.

Number Theoretic Notations

This section provides a brief review of notions from elementary number theory concerning the set Z = {. . . , -2, -1, 0, 1, 2 . . .} of integers and the set N = {0, 1, 2 . . .} of natural numbers.

Divisibility and Divisors

The notion of one integer being divisible by another is a central one in the theory of numbers. The notation d | a (read "d divides a") means that a = kd for some integer k. Every integer divides 0. If a > 0 and d | a, then | d | ≤ | a |. If d | a, then we also say that 'a' is a multiple of d. If d | a and d≥0, we say that d is a divisor of a. Note that d | a if and only if -d | a, so that no generality is lost by defining the divisors to be nonnegative, with the understanding that the negative of any divisor of 'a' also divides a. A divisor of an integer 'a' is at least 1 but not greater than | a |.

For example, the divisors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24.

Every integer a is divisible by the trivial divisors 1 and a. Nontrivial divisors of 'a' are also called factors of a. For example, the factors of 20 are 2, 4, 5, and 10.

Prime and Composite Numbers

An integer a > 1 whose only divisors are the trivial divisors 1 and 'a' is said to be a prime number (or, more simply, a prime). Primes have many special properties and play a critical role in number theory. The small primes, in order, are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59...

An integer a > 1 that is not prime is said to be a composite number (or, more simply, a composite). For example, 39 is composite because 3 | 39. The integer 1 is said to be a unit and is neither prime nor composite. Similarly, the integer 0 and all negative integers are neither prime nor composite.

Common Divisors and Greatest Common Divisors

If d is a divisor of 'a' and also 'a' divisor of b, then d is a common divisor of 'a' and b. For example, the divisors of 30 are 1, 2, 3, 5, 6, 10, 15, and 30, and so the common divisors of 24 and 30 are 1, 2, 3, and 6. Note that 1 is a common divisor of any two integers. An important property of common divisors is that

- d | a and d | b implies d | (a + b) and d | (a - b).
- d | a and d | b implies d | (ax + by)

The greatest common divisor of two integers a and b, not both zero, is the largest of the common divisors of a and b; it is denoted gcd(a, b).

For example, gcd(24, 30) = 6, gcd(5, 7) = 1, and gcd(0, 9) = 9. If a and b are not both 0, then gcd(a, b) is an integer between 1 and min(|a|, |b|). We define gcd(0, 0) to be 0; this definition is necessary to make standard properties of the gcd function universally valid. The following are elementary properties of the gcd function:

- gcd(a, b) = gcd(b, a)
- gcd(a, b) = gcd(-a, b)
- gcd(a, b) = gcd(|a| , |b|)
- gcd(a, 0) = |a|
- gcd(a, ka) = |a| for any k Z.

Corollary 1

For any integers a and b, if d | a and d | b then d | gcd(a, b).

Corollary 2

For all integers a and b and any nonnegative integer n,

gcd(an, bn) = n gcd(a, b)

Corollary 3

For all positive integers n, a, and b, if n | ab and gcd(a, n) = 1, then n | b.

Relatively Prime Integers

Two integers a, b are said to be relatively prime if their only common divisor is 1, that is, if gcd(a, b) = 1. For example, 8 and 15 are relatively prime, since the divisors of 8 are 1, 2, 4, and 8, while the divisors of 15 are 1, 3, 5, and 15. The following theorem states that if two integers are each relatively prime to an integer p, then their product is relatively prime to 'p'.

- For any integers a, b, and p, if gcd(a, p) = 1 and gcd(b, p) = 1, then gcd(ab, p) = 1.

Theorem 1

For any integers a, b, and p, if both gcd(a, p) = 1 and gcd(b, p) = 1, then

gcd(ab, p) = 1.

Theorem 2

For all primes p and all integers a, b, if p | ab, then p|a or p | b (or both).

Euclid's Algorithm

In mathematics, the Euclidean algorithm is an efficient method for computing the greatest common divisor of two numbers, the largest number that divides both of them without leaving a remainder. It is named after the ancient Greek mathematician Euclid, who first described it in his Elements.

The following gcd algorithm is described in the elements of although it may be of even earlier origin. It is written as a recursive program based directly. The inputs 'a' and 'b' are arbitrary nonnegative integers.

```
EUCLID (a, b)
{
        If b = 0
                Then return a
        Else
                Return EUCLID (b, a mod b)
}
```

Analysis

Since it is recursive algorithm so we need to find their recurrence relation,

Since every time the problem is divided into two parts one is b and another part is a mod b.

Thus the size of sub problem=n/2

Dividing and merging time = constant = O(1)

Thus recurrence relation is,

$T(n) = T(n/2) + O(n)$

By solving this we get, $T(n) = O(\log n)$

Example 1: Find gcd (30, 21) by Euclid's Algorithm

EUCLID (30, 21)
= EUCLID (21, 30 mod 21)
= EUCLID (21, 9)
= EUCLID (9, 21 mod 9)
= EUCLID (3, 0)
Since b ==0 so return a
= 3

In this computation, there are three recursive invocations of EUCLID.

Example 2: Find gcd (270, 192) by Euclid's Algorithm

EUCLID (270, 192)
= EUCLID (192, 270 mod 192)
= EUCLID (192, 78)
= EUCLID (78, 192 mod 78)
= EUCLID (78, 36)
= EUCLID (36, 78 mod 36)
= EUCLID (36, 6)
= EUCLID (6, 36 mod 6)
= EUCLID (6, 0)
Since b =0 so return a
= 6

Example 3: Find gcd (270, 192) by Euclid's Algorithm

q	r1	r2	r
1	270	192	78
2	192	78	36
2	78	36	6
6	36	6	0
	6	0	0

Extended Euclid's Algorithms

In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the greatest common divisor of integers a and b, also the coefficients of Bezout's identity, which are integers x and y such that

$$d = \gcd(a, b) = ax + by$$

Where x and y may be zero or negative. We shall find these coefficients useful later for the computation of modular multiplicative inverses. The procedure EXTENDED-EUCLID takes as input a pair of nonnegative integers and returns a triple of the form (d, x, y) that satisfies given equation.

It is used for finding the greatest common divisor of two positive integers 'a' and b and writing this greatest common divisor as an integer linear combination of a and b. The steps of this algorithm are given below.

1. Set the value of the variable c to the larger of the two values 'a' and b, and set d to the smaller of a and b.
2. Find the quotient and the remainder when c is divided by d. Call the quotient q and the remainder r. Use the division algorithm and expressions for previous remainders to write an expression for r in terms of 'a' and b.
3. If r=0 then gcd(a, b) = d. the expression for the previous value of r gives an expression for gcd(a, b) in terms of a and b stop.
4. Otherwise, use the current values of d and r as the new values of c and d, respectively, and go back to step 2.

Algorithm

```
EXTENDED-EUCLID (a, b)
{
        if b = 0
                Then return (a, 1, 0)
        (d', x', y') ← EXTENDED-EUCLID (b, a mod b)
        (d, x, y) ← (d', y', x'- floor (a/b)y')
        Return (d, x, y)
}
```

Analysis

Since the number of recursive calls made in EUCLID is equal to the number of recursive calls made in EXTENDED-EUCLID, the running times of EUCLID and EXTENDED-EUCLID are the same, to within a constant factor. That is, for $a > b > 0$, the number of recursive calls is $O(\log b)$.

Example: find GCD (161, 28) and value of x and y by using extended Euclidean's algorithm.

Solution: we have a=161, b=28 and ax+by = GCD (a, b)

Let's define following three equations,

$r=r_1-q*r_2$

$x=x_1-q*x_2$ and

$y= y_1-q*y_2$

Consider $a=r_1$ and $b=r_2$

q	r_1	r_2	r	x_1	x_2	x	y_1	y_2	y
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

x=-1, y=6

gcd(a, b) = ax + by

or, (-1*161) + (6*28) = 7

or, - 161 + 168 = 7

or, 7 = 7

Chinese Remainder Theorem

The Chinese remainder theorem is a theorem of number theory, which states that if one knows the remainders of the Euclidean division of an integer n by several integers, then one can determine uniquely the remainder of the division of n by the product of these integers, under the condition that the divisors are pairwise co-prime.

Statement: If m1, m2... mk are pairwise relatively prime positive integers, and if a1, a2... ak are any integers, then the simultaneous congruences

$x \equiv a_1 \pmod{m_1}$,

$x \equiv a_2 \pmod{m_2}$, ...

$x \equiv a_k \pmod{m_k}$ have a solution, and the solution is unique modulo m.

Here we need to calculate x with the help of following four formulas,

$$x = (M_1 x_1 a_1 + M_2 x_2 a_2 + ...+ M_k x_k a_k) \pmod{M}$$

$$M = m_1 m_2 .....m_k$$

$$M_i = \frac{M}{mi}$$

And $M_i x_i = 1 \pmod{m_i}$

Example: solve following congruences by using Chinese remainder theorem.

$x \equiv 1 \pmod 5$

$x \equiv 1 \pmod 7$

$x \equiv 3 \pmod{11}$

Solution: we have $a_1 = 1, a_2 = 1, a_3 = 3$

$m_1=5, m_2=7, m_3=11$

$x = (M_1X_1a_1 + M_2X_2a_2 + M_3X_3a_3) \pmod M$

$M = m_1.m_2.m_3 = 5 \times 7 \times 11 = 385$

$$M_1 = \frac{M}{m1} = \frac{385}{5} = 77$$

$$M_2 = \frac{M}{m2} = \frac{385}{7} = 55$$

$$M_3 = \frac{M}{m3} = \frac{385}{11} = 35$$

$M_1 X_1 = 1 \pmod{m_1}$

Or, $77 X_1 = 1 \pmod 5$

Or, $2 X_1 = 1 \pmod 5$

Multiplying both side by 3 since we need to make coefficient of X_1 to 1,

$6 X_1 = 3 \pmod 5$

Or, $1 X_1 = 3$

⇨ $X_1 = 3$

Also, $M_2 X_2 = 1 \pmod{m_2}$

Or, $55 X_2 = 1 \pmod 7$

Or, $6 X_2 = 1 \pmod 7$

Multiplying both side by 6 since we need to make coefficient of X_1 to 1,

$36 X_2 = 6 \pmod 7$

Or, $1 X_2 = 6$

⇨ $X_2 = 6$

Also, $M_3 X_3 = 1 \pmod{m_3}$

Or, $35 X_3 = 1 \pmod{11}$

Or, $2 X_3 = 1 \pmod{11}$

Multiplying both side by 6 since we need to make coefficient of X_1 to 1,

$12 X_3 = 6 \pmod{11}$

Or, $1 X_3 = 6$

⇨ $X_3 = 6$

$$x = (M_1x_1a_1 + M_2x_2a_2 + M_3x_3a_3) \pmod M$$
$$= (77 \times 3 \times 1 + 55 \times 6 \times 1 + 35 \times 6 \times 3) \pmod{385}$$
$$= (231 + 330 + 630) \pmod{385}$$
$$= 1191 \pmod{385}$$
$$= 36 \text{ Ans.}$$

Also we can test the solution as,

$36 \pmod 5 = 1$

$36 \pmod 7 = 1$

$36 \pmod{11} = 3$

Solving Modular Linear Equations

We now consider the problem of finding solutions to the equation

$$ax \equiv b \pmod n$$

Where, $a > 0$ and $n > 0$. There are several applications for this problem; for example, we will use it as part of the procedure for finding keys in the RSA public key cryptosystem. We assume that a, b, and n are given, and we are to find all values of x, modulo n, that satisfy given equation. There may be zero, one, or more than one such solution.

Algorithm

Modular_Linear_Equation_Solver(a, b, n)

1. (d, x′, y′) = Extended_Euclid (a, n)
2. If d | b
   a. $x_0 = x' (b/d) \bmod n$
   b. for i=0 to d-1
      i. print $(x_0 + i(n/d)) \bmod n$
3. else print " No Solutions"

Corollary 1

The equation $ax \equiv b \pmod n$ is solvable for the unknown x if and only if gcd(a, n) | b.

Corollary 2

The equation $ax \equiv b \pmod n$ either has d distinct solutions modulo n, where d = gcd(a, n), or it has no solutions.

Corollary 3

For any $n > 1$, if gcd(a, n) = 1, then the equation $ax \equiv b \pmod n$ has a unique solution, modulo n.

Corollary 4

For any $n > 1$, if gcd(a, n) = 1, then the equation $ax \equiv 1 \pmod n$ has a unique solution, modulo n. Otherwise, it has no solution.

Miller-Rabin Randomized Primality Test

The Miller–Rabin primality test or Rabin–Miller primality test is a primality test: an algorithm which determines whether a given number is prime or not. The Miller-Rabin primality test overcomes the problems of the simple test PSEU-DOPRIME with two modifications,

- It tries several randomly chosen base values a instead of just one base value.
- While computing each modular exponentiation, it notices if a nontrivial square root of 1, modulo n, is discovered during the final set of squaring. If so, it stops and outputs COMPOSITE.

Algorithm

/* It returns false if n is composite and returns true if n is probably prime. k is an input parameter that determines accuracy level. Higher value of k indicates more accuracy.*/

bool IsPrime(int n, int k)

1) Handle base cases for n < 3
2) If n is even, return false.
3) Find an odd number d such that n-1 can be written as d*2r.
   Note that since n is odd, (n-1) must be even and r must be greater than 0.
4) Do following k times
   if (millerTest(n, d) == false)
   return false
5) Return true.

bool millerTest(int n, int d)

1) Pick a random number 'a' in range [2, n-2]
2) Compute: x = pow(a, d) % n
3) If x == 1 or x == n-1, return true.
   // below loop mainly runs 'r-1' times.
4) Do following while d doesn't become n-1.
   a) x = (x*x) % n.
   b) If (x == 1) return false.
   c) If (x == n-1) return true.

Example: Input: n = 13, k = 2.

1) Compute d and r such that d*2r = n-1,
   d = 3, r = 2.
2) Call millerTest k times.

1st Iteration:

1) Pick a random number 'a' in range [2, n-2]
   Suppose a = 4
2) Compute: x = pow(a, d) % n
   x = 43 % 13 = 12
3) Since x = (n-1), return true.

2nd Iteration:

1) Pick a random number 'a' in range [2, n-2]
   Suppose a = 5
2) Compute: x = pow(a, d) % n
   x = 53 % 13 = 8

3) x neither 1 nor 12.

4) Do following (r-1) = 1 times

a) x = (x * x) % 13 = (8 * 8) % 13 = 12

b) Since x = (n-1), return true.

Since both iterations return true, we return true.

DISCUSSION EXERCISE

1. Define Number theoretic notations with suitable example.
2. What is the purpose of Euclid's algorithm? Explain with suitable example.
3. Define extended Euclid's algorithm. Explain it with suitable example.
4. Differentiate between Euclid's algorithm and extended Euclid's algorithm with suitable example.
5. Define relatively prime integers with suitable example.
6. Prove that for any integers a, b, and p, if both gcd(a, p) = 1 and gcd(b, p) = 1, then
   gcd(ab, p) = 1
7. Write down the algorithm for Miller-Rabin Randomized Primality Test and analyze it.
8. Solve following congruences by using Chinese remainder theorem.
   $x \cong 3 \pmod 5$
   $x \equiv 5 \pmod 7$
   $x \equiv 2 \pmod{11}$
9. State and explain Chinese Remainder Theorem with example.
10. How to solve Solving Modular Linear Equations? Explain.
11. Prove that for any integers a, b, and p, if both gcd(a, p) = 1 and gcd(b, p) = 1, then
    gcd(ab, p) = 1.
12. Prove that for all primes p and all integers a, b, if p | ab, then p | a or p | b (or both).
13. Find GCD (198, 128) and value of x and y by using extended Euclidean's algorithm.
14. Show that there are exactly (p - 1)/2 quadratic residues, modulo p.
15. Prove that if a and b are both even, then gcd(a, b) = 2gcd(a/2, b/2).
16. Write down the algorithm for Solving Modular Linear Equations and analyze it.
17. Find gcd (270, 192) by Euclid's Algorithm.
18. Find gcd (70, 92) by Euclid's Algorithm.
19. Find GCD (98, 12) and value of x and y by using extended Euclidean's algorithm.
20. Write down the algorithm for Chinese remainder theorem and analyze it.

❑❑❑