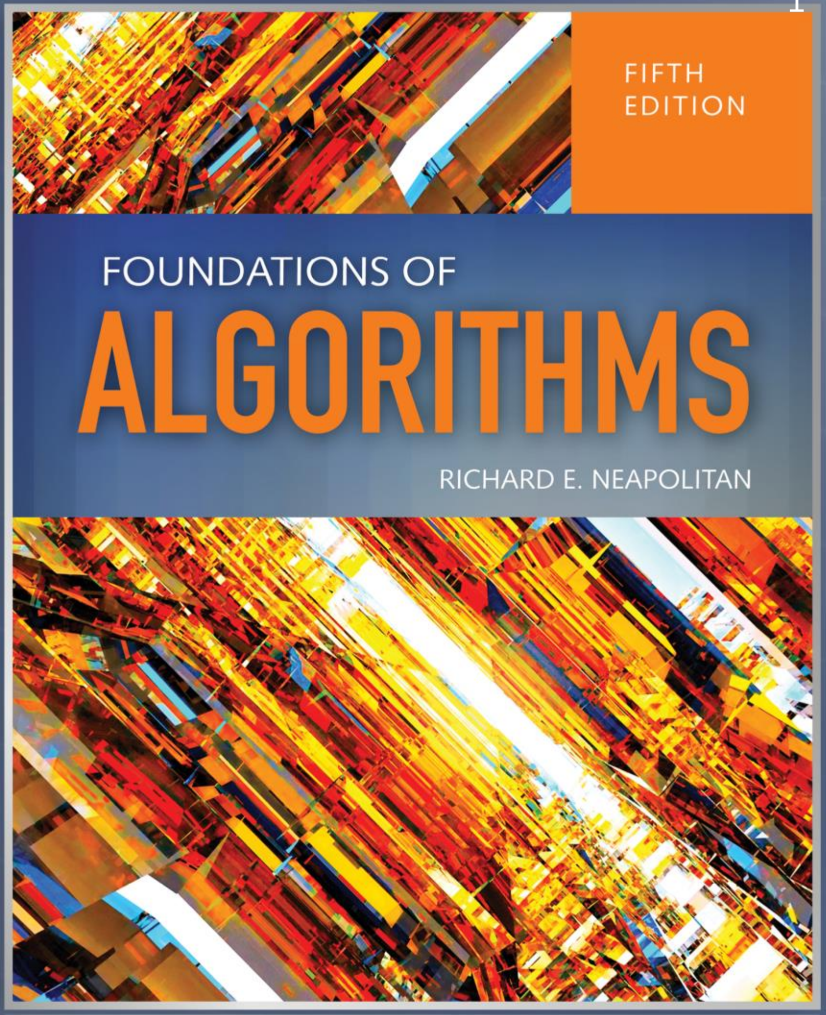

Number-Theoretic Algorithms

Chapter 11

Objectives

- Define prime number
- Define greatest common divisor
- Develop the theoretic basis for Euclid’s Algorithm to determine the greatest common divisor of two integers.
- Determine the worst-case complexity analysis of Euclid’s Algorithm
- Develop the theoretic basis for an algorithm to solve modular linear equations
- Describe public key encryption
- Describe the steps taken in the RSA cryptosystem

Number Theory

- Number theory is the branch of mathematics concerned with the properties of the integers.
- Number-theoretic algorithms are algorithms that solve problems involving the integers
- Cryptography: important application of number-theoretic algorithms

Definitions

- Z = { . . . , -2, -1, 0, 1, 2, . . . } set of integers
- Any two integers n, h ε Z, say h ***divides*** n, denoted h|n if there is some integer k such that n = kh
- If h|n we say
  - n is ***divisible*** by h
  - n is a ***multiple*** of h
  - H is a ***divisor*** or ***factor*** of n
- Prime Number: an integer n > 1 whose only positive divisors are 1 and n
  - Prime number has no factors
- Composite Number: has at least one factor

Greatest Common Divisor

- If h|n and h|m, h is called a common divisor of n and m
- If n and m are not both 0, ***the greatest common divisor*** of n and m, gcd(n,m) is the largest integer that divides both n and m

Theorem 11.1

- If $h|n$ and $h|m$ then for any integers i and j
  - $h|(in + jm)$
- Proof: since $h|n$ and $h|m$, there exists integers k and l such that $n = kh$ and $m = lh$
  - Therefore: $in + jm = ikh + ilh = (ik + jl)h$, which means $h|(in + jm)$

More Definitions

- For any two integers n and m where m !=0, the quotient q of n divided by m is given by
  - $q = \lfloor n/m \rfloor$
  - The remainder r of dividing n by m is r = n – qm
- Remainder denoted n mod m
- If m > 0, 0 <= r < m
- If m < 0, m < r <=0
- 10.1:
  - n = qm + r AND
  - m > 0, 0 <= r <m; m <0, m < r <=0

Theorem 11.2

- Let n and m be integers, not both 0, and let
  - d = min{in+jm such that I, jεZ and in + jm >0}
  - That is, d is the smallest positive linear combination of n and m, then d = gcd(n,m)

Corollary 11.1

- Suppose n and m are integers, not both 0. Then every common divisor of n and m is a divisor of gcd(n,m). That is if h|n and h|m, then h|gcd(n,m)
- Proof: by Theorem 10.2, gcd(n,m) is a linear combination of n and m. Proof follows from Theorem 10.1

Theorem 11.3

- Suppose we have integers n >= 0 and m >0. If r = n mod m, then gcd(n,m) = gcd(m,r)

Prime Factorization

- Every integer > 1 can be written as a unique product of primes
- Two integers n and h not both 0 are called relatively prime if gcd(n,h) = 1

Theorem 11.4

- If h and m are relatively prime and h divides nm, then h divides n. i.e. gcd(h,m) = 1 and h|nm implies h|n
- Corollary 10.2
  - Given integers n, m and prime integer p, if p|nm, then p|n or p|m (inclusive)

Corollary 11.2

- Given integers n, m, and prime integer p, if p|nm, then p|n or p|m (inclusive).
- Roof follows easily from Theorem 11.4

Theorem 11.5

- Every Integer n > 1 has a unique factorization as a product of prime numbers.
  - i.e $n = p1^{k1}p2^{k2} \ldots pj^{kj}$ where $p1<p2< \ldots < pj$ are primes and this representation of n is unique.
  - The integer ki is called the order of pi in n
- Proof is by Induction
- Unique factorization theorem and the fundamental theorem of arithmetic

Theorem 11.6

- The gcd(n,m) is a product of the primes that are common to n and m, where the power of each prime in the product is the smaller of its orders I n and m

Least Common Multiple (lcm)

- Concept similar to gcd
- If n and m are both nonzero, lcm(n,m) is the smallest positive integer that they both divide

Theorem 11.7

- The lcm(n,m) is a product of the primes that are common to n and m, where the power of each prime in the product is the larger of its orders in n and m

Euclid’s Algorithm

- Developed by Euclid around 300 B.C.
- Recursively applies Theorem 10.1 to determine the greatest common divisor of two integers

```
int gcd(int n, int m)
{
        if (m == 0)
                return n;
        else
                return gcd(m, n
mod m); //C++ code n % m
}
```

Lemma 11.1

- If n > m >= 1 and the call gcd(n,m) results in k recursive calls where k >= 1, then
  - N >= f_{k+2} and m >= k_{k+1} where f_k is the kth number in the Fibonacci sequence
- Proof is by Induction

Theorem 11.8

- For every integer k >= 1 if n > m >=1 and m <= f_k, the kth number in the Fibonacci sequence, then the call gcd(n,m) results in less than k recursive calls
- Proof follows from Lemma 11.1

Worst-Case Time Complexity of Euclid's Algorithm

- Basic Operation: one bit manipulation in the computation of remainder
- Input size: the number of bits s it takes to encode n and the number of bits t it takes to encode m
  - $s = \lfloor \lg m \rfloor + 1 \quad t = \lfloor \lg n \rfloor + 1$
- W(s,t) ε O(st)

Corollary 11.6

- The equation $[m]_n\, x = [k]_n$ has a solution if and only if $d|k$, where $d = \gcd(n,m)$

Theorem 11.24

- Let d = gcd(n,m) and let i and j be integers such that d = in + jm.
- From Theorem 11.2, i and j exist
- Then the equation $[m]_n\ x = [k]_n$, has solution $x = [jk/d]_n$

Theorem 11.25

- Suppose the equation $[m]_n\ x = [k]_n$ is solvable, $x = [j]_n$ is one solution, and $d = gcd(n,m)$. Then the d distinct solutions of this equation are
  - $[j = ln/d]_n$ for $l = 0, 1, . . . , d-1$

Solve Modular Linear Equations

- Using Corollary 11.6, Theorem 11.24, and theorem 11.25
- Write algorithm to solve modular linear equations
- Algorithm 11.3
- Input size is the number of bits it takes to encode the input
  - $s = \lfloor \lg n \rfloor + 1$ $t = \lfloor \lg m \rfloor + 1$ $u = \lfloor \lg k \rfloor + 1$
- Time complexity includes time complexity of Algorithm 11.2 O(st) plus the time complexity of the for l loop

Solve Modular Linear Equations

- Time complexity is worst-case exponential in terms of the input size

```
void solve_linear (int n, int m, int k)
{
        index l;
        int i, j, d;
        Euclid(n, m, d, I, j); // call Algorithm
11.2
        if (d|k)
                for (l = 0;l<=d-1;l++)
                        cout<<[jk/d =
ln/d]n;
}
```

Algorithm 11.4

- Compute Modular Power
- Uses the method of repeated squaring

Polynomial Determine Prime

- Algorithm 11.5
- Returns true if an integer n is prime and false if n is composite
- Worst-case time complexity
  - $W(s) \in O(s^{12})$

Public-Key Cryptosystems

- Public key
- Secret key
- Network for sending messages among participants

RSA Cryptosystem

- Find large primes

RSA public-key cryptosystem steps

1. Select two very large prime number p and q
2. Compute n = pq $\vartheta(n) = (p – 1)(q – 1)$; Formula for $\vartheta(n)$ is owing to Theorem 11.17
3. Select a small prime number g that is relatively prime to $\vartheta(n)$
4. Using algorithm 11.3 compute the multiplicative inverse $[h]_{\vartheta(n)}$ of $[g]_{\vartheta(n)}$. That is $[g]_{\vartheta(n)}$ $[h]_{\vartheta(n)}$ = $[l]_{\vartheta(n)}$ Owing to Corollary 11.8, this inverse exists and is unique.
5. Let pkey = (ng) be the public key, and skey = (n,h) be the secret key