CS483-15 Algorithms with Numbers

Instructor: Fei Li

Room 443 ST II

Office hours: Tue. & Thur. 1:30pm - 2:30pm or by appointments

lifei@cs.gmu.edu with subject: CS483

http://www.cs.gmu.edu/∼ lifei/teaching/cs483_fall07/

Part of the slides is based on the book "Algorithms" by S. Dasgupta, C. Papadimitriou, and U. Vazirani.

Outline

- Primality testing
- Cryptography
- Universal hashing

Factoring vs. Primality

- FACTORING: Given a number N, express it as a product of its prime factors.
- PRIMALITY: Given a number N, determine whether it is a prime.

Factoring vs. Primality

- FACTORING: Given a number N, express it as a product of its prime factors.

  The fastest method for factoring a number N takes time exponential in the number of bits N.

- PRIMALITY: Given a number N, determine whether it is a prime.

  There are efficient algorithms for PRIMALITY.

- This strange disparity between these 2 intimately related problems lies the heart of current secure communication.

Basic Arithmetic

- **Modular arithmetic**: How do we handle numbers that are significantly large?
- x **modulo** N: The remainder when x is divided by N. $r = x$ modulo N if $x = q \cdot N + r$ with $0 \leq r < N$.
- x **are** y **are congruent modulo** N if they differ by a multiple of N.

$$x \equiv y \pmod{N} \Leftrightarrow N \text{ divides } (x - y).$$

E.g. $253 \equiv 13 \pmod{60}$. 253 minutes is 4 hours and 13 minutes.
$59 \equiv -1 \pmod{60}$.

Basic Arithmetic

- **Modular arithmetic**: Modular arithmetic deals with all the integers, but divides N *equivalent classes*, each of the form $\{i + k \cdot N, k \in \mathbb{Z}\}$ for some i between 0 and $N - 1$.

- Some rules
  - If $x \equiv x' \pmod N$ and $y \equiv y' \pmod N$, then

$$x + y \equiv x' + y' \pmod N \text{ and } x \cdot y \equiv x' \cdot y' \pmod N.$$

  - **Associatively**: $x + (y + z) \equiv (x + y) + z \pmod N$.
  - **Commutativity**: $x \cdot y \equiv y \cdot x \pmod N$.
  - **Distributively**: $x \cdot (y + z) \equiv x \cdot y + x \cdot z \pmod N$.

Exercises

- $2^{345} \equiv ? \pmod{31}$.

Exercises

- $2^{345} \equiv ? \pmod{31}$.
- $2^{345} \equiv (2^5)^{69} \equiv (32)^{69} \equiv 1^{69} \equiv 1 \pmod{31}$.
- Consider the question: compute x^y mod N for values of x, y, and N that are several hundreds bits long.

  Can this be done quickly?

  If x and y are 20-bits, how long the size of x^y?

  $(2^{19})^{2^{19}} = 2^{19 \cdot 524288}$, about 10 million bits long.

Modular Exponentiation

- Compute x^y mod N for values of x, y, and N that are several hundreds bits long.

$$x \text{ mod } N \longrightarrow x^2 \text{ mod} N \longrightarrow x^3 \text{ mod } N \longrightarrow \cdots \longrightarrow x^y \text{ mod} N.$$

If y is 500 bits long, we need to perform $y - 1 \approx 2^{500}$ multiplications.

- An alternative approach:

$$x \text{ mod } N \longrightarrow x^2 \text{ mod} N \longrightarrow x^4 \text{ mod } N \longrightarrow \cdots \longrightarrow x^{2^{\lfloor \log y \rfloor}} \text{ mod} N.$$

Each takes $O(\log^2 N)$ time to compute, and there are only $\log y$ multiplications.

$x^y = (x^{\lfloor y/2 \rfloor})^2$ if y is even.

$x^y = x \cdot (x^{\lfloor y/2 \rfloor})^2$ if y is odd.

Let n be the largest size in bits of x, y, and N. Running time: $O(n^3)$.

Extension of Euclid Algorithm

- Assume the instructor of CS483 claims that d is the greatest common divisor of a and b, how can we check this?
- **Lemma**: If d divides both a and b, and $d = a \cdot x + b \cdot y$ for some integers of x and y, then necessarily $d = gcd(a, b)$.
- **Proof**:
  1. $d \leq gcd(a, b)$.
  2. $gcd(a, b)$ must divide $a \cdot x + b \cdot y$. So, $gcd(a, b)$ divides d, $gcd(a, b) \leq b$.
- **Example**: $gcd(13, 4) = 1$ since $13 \cdot 1 + 4 \cdot (-3) = 1$.

Extension of Euclid Algorithm

- Input: 2 positive integers a and b. with $a \geq b \geq 0$.
- Output: Integers x, y, and d such that $d = gcd(a, b)$ and $a \cdot x + b \cdot y = d$.

Algorithm 0.1: EXT-GCD(a, b)

if $b = 0$

return $((1, 0, a))$

else

$$\begin{cases} (x', y', d) = \texttt{ext} - \texttt{gcd}(b, a \bmod b) \\ \textbf{return } ((y', x' - \lfloor a/b \rfloor y', d)) \end{cases}$$

- **Proof**: mathematical induction.

Modular division

- In real arithmetic, every number $a \neq 0$ has an inverse $1/a$.
- x is the multiplicative inverse of a modulo N if $ax \equiv 1 \pmod{N}$.
- **Example**: Compute 11^{-1} mod 25.

  $(1.)$ Use extended Euclid algorithm, $15 \cdot 25 - 34 \cdot 11 = 1$.

  $(2.)$ Reduce both sides modulo 25, we have $-34 \cdot 11 \equiv 1$ mod 25. So, $-34 \equiv 16$ mod 25 is the inverse of 11 mod 25.

- $gcd(a, N)$ divides ax mod N because $gcd(a, N) = ax + Ny$.

  If $gcd(a, N) > 1$, $ax \not\equiv 1$ mod N. a cannot have a multiplicative inverse modulo N.

Modular Division Theorem

- For any a mod N, a has a multiplicative inverse modulo N if and only if it is relatively prime to N.
- When this inverse exists, it can be found in time $O(n^3)$ (where as usual n denotes the number of bits of N) by running the the extended Euclid algorithm.

Outline

- Primality testing
- Cryptography
- Universal hashing

Primality Testing

- **Fermat's Little Theorem**

  If p is a prime, then for every $1 \le a < p$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

- **Proof**.

  The numbers $a \cdot i \pmod{p}$ are distinct because if $a \cdot i \equiv a \cdot j \pmod{p}$, then, dividing both sides by a gives $i \equiv j \pmod{p}$.

$$S = \{1, 2, \ldots, p-1\} = \{a \cdot 1 \bmod p, a \cdot 2 \bmod p, \ldots, a \cdot (p-1) \bmod p\}$$

$$(p-1)! \equiv a^{p-1} \cdot (p-1)! \pmod{p}.$$

Fermat's Test

- If $a^{N-1} \equiv 1$ mod N?

  Pass: N is a prime.

  Fail: N is a composite.

- **Lemma** If $a^{N-1} \not\equiv 1 \pmod N$ for some a relatively prime to N, then, it must hold for at least half the choices of $a < N$.

  Proof: Fix some value of a for which $a^{N-1} \not\equiv 1 \pmod N$. Every $b < N$ that passes Fermat's test with respect to N has a twin $a \cdot b$ that fails the test

$$(a \cdot b)^{N-1} \equiv a^{N-1} \cdot b^{N-1} \equiv a^{N-1} \not\equiv 1 \text{ mod } N.$$

- Pick positive integers $a_1, a_2, \ldots, a_k < N$ at random

  If $a_i^{N-1} \equiv 1 \; (\text{mod} N)$ for $i = 1, 2, \ldots, k$, then, output Y, else output N.

  The error of N is not a prime is low: $\frac{1}{2^k}$.

Cryptography

- Alice sends msg x to Bob.
- $x \rightarrow e(x)$.
- $x \leftarrow d(e(x))$.
- $e(x)$ to eavesdropper Eve.
- Ideally, $e(.)$ is chosen that without knowing $d(.)$.

Private-key Scheme

- Alice and Bob meet beforehand and choose a string r of the same length.

$$e :< \text{message} > \;\longrightarrow\; < \text{encoded message} >$$

$$e_r(11110000) = 11110000 \bigoplus 01110010 = 10000010$$

$$e_r(e_r(x)) = (x \bigoplus r) \bigoplus r = x \bigoplus (r \bigoplus r) = x \bigoplus \bar{0} = x.$$

Public Key Cryptography

- **Public-key cryptography**: anybody can send a message to anybody else using publicly available information.
- Each person has a public key known to the whole world and a secret key known only to him- or herself.
- When Alice wants to send message x to Bob, she encodes it using Bob's public key. Bob decrypts it using his secret key.
- **Approach**: Think of messages from Alice to Bob as numbers modulo N.

Public Key Cryptography

- **Property**: Pick any 2 primes p and q and let $N = p \cdot q$. For any e relatively prime to $(p-1) \cdot (q-1)$.

  1. The mapping $x \rightarrow x^e$ mod N is a bijection on $\{0, 1, \ldots, N-1\}$.
  2. The inverse mapping is easily realized. Let d be the inverse of e mod$(p-1) \cdot (q-1)$. Then, $\forall x \in \{0, 1, \ldots, N-1\}$.

$$(x^e)^d \equiv x \text{ mod } N.$$

- The first property says $x \rightarrow x^e$ mod N is a reasonable way to encode x, given (N, e) is Bob's public key.

- Bob uses d to decrypt x.

Proof of RSA

1. $(2.)$ implies $(1.)$ since the mapping is invertible.

2. e is invertible module $(p-1)\cdot(q-1)$ because e is relatively prime to this number.

3. $e \cdot d \equiv 1 \bmod (p-1)\cdot(q-1)$, then, $e \cdot d = 1 + k \cdot (p-1)\cdot(q-1)$ for some k. Show

$$x^{e \cdot d} - x = x^{1+k\cdot(p-1)\cdot(q-1)} - x$$

is always 0 modulo N.

Since p and q are primes, using Fermat's theorem, we can prove above statement as this expression is divisible by the produce p and q.

RSA: Ron Rivest, Adi Shamir and Leonard Adleman at MIT

➢ RSA

- Bob picks up 2 large prime numbers p and q. His public key is (N, e), where $N = p \cdot q$ and e is relatively prime to $(p-1) \cdot (q-1)$. Bob's secret key is d, the inverse of e modulo $(p-1) \cdot (q-1)$. (Use extended Euclid algorithm to get d).
- Alice sends Bob $y = x^e \bmod N$. (Use efficient modular exponentiation algorithm.)
- Bob decodes x by computing $y^d \bmod N$.

➢ Basic

a. Given N, e, and $y = x^e \bmod N$, it is computational intractable to determine x.

b. FACTORING is HARD.