Lecture 21: Primes, iterates, GCD, modular exponentiation

Dave Naumann

Department of Computer Science
Stevens Institute of Technology

CS 135 Discrete Structures Spring 2015

Outline of lecture

Iterates

Prime numbers

Number representations and algos

Iterated composition

Recall from lect 19: the reflexive, transitive closure of a relation R is written R^* and defined by $R^* = (\bigcup_{i \in \mathbf{N}} R^i)$ where R^i is defined by $R^0 = id$ and $R^{i+1} = R \circ R^i$

Iterated composition

Recall from lect 19: the reflexive, transitive closure of a relation R is written R^* and defined by $R^* = (\bigcup_{i \in \mathbf{N}} R^i)$ where R^i is defined by $R^0 = id$ and $R^{i+1} = R \circ R^i$

In the case of a unary function f on some set, $f^0(x) = x$, $f^1(x) = f(x)$, $f^2(x) = f(f(x))$, $f^3(x) = f(f(f(x)))$,....

Notation hazard: superscript sometimes used for exponent, sometimes for iterate. Rosen writes $f^{(2)}$ to be clear, since for numeric functions, $f^2(x)$ could plausibly mean $f(x) \cdot f(x)$.

Iterated composition

Recall from lect 19: the reflexive, transitive closure of a relation R is written R^* and defined by $R^* = (\bigcup_{i \in \mathbf{N}} R^i)$ where R^i is defined by $R^0 = id$ and $R^{i+1} = R \circ R^i$

In the case of a unary function f on some set, $f^0(x) = x$, $f^1(x) = f(x)$, $f^2(x) = f(f(x))$, $f^3(x) = f(f(f(x)))$, $\ldots$.

Notation hazard: superscript sometimes used for exponent, sometimes for iterate. Rosen writes $f^{(2)}$ to be clear, since for numeric functions, $f^2(x)$ could plausibly mean $f(x) \cdot f(x)$.

Example: $sqr^0(3) = id(3) = 3$, $sqr^1(3) = sqr(3) = 9$, $sqr^2(3) = sqr(sqr(3)) = 81$

Iterates and sequences

Consider this recursive definition of a sequence of integers:

$x_0 = 3$ and $x_{n+1} = 5 \cdot x_n \bmod 14$

3, 1, 5, 11, 13, 9, . . .

Iterates and sequences

Consider this recursive definition of a sequence of integers:

$x_0 = 3$ and $x_{n+1} = 5 \cdot x_n \ \mathbf{mod}\ 14$
3, 1, 5, 11, 13, 9, . . .

(Recall that a sequence can be viewed as function on naturals; here x is that function and x_n is notation for $x(n)$.)

Iterates and sequences

Consider this recursive definition of a sequence of integers:

$x_0 = 3$ and $x_{n+1} = 5 \cdot x_n \ \mathbf{mod}\ 14$
3, 1, 5, 11, 13, 9, ...

(Recall that a sequence can be viewed as function on naturals; here x is that function and x_n is notation for $x(n)$.)

Define $g(m) = 5 \cdot m \ \mathbf{mod}\ 14$, to reformulate:
$x_0 = 3$ and $x_{n+1} = g(x_n)$
Now $x_1 = g(x_0)$, $x_2 = g(g(x_0)) = g^2(x_0)$, $x_3 = g^3(x_0)$, ...

Primes

$a \mid b$ means $\exists d\ (b = ad)$ (a divides b, a is a factor of b, b is a multiple of a)

A *prime* is a natural n such that $n > 1$ and the only positive factors of n are 1 and n. In other words, n has exactly two positive factors. A non-prime is called a *composite*.

Fundamental Theorem of Arithmetic (lecture 9): Any natural number $n > 1$ can be written as a product of primes.

Factoring

It's straightforward to define a procedure to check whether a number is prime, and therefore to find primes; also to find factorization. How?

But for extremely large numbers (e.g., used to encode pictures or texts that we want to encrypt), arithmetic operations take time! (E.g., addition is linear in the number of bits.) Factoring large numbers is believed to be inherently difficult, in the sense of computational complexity. Some encryption schemes make use of numbers of the form $p \cdot q$ where p and q are large primes.

Greatest common divisor

For integers $a, b, c, d \ldots$

Define $gcd(a, b)$ by $gcd(a, b) = max\{d \mid (d \mid a) \wedge (d \mid b)\}$
Note that the set contains at least 1, so max is well defined.

Greatest common divisor

For integers $a, b, c, d \ldots$

Define $gcd(a, b)$ by $gcd(a, b) = max\{d \mid (d \mid a) \wedge (d \mid b)\}$
Note that the set contains at least 1, so max is well defined.

For a and b to be *relatively prime* means $gcd(a, b) = 1$.

Greatest common divisor

For integers $a, b, c, d \ldots$

Define $gcd(a, b)$ by $gcd(a, b) = max\{d \mid (d \mid a) \wedge (d \mid b)\}$
Note that the set contains at least 1, so max is well defined.

For a and b to be *relatively prime* means $gcd(a, b) = 1$.

A set of integers S is *pairwise relatively prime*: means any two elements of S are relatively prime.

Least common multiple

Define $lcm(a, b)$ by $lcm(a, b) = min\{d \mid d > 0 \wedge (a \mid d) \wedge (b \mid d)\}$

Theorem: for any a and b in $\mathbf{Z}^+$, we have
$a \cdot b = gcd(a, b) \cdot lcm(a, b)$
(For the proof, think about prime factorization.)

Base b expansion

The base 2 representation of 13 is $(1101)_2$

For $b > 1$ and $n > 0$, the base b representation of n is written $(a_k \ a_{k-1} \ \dots \ a_1 \ a_0)_b$ where

- $k \geqslant 0$
- each a_i is in $0..b-1$
- $a_k \neq 0$ (though sometimes this condition is dropped)
- $n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a_0$

(Here superscript means exponent. The subscript b just indicates what base is intended.)

Arithmetic algorithms

For large numbers, measure complexity in terms of the number of bits in the base 2 representation. (Two's complement isn't much different.)

Addition is linear; division quadratic, multiplication a little better.

Euclid's algorithm

Lemma: if $a > b$ then $gcd(a, b) = gcd(a - b, b)$ (why?) Also $gcd(a, a) = a$.

```
Assume a>0 and b>0
x := a; y := b;
while x ≠ y {
   if x > y then x := x - y;
             else y := y - x;
}
Assert x = gcd(a, b)
```

Euclid's algorithm

Lemma: if $a > b$ then $gcd(a, b) = gcd(a - b, b)$ (why?) Also $gcd(a, a) = a$.

```
Assume a>0 and b>0
x := a; y := b;
while x ≠ y {
  if x > y then x := x - y;
           else y := y - x;
}
Assert x = gcd(a, b)
```

What's invariant? What decreases but is bounded?

Euclid's algorithm using division

Lemma: if $a = bq + r$ then $gcd(a, b) = gcd(b, r)$ (why?)

```
Assume a > b > 0
x := a;
y := b;
while y ≠ 0 {
   r := x mod y;
   x := y;
   y := r;
}
Assert x = gcd(a,b)
```

Euclid's algorithm using division

Lemma: if $a = bq + r$ then $gcd(a, b) = gcd(b, r)$ (why?)

```
Assume a > b > 0
x := a;
y := b;
while y ≠ 0 {
   r := x mod y;
   x := y;
   y := r;
}
Assert x = gcd(a,b)
```

Number of divisions is $O(log\ b)$ (see Rosen if interested)

Review/exercises

Definition of "congruent modulo m", for $m > 0$, is

$$\boxed{a \equiv b \pmod{m} \quad \equiv \quad m \mid (a - b)}$$

Review/exercises

Definition of "congruent modulo m", for $m > 0$, is

$$\boxed{a \equiv b \pmod{m} \quad \equiv \quad m \mid (a - b)}$$

div and **mod** are defined by this property:

$$\boxed{\text{For } a \in \mathbf{Z} \text{ and } m \in \mathbf{Z}^{+}, \text{ we have } a = (a \ \mathbf{div}\ m) * m + (a \ \mathbf{mod}\ m)}$$

Note: **mod** operation versus $\ldots \equiv \ldots \pmod{m}$ relation.

Review/exercises

Definition of "congruent modulo m", for $m > 0$, is

$$\boxed{a \equiv b \pmod{m} \quad \equiv \quad m \mid (a - b)}$$

div and **mod** are defined by this property:

$$\boxed{\text{For } a \in \mathbf{Z} \text{ and } m \in \mathbf{Z}^{+}, \text{ we have } a = (a \operatorname{\mathbf{div}} m) * m + (a \operatorname{\mathbf{mod}} m)}$$

Note: **mod** operation versus $\ldots \equiv \ldots \pmod{m}$ relation.

Important properties to use in following algorithm:

$$(a + b) \operatorname{\mathbf{mod}} m = ((a \operatorname{\mathbf{mod}} m) + (b \operatorname{\mathbf{mod}} m)) \operatorname{\mathbf{mod}} m$$

$$ab \operatorname{\mathbf{mod}} m = ((a \operatorname{\mathbf{mod}} m)(b \operatorname{\mathbf{mod}} m)) \operatorname{\mathbf{mod}} m$$

Proving the properties

Thm 3: $a \equiv b \pmod{m}$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$

Proving the properties

Thm 3: $a \equiv b \pmod{m}$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$

Thm 4: $a \equiv b \pmod{m}$ iff $\exists k : \mathbf{Z}.\ a = km + b$

Proving the properties

Thm 3: $a \equiv b \pmod{m}$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$

Thm 4: $a \equiv b \pmod{m}$ iff $\exists k : \mathbf{Z}.\ a = km + b$

Proof of Thm 4: (with justifications left to you)

$a \equiv b \pmod{m}$ iff $m \mid a - b$ iff $(\exists k.\ a - b = km)$ iff $(\exists k.\ a = km + b)$.

Proving the properties

Thm 3: $a \equiv b (\bmod m)$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$

Thm 4: $a \equiv b (\bmod m)$ iff $\exists k : \mathbf{Z}.\ a = km + b$

Proof of Thm 4: (with justifications left to you)

$a \equiv b (\bmod m)$ iff $m \mid a - b$ iff $(\exists k.\ a - b = km)$ iff $(\exists k.\ a = km + b)$.

Thm 5: If $a \equiv b (\bmod m)$ and $c \equiv d (\bmod m)$ then

$a + c \equiv b + d (\bmod m)$ and $ac \equiv bd (\bmod m)$.

Proving the properties

Thm 3: $a \equiv b \pmod{m}$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$

Thm 4: $a \equiv b \pmod{m}$ iff $\exists k : \mathbf{Z}.\ a = km + b$

Proof of Thm 4: (with justifications left to you)

$a \equiv b \pmod{m}$ iff $m \mid a - b$ iff $(\exists k.\ a - b = km)$ iff $(\exists k.\ a = km + b)$.

Thm 5: If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then

$a + c \equiv b + d \pmod{m}$ and $ac \equiv bd \pmod{m}$.

Proof of Thm 5:

1. $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ (by assumption)
2. $a = km + b$ and $c = k'm + d$ for some k, k' (from 1 by Thm 4)
3. $a + c = (k + k')m + (b + d)$ (from 2 by arith)
4. $a + c \equiv b + d \pmod{m}$ (from 3 by Thm 4)
5. $ac = (km + b)(k'm + d)$ (from 2 by arith)
6. $ac = (\ldots \cdot m) + bd$ (from 5 by arith)
7. $ac \equiv bd \pmod{m}$ (from 6 by Thm 4)

(so we get Thm 5 by discharge hypothesis, using lines 4 and 7)

More properties

Thm 3: $a \equiv b (\operatorname{mod} m)$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$
Thm 4: $a \equiv b (\operatorname{mod} m)$ iff $\exists k : \mathbf{Z}.\ a = km + b$
Thm 5: If $a \equiv b (\operatorname{mod} m)$ and $c \equiv d (\operatorname{mod} m)$ then
$a + c \equiv b + d (\operatorname{mod} m)$ a and $ac \equiv bd (\operatorname{mod} m)$.

Corollary 2: $(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$

More properties

Thm 3: $a \equiv b \pmod{m}$ iff $a \bmod m = b \bmod m$ $(\forall m > 0, \forall a, b)$
Thm 4: $a \equiv b \pmod{m}$ iff $\exists k : \mathbf{Z}.\ a = km + b$
Thm 5: If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then
$a + c \equiv b + d \pmod{m}$ a and $ac \equiv bd \pmod{m}$.

Corollary 2: $(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$

Proof:

1. $a \equiv (a \bmod m) \pmod{m}$ and $b \equiv (b \bmod m) \pmod{m}$ (why?)
2. $a + b \equiv (a \bmod m) + (b \bmod m) \pmod{m}$ (from 1 by Thm 5)
3. $(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$
(from 2 by Thm 3)

Modular exponentiation

For large numbers, it's impractical to compute $b^n \ \mathbf{mod}\ m$ by first computing b^n.

Modular exponentiation

For large numbers, it's impractical to compute $b^n \mathbf{\ mod\ } m$ by first computing b^n.

Suppose n has expansion $(a_{k-1} \ldots a_1 a_0)_2$, i.e., n equals $a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \ldots + a_1 \cdot 2^1 + a_0$ with each a_i in $\{0, 1\}$.

Modular exponentiation

For large numbers, it's impractical to compute $b^n \textbf{ mod } m$ by first computing b^n.

Suppose n has expansion $(a_{k-1} \ldots a_1 a_0)_2$, i.e., n equals $a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \ldots + a_1 \cdot 2^1 + a_0$ with each a_i in $\{0, 1\}$.

Observe that

$$
\begin{aligned}
& b^n \textbf{ mod } m \\
= \; & b^{(a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \ldots + a_1 \cdot 2^1 + a_0)} \textbf{ mod } m \\
= \; & (b^{a_{k-1} \cdot 2^{k-1}} \cdot b^{a_{k-2} \cdot 2^{k-2}} \cdot \ldots \cdot b^{a_1 \cdot 2^1} \cdot a_0) \textbf{ mod } m \\
= \; & ((b^{a_{k-1} \cdot 2^{k-1}} \textbf{ mod } m) \cdot \ldots \cdot (b^{a_1 \cdot 2^1} \textbf{ mod } m) \cdot (a_0 \textbf{ mod } m)) \textbf{ mod } m
\end{aligned}
$$

Modular exponentiation

For large numbers, it's impractical to compute $b^n \textbf{ mod } m$ by first computing b^n.

Suppose n has expansion $(a_{k-1} \ldots a_1 a_0)_2$, i.e., n equals $a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \ldots + a_1 \cdot 2^1 + a_0$ with each a_i in $\{0, 1\}$.

Observe that

$$\begin{aligned}
& b^n \textbf{ mod } m \\
= \; & b^{(a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \ldots + a_1 \cdot 2^1 + a_0)} \textbf{ mod } m \\
= \; & (b^{a_{k-1} \cdot 2^{k-1}} \cdot b^{a_{k-2} \cdot 2^{k-2}} \cdot \ldots \cdot b^{a_1 \cdot 2^1} \cdot a_0) \textbf{ mod } m \\
= \; & ((b^{a_{k-1} \cdot 2^{k-1}} \textbf{ mod } m) \cdot \ldots \cdot (b^{a_1 \cdot 2^1} \textbf{ mod } m) \cdot (a_0 \textbf{ mod } m)) \textbf{ mod } m
\end{aligned}$$

and also $b^{2^i} = b^{2 \cdot 2^{i-1}} = b^{2^{i-1} + 2^{i-1}} = b^{2^{i-1}} \cdot b^{2^{i-1}}$.

Modular exponentiation

For large numbers, it's impractical to compute $b^n \textbf{ mod } m$ by first computing b^n.

Suppose n has expansion $(a_{k-1} \dots a_1 a_0)_2$, i.e., n equals $a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \dots + a_1 \cdot 2^1 + a_0$ with each a_i in $\{0, 1\}$.

Observe that

$$
\begin{aligned}
& b^n \textbf{ mod } m \\
= \; & b^{(a_{k-1}\cdot 2^{k-1} + a_{k-2}\cdot 2^{k-2} + \dots + a_1 \cdot 2^1 + a_0)} \textbf{ mod } m \\
= \; & (b^{a_{k-1}\cdot 2^{k-1}} \cdot b^{a_{k-2}\cdot 2^{k-2}} \cdot \ldots \cdot b^{a_1 \cdot 2^1} \cdot a_0) \textbf{ mod } m \\
= \; & ((b^{a_{k-1}\cdot 2^{k-1}} \textbf{ mod } m) \cdot \ldots \cdot (b^{a_1\cdot 2^1} \textbf{ mod } m) \cdot (a_0 \textbf{ mod } m)) \textbf{ mod } m
\end{aligned}
$$

and also $b^{2^i} = b^{2 \cdot 2^{i-1}} = b^{2^{i-1} + 2^{i-1}} = b^{2^{i-1}} \cdot b^{2^{i-1}}$.

Note: $b^{2^i} = c_i$ where the sequence c is defined by iterated squaring: $c_0 = b$ and $c_i = c_{i-1} \cdot c_{i-1}$

Modular exponentiation algorithm

Preceding theory says we can iteratively get the terms b^{2^i}, i.e., $b^{a_i \cdot 2^i}$ when $a_i = 1$. And also apply $\mathbf{mod}\ m$ to each $b^{a_i \cdot 2^i}$ as we go.

```
Assume b, n, m positive integers with expansion of n in array a.
x := 1;
power := b mod m;
for i := 0 to k - 1 {
   if a[i] = 1 then x := (x * power) mod m;
   power := (power * power) mod m;
}
```

Assert $x = b^n\ \mathbf{mod}\ m$

Induction exercises

Exercises 31–34 in sect 5.1 of Rosen.

Prove that 2 divides $n^2 + n$ whenever n is a positive integer.
(Can also be proved for any integer n, by cases on whether n is even: do the even case first.)
Prove that 3 divides $n^3 + 2n$ whenever n is a positive integer.

Prove that 5 divides $n^5 - n$ whenever n is a nonnegative integer.

Prove that 6 divides $n^3 - n$ whenever n is a nonnegative integer.