Modular Arithmetic

CS70 Summer 2016 - Lecture 7A

David Dinh

01 August 2016

UC Berkeley

Announcements

Midterm 2 scores out.

Homework 7 is out.

Announcements

Midterm 2 scores out.

Homework 7 is out. Longer, but due next Wednesday before class, not next Monday.

There will be no homework 8.

Agenda

Some basic number theory:

- Modular arithmetic
- GCD, Euclidean algorithm, and multiplicative inverses
- Exponentiation in modular arithmetic

Mathematics is the queen of the sciences and number theory is the queen of mathematics. -Gauss

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours?

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours?

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

- What time is it in 2 hours? 3:00!
- What time is it in 5 hours? 6:00!
- What time is it in 15 hours?

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

- What time is it in 2 hours? 3:00!
- What time is it in 5 hours? 6:00!
- What time is it in 15 hours? 16:00!
  - Actually 4:00.
- 16 is the “same as 4” with respect to a 12 hour clock system.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

- What time is it in 2 hours? 3:00!
- What time is it in 5 hours? 6:00!
- What time is it in 15 hours? 16:00!
  - Actually 4:00.
- 16 is the “same as 4” with respect to a 12 hour clock system.
  - Clock time equivalent up to to addition/subtraction of 12.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

- What time is it in 2 hours? 3:00!
- What time is it in 5 hours? 6:00!
- What time is it in 15 hours? 16:00!
  - Actually 4:00.
- 16 is the “same as 4” with respect to a 12 hour clock system.
  - Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours?

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

- What time is it in 2 hours? 3:00!
- What time is it in 5 hours? 6:00!
- What time is it in 15 hours? 16:00!
  - Actually 4:00.
- 16 is the "same as 4" with respect to a 12 hour clock system.
  - Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00!

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Actually 4:00.

16 is the “same as 4” with respect to a 12 hour clock system.

Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00! or 5:00.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Actually 4:00.

16 is the "same as 4" with respect to a 12 hour clock system.

Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00! or 5:00.

$101 = 12 \times 8 + 5$.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Actually 4:00.

16 is the "same as 4" with respect to a 12 hour clock system.

Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00! or 5:00.

$101 = 12 \times 8 + 5$.

5 is the same as 101 for a 12 hour clock system.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Actually 4:00.

16 is the "same as 4" with respect to a 12 hour clock system.

Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00! or 5:00.

$101 = 12 \times 8 + 5$.

5 is the same as 101 for a 12 hour clock system.

Clock time equivalent up to addition of any integer multiple of 12.

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Actually 4:00.

16 is the "same as 4" with respect to a 12 hour clock system.

Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00! or 5:00.

$101 = 12 \times 8 + 5$.

5 is the same as 101 for a 12 hour clock system.

Clock time equivalent up to addition of any integer multiple of 12.

Custom is only to use the representative in $\{12, 1, \ldots, 11\}$

Modular Arithmetic Motivation: Clock Math

If it is 1:00 now.

What time is it in 2 hours? 3:00!

What time is it in 5 hours? 6:00!

What time is it in 15 hours? 16:00!

Actually 4:00.

16 is the "same as 4" with respect to a 12 hour clock system.

Clock time equivalent up to to addition/subtraction of 12.

What time is it in 100 hours? 101:00! or 5:00.

$101 = 12 \times 8 + 5$.

5 is the same as 101 for a 12 hour clock system.

Clock time equivalent up to addition of any integer multiple of 12.

Custom is only to use the representative in $\{12, 1, \ldots, 11\}$

(Almost remainder, except for 12 and 0 are equivalent.)

Congruences

x is congruent to y modulo m, denoted "$x \equiv y \pmod{m}$"...

- if and only if $(x - y)$ is divisible by m (denoted $m|(x - y)$).
- if and only if x and y have the same remainder w.r.t. m.
- $x = y + km$ for some integer k.

(these definitions are equivalent).

Congruences

x is congruent to y modulo m, denoted "$x \equiv y \pmod{m}$"...

- if and only if $(x - y)$ is divisible by m (denoted $m|(x - y)$).
- if and only if x and y have the same remainder w.r.t. m.
- $x = y + km$ for some integer k.

(these definitions are equivalent).

Congruence partitions the integers into equivalence classes ("congruence classes"). For instance, here are equivalence classes mod 7: $\{\ldots, -7, 0, 7, 14, \ldots\}$

Congruences

x is congruent to y modulo m, denoted "$x \equiv y \pmod{m}$"...

- if and only if $(x - y)$ is divisible by m (denoted $m|(x - y)$).
- if and only if x and y have the same remainder w.r.t. m.
- $x = y + km$ for some integer k.

(these definitions are equivalent).

Congruence partitions the integers into equivalence classes ("congruence classes"). For instance, here are equivalence classes mod 7: $\{\ldots, -7, 0, 7, 14, \ldots\}$ $\{\ldots, -6, 1, 8, 15, \ldots\}$

Theorem: If $a \equiv c \pmod{m}$ and $b \equiv d \pmod{m}$, then $a + b \equiv c + d \pmod{m}$ and $a \cdot b = c \cdot d \pmod{m}$.

Theorem: If $a \equiv c \pmod{m}$ and $b \equiv d \pmod{m}$, then $a + b \equiv c + d \pmod{m}$ and $a \cdot b = c \cdot d \pmod{m}$.

Proof: Addition: $(a + b) - (c + d) = (a - c) + (b - d)$. Since $a \equiv c \pmod{m}$ the first term is divisible by m, likewise for the second term. Therefore the entire expression is divisible by m, so $a + b \equiv c + d \pmod{m}$.

Modular Arithmetic

Theorem: If $a \equiv c \pmod{m}$ and $b \equiv d \pmod{m}$, then $a + b \equiv c + d \pmod{m}$ and $a \cdot b = c \cdot d \pmod{m}$.

Proof: Addition: $(a + b) - (c + d) = (a - c) + (b - d)$. Since $a \equiv c \pmod{m}$ the first term is divisible by m, likewise for the second term. Therefore the entire expression is divisible by m, so $a + b \equiv c + d \pmod{m}$.

Multiplication: Let a = $k_1 m + c$ and $b = k_2 m + d$. Then

$$ab = (k_1 m + c)(k_2 m + d) = (k_1 k_2 m + k_1 d + k_2 c)m + cd$$

so $ab \equiv cd \pmod{m}$.

Multiplicative Inverses: Motivation

We have addition, subtraction, and multiplication. What about division?

Multiplicative Inverses: Motivation

We have addition, subtraction, and multiplication. What about division?

What is division? Multiplication by a multiplicative inverse. $x/y = x(1/y)$.

Multiplicative Inverses: Motivation

We have addition, subtraction, and multiplication. What about division?

What is division? Multiplication by a multiplicative inverse. $x/y = x(1/y)$.

Formally, a multiplicative inverse of x is a number y such that $xy = 1$, the multiplicative identity.

Multiplicative Inverses: Motivation

We have addition, subtraction, and multiplication. What about division?

What is division? Multiplication by a multiplicative inverse. $x/y = x(1/y)$.

Formally, a multiplicative inverse of x is a number y such that $xy = 1$, the multiplicative identity.

Is there a concept of multiplicative inverse in modular arithemtic?

Multiplicative Inverses: Motivation

We have addition, subtraction, and multiplication. What about division?

What is division? Multiplication by a multiplicative inverse. $x/y = x(1/y)$.

Formally, a multiplicative inverse of x is a number y such that $xy = 1$, the multiplicative identity.

Is there a concept of multiplicative inverse in modular arithemtic?

When is there a solution to the equation $xy = 1 + km$?

Multiplicative Inverses: Existence

Theorem: If greatest common divisor of x and m, $\gcd(x, m)$, is 1, then x has a multiplicative inverse modulo m.

Multiplicative Inverses: Existence

Theorem: If greatest common divisor of x and m, $\gcd(x, m)$, is 1, then x has a multiplicative inverse modulo m.

Proof: It suffices to show: all elements of $S = \{0x, 1x, \ldots, (m-1)x\}$ are distinct mod m.

Multiplicative Inverses: Existence

Theorem: If greatest common divisor of x and m, $\gcd(x, m)$, is 1, then x has a multiplicative inverse modulo m.

Proof: It suffices to show: all elements of $S = \{0x, 1x, \ldots, (m-1)x\}$ are distinct mod m. Why? Pigeonhole principle. All distinct means that one of them has to correspond to $1 \mod m$.

Multiplicative Inverses: Existence

Theorem: If greatest common divisor of x and m, $\gcd(x, m)$, is 1, then x has a multiplicative inverse modulo m.

Proof: It suffices to show: all elements of $S = \{0x, 1x, \ldots, (m-1)x\}$ are distinct mod m. Why? Pigeonhole principle. All distinct means that one of them has to correspond to $1 \mod m$.

Suppose for contradiction that they are not distinct. Then there exist a, b in $\{0, ..., m-1\}$ such that ax, bx are in the same congruence class $\mod m$, i.e. $(a-b)x = km$ for some integer k.

Multiplicative Inverses: Existence

Theorem: If greatest common divisor of x and m, $\gcd(x, m)$, is 1, then x has a multiplicative inverse modulo m.

Proof: It suffices to show: all elements of $S = \{0x, 1x, \ldots, (m-1)x\}$ are distinct mod m. Why? Pigeonhole principle. All distinct means that one of them has to correspond to $1 \mod m$.

Suppose for contradiction that they are not distinct. Then there exist a, b in $\{0, ..., m-1\}$ such that ax, bx are in the same congruence class $\mod m$, i.e. $(a-b)x = km$ for some integer k.

Since $\gcd(x, m) = 1$, we must have that $m | (a-b)$, which implies that $a - b \geq m$. But $a, b \in \{0, 1, \ldots, m-1\}$, so this is impossible. Contradiction. $\square$

Finding GCD

How do we find GCD of x, m?

Finding GCD

How do we find GCD of x, m?

Naive approach: try every single number in $[1, \min(x, m)]$ and see if it divides x and m both. Keep the biggest number that does.

Finding GCD

How do we find GCD of x, m?

Naive approach: try every single number in $[1, \min(x, m)]$ and see if it divides x and m both. Keep the biggest number that does.

Obviously works, but how long does that take?

Finding GCD

How do we find GCD of x, m?

Naive approach: try every single number in $[1, \min(x, m)]$ and see if it divides x and m both. Keep the biggest number that does.

Obviously works, but how long does that take?

I need $\min(x, m)$ divisions. For 64-bit integers, that means up to $2^64 = 18446744073709551616$ divisions - assuming one division per nanosecond (1 GHz), that's about 585 years to compute a single gcd :(

Euclid to the Rescue

Can we do better?

Euclid to the Rescue

Can we do better?

Lemma: Suppose $d|x$ and $d|y$. Then $d|(x + ay)$ for all integers a.

Euclid to the Rescue

Can we do better?

Lemma: Suppose $d|x$ and $d|y$. Then $d|(x + ay)$ for all integers a.

Proof: Write $x = k_1 d$ and $y = k_2 d$ for some integers k_1, k_2 (we know this is possible because $d|x$ and $d|y$). Then $x + ay = (k_1 + ak_2)d$. $\square$

Euclid to the Rescue

Can we do better?

Lemma: Suppose $d|x$ and $d|y$. Then $d|(x + ay)$ for all integers a.

Proof: Write $x = k_1 d$ and $y = k_2 d$ for some integers k_1, k_2 (we know this is possible because $d|x$ and $d|y$). Then $x + ay = (k_1 + ak_2)d$. $\square$

Theorem: $\gcd(x, y) = \gcd(x, y + ax)$ for all integers a.

Euclid to the Rescue

Can we do better?

Lemma: Suppose $d|x$ and $d|y$. Then $d|(x + ay)$ for all integers a.

Proof: Write $x = k_1 d$ and $y = k_2 d$ for some integers k_1, k_2 (we know this is possible because $d|x$ and $d|y$). Then $x + ay = (k_1 + ak_2)d$. $\square$

Theorem: $\gcd(x, y) = \gcd(x, y + ax)$ for all integers a.

Proof: Suppose k divides both x and y. Then by the lemma, it divides $y + ax$ as well.

Euclid to the Rescue

Can we do better?

Lemma: Suppose $d|x$ and $d|y$. Then $d|(x + ay)$ for all integers a.

Proof: Write $x = k_1 d$ and $y = k_2 d$ for some integers k_1, k_2 (we know this is possible because $d|x$ and $d|y$). Then $x + ay = (k_1 + ak_2)d$. $\square$

Theorem: $\gcd(x, y) = \gcd(x, y + ax)$ for all integers a.

Proof: Suppose k divides both x and y. Then by the lemma, it divides $y + ax$ as well.

Now suppose k divides both x and $y + ax$. Then again by lemma, it must divide $y + ax - ax = y$.

Euclid to the Rescue

Can we do better?

Lemma: Suppose $d|x$ and $d|y$. Then $d|(x+ay)$ for all integers a.

Proof: Write $x = k_1 d$ and $y = k_2 d$ for some integers k_1, k_2 (we know this is possible because $d|x$ and $d|y$). Then $x + ay = (k_1 + ak_2)d$. $\square$

Theorem: $\gcd(x, y) = \gcd(x, y + ax)$ for all integers a.

Proof: Suppose k divides both x and y. Then by the lemma, it divides $y + ax$ as well.

Now suppose k divides both x and $y + ax$. Then again by lemma, it must divide $y + ax - ax = y$.

Therefore, the set of common divisors of x, y is the same as the set of divisors of $x, y + ax$ which means that the gcd must be the same as well. $\square$

The Euclidean Algorithm

This leads to an algorithm for computing the gcd of x and y (assuming $x \geq y \geq 0$):

1. If y is zero, just return x.

The Euclidean Algorithm

This leads to an algorithm for computing the gcd of x and y (assuming $x \geq y \geq 0$):

1. If y is zero, just return x.
2. Otherwise, let $x' = x - y \left\lfloor \frac{x}{y} \right\rfloor$, and apply the algorithm recursively to find the $\gcd(y, x')$; this is also $\gcd(x, y)$.

($\lfloor k \rfloor$ is the smallest integer less than or equal to x)

The Euclidean Algorithm

This leads to an algorithm for computing the gcd of x and y (assuming $x \geq y \geq 0$):

1. If y is zero, just return x.
2. Otherwise, let $x' = x - y \left\lfloor \frac{x}{y} \right\rfloor$, and apply the algorithm recursively to find the $\gcd(y, x')$; this is also $\gcd(x, y)$.

($\lfloor k \rfloor$ is the smallest integer less than or equal to x)

By the theorem on the previous slide this is guaranteed to give the right result.

The Euclidean Algorithm

This leads to an algorithm for computing the gcd of x and y (assuming $x \geq y \geq 0$):

1. If y is zero, just return x.
2. Otherwise, let $x' = x - y \left\lfloor \frac{x}{y} \right\rfloor$, and apply the algorithm recursively to find the $\gcd(y, x')$; this is also $\gcd(x, y)$.

($\lfloor k \rfloor$ is the smallest integer less than or equal to x)

By the theorem on the previous slide this is guaranteed to give the right result.

How long does it take to run? $O(\log y)$ iterations. Proof: not today.

A lot faster than brute force!

Finding the Inverse with EGCD

Now we have a way to tell if there is an inverse. How do we find the inverse?

Finding the Inverse with EGCD

Now we have a way to tell if there is an inverse. How do we find the inverse?

Theorem: For any integers x, y, there exist integers a, b such that $ax + by = \gcd(x, y)$.

Finding the Inverse with EGCD

Now we have a way to tell if there is an inverse. How do we find the inverse?

Theorem: For any integers x, y, there exist integers a, b such that $ax + by = \gcd(x, y)$.

How do we find the multiplicative inverse $\mod m$? If $\gcd(x, m) = 1$, then we can find a, b such that $ax + bm = 1$.

Finding the Inverse with EGCD

Now we have a way to tell if there is an inverse. How do we find the inverse?

Theorem: For any integers x, y, there exist integers a, b such that $ax + by = \gcd(x, y)$.

How do we find the multiplicative inverse $\mod m$? If $\gcd(x, m) = 1$, then we can find a, b such that $ax + bm = 1$. Equivalently: $ax = 1 - bm \equiv 1 \pmod{m}$.

Finding the Inverse with EGCD

Now we have a way to tell if there is an inverse. How do we find the inverse?

Theorem: For any integers x, y, there exist integers a, b such that $ax + by = \gcd(x, y)$.

How do we find the multiplicative inverse $\pmod m$? If $\gcd(x, m) = 1$, then we can find a, b such that $ax + bm = 1$. Equivalently: $ax = 1 - bm \equiv 1 \pmod m$. So $a = x^{-1} \pmod m$.

Finding the Inverse with EGCD

Now we have a way to tell if there is an inverse. How do we find the inverse?

Theorem: For any integers x, y, there exist integers a, b such that $ax + by = \gcd(x, y)$.

How do we find the multiplicative inverse $\mod m$? If $\gcd(x, m) = 1$, then we can find a, b such that $ax + bm = 1$. Equivalently: $ax = 1 - bm \equiv 1 \pmod{m}$. So $a = x^{-1} \pmod{m}$.

How do we find a, b?

EGCD: Motivation

Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$(3)12 + (-1)35 = 1$.

$a = 3$ and $b = -1$.

The multiplicative inverse of $12 \pmod{35}$ is 3.

EGCD: Motivation

Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$(3)12 + (-1)35 = 1$.

$a = 3$ and $b = -1$.

The multiplicative inverse of $12 \pmod{35}$ is 3.

How do we get there using Euclid?

$$\gcd(35, 12) = \gcd(12, 11) = \gcd(11, 1) = \gcd(1, 0) = 1$$

EGCD: Motivation

Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$(3)12 + (-1)35 = 1$.

$a = 3$ and $b = -1$.

The multiplicative inverse of $12 \pmod{35}$ is 3.

How do we get there using Euclid?

$$\gcd(35, 12) = \gcd(12, 11) = \gcd(11, 1) = \gcd(1, 0) = 1$$

How did we get 11 from 35 and 12? $35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11$.

EGCD: Motivation

Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$(3)12 + (-1)35 = 1$.

$a = 3$ and $b = -1$.

The multiplicative inverse of $12 \pmod{35}$ is 3.

How do we get there using Euclid?

$$\gcd(35, 12) = \gcd(12, 11) = \gcd(11, 1) = \gcd(1, 0) = 1$$

How did we get 11 from 35 and 12? $35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11$. How did gcd get 1 from 12 and 11? $12 - \left\lfloor \frac{12}{11} \right\rfloor 11 = 12 - (1)11 = 1$.

EGCD: Motivation

Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$(3)12 + (-1)35 = 1$.

$a = 3$ and $b = -1$.

The multiplicative inverse of $12 \pmod{35}$ is 3.

How do we get there using Euclid?

$$\gcd(35, 12) = \gcd(12, 11) = \gcd(11, 1) = \gcd(1, 0) = 1$$

How did we get 11 from 35 and 12? $35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11$. How did gcd get 1 from 12 and 11? $12 - \left\lfloor \frac{12}{11} \right\rfloor 11 = 12 - (1)11 = 1$.

What if we work backwards?

$$1 = 12 - 1(11) = 12 - 1(35 - 2(12)) = 3(12) - 1(35) \quad .$$

Just keep back-substituting.

EGCD Algorithm

How do we turn this into an algorithm?

EGCD Algorithm

How do we turn this into an algorithm?

Just run normal GCD but keep track of the coefficients.

EGCD Algorithm

How do we turn this into an algorithm?

Just run normal GCD but keep track of the coefficients.

Extended GCD algorithm.

Inputs: $x \geq y \geq 0$ with $x > 0$. Outputs: integers (d, a, b) where $d = \gcd(x, y) = ax + by$.

EGCD Algorithm

How do we turn this into an algorithm?

Just run normal GCD but keep track of the coefficients.

Extended GCD algorithm.

Inputs: $x \geq y \geq 0$ with $x > 0$. Outputs: integers (d, a, b) where $d = \gcd(x, y) = ax + by$.

1. If $y = 0$, return $(x, 1, 0)$: $x = 1x + 0y$.

EGCD Algorithm

How do we turn this into an algorithm?

Just run normal GCD but keep track of the coefficients.

Extended GCD algorithm.

Inputs: $x \geq y \geq 0$ with $x > 0$. Outputs: integers (d, a, b) where $d = \gcd(x, y) = ax + by$.

1. If $y = 0$, return $(x, 1, 0)$: $x = 1x + 0y$.
2. Otherwise, let (d, a, b) be the return value of the extended GCD algorithm on $(y, x - y \lfloor x/y \rfloor)$.

EGCD Algorithm

How do we turn this into an algorithm?

Just run normal GCD but keep track of the coefficients.

Extended GCD algorithm.

Inputs: $x \geq y \geq 0$ with $x > 0$. Outputs: integers (d, a, b) where $d = \gcd(x, y) = ax + by$.

1. If $y = 0$, return $(x, 1, 0)$: $x = 1x + 0y$.
2. Otherwise, let (d, a, b) be the return value of the extended GCD algorithm on $(y, x - y\lfloor x/y \rfloor)$.
3. Return $(d, b, a - b\lfloor x/y \rfloor)$.

EGCD Algorithm

How do we turn this into an algorithm?

Just run normal GCD but keep track of the coefficients.

Extended GCD algorithm.

Inputs: $x \geq y \geq 0$ with $x > 0$. Outputs: integers (d, a, b) where $d = \gcd(x, y) = ax + by$.

1. If $y = 0$, return $(x, 1, 0)$: $x = 1x + 0y$.
2. Otherwise, let (d, a, b) be the return value of the extended GCD algorithm on $(y, x - y\lfloor x/y \rfloor)$.
3. Return $(d, b, a - b\lfloor x/y \rfloor)$.

Since this is just GCD (except we track some more numbers), $d = \gcd(x, y)$.

Need to show that $d = ax + by$.

EGCD: Proof of Correctness

Proof: by

EGCD: Proof of Correctness

Proof: by induction on y.

EGCD: Proof of Correctness

Proof: by induction on y.

For the base case, $y = 0$. We return $(x, 1, 0)$ and $x = 1x + 0y$, as desired.

EGCD: Proof of Correctness

Proof: by induction on y.

For the base case, $y = 0$. We return $(x, 1, 0)$ and $x = 1x + 0y$, as desired.

Now suppose for induction that extended GCD returns the correct coefficients for all y in $[0, k]$. It suffices to show the claim for $y = k + 1$.

EGCD: Proof of Correctness

Proof: by induction on y.

For the base case, $y = 0$. We return $(x, 1, 0)$ and $x = 1x + 0y$, as desired.

Now suppose for induction that extended GCD returns the correct coefficients for all y in $[0, k]$. It suffices to show the claim for $y = k + 1$.

Return value: $(d, b, a - b \lfloor x/y \rfloor)$ where (d, a, b) is return value of the extended GCD algorithm on $(y, x - y \lfloor x/y \rfloor)$. By inductive hypothesis, (d, a, b) is the correct return value for the recursive call, i.e. $ay + b(x - y \lfloor x/y \rfloor) = d$.

EGCD: Proof of Correctness

Proof: by induction on y.

For the base case, $y = 0$. We return $(x, 1, 0)$ and $x = 1x + 0y$, as desired.

Now suppose for induction that extended GCD returns the correct coefficients for all y in $[0, k]$. It suffices to show the claim for $y = k + 1$.

Return value: $(d, b, a - b\lfloor x/y \rfloor)$ where (d, a, b) is return value of the extended GCD algorithm on $(y, x - y\lfloor x/y \rfloor)$. By inductive hypothesis, (d, a, b) is the correct return value for the recursive call, i.e. $ay + b(x - y\lfloor x/y \rfloor) = d$.

Therefore:

$$d = ay + b(x - y\lfloor x/y \rfloor) = ay + bx - by\lfloor x/y \rfloor = bx + (a - \lfloor x/y \rfloor b)y \quad ,$$

as desired. $\square$

More Arithmetic...

We have addition, subtraction, multiplication, and ”division” now.

More Arithmetic...

We have addition, subtraction, multiplication, and ”division” now.

What about exponentiation? After the break.

Break!

Exponentiation: Motivation

Can we just simplify exponentiation under congruence the same way we did with addition and multiplication?

Exponentiation: Motivation

Can we just simplify exponentiation under congruence the same way we did with addition and multiplication?

$$2^6 \equiv 64 \equiv 4 \not\equiv 2^1 \pmod 5 \; .$$

Guess not.

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$
$51^2 = (51) * (51) = 2601 \equiv 60 \pmod{77}$

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$
$51^2 = (51) * (51) = 2601 \equiv 60 \pmod{77}$
$51^4 = (51^2) * (51^2) = 60 * 60 = 3600 \equiv 58 \pmod{77}$

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$
$51^2 = (51) * (51) = 2601 \equiv 60 \pmod{77}$
$51^4 = (51^2) * (51^2) = 60 * 60 = 3600 \equiv 58 \pmod{77}$
$51^8 = (51^4) * (51^4) = 58 * 58 = 3364 \equiv 53 \pmod{77}$

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$
$51^2 = (51) * (51) = 2601 \equiv 60 \pmod{77}$
$51^4 = (51^2) * (51^2) = 60 * 60 = 3600 \equiv 58 \pmod{77}$
$51^8 = (51^4) * (51^4) = 58 * 58 = 3364 \equiv 53 \pmod{77}$
$51^{16} = (51^8) * (51^8) = 53 * 53 = 2809 \equiv 37 \pmod{77}$

Repeated Squaring

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$
$51^2 = (51) * (51) = 2601 \equiv 60 \pmod{77}$
$51^4 = (51^2) * (51^2) = 60 * 60 = 3600 \equiv 58 \pmod{77}$
$51^8 = (51^4) * (51^4) = 58 * 58 = 3364 \equiv 53 \pmod{77}$
$51^{16} = (51^8) * (51^8) = 53 * 53 = 2809 \equiv 37 \pmod{77}$
$51^{32} = (51^{16}) * (51^{16}) = 37 * 37 = 1369 \equiv 60 \pmod{77}$

One way to do this efficiently: repeated squaring. Keep squaring the base and simplifying (since multiplication can easily be simplified under congruence).

Example: compute $51^{43} \pmod{77}$.

$51^1 \equiv 51 \pmod{77}$
$51^2 = (51) * (51) = 2601 \equiv 60 \pmod{77}$
$51^4 = (51^2) * (51^2) = 60 * 60 = 3600 \equiv 58 \pmod{77}$
$51^8 = (51^4) * (51^4) = 58 * 58 = 3364 \equiv 53 \pmod{77}$
$51^{16} = (51^8) * (51^8) = 53 * 53 = 2809 \equiv 37 \pmod{77}$
$51^{32} = (51^{16}) * (51^{16}) = 37 * 37 = 1369 \equiv 60 \pmod{77}$

$$51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 = (60) * (53) * (60) * (51) \equiv 2 \pmod{77} .$$

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute x^1,

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2,$

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4,$

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4, \ldots,$

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lfloor \log y \rfloor}}$.

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lfloor \log y \rfloor}}$.
2. Multiply together x^i where the $(\log(i))$th bit of y (in binary) is 1.

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lfloor \log y \rfloor}}$.
2. Multiply together x^i where the $(\log(i))$th bit of y (in binary) is 1. Example:

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lfloor \log y \rfloor}}$.
2. Multiply together x^i where the $(\log(i))$th bit of y (in binary) is 1.
   Example: $43 = 101011$ in binary.

Repeated Squaring, Formally

To compute $x^y \pmod{n}$:

1. x^y: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lfloor \log y \rfloor}}$.
2. Multiply together x^i where the $(\log(i))$th bit of y (in binary) is 1.
   Example: $43 = 101011$ in binary.

$$x^{43} = x^{32} * x^8 * x^2 * x^1$$

.

How many multiplications required? $O(\log y)$. Much faster than multiplying y times!

Algebraic simplification?

Repeated squaring is less useful when you're dealing with symbolic expressions… what else do we have in our toolbox?

Reduced Residue Systems

Remember that we can divide up the integers into congruence classes mod n for any n.

Any set of n integers, one from each congruence class, is known a **complete residue system** mod n.

One complete residue system mod n: $\{0, 1, 2, ..., n-1\}$.

Reduced Residue Systems

Remember that we can divide up the integers into congruence classes mod n for any n.

Any set of n integers, one from each congruence class, is known a **complete residue system** mod n.

One complete residue system mod n: $\{0, 1, 2, ..., n-1\}$.

A subset of a complete residue system only consisting of numbers relatively prime to n is called a **reduced residue system**.

One reduced residue system mod n: list of all nonnegative numbers smaller than n that are relatively prime to it (i.e. numbers whose gcd with n is 1).

Euler's Totient Function

For $n \geq 1$, the *totient function* $\phi(n)$ denotes the number of elements in any reduced residue system mod n. Equivalently: the number of nonnegative numbers smaller than n that are relatively prime to n.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) I

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} = 1$.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) I

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} = 1$.

Lemma 1: Suppose $\gcd(a, n) = 1$, and $\{a_1, ..., a_n\}$ is a complete residue system mod n. Then for all b, $\{aa_1 + b, ..., aa_n + b\}$ forms a complete residue system mod n.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) I

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} = 1$.

Lemma 1: Suppose $\gcd(a, n) = 1$, and $\{a_1, ..., a_n\}$ is a complete residue system mod n. Then for all b, $\{aa_1 + b, ..., aa_n + b\}$ forms a complete residue system mod n.

Proof of Lemma 1: Since $\gcd(a, n) = 1$, we know that there must exist some c such that $ac \equiv 1 \pmod{n}$.

Now suppose $\{a_1, ..., a_n\}$ is a complete residue system mod n. Then for any integer d, there is a unique k such that $c(d - b) \equiv a_k \pmod{n}$.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) I

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} = 1$.

Lemma 1: Suppose $\gcd(a, n) = 1$, and $\{a_1, ..., a_n\}$ is a complete residue system mod n. Then for all b, $\{aa_1 + b, ..., aa_n + b\}$ forms a complete residue system mod n.

Proof of Lemma 1: Since $\gcd(a, n) = 1$, we know that there must exist some c such that $ac \equiv 1 \pmod{n}$.

Now suppose $\{a_1, ..., a_n\}$ is a complete residue system mod n. Then for any integer d, there is a unique k such that $c(d - b) \equiv a_k \pmod{n}$.

Therefore: $(d - b) \equiv ac(d - b) \equiv aa_k \pmod{n}$ so $d \equiv aa_k + b \pmod{n}$. So each integer is congruent with at least one element in set.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) I

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} = 1$.

Lemma 1: Suppose $\gcd(a, n) = 1$, and $\{a_1, \ldots, a_n\}$ is a complete residue system mod n. Then for all b, $\{aa_1 + b, \ldots, aa_n + b\}$ forms a complete residue system mod n.

Proof of Lemma 1: Since $\gcd(a, n) = 1$, we know that there must exist some c such that $ac \equiv 1 \pmod{n}$.

Now suppose $\{a_1, \ldots, a_n\}$ is a complete residue system mod n. Then for any integer d, there is a unique k such that $c(d - b) \equiv a_k \pmod{n}$.

Therefore: $(d - b) \equiv ac(d - b) \equiv aa_k \pmod{n}$ so $d \equiv aa_k + b \pmod{n}$. So each integer is congruent with at least one element in set.

Now suppose $d \equiv aa_j + b \pmod{n}$ and $d \equiv aa_k + b \pmod{n}$. Then $c(d - b) = aca_j = a_j = aca_k = a_k \pmod{n}$. So each integer is congruent with **exactly** one element in set. So set is a CRS. $\square$

Lemma 2: Suppose $\gcd(a, n) = 1$, and $\{a_1, ..., a_{\phi(n)}\}$ is a reduced residue system mod n. Then $\{aa_1, ..., aa_{\phi(n)}\}$ is also a reduced resude system mod n.

Proof of Lemma 2: Each of $\{aa_1, ..., aa_{\phi(n)}\}$ must be a distinct element in a complete residue system mod n by Lemma 1. Since a reduced residue system has $\phi(n)$ elements, it suffices to show that each of $\{aa_1, ..., aa_{\phi(n)}\}$ is relatively prime to n. But this follows immediately from the fact that both a and a_k are relatively prime to n for all k. $\square$

Euler's Theorem (a.k.a. Euler-Fermat Theorem) III

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof:

Euler's Theorem (a.k.a. Euler-Fermat Theorem) III

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof: Let $\{a_1, ..., a_{\phi(n)}\}$ be a reduced residue system mod n. Then $\{aa_1, ..., aa_{\phi(n)}\}$ must also be a reduced residue system.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) III

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof: Let $\{a_1, \ldots, a_{\phi(n)}\}$ be a reduced residue system mod n. Then $\{aa_1, \ldots, aa_{\phi(n)}\}$ must also be a reduced residue system.

Multiply all the elements of the sets together. They have to be the same.

Euler's Theorem (a.k.a. Euler-Fermat Theorem) III

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof: Let $\{a_1, ..., a_{\phi(n)}\}$ be a reduced residue system mod n. Then $\{aa_1, ..., aa_{\phi(n)}\}$ must also be a reduced residue system.

Multiply all the elements of the sets together. They have to be the same.

$$(aa_1)(aa_2)(aa_3)...(aa_{\phi(n)}) \equiv a_1 a_2 ... a_{\phi(n)} \pmod{n} \ .$$

Euler's Theorem (a.k.a. Euler-Fermat Theorem) III

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof: Let $\{a_1, ..., a_{\phi(n)}\}$ be a reduced residue system mod n. Then $\{aa_1, ..., aa_{\phi(n)}\}$ must also be a reduced residue system.

Multiply all the elements of the sets together. They have to be the same.

$$(aa_1)(aa_2)(aa_3)...(aa_{\phi(n)}) \equiv a_1 a_2 ... a_{\phi(n)} \pmod{n} \quad .$$

Since each a_k is relatively prime to n: we can cancel it on both sides (by existence of multiplicative inverse).

Euler's Theorem (a.k.a. Euler-Fermat Theorem) III

Theorem: Suppose $\gcd(a, n) = 1$. Then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof: Let $\{a_1, ..., a_{\phi(n)}\}$ be a reduced residue system mod n. Then $\{aa_1, ..., aa_{\phi(n)}\}$ must also be a reduced residue system.

Multiply all the elements of the sets together. They have to be the same.

$$(aa_1)(aa_2)(aa_3)...(aa_{\phi(n)}) \equiv a_1a_2...a_{\phi(n)} \pmod{n} .$$

Since each a_k is relatively prime to n: we can cancel it on both sides (by existence of multiplicative inverse).

So:

$$a^{\phi(n)} \equiv 1 \pmod{n} .$$

$\square$

Fermat's Little Theorem

Fermat's little theorem follows immediately from Euler's theorem.

Theorem: Suppose p is prime. Then $a^p \equiv a \pmod{p}$. Furthermore, if $p \nmid a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Fermat's Little Theorem

Fermat's little theorem follows immediately from Euler's theorem.

Theorem: Suppose p is prime. Then $a^p \equiv a \pmod{p}$. Furthermore, if $p \not| a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Proof: Suppose $p|a$. Then obviously $a^p \equiv 0 \equiv a \pmod{p}$.

Fermat's Little Theorem

Fermat's little theorem follows immediately from Euler's theorem.

Theorem: Suppose p is prime. Then $a^p \equiv a \pmod{p}$. Furthermore, if $p \not| a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Proof: Suppose $p|a$. Then obviously $a^p \equiv 0 \equiv a \pmod{p}$.

On the other hand, suppose $p \not| a$. How many nonnegative numbers smaller than p are relatively prime to it?

Fermat's Little Theorem

Fermat's little theorem follows immediately from Euler's theorem.

Theorem: Suppose p is prime. Then $a^p \equiv a \pmod{p}$. Furthermore, if $p \not| a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Proof: Suppose $p|a$. Then obviously $a^p \equiv 0 \equiv a \pmod{p}$.

On the other hand, suppose $p \not| a$. How many nonnegative numbers smaller than p are relatively prime to it? $p - 1$ (all except 0).

Fermat's Little Theorem

Fermat's little theorem follows immediately from Euler's theorem.

Theorem: Suppose p is prime. Then $a^p \equiv a \pmod{p}$. Furthermore, if $p \not| a$, then $a^{p-1} \equiv 1 \pmod{p}$.

Proof: Suppose $p|a$. Then obviously $a^p \equiv 0 \equiv a \pmod{p}$.

On the other hand, suppose $p \not| a$. How many nonnegative numbers smaller than p are relatively prime to it? $p - 1$ (all except 0). So by Euler's theorem: $a^{p-1} = a^{\phi(p)} = 1$. $\square$

Gig(ish): A Combinatorial Look at Fermat's Little Theorem

Questions?