Lecture 7. Outline.

1. Modular Arithmetic. Clock Math!!!
2. Inverses for Modular Arithmetic: Greatest Common Divisor. Division!!!
3. Euclid’s GCD Algorithm. A little tricky here!

Years and years...

80 years from now? 20 leap years. 366 × 20 days
60 regular years. 365 × 60 days
Today is day 2.
It is day 2 × 366 × 20 + 365 × 60. Equivalent to?
Hmm.
What is remainder of 366 when dividing by 7? 52 × 7 + 2.
What is remainder of 365 when dividing by 7? 1
Today is day 2.
Get Day: 2 × 2 × 20 + 1 × 60 = 102
Remainder when dividing by 7? 14 × 7 + 4.
Or February 7, 2096 is Thursday!
Further Simplify Calculation:
20 has remainder 6 when divided by 7.
60 has remainder 4 when divided by 7.
Get Day: 2 × 2 × 6 + 1 × 4 = 18
Or Day 4. February 9, 2095 is Thursday.
"Reduce" at any time in calculation!

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 addition/subtraction of 12.
What time is it in 100 hours? 101:00! or 5:00.
101 = 12 × 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, . . . , 11)
(Against remainder, except for 12 and 0 are equivalent.)

Modular Arithmetic: refresher.

x is congruent to y modulo m or "x = y (mod m)"
If and only if (x − y) is divisible by m.
...or x and y have the same remainder w.r.t. m.
...or x = y + km for some integer k.
Mod 7 equivalence classes:
{...−7, 0, 7, 14,...} {...−6, 1, 8, 15,...} ...
Useful Fact: Addition, subtraction, multiplication can be done with any equivalent x and y.
or * a = c (mod m) and b = d (mod m)
⇒ a + b = c + d (mod m) and a − b = c − d (mod m)"
Proof: If a = c (mod m), then a = c + km for some integer k.
If b = d (mod m), then b = d + jm for some integer j.
Therefore, a + b = c + d + (k + j)m and since k + j is integer.
⇒ a + b = c + d (mod m).
Can calculate with representative in {0, . . . , m − 1}.

Day of the week.

Today is Monday.
What day is it a year from now? on February 9, 2016?
Number days.
0 for Sunday, 1 for Monday, . . . , 6 for Saturday.
Today: day 2.
5 days from now. day 7 or day 0 or Sunday.
25 days from now. day 27 or day 6.
two days are equivalent up to addition/subtraction of multiple of 7.
11 days from now is day 6 which is Saturday!
What day is it a year from now?
This year is not a leap year. So 365 days from now.
Day 2+365 or day 367.
Smallest representation:
subtract 7 until smaller than 7.
divide and get remainder.
367/7 leaves quotient of 52 and remainder 3.
or February 7, 2018 is a Wednesday.

Notation

x (mod m) mod (x, m)
- remainder of x divided by m in {0, . . . , m − 1}.
mod (x, m) = x − [x m]
[x m] is quotient.
mod (29, 12) = 29 − ((29 12)) × 12 = 29 − (2) × 12 = X = 5
Work in this system.
a = b (mod m).
Says two integers a and b are equivalent modulo m.
Modulus is m
6 = 3 + 3 = 3 + 10 (mod 7).
6 = 3 + 3 = 3 + 10 (mod 7).
Generally, not 6 (mod 7) = 13 (mod 7).
But ok, if you really want.
Proof Review 2: Bijections.

If \( \gcd(x, m) = 1 \).
Then the function \( f(a) = xa \mod m \) is a bijection.
One to one: there is a unique inverse.
Onto: the sizes of the domain and co-domain are the same.
\[ x = 3, m = 4. \]
\( f(1) = 3(1) \equiv 3 \mod 4, f(2) = 6 \equiv 2 \mod 4, f(3) = 1 \mod 3. \)
Oh yeah. \( f(0) = 0 \).
Bijection = unique inverse and same size.
Proved unique inverse.
\[ x = 2, m = 4. \]
\( f(1) = 2, f(2) = 0, f(3) = 2 \).
Oh yeah. \( f(0) = 0 \).
Not a bijection.

Finding inverses.

How to find the inverse?
How to find if \( x \) has an inverse modulo \( m \)?
Find \( \gcd(x, m) \).
Greater than 1? No multiplicative inverse.
Equal to 1? Multiplicative inverse.
Algorithm: Try all numbers up to \( x \) to see if it divides both \( x \) and \( m \).
Very slow.

Proof review. Consequence.

Thm: If \( \gcd(x, m) = 1 \), then \( x \) has a multiplicative inverse modulo \( m \).
Proof Sketch: The set \( S = \{0x, 1x, \ldots, (m-1)x\} \) contains \( y = 1 \mod m \) if all distinct modulo \( m \).

Inverses

Next up.
Euclid’s Algorithm.
Runtime.
Euclid’s Extended Algorithm.
Divisibility...

**Notation:** \(d|x\) means "\(d\) divides \(x\)" or \(x = kd\) for some integer \(k\).

**Fact:** If \(d|x\) and \(d|y\) then \(d|(x+y)\) and \(d|(x-y)\).

**Proof:** \(d|x\) and \(d|y\) or \(x = id\) and \(y = kd\)

\[x - y = kd - id = (k-i)d \implies d|(x-y)\]

---

Excursion: Value and Size.

Before discussing running time of gcd procedure...

What is the value of 1,000,000?

- one million or 1,000,000!

What is the "size" of 1,000,000?

- Number of digits: 7.
- Number of bits: 21.

For a number \(x\), what is its size in bits?

\[n = \ell(x) \approx \log_2 x\]

---

Euclid's algorithm.

**GCD Mod Corollary:** \(\gcd(x,y) = \gcd(y, \text{mod}(x,y))\).

Hey, what's \(\gcd(7,0)\)? 7 since 7 divides 7 and 7 divides 0.

What's \(\gcd(0,7)\)?

**Proof:** Use Strong Induction.

**Base Case:** \(y = 0\), "\(x\) divides \(y\) and \(x\)"

\[\iff \ gcd(x,y) \] is common divisor and clearly largest."

**Induction Step:** \(\text{mod}(x,y) < y \leq x\) when \(x \geq y\)

call in line (***)) meets conditions plus arguments "smaller" and by strong induction hypothesis

computes \(\gcd(y, \ \text{mod}(x,y))\)

which is \(\gcd(x,y)\) by GCD Mod Corollary.

---

More divisibility

**Notation:** \(d|x\) means "\(d\) divides \(x\)" or \(x = kd\) for some integer \(k\).

**Lemma 1:** If \(d|x\) and \(d|y\) then \(d|y\) \(\implies\) \(d|\text{mod}(x,y)\).

**Proof:** \(\text{mod}(x,y) = x - \lfloor x/y \rfloor \cdot y\)

**Lemma 2:** If \(d|y\) and \(d|\text{mod}(x,y)\) then \(d|y\) \(\implies\) \(d|x\).

**Proof:** Similar. Try this at home.

**GCD Mod Corollary:** \(\gcd(x,y) = \gcd(y, \ \text{mod}(x,y))\).

---

Euclid procedure is fast.

**Theorem:** \((\text{euclid } x y)\) uses \(2n\) "divisions" where \(n = \ell(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \([2, y/2]\)?

Check 2, check 3, check 4, check 5 . . . , check \(y/2\).

If \(y = x\) roughly \(y\) uses \(n\) bits...

\(2^{n-1}\) divisions! Exponential dependence on size!

101 bit number, \(2^{101} = 10^{30} = \text{"million, trillion, trillion" divisions!}\)

\(2n\) is much faster! .. roughly 200 divisions.
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check \( y/2 \).
"(gcd x y)" at work.

\[
euclid(700, 568) \\
euclid(568, 132) \\
euclid(132, 40) \\
euclid(40, 12) \\
euclid(12, 4) \\
euclid(4, 0)
\]

Notice: The first argument decreases rapidly.
At least a factor of 2 in two recursive calls.
(The second is less than the first.)

Finding an inverse?

We showed how to efficiently tell if there is an inverse.
Extend euclid to find inverse.

Euclid's GCD algorithm.

\[
\begin{align*}
&\text{(define (euclid x y)} \\
&(\text{if (= y 0)} \\
&\quad x \\
&\quad (euclid y (mod x y))))
\end{align*}
\]

Computes the gcd\((x, y)\) in \(O(n)\) divisions.
For \(x \) and \( m \), if gcd\((x, m) = 1\) then \(x\) has an inverse modulo \(m\).

Proof.

\[
\text{(define (euclid x y)} \\
&(\text{if (= y 0)} \\
&\quad x \\
&\quad (euclid y (mod x y))))
\]

Theorem: \(\text{euclid x y}\) uses \(O(n)\) "divisions" where \(n = b(x)\).
Proof:

Fact:
First arg decreases by at least factor of two in two recursive calls.
Proof of Fact: By induction.
Base case: \(y = 0\).
Inductive step: \(\text{euclid}\) calls remainder function.
\(\text{mod} (x, y)\leq x/2\). 
Diagonal line: \(\text{euclid}\) divides the second argument in next recursive call,
and becomes the first argument in the next one.

\[
\text{mod} (x, y) = x - y \lfloor x/y \rfloor = x - y \leq x - x/2 = x/2 
\]

Finding an inverse?

We showed how to efficiently tell if there is an inverse.
Extend euclid to find inverse.

Multiplicative Inverse.

GCD algorithm used to tell if there is a multiplicative inverse.
How do we find a multiplicative inverse?
Extended GCD

**Euclid’s Extended GCD Theorem:** For any $x,y$ there are integers $a,b$ such that
$$ax + by = d \quad \text{where } d = \gcd(x,y).$$

“Make $d$ out of sum of multiples of $x$ and $y$.”

What is multiplicative inverse of $x$ modulo $m$?

By extended GCD theorem, when $\gcd(x,m) = 1$.
$$ax \equiv 1 \!\pmod{m}.$$  

So a multiplicative inverse of $x$ (mod $m$)!

Example: For $x = 12$ and $y = 35 \Rightarrow \gcd(12,35) = 1$.
$(3)12 + (−1)35 = 1$.

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

---

**Make $d$ out of $x$ and $y$.**

$$\gcd(35,12)$$  
$$\gcd(12, 11) \quad \Rightarrow \quad \gcd(12, 35\%12)$$  
$$\gcd(11, 1) \quad \Rightarrow \quad \gcd(11, 12\%11)$$  
$$\gcd(1,0) \quad \Rightarrow \quad 1.$$  

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

How does $\gcd$ get 1 from 12 and 11?
$$12 - \left\lfloor \frac{12}{11} \right\rfloor \cdot 11 = 12 - (1)11 = 1.$$  

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?
Get 1 from 12 and 11.
$$1 = 12 - (1)11 = 12 - (1)(35 - (2)12) = (3)12 + (-1)35.$$  

Get 11 from 35 and 12 and plugin,... Simplify: $a = 3$ and $b = -1$.

---

**Extended GCD Algorithm.**

```plaintext
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
  else
    (d, a, b) := ext-gcd(y, mod(x,y))
    return (d, b, a - floor(x/y) * b)

Theorem: Returns $(d,a,b)$, where $d = \gcd(a,b)$ and $d = ax + by$.
```

**Correctness.**

**Proof:** Strong Induction.\(^1\)

**Base:** $\text{ext-gcd}(x,0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

**Induction Step:** $\text{Returns } (d,A,B)$ with $d = Ax + By$

Ind hyp: $\text{ext-gcd}(y, \mod(x,y))$ returns $(d, a, b)$ with $d = ay + bx \mod(x,y)$

$\text{ext-gcd}(x,y)$ calls $\text{ext-gcd}(y, \mod(x,y))$ so

\[
d = ay + bx \mod(x,y)
= ay + bx - \left\lfloor \frac{ax}{y} \right\rfloor y
= bx + (a - \left\lfloor \frac{x}{y} \right\rfloor) y
\]

And $\text{ext-gcd}$ returns $(d,b,(a - \left\lfloor \frac{x}{y} \right\rfloor) b)$ so theorem holds! \(\square\)

---

**Proof:** Strong Induction.\(^1\)

**Base:** $\text{ext-gcd}(x,0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

**Induction Step:** $\text{Returns } (d,A,B)$ with $d = Ax + By$

Ind hyp: $\text{ext-gcd}(y, \mod(x,y))$ returns $(d, a, b)$ with $d = ay + bx \mod(x,y)$

$\text{ext-gcd}(x,y)$ calls $\text{ext-gcd}(y, \mod(x,y))$ so

\[
d = ay + bx \mod(x,y)
= ay + bx - \left\lfloor \frac{ax}{y} \right\rfloor y
= bx + (a - \left\lfloor \frac{x}{y} \right\rfloor) y
\]

And $\text{ext-gcd}$ returns $(d,b,(a - \left\lfloor \frac{x}{y} \right\rfloor) b)$ so theorem holds! \(\square\)

---

**Review Proof:** step.

```plaintext
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
  else
    (d, a, b) := ext-gcd(y, mod(x,y))
    return (d, b, a - floor(x/y) * b)

Claim: Returns $(d,a,b)$: $d = \gcd(a,b)$ and $d = ax + by$.
Example: $a - \lfloor x/y \rfloor \cdot d = \gcd(a,b)$ and $d = ax + by$.  
```

\[
\text{ext-gcd}(35,12)
\text{ext-gcd}(12, 11)
\text{ext-gcd}(11, 1)
\text{ext-gcd}(1,0)
\]

return (1,1,0); $1 = (1)1 + (0)0$
return (1,0,1); $1 = (0)11 + (1)1$
return (1,1,-1); $1 = (1)12 + (-1)11$
return (1,-1, 3); $1 = (-1)35 + (3)12$

\[29/32\]
Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!
Very different from elementary school: try 1, try 2, try 3...

$2^{\sqrt{n}}$

Inverse of 500,000,357 modulo 1,000,000,000,000,000,000,000,000,000?
≤ 80 divisions.
versus 1,000,000

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(1000000000000000000000000000000000000000000)5$ divisions.

Internet Security: Next Week!