Lecture Outline

Continue with modular arithmetic.

Euclid’s Algorithm for computing GCD.
Runtime.
Euclid’s Extended Algorithm.
Fundamental Theorem of Arithmetic.
Chinese Remainder Theorem.
Recap: Review of theorem from last time.

**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 \( y \equiv 1 \mod m \) if all distinct modulo \( m \).

... For \( x = 4 \) and \( m = 6 \). All products of 4...
\[
S = \{0(4), 1(4), 2(4), 3(4), 4(4), 5(4)\} = \{0, 4, 8, 12, 16, 20\}
\]
reducing \( \mod 6 \)
\[
S = \{0, 4, 2, 0, 4, 2\}
\]
Not distinct. Common factor 2.

For \( x = 5 \) and \( m = 6 \).
\[
S = \{0(5), 1(5), 2(5), 3(5), 4(5), 5(5)\} = \{0, 5, 4, 3, 2, 1\}
\]
All distinct, contains 1! 5 is multiplicative inverse of 5 \( \mod 6 \).

\( 5x = 3 \mod 6 \) What is \( x \)? Multiply both sides by 5.
\[
x = 15 = 3 \mod 6
\]
\( 4x = 3 \mod 6 \) No solutions. Can’t get an odd.
\( 4x = 2 \mod 6 \) Two solutions! \( x = 2, 5 \mod 6 \)

Very different for elements with inverses.
$x$ has an inverse modulo $m$ if $\gcd(x, m) = 1$

Next:
  - Compute $\gcd$!
  - Compute Inverse modulo $m$. 

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 $x = \ell d$ and $y = kd$

$$\Rightarrow x - y = kd - \ell d = (k - \ell)d \Rightarrow d|\,(x - y)$$
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 $d|y$ and $d|\text{mod}(x, y)$.

**Proof:**
\[
\begin{align*}
\text{mod}(x, y) &= x - \lfloor x/y \rfloor \cdot y \\
&= x - s \cdot y \quad \text{for integer } s \\
&= kd - s\ell d \quad \text{for integers } k, \ell \\
&= (k - s\ell)d
\end{align*}
\]

Therefore $d|\text{mod}(x, y)$. And $d|y$ since it is in condition. \hfill \square

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

**Proof...:** Similar. Try this at home. \hfill \square.

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

**Proof:** $x$ and $y$ have **same** set of common divisors as $x$ and $\text{mod}(x, y)$ by Lemma.

Same common divisors $\implies$ largest is the same. \hfill \square
Euclid’s algorithm.

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

gcd (x, y)
    if (y = 0) then
        return x
    else
        return gcd(y, mod(x, y)) ***

**Theorem:** Euclid’s algorithm computes the greatest common divisor of \( x \) and \( y \) if \( x \geq y \).

**Proof:** Use Strong Induction.

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

\[ \implies \text{“} x \text{ is common divisor and clearly largest.”} \]

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

call in line (*** ) meets conditions plus arguments “smaller”
and by strong induction hypothesis
computes \( \gcd(y, \mod(x, y)) \)
which is \( \gcd(x, y) \) by GCD Mod Corollary.
Before discussing running time of gcd procedure...

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?

$$n = b(x) \approx \log_2 x$$
GCD procedure is fast.

**Theorem:** GCD uses $2n$ “divisions” where $n$ is the number of bits.

Is this good? Better than trying all numbers in \{2, \ldots \ y/2\}?

Check 2, check 3, check 4, check 5 \ldots, check $y/2$.

$2^{n-1}$ divisions! Exponential dependence on size!

101 bit number. $2^{100} \approx 10^{30} = \text{“million, trillion, trillion” divisions}!$

$2n$ is much faster! \ldots roughly 200 divisions.
Algorithms at work.

“gcd(x, y)” at work.

gcd(700, 568)
gcd(568, 132)
gcd(132, 40)
gcd(40, 12)
gcd(12, 4)
gcd(4, 0)

4

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

\[
\text{gcd}(x, y) = \begin{cases} 
  x & \text{if } (y = 0) \\
  \text{gcd}(y, \text{mod}(x, y)) & \text{else}
\end{cases}
\]

**Theorem:** GCD uses \(O(n)\) ”divisions” where \(n\) is the number of bits.

**Proof:**

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

**Proof of Fact:** Recall that first argument decreases every call. After \(2 \log_2 x\) recursive calls, argument \(x\) is \(1\) bit number. One more recursive call to finish. "mod\((x, y) \leq x/2.\)"

**Case 1:** \(y \leq x/2\), first argument is \(y\)

**Case 2:** \(y > x/2\), \(\text{mod}(x, y) = x - y\lfloor x/y \rfloor = x - y \leq x - x/2 = x/2\)

\[
\text{mod} (x, y) = x - y \left\lfloor \frac{x}{y} \right\rfloor = x - y \leq x - x/2 = x/2
\]
Multiplicative Inverse.

GCD algorithm used to tell if there is a multiplicative inverse.

How do we find a multiplicative inverse?
Euclid’s Extended GCD Theorem: For any \( x, y \) there are integers \( a, b \) such that
\[
ax + by = \gcd(x, y) = d \quad \text{where } d = \gcd(x, y).
\]

“Make \( d \) out of sum of multiples of \( x \) and \( y \).”

What is multiplicative inverse of \( x \) modulo \( m \)?

By extended GCD theorem, when \( \gcd(x, m) = 1 \).
\[
ax + bm = 1
\]
\[
ax \equiv 1 - bm \equiv 1 \pmod{m}.
\]

So \( a \) is multiplicative inverse of \( x \) if \( \gcd(a, x) = 1 \)!!

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

\[
(3)12 + (-1)35 = 1.
\]
\[
a = 3 \text{ and } b = -1.
\]
The multiplicative inverse of 12 (mod 35) is 3.
Make $d$ out of $x$ and $y$..?

```
gcd(35, 12)
gcd(12, 11) ;; gcd(12, 35\%12)
gcd(11, 1) ;; gcd(11, 12\%11)
gcd(1, 0)
  1
```

How did gcd get 11 from 35 and 12?

$$35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11$$

How does gcd get 1 from 12 and 11?

$$12 - \left\lfloor \frac{12}{11} \right\rfloor 11 = 12 - (1)11 = 1$$

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.

\[ \text{ext-gcd}(x, y) \]
\[
\text{if } y = 0 \text{ then return } (x, 1, 0) \\
\text{else} \\
(d, a, b) := \text{ext-gcd}(y, \text{mod}(x,y)) \\
\text{return } (d, b, a - \text{floor}(x/y) \times b)
\]

Claim: Returns \((d, a, b)\): \(d = \gcd(x, y)\) and \(d = ax + by\).

Example: \(a - \lfloor x/y \rfloor \cdot b = 1 - \lfloor 11/1 \rfloor \cdot 0 = 10 - \lfloor 12/11 \rfloor \cdot 1 = -11 - \lfloor 35/12 \rfloor \cdot (-1) = 3\)

\[
\text{ext-gcd}(35, 12) \\
\text{ext-gcd}(12, 11) \\
\text{ext-gcd}(11, 1) \\
\text{ext-gcd}(1, 0) \\
\text{return } (1, 1, 0) ;; 1 = (1)1 + (0)0 \\
\text{return } (1, 0, 1) ;; 1 = (0)11 + (1)1 \\
\text{return } (1, 1, -1) ;; 1 = (1)12 + (-1)11 \\
\text{return } (1, -1, 3) ;; 1 = (-1)35 + (3)12\]
Extended GCD Algorithm.

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 \( d = \gcd(x, y) \) 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:** Returns \((d, A, B)\) with \(d = Ax + By\)

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

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

\[
d = d^* = ay + b \cdot (\text{mod}(x, y))
\]

\[
= ay + b \cdot (x - \left\lfloor \frac{x}{y} \right\rfloor y)
\]

\[
= bx + (a - \left\lfloor \frac{x}{y} \right\rfloor \cdot b)y
\]

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

---

\(^1\)Assume \(d\) is \(gcd(x, y)\) by previous proof.

\[
\text{ext-gcd}(x, y) \\
\text{if } y = 0 \text{ then return } (x, 1, 0) \\
\text{else} \\
\quad (d, a, b) := \text{ext-gcd}(y, \text{mod}(x, y)) \\
\quad \text{return } (d, b, a - \text{floor}(x/y) \times b)
\]

Recursively: \[d = ay + b(x - \lfloor \frac{x}{y} \rfloor \cdot y) \implies d = bx + (a - \lfloor \frac{x}{y} \rfloor b)y\]

Returns \((d, b, (a - \lfloor \frac{x}{y} \rfloor \cdot b)).\)
Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Proof: $n$ is either prime (base cases)
  or $n = a \times b$ and $a$ and $b$ can be written as product of primes.

Thm: The prime factorization of $n$ is unique up to reordering.

Fundamental Theorem of Arithmetic: Every natural number can be written as a unique (up to reordering) product of primes.
Claim: For $x, y, z \in \mathbb{Z}^+$ with $gcd(x, y) = 1$ and $x|yz$ then $x|z$.

Idea: $x$ doesn’t share common factors with $y$ so it must divide $z$.

Euclid: $1 = ax + by$.

Observe: $x|axz$ and $x|byz$ (since $x|yz$), and $x$ divides the sum.

$\implies x|axz + byz$

And $axz + byz = z$, thus $x|z$. 

$\square$
Fundamental Theorem of Arithmetic: Uniqueness

Thm: The prime factorization of $n$ is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some $j$.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.

If $gcd(p, q_l) = d$, then $d$ is a common factor.

If both prime, both only have 1 and themselves as factors. Thus, $p = q_l = d$.

End proof of fact.

Proof by induction.

Base case: If $l = 1$, $p_1 \cdots p_k = q_1$.

But if $q_1$ is prime, only prime factor is $q_1$ and $p_1 = q_1$ and $l = k = 1$.

Induction step: From Fact: $p_1 = q_j$ for some $j$.

$n/p_1 = p_2 \cdots p_k$ and $n/q_j = \prod_{i \neq j} q_i$.

These two expressions are the same up to reordering by induction. And $p_1$ is matched to $q_j$. 

$\square$
**Simple Chinese Remainder Theorem.**

**CRT Thm:** For $m, n$ s.t. $\gcd(m, n)=1$, there exists a unique solution $x \pmod{mn}$ s.t.

$x = a \pmod{m}$ and $x = b \pmod{n}$

**Proof (solution exists):**
Consider $u = n(n^{-1} \pmod{m})$.

$u = 0 \pmod{n}$ \quad $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.

$v = 1 \pmod{n}$ \quad $v = 0 \pmod{m}$

Let $x = au + bv$.

$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$

$x = b \pmod{n}$ since $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

This shows there is a solution. \qed
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$

$\gcd(m, n) = 1 \implies$ no common primes in factorization $m$ and $n$

$\implies mn|(x - y)$

$\implies x - y \geq mn \implies x, y \not\in \{0, \ldots, mn - 1\}$.

Thus, only one solution modulo $mn$.  