CSE 311: Foundations of Computing

Lecture 11: Application, Primes, GCD

Last class: Modular Arithmetic: Properties

If $\boldsymbol{a} \equiv \boldsymbol{b} \pmod{m}$ and $\boldsymbol{b} \equiv \boldsymbol{c} \pmod{m}$ then $\boldsymbol{a} \equiv \boldsymbol{c} \pmod{m}$

If $\boldsymbol{a} \equiv \boldsymbol{b} \pmod{m}$ and $\boldsymbol{c} \equiv \boldsymbol{d} \pmod{m}$ then
$\boldsymbol{a} + \boldsymbol{c} \equiv \boldsymbol{b} + \boldsymbol{d} \pmod{m}$ and
$\boldsymbol{ac} \equiv \boldsymbol{bd} \pmod{m}$

Corollary: If $\boldsymbol{a} \equiv \boldsymbol{b} \pmod{m}$ then
$\boldsymbol{a} + \boldsymbol{c} \equiv \boldsymbol{b} + \boldsymbol{c} \pmod{m}$ and
$\boldsymbol{ac} \equiv \boldsymbol{bc} \pmod{m}$

These allow us to solve problems in modular arithmetic, e.g.

- **add/subtract numbers from both sides of equations**
- **multiply numbers on both sides of equations.**
- **use chains of equivalences**

Basic Applications of mod

- Two's Complement
- Hashing
- Pseudo random number generation

n-bit Unsigned Integer Representation

- Represent integer x as sum of powers of 2:

99 = 64 + 32 + 2 + 1 = $2^6 + 2^5 + 2^1 + 2^0$

18 = 16 + 2 = $2^4 + 2^1$

If $b_{n-1}2^{n-1} + \cdots + b_1 2 + b_0$ with each $b_i \in \{0,1\}$
then binary representation is $b_{n-1}...b_2\, b_1\, b_0$

- For n = 8:

99: 0110 0011

18: 0001 0010

Easy to implement arithmetic **mod 2^n**
... just throw away bits n+1 and up

$2^n \mid 2^{n+k}$ so $b_{n+k}2^{n+k} \equiv 0 \pmod{2^n}$
for $k \geq 0$

n-bit Unsigned Integer Representation

- Largest representable number is $2^n - 1$

$$2^n = 100...000 \qquad \text{(n+1 bits)}$$

$$2^n - 1 = 11...111 \qquad \text{(n bits)}$$

THE WALL STREET JOURNAL.

32 bits
1 = $0.0001
$429,496.7295 max

Berkshire Hathaway Inc. (BRK-A)
NYSE - Nasdaq Real Time Price. Currency in USD

436,401.00 +679.50 (+0.16%)

At close: 4:00PM EDT

Sign-Magnitude Integer Representation

n-bit signed integers

Suppose that $-2^{n-1} < x < 2^{n-1}$

First bit as the sign, $n-1$ bits for the value

99 = 64 + 32 + 2 + 1

18 = 16 + 2

18 = 0001 0010

For n = 8:

99: 0110 0011

-18: 1001 0010

-18 = 1001 0010

0000 0000

1000 0000 "−0"

Problem: this has both +0 and -0 (annoying)

Two's Complement Representation

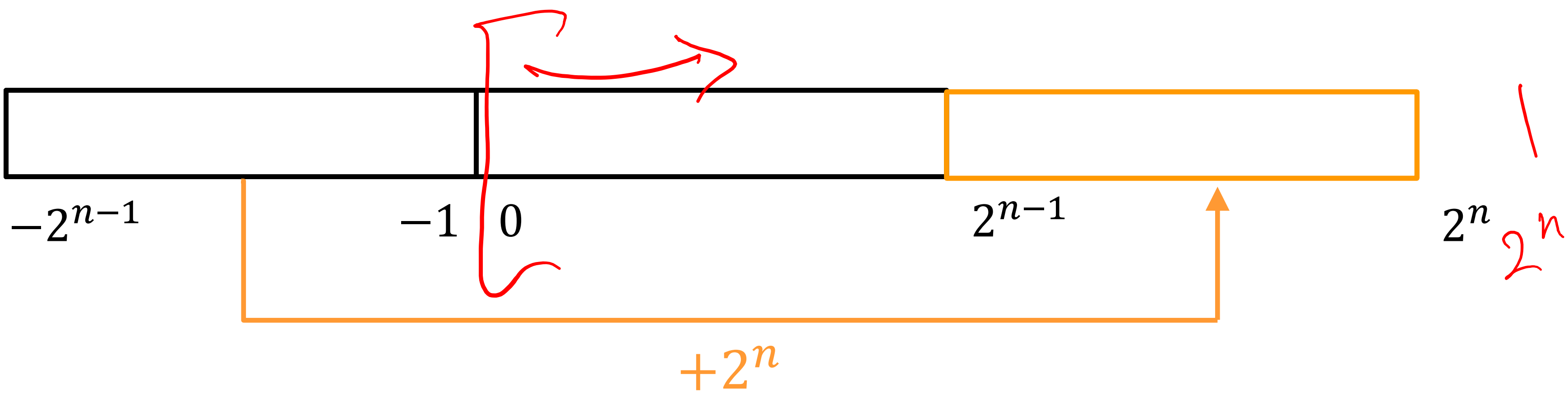
Suppose that $0 \leq x < 2^{n-1}$

x is represented by the binary representation of x

Suppose that $-2^{n-1} \leq x < 0$

x is represented by the binary representation of $x + 2^n$

result is in the range $2^{n-1} \leq x < 2^n$



0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Two's Complement Representation

Suppose that $0 \leq x < 2^{n-1}$

x is represented by the binary representation of x

Suppose that $-2^{n-1} \leq x < 0$

x is represented by the binary representation of $x + 2^n$

result is in the range $2^{n-1} \leq x < 2^n$

0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

99 = 64 + 32 + 2 + 1

18 = 16 + 2

For n = 8:

99: 0110 0011

-18: 1110 1110 (-18 + 256 = 238)

Two's Complement Representation

Suppose that $0 \le x < 2^{n-1}$

x is represented by the binary representation of x

Suppose that $-2^{n-1} \le x < 0$

x is represented by the binary representation of $x + 2^n$

result is in the range $2^{n-1} \le x < 2^n$

0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Key property: First bit is still the sign bit!

Key property: Twos complement representation of any number $\boldsymbol{y}$ is equivalent to $\boldsymbol{y} \pmod{\mathbf{2}^{\boldsymbol{n}}}$ so arithmetic works $\pmod{\mathbf{2}^{\boldsymbol{n}}}$

$$y + 2^n \equiv y \pmod{2^n}$$

0111 1111 0110

$7 + -1 \quad 6$

vs

$7 + 15 \quad 22 \equiv 6 \pmod{16}$

0111 1111 10110

Two's Complement Representation

- For $0 < x \leq 2^{n-1}$, $-x$ is represented by the binary representation of $-x + 2^n$
  - How do we calculate –x from x?
  - E.g., what happens for "`return -x;`" in Java?

$$-x + 2^n = (2^n - 1) - x + 1$$

- To compute this, flip the bits of x then add 1!
  - All 1's string is $2^n - 1$, so

    Flip the bits of x means replace x by $2^n - 1 - x$

    Then add 1 to get $-x + 2^n$

More Number Theory

Primes and GCD

Primality

An integer *p* greater than 1 is called *prime* if the only positive factors of *p* are 1 and *p*.

$$p > 1 \land \forall x\,((x > 0) \land (x \mid p)) \rightarrow ((x = 1) \lor (x = p)))$$

A positive integer that is greater than 1 and is not prime is called *composite*.

$$p > 1 \land \exists x\,((x > 0) \land (x \mid p) \land (x \neq 1) \land (x \neq p))$$

Fundamental Theorem of Arithmetic

Every positive integer greater than 1 has a "unique" prime factorization

48 = 2 • 2 • 2 • 2 • 3 = 3 · 2 · 2 · 2 · 2

591 = 3 • 197

45,523 = 45,523

321,950 = 2 • 5 • 5 • 47 • 137

1,234,567,890 = 2 • 3 • 3 • 5 • 3,607 • 3,803

Algorithmic Problems

- **Multiplication**
  - Given primes $p_1, p_2, \ldots, p_k$, calculate their product $p_1 p_2 \ldots p_k$
- **Factoring**
  - Given an integer n, determine the prime factorization of n

Factoring

Factor the following 232 digit number [RSA768]:

1230186684530117755130494958384962720772
8535695953347921973224521517264005072636
5751874520219978646938995647494277406384
5925192557326303453731548268507917026122
1429134616704292143116022212404792747377
940806653514195974598569021434l3

1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413

=

33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489

×

36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

Famous Algorithmic Problems

- Factoring
  - Given an integer n, determine the prime factorization of n
- Primality Testing
  - Given an integer n, determine if n is prime

- Factoring is hard
  - (on a classical computer)
- Primality Testing is easy

Greatest Common Divisor

GCD(a, b):

Largest integer d such that $d \mid a$ and $d \mid b$

- GCD(100, 125) = 25
- GCD(17, 49) = 1
- GCD(11, 66) = 11
- GCD(13, 0) = 13
- GCD(180, 252) = $2^2 \cdot 3^2 = 36$

$(2^2 \cdot 3^2 \cdot 5,\ 2^2 \cdot 3^2 \cdot 7)$

$d = \text{GCD}(a,b)$ iff $(d \mid a) \wedge (d \mid b) \wedge \forall x\, (((x \mid a) \wedge (x \mid b)) \rightarrow (x \leq d))$

GCD and Factoring

$a = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11 = 46{,}200$

$b = 2 \cdot 3^2 \cdot 5^3 \cdot 7 \cdot 13 = 204{,}750$

$GCD(a, b) = 2^{\min(3,1)} \cdot 3^{\min(1,2)} \cdot 5^{\min(2,3)} \cdot 7^{\min(1,1)} \cdot 11^{\min(1,0)} \cdot 13^{\min(0,1)}$

$2^1 \cdot 3^1 \cdot 5^2 \cdot 7$

Factoring is hard!

Can we compute GCD(a,b) without factoring?

Useful GCD Fact

Let *a* and *b* be positive integers.
We have gcd(*a*,*b*) = gcd(*b*, *a* mod *b*)

Proof:
We will show that the numbers dividing a and b are
the same as those dividing b and $a \bmod b$.
i.e., $d|a$ and $d|b$ iff $d|b$ and $d|(a \bmod b)$.

Hence, their set of common divisors are the same,
which means that their greatest common divisor is the same.

Useful GCD Fact

$\gcd(a,b) = \gcd(b,a)$

Let *a* and *b* be positive integers.
We have gcd(*a*,*b*) = gcd(*b*, *a* mod *b*)

Proof:

By definition of mod, $a = qb + (a \bmod b)$ for some integer $q = a \text{ div } b$.

(⇒) Suppose that $d|b$ and $d|(a \bmod b)$.
Then $b = md$ and $(a \bmod b) = nd$ for some integers m and n.
Therefore $a = qb + (a \bmod b) = qmd + nd = (qm+n)d$.
So $d|a$. Therefore $d|a$ and $d|b$.

$a \bmod b = a - qb$

(⇐) Suppose that $d|a$ and $d|b$.
Then $a = kd$ and $b = jd$ for some integers k and j.
Therefore $(a \bmod b) = a - qb = kd - qjd = (k - qj)d$.
So $d|(a \bmod b)$ also. Therefore $d|b$ and $d|(a \bmod b)$.

Since they have the same common divisors, $\gcd(a,b) = \gcd(b, a \bmod b)$. ∎

Another simple GCD fact

Let a be a positive integer.
We have gcd(a,0) = a.

Euclid's Algorithm

gcd(a, b) = gcd(b, a mod b)

gcd(a, 0) = a

```
int gcd(int a, int b){ /* Assumes: a >= b, b >= 0 */
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}
```

Note: gcd(b, a) = gcd(a, b)

Euclid's Algorithm

Repeatedly use $\gcd(a, b) = \gcd(b, a \bmod b)$ to reduce numbers until you get $\gcd(g, 0) = g$.

gcd(660,126)

Euclid's Algorithm

Repeatedly use $\gcd(a, b) = \gcd(b, a \bmod b)$ to reduce numbers until you get $\gcd(g, 0) = g$.

Equations with recursive calls:

gcd(660,126) = gcd(126, 660 mod 126) = gcd(126, 30)
= gcd(30, 126 mod 30) = gcd(30, 6)
= gcd(6, 30 mod 6) = gcd(6, 0)
= 6

660 = 5 * 126 + 30
126 = 4 * 30 + 6
30 = 5 * 6 + 0

Euclid's Algorithm

Repeatedly use $\gcd(a, b) = \gcd(b, a \bmod b)$ to reduce numbers until you get $\gcd(g, 0) = g$.

Equations with recursive calls:

gcd(660,126) = gcd(126, 660 mod 126) = gcd(126, 30)
= gcd(30, 126 mod 30) = gcd(30, 6)
= gcd(6, 30 mod 6) = gcd(6, 0)
= 6

Tableau form (which is much easier to work with and will be more useful):

660 = 5 * 126 + 30
126 = 4 * 30 + 6
30 = 5 * 6 + 0

Each line computes both quotient and remainder of the shifted numbers

Division $(\mathrm{mod}\ m)$

We already can

- Add, subtract, and, multiply numbers $(\mathrm{mod}\ m)$

What about dividing numbers $(\mathrm{mod}\ m)$?

In ordinary arithmetic, to divide by a we can multiply by $b = a^{-1} = 1/a$, the *multiplicative inverse* of a

- It doesn't always exist
  - if $a = 0$
  - if the domain is integers and $a \neq 1, -1$
- If it does exist then $ab = 1$

Multiplicative inverse $(\mathrm{mod}\ m)$

Let $0 \leq a, b < m$. Then, b is the *multiplicative inverse of* a *(modulo* m*)* iff $ab \equiv 1 \pmod{m}$.

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

mod 7

$3 \cdot 5 \equiv 1 \pmod{7}$

x	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

mod 10

Finding inverses with Euclid I: Bézout's theorem

If *a* and *b* are positive integers, then there exist integers ***s*** and ***t*** such that

$$\gcd(a,b) = \boldsymbol{s}a + \boldsymbol{t}b.$$

note:
s or t
will
be negative

$$\forall a\, \forall b\, ((a > 0 \land b > 0) \rightarrow \exists s\, \exists t\, (\gcd(a,b) = sa + tb))$$

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Step 1 (Compute GCD(a,b) in tableau form):

Example: $a = 35$, $b = 27$

Compute $\gcd(35, 27)$:

a = q * b + r

$$35 = 1 * 27 + 8$$
$$27 = 3 * 8 + 3$$
$$8 = 2 * 3 + 2$$
$$3 = 1 * 2 + 1$$
$$2 = 2 * 1 + 0$$

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Step 2 (Solve the equations for r):

Example: $a = 35$, $b = 27$

a = q * b + r	r = a – q * b
$35 = 1 * 27 + 8$	$8 = 35 - 1 * 27$
$27 = 3 * 8 + 3$	$3 = 27 - 3 * 8$
$8 = 2 * 3 + 2$	$2 = 8 - 2 * 3$
$3 = 1 * 2 + 1$	$1 = 3 - 1 * 2$
$2 = 2 * 1 + 0$	

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35$, $b = 27$

$$8 = 35 - 1 * 27$$
$$3 = 27 - 3 * 8$$
$$2 = 8 - 2 * 3$$
$$1 = 3 - 1 * 2$$

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35, b = 27$ $\qquad 1 = 3 - 1 * 2$

$$8 = 35 - 1 * 27$$

$$3 = 27 - 3 * 8$$

$$2 = 8 - 2 * 3$$

$$1 = 3 - 1 * 2$$

Extended Euclidean algorithm

- **Can use Euclid's Algorithm to find s, t such that**

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35, b = 27$

$$1 = 3 - 1 * 2$$

Plug in for 2

$$= 3 - 1 * (8 - 2 * 3)$$

$$8 = 35 - 1 * 27$$

$$3 = 27 - 3 * 8$$

$$2 = 8 - 2 * 3$$

$$1 = 3 - 1 * 2$$

Extended Euclidean algorithm

- **Can use Euclid's Algorithm to find s, t such that**

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35, b = 27$

$$1 = 3 - 1 * 2$$
$$= 3 - 1 * (8 - 2 * 3)$$
$$= 3 - 8 + 2 * 3$$
$$= (-1) * 8 + 3 * 3$$

Re-arrange into 8's and 3's

$$8 = 35 - 1 * 27$$
$$\boxed{3 = 27 - 3 * 8}$$
$$2 = 8 - 2 * 3$$
$$1 = 3 - 1 * 2$$

Extended Euclidean algorithm

- **Can use Euclid's Algorithm to find s, t such that**

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35, b = 27$

$$1 = 3 - 1 * 2$$
$$= 3 - 1 * (8 - 2 * 3)$$
$$= 3 - 8 + 2 * 3$$
$$= (-1) * 8 + 3 * 3$$

$$8 = 35 - 1 * 27$$
$$3 = 27 - 3 * 8$$
$$2 = 8 - 2 * 3$$
$$1 = 3 - 1 * 2$$

Plug in for 3

$$= (-1) * 8 + 3 * (27 - 3 * 8)$$

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35$, $b = 27$

$8 = 35 - 1 * 27$

$3 = 27 - 3 * 8$

$2 = 8 - 2 * 3$

$1 = 3 - 1 * 2$

$$\begin{aligned} 1 &= 3 - 1 * 2 \\ &= 3 - 1 * (8 - 2 * 3) \\ &= 3 - 8 + 2 * 3 \\ &= (-1) * 8 + 3 * 3 \\ &= (-1) * 8 + 3 * (27 - 3 * 8) \\ &= (-1) * 8 + 3 * 27 + (-9) * 8 \\ &= 3 * 27 + (-10) * 8 \end{aligned}$$

Re-arrange into 27's and 8's

Extended Euclidean algorithm

- Can use Euclid's Algorithm to find s, t such that

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35, b = 27$

$$8 = 35 - 1 * 27$$
$$3 = 27 - 3 * 8$$
$$2 = 8 - 2 * 3$$
$$1 = 3 - 1 * 2$$

$$1 = 3 - 1 * 2$$
$$= 3 - 1 * (8 - 2 * 3)$$
$$= 3 - 8 + 2 * 3$$
$$= (-1) * 8 + 3 * 3$$

Plug in for 8

$$= (-1) * 8 + 3 * (27 - 3 * 8)$$
$$= (-1) * 8 + 3 * 27 + (-9) * 8$$
$$= 3 * 27 + (-10) * 8$$
$$= 3 * 27 + (-10) * (35 - 1 * 27)$$

Extended Euclidean algorithm

- **Can use Euclid's Algorithm to find s, t such that**

$$\gcd(a, b) = sa + tb$$

Step 3 (Backward Substitute Equations):

Example: $a = 35$, $b = 27$

$$8 = 35 - 1 * 27$$
$$3 = 27 - 3 * 8$$
$$2 = 8 - 2 * 3$$
$$1 = 3 - 1 * 2$$

$$
\begin{aligned}
1 &= 3 - 1 * 2 \\
&= 3 - 1 * (8 - 2 * 3) \\
&= 3 - 8 + 2 * 3 \\
&= (-1) * 8 + 3 * 3 \\
&= (-1) * 8 + 3 * (27 - 3 * 8) \\
&= (-1) * 8 + 3 * 27 + (-9) * 8 \\
&= 3 * 27 + (-10) * 8 \\
&= 3 * 27 + (-10) * (35 - 1 * 27) \\
&= 3 * 27 + (-10) * 35 + 10 * 27 \\
&= (-10) * 35 + 13 * 27
\end{aligned}
$$

Re-arrange into 35's and 27's

Optional Check:

$(-10) * 35 = -350$

$13 * 27 = 351$

Finding multiplicative inverse $\bmod m$

Suppose that $\gcd(a, m) = 1$.

By Bézout's Theorem, there exist integers s and t such that $sa + tm = 1$.

Therefore $sa \equiv 1 \pmod{m}$.

The multiplicative inverse b of a modulo m must also satisfy $0 \leq b < m$ so we set $b = s \bmod m$.

It works since $ba \equiv sa \equiv 1 \pmod{m}$

So… we can compute multiplicative inverses with the extended Euclidean algorithm.