### 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!

### Hypercubes.

Complete graphs, really connected! But lots of edges.

Trees, few edges. $|V| \log |V|$ edges!

Also represents bit-strings nicely.

$G = (V,E)$

$|V| = (0,1)^n$.

$|E| = \{(x,y) | x$ and $y$ differ in one bit position.$\}$

2$^n$ vertices. number of $n$-bit strings!

$n2^{n-1}$ edges.

2$^n$ vertices each of degree $n$

total degree is $n2^n$ and half as many edges!

### Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x,1x)$.

### Proof of Large Cuts.

**Thm:** For any cut $(S, V-S)$ in the hypercube, the number of cut edges is at least the size of the small side.

**Proof:**

Base Case: $n = 1$ $V = \{0,1\}$.

$S = \emptyset$ has one edge leaving. $|S| = 0$ has 0.

### Induction Step Idea

**Thm:** For any cut $(S, V-S)$ in the hypercube, the number of cut edges is at least the size of the small side.

Use recursive definition into two subcubes.

Two cubes connected by edges.

**Case 1:** Count edges inside subcube inductively.

**Case 2:** Count inside and across.
### Summary

Euler: \(v + f = e + 2\).
- Tree: Plus adding edge adds face.
- Planar graphs: \(e \leq 3v = 6\).
- Count face-edge incidences to get \(2e \leq 3f\).
- Replace \(f\) in Euler.
- Coloring: degree \(d\) vertex can be colored if \(d + 1\) colors.
- Small degree vertex in planar graph: 6 color theorem.
- Recolor separate and planarity: 5 color theorem.
- Graphs:
  - Trees: sparsest connected.
  - Complete densest
  - Hypercube: middle.

### Modular Arithmetic

**Theorem:** If \(d|\ x\) and \(d|\ y\), then \(d|(y - x)\).

**Proof:**
\[
x = ad, \ y = bd, \\
(x - y) = (ad - bd) = d(a - b) \implies d|(x - y).
\]

### Induction Step

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.
- Recursive definition: \(H_0 = (V_0, E_0), H_1 = (V_1, E_1)\), edges \(E\) that connect them.
- \(H = (V_0 \cup V_1, E_0 \cup E_1)\)
- \(S = S_0 \cup S_1\) where \(S_0\) in first, and \(S_1\) in other.

**Case 1:** \(|S_0| \leq |V_0|/2, |S_1| \leq |V_1|/2\)
- Both \(S_0\) and \(S_1\) are small sides. So by induction.
  - Edges cut in \(H_0 \geq |S_0|\).
  - Edges cut in \(H_1 \geq |S_1|\).
  - Total cut edges \(\geq |S_0| + |S_1| = |S|\).

**Induction Step. Case 2.**

**Thm:** For any cut \((S, V - S)\) in the hypercube, the number of cut edges is at least the size of the small side, \(|S|\).

**Proof:** Induction Step.
- \(|S_0| \geq |V_0|/2\).
- Recall Case 1: \(|S_0|, |S_1| \leq |V|/2\)
  - \(|S_1| \leq |V_1|/2\) since \(|S_1| \leq |V_1|/2\)
  - \(|S_0| \geq |V_0|/2\) \(\implies |V_0| - |S_0| \leq |V_0|/2\)
  - \(|S_0| \geq |V_0|/2 \implies |V_0| - |S_0| \leq |V_0|/2\)
- Edges in \(E_0\) connect corresponding nodes.
  - \(|S_0| - |S_1|\) edges cut in \(E_0\).
- Total edges cut:
  - \(\geq |S_1| + |V_0| - |S_0| - |S_1| = |V_0|\)
  - \(|V_0| = |V|/2 \geq |S_1|\).
- Also, case 3 where \(|S_1| \geq |V|/2\) is symmetric.

### Hypercubes and Boolean Functions

The cuts in the hypercubes are exactly the transitions from 0 sets to 1 set on boolean functions on \(\{0,1\}^n\).
- Central area of study in computer science!
- Yes/No Computer Programs = Boolean function on \(\{0,1\}^n\)
- Central object of study.

### Key ideas for modular arithmetic

**Theorem:** If \(d|x\) and \(d|y\), then \(d|(y - x)\).

**Proof:**
\[
x = ad, \ y = bd, \\
(x - y) = (ad - bd) = d(a - b) \implies d|(x - y).
\]

**Theorem:** Every number \(n \geq 2\) can be represented as a product of primes.

**Proof:** Either prime, or \(n = a \times b\), and use strong induction. (Uniqueness? Later.)
What did we use in our proofs of key ideas?
(A) Distributive Property of multiplication over addition.
(B) Euler’s formula.
(C) The definition of a prime number.
(D) Euclid’s Lemma.
(A) and (C)

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] (Almost 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.
Mod 7 equivalence or residue 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].

Next Up.

Modular Arithmetic.

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.
Mod 7 equivalence or residue 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].

Years and years...

80 years? 20 leap years. 366 × 20 days
60 regular years. 365 × 60 days
Today is day 4.
It is day 4 + 366 × 20 + 365 × 60. Equivalent to?
Hm.
What is remainder of 366 when dividing by ?? 52 × 7 + 2.
What is remainder of 365 when dividing by ?? 1
Today is day 4.
Get Day: 4 + 2 × 20 + 1 × 60 = 104
Remainder when dividing by ?? 104 = 14 × 7 + 6.
Or September 15, 2102 is Saturday!
Further Simplify Calculation:
20 has remainder 6 when divided by 7.
60 has remainder 4 when divided by 7.
Get Day: 4 + 2 × 6 + 1 × 4 = 20.
Or Day 6. September 14, 2103 is Saturday.
“Reduce” at any time in calculation!

Day of the week.
This is Thursday is September 14, 2023.
What day is it a year from now? on September 14, 2023?
Number days.
0 for Sunday, 1 for Monday, ..., 6 for Saturday.
Today: day 4.
5 days from then. day 9 or day 2 or Tuesday.
25 days from then. day 29 or day 1. 29 = (7)4 + 1
two days are equivalent up to addition/subtraction of multiple of 7.
11 days from then is day 1 which is Monday!
What day is it a year from then?
Next year is not a leap year. So 365 days from then.
Day 4+365 or day 369.
Smallest representation:
subtract 7 until smaller than 7.
divide and get remainder.
369/7 leaves quotient of 52 and remainder 5. 369 = 7(52) + 5
or September 15, 2022 is a Friday.
Notation

\[ x \mod m \] or \[ \text{mod}(x, m) \]
- remainder of \( x \) divided by \( m \) in \( \{0, \ldots, m-1\} \).

\[ \text{mod}(x, m) = x - \left\lfloor \frac{x}{m} \right\rfloor m \]
\[ \left\lfloor \frac{x}{m} \right\rfloor \] is quotient.

\[ \text{mod}(29, 12) = 29 - \left(\left\lfloor\frac{29}{12}\right\rfloor\right) \times 12 = 29 - 2 \times 12 = 5 \]

Work in this system.
\[ a \equiv b \pmod m \]
Says two integers \( a \) and \( b \) are equivalent modulo \( m \).

Modulus is \( m \)
\[ 6 = 3 + 3 = 3 + 10 \pmod 7. \]
\[ 6 = 3 + 3 = 3 + 10 \pmod 7. \]
Generally, not 6 (mod 7) = 13 (mod 7).
But probably won’t take off points, still hard for us to read.

Greatest Common Divisor and Inverses.

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

Proof
\[ \implies \]
Claim: The set \( S = \{0x, 1x, \ldots, (m - 1)x\} \) contains \( y = 1 \mod m \) if all distinct modulo \( m \).
Each of \( m \) numbers in \( S \) correspond to one of \( m \) equivalence classes modulo \( m \).
\[ \implies \] One must correspond to 1 modulo \( m \). Inverse Exists!
Proof of Claim: If not distinct, then \( \exists a, b \in \{0, \ldots, m - 1\}, a \neq b \), where \( (ax = bx \pmod m) \implies (a - b)x = 0 \pmod m \)
Or \( (a - b)x = km \) for some integer \( k \).
\[ \gcd(x, m) = 1 \]
\[ \implies \] Prime factorization of \( m \) and \( x \) do not contain common primes.
\[ \implies \] (a - b) factorization contains all primes in \( m \)'s factorization.
So \( (a - b) \) has to be multiple of \( m \).
\[ \implies (a - b) \geq m. \] But \( a, b \in \{0, \ldots, m - 1\}. \) Contradiction.

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 \( m \).
\[ \implies \]
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. Can’t be 1. No inverse.
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).
(Hmm. What normal number is it own multiplicative inverse?) \( 1 \cdot 1 \).
\[ 5x = 3 \pmod 6 \] What is \( x \)? Multiply both sides by 5.
\[ x = 15 \equiv 3 \pmod 6 \]
\[ 4x = 3 \pmod 6 \] No solutions. Can’t get an odd.
\[ 4x = 2 \pmod 6 \] Two solutions! \( x = 2, 5 \pmod 6 \)

Poll

Mark true statements.
(A) Multiplicative inverse of 2 mod 5 is 3 mod 5.
(B) The multiplicative inverse of \( ((n - 1) \pmod n) \) is \( (n - 1) \pmod n \).
(C) Multiplicative inverse of 2 mod 5 is 0.5.
(D) Multiplicative inverse of 4 mod 5 is 0.5.
(E) \( -1 \cdot (-1) = 1 \mod 5 \).

Proof Review 2: Bijections.

If \( \gcd(x, m) = 1 \).
Then the function \( f(a) = xa \pmod m \) is a bijection.
One to one: there is a unique pre-image(simple x where \( y = f(x) \).)
Onto: the sizes of the domain and co-domain are the same.
\[ x = 3, m = 4, \]
\[ f(1) = 3(1) = 3 \pmod 4 \]
\[ f(2) = 6 = 2 \pmod 4 \]
\[ f(3) = 1 \pmod 3 \]
Oh yeah. \( f(0) = 0 \pmod 3 \).

Bijection \( \implies \) unique pre-image and same size.
All the images are distinct. \( \implies \) unique pre-image for any image.
\[ x = 2, m = 4, \]
\[ f(1) = 2 \]
\[ f(2) = 0 \]
\[ f(3) = 2 \]
Oh yeah. \( f(0) = 0 \).

Not a bijection.
Poll

Which is bijection?
(A) \(f(x) = x\) for domain and range being \(\mathbb{R}\)
(B) \(f(x) = ax \pmod{n}\) for \(x \in \{0, \ldots, n-1\}\) and \(\gcd(a, n) = 2\)
(C) \(f(x) = ax \pmod{n}\) for \(x \in \{0, \ldots, n-1\}\) and \(\gcd(a, n) = 1\)

(B) is not.

25 / 43

Only if

Thm: If \(\gcd(x, m) \neq 1\) then \(x\) has no multiplicative inverse modulo \(m\).
Assume the inverse of \(a\) is \(x^{-1}\), or \(ax = 1 + km\).
Thus,
\[a(nd) = 1 + k\ell d\]
\[d(na - k\ell) = 1.\]
But \(d > 1\) and \(z = (na - k\ell) \in \mathbb{Z}\).
so \(dz \neq 1\) and \(dz = 1\). Contradiction.

26 / 43

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.

27 / 43

Inverses

Next up.
Euclid's Algorithm.
Runtime.
Euclid's Extended Algorithm.

Refresher

Does \(2\) have an inverse mod 8? No.
Any multiple of \(2\) is \(2\) away from \(0 + 8k\) for any \(k \in \mathbb{N}\).
Does \(2\) have an inverse mod 9? Yes. 5
\(2(5) = 10 = 1 \pmod{9}\).
Does \(6\) have an inverse mod 9? No.
Any multiple of \(6\) is \(3\) away from \(0 + 9k\) for any \(k \in \mathbb{N}\).\n\(3 = \gcd(6, 9)\!\).
\(x\) has an inverse modulo \(m\) if and only if \(\gcd(x, m) > 1\)? No.
\(\gcd(x, m) = 1\)? Yes.
Now what?:
Compute \(\gcd!\)
Compute Inverse modulo \(m\).

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 \(d|(x-y)\).
Is it a fact? Yes? No?
Proof: \(d|\!\!x\) and \(d|\!\!y\) or
\(x = \ell d\) and \(y = k d\)
\(\Rightarrow x - y = kd - \ell d = (k - \ell)d \Rightarrow d|(x-y)\)

28 / 43
Euclid's algorithm.

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

Hey, what's gcd(7, 0)? 7 since 7 divides 7 and 7 divides 0
What's gcd(x, 0)? x

(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y)))))

Theorem: (euclid x y) = gcd(x, y) if x \geq y.

Proof: Use Strong Induction.

Base Case: y = 0, "x divides y and x" 
\Rightarrow "x 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.

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| mod(x, y).

Proof:
\[ \text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y \]
\[ = x - s \cdot y \text{ for integer s} \]
\[ = kd - s\cdot d \text{ for integers k, s where } x = kd \text{ and } y = s\cdot d \]
\[ = (k - s)\cdot d \]

Therefore d| mod(x, y). And d|y since it is in condition.

Lemma 2: If d|y and d| mod(x, y) then d|y and d|x.

Proof: Similar. Try this at home.

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

Proof: x and y have same set of common divisors as x and
mod(x, y) by Lemma 1 and 2.

Same common divisors \Rightarrow largest is the same.

Euclid procedure is fast.

Theorem: (euclid x y) uses 2n "divisions" where n = b(x) \approx log_2 x.

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.

If y \approx x roughly y uses n bits ...

\[ 2^{n-1} \text{ divisions! Exponential dependence on size!} \]

101 bit number. \[ 2^{100} \approx 10^{30} = \text{"million, trillion, trillion" divisions!} \]

2n is much faster! \ldots roughly 200 divisions.

Poll

Assume \( \log_2 1,000,000 \) is 20 to the nearest integer.
Mark what’s true.

(A) The size of 1,000,000 is 20 bits.
(B) The size of 1,000,000 is one million.
(C) The value of 1,000,000 is one million.
(D) The value of 1,000,000 is 20.

(A) and (C).

Poll

Which are correct?

(A) gcd(700,568) = gcd(568,132)
(B) gcd(8,3) = gcd(3,2)
(C) gcd(8,3) = 1
(D) gcd(4,0) = 4

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 in base 10: 7.
Number of bits (a digit in base 2): 21.

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

\[ n = b(x) = \log_2 x \]
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 \ldots , check \frac{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)

4

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

Remark

(define (euclid x y) (if (= y 0) x (euclid y (- x y))))

Didn't necessarily need to do gcd.
Runtime proof still works.

Runtime Proof.

(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

Theorem: (euclid x y) uses $O(n)$ “divisions” where $n = \log_2(x)$.

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.

Case 1: $y < \frac{x}{2}$, first argument is $y$ ⇒ true in one recursive call;

Case 2: Will show "$y \geq \frac{x}{2}$ ⇒ $\text{mod}(x, y) \leq \frac{x}{2}$.”
$\text{mod}(x, y)$ is second argument in next recursive call, and becomes the first argument in the next one.
When $y \geq \frac{x}{2}$, then
\[
\left\lfloor \frac{x}{y} \right\rfloor = 1,
\]
$\text{mod}(x, y) = x - y \left\lfloor \frac{x}{y} \right\rfloor = x - y \leq x - \frac{x}{2} = \frac{x}{2}$

Finding an inverse?

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

Euclid’s GCD algorithm.

(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

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

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

Claim: \( \text{Returns } (d, a, b) : d = \text{gcd}(a, b) \text{ and } d = ax + by. \)

Example: \( a = 35, b = -1 \)
\[
\text{ext-gcd}(35, 12) \
\text{ext-gcd}(12, 11) \
\text{ext-gcd}(11, 1) \
\text{ext-gcd}(1, 0) \
\text{return } (1, 1, 0) ; ; 1 = (1)1 + (0)0 \
\text{return } (1, 0, 1) ; ; 1 = (0)1 + (1)1 \
\text{return } (1, 1, -1) ; ; 1 = (1)12 + (-1)35 \
\text{return } (1, -1, 3) ; ; 1 = (-1)35 + (3)12 
\]

Theorem: \( \text{Returns } (d, a, b), \text{ where } d = \text{gcd}(a, b) \text{ and } d = ax + by. \)
Correctness.

Proof: Strong Induction.1
Base: 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: ext-gcd(y, mod(x,y)) returns (d,a,b) with d = ay + b( mod(x,y))

ext-gcd(x,y) calls ext-gcd(y, mod(x,y)) so

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

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


Prove: returns (d,A,B) where d = Ax + By.

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)

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

Returns (d,b,(a − \left\lfloor \frac{x}{y} \right\rfloor b)).

Hand Calculation Method for Inverses.

Example: gcd(7,60) = 1.
egcd(7,60).
7(0) + 60(1) = 60
7(1) + 60(0) = 7
7(−8) + 60(1) = 4
7(9) + 60(−1) = 3
7(−17) + 60(2) = 1
Confirms: −119 + 120 = 1

Wrap-up

Conclusion: Can find multiplicative inverses in \(O(n)\) time!

Very different from elementary school: try 1, try 2, try 3...

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

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs. (10000000000000000000000000000000000000000000)5 divisions.

Internet Security: Next Week.