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 \( \gcd(x, 0) \)?

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

Theorem: \( \gcd(x, y) = \gcd(x, y) \) 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.

Euclid procedure is fast.

Theorem: \( (euclid x y) \) uses \( 2n \) “divisions” where \( n = \log_2 x \).

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

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

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

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

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

\( 2n \) is much faster! … roughly 200 divisions.

To homework or not to homework.

Form extended.

There is NOT a “scoring bump” nor is there a “scoring detriment” for doing homework option versus non-homework option.

We grade by making buckets according to quality of exams on exam totals.

Determines number of grades at each level. Then re-sort homework students only. So only their grades are affected.

Difference is about how you want to use your time and effort to do your best learning.

Today

Finish Euclid.

Bijection/CRT/Isomorphism.

Fermat's Little Theorem.

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?

\[
\begin{align*}
\text{number of bits} & = \log_2 x
\end{align*}
\]

Euclid procedure is fast.

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:

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

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

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

Proof: Similar. Try this at home.

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.

Check 5 …, check 7 …, check 4 …, check 3 ….

Try this at home.
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).

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

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

Mark correct answers.

(A) mod(x,y) < y
(B) If euclid(x,y) calls euclid(u,v) calls euclid(a,b) then a <= x/2.
(C) euclid(x,y) calls euclid(u,v) means u = y.
(D) if $y > x/2$, then mod(x,y) < y/2
(E) if $y > x/2$, then mod(x,y) = (y - x)

(D) is not always true.
Finding an inverse?

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

Euclid’s GCD algorithm.

We saw how to efficiently tell if there is an inverse.

Make d out of multiples of x and y .

Euclid’s GCD theorem: For any x, y there are integers
a, b such that
\[ ax + by = d \]
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 \( \gcd(x, m) = 1 \).

\[ ax + bm = 1 \]

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

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

(3)12 + (−1)35 = 1.

The multiplicative inverse of 12 (mod 35) is 3.

Check: 3(12) = 36 = 1 (mod 35).

Extended GCD Algorithm.

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
\[ ax + by = d \]
where \( d = \gcd(x, y) \).

Compute the \( \gcd(x, y) \) in \( O(n) \) divisions. (Remember \( n = \log_2(x) \).

For x and m, if \( \gcd(x, m) = 1 \) then x has an inverse modulo m.

Make d out of multiples 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?

\[ 35 - (\frac{3}{2})12 = 35 - (2)12 = 11 \]

How does gcd get 1 from 12 and 11?

\[ 12 - (\frac{3}{2})11 = 12 - (1)11 = 1 \]

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 \( b = −1 \).

Multiplicative Inverse.

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

How do we find a multiplicative inverse?
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(a,b) and
\[ d = ax + by. \]

Hand Calculation Method for Inverses.

Example: gcd(7,60) = 1.
\[ \text{egcd}(7,60): \]
\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]

Confirm: -119 + 120 = 1
Note: an “iterative” version of the e-gcd algorithm.

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

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

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

Wrap-up

Conclusion: Can find multiplicative inverses in O(n) time!
Very different from elementary school: try 1, try 2, try 3...
\[ x \equiv y \mod \phi(n) \]
Inverse of 500,000,357 modulo 1,000,000,000,000?
\[ \leq 80 \text{ divisions.} \]
versus 1,000,000

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs. \((1000000000000000000000000000000000000000000)\) divisions.
Internet Security: Soon.


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

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

Bijections

Bijection is one to one and onto.
Bijection:
f : A -> B.
Domain: A, Co-Domain: B.

Versus Range.
E.g. \[ \sin(x). \]
\[ A = \mathbb{R} \text{ reals.} \]
Range is \([-1,1]\]. Onto: \([-1,1]\].
Not one-to-one. \[ \sin(\pi) = \sin(0) = 0. \]

Range Definition always is onto.
Consider \[ f(x) = ax \mod m. \]
of \[ (0,\ldots,m-1) \rightarrow (0,\ldots,m-1). \]
\[ f(0) = f(1) = \ldots = f(m-1). \]
Domain/Co-Domain: \([0,\ldots,m-1]\).

When is it a bijection?
When \[ \text{gcd}(a,m) \text{ is } 1. \]
Not Example: \[ a = 2, m = 4, f(0) = f(2) = 0 \mod 4. \]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \ (\text{mod } m) \) and \( x = b \ (\text{mod } n) \) where \( \gcd(m, n) = 1 \).

**Proof (solution exists):**

Consider \( u = n(m^{-1}) \ (\text{mod } m) \).

\[ u = 0 \ (\text{mod } n) \quad u = 1 \ (\text{mod } m) \]

Consider \( v = m(m^{-1}) \ (\text{mod } m) \).

\[ v = 1 \ (\text{mod } n) \quad v = 0 \ (\text{mod } m) \]

Let \( x = au + bv \).

\[ x = a \ (\text{mod } m) \quad \text{since } bv = 0 \ (\text{mod } m) \quad \text{and } au = a \ (\text{mod } m) \]

\[ x = b \ (\text{mod } n) \quad \text{since } au = 0 \ (\text{mod } n) \quad \text{and } bv = b \ (\text{mod } n) \]

This shows there is a solution.

**Proof (uniqueness):**

If not, two solutions, \( x \) and \( y \).

\[ (x - y) = 0 \ (\text{mod } m) \quad \text{and } (x - y) = 0 \ (\text{mod } n) \]

\[ \Rightarrow (x - y) \text{ is multiple of } m \text{ and } n \]

\[ \gcd(m,n) = 1 \Rightarrow \text{no common primes in factorization } m \text{ and } n \]

\[ \Rightarrow mn(x - y) \]

\[ x - y \geq mn \Rightarrow x, y \not\in \{0, \ldots, mn - 1\} \]

Thus, only one solution modulo \( mn \).
Which was used in Fermat’s theorem proof?
(A) The mapping \( f(x) = ax \mod p \) is a bijection.
(B) Multiplying a number by 1, gives the number.
(C) All nonzero numbers mod \( p \), have an inverse.
(D) Multiplying a number by 0 gives 0.
(E) Multiplying elements of sets \( A \) and \( B \) together is the same if \( A = B \).

(A), (C), and (E)

Fermat and Exponent reducing.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),
\[
a^{p-1} \equiv 1 \pmod{p}.
\]

What is \( 2^{101} \pmod{7} \)?

Wrong: \( 2^{101} = 2^{7 \cdot 14 + 3} = 2^3 \pmod{7} \)

Fermat: 2 is relatively prime to 7. \( \implies 2^6 \equiv 1 \pmod{7} \).

Correct: \( 2^{101} = 2^{6 \cdot 16 + 5} = 2^5 = 32 = 4 \pmod{7} \).

For a prime modulus, we can reduce exponents modulo \( p - 1 \)!

Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).

Proof Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse: \( ax + by = ax = \gcd(x, y) \pmod{y} \)
If \( \gcd(x, y) = 1 \), we have \( ax = 1 \pmod{y} \)
\( a = x^{-1} \pmod{y} \).

Chinese Remainder Theorem:
If \( \gcd(n, m) = 1 \), \( x = a \pmod{n} \), \( x = b \pmod{m} \) unique sol.
Proof: Find \( u = 1 \pmod{n} \), \( u = 0 \pmod{m} \),
and \( v = 0 \pmod{n} \), \( v = 1 \pmod{m} \).
Then \( x = au + bv \pmod{mn} \).

Fermat: Prime \( p \), \( a^{p-1} = 1 \pmod{p} \).
Proof Idea: \( f(x) = a(x) \pmod{p} \): bijection on \( S = \{1, \ldots, p-1\} \).
Product of elts \( \equiv \) for range/domain: \( a^{p-1} \) factor in range.