Lecture 2: Greatest Common Divisor

Lecture 2

Objectives

By the end of this lecture you should be able to understand

1. Greatest Common Divisor
2. Euclid's Algorithm for computing GCD
3. Extended Euclid's Algorithm for computing GCD and its certificate
4. Applications of GCD

Outline

1. Greatest Common Divisor
2. Euclid Algorithm
3. Extended Euclid Algorithm
4. Applications of GCD
   - Least Common Multiple
   - Diophantine Equations
   - Modular Division

Greatest Common Divisor

Definition

The greatest common divisor, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest positive integer d that divides both a and b, i.e.,

- $d \mid a$ and $d \mid b$
- If $c \mid a$ and $c \mid b$, then $c \leq d$

Example

$\gcd(24, 16) = 8, \gcd(9, 17) = 1, \gcd(239; 0) = 239$

Note: Since $d \mid a$ implies that $d \mid -a$, then we assume that a and b are non-negative since negative numbers have same gcd.

Some Applications

- Computing Modular Inverse: Given integers a and n, how to find an integer k such that $ak \equiv 1 \pmod{n}$? Basic primitive in modern crypto protocols, used billions times per day

- Computing Fractions:

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

```python
from fractions import Fraction
print(Fraction(31, 177) + Fraction(29, 59))
```

Naive Algorithm

To find the greatest common divisor, simply try all numbers and select the largest one.

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0
    if a == 0 or b == 0:
        return max( a, b )
    for d in range ( min ( a, b ), 0, -1):
        if a % d == 0 and b % d == 0:
            return d
    return 1
```

Naive Algorithm: Analysis

- If $\gcd = 1$, the algorithm will perform $\min\{a, b\}$ divisions
- Modern laptops perform roughly one billion (10^9) operations per second
- On your laptop, the call
  print (gcd(790933790547, 1849639579327))
  will take more than one minute
- If a and b consist of hundreds of digits (typical case for crypto protocols), even on a supercomputer with quadrillion operations per second this algorithm will run for more than thousand years

Euclid Algorithm

- Efficient algorithm for computing the greatest common divisor of two integers
- *"We might call Euclid's method the granddaddy of all algorithms, because it is the oldest nontrivial algorithm that has survived to the present day."* Donald E. Knuth, The Art of Computer Programming : Seminumerical Algorithms
- Euclid's algorithm for finding the greatest common divisor of two integers is perhaps the oldest nontrivial algorithm that has survived to the present day

Euclid's Lemma

Lemma

$d \mid a$ and $d \mid b$, iff $d \mid a - b$ and $d \mid b$, i.e., $\mathrm{gcd(a, b) = gcd(a - b, b)}$

Proof.

$\Rightarrow$ if $a = dp$ and $b = dq$, then $a - b = d(p - q)$
$\Leftarrow$ if $a - b = dp$ and $b = dq$, then $a = d(p + q)$

- Every common divisor of a and b is also a common divisor of $a - b$ and b
- Every common divisor of $a - b$ and b is also a common divisor of a and b
- Therefore, $\mathrm{gcd(a, b) = gcd(a - b, b)}$

□

A Better Algorithm

Main Idea: Use this concept to reduce the numbers by differencing

```
def gcd(a, b ):
    assert a >= 0 and b >= 0 and a + b > 0
    while a > 0 and b > 0:
        if a >= b:
            a = a - b
        else:
            b = b - a
    return max(a, b)
```

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow: gcd(790933790548, 7)
- Reason: the code will subtract 7 billions of times!
- Idea: what is left is the reminder modulo 7

Euclid's Algorithm

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0
    while a > 0 and b > 0:
        if a >= b:
            a = a % b
        else:
            b = b % a
    return max(a, b)
```

Computing the GCD

Theorem

Let $a = qb + r$*, where* $a, b, q, r \in \mathbb{Z}$*. Then* $\gcd(a, b) = \gcd(b, r)$ *or in other words* $\gcd(a, b) = \gcd(b, a \bmod b)$.

Proof.

- Assume $d \mid a$ and $d \mid b$, then $d \mid a - bq$. It follows that every common divisor of a and b is also a common divisor of b and r
- Conversely, assume $d \mid b$ and $d \mid r$, then $d \mid qb + r$. It follows that every common divisor of b and r is also a common divisor of a and b.
- Hence, $\gcd(a, b) = \gcd(b, r)$

$\square$

Euclid Algorithm with Logging

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0
    while a > 0 and b > 0:
        print( f"gcd({a}, {b}) = ")
        if a >= b:
            a = a % b
        else:
            b = b % a
        print( f"gcd({a}, {b}) = ")
    return max(a, b)
```

Euclid Algorithm with Logging

```
gcd(790933790547, 1849639579327) =
gcd(790933790547, 267771998233) =
gcd(255389794081, 267771998233) =
gcd(255389794081, 12382204152) =
gcd(7745711041, 12382204152) =
gcd(7745711041, 4636493111) =
gcd(3109217930, 4636493111) =
gcd(3109217930, 1527275181) =
gcd(54667568, 1527275181) =
gcd(54667568, 51250845) =
gcd(3416723, 51250845) =
gcd(3416723, 0) =
3416723
```

Analysis

- Already quite fast: if a and b are 100 digits long, the number of iterations of the while loop is at most 660
- Each iteration is a division. Can we avoid division?
- The numbers are getting shorter and shorter item A more quantitative statement: at each iteration of the while loop the larger number drops by at least a factor of 2

Lemma

Let $a \geq b > 0$. Then $a \bmod b < a/2$.

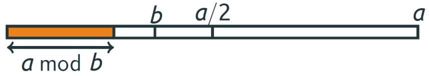
Analysis

Lemma

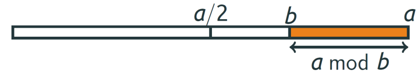
Let $a \geq b > 0$. Then $a \bmod b < a/2$.

Proof.

- If $b \leq a/2$, then $a \bmod b < b \leq a/2$



- If $b > a/2$, then $a \bmod b = a - b < a/2$



$\square$

Compact Code

This the compact code for Euclid Algorithm that is usually used

```
def gcd ( a , b ) :
    assert a >= b and b >= 0 and a + b > 0
    return gcd (b, a % b) if b > 0 else a
```

Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2
- Thus, the total number of iterations is at most $\log_2 a + \log_2 b$
- If a consists of less than 5000 decimal digits, i.e., $a < 10^{5000}$, then $\log_2 a < 16600$

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d
- You can check that d divides both a and b, but this only shows that d is a common divisor of a and b, but does not guarantees that is the greatest one
- It turns out that it is enough to represent d as $ax + by$ (for integers x and y)!

Test

Lemma

If $d \mid a$, $d \mid b$, and $d = ax + by$ for integers x and y, then $d = \gcd(a, b)$

Proof.

- d is a common divisor of a and b, then $d \leq \gcd(a, b)$
- $\gcd(a, b)$ divides both a and b, then it also divides $d = ax + by$, and $\gcd(a, b) \leq d$

$\square$

Examples

- $\gcd(10, 6) = 2 = 10 \times -1 + 6 \times 2$
- $\gcd(7, 5) = 1 = 7 \times -2 + 5 \times 3$
- $\gcd(391, 299) = 23 = 391 \times -3 + 299 \times 4$
- $\gcd(239, 201) = 1 = 239 \times -37 + 201 \times 44$

Extending Euclid's Algorithm

Given certificate of $\gcd(b, a \bmod b)$, how to get certificate of $\gcd(a, b)$??

- Recall that Euclid's algorithm uses the fact that for $a \geq b$, we have $\gcd(a, b) = \gcd(b, a \bmod b)$
- Assume that $d = \gcd(b, a \bmod b)$ and that $d = bp + (a \bmod b)q$
- Then

$$\begin{aligned} d &= bp + (a \bmod b)q \\ &= bp + \left(a - \lfloor \frac{a}{b} \rfloor b\right) q \\ &= aq + b(p - \lfloor \frac{a}{b} \rfloor q) \end{aligned}$$

Extended Euclid Algorithm

```
# returns gcd (a, b), x, y: gcd (a, b) = ax+by
def extended_gcd (a, b):
    assert a >= b and b >= 0 and a + b > 0
    if b == 0:
        d , x , y = a , 1 , 0
    else:
        (d, p, q) = extended_gcd(b , a % b)
        x = q
        y = p - q * ( a // b )
    assert a % d == 0 and b % d == 0
    assert d == a * x + b * y
    return (d, x, y)
```

Least Common Multiple

Definition

The least common multiple, $\operatorname{lcm}(a, b)$, of integers a and b (both different from zero) is the smallest positive integer that is divisible by both a and b, i.e.,

- $a \mid \operatorname{lcm}(a, b)$ and $b \mid \operatorname{lcm}(a, b)$
- If $a \mid c$ and $b \mid c$, then $c \geq \operatorname{lcm}(a, b)$

Example

$\operatorname{lcm}(24, 16) = 48$, $\operatorname{lcm}(9, 17) = 153$, $\operatorname{lcm}(239, 0)$ is undefined

Naive Algorithm

ab is divisible by a and b. To find the least common multiple, try all numbers up to ab and select the smallest one.

```
def lcm(a, b):
assert a > 0 and b > 0
for d in range (1, a * b + 1):
    if d % a == 0 and d % b == 0 :
        return d
```

- If $\operatorname{lcm}(a,b) = ab$, the algorithm will perform $a \times b$ divisions
- Can we use efficient Euclid's algorithm to compute lcm? YES!

Relation between lcm and gcd

Lemma

If $a, b > 0$, *then* $\text{lcm}(a, b) = ab/\text{gcd}(a, b)$

Proof.

- Let $d = \text{gcd}(a, b)$, $a = dp$, $b = dq$
- Then, $m = ab/d = dpq = pb = qa$ is a multiple of both a and b
- If there was a smaller multiple $\widehat{m} < m$, then $\widehat{d} = ab/\widehat{m} > d$ would be a common divisor: $a/\widehat{d} = \widehat{m}/b$, $b/\widehat{d} = \widehat{m}/a$
- But this leads to a contradiction because d should be the greatest common divisor and $\widehat{d}$ should not be less than d.

$\square$

Outline

1. Greatest Common Divisor
2. Euclid Algorithm
3. Extended Euclid Algorithm
4. Applications of GCD
   - Least Common Multiple
   - Diophantine Equations
   - Modular Division

Diophantine Equations

- A Diophantine equation is an equation where only integer solutions are allowed.
- The simplest type is linear Diophantine equation in 2 unknowns

$$ax + by = c$$

  where a, b, c are given integers, and a, b are not both zero
- A Diophantine such as $3x + 6y = 18$ can have a number of solutions, or no solutions such as $2x + 10y = 17$

What are the conditions at which a solution exists? and what is the general form of solution?

Solutions of Diophantine Equations

Theorem

Given integers a, b and c, where at least one of a and b $\neq 0$, the Diophantine equation

$$ax + by = c$$

has a solution (where x and y are integers) iff

$$\gcd(a, b) \mid c$$

Proof of Theorem

Theorem

$ax + by = c$ *has an integer solution* $\iff \gcd(a, b) \mid c$

Proof.

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

$\Rightarrow$

- $a = dp$ and $b = dq$
- Thus $c = ax + by = d(px + qy)$, i.e., $d \mid c$

$\Leftarrow$

- Extended Euclid's algorithm: $a\bar{x} + b\bar{y} = d$
- $d \mid c$ implies $c = td$, then $c = at\bar{x} + bt\bar{y} = ax + by$

$\square$

Numerical Examples

- $10x + 6y = 14$
  - Extended Euclid's algorithm:

$$\gcd(10, 6) = 2 = 10(-1) + 6(2)$$

  - $14 = 2(7) = 10(-7) + 6(14)$
  - Then $x = -7, y = 14$
- $391x + 299y = -69$
  - Extended Euclid's algorithm:

$$\gcd(391, 299) = 23 = 391(-3) + 299(4)$$

  - $-69 = 23(-3) = 391(9) = 299(-12)$
  - Then $x = 9, y = -12$
  - But $x = -4, y = 5$ is also a solution. How to find all solutions?

Euclid's Lemma

Lemma

If $n \mid ab$ *and* $\gcd(a, n) = 1$*, then* $n \mid b$

Proof.

- From Extended Euclid's algorithm on (a, n):
- $ax + ny = 1 \Rightarrow axb + nyb = b \quad (1)$
- Since $n \mid ab$, then $ab = kn$.
- substitute in (1):

$$b = nkx + nyb = n(xk + yb)$$

$\square$

If $\gcd(a, b) = 1$, a and b are called co-primes or relatively primes.

Finding All Solutions

Theorem

Let $\gcd(a,b) = d$*,* $a = dp$*,* $b = dq$*. If* (x_0, y_0) *is a solution of the Diophantine equation* $ax + by = c$*, i.e.,*

$$ax_0 + by_0 = c,$$

then all solutions have the form

$$a(x_0 + tq) + b(y_0 - tp) = c,$$

where t *is any arbitrary integers.*

Proof of the Theorem – Part 1: Existence

Proof.

- $a = dp$, $b = dq$, $ax_0 + by_0 = c$
- For any integer t:

$$\begin{aligned} & a(x_0 + tq) + b(y_0 - tp) \\ &= ax_0 + by_0 + t(aq - bp) \\ &= c + t(dpq - dpq) = c \end{aligned}$$

  is a solution

$\square$

Proof of the Theorem – Part 2: Uniqueness

Proof.

- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)
- Subtract the 2 equations:

$$a(x_1 - x_2) + b(y_1 - y_2) = c - c = 0$$

- Divide by d:

$$p(x_1 - x_2) + q(y_1 - y_2) = 0$$

- Since $\gcd(a, b) = d$, $a = dp$, $b = dq \Rightarrow \gcd(p, q) = 1$
- Euclid's lemma: $q \mid p(x_1 - x_2)$, $\gcd(p, q) = 1 \Rightarrow x_1 - x_2 = tq$
- Then $y_1 - y_2 = -tp$

$\square$

Word Problem Example

Example

A customer bought 12 pieces of fruit, apples and oranges, for \$1.32. If an apple costs 3 cents more than an orange and more apples than oranges were purchased, how many pieces of each kind were bought?

Solution

- *Let x be number of apples, y number of oranges, and z is the cost of an orange in cents.*
- *Then we have* $(z+3)x+zy=132$
- *or equivalently* $3x+(x+y)z=132$
- *but* $x+y=12$, *then* $3x+12z=132$ *or* $x+4z=44$
- *Look in Burton's textbook page 36 for the rest of solution*

Outline

1. Greatest Common Divisor
2. Euclid Algorithm
3. Extended Euclid Algorithm
4. Applications of GCD
   - Least Common Multiple
   - Diophantine Equations
   - Modular Division

Division Mod 7

$\times$	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

- Given $a \neq 0$ and b, there exists x such that $a \times x \equiv b \pmod{7}$
- x plays the role of modular division $x \equiv b/a \pmod{7}$

Division Mod 6

$\times$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

- $2/5 \equiv 4(\text{mod}6)$
- But there is no x such that $3 \times x \equiv 1 \pmod 6$
- We cannot divide 1 by 3 modulo 6!

Multiplicative Inverse

- A multiplicative inverse of a mod n is $\bar{a}$ such that

$$a \times \bar{a} \equiv 1 \pmod{n}$$

- If a has a multiplicative inverse $\bar{a}$, then we can divide by a mod n:

$$b/a \equiv b \times \bar{a} \pmod{n}$$

- Indeed, for every b:

$$b/a \times a \equiv b \times \bar{a} \times a \equiv b \pmod{n}$$

Uniqueness of Inverses

Lemma

If a has a multiplicative inverse, then it is unique

Proof.

If x and y are multiplicative inverses of a, then

$$x \equiv x \times (a \times y) \equiv (x \times a) \times y \equiv y \pmod{n}$$

$\square$

Existence of Inverses

Theorem

a has a multiplicative inverse modulo n iff $\gcd(a, n) = 1$, i.e., a and n are co-primes or relatively primes.

Proof.

- $ax \equiv 1 \pmod{n}$ iff $ax + kn = 1$
- For fixed a and n, this Diophantine equation has a solution x iff $\gcd(a, n) = 1$

$\square$

Steps of Modular Division

- If $\gcd(a, n) = 1$, then the inverse exists and we can divide by a modulo n
- Given a, b, n, we want to find $x \equiv b/a \pmod{n}$:
  - First, use Extended Euclid's theorem to find (s, t) such that

$$nt + as = 1$$

  - Then s is the multiplicative inverse of a modulo n
  - Now,

$$x \equiv b/a \equiv b \times s \pmod{n}$$

Numerical Example

Find x such that $x \times 2 \equiv 7 \pmod 9$

- $\gcd(9, 2) = 1$, so we can compute

$$7/2 \pmod 9$$

- Extended Euclid's algorithm gives us

$$9(1) + 2(-4) = 1$$

- $-4 \equiv 5 \pmod 9$ is the inverse of 2 mod 9
- $7/2 \equiv 7 \times 5 \equiv 8 \pmod 9$
- Check: $8 \times 2 \equiv 7 \pmod 9$