CISC-102
Winter 2016
Lecture 12

Euclid's Algorithm

Suppose a,b are non-zero integers then we can define a function on the integers, gcd(a,b), that returns the greatest common divisor of a and b. It will be convenient to further assume that $|a| \geq |b|$.

Euclid's algorithm to compute gcd(a,b) is way more efficient than computing all the divisors a and b. The algorithm is based on the following theorem.

Euclid's Theorem:

Let a,b,q,r be integers such that $a = qb + r$ then

$$\mathbf{gcd(a,b) = gcd(b,r)}$$

This translates to the following iterative algorithm, implemented in Python.

Euclid's Algorithm in the Python programming language.

```
def euclid_gcd(a,b):
# Assume |a| >= |b| > 0
    r = a % b # this returns r s.t. a = bq + r
    while r > 0:
        a,b = b,r
        r = a % b # this returns r s.t. a = bq + r
    return b
```

NOTE: The % (mod) operator is found in many programming languages and returns the remainder when doing integer division.

For example: a = 154, b = 18.

so
iteration 0: (before the while loop) r = 154 % 18 = 10
iteration 1: r = 18 % 10 = 8
iteration 2: r = 10 % 8 = 2
iteration 3: r = 8 % 2 = 0

concluding that gcd(154,18) = 2

or
gcd(154,18) = gcd(18,10) = gcd(10,8) = gcd(8,2) = gcd(2,0) = 2.

Observe that as a side effect of Euclid's algorithm we can always find integers x,y such that gcd(a,b) = ax + by.

This can be illustrated with the previous example.

(1) 154 = (8) 18 + 10 implies 10 = 154 - (8)18
(2) 18 = (1) 10 + 8 implies 8 = 18 - (1)10
(3) 10 = (1) 8 + 2 implies 2 = 10 - (1)8

Now we can write gcd(154,18) = 2 as:

2 = 10 - (1)8 equation (3)
2 = 10 - (1)[18 - (1)10] equation (2)
2 = 154 - (8)18 - (1)[18 - (1)(154-(8)18)] equation (1)
2 = (2)154 - (17)18

The proof of Euclid's Theorem appears in lecture 11.

It can also be shown that this function is extremely efficient when compared to looking at all the divisors of a and b.

Let a = 250, and b = 575. We can construct a prime factorization of a and b.

Prime factorization:
$250 = (2)(5^3)$
$575 = (5^2)(23)$

We can inspect the prime factorization of a and b to obtain a greatest common divisor.

Observe that 5^2 is the greatest number that divides both a and b, that is, gcd(a,b). Using the prime factorizations of a and b is much less efficient than Euclid's algorithm. Nevertheless, the prime factorization is useful for obtaining other properties of the greatest common divisor.

Least Common Multiple

Given two non-zero[1] integers a,b we can have many values that are positive common multiples of both a & b. By the well ordering principle we know that amongst all of those multiples there is one that is smallest, and this is known as the *least common multiple* of a and b. We can define a function lcm(a,b) that returns this value.

Example: Suppose $a = 12$, and $b = 24$,
so we have $\mathrm{lcm}(a,b) = 24$.
In general if $a \mid b$ then $\mathrm{lcm}(a,b) = |b|$.
At this point it is worth mentioning that if $a \mid b$ then $\gcd(a,b) = |a|$, and that $\mathrm{lcm}(a,b) \times \gcd(a,b) = |ab|$.

[1] Multiples of zero are always zero, so this is a boring case.

Example: Suppose a = 13, and b = 24, we have lcm(a,b) = (13)(24). We can also observe that gcd(a,b) = 1, that is the numbers are relatively prime. In general if a and b are relatively prime, that is, if gcd(a,b) = 1 then lcm(a,b) = |ab|

So when gcd(a,b) = 1, we can observe that lcm(a,b) × gcd(a,b) = |ab|.

Let a = 250, and b = 575. We can construct a prime factorization of a and b

Prime factorization
$250 = (2)(5^3)$
$575 = (5^2)(23)$

We can inspect the prime factorization of a and b to obtain the least common multiple and the greatest common divisor using the formulae:

$$\gcd(575,250) = 2^{\min(1,0)} \times 5^{\min(3,2)} \times 23^{\min(0,1)} = 5^2$$

and

$$\begin{aligned} \operatorname{lcm}(575,250) &= 2^{\max(1,0)} \times 5^{\max(3,2)} \times 23^{\max(0,1)} \\ &= 2^1 \times 5^3 \times 23^1 \end{aligned}$$

So in this case we also have lcm(a,b) × gcd(a,b) = |ab|

$630 = (\ 2\ )\ (3^2)\ (\ 5\ )\ (\ 7\ )$
$84\ =\ (2^2)\ (\ 3\ )\ (\ 7\ )$

Using the formulae we get:

$$\begin{aligned} \gcd(630,84) &= 2^{\min(1,2)} \times 3^{\min(2,1)} \times 5^{\min(1,0)} \times 7^{\min(1,1)} \\ &= 2 \times 3 \times 7 \end{aligned}$$

and

$$\begin{aligned} \operatorname{lcm}(630,84) &= 2^{\max(1,2)} \times 3^{\max(2,1)} \times 5^{\max(1,0)} \times 7^{\max(1,1)} \\ &= 2^2 \times 3^2 \times 5 \times 7 \end{aligned}$$

Again we have

$$\begin{aligned} 630 \times 84 &= (\ 2\ )\ (3^2)\ (\ 5\ )\ (\ 7\ ) \times (2^2)\ (\ 3\ )\ (\ 7\ ) \\ &= (\ 2\ )\ (\ 3\ )\ (\ 7\ ) \times (2^2)\ (3^2)\ (\ 5\ )\ (\ 7\ ) \\ &= \gcd(630,84) \times \operatorname{lcm}(630,84) \end{aligned}$$

These ideas lead to the following theorem that is given without formal proof.

Theorem: Let a,b be non-zero integers, then

$$\gcd(a,b)\operatorname{lcm}(a,b) = |ab|.$$

Factoring vs. GCD

Factoring an integer N into its prime factors will use roughly $\sqrt{N}$ operations.

Computing gcd(N,m) with Euclid's algorithm for N > m ≥ 0 will use roughly $\log_2 N$ operations.

N	$\log_2 N$	$\sqrt{N}$
1024	10	32
1099511627776	40	1,048,576
1×10^{301}	1000	3.27×10^{150}

The efficiency of Euclid's gcd algorithm is essential for implementing current public key crypto systems that are commonly used for e-commerce applications.

With a "key" decoding an encrypted message using Euclid's algorithm takes about 1000 operations. Without a "key" breaking an encrypted message takes about 3.27×10^{150} operations. This amounts to a small fraction of a second for decoding and many millions of years for breaking the encrypted message.

Congruence Relations

Let a,b,m be integers, $m > 0$, such that

$a \% m = b \% m$ that is:

$a = (p)m + r$ and $b = (q)m + r$

for example: let $a = 7$, $b = 19$, $m = 12$

$7 = (0)12 + 7$

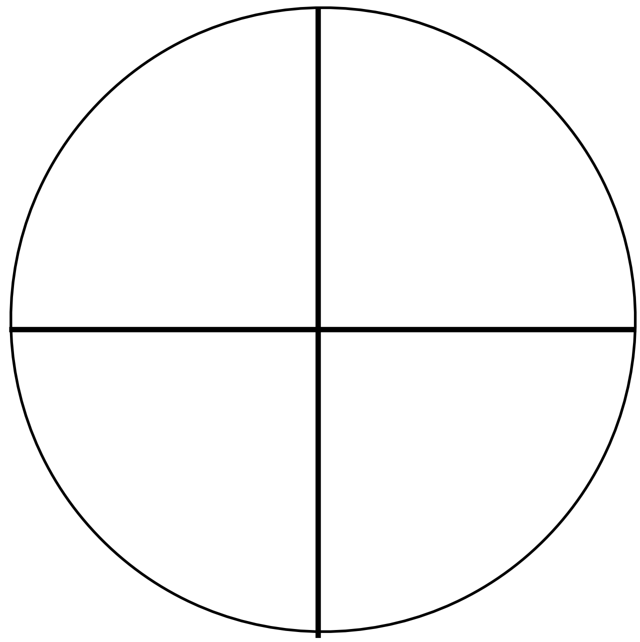
$19 = (1)12 + 7$

$a \% m = 7 = b \% m$

We say that a is congruent to b modulo m written as:

$a \equiv b \pmod{m}$

Integers modulo 4.



Definition:
$a \equiv b \pmod{m}$ if a % m = b % m.

An equivalent definition is:

Definition:
$a \equiv b \pmod{m}$ if $m \mid (a-b)$.

To show that the two definitions are equivalent we need to show that:

if a % m = b % m **then** m | (a-b)

and

if m | (a-b) **then** a % m = b % m

if a % m = b % m **then** m | (a-b)

Say a % m = b % m = r. Then we can write:

a = pm + r **and** b = qm + r

where p and q are integers.

we have:

(1) (a-b) = pm + r - qm - r = m(p-q)

and equation (1) implies (a-b) = m(p-q) so m | (a-b).

On the other hand:

if m | (a-b) **then** a % m = b % m

The proof of this proposition is a bit involved so I will omit it.

Example: Let m = 12. Then we have:

$13 \equiv 1 \pmod{12}$

$17 \equiv 5 \pmod{12}$

Which is familiar to everyone who uses a 24 hour clock.

And we can also have:

$241 \equiv 1 \pmod{12}$

$166 \equiv 10 \pmod{12}$

$120 \equiv 0 \pmod{12}$

Similarly

$90 \equiv 30 \pmod{60}$

$75 \equiv 15 \pmod{60}$

$120 \equiv 0 \pmod{60}$

We now show that congruence is an equivalence relation.

Theorem: Let m be a positive integer then

1. For any integer a we have $a \equiv a \pmod{m}$ (reflexive)
2. if $a \equiv b \pmod{m}$ then $b \equiv a \pmod{m}$ (symmetric)
3. if $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$
   then $a \equiv c \pmod{m}$ (transitive)

I will prove 3.

Theorem: if a ≡ b (mod m) and b ≡ c (mod m) then a ≡ c (mod m).

Proof: if a ≡ b (mod m) then m | (a-b), and if b ≡ c (mod m) then m | (b-c).

And by one of the divisibility theorems we have:

m | (a-b+b-c) or, m | (a-c) so a ≡ c (mod m). □