CISC-102
Winter 2016
Lecture 11

Greatest Common Divisor

Consider any two integers, a,b, at least one non-zero. If we list the positive divisors in numeric order from smallest to largest, we would get two lists:

a: (1, c1, c2, ... |a|)
b: (1, d1, d2, ... |b|)

Since both lists must contain the number 1, we see that 1 is a common divisor of a and b. Since the greatest divisor of a is |a| and the greatest divisor of b is |b|, we can deduce that amongst the common divisors of a and b, there must be one that is the greatest.

Thus we can say that given two integers a,b, at least one not zero, there is a unique greatest common divisor of a and b.

Computing the greatest common divisor of a non-zero integer a, and 0, is somewhat boring because all non-zero integers divide 0, so the greatest common divisor of a and 0 is always |a|. So let's just assume from now on that neither a nor b is 0.

Example:
Let a = 111, and b = 250. We can construct sorted lists of divisors of a and b yielding:

a: (1, 3, 37, 111)
b: (1, 2, 5, 10, 25, 50, 125, 250)

And by inspection we can deduce that 1 is the greatest common divisor of a and b. When the greatest common divisor of two numbers a,b is 1 we say that a and b are *relatively prime* or *coprime*.

Another example:
Let a = 250, and b = 575. We can construct sorted lists of divisors of a and b yielding:

a: (1, 2, 5, 10, 25, 50, 125, 250)
b:(1, 5, 23, 25, 115, 575)

And by inspection we can deduce that 25 is the greatest common divisor of a and b.

This method of obtaining all divisors of a and b is very computationally intensive, and would make some essential steps of public key encryption schemes un-feasible. Remarkably an algorithm invented by Euclid (~ 300 BC) finds greatest common divisors in a much more efficient way.

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

gcd (a,b) = gcd(b,r)

For example: a = 575, b = 250.
(Note: We already know that gcd(575,250) = 25)

575 = (2)(250) + 75 (Use long division to get q & r)

So the claim is that gcd(575, 250) = gcd(250,75).

Since we already know that gcd(575,250) = 25 this can be verified by listing the divisors of 250 and 75.

250: (1, 2, 5, 10, 25, 50, 125, 250)
75: (1, 3, 5, 15, 25, 75)

We can now “iterate” this process by renaming a = 250, b = 75 and repeat the previous calculation. That is:

250 = (3)(75) + 25 (Use long division to get q & r)

We can again verify that gcd(250,75) = gcd(75,25) by listing the divisors of 75 and 25.

75: (1, 3, 5, 15, 25, 75)
25: (1, 5, 25)

Let’s repeat this again, so a = 75 and b = 25

75 = (3)(25) + 0

so we have gcd(75,25) = gcd(25,0), and we have already seen that the greatest common divisor of any non-zero integer a and 0 is |a|.

Therefore by Euclid’s algorithm we have
gcd(575,250) = 25.

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.

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) 575 = (2) 250 + 75 implies 75 = 575 - (2)250
(2) 250 = (3) 75 + 25 implies 25 = 250 - (3)75
(3) 75 = (3) 25 + 0

Now we can write gcd(575,250) = 25 as:

25 = 250 - (3)75 (Using (2) above)
25 = 250 - (3)[575 - (2)250] (Using (1) above)
25 = (7)250 - (3)575 (Simplify)

To prove Euclid's Theorem we will need a preliminary result. Math convention uses the word "lemma" for preliminary results that are proved in preparation for the proof of the main theorem.

Lemma: Let g,a,b be non-zero integers. If g | a and g | b then g | (pa + b) for all integers p.

Proof: Since g | a and g | b we can write

(1) a = cg and b = dg, where c,d are integers.

Replacing the values of a and b in g | (pa + b) using equation (1) we get:

$$g \mid (pcg + dg)$$

which simplifies to:

$$g \mid g(pc + d)$$

Now it should be clear that g divides g(pc+d) and thus we conclude that g divides pa + b. □

Theorem: Let a,b,q,r be integers such that:
$a = qb + r$, $0 \leq r < |b|$, then gcd (a,b) = gcd(b,r).

Proof:
(0) Let g1= gcd(a,b) and g2 = gcd(b,r).

(1) Observe that g2 | b and g2 | r, so g2 | pb + r for all integers p, and in particular for q, where a = qb + r.

(a) Therefore, g2 | a, and we have established that g2 is a common divisor of both a and b.

(b) Furthermore, observe that $g2 \leq g1 = \gcd(a,b)$

(2) Using the equation a = qb + r we can write
r = - qb + a.
g1 | b and g1 | a so use the lemma (with p = -q) to get g1 | -qb + a or g1 | r.

(a) Therefor g1 | r and we have established that g1 is a common divisor of b and r.

(b) Furthermore, observe that $g1 \leq g2 = \gcd(b,r)$.

(3) $g2 \leq g1$ and $g1 \leq g2$ implies that g1 = g2, so we can conclude that gcd(a,b) = gcd(b,r). □

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.

We will argue that euclid_gcd(a,b) finds gcd(a,b) assuming that $a \geq b > 0$.

We first argue that the loop terminates, that is r eventually becomes 0. This is easy to see because the remainder when we divide a by b is less than b. The value of r begins positive and always decreases so it eventually must be zero.

The correctness follows from Euclid's theorem.

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