Number Theory 1: GCD, Linear Combinations, Inverses (Revised 1/27/2014)

Quick Review

Here are the topics on number theory that were covered in Math/CS11 and that you need to be familiar with (if not, review it!) in order to follow the lectures on number theory:

1. Prime and composite numbers (why there are infinitely many primes?)
2. Factorization, uniqueness (the fundamental theorem of arithmetic)
3. Relatively prime numbers
4. Greatest common divisor, least common multiple
5. Basic modular arithmetic and congruences

Gcd and Euclid's Algorithm

Given two numbers a, b we want to compute their greatest common divisor $c = \gcd(a, b)$. This can be done using Euclid's algorithm, that is based on the following easy-to-prove theorem.

Theorem 1 *Let* $a > b$. *Then* $\gcd(a, b) = \gcd(a - b, b)$.

Proof: The theorem follows from the following claim: x is a common divisor of a, b if and only if x is a common divisor of $a - b, b$. To prove the claim, we show each implication separately.

($\Rightarrow$) Suppose that x is a common divisor of a and b. Then $a = \alpha x$ and $b = \beta x$ for some integers α, β. Therefore $a - b = \alpha x - \beta x = (\alpha - \beta)x$, so x is a divisor of $a - b$.

($\Leftarrow$) Suppose now that x is a common divisor of $a - b$ and b. THen $a - b = \gamma x$ and $b = \delta x$ for some integers γ, δ. This implies that $a = (a - b) + b = \gamma x + \delta x = (\gamma + \delta)x$, so x is a divisor of a. □

Euclid's Algorithm. Euclid's Algorithm computes the greatest common divisor of two positive integers, and it can be written in a recursive form as follows.

```
function gcd($a, b$)
    if $a = b$ then return $a$
    if $a < b$ then swap($a, b$)
    return gcd($a - b, b$)
```

Example. Suppose we want to compute $\gcd(1034, 222)$. The algorithm will produce the following pairs of numbers: $1034, 222$; $810, 222$; $588, 222$; $366, 222$; $222, 144$; $144, 78$; $78, 66$; $66, 12$; $54, 12$; $42, 12$; $30, 12$; $24, 12$; $12, 12$. So the algorithm will return 12.

Faster version of Euclid's Algorithm . As you can see from the example below, if a is much bigger than b then the algorithm will replace a by $a - b$, $a - 2b$, $a - 3b$, ... For large numbers, say $a = 10^{100}$ and $b = 13$ then this process would take forever. But that's easy to get around: the final result of these subtractions will be $a \,\mathrm{rem}\, b$, so instead of repeated multiplications we can replace a by $a \,\mathrm{rem}\, b$ in one step. (This will require to change the base case $a = b$ by $b = 0$ – a minor detail.) This new algorithm is very fast – it runs in time $O(\log a + \log b)$, that is the number of iterations does not exceed the total number of bits in a and b.

Gcd as a Linear Combination

Theorem 2 *If a, b are positive integers then there exist integers α, β such that* $\gcd(a, b) = \alpha a + \beta b$.

Proof: To prove this theorem we modify Euclid's Algorithm so that it actually computes these numbers α, β, as follows:

```
function gcd($a, b$)
    if $a = b$ then return $(a, 0, 1)$
    if $a > b$ then
        $(c, \alpha', \beta') = \gcd(a - b, b)$
        return $(c, \alpha', \beta' - \alpha')$
    else
        $(c, \alpha', \beta') = \gcd(b - a, a)$
        return $(c, \beta' - \alpha', \alpha')$
```

This is sometimes called the *Extended Euclid's Algorithm.* It remains to prove that it's correct.

The proof is by induction by the number of steps of the algorithm. The base case is when there are no recursive calls, that is when $a = b$. In this case the algorithm returns $\alpha = 0$ and $\beta = 1$. Since $0 \cdot a + 1 \cdot b = b = \gcd(a, b)$, the computed values are correct.

Suppose now that the we make some number n of recursive calls in the algorithm. If the inputs are a, b, where $a > b$ the algorithm will compute (c, α', β'), which by the induction assumptions are correct, that is $\gcd(a - b, b) = c$ and $\alpha' \cdot (a - b) + \beta' \cdot b = c$. The algorithm will then return $\alpha = \alpha'$ and $\beta = \beta' - \alpha'$. But then $\gcd(a, b) = c$ as well and $c = \alpha' \cdot (a - b) + \beta' \cdot b = \alpha \cdot a + \beta \cdot b$, so α and β are computed correctly. The case when $a < b$ is similar. □

In fact, the faster version of Euclid's Algorithm can be modified in a similar way to compute the coefficients α and β. Therefore these coefficients can be computed in polynomial time. (This is important for what we will do later.)

In hand calculations we will typically not use Euclid's Algorithm to find the values of α and β. Instead, for small numbers a, b, one can simply list all multiples a and b, until finding two whose difference is $c = \gcd(a, b)$.

Example. Consider $a = 22$ and $b = 32$. Then $\gcd(a, b) = 2$, so we want to find their multiples that differ by 2. The list of their multiples is

$$\begin{array}{cccc} 0 & 22 & 44 & 66 \\ 0 & 32 & 64 & \end{array}$$

so we get $3 \cdot 22 - 2 \cdot 32 = 2$. Thus $\alpha = 3$ and $\beta = -2$.

Inverses Modulo a Prime

As discussed in CS11 and earlier in this class, for some number n we can consider numbers $0, 1, ..., n-1$ as a small "universe" and redefine the operations of addition, subtraction and multiplication by taking the result modulo n, so that the resulting value is in the set $0, 1, ..., n-1$. This way this set is closed under these operations.

We can also compute powers. For example, what is $3^{44208} \operatorname{rem} 8$? We can write $3^{44208} = (3^2)^{22104} = 9^{22104}$, so $3^{44208} \operatorname{rem} 8 = 9^{22104} \operatorname{rem} 8 = 1^{22104} \operatorname{rem} 8 = 1$. As this example illustrates, when you do computation in a modular arithmetic, you can compute the remainders at each step, rather than computing the complete value first and then taking the remainder.

We would like to extend this even further. In real numbers, we can also divide x/y. Can we do the division in our modular arithmetic? As it turns out, the answer is yes if n is a prime. (This can be generalized to all numbers n, but we will not cover it in this class.)

So let us consider a prime p and numbers $1, 2, ..., p-1$. We do not include 0 because division by 0 is not allowed, even for real numbers. In real arithmetic, $x/y = x \cdot y^{-1}$, and we could do the same for modular arithmetic, but then we need to figure out what is y^{-1}, the inverse of y. This is easy: Define $y^{-1} \pmod{p} = z$, for $z \in \{1, 2, ..., p-1\}$, iff $yz = 1 \pmod{p}$. We now show that this inverse exists and is uniquely defined.

Theorem 3 *Let p be a prime and $y \in \{1, 2, ..., p-1\}$. Then the inverse of y modulo p exists and is unique (among numbers between 1 and $p-1$.)*

Proof: We show existence first. Since y and p are relatively prime, there are α, β such that $\alpha y + \beta p = 1$. Taking the remainder modulo p of both sides, we get $\alpha y = 1 \pmod{p}$. So α is the inverse of y. This is almost right, but not exactly, because α may not be in the range $1, 2, ..., p-1$. If so, replace α by $\alpha' + dp$, where d is chosen so that $\alpha' \in \{1, 2, ..., p-1\}$.

To show uniqueness, towards contradiction, suppose there are two different inverses z, z' of y between 1 and $p-1$. So $zy = 1 \pmod{p}$ and $z'y = 1 \pmod{p}$. Thus $y(z - z') = 0 \pmod{p}$, and since $\gcd(y, p) = 1$, we conclude that $z = z' \pmod{p}$, so z, z' must be equal. □

Example. Find $10^{-1} \pmod{13}$. We first find α and β for which $\alpha \cdot 10 + \beta \cdot 13 = 1$. We list the multiples of 10 and 13:

0	10	20	30	40
0	13	26	39	

so we get $4 \cdot 10 - 3 \cdot 13 = 1$. Thus $4 \cdot 10 = 1 \pmod{13}$, so $10^{-1} = 4 \pmod{13}$.

Solving Linear Congruences Modulo a Prime

A linear congruence has the form

$$ax \quad \equiv \quad b \pmod{p}.$$

We will conisder only the case when p is a prime number (for now). It's essentially an equation that we want to solve for x (where a, b are given.) Is it always possible? If so, is the solution unique?

Example: Let's say we want to solve $7x \equiv 5 \pmod{9}$. Determine first the inverse of 7 modulo 9, that is, we want to find z for which $7z \equiv 1 \pmod{9}$. This is equivalent to finding z and s for which $7z = 9s + 1$. So we search for a multiple of 7 among the numbers of the form $9s + 1$: 10, 19, 28, 37, ... The first number in this sequence that will work is 28, since it's a multiple of 7. Since $7 \cdot 4 = 28$, we get $z = 4$. Indeed, $7 \cdot 4 \equiv 1 \pmod{9}$.

Now, to solve the original congruence, multiply it by 4, and we get $x = 20 \pmod{9} = 2$.