EMAT10001 Lecture 4.

Conor Houghton 2013-10-16

Preface

These are outline notes for lecture 4; they are based on *Number Theory with Computer Applications* by Ramanujachary Kumanduri and Cristina Romero. This is an excellent book, but the material can be found in many number theory and discrete mathematics books. As usual there is a bounty of between 20p and £2 for errors, you can tell me at the end of a lecture or email me at `conor.houghton@bristol.ac.uk`. A manicule (☜ or ☞) is used to indicate that a proof, derivation or piece of material has been omitted from the lecture but will be covered in the workshop.

Introduction

There are two types of mathematicians, those who think of themselves as discovering mathematics and those who think of themselves as creating it. It is the sort of mathematics done by the second type of mathematicians that we will look at over the next few lectures, in particular, we will look at number theory and group theory. Our aim is three fold:

- The mathematics we will look at is the basis of cryptography and is useful in a number of areas in computer science.
- The concepts we will see occur, often in a more complicated way, across all sorts of mathematical constructions.
- This area has some nice proofs, which will let us practise proving things before learning about it more precisely from Kerstin.

In the case of Boolean algebra we saw the *and* table:

$\wedge$	0	1
0	0	0
1	0	1

which a lot like a multiplication table and the *xor* table:

$\oplus$	0	1
0	0	1
1	1	0

which looks a bit like an addition table, except $1 + 1 = 2$ whereas $1 \oplus 1 = 0$. In fact, this could also be thought of as modular addition. Here we are going to look at modular arithmetic and then at group theory, which answers the question of when something 'looks a bit like an addition table'. Modular arithmetic is the basis of the RSA encryption algorithm, it gives examples that are useful for studying group theory, it is a good example for looking at equivalence relations and it's a fun piece of mathematics.

Remainders

We write

$$r = a \bmod b \tag{1}$$

to mean that r is the remainder when a is divided by b. This is ambiguous, a non-zero r could be positive or negative; it is normal to take r non-negative. This is the same as saying

$$a = mb + r \tag{2}$$

for some m an integer, with $0 \leq r < b$. Hence

$$4 = 104 \bmod 50 \tag{3}$$

because $104 = 2 \cdot 50 + 4$.[1] ☞ It is easy to check that

$$[a + b \bmod c] = [[a \bmod c] + [b \bmod c] \bmod c] \tag{4}$$

and

$$[ab \bmod c] = [[a \bmod c][b \bmod c] \bmod c] \tag{5}$$

which helps when calculating remainders for large numbers, for example, for numbers whose product might exceed the capacity of an `int`.

We should also note that there is a theorem called The Division Theorem that shows that this is all well defined, that is, given integers a and b with $b \neq 0$ there are unique integers m and r such that $a = bm + r$ and $0 \leq r < |b|$. This isn't a hard theorem but we are leaving it out for brevity.

If $0 = a \bmod b$ then we say b *divides* a and write $b|a$. Obviously, this means $a = mb$ for some m. Hence $8|24$ and $3|27$ but $8 \not|27$. Notice also that $a|0$ since $0 = 0a$. ☞It is useful to list some basic properties as a lemma.

Lemma. If a, b, c, x and y are positive integers

1. If $a|b$ and $x|y$ then $ax|by$.
2. If $a|b$ and $b|c$ then $a|c$.
3. If $a|b$ and $b \neq 0$ then $a \leq b$.

[1] There are two simple Python programs on the website for you to practise modular arithmetic.

4. If $a|b$ and $a|c$ then $a|(bx+cy)$.

As you probably know if $p>1$ has only one and itself as divisors then p is a *prime* number. Thus, seven is a prime, eight is not. Perhaps the most important thing about primes is that the set of integers is a *unique factorization domain*, which means that every number can be written as

$$n = p_1 p_2 \dots p_r \tag{6}$$

where p_1, p_2 and so on up to p_r are primes and up to the order you write the primes in, there is only one way to do this. In this way

$$18 = 2 \cdot 3 \cdot 3 = 2 \cdot 3^2. \tag{7}$$

Greatest Common Divisor

Another important concept is the greatest common divisor of two numbers a and b, written $\gcd(a,b)$ or just (a,b).[2] A *common divisor* of a and b is a number d which divides both, that is $d|a$ and $d|b$; the *greatest common divisor* is the largest such number. Hence two is a common divisor of six and 24 but

$$(6,24) = 6 \tag{8}$$

If $(a,b)=1$ we say a and b are *co-prime.*

There is an obvious, but slow way to find the greatest common divisor, take for example 18 and 24. We know

$$\begin{aligned} 18 &= 2\cdot 3\cdot 3 \\ 24 &= 2\cdot 2\cdot 2\cdot 3 \end{aligned} \tag{9}$$

so we look at the prime factors in common, one two and one three, so $(18,24)=6$. There is a much quicker way, the Euclid algorithm, which we will return to shortly, first though we want to prove a slightly surprising property of the greatest common divisor; for any a and b there are integers m and n such that

$$(a,b) = ma + nb \tag{10}$$

This works for all integers a and b but for simplicity we assume a and b are non-negative. ☞We first need a simple lemma.

Lemma. $(a,b) = (a-b,b)$.

Now the theorem itself.

[2] Because it is such a straight-forward notation, (a,b) is used to mean very different things in different areas of mathematics, so be careful with it.

Theorem. For any non-negative integers a and b there are integers m and n such that $(a,b)=ma+nb$.
Proof: Write $d=(a,b)$. It is clear that $d|(ma+nb)$ for any m and n, the problem is showing that there is an m and n such that $d=ma+nb$. Consider the set of all linear combinations

$$S = \{ma + nb | m, n \in \mathbf{Z}\} \tag{11}$$

Let c be the smallest positive integer in S, so $c>0$ and, if $e\in S$ and $e>0$ then $e\geq c$. Since $c\in S$ there are m_1 and n_1 such that

$$c = m_1 a + n_1 b \tag{12}$$

Now by choosing $m=1$ and $n=0$ or the other way around we can see $c\leq a$ and $c\leq b$. Now from the division theorem we know

$$a = qc + r \tag{13}$$

with $0\leq r<c$ for some q. Turning this around

$$r = a - qc = a - q(m_1 a + n_1 b) = (1 - q m_1)a - q n_1 b \tag{14}$$

so since this in the form $ma+nb$, with $m=1-qm_1$ and $n=-qn_1$, $r\in S$. If $r\neq 0$ then this r is smaller than S, contradicting the choice of c as the smallest non-negative element in S. Thus r must be zero, and hence $a=qc$ so $c|a$. The same method can be used to show $c|b$ and hence c is a divisor of a and b. Since $c\in S$ we know $d|c$ and since d is the greatest divisor, this means $d=c$. □

Related to the greatest common divisor is the *Euler Totient function*:

$$\phi(a) = |\{b < a | (a,b) = 1\}| \tag{15}$$

so $\phi(a)$ is the number of numbers less than a and co-prime with it.☜

Euclid algorithm

The Euclid algorithm is an efficient algorithm for finding the greatest common divisor of two numbers, it also allows us to find the m and n in the equation

$$(a,b) = ma + nb \tag{16}$$

for some m and n. It is perhaps surprising that it is far easier to find the greatest common divisor than to factorize the numbers involved. This is part of what makes public key cryptography work. The algorithm is

Algorithm. For positive integers a and $b\neq 0$

1. Set $x=a$ and $y=b$.

2. If $y = 0$ then x is the answer.

3. Set $r = x \bmod y$ and then let $x = y$ and $y = r$ and return to 2.

The algorithm works because $(a, b) = (a - b, b)$, the step $r = x$y is replacing x with $x - my$ for some y. It also swaps using $(a, b) = (b, a)$ so that the numbers keep getting lower.

As an example consider finding (56,106).

$$\begin{aligned} 50 &= 106 \bmod 56 \\ 6 &= 56 \bmod 50 \\ 2 &= 50 \bmod 6 \\ 0 &= 6 \bmod 2 \end{aligned} \tag{17}$$

so $(56, 106) = 2$. Further, working backwards through the modular divisions, from the first non-zero modular equation

$$2 = 50 - 8 * 6 \tag{18}$$

Now the next one tells us that $6 = 56 - 50$, so substituting that back in gives

$$2 = 50 - 8(56 - 50) = 9 \cdot 50 - 8 \cdot 56 \tag{19}$$

and $50 = 106 - 56$ gives

$$2 = 9 \cdot (106 - 56) - 8 \cdot 56 = 9 \cdot 106 - 17 \cdot 56 \tag{20}$$

Modular arithmetic

The idea behind modular arithmetic is to use the idea of remainders and so on and to turn it into an actual arithmetic system. It relies on *congruence*. If a, b and c are integers, we say that a *is congruent to* b *modulo* m if $m|(a - b)$. This is written $a \equiv b \pmod{m}$. There are different ways of thinking about this, we could say a is the same as b up to a multiple of m, or we could use the language of remainders we developed above and say $a \equiv b \pmod{m}$ is $(a \bmod m) = (b \bmod m)$. Note the way the use of mod differs, if $r = a \bmod m$ then r is the remainder and $0 \leq r < b$ whereas if $a \equiv b \pmod{m}$ then a and b are conjugate, neither has to be less than m.

Congruence is an example of an *equivalence relation*, an equivalence relation is any relationship between pairs of elements in a set that satisfies reflexivity, symmetry and transitivity. So, for a set X the relationship $\sim$ is an equivalence relationship if

1. Reflexivity: if $x \in X$ then $x \sim x$.

2. Symmetry: if x and y are in X and $x \sim y$ then $y \sim x$.

3. Transitivity: if x, y and z are in X and $x \sim y$ and $y \sim z$ then $x \sim z$.

Verifying that congruence is an equivalence relation is just a matter of checking each of those three properties.☜