Notes 4

7 Natural Numbers and Integers

Having introduced the basic ideas of proofs, especially induction, we now turn to introducing the basic discrete structures used in computer science.

These are:
Natural numbers and integers.
Sets, relations, functions.
Equivalence relations, posets, trees.
Graphs.
State machines.

Then on to analysis techniques (counting, recurrences, etc.).

Natural numbers are $0, 1, \cdots$.
Integers also include negatives.
Operations are usual arithmetic operations $+$, $\times$, etc.
Ordering is the usual order.

We will review some basic concepts: divisibility, primes, composite numbers, modular arithmetic, etc.
Give basic theorems about these notions,
and some simple algorithims based on these theorems.

Fundamental theorem of arithmetic – every number has unique factorization into primes.
Algorithm for finding GCD, the greatest common divisor of two numbers.
RSA algorithm for encrypting/decrypting data.

Subject matter usually called *number theory*.
Used to be just for fun, but recently has become very important in practice, mainly because of applications to security.

7.1 Divisibility and Primes

Definition 7.1 If a and b are integers, $a \neq 0$, then a *divides* b (written $a \mid b$) if $(\exists c)$, c an integer, such that $b = ac$.
We say *a goes into* b (in elementary school), or b is *divisible* by a.

Makes sense even for negative integers, but we'll mainly think of positive integers.

Definition 7.2 If a is a natural number ≥ 2, then a is *prime* if the only natural numbers dividing a are 1 and a.

Definition 7.3 If a is a natural number ≥ 2, then a is *composite* if $a = bc$ for natural numbers $b, c \neq 1, a$.

Basic divisibility properties:

Theorem 7.4
1. If $a \mid b$ and $a \mid c$, then $a \mid (b + c)$ and $a \mid (b - c)$.
2. If $a \mid b$ then $a \mid bc$ for any c.
3. If $a \mid b$ and $b \mid c$ then $a \mid c$.

Proof. (Of part 1) By definition of divisibility, $(\exists t) b = at$ and $(\exists s) c = as$.
Therefore, $b + c = at + as = a(t + s)$.
Therefore, by definition, $a \mid b + c$.

Parts 2 and 3 are similar (use 2 to prove 3). ■

In lecture 3 and last Friday's recitation, already did some results about these notions – e.g., the division algorithm, and half of the Fundamental Theorem of Arithmetic:

Theorem 7.5 (Fundamental Theorem of Arithmetic)
Every natural number ≥ 1 has a factorization into primes.
Moreover, the factorization is unique (except for reordering the factors)

We proved existence using strong induction.
Later today, we will also prove uniqueness.

One other useful def:

Definition 7.6 If x and y are integers, where $y > 0$, then $x \bmod y$ is the remainder on division of x by y.

$x \bmod y$ is well defined by the division algorithm.
$0 \leq x \bmod y < y$.

7.2 GCDs and LCMs

Remember Die Hard?
Why did 5 and 3 buckets work but 6 and 3 not?
What if buckets are 21 and 133, and goal is 13 gallons?

Definition 7.7 Let a, b be integers, not both 0. The *greatest common divisor* of a and b, written $\text{GCD}(a, b)$, is the largest natural number g such that $g \mid a$ and $g \mid b$.

We can prove GCD is well defined:
Can use well-ordering to prove that any nonempty finite set has a *largest* element.
The set of common divisors includes 1, and is finite because it is bounded above by either number.

Note GCD is always positive.

GCD of 24 and 36? 21 and 133?

It turns out that in Die Hard, buckets of size a, b always produce numbers divisible by $\text{GCD}(a, b)$.
Q: Can all such numbers in range $0, \ldots, \max(a, b)$ be produced?

Here is an interesting theorem that shows that a GCD is always expressible as a sum.
This theorem will be used for many purposes, e.g., to help in proving the uniqueness part of the Fundamental Theorem of Arithmetic.

Theorem 7.8 Let x and y be integers, not both 0.
Then $\text{GCD}(x, y)$ can be expressed in the form $ax + by$ for some integers a and b.

Examples: $x = 22$, $y = 16$
Then GCD is 2, express as $2 = 3(22) + (-4)(16)$. Note can use negative integers.

$x = -22$, $y = 16$
Again, GCD is 2, express as $2 = (-3)(-22) + (-4)(16)$. (The GCD is always nonnegative.)

Proof. Assume wlog that $x, y \geq 0$. (Otherwise, use negative coefficients as above.)
If either is 0, then the GCD is defined to be the other – unwind the definition, which is certainly expressible in this way.
So we're left with the case where x and y are both positive.

Let S be the set of positive integers expressible in the form $ax + by$, where a and b are

integers.
Fix l to be the smallest element in S – exists by well-ordering, S is nonempty since $x \in S$.
Fix a and b such that $l = ax + by$.

As an example, let $x = 22$, and $y = 16$. S includes all numbers $ax + by$, for all integers x and y. It includes:

$$\begin{aligned} 0x + 1y &= 16 \\ 1x + (-1)y &= 16 \\ (-1)x + 2y &= 10 \\ 2x + (-2)y &= 12 \\ 3x + (-4)y &= 2 \end{aligned}$$

and many others. $l = 2$ turns out to be the smallest element in S. We claim l is what we want, *i.e.*, $\text{GCD}(x, y)$. We need to show:

1. l is a common divisor of x and y.

2. Any common divisor of x and y also divides l (so l is the greatest – if a number c goes into a positive number l, then $c \leq l$).

1. l is a common divisor of x and y.

   First show $l \mid x$.
   Write $x = ql + r$ by division algorithm, where $0 \leq r < l$.
   Then rewrite:

   $$\begin{aligned} r &= x - ql \\ &= x - q(ax + by) \\ &= (1 - qa)x + (-qb)y \end{aligned}$$

   So r is also expressible in the sum form.
   We know $0 \leq r < l$.
   If r were positive, it would be in S, and would be smaller than l, which would contradict our choice of l as the smallest element of S.
   So it must be that $r = 0$.
   So $x = ql$, which says that $l \mid x$.

   Similarly, $l \mid y$.

2. Any common divisor of x and y also divides l.

   Easy: $c \mid x$ and $c \mid y$ imply $c \mid ax + by = l$.

■

The symmetric concept to GCD is LCM:

Definition 7.9 Let a, b be positive integers. The *least common multiple* of a and b, written $\mathrm{LCM}(a, b)$, is the smallest natural number l such that $a \mid l$ and $b \mid l$.

7.3 Fundamental Theorem of Arithmetic

With the addition characterization we just proved for the GCD, we are finally ready to prove the Fundamental Theorem.

Recall statement. Already proved existence. So remains to prove uniqueness.
For this, use basic number theory lemmas:

Lemma 7.10 Let p be a prime, and a and b any integers. If $p \mid ab$, then $p \mid a$ or $p \mid b$.

Proof. If $p \mid a$, we are done, so assume not.
Then $\mathrm{GCD}(p, a) = 1$.
(Because the only positive divisors of p are p and 1, and p doesn't divide a.)
So by the addition form above, can write 1 as a sum of the form $1 = sa + tp$, where s and t are integers.
Multiplying by b:
$b = sab + tpb$.
We have $p \mid sab$, since $p \mid ab$.
Also, $p \mid tpb$.
So $p \mid b$ (by Theorem 7.4). ■

Corollary 7.11 Let p be a prime, and $a_1, \cdots, a_k$ any integers.
If $p \mid a_1 a_2 \cdots a_k$, then $(\exists i) p \mid a_i$.

Now prove uniqueness, in Fundamental Theorem:

Proof. (Uniqueness of prime decomposition) We argue by (strong) induction. The base case $n = 1$ is trivial – the product of no factors is defined to be 1.
INDUCTIVE STEP:
Suppose the claim true for all numbers less than n.
Now show for n.
Suppose for contradiction that there are two different factorizations, $n = p_1 p_2 \cdots p_k$ and $n = q_1 q_2 \cdots q_l$.

Assume that the primes in each factorization are written in nondecreasing order.
Then $p_1 \mid q_1 \cdots q_l$.
By corollary, $(\exists i) p_1 \mid q_i$.
But since q_i is prime, $p_1 = q_i$.
Divide both sides, and get two factorizations of n/p_1.
This contradicts our inductive hypothesis, so the theorem is proved. ■

Unique factorization gives a method for computing GCD:
Factor both numbers, and then multiply out all their common factors (watch multiplicities).

Example: 100 vs. 120:
$100 = 2 \times 2 \times 5 \times 5$
$120 = 2 \times 2 \times 2 \times 3 \times 5$
Common factors (including multiplicities): 2, 2, 5.
Product $2 \times 2 \times 5 = 20$ is GCD.

Taking min number of occurrences of each prime.
Likewise, unique factorization gives a method for computing LCM: Factor both, then multiply the max number of occurrences of each prime:

Example: 100 vs. 120:
Product $2 \times 2 \times 2 \times 3 \times 5 \times 5 = 600$ is LCM.

These observations lead to the following nice connection between GCD and LCM:

Theorem 7.12 Let a, b be positive integers. Then $ab = \mathrm{GCD}(a, b) \times \mathrm{LCM}(a, b)$

Example: 100 vs. 120:
$100 \times 120 = 20 \times 600$
$2^2 \cdot 5^2 \times 2^3 \cdot 3 \cdot 5 = 2^2 \cdot 5 \times 2^3 \cdot 3 \cdot 5^2$

Proof. For each distinct prime in the prime factorizations, the LHS uses the sum of occurrences in a and b,
while the RHS uses the min plus the max occurrences.
Same (sum of two numbers is the same as their min plus their max). ■

7.4 Euclid's GCD Algorithm

Unfortunately, the "algorithms" given above to find GCD's and LCM's – factor and multiply some of the factors – don't work in practice!
Why?
Factoring is expensive.

No one knows how to do it very well – exponential behavior.

In this section, we present an algorithm for GCD (and so LCM) that *is* practical.
This doesn't help us in factoring, though.

Euclid
One of first algorithms.
Algorithm based on division.

Do correctness proof – not obvious.
Augment to show how to get the coefficients in the sum form of the GCD

7.4.1 The Algorithm

Procedure Euclid(x, y), x and y nonnegative integers, not both 0, $x \geq y$:

1. If $y = 0$, then return x.
2. Otherwise, return Euclid$(y, x \bmod y)$.

Example execution: Euclid$(22, 16)$
Call Euclid$(16, 6)$.
Call Euclid$(6, 4)$.
Call Euclid$(4, 2)$.
Call Euclid$(2, 0)$.
Return 2.

7.4.2 Correctness Proof

Two properties to show:

- If it ever terminates, it gives the right answer (partial correctness).
- It eventually terminates.

Both of these are normally part of correctness proofs (for standard sequential algorithms).

Partial correctness Why does this give the correct answer, if it terminates?
We show that each step "preserves" the GCD, which implies (by simple induction) that the whole algorithm does.

Lemma 7.13 If x, y are any integers, $y \neq 0$, then $\text{GCD}(x, y) = \text{GCD}(y, x \bmod y,)$.

Proof. We show that any nonzero integer c is a common divisor of x and y if and only if it's a common divisor of y and $(x \bmod y)$.

Write $x = qy + x \bmod y$, using the division algorithm.

Suppose c is a common divisor of x and y: $c \mid x$ and $c \mid y$.
Then also $c \mid x - qy = x \bmod y$, by Theorem 7.4.
So c is a common divisor of y and $x \bmod y$.

On the other hand, suppose that c is a common divisor of y and $x \bmod y$ – $c \mid y$ and $c \mid x \bmod y$.
Then $c \mid qy + x \bmod y = x$ by Theorem 7.4.
So c is a common divisor of x and y. ∎

Since each call reduces the problem to one with the same GCD, the algorithm, *if it terminates*, produces the correct GCD for the original arguments.

Theorem 7.14 If $\text{Euclid}(x, y)$ terminates, then its output is $\text{GCD}(x, y)$.

Proof. By induction on the number of passes. ∎

Termination

Theorem 7.15 Euclid's algorithm terminates after at most $x + y$ passes.

Proof. Strong induction on the sum of the two arguments, $x + y$.
BASE: Sum is 1.
Then one argument is 0, the other 1.
Terminates in 1 step, with 1 as output.
INDUCTIVE STEP:
Suppose terminates whenever sum of arguments is $< n$, and consider $\text{Euclid}(x, y)$ where $x + y = n$.
Must argue terminates in at most n passes.
If terminates in 1 pass, we're done, since $1 \leq n$.
If not, then calls $\text{Euclid}(y, x \mod y)$.

The sum of the arguments in the call is $y + (x \mod y)$,
which we claim is strictly less than $n = x + y$, *i.e.*, is $\leq n - 1$.

If we show this, then we can apply strong induction to show that the number of passes remaining is at most $n-1$, which adds to the present pass to yield the result.

So we just need to show $y + (x \mod y) < x + y$.
Or, $x \mod y < x$.
True? Yes since $x \geq y$. ■

Example 1 We show $\text{GCD}(949, 273) = 13$ as follows:

```
quotient    x      y
           949    273
   3       273    130
   2       130     13
  10        13      0
```

7.4.3 Finding GCD coefficients

The previous algorithm finds the GCD.
It can be modified to give the coefficients in the sum expression.

Procedure Euclid-Coefficients(x, y), x and y not both 0, $x \geq y$

1. If $y = 0$, return $(1, 0)$.
2. Else $x = qy + r$
3. Use Euclid-Coefficients(y, r) to find (a, b) such that $\text{GCD}(y, r) = ay + br$
4. Return $(b, a - qb)$.

Theorem 7.16 If Euclid-Coefficients(x, y) terminates, its output (a, b) satisfies $ax + by = \text{GCD}(x, y)$.

Proof. Induction on the number of passes the algorithm takes to terminate.

BASE: 1 pass:
Then $y = 0$, the GCD is x, and the return values are $(1, 0)$. Since $(1)(x) + (0)(y) = x$, this is correct.

INDUCTIVE STEP:
Show for n passes, assuming OK for $n - 1$ passes:

Say $x = qy + r$.
By induction, recursive call returns (a, b) with $ay + br = g$, where g is $\mathrm{GCD}(y, x \bmod y)$.
Remember, this is also $\mathrm{GCD}(x, y)$.
Just rewrite g in terms of x and y instead of in terms of y and $x \bmod y$:
$g = ay + br = ay + b(x - qy) = bx + (a - bq)y$.
Says the return values $(b, a - bq)$ are as claimed. ■

Example 2 Table method for getting GCD and Coefficients: run Euclid's algorithm, then fill in the coefficients from the bottom up.

```
      949  273     -2   7
3     273  130      1  -2
2     130   13      0   1
10     13    0      1   0
```

Filling in table entries from bottom: $g = (1)(13) + (0)(0)$.
Then $g = (0)(130) + (1)(13)$.
Then $g = (1)(273) + (-2)(130)$.
Then $g = (-2)(949) + (7)(273)$.

7.5 RSA cryptosystem

We finish today with a nice application of number theory to cryptography.
RSA is a *public key* cryptosystem: anyone can encrypt (code) a message to me, but no one else can decrypt (decode) it.

Uses two *keys*, which are positive integers.
e, the *encryption key*. Public.
d, the *decryption key*. Private (to the decrypter).

A very big positive integer modulus n is also used.

A message m is interpreted as an integer (e.g., by binary representation).

- To encrypt a message $m < n$ into a coded message c:
  Compute $c = m^e \bmod n$.
  Anyone can do this, because e and n are public.
- To decrypt a coded message c into an uncoded message m:
  Compute $m = c^d \bmod n$.

Only someone who knows the secret key d can do this.

Of course, to give the correct message, we need the property:

For any $m < n$, $(m^e)^d \bmod n = m$.
In other words, $m^{ed} \bmod n = m$.

What kinds of numbers satisfy this condition?
Uses a little number theory:

Choose $n = pq$, where p and q are primes.
Choose e, d such that $ed \bmod (p-1)(q-1) = 1$.

Theorem 7.17 Suppose that p and q are primes, $n = pq$, and e, d are positive integers with $ed \bmod (p-1)(q-1) = 1$.
Suppose that m is any natural number $< n$ such that (technicality, usually holds) neither p nor q divides m.
Then $m^{ed} \bmod n = m$.

Proof. In book, see p. 147.
Basically, a bunch of exponential calculations and modular arithmetic. ■

Example (from book): Let $p = 43$, $q = 59$, so $n = 43 \times 59 = 2537$.
Let $e = 13$, $d = 937$. Then $ed \bmod (p-1)(q-1) = 1$, as needed (check).
Encrypt a number 1819 using exponentiation with 13, mod 2537, get 2081.
Decrypt 2081 using exponentiation with 937, mod 2537, get back 1819. Left to the reader to check this.

Fast exponentiation:
In recitation, we will show how to compute the big exponentiations needed to implement RSA quickly. It relies on the fact that $x^{ab} \bmod n = (x^a \bmod n)^b \bmod n$.

This all works to encrypt and decrypt.
But we haven't addressed the important issue: *secrecy*,
i.e., the property that the coded message is hard to decrypt without having access to d, the decryption key.
This turns out to depend on p and q being very large (200 digits?)
And keep them secret, as well as d.
(Finding d actually lets you find p, q, and vice versa.)

In case p and q are large, no one yet knows how to "break" RSA:

- No one knows how to compute p, q efficiently from n. (Factoring large numbers is hard, though there's been progress.)

- No one knows how to compute d efficiently without p, q.
- No one knows how to find m efficiently without d (ie, solve $c = n^e$ for m).

In practice, how do we find suitable p, q, e and d?

There are fast algorithms for generating large prime numbers p, q.

For e and d:
Recall our objective:
Need $de \bmod k = 1$, where $k = (p-1)(q-1)$.
Pick e relatively prime to k (lots of choices here).
That is, $\mathrm{GCD}(e, k) = 1$.
By GCD theorems, we can find coefficients a, b such that $ae + bk = 1$.
Just use $d = a$ – then $de \bmod k = 1$ since $de + bk = 1 \longrightarrow de = -bk + 1 \longrightarrow de \bmod k = 1$.

Used by Netscape, e.g.