3 The rational numbers

3.1 Fractions

You are familiar with fractions from your school days; they are pairs of integers $\frac{a}{b}$, where $b \neq 0$, expressing the idea of a number a of entities shared equally among b individuals. Fractions figure prominently in the oldest mathematical texts that have survived from ancient times, the Rhind Papyrus (kept in the British Museum) and the Moscow Mathematical Papyrus. So unlike negative numbers, fractions appear very early in the historical record.

In school you also learned how to perform the arithmetic operations with fractions, for example, the rules for addition and multiplication are

$$\frac{a}{b}+\frac{c}{d}=\frac{ad+cb}{bd} \quad \text{and} \quad \frac{a}{b}\times\frac{c}{d}=\frac{ac}{bd}$$

We also have a zero: $\frac{0}{1}$ and a one: $\frac{1}{1}$, and these behave as neutral elements for addition and multiplication, respectively. The ring laws can now be checked and they are all valid — or are they? Actually, no, the distributivity law almost always fails:

21

What's going on?

3.2 The rational numbers

Well, what is going on is that the same *ratio* can be expressed in many different ways: Whether we share one chocolate bar among two friends, or share two among four gives the same ratio; everyone gets half a bar. Mathematically we say that the two fractions $\frac{1}{2}$ and $\frac{2}{4}$ represent the same *rational number*. Here is a simple test to check whether two fractions represent the same ratio:

$$\frac{a}{b}=\frac{c}{d} \quad \text{if and only if} \quad a\times d = b\times c$$

We can use it to check that the two fractions we got from testing the distributivity law are equal as rational numbers:

22

Now we look again at all the ring laws and we find that the rational numbers, usually denoted by $\mathbb{Q}$, do indeed form a ring. In fact, rational numbers have a further property, namely, every rational $q \neq 0$ has a **multiplicative inverse** q^{-1}. If q is given as the fraction $\frac{a}{b}$ then the inverse is $\frac{b}{a}$ since $\frac{a}{b}\times\frac{b}{a}=\frac{ab}{ab}=\frac{1}{1}$.

Whenever a (commutative unital) ring has a multiplicative inverse for every element other than zero, we speak of a **field**. For ease of reference we list all the laws again:

23: The field laws

$a+0$	$=$	a	$a\times 1$	$=$	a	(neutral elements)
$a+b$	$=$	$b+a$	$a\times b$	$=$	$b\times a$	(commutativity)
$(a+b)+c$	$=$	$a+(b+c)$	$(a\times b)\times c$	$=$	$a\times(b\times c)$	(associativity)
$a+(-a)$	$=$	0	$a\times a^{-1}$	$=$	$1\ (a\neq 0)$	(inverses)
		$a\times(b+c)=a\times b+a\times c$				(distributivity of $\times$ over $+$)

In Box 13 we showed that the cancellation law for addition is automatically valid once we have an additive inverse for every number; essentially the same derivation shows that the cancellation law for multiplication follows from the existence of multiplicative inverses (for non-zero numbers).

3.3 A normal form and the Euclidean algorithm

Among the infinitely many fractions that represent the same rational number, there is one which is particularly simple. It is the fraction $\frac{a}{b}$ where a and b are as small as possible, and b is positive. We call this simplest fraction **reduced** or a **normal form** for the corresponding rational number.

Given any fraction $\frac{a}{b}$ we can compute the normal form by cancelling out common factors from a and b, and by replacing $\frac{a}{b}$ with $\frac{-a}{-b}$ should b be negative. To find the largest common factor $\mathsf{lcf}(a,b)$ of two natural numbers a and $b \neq 0$, we employ **Euclid's algorithm**.[1] It starts by writing a as $m \times b + r$ with $0 \leq r < b$, which is always possible as we stated in Box 5. If it turns out that $r = 0$, then b divides a and clearly b is the largest common factor of the two. If $r \neq 0$ then we repeat the process with b and r. Since $r < b$, the process will eventually end.

Using pseudocode instead of words, here is the algorithm again. It uses "registers" `x` and `y` to hold the (changing) pair of numbers of which we want to compute the largest common factor:

```
x <- a
y <- b
while ( y != 0 ) {
   r <- x mod y
   x <- y
   y <- r }
return x
```

Why does this work? For this we consider a **loop invariant**, i.e., a logical statement that is true at the beginning of the `while`-loop and also after each of its iterations:

$$\mathsf{lcf}(x,y) = \mathsf{lcf}(a,b)$$

It holds at the start of the `while`-loop because at that point we just initialised `x` with `a` and `y` with `b`. Assuming now that it holds at the start of the execution of the body, we want to show that it holds afterwards, in other words, we need to prove that

$$\mathsf{lcf}(x,y) = \mathsf{lcf}(y,r) \quad \text{where} \quad x = m \times y + r$$

Here is the argument:

24

There is also a **loop variant**, i.e., something which changes in each iteration of the `while`-loop, namely, the variable y; each time the body of the loop is executed, y is replaced with $r = x \bmod y$ and by Box 5 (on page 4) the new value (the remainder of the division of x by y) is strictly smaller than y. The same theorem tells us that y is always greater than or equal to zero, so the sequence of values stored in the variable y is strictly decreasing but bounded from below by zero. It must therefore become equal to zero after finitely many iterations. In other words, the `while`-loop will eventually stop.

3.4 The extended Euclidean algorithm and finite fields

With a little bit more book-keeping we can obtain a very useful statement about the largest common factor, namely, that it can be written as a *linear combination* of the original numbers a and b (named after Étienne Bézout, 1730-1783.):

[1] Named after the ancient Greek mathematician Euclid who lived around 300 BC.

Theorem 1 (Bézout's identity) *Let a and b be natural numbers not both of which are zero. Then there exist integers u and v such that*

$$\mathsf{lcf}(a,b) = u \times a + v \times b\ .$$

The proof is *constructive* in that we give an algorithm to compute u and v, an extension of Euclid's algorithm:

```
x <- a
y <- b
u_x <- 1; v_x <- 0
u_y <- 0; v_y <- 1
while ( y != 0 ) {
   r <- x mod y; k <- x div y
   u <- u_x; v <- v_x
   u_x <- u_y; v_x <- v_y
   u_y <- u - k*u_y; v_y <- v - k*v_y
   x <- y
   y <- r }
return x, u_x, v_x
```

Here is an example run of this algorithm, for $a = 288$ and $b = 150$. We record the contents of each variable at the beginning of the `while`-loop:

25

run	x	y	u_x	v_x	u_y	v_y
0	288	150	1	0	0	1
1						
2						
3						
4						

We obtain $\mathsf{lcf}(288,150) = 6 = 12 \times 288 - 23 \times 150$.

As an exercise you may now show that the following is an invariant for the `while`-loop of the extended algorithm:

$$x = u_x \times a + v_x \times b \qquad \text{and} \qquad y = u_y \times a + v_y \times b\ .$$

Let us now use Bézout's identity to show that $\mathbb{Z}_m$ is actually a field whenever m is a prime number. Remember that we already know that $\mathbb{Z}_m$ is a ring, and the difference between a ring and a field is that in the latter every non-zero element a has a multiplicative inverse, that is, there is an element b such that $a \times b \equiv 1 \bmod m$. The proof is very simple:

26

As we demonstrated, the multiplicative inverse in such a **finite field** can be calculated efficiently with the extended Euclidean algorithm. For small prime numbers m we can also read them off the multiplication table of $\mathbb{Z}_m$. Let's look at two examples and one non-example:

27

It's important to remember that the multiplicative inverse that exists for every $a \neq 0$ in a finite field $\mathbb{Z}_m$, has *nothing at all* to do with the inverse $\frac{1}{a}$ in the field $\mathbb{Q}$.

For $m = 2$ we get the field $\mathbb{Z}_2$, with its two elements 0 and 1 (which we could also call "even" and "odd"). It is also called GF(2) , or "**Galois field** with two elements" in honour of Évariste Galois (1811–32). It is obviously of great interest to computer scientists.

3.5 Fractions and rational numbers on a computer

It is easy to represent fractions in Java; all we need to do is define a class such that every instance of that class has two `BigInteger` variables in which to store numerator and denominator. The arithmetic operations can be defined as methods along the lines indicated in Section 3.1. The test for equality becomes a method with result type `boolean`, and in it we employ the criterion discussed in Section 3.2. We should also have a method `normalise` which uses the Euclidean algorithm to reduce the fraction stored in the class instance.

The implementation suggested here is an example of an **abstract datatype**. On the outside, objects in this class behave like *rationals* while the implementation (invisible to the programmer) works with *fractions*.

Exercises

1. Let $\frac{a}{b}$ be a fraction which is *not* reduced, which means that $\mathsf{lcf}(a,b) > 1$. Let $\frac{c}{d}$ be any fraction. Show that $\frac{a}{b} + \frac{c}{d}$ is also not reduced.
2. What happens in the first round of the Euclidean algorithm when the initial values a and b are such that $a < b$?
3. Find the multiplicative inverse of 4 in the finite field $\mathbb{Z}_{17}$.

Practical advice

In the exam, I expect you to be able to

- prove a statement that follows from the field laws;
- find the multiplicative inverse in a finite field $\mathbb{Z}_m$ where m is prime.

I will also expect you to know

- the difference between fractions and rational numbers;
- the notation $\mathbb{Q}$ for the rational numbers.

More information

Finite rings and finite fields $\mathbb{Z}_m$ are central to cryptography. Every book on cryptography starts with a discussion of their (many and fascinating) properties. Here is a taster, known as *Fermat's Little Theorem*: If $\mathbb{F}$ is a finite field with m elements and $a \neq 0$ is an element of $\mathbb{F}$, then $a^{m-1} = 1$. (Check it in $\mathbb{Z}_5$.)

Mathematics books on the topic are not so easy to read, unfortunately, since fields appear only after a lengthy (albeit useful) discussion of "groups" (yet another concept) and rings.