INF 240 - Exercise problems - 4

Nikolay Kaleyski

1 The Euclidean algorithm for integers

Let us briefly recall the basic notions of divisibility for integers. If $a = bk$ for some $a, b, k \in \mathbb{Z}$, we say that a is a *multiple* of b, or that b *divides* a, or that b is a *divisor* of a; we write this as $b \mid a$. This is the same as saying $a \mod b = 0$.

A *common divisor* of two integers a, b is a number which is a divisor of a as well as a divisor of b. For example, 5 divides both 20 and 30, so it is a common divisor. The *greatest common divisor (GCD)* is, intuitively, the largest (greatest) number which has this property: for example, 10 also divides both 20 and 30, so although 5 is a common divisor, it is definitely not their greatest common divisor.

A related concept is that of the *least common multiple (LCM)*. Again, a *common multiple* of a and b is a number that is a multiple of a as well as a multiple of b; the LCM is the smallest among all numbers having this property.

The GCD and LCM are linked by the formula

$$\gcd(a, b)\mathrm{lcm}(a, b) = ab. \tag{1}$$

Thus, provided one has a way of obtaining one of the LCM or GCD of a pair of given numbers, the other one can be obtained from (1).

Recall the *Euclidean algorithm* for finding the GCD of two positive integers $a, b \in \mathbb{N}$. Initially, we set $r_0 = a$, $r_1 = b$ (assuming $a > b$). In each step, we divide r_{i-2} by r_{i-1} with remainder r_i, i.e. we write $r_{i-2} = r_{i-1}q + r_i$. We repeat the same procedure again and again, until we obtain a zero remainder. The last non-zero remainder obtained using this procedure is the GCD of a and b.

Exercise 1. *Use the Euclidean algorithm to find* $\gcd(115, 69)$ *and* $\gcd(115, 48)$.

Exercise 2. *Find* $\mathrm{lcm}(115, 60)$ *and* $\mathrm{lcm}(115, 48)$.

2 The Euclidean algorithm for polynomials

The divisibility properties, least common multiple (LCM), greatest common divisor (GCD), etc. of two polynomials $f(x), g(x)$ are defined and computed according to the same logic as in the case of integers.

The *(extended) Euclidean algorithm for polynomials* is performed analogically to the algorithm for integers, simply by replacing the notions of divisibility for integers with their analogues for polynomials.

Exercise 3. *Find the greatest common divisors of* $f(x) = x^5 + 2x^4 - x^2 + 1$ *and* $g(x) = x^4 - 1$ *over* $\mathbb{Q}$.

Exercise 4. *Find the least common multiple of $f(x)$ and $g(x)$ from the previous exercise.*

Performing calculations of this type over finite structures (e.g. $\mathbb{F}_7$ as opposed to $\mathbb{Q}$) is typically easier since all coefficients are integers and no fractions are involved. The principles remain the same.

Exercise 5. *Let $f(x) = x^7 + 1$ and $g(x) = x^5 + x^3 + x + 1$ over $\mathbb{F}_2$. Compute the greatest common divisor $\gcd(f(x), g(x))$ of the two polynomials.*

Exercise 6. *Compute $\mathrm{lcm}(f(x), g(x))$ for $f(x)$ and $g(x)$ from the previous exercise.*

3 Computing with polynomials in *Magma*

3.1 Creating the base structure

The *Magma* programming language makes it very easy to work with polynomials defined over an arbitrary field or ring. Operators and functions allowing one to add, multiply and divide polynomials, as well as to calculate the GCD or LCM of a pair of polynomials are readily available.

The first step is to create a representation for the base structure over which the polynomials are defined. The infinite sets $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ of integer, rational, real, and complex numbers, respectively, can be obtained by invoking built-in functions in *Magma*:

```
base := IntegerRing();
//or
base := RationalField();
//or
base := RealField();
//or
base := ComplexField();
```

To create a representation of $\mathbb{F}_n$, e.g. for $n = 16$, we would call

```
base := FiniteField(16);
```

3.2 Creating the polynomial ring

Once the base structure $\mathbb{B}$ is created and stored in a variable, we can define the ring $\mathbb{B}[x]$ of polynomials in indeterminate x over $\mathbb{B}$. The name of the indeterminate does not have to be x; it can be e.g. y, z, A, *ind*, or any other valid keyword, but we have to specify it during the creating of the polynomial ring.

```
P<z> := PolynomialRing(base);
```

The above will create a polynomial ring over the structure stored in *base* with indeterminate z. The polynomial ring is stored in the variable P.

3.3 Defining polynomials

Once the polynomial ring is created, one can define polynomials using the indeterminate that was specified during its creation:

```
f_1 := z^5 + z^2 + z;
f_2 := z^8 - 3 * z + 1;
```

The coefficients, of course, must belong to the base structure; if the base structure is one of the infinite sets of numbers or $\mathbb{F}_p$ for some prime number p, then we can assume that all possible coefficients are numbers and can be entered exactly as above.

3.4 Working with polynomials

Once one (or several) polynomials are defined, a lot of standard operations can easily be performed with *Magma*. For instance, invoking

```
GCD( f_1 , f_2 );
```

or

```
LCM( f_1 , f_2 );
```

will compute and return the LCM, resp. GCD of two given polynomials. Addition, subtraction and multiplication of polynomials is performed exactly as one would expect, e.g.

```
( f_1 + f_2 )* f_1 ;
```

Division is somewhat more complicated due to the fact that when dividing $a(x)$ by $b(x)$, there are *two outputs*, viz. the quotient $q(x)$ and the remainder $r(x)$: $a(x) = b(x)q(x) + r(x)$. The quotient, resp. remainder, is obtained using the *div*, resp. *mod* command:

```
quotient := f_1 div f_2;
remainder := f_1 mod f_2;
assert f_1 eq quotient * f_2 + remainder;
```

Exercise 7. *Define polynomial rings over $\mathbb{Q}$ and over $\mathbb{F}_2$, and find the GCD's and LCM's of the polynomials from the previous exercises and verify your answers.*