Polynomial division

Application of discrete models

Previous documents (Divisors, prime numbers; Greatest common divisor) discussed divisors, divisibility and related concepts for integers. This document introduces their equivalents for polynomials. Many of these concepts will be defined more generally, in rings (see: Algebraic structures). This allows us to easily apply them to both integers and polynomials, and to potentially extend them to other structures.

1 Division with remainder

Just like for integers, division doesn't generally work for polynomials (e.g. $x^2/(x+1)$ is not a polynomial, in the same way as $2/3 \notin \mathbb{Z}$). But just like for integers, we can introduce **division with remainder** for polynomials (for integers, see: Divisors, prime numbers, Theorem 1.1):

Theorem 1.1 (Division theorem for polynomials)**.** *If K is a field, $a, b \in K[x]$ and $b \neq 0$, then there exist unique $q, r \in \mathbb{R}[x]$ for which $a = qb + r$ and $\deg r < \deg b$.*

Definition 1.2. q is the **quotient polynomial** of a and b, and r is their **remainder polynomial**.

Example. If $a = 2x^3 + x^2 - 2x - 1$ and $b = x^2 - 2x + 1$, then $q = 2x + 5$ and $r = 6x - 6$, because $a = 2x^3 + x^2 - 2x - 1 = (2x+5)(x^2 - 2x + 1) + (6x - 6)$ and $\deg(6x - 6) < \deg(x^2 - 2x + 1)$.

Manually, division with remainder can be performed very similarly to integer long division, but instead of going by the positions of the digits, we need to go by the degrees of the terms. Starting from a, multiples of b are subtracted from it to cancel its leading coefficient, thus decreasing the degree of the polynomial step by step. This is repeated until the degree goes below $\deg b$, when we get the remainder. For the above example ($a = 2x^3 + x^2 - 2x - 1$ and $b = x^2 - 2x + 1$):

$$\begin{aligned} p_0 &:= a &&= 2x^3 + x^2 - 2x - 1, \\ p_1 &:= p_0 - 2x \cdot b &&= 5x^2 - 4x - 1, \\ p_2 &:= p_1 - 5 \cdot b &&= 6x - 6, \end{aligned}$$

where $\deg p_2 < \deg b$, so the remainder is $r := p_2 = 6x - 6$, and the quotient can be assembled from the multipliers of b: $q = 2x + 5$.

In Sage, polynomial division works in the same way as integer division (`//` and `%`):

```
sage: P.<x> = QQ[]    # not ZZ[]!
sage: a = 2*x^3 + x^2 - 2*x - 1; b = x^2 - 2*x + 1
sage: a // b
2*x + 5
sage: a % b
6*x - 6
```

Attention: avoid regular division (/), because the type of the result is not polynomial:

```
sage: a / b          # the result is not polynomial
(2*x^2 + 3*x + 1)/(x - 1)
sage: (b*x) / b      # not even when it looks so!
x
sage: _.degree()     # doesn't work, because it is not a polynomial
[...]
AttributeError: '[...]' object has no attribute 'degree'
```

It is important that division with remainder works only over *fields* (in general). For example, in $\mathbb{Z}[x]$ (where $\mathbb{Z}$ is not a field), the quotient of $a = x^2 + 3$ and $b = 2x$ is $q = 1/2 \cdot x$, and the remainder is $r = 3$, i.e. we need $\mathbb{Q}[x]$ to calculate the result. But we can do it over other fields too, e.g. in $\mathbb{Z}_5[x]$, the quotient of $a = x^2 + \overline{3}$ and $b = \overline{2}x$ is $q = \overline{3}x$, and the remainder is $r = \overline{3}$ (since in $\mathbb{Z}_5$, $\overline{2}^{-1} = \overline{3}$).

An interesting special case is when the divisor is of the form $b = x - c$, i.e. it is a root factor. Then, by the division theorem, $p = (x - c)\,q + r$, where $\deg r < \deg b = 1$, i.e. the remainder is a constant polynomial. Substituting c into this, we get $p(c) = 0\,q(c) + r = r$, which means that the remainder is the value of p at c. In particular, if c is a root of p, then the remainder is 0, i.e. $p = (x - c)\,q$. If $p(c)$ is calculated using Horner's method (see: Basics of polynomials), then it can be proven that we automatically get the quotient polynomial q too:

Theorem 1.3. *If $p \in K[x]$ and $c \in K$, then the remainder of $p = a_n x^n + \ldots + a_1 x + a_0$ and $b = x - c$ is the constant polynomial $p(c)$, and their quotient is $q = b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$, where the b_k's are the intermediate values of the variable b in the algorithm of Horner's method, more precisely they can be calculated by the following recursion: $b_{n-1} := a_n$ and $b_k := b_{k+1}c + a_{k+1}$.*

Example. If $p = 2x^3 - 3x^2 - 4x + 7$ and $c = 2$, then, by Horner's method, $p(c) = ((2c - 3)c - 4)c + 7$, i.e. $b_2 = 2$ (= p's leading coefficient), $b_1 = b_2 c - 3 = 1$, $b_0 = b_1 c - 4 = -2$ and $b_{-1} = b_0 c + 7 = 3$, so $q = b_2 x^2 + b_1 x + b_0 = 2x^2 + x - 2$ and $r = p(c) = b_{-1} = 3$. The same can be written in a table form, where the first row contains the coefficients of p, and the b_k's are calculated in the second row:

	2	-3	-4	7	$\longleftarrow$	$p = 2x^3 - 3x^2 - 4x + 7$
$c = 2$	2	1	-2	3	$\longleftarrow$	$q = 2x^2 + x - 2,\ \ p(c) = 3$

2 Divisibility

In this section, we generalize the concept of divisibility ($b \mid a$) – already well-known for integers – to rings, more specifically to integral domains. Reminder: the main integral domains are the integers ($\mathbb{Z}$), the polynomial rings ($K[x]$, where K is an integral domain) and all fields.

Definition 2.1. In an integral domain R, $a \in R$ is **divisible** by $b \in R$ (or b is a **divisor** of a) if $\exists c \in R : a = bc$. Notation: $b \mid a$.

Example. In $\mathbb{Z}$, $2 \mid 6$, because $6 = 2 \cdot 3$.

Example. In $\mathbb{R}[x]$, $2x^2 - x - 1$ is divisible by $x - 1$, because $2x^2 - x - 1 = (x - 1)(2x + 1)$.

Example. In fields, divisibility is trivial, because all elements are divisible by all nonzero elements, since $c := b/a$ always exists. For example, in $\mathbb{Q}$, 5 is divisible by 2, because $5 = 2 \cdot (5/2)$.

The point of divisibility for polynomials is that it has a strong connection with the roots: if $b \mid a$, then all roots of b are also roots of a (and at least as many times as for b). For example, 1 is a root of $x-1$, and $x-1 \mid 2x^2-x-1$, so we know that 1 must also be a root of $2x^2-x-1$.

Just like for integers, divisibility of polynomials can be checked using division with remainder: if $b \neq 0$, then $b \mid a \iff a \bmod b = 0$.

To discuss divisibility in general, we need a few more definitions:

Definition 2.2. In an integral domain R, $u \in R$ is a **unit** if it divides the identity element.

Example. $\mathbb{Z}$ has two units: 1 and -1 (the divisors of 1).

Example. In $\mathbb{R}[x]$, all nonzero constant polynomials are units.

Example. In fields, all nonzero elements are units.

Example. In general, the units of $K[x]$ are the units of K (as constant polynomials), e.g. in $\mathbb{Z}[x]$, the units are 1 and -1, and in $\mathbb{Q}[x]$, all nonzero constant polynomials are units.

Note: do not confuse *units* with the *identity element*, as the latter is unique, but there can be more than one units in an integral domain (as we have seen above).

We also need another important concept related to divisibility:

Definition 2.3. In an integral domain R, $a \in R$ and $b \in R$ are **associates** if they divide each other: $a \mid b$ and $b \mid a$.

Example. In $\mathbb{Z}$, the associates are integers with the same magnitude, e.g. 5 and -5.

Example. In $\mathbb{R}[x]$, the associates are the constant multiples of each other, e.g. $x-1$, $2x-2$, $3x-3$, $-x+1$, $-x/2+1/2$ etc. are all associates.

Example. In fields, any two nonzero elements are associates.

Notice that the associates are always unit multiples of each other, e.g. in $\mathbb{Z}$, $-5 = -1 \cdot 5$, in $\mathbb{R}[x]$, $2x-2 = 2(x-1)$ etc. The point of associates is that they behave in exactly the same way regarding divisibility, e.g. $2 \mid 6 \implies 2 \mid -6,\ -2 \mid 6,\ -2 \mid -6$. Therefore, we can say that the associates are "essentially" the same.

3 Irreducible elements

The concept of *prime numbers* can be generalized as follows:

Definition 3.1. In an integral domain R, $a \in R$ is **irreducible** if it is nonzero, non-unit, and its only divisors are units and associates of a.

The "nonzero, non-unit" corresponds to excluding 0 and 1 from the definition of prime numbers, and the "units and associates of a" corresponds to "its only divisors are 1 and n".

Example. The irreducible elements of $\mathbb{Z}$ are the prime numbers and their negatives: $2, 3, 5, 7, 11, \ldots$, $-2, -3, -5, -7, \ldots$. (For simplicity, prime numbers were defined to be positive, but for irreducible elements in general, it is not so easy to make a convenient choice from among the associates.)

Example. x^2-2 is irreducible in $\mathbb{Z}[x]$, but not in $\mathbb{R}[x]$, because $x^2-2 = (x-\sqrt{2})(x+\sqrt{2})$.

Example. Fields have no irreducible elements, because all nonzero elements are units.

In polynomial rings ($K[x]$), the irreducible polynomials have the following properties:

1. Over fields, all linear polynomials are irreducible, because the divisors of $p = ax+b$ are constant and linear polynomials, the former of which are units, and the latters are p's associates.
2. Irreducible polynomials of degree ≥ 2 cannot have roots (in the coefficient ring K), otherwise we could extract a root factor, which would mean that the polynomial is not irreducible.
3. In $\mathbb{C}[x]$, *only* linear polynomials are irreducible, because the fundamental theorem of algebra (Basics of polynomials, Theorem 4.7) states that all nonconstant polynomials have roots.
4. In $\mathbb{R}[x]$, only linear and quadratic polynomials can be irreducible (e.g. $2x + 3$ and $x^2 + 1$).
5. But in $\mathbb{Z}[x]$ and $\mathbb{Q}[x]$, irreducible polynomials can have arbitrary degree, e.g. all polynomials of the form $x^n - 2$ are irreducible.

The significance of prime numbers was shown by the fundamental theorem of arithmetic (Divisors, prime numbers, Theorem 3.7), which guaranteed that the prime factorization is unique. But if we try to generalize this, it will not be true for all integral domains, only for specific ones:

Definition 3.2. An integral domain R is called a **unique factorization domain (UFD)** if all nonzero non-unit elements can be written uniquely as a product of irreducible elements, up to associatedness and the order of the factors.

Example. $\mathbb{Z}$ is a UFD by the fundamental theorem of arithmetic. For example, $-10 \in \mathbb{Z}$ can be factored in multiple ways: $-10 = (-2)\cdot 5 = 2\cdot(-5) = 5\cdot(-2) = (-5)\cdot 2$, but these factorizations differ only in order and associatedness (i.e. sign), i.e. they are "essentially" the same. (For the fundamental theorem of arithmetic, we could get around associatedness by working with only positive integers.)

Example. All fields are trivially UFDs (since they have no nonzero non-unit elements).

Example. $\mathbb{R}[x]$ is a UFD, e.g. $x^2-1 = (x+1)(x-1) = (2x+2)(x/2-1/2) = (-3x+3)(-x/3-1/3)$ etc., but these factorizations differ only in order and associatedness (i.e. constant multiplier).

Example. If K is a UFD, then $K[x]$ is also a UFD, i.e. $\mathbb{Z}[x]$, $\mathbb{Q}[x]$, $\mathbb{Z}_5[x]$ etc. are all UFDs.

In Sage, the `factor()` function (see: Divisors, prime numbers) works for arbitrary UFDs, with the following caveat: it selects the irreducible elements (from its associates) so that they are as simple as possible (e.g. in $\mathbb{Z}$, it selects the positive ones), but if it doesn't work that way (e.g. if the factored number is negative), then it adds a unit (e.g. -1) to the beginning of the product. The result of `factor()` is a `Factorization` object, which behaves like a sequence (e.g. it can be iterated through by a `for` loop), but the unit, if any, is not part of this sequence, instead, it can be queried separately by `unit()`. For example:

```
sage: F = factor(-40)
sage: F
-1 * 2^3 * 5
sage: for (p, k) in F:
....:     print(p, k)
2 3
5 1
sage: F.unit()
-1
```

```
sage: P.<x> = QQ[]
sage: F = factor(2*x^2 - 1/2)
sage: F
(2) * (x - 1/2) * (x + 1/2)
sage: [p for (p, k) in F]
[x - 1/2, x + 1/2]
sage: F.unit()
2
```

4 Greatest common divisor

Now we extend the greatest common divisor and least common multiple (see the document called Greatest common divisor) from integers to arbitrary integral domains:

Definition 4.1. In an integral domain R, a **greatest common divisor** of $a \in R$ and $b \in R$ is $c \in R$ if $c \mid a \ \wedge \ c \mid b \ \wedge \ \forall d \in R : d \mid a \ \wedge \ d \mid b \implies d \mid c$. Notation: $\gcd(a, b)$.
Furthermore, a and b are **coprime** if their greatest common divisor is a unit.

Definition 4.2. In an integral domain R, a **least common multiple** of $a \in R$ and $b \in R$ is $c \in R$ if $a \mid c \ \wedge \ b \mid c \ \wedge \ \forall d \in R : a \mid d \ \wedge \ b \mid d \implies c \mid d$. Notation: $\mathrm{lcm}(a, b)$.

Example. In $\mathbb{Z}$, a greatest common divisor of 12 and 20 is 4, and a least common multiple is 60.

Example. In $\mathbb{R}[x]$, a greatest common divisor of $p = x^2 - 2x + 1 = (x-1)^2$ and $q = 2x^2 - x - 1 = (x-1)(2x+1)$ is $x - 1$, and a least common multiple is $(x-1)^2(2x+1) = 2x^3 - 3x^2 + 1$.

But in such a general context, there are several problems with these concepts. First, they do not necessarily exist. But fortunately, in the important special cases that we care about, they do exist:

Theorem 4.3. *In a unique factorization domain (UFD), any two elements have a greatest common divisor and a least common multiple.*

So in $\mathbb{Z}$, $\mathbb{Z}[x]$, $\mathbb{R}[x]$ etc., gcd and lcm do always exist. (The theorem is not surprising if we consider how gcd and lcm can be calculated from the prime factorization in $\mathbb{Z}$.)

But there is another problem: in general, gcd and lcm are not unique!

Example. In $\mathbb{Z}$, the above definition for $\gcd(6, 4)$ is satisfied by both 2 and -2. In the original definition (Greatest common divisor, Definition 1.1), the only reason gcd was unique is that we restricted it to $\mathbb{N}$.

Example. For the above $p = x^2 - 2x + 1$ and $q = 2x^2 - x - 1$, in $\mathbb{R}[x]$, the following polynomials all qualify as their greatest common divisors: $r_1 = x - 1$, $r_2 = -x + 1$, $r_3 = 2x - 2$, $r_4 = -x/2 + 1/2$ etc. (But in $\mathbb{Z}[x]$, only r_1 and r_2 do, because there $r_3 \nmid p$ and $r_4 \notin \mathbb{Z}[x]$.)

But even if we can't have complete uniqueness, there is uniqueness "in some sense":

Theorem 4.4. *The greatest common divisor and the least common multiple, if they exist, are unique up to associatedness.*

Example. In $\mathbb{Z}$, gcd is unique *up to sign*: $\gcd(6, 4)$ can be both 2 and -2, but nothing else. And if we restrict the gcd to $\mathbb{N}$, then it is completely unique.

Example. For polynomials in $\mathbb{R}[x]$, gcd is unique *up to a constant multiplier*: in the above example, $r_1 = -r_2 = 1/2 \cdot r_3 = -2r_4$. For complete uniqueness, we can require e.g. that the leading coefficient of the gcd must be 1 (e.g. then $r_1 = x-1$ is the "true" gcd). (But in $\mathbb{Z}[x]$, this is not always possible.)

But why is polynomial gcd interesting? Because it tells something important about the roots:

Theorem 4.5. *If $p, q, r \in K[x]$ and $r = \gcd(p, q)$, then the roots of r are exactly the common roots of p and q, and their multiplicity in r is the smaller one of their multiplicities in p and q.*

Example. $p = x^2 - 2x + 1$ has only one root: 1 (a double root); $q = 2x^2 - x - 1$ has two roots: 1 and $-1/2$ (simple roots); so $r = \gcd(p, q)$ (or any other r_i) has only one root: 1 (as a simple root).

So if we need to find the common roots of two polynomials, we only need to calculate the roots of their gcd, which is usually easier, because it has a smaller degree; and if it is a constant, i.e. the two polynomials are coprime, then we know immediately that they cannot have common roots.

But how do we calculate the gcd of two polynomials? In the same way as for integers: using the Euclidean algorithm (see the document: Greatest common divisor), which works for polynomials too, because they also have a remainder operation (see the beginning of this document).

Example. The Euclidean algorithm for $p = x^4 - 4x^3 + 1$ and $q = x^3 - 3x^2 + 1$:

$$\begin{aligned}
r_0 &:= p = x^4 - 4x^3 + 1, \\
r_1 &:= q = x^3 - 3x^2 + 1, \\
r_2 &:= r_0 \bmod r_1 = -3x^2 - x + 2, \\
r_3 &:= r_1 \bmod r_2 = 16/9 \cdot x - 11/9, \\
r_4 &:= r_2 \bmod r_3 = -27/256, \\
r_5 &:= r_3 \bmod r_4 = 0,
\end{aligned}$$

therefore, $\gcd(p, q) = -27/256$. But as we have seen, the gcd is not unique, so we can multiply it by any nonzero constant, e.g. $\gcd(p, q) = -27$ and $\gcd(p, q) = 1$ are also true. (In general, if the result is a constant, then any other nonzero constant is also correct.) This means that p and q are coprime, i.e. they cannot have common roots. Note: since constant multipliers don't change the correctness of the result, we can also multiply the earlier polynomials by constants to simplify the calculations, e.g. we can use $r_3' := 9r_3 = 16x - 11$ instead of r_3, and so on.

Important: in general, the Euclidean algorithm works only over *fields*, e.g. in $\mathbb{R}[x]$ or $\mathbb{Q}[x]$. It doesn't work in $\mathbb{Z}[x]$, because the algorithm can introduce fractions (as we have seen in the above example). This is true even if a gcd exists in $\mathbb{Z}[x]$, e.g. $\gcd(p, q) = 1 \in \mathbb{Z}[x]$ was true above – but to know this, we first had to calculate another gcd in $\mathbb{Q}[x]$: $-27/256$.

Not only the regular Euclidean algorithm, but its extended version can also be applied to polynomials – just like for integers. This is guaranteed by the following theorem:

Theorem 4.6 (Bézout's identity for polynomials)**.** *If K is a field, and $r \in K[x]$ is a greatest common divisor of $p \in K[x]$ and $q \in K[x]$, then there exist $u, v \in K[x]$ for which $pu + qv = r$.*

Example. The gcd of $p = x^3 - 3x^2 + 3x - 1 = (x-1)^3$ and $q = 2x^2 - x - 1 = (x-1)(2x+1)$ is $r = x - 1$, and indeed: $p \cdot 4/9 + q \cdot (-2/9 \cdot x + 5/9) = r$, i.e. $u = 4/9$ and $v = -2/9 \cdot x + 5/9$.

In Sage, `gcd(p, q)` and `xgcd(p, q)` can be used for polynomials too, just like for integers. (Except that for polynomials, `gcd(p, q)` is not the only possible solution, as the gcd is not unique.)

5 Formal derivative

From calculus, we know the **derivative**, and it has useful properties for polynomials too. But here, we don't want to use all the machinery of calculus (e.g. limits, continuity), and in some cases, we simply can't (because there is no continuity in *discrete* structures like $\mathbb{Z}$ and $\mathbb{Z}_m$). Therefore, instead of using calculus, we simply *define* the derivative for polynomials (and call it *formal* derivative, to indicate that it is not based on calculus):

Definition 5.1. If K is an integral domain, and

$$p = a_n x^n + \ldots + a_k x^k + \ldots + a_2 x^2 + a_1 x + a_0 \in K[x],$$

then the **formal derivative** of p is the following polynomial:

$$p' := n a_n x^{n-1} + \ldots + k a_k x^{k-1} + \ldots + 2a_2 x + a_1 \in K[x],$$

where the product ka_k means $a_k + a_k + \ldots + a_k$ with exactly k terms.

Example. The derivative of $p = x^3 + 2x^2 - 3x + 5 \in \mathbb{Z}[x]$ is $p' = 3x^2 + 4x - 3$.

In the above definition, the meaning of ka_k had to be given, because the coefficients a_k are not necessarily numbers, but just elements of an arbitrary integral domain K, and we haven't yet defined how to multiply them by an integer (since the exponents are always integers, no matter what ring the coefficients are from). For example, in $\mathbb{Z}_3[x]$, the formal derivative of $p = \overline{2}x^5$ is $p' = 5 \cdot \overline{2}x^4 = (\overline{2} + \overline{2} + \overline{2} + \overline{2} + \overline{2})x^4 = \overline{1}x^4 = x^4$.

Also, non-number polynomials can lead to unexpected results. The definition suggests that the formal derivative of a degree-n polynomial has degree $n-1$. Which is true for real polynomials ($\mathbb{R}[x]$), but not necessarily in $\mathbb{Z}_m[x]$: e.g. the cubic polynomial $p = x^3 + \overline{2}x^2 + \overline{2}x + \overline{1} \in \mathbb{Z}_3[x]$ has only a linear derivative: $p' = x + \overline{2}$, because the quadratic term of p' vanishes in $\mathbb{Z}_3$: $3 \cdot \overline{1}x^2 = (\overline{1} + \overline{1} + \overline{1})x^2 = \overline{0}x^2$. It can also happen that the whole polynomial vanishes, e.g. the derivative of $p = x^6 + x^3 + \overline{1} \in \mathbb{Z}_3[x]$ is $p' = \overline{0}$.

In Sage, the derivative of a polynomial can be obtained by the member function `p.diff()`.

Why is the derivative useful for polynomials? Because of the following property about the roots:

Theorem 5.2. *If $c \in K$ is a root of multiplicity $k \geq 1$ of $p \in K[x]$, then c's multiplicity in p' is $k-1$.*

Example. $c = 3$ is a double root of $p = x^3 - 4x^2 - 3x + 18 = (x + 2)(x - 3)^2$, therefore, the same c is a simple root of $p' = 3x^2 - 8x - 3 = (x - 3)(3x + 1)$. (Also, p has a simple root $c = -2$, which is not a root of p', i.e. its multiplicity is zero.)

That is, the derivative preserves the *multiple* roots of the original polynomial (with one less multiplicity). However, a slight problem is that p' can have additional roots, e.g. $c = -1/3$ above. But we can easily filter them out by taking $\gcd(p, p')$, which contains only their common roots.

Example. Continuing the above example, $\gcd(p, p') = x - 3$, whose only root is the simple root $c = 3$. This shows clearly that the only multiple root of p is $c = 3$.

So if we are only interested in the multiple roots of a polynomial p, then we only need to find the roots of $\gcd(p, p')$, which is easier, because it has a lower degree than p, and the gcd can be calculated efficiently using the Euclidean algorithm. Furthermore, if $\gcd(p, p')$ is constant, then we know immediately that p cannot have multiple roots, without calculating any roots.

Also, if we divide the polynomial by $\gcd(p, p')$, i.e. calculate $p \,/\, \gcd(p, p')$ (which is also a polynomial, because $\gcd(p, p') \mid p$), then it has exactly the same roots as p, but only as simple roots (since each $(x-c)^k$ was divided by $(x-c)^{k-1}$). So if we need to find the roots of a polynomial p, it is useful to first divide it by $\gcd(p, p')$, because doing so will not change the roots (only their multiplicities),

but it may reduce the degree.

Note: for number polynomials (e.g. in $\mathbb{R}[x]$), this method works perfectly, but in other structures, e.g. in $\mathbb{Z}_m[x]$, we may encounter problems. For example, we have seen above that the derivative of $p = x^6 + x^3 + \overline{1} \in \mathbb{Z}_3[x]$ vanishes: $p' = \overline{0}$. Which is a problem, because then $p \,/\, \gcd(p, p') = p/p = 1$, so we can't filter out the multiple roots – even though p has plenty of them: $p = x^6 + x^3 + \overline{1} = (x - \overline{1})^6$, i.e. it has a sextuple root, $c = \overline{1}$.

Now let's see how to efficiently evaluate the derivative of a polynomial at a given place.

In Basics of polynomials, we have seen Horner's method, which calculated the value $p(c)$ of a polynomial p efficiently. If we also need the value of the derivative, $p'(c)$, we can of course calculate it in the same way from p'. However, we need to calculate the derivative polynomial p' for this, which requires further multiplications (and more memory). But we can avoid that: Horner's method can be extended to calculate $p'(c)$ directly, along with $p(c)$, without calculating p' itself. This is made possible by the following theorem:

Theorem 5.3. *If $p \in K[x]$, $c \in K$ and the quotient of p by $x - c$ is $q \in K[x]$, i.e. $p = (x - c)\,q + p(c)$, then $p'(c) = q(c)$.*

This is useful, because Horner's method calculates q automatically (see above, after the division with remainder), so we can just repeat Horner's method to evaluate $q(c)$, i.e. $p'(c)$. Also, these two phases can be combined, so we don't need to store the whole q. (This means a total of approx. $2n$ multiplications and $2n$ additions, whereas calculating p' would require n more multiplications.)

Example. For $p = x^4 - 3x^3 + 6x + 3$ and $c = 2$, the extended Horner's method can be written as:

	1	−3	0	6	3	⟵	$p = x^4 - 3x^3 + 6x + 3$
$c = 2$	1	−1	−2	2	7	⟵	$q = x^3 - x^2 - 2x + 2$, $p(c) = 7$
$c = 2$	1	1	0	2		⟵	$q(c) = p'(c) = 2$

Extended Horner's method:

Input: the coefficients of p: $a_0, a_1, \ldots, a_n$, and the value c
Output: $(p(c), p'(c))$
$b := a_n$
$d := 0$
for $k = n-1$ *to* 0 **do**
  $d := d \cdot c + b$
  $b := b \cdot c + a_k$
→ *(b, d)*

6 Cyclic redundancy check (CRC)

When data is transmitted through a physical channel (e.g. in a wire or with radio signals), errors can happen, i.e. the received data may not be exactly the same as what was sent. How can we detect such errors?

The solution is *redundancy*: extend the message by a new part called the *check value*, which is calculated from the message, i.e. it doesn't add any new information, but it can be used to check whether the data is correct. If the receiver finds that the check value doesn't correspond to the message, than they can be sure that the message or the check value (or both) was corrupted.

A simple example is the **parity bit**: the message, which is a bit sequence, is extended by a single new bit: if the message contains an even number of 1 bits, then append 0, otherwise append 1. Then the result always has an even number of 1's. For example: $0101 \to 01010$, and $1101 \to 11011$. If we get 11010 on the other end of the channel, then we know that an error must have happened, because it contains an odd number of 1's. But we don't know *where* the error is, e.g. both of the above two example bit sequences differ only in one bit from this one. The parity bit method always detects a single-bit error, but it can't detect two bits (because then the number of 1's remains even).

A more advanced method for error-detection is **Cyclic Redundancy Check (CRC)**.

CRC treats the bit sequences as polynomials, where each coefficient is a bit (0 or 1). For example, the bit sequence 101011 corresponds to the polynomial $x^5 + x^3 + x + 1$ (the last bit is the constant term, the previous one is the linear term, and so on). The polynomials are in $\mathbb{Z}_2[x]$, i.e. all calculations are done modulo 2, e.g. $1 + 1 = 0$, $0 - 1 = 1$ etc. As a consequence of this, for all polynomials, $p + p = 0$ (e.g. $(x^2 + 1) + (x^2 + 1) = (1 + 1)x^2 + (1 + 1) = 0$), so $p = -p$, therefore addition is the same as subtraction: $p - q = p + (-q) = p + q$. This addition/subtraction is also the same as the bitwise "exclusive or" (XOR, $\oplus$) operation (e.g. $1 + 1 = 0 \longrightarrow$ "true" $\oplus$ "true" = "false").

The main operation of CRC is polynomial division with remainder in $\mathbb{Z}_2[x]$. The divisor is a fixed polynomial $g \in \mathbb{Z}_2[x]$, called the **generator polynomial** of CRC. If $n := \deg g$ (i.e. g has n+1 bits), then we call this an ***n*-bit CRC** (notation: CRC-n), since the remainder by g fits into n bits (as its degree is at most $n-1$).

CRC message sending:

Input: $m \in \mathbb{Z}_2[x]$, the message to be transmitted *(it can be treated as a bit sequence)*
Calculate the polynomial mx^n *(i.e. extend m by n zeros)*
$c := mx^n \bmod g$ *(where c is the **check value**)*
Sent **codeword**: $mx^n + c$ *(i.e. extend m by the n-bit check value c)*

CRC checking (method 1):

Input: $mx^n + c \in \mathbb{Z}_2[x]$, the received codeword (see above)
Split the codeword into m and c *(i.e. separate the last n bits)*
$c' := mx^n \bmod g$
if $c' \neq c$ **then** $\to$ *error*

CRC checking (method 2):

Input: $mx^n + c \in \mathbb{Z}_2[x]$, the received codeword (see above)
$e := (mx^n + c) \bmod g$
if $e \neq 0$ **then** $\to$ *error*

The two methods are equivalent, because if the remainder of mx^n is c, i.e. $mx^n = qg + c$, then $mx^n - c = qg$; but in $\mathbb{Z}_2[x]$, subtraction is the same as addition, so $mx^n - c = mx^n + c$, therefore $g \mid mx^n + c$, i.e. $mx^n + c$ indeed gives the remainder 0.

Example. Let the generator be $g = x^4 + x + 1$, or as a bit sequence, $g = 10011$ (so $n = 4$), and let the message be $m = 11000101$. For transmission, calculate the polynomial $mx^n = 11000101|0000$ (i.e. append four zeros to m), and divide it by g (see below on the left). The remainder is $c = 110$, extend it to n bits: $c = 0110$, and append it to the message: $mx^n + c = 11000101|0110$, to get the

transmitted codeword. The receiver checks this value by dividing it by g (see below on the right), and since the remainder is 0, the message is accepted as valid. (Again: these are not numbers in binary, but polynomials in $\mathbb{Z}_2[x]$, so subtraction is really XOR.)

Sending ($mx^n \bmod g$):

$$
\begin{array}{r}
11000101|0000 \bmod 10011 \\
\underline{-10011} \\
10111 \\
\underline{-10011} \\
10001 \\
\underline{-10011} \\
10000 \\
\underline{-10011} \\
110
\end{array}
$$

Checking ($mx^n + c \bmod g$):

$$
\begin{array}{r}
11000101|0110 \bmod 10011 \\
\underline{-10011} \\
10111 \\
\underline{-10011} \\
10001 \\
\underline{-10011} \\
10011 \\
\underline{-10011} \\
00
\end{array}
$$

The advantage of CRC is that this polynomial division can be done very efficiently on computers, because it needs only bit manipulations, which computers are already very good at.

Unfortunately, it is impossible to detect *all* errors, and the check will occasionally accept some invalid codewords too. The goal is to make this as unlikely as possible. An error means that instead of the transmitted mx^n+c, the receiver gets mx^n+c+e, where e is the **error polynomial**, i.e. which has a coefficient 1 for each bit that was corrupted during transmission. If an error is undetected, it is because $g \mid mx^n + c + e$ (i.e. the remainder is 0), therefore $g \mid e$ (since for the correct codeword, $g \mid mx^n + c$). A good g therefore does not divide e for the most frequently occuring error patterns.

Here are some aspects of choosing a generator polynomial (g):

1. Its degree (n) equals the number of check bits, so the bigger it is, the smaller the error probability can be (a random bit sequence is accepted with $1/2^n$ probability).
2. The constant term (last bit) of g should be 1, because:
   - If it were 0, then $x \mid g$, so the last (few) bits of $c = mx^n \bmod g$ were always 0, which would reduce the number of useful bits of c.
   - Then all one-bit errors are detected (because then $e = x^k$, which is only divisible by x^l).
   - Then all such errors are detected where the failed bits are within a single n-bit window.
3. If $x+1 \mid g$ (or equivalently, g has an even number of 1 bits), then all errors with an odd number of bits are detected (because the multiples of g also have an even number of 1 bits).

Some frequently used CRC generator polynomials in practice:

- CRC-32: $g = 1\,00000100\,11000001\,00011101\,10110111$ (e.g. Ethernet, Wi-Fi, PNG);
- CRC-16: $g = 1\,10000000\,00000101$ (e.g. USB);
- CRC-1: $g = 11$ $(= x+1)$ – equivalent to the parity bit (e.g. HDDs and many other hardware).

7 Author

This document was written by Uray M. János, partly using the following sources:

[1] https://compalg.elte.gitlab-pages.hu/dimoa-web/tematika/dimoa (Hungarian)

[2] https://en.wikipedia.org/wiki/Cyclic_redundancy_check

[3] W.H. Press et all (2007): Numerical Recipes: The Art of Scientific Computing, Third Edition; chapter 22.4: Cyclic Redundancy and Other Checksums

Errors and suggestions can be reported here: uray.janos@inf.elte.hu.