Polinomosztás

Diszkrét modellek alkalmazásai

A korábbi jegyzetekben (Oszthatóság, prímszámok; Legnagyobb közös osztó) szó volt az egész számok osztásáról, oszthatóságáról és kapcsolódó tulajdonságokról. Ebben a jegyzetben ezek megfelelőiről lesz szó polinomokra. Sőt, sok esetben általánosabban, gyűrűkre fogjuk vizsgálni ezeket a fogalmakat (lásd: Algebrai struktúrák). Ezáltal egy csapásra intézzük el az egész számokat és a polinomokat, és lehetővé válik a más struktúrákra való kiterjesztés is.

1. Maradékos osztás

Ahogy az egész számok körében nem lehet általában osztani (pl. $2/3 \notin \mathbb{Z}$), úgy a polinomoknál sem (például $x/(x+1)$ nem polinom). Viszont ahogy az egész számoknál, úgy itt is bevezethetjük a **maradékos osztás** fogalmát (vö.: Oszthatóság, prímszámok, 1.1. Tétel):

1.1. Tétel (Maradékos osztás tétele polinomokra)**.** *Ha K test, $a, b \in K[x]$ és $b \neq 0$, akkor egyértelműen létezik olyan $q, r \in K[x]$, hogy $a = qb + r$ és* $\deg r < \deg b$.

1.2. Definíció. A fenti tételben q az a és b **hányadospolinomja**, r pedig a **maradékpolinomja**.

Példa. Ha $a = 2x^3 + x^2 - 2x - 1$ és $b = x^2 - 2x + 1$, akkor $q = 2x + 5$ és $r = 6x - 6$, mert $a = 2x^3 + x^2 - 2x - 1 = (2x + 5)(x^2 - 2x + 1) + (6x - 6)$ és $\deg(6x - 6) < \deg(x^2 - 2x + 1)$.

Kézzel nagyon hasonlóan lehet polinomokat maradékosan osztani, mint egész számokat, csak nem a számjegyek helyiértéke szerint haladunk, hanem a tagok fokszáma szerint. Az a-ból kiindulva, minden lépésben levonjuk belőle b-nek azt a többszörösét, hogy a főegyüttható éppen eltűnjön, ezáltal a maradék fokszáma kisebb legyen. Ezt addig folytatjuk, amíg a fokszám $\deg b$ alá nem csökken, és akkor megkaptuk a maradékot. A fenti példára ($a = 2x^3 + x^2 - 2x - 1$ és $b = x^2 - 2x + 1$):

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

ahol $\deg p_2 < \deg b$, ezért készen vagyunk, és a maradék $r := p_2 = 6x - 6$, a hányadost pedig a b szorzóiból állíthatjuk össze: $q = 2x + 5$.

Sage-ben a maradékos osztás ugyanúgy működik, mint egészekre (`//` és `%`):

```
sage: P.<x> = QQ[]    # ZZ[] nem jó!
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
```

Vigyázat: a sima osztást (/) itt is kerüljük, mert nem polinom típusú lesz az eredmény:

```
sage: a / b          # nem polinom az eredmény
(2*x^2 + 3*x + 1)/(x - 1)
sage: (b*x) / b      # akkor sem, ha annak tűnik!
x
sage: _.degree()     # nem működik, mert nem polinom típusú
[...]
AttributeError: '[...]' object has no attribute 'degree'
```

Fontos, hogy a maradékos osztás általában csak *test* fölött működik. Például $\mathbb{Z}[x]$-ben (ahol $\mathbb{Z}$ nem test) $a = x^2+3$ és $b = 2x$ hányadosa $q = 1/2 \cdot x$, maradéka $r = 3$, vagyis $\mathbb{Z}[x]$-ben nem működik, csak $\mathbb{Q}[x]$-ben, mert b főegyütthatójával kellett hozzá osztani. De más testet is használhatunk, például $\mathbb{Z}_5[x]$-ben $a = x^2 + \overline{3}$ és $b = \overline{2}x$ hányadosa $q = \overline{3}x$, maradéka $r = \overline{3}$ (hiszen $\mathbb{Z}_5$-ben $\overline{2}^{-1} = \overline{3}$).

Érdekes speciális eset, ha az osztó $b = x - c$ alakú, vagyis egy gyöktényező. Ekkor a maradékos osztás tétele miatt $p = (x - c)\, q + r$, ahol $\deg r < \deg b = 1$, azaz a maradék egy konstans polinom. Ebbe c-t behelyettesítve azt kapjuk, hogy $p(c) = 0\, q(c) + r = r$, vagyis a maradék nem más, mint p helyettesítési értéke c-ben. Speciálisan, ha c gyöke p-nek, akkor a maradék 0, azaz $p = (x - c)\, q$. Ha $p(c)$-t a Horner-módszerrel számítjuk ki (lásd: Polinomok alapjai), akkor belátható, hogy a q hányadospolinomot is automatikusan megkapjuk:

1.3. Tétel. *Ha $p \in K[x]$ és $c \in K$, akkor $p = a_n x^n + \ldots + a_1 x + a_0$ és $b = x - c$ maradéka a $p(c)$ konstans polinom, hányadosa pedig $q = b_{n-1} x^{n-1} + \ldots + b_1 x + b_0$, ahol a b_k-k a Horner-módszer algoritmusában szereplő b változó menet közbeni értékei, egészen pontosan a következő rekurzióval írhatók fel: $b_{n-1} := a_n$ és $b_k := b_{k+1} c + a_{k+1}$.*

Példa. $p = 2x^3 - 3x^2 - 4x + 7$ és $c = 2$ esetén a Horner-elrendezés szerint $p(c) = ((2c-3)c-4)c+7$, azaz $b_2 = 2$ ($= p$ főegyütthatója), $b_1 = b_2 c - 3 = 1$, $b_0 = b_1 c - 4 = -2$ és $b_{-1} = b_0 c + 7 = 3$, azaz $q = b_2 x^2 + b_1 x + b_0 = 2x^2 + x - 2$ és $r = p(c) = b_{-1} = 3$. Ugyanezt táblázatos formában is felírhatjuk, ahol az első sorba írjuk p-nek az együtthatóit, a második sorba pedig kiszámítjuk a b_k-kat:

	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. Oszthatóság

Ebben a szakaszban az egész számokra jól ismert oszthatóság ($b \mid a$) fogalmát terjesztjük ki gyűrűkre – egészen pontosan integritási tartományokra. Emlékeztető: a főbb integritási tartományok a $\mathbb{Z}$, a polinomgyűrűk ($K[x]$, ahol K integritási tartomány) és bármely test.

2.1. Definíció. Egy R integritási tartományban $a \in R$ **osztható** $b \in R$-rel (azaz b **osztója** a-nak), ha $\exists c \in R : a = bc$. Jelölés: $b \mid a$.

Példa. $\mathbb{Z}$-ben $2 \mid 6$, mert $6 = 2 \cdot 3$.

Példa. $\mathbb{R}[x]$-ben $2x^2 - x - 1$ osztható $x-1$-gyel ($x-1 \mid 2x^2 - x - 1$), mert $2x^2 - x - 1 = (x-1)(2x+1)$.

Példa. Testben a fogalom triviális, mert minden elem osztható minden nemnull elemmel, hiszen mindig létezik $c := b/a$. Például $\mathbb{Q}$-ban 5 osztható 2-vel, mert $5 = 2 \cdot (5/2)$.

Az oszthatóság polinomok esetén azért érdekes, mert következtetni lehet belőle a gyökökre: ha $b \mid a$, akkor b minden gyöke egyúttal a-nak is gyöke (és legalább annyiszor, mint b-nek). Például $x-1$-nek gyöke az 1, és $x-1 \mid 2x^2-x-1$, ezért ebből tudjuk, hogy $2x^2-x-1$-nek is gyöke az 1.

Akárcsak egészek esetén, polinomokra is könnyen eldönthető az oszthatóság a maradékos osztás segítségével: ha $b \neq 0$, akkor $b \mid a \iff a \bmod b = 0$.

Az oszthatóság általános tárgyalásához szükségünk lesz néhány új fogalomra:

2.2. Definíció. Az R integritási tartományban $u \in R$ **egység** [*unit*], ha osztója az egységelemnek.

Példa. $\mathbb{Z}$-ben két egység van: 1 és -1 (az 1 osztói).

Példa. $\mathbb{R}[x]$-ben minden nulladfokú polinom (azaz nemnulla konstans polinom) egység.

Példa. Testben minden elem egység, kivéve a nullelemet.

Példa. Egy általános $K[x]$ polinomgyűrűben az egységek a K egységei (mint konstans polinomok), pl. $\mathbb{Z}[x]$-ben 1 és -1, $\mathbb{Q}[x]$-ben pedig az összes nemnulla konstans.

Fontos: ne keverjük össze az *egység* [*unit*] és az *egységelem* [*identity element*] fogalmát: egységelemből csak egy van, egységből viszont, mint láttuk, több is lehet.

Az oszthatósággal kapcsolatban szükségünk lesz egy másik fontos fogalomra:

2.3. Definíció. Az R integritási tartományban $a \in R$ és $b \in R$ **asszociáltak** [*associates*], ha egymás osztói: $a \mid b$ és $b \mid a$.

Példa. $\mathbb{Z}$-ben az asszociáltak az azonos abszolút értékű számok, pl. 5 és -5.

Példa. $\mathbb{R}[x]$-ben az asszociáltak az egymás nemnulla konstansszorosai, például a következők mind asszociáltak: $x-1$, $2x-2$, $3x-3$, $-x+1$, $-x/2+1/2$ stb.

Példa. Testben bármely két elem asszociált, kivéve a nullelemet.

Vegyük észre, hogy az asszociáltak mindig egymás egységszeresei, pl. $\mathbb{Z}$-ben $-5 = -1 \cdot 5$, $\mathbb{R}[x]$-ben $2x-2 = 2 \cdot (x-1)$ stb. Az asszociáltaknak az a jelentősége, hogy oszthatóság szempontjából pontosan ugyanúgy viselkednek, pl. $2 \mid 6 \implies 2 \mid -6,\ -2 \mid 6,\ -2 \mid -6$. Ezt úgy is megfogalmazhatjuk, hogy az asszociáltak "lényegében" ugyanazok.

3. Irreducibilis elemek

Most következzen a *prímszám* fogalmának általánosítása:

3.1. Definíció. Egy R integritási tartományban $a \in R$ **irreducibilis** elem (vagy **felbonthatatlan**), ha nem a nullelem, nem egység, és nincs más osztója, csak az egységek és a-nak az asszociáltjai.

A "nem a nullelem, nem egység" annak felel meg, hogy a prímszámoknál kizártuk a 0-t és az 1-et, a "csak az egységek és a-nak az asszociáltjai" pedig annak, hogy "csak 1 és n az osztója".

Példa. $\mathbb{Z}$ irreducibilis elemei a prímszámok és azok ellentettjei: $2, 3, 5, 7, 11, \ldots, -2, -3, -5, -7, \ldots$. (A prímszámokat az egyszerűség kedvéért úgy definiáltuk, hogy pozitívak legyenek, azaz kiválasztottunk egy-egy kitüntetett elemet az asszociáltak közül. Ez azonban általában nem ilyen egyszerű.)

Példa. $\mathbb{Z}[x]$-ben az x^2-2 polinom irreducibilis, de $\mathbb{R}[x]$-ben nem az, mert $x^2-2 = (x-\sqrt{2})(x+\sqrt{2})$.

Példa. Testben nincsenek irreducibilis elemek, mert minden elem nullelem vagy egység.

Polinomgyűrűkben ($K[x]$) vizsgáljuk meg alaposabban is az irreducibilis polinomokat:

1. Test fölött minden elsőfokú polinom irreducibilis, hiszen $p = ax + b$ osztói csak konstans és elsőfokú polinomok lehetnek, az előbbiek egységek, az utóbbiak pedig p asszociáltjai.
2. Elsőnél magasabbfokú irreducibilis polinomnak nem lehet gyöke (az adott K alapgyűrűben), mert ha lenne, akkor kiemelhetnénk belőle egy gyöktényezőt, és akkor nem lenne irreducibilis.
3. $\mathbb{C}[x]$-ben *csak* az elsőfokú polinomok irreducibilisek, mert az algebra alaptétele miatt (Polinomok alapjai, 4.7. Tétel) minden nemkonstans polinomnak van gyöke.
4. $\mathbb{R}[x]$-ben csak első- és másodfokú polinomok lehetnek irreducibilisek (pl. $2x + 3$ és $x^2 + 1$).
5. $\mathbb{Z}[x]$-ben és $\mathbb{Q}[x]$-ben viszont az irreducibilis polinomok fokszáma tetszőlegesen nagy lehet, pl. ott minden $x^n - 2$ alakú polinom irreducibilis.

A prímszámok jelentőségét a számelmélet alaptétele mutatta meg (Oszthatóság, prímszámok, 3.7. Tétel), amely a prímtényezős felbontás egyértelműségéről szólt. Ezt közvetlenül nem tudjuk általánosítani, mert nem minden integritási tartományra fog teljesülni, csak egy részükre:

3.2. Definíció. Az R integritási tartomány **Gauss-gyűrű** [*unique factorization domain (UFD)*], ha a nullelem és az egységek kivételével minden elem egyértelműen felírható irreducibilis elemek szorzataként, ahol az egyértelműséget sorrendtől és asszociáltságtól eltekintve értjük.

Példa. $\mathbb{Z}$ a számelmélet alaptétele miatt Gauss-gyűrű. Például $-10 \in \mathbb{Z}$-t többféleképpen is fel lehet bontani: $-10 = (-2) \cdot 5 = 2 \cdot (-5) = 5 \cdot (-2) = (-5) \cdot 2$, de ezek mind csak sorrendben és asszociáltságban (azaz előjelban) térnek el, vagyis "lényegében" ugyanazok. (A számelmélet alaptételénél az asszociáltságról szóló kitételt megúsztuk, mert azt csak pozitív számokra mondtuk ki.)

Példa. Minden test triviálisan Gauss-gyűrű, hiszen nincs más elem, mint a nullelem és az egységek.

Példa. $\mathbb{R}[x]$ Gauss-gyűrű, pl. $x^2-1 = (x+1)(x-1) = (2x+2)(x/2-1/2) = (-3x+3)(-x/3-1/3)$ stb., de ezek mind csak sorrendben és asszociáltságban (azaz konstans szorzóban) térnek el egymástól.

Példa. Ha K Gauss-gyűrű, akkor $K[x]$ is Gauss-gyűrű, azaz $\mathbb{Z}[x]$, $\mathbb{Q}[x]$, $\mathbb{Z}_5[x]$ stb. mind Gauss-gyűrű.

Sage-ben az egész számokra megismert `factor()` függvény tetszőleges Gauss-gyűrűre működik, a következő kiegészítéssel: az irreducibilis tényezőket úgy választja meg (az asszociáltak közül), hogy a lehető legegyszerűbbek legyenek (pl. $\mathbb{Z}$-ben pozitívak), viszont ha így nem jönne ki a szorzat (mert pl. negatív számot bontunk fel), akkor a szorzat elejére kiemel egy egységet (pl. -1). A `factor()` eredménye egy `Factorization` típusú objektum, amely sorozatként viselkedik (pl. egy `for`-ciklussal felsorolható), de az esetleges egység ebben nem szerepel, hanem külön kell lekérdezni a `unit()` tagfüggvénnyel. Például:

```
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. Legnagyobb közös osztó

Most kiterjesztjük a legnagyobb közös osztó és a legkisebb közös többszörös fogalmát (lásd a Legnagyobb közös osztó c. jegyzetet) egészekről integritási tartományokra:

4.1. Definíció. Az R integritási tartományban $a \in R$ és $b \in R$ **legnagyobb közös osztója** $c \in R$, ha $c \mid a \ \wedge \ c \mid b \ \wedge \ \forall d \in R : d \mid a \ \wedge \ d \mid b \implies d \mid c$. Jelölés: $\text{lnko}(a, b)$ vagy $\gcd(a, b)$.
Továbbá a és b **relatív prím**, ha legnagyobb közös osztójuk egység.

4.2. Definíció. Az R integr. tartományban $a \in R$ és $b \in R$ **legkisebb közös többszöröse** $c \in R$, ha $a \mid c \ \wedge \ b \mid c \ \wedge \ \forall d \in R : a \mid d \ \wedge \ b \mid d \implies c \mid d$. Jelölés: $\text{lkkt}(a, b)$ vagy $\text{lcm}(a, b)$.

Példa. $\mathbb{Z}$-ben 12 és 20 legnagyobb közös osztója 4, legkisebb közös többszöröse 60.

Példa. $\mathbb{R}[x]$-ben a $p = x^2 - 2x + 1 = (x-1)^2$ és a $q = 2x^2 - x - 1 = (x-1)(2x+1)$ polinomok legnagyobb közös osztója $x - 1$, legkisebb közös többszöröse pedig $(x-1)^2(2x+1) = 2x^3 - 3x^2 + 1$.

De ezekkel a fogalmakkal ilyen általánosan van néhány probléma. Például nem biztos, hogy léteznek. Szerencsére azonban a számunkra fontos speciális esetekben garantálható a létezés:

4.3. Tétel. *Gauss-gyűrűben bármely két elemnek van legnagyobb közös osztója és legkisebb közös többszöröse.*

Tehát $\mathbb{Z}$-ben, $\mathbb{Z}[x]$-ben, $\mathbb{R}[x]$-ben stb. mindig létezik lnko és lkkt. (A tétel nem meglepő, gondoljunk csak arra, hogy $\mathbb{Z}$-ben hogyan lehet ezeket kiszámítani a prímtényezős felbontás segítségével.)

De van egy másik probléma is: az lnko és az lkkt nem egyértelmű!

Példa. $\mathbb{Z}$-ben $\text{lnko}(6, 4)$ fenti definícióját kielégíti 2 és -2 is. Az eredeti definícióban (Legnagyobb közös osztó, 1.1. Definíció) is csak azért volt egyértelműség, mert kikötöttük, hogy az eredménynek $\mathbb{N}$-ben kell lennie.

Példa. A fenti $p = x^2 - 2x + 1$ és $q = 2x^2 - x - 1$ polinomoknak, ha $\mathbb{R}[x]$-ben nézzük, legnagyobb közös osztója az $r_1 = x - 1$, az $r_2 = -x + 1$, az $r_3 = 2x - 2$, az $r_4 = -x/2 + 1/2$ stb. (De ha $\mathbb{Z}[x]$-ben nézzük, akkor csak r_1 és r_2 az, mert ott $r_3 \nmid p$ és $r_4 \notin \mathbb{Z}[x]$.)

Ha teljes egyértelműség nem is, de "bizonyos értelemben" való egyértelműség fennáll:

4.4. Tétel. *A legnagyobb közös osztó és a legkisebb közös többszörös, ha létezik, asszociáltságtól eltekintve egyértelmű.*

Példa. $\mathbb{Z}$-ben az lnko *előjeltől eltekintve* egyértelmű: $\text{lnko}(6, 4)$ lehet 2 és -2 is, de más nem. Ha pedig leszűkítjük $\mathbb{N}$-re, akkor teljesen egyértelmű lesz.

Példa. $\mathbb{R}[x]$-beli polinomoknál az lnko *konstans szorzótól eltekintve* egyértelmű: a fenti példában $r_1 = -r_2 = 1/2 \cdot r_3 = -2r_4$. A teljes egyértelműséghez pedig például kiköthetjük, hogy a főegyüttható 1 legyen (pl. akkor $r_1 = x - 1$ az "igazi" lnko). (De $\mathbb{Z}[x]$-ben ez nem mindig működik.)

Most vizsgáljuk meg, hogy miért érdekes a polinomok legnagyobb közös osztója:

4.5. Tétel. *Ha $p \in K[x]$ és $q \in K[x]$ legnagyobb közös osztója $r \in K[x]$, akkor r gyökei pontosan a p és q közös gyökei, és r-beli multiplicitásuk a p-beli és a q-beli multiplicitásaik közül a kisebb.*

Példa. A fenti $p = x^2 - 2x + 1$ egyetlen gyöke az 1 (de kétszeres); $q = 2x^2 - x - 1$ gyökei 1 és $-1/2$ (egyszeresek); ezért $r = \text{lnko}(p, q)$-nak (vagy bármelyik r_i-nek) egyetlen gyöke az 1 (és az egyszeres).

Vagyis ha két polinom közös gyökeit szeretnénk meghatározni, akkor elég csak az lnko-juk gyökeit kiszámítani, amely általában alacsonyabbfokú, mint az eredetiek, ezért könnyebb vele számolni; ha pedig konstans, azaz a két polinom relatív prím, akkor rögtön tudjuk, hogy nem lehet közös gyökük.

Polinomok legnagyobb közös osztóját ugyanúgy tudjuk kiszámítani, mint egész számokét: az euklideszi algoritmussal (lásd a Legnagyobb közös osztó c. jegyzetet), hiszen polinomokat is lehet maradékosan osztani (lásd ezen jegyzet elejét).

Példa. Euklideszi algoritmus a $p = x^4 - 4x^3 + 1$ és $q = x^3 - 3x^2 + 1$ polinomokra:

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

vagyis $\text{lnko}(p, q) = -27/256$. De, mint láttuk, az lnko nem egyértelmű, ezért vehetjük bármely konstansszorosát is, pl. $\text{lnko}(p, q) = -27$ és $\text{lnko}(p, q) = 1$ is igaz. (Általában, ha konstans az eredmény, akkor bármilyen más nemnulla konstans is jó.) Ez azt jelenti, hogy p és q relatív prímek, tehát nem lehet közös gyökük. Megjegyzés: mivel az eredményben úgysem számít a konstans szorzó, ezért akár menet közben is egyszerűsíthetjük a polinomokat egy-egy konstans szorzóval, pl. r_3 helyett számolhatunk az egyszerűbb $r_3' := 9r_3 = 16x - 11$ polinommal, és így tovább.

Fontos: az euklideszi algoritmus általában csak *test* fölötti polinomokra működik, pl. $\mathbb{R}[x]$-ben vagy $\mathbb{Q}[x]$-ben. Nem működik viszont $\mathbb{Z}[x]$-ben, mert törtekkel kell számolni (ahogy a fenti példában is). Ez annak ellenére igaz, hogy maga a legnagyobb közös osztó $\mathbb{Z}[x]$-ben is létezik, például fent $\text{lnko}(p, q) = 1 \in \mathbb{Z}[x]$ – de ehhez először $\mathbb{Q}[x]$-ben kellett kiszámítani egy másik lnko-t: $-27/256$.

Nemcsak a normál euklideszi algoritmus, hanem a bővített verziója is működik polinomokra, és pontosan ugyanúgy, mint egészekre. Ezt a következő tétel biztosítja:

4.6. Tétel (Bézout-lemma polinomokra)**.** *Ha K test, és $p \in K[x]$ és $q \in K[x]$ legnagyobb közös osztója $r \in K[x]$, akkor létezik olyan $u, v \in K[x]$, hogy $pu + qv = r$.*

Példa. $p = x^3 - 3x^2 + 3x - 1 = (x - 1)^3$ és $q = 2x^2 - x - 1 = (x - 1)(2x + 1)$ legnagyobb közös osztója $r = x - 1$, és valóban: $p \cdot 4/9 + q \cdot (-2/9 \cdot x + 5/9) = r$, azaz $u = 4/9$ és $v = -2/9 \cdot x + 5/9$.

Sage-ben polinomokra ugyanúgy működik a `gcd(p, q)` és az `xgcd(p, q)`, mint egészekre. (Azt leszámítva, hogy itt `gcd(p, q)` nem az egyetlen helyes megoldás, hiszen az lnko nem egyértelmű.)

5. Algebrai derivált

Analízisből ismerjük a **derivált** fogalmát, amelynek polinomokra is hasznos tulajdonságai vannak. Itt azonban nem szeretnénk az analízis eszköztárát (pl. határérték, folytonosság) használni, bizonyos esetekben pedig nem is tudjuk (mert a *diszkrét* struktúrákban, mint $\mathbb{Z}$ vagy $\mathbb{Z}_m$, nincs folytonosság). Ezért ahelyett, hogy *levezetnénk* a deriváltat analízisből, inkább egyszerűen csak *definiáljuk* (és *algebrai* deriváltnak nevezzük, arra utalva, hogy analízis nélkül is működik):

5.1. Definíció. Egy K integritási tartomány fölötti

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

polinom **algebrai deriváltja** a következő polinom:

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

ahol a ka_k szorzat a k-tagú $a_k + a_k + \ldots + a_k$ összeget jelenti.

Példa. A $p = x^3 + 2x^2 - 3x + 5 \in \mathbb{Z}[x]$ polinom deriváltja $p' = 3x^2 + 4x - 3$.

Megjegyzés: a fenti definícióban a ka_k szorzat jelentését azért kellett megadni, mert az a_k együtthatók nem feltétlenül számok, hanem egy tetszőleges K integritási tartományból vannak, azt pedig eddig nem definiáltuk, hogy azokat hogyan kell megszorozni egy számmal (a kitevőben ugyanis mindig egész számok vannak, bármilyen gyűrűből is vannak az együtthatók). Például $\mathbb{Z}_3[x]$-ben $p = \overline{2}x^5$ algebrai deriváltja $p' = 5 \cdot \overline{2}x^4 = (\overline{2}+\overline{2}+\overline{2}+\overline{2}+\overline{2})x^4 = \overline{1}x^4 = x^4$.

További meglepetéseket okozhat, ha a polinomok együtthatói nem számok. A definícióból ugyanis úgy tűnik, hogy egy n-edfokú polinom deriváltja $n-1$-edfokú. Ez valós polinomokra ($\mathbb{R}[x]$) igaz is, de $\mathbb{Z}_m[x]$-ben nem feltétlenül: pl. a $p = x^3 + \overline{2}x^2 + \overline{2}x + \overline{1} \in \mathbb{Z}_3[x]$ harmadfokú polinom deriváltja csak elsőfokú: $p' = x + \overline{2}$, ugyanis p' másodfokú együtthatója $\mathbb{Z}_3$-ban eltűnt: $3 \cdot \overline{1}x^2 = (\overline{1}+\overline{1}+\overline{1})x^2 = \overline{0}x^2$. Sőt, az is előfordulhat, hogy a teljes polinom eltűnik, pl. $p = x^6 + x^3 + \overline{1} \in \mathbb{Z}_3[x]$ deriváltja $p' = \overline{0}$.

Sage-ben egy polinom deriváltját a `p.diff()` tagfüggvénnyel kérdezhetjük le.

Mire jó egy polinom deriváltja? Erre a következő tétel ad választ:

5.2. Tétel. *Ha $c \in K$ a $p \in K[x]$-nek k-szoros gyöke ($k \geq 1$), akkor p'-nek $k-1$-szeres gyöke.*

Példa. $p = x^3 - 4x^2 - 3x + 18 = (x+2)(x-3)^2$-nek $c = 3$ kétszeres gyöke, $p' = 3x^2 - 8x - 3 = (x-3)(3x+1)$-nek pedig egyszeres. (Továbbá p-nek $c = -2$ egyszeres gyöke, p'-nek pedig nem gyöke, azaz "nullaszoros" gyöke.)

Vagyis a derivált megőrzi az eredeti polinom *többszörös* gyökeit (eggyel alacsonyabb multiplicitással). Ezt kicsit megzavarja, hogy p'-nek lehetnek új gyökei is, mint a fenti példában $c = -1/3$. De ezeket könnyen kiszűrhetjük, ha vesszük $\mathrm{lnko}(p, p')$-t, ami csak a közös gyököket tartalmazza.

Példa. A fenti példát folytatva $\mathrm{lnko}(p, p') = x - 3$, amelynek $c = 3$ egyszeres gyöke, és más gyöke nincs. Ez tisztán jelzi, hogy p-nek egyetlen többszörös gyöke van, a $c = 3$.

Ha tehát egy p polinomnak csak a többszörös gyökeire vagyunk kíváncsiak, akkor elég csak az $\mathrm{lnko}(p, p')$ gyökeit kiszámítani, amely könnyebb, mert alacsonyabbfokú, és az lnko-t hatékonyan ki tudjuk számítani az euklideszi algoritmussal. Sőt, ha $\mathrm{lnko}(p, p')$ konstans, akkor rögtön tudjuk, hogy p-nek nem lehetnek többszörös gyökei, anélkül, hogy bármilyen gyököt ki kellett volna számítani.

Ha pedig leosztjuk a polinomot $\mathrm{lnko}(p, p')$-vel: $p \,/\, \mathrm{lnko}(p, p')$ (amely szintén polinom, hiszen $\mathrm{lnko}(p, p') \mid p$), akkor ennek pontosan ugyanazok a gyökei, mint p-nek, csak mindegyik egyszeres (hiszen minden $(x-c)^k$-ból leosztottunk $(x-c)^{k-1}$-t). Ez azért hasznos, mert ha szeretnénk p gyökeit kiszámítani, akkor érdemes először leosztani $\mathrm{lnko}(p, p')$-vel, mert így a gyökök nem változnak,

de csökkenthetjük a polinom fokszámát. Ezt az eljárást **négyzetmentesítés**nek nevezzük.

Megjegyzés: a négyzetmentesítés számpolinomokra (pl. $\mathbb{R}[x]$-ben) tökéletesen működik, de más struktúrákban, pl. $\mathbb{Z}_m[x]$-ben problémákba ütközhet. Például fent láttuk, hogy $p = x^6 + x^3 + \overline{1} \in \mathbb{Z}_3[x]$ deriváltja eltűnik: $p' = \overline{0}$. Ez azért baj, mert ilyenkor $p \,/\, \text{lnko}(p, p') = p/p = 1$, vagyis nem tudjuk kiszűrni a többszörös gyököket. Pedig lenne mit kiszűrni: $p = x^6 + x^3 + \overline{1} = (x - \overline{1})^6$, azaz van egy hatszoros gyöke, $c = \overline{1}$.

Most nézzük meg, hogyan tudjuk hatékonyan kiértékelni egy polinom deriváltját egy adott helyen.

A Polinomok alapjai c. jegyzetben megismertük a Horner-módszert, amellyel hatékonyan ki tudunk értékelni egy p polinomot egy c helyen: $p(c)$. Ha szükségünk van a derivált helyettesítési értékére, $p'(c)$-re is, akkor azt nyilván kiszámíthatjuk ugyanígy p'-ből. Ehhez azonban szükségünk van a p' polinomra, amelynek kiszámítása további szorzásokat (és memóriát) igényel. De ezt megúszhatjuk: a Horner-módszer kiterjeszthető úgy, hogy a $p(c)$-vel együtt kiszámítsa $p'(c)$-t is, anélkül, hogy közben kiszámítaná a p' polinomot. Erre a következő tétel ad lehetőséget:

5.3. Tétel. *Ha $p \in K[x]$, $c \in K$ és p-nek az $x-c$-vel vett maradékos osztásának hányadosa $q \in K[x]$, azaz $p = (x - c)\, q + p(c)$, akkor $p'(c) = q(c)$.*

Ez azért jó, mert a Horner-módszer automatikusan kiszámítja q-t (lásd fent a maradékos osztásnál), tehát abból egy újabb Horner-módszerrel kiszámíthatjuk $q(c)$-t, azaz $p'(c)$-t is. Ráadásul ezt a két lépést egyszerre is végezhetjük, azaz nem kell eltárolni a teljes q-t. (Ennek a műveletigénye összesen kb. $2n$ szorzás és $2n$ összeadás, míg a p' kiszámítása további n szorzást igényelne.)

Példa. $p = x^4 - 3x^3 + 6x + 3$ és $c = 2$ esetén a Horner-módszer két lépése így írható fel:

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

Dupla Horner-módszer:

Bemenet: a p polinom $a_0, a_1, \ldots, a_n$ együtthatói és a c érték
Kimenet: $(p(c), p'(c))$
$b := a_n$
$d := 0$
ciklus $k = n-1$-*től* 0-*ig*
  $d := d \cdot c + b$
  $b := b \cdot c + a_k$
$\rightarrow$ *(b, d)*

6. Cyclic redundancy check (CRC)

Amikor adatot továbbítunk egy fizikai csatornán keresztül (például vezetékben vagy rádiójelekkel), akkor előfordulhatnak adatátviteli hibák, azaz hogy nem pontosan ugyanaz az adat érkezik meg, mint amit elküldtünk. Hogyan tudjuk észrevenni az ilyen hibákat?

Ehhez *redundanciát* használunk: az üzenetet kiegészítjük egy ún. ellenőrző kóddal, amely új információt nem tartalmaz, de segít ellenőrizni, hogy az átvitt adat helyes-e. Ha ugyanis a továbbítás után az ellenőrző kód nem felel meg az üzenetnek, akkor biztosak lehetünk benne, hogy vagy az üzenet, vagy az ellenőrző kód (vagy mindkettő) hibás.

Erre egy egyszerű példa a **paritásbit**: a küldendő üzenetet, ami egy bitsorozat, egy új bittel egészítjük ki a következőképpen: ha az üzenetben páros számú 1-es bit volt, akkor 0-t írunk a végére, ha pedig páratlan, akkor 1-et. Így a végeredményben mindenképpen páros számú 1-es bit lesz. Például: $0101 \to 01010$, ill. $1101 \to 11011$. Ha a csatorna másik végén azt kapjuk, hogy 11010, akkor az biztosan hibás, mert páratlan sok 1-est tartalmaz. De hogy hol a hiba, azt nem tudjuk, például a fenti két bitsorozat bármelyikétől csak egy bitben tér el. Ezzel a módszerrel mindig észrevesszük, ha egyetlen bit romlik el, de kettőt már nem (mert attól páros marad az 1-esek száma).

A paritásbitnél jobb megoldást ad hibadetektálásra a **Cyclic Redundancy Check (CRC)**.

A CRC a bitsorozatokat polinomokként kezeli, ahol minden együttható egy-egy bit (0 vagy 1). Például az 101011 bitsorozat az $x^5 + x^3 + x + 1$ polinomnak felel meg (az utolsó bit a konstans tag, az azelőtti az elsőfokú tag, és így tovább). A polinomok $\mathbb{Z}_2[x]$-ben vannak, azaz minden számítás modulo 2 történik, pl. $1+1=0$, $0-1=1$ stb. Ebből következik, hogy bármely polinomra $p+p=0$ (pl. $(x^2+1)+(x^2+1)=(1+1)x^2+(1+1)=0$), tehát $p=-p$, ezért az összeadás megegyezik a kivonással: $p-q=p+(-q)=p+q$. Ez az összeadás/kivonás leírható a bitenkénti "kizáró vagy" (XOR, $\oplus$) logikai művelettel is (pl. $1+1=0 \longrightarrow$ "igaz" $\oplus$ "igaz" = "hamis").

A CRC fő művelete a polinomok maradékos osztása $\mathbb{Z}_2[x]$-ben. Az osztó egy rögzített $g \in \mathbb{Z}_2[x]$ polinom, a CRC **generátorpolinom**ja. Ha $n := \deg g$ (azaz g bitsorozata $n+1$ bitből áll), akkor ***n*-bites CRC**-ről beszélünk (jelölés: CRC-n), mert g-vel osztva a maradék belefér n bitbe (hiszen legfeljebb $n-1$-edfokú).

CRC üzenetküldés:

Bemenet: $m \in \mathbb{Z}_2[x]$, a továbbítandó üzenet *(amit bitsorozatként kezelünk)*
Számítsuk ki az mx^n polinomot *(azaz írjunk m végére n darab nullát)*
$c := mx^n \bmod g$ *(ahol c az ellenőrző kód, "check")*
Elküldött **kódszó**: $mx^n + c$ *(azaz írjuk m végére az n-bites c-t)*

CRC ellenőrzés (1. módszer):

Bemenet: $mx^n + c \in \mathbb{Z}_2[x]$, a megkapott kódszó (lásd fent)
Bontsuk fel a kódszót m-re és c-re *(azaz válasszuk le az utolsó n bitjét)*
$c' := mx^n \bmod g$
ha $c' \neq c \to$ *hibás*

CRC ellenőrzés (2. módszer):

Bemenet: $mx^n + c \in \mathbb{Z}_2[x]$, a megkapott kódszó (lásd fent)
$e := (mx^n + c) \bmod g$
ha $e \neq 0 \to$ *hibás*

A két ellenőrző módszer ekvivalens, mert ha mx^n maradéka c, azaz $mx^n = qg + c$, akkor $mx^n - c = qg$; de $\mathbb{Z}_2[x]$-ben a kivonás ugyanaz, mint az összeadás, ezért $mx^n - c = mx^n + c$, vagyis $g \mid mx^n + c$, tehát ekkor $mx^n + c$ valóban nullát ad maradékul g-vel osztva.

Példa. Legyen a generátor $g = x^4 + x + 1$, bitsorozatként $g = 10011$ (ekkor $n = 4$), és legyen az üzenet $m = 11000101$. A küldéshez előállítjuk az $mx^n = 11000101|0000$ polinomot (azaz m-hez hozzáírunk négy nullát), és ezt elosztjuk maradékosan g-vel (a lenti bal oldali osztás). A maradék $c = 110$, n-bitesre kiegészítve $c = 0110$, ezt hozzáfűzzük az üzenet végére: $mx^n + c = 11000101|0110$,

és ez lesz a kódszó, amit elküldünk. A fogadó fél ezt úgy ellenőrzi, hogy elosztja g-vel (a lenti jobb oldali osztás), és mivel a maradék 0, ezért az üzenetet helyesnek találja. (Még egyszer: ezek nem bináris számok, hanem polinomok $\mathbb{Z}_2[x]$-ben, tehát a kivonás valójában "kizáró vagy" (XOR).)

Küldés ($mx^n \bmod g$):

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

Ellenőrzés ($mx^n + c \bmod g$):

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

A CRC előnye, hogy ezeket a polinomműveleteket egy számítógép nagyon hatékonyan el tudja végezni, mert csak bitsorozatokat kell manipulálni, amiben a számítógépek amúgy is jók.

Sajnos elkerülhetetlen, hogy az ellenőrzés néha hibás üzeneteket is helyesnek találjon. A cél olyan g generátorpolinom keresése, hogy ez minél ritkábban fordulhasson elő. A hiba azt jelenti, hogy az elküldött $mx^n + c$ kódszó helyett $mx^n + c + e$ érkezik meg, ahol e a **hibapolinom**, azaz amelyben azok az együtthatók 1-esek, amely bitek a küldés során megváltoztak. Ha nem vesszük észre a hibát, az azért van, mert $g \mid mx^n + c + e$ (ekkor lesz 0 a maradék), következésképp $g \mid e$ (mivel a helyes kódszóra is $g \mid mx^n + c$). A jó g tehát olyan, hogy a gyakran előforduló hibamintázatokra $g \nmid e$.

A generátorpolinom (g) kiválasztásának néhány szempontja:

1. A fokszáma (n) megegyezik az ellenőrzőbitek számával, ezért minél nagyobb, annál kisebb lehet a tévedés valószínűsége (véletlen bitsorozatot $1/2^n$ valószínűséggel talál helyesnek).
2. A generátorpolinom konstans tagja (utolsó bitje) legyen 1, mert:
   - Ha 0 lenne, akkor $x \mid g$, ezért a $c = mx^n \bmod g$ utolsó (néhány) bitje is mindig 0 lenne, vagyis annál kevesebb értékes bitje maradna.
   - Így minden egybites hiba felismerhető (mert akkor $e = x^k$, amit csak $g = x^l$ oszthatna).
   - Így minden olyan hiba felismerhető, ahol a hibás bitek egyetlen n-hosszú részben vannak.
3. Ha $x+1 \mid g$ (ami ekvivalens azzal, hogy g-nek páros sok 1-es bitje van), akkor minden páratlan sok bitből álló hiba felismerhető (mert g többszöröseinek is páros sok 1-es bitjük lesz).

A gyakorlatban használt néhány CRC generátorpolinom:

- CRC-32: $g = 1\,00000100\,11000001\,00011101\,10110111$ (pl. Ethernet, Wi-Fi, PNG);
- CRC-16: $g = 1\,10000000\,00000101$ (pl. USB);
- CRC-1: $g = 11$ $(= x + 1)$ – ez ekvivalens a paritásbittel (pl. HDD-k és számos más hardver).

7. Szerző

Ezt a jegyzetet Uray M. János írta, részben felhasználva az alábbi forrásokat:

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

[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

Hibákat, észrevételeket itt lehet jelezni: uray.janos@inf.elte.hu.