Polinomműveletek

Diszkrét modellek alkalmazásai

Az előző jegyzetben definiáltuk a polinomok alapfogalmait. Ebben a jegyzetben arról lesz szó, hogy hogyen lehet polinomokon különböző műveleteket végezni – akár kézzel, akár programmal. A műveletek alatt elsősorban az egész számoknál is ismert alapműveleteket értjük: összeadás, kivonás, szorzás és maradékos osztás, de lesz szó kifejezetten polinomokra érvényes műveletekről is.

A műveleteknél azt feltételezzük, hogy a bemenő polinomok kanonikus alakban vannak megadva (pl. $p = 2x^3 - 6x + 4$, lásd az előző jegyzetet), és az eredményt is ilyen alakban szeretnénk előállítani. Számítógépes ábrázoláshoz elég az együtthatók sorozatát tárolni (a kihagyott tagok helyén 0-kkal), tipikusan a konstans tagtól kezdve (pl. $p = 2x^3-6x+4$ esetén: $4, -6, 0, 2$), hogy a k-adfokú együttható éppen a lista k-adik eleme legyen (ha 0-tól indexelünk).

1. Összeadás, kivonás

A legegyszerűbb polinomműveletek közé tartozik az összeadás ($p+q$) és a kivonás ($p-q$). Ezekhez semmi mást nem kell tenni, mint az azonos fokszámú tagokat összeadni, ill. kivonni. Ha kézzel számolunk, akkor célszerű egymás alá írni a megfelelő tagokat. (Csak arra figyeljünk, hogy a hiányzó tagoknak hagyjunk ki helyet, nehogy véletlenül rossz taghoz adjuk hozzá.)

1. Példa. $\mathbb{Z}[x]$-ben $p = 2x^3 + x^2 - 5$ és $q = 2x^2 - 2x + 1$ összege:

$$\begin{array}{rrrrr} & 2x^3 & +\,x^2 & & -\,5 \\ + & & 2x^2 & -\,2x & +\,1 \\ \hline & 2x^3 & +\,3x^2 & -\,2x & -\,4 \end{array}$$

2. Példa. $\mathbb{Z}[x]$-ben $p = x^3 + 2x^2 + x$ és $q = x^3 + 3x^2 - 2x + 4$ különbsége:

$$\begin{array}{rrrrrr} & x^3 & +\,2x^2 & +\,x & & \\ -(& x^3 & +\,3x^2 & -\,2x & +\,4 &) \\ \hline & & -\,x^2 & +\,3x & -\,4 & \end{array}$$

3. Példa. Az együtthatók nemcsak számok, hanem $\mathbb{Z}_m$-ből is lehetnek (ahol modulo m kell számolni, lásd: Kongruenciák). $\mathbb{Z}_5[x]$-ben $p = \overline{2}x^4 + x^3 + \overline{4}x^2 + \overline{3}$ és $q = \overline{3}x^4 + \overline{4}x^3 + \overline{2}x^2 + \overline{2}x + \overline{1}$ összege:

$$\begin{array}{rrrrrr} & \overline{2}x^4 & +\,x^3 & +\,\overline{4}x^2 & & +\,\overline{3} \\ + & \overline{3}x^4 & +\,\overline{4}x^3 & +\,\overline{2}x^2 & +\,\overline{2}x & +\,\overline{1} \\ \hline & & & x^2 & +\,\overline{2}x & +\,\overline{4} \end{array}$$

Írjuk fel formálisan is a két művelet szabályát, azaz hogy hogyan lehet kiszámítani az eredmény együtthatóit a bemenő polinomok együtthatóiból. Legyen p_k a p polinom k-adfokú együtthatója (ha nem létezik, mert $k > \deg p$, akkor legyen 0), és hasonlóan q-ra, $(p + q)$-ra stb. is. Ekkor:

$$(p+q)_k := p_k + q_k,$$
$$(p-q)_k := p_k - q_k.$$

1.1. Tétel (összeg és különbség fokszáma)**.** *Ha* $p, q \in K[x]$*, akkor*

$$\deg(p+q) \le \max(\deg p, \deg q),$$
$$\deg(p-q) \le \max(\deg p, \deg q),$$

és ha $\deg p \neq \deg q$*, akkor egyenlőség (=) áll fenn mindkét esetben.*

Példa. A fenti első két példában $\deg(p+q) = 3 = \max(3,2)$ és $\deg(p-q) = 2 < \max(3,3) = 3$.

Vagyis az eredmény fokszáma nem lehet nagyobb, mint a magasabbfokú polinomé. Kisebb viszont lehet, ha a két fokszám egyenlő, és a főegyütthatók éppen kiesnek (mint a fenti 2. és 3. példában).

Műveletigény: két n-edfokú polinom összeadásához vagy kivonásához legfeljebb $n+1$ elemi összeadást vagy kivonást kell elvégezni, tehát ezek a műveletek *lineáris* időben elvégezhetőek.

A polinomműveletek nagyon hasonlítanak a többszámjegyű egész számok megfelelő műveleteire. Például:

$$\begin{array}{r} 314 \\ +\ 271 \\ \hline 585 \end{array} \qquad \begin{array}{r} 31415 \\ +\ 2717 \\ \hline 34132 \end{array} \qquad \begin{array}{r} 31415 \\ -\ 2717 \\ \hline 28698 \end{array} \qquad \begin{array}{r} 98765 \\ +\ 1235 \\ \hline 100000 \end{array}$$

Ez azért van így, mert az egész számok felírhatók a számjegyeiknek polinomjaként $x = 10$-zel, pl. $314 = 3\cdot 10^2 + 1\cdot 10 + 4$. Az első összeadás egy az egyben le is fordítható polinomokra: $(3x^2 + x + 4) + (2x^2 + 7x + 1) = 5x^2 + 8x + 5$.

A többi azonban nem. Az egész számok összeadásánál ugyanis lehet *átvitel*, azaz ha egy számjegy eredménye nem fér bele 0 és 9 közé, akkor az az előző számjegyet is módosítani fogja. Például a második összeadás utolsó számjegyénél $5 + 7 = 12$, amelyből csak a 2-es lett az utolsó számjegy, az 1-est pedig (azaz az átvitelt) hozzáadtuk az előző számjegyhez ($2 \to 3$). Ha ezek polinomok lennének, akkor az utolsó együttható maradna 12, az előző pedig 2.

Az átvitel nemcsak az előző számjegyet befolyásolhatja, hanem tovább is gyűrűzhet. Például a jobb szélső összeadásnál az utolsó számjegy átvitele továbbgyűrűzött az összes többi számjegyre. Sőt, ez arra is példa, hogy – a polinomokkal ellentétben – összeadáskor hosszabb számot is kaphatunk, mint a hosszabbik bemenő szám.

2. Szorzás

Két polinom összeszorzása már egy bonyolultabb művelet, mint az összeadás és a kivonás: minden tagot minden másik taggal össze kell szorozni, és végül a megfelelő tagokat összeadni.

Példa. $p = x^3 + 2x^2 - 3x + 4 \in \mathbb{Z}[x]$ és $q = 2x^2 - x + 3 \in \mathbb{Z}[x]$ szorzatát úgy számítjuk ki, hogy p mind a négy tagját megszorozzuk q mind a három tagjával. Táblázatos formában:

$(\cdot)$	x^3	$2x^2$	$-3x$	4
$2x^2$	$2x^5$	$4x^4$	$\boldsymbol{-6x^3}$	$8x^2$
$-x$	$-x^4$	$\boldsymbol{-2x^3}$	$3x^2$	$-4x$
3	$\boldsymbol{3x^3}$	$6x^2$	$-9x$	12

Ezután össze kell adni a táblázatban szereplő 12 tagot, kitevőnként csoportosítva (vegyük észre, hogy az azonos kitevőjű tagok átlósan helyezkednek el, mint például a fent vastagon kiemelt x^3-tagok):

$$p \cdot q = 2x^5 + (4x^4 - x^4) + (-6x^3 - 2x^3 + 3x^3) + (8x^2 + 3x^2 + 6x^2) + (-4x - 9x) + 12 =$$
$$= 2x^5 + 3x^4 - 5x^3 + 17x^2 - 13x + 12.$$

Formálisan így írhatjuk fel a szorzatpolinom egyes együtthatóit:

$$(p \cdot q)_k = \sum_{i=0}^{k} p_i q_{k-i} = p_0 q_k + p_1 q_{k-1} + \ldots + p_{k-1} q_1 + p_k q_0.$$

Példa. A fenti példában $p \cdot q$ harmadfokú tagjának $(-5x^3)$ együtthatóját így számítjuk ki:

$$(p \cdot q)_3 = p_0 q_3 + p_1 q_2 + p_2 q_1 + p_3 q_0 = 4 \cdot 0 + (-3) \cdot 2 + 2 \cdot (-1) + 1 \cdot 3 = -6 - 2 + 3 = -5.$$

Ha írásban számolunk, akkor nem a fenti táblázotos formát célszerű használni, hanem úgy írjuk fel a sorokat, hogy az azonos fokszámú tagok éppen egymás alá kerüljenek, megkönnyítve az összeadást.

Példa. A fenti p és q polinomok szorzatát írásban így számíthatjuk ki:

$$\begin{array}{rr} & (\ x^3 + 2x^2 - 3x + 4\ ) \cdot (2x^2 - x + 3) \\ \hline & 2x^5 + 4x^4 - 6x^3 + 8x^2 \\ & \quad -x^4 - 2x^3 + 3x^2 - 4x \\ + & \quad\quad 3x^3 + 6x^2 - 9x + 12 \\ \hline & 2x^5 + 3x^4 - 5x^3 + 17x^2 - 13x + 12 \end{array}$$

2.1. Tétel (szorzat fokszáma)**.** *Legyen $p, q \in K[x]$. Ha K integritási tartomány, akkor*

$$\deg(p \cdot q) = \deg p + \deg q,$$

ellenkező esetben (azaz ha vannak benne nullosztók):

$$\deg(p \cdot q) \leq \deg p + \deg q.$$

(Emlékeztető: integritási tartomány minden ismert számhalmaz: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, valamint prímszámokra $\mathbb{Z}_p$, lásd az Algebrai struktúrák c. jegyzetet.)

Példa. A fenti példában $\deg p = 3$ és $\deg q = 2$, és valóban: $\deg(p \cdot q) = 5 = 3 + 2$.

Példa. $\mathbb{Z}_6$ nem integritási tartomány (pl. nullosztók: $\overline{2} \cdot \overline{3} = \overline{0}$), ezért $\mathbb{Z}_6[x]$-ben csak "$\leq$" teljesül:

$$p = \overline{2}x + \overline{1},\ q = \overline{3}x + \overline{1} \quad \Longrightarrow \quad p \cdot q = \overline{5}x + \overline{1} \quad \Longrightarrow \quad \deg(p \cdot q) = 1 \leq 1 + 1.$$

Ez azért van így, mert a nullosztók miatt a másodfokú tag eltűnik: $\overline{2}x \cdot \overline{3}x = \overline{0}x^2$.

Példa. A tétel a nullpolinomra is igaz (amelyre $\deg 0 = -\infty$, lásd az előző jegyzetet):

$$p = x^2,\ q = 0 \quad \Longrightarrow \quad p \cdot q = 0 \quad \Longrightarrow \quad \deg(p \cdot q) = -\infty = 2 + (-\infty).$$

Műveletigény: két n-edfokú polinom összeszorzásához legfeljebb $(n+1)^2$ elemi szorzásra és kb. ugyanennyi összeadásra van szükség, azaz a polinomszorzás *négyzetes* futási idejű.

A polinomszorzásnak érdemes különvenni azt a speciális esetét, amikor az egyik polinom konstans, mert ekkor jelentősen leegyszerűsödik a művelet: minden együtthatót csak meg kell szorozni a konstanssal. Ekkor a műveletigény *lineáris* lesz (mint az összeadás és a kivonás esetén).

Példa. Ha $p = x^3 - 3x^2 + 2x - 4 \in \mathbb{Z}[x]$ és $q = -3$, akkor $p \cdot q$-t így számíthatjuk ki:

$$\begin{array}{r} (\ x^3 - 3x^2 + 2x - 4\ ) \cdot (-3) \\ \hline -3x^3 + 9x^2 - 6x + 12 \end{array}$$

3. 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):

3.1. 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 $\deg r < \deg b$.*

3.2. 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 $\deg(6x - 6) < \deg(x^2 - 2x + 1)$.

Fontos, hogy a maradékos osztás általában csak *test* fölött működik. (Emlékeztető: testben az osztás mindig elvégezhető, pl. $\mathbb{Q}$ és $\mathbb{R}$ test, de $\mathbb{Z}$ nem az, lásd az Algebrai struktúrák c. jegyzetet.)
Példa. $a = x^2 + 3$ és $b = 2x$ hányadosa $q = 1/2 \cdot x$, maradéka $r = 3$. Az osztás tehát nem mindig végezhető el $\mathbb{Z}[x]$-ben, csak $\mathbb{Q}[x]$-ben, mert (mint látni fogjuk) b főegyüttható́jával kell hozzá osztani.
Példa. De más test fölött is működik: $\mathbb{Z}_5[x]$-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 $\overline{2}^{-1} = \overline{3}$).

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

Írásban nagyon hasonlóan lehet polinomokat maradékosan osztani, mint többszámjegyű egész számokat. Az osztandóból, a-ból kiindulva, minden lépésben levonjuk belőle az osztónak, 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.
1. Példa. Induljunk ki a fenti $a = 2x^3 + x^2 - 2x - 1$ és $b = x^2 - 2x + 1$ polinomokból. Az első lépés az, hogy eltüntetjük a legnagyobb fokszámú tagot ($2x^3$). A két főtag hányadosa $2x^3/x^2 = 2x$, ezért b-nek a $2x$-szeresét kell levonni az első lépésben:

$$\begin{array}{r} (\ 2x^3 \ + x^2 - 2x - 1\ ) : (\ x^2 - 2x + 1\ ) = 2x \\ \hline -(\ 2x^3 - 4x^2 + 2x\ ) \\ \hline 5x^2 - 4x - 1 \end{array}$$

Az egyenlőségjel ($=$) után gyűjtjük a hányados tagjait, azaz b aktuális szorzóit ($2x$). A második sorban b-t visszaszoroztuk $2x$-szel, majd ezt kivontuk az első sorból. Innen folytatva, az új főtagok hányadosa $5x^2/x^2 = 5$, ezért b-nek most az 5-szörösét kell kivonni (és a hányados következő tagja 5):

$$\begin{array}{r} (\ 2x^3 \ + x^2 \ - 2x - 1\ ) : (\ x^2 - 2x + 1\ ) = 2x + 5 \\ \hline -(\ 2x^3 \ - 4x^2 \ + 2x\ ) \\ \hline 5x^2 \ - 4x - 1 \\ -(\ 5x^2 - 10x + 5\ ) \\ \hline 6x - 6 \end{array}$$

Innen már nem tudjuk folytatni, mert az eredmény fokszáma a osztó fokszáma alá csökkent. A maradék tehát $r = 6x - 6$, a hányadost pedig a szorzókból gyűjtöttük össze: $q = 2x + 5$.

2. Példa. Legyen $a = 3x^2 - 2x + 1$ és $b = 2x + 1$. Itt a számolást nehezíti, hogy az osztónak nem 1 a főegyütthatója, ezért törtek is megjelennek (pl. $3x^2/2x = \frac{3}{2}x$).

$$\begin{array}{r} (\ 3x^2\ -\ 2x\ +\ 1\ ) : (\ 2x+1\ ) = \frac{3}{2}x - \frac{7}{4} \\ \underline{-(\ 3x^2\ +\ \frac{3}{2}x\ )\qquad\qquad} \\ -\frac{7}{2}x\ +\ 1 \\ \underline{-(\ -\frac{7}{2}x\ -\ \frac{7}{4}\ )} \\ \frac{11}{4} \end{array}$$

Tehát a hányados $q = \frac{3}{2}x - \frac{7}{4}$, a maradék pedig a konstans polinom $r = \frac{11}{4}$.

3. Példa. Legyen $a = 3x^4 - 7x^3 - 5x + 10$ és $b = x - 2$. Itt figyeljünk arra, hogy hagyjunk ki helyet a hiányzó másodfokú tagnak ($0x^2$), nehogy rossz tagokat vonjuk ki egymásból:

$$\begin{array}{r} (\ 3x^4\ -\ 7x^3\qquad -\ 5x\ +\ 10\ ) : (\ x-2\ ) = 3x^3 - x^2 - 2x - 9 \\ \underline{-(\ 3x^4\ -\ 6x^3\ )\qquad\qquad\qquad\qquad} \\ -\ x^3 \\ \underline{-(\ -\ x^3\ +\ 2x^2\ )\qquad\qquad} \\ -\ 2x^2\ -\ 5x \\ \underline{-(\ -\ 2x^2\ +\ 4x\ )\qquad} \\ -\ 9x\ +\ 10 \\ \underline{-(\ -\ 9x\ +\ 18\ )} \\ -8 \end{array}$$

Tehát a hányados $q = 3x^3 - x^2 - 2x - 9$, a maradék pedig $r = -8$.

Műveletigény: ha az osztó és a hányados legfeljebb n-edfokú polinomok, akkor a maradékos osztáshoz legfeljebb $(n+1)^2$ elemi szorzásra és kb. ugyanennyi kivonásra van szükség, azaz ez a művelet is *négyzetes* futási idejű (mint a szorzás).

A következő szakasz végén látni fogunk egy másik módszert is a maradékos osztásra, de az csak abban az esetben működik, ha az osztó $x - c$ alakú elsőfokú polinom (mint a fenti utolsó példában).

4. Kiértékelés

Polinomoknál érdemes külön foglalkozni a *kiértékelés* (vagy *behelyettesítés*) műveletével, azaz amikor egy adott p polinomra és c értékre kiszámítjuk a $p(c)$ értéket.

Ezt megtehetjük közvetlenül a kanonikus alakból, azaz a $p(c) = a_n c^n + a_{n-1}c^{n-1} + \ldots + a_1 c + a_0$ felírásból – de ez így nem hatékony: az $a_n c^n$ kiszámításához n szorzás kell, az $a_{n-1}c^{n-1}$-hez $n-1$ stb., ami összesen $n + (n-1) + \ldots + 2 + 1 = n(n+1)/2$ szorzás, és kell még n összeadás. (Ez például egy tizedfokú polinomra 55 szorzást és 10 összeadást jelent.)

Ezt javíthatjuk úgy, ha jobbról balra számolunk, és mindig megjegyezzük az utolsó c^k értéket, mert akkor a következő c^{k+1}-hez már csak plusz egy szorzás kell: $c^{k+1} = c^k \cdot c$. Így összesen csak $2n-1$ szorzás és n összeadás kell (ami tizedfokú polinomra 19 szorzás és 10 összeadás).

Ennél is hatékonyabb a **Horner-módszer** (vagy *Horner-elrendezés*), amely a polinomot kiemelések sorozatával olyan alakra hozza, amelyből nagyon hatékonyan lehet kiértékelni. Ehhez emeljünk ki x-et minden tagból, amelyből lehet, majd a zárójelben kapott kifejezésben ezt ismételjük meg, és ezt addig folytassuk, amíg minden kitevő el nem tűnik. Például:

$$\begin{aligned} p &= 2x^4 + 3x^3 - 2x^2 - 4x + 3 = \\ &= (2x^3 + 3x^2 - 2x - 4)x + 3 = \\ &= ((2x^2 + 3x - 2)x - 4)x + 3 = \\ &= (((2x + 3)x - 2)x - 4)x + 3. \end{aligned}$$

Általában ez a következő átírást jelenti:

$$\begin{aligned} p &= a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0 = \\ &= ((\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_2)x + a_1)x + a_0. \end{aligned}$$

Ebből a felírásból a behelyettesítés hatékonyan elvégezhető: minden lépésben megszorozzuk az egyik zárójeles kifejezést (legelőször a_n-et) x-szel (vagyis c-vel), majd ehhez hozzáadjuk a következő együtthatót, és így megkapjuk a következő zárójeles kifejezést; ezt folytatva a legvégén eljutunk a teljes helyettesítési értékhez. Ehhez tehát összesen csak n szorzás és n összeadás kell.

Horner-módszer:

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

Itt a b változó menet közbeni értékei a fenti felírás egy-egy zárójeles részkifejezését adják, például az első lépés után $b = a_n c + a_{n-1}$, aztán $b = (a_n c + a_{n-1})c + a_{n-2}$ stb.

(Megjegyzés: a nullpolinomra így ez az algoritmus nem működik, mert annak nincsenek együtthatói. Ezt megoldhatjuk úgy, ha egyetlen nulla együtthatót adunk meg: $n = 0$ és $a_0 = 0$. Általában is igaz, hogy az algoritmushoz nem kell, hogy $a_n \neq 0$ legyen, azaz lehetnek fölösleges 0-k az együtthatólista végén.)

Példa. Legyen $p = 3x^4 - 7x^3 - 5x + 10$ és $c = 2$. Ekkor a Horner-elrendezés így néz ki:

$$p = (((3x - 7)x + 0)x - 5)x + 10,$$

vagyis a b változó a következő értékeket veszi föl az algoritmus során:

$$\begin{aligned} &b := 3, \\ &b := b \cdot 2 - 7 = -1, \\ &b := b \cdot 2 + 0 = -2, \\ &b := b \cdot 2 - 5 = -9, \\ &b := b \cdot 2 + 10 = -8 \quad \longrightarrow \quad p(c) = -8. \end{aligned}$$

Írásban a Horner-módszert a következő táblázatos formában célszerű végrehajtani. Az első sorban felírjuk a polinom együtthatóit (a hiányzó együtthatók helyére 0-t írunk), a harmadik sorban kiszámítjuk a b értékeket a fölötte lévő számok összegeként (az első oszlopban ez csak a főegyüttható), a második sorban pedig az előző b érték c-szeresét írjuk.

Példa. A fenti $p = 3x^4 - 7x^3 - 5x + 10$ polinom esetén ez így néz ki:

	3	−7	0	−5	10
$c = 2$		6	−2	−4	−18
	3	−1	−2	−9	−8

$\longrightarrow \quad p(2) = -8$

Azaz a 3-at lemásoljuk a vonal alá, megszorozzuk $c = 2$-vel (6), összegezzük a második oszlopot $(-7 + 6 = -1)$, az eredményt ismét megszorozzuk $c = 2$-vel (-2), és így tovább cikkcakkban.

A Horner-módszer azonban nemcsak a helyettesítési érték kiszámítására alkalmas. Hasonlítsuk össze a fenti b-értékeket (a táblázat harmadik sorát) a maradékos osztásnál bemutatott 3. példával, ahol ugyanezt a polinomot osztottuk el $b = x-2$-vel. Vegyük észre, hogy az utolsó b-érték (-8) éppen az osztás maradéka: $r = -8$, az előző b-értékek $(3, -1, -2, -9)$ pedig éppen az osztás hányadosának együtthatói: $q = 3x^3 - x^2 - 2x - 9$. Ez igaz általában is:

4.1. 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, azaz $b_{n-1} := a_n$ és $b_k := b_{k+1}c + a_{k+1}$.*

(A maradékra vonatkozó állítás bizonyítása egyszerű: a maradékos osztás tétele miatt (lásd fent) $p = (x-c)\,q + r$, ahol $\deg r < \deg b = 1$, azaz a maradék konstans polinom. Ebbe c-t behelyettesítve megkapjuk a kívánt állítást: $p(c) = (c-c)\,q(c) + r = r$. A hányadosra vonatkozó állítás picit bonyolultabb, ezért azt itt nem bizonyítjuk.)

5. Algebrai deriváltak

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*:

5.1. Definíció. Legyen $p \in K[x]$ egy n-edfokú polinom:

$$p = a_n x^n + \ldots + a_k x^k + \ldots + a_2 x^2 + a_1 x + a_0.$$

Ennek **algebrai deriváltja** a következő $p' \in K[x]$ polinom:

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

ahol a ka_k szorzat a k-tagú $a_k + a_k + \ldots + a_k$ összeget jelenti. (p' kiejtése: "p vessző".)

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

Megjegyzés: az "algebrai" jelző arra utal, hogy ez a deriváltfogalom analízis nélkül is működik.

Megjegyzés 2: a fenti definícióban a ka_k szorzat jelentését azért kellett megadni, mert az a_k együtthatók nem feltétlenül számok, hanem egy tetszőleges K gyűrűből 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 $p' = 5 \cdot \overline{2}x^4 = (\overline{2} + \overline{2} + \overline{2} + \overline{2} + \overline{2})x^4 = \overline{1}x^4 = x^4$.

Megjegyzés 3: a definícióból 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 mindig (még akkor sem, ha $\mathbb{Z}_m$ test): 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 tagja $\mathbb{Z}_3$-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()` tagfüggvény számítja ki.

5.2. Tétel (az algebrai derivált tulajdonságai)**.** *Bármely $p, q \in K[x]$ polinomra és $c \in K$ konstansra:*

1. $c' = 0$;
2. $x' = 1$ *(= K egységeleme);*
3. $(p+q)' = p' + q'$;
4. $(p-q)' = p' - q'$;
5. $(c \cdot p)' = c \cdot p'$;
6. $(p \cdot q)' = p' \cdot q + p \cdot q'$;
7. $(p^n)' = n \cdot p' \cdot p^{n-1}$.

Az algebrai derivált kiszámítása a definíció alapján triviális, csak meg kell szorozni az együtthatókat a megfelelő konstansokkal (és eltolni az indexüket eggyel). Ehhez összesen n szorzásra van szükség, tehát ez a művelet is *lineáris*.

Néha azonban nem magára a p' polinomra van szükség, hanem csak egy helyettesítési értékére egy adott c helyen, azaz a $p'(c)$ értékre. Ezt nyilván kiszámíthatjuk úgy, hogy előállítjuk p'-t a definíció alapján, majd ezt kiértékeljük a c helyen a fenti Horner-módszerrel. Ehhez összesen $2n$ szorzásra és n összeadásra van szükség (valamint memóriára, ha a p'-t eltároljuk).

De $p'(c)$-t ki lehet számítani másképpen is: a Horner-módszer kiterjeszthető úgy, hogy a $p(c)$-vel együtt kiszámítsa $p'(c)$-t is, a p' kiszámítása nélkül. 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 $p'(c) = q(c)$.*

(Bizonyítás: $p' = ((x-c)\,q + p(c))' = q + (x-c)\,q'$, ebbe c-t helyettesítve kijön, hogy $p'(c) = q(c)$.)

Fent láttuk, hogy a Horner-módszer automatikusan kiszámítja q-t is, tehát ha abból egy újabb Horner-módszerrel kiszámítjuk $q(c)$-t, akkor ez a tétel szerint megegyezik $p'(c)$-vel. 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, tehát ha a $p(c)$-re amúgyis szükségünk van, akkor ezenfelül már csak n szorzás és n összeadás kell $p'(c)$-hez, ami hatékonyabb, mint p' kiszámítása és kiértékelése.)

Példa. $p = 3x^4 - 7x^3 - 5x + 10$ és $c = 2$ esetén a dupla Horner-módszer így néz ki:

	3	-7	0	-5	10	$\longleftarrow$	$p = 3x^4 - 7x^3 - 5x + 10$
$c = 2$		6	-2	-4	-18		
	3	-1	-2	-9	-8	$\longleftarrow$	$q = 3x^3 - x^2 - 2x - 9, \quad p(2) = -8$
$c = 2$		6	10	16			
	3	5	8	7		$\longleftarrow$	$q(2) = p'(2) = 7$

Dupla Horner-módszer:

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

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, mert a fenti két bitsorozat bármelyikéből megkaphattuk egy bit megváltozásával. Ezzel a módszerrel mindig észrevesszük, ha egyetlen bit romlik el, de kettőt már nem (mert attól páros marad az 1-esek száma).

Az alábbiakban bemutatjuk a **Cyclic Redundancy Check (CRC)** nevű módszert, amely a paritásbitnél jobb megoldást ad hibadetektálásra.

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 számolás 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 átrendezve $mx^n - c = qg$; 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, amelyekre a fent tárgyalt maradékos osztás módszerét használjuk, csak nem írjuk ki az x-eket. A kivonás tehát valójában "kizáró vagy" (XOR).)

Küldés ($mx^n \bmod g$):

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

Ellenőrzés ($mx^n + c \bmod g$):

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

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 hibás átvitel matematikailag úgy írható le, hogy az elküldött $mx^n + c$ kódszó helyett $mx^n + c + e$ érkezik meg, ahol e a **hibapolinom**, azaz amelyben 1-esek jelzik az elromlott bitek helyét (ugye XOR-ban egy 1-es bit éppen megfordítja a másik bitet). 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 $g \mid mx^n + c$). Egy jó g tehát olyan, amely a gyakorlatban előforduló hibamintázatokra minél ritkábban osztja e-t.

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/Polynomial_long_division

[3] https://en.wikipedia.org/wiki/Synthetic_division

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

[5] 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.