POČÍTAČOVÁ ALGEBRA

DAVID STANOVSKÝ

Verze ze dne 3.4.2007

Toto jsou provizorní skripta k přednášce z počítačové algebry. Velký dík za jejich průběžný vznik patří Ivanu Štubňovi, který pomáhá přepisovat zápisky z přednášek do elektronické formy. Veškeré chyby, jako že jich tam zřejmě pár bude, ovšemže padají na mou hlavu.

Počítačová algebra se zabývá algoritmy pro operace v různých algebraických oborech. Nebudeme se snažit definovat přesný obsah, ale pokusíme se vymezit tento obor vůči tzv. *numerické matematice*. Numerická matematika vyvíjí algoritmy na *přibližné* výpočty, zpravidla nad reálnými či komplexními čísly a její důležitou součástí je teorie odhadu chyb. Oproti tomu počítačová algebra vyvíjí algoritmy, které počítají *přesně*. Typickým oborem, se kterým pracuje, jsou proto celá a racionální čísla, jejich algebraická rozšíření či konečná tělesa, a dále polynomy či matice nad těmito obory. Na pravou míru bychom měli uvést zdání, že numerická matematika je „méně dokonalá": ano, počítačová algebra dává výsledky přesné a je tedy žádoucí použít její algoritmy, kdykoliv jsou k dispozici a vedou k řešení. Daní za přesnost je ovšem často vyšší výpočetní složitost (v některých případech natolik vyšší, že vede k pratické nepoužitelnosti). Navíc k mnoha důležitým problémům přesné algoritmy ani neexistují — např. k řešení většiny diferenciálních rovnic.

V této přednášce se zaměříme na algoritmy pro operace s polynomy. To je v současnosti jeden z nejdůležitějších směrů počítačové algebry, i díky svým aplikacím v kryptografii (rozklady čísel, polynomů) a výpočetní geometrii. Mezi další důležité směry počítačové algebry patří např. algoritmy pro lineární algebru a řešení lineárních a polynomiálních rovnic, algoritmy na výpočet primitivních funkcí a (přesné) řešení diferenciálních rovnic, ale také výpočetní teorie grup a zahrnout by se dala částečně i rovnicová logika, obsahující problémy okolo úpravy výrazů či automatické dokazování vět.

Obsah přednášky by se dal shrnout takto: nejprve představíme základní obory, kterými se budeme zabývat (půjde zejména o celá a racionální čísla, jejich algebraická rošíření a konečná tělesa, a dále polynomy nad nimi), předvedeme rychlé algoritmy pro základní operace v těchto oborech (sčítání, násobení, dělení, apod.), a poté se budeme zabývat složitejšími operacemi jako největší společný dělitel či rozklad na ireducibilní činitele. Mnoho rychlých algoritmů je založeno na tzv. *modulární reprezentaci*, kterou představíme v druhé kapitole: jejím principem je trik, že místo jednoho výpočtu v komplikovaném oboru provedeme úlohu v několika jednodušších oborech. V některých případech dochází k výrazné úspoře času. Na závěr pak probereme tzv. metodu Gröbnerových bází, která umožňuje řadu efektivních výpočtů s polynomy více proměnných. Mezi hlavní oblasti využití algoritmů, které se naučíme, patří výpočetní teorie čísel (rozklady, testy prvočíselnosti, apod.), výpočetní geometrie (algoritmy pro počítání s algebraickými křivkami, které se používají v moderních kryptografických systémech), či symbolická integrace.

Probírané algoritmy jsou samozřejmě součástí velkých softwarových balíků na počítačovou algebru. Největší a nejpoužívanější se zdají být v současné době systémy Maple a Mathematica. Pro výpočetní teorii grup je pak nejrozšířenějším systémem GAP (`www.gap-system.org`).

Nebudeme zde definovat základní pojmy teorie složitosti. Pouze pro úplnost připomeneme značení týkající se asymptotického chování funkcí.

Definice. Nechť $f, g : \mathbb{N}^k \to \mathbb{R}^+$ jsou k-ární funkce. Řekneme, že $f = \mathcal{O}(g)$ (f je *asymptoticky menší nebo rovná* g), pokud existuje konstanta C taková, že $f(x_1, \dots, x_k) \leq Cg(x_1, \dots, x_k)$ pro všechna $x_1, \dots, x_k \in \mathbb{N}$. Řekneme, že $f \sim g$ (*asymptoticky ekvivalentní*), pokud $f = \mathcal{O}(g)$ a $g = \mathcal{O}(f)$.

Ještě připomeneme terminologii okolo polynomů. Je-li

$$p = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 \in \mathbf{R}[x]$$

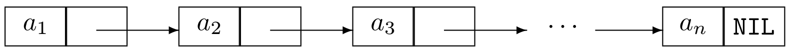
a $a_n \neq 0$, pak n nazýváme *stupeň polynomu* a značíme $\deg p$ (pro úplnost definujeme $\deg 0 = -1$), a_0 nazýváme *absolutní člen*, $a_n x^n$ nazýváme *vedoucí člen* a značíme $\mathrm{lm}(p)$ (lm jako *leading monomial*) a a_n nazýváme *vedoucí koeficient* a značíme $\mathrm{lc}(p)$ (lc jako *leading coefficient*). V případě polynomů více proměnných používáme analogickou terminologii. Není ovšem zřejmé, který člen je vedoucí. Tento problém bude rozebrán podrobně v kapitole o Gröbnerových bázích.

Základní obory a operace

1. Počítačová reprezentace dat

Ještě než se pustíme do diskuse jednotlivých oborů, měli bychom zdůraznit, že algoritmy počítačové algebry počítají *přesně*. Není tedy možné shora omezit velikost dat (počet cifer čísla, stupeň polynomu, velikost matice, apod.), neboť jednak nevíme, jak velký vstup nám uživatel zadá (i velké číslo lze zadat na malém prostoru — např. číslo 100!), ale, co je možná důležitější, i při malých vstupech mohou v průběhu výpočtu vznikat veliká čísla (typickým příkladem je Eukleidův algoritmus na výpočet největšího společného dělitele dvou polynomů). Anglickým termínem, který neumím přeložit, se tato vlastnost označuje jako *multiprecision arithmetics*. Jsou víceméně dvě základní datové struktury vhodné pro tuto situaci.

(1) *Spojový seznam.* `struct seznam = (TYP hodnota, seznam * dalsi).`
Základní jednotkou je buňka obsahující dvě políčka: jedno s hodnotou a druhé s ukazatelem (pointrem) na další buňku. Ukazatel poslední buňky neukazuje nikam (je nastaven na `NIL`). Graficky můžeme spojový seznam znázornit následovně:



Výhody: Čistá práce (ve chvíli, kdy máme k dispozici základní funkce typu přídávání a odebírání prvků ze seznamu, spojování seznamů, atd., se nemusíme vůbec starat o správu paměti), snadné prodlužování a navazování seznamů.
Nevýhody: Velká spotřeba paměti, neboť ke každé hodnotě potřebujeme uložit i ukazatel. Pomalé prohledávání.
Poznamenejme, že tato datová struktura je implicitní v jazycích typu Prolog či Lisp (přesněji řečeno, v těchto jazycích ani jiná datová struktura k dispozici není).

(2) *Dynamické pole.* `struct dyn_pole = (int`[1] `delka, MUJTYP * pole).`
Jedná se o pole, jehož velikost můžeme podle potřeby měnit. První políčko obsahuje současnou délku pole, další políčka postupně hodnoty. Graficky můžeme dynamické pole znázornit následovně:

n	a_1	a_2	$\ldots$	a_n

Výhody: Oproti spojovému seznamu šetří paměť a v jazycích typu Pascal nebo C se s ní snadno manipuluje.

[1] Abychom byli korektní, tato proměnná by také měla být schopna nabývat libovolně velkých hodnot. Ovšem při praktické implementaci samozřejmě stačí zvolit tak velký typ, jak velké pole nám umožní naalokovat použitý jazyk (což je v každém případě méně, než veškerá fyzická paměť použitého počítače; např. do 1GB paměti se vejde číslo s přibližně miliardou cifer, stačí tedy 32-bitový typ, což je obvykle něco jako `longint`).

Nevýhody: Nemáme-li k dispozici chytrý manažer paměti, pak nastává problém při prodlužovaní již jednou alokovaného pole.
Poznamenejme, že tuto datovou strukturu používá např. systém Maple.

1.1. Datová reprezentace celých čísel. Čísla reprezentujeme pomocí cifer ve vhodně zvolené bázi. Znaménko se zpravidla ukládá nějakým způsobem do nejvyšší cifry (abychom nezapadli do technických detailů, budeme nadále považovat všechna celá čísla za nezáporná :–)). Jak zvolit bázi, abychom efektivně využili paměť a možnosti procesoru? Protože základní operace $+, \cdot$ jsou v procesoru implementovány na úrovni slov, volí se obvykle jedna z následujících bází (zde n značí počet bitů ve slově, což je dnes obvykle 32):

(1) 2^n — Výhoda: maximální využití paměti. Nevýhoda: vyšší časová náročnost, neboť uživatel zpravidla vyžaduje vstupy a výstupy v desítkové soustavě (viz složitost převodu mezi bázemi).
(2) 10^k, kde k je největší číslo splňující $10^k \leq 2^n$ — Výhoda: báze jakou chce běžný uživatel. Nevýhoda: mrhání pamětí, neboť spousta hodnot je nevyužita.

V obou reprezentacích platí, že nejnižší cifry čísla je výhodné psát na začátek (tj. číslo psát pozpátku), protože při základních operacích se čísla čtou obvykle od nejnižších cifer a navíc dynamické pole se lépe prodlužuje na konci.

Příklad. Uvažujme například 8-bitový počítač a reprezentujme číslo $2^{17} = 131072$. Pak

(1) jestliže si za bázi zvolíme $2^8 = 256$, pak zápis čísla v této bázi bude

0	0	2

(2) jestliže si za bázi zvolíme $10^2 = 100 \leq 2^8$, pak zápis čísla v této bázi bude

72	10	13

Počet cifer čísla n v bázi B je $\log_B n = \left\lceil \frac{\log n}{\log B} \right\rceil + 1$. Abychom se vyhnuli počítání s konstantami, *nadále budeme považovat všechny logaritmy implicitně se základem B, kde B je zvolená báze.*

Definice. Počet cifer čísla n ve zvolené bázi B značíme $L_B(n)$ nebo krátce $L(n)$.

Poznamenejme, že asymptotické chování časové složitosti nezáleží na volbě báze: počítáme-li složitost algoritmu v počtu operací s bajty zadaného čísla, tj. rozhoduje-li počet cifer n, pak ovšem v bázi B je parametrem $L_B(n)$, zatímco v bázi C je parametrem $L_C(n)$. Jenže $\log_B n = \frac{\log C}{\log B} \log_B n$, a tedy $L_B(n) \sim L_C(n)$.

1.2. Datová reprezentace racionálních čísel. Racionální číslo reprezentujeme jako dvojici (čitatel, jmenovatel).

čitatel	jmenovatel

Racionální čísla reprezentujeme obvykle v *základním tvaru*, kdy čitatel a jmenovatel jsou nesoudělní. Poznamenejme, že toto je časově poměrně náročná operace (viz níže), v některých výpočtech tedy stojí za úvahu, zda krácení neprovádět jen v některých krocích.

1.3. Datová reprezentace algebraických rozšíření racionálních čísel. Z hlediska počítačové algebry se má smysl zabývat rozšířeními konečného stupně, tedy tělesy tvaru $\mathbb{Q}(\alpha_1, \dots, \alpha_n)$, kde $\alpha_1, \dots, \alpha_n \in \mathbb{C}$ jsou nějaké algebraické prvky nad $\mathbb{Q}$. Navíc stačí uvažovat pouze rozšíření tvaru $\mathbb{Q}(\alpha)$, neboť iterací níže uvedeného postupu získáme reprezentaci rozšíření s obecně n prvky.

K reprezentci tělesa $\mathbb{Q}(\alpha)$ je výhodné využít větu, která říká, že toto těleso je izomorfní faktorokruhu $\mathbb{Q}[x]/m_{\alpha,\mathbb{Q}}\mathbb{Q}[x]$, kde $m_{\alpha,\mathbb{Q}}$ je minimální polynom prvku α nad tělesem $\mathbb{Q}$. Z toho plyne, že $\mathbb{Q}(\alpha)$ lze považovat za vektorový prostor nad tělesem $\mathbb{Q}$ dimenze $n = \deg m_{\alpha,\mathbb{Q}}$, jehož prvky korespondují s polynomy stupně $0, \dots, n-1$ s racionálními koeficienty. Operace s prvky tělesa $\mathbb{Q}(\alpha)$ se převádějí na operace s polynomy modulo $m_{\alpha,\mathbb{Q}}$.

Příklad. Těleso $\mathbb{Q}(\sqrt[3]{2})$ je izomorfní faktorokruhu $\mathbb{Q}[x]/(x^3-2)\mathbb{Q}[x]$. Prvku $a + b\sqrt[3]{2} + c\sqrt[3]{4}$ odpovídá polynom $cx^2 + bx + a$.

Tedy problém datové reprezentace těchto rozšíření se převádí na problém datové reprezentace polynomů, který bude diskutován níže.

Poznamenejme, že algebraická rošíření racionálních čísel se přirozeně objevují v algoritmech jako je Rychlá Fourierova transformace či v algoritmech na (přesné) řešení polynomiálních rovnic s racionálními koeficienty — viz např. kapitolka ze Základů algebry o konstrukcích pravítkem a kružítkem.

1.4. Datová reprezentace konečných těles. Prvky tělesa $\mathbb{F}_p$ jsou čísla $0, \dots, p-1$, problém se tedy převádí na reprezentaci celých čísel. Ostatní konečná tělesa jsou, jak známo, algebraická rozšíření těles $\mathbb{F}_p$, problém lze tedy řešit analogicky jako v předchozím odstavci.

1.5. Datová reprezentace polynomů.

1.5.1. *Polynomy jedné proměnné.* Polynom $p = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ lze reprezentovat dvěma způsoby:

(1) *Hustá reprezentace.* Do paměti se uloží všechny koeficienty, včetně nulových.

n	a_0	a_1	$\dots$	a_{n-1}	a_n

(2) *Řídká reprezentace.* Do paměti se uloží pouze nenulové koeficienty, spolu s exponentem

počet členů	člen 1	člen 2	$\dots$	člen m

přičemž každý člen je dvojice tvaru

koeficient	exponent

Často hovoříme o řídkých resp. hustých polynomech, v závislosti na tom, zda je nebo není většina koeficientů nulová. Velký význam má toto dělení při generování náhodných polynomů.

Příklad. Uvažujme polynom $x^{100} - x + 2$. Jeho hustá reprezentace je

100	2	-1	0	0	$\dots$	0	0	1

(tedy spotřebujeme 102 jednotek paměti), zatímco řídká reprezentace je

3	2	0	-1	1	1	100

(tedy stačí 7 jednotek paměti).

Výhody a nevýhody té které reprezentace jsou zřejmé. Jejich užití se řídí tím, jaká data bude uživatel algoritmu poskytovat, a také tím, co je vlastně pro ten který algoritmus potřeba (např. při Rychlé Fourierově transformaci se husté reprezentaci nevyhneme).

1.5.2. *Polynomy více proměnných.* Uvažujme obor $\mathbf{R}[x_1, \dots, x_n]$. Jeho prvky lze reprezentovat dvěma základními způsoby:

(1) *Distributivní forma.* Členy jsou výrazy tvaru koeficient z R krát term v proměnných $x_1, \dots, x_n$.
(2) *Rekurzivní forma.* Vybereme jednu význačnou proměnnou, např. x_n, a polynom více proměnných považujeme za polynom jedné proměnné x_n s koeficienty v oboru $\mathbf{R}[x_1, \dots, x_{n-1}]$. Jinými slovy, obor $\mathbf{R}[x_1, \dots, x_n]$ ztotožníme s oborem $(\mathbf{R}[x_1, \dots, x_{n-1}])[x_n]$.

Obě reprezentace dále můžeme uvažovat *husté* nebo *řídké*. Poznamenejme, že hustá distributivní forma je obvykle velmi nevýhodná, neboť s růstem počtu proměnných prudce roste počet možných termů daného stupně (např. polynom stupně 6 ve třech proměnných může mít až 120 členů).

Příklad. Uvažujme polynom

$$p = x^3y + 2xyz + z^2 - x^2y^3z + 3y^2z^2.$$

Tento výraz odpovídá distributivní formě, v ní je jeho stupeň 6 (viz čtvrtý člen). Jeho řídká reprezentace je

5	1	x^3y	2	xyz	1	z^2	-1	x^2y^3z	3	y^2z^2

(jak už bylo řečeno, hustá reprezentace by měla 120 členů, byla by tedy velmi neefektivní.)

Rekurzivní forma polynomu p vzhledem k proměnné z je

$$(1 + y^2)z^2 + (xy + x^2y^3)z + x^3y,$$

má tedy stupeň 2, zatímco vzhledem k proměnné x to je

$$yx^3 + (y^3z)x^2 + (yz)x + (z^2 + y^2z^2),$$

má tedy stupeň 3. Hustá rekurzivní reprezentace vzhledem k proměnné z by tedy byla

2	x^3y	$xy + x^2y^3$	$1 + y^2$

zatímco řídká by byla

3	x^3y	0	$xy + x^2y^3$	1	$1 + y^2$	2

Opět, volba reprezentace záleží na užitém algoritmu a na předpokládané struktuře dat.

Vznikat také otázka, co to je „hezký tvar polynomu": máme zadaný polynom expandovat do distributivního tvaru? Nebo jej naopak rozložit na tzv. součinovou formu? Nebo něco jiného? Uvažujte následující příklady: polynom

$$x^{1000} - y^{1000} = (x - y)(x^{999} + \dots + y^{999})$$

je evidentně „hezčí" v expandovaném tvaru, zatímco polynom

$$(x + y)^{1000}$$

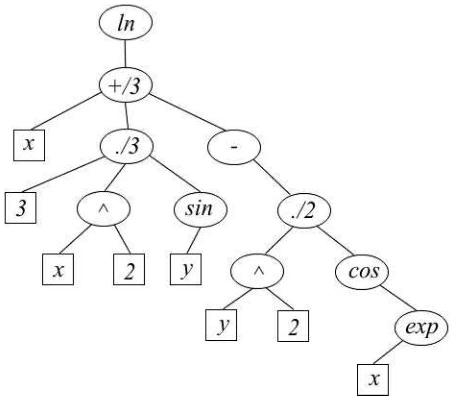
v součinovém, a polynom

$$(x + y)^{1000} - y^{1000}$$

je nejlepší nechat tak, jak je. Navíc si uvědomte, že zatímco násobení je mnohem jednodušší v součinové formě, sčítání zase v expandované. A to už nemluvíme o složitosti převodu mezi jednotlivými tvary. Závěr je, že žádnou z těchto forem nelze považovat za lepší a že je často dobré si pamatovat „historii úprav". Např. systém Maple to řeší tím, že ponechává polynom v zadaném tvaru, dokud ho uživatel explicitně nepožádá o úpravu (viz funkce `expand` a `factorize`).

1.6. **Reprezentace obecných výrazů.** Obecné výrazy je přirozené reprezentovat pomocí stromu: `struct vyraz = (SYMBOL symbol, int arita, vyraz * arg1, ..., vyraz * argn)`.

Příklad. Uvažujme výraz $\ln(x + 3x^2 \sin(y) - y^2 \cos(e^x))$. Jeho stromová reprezentace je



Velkým problémem je rozhodnout, kdy dva výrazy určují stejnou funkci na celých (racionálních, reálných, komplexních) číslech. Pro polynomy je to snadné — stačí je roznásobit na expandovanou formu a porovnat koeficienty. Lze to i v některých dalších omezených případech: např. Caviness a Fateman dokázali, že lze rozhodnout, kdy dva výrazy používající operace $+, -, \cdot, /$ a do sebe nezanořené odmocniny určují stejnou funkci na reálných číslech. Naopak Caviness (s užitím slavných výsledků Matijaseviče diskutovaných v poslední kapitole) dokázal, že rovnost výrazů používajících operace $+, -, \cdot, \sin$, absolutní hodnotu a konstantu π na reálných číslech nelze algoritmicky rozhodnout.

2. Základní operace

V následující tabulce podáváme přehled složitosti základních operací v $\mathbb{Z}$, $\mathbb{Q}$ a na polynomech.

	n	časová složitost	$+$	$\cdot$	div , mod	NSD
$\mathbb{Z}$	počet cifer	operace s ciframi	$\mathcal{O}(n)$	$\mathcal{O}(n^{\log_2 3})$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
$\mathbb{Q}$	počet cifer	operace s ciframi	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	—	—
$\mathbf{R}[x]$	deg	operace v $\mathbf{R}$	$\mathcal{O}(n)$	$\mathcal{O}(n^{\log_2 3})$ $\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$ $\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$

Zde je důležité zmínit, jak počítáme časovou složitost.

(1) Pro *čísla* je parametrem počet cifer a v časové složitosti bereme jako jednotkové operace s ciframi (sčítání a násobení). (Pro racionální čísla obvykle počítáme to větší z počtu cifer čitatele a jmenovatele.)

(2) Pro *polynomy* je parametrem stupeň a v časové složitosti bereme jako jednotkové operace v okruhu koeficientů (sčítání a násobení).

V tabulce pro jednoduchost předpokládáme, že parametr je stejný pro obě vstupní čísla (oba polynomy), a to n. Jemnější odhady budou dokázány níže. Připomínáme, že asymptotická složitost nezáleží na volbě báze.

Co se týče dvojích hodnot u násobení a dělení polynomů, záleží na vlastnostech oboru **R**. Viz kapitola o Rychlé Fourierově transformaci.

Poznamenejme, že náš předpoklad, že všechny operace v oboru koeficientů jsou stejně časově náročné, není vždy rozumný. Např. při běhu Eukleidova algoritmu v $\mathbb{Q}[x]$ může docházet k prudkému růstu koeficientů, což algoritmus značně zpomalí. Tento problém bude diskutován později. V případě základních operací se lze spokojit s tímto zjednodušením.

2.1. **Operace v oboru celých čísel.** Uvažujme dvě celé čísla

$$a = \sum_{i=0}^{n-1} a_i B^i \quad \text{a} \quad b = \sum_{i=0}^{m-1} b_i B^i,$$

kde B je zvolená báze, n počet cifer čísla a a m počet cifer čísla b. Předpokládejme $m < n$.

2.1.1. *Sčítání.* Uvažujme algoritmus sčítání dobře známý ze základní školy.

$$\begin{array}{r} a_{n-1} \dots a_m a_{m-1} \dots a_1 a_0 \\ 0 \dots 0 b_{m-1} \dots b_1 b_0 \\ \hline c_{n-1} \dots c_m c_{m-1} \dots c_1 c_0 \end{array}$$

Cifru c_0 dostaneme jako součet $a_0 + b_0$ mod B. Jestliže $a_0 + b_0 \geq B$, pak se přenáší do dalšího řádu 1, jinak 0. Cifru c_1 dostaneme jako součet $a_1 + b_1 +$ přenos mod B. Analogicky, jestliže $a_1 + b_1 +$ přenos $\geq B$, pak přenášíme dále 1, jinak 0. A tak dále. Je vidět, že potřebujeme $\max(m, n) = n$ kroků, v každém děláme konstantní množství operací s ciframi. Časová složitost sčítání je tedy

$$\mathcal{O}(\max(m, n)).$$

2.1.2. *Násobení.* Uvažujme opět algoritmus dobře známý ze základní školy.

$$\begin{array}{r} \boxed{a = a_{n-1} \dots a_0} \\ \boxed{b = b_{m-1} \dots b_0} \\ \hline \boxed{b_0 a} \\ \boxed{b_1 a} \\ \vdots \\ \boxed{b_{m-1} a} \\ \hline \boxed{ab} \end{array}$$

Vidíme, že násobení dvou čísel délek m, n obnáší (1) m-krát násobit číslo délky n jednou cifrou, k čemuž je při provedení postupu analogického algoritmu na sčítání potřeba $\mathcal{O}(n)$ operací s ciframi, (2) součet m čísel délky n, k čemuž je potřeba $m \cdot \mathcal{O}(n)$ operací. Celková časová složitost školského násobení je tedy

$$\mathcal{O}(mn).$$

Na rozdíl od sčítání, školské násobení není nejefektivnější metodou. Způsob, jak jej zefektivnit, je metoda *„rozděl a panuj"*. Ilustrujeme si ji nejprve na následujícím algoritmu. Předpokládejme, že čísla a a b jsou stejně dlouhá (kratší stačí doplnit zleva nulami) a mají sudou délku. Rozdělme si je na poloviny, tj. nechť

$$a = rB^{n/2} + s \quad \text{a} \quad b = tB^{n/2} + u.$$

Pak lze počítat

$$a \cdot b = (rB^{n/2} + s)(tB^{n/2} + u) = rtB^n + (ru + st)B^{n/2} + su.$$

Tedy místo výpočtu jednoho součinu ab dvou čísel délky n počítáme čtyři součiny čísel čísel délky $\frac{n}{2}$, konkrétně rt, ru, st a su. Abychom mohli i v dalším kroku dělit úlohu na dvě části, budeme předpokládat, že $n = 2^k$ pro nějaké k.

Algoritmus 2.1. `NÁSOB`(a, b)
`VSTUP:` $a, b \in \mathbb{Z}$ délky 2^k.
`VÝSTUP:` $a \cdot b$.
0. `IF` $k = 0$ `THEN RETURN` ab, `STOP`.
1. $w_1 :=$ `NÁSOB`(r, t), $w_2 :=$ `NÁSOB`(r, u), $w_3 :=$ `NÁSOB`(s, t), $w_4 :=$ `NÁSOB`(s, u).
2. `RETURN` $w_1 B^n + (w_2 + w_3)B^{n/2} + w_4$.

Z výše uvedeného vzorce je zřejmé, že algoritmus funguje.

Tvrzení 2.2. *Časová složitost Algoritmu 2.1 je $\mathcal{O}(n^2)$, kde $n = 2^k$ je počet cifer obou čísel.*

Důkaz. Označme $T(n)$ počet operací, které algoritmus potřebuje na vstupu délky n. Pro čísla délky 1 máme $T(1) = 1$, neboť násobíme dvě jednociferná čísla. Pro čísla délky n platí

$$T(n) = 4T(n/2) + Cn$$

pro jistou konstantu C, neboť úlohu převádíme na čtyři úlohy poloviční délky (krok 1.) a v druhém kroku sčítáme čtyři čísla délky n. Vynecháme konkrétní výpočet – je velmi podobný výpočtu v důkazu složitosti Karacubova násobení. Rekurentní vztahy podobné výše uvedenému vznikají často při odhadech složitosti algoritmů založených na metodě „rozděl a panuj". Existují obecná tvrzení na jejich řešení (Master theorem, Akra-Bazzi theorem). $\square$

Jak vidíme, tento algoritmus má stejnou složitost jako klasické násobení. Existuje ovšem trik, připisovaný A. A. Karacubovi, jak tento algoritmus mírně upravit a získat postup asymptoticky výrazně lepší. Karacubův trik spočívá v tom, že se použije vyjádření $ru + st = rt + su + (r - s)(u - t)$, a tedy dostáváme vzorec

$$a \cdot b = rtB^n + (rt + su + (r - s)(u - t))B^{n/2} + su.$$

Všimněte si, že nyní stačí volat pouze tři násobení čísel poloviční délky: $r \cdot t$, $s \cdot u$, $(r - s) \cdot (u - t)$. Cena, kterou platíme, je větší počet operací sčítání. To se ovšem v asymptotice neprojeví.

Algoritmus 2.3. `KARACUBA`(a, b)
`VSTUP:` $a, b \in \mathbb{Z}$ délky 2^k.
`VÝSTUP:` $a \cdot b$.
`0.` `IF` $k = 0$ `THEN RETURN` ab, `STOP`.
`1.` $w_1 :=$ `KARACUBA`(r, t), $w :=$ `KARACUBA`$(r - s, u - t)$, $w_4 :=$ `KARACUBA`(s, u).
`2.` `RETURN` $w_1 B^n + (w_1 + w_4 + w) B^{n/2} + w_4$.

Správnost algoritmu vyplává, tak jako předtím, z výše uvedeného vzorce.

Tvrzení 2.4. *Časová složitost Algoritmu 2.3 je $\mathcal{O}(n^{\log_2 3})$, kde $n = 2^k$ je počet cifer obou čísel.*

Důkaz. Označme $T(n)$ počet operací, které algoritmus potřebuje na vstupu délky n. Pro čísla délky 1 máme $T(1) = 1$, neboť násobíme dvě jednociferná čísla. Pro čísla délky n platí

$$T(n) = 3T(n/2) + Cn$$

pro jistou konstantu C, neboť v kroku 1. provádíme dvě odčítání a převádíme úlohu na tři úlohy poloviční délky a v druhém kroku sčítáme pět čísel délky n. Protože $n = 2^k$, můžeme počítat

$$\begin{aligned}
T(n) = T(2^k) &= 3T(2^{k-1}) + C2^k = 3(3T(2^{k-2}) + C2^{k-1}) + C2^k \\
&= 3^2 T(2^{k-2}) + C2^k(3/2 + 1) \\
&= 3^2(3T(2^{k-3}) + C2^{k-2}) + C2^k(3/2 + 1) \\
&= 3^3 T(2^{k-3}) + C2^k((3/2)^2 + 3/2 + 1) \\
&= \dots \\
&= 3^k T(2^{k-k}) + C2^k((3/2)^{k-1} + \dots + 1) = 3^k T(1) + C2^k \frac{(3/2)^k - 1}{3/2 - 1} \\
&= 3^k + 2C3^k - 2C2^k = 2^{k \log_2 3} + 2C2^{k \log_2 3} - 2C2^k \\
&= n^{\log_2 3} + 2Cn^{\log_2 3} - 2n = \mathcal{O}(n^{\log_2 3})
\end{aligned}$$

□

Jak již bylo řečeno, daní za lepší asymptotickou složitost je vyšší režie. Při praktických implementacích se uvádí, že Karacubovo násobení se vyplatí pro čísla s řádově desítkami 32-bitových cifer. Poznamenejme, že $\log_2 3$ je rovno přibližně 1.58. Následující tabulka uvádí několik hodnot. Interpretace je taková, že na desetkrát větších číslech poběží klasické násobení zhruba stokrát pomaleji, zatímco Karacubovo pouze zhruba čtyřicetkrát pomaleji.

n	1	10	100	1000
n^2	1	100	10000	10^6
$n^{\log_2 3}$	1	38	1445	55000

Poznámka. Podobným trikem jako u Karacubova násobení lze asymptotickou složitost zlepšit až na $\mathcal{O}(n^{1+\epsilon})$, kde $\epsilon > 0$ je libovolné (tzv. Toom-Cookovo násobení).

Poznámka. Pro násobení celých čísel existuje teoreticky ještě rychlejší algoritmus, tzv. *Schönhage-Strassenův*, s časovou složitostí $\mathcal{O}(n \log n \log\log n)$. Problém je, že konstanta skrytá v symbolu $\mathcal{O}$ je tak velká, že praktická použitelnost tohoto algoritmu se pohybuje v řádu čísel velikosti 2^{10000} a více, což je hodně i na kryptografické aplikace (poznamenejme, že za bezpečné velikosti klíče pro RSA se v současné době považují čísla řádu 2^{1024} a i pro vzdálenější budoucnost by měly stačit klíče řádu 2^{4096}). Algoritmus má nicméně velký teoretický význam, neboť umožňuje konstruovat algoritmy se subkvadratickou složitostí (tedy rychlejší než jedno klasické násobení) na řadu problémů teorie čísel (např. testy prvočíselnosti).

Princip Schönhage-Strassenova násobení je takový, že dané číslo délky n, kde $n = 2^k$, rozdělíme na b bloků délky l, kde b a l jsou mocniny dvojky rovné zhruba $\sqrt{n}$ (pro k liché se vezme l větší). Jednotlivé cifry součinu se pak se počítají jednak modulo b (jakkoliv, neboť b je malé), a dále modulo $2^{2l} + 1$, kde se použitím jistého triku s konvolucemi výpočet převede pomocí rychlé Fourierovy transformace na násobení b čísel délky $2l$ a pokračuje se rekurzí.

Cvičení.
- Dokažte, že mocninu $c = a^b$ lze spočítat v čase $\mathcal{O}(\log^2 c)$ při použití standardní aritmetiky. Najděte podobný odhad při použití Karacubova algoritmu na násobení.
- Dokažte, že mocninu $c = a^b \bmod m$ lze spočítat v čase $\mathcal{O}(\log a \log m + \log b \log^2 m)$.

2.1.3. *Dělení se zbytkem.* Uvažujme opět algoritmus dobře známý ze základní školy.

$$
\begin{array}{ccc}
a & b & a \text{ div } b \\
\begin{array}{|c|c|c|c|}\hline a_{n-1} & \cdots & a_{n-m+1} & \cdots \\ \hline\end{array}
\; : &
\begin{array}{|c|c|}\hline b_{m-1} & \cdots \\ \hline\end{array}
\; = &
\begin{array}{|c|c|}\hline c_{d-1} & \cdots \\ \hline\end{array}
\end{array}
$$

$$
\begin{array}{l}
\boxed{\quad -c_{d-1}b \quad} \\ \hline
\quad\boxed{\quad \cdots \quad} \\
\qquad\quad \ddots \\
\qquad\qquad \boxed{a \bmod b}
\end{array}
$$

Tvrzení 2.5. *Časová složitost školského dělení je $\mathcal{O}(dm) = \mathcal{O}((n-m+1)m)$, kde n je počet cifer dělence, m je počet cifer dělitele a d je počet cifer podílu.*

Důkaz. K vydělení čísla délky n číslem délky m potřebujeme nejvýše d kroků ($d \leq n - m + 1$). V i-tém kroku pak hledáme největší c_{d-i} takové, že $c_{d-i}b < u$, kde u je číslo se zápisem $a_{n-i} \ldots a_{d-i+1}$. Vzhledem k tomu, že c_{d-i} je jednociferné, tedy, $0 \leq c_{d-i} < B$, stačí vyzkoušet nejvýše B možností, to znamená konstantně mnoho kroků. V každém z těchto nejvýše B kroků násobíme $c_{d-i}b$ (což má složitost $\mathcal{O}(m)$) a porovnáváme výsledek s číslem u (což má také složitost $\mathcal{O}(m)$). Celková časová složitost dělení je tedy $\mathcal{O}(dm) = \mathcal{O}((n-m+1)m)$. $\square$

2.1.4. *Převod mezi bázemi.* Nechť B, C jsou dvě báze a označme

$$a = \sum_{i=0}^{n-1} a_i B^i = \sum_{i=0}^{m-1} b_i C^i$$

zápisy čísla a v těchto bázích. Úloha zní: máme-li dány $a_0, \ldots, a_{n-1}$, jak zjistit $b_0, \ldots, b_{m-1}$?

Klasický postup při převodu z báze B do báze C je dělení a zapisování zbytků:

$$\begin{aligned} b_0 &= a \bmod C \\ b_1 &= (a \operatorname{div} C) \bmod C \\ b_2 &= (a \operatorname{div} C^2) \bmod C \\ &\dots \end{aligned}$$

Algoritmus 2.6.
`VSTUP:` $a = \sum_{i=0}^{n-1} a_i B^i$.
`VÝSTUP:` $b_0, \dots, b_{m-1}$ taková, že $a = \sum_{i=0}^{m-1} a_i C^i$.
`0.` $u_0 := a$.
`i.` $b_i := u_i \bmod C$, $u_{i+1} := u_i \operatorname{div} C$.
`IF` $u_{i+1} = 0$ `THEN STOP`.

Správnost algoritmu je zřejmá.

Tvrzení 2.7. *Časová složitost Algoritmu 2.6 je $\mathcal{O}(n^2)$ (kde $n = L(a)$, tj., až na konstantu, počet cifer v jakékoliv bázi).*

Důkaz. Vykoná se přesně m kroků a jak bylo již řečeno, $m = \mathcal{O}(n)$. V každém kroku dochází k dělení čísla u_i číslem C, tedy čísla nejvýše n-ciferného číslem K-ciferným (K je jistá konstanta stálá po celou dobu výpočtu), což je operace se složitostí $\mathcal{O}((n-K+1)K) = \mathcal{O}(n) = \mathcal{O}(n)$. Celkem tedy $\mathcal{O}(n^2)$. □

2.1.5. *Největší společný dělitel.* Mějme opět n-ciferné číslo a a m-ciferné číslo b. Nejefektivnější metodou na nalezení největšího společného dělitele čísel a a b je Eukleidův algoritmus, dobře známý ze základního kurzu algebry.

Algoritmus 2.8 (Eukleidův algoritmus)**.**
`VSTUP:` $a, b \in \mathbb{Z}$.
`VÝSTUP:` $\mathrm{NSD}(a, b)$.
`1.` $a_1 := \max(a, b)$.
`2.` $a_2 := \min(a, b)$.
`i.` `IF` $a_{i-1} = 0$ `RETURN` a_{i-2}.
$a_i := a_{i-2} \bmod a_{i-1}$.

Tvrzení 2.9. *Algoritmus 2.8 funguje.*

Důkaz. Je potřeba dokázat dvě věci: že algoritmus po konečném počtu kroků skončí, a že na výstupu se objeví $\mathrm{NSD}(a, b)$. Obojí plyne ze vztahu

$$\mathrm{NSD}(a, b) = \mathrm{NSD}(b, a \bmod b)$$

, který platí v libovolném Eukleidovském oboru. Odkazujeme do základního kurzu algebry. □

K výpočtu složitosti Eukleidova algoritmu potřebujeme následující snadné tvrzení. Důkaz přenecháme jako cvičení.

Lemma 2.10. *Nechť $a_1, \dots, a_n \geq 2$. Pak $\sum_{i=1}^{n} L(a_i) \leq C L\left(\prod_{i=1}^{n} a_i\right)$ pro vhodnou konstantu C závisející pouze na bázi.*

Tvrzení 2.11. *Časová složitost Algoritmu 2.8 je $\mathcal{O}((n-d+1)m) \leq \mathcal{O}(nm)$, kde $n = L(a)$, $m = L(b)$, $d = L(\mathrm{NSD}(a, b))$.*

Důkaz. Bez újmy na obecnosti předpokládejme, že $a > b$ a označme ℓ nejmenší číslo splňující $a_\ell = 0$ (tj. algoritmus se zastaví v kroku $i = \ell + 1$). Dále nechť $q_i := a_{i-2}$ div a_{i-1}, neboli $a_{i-2} = q_i a_{i-1} + a_i$, $i = 3, \dots, \ell$. Časovou složitost Eukleidova algoritmu pak lze podle Tvrzení 2.5 vyjádřit vzorcem

$$\mathcal{O}\left(\sum_{i=3}^{\ell} L\left(a_{i-1}\right) L\left(q_i\right)\right).$$

Využijeme odhad $a_{i-1} \leq a_2 = b$, $i = 3, \dots, \ell$:

$$\sum_{i=3}^{\ell} L\left(a_{i-1}\right) L\left(q_i\right) \leq m \cdot L\left(q_\ell\right) \cdot \sum_{i=3}^{\ell-1} L\left(q_i + 1\right).$$

Protože $q_{\ell-2}, q_i + 1 \geq 2$, můžeme podle předchozího lemmatu psát

$$L\left(q_\ell\right) \cdot \sum_{i=3}^{\ell-1} (L\left(q_i + 1\right)) = \mathcal{O}\left(L\left(q_\ell \cdot \prod_{i=3}^{\ell-1} (q_i + 1)\right)\right).$$

Všimneme si, že $q_\ell = a_{\ell-2}/a_{\ell-1}$ a

$$q_i + 1 = \frac{a_i q_i + a_i}{a_i} < \frac{a_{i-1} q_i + a_i}{a_i} = \frac{a_{i-2}}{a_i},$$

a můžeme odhadnout

$$\begin{aligned} q_\ell \cdot \prod_{i=3}^{\ell-1} (q_i + 1) &\leq \frac{a_{\ell-2}}{a_{\ell-1}} \cdot \prod_{i=3}^{\ell-1} \frac{a_{i-2}}{a_i} = \frac{a_{\ell-2} \cdot a_1 a_2 \cdot a_{\ell-3}}{a_{\ell-1} \cdot a_3 a_4 \dots a_{\ell-1}} = \frac{a_1 a_2}{a_{\ell-1}^2} \\ &\leq \left(\frac{a_1}{a_{\ell-1}^2}\right)^2 = \left(\frac{a}{\mathrm{NSD}(a,b)}\right)^2. \end{aligned}$$

Složitost je tedy

$$\mathcal{O}\left(m \cdot L\left(\left(\frac{a}{NSD(a,b)}\right)^2\right)\right) = \mathcal{O}(m(n - d + 1)).$$

$\square$

2.1.6. *Rozšířený Eukleidův algoritmus.* Ze základního kurzu by mělo být známo, že v Eukleidovském oboru existují pro každé prvky a, b nějaké prvky c, d splňující $\mathrm{NSD}(a, b) = ca + db$. Rozšířený Eukleidův algoritmus vrací kromě NSD též tyto koeficienty. Algoritmus prezentujeme pro obecný Eukleidův obor.

Algoritmus 2.12 (Rozšířený Eukleidův algoritmus)**.**
`VSTUP:` $a, b \in \mathbf{R}$, předpokládejme $\nu(a) \geq \nu(b)$.
`VÝSTUP:` $\mathrm{NSD}(a, b)$ a c, d splňující $\mathrm{NSD}(a, b) = ca + db$.
`1.` $(a_1, c_1, d_1) := (a, 1, 0)$,
`2.` $(a_2, c_2, d_2) := (b, 0, 1)$,
`i.` `IF` $a_{i-1} = 0$ `RETURN` $NSD(a, b) = a_{i-2}, c = c_{i-2}, d = d_{i-2}$
$q_i := a_{i-2}$ div a_{i-1},
$(a_i, b_i, c_i) := (a_{i-2} \bmod a_{i-1}, c_{i-2} - q_i c_{i-1}, d_{i-2} - q_i d_{i-1})$,

Tvrzení 2.13. *Je-li* $\mathbf{R}$ *Eukleidovský obor, pak Algoritmus 2.12 funguje.*

Důkaz. Provedeme důkaz pouze pro část týkající se koeficientů c, d; zbytek viz základní kurz. Indukcí dokážeme, že

$$a_i = c_i a + d_i b$$

pro každé i. Pro $i = 1, 2$ je tak postulováno v kroku `1.` Dále, pro $i \geq 3$,

$$\begin{aligned} a_i &= a_{i-2} - q_i a_{i-1} = c_{i-2}a + d_{i-2}b - q_i(c_{i-1}a + d_{i-1}b) \\ &= (c_{i-2} - q_i c_{i-1})a + (d_{i-2} - q_i d_{i-1})b, \end{aligned}$$

a tedy $c_i = c_{i-2} - q_i c_{i-1}$ a $d_i = d_{i-2} - q_i d_{i-1}$. □

Není těžké nahlédnout, že asymptotická časová složitost Algoritmu 2.12 je stejná jako pro klasický Eukleidův algoritmus nad $\mathbf{R}$, protože v i-tém kroku přibyde k dělení prvků a_i, b_i pouze několik operací sčítání a odčítání, jejichž asymptotické složitosti jsou menší nebo stejné než dělení.

2.2. **Operace v oboru racionálních čísel.**

2.2.1. *Sčítání.* Sčítání dvou racionálních čísel probíhá podle vzorce

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}.$$

Tedy je třeba provést tři násobení celých čísel, jedno sčítání, jeden algoritmus na hledání NSD a dvě dělení (kvůli zkrácení na základní tvar). Časová složitost tedy bude

$$\mathcal{O}(n^2),$$

kde n je větší z počtu cifer čitatele a jmenovatele.

2.2.2. *Násobení.* Násobení dvou racionálních čísel probíhá podle vzorce

$$\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}.$$

Tedy je třeba provést dvě násobení celých čísel, jeden algoritmus na hledání NSD a dvě dělení (kvůli zkrácení na základní tvar). Časová složitost bude opět

$$\mathcal{O}(n^2),$$

kde n je větší z počtu cifer čitatele a jmenovatele.

2.2.3. *Dělení.* Dělení dvou racionálních čísel probíhá podle vzorce

$$\frac{\frac{a}{b}}{\frac{c}{d}} = \frac{a}{b} \cdot \frac{d}{c}$$

Dělení se tedy převádí na násobení a má proto stejnou časovou složitost

$$\mathcal{O}(n^2).$$

Zdůrazňujeme, že operace s racionálními čísly jsou řádově pomalejší než s čísly celými, z důvodu krácení na základní tvar (složitost Eukleidova algoritmu dominuje nad sčítáním a násobením). V některých aplikacích se můžeme vzdát požadavku krácení a operace tak výrazně zrychlit. Musíme dát ovšem pozor na to, aby nám nerostly hodnoty v čitateli a jmenovateli do závratných výšin, což je typický důsledek.

2.3. Operace v algebraických rozšířeních $\mathbb{Q}$. Jak už bylo řečeno,

$$\mathbb{Q}(\alpha) \simeq \mathbb{Q}[x]/m_{\alpha,\mathbb{Q}}\mathbb{Q}[x],$$

takže prvky $\mathbb{Q}(\alpha)$ můžeme reprezentovat pomocí polynomů stupně menšího než $n = \deg m_{\alpha,\mathbb{Q}}$ a operace $+, \cdot$ provádíme modulo polynom $m_{\alpha,\mathbb{Q}}$. Z toho lze odhadnout časovou složitost operací — ke sčítání potřebujeme $\mathcal{O}(n)$ operací v $\mathbb{Q}$ a k násobení $\mathcal{O}(n^2)$ (je třeba jednak násobit dané polynomy a následně dělit výsledek polynomem $m_{\alpha,\mathbb{Q}}$).

Protože $\mathbb{Q}(\alpha)$ je těleso, potřebujeme také algoritmus na výpočet inverzu. Mějme polynom $p \in \mathbb{Q}[x]$ stupně méně než n, chceme najít polynom q stupně méně než n takový, že $pq \equiv 1 \pmod{m_{\alpha,\mathbb{Q}}}$. Jinými slovy, takový, že $p(\alpha)q(\alpha) = 1$. Víme, že $\mathrm{NSD}(p, m_{\alpha,\mathbb{Q}}) = 1$, neboť $m_{\alpha,\mathbb{Q}}$ je ireducibilní a $\deg p < \deg m_{\alpha,\mathbb{Q}}$, a tedy pomocí rozšířeného Eukleidova algoritmu lze najít polynomy $a, b \in \mathbb{Q}[x]$ splňující $1 = ap + bm_{\alpha,\mathbb{Q}}$. Dosazením prvku α zjistíme, že $1 = a(\alpha)p(\alpha) + b(\alpha)m_{\alpha,\mathbb{Q}}(\alpha) = a(\alpha)p(\alpha) + 0$. Tedy a je hledaný polynom. Časová složitost invertování je tedy stejná, jako složitost Eukleidova algoritmu.

2.4. Operace v konečných tělesech. Sčítání a násobení v tělesech $\mathbb{F}_p$ netřeba rozebírat, jde o sčítání a násobení celých čísel s následným dělením. Co se týče inverzu, buď můžeme použít způsob analogický předchozímu odstavci, nebo lze užít vzorce z malé Fermatovy věty, která říká, že v $\mathbb{F}_p$ je $a^{-1} = a^{p-2} \bmod p$.

Ostatní konečná tělesa jsou, jak známo, algebraická rozšíření těles $\mathbb{Z}_p$, problém lze tedy řešit analogicky jako pro algebraická rozšíření $\mathbb{Q}$.

2.5. Operace s polynomy. Uvažujme dva polynomy

$$p = \sum_{i=0}^{n} a_i x^i \quad \text{a} \quad q = \sum_{i=0}^{m} b_i x^i$$

v husté reprezentaci, resp. dva polynomy

$$u = \sum_{t \in T_1} a_t t \quad \text{a} \quad v = \sum_{t \in T_2} b_t t$$

v řídké reprezentaci, kde T_1, T_2 jsou nějaké množiny termů, $|T_1| = n$, $|T_2| = m$. Zde tedy m, n neznačí stupeň, nýbrž *počet členů* těchto polynomů. (Pro úplnost předpokládejme, že $a_t = 0$ pro všechna $t \in T_2 \smallsetminus T_1$ a $b_t = 0$ pro všechna $t \in T_1 \smallsetminus T_2$.)

2.5.1. *Sčítání.* Sčítání je v principu stejné jako pro čísla, akorát bez přenosů. V husté reprezentaci probíhá podle vzorce

$$p + q = \Big(\sum_{i=0}^{\max(m,n)} (a_i + b_i) x^i \Big),$$

potřebujeme tedy přesně $\max(m, n) + 1$ operací v okruhu R. Podobně, v řídké reprezentaci probíhá podle vzorce

$$u + v = \Big(\sum_{t \in T_1 \cup T_2} (a_t + b_t) t \Big),$$

potřebujeme tedy nejvýše $\min(m, n)$ operací v okruhu R (přičítání nuly nepovažujeme za operaci).

2.5.2. *Násobení.* V husté reprezentaci můžeme použít vzorec

$$p \cdot q = \sum_{i=0}^{m+n} \Big(\sum_{j=0}^{i} a_j b_{i-j} \Big) x^i,$$

ve kterém provádíme $1+2+\cdots+(m+n) = \mathcal{O}((m+n)^2) = \mathcal{O}(\max(m,n)^2)$ operací v okruhu R. Při jemnějším výpočtu (všimněte si, že $a_i = 0$ pro $i > n$, resp. $b_i = 0$ pro $i > m$) dostáváme časovou složitost $1+2+\cdots+\min(m,n)+\cdots+\min(m,n) = \mathcal{O}(mn)$.

Nebo můžeme použít školského násobení, které proběhne podobně jako pro čísla, ale bez přenosů. Zde opět dostáváme časovou složitost

$$\mathcal{O}(mn).$$

Výhoda je, že tato metoda funguje i v řídké reprezentaci, kde dostáváme analogickou složitost *vzhledem k počtu členů.*

V případě husté reprezentace se pak vyplatí použít Karacubovo násobení (v algoritmu je pouze potřeba nahradit mocniny B^i za x^i). Tím získáme algoritmus se složitostí

$$\mathcal{O}(\max(m,n)^{\log_2 3}).$$

Cvičení. Aplikujte všechny tři postupy na výpočet součinu

$$(x^{10} + x + 1) \cdot (x^3 - x).$$

Poznamenejme, že v některých okruzích $\mathbf{R}$ funguje ještě řádově rychlejší algoritmus, se složitostí $\mathcal{O}(\max(m,n) \log \max(m,n))$, založený na modulární reprezentaci a Rychlé Fourierově transformaci. Ukážeme si jej později.

2.5.3. *Dělení a NSD nad tělesem.* Uvažujme polynomy, jejichž koeficienty jsou prvky tělesa. Podíl a zbytek jsou, jak známo, jednoznačně určeny a k jejich výpočtu lze použít školský algoritmus.

$$\boxed{a_n x^n + \ldots + a_0} : \boxed{b_m x^m + \ldots + b_0} = \boxed{c_{n-m} x^{n-m} + \ldots + c_0}$$

Podobně jako u dělení čísel je třeba $n - m + 1$ kroků, v každém z nich provádíme následující operace:

(1) Urči koeficient c_i jako podíl vedoucího koeficientu právě děleného polynomu a prvku $\mathrm{lc}(q)$.
(2) Odečti od děleného polynomu polynom $c_i \cdot q$.

V prvním kroku provádíme pouze jednu operaci v tělese, v druhém pak nejprve násobíme polynom q konstantou c_i, což spotřebuje $m+1$ operací, a následně odečítáme tento polynom stupně m od děleného polynomu, což spotřebuje $m+1$ operací. Celková časová složitost je tedy $(m+1)(n-m+1) = \mathcal{O}(m(n-m+1))$ jako u celých čísel.

Cvičení. Jak jistě znáte ze základního kurzu algebry, největší společný dělitel dvou polynomů jedné proměnné nad tělesem lze vypočítat pomocí Eukleidova algoritmu. Spočtěte jeho časovou složitost pro vstupní polynomy stupňů m, n. Můžete se inspirovat důkazem Tvrzení 2.11. Uvědomte si, že rozšířený Eukleidův algoritmus funguje pro polynomy také.

V praxi se bohužel často setkáváme s polynomy, jejichž koeficienty nejsou prvky tělesa: jde zejména o polynomy nad celými čísly a o polynomy více proměnných (které můžeme chápat jako polynomy jedné proměnné nad $\mathbf{R}[x_1, \ldots, x_{n-1}]$). V těchto oborech vůbec nemusí podíl existovat.

Příklad.

- V $\mathbb{Z}[x]$ neexistuje podíl polynomů $x^2 - 1$ a $2x - 2$. Při řešení rovnosti $(x^2 - 1) = q(2x - 2) + r$ (kde $\deg r < \deg(2x - 2) = 1$) docházíme k tomu, že koeficient u x je sudý, spor.
- V $\mathbf{R}[x, y] = (\mathbf{R}[x])[y]$ neexistuje podíl polynomů y^2 a $xy + 1$. Při řešení rovnosti $y^2 = q(xy + 1) + r$ (kde $\deg r < 1$ ve smyslu polynomu jedné proměnné y) docházíme k tomu, že koeficient u y^2 je dělitelný x, spor.

Tento problém lze vyřešit pomocí tzv. *pseudodělení*, které bude diskutováno v kapitole věnované největšímu společnému děliteli polynomů nad obecným Gaussovým oborem. (Připomeňme, že tzv. Gaussova věta, která se v základním kurzu často nedokazuje, říká, že obor polynomů nad Gaussovým oborem je Gaussův obor.)

2.5.4. *Vyhodnocování polynomů.* Nechť $p = \sum_{i=0}^{n} a_i x^i \in \mathbf{R}[x]$ a $\alpha \in \mathbf{R}$. Zajímá nás otázka, jak efektivně vyčíslit hodnotu polynomu p v bodě α.

Na první pohled by čtenáře mohlo napadnout prostě dosadit písmeno α místo písmena x a výsledný výraz v okruhu $\mathbf{R}$ vyhodnotit tak jak leží, tj. jako $a_0 + a_1\alpha + a_2\alpha^2 + \ldots + a_n\alpha^n$. Budeme-li si pamatovat předchozí mocninu α, pak potřebujeme n sčítání a $2n$ násobení v $\mathbf{R}$, neboť v každém kroku násobíme $a_i \cdot \alpha^{i-1} \cdot \alpha$.

Efektivnějším způsobem je tzv. *Hornerovo schéma*

$$p(\alpha) = a_0 + \alpha(a_1 + \alpha(\ldots \alpha(a_{n-1} + \alpha a_n))).$$

Vidíte, že takto potřebujeme n sčítání a pouze n násobení.

Modulární reprezentace a rychlé násobení a dělení

3. Modulární reprezentace

Připomeňme si dvě dobře známé věty.

Věta 3.1 (Čínská věta o zbytcích). *Nechť $m_1, \dots, m_n$ jsou po dvou nesoudělná čísla, označme $m = m_1 \cdot \ldots \cdot m_n$. Pak pro každá $u_1, \dots, u_n \in \mathbb{Z}$ existuje právě jedno řešení $x \in \{0, 1, \dots, m-1\}$ soustavy*

$$\begin{aligned} x &\equiv u_1 \pmod{m_1} \\ x &\equiv u_2 \pmod{m_2} \\ &\vdots \\ x &\equiv u_n \pmod{m_n} \end{aligned}$$

Věta 3.2 (o interpolaci). *Nechť $\alpha_0, \alpha_1, \dots, \alpha_n$ jsou po dvou různé prvky tělesa $\mathbf{T}$. Pak pro každá $u_0, u_1, \dots, u_n \in \mathbf{T}$ existuje právě jeden polynom $p \in \mathbf{T}[x]$ stupně nejvýše n splňující*

$$\begin{aligned} p(\alpha_0) &= u_0 \\ p(\alpha_1) &= u_1 \\ &\vdots \\ p(\alpha_n) &= u_n \end{aligned}$$

Navíc si všimněte, že zobrazení

$$\mathbb{Z}_m \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_n}, \quad x \mapsto (x \bmod m_1, \dots, x \bmod m_n)$$

je homomorfismus, a Čínská věta o zbytcích říká, že to je bijekce, tedy izomorfismus. I v druhém případě, zobrazení

$$\mathbf{T}[x] \to \mathbf{T} \times \cdots \times \mathbf{T}, \quad p \mapsto (p(\alpha_0), \dots, p(\alpha_n))$$

je homomorfismus, a věta o interpolaci říká, že to je bijekce na množině všech polynomů stupně nejvýše n. Tato pozorování motivují následující definici.

Definice. *Modulární reprezentací* okruhu $\mathbf{R}$ podle ideálů $\mathbf{M}_1, \dots, \mathbf{M}_n$ rozumíme epimorfismus

$$\begin{gathered} \varphi : \mathbf{R} \twoheadrightarrow \mathbf{R}/\mathbf{M}_1 \times \ldots \times \mathbf{R}/\mathbf{M}_n \\ x \mapsto (x + \mathbf{M}_1, \dots, x + \mathbf{M}_n) \end{gathered}$$

Modulární reprezentaci nazveme *věrnou*, pokud je φ izomorfismus.

Obvykle se volí $\mathbf{M}_1, \dots, \mathbf{M}_n$ hlavní ideály, tj. $\mathbf{M}_i = m_i \mathbf{R}$ pro nějaká $m_i \in \mathbf{R}$. Budeme psát zkráceně $\mathbf{R}/m_i$ místo $\mathbf{R}/m_i\mathbf{R}$.

Pokud obor $\mathbf{R}$ umožňuje dělit se zbytkem (jako např. $\mathbb{Z}$ nebo polynomy nad tělesem), můžeme ztotožnit prvky $\mathbf{R}/m_i$ s vybranými reprezentanty zbytků modulo m_i. Modulární reprezentaci pak můžeme psát ve tvaru

$$\begin{gathered} \varphi : \mathbf{R} \twoheadrightarrow \mathbf{R}/m_1 \times \ldots \times \mathbf{R}/m_n \\ x \mapsto (x \bmod m_1, \dots, x \bmod m_n). \end{gathered}$$

Např. prvky $\mathbb{Z}/m$ ztotožníme s reprezentanty zbytků modulo m, které se obvykle volí $0,\dots,m-1$ (alternativně bychom mohli vzít např. $-\frac{m}{2}+1,\dots,\frac{m}{2}$ pro m sudé, resp. $-\frac{m-1}{2},\dots,\frac{m-1}{2}$ pro m liché). Podobně, prvky $\mathbf{T}[x]/(x-\alpha)$ ztotožníme s prvky tělesa $\mathbf{T}$, neboť

$$p \bmod (x-\alpha) = p(\alpha) \in \mathbf{T}$$

(dokažte dosazením hodnoty α do definice podílu a zbytku!).

Příklad. Zobrazení

$$\mathbb{Z} \to \mathbb{Z}/m_1 \times \ldots \times \mathbb{Z}/m_n, \quad x \mapsto (x \bmod m_1, \dots, x \bmod m_n)$$

je modulární reprezentací přirozených čísel.

Příklad. Zobrazení

$$\mathbf{T}[x] \to \mathbf{T}[x]/(x-\alpha_0) \times \ldots \times \mathbf{T}[x]/(x-\alpha_n), \quad p \mapsto (p(\alpha_0), \dots, p(\alpha_n))$$

je modulární reprezentací polynomů nad tělesem $\mathbf{T}$.

K čemu jsou modulární reprezentace? Princip tzv. *modulární metody* je, že daný problém neřešíme v daném komplikovaném oboru $\mathbf{R}$ (tj. ve standardní reprezentaci), nýbrž v několika menších jednodušších oborech $\mathbf{R}/m_i$ (tj. v modulární reprezentaci). Přitom některé operace jsou v modulární reprezentaci mnohem rychlejší, než v klasické. Typickým příkladem je násobení: máme-li dán polynom stupně n nad tělesem $\mathbf{T}$ (resp. n-ciferné číslo), ve standardní reprezentaci potřebujeme $\mathcal{O}(n^2)$ operací v $\mathbf{T}$ (resp. operací s ciframi). Je-li ovšem tento polynom (číslo) v modulární reprezentaci, tj. reprezentován vektorem délky $n+1$ svých hodnot v nějakých bodech z $\mathbf{T}$ (resp. n zbytky modulo jednociferná čísla m_i), stačí nám k násobení pouze $n+1$ operací v tělese $\mathbf{T}$ (resp. $2n$ operací s jednocifernými čísly) — vektory stačí vynásobit po složkách, neboť modulární reprezentace je *homomorfismus*!

Vidíme tedy, že násobení (a další úlohy) jsou v modulární reprezentaci řádově rychlejší. Ovšem uživatel zpravidla vyžaduje vstup i výstup (polynom, číslo) v reprezentaci standardní. Problém tedy je, nalézt rychlý algoritmus na převod mezi standardní a modulární reprezentací. V obecném případě nalezneme algoritmy (*Lagrangeův* a *Garnerův*) s časovou složitostí kvadratickou (tedy pro rychlé násobení bezcenné, ovšem cenné pro jiné úlohy). V jistém speciálním případě pro polynomy pak nalezneme algoritmus s časovou složitostí $\mathcal{O}(n \log n)$ (tzv. *rychlá Fourierova transformace*), čímž získáme asymptoticky nejrychlejší známý algoritmus pro násobení polynomů.

4. Zobecněná Čínská věta o zbytcích

Dříve než pokročíme ke zmíněným algoritmům, dokážeme tzv. *zobecněnou Čínskou větu o zbytcích*, která zahrnuje jak obě věty zmíněné v úvodu této kapitoly, tedy klasickou ČVZ a větu o interpolaci polynomů, tak mnoho dalších situací. Tato věta říká, na jakém oboru se daná modulární reprezntace chová věrně, proto též název *Věta o modulární reprezentaci.*

Připomeňme, že jsou-li $\mathbf{A}, \mathbf{B}$ ideály komutativního okruhu $\mathbf{R}$, pak

$$\mathbf{A} + \mathbf{B} = \{a + b : a \in \mathbf{A}, b \in \mathbf{B}\}$$

také tvoří ideál a je to nejmenší ideál obsahující $A \cup B$. Potřebujeme následující pomocné lemma.

Lemma 4.1. *Nechť* $\mathbf{R}$ *je komutativní okruh s jednotkou a* $\mathbf{M}, \mathbf{M}_1, \ldots, \mathbf{M}_n$ *jeho ideály splňující* $\mathbf{M} + \mathbf{M}_i = \mathbf{R}$ *pro všechna* i*. Označme*

$$\mathbf{K} = \bigcap_{i=1}^{n} \mathbf{M}_i.$$

Pak $\mathbf{M} + \mathbf{K} = \mathbf{R}$.

Důkaz. Nejprve si uvědomte, že stačí dokázat, že $1 \in \mathbf{M} + \mathbf{K}$. V tom případě, $1 = a + b$ pro nějaká $a \in \mathbf{M}$, $b \in \mathbf{K}$ a libovolné $r \in \mathbf{R}$ lze psát jako $r = r \cdot 1 = ra + rb \in \mathbf{M} + \mathbf{K}$.

Víme, že pro všechna i platí $\mathbf{M} + \mathbf{M}_i = \mathbf{R}$, tedy existují $c_i \in \mathbf{M}$ a $d_i \in \mathbf{M}_i$ splňující $1 = c_i + d_i$. Pak ovšem

$$\begin{aligned} 1 &= 1 \cdot \ldots \cdot 1 = (c_1 + d_1) \cdot \ldots \cdot (c_n + d_n) = \\ &= \big(c_1 c_2 \cdots c_n + c_1 \cdots c_{n-1} d_n + \ldots + d_1 \cdots d_{n-1} c_n\big) + \big(d_1 \cdots d_n\big). \end{aligned}$$

Ovšem první závorka je prvkem $\mathbf{M}$, protože v každém sčítanci je nějaké $c_k \in \mathbf{M}$ (a podle definice ideálu, libovolný jeho násobek tam je a jejich součet též). A druhá závorka, tj. $d_1 \cdots d_n$, náleží $\mathbf{K}$, neboť pro každé i máme $d_i \in \mathbf{M}_i$, tedy i $d_1 \cdots d_n \in \mathbf{M}_i$, a proto $d_1 \cdots d_n \in \bigcap_{i=1}^{n} \mathbf{M}_i = \mathbf{K}$. Takže $1 \in \mathbf{M} + \mathbf{K}$. □

Věta 4.2 (o modulární reprezentaci, zobecněná Čínská věta o zbytcích)**.** *Nechť* $\mathbf{R}$ *je komutativní okruh s jednotkou,* $\mathbf{M}_1, \ldots, \mathbf{M}_n$ *jeho ideály splňující* $\mathbf{M}_i + \mathbf{M}_j = \mathbf{R}$ *pro všechna* $i \neq j$ *a označme*

$$\mathbf{M} = \bigcap_{i=1}^{n} \mathbf{M}_i.$$

Pak je zobrazení

$$\mathbf{R}/\mathbf{M} \to \mathbf{R}/\mathbf{M}_1 \times \ldots \times \mathbf{R}/\mathbf{M}_n$$

$$x + \mathbf{M} \mapsto (x + \mathbf{M}_1, \ldots, x + \mathbf{M}_n)$$

izomorfismus.

Důkaz. Uvažujme zobrazení

$$\varphi : \mathbf{R} \to \mathbf{R}/\mathbf{M}_1 \times \ldots \times \mathbf{R}/\mathbf{M}_n$$

$$x \mapsto (x + \mathbf{M}_1, \ldots, x + \mathbf{M}_n).$$

Je to homomorfismus, protože v každé složce máme přirozenou projekci $\mathbf{R} \twoheadrightarrow \mathbf{R}/\mathbf{M}_i$. Nejprve dokážeme, že $\mathbf{Ker}(\varphi) = \mathbf{M}$:

$$\begin{aligned} \mathrm{Ker}(\varphi) &= \{x \in \mathbf{R} : \varphi(x) = (0, \ldots, 0)\} = \{x \in \mathbf{R} : x \in \mathbf{M}_1, \ldots, x \in \mathbf{M}_n\} \\ &= \{x \in \mathbf{R} : x \in \bigcap_{i=1}^{n} \mathbf{M}_i\} = \{x \in \mathbf{R} : x \in \mathbf{M}\} = \mathbf{M}. \end{aligned}$$

Tedy podle 1. věty o izomorfismu platí $\mathbf{R}/\mathbf{M} \simeq \mathbf{Im}(\varphi)$, s izomorfismem uvedeným ve znění věty. Zbývá dokázat, že $\mathbf{Im}(\varphi) = \mathbf{R}/\mathbf{M}_1 \times \ldots \times \mathbf{R}/\mathbf{M}_n$. Zvolíme $u_1, \ldots, u_n \in \mathbf{R}$ a zkonstruujeme $x \in \mathbf{R}$ splňující

$$\varphi(x) = (u_1 + \mathbf{M}_1, \ldots, u_n + \mathbf{M}_n).$$

Označme $\mathbf{K}_i = \bigcap_{j \neq i} \mathbf{M}_j$. Podle Lemmatu 4.1 platí $\mathbf{M}_i + \mathbf{K}_i = \mathbf{R}$, tedy $1 \in \mathbf{M}_i + \mathbf{K}_i$, a proto existují $a_i \in \mathbf{M}_i$, $c_i \in \mathbf{K}_i$ splňující $1 = a_i + c_i$. Položme

$$x = \sum_{i=1}^{n} c_i u_i.$$

Dokážeme, že pro všechna j platí $x + \mathbf{M}_j = u_j + \mathbf{M}_j$, neboli že $x - u_j \in \mathbf{M}_j$. Rozepíšeme si

$$x - u_j = \sum_{i=1}^{n} c_i u_i - u_j = \Big(\sum_{i \neq j} c_i u_i\Big) + (c_j - 1) u_j.$$

Protože $c_i \in \mathbf{M}_j$ pro všechna $i \neq j$, máme také $c_i u_i \in \mathbf{M}_j$ pro všechna $i \neq j$, a tedy celá suma $\sum_{i \neq j} c_i u_i \in \mathbf{M}_j$. Navíc $c_j - 1 = -a_j \in \mathbf{M}_j$, tedy i $(c_j - 1) u_j \in \mathbf{M}_j$. Tedy také $x - u_j$ náleží do $\mathbf{M}_j$. □

Důsledek 4.3. *Nechť* $\mathbf{R}$ *je Eukleidovský obor,* $m_1, \ldots, m_n \in \mathbf{R}$ *jeho prvky splňující* $\mathrm{NSD}(m_i, m_j) = 1$ *pro všechna* $i \neq j$ *a označme* $m = m_1 \cdot \ldots \cdot m_n$. *Pak je zobrazení*

$$\mathbf{R}/m \to \mathbf{R}/m_1 \times \ldots \times \mathbf{R}/m_n, \quad x \bmod m \mapsto (x \bmod m_1, \ldots, x \bmod m_n)$$

izomorfismus.

Důkaz. V předchozí větě dosaďte za $\mathbf{M}_i$ hlavní ideál $m_i\mathbf{R}$ a připomeňte si zavedené ztotožnění prvků $\mathbf{R}/m_i$ s reprezentanty zbytkových tříd modulo m_i. Pro úplnost je třeba ověřit, že (1) $\mathbf{M}_i + \mathbf{M}_j = \mathbf{R}$ pro všechna $i \neq j$ a (2) $\bigcap_{i=1}^n \mathbf{M}_i = m\mathbf{R}$.

(1) Protože je obor $\mathbf{R}$ Eukleidovský, z rozšířeného Eukleidova algoritmu plyne, že existují $c, d \in \mathbf{R}$ splňující $1 = \mathrm{NSD}(m_i, m_j) = cm_i + dm_j \in \mathbf{M}_i + \mathbf{M}_j$. Tedy $\mathbf{M}_i + \mathbf{M}_j = \mathbf{R}$.

(2) Jinými slovy, chceme dokázat, že prvek a je dělitelný všemi m_i právě tehdy, když je dělitelný m. Což plyne z nesoudělnosti prvků m_i. (Připomeňme, že a je prvek $m\mathbf{R}$ právě tehdy, když $m \mid a$.) □

Příklad. Pro $\mathbf{R} = \mathbb{Z}$ dostáváme Čínskou větu o zbytcích.

Příklad. Pro $\mathbf{R} = \mathbf{T}[x]$ a $m_i = x - \alpha_i$ dostáváme upřesnění věty o interpolaci: modulární reprezentace je věrná, bereme-li polynomy modulo polynom

$$m = (x - \alpha_0) \cdot \ldots \cdot (x - \alpha_n)$$

stupně $n + 1$. Neboli

$$\mathbf{T}[x]/m \simeq \mathbf{T}^{n+1}, \quad p \mapsto (p(\alpha_0), \ldots, p(\alpha_n)).$$

I zde je třeba považovat p za reprezentanta zbytkových tříd modulo m, tedy $\deg p < n + 1$.

5. Algoritmy na Čínskou větu o zbytcích

Uvažujme situaci popsanou Důsledkem 4.3 a zejména dva výše uvedené konkrétní případy. Algoritmus pro převod ze standardní do modulární reprezentace je očividný: počítáme n zbytků po dělení prvky $m_1, \ldots, m_n$. Otázka je, jak provést efektivně opačný převod.

V případě polynomů vede úloha přímočaře na řešení soustavy lineárních rovnic s tzv. Vandermondovou maticí: chceme-li najít polynom $p = \sum_{i=0}^{n} a_i x^i$ splňující $p(\alpha_0) = u_0, \ldots, p(\alpha_n) = u_n$, dostáváme soustavu

$$\left(\begin{array}{ccccc|c} \alpha_0^0 & \alpha_0^1 & \alpha_0^2 & \ldots & \alpha_0^n & u_0 \\ \alpha_1^0 & \alpha_1^1 & \alpha_1^2 & \ldots & \alpha_1^n & u_1 \\ \vdots & & & & & \vdots \\ \alpha_n^0 & \alpha_n^1 & \alpha_n^2 & \ldots & \alpha_n^n & u_n \end{array}\right)$$

Jak známo, determinant Vandermondovy matice je roven $\prod_{i<j}(\alpha_i - \alpha_j) \neq 0$, existuje tedy právě jedno řešení této soustavy.

Právě popsaná metoda jednak není příliš efektivní, a navíc ji nelze aplikovat v druhém případě, totiž na řešení soustav kongruenčních rovnic. Předvedeme dva obecné algoritmy.

První z nich je de facto popsán v důkazu zobecněné ČVZ. Pouze je třeba přeložit symbolický zápis z řeči ideálů do řeči prvků oboru.

Algoritmus 5.1 (Lagrangeův)**.**
`VSTUP:` $m_1, \ldots, m_n \in \mathbf{R}$, $u_1, \ldots, u_n \in \mathbf{R}$.
`VÝSTUP:` Řešení soustavy $x \equiv u_i \pmod{m_i}$, $i = 1, \ldots, n$.
`1.` Polož $m := m_1 \cdot \ldots \cdot m_n$ a $k_i := \frac{m}{m_i}$, $i = 1, \ldots, n$.
`2.` Najdi a_i, b_i splňující $a_i m_i + b_i k_i = 1$, $i = 1, \ldots, n$.
`3.` `RETURN` $x = \sum_{i=1}^{n} b_i k_i u_i$.

Poznamenejme, že je-li $\mathbf{R}$ Eukleidovský obor, lze použít v kroku `2.` rozšířený Eukleidův algoritmus, neboť $\mathrm{NSD}(m_i, k_i) = 1$. V případě $\mathbf{R} = \mathbf{T}[x]$ a $m_i = x - \alpha_i$ je však efektivnější následující trik: položme

$$b_i = \frac{1}{k_i(\alpha_u)} = \frac{1}{\prod_{j\neq i}(\alpha_i - \alpha_j)}$$

(všimněte si, že a_i vlastně k ničemu nepotřebujeme). Potřebujeme ověřit, že $b_i k_i \equiv 1 \pmod{x - \alpha_i}$, tedy že $b_i k_i(\alpha_i) = 1$. Ovšem $k_i = \prod_{j\neq i}(x - \alpha_j)$, takže to platí.

Tvrzení 5.2. *Je-li* $\mathbf{R}$ *Eukleidovský obor a* $m_1, \ldots, m_n$ *jeho po dvou nesoudělné prvky, pak Algoritmus 5.1 funguje.*

Důkaz. Jak již bylo uvedeno, postup odpovídá důkazu Věty 4.2. □

Tvrzení 5.3. *Nechť* $\mathbf{T}$ *těleso,* $\alpha_1, \ldots, \alpha_n \in \mathbf{T}$. *Jestliže* $\mathbf{R} = \mathbf{T}[x]$ *a* $m_i = x - \alpha_i$, *pak má Algoritmus 5.1 časovou složitost* $\mathcal{O}(n^2)$.

Důkaz. Probereme jednotlivé kroky. `1.` Nejprve se vypočte $m = m_1 \cdot \ldots \cdot m_n$, což je $n-1$ násobení polynomu stupně nejvýše $n-1$ polynomem stupně 1, tedy časová složitost je $\mathcal{O}(n^2)$. Následně počítáme $k_i = \frac{m}{m_i}$, tedy n podílů polynomu stupně n polynomem stupně 1, takže časová složitost je $n\mathcal{O}(n) = \mathcal{O}(n^2)$. `2.` Provádí se výpočet $b_1, \ldots, b_n$. Podle výše uvedeného vzorce, každé b_i je inverzem součinu $n-1$ prvků tělesa, tedy časová složitost je opět $n\mathcal{O}(n) = \mathcal{O}(n^2)$. `3.` Poslední krok zahrnuje $n-1$ sčítání, přičemž každý sčítanec je součin polynomu n_i stupně $n-1$ a konstant $b_i, u_i \in \mathbf{T}$. Vidíme, že i tento krok má složitost $\mathcal{O}(n^2)$. □

Tvrzení 5.4. *Jestliže* $\mathbf{R} = \mathbb{Z}$ *a* m_i *jsou jednociferná čísla, pak má Algoritmus 5.1 časovou složitost* $\mathcal{O}(n^2)$.

Důkaz. Analogicky, přičemž místo polynomů stupně 1 vystupují jednociferná čísla. Rozdílný je krok `2.`, kde provádíme n-krát rozšířený Eukleidův algoritmus. Podle Tvrzení 2.11 je složitost Eukleidova algoritmu provedeného na n-ciferné a jednociferné číslo $\mathcal{O}(n)$, tedy i tento krok má složitost $\mathcal{O}(n^2)$. □

Příklad. Řešte v $\mathbb{Z}$ soustavu $x \equiv 0 \pmod 2$, $x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 5$.

i	u_i	m_i	k_i	a_i	b_i	$b_i n_i$
1	0	2	15	-7	1	15
2	2	3	10	-3	1	10
3	3	5	6	-1	1	6

Protože $m = 2 \cdot 3 \cdot 5 = 30$, dostáváme jako řešení každé

$$x \equiv 15 \cdot 0 + 10 \cdot 2 + 6 \cdot 3 = 38 \equiv 8 \pmod{30}.$$

Příklad. Najděte $p \in \mathbb{Z}_5[x]$ stupně nejvýše 2 tak, aby $p(0) = 2$, $p(1) = 2$, $p(2) = 1$.

i	u_i	m_i	k_i	b_i
1	2	x	$(x-1)(x-2)$	$\frac{1}{(-1)(-2)} = 3$
2	2	$(x-1)$	$x(x-2)$	$\frac{1}{1(-1)} = 4$
3	1	$(x-2)$	$x(x-1)$	$\frac{1}{2 \cdot 1} = 3$

Hledaný polynom je tedy

$$x = 6(x-1)(x-2) + 8x(x-2) + 3x(x-1) = 2x^2 + 3x + 2.$$

Jednou z výhod Lagrangeova algoritmu je, že při změně hodnot u_i stačí opakovat jen krok `3.` Jinou pěknou vlastností je snadná paralelizace Lagrangeova algoritmu (tedy úprava do formy vhodné pro velké paralelní počítače): výpočet x v kroku `3.` lze napsat pro n sudé jako

$$x = \Big(\sum_{i=1}^{n/2} b_i' n_i' u_i\Big) \cdot m_{n/2+1} \cdots m_n + \Big(\sum_{i=n/2+1}^{n} b_i' n_i' u_i\Big) \cdot m_1 \cdots m_{n/2},$$

kde $n_i' = \frac{m_1 \cdots m_{n/2}}{m_i}$ pro $i = 1, \dots, n/2$, $n_i' = \frac{m_{n/2+1} \cdots m_n}{m_i}$ pro $i = n/2+1, \dots, n$ a a_i', b_i' jsou prvky splňující $a_i' m_i + b_i' n_i' = 1$. Všimněte si, že první i druhou polovinu výrazu lze počítat nezávisle, tedy lze aplikovat metodu *rozděl a panuj*, velmi vhodnou pro paralelizaci. Není těžké ověřit, že výsledná složitost je opět kvadratická.

Uvedené výhody Lagrangeova algoritmu jsou zneváženy jednou velkou nevýhodou: při zvýšení n (tj. při přidání dodatečné podmínky) se musí celý výpočet opakovat. To je poměrně častá situace při práci s modulárními reprezentacemi, neboť v mnoha úlohách v průběhu výpočtu výrazně rostou velikosti proměnných. Následující algoritmus tuto vadu nemá. (Poznamenejme, že jeho asympotická časová složitost je stejná.)

Trik Garnerova algoritmu je následující:
Je-li a_n řešením soustavy $x \equiv u_i \pmod{m_i}$, $i = 1, \dots, n$ a přidáme-li novou rovnici $x \equiv u_{n+1} \pmod{m_{n+1}}$, pak řešení nové soustavy hledáme ve tvaru

$$a_{n+1} = a_n + m_1 \cdot \ldots \cdot m_n b_n$$

pro jisté b_n. Všimněte si, že a_{n+1} je řešením původní soustavy pro jakékoliv b_n, neboť druhý sčítanec je dělitelný každým m_i, $i = 1, \dots, n$. Aby $a_{n+1} \equiv u_{n+1} \pmod{m_{n+1}}$, musí platit

$$\begin{aligned} a_n + m_1 \cdot \ldots \cdot m_n b_n &\equiv u_{n+1} \pmod{m_{n+1}} \\ m_1 \cdot \ldots \cdot m_n b_n &\equiv u_{n+1} - a_n \pmod{m_{n+1}}, \end{aligned}$$

tedy stačí (a je nutné), aby $b_n \equiv (u_{n+1} - a_n)c_n \pmod{m_{n+1}}$, kde c_n je prvek, pro který $c_n \cdot m_1 \dots m_n \equiv 1 \pmod{m_{n+1}}$.

Algoritmus 5.5 (Garnerův).
`VSTUP:` $m_1, \dots, m_n \in \mathbf{R}$, $u_1, \dots, u_n \in \mathbf{R}$.
`VÝSTUP:` Řešení soustavy $x \equiv u_i \pmod{m_i}$, $i = 1, \dots, n$.
`0.` $(a_0, b_0, k_0) := (0, u_1, 1)$
`i.` $(a_i, b_i, k_i) := (a_{i-1} + b_{i-1}k_{i-1}, (u_{i+1} - a_i)c_i \bmod m_{i+1}, m_i k_{i-1})$, kde
c_i je prvek splňující $k_i c_i \equiv 1 \pmod{m_{i+1}}$
`n.` `RETURN` $a_n := a_{n-1} + b_{n-1}k_{n-1}$.

Poznamenejme (Lemma 4.1), že je-li $\mathbf{R}$ Eukleidovský obor a prvky $m_1, \dots, m_n$ jsou po dvou nesoudělné, pak je $k_i = m_1 \cdot \ldots \cdot m_i$ nesoudělný s m_{i+1}, a tedy prvek c_i lze nalézt pomocí rozšířeného Eukleidova algoritmu:

$$1 = \mathrm{NSD}(k_i, m_{i+1}) = c_i k_i + d_i m_{i+1}.$$

(Všimněte si, že modulo m_{i+1} dostáváme $1 \equiv c_i k_i \pmod{m_{i+1}}$.) V našich dvou speciálních případech lze postupovat efektivněji:

(1) v případě $\mathbf{R} = \mathbb{Z}$ vlastně hledáme inverz prvku k_i v grupě $\mathbb{Z}^*_{m_{i+1}}$. Podle Eulerovy věty můžeme volit
$$c_i = k_i^{\varphi(m_{i+1})-1}.$$
(2) v případě $\mathbf{R} = \mathbf{T}[x]$ a $m_i = x - \alpha_i$ lze volit (podobně jako u Lagrangeova algoritmu)
$$c_i = \frac{1}{k_i(\alpha_{i+1})} = \frac{1}{(\alpha_{i+1} - \alpha_1) \cdot \ldots \cdot (\alpha_{i+1} - \alpha_i)}.$$
(Ověřte!)

Tvrzení 5.6. *Je-li $\mathbf{R}$ Eukleidovský obor a $m_1, \dots, m_n$ jeho po dvou nesoudělné prvky, pak Algoritmus 5.5 funguje.*

Důkaz. Všimněte si, že
$$k_i = m_1 \cdot \ldots \cdot m_i$$
a v i-tém kroku platí
$$a_{i+1} = a_i + b_i k_i \equiv u_j \pmod{m_j}, \quad j = 1, \dots, i+1,$$
podle výpočtu uvedeného před algoritmem. □

Tvrzení 5.7. *Nechť $\mathbf{R} = \mathbb{Z}$ nebo $\mathbf{R} = \mathbf{T}[x]$, $\mathbf{T}$ těleso, a předpokládejme, že délka všech m_i je 1 (tj. jsou to jednociferná čísla, resp. polynomy stupně 1). Pak má Algoritmus 5.5 časovou složitost $\mathcal{O}(n^2)$.*

Důkaz. Algoritmus probíhá v $n+1$ krocích. Přitom každý z nich má časovou složitost $\mathcal{O}(n)$: prověřte, že každá operace je buď sčítání a odčítání prvků délky nejvýše n, nebo násobení prvku délky nejvýše n prvkem délky 1, nebo počítání zbytku modulo prvek délky 1. S vyjímkou výpočtu c_i, kde voláme rozšířený Eukleidův algoritmus na prvky délky n a 1: ovšem podle Tvrzení 2.11 má i tento výpočet složitost $\mathcal{O}(n)$. □

Jak je vidět, výhodou Garnerova algoritmu oproti Lagrangeovu je, že dodatečné přidání jedné podmínky znamená prodloužení výpočtu o pouhý jeden krok, který proběhne v lineárním čase. Tento fakt si ilustrujeme na následujících příkladech, kdy k třem podmínkám použitých v příkladech výše přidáme čtvrtou, které vyhovuje stejné řešení.

Příklad. Řešte v $\mathbb{Z}$ soustavu $x \equiv 1 \pmod 2$, $x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 5$. Řešte tuto soustavu doplněnou o $x \equiv 1 \pmod 7$.

i	u_i	m_i	k_i	c_i	a_i	b_i
0	—	—	1	—	0	1
1	1	2	2	2	$1 = 0 + 1 \cdot 1$	$2 = (2-1) \cdot 2$
2	2	3	6	1	$5 = 1 + 2 \cdot 2$	$3 = (3-5) \cdot 1 \bmod 5$
3	3	5	30	4	$23 = 5 + 3 \cdot 6$	$3 = (1-23) \cdot 4 \bmod 7$
4	1	7	—	—	$113 = 23 + 3 \cdot 30$	—

Řešením prvních třech rovnic tedy je každé $x \equiv 23 \pmod{30}$. Řešením všech čtyřech pak je každé $x \equiv 113 \pmod{210}$.

Příklad. Najděte $p \in \mathbb{Z}_5[x]$ stupně nejvýše 2 tak, aby $p(0) = 2$, $p(1) = 2$, $p(2) = 1$. Najděte $p \in \mathbb{Z}_5[x]$ stupně nejvýše 3 splňující navíc $p(4) = 2$.

i	u_i	m_i	k_i	c_i	a_i	b_i
0	—	—	1	—	0	2
1	2	x	x	$1 = \frac{1}{k_1(1)}$	2	0
2	2	$x-1$	$x(x-1)$	$3 = \frac{1}{k_2(2)} \bmod 5$	2	2
3	1	$x-2$	$x(x-1)(x-2)$	$4 = \frac{1}{k_3(4)} \bmod 5$	$2 + 2x(x-1)$	4
4	2	$x-4$	…	—	$2 + 2x(x-1) +$ $4x(x-1)(x-2)$	—

(Při výpočtu např. b_3 počítáme $(u_4 - a_3) \cdot c_3 \bmod (x-4)$, neboli dosazujeme hodnotu $x = 4$ do polynomu $(2 - (2 + 2x(x-1))) \cdot 4$.) Prvním třem podmínkám tedy vyhovuje polynom $p = 2 + 2x(x-1) = 2x^2 + 3x + 2$. Všem čtyřem pak polynom $p = 2 + 2x(x-1) + 4x(x-1)(x-2) = 4x^3 + x + 2$.

6. Rychlá Fourierova transformace

V této sekci nechť $\mathbf{T}$ značí těleso a ω jeho prvek.

Diskrétní Fourierovou transformací $\mathrm{DFT}(\omega)$ s parametrem ω rozumíme výpočet hodnot polynomu $p \in \mathbf{T}[x]$ stupně nejvýše $n-1$ v bodech $1, \omega, \omega^2, \ldots, \omega^{n-1}$. Diskrétní Fourierovu transformaci lze chápat jako zobrazení mezi vektorovými prostory $\mathbf{T}^n \to \mathbf{T}^n$, které vektoru koeficientů přiřadí vektor hodnot ve zmíněných bodech. Tedy pro polynom $p = \sum_{i=0}^{n-1} a_i x^i$ platí

$$\mathrm{DFT}(\omega)(a_0, a_1, \ldots, a_{n-1}) = (p(1), p(\omega), \ldots, p(\omega^{n-1})).$$

Protože hodnotu polynomu v libovolném bodě α lze dostat násobením jistého řádkového vektoru závislého na α a stále stejného sloupcového vektoru podle vzorce

$$p(\alpha) = \sum_{i=0}^{n-1} a_i \alpha^i = \left(1, \alpha, \alpha^2 \ldots, \alpha^{n-1}\right) \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix},$$

pro diskrétní Fourierovu transformaci platí vzorec

$$\mathrm{DFT}(\omega)(u) = A(\omega) \cdot u,$$

kde u je sloupcový vektor $(a_0, a_1, \dots, a_{n-1})^T$ a $A(\omega)$ je Vandermondova matice

$$A(\omega) = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & & & & \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{pmatrix}$$

Tedy $\mathrm{DFT}(\omega)$ je lineární zobrazení. Navíc, pokud jsou prvky $1, \omega, \dots, \omega^{n-1}$ po dvou různé, pak je toto zobrazení bijektivní, protože, jak známo, determinant Vandermondovy matice $A(\omega)$ je roven součinu $\prod_{i>j}(\omega^i - \omega^j)$. Za tohoto předpokladu můžeme uvažovat inverzní zobrazení $\mathrm{DFT}(\omega)^{-1}$: nazýváme jej *inverzní diskrétní Fourierovou transformací* a budeme jej značit $\mathrm{IDFT}(\omega)$. Všimněte si, že

$$\mathrm{IDFT}(\omega)(u) = A(\omega)^{-1} \cdot u.$$

IDFT je vlastně *interpolací polynomu* z hodnot v bodech $1, \omega, \omega^2, \dots, \omega^{n-1}$ a celá diskrétní Fourierova transformace a její inverz jsou speciálním případem modulární reprezentace. Její smysl je v tom, že pro jistá ω existuje velmi rychlý algoritmus na jejich výpočet.

Definice. Řekneme, že prvek $\omega \in \mathbf{T}$ je *primitivní n-tá odmocnina z jedné* v tělese $\mathbf{T}$, jestliže platí

(1) $\omega^n = 1$,
(2) $\omega^i \neq 1$ pro všechna $i = 1, 2, \dots, n-1$.

Jinými slovy, jestliže je řád prvku ω v grupě $\mathbf{T}^*$ roven n.

Příklad. (1) V tělese $\mathbb{C}$ je $\omega = e^{\frac{2\pi i}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ primitivní n-tou odmocninou z jedné, jak lze snadno nahlédnout, když si nakreslíte její mocniny v komplexní rovině (tvoří vrcholy pravidelného n-úhelníka)

(2) V tělese $\mathbb{Z}_5$ je $\omega = 2$ primitivní čtvrtou odmocninou z jedné, neboť $2^2 = 4$, $2^3 = 3$ a $2^4 = 1$. Obecně, každý generátor grupy $\mathbb{Z}_p^*$ je primitivní $p-1$-tou odmocninou z jedné v tělese $\mathbb{Z}_p$.

Všimněte si, že je-li ω primitivní n-tá odmocnina z jedné, pak je matice $A(\omega)$ regulární.

Tvrzení 6.1. *Je-li ω primitivní n-tá odmocnina z jedné v tělese $\mathbf{T}$ a*

$$A(\omega) = \left(\omega^{ij}\right)_{i,j=0}^{n-1},$$

pak

$$A(\omega)^{-1} = \frac{1}{n} \cdot \left(\omega^{-ij}\right)_{i,j=0}^{n-1}.$$

Jinými slovy,

$$\mathrm{IDFT}(\omega) = \frac{1}{n} \cdot \mathrm{DFT}(\omega^{-1}).$$

Důkaz. Dokážeme, že součin těch dvou matic je jednotková matice (z toho plyne, že jsou navzájem inverzní). Podle vzorce pro součin matic platí

$$\left(\omega^{ij}\right)_{i,j=0}^{n-1} \cdot \frac{1}{n} \cdot \left(\omega^{-ij}\right)_{i,j=0}^{n-1} = \frac{1}{n} \cdot \Big(\sum_{k=0}^{n-1} \omega^{ik}\omega^{-kj}\Big)_{i,j=0}^{n-1}.$$

Přitom pro $i = j$ máme

$$\frac{1}{n} \cdot \sum_{k=0}^{n-1} \omega^{ik}\omega^{-ki} = \frac{1}{n} \cdot \sum_{k=0}^{n-1} 1 = \frac{1}{n} \cdot (n \cdot 1) = 1.$$

Naopak, pro $i \neq j$

$$\sum_{k=0}^{n-1} \omega^{ik}\omega^{-kj} = \sum_{k=0}^{n-1} \omega^{k(i-j)} = \sum_{k=0}^{n-1} (\omega^{i-j})^k,$$

dostali jsme tedy geometrickou řadu. Protože ω je *primitivní* odmocnina, máme $\omega^{i-j} \neq 1$ a můžeme použít známý vzorec, který říká, že

$$\sum_{k=0}^{n-1} (\omega^{i-j})^k = \frac{(\omega^{i-j})^n - 1}{\omega^{i-j} - 1}.$$

Zároveň však $\omega^n = 1$, a tedy $(\omega^{i-j})^n = (\omega^n)^{i-j} = 1^{i-j} = 1$. Čili výsledek je 0.

Dokázali jsme, že na diagonále součinu těch dvou matic jsou jednotky a mimo diagonálu nuly. Součinem je tedy jednotková matice. □

Ve zbytku kapitoly se tedy nemusíme zabývat výpočtem IDFT, neboť tu lze provést stejně jako DFT, pouze s jiným parametrem.

Principem rychlého algoritmu na výpočet DFT je metoda *rozděl a panuj*. Je-li n sudé, můžeme počítat hodnotu polynomu $p = \sum_{i=0}^{n-1} a_i x^i$ v bodě α rekurzivně takto:

$$p(\alpha) = \underbrace{(a_0 + a_2\alpha^2 + a_4\alpha^4 + \ldots + a_{n-2}\alpha^{n-2})}_{q(\alpha^2)} + \underbrace{(a_1\alpha + a_3\alpha^3 + \ldots + a_{n-1}\alpha^{n-1})}_{\alpha r(\alpha^2)},$$

tj.

$$p(\alpha) = q(\alpha^2) + \alpha r(\alpha^2),$$

přičemž q, r jsou polynomy definované

$$q(x) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i} x^i \quad \text{a} \quad r(x) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1} x^i.$$

Tedy úlohu dosazení hodnoty α do polynomu s n koeficienty jsme rozdělili na dvě úlohy dosazení hodnoty α^2 do polynomů poloviční velikosti.

Abychom mohli úlohu dělit na poloviční ve všech krocích, předpokládejmě nadále, že n je mocninou dvojky.

Algoritmus 6.2 (Rychlá Fourierova transformace). $\mathtt{FFT}(\omega)$

`VSTUP:` $a_0, a_1, \dots, a_{n-1}$.
`VÝSTUP:` $\mathrm{DFT}(\omega)(a_0, a_1, \dots, a_{n-1})$.
`0.` `IF` $n = 1$ `THEN RETURN` a_0, `STOP`.
`1.` $(b_0, \dots, b_{\frac{n}{2}-1}) := \mathtt{FFT}(\omega^2)(a_0, a_2, \dots, a_{n-2})$,
$(c_0, \dots, c_{\frac{n}{2}-1}) := \mathtt{FFT}(\omega^2)(a_1, a_3, \dots, a_{n-1})$.
`2.` Pro $i = 0, \dots, \frac{n}{2} - 1$ polož $d_i := b_i + \omega^i c_i$, $d_{i+\frac{n}{2}} := b_i - \omega^i c_i$,
`RETURN` $(d_0, \dots, d_{n-1})$.

Tvrzení 6.3. *Je-li n mocnina dvojky a ω primitivní n-tá odmocnina z jedné v tělese $\mathbf{T}$, pak Algoritmus 6.2 funguje.*

Důkaz. Důkaz provedeme indukcí podle n. Pro $n = 1$ je $\mathrm{DFT}(\omega)$ dosazení do konstantního polynomu s koeficientem a_0, tedy výsledek je a_0. Provedeme indukční krok. Nechť $p = \sum_{i=0}^n a_i x^i$ a definujeme polynomy q, r jako výše. Podle indukčního předpokladu

$$(b_0, \dots, b_{\frac{n}{2}-1}) = (q(1), q(\omega^2), q(\omega^4), \dots, q(\omega^{n-2}))$$

a

$$(c_0, \dots, c_{\frac{n}{2}-1}) = (r(1), r(\omega^2), r(\omega^4), \dots, r(\omega^{n-2})).$$

Chceme dokázat, že pro $i = 0, 1, \dots, \frac{n}{2} - 1$ platí

$$d_i = p(\omega^i) \quad \text{a} \quad d_{i+\frac{n}{2}} = p(\omega^{i+n/2}).$$

První vztah plyne přímo ze vzorce odvozeného výše:

$$p(\omega^i) = q(\omega^{2i}) + \omega^i r(\omega^{2i}) = b_i + \omega^i c_i = d_i.$$

Podobně odvodíme i druhý vztah:

$$p(\omega^{i+n/2}) = q(\omega^{2i+n}) + \omega^{i+n/2} r(\omega^{2i+n}) = b_i - \omega^i c_i = d_{i+\frac{n}{2}}.$$

Zde využíváme snadného pozorování, že $\omega^{2i+n} = \omega^{2i}\omega^n = \omega^{2i}$ a že $\omega^{i+n/2} = \omega^i \omega^{n/2} = -\omega^i$. Přitom $\omega^{n/2} = -1$, protože to je druhá odmocnina z jedné, a ty jsou pouze dvě: 1 (ta to není, neboť ω je *primitivní* odmocnina) a -1.

K funkčnosti algoritmu zbývá dokázat, že ω^2 je primitivní $\frac{n}{2}$-tá odmocnina z jedné. Zřejmě $(\omega^2)^{n/2} = \omega^n = 1$ a dále pro všechna $i = 1, 2, \dots, \frac{n}{2} - 1$ platí $(\omega^2)^i = \omega^{2i} \neq 1$, protože $2i < n$ a ω je primitivní odmocnina. □

Tvrzení 6.4. *Algoritmus 6.2 má časovou složitost $\mathcal{O}(n \log n)$ (jako jednotkovou operaci uvažujeme jakoukoliv operaci v tělese $\mathbf{T}$).*

Důkaz. Budeme postupovat podle již několikrát použitého schématu pro algoritmy rozděl a panuj. Předpokládejme, že $n = 2^k$. Označme $T(n)$ počet operací v tělese $\mathbf{T}$, které algoritmus provede na vstupu délky n. Všimněme si, že $T(1) = 0$ a

$$T(n) = 2T(\frac{n}{2}) + cn$$

pro jistou konstantu c. Platí tedy

$$\begin{aligned}
T(2^k) &= 2T(2^{k-1}) + c2^k \\
&= 2(2T(2^{k-2}) + c2^{k-1}) + c2^k = 4T(2^{k-2}) + c(2^k + 2^k) \\
&= \dots \\
&= 2^k T(2^{k-k}) + ck2^k = 2^k T(1) + ck2^k = \mathcal{O}(k2^k).
\end{aligned}$$

Tedy $T(n) = \mathcal{O}(n \log n)$. $\square$

Příklad. Uvažujme polynom $p = 5x^3 + x + 1 \in \mathbb{Z}_{41}[x]$. Můžeme zvolit $\omega = -9$, neboť $\omega^2 = -1$, $\omega^3 = 9$ a $\omega^4 = 1$. Spočteme $\mathrm{DFT}(\omega)(1,1,0,5)$ pomocí Rychlé Fourierovy transformace: $\mathtt{FFT}(\omega^2)(1,0) = (1,1)$, $\mathtt{FFT}(\omega^2)(1,5) = (6,-4)$, výsledek tedy je $(1+\omega^0 6, 1+\omega^1 \cdot (-4), 1-\omega^0 6, 1-\omega^1(-4)) = (1+6, 1+(-9)\cdot(-4), 1-6, 1-(-9)\cdot(-4)) = (7,-4,-5,6)$.

Zbývá vyřešit otázku, jak zvolit parametr ω, tj. kde vzít v tělese $\mathbf{T}$ primitivní n-tou odmocninu z jedné. Jak již bylo řečeno, v tělese $\mathbb{C}$ existuje primitivní n-tá odmocnina z jedné pro každé n, např.

$$\omega = e^{\frac{2\pi i}{n}} = \cos\frac{2\pi i}{n} + i \sin\frac{2\pi i}{n}.$$

Pro tělesa $\mathbb{Z}_p$ platí následující tvrzení:

Tvrzení 6.5. *V tělese $\mathbb{Z}_p$ existuje primitivní n-tá odmocnina z jedné právě tehdy, když $n \mid p-1$. V tom případě je primitivní n-tou odmocninou každý prvek $a^{\frac{p-1}{n}}$, kde a je generátor grupy $\mathbb{Z}_p^*$.*

Důkaz. Připomeňme, že primitivní n-tá odmocnina z jedné je vlastně prvek $\omega \in \mathbb{Z}_p^*$ řádu n. Víme, že $|\mathbb{Z}_p^*| = p-1$, a tedy, podle Lagrangeovy věty, jestliže $n \nmid p-1$, pak žádný prvek řádu n neexistuje. V opačném případě využijeme faktu, že grupa $\mathbb{Z}_p^*$ je cyklická, tedy existuje prvek $a \in \mathbb{Z}_p^*$, který tuto grupu generuje, a tudíž má řád $p-1$. Zřejmě $\omega = a^{\frac{p-1}{n}}$ je prvek řádu n, neboť $\omega^i = a^{i\frac{p-1}{n}} \neq 1$ pro každé $i < n$ a přitom $\omega^n = a^{p-1} = 1$ v $\mathbb{Z}_p$. $\square$

Jak ale takové a v $\mathbb{Z}_p$ najít? V základním kruzu jsme dokázali, že grupa $\mathbb{Z}_p^*$ obsahuje $\varphi(p-1)$ generátorů, kde φ značí Eulerovu funkci. Jejich hustota je tedy relativně velká a existuje odhad (dokazovaný obvykle v teorii čísel)

$$\frac{\varphi(p-1)}{p} \sim \frac{3}{\pi^2} \doteq 0,3.$$

Nejefektivnější metoda je tedy náhodná volba a následné ověření, zda skutečně řád náhodně zvoleného prvku je $p-1$. Zmíněný odhad říká, že se trefíme do generátoru v průměru v každém třetím případě.

Poznamenejme, že nás vlastně zajímají primitivní n-té odmocniny z jedné pro n rovno mocnině dvojky. Zajímavá jsou tělesa $\mathbb{Z}_p$, kde $2^k \mid p-1$ pro hodně velká k (např. 17, 41, atd.)

Závěrem okomentujeme nepříjemnou námitku, která vás již možná napadla: co když v mém oblíbeném tělese $\mathbf{T}$ (jako třeba v racionálních číslech) žádné primitivní n-té odmocniny z jedné nejsou? Pak v $\mathbf{T}$ nemůžeme provádět Rychlou Fourierovu transormaci. Nikdo nám ovšem nebrání si příslušnou odmocninu k $\mathbf{T}$ adjungovat a pracovat v algebraickém rozšíření $\mathbf{T}(\omega)$. Jako $\mathbf{T}(\omega)$ si samozřejmě zvolíme vhodné kořenové nadtěleso polynomu $x^n - 1$. V případě racionálních čísel je přirozenou volbou $\mathbb{Q}(e^{\frac{2\pi i}{n}})$.

7. Rychlé násobení polynomů

Principem rychlého algoritmu na násobení a dělení polynomů je následující pozorování: je-li

$$(c_0, \dots, c_n)$$

modulární reprezentace polynomu p a

$$(d_0, \dots, d_n)$$

modulární reprezentace polynomu q vzhledem k daným bodům $\alpha_0, \dots, \alpha_n$, a je-li navíc $n \geq \deg(p) + \deg(q)$ (!), pak modulární reprezentace polynomu $p \cdot q$ vzhledem k těmto bodům je

$$(c_0 d_0, \dots, c_n d_n),$$

neboť $(p \cdot q)(\alpha_i) = p(\alpha_i) \cdot q(\alpha_i)$ pro libovolný bod α_i. Analogicky, pokud $q \mid p$, pak $\frac{p}{q}$ má modulární reprezentaci

$$(\frac{c_0}{d_0}, \dots, \frac{c_n}{d_n}).$$

Přitom k výpočtu součinu a podílu v takové modulární reprezentaci stačí $n+1$ operací (násobení, resp. dělení) v tělese $\mathbf{T}$. Vzhledem k tomu, že uživatel obvykle vyžaduje vstup i výstup ve standardní reprezentaci, složitost násobení a dělení závisí na algoritmu pro převod do vhodně zvolené modulární báze.

Algoritmus 7.1 (Rychlé násobení).
`VSTUP:` $p = \sum_{i=0}^{n} a_i x^i, q = \sum_{i=0}^{m} b_i x^i \in \mathbf{T}[x]$.
`VÝSTUP:` $p \cdot q = \sum_{i=0}^{m+n} f_i x^i$.
`1.` Zvol $N = 2^k > m + n$ a nějakou primitivní n-tou odmocninu z jedné ω v $\mathbf{T}$.
`2.` $\bar{c} := \mathrm{FFT}(\omega)(a_0, \dots, a_n, 0, \dots, 0)$,
$\bar{d} := \mathrm{FFT}(\omega)(b_0, \dots, b_m, 0, \dots, 0)$.
`3.` $\bar{e} := \bar{c} \cdot \bar{d} = (c_0 \cdot d_0, \dots, c_{N-1} \cdot d_{N-1})$.
`4.` $\bar{f} := \frac{1}{N} \mathrm{FFT}(\omega^{-1})(e_0, \dots, e_{N-1})$.
`5.` `RETURN` $\sum_{i=0}^{N-1} f_i x^i$.

Tvrzení 7.2. *Předpokládejme, že ω, ω^2, ... jsou dány. Časová složitost Algoritmu 7.1 je $\mathcal{O}(n \log n)$, kde n je větší ze stupňů p, q.*

Důkaz. Rozeberme složitost jednotlivých kroků: `1.` je triviální. Krok `2.` má složitost $2\mathcal{O}(N \log N)$. Krok `3.` má složitost N. Krok `4.` má složitost $\mathcal{O}(N \log N)$. A krok `5.` je triviální. Protože $N \leq 4n$, máme celkoucou časovou složitost algoritmu $\mathcal{O}(N \log N) = \mathcal{O}(n \log n)$. □

Poznámka. Složitost hledání primitivní odmocniny z jedné jsme nepočítali, neboť zde velmi záleží na tělese $\mathbf{T}$. Např. v případě $\mathbb{Q}$ nemusíme nic hledat, stačí položit $\omega = e^{\frac{2\pi i}{N}}$ a $\omega^{-1} = e^{-\frac{2\pi i}{N}}$ a počítat v tělese $\mathbb{Q}(\omega)$. V případě $\mathbb{Z}_p$ můžeme hledat pravděpodobnostním algoritmem popsaným v předešlé sekci. Pokud v $\mathbb{Z}_p$ žádná primitivní N-tá odmocnina z jedné neexistuje, pracujeme opět v příslušném rozšíření.

Příklad. Spočteme součin $(3x^3 + x^2 - 4x + 1) \cdot (x^3 + 2x^2 + 5x - 3)$ v $\mathbb{Z}_{41}$.

(1) zvolme $N = 2^3 = 8 > 3 + 3$ a $\omega = 14$.
(2) $\bar{c} = \mathrm{FFT}(14)(1, -4, 1, 3, 0, 0, 0, 0) = (1, 9, -19, -18, 3, 16, 19, -3)$,
$\bar{d} = \mathrm{FFT}(14)(-3, 5, 2, 1, 0, 0, 0, 0) = (5, 5, 0, 14, -7, -6, -10, 16)$.
(3) $\bar{e} = (5, 4, 0, -6, 20, -14, 15, -7)$.
(4) Platí $\omega^{-1} = 3$ a $\frac{1}{N} = \frac{1}{8} = -5$. Tedy
$\bar{f} = -5 \cdot \mathrm{FFT}(3)(5, 4, 0, -6, 20, -14, 15, -7) = (-3, 17, 20, -11, 13, 7, 3, 0)$.
(5) Součin je $-3 + 17x + 20x^2 - 11x^3 + 13x^4 + 7x^5 + 3x^6$

Cvičení. Aplikujte uvedený algoritmus na výpočet součinu $(x^{10}+x+1)\cdot(x^3-x)$ v $\mathbb{Q}[x]$ a porovnejte výpočet s Karacubovým a klasickým násobením (viz cvičení v první kapitole).

Poznámka. Algoritmus na rychlé dělení funguje analogicky. V kroku `1.` stačí $N = 2^k > n$ a v kroku `3.` počítáme $\bar{e} = (\frac{c_0}{d_0}, \dots, \frac{c_{N-1}}{d_{N-1}})$. I v tomto případě bude časová složitost $\mathcal{O}(n \log n)$.

8. Rychlé dělení se zbytkem

Algoritmus dělení z předchozí sekce funguje pouze v tom případě, kdy je polynom p polynomem q dělitelný beze zbytku. To je problematické omezení. V této sekci vyvineme algoritmus pro obecné dělení se zbytkem. Ten má dvě velmi rozdílné části: nejprve uvedenou úlohu převedeme na problém invertování polynomu v oboru formálních mocninných řad a v druhé části nalezneme rychlý algoritmus na výpočet počátečních členů této řady.

8.1. **Algoritmus na dělení.** Zopakujme nejprve stručně, co to jsou *formální mocninné řady* nad okruhem $\mathbf{R}$: jsou to výrazy tvaru

$$\sum_{i=0}^{\infty} a_i x^i,$$

kde a_i jsou koeficienty z okruhu $\mathbf{R}$. Na těchto prvcích definujeme okruhové operace podobně jako pro polynomy:

$$\sum_{i=0}^{\infty} a_i x^i \pm \sum_{i=0}^{\infty} b_i x^i = \sum_{i=0}^{\infty} (a_i \pm b_i) x^i,$$
$$\sum_{i=0}^{\infty} a_i x^i \cdot \sum_{i=0}^{\infty} b_i x^i = \sum_{i=0}^{\infty} (\sum_{j=0}^{i} a_j \cdot b_{i-j}) x^i,$$
$$0 = 0+0+0+\dots, \quad 1 = 1+0+0+\dots.$$

Okruh formálních mocninných řad na okruhem $\mathbf{R}$ značíme $\mathbf{R}[[x]]$. Polynomy nad $\mathbf{R}$ tvoří jeho podokruh.

Pro účely této sekce zavedeme technické označení: pro polynom p definujeme

$$p^* = x^{\deg p} \cdot p(x^{-1}).$$

Jinými slovy, p^* je polynom, který vznikne z p, když napíšeme jeho koeficienty v opačném pořadí. Např. je-li $p = 3x^3 + 2x^2 - 1$, pak $p^* = x^3(3x^{-3} + 2x^{-2} - 1) = 3 + 2x - x^3$.

Nyní uvažujme polynomy $a, b \in \mathbf{T}[x]$, $b \neq 0$, a označme $n = \deg a$, $m = \deg b$, $n \geq m$. Chceme spočítat podíl a zbytek, tj. hledáme polynomy $q, r \in \mathbf{T}[x]$ splňující

$$a = bq + r, \quad \deg r < m.$$

Má tedy platit

$$a(x^{-1}) = b(x^{-1})q(x^{-1}) + r(x^{-1}).$$

Po vynásobení x^n dostáváme

$$\begin{aligned} x^n a(x^{-1}) &= x^m b(x^{-1}) x^{n-m} q(x^{-1}) + x^{n-\deg r} x^{\deg r} r(x^{-1}) \\ a^* &= b^* q^* + x^{n-\deg r} r^*. \end{aligned}$$

Po vydělení b^* dostáváme

$$q^* = \frac{a^*}{b^*} - x^{n-\deg r}\frac{r^*}{b^*}.$$

Polynom q^* má stupeň $n-m$, tedy je roven prvním $n-m+1$ členům mocniné řady na pravé straně rovnosti. Ale $n-m < n-\deg r$, takže $x^{n-\deg r}\frac{r^*}{b^*}$ se v prvních $n-m+1$ členech neprojeví. Polynom q^* je proto roven prvním $n-m+1$ členům mocninné řady $\frac{a^*}{b^*}$. Budeme-li umět najít těchto prvních $n-m+1$ členů, snadno dopočítáme q, r.

Algoritmus 8.1 (Rychlé dělení se zbytkem)**.**
`VSTUP:` $a, b \in \mathbf{T}[x]$, $b \neq 0$. Označme $n = \deg a$, $m = \deg b$, $n \geq m$.
`VÝSTUP:` a div b, a mod b.
`1.` Spočti a^*, b^*.
`2.` Spočti prvních $n-m+1$ členů mocninné řady $\frac{1}{b^*}$ a ulož je do c.
`3.` Spočti $q^* = a^*c$.
`4.` `RETURN` $q = a$ div $b = x^{n-m}q^*(\frac{1}{x})$ a a mod $b = a - bq$.

Správnost algoritmu jsme odvodili v předešlém textu.

Tvrzení 8.2. *Jestliže těleso* $\mathbf{T}$ *umožňuje FFT, pak má Algoritmus 8.1 časovou složitost* $\mathcal{O}(n\log n + m\log^2 m)$*. V opačném případě má složitost* $\mathcal{O}(n(n-m+1))$*.*

Důkaz. Rozebereme si složitost jednotlivých kroků:
(1) Přepsání koeficientů a a b v opačném pořadí má složitost $\mathcal{O}(n) + \mathcal{O}(m)$
(2) Ve zbytku kapitoly si ukážeme (Tvrzení 8.7), že s použitím FFT je složitost tohoto kroku $\mathcal{O}((n-m+1)\log(n-m+1) + m\log^2 m)$, bez použití $\mathcal{O}(n(n-m+1))$.
(3) $\mathcal{O}(n\log n)$, resp. $\mathcal{O}(n(n-m+1))$.
(4) Výpočet a div b je vlastně přepsání koeficientů q^* v opačném pořadí, má tedy složitost $\mathcal{O}(n-m)$. Dále, násobení bq má složitost $\mathcal{O}(n\log n)$, resp. $\mathcal{O}(m(n-m+1))$, odečítání je lineární.

Celková časová složitost algoritmu je tedy $\mathcal{O}(n\log n + m\log^2 m)$, resp. $\mathcal{O}(n(n-m+1))$. □

Poznamenejme, že nad tělesem, jež FFT neumožňuje, je tento způsob dělení pomalejší, než klasický. Tedy nezajímavý.

8.2. **Algoritmus na invertování.** V druhé části této sekce budeme řešit nově vzniklý problém invertování polynomu $p \in \mathbf{T}[x]$ v oboru formálních mocninných řad nad $\mathbf{T}$. Inverzem p v $\mathbf{T}[[x]]$ je mocninná řada $q \in \mathbf{T}[[x]]$ splňující $pq = 1$. Prvních n členů této mocninné řady je tedy polynom $\bar{q}$ stupně nejvýše $n-1$ splňující

$$p\bar{q} = \underbrace{1 + 0 + 0 + \ldots + 0}_{n} + x^n r$$

pro nějakou mocninnou řadu $r \in \mathbf{T}[[x]]$. Jinými slovy,

$$p\bar{q} \equiv 1 \pmod{x^n}.$$

Uvědomte si, že modx^n značí škrtnutí všech členů v n-té a vyšší mocnině.

Nejprve si ukážeme klasický postup výpočtu inverzu, a to obecněji, za předpokladu, že p je libovolná mocninná řada (ne nutně konečného stupně). Následující tvrzení říká, kdy takový inverz existuje.

Tvrzení 8.3. *Nechť* $\mathbf{T}$ *je těleso a* $p \in \mathbf{T}[[x]]$. *Pak* p *je invertibilní v* $\mathbf{T}[[x]]$ *právě tehdy, když* $p(0) \neq 0$.

Důkaz. ($\Rightarrow$) Kdyby $p(0) = 0$, pak $x \mid p$, a tedy $x \mid pq$ pro jakékoliv q. Speciálně, $pq \neq 1$. Tedy inverz neexistuje.

($\Leftarrow$) Označme $p = \sum_{i=0}^{\infty} a_i x^i$, platí tedy $a_0 \neq 0$. Hledáme $q = \sum_{i=0}^{\infty} b_i x^i$ splňující

$$pq = \sum_{i=0}^{\infty} (\sum_{j=0}^{i} a_j b_{i-j}) x^i = 1.$$

Koeficienty b_i tedy můžeme konstruovat indukcí:

(0) Musí platit $a_0 b_0 = 1$. Tedy $b_0 = a_0^{-1}$.
(1) Musí platit $a_0 b_1 + a_1 b_0 = 0$. Tedy $b_1 = -a_0^{-1}(a_1 b_0)$.
...
(i) Musí platit $\sum_{j=0}^{i} a_j b_{i-j} = 0$. Tedy $b_i = -a_0^{-1}(\sum_{j=1}^{i} a_j b_{i-j})$.

□

V případě, že p je polynom, důkaz obsahuje přímo algoritmus na jeho invertování. Prvních n členů mocninné řady p^{-1} získáme v n krocích tohoto algoritmu, přičemž nultý krok má časovou složitost 1 a i-tý krok ($i = 1, 2, \dots$) má časovou složitost $2i + 1$. Tedy celková složitost bude

$$1 + \sum_{i=1}^{n} (2i + 1)) = 1 + n + 2 \sum_{i=1}^{n} i = \mathcal{O}(n^2).$$

To je ale pomalé. My bychom potřebovali „něco jako $n \log n$". Efektivnímu algoritmu by tedy mělo stačit $\mathcal{O}(\log n)$ kroků, neboli v každém kroku je potřeba počet platných členů zdvojnásobit (nikoliv zvýšit o 1 jako v uvedeném výpočtu). Trik, kterým to lze provést, je založen na obecné *Newtonově metodě*, zmíněné v závěru této sekce.

Označme p polynom, který chceme invertovat, a q polynom sestávající z prvních n členů formální mocninné řady $\frac{1}{p}$. Algoritmus je založen na následujícím triku: protože $pq \equiv 1 \pmod{x^n}$, tedy jinými slovy $x^n \mid pq - 1$, platí také $x^{2n} \mid (pq-1)^2 = pq(pq-2) + 1$ a dostáváme

$$pq(2 - pq) \equiv 1 \pmod{x^{2n}}.$$

Vezmeme-li tedy polynom $q(2-pq)$, dostáváme prvních $2n$ členů formální mocninné řady $\frac{1}{p}$.

Algoritmus 8.4 (Newtonův)**.**
`VSTUP:` $p \in \mathbf{T}[x]$, $\deg p = m$, $p(0) \neq 0$, n.
`VÝSTUP:` Prvních n členů mocninné řady $\frac{1}{p}$.
`0.` $q_0 := a_0^{-1}$.
i. $q_i := q_{i-1}(2 - pq_{i-1}) \bmod x^{2^i}$ pro $i = 1, \dots, \lceil \log n \rceil$.

Tvrzení 8.5. *Polynom* q_i *sestává z prvních* 2^i *členů formální mocninné řady* $\frac{1}{p}$. *Tedy Algoritmus 8.4 funguje.*

Důkaz. Dokážeme indukcí podle i. Pro $i = 0$ to evidentně platí (viz důkaz Tvrzení 8.3). A platí-li $pq_{i-1} \equiv 1 \bmod x^{2^{i-1}}$, tj. $pq_{i-1} = 1 + x^{2^{i-1}} r$ pro nějaké $r \in \mathbf{T}[[x]]$, pak

$2-pq_{i-1} = 1-x^{2^{i-1}}r$. Dostáváme $pq_i = pq_{i-1}(2-pq_{i-1}) = (1+x^{2^{i-1}}r)(1-x^{2^{i-1}}r) = 1 - x^{2^i}r^2 \equiv 1 \pmod{x^{2^i}}$. □

K důkazu složitosti Algoritmu 8.4 budeme potřebovat součet následující řady.

Lemma 8.6. $\sum_{i=1}^{N} i2^i = (N+1)2^{N+1} - 2^{N+2} + 2$.

Důkaz. Dokážeme indukcí podle N. Pro $N = 1$ to evidentně platí. A z platnosti vzorce pro N dostáváme

$$\begin{aligned}\sum_{i=1}^{N+1} i2^i = \sum_{i=1}^{N} i2^i + (N+1)2^{N+1} &= (N+1)2^{N+1} - 2^{N+2} + 2 + (N+1)2^{N+1} \\ &= (N+1)2^{N+2} - 2^{N+2} + 2 = (N+2)2^{N+2} - 2^{N+2} - 2^{N+2} + 2 \\ &= (N+2)2^{N+2} - 2^{N+3} + 2.\end{aligned}$$

□

Tvrzení 8.7. *Jestliže těleso* $\mathbf{T}$ *umožňuje FFT, pak má Algoritmus 8.4 časovou složitost* $\mathcal{O}(n\log n + m\log^2 m)$. *V opačném případě* $\mathcal{O}(n^2 + mn)$.

Důkaz. Bez újmy na obecnosti předpokládejme, že $n = 2^k$. Tedy algoritmus proběhne v $k+1$ krocích. Složitost 0. kroku je 1. V dalších krocích je zřejmě nejnáročnější operací násobení polynomů q_{i-1} a $2 - pq_{i-1}$. Přitom $\deg p = m$ a $\deg q_{i-1} < 2^{i-1}$, tedy jde o násobení polynomů stupňů nejvýše 2^{i-1} a $m + 2^{i-1}$. Při použití klasického násobení dostáváme celkovou časovou složitost

$$\begin{aligned}\mathcal{O}\left(1 + \sum_{i=1}^{k} \mathcal{O}(2^{i-1}\cdot(2^{i-1}+m))\right) &= \mathcal{O}\left(\sum_{i=0}^{k-1} 4^i + m\sum_{i=0}^{k-1} 2^i\right) = \mathcal{O}\left(\frac{4^k-1}{3} + m(2^k-1)\right) \\ &= \mathcal{O}((2^k)^2 + m2^k) = \mathcal{O}(n^2 + mn).\end{aligned}$$

V případě použití algoritmu pro rychlé násobení 7.1 dostáváme časovou složitost

$$\mathcal{O}\left(1 + \sum_{i=1}^{k} \mathcal{O}\left((2^{i-1}+m)\cdot\log(2^{i-1}+m)\right)\right).$$

Tuto sumu si rozdělíme na dvě části: dokud $2^{i-1} < m$, použijeme odhad $2^{i-1} + m < 2m$. V opačném případě použijeme odhad $2^{i-1} + m < 2\cdot 2^{i-1} = 2^i$. Dostáváme tedy

$$\begin{aligned}\sum_{i=1}^{k}(2^{i-1}+m)\cdot\log(2^{i-1}+m) &= \sum_{i<1+\log_2 m} 2m\log(2m) + \sum_{i\geq 1+\log_2 m} 2^i \log 2^i \\ &\leq \log_2 m \cdot 2m\log(2m) + \sum_{i=0}^{k} i2^i \\ &= \mathcal{O}(m\log^2 m) + \mathcal{O}(k2^k) = \mathcal{O}(m\log^2 m + n\log n).\end{aligned}$$

□

Příklad. Spočtěte první 4 členy mocninné řady $\frac{1}{p}$ pro $p = 1 - 2x + 3x^2 + x^4 - x^5 \in \mathbb{Q}[x]$. (Vidíme, že $p(0) \neq 0$.)
Použitím Newtonova algoritmu dostáváme:

$$\begin{aligned}&q_0 = 1 \\ &q_1 = 1\cdot(2 - p\cdot 1) \bmod x^2 = 2 - 1 + 2x + \ldots \bmod x^2 = 1 + 2x\end{aligned}$$

$q_2 = (1+2x)(2-p\cdot(1+2x)) \bmod x^4 = 2-p-2xp+4x-2xp-4x^2p = 1+2x+x^2-4x^3$

Požitím klasického algoritmu dostáváme:

$b_0 = a_0^{-1} = 1$
$b_1 = -a_0^{-1}(a_1b_0) = -(-2\cdot 1) = 2$
$b_2 = -a_0^{-1}(a_1b_1 + a_2b_0) = -((-2\cdot 2) + 3\cdot 1) = 1$
$b_3 = -a_0^{-1}(a_1b_2 + a_2b_1 + a_3b_0) = -((-2\cdot 1) + 3\cdot 2 + 0\cdot 1) = -4$

8.3. **Obecná Newtonova metoda.** Nejprve si všimněte, že jednoduchou analogii Newtonova algoritmu můžeme použít i na přibližný výpočet zlomků. Tato metoda se dříve skutečně používala v počítačových systémech pro numerickou matematiku. Dnes už je aritmetika v plovoucí čárce implementována na hardwarové úrovni.

Algoritmus 8.8 (Aproximace zlomků)**.**
`VSTUP:` $a, n \in \mathbb{N}$.
`VÝSTUP:` $\frac{1}{a}$ na n desetinných míst.
`0.` Zvol q_0 nějaký kladný dolní odhad $\frac{1}{a}$.
i. $q_i := q_{i-1}(2 - aq_{i-1})$.

Dokážeme, že q_{i+1} *obsahuje přibližně dvakrát více platných cifer než* q_i. Toto tvrzení je poněkud vágní, ale jeho přesná formulace není v tomto okamžiku našim cílem. Cílem je si uvědomit, že tato metoda aproximace zlomků pracuje *velmi rychle.*

Předpokládejme, že $\frac{1}{a} = q_i + 10^{-k}r$, kde $r \in (0,1)$. Postupnými úpravami dostáváme

$$q_i = \frac{1}{a} - 10^{-k}r,$$
$$aq_i = 1 - 10^{-k}ar,$$
$$2 - aq_i = 1 + 10^{-k}ar.$$

Vynásobením předchozích dvou řádků dostáváme aq_{i+1}, tedy platí

$$aq_{i+1} = aq_i\cdot(2 - aq_i) = 1 - 10^{-2k}a^2r^2,$$

a tedy

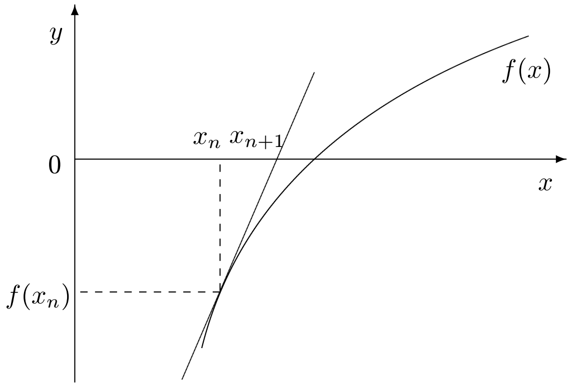
$$\frac{1}{a} = q_{i+1} + 10^{-2k}ar^2.$$

Pokud je $ar^2 \in (0,1)$, pak q_{i+1} má přesně dvakrát více platných cifer než q_i. V opačném případě jich má přiměřeně méně, ovšem uvědomte si, že s přibývajícím i se případný přesah marginalizuje.

Příklad. Spočtěte $\frac{1}{7}$.

	počet platných cifer
$q_0 = 0,1$	1
$q_1 = 0,13$	1
$q_2 = 0,1417$	2
$q_3 = 0,14284777$	4
$q_4 = 0,1428571432\ldots$	8

Na závěr si ve zkratce ukážeme *obecnou Newtonovu metodu*, která slouží k přibližnému výpočtu kořene rovnice $f(x) = 0$ pro libovolnou diferencovatelnou funkci f na reálných číslech. Nechť x_0 je nějaká aproximace tohoto kořene. Budeme konstruovat posloupnost $x_1, x_2, \ldots$ postupně zpřesňující tento odhad.



Všimněte si, že platí $f'(x_n) = \frac{-f(x_n)}{x_{n+1}-x_n}$ (protilehlá ku přilehlé), a tedy novou (lepší) aproximaci dostaneme z předchozí volbou

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Dá se dokázat, že pro *hezké funkce* f a *při vhodné volbě* x_0 konverguje posloupnost $x_0, x_1, x_2, \dots$ k nějakému kořeni, a to kvadratickou rychlostí (tj. existuje konstanta C taková, že pro všechna n platí $|x_{n+1} - x| \leq C \cdot |x_n - x|^2$). Co to znamená *hezká funkce* a *vhodná volba* x_0 necháváme na starosti numerické matematice.

Všimněte si, že pro $f = a - \frac{1}{x}$ dostáváme přesně vzorec z Newtonova algoritmu na aproximaci zlomků: platí $f' = \frac{1}{x^2}$, a tedy $x_{n+1} = x_n - (ax_n^2 - x_n) = x_n(2 - ax_n)$.

Největší společný dělitel

9. Primitivní polynomy a Gaussova věta

V této kapitole se seznámíme s dvěma základními algoritmy na výpočet největšího společného dělitele polynomů. Tato sekce pak slouží jako teoretická příprava: nejprve dokážeme, že NSD polynomů skutečně existují (připomeňme, že v základním kurzu byl tento fakt pouze konstatován).

Cílem je tedy dokázat tzv. *Gaussovu větu*, která říká, že obor polynomů (libovolně mnoha proměnných) nad Gaussovým oborem je opět Gaussův obor. Tedy, speciálně, že v oboru polynomů nad okruhem $\mathbf{R}$ existují NSD právě tehdy, když existují v $\mathbf{R}$. Poznamenejme, že pro obor polynomů jedné proměnné nad tělesem jsme již tento fakt dokázali dříve: takový obor je dokonce Eukleidovský. Obecné obory polynomů však tuto vlastnost nemají (např. $\mathbb{Z}[x]$ ani $\mathbb{Q}[x, y]$ Eukleidovské nejsou). Připomeňme, že Gaussovskost oboru $\mathbf{R}$ lze definovat ekvivalentně dvěma způsoby:

- v $\mathbf{R}$ lze každý prvek rozložit *jednoznačným* způsobem na součin ireducibilních prvků (připomeňme, že ireducibilní prvek je neinvertibilní prvek jež se nedá netriviálně rozložit);
- v $\mathbf{R}$ existují největší společní dělitelé libovolných dvou prvků a přitom neexistuje nekonečná posloupnost vlastních dělitelů.

Uvědomte si, že důkaz druhé části podmínky (2) je pro libovolný okruh polynomů nad Gaussovým oborem takřka očividný.

Lemma 9.1. *Nechť $\mathbf{R}$ je Gaussův obor. Pak v oboru $\mathbf{R}[x]$ neexistuje nekonečná posloupnost vlastních dělitelů.*

Důkaz. Nechť $p_1, p_2, p_3, \dots$ je taková posloupnost, tj. $p_{i+1} \mid p_i$ a $p_i \nmid p_{i+1}$. Pak zřejmě $\deg p_1 \geq \deg p_2 \geq \deg p_3 \geq \dots \geq 0$, a tedy existuje N takové, že $\deg p_N = \deg p_{N+1} = \dots$ Označme $a_N = \mathrm{lc}(p_N)$, $a_{N+1} = \mathrm{lc}(p_{N+1})$. Pak $a_N, a_{N+1}, \dots$ je nekonečná posloupnost vlastních dělitelů v $\mathbf{R}$, spor. □

Ve zbytku této sekce dokážeme pro okruhy polynomů první část podmnínky (2). Nebude-li řečeno jinak, $\mathbf{R}$ bude značit nějaký Gaussův obor.

Definice. Nechť $p = \sum_{i=0}^{n} a_i x^i \in \mathbf{R}[x]$, $a_n \neq 0$.

- Polynom p nazýváme *primitivní*, jestliže $\mathrm{NSD}(a_0, \dots, a_n) = 1$.
- *Obsahem* polynomu p rozumíme $\mathrm{cont}(p) = \mathrm{NSD}(a_0, \dots, a_n)$.
- *Primitivní částí* polynomu p rozumíme $\mathrm{pp}(p) = \sum_{i=0}^{n} \frac{a_i}{\mathrm{cont}(p)} x^i$.

Všimněte si, že v okruhu polynomů nad tělesem je každý polynom primitivní. V okruhu $\mathbb{Z}[x]$ máme např. pro polynom $p = 3x^2 + 6x - 3$ obsah $\mathrm{cont}(p) = 3$ a primitivní část $\mathrm{pp}(p) = x^2 + 2x - 1$.

Následující snadné tvrzení ukazuje, že při hledání největšího společného dělitele polynomů se můžeme zaměřit pouze na primitivní polynomy.

Tvrzení 9.2. *Nechť* $\mathbf{R}$ *je obor integrity a* $p, q \in \mathbf{R}[x]$. *Pak*

$$\mathrm{NSD}_{\mathbf{R}[x]}(p,q) = \mathrm{NSD}_{\mathbf{R}}(\mathrm{cont}(p), \mathrm{cont}(q)) \cdot \mathrm{NSD}_{\mathbf{R}[x]}(\mathrm{pp}(p), \mathrm{pp}(q))$$

za předpokladu, že oba největší společní dělitelé na pravé straně rovnosti existují.

Důkaz. Označme r výraz na pravé straně. Je třeba dokázat, že r je společným dělitelem p a q (v $\mathbf{R}[x]$) a dokázat, že r je nejmenší s touto vlastností (tedy že, pro jakékoliv $a \in \mathbf{R}[x]$, pokud $a \mid p, q$, pak $a \mid r$).

Protože $\mathrm{NSD}_{\mathbf{R}}(\mathrm{cont}(p), \mathrm{cont}(q)) \mid \mathrm{cont}(p)$ a $\mathrm{NSD}_{\mathbf{R}[x]}(\mathrm{pp}(p), \mathrm{pp}(q)) \mid \mathrm{pp}(p)$, platí $r \mid p$. Podobně $r \mid q$.

Pokud $a \mid p, q$, pak $\mathrm{cont}(a) \mid \mathrm{cont}(p), \mathrm{cont}(q)$, tedy $\mathrm{cont}(a) \mid \mathrm{NSD}_{\mathbf{R}}(\mathrm{cont}(p), \mathrm{cont}(q))$. Podobně $\mathrm{pp}(a) \mid \mathrm{NSD}_{\mathbf{R}[x]}(\mathrm{pp}(p), \mathrm{pp}(q))$. Dohromady $a|r$. □

Následující tvrzení je klíčové:

Tvrzení 9.3 (Gaussovo lemma)**.** *Nechť* $\mathbf{R}$ *je Gaussův obor a* $p, q \in \mathbf{R}[x]$. *Pak* $p \cdot q$ *jsou primitivní právě tehdy, když oba polynomy* p, q *jsou primitivní.*

Důkaz. Implikace zleva doprava je triviální, protože zřejmě $\mathrm{cont}(p), \mathrm{cont}(q) \mid \mathrm{cont}(pq)$.

Nyní předpokládáme, že p a q jsou primitivní a chceme dokázat, že pq je také primitivní polynom. Označme $p = \sum_{i=0}^{n} a_i x^i$ a $q = \sum_{i=0}^{m} b_i x^i$. Dodefinujme $a_{n+1} = \ldots = a_{n+m} = 0$ a $b_{m+1} = \ldots = b_{m+n} = 0$. Máme tedy $p \cdot q = \sum_{i=0}^{m+n} (\sum_{j=0}^{i} a_j b_{i-j}) x^i$, označme i-tý koeficient c_i. Protože R je Gaussův obor, stačí dokázat, že neexistuje ireducibilní prvek $u \in \mathbf{R}$, který by dělil všechna c_i (ta pak musí být nutně nesoudělná). Pro spor předpokládejme opak.

Protože jsou polynomy p, q primitivní, u nemůže dělit všechna a_i ani všechna b_i. Zvolme nejmenší j takové, že $u \nmid a_j$ a nejmenší k takové, že $u \nmid b_k$. Dokážeme, že $u \nmid c_{j+k}$, což dá spor. Platí

$$c_{j+k} = a_0 b_{j+k} + a_1 b_{j+k-1} + \ldots + a_{j-1} b_{k+1} + a_j b_k + a_{j+1} b_{k-1} + \ldots + a_{j+k} b_0.$$

Protože $u \mid a_l$ pro všechna $l < j$, máme

$$u \mid a_0 b_{j+k} + a_1 b_{j+k-1} + \ldots + a_{j-1} b_{k+1}.$$

Protože $u \mid b_l$ pro všechna $l < k$, máme

$$u \mid a_{j+1} b_{k-1} + \ldots + a_{j+k} b_0.$$

Ovšem $u \nmid a_j b_k$, a tedy $u \nmid c_{j+k}$, spor. □

Tvrzení 9.4. *Nechť* R *je Gaussův obor,* $\mathbf{Q}$ *jeho podílové těleso a* $p, q \in \mathbf{R}[x]$. *Pokud* $p \mid q$ *v* $\mathbf{Q}[x]$, *pak* $p \mid q$ *v* $\mathbf{R}[x]$.

Důkaz. Předpokládejme, že $p \mid q$ v $\mathbf{Q}[x]$, čili existuje $r \in \mathbf{Q}[x]$ takový, že $pr = q$. Vezmeme $k \in \mathbf{Q}$ tak, aby rk byl primitivní polynom v $\mathbf{R}[x]$. Máme $p \cdot rk = qr$. Na levé straně je součin primitivních polynomů, takže podle Gaussova lemmatu qr je primitivní polynom. Protože q a qr jsou primitivní, je $r \in \mathbf{R}$, $r \parallel 1$. Takže ve skutečnosti $r = rk \cdot k^{-1}$ je polynom v $\mathbf{R}[x]$ a jsme hotovi. □

Důsledkem předchozího tvrzení je, že největší společný dělitel primitivních polynomů v $\mathbf{R}[x]$ lze počítat $\mathbf{Q}[x]$:

Tvrzení 9.5. *Nechť* $\mathbf{R}$ *je Gaussův obor,* $\mathbf{Q}$ *jeho podílové těleso,* $p, q \in \mathbf{R}[x]$ *primitivní polynomy a* $r = \mathrm{NSD}_{\mathbf{Q}[x]}(p,q)$. *Pak* $\mathrm{NSD}_{\mathbf{R}[x]}(p,q)$ *existuje a je roven* kr, *kde* k *je takový prvek* $\mathbf{Q}$, *že* kr *je primitivní polynom.*

Důkaz. Dokážeme, že $a \mid p$ a $a \mid q$ v $\mathbf{R}[x]$ právě tehdy, když $a \mid kr$ v $\mathbf{R}[x]$.

($\Rightarrow$) Protože $a \mid p$, $a \mid q$ a $kr = \mathrm{NSD}_{\mathbf{Q}[x]}(p, q)$, platí $a \mid kr$ (v $\mathbf{Q}[x]$). Podle Tvrzení 9.4 také $q|kr$ v $\mathbf{R}[x]$.

($\Leftarrow$) Předpokládejme, že $a \mid kr = \mathrm{NSD}_{\mathbf{Q}[x]}(p, q)$ v $\mathbf{R}[x]$. Pak $a \mid p, q$ v $\mathbf{Q}[x]$ a proto (opět podle Tvrzení 9.4) $a \mid p, q$ v $\mathbb{Q}[x]$. □

Příklad. $\mathrm{NSD}_{\mathbb{Z}[x]}(4x^2 + 8x + 4, -6x^2 + 6) = \mathrm{NSD}_{\mathbb{Z}}(4, 6) \cdot \mathrm{NSD}_{\mathbb{Q}[x]}(x^2 + 2x + 1, x^2 - 1) = 2(x + 1)$.

Užitím Tvrzení 9.2 a Tvrzení 9.5 dostáváme:

Důsledek 9.6. *Nechť* $\mathbf{R}$ *je Gaussův obor,* $\mathbf{Q}$ *jeho podílové těleso a* $p, q \in \mathbf{R}[x]$ *libovolné polynomy. Pak* $\mathrm{NSD}_{\mathbf{R}[x]}(p, q)$ *existuje a platí* $\mathrm{NSD}_{\mathbf{R}[x]}(p, q) = \mathrm{NSD}_{\mathbf{Q}[x]}(p, q)$ *v* $\mathbf{Q}[x]$.

Věta 9.7 (Gaussova)**.** *Nechť* $\mathbf{R}$ *je Gaussův obor a* X *libovolná neprázdná množina. Pak je* $\mathbf{R}[X]$ *také Gaussův obor.*

Důkaz. (A) Nejprve předpokládejme, že X je konečná. Důkaz provedeme indukcí podle počtu prvků množiny X.

(1) Je-li $|X| = 1$, je tvrzení obsaženo v Důsledku 9.6 a Tvrzení 9.1.
(2) Je-li $|X| = n$, označme $X = \{x_1, \ldots, x_n\}$. Podle indukčního předpokladu je $\mathbf{R}[x_1, \ldots, x_{n-1}]$ Gaussův obor. Tedy $\mathbf{R}[x_1, \ldots, x_n] \simeq (\mathbf{R}[x_1, \ldots, x_{n-1}])[x_n]$ je také Gaussův, podle bodu (1).

(B) Nyní předpokládejme, že X je nekonečná. Nechť $p, q \in \mathbf{R}[X]$. Protože p, q mají jen konečné množství členů, existují $Y, Z \subset X$ konečné takové, že $p \in \mathbf{R}[Y]$ a $q \in \mathbf{R}[Z]$. Evidentně $\mathrm{NSD}_{\mathbf{R}[X]}(p, q) = \mathrm{NSD}_{\mathbf{R}[Y \cup Z]}(p, q)$ a ten existuje podle bodu (A). Neexistenci nekonečné posloupnosti vlastních dělitelů můžeme dokázat také odkazem na konečný případ: první člen takové posloupnosti by obsahoval pouze konečně mnoho proměnných; přitom všichni jeho vlastní dělitelé obsahují pouze podmnožinu těchto proměnných. Šlo by tedy o posloupnost v nějakém $\mathbf{R}[Y]$, Y konečné, a ta neexistuje podle bodu (A). □

Tvrzení 9.2 a 9.5 nám zároveň dávají algoritmus, podle kterého lze počítat $\mathrm{NSD}(p, q)$ v Gaussově oboru $\mathbf{R}[x]$ za předpokladu, že máme algoritmus pro výpočet NSD v $\mathbf{R}$ (připomeňme, že Eukleidův algoritmus obecně funguje pouze nad tělesem, pro výpočet v $\mathbf{Q}[x]$ jej tedy použít lze). Algoritmus probíhá takto:

(1) spočteme obsahy a primitivní části,
(2) provedeme NSD obsahů v $\mathbf{R}$,
(3) provedeme NSD primitivních částí v $\mathbf{Q}[x]$ a vynásobíme jej nejmenším společným násobkem jmenovatelů koeficientů,
(4) výsledky vynásobíme.

Typickým příkladem je obor $\mathbb{Z}[x]$: kroky (2) a (3) lze provést pomocí Eukleidova algoritmu v $\mathbb{Z}$, resp. v $\mathbb{Q}[x]$. Podobně v oboru $\mathbb{Z}[x_1, \ldots, x_n]$: krok (2), tedy výpočet NSD v oboru $\mathbb{Z}[x_1, \ldots, x_{n-1}]$, lze provést iterovaným použitím téhož algoritmu.

Nevýhodou tohoto algoritmu je fakt, že místo v $\mathbf{R}$ se většina operací odehrává v jeho podílovém tělese $\mathbf{Q}$. Přitom operace v podílovém tělese jsou časově *mnohem* náročnější, např. kvůli neustálému krácení na základní tvar. Tento problém je nejmarkantnější právě na příkladě polynomů více proměnných, kde bychom museli pracovat s oborem racionálních funkcí. Ve zbytku této kapitoly předvedeme algoritmy efektivnější:

- ve druhé sekci se pokusíme o úpravu Eukleidova algoritmu tak, aby nepoužíval zlomky;
- ve třetí sekci si ukážeme algoritmus založený na modulární metodě.

Na závěr dokážeme ještě jeden teoretický důsledek Gaussova lemmatu týkající se ireducibility polynomů

Jak plyne z Tvrzení 9.2 a 9.5, *největší společní dělitelé* dvou polynomů počítané nad Gaussovým oborem $\mathbf{R}$, resp. nad jeho podílovým tělesem $\mathbf{Q}$, spolu úzce souvisejí, podle vzorce

$$\mathrm{NSD}_{\mathbf{R}[x]}(p,q) = \mathrm{NSD}_{\mathbf{R}}(\mathrm{cont}(p), \mathrm{cont}(q)) \cdot \mathrm{NSD}_{\mathbf{Q}[x]}(\mathrm{pp}(p), \mathrm{pp}(q)),$$

přičemž $\mathrm{NSD}_{\mathbf{Q}[x]}$ bereme takový, aby byl výsledný polynom primitivní. Nyní dokážeme, jak spolu souvisí *ireducibilita* polynomů nad Gaussovým oborem a nad jeho podílovým tělesem.

Lemma 9.8. *Nechť* $\mathbf{R}$ *je Gaussův obor,* $\mathbf{Q}$ *jeho podílové těleso a* $p \in \mathbf{R}[x]$ *primitivní polynom. Pak* p *je ireducibilní v* $\mathbf{R}[x]$ *právě tehdy, když* p *je ireducibilní v* $\mathbf{Q}[x]$.

Důkaz. Dokážeme, že p lze netriviálně rozložit v $\mathbf{R}[x]$ právě tehdy, když p lze netriviálně rozložit v $\mathbf{Q}[x]$.

($\Rightarrow$) Nechť $p = ab$, $a, b \in \mathbf{R}[x]$, $a, b \nparallel 1$ v $\mathbf{R}[x]$. Je otázka, zda $a, b \nparallel 1$ v $\mathbf{Q}[x]$. Kdyby aspoň jeden z a, b byl invertibilní, musel by být stupně 0 a tedy by p nebyl primitivní.

($\Leftarrow$) Nechť $p = ab$, $a, b \in \mathbf{Q}[x]$, $a, b \nparallel 1$ v $\mathbf{Q}[x]$. Vezměme $k, l \in \mathbf{Q}$ takové, že ka a lb jsou primitivní polynomy v $\mathbf{R}[x]$. Pak $kl \cdot p = ka \cdot lb$ a protože ka i lb jsou primitivní, je podle Gaussova lemmatu i jejich součin primitivní. Tedy $kl \parallel 1$ a $p = (kl)^{-1} ka \cdot lb$ je netriviální rozklad p v $\mathbf{R}[x]$. □

Tvrzení 9.9. *Nechť* $\mathbf{R}$ *je Gaussův obor,* $\mathbf{Q}$ *jeho podílové těleso a* $p \in \mathbf{R}[x]$. *Pak* p *je ireducibilní v* $\mathbf{R}[x]$ *právě tehdy, když*

- *buď* $\deg p = 0$ *a* p *je ireducibilní v* $\mathbf{R}$*;*
- *nebo* $\deg p > 0$*,* p *je primitivní a* p *je ireducibilní v* $\mathbf{Q}[x]$.

Důkaz. Vzhledem k tomu, že se p rozkládá na obsah a primitivní část, musí být v případě ireducibility jedna z těchto částí triviální a druhá ireducibilní. Zbytek plyne z předchozího lemmatu. □

Příklad.
- Polynom $2x-2$ je ireducibilní v $\mathbb{Q}[x]$, ale není ireducibilní v $\mathbb{Z}[x]$, protože není primitivní.
- Polynom 2 není ireducibilní v $\mathbb{Q}[x]$, protože je invertibilní, ale je ireducibilní v $\mathbb{Z}[x]$.
- Polynom $x^4 - 2x^2 + 6x + 2$ je ireducibilní $\mathbb{Q}[x]$ (jak plyne z Eisensteinova kritéria) i v $\mathbb{Z}[x]$ (protože je navíc primitivní).

10. Pseudodělení

Jak jsme již zmínili, v neeukleidovských oborech polynomů obecně neexistuje nic jako podíl a zbytek. Abychom však mohli adaptovat Eukleidův algoritmus na tyto obory, něco jako dělení bychom potřebovali.

Příklad.
- V $\mathbb{Z}[x]$ neexistuje podíl polynomů $x^2 - 1$ a $2x - 2$. Při řešení rovnosti $(x^2 - 1) = q(2x - 2) + r$ (kde $\deg r < \deg(2x - 2) = 1$) docházíme k tomu, že koeficient u x je sudý, spor.
- V $\mathbf{R}[x, y] = (\mathbf{R}[x])[y]$ neexistuje podíl polynomů y^2 a $xy + 1$. Při řešení rovnosti $y^2 = q(xy + 1) + r$ (kde $\deg r < 1$ ve smyslu polynomu jedné proměnné y) docházíme k tomu, že koeficient u y^2 je dělitelný x, spor.

Co nám překáží, je „příliš malý” vedoucí koeficient dělence. Všimněte si, že $2(x^2-1) = (x+1)(2x-2)+0$, tj. podíl a zbytek po dělení polynomů $2(x^2+1)$ a $2x-2$ existuje, a také $x^2y^2 = (xy - 1)(xy + 1) + 1$, tj. podíl a zbytek po dělení polynomů x^2y^2 a $xy - 1$ existuje. Otázku, jak velkým koeficientem je třeba přenásobit, řeší následující věta.

Tvrzení 10.1 (o pseudodělení)**.** *Nechť $\mathbf{R}$ je obor integrity, $a, b \in \mathbf{R}[x]$, $b \neq 0$, $n = \deg a$, $m = \deg b$ a předpokládejme $m \leq n$. Pak existuje právě jedna dvojice $q, r \in \mathbf{R}[x]$ taková, že*

$$\mathrm{lc}(b)^{n-m+1}a = qb + r \quad \text{a} \quad \deg r < m.$$

Polynom q nazýváme *pseudopodíl* a značíme a pdiv b. Polynom r nazýváme *pseudozbytek* a značíme a pmod b. Jinými slovy,

$$a \text{ pdiv } b = (\mathrm{lc}(b)^{n-m+1}a) \text{ div } b,$$
$$a \text{ pmod } b = (\mathrm{lc}(b)^{n-m+1}a) \text{ mod } b.$$

Důkaz. Nejprve dokážeme jednoznačnost. Předpokládejme, že $\mathrm{lc}(b)^{n-m+1}a = q_1b + r_1 = q_2b + r_2$, přičemž $\deg r_1, \deg r_2 < m$. Pak $q_1b - q_2b = b(q_1 - q_2) = r_2 - r_1$, tedy $b \mid r_2 - r_1$. Ovšem $\deg(r_2 - r_1) < \deg b$, a tedy musí být $r_2 - r_1 = 0$. Ovšem pak $b(q_1 - q_2) = 0$, a protože $b \neq 0$ a $\mathbf{R}$ je obor integrity, dostáváme $q_1 - q_2 = 0$. Tedy $r_1 = r_2$ a $q_1 = q_2$.

Nyní dokážeme, že vůbec nějaké takové q, r existuje. Důkaz provedeme indukcí podle $n - m$.

(I) Předpokládejme $n - m = 0$. Položme $q = \mathrm{lc}(a)$ a $r = \mathrm{lc}(b)a - \mathrm{lc}(a)b$. Pak platí rovnost $\mathrm{lc}(b)a = \mathrm{lc}(a)b + \mathrm{lc}(b)a - \mathrm{lc}(a)b = qb + r$ a stačí dokázat, že $\deg r < n$. Vzhledem k tomu, že polynomy a, b lze napsat jako $a = \mathrm{lc}(a)x^n + a'$ a $b = \mathrm{lc}(b)x^n + b'$, kde $\deg a' < n$, $\deg b' < m$, vidíme, že $r = \mathrm{lc}(b)\mathrm{lc}(a)x^n + \mathrm{lc}(b)a' - \mathrm{lc}(a)\mathrm{lc}(b)x^n - \mathrm{lc}(a)b'$ má koeficient u x^n nulový, a tedy že $\deg r < n$.

(II) Předpokládejme, že $n - m > 0$ a že pro polynomy s menším rozdílem stupňů tvrzení platí. Definujme

$$(\dagger) \qquad c = \mathrm{lc}(b)a - \mathrm{lc}(a)x^{n-m}b,$$

tedy zbytek po prvním kroku školského dělení. Podobně jako v předchozím případě vidíme, že $\deg c < n$.

(1) Jestliže dokonce $\deg c < m$, položme $q' = 0$ a $r' = \mathrm{lc}(b)^{\deg c - m+1}c$. Pak

$$(\ddagger) \qquad \mathrm{lc}(b)^{\deg c-m+1}c = q'b + r', \text{ přičemž } \deg r' < m.$$

(2) Jestliže $m \leq \deg c < n$, pak $\deg c - m < n - m$, a proto z indukčního předpokladu existují pseudopodíl a pseudozbytek c po dělení b, tj. existují q', r' takové, že

$$(\ddagger) \qquad \mathrm{lc}(b)^{\deg c-m+1}c = q'b + r', \text{ přičemž } \deg r' < m.$$

Dosadíme-li definici (†) do (‡), dostáváme rovnost

$$\mathrm{lc}(b)^{\deg c-m+2}a = (q' + \mathrm{lc}(a)\mathrm{lc}(b)^{\deg c-m+1}x^{n-m})b + r'.$$

Přenásobením obou stran koeficientem $\mathrm{lc}(b)^{n-\deg c-1}$ zjistíme, že lze zvolit

$$q = \mathrm{lc}(b)^{n-\deg c-1}q' + \mathrm{lc}(a)\mathrm{lc}(b)^{n-m}x^{n-m} \quad \text{a} \quad r = \mathrm{lc}(b)^{n-\deg c-1}r'.$$

□

11. Posloupnosti polynomiálních zbytků

V této sekci adaptujeme Eukleidův algoritmus na výpočet největšího společného dělitele polynomů tak, aby fungoval pro polynomy nad libovolným Gaussovým oborem, v němž máme k dispozici algoritmus na výpočet NSD. Speciálně tím získáme algoritmus pro výpočet NSD v oborech polynomů více proměnných, pomocí rekurzivní aplikace vztahu $\mathbf{R}[x,y] = (\mathbf{R}[x])[y]$. V dalším textu bude $\mathbf{R}$ značit libovolný Gaussův obor.

Připomeňme, že Eukleidův algoritmus funguje v libovolném Eukleidovském oboru, kterými jsou např. obory polynomů jedné proměnné nad tělesem, nikoliv však obory polynomů více proměnných či polynomy nad $\mathbb{Z}$. Důvod, proč obecně nefunguje, je zhruba řečeno ten, že v obecném oboru $\mathbf{R}[x]$ nemáme k dispozici dělení se zbytkem (např. v $\mathbb{Z}[x]$ neexistuje nic jako podíl a zbytek polynomů x^2-1 a $2x+2$). Na druhou stranu, v úvodní kapitole jsme dokázali, že v každém oboru polynomů máme tzv. *pseudodělení* (viz Tvrzení 10.1), definované vztahy

$$p \text{ pdiv } q = (\mathrm{lc}(q)^{\deg p-\deg q+1}p) \text{ div } q,$$
$$p \text{ pmod } q = (\mathrm{lc}(q)^{\deg p-\deg q+1}p) \text{ mod } q.$$

Motivací následující definice je posloupnost polynomů, která vzniká při chodu Eukleidova algoritmu, přičemž dělení je nahrazeno pseudodělením.

Řekneme, že dva polynomy $p, q \in \mathbf{R}[x]$ jsou *podobné*, píšeme $p \sim q$, pokud existují $k, l \in \mathbf{R}$ takové, že $kp = lq$. (Ekvivalentně $p \sim q$, pokud $p \parallel q$ v $\mathbf{Q}[x]$, kde $\mathbf{Q}$ je podílové těleso $\mathbf{R}$.) Např. $2x+4 \sim 3x+6$ v $\mathbb{Z}[x]$.

Definice. *Posloupností polynomiálních zbytků* v $\mathbf{R}[x]$ (často budeme psát anglickou zkratku *PRS*) rozumíme každou posloupnost $p_1, p_2, \dots, p_k \in \mathbf{R}[x]$ splňující

1. $p_1, \dots, p_{k-1} \neq 0$, $p_k = 0$;
2. $\deg p_1 \geq \deg p_2$;
3. $p_{i+1} \sim (p_{i-1} \text{ pmod } p_i)$.

Všimněte si, že při běhu Eukleidova algoritmu v $\mathbf{T}[x]$ vzniká posloupnost polynomiálních zbytků, neboť každý další polynom p_{i+1} získáme vztahem $p_{i+1} = p_{i-1} \text{ mod } p_i \sim p_{i-1} \text{ pmod } p_i$. Eukleidův algoritmus končí nulovým polynomem, k je tedy počet kroků a $p_{k-1} = \mathrm{NSD}(p_1, p_2)$.

Tvrzení 11.1. *Nechť $\mathbf{R}$ je Gaussův obor a $p_1, p_2, \dots, p_k$ nějaká posloupnost polynomiálních zbytků v $\mathbf{R}[x]$. Pak*

$$\mathrm{NSD}(p_1, p_2) \sim p_{k-1}.$$

Důkaz. Označme $\mathbf{Q}$ podílové těleso $\mathbf{R}$. Protože $p_{i+1} \sim (p_{i-1} \text{ pmod } p_i)$, platí v $\mathbf{Q}[x]$ $p_{i+1} \parallel (p_{i-1} \text{ mod } p_i)$. Protože $\mathrm{NSD}(p,q) = \mathrm{NSD}(q, p \text{ mod } q)$ v libovolném Eukleidovském oboru (viz důkaz Tvrzení 2.9), máme $p_{k-1} = \mathrm{NSD}_{\mathbf{Q}[x]}(p_1, p_2)$, tedy důkaz je hotov užitím Důsledku 9.6. □

Důsledek 11.2. *Nechť* $\mathbf{R}$ *je Gaussův obor,* $p, q \in \mathbf{R}[x]$ *primitivní polynomy a* $p, q, p_3, p_4, \ldots, p_k$ *nějaká posloupnost polynomiálních zbytků v* $\mathbf{R}[x]$*. Pak*

$$\mathrm{NSD}(p, q) \sim p_{k-1}.$$

Tohoto tvrzení využijeme ke konstrukci algoritmu pro výpočet NSD pro polynomy nad libovolným Gaussovým oborem, v němž máme k dispozici algoritmus na výpočet NSD.

Algoritmus 11.3 (NSD pomocí PRS)**.**
`VSTUP:` $p, q \in \mathbf{R}[x]$, předpokládejme $\deg p \geq \deg q$.
`VÝSTUP:` $\mathrm{NSD}(p, q)$.
1. Polož $p_1 := \mathrm{pp}(p)$, $p_2 := \mathrm{pp}(q)$.
2. Spočti nějakou PRS $p_1, p_2, \ldots, p_k$.
3. Spočti $c := \mathrm{NSD}_{\mathbf{R}}(\mathrm{cont}(p), \mathrm{cont}(q))$.
4. `RETURN` $c \cdot \mathrm{pp}(p_{k-1})$.

Správnost algoritmu plyne z Tvrzení 9.2 a Důsledku 11.2. Jeho druhý krok však je poněkud nejednoznačný: jakou PRS máme vzít? Ukážeme si čtyři základní příklady. První dva by napadly asi každého, nejsou však příliš efektivní. Bez důkazu pak uvedeme dva nejpoužívanější.

Definice PRS říká, že p_{i+1} volíme tak, aby $p_{i+1} \sim p_{i-1}$ pmod p_i. Naše snaha je udržovat členy PRS s „co nejmenšími koeficienty", budeme tedy volit

$$p_{i+1} = \frac{1}{\alpha_{i+1}} \cdot (p_{i-1} \text{ pmod } p_i)$$

pro nějaké $\alpha_{i+1} \in \mathbf{R}$ takové, aby výsledek náležel $\mathbf{R}[x]$. Snaha je volit α_{i+1} „co největší", ale zároveň takové, aby jeho výpočet netrval příliš dlouho.
Eukleidovská PRS. Nejjednodušší variantou je nekrátit nic, tedy

$$\alpha_{i+1} = 1, \quad \text{tj. } p_{i+1} = p_{i-1} \text{ pmod } p_i,$$

Tato metoda je nejméně efektivní (srovnatelná s prováděním Eukleidova algoritmu nad podílovým tělesem), kvůli obrovskému, v nejhorším případě až exponenciálně rychlému, růstu koeficientů. Na druhou stranu, výpočet α_{i+1} nic nestojí.

Příklad. Spočtěte $\mathrm{NSD}(x^8+x^6-3x^4-3x^3+8x^2+2x-5, 3x^6+5x^4-4x^2-9x+21)$ v $\mathbb{Z}[x]$.

$$\begin{aligned}
p_1 &= x^8 + x^6 - 3x^4 + 8x^2 + 2x - 5\\
p_2 &= 3x^6 + 5x^4 - 4x^2 - 9x + 21\\
p_3 &= -15x^4 + 3x^2 - 9\\
p_4 &= 15795x^2 + 30375x - 59535\\
p_5 &= 1254542875143750x - 1654608338437500\\
p_6 &= 12593338795500743100931141992187500
\end{aligned}$$

Primitivní PRS. Druhou přirozenou variantou je metoda „vykrať co můžeš", tedy

$$\alpha_{i+1} = \mathrm{cont}(p_{i-1} \text{ pmod } p_i), \quad \text{tj. } p_{i+1} = \mathrm{pp}(p_{i-1} \text{ pmod } p_i).$$

Tato varianta dá samozřejmě optimální koeficienty (rostou nejvýše lineární rychlostí), její velkou nevýhodou však je pomalý výpočet koeficientů α_i, protože se v každém kroku provádí NSD všech koeficientů polynomu p_{i-1} pmod p_i.

Příklad. Spočtěte $\mathrm{NSD}(x^8+x^6-3x^4-3x^3+8x^2+2x-5, 3x^6+5x^4-4x^2-9x+21)$ v $\mathbb{Z}[x]$.

$$\begin{aligned}
p_1 &= x^8 + x^6 - 3x^4 + 8x^2 + 2x - 5\\
p_2 &= 3x^6 + 5x^4 - 4x^2 - 9x + 21\\
p_3 &= 5x^4 - x^2 + 3\\
p_4 &= 13x^2 + 25x - 49\\
p_5 &= 4663x - 6150\\
p_6 &= 1
\end{aligned}$$

Redukovaná PRS. Tato metoda je kompromisem, na který přišel Sylvester kolem roku 1850. Volbou

$$\alpha_3 = 1, \quad \alpha_{i+1} = \mathrm{lc}(p_{i-1})^{\deg p_{i-1} - \deg p_i + 1},\ i \geq 3$$

dostáváme prvek, který lze počítat velmi rychle, a přitom koeficienty polynomů v PRS rostou znatelně pomaleji, než v Eukleidovském případě. Pokud se stupně polynomů p_i a p_{i-1} liší pouze o 1 (pro každé i), pak je tento růst pouze lineární. Důkaz, že výsledný polynom má koeficienty v $\mathbf{R}$, neuvádíme.

Příklad. Spočtěte $\mathrm{NSD}(x^8+x^6-3x^4-3x^3+8x^2+2x-5, 3x^6+5x^4-4x^2-9x+21)$ v $\mathbb{Z}[x]$.

$$\begin{aligned}
p_1 &= x^8 + x^6 - 3x^4 + 8x^2 + 2x - 5\\
p_2 &= 3x^6 + 5x^4 - 4x^2 - 9x + 21\\
p_3 &= -15x^4 + 3x^2 - 9\\
p_4 &= 585x^2 + 1125x - 2205\\
p_5 &= -18885150x + 24907500\\
p_6 &= 527933700
\end{aligned}$$

Subrezultantová PRS. Nejefektivnější v současné době známá metoda, objevená Collinsem roku 1967, je v jistém smyslu vylepšením Sylvesterovy metody; pokud se stupně polynomů p_i a p_{i-1} liší pouze o 1 (pro každé i), jsou tyto PRS totožné. Subrezultantová PRS má lineárně rychlý růst koeficientů v každém případě, a přitom je výpočet α_{i+1} asymptoticky stejně rychlý, jako u redukované PRS. Vzorec však je poněkud komplikovaný:

$$\begin{aligned}
\alpha_3 &= (-1)^{\delta_1+1},\\
\alpha_{i+1} &= (-1)^{\delta_{i-1}+1}\mathrm{lc}(p_{i-1})h_{i-1}^{\delta_{i-1}} \quad \text{pro } i = 3, 4, \ldots, k,
\end{aligned}$$

přičemž $\delta_i = \deg p_i - \deg p_{i+1}$ a koeficienty h_i jsou definovány induktivně

$$h_2 = \mathrm{lc}(p_2)^{\delta_1}, \quad h_i = \mathrm{lc}(p_i)^{\delta_{i-1}} h_{i-1}^{1-\delta_{i-1}} \quad \text{pro } i = 3, 4, \ldots, k.$$

Důkaz, že výsledný polynom má koeficienty v $\mathbf{R}$, je dosti komplikovaný a uvádět jej nebudeme. Zájemce můžeme odkázat na skripta Aleše Drápala dostupná na adrese `www.karlin.mff.cuni.cz/~krypto/seminar/nfs.pdf`.

Příklad. Spočtěte $\mathrm{NSD}(x^8+x^6-3x^4-3x^3+8x^2+2x-5, 3x^6+5x^4-4x^2-9x+21)$ v $\mathbb{Z}[x]$.

$$\begin{aligned}
p_1 &= x^8 + x^6 - 3x^4 + 8x^2 + 2x - 5\\
p_2 &= 3x^6 + 5x^4 - 4x^2 - 9x + 21\\
p_3 &= 15x^4 - 3x^2 + 9\\
p_4 &= 65x^2 + 125x - 245\\
p_5 &= 9326x - 12300\\
p_6 &= 260708
\end{aligned}$$

12. Rezultant a Sylvesterovo kritérium nesoudělnosti

Jak jsme zmínili v předešlé sekci, úplný důkaz subrezultantové PRS provádět nebudeme. Přesto je však užitečné říci, co to vlastně ten (sub)rezultant je a jak souvisí s dělitelností polynomů. Sylvesterovo kritérium nesoudělnosti využívající rezultant ještě v budoucnu použijeme.

Definice. Nechť $p, q \in \mathbf{R}[x]$, $n = \deg p$, $m = \deg q$ a označme $p = \sum_{i=0}^{n} a_i x^i$, $q = \sum_{i=0}^{m} b_i x^i$. *Sylvestrovou maticí* $M(p,q)$ rozumíme matici typu $(m+n)\times(m+n)$ nad $\mathbf{R}$ definovanou předpisem

$$M(p,q) = \begin{pmatrix} a_n & a_{n-1} & \dots & \dots & a_0 & \dots & \dots & \dots & 0 \\ 0 & a_n & a_{n-1} & \dots & \dots & a_0 & \dots & \dots & 0 \\ 0 & 0 & a_n & a_{n-1} & \dots & \dots & a_0 & \dots & 0 \\ \vdots & & & & & & & & \\ 0 & \dots & \dots & \dots & 0 & a_n & \dots & \dots & a_0 \\ b_m & b_{m-1} & \dots & \dots & b_0 & 0 & \dots & \dots & 0 \\ 0 & b_m & b_{m-1} & \dots & \dots & b_0 & 0 & \dots & 0 \\ 0 & 0 & b_m & b_{m-1} & \dots & \dots & b_0 & \dots & 0 \\ \vdots & & & & & & & & \\ 0 & \dots & \dots & \dots & \dots & 0 & b_m & \dots & b_0 \end{pmatrix}$$

Determinant této matice se nazývá *rezultant* polynomů p, q a značí se

$$\operatorname{res}(p,q) = \det M(p,q).$$

Příklad. Nechť $p = 2x^4 + x^2 - 4$ a $q = 3x^2 + 2$. Pak

$$M(p,q) = \begin{pmatrix} 2 & 0 & 1 & 0 & -4 & 0 \\ 0 & 2 & 0 & 1 & 0 & -4 \\ 3 & 0 & 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 & 0 & 2 \end{pmatrix}$$

a není těžké spočítat, že $\operatorname{res}(p,q) = 1156$.

Resultant dvou polynomů rozhoduje o (ne)soudělnosti polynomů.

Tvrzení 12.1. *Buď* $\mathbf{R}$ *Gaussův obor,* $p, q \in \mathbf{R}[x]$, $n = \deg p > 0$ *a* $m = \deg q > 0$. *Pak existují nenulové polynomy* $s, t \in \mathbf{R}[x]$ *takové, že* $\deg s < \deg q$, $\deg t < \deg p$ *a*

$$\operatorname{res}(p,q) = ps + qt.$$

Důkaz. (A) Nejprve předpokládejme, že $\operatorname{res}(p,q) \neq 0$, tj. že matice $M(p,q)$ je regulární. Všimněme si, že platí

$$M(p,q) \cdot \begin{pmatrix} x^{m+n-1} \\ x^{m+n-2} \\ \dots \\ x^{n+1} \\ x^n \\ x^{n-1} \\ \dots \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} x^{m-1}p \\ x^{m-2}p \\ \dots \\ xp \\ p \\ x^{n-1}q \\ \dots \\ xq \\ q \end{pmatrix}$$

Máme tedy soustavu lineárních rovnic

$$\left(\begin{array}{c|c} & x^{m-1}p \\ M(p,q) & \vdots \\ & q \end{array} \right)$$

a víme, že vektor $(x^{m+n-1}, \dots, 1)$ je jejím jediným řešením (jediným, protože je $M(p,q)$ regulární). Řešení regulární soustavy lze ale spočítat také pomocí Cramérova pravidla, které říká, že soustava $Ax = b$ má řešení $x = (x_1, \dots, x_k)$, přičemž i-tou souřadnici lze spočítat podle vzorce $x_i = \frac{\det A_i}{\det A}$, kde A_i je matice, která vznikne z matice A nahrazením i-tého sloupce za sloupec b. V našem případě, pro $i = m+n$, dostáváme

$$1 = \frac{\det(M(p,q)_{m+n})}{\det M(p,q)},$$

a tedy

$$\operatorname{res}(p,q) = \det(M(p,q)_{m+n}).$$

Determinant matice $M(p,q)_{m+n}$ spočítáme rozvojem podle posledního sloupce. (Připomeňme, že rozvojem determinantu matice $A = (a_{ij})$ podle j-tého sloupce se rozumí vzorec $\det A = \sum_{i=1}^{k} (-1)^{i+j} a_{ij} \det A_{ij}$, kde matice A_{ij} vzniká z matice A vynecháním i-tého řádku a j-tého sloupce.) V našem případě je poslední sloupec ten nahrazený, tedy $(x^{m-1}p, \dots, p, x^{n-1}q, \dots, q)^T$. Dostáváme

$$\begin{aligned}
\operatorname{res}(p,q) &= \sum_{i=1}^{m} (-1)^{m+n+i} \cdot x^{m-i} \cdot p \cdot \det(M(p,q)_{m+n})_{i,m+n} + \\
&\quad + \sum_{i=1}^{n} (-1)^{m+i+m+n} \cdot x^{n-i} \cdot q \cdot \det(M(p,q)_{m+n})_{m+i,m+n} \\
&= p \cdot \sum_{i=1}^{m} \underbrace{(-1)^{m+n+i} \cdot \det(M(p,q)_{m+n})_{i,m+n}}_{u_i} \cdot x^{m-i} + \\
&\quad + q \cdot \sum_{i=1}^{n} \underbrace{(-1)^{m+i+m+n} \cdot \det(M(p,q)_{m+n})_{m+i,m+n}}_{v_i} \cdot x^{n-i} \\
&= p \cdot \sum_{i=1}^{m} u_i \cdot x^{m-i} + q \cdot \sum_{i=1}^{n} v_i \cdot x^{n-i} = ps + qt,
\end{aligned}$$

přičemž $u_i, v_i \in \mathbf{R}$ jsou koeficienty definované výše, $s = \sum_{i=1}^{m} u_i \cdot x^{m-i}$ je polynom stupně méně než m a $t = \sum_{i=1}^{n} v_i \cdot x^{n-i}$ je polynom stupně méně než n.

(B) Nyní předpokládejme, že $\operatorname{res}(p,q) = 0$, označme $M(p,q) = (m_{ij})_{i,j=1}^{m+n}$. Protože je matice $M(p,q)$ singulární, existuje netriviální lineární kombinace řádkových vektorů rovná nule. Tedy existují prvky $\alpha_1, \dots, \alpha_{m+n} \in \mathbf{R}$, aspoň jeden z nich nenulový, splňující

$$\sum_{i=1}^{m+n} \alpha_i \cdot m_{ij} = 0$$

pro každé j. Pak také samozřejmě

$$x^{m+n-j} \cdot \sum_{i=1}^{m+n} \alpha_i \cdot m_{ij} = \sum_{i=1}^{m+n} \alpha_i \cdot m_{ij} \cdot x^{m+n-j} = 0$$

pro každé j a součet přes všechna j dává

$$\sum_{j=1}^{m+n} \sum_{i=1}^{m+n} \alpha_i \cdot m_{ij} \cdot x^{m+n-j} = 0.$$

Prohozením sum dostáváme

$$0 = \sum_{i=1}^{m+n} \sum_{j=1}^{m+n} \alpha_i \cdot m_{ij} \cdot x^{m+n-j} = \sum_{i=1}^{m} \alpha_i \cdot x^{m-i} \cdot p + \sum_{i=1}^{n} \alpha_{m+1} \cdot x^{n-i} \cdot q.$$

Položíme-li $s = \sum_{i=1}^{m} \alpha_i x^{m-i}$ a $t = \sum_{i=1}^{n} \alpha_{m+1} x^{n-1}$, dostáváme

$$\operatorname{res}(p,q) = 0 = sp + tq.$$

Přitom $s, t \neq 0$, protože aspoň jeden z koeficientů α_i je nenulový. □

Důsledek 12.2 (Sylvestrovo kritérium)**.** *Nechť* $\mathbf{R}$ *je Gaussův obor,* $\mathbf{Q}$ *jeho podílové těleso a* $p, q \in \mathbf{R}[x]$*. Pak následující tvrzení jsou ekvivalentní:*

(1) $\deg(\mathrm{NSD}_{\mathbf{R}[x]}(p,q)) = 0$.
(2) $\mathrm{NSD}_{\mathbf{Q}[x]}(p,q) = 1$.
(3) $\operatorname{res}(p,q) \neq 0$.

Důkaz. (1) $\Leftrightarrow$ (2) plyne z Tvrzení 9.2.

(2) $\Rightarrow$ (3) Pro spor předpokládejme, že $\operatorname{res}(p,q) = 0$ a napišme $0 = sp + tq$ pro nějaké polynomy $s, t \neq 0$, $\deg s < \deg q$, $\deg t < \deg p$. Pak $sp = -tq$, a tedy $p \mid tq$. Ovšem p, q jsou nesoudělné, takže $p \mid t$. Jenže $\deg t < \deg p$, tedy $t = 0$, spor.

(3) $\Rightarrow$ (2) Napišme $\operatorname{res}(p,q) = sp+tq$ a uvažujme netriviálního společného dělitele u, tj. $\deg u > 0$ a $u \mid p, q$. Pak $u \mid sp + tq$, ovšem $\deg(sp + tq) = 0$, spor. □

Poznámka. Platí i silnější věta, jejíž důkaz leží stranou našeho zájmu: *Stupeň polynomu* $\mathrm{NSD}(p,q)$ *je roven nejmenšímu* j *takovému, že* $\det M_j(p,q) \neq 0$. Matice $M_j(p,q)$ vzniká z $M(p,q)$ vyškrtnutím posledních j řádků z horní části (část matice s koeficienty a_i), posledních j řádků z dolní části (část matice s koeficienty b_i) a posledních $2j$ sloupců.

Příklad. Nechť $p = x^2 - 1$ a $q = x^2 + 2x + 1$. Spočítáme-li Sylvesterovu matici

$$M(p,q) = M_0(p,q) = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \end{pmatrix},$$

vidíme, že $\operatorname{res}(p,q) = \det M(p,q) = 0$. Pokračujeme-li dále, dostáváme

$$M_1(p,q) = \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix},$$

a vidíme, že $\det M_1(p,q) = 2$. Podle výše uvedené poznámky se dozvídáme, že stupeň polynomu $\mathrm{NSD}(p,q)$ je 1. (Každý už od začátku vidí, že $\mathrm{NSD}(p,q) = x+1$.)

Definice. Nechť $M_{i,j}(p,q)$ značí matici, která vznikne z $M(p,q)$ vyškrtnutím posledních j řádků z horní části, posledních j řádků z dolní části a posledních $2j+1$ sloupců s výjimkou $(m+n-i-j)$-tého. Definujeme j*-tý subrezultant* předpisem

$$S_j(p,q) = \sum_{i=0}^{j} \det(M_{i,j}(p,q)) \cdot x^i.$$

Příklad. Nechť $p = 2x^4 + x^2 - 4$ a $q = 3x^2 + 2$. Pak

$$\det M_{0,1}(p,q) = \begin{vmatrix} 2 & 0 & -1 & 4 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 2 \end{vmatrix} = 102$$

a

$$\det M_{1,1}(p,q) = \begin{vmatrix} 2 & 0 & 1 & 0 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 2 \\ 0 & 0 & 3 & 0 \end{vmatrix} = 0$$

Tedy

$$S_0(p,q) = \operatorname{res}(p,q) = 1156,$$
$$S_1(p,q) = \det M_{0,1}(p,q) \cdot x^0 + \det M_{1,1}(p,q) \cdot x^1 = 102 + 0 \cdot x = 102.$$

Všimněte si, že pro polynomy z předchozího příkladu platí $p \operatorname{pmod} q = 102$. Tato podobnost *není* náhodná! Vztah subrezultantů a dělitelnosti je zakódován v následující větě.

Věta 12.3 (Fundamentální věta o PRS). *Buď* $\mathbf{R}$ *Gaussův obor,* $p_1, p_2, \dots, p_k$ *nějaká PRS v* $\mathbf{R}[x]$ *a označme* $n_i = \deg p_i$. *Pak pro všechna* $i = 3, \dots, k-1$ *platí*

$$S_{n_{i-1}-1}(p_1,p_2) \sim p_i,$$
$$S_{n_i}(p_1,p_2) \sim p_i,$$
$$S_j(p_1,p_2) = 0 \text{ pro všechna } j = n_i+1, \dots, n_{i-1}-2.$$

Důkaz. Důkaz této věty je dosti technický a uvádět jej nebudeme. Případné zájemce můžeme odkázat na skripta Aleše Drápala dostupná na adrese `www.karlin.mff.cuni.cz/~krypto/seminar/nfs.pdf`. □

Tato věta říká následující: máme-li libovolnou PRS začínající polynomy p, q a sledujeme-li hodnoty $S_j(p,q)$ pro $j = 0, 1, 2, \dots, \deg q - 1$, dostaneme následující posloupnost:

$$0, \dots, 0, p_{k-1}, 0, \dots, 0, p_{k-1}, p_{k-2}, 0, \dots, 0, p_{k-2}, p_{k-3}, \dots, \dots, p_4, p_3, 0, \dots, 0, p_3.$$

Tedy bez ohledu na krácení v té PRS (ve smyslu bez ohledu na volbu α_i) platí, že každé p_i je podobné některému subrezultantu p, q, a to konkrétně n_i-tému a $(n_{i-1}-1)$-tému (pokud $\delta_i = 1$, pak jde o tentýž člen té posloupnosti).

Fundamentální věta o PRS dává návod, jak získat subrezultantovou PRS: pokud vyjdeme ze vztahu $p_i = S_{n_i}(p, q)$, získáme po dlouhém výpočtu rekurentní vzorec uvedený v předchozí sekci. (Odvození je opět možné najít v již zmíněných skriptech Aleše Drápala.)

13. Modulární algoritmus v $\mathbb{Z}[x]$

Jiný způsob, jak bojovat s příliš rychle rostoucími koeficienty v Eukleidově algoritmu, je počítání modulo (malé) prvočíslo. Jinými slovy, místo v $\mathbb{Z}[x]$ budeme počítat NSD v $\mathbb{Z}_k[x]$ pro různá prvočísla k a původní řešení zrekonstruujeme za pomoci Čínské věty o zbytcích. Hlavní problém, který budeme muset vyřešit, je ten, že modulo k se někdy daný polynom rozkládá na více faktorů, než nad $\mathbb{Z}$, a tedy NSD často vyjde modulo k větší, než by měl. Jádrem algoritmu je tvrzení, že prvočísel, pro která tato nešťastná situace nastane (nazývají se *smolná)*, není příliš mnoho.

Nebude-li výslovně uvedeno jinak, v celé sekci budeme předpokládat, že p, q jsou nějaké nenulové polynomy ze $\mathbb{Z}[x]$ a $p \bmod k, q \bmod k$ považujeme za prvky $\mathbb{Z}_p[x]$.

Příklad. Uvažujme polynomy

$$p = x^3 - x^2 + x - 1 = (x^2 + 1)(x - 1),$$
$$q = x^3 + 2x^2 - x - 2 = (x - 1)(x + 1)(x + 2).$$

Vidíme, že $\mathrm{NSD}(p, q) = x - 1$.

- Počítejme modulo 2. Protože $p \bmod 2 = (x + 1)^3$ a $q \bmod 3 = x(x + 1)^2$, dostáváme
$$\mathrm{NSD}_{\mathbb{Z}_2[x]}(p \bmod 3, q \bmod 3) = (x + 1)^2.$$
Tedy 2 je v tomto případě smolné prvočíslo.
- Počítejme modulo 3. Protože $p \bmod 3 = (x^2 + 1)(x - 1)$ a $q \bmod 3 = (x - 1)(x + 1)(x - 1)$, dostáváme
$$\mathrm{NSD}_{\mathbb{Z}_3[x]}(p \bmod 3, q \bmod 3) = x - 1.$$
Tedy 3 je v tomto případě šťastné prvočíslo.
- Počítejme modulo 5. Protože $p \bmod 5 = (x - 2)(x + 2)(x - 1)$ a $q \bmod 5 = (x - 1)(x + 1)(x + 2)$, dostáváme
$$\mathrm{NSD}_{\mathbb{Z}_5[x]}(p \bmod 5, q \bmod 5) = (x - 1)(x + 2).$$
Tedy 5 je v tomto případě také smolné prvočíslo.
- Počítejme modulo 7. Protože $p \bmod 7 = (x^2 + 1)(x - 1)$ a $q \bmod 7 = (x - 1)(x + 1)(x + 2)$, dostáváme
$$\mathrm{NSD}_{\mathbb{Z}_7[x]}(p \bmod 7, q \bmod 7) = x - 1.$$
Tedy 7 je šťastné prvočíslo.

Otázka je: jak poznat, že je prvočíslo smolné, aniž bychom znali výsledek předem? Postup je takový, že budeme sledovat stupně největších společných dělitelů modulo k. Podle následujících tvrzení jsou tito vždy větší než stupeň NSD v $\mathbb{Z}[x]$. Dokud se tedy tyto stupně zmenšují, můžeme si být jisti, že jsme šťastné číslo neobjevili.

Lemma 13.1. *Označme* $r = \mathrm{NSD}_{\mathbb{Z}[x]}(p, q)$. *Pak*

$$\mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k) = (r \bmod k) \cdot \mathrm{NSD}_{\mathbb{Z}_k[x]}\left(\frac{p}{r} \bmod k, \frac{q}{r} \bmod k\right).$$

Důkaz. Označme $a = \frac{p}{r}$ a $b = \frac{q}{r}$. Platí tedy $p = ra$ a $q = rb$, a proto také $p \bmod k = (r \bmod k)(a \bmod k)$ a $q \bmod k = (r \bmod k)(b \bmod k)$. Vidíme, že $\mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k) = (r \bmod k) \cdot \mathrm{NSD}_{\mathbb{Z}_k[x]}(a \bmod k, b \bmod k) = \mathrm{NSD}_{\mathbb{Z}_k[x]}(\frac{p}{r} \bmod k, \frac{q}{r} \bmod k)$. □

Podívejte se ještě jednou na úvodní příklad!

Důsledek 13.2. *Jestliže číslo k nedělí $\mathrm{lc}(p)$ ani $\mathrm{lc}(q)$, pak*

$$\deg \mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k) \geq \deg \mathrm{NSD}_{\mathbb{Z}[x]}(p, q).$$

Důkaz. Pokud k nedělí $\mathrm{lc}(p)$ ani $\mathrm{lc}(q)$, pak nědělí ani $\mathrm{lc}(\mathrm{NSD}(p, q))$, a tedy stupeň $\mathrm{NSD}(p, q) \bmod k$ je stejný jako stupeň $\mathrm{NSD}(p, q)$. Nerovnost plyne z předchozího lemmatu. □

Poznamenejme, že podmínka na nedělitelnost vedoucích koeficientů je nutná: např. pro $p = q = 2x + 1$, $k = 2$ máme $\mathrm{NSD}_{\mathbb{Z}[x]}(p, q) = 2x + 1$, ale $\mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k) = 1$.

Důsledkem uvedených faktů je také následující jednoduchý test na nesoudělnost:

Důsledek 13.3. *Pokud k nedělí $\mathrm{lc}(p)$ ani $\mathrm{lc}q$ a jsou-li $p \bmod k$ a $q \bmod k$ nesoudělné v některém $\mathbb{Z}_k[x]$, pak jsou p, q nesoudělné v $\mathbb{Z}[x]$.*

Důsledek 13.2 motivuje následující definici.

Definice. Řekneme, že prvočíslo k je *použitelné* pro polynomy p, q, pokud k nedělí $\mathrm{lc}(p)$ ani $\mathrm{lc}(q)$.

- Použitelné prvočíslo k se nazývá *šťastné* pro polynomy p, q, pokud
$$\deg \mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k) = \deg \mathrm{NSD}_{\mathbb{Z}[x]}(p, q).$$
- Použitelné prvočíslo k se nazývá *smolné* pro polynomy p, q, pokud
$$\deg \mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k) > \deg \mathrm{NSD}_{\mathbb{Z}[x]}(p, q).$$

Lemma 13.4. *Nechť k je použitelné smolné prvočíslo pro polynomy p, q. Pak*

$$k \mid \mathrm{res}\left(\frac{p}{\mathrm{NSD}(p, q)}, \frac{q}{\mathrm{NSD}(p, q)}\right).$$

Důkaz. Protože k je použitelné smolné, platí $\deg r < \deg \mathrm{NSD}(p \bmod k, q \bmod k)$. Tedy, podle vzorce 13.1, $\deg \mathrm{NSD}_{\mathbb{Z}_k[x]}(\frac{p}{r} \bmod k, \frac{q}{r} \bmod k) \neq 1$. Podle Sylvestrova kritéria 12.2 tedy platí $\mathrm{res}_{\mathbb{Z}_k[x]}(\frac{p}{r} \bmod k, \frac{q}{r} \bmod k) = 0$. Protože rezultant je determinantem jisté matice, nezáleží, zda nejprve dosazujeme hodnoty modulo k a poté počítáme v $\mathbb{Z}_k$, nebo zda nejprve počítáme tento determinant v $\mathbb{Z}$ a poté výsledek upravíme modulo k. Tedy platí $\mathrm{res}_{\mathbb{Z}[x]}(\frac{p}{r}, \frac{q}{r}) \equiv 0 \pmod{k}$, neboli $k \mid \mathrm{res}(\frac{p}{r}, \frac{q}{r})$. □

Důsledek 13.5. *Použitelných smolných prvočísel pro dané polynomy p, q je jen konečně mnoho.*

Důkaz. Smolných prvočísel je jen tolik, kolik je prvočíselných dělitelů toho resultantu. □

Protože je smolných prvočísel jen konečně mnoho, při modulárním výpočtu NSD musíme dříve nebo později narazit na nějaké šťastné. Zbývá vyřešit otázku, kolik šťastných prvočísel potřebujeme k provedení Čínské věty o zbytcích, abychom zrekonstruovali NSD v $\mathbb{Z}[x]$. Následující tvrzení dává horní mez na velikost koeficientů největšího společného dělitele.

Tvrzení 13.6 (Landau-Mignottova mez)**.** *Předpokládejme, že polynom* $r = \sum_{i=0}^{k} c_i x^i \in \mathbb{Z}[x]$ *dělí polynom* $p = \sum_{i=0}^{n} a_i x^i \in \mathbb{Z}[x]$ *(*$a_n, c_k \neq 0$*). Pak*

$$\sum_{i=0}^{k} |c_i| \leq 2^k \cdot \left|\frac{c_k}{a_n}\right| \cdot \sqrt{\sum_{i=0}^{n} a_i^2}.$$

Důkaz. Nebude (viz []). □

Důsledek 13.7. *Nechť* $p = \sum_{i=0}^{n} a_i x^i$, $q = \sum_{i=0}^{m} b_i x^i$ *a* $\mathrm{NSD}(p,q) = \sum_{i=0}^{k} c_i x^i$ *(*$a_n, b_m, c_l \neq 0$*, vše v* $\mathbb{Z}[x]$*). Pak pro všechna i platí*

$$|c_i| \leq 2^{\min(m,n)} \cdot |\mathrm{NSD}(a_n, b_m)| \cdot \min\left(\frac{1}{|a_n|}\sqrt{\sum_{i=0}^{n} a_i^2}, \frac{1}{|b_n|}\sqrt{\sum_{i=0}^{m} b_i^2}\right).$$

Důkaz. Uvědomíme si, že c_k dělí a_n i b_m, tedy i jejich NSD a použijeme předchozí tvrzení. □

Pro účely algoritmu označme $\mathrm{LM}(p,q)$ jakoukoliv mez na velikost koeficientů největšího společného dělitele polynomů p, q. Podle Důsledku 13.7 lze zvolit např. $\mathrm{LM}(p,q) = 2^{\min(m,n)} \cdot |\mathrm{NSD}(a_n, b_m)| \cdot \min(\frac{1}{|a_n|}\sqrt{\sum_{i=0}^{n} a_i^2}, \frac{1}{|b_n|}\sqrt{\sum_{i=0}^{m} b_i^2})$.

Nyní si ukážeme nejjednodušší způsob, jak využít modulární metodu k výpočtu NSD: Zvolíme dostatečně velké prvočíslo, abychom se vyhnuli aplikaci Čínské věty o zbytcích. Tento postup není příliš efektivní, ale je na něm dobře vidět, jak testovat, zda je zvolené prvočíslo šťastné.

Poznámka. $\mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k)$ je polynom ze $\mathbb{Z}_k[x]$, ale my jej budeme považovat také za polynom ze $\mathbb{Z}[x]$, přičemž jeho koeficienty bereme z množiny $\{-\frac{k-1}{2}, \dots, \frac{k+1}{2}\}$. Přistoupíme-li na tuto konvenci, vidíme, že pro šťastné $k > 2\mathrm{LM}(p,q)$ platí

$$\mathrm{NSD}_{\mathbb{Z}[x]}(p,q) = \mathrm{NSD}_{\mathbb{Z}_k[x]}(p,q),$$

pokud má polynom na pravé straně stejný vedoucí koeficient jako polynom na levé straně. (Připomeňme, že NSD nad tělesem je určen jednoznačně až na násobek libovolným nenulovým prvkem tělesa.)

Setkáváme se s tzv. *problémem vedoucího koeficientu.* Lze jej řešit takto: Předpokládáme, že polynomy p, q jsou primitivní (to můžeme podle 9.2). Vezmeme $k > 2d\mathrm{LM}(p,q)$, spočteme ten $\mathrm{NSD}_{\mathbb{Z}_k[x]}(p,q)$, jehož vedoucím koeficientem je $d := \mathrm{NSD}_{\mathbb{Z}}(\mathrm{lc}(p), \mathrm{lc}(q))$, a spočteme jeho primitivní část. Protože vedoucí koeficient $\mathrm{NSD}_{\mathbb{Z}[x]}(p,q)$ dělí d a $\mathrm{NSD}_{\mathbb{Z}[x]}(p,q)$ je primitivní (viz Gaussovo lemma), uvedený postup funguje.

Algoritmus 13.8 (Modulární algorimus na NSD, verze I)**.**
`VSTUP:` $p, q \in \mathbb{Z}[x]$ primitivní.
`VÝSTUP:` $\mathrm{NSD}(p,q)$.
`0.` Spočti $d := \mathrm{NSD}(\mathrm{lc}(p), \mathrm{lc}(q))$.
`1.` Zvol k nejmenší použitelné prvočíslo větší než $2d\mathrm{LM}(p,q)$.
`2.` Spočti $r := \mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k)$, aby $\mathrm{lc}(r) = d$.
`3.` `IF` $\mathrm{pp}(r) \mid p$ a $\mathrm{pp}(r) \mid q$ `THEN RETURN` $\mathrm{pp}(r)$, `STOP`
`ELSE` předefinuj k jako nejmenší větší nepoužité použitelné prvočíslo větší než k, `GOTO 2`.

Tvrzení 13.9. *Algoritmus 13.8 funguje.*

Důkaz. Pokud je prvočíslo k šťastné, z diskuze nad algoritmem plyne, že algoritmus vrátí správný výsledek. Algoritmus opakuje kroky `2.,3.` tak dlouho, dokud nenalezne šťastné prvočíslo: díky vzorci 13.2 je $\mathrm{pp}(r)$ společným dělitelem právě tehdy, když je největším společným dělitelem. Jinými slovy, šťastnost k je ověřená tím, zda je kandidát na NSD skutečně společným dělitelem. Algoritmus se zastaví po konečně mnoha cyklech díky Důsledku 13.5. □

Problémem uvedeného algoritmu je, že mez LM je obvykle mnohem větší, než jsou ve skutečnosti koeficienty NSD (jde o mez *v nejhorším případě*, typicky je situace mnohem příznivější), a tedy úspora na velikosti koeficientů je diskutabilní. O něco komplikovanější, ale v praxi o dost efektivnější algoritmus je následující, využívající malých prvočísel a Čínské věty o zbytcích.

Algoritmus 13.10 (Modulární algorimus na NSD, verze II)**.**

`VSTUP:` $p, q \in \mathbb{Z}[x]$ primitivní.

`VÝSTUP:` $\mathrm{NSD}(p, q)$.

`0.` Spočti $d := \mathrm{NSD}_{\mathbb{Z}}(\mathrm{lc}(p), \mathrm{lc}(q))$

`1.` Zvol k nejmenší dosud nepoužité použitelné prvočíslo.

`2.` Spočti $r := \mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k)$, aby $\mathrm{lc}(r) \equiv d \pmod k$

`3.` `IF` $\deg r = 0$ `THEN RETURN` 1, `STOP`
  `ELSE` polož $R := r$ a $K := k$.

`4.` `WHILE` $K \leq 2d\mathrm{LM}(p, q)$ `DO`:
  Nechť k je nejmenší dosud nepoužité použitelné prvočíslo.
  Spočti $r := \mathrm{NSD}_{\mathbb{Z}_k[x]}(p \bmod k, q \bmod k)$, aby $\mathrm{lc}(r) \equiv d \pmod k$
  `IF` $\deg r < \deg R$ `THEN GOTO 3.`
  `IF` $\deg r = \deg R$ `THEN`
    Spočti polynom s splňující $s \equiv r \pmod k$ a $s \equiv R \pmod K$
    (aplikuj ČVZ na každý koeficient (tj. $\deg R$-krát)).
    Polož $R := s$ a $K := K \cdot k$.

`5.` `IF` $\mathrm{pp}(R) \mid p$ a $\mathrm{pp}(R) \mid q$ `THEN RETURN` $\mathrm{pp}(R)$ `ELSE GOTO 1.`

Tvrzení 13.11. *Algoritmus 13.10 funguje.*

Důkaz. Okomentujeme běh algoritmu. Na začátku zvolíme nové použitelné prvočíslo k. Pro něj spočteme NSD daných polynomů mod k. Pokud vyjde NSD stupně nula, jsou polynomy p, q díky Důsledku 13.3 nesoudělné. V opačném případě budeme předpokládat, že jde o šťastné prvočíslo a začneme střádat dostatečné množství NSD modulo různá k tak, aby z nich bylo možno zrekonstruovat NSD v $\mathbb{Z}[x]$. V K udržujeme součin všech aktuálních prvočísel, jež považujeme za šťastná, v R udržujeme hodnotu $\mathrm{NSD}(p, q)$ mod K. Dokud K nepřevýší $2d\mathrm{LM}(p, q)$, generujeme další použitelná prvočísla k. Pro každé z nich spočítáme NSD mod k. Pokud má tento polynom větší stupeň než všechny předchozí, je toto prvočíslo smolné a ignorujeme jej (viz 13.2). Pokud má tento polynom menší stupeň než všechny předchozí, znamená to, že všechna předchozí prvočísla byla smolná (viz 13.2) a vracíme se zpět na začátek výpočtu. V případě rovnosti stupňů spočítáme pomocí některého algoritmu na Čínskou větu o zbytcích NSD mod $K \cdot k$ a pokračujeme dalším prvočíslem. V kroku `5.` si stále nemůžeme být jisti, že jsme našli nějaké šťastné prvočíslo. Vyzkoušíme tedy, zda je výsledek společným dělitelem polynomů p, q. Pokud ano, je to výsledek. Pokud ne, všechna prvočísla byla dosud smolná a algoritmus pokračuje znovu od začátku. (Viz důkaz Tvrzení 13.9).

Všimněte si, že algoritmus skončí v konečném čase, protože všechny skoky zpět probíhají pouze v případě, kdy jsme prokazatelně narazili na smolné prvočíslo. Těch je jen konečně mnoho podle Důsledku 13.5. □

Poznamenejme, že Landau-Mignottova mez je pro většinu polynomů značně nadnesená, nemá tedy obvykle smysl počítat tak dlouho, až K převýší $\mathrm{LM}(p,q)$ mez, ale vyplatí se již v průběhu výpočtu zkoušet, zda R není náhodou společným dělitelem. Účinnou *heuristikou* znatelně zrychlující chod algoritmu je následující podmínka: Pokud v cyklu v kroku `4.` vyšlo dvakrát po sobě stejné R, aplikuj test z kroku `5.` Pokud vyjde, tj. pokud $\mathrm{pp}(R) \mid p$ a $\mathrm{pp}(R) \mid q$, `RETURN` $\mathrm{pp}(R)$, `STOP`. Jinak pokračuj výpočet jako by se nic nestalo.

Tvrzení 13.12. *Časová složitost Algoritmu 13.10 je $\mathcal{O}(n^3(\log n + c)^2)$ operací s ciframi, přičemž n značí větší ze stupňů p, q a c značí počet cifer největšího z koeficientů p, q.*

Důkaz. Nebude (viz []). □

Asymptotický odhad složitosti toho algoritmu sice vypadá hůře, než u metody PRS, ale uvědomte si, že zde počítáme *počet operací s ciframi*, zatímco v odhadu pro PRS jsme počítali *počet operací v $\mathbb{Z}$*! Navíc, toto je odhad složitosti *v nejhorším případě*, který ale mnoho neříká o typickém chování. Např. pro nesoudělné polynomy je modulární algoritmus poměrně rychlý (díky podmínce v kroku 3.), zatímco pro metodu PRS jsou nesoudělné polynomy nejpomalejším případem.

14. Modulární algoritmus pro polynomy více proměnných

Předpokládejme, že máme algoritmus pro výpočet NSD v oboru $\mathbf{R}[x]$. Adaptujeme modulární algoritmus z předchozí sekce na výpočet NSD v oboru $\mathbf{R}[x,y]$. Rekurzivní aplikací tak získáme algoritmus pro výpočet NSD v oborech polynomů libovolného množství proměnných nad $\mathbb{Z}$ či nad libovolným tělesem. V témto sekci budeme předpokládat, že p, q jsou nějaké nenulové polynomy z $\mathbf{R}[x,y]$.

Finta je analogická: budeme počítat modulo polynom $y-\alpha$, pro různá $\alpha \in \mathbf{R}$. Jinými slovy, za proměnnou y budeme dosazovat různé hodnoty α a výsledek opět rekonstruovat pomocí Čínské věty o zbytcích. Polynomy z $\mathbf{R}[x,y]$ tedy budeme uvažovat v rekurzivní reprezentaci $(\mathbf{R}[y])[x]$, tj. koeficienty jsou polynomy z $\mathbf{R}[y]$.

Příklad. Uvažujme polynomy

$$p = yx^2 + (1-y^2)x - y = (xy+1)(x-y),$$
$$q = yx^2 + (1+y^2)x + y = (xy+1)(x+y).$$

Všimněme si, že všechny koeficienty (jako polynomy nad y) mají stupeň nejvýše 2. Totéž musí platit i pro jejich největšího společného dělitele, a tedy na jeho zrekonstruování budou stačit tři hodnoty.

- $\alpha = 1$: Protože $p(x,1) = x^2 - 1$ a $q(x,1) = x^2 + 2x + 1$, dostáváme $\mathrm{NSD}(p,q)(x,1) = x + 1$.
- $\alpha = 2$: Protože $p(x,2) = 2x^2 - 3x - 2$ a $q(x,2) = 2x^2 + 5x + 2$, dostáváme $\mathrm{NSD}(p,q)(x,2) = 2x + 1$.
- $\alpha = 3$: Protože $p(x,3) = 3x^2 - 8x - 3$ a $q(x,3) = 3x^2 + 10x + 3$, dostáváme $\mathrm{NSD}(p,q)(x,3) = 3x + 1$.

Tedy $\mathrm{NSD}(p,q) = yx + 1$.

V tomto případě nám náhodou vyšly všechny hodnoty šťastné. Proč jsme nevolili $\alpha = 0$? Tato hodnota není použitelná, protože $y - 0$ dělí vedoucí koeficienty p, q. (Kdybychom ji použili, vyšlo by nám $\mathrm{NSD}(p,q)(x,0) = x$, což je špatně.)

Následující tvrzení lze dokázat analogicky jako v případě polynomů jedné proměnné nad $\mathbb{Z}$: všude je obor $\mathbb{Z}_k$ je nahrazen oborem $\mathbf{R}[x]$ a výraz $\bmod k$ je nahrazen výrazem $\bmod (y-\alpha)$. Přitom místo $p(x,y) \bmod (y-\alpha)$ píšeme $p(x,\alpha)$.

Lemma 14.1. *Označme* $r = \mathrm{NSD}_{\mathbf{R}[x,y]}(p,q)$. *Pak*

$$\mathrm{NSD}_{\mathbf{R}[x]}(p(x,\alpha), q(x,\alpha)) = r(x,\alpha) \cdot \mathrm{NSD}_{\mathbf{R}[x]}\big(\frac{p}{r}(x,\alpha), \frac{q}{r}(x,\alpha)\big).$$

Důsledek 14.2. *Jestliže polynom* $y-\alpha$ *nedělí* $\mathrm{lc}(p)$ *ani* $\mathrm{lc}(q)$ *(neboli* $\mathrm{lc}(p)(\alpha), \mathrm{lc}(q)(\alpha) \neq 0$*), pak*

$$\deg \mathrm{NSD}_{\mathbf{R}[x]}(p(x,\alpha), q(x,\alpha)) \geq \deg \mathrm{NSD}_{\mathbf{R}[x,y]}(p,q).$$

Důsledek 14.3. *Jestliže* $y - \alpha$ *nedělí* $\mathrm{lc}(p)$ *ani* $\mathrm{lc}(q)$ *a jsou-li polynomy* $p(x,\alpha)$ *a* $q(x,\alpha)$ *nesoudělné v* $\mathbf{R}[x]$ *pro nějaké* $\alpha \in \mathbf{R}$*, pak jsou* p, q *nesoudělné v* $\mathbf{R}[x,y]$.

Analogicky definujeme použitelné, šťastné a smolné hodnoty α.

Definice. Řekneme, že hodnota $\alpha \in \mathbf{R}$ je *použitelná* pro polynomy p, q, pokud $\mathrm{lc}_x(p)(\alpha) \neq 0$ a $\mathrm{lc}_x(q)(\alpha) \neq 0$.

- Použitelná hodnota $\alpha \in \mathbf{R}$ se nazývá *šťastná* pro polynomy p, q, pokud

$$\deg \mathrm{NSD}_{\mathbf{R}[x]}(p(x,\alpha), q(x,\alpha)) = \deg_x \mathrm{NSD}_{\mathbf{R}[x,y]}(p,q).$$

- Použitelná hodnota $\alpha \in \mathbf{R}$ se nazývá *smolná* pro polynomy p, q, pokud

$$\deg \mathrm{NSD}_{\mathbf{R}[x]}(p(x,\alpha), q(x,\alpha)) > \deg_x \mathrm{NSD}_{\mathbf{R}[x,y]}(p,q).$$

I následující lemma se dokazuje zcela analogicky, jako v předchozí sekci.

Lemma 14.4. *Nechť* α *je použitelná smolná hodnota pro polynomy* p, q. *Pak*

$$(y-\alpha) \mid \mathrm{res}_x\left(\frac{p}{\mathrm{NSD}(p,q)}, \frac{q}{\mathrm{NSD}(p,q)}\right).$$

Důsledek 14.5. *Použitelných smolných hodnot pro dané polynomy* p, q *je jen konečně mnoho.*

Důkaz. Výraz $\mathrm{res}_x(\frac{p}{\mathrm{NSD}(p,q)}, \frac{q}{\mathrm{NSD}(p,q)})$ je polynomem v proměnné y nad $\mathbf{R}$. Je dělitelný $y - \alpha$ právě tehdy, když je α jeho kořenem. Každý polynom má jen konečně mnoho kořenů. □

Analogie Landau-Mignottovy meze je v případě polynomů více proměnných dokonce mnohem jednodušší, než předtím: nechť d značí největší stupeň mezi všemi koeficienty v p, q (připomínáme, že koeficienty jsou polynomy z $\mathbf{R}[y]$) a položme

$$\mathrm{LM}(p,q) = 1 + d.$$

Je zřejmé, že všechny koeficienty v jakémkoliv společném děliteli polynomů p, q budou mít stupeň nejvýše $\mathrm{LM}(p,q)$. Na jejich určení postačí $\mathrm{LM}(p,q) + 1$ šťastných hodnot α.

Algoritmus 14.6 (Modulární algorimus na NSD, polynomy více proměnných)**.**

```
VSTUP:   p, q ∈ R[x, y] primitivní.
VÝSTUP:  NSD(p, q).
1.  Nechť α_1 je dosud nepoužitá použitelná hodnota z R.
2.  Spočti r := NSD_{R[x]}(p(x, α_1), q(x, α_1)).
3.  R := r, i := 1.
4.  WHILE i ≤ LM(p, q) DO:
        Nechť α_{i+1} je dosud nepoužitá použitelná hodnota z R.
        Spočti r := NSD_{R[x]}(p(x, α_{i+1}), q(x, α_{i+1})).
        IF deg_x r < deg_x R THEN α_1 := α_{i+1}, GOTO 3.
        IF deg_x r = deg_x R THEN
            Zkombinuj pomocí ČVZ R, r, výsledek ulož do R, i := i + 1.
5.  IF R | p a R | q THEN RETURN pp(R) ELSE GOTO 1.
```

Důkaz správnosti tohoto algoritmu je opět zcela analogický verzi pro polynomy ze $\mathbb{Z}[x]$. Podobným způsobem se vyplatí zabudovat i uvedenou heuristiku.

Příklad. Nechť

$$p = 2x^2y^3 - xy^3 + x^3y^2 + 2x^4y - x^3y - 6xy + 3y + x^5 - 3x^2$$
$$q = 2xy^3 - y^3 - x^2y^2 + xy^2 - x^3y + 4xy - 2y + 2x^2$$

Uvažujme tyto polynomy jako prvka a) oboru $(\mathbb{Z}[x])[y]$, b) oboru $(\mathbb{Z}[y])[x]$. V obou případech je $\mathrm{LM}(p, q) = 3$. Následující tabulka ukazuje běh modulárního algoritmu v obou případech:

	$(\mathbb{Z}[x])[y]$		$(\mathbb{Z}[y])[x]$	
$\alpha = 1$	$x = 1$	$y + 1$	$y = 1$	$x^2 + 2x - 1$
$\alpha = -1$	$x = -1$	$-3y + 1$	$y = -1$	$x^2 - 2x + 1$
$\alpha = 2$	$x = 2$	$3y + 4$	$y = 2$	$x^2 + 4x - 2$
$\alpha = -2$	$x = -2$	$-5y + 4$	$y = -2$	$x^2 - 4x + 2$
$\mathrm{NSD}(p, q)$		$(2x - 1)y + x^2$		$x^2 + 2yx - y$

Na závěr poznamenejme, že kombinací Algoritmů 13.10 a 14.6 dostáváme algoritmus pro výpočet NSD v oboru $\mathbb{Z}[x_1, \dots, x_k]$. Dá se dokázat, že jeho časová složitost je $\mathcal{O}(n^{2k+1} \cdot (k \cdot \log n + c)^2)$ operací s ciframi v $\mathbb{Z}$, kde opět n značí větší ze stupňů p, q a c značí počet cifer největšího koeficientu.

Faktorizace

Faktorizace polynomu, neboli rozklad polynomu na součin ireducibilních činitelů, je následující problém: vstupem je polynom $f \in \mathbf{R}[x_1, \dots, x_k]$ (kde $\mathbf{R}$ je nějaký Gaussův obor), výstupem jsou po dvou neasociované ireducibilní polynomy $a_1, \dots, a_m \in \mathbf{R}[x_1, \dots, x_k]$ a čísla $n_1, \dots, n_m$ taková, že

$$f = a_1^{n_1} \cdot \ldots \cdot a_m^{n_m}.$$

Tyto polynomy a exponenty jsou určeny jednoznačně až na pořadí a asociovanost (neboť obory polynomů nad Gaussovým oborem jsou Gaussovy obory).

V celé kapitole budeme předpokládat, že polynom f je primitivní. Pokud ne, pak se rozkládá na součin $\mathrm{cont}(f) \cdot \mathrm{pp}(f)$ a problém faktorizace f se dělí na problém faktorizace $\mathrm{pp}(f)$ v $\mathbf{R}[x_1, \dots, x_k]$ a problém faktorizace $\mathrm{cont}(f)$ v $\mathbf{R}$. Tím druhým se zabývat nebudeme. (Nad tělesy je to problém triviální, nad $\mathbb{Z}$ naopak příliš obtížný a zapadá spíše do výpočetní teorie čísel než do počítačové algebry).

Většina pokročilých algoritmů na faktorizaci dále předpokládá, že je vstupní polynom f tzv. *bezčtvercový*, tedy že v jeho rozkladu jsou všechny exponenty $n_1 = \dots = n_m = 1$. V první sekci proto probereme algoritmus, který libovolný polynom rozloží na součin bezčtvercových (ne nutně ireducibilních), a to v čase $\mathcal{O}(n^3)$.

Pro faktorizaci polynomů nad konečnými tělesy se používá tzv. Berlekampův algoritmus, který se probírá na přednášce Konečná tělesa. Je časová složitost je $\mathcal{O}(n^3)$. Ten se využije v modulárním algoritmu pro faktorizaci polynomů nad celými čísly (tzv. Berlekamp-Henselův algoritmus), jehož složitost je sice teoreticky exponenciální (v nejhorším případě), ale většina algoritmu probíhá v čase polynomiálním a závěrečná fáze je neefektivní pouze ve speciálních případech; obecně tedy běží poměrně rychle. Pro faktorizaci polynomů nad celými čísly existuje i algoritmus s polynomiální složitostí v nejhorším případě, tzv. Lenstra-Lenstra-Lovászův algoritmus (jeho složitost je $\mathcal{O}(n^{12+\varepsilon})$), v praxi se však ukazuje pomalejší, než algoritmus Berlekamp-Henselův.

(Poznamenejme, že faktorizace v $\mathbb{Z}$ je ve složitostní třídě NP a dosud není znám deterministický polynomiální algoritmus. Tedy polynomy nad $\mathbb{Z}$ umíme rozkládat efektivně až na rozklad jejich obsahu.)

Rychlé algoritmy na faktorizaci polynomů mají dalekosáhlé aplikace, např. v algebraické geometrii. Pro tyto účely existují další speciální algoritmy pro polynomy nad algebraicky uzavřeným tělesem, nad algebraickými rozšířeními, atd., kterými se zabývat nebudeme.

15. Bezčtvercová faktorizace

Jak již bylo řečeno, většina pokročilých algoritmů na faktorizaci vyžaduje vstupní polynom f bezčtvercový. Předvedeme algoritmus, který libovolný polynom rozloží na součin bezčtvercových (ne nutně ireducibilních).

Definice. Polynom f se nazývá *bezčtvercový*, pokud v jeho ireducibilním rozkladu $f = a_1^{n_1} \cdot \ldots \cdot a_m^{n_m}$ není žádná vyšší mocnina, tj. pokud $n_1 = \dots = n_m = 1$.

Bezčtvercovým rozkladem polynomu f rozumíme po dvou nesoudělné bezčtvercové polynomy $g_1, \dots, g_k$ splňující

$$f = g_1 \cdot g_2^2 \cdot g_3^3 \cdot \dots \cdot g_k^k$$

(tedy g_i obsahuje právě ty ireducibilní činitele, které se v f vyskytují v i-té mocnině).

Příklad. V $\mathbb{Z}[x]$ je polynom $4(x-1)(x+1)$ bezčtvercový, ale $(x-1)^2$ není.

Nadále budeme uvažovat případ polynomů jedné proměnné nad tělesem $\mathbf{T}$. (Místo $\mathbb{Z}$ uvažujeme podílové těleso.) Případ polynomů více proměnných lze řešit analogicky, avšak technicky je o něco náročnější. Necháváme jej jako cvičení.

Věta 15.1. *Nechť* $\mathbf{T}$ *je těleso a* $0 \neq f \in \mathbf{T}[x]$. *Pak*

(1) f *je bezčtvercový právě tehdy, když* $\mathrm{NSD}(f, f') = 1$*;*

(2) *jestliže* $\mathrm{char}(\mathbf{T}) = 0$ *a* $f = \prod_{i=1}^k g_i^i$ *je bezčtvercový rozklad, pak*

$$\mathrm{NSD}(f, f') = \prod_{i=1}^k g_i^{i-1}.$$

Důkaz. (1) ($\Leftarrow$) Předpokládejme, že f není bezčtvercový. Pak $f = g^2 \cdot h$ pro nějaké $g, h \in \mathbf{T}[x]$, $\deg g > 0$ a po zderivování dostáváme $f' = 2gg'h + g^2h'$. Tedy g je společný dělitel f i f', spor.

($\Rightarrow$) Nechť $f = \prod_{i=1}^k a_i$ je ireducibilní rozklad polynomu f, tj. $a_1, \dots, a_k$ jsou po dvou nesoudělné ireducibilní polynomy. Pak

$$f' = a_1' \cdot a_2 \cdot \dots \cdot a_k + a_1 \cdot a_2' \cdot a_3 \cdot \dots \cdot a_k + \dots + a_1 \cdot \dots \cdot a_{k-1} \cdot a_k'.$$

Předpokládejme, že f a f' mají nějakého netriviálního společného dělitele. Můžeme navíc předpokládat, že je tento společný dělitel ireducibilní, a tedy je roven a_i pro nějaké i. Protože $a_i \mid f'$ a přitom se vyskytuje ve všech členech kromě i-tého, musí a_i dělit i i-tý člen. Protože $a_i \nmid a_j$ pro žádné $j \neq i$, musí a_i dělit a_i'. Jenže $\deg a_i' < \deg a_i$, a tedy $a_i' = 0$. V případě charakteristiky 0 to znamená, že je a_i konstantní, spor. V případě charakteristiky $p \neq 0$ může nastat ještě následující možnost: při derivování polynomu a_i' vznikají koeficienty, které jsou násobky p. To nastane, pokud jsou všechny nenulové termy a_i tvaru x^{pj}, pro nějaké j. Pak ale můžeme psát $a_i = g(x^p)$ pro nějaký polynom $g \in \mathbf{T}[x]$. Jenže v tělese charakteristiky p platí $g(x^p) = g(x)^p$ (použijte rovnost $(a+b)^p = a^p + b^p$), což je ve sporu s ireducibilitou polynomu a_i.

(2) Derivací bezčtvercového rozkladu f dostáváme

$$f' = \sum_{j=1}^k (g_j^j)' \cdot (\prod_{i \neq j} g_i^i) = \sum_{j=1}^k j \cdot g_j^{j-1} \cdot g_j' \cdot (\prod_{i \neq j} g_i^i).$$

Tedy bez ohledu na charakteristiku $\mathbf{T}$ platí, že $\prod_{i=1}^k g_i^{i-1}$ je společným dělitelem polynomů f a f'. Zbývá dokázat, že to je *největší* společný dělitel. Předpokládejme, že ne, tedy že existuje nekonstatní polynom g, který dělí polynomy

$$\frac{f}{\prod_{i=1}^k g_i^{i-1}} = \prod_{i=1}^k g_i \quad \text{i} \quad \frac{f'}{\prod_{i=1}^k g_i^{i-1}} = \sum_{i=1}^k j \cdot g_j' \prod_{i \neq j} g_i.$$

Opět můžeme předpokládat, že g je ireducibilní a tedy $g \mid g_m$ pro nějaké m. Podobně jako v (1), protože se g_m vyskytuje ve všech členech sumy $\sum_{i=1}^k j \cdot g_j' \prod_{i \neq j} g_i$ kromě

m-tého, musí g dělit mg'_m. Jenže g_m je bezčtvercové, tedy podle (1) jsou g_m a g'_m nesoudělné, spor. □

Poznámka. V důkazu bodu (2) jsme konstatovali, že polynom $\prod_{i=1}^k g_i^{i-1}$ je společným dělitelem polynomů f, f', a to v libovolné charakteristice. Tedy polynom $\frac{f}{\text{NSD}(f,f')}$ je dělitelem polynomu $\prod_{i=1}^k g_i$, a proto je bezčtvercový.

Na této větě jsou založeny algoritmy na bezčtvercovou faktorizaci: jednodušší a rychlejší pro případ charakteristiky 0, o něco složitější obecný.

Algoritmus 15.2 (Bezčtvercová faktorizace v charakteristice 0)**.**
`VSTUP:` $f \in \mathbf{T}[x]$, předpokládáme $\text{char}(\mathbf{T}) = 0$.
`VÝSTUP:` Bezčtvercový rozklad $g_1 \dots, g_k$ polynomu f.
`0.` $f_0 := f$.
`i.` $f_i := \text{NSD}(f_{i-1}, f'_{i-1})$ pro $i = 1, \dots, k$.
$h_i := \frac{f_{i-1}}{f_i}$.
`IF` $i \geq 2$ `THEN` $g_{i-1} := \frac{h_{i-1}}{h_i}$.
`IF` $f_i = 1$ `THEN RETURN` $g_1, \dots, g_{i-1}, g_i = f_{i-1}$, `STOP`.

Tvrzení 15.3. *Algoritmus 15.2 funguje.*

Důkaz. Indukcí snadno dokážeme, že $f_i = \prod_{j=i+1}^k g_j^{j-i}$: pro $i = 0$ to platí triviálně a v indukčním kroku použijeme Větu 15.1 (2), která říká, že se exponenty v bezčtvercovém rozkladu f_{i+1} zmenší oproti f_i o jedna. První činitel tedy zmizí a platí

$$h_i = \frac{\prod_{j=i}^k g_j^{j-i+1}}{\prod_{j=i+1}^k g_j^{j-i}} = g_i \cdot \ldots \cdot g_k.$$

Proto

$$\frac{h_{i-1}}{h_i} = \frac{g_{i-1} \cdot \ldots \cdot g_k}{g_i \cdot \ldots \cdot g_k} = g_{i-1}.$$

Algoritmus se zastaví přesně po k krocích, přičemž $f_k = 1$ a $f_{k-1} = g_k$. □

Příklad. Nechť

$$f = x^7 + x^6 - x^5 - x^4 - x^3 - x^2 + x + 1 \in \mathbb{Q}[x].$$

(Vidíme, že $f = (x^2 + 1)(x - 1)^2(x + 1)^3$.) Algoritmus 15.2 proběhne následovně:
`0.` $f_0 := f$
`1.` $f_1 := \text{NSD}(f_0, f'_0) = x^3 + x^2 - x - 1$.
$h_1 := \frac{f_0}{f_1} = x^4 - 1$.
`2.` $f_2 := \text{NSD}(f_1, f'_1) = x + 1$.
$h_2 := \frac{f_1}{f_2} = x^2 - 1$.
$g_1 := \frac{h_1}{h_2} = x^2 + 1$.
`3.` $f_3 := \text{NSD}(f_2, f'_2) = 1$.
$h_3 := \frac{f_2}{f_3} = x + 1$.
$g_2 := \frac{h_2}{h_3} = x - 1$.
Výsledek je tedy $g_1 = x^2 + 1$, $g_2 = x - 1$, $g_3 = f_2 = x + 1$ a platí $f = g_1 \cdot g_2^2 \cdot g_3^3$.

Tvrzení 15.4. *Časová složitost Algoritmu 15.2 je $\mathcal{O}(n^3)$, kde $n = \deg f$.*

Důkaz. Proběhne nejvýše n kroků, v každém z nich počítáme NSD a dvakrát dělíme, to vše s polynomy stupně nejvýše n. Každý krok má tedy složitost $\mathcal{O}(n^2)$ a celková složitost je $n \cdot \mathcal{O}(n^2)$. □

Algoritmus 15.5 (Bezčtvercová faktorizace v kladné charakteristice)**.**
`VSTUP:` $f \in \mathbf{T}[x]$, předpokládáme $\mathrm{char}(\mathbf{T}) = p \neq 0$.
`VÝSTUP:` Bezčtvercový rozklad $g_1 \dots, g_k$ polynomu f.
`1.` $d := \mathrm{NSD}(f, f')$.
`2.` `IF` $d = 1$ `THEN RETURN` f.
`3.` `IF` $d = f$ `THEN`
  najdi g splňující $f = g(x^p)$,
  spočti bezčtvercový rozklad $h_1, \dots, h_k$ polynomu g,
  `RETURN` $\underbrace{1, \dots, 1}_{p-1}, h_1, \underbrace{1, \dots, 1}_{p-1}, h_2, \underbrace{1, \dots, 1}_{p-1}, h_3, \dots, h_k$.
`4.` `IF` $1 \neq d \neq f$ `THEN`
  polož $g := \frac{f}{d}$,
  spočti bezčtvercový rozklad $h_1, \dots, h_k$ polynomu d, polož $h_{i+1} := 1$,
  pro $i = 1, \dots, k$ dělej
    spočti $u := \mathrm{NSD}(g, h_i)$ a polož $h_i := \frac{h_i}{u}$, $h_{i+1} := h_{i+1} \cdot u$, $g := \frac{g}{u}$,
  `RETURN` $h_1 \cdot g, h_2, \dots, h_{k+1}$.

Tvrzení 15.6. *Algoritmus 15.5 funguje.*

Důkaz. Algoritmus se zastaví, neboť každé rekurzivní volání má jako parametr polynom stupně menšího než původní vstup. Správnost plyne z následujícího komentáře k jednotlivým krokům.

(2) Jestliže je $\mathrm{NSD}(f, f') = 1$, pak je podle Věty 15.1 (1) polynom f bezčtvercový.

(3) Podmínka $d = f$ je splněna právě tehdy, když $f' = 0$, tedy když se všechny koeficienty při derivování vynulovaly. Podle argumentu uvedeného v důkaze Věty 15.1 existuje polynom g splňující $f = g(x^p)$. Je-li $h_1, \dots, h_k$ bezčtvercová faktorizace polynomu g, pak $f = g(x^p) = g^p = (\prod_i h_i^i)^p = \prod_i h_i^{ip}$, tedy bezčtvercový rozklad polynomu f je posloupnost, kde ip-tý člen je h_i a ostatní členy jsou 1.

(4) Je třeba zakomponovat polynom $g = \frac{f}{d}$ do bezčtvercového rozkladu polynomu d. Podle poznámky za Větou 15.1 je g bezčtvercový, stačí tedy ověřit jeho soudělnost s každým z bezčtvercových faktorů polynomu d. V případě soudělnosti pak je třeba přesunout $\mathrm{NSD}(g, h_i)$ z i-té do $(i+1)$-té složky bezčtvercového rozkladu. To, co z polynomu g zbyde (tj. to, co není soudělné s d), se přidá k první složce. □

Tvrzení 15.7. *Časová složitost Algoritmu 15.5 je $\mathcal{O}(n^4)$, kde $n = \deg f$.*

Důkaz. Algoritmus rekurzivně volá sám sebe nejvýše jednou v každém průběhu, a to na polynom stupně menšího, než je stupeň vstupního polynomu. Dochází tedy k nejvýše n opakováním. Spočítáme složitost jednoho průběhu.

Výpočet NSD v kroku `1.` má složitost $\mathcal{O}(n^2)$. Je-li splněna podmínka v kroku `2.`, žádné další operace se neprovádí. Je-li splněna podmínka v kroku `3.`, výpočet polynomu g má složitost $\mathcal{O}(n)$ a přepočet bezčtvercového rozkladu také $\mathcal{O}(n)$. Je-li splněna podmínka v kroku `4.`, je nutné testovat soudělnost $\frac{f}{d}$ s každým z polynomů h_i, tedy $\mathcal{O}(n)$-krát opakujeme operace se složitostí $\mathcal{O}(n^2)$.

Celkem tedy máme nejvýše n průběhů se složitostí $\mathcal{O}(n^3)$. □

Cvičení. Důkaz složitosti byl proveden poněkud „nahrubo", dostali jsme tedy složitost kvartickou. Dokažte, že ve skutečnosti má Algoritmus 15.5 časovou složitost $\mathcal{O}(n^3)$ *???aspoň doufám???*.

Příklad. Nechť
$$f = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \in \mathbb{Z}_2[x].$$
(Vidíme, že $f = (x+1)^7$.) Algoritmus 15.5 proběhne následovně:

(1) Nejprve spočteme $f' = x^6 + x^4 + x^2 + 1$ a vidíme, že $d = \mathrm{NSD}(f, f') = x^6 + x^4 + x^2 + 1$. Nastává tedy podmínka z kroku `4.`, provedeme rozklad d a zkombinujeme ho s $\frac{f}{d} = x + 1$.
(2) Nyní je tedy na vstupu polynom d. Protože $d' = 0$, pokračujeme krokem `3.`, vezmeme polynom $g = x^3 + x^2 + x + 1$, spočteme jeho bezčtvercový rozklad a poté ho budeme brát v druhé mocnině.
(3) Nyní je na vstupu polynom g. Spočteme $g' = x^2+1$, $\mathrm{NSD}(g, g') = x^2+1$, a tedy pokračujeme krokem `4.`, kde najdeme rozklad x^2+1 a zkombinujeme ho s $\frac{g}{\mathrm{NSD}(g,g')} = x + 1$.
(4) Nyní pokračujeme s polynomem $x^2 + 1$. Protože $(x^2 + 1)' = 0$, podle `3.` najdeme rozklad $x + 1$ a budeme ho brát v druhé mocnině.
(5) Polynom $x+1$ je v kroku `2.` prohlášen za bezčtvercový, nastává tedy zpětné vynořování z rekurze.

Při vynořování dostáváme postupně polynomy (5) $x+1$, (4) $(x+1)^2$, (3) $(x+1)^3$, (2) $(x+1)^6$, (1) $(x+1)^7$.

Příklad. Nechť
$$f = x^6 + x^5 + x^4 + x^3 + x^2 + x \in \mathbb{Z}_2[x].$$
Algoritmus 15.5 proběhne následovně:

(1) Nejprve spočteme $f' = x^4+x^2+1$ a vidíme, že $d = \mathrm{NSD}(f, f') = x^4+x^2+1$. Nastává tedy podmínka z kroku `4.`, provedeme rozklad d a zkombinujeme ho s $\frac{f}{d} = x^2 + x$.
(2) Nyní je tedy na vstupu polynom d. Protože $d' = 0$, pokračujeme krokem `3.`, vezmeme polynom $g = x^2 + x + 1$, spočteme jeho bezčtvercový rozklad a poté ho budeme brát v druhé mocnině.
(3) Nyní je na vstupu polynom g. Spočteme $g' = 1$, tedy g je podle kroku `2.` bezčtvercový a nastává vynořování z rekurze.

Při vynořování dostáváme postupně polynomy (3) x^2+x+1, (2) $(x^2+x+1)^2$, (1) $(x^2+x)(x^2+x+1)^2$.

Cvičení. Proveďte bezčtvercovou faktorizaci polynomu $x^8+2x^6+x^5+2x^3+x^2+2$ v $\mathbb{Z}_3[x]$.

Cvičení. Nechť $\mathbf{T}$ je těleso a $0 \neq f \in \mathbf{T}[x_1, \dots, x_k]$. Dokažte, že

(1) f je bezčtvercový právě tehdy, když $\mathrm{NSD}(f, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_k}) = 1$;
(2) jestliže $\mathrm{char}(\mathbf{T}) = 0$ a $f = \prod_{i=1}^k g_i^i$ je bezčtvercový rozklad, pak
$$\mathrm{NSD}(f, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_k}) = \prod_{i=1}^k g_i^{i-1}.$$

Na základě této věty navrhněte algoritmus pro bezčtvercovou faktorizaci polynomů více proměnných.

16. Faktorizace polynomů nad konečným tělesem

Nejběžnějším nástrojem je tzv. *Berlekampův algoritmus.* Vstupem je bezčtvercový polynom nad konečným tělesem, výstupem je jeho faktorizace. Jeho složitost je $\mathcal{O}(n^3)$. Veškeré detaily viz přednáška Konečná tělesa.

17. Faktorizace polynomů nad celými čísly

Zřejmě nejjednoduším rychlým algoritmem na faktorizaci polynomů v $\mathbb{Z}[x]$ je algoritmus *modulární*, který využívá Berlekampův algoritmus pro faktorizaci v $\mathbb{Z}_p[x]$ a Henselův algoritmus na „liftování" pro rekonstrukci řešení nad $\mathbb{Z}$. Nejprve si ukážeme, proč místo relativně komplikovaného liftování nelze jednoduše použít Čínskou větu o zbytcích podobně jako v kapitole o největším společném děliteli.

Uvažujme polynom
$$f = x^2 - 9x + 20 \in \mathbb{Z}[x].$$
Vidíme, že $f = (x-4)(x-5)$. Modulo 2 dostáváme
$$f \bmod 2 = x^2 + x = x(x+1),$$
modulo 3 dostáváme
$$f \bmod 3 = x^2 - 1 = (x+1)(x-1)$$
a modulo 5 dostáváme
$$f \bmod 5 = x^2 + x = x(x+1).$$
Tolik prvočísel by mělo stačit k rekonstrukci rozkladu. Jenže: které z těch dvou činitelů modulo jednotlivá p máme dát k sobě? Celkem bychom museli vyzkoušet všech osm kombinací (první první první, první první druhý, atd., druhý druhý druhý), na každou z nich aplikovat ČVZ a vyzkoušet, zda výsledek skutečně dělí zadaný polynom. Tento přístup má ovšem exponenciální složitost.

Henselův algoritmus tento problém obchází: na vstupu mu stačí rozklad modulo jediné (použitelné) prvočíslo p a on ho umí „naliftovat" na rozklad modulo p^k. Zvolíme-li k tak velké, abychom převýšili analogii Landau-Mignottovy meze, dostaneme *téměř* rozklad nad $\mathbb{Z}$.

Začneme jedním pomocným algoritmem, který potom využijeme v algoritmu Henselově.

Lemma 17.1. *Nechť* $\mathbf{T}$ *je těleso,* f, a, b *polynomy z* $\mathbf{T}[x]$ *a předpokládejme, že* a, b *jsou nesoudělné a* $\deg f < \deg a + \deg b$. *Pak existuje právě jedna dvojice polynomů* $u, v \in \mathbf{T}[x]$ *splňující*
$$f = u \cdot a + v \cdot b$$
taková, že $\deg u < \deg b$ *a* $\deg v < \deg a$.

Důkaz. Nejprve dokážeme jednoznačnost. Je-li
$$f = u_1 \cdot a + v_1 \cdot b = u_2 \cdot a + v_2 \cdot b$$
a $\deg u_1 < \deg b$, $\deg u_2 < \deg b$, $\deg v_1 < \deg a$, $\deg v_2 < \deg a$, pak
$$a(u_1 - u_2) = b(v_2 - v_1).$$
Protože a dělí levou stranu rovnosti, musí dělit také pravou stranu. Protože a, b jsou nesoudělné, musí a dělit $v_2 - v_1$. Ovšem $\deg(v_2 - v_1) < \deg a$, a tedy $v_2 - v_1 = 0$. Analogicky b dělí pravou stranu, tedy musí dělit i levou stranu a dostáváme $u_1 = u_2$.

Nyní dokážeme existenci. Protože jsou a, b nesoudělné, existují $\bar{u}, \bar{v} \in \mathbf{T}[x]$ splňující $\mathrm{NSD}(a, b) = 1 = \bar{u} \cdot a + \bar{v} \cdot b$ (viz rozšířený Eukleidův algoritmus). Pak $f = \bar{u}f \cdot a + \bar{v}f \cdot b$, ovšem nejspíš nebude splněna ta podmínka na stupně. Označme

$$u = \bar{u}f \bmod b \quad \text{a} \quad v = \bar{v}f + a \cdot (\bar{u}f \text{ div } b).$$

Ověříme, že $f = u \cdot a + v \cdot b$. Označme $q = \bar{u}f$ div b. Tedy $\bar{u}f = qb + u$, $v = \bar{v}f + aq$ a dostáváme

$$u \cdot a + v \cdot b = (\bar{u}f - qb) \cdot a + (\bar{v}f + aq) \cdot b = \bar{u}fa - qba + \bar{v}fb + aqb = \bar{u}fa + \bar{v}fb = f.$$

Zbývá ověřit, že $\deg u < \deg b$ a $\deg v < \deg a$. První nerovnost je zřejmá. Druhá plyne z rovnosti $f = u \cdot a + v \cdot b$, protože $\deg f < \deg a + \deg b$, $\deg u \cdot a = \deg u + \deg a < \deg b + \deg a$, a tedy i $\deg v \cdot b = \deg v + \deg b < \deg a + \deg b$. Proto $\deg v < \deg a$. □

Tvrzení 17.2. *Nechť $\mathbf{T}$ je těleso, $f, a_1, \dots, a_n$ polynomy z $\mathbf{T}[x]$ a předpokládejme, že $a_1, \dots, a_n$ jsou po dvou nesoudělné a $\deg f < \deg a_1 + \ldots + \deg a_n$. Označme $\tilde{a}_i = \prod_{j \neq i} a_j$. Pak existuje právě jedna n-tice polynomů $b_1, \dots, b_n \in \mathbf{T}[x]$ splňující*

$$f = \sum_{i=1}^{n} b_i \tilde{a}_i$$

taková, že pro všechna i je $\deg b_i < \deg a_i$.

Důkaz. Jednoznačnost se dokáže analogicky jako v předchozím lemmatu, detaily přenecháváme čtenáři jako cvičení.

K důkazu existence také využijeme přechozí lemma. Označme $a_i^* = a_{i+1} \cdot \ldots \cdot a_n$ a položme $d_0 = f$. Pro $i = 1, \dots, n-1$ zvolíme (pomocí toho lemmatu) d_i, b_i splňující

$$d_{i-1} = d_i a_i + b_i a_i^*, \qquad \deg d_i < \deg a_i^*, \deg b_i < \deg a_i$$

a na závěr položíme $b_n = d_{n-1}$. Podmínka $\deg b_i < \deg a_i$ je tedy splněna pro každé i a zbývá ověřit, že

$$f = \sum_{i=1}^{n} b_i \tilde{a}_i.$$

Pro všechna $i = 1, \dots, n-1$ platí

$$b_i \tilde{a}_i = a_1 \cdot \ldots \cdot a_{i-1} \cdot b_i a_i^* = a_1 \cdot \ldots \cdot a_{i-1} \cdot (d_{i-1} - d_i a_i).$$

Označíme-li $y_i = a_1 \cdot \ldots \cdot a_i \cdot d_i$, máme $b_i \tilde{a}_i = y_{i-1} - y_i$. Tedy

$$\begin{aligned}\sum_{i=1}^{n} b_i \tilde{a}_i &= \sum_{i=1}^{n-1} (y_{i-1} - y_i) + b_n \tilde{a}_n = (y_0 - y_1) + (y_1 - y_2) + \ldots + (y_{n-2} - y_{n-1}) + b_n \tilde{a}_n \\ &= y_0 - y_{n-1} + b_n \tilde{a}_n = d_0 - a_1 \cdot \ldots \cdot a_{n-1} \cdot d_{n-1} + \tilde{a}_n \cdot d_{n-1} = d_0 = f.\end{aligned}$$

□

Podíváme-li se na důkaz Tvrzení 17.2 a Lemmatu 17.1, můžeme ihned zformulovat algoritmus, který polynomy $b_1, \dots, b_n$ nalezne.

Algoritmus 17.3.
`VSTUP:` $f, a_1, \dots, a_n \in \mathbf{T}[x]$, $a_1, \dots, a_n$ po dvou nesoudělné, $\deg f < \sum_i \deg a_i$.
`VÝSTUP:` $b_1, \dots, b_n$ splňující $f = \sum_{i=1}^n b_i \tilde{a}_i$ a $\deg b_i < \deg a_i$ pro všechna i.
`1.` Polož $d := f$.
`2.` `FOR` $i = 1, \dots, n-1$ `DO`
najdi $\bar{u}, \bar{v}$ splňující $1 = \bar{u} \cdot a_i + \bar{v} \cdot a_i^*$ (rozšířený Eukleidův algoritmus),
$u := d \cdot \bar{u} \bmod a_i^*$,
$v := d \cdot \bar{v} + a_i \cdot (d \cdot \bar{u} \operatorname{div} a_i^*)$,
$b_i := v$,
$d := u$.
`3.` Polož $b_n := d$, `RETURN` $b_1, \dots, b_n$.

Tvrzení 17.4. *Časová složitost Algoritmu 17.3 je $\mathcal{O}(n \cdot m^2)$, kde $m = \sum_{i=1}^n \deg a_i$.*

Důkaz. Jediný náročný krok je `2.` Všimněte si, že všechny polynomy, které se vyskytují v cyklu, mají stupeň nejvýše m, tedy operace násobení a dělení i Eukleidův algoritmus proběhnou v čase $\mathcal{O}(m^2)$. Cyklus probíhá $(n-1)$-krát, celkem tedy $\mathcal{O}(n \cdot m^2)$. □

Příklad. Nechť $f = x^2 + 2x + 2$, $a_1 = 2x + 1$, $a_2 = x + 1$, $a_3 = x$ v $\mathbb{Z}_3[x]$. Označme d_i hodnotu proměnné d z i-tého cyklu.
`1.` $d_0 := x^2 + 2x + 2$.
`2.` $\underline{i = 1.}$ Má platit $d_0 = d_1 \cdot (2x + 1) + b_1 \cdot (x + 1)x$. Vidíme, že $d_1 = 2$ a $b_1 = 1$.
$\underline{i = 2.}$ Má platit $d_1 = d_2 \cdot (x + 1) + b_2 \cdot x$. Vidíme, že $d_2 = 2$ a $b_2 = 1$.
`3.` $b_3 := 2$.
Výstupem algoritmu je tedy $b_1 = 1$, $b_2 = 1$, $b_3 = 2$.

Nyní dokážeme klíčovou větu týkající se liftování rozkladu modulo p na rozklad modulo p^k. Její důkaz bude opět konstruktivní a na jeho základě zformulujeme Henselův algoritmus.

Věta 17.5 (Henselovo lemma). *Buď f primitivní bezčtvercový polynom v $\mathbb{Z}[x]$, p prvočíslo, které nedělí $\operatorname{lc}(f)$, a nechť $a_1, \dots, a_n \in \mathbb{Z}_p[x]$ jsou po dvou nesoudělné polynomy splňující*
$$f \equiv a_1 \cdot \ldots \cdot a_n \pmod{p},$$
přičemž $\operatorname{lc}(a_1) = \operatorname{lc}(f) \bmod p$ a $\operatorname{lc}(a_2) = \ldots = \operatorname{lc}(a_n) = 1$.
Pak pro všechna k přirozená existují $a_1^{(k)}, \dots, a_n^{(k)} \in \mathbb{Z}_{p^k}[x]$ splňující
$$f \equiv a_1^{(k)} \cdot \ldots \cdot a_n^{(k)} \pmod{p^k},$$
přičemž $\operatorname{lc}(a_1^{(k)}) = \operatorname{lc}(f) \bmod p^k$ a $\operatorname{lc}(a_2^{(k)}) = \ldots = \operatorname{lc}(a_n^{(k)}) = 1$ a pro všechna i platí
$$a_i^{(k)} \equiv a_i \pmod{p}.$$

Výpočtu polynomů $a_1^{(k)}, \dots, a_n^{(k)}$ se někdy říká *Henselovo liftování* rozkladu polynomu f. Ještě než se pustíme do důkazu, předvedeme na příkladě, co tato věta tvrdí. (K řešení nijak nevyužijeme postup, který bude popsán v důkazu.)

Příklad. Nechť $f = 5x^3 + 9x^2 - 146x - 120 \in \mathbb{Z}[x]$, $p = 3$ a uvažujme rozklad
$$f \equiv (2x + 1)(x + 1)x \pmod{3}.$$
Chceme tento rozklad modulo 3 naliftovat na rozklad modulo $3^2 = 9$. Hledáme tedy rozklad
$$f \equiv (5x + u)(x + v)(x + w) \pmod{9},$$

kde $u, v, w \in \mathbb{Z}_9$ jsou neznámé, přičemž má platit

$$\begin{aligned} 2x+1 &\equiv 5x+u \pmod 3 \\ x+1 &\equiv x+v \pmod 3 \\ x &\equiv x+w \pmod 3 \end{aligned}$$

Tedy

$$\begin{aligned} u &\equiv 1 \pmod 3 \\ v &\equiv 1 \pmod 3 \\ w &\equiv 0 \pmod 3 \end{aligned}$$

a možní kandidáti jsou $u, v = 1, 4, 7$ a $w = 0, 3, 6$. Řešením soustavy rovnic

$$\begin{aligned} uvw &\equiv -120 \equiv 6 \pmod 9 \\ uv+uw+5vw &\equiv -146 \equiv 7 \pmod 9 \\ u+5v+5w &\equiv 9 \equiv 0 \pmod 9 \end{aligned}$$

dostáváme $u = 4$, $v = 4$, $w = 6$, hledaným rozkladem je tedy

$$f \equiv (5x+4)(x+4)(x+6) \pmod 9.$$

Důkaz. Důkaz provedeme indukcí podle k. Pro $k = 1$ volíme $a_i^{(1)} = a_i$.

Předpokládejme tedy, že máme $a_1^{(k)}, \dots, a_n^{(k)}$ splňující podmínky věty, zkonstruujeme $a_1^{(k+1)}, \dots, a_n^{(k+1)}$. Existuje tedy $\tilde{d} \in \mathbb{Z}_p[x]$ takové, že

$$f - a_1^{(k)} \cdot \ldots \cdot a_n^{(k)} \equiv p^k \tilde{d} \pmod{p^{k+1}}.$$

Předefinujeme $a_1^{(k)}$ tak, že nahradíme jeho vedoucí koeficient za $\mathrm{lc}(f) \bmod p^{k+1}$. Pak dokonce existuje $d \in \mathbb{Z}_p[x]$ splňující

$$f - a_1^{(k)} \cdot \ldots \cdot a_n^{(k)} \equiv p^k d \pmod{p^{k+1}}$$

a navíc $\deg d < \deg f$, protože $\deg a_1^{(k)} \cdot \ldots \cdot a_n^{(k)} = \deg f$ a vedoucí členy obou polynomů jsou stejné. Položme

$$a_i^{(k+1)} = (a_i^{(k)} + p^k b_i) \bmod p^{k+1},$$

kde $b_i \in \mathbb{Z}_p[x]$ jsou polynomy splňující $\deg b_i < \deg a_i$ pro všechna i a

$$d \equiv \sum_{i=1}^{n} b_i \tilde{a}_i \pmod p,$$

kde $\tilde{a}_i = \prod_{j \neq i} a_j$. Existence takových b_i plyne z Tvrzení 17.2. Ověříme, že takto definovaná $a_i^{(k+1)}$ splňují podmínky věty, tj. že

(1) $f \equiv a_1^{(k+1)} \cdot \ldots \cdot a_n^{(k+1)} \pmod{p^{k+1}}$;
(2) $a_i^{(k+1)} \equiv a_i \pmod p$ pro všechna i;
(3) platí podmínky na vedoucí členy polynomů $a_1^{(k+1)}, \dots, a_n^{(k+1)}$.

Platí

$$\begin{aligned} a_1^{(k+1)} \cdot \ldots \cdot a_n^{(k+1)} &\equiv (a_1^{(k)} + p^k b_1) \cdot \ldots \cdot (a_n^{(k)} + p^k b_n) \\ &= \prod_{i=1}^{n} a_i^{(k)} + p^k \cdot (\sum_{i=1}^{n} b_i \tilde{a}_i^{(k)}) + (p^k)^2 \cdot (\text{zbytek}) \\ &\equiv \prod_{i=1}^{n} a_i^{(k)} + p^k \cdot (\sum_{i=1}^{n} b_i \tilde{a}_i^{(k)}) \pmod{p^{k+1}}. \end{aligned}$$

Protože $\prod_{i=1}^n a_i^{(k)} \equiv f - p^k d \pmod{p^{k+1}}$, dostáváme

$$a_1^{(k+1)} \cdot \ldots \cdot a_n^{(k+1)} \equiv f - p^k d + p^k \sum_{i=1}^n b_i \tilde{a}_i^{(k)} = f + p^k \Big(\sum_{i=1}^n b_i \tilde{a}_i^{(k)} - d\Big) \pmod{p^{k+1}}.$$

Přitom $\sum_{i=1}^n b_i \tilde{a}_i^{(k)} \equiv \sum_{i=1}^n b_i \tilde{a}_i \equiv d \pmod{p}$ (polynomy b_i jsme volili tak, aby to platilo), a tak dostáváme (1). K důkazu (2) stačí použít indukční předpoklad: protože $a_i^{(k)} \equiv a_i \pmod{p}$, dostáváme

$$a_i^{(k+1)} \equiv a_i^{(k)} + p^k b_i \equiv a_i^{(k)} \equiv a_i \pmod{p}.$$

A dále $\deg p^k b_i = \deg b_i < \deg a_i = \deg a_i^{(k)}$, a tedy $\mathrm{lc}(a_i^{(k+1)}) = \mathrm{lc}(a_i^{(k)})$, který jsme ovšem hned na začátku předefinovali tak, aby splňoval podmínky věty. □

Podle důkazu Henselova lemmatu můžeme ihned zformulovat algoritmus, který provádí Henselovo liftování.

Algoritmus 17.6 (Henselův)**.**
`VSTUP:` $f, a_1, \ldots, a_n$ jako ve Větě 17.5, $K \in \mathbb{N}$.
`VÝSTUP:` $a_1^{(K)}, \ldots, a_n^{(K)}$ jako ve Větě 17.5.
1. $\bar{a}_i := a_i$, $\tilde{a}_i := \prod_{j \neq i} a_j \bmod p$, pro všechna i.
2. `FOR` $k = 2, \ldots, K$ `DO`
   nahraď vedoucí člen $\bar{a}_1$ za $\mathrm{lc}(f) \bmod p^k$,
   spočti d splňující $p^{k-1} d \equiv f - \bar{a}_1 \cdot \ldots \cdot \bar{a}_n \pmod{p^k}$,
   spočti $b_1, \ldots, b_n$ splňující $d \equiv \sum_{i=1}^n b_i \tilde{a}_i \pmod{p}$ a $\deg b_i < \deg a_i$,
   polož $\bar{a}_i := \bar{a}_i + p^{k-1} b_i \bmod p^k$, pro všechna i.
3. `RETURN` $\bar{a}_1, \ldots, \bar{a}_n$.

Druhý řádek v cyklu znamená, že každý koeficient polynomu $f - \bar{a}_1 \cdot \ldots \cdot \bar{a}_n$ vydělíme číslem p^{k-1} a výsledek vezmeme modulo p^k. Z důkazu Henselova lemmatu plyne, že to lze. Na třetím řádku pak voláme Algoritmus 17.3 v tělese $\mathbb{Z}_p$.

Tvrzení 17.7. *Časová složitost Algoritmu 17.6 je $\mathcal{O}(K \cdot n \cdot m^2)$, kde $m = \deg f$.*

Důkaz. Cyklus je opakován $(K-1)$-krát. V každém kroku (1) násobíme $\bar{a}_1 \cdot \ldots \cdot \bar{a}_n$, tedy máme n násobení polynomů stupně nejvýše m se složitostí $n \cdot \mathcal{O}(m^2)$, (2) odečítáme $f - \bar{a}_1 \cdot \ldots \cdot \bar{a}_n$ v čase $\mathcal{O}(m)$, (3) dělíme výsledek p^{k-1} v čase $\mathcal{O}(m)$, (4) provádíme Algoritmus 17.3 se složitostí $\mathcal{O}(n \cdot m^2)$ a (5) počítáme nové $\bar{a}_i$ v čase $\mathcal{O}(m)$. □

Poznamenejme, že zaručeně $n \leq m$, a tedy složitost můžeme formulovat pouze v proměnných $m = \deg f$ a K jako $\mathcal{O}(K \cdot m^3)$. Existuje i verze Henselova algoritmu se složitostí $\mathcal{O}(\log K \cdot m^3)$: každý krok cyklu má podobnou složitost jako v naší verzi, ovšem tento krok spočte nikoliv $a_i^{(k+1)}$, nýbrž $a_i^{(pk)}$. Čili k se v každém kroku z-p-násobí a proto stačí $\log_p K$ kroků. Tato varianta je však technicky náročnější, zájemce odkazujeme na literaturu [].

Příklad. Nechť $f = 5x^3 + 9x^2 - 146x - 120 \in \mathbb{Z}[x]$, $p = 3$ a uvažujme rozklad

$$f \equiv (2x+1)(x+1)x \pmod{3}.$$

Chceme tento rozklad modulo 3 naliftovat na rozklad modulo $3^2 = 9$. Tentokrát použijeme Algoritmus 17.6.

Označme $a_1 = 2x+1$, $a_2 = x+1$, $a_3 = x$. Nejprve předefinujeme $a_1 = 5x+1$ a hledáme d splňující

$$3d \equiv f - a_1 \cdot a_2 \cdot a_3 \equiv (5x^3+7x+6)-(5x^3+6x^2+x) = 3x^2+6x+6 \pmod 9.$$

Tedy $d = x^2+2x+2$. Dále hledáme b_1, b_2, b_3 splňující

$$d \equiv b_1(x+1)x + b_2(5x+1)x + b_3(5x+1)(x+1) \pmod 3.$$

Provedením Algoritmu 17.3 získáme $b_1 = 1, b_2 = 1, b_3 = 2$. Tedy

$$a_1^{(2)} = a_1 + 3b_1 = 5x+4, \quad a_2^{(2)} = a_2 + 3b_2 = x+4, \quad a_3^{(2)} = a_3 + 3b_3 = x+6.$$

Nyní můžeme konečně formulovat modulární algoritmus na faktorizaci celočíselných polynomů.

Nejprve zvolíme vhodné prvočíslo p, a to tak, aby nedělilo vedoucí člen vstupního plynomu f, a tak, aby $f \bmod p$ byl bezčtvercový polynom v $\mathbb{Z}_p[x]$. Následně spočteme pomocí Berlekampova algoritmu rozklad polynomu $f \bmod p$ v $\mathbb{Z}_p[x]$. Tento rozklad dále naliftujeme pomocí Henselova algoritmu na rozklad modulo p^k pro jisté k. Jak toto k určit? Připomeňme Tvrzení 13.6, které říká, že jakýkoliv polynom $g = \sum_{i=0}^m b_i x^i$, který dělí polynom $f = \sum_{i=0}^n a_i x^i$ ($a_n, b_m \neq 0$), splňuje Landau-Mignottovu nerovnost

$$\sum_{i=0}^m |b_i| \leq 2^m \cdot \left|\frac{b_m}{a_n}\right| \cdot \sqrt{\sum_{i=0}^n a_i^2}.$$

Tedy, označíme-li

$$\mathrm{LM}(f) = 2^n \cdot \sqrt{\sum_{i=0}^n a_i^2},$$

pak všechny koeficienty jakéhokoliv členu ireducibilního rozkladu polynomu f v $\mathbb{Z}[x]$ jsou v absolutní hodnotě menší, než $\mathrm{LM}(f)$. Zvolíme-li tedy k tak, aby $p^k > 2 \cdot \mathrm{LM}(f)$, máme částečně vyhráno.

Máme-li štěstí, získali jsme rozklad f v $\mathbb{Z}[x]$ (v tom případě to bude rozklad ireducibilní). Může se ovšem stát (a obvykle tomu tak bude), že se f rozkládá modulo p na více činitelů, než v oboru $\mathbb{Z}[x]$. Potom některé složky rozkladu f modulo p^k vůbec nědělí f a je potřeba je nakombinovat do větších celků (viz následující příklad a poslední krok Algoritmu 17.8). Tato kombinační fáze má v nejhorším případě exponenciální složitost (je třeba vyzkoušet všechny možné kombinace faktorů) a teoreticky je tedy asymptoticky stejně špatná jako naivní přístup přes Čínskou větu o zbytcích. V praxi se však ukazuje, že tento přístup je *mnohem* rychlejší.

Příklad. Uvažujme polynom $f = x^2+x+2 \in \mathbb{Z}[x]$. Rozklad modulo $p=2$ je

$$f \bmod 2 = x^2 + x = x(x+1).$$

Liftováním dostaneme pro libovolné k rozklad tvaru

$$f \bmod 2^k = (x+u)(x+v)$$

pro jisté u sudé a v liché. Protože $\mathrm{LM}(f) = 4 \cdot \sqrt{6}$, zvolme $k=5$ a vychází

$$f \bmod 32 = (x+6)(x-5).$$

Ovšem polynom $x+6$ ani $x-5$ nedělí f, takže f musí být nerozložitelný.

Formální zápis algoritmu následuje. Podobně jako v kapitole o modulárním algoritmu na NSD chápeme polynom f mod p^k nejen jako prvek $\mathbb{Z}_{p^k}[x]$, ale také jako prvek $\mathbb{Z}[x]$ s tím, že koeficienty uvažujeme z intervalu $\{-\frac{p^k-1}{2}, \frac{p^k-1}{2}\}$ (resp. $\{-2^{k-1}+1, 2^{k-1}\}$ pro $p=2$).

Algoritmus 17.8 (Berlekamp-Henselův).
`VSTUP:` $f \in \mathbb{Z}[x]$ primitivní bezčtvercový.
`VÝSTUP:` Ireducibilní rozklad $a_1, \dots, a_z$ polynomu f.
`1.` Zvol prvočíslo p takové, že $p \nmid \mathrm{lc}(f)$ a f mod p je bezčtvercový v $\mathbb{Z}_p[x]$.
`2.` Spočti ireducibilní rozklad $b_1, \dots, b_l$ polynomu f mod p v $\mathbb{Z}_p[x]$,
a to tak, aby $\mathrm{lc}(b_1) = \mathrm{lc}(f)$ mod p a $\mathrm{lc}(b_2) = \ldots = \mathrm{lc}(b_l) = 1$.
`3.` Najdi nejmenší k takové, že $p^k > 2 \cdot \mathrm{LM}(f) \cdot |lc(f)|$.
`4.` Spočti $c_1, \dots, c_l \in \mathbb{Z}_{p^k}[x]$ splňující $f \equiv c_1 \cdot \ldots \cdot c_l \pmod{p^k}$,
$\mathrm{lc}(c_1) = \mathrm{lc}(f)$ mod p^k, $\mathrm{lc}(c_2) = \ldots = \mathrm{lc}(c_l) = 1$ a $c_i \equiv b_i \pmod{p}$ pro všechna i.
`5.` <u>Kombinace faktorů:</u>
$C := \{2, \dots, l\}$, $i := 0$, $m := 0$,
`WHILE` $m < |C|$ `DO`
  $m := m + 1$
  pro všechna $i_1, \dots, i_m \in C$ různá dělej
    $\tilde{g} := \mathrm{lc}(f) \cdot c_{i_1} \cdot \ldots \cdot c_{i_m}$ mod $p^k \in \mathbb{Z}[x]$,
    $g := \mathrm{pp}(\tilde{g})$,
    `IF` $g \mid f$ `THEN` $i := i + 1$, $a_i := g$, $f := \frac{f}{g}$, $C := C \setminus \{i_1, \dots, i_m\}$,
`RETURN` $a_1, \dots, a_i, f$.

Bezčtvercovost polynomu f mod p v kroku `1.` testujeme pomocí kritéria z Věty 15.1, tj. požadujeme $\mathrm{NSD}(f \bmod p, (f \bmod p)') = 1$ (takové p zaručeně existuje, stačí vzít libovolné prvočíslo větší než každý koeficient f a větší než $\deg f$).

V kroku `2.` použijeme Berlekampův algoritmus na polynom f mod p a výsledné polynomy znormujeme tak, aby platila podmínka na vedoucí členy.

V kroku `4.` použijeme Henselův algoritmus na polynomy $b_1, \dots, b_l$.

V kroku `5.` kombinujeme faktory $c_1, \dots, c_l$ do větších celků tak, aby dávaly dělitele polynomu f. Proměnné mají následující význam: C je množina indexů dosud volných faktorů, i je čítač faktorů polynomu f a m značí, že nyní zkoušíme kombinovat dohromady m faktorů. Cyklus probíhá pro m od jedné (nejprve testujeme každý zvlášť) až do té doby, dokud ještě v množině C zbývají nějaké neotestované indexy. V každém kroku se testuje součin všech možných m-tic činitelů c_i. Pro každý takový součin se otestuje, zda dělí zadaný polynom f (před tím je třeba upravit vedoucí koeficient: nejprve jej položíme maximální možný, tj. rovný $\mathrm{lc}(f)$, a z výsledku vezmeme primitivní část; zde využíváme předpokladu, že f je primitivní). Pokud jej dělí, pak jsme našli ireducibilního dělitele (ireducibilní v $\mathbb{Z}[x]$ je proto, že všechny menší kombinace jsme zkoušeli dříve). Příslušnou m-tici vymažeme z množiny C a její součin z rozkládaného polynomu f. To, co nám zbyde v množině C po doběhu cyklu, je potřeba zkombinovat s členem c_1.

Správnost Algoritmu 17.8 plyne z poznámek uvedených před a za algoritmem. Zbývá odhadnout jeho časovou složitost. Nechť $n = \deg f$. Výpočet provedeme zvlášť pro jednotlivé kroky, aby bylo dobře vidět, které kroky jsou z hlediska času kritické.

(1) V prvním kroku je nejnáročnější operací počítání NSD v oboru $\mathbb{Z}_p[x]$, což má složitost $\mathcal{O}(n^2)$; tuto operaci je potřeba eventuálně provést vícekrát, ale

ne více než n-krát, resp. tolikrát, kolik prvočíselných dělitelů má vedoucí člen f.

(2) V druhém kroku se volá Berlekampův algoritmus se složitostí $\mathcal{O}(n^3)$.

(3) Výpočet k ve třetím kroku má lineární složitost vzhledem k n. Všimněte si, že $k \leq 1 + \log_p(2^n\sqrt{(n+1)c^2}) = \mathcal{O}(n + \log c)$, kde c je absolutní hodnota největšího koeficientu v f.

(4) Henselův algoritmus proběhne v čase $\mathcal{O}(k \cdot n^3) = \mathcal{O}(n^4 + n^3 \log c)$.

(5) Nejhorší případ, který může nastat, je následující: f je ireducibilní v $\mathbb{Z}[x]$, avšak rozkládá se na součin lineárních činitelů v $\mathbb{Z}_p[x]$. Potom cyklus proběhne pro všechny možné podmnožiny množiny C, tedy 2^{n-1}-krát. Obecně má tento krok složitost $\mathcal{O}(2^l \cdot n^2)$, kde l je počet prvků rozkladu f mod p.

Kroky `1.-4.` mají tedy složitost polynomiální, krok `5.` exponenciální v nejhorším případě. Ovšem jak je vidět, popsaný nejhorší případ je dosti obskurní a čím méně dodatečných faktorů mod p vzniká, tím je tento krok rychlejší. V praxi se ukazuje, že obvykle mnoho nových faktorů mod p nevzniká.

Poznámka. Pro Berlekamp-Henselův algoritmus se nabízejí dvě účinné heuristiky:

- Díky analýze složitosti vidíme, že kritickým parametrem je počet prvků rozkladu polynomu f mod p získaného Berlekampovým algoritmem. Můžeme tedy zkusit více (přípustných) prvočísel p (tj. opakovat několikrát kroky `1., 2.`) a nakonec zvolit to p, pro které vyšel nejkratší rozklad.
- Jsou-li koeficienty polynomu f velké, lze použít podobnou heuristiku, jako u modulárního algoritmu pro NSD: zvolíme k menší, než vychází z Landau-Mignottovy meze, a pokud krok `5.` nevyprodukuje korektní rozklad (je třeba přidat závěrečný test i pro a_k), zopakujeme kroky `4., 5.` s větším k. Tato heuristika samozřejmě dává smysl pouze v případě, kdy je krok `5.` rychlejší, než případná další iterace Henselova algoritmu (tedy použitelná především pro polynomy malého stupně s obrovskými koeficienty).

Příklad. Faktorizujte v oboru $\mathbb{Z}[x]$ polynom

$$f = 6x^7 + 7x^6 + 4x^5 + x^4 + 6x^3 + 7x^2 + 4x + 1.$$

`1.` První přípustné prvočíslo je $p = 5$. Máme

$$f \bmod 5 = x^7 + 2x^6 - x^5 + x^4 + x^3 + 2x^2 - x + 1$$

a $(f \bmod 5)' = 2x^6 + 2x^5 - x^3 - 2x^2 - x - 1$, a tedy $\mathrm{NSD}(f \bmod 5, f' \bmod 5) = 1$.
`2.` Berlekampův algoritmus dává rozklad

$$f \equiv (x-2)(x^2-2)(x^2+2)(x^2-x+2) \pmod{5},$$

tedy $b_1 = x - 2$, $b_2 = x^2 - 2$, $b_3 = x^2 + 1$ a $b_4 = x^2 - x + 2$.
`3.` Pro jednoduchost zvolme $\mathrm{LM}(f) = 10$. Pak stačí $k = 2$.
`4.` Henselův algoritmus dává rozklad

$$f \equiv (6x+3)(x^2-7)(x^2+7)(x^2+9x-8) \pmod{25},$$

tedy $c_1 = 6x + 3$, $c_2 = x^2 - 7$, $c_3 = x^2 + 7$, $c_4 = x^2 + 9x - 8$.
`5.` Na začátku položme $C = \{2, 3, 4\}$. Následující tabulka znázorňuje průběh cyklem.

$$\begin{array}{lll}
m=1 & c_2 \rightsquigarrow & \tilde{g} = 6\cdot(x^2-7) \bmod 25 = 6x^2+8 \\
 & & g = 3x^2+4 \nmid f \\
 & c_3 \rightsquigarrow & \tilde{g} = 6(x^2+7) \bmod 25 = 6x^2-8 \\
 & & g = 3x^2-4 \nmid f \\
 & c_4 \rightsquigarrow & \tilde{g} = 6\cdot(x^2+9x-8) \bmod 25 = 6x^2+4x+2 \\
 & & g = 3x^2+2x+1 \mid f, \quad \text{tedy } a_1 := 3x^2+2x+1, \\
 & & \quad f := \frac{f}{a_1} = 2x^5+x^4+2x+1, C := \{2,3\} \\
m=2 & c_2, c_3 \rightsquigarrow & \tilde{g} = 2\cdot(x^2-7)(x^2+7) \bmod 25 = 2x^4+2 \\
 & & g = x^4+1 \mid f, \quad \text{tedy } a_2 := x^4+1, f := \frac{f}{a_2} = 2x+1, C := \emptyset
\end{array}$$

Cyklus se zastaví s $C = \emptyset$, posledním ireducibilním činitelem je tedy $a_3 := f = 2x+1$. Výsledkem úlohy je rozklad

$$f = (3x^2+2x+1)(x^4+1)(2x+1).$$

Poznámka. Algoritmus Lenstra-Lenstra-Lovászův (1982) faktorizuje polynomy nad $\mathbb{Z}$ v polynomiálním čase v nejhorším případě (konkrétně se složitostí $\mathcal{O}(n^{12})$, resp. $\mathcal{O}(n^{9+\varepsilon})$ při použití FFT), nicméně v praxi se ukazuje zpravidla pomalejší, než algoritmus Henselův. Na druhou stranu, některé myšlenky LLL algoritmu lze využít i pro řešení dalších problémů, např. na (relativně rychlou) faktorizaci čísel.

Cvičení. Polynom

$$f = x^4 + 1$$

má následující zajímavou vlastnost: je ireducibilní v oboru $\mathbb{Z}[x]$, avšak rozkládá se v každém $\mathbb{Z}_p[x]$ pro libovolné prvočíslo p. Dokažte!

18. Faktorizace polynomů více proměnných

Na faktorizaci polynomů více proměnných existuje řada algoritmů. Zmíníme se o třech:

- *Kroneckerův algoritmus* (1882) má sice v nejhorším případě exponenciální složitost, ale zato je poměrně jednoduchý; předvedeme jej se všemi detaily.
- *analogie Berlekamp-Henselova algoritmu* (1960???) má sice také exponenciální složitost, ale v praxi je celkem rychlá (určitě rychlejší než Kroneckerův algoritmus); předvedeme myšlenku, na které je založena, ale důkaz správnosti kvůli jeho technické náročnosti uvádět nebudeme.
- *Kaltofenův algoritmus* (1982) má teoreticky polynomiální složitost; v praxi je však relativně pomalý a více se o něj zajímat nebudeme.

Společné všem třem (??) metodám je, že problém faktorizace polynomů více proměnných je převáděn na faktorizaci polynomů v jedné proměnné nad týmž (Gaussovým) oborem.

18.1. **Kroneckerův algoritmus.** Principem Kroneckerova algoritmu je dosazení různých mocnin jedné (nové) proměnné za všechny proměnné zadaného polynomu. Definujme zobrazení

$$\begin{aligned} \varphi_d : \mathbf{R}[x_1,\dots,x_k] &\to \mathbf{R}[y] \\ f &\mapsto f(y, y^d, y^{d^2}, \dots, y^{d^{k-1}}). \end{aligned}$$

Všimněte si, že to je (dosazovací) homomorfismus okruhů.

Příklad. Nechť $f = x_1^2x_2 + x_1x_2^2 + x_1 + x_2 \in \mathbb{Z}[x_1,x_2]$ a položme $d = 3$. Za x_1 tedy dosazujeme y a za x_2 dosazujeme y^3. Dostáváme $\varphi_3(f) = y^5 + y^7 + y + y^3$.

Dále si všimněte, že zobrazení φ_d omezené na množinu

$$\{f \in \mathbf{R}[x_1, \dots, x_k] : \deg_{x_i} f < d \text{ pro všechna } i\}$$

je bijekcí. Jinými slovy, pokud $\deg_{x_i} f < d$ pro všechna i, pak lze f jednoznačně zrekonstruovat ze svého obrazu $g = \varphi_d(f)$. V tomto kontextu budeme používat značení $f = \varphi_d^{-1}(g)$.

Příklad. Najděte polynom $f \in \mathbb{Z}[x_1, x_2]$ takový, že $\deg_{x_1} f < 3$, $\deg_{x_2} f < 3$ a $\varphi_3(f) = y^7 + y^5 + y^3 + y$.
Jako stavební kameny pro $\varphi_3(f)$ máme y a y^3. Člen y^7 lze tedy napsat (jednoznačně) jako $y \cdot (y^3)^2$ a odpovídá členu $x_1 x_2^2$. Podobně, člen $y^5 = (y)^2 \cdot y^3$ odpovídá členu $x_1^2 \cdot x_2$, člen y^3 členu x_2 a člen y členu x_1. Dostáváme $f = x_1 x_2^2 + x_1^2 x_2 + x_2 + x_1$.

Lemma 18.1. *Nechť* $\mathbf{R}$ *je Gaussův obor,* $f, g \in \mathbf{R}[x_1, \dots, x_k]$ *a předpokládejme, že* $g \mid f$*. Dále, nechť* $d > \max_i \deg_{x_i} f$*. Pak*

$$g = \varphi_d^{-1}(u)$$

pro nějaký dělitel u *polynomu* $\varphi_d(f)$*.*

Důkaz. Napišme $f = g \cdot h$ pro nějaký polynom h. Protože je φ_d homomorfismus, platí $\varphi_d(f) = \varphi_d(g) \cdot \varphi_d(h)$, a tedy $\varphi_d(g) \mid \varphi_d(f)$. Můžeme zvolit $u = \varphi_d(g)$. □

Lemma tedy říká, že každý dělitel polynomu f lze zrekonstruovat z některého dělitele polynomu $\varphi_d(f)$. Opačná implikace však neplatí, ve smyslu, že ne každý dělitel polynomu $\varphi_d(f)$ se invertuje na dělitele polynomu f.

Příklad. V předchozím příkladě $y \mid \varphi_3(f)$, avšak $x_1 = \varphi_3^{-1}(y) \nmid f$.

Na této myšlence je založen Kroneckerův faktorizační algoritmus. Polynom f převedeme pomocí zobrazení φ_d (pro dostatečně velké d) na polynom, který rozložíme pomocí nějakého algoritmu na faktorizaci v jedné proměnné. Vhodnou kombinací ireducibilních činitelů polynomu $\varphi_d(f)$ pak získáme ireducibilní faktory polynomu f, podobně jako v kroku `5.` Berlekamp-Henselova algoritmu.

Algoritmus 18.2 (Kroneckerův)**.**
`VSTUP:` $f \in \mathbf{R}[x_1, \dots, x_k]$.
`VÝSTUP:` Ireducibilní rozklad $a_1, \dots, a_l$ polynomu f.
`1.` Polož $d := 1 + \max_{i=1,\dots,k} \deg_{x_i} f$.
`2.` Najdi faktorizaci $g_1, \dots, g_r$ polynomu $\varphi_d(f)$ v oboru $\mathbf{R}[y]$.
`3.` Kombinace faktorů:
$C := \{1, \dots, r\}$, $m := 0$, $i := 0$,
`WHILE` $C \neq \emptyset$ `DO`
  $m := m + 1$,
  pro všechna $i_1, \dots, i_m \in C$ různá dělej
    $g := \varphi_d^{-1}(g_{i_1} \cdot \dots \cdot g_{i_m})$,
    jestliže $g \mid f$ pak $i := i + 1$, $a_i := g$, $f := \frac{f}{g}$, $C := C \setminus \{i_1, \dots, i_m\}$.
`RETURN` $a_1, \dots, a_i$.

Správnost algoritmu plyne z Lemmatu 18.1, neboť každý ireducibilní činitel f lze nakombinovat z φ_d^{-1}-obrazu některých ireducibilních činitelů polynomu $\varphi_d(f)$. Ireducibilita je v algoritmu zajištěna tím, že hledání probíhá kombinací všech m-tic pro postupně se zvyšující m.

Příklad. Pomocí Algoritmu 18.2 najděte rozklad polynomu
$$f = x_1^2x_2 + x_1x_2^2 + x_1 + x_2 \in \mathbb{Z}[x_1, x_2].$$
Zvolíme $d = 3$ a najdeme rozklad
$$\varphi_3(f) = y^7 + y^5 + y^3 + y = y \cdot (y^2 + 1) \cdot (y^4 + 1).$$
Tedy $g_1 = y$, $g_2 = y^2 + 1$ a $g_3 = y^4 + 1$. Tabulka ukazuje průběh kroku `3.`

$$\begin{array}{lll}
m = 1 & g_1 \rightsquigarrow & \varphi_d^{-1}(g_1) = x_1 \nmid f \\
 & g_2 \rightsquigarrow & \varphi_d^{-1}(g_2) = x_1^2 + 1 \nmid f \\
 & g_3 \rightsquigarrow & \varphi_d^{-1}(g_3) = x_1x_2 + 1 \mid f \\
 & & \quad \text{tedy } a_1 = x_1x_2 + 1, f := \frac{f}{a_1} = x_1 + x_2, C := \{1, 2\} \\
m = 2 & g_1, g_2 \rightsquigarrow & \varphi_d^{-1}(g_1g_2) = \varphi_d^{-1}(y^3 + y) = x_1 + x_2 \mid f \\
 & & \quad \text{tedy } a_2 = x_1 + x_2, f := 1, C := \emptyset
\end{array}$$

Výsledkem úlohy je rozklad $f = (x_1x_2 + 1)(x_1 + x_2)$.

Na závěr prodiskutujeme časovou složitost Algoritmu 18.2. Předpokládejme, že $\mathbf{R}$ je konečné těleso nebo $\mathbb{Z}$. V tomto případě mají první dva kroky složitost polynomiální. Poznamenejme, že je-li $n = \deg f$, pak hodnota d může být až $n + 1$, a tedy $\deg \varphi_d(f)$ může být až $(n+1)^k - 1$. V případě konečného tělesa tak dostáváme pro časovou složitost kroku `2.` odhad $\mathcal{O}((n^k)^3) = \mathcal{O}(n^{3k})$ a v $\mathbb{Z}$ (při použití algoritmu Lenstra-Lenstra-Lovász) odhad $\mathcal{O}(n^{9k+\varepsilon})$.

Složitost kroku `3.` je v nejhorším případě $\mathcal{O}(2^r)$. Tedy v krajním (avšak extrémně řídkém) případě, kdy je polynom f ireducibilní, avšak $\varphi_d(f)$ se rozkládá na $\mathcal{O}(n^k)$ činitelů, máme složitost $\mathcal{O}(2^{n^k})$. V praxi je však, podobně jako u Berlekamp-Henselova algoritmu, tento krok mnohem rychlejší, než teoretický odhad napovídá.

18.2. **Analogie Berlekamp-Henselova algoritmu.** Principem tohoto algoritmu je dosazení nějakých hodnot za všechny proměnné zadaného polynomu kromě jedné vybrané. Algoritmus je založen na zobecnění Henselova lemmatu, tzv. Větě o liftování.

Věta 18.3 (o liftování). *Nechť $\mathbf{R}$ je obor integrity, $\mathbf{I}$ konečně generovaný ideál v $\mathbf{R}$, $f_1, \dots, f_n$ polynomy z $\mathbf{R}[x_1, \dots, x_k]$ a $a_1, \dots, a_k \in \mathbf{R}$ takové, že*
$$f_1(a_1, \dots, a_k) \in \mathbf{I}, \dots, f_n(a_1, \dots, a_k) \in \mathbf{I}.$$
Označme $U = (u_{ij})$ matici $n \times k$, kde
$$u_{ij} = \frac{\partial f_i}{\partial x_j}(a_1, \dots, a_k)$$
(tedy Jakobián *polynomů $f_1, \dots, f_n$ v bodě $(a_1, \dots, a_k)$) a předpokládejme, že je matice U zprava invertibilní modulo $\mathbf{I}$ (tj. existuje matice W typu $k \times n$ splňující $U \cdot W \equiv E_n \pmod{\mathbf{I}}$).*
Pak pro všechna t přirozená existují $a_1^{(t)}, \dots, a_k^{(t)} \in \mathbf{R}$ splňující
$$f_1(a_1^{(t)}, \dots, a_k^{(t)}) \in \mathbf{I}^t, \dots, f_n(a_1^{(t)}, \dots, a_k^{(t)}) \in \mathbf{I}^t$$
a zároveň
$$a_j^{(t)} \equiv a_j \pmod{\mathbf{I}}.$$
Zde $\mathbf{I}^t$ značí ideál $\underbrace{\mathbf{I} \cdot \ldots \cdot \mathbf{I}}_{t} = \{i_1 \cdot \ldots \cdot i_t : i_1, \dots, i_t \in \mathbf{I}\}$.

Tuto větu dokazovat nebudeme, ale ukážeme, jak z ní plyne Henselovo lemma a další zajímavé důsledky.

Henselovo lemma. Nechť $f \in \mathbb{Z}[x]$ a $a_1, \dots, a_k \in \mathbb{Z}[x]$ splňují $f \equiv a_1 \cdot \ldots \cdot a_k \pmod p$. Položíme-li

$$\mathbf{R} = \mathbb{Z}[x],\ \mathbf{I} = p\mathbb{Z}[x],\ n = 1 \ \text{ a } \ f_1 = x_1 \cdot \ldots \cdot x_m - f$$

(polynom f_1 rozumíme jako prvek oboru $\mathbf{R}[x_1, \dots, x_m]$, člen $f \in \mathbf{R} = \mathbb{Z}[x]$ je tedy *konstantní člen* tohoto polynomu!), pak dostáváme Henselovo lemma (s vyjímkou podmínky na vedoucí koeficienty). Podmínka $f_1(a_1, \dots, a_m) \in \mathbf{I}$ se překládá jako $a_1 \cdot \ldots \cdot a_m - f \in p\mathbb{Z}[x]$, čili $f \equiv a_1 \cdot \ldots \cdot a_m \pmod p$. A ideál $\mathbf{I}^t$ sestává ze součinů t polynomů, z nichž každý je dělitelný p, tedy je roven ideálu $p^t\mathbb{Z}[x]$.

Zbývá ověřit, že Jakobián polynomu f_1 je zprava invertibilní modulo $\mathbf{I} = p\mathbb{Z}[x]$. Protože ale $U = (\frac{\partial f_1}{\partial x_1}(a_1, \dots, a_m), \dots, \frac{\partial f_1}{\partial x_m}(a_1, \dots, a_m)) = (\tilde{a}_1, \dots, \tilde{a}_m)$, podle Tvrzení 17.3 existuje $W = (b_1, \dots, b_m)^\top$ splňující $\sum_{i=1}^m \tilde{a}_i b_i \equiv 1 \pmod p$.

Faktorizace polynomů více proměnných. Nechť $f \in \mathbf{R}[x, y, z, \dots]$. Uvažujeme-li obor $\mathbf{R}[x, y, z, \dots]$ jako obor $\mathbf{R}$ ve Větě o liftování a položíme-li

$$\mathbf{I} = \langle y - \alpha, z - \beta, \dots \rangle,\ n = 1 \ \text{ a } \ f_1 = x_1 \cdot \ldots \cdot x_m - f$$

(opět, f je konstatním členem polynomu f_1), dostáváme analogii Henselova lemmatu. Zhruba řečeno, daný rozklad polynomu f modulo $\mathbf{I}$ (tj. jako polynomu jedné proměnné po dosazení $y = \alpha, z = \beta, \dots$) můžeme liftovat na rozklad modulo ideál $\mathbf{I}^t$ a pro dostatečně velké t dostaneme rozklad, z nějž lze kombinováním členů zrekonstruovat ireducibilní činitele polynomu f. Výsledkem je následující analogie Berlekamp-Henselova algoritmu:

Algoritmus 18.4.
`VSTUP:` $f \in \mathbf{R}[x, y, z, \dots]$.
`VÝSTUP:` Ireducibilní rozklad $a_1, \dots, a_l$ polynomu f.
`1.` Zvol $\alpha, \beta, \dots \in \mathbf{R}$ a $g = f(x, \alpha, \beta, \dots)$ takové, že
$\deg_x f = \deg g$ a polynom g je bezčtvercový v $\mathbf{R}[x]$.
`2.` Spočti ireducibilní rozklad $b_1, \dots, b_m$ polynomu g v $\mathbf{R}[x]$.
`3.` Nechť t je větší než stupeň všech koeficientů f vzhledem k proměnné x.
`4.` Spočti $c_1, \dots, c_m$ splňující
$f \equiv c_1 \cdot \ldots \cdot c_m \pmod{\mathbf{I}^t}$ a $c_i \equiv b_i \pmod{\mathbf{I}}$ pro všechna i.
`5.` Zkombinuj ireducibilní činitele polynomu f.

Krok `5.` má opět v nejhorším případě exponenciální složitost, ale v praxi běží rychleji, než Kroneckerův algoritmus.

Soustavy polynomiálních rovnic. Uvažujme situaci $\mathbf{R} = \mathbb{Z}$, $\mathbf{I} = p\mathbb{Z}$ a nechť $f_1, \dots, f_n$ jsou nějaké polynomy ze $\mathbb{Z}[x_1, \dots, x_k]$. Zajímá nás řešení soustavy rovnic

$$\begin{aligned} &f_1(x_1, \dots, x_k) = 0 \\ &\dots \\ &f_n(x_1, \dots, x_k) = 0 \end{aligned}$$

Věta o liftování říká, že kdykoliv najdeme nějaké řešení $(a_1, \dots, a_k)$ modulo prvočíslo p, tj. kdykoliv

$$f_1(a_1, \dots, a_k) \equiv 0 \pmod p, \dots, f_n(a_1, \dots, a_k) \equiv 0 \pmod p,$$

pak existuje pro každé t řešení $(a_1^{(t)}, \dots, a_r^{(t)})$ modulo p^t, tj.

$$f_1(a_1^{(t)}, \dots, a_k^{(t)}) \equiv 0 \pmod{p^t}, \dots, f_n(a_1^{(t)}, \dots, a_k^{(t)}) \equiv 0 \pmod{p^t}.$$

Navíc, algoritmická forma (kterou jsme neuváděli) této věty jej umožňuje efektivně nalézt.

Najít celočíselné řešení dané soustavy polynomiálních rovnic může být problém. Ovšem najít řešení modulo nějaké malé p je relativně snadné. Někdy se stane, že liftováním řešení modulo p na řešení modulo p^t pro dostatečně velké t získáme řešení v $\mathbb{Z}$. Někdy ne. Předvedeme obě situace na příkladě.

Příklad. Řešte soustavu

$$x_1 x_2 - x_2^2 - 10 = 0,$$
$$x_1^2 - 4x_1 x_2 + x_1 = 0.$$

Snadno nalezneme řešení $a_1 = 1$, $a_2 = -1$ modulo 3. Přitom $U = \left(\begin{smallmatrix} -1 & 3 \\ 7 & -4 \end{smallmatrix}\right)$, tedy U mod 3 je invertibilní nad $\mathbb{Z}_3$. Použitím Věty o liftování spočteme řešení mod 9. Vyjde nám $a_1^{(2)} = 7$, $a_2^{(2)} = 2$ a není těžké ověřit, že jde o řešení v $\mathbb{Z}$.

Příklad. Řešte rovnici

$$x_1 x_2 - x_2^2 - 10 = 0.$$

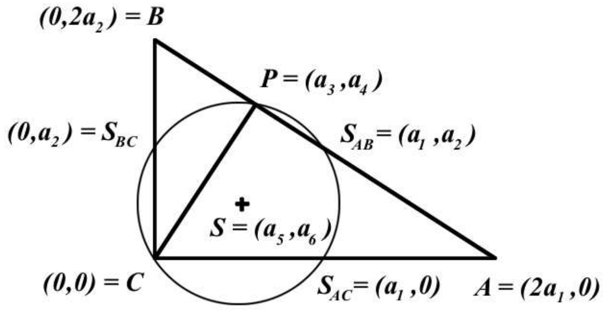
Opět vezmeme řešení $a_1 = 1, a_2 = -1$ modulo 3. Přitom $U = (-1, 3)$, tedy $U \cdot (-1, 0)^\top \equiv 1 \pmod 3$. Ovšem pro žádné t nevznikne liftováním řešení nad celými čísly (konkrétně, pro $t = 2$ dostáváme $a_1^{(2)} = -2$, $a_2^{(2)} = 2$, pro $t = 3$ dostáváme $a_1^{(3)} = 20$, $a_2^{(3)} = -20$, atd.)

Gröbnerovy báze

19. Úvod

Gröbnerovy báze jsou nástrojem pro algoritmickou práci s okruhy polynomů více proměnných. Mají řadu aplikací; některé předvedeme v závěru této kapitoly, mezi těmi ostatními zmiňme např. počítání soustav polynomiálních rovnic, počítání NSD, útoky proti některým typům kryptosystémů, byly i pokusy založit kryptosystémy na metodě Gröbnerových bází. Jako úvodní motivaci ukážeme, jak mohou Gröbnerovy báze pomoci při (automatickém) řešení úloh z klasické středoškolské geometrie.

Úloha. *Dokažte, že v pravoúhlém trojúhelníku leží vrchol C, středy stran AB, AC, BC a pata výšky na stranu AB na kružnici.*



Tuto úlohu lze jistě snadno řešit elementárně: v pravoúhlém trojúhelníku tvoří body $CS_{BC}S_{AB}S_{AC}$ obdélník, a tedy existuje kružnice k, která ho opíše. Uvažujme k jako Thaletovu kružnici nad úhlopříčkou CS_{AB}. Protože je úhel CPS_{AB} pravý, leží bod P na k také.

Podobné úvahy je vcelku snadné naučit člověka, nikoliv však počítač. Předvedeme alternativní způsob, založený na metodě Gröbnerových bází, který umožňuje řešit podobné úlohy algoritmicky.

Úlohu budeme řešit pomocí analytické geometrie; souřadnice zavedeme jako na obrázku. Ze zadání úlohy víme, že

- výška CP je kolmá na stranu AB, neboli vektor (a_3, a_4) je kolmý na vektor $(a_1, -a_2)$, tedy platí rovnice

$$a_1a_3 - a_2a_4 = 0. \tag{1}$$

- bod P náleží straně AB, a protože přímku AB můžeme popsat rovnicí $a_2x + a_1y - 2a_1a_2 = 0$, dostáváme rovnici

$$a_2a_3 + a_1a_4 - 2a_1a_2 = 0. \tag{2}$$

- S je střed kružnice, která prochází body $C, S_{AB}, S_{AC}, S_{BC}$ (počítač by samozřejmě „neviděl”, že všechny čtyři body leží na kružnici, museli bychom jej nechat tento fakt dokázat; pro jednoduchost se soustředíme na problém bodu P). Protože $|SC| = |SS_{AB}| = |SS_{AC}| = |SS_{BC}|$, máme pro souřadnice bodu S rovnice

$$a_5^2 + a_6^2 = (a_5 - a_1)^2 + (a_6 - a_2)^2 = (a_5 - a_1)^2 + a_6^2 = a_5^2 + (a_6 - a_2)^2 \tag{3}$$

Potřebujeme dokázat, že $|SP|$ je rovno poloměru kružnice k, tj. že

$$(a_5 - a_3)^2 + (a_6 - a_4)^2 = a_5^2 + a_6^2. \tag{4}$$

Shrnuto, chceme dokázat, že kdykoliv nějaká šestice čísel $(a_1, \dots, a_6) \in \mathbb{R}^6$ splňuje rovnice (1),(2),(3), pak splňuje také rovnici (4). (Přesněji řečeno, uvažujeme jen taková $(a_1, \dots, a_6)$, že úloha dává smysl, tj. body A, B, C skutečně tvoří trojúhelník.) Všimněte si, že každou z uvedených rovnic lze napsat ve tvaru $f(a_1, \dots, a_6) = 0$, kde f je polynom z $\mathbb{Q}[x_1, \dots, x_6]$; označme polynomy z rovnic (1),(2),(3) $f_1, \dots, f_5$ a polynom z rovnice (4) g.

Definice. Nechť $\mathbf{T}$ je těleso, $a_1, \dots, a_k \in \mathbf{T}$ a $f_1, \dots, f_\ell \in \mathbf{T}[x_1, \dots, x_k]$. Definujme množiny

$$\mathrm{I}(a_1, \dots, a_k) = \{f \in \mathbf{T}[x_1, \dots, x_k] : f(a_1, \dots, a_k) = 0\},$$
$$\mathrm{V}(f_1, \dots, f_\ell) = \{(a_1, \dots, a_k) \in \mathbf{T}^k : f_i(a_1, \dots, a_k) = 0 \text{ pro všechna } i = 1, \dots, \ell\}.$$

Pozorování 19.1. *Množina* $\mathrm{I}(a_1, \dots, a_k)$ *tvoří ideál v oboru* $\mathbf{T}[x_1, \dots, x_k]$.

Důkaz. Leží-li polynomy p, q v množině $\mathrm{I}(a_1, \dots, a_k)$, pak $p+q$ leží také v $\mathrm{I}(a_1, \dots, a_k)$, neboť $p(a_1, \dots, a_k) + q(a_1, \dots, a_k) = 0 + 0 = 0$. Je-li r libovolný polynom, pak $p \cdot r \in \mathrm{I}(a_1, \dots, a_k)$, protože $p(a_1, \dots, a_k) \cdot r(a_1, \dots, a_k) = 0 \cdot r(a_1, \dots, a_k) = 0$. □

Řečeno právě zavedenou terminologií, chceme dokázat, že kdykoliv (netriviální) $(a_1, \dots, a_6)$ náleží $\mathrm{V}(f_1, \dots, f_5)$, pak náleží také $\mathrm{V}(g)$. Ekvivalentně, že kdykoliv polynomy $f_1, \dots, f_5$ náleží nějakému $\mathrm{I}(a_1, \dots, a_6)$, pak polynom g náleží $\mathrm{I}(a_1, \dots, a_6)$.

Základní úloha, kterou řeší metoda Gröbnerových bází, je tzv. problém *náležení ideálu*: rozhodnout, zda daný polynom f náleží danému ideálu. Speciálně, umí rozhodnout, zda pro *každý* ideál $\mathbf{I}$ platí, že kdykoliv $f_1, \dots, f_m \in \mathbf{I}$, pak $g \in \mathbf{I}$ (tj. g patří do ideálu generovaného polynomy $f_1, \dots, f_m$). Ne každý ideál je sice tvaru $\mathrm{I}(a_1, \dots, a_k)$, avšak přesto je možné tento algoritmus aplikovat v případě uvedené geometrické úlohy. Je-li odpověď kladná, úloha je dokázána. Je-li záporná, nemůžeme říci nic. (Situaci jsme zde poněkud zjednodušili, detaily jsou uvedeny v poslední sekci.)

20. Báze ideálu

Ve této sekci se budeme věnovat teoretickému zázemí metody Gröbnerových bází.

Ideály v oborech $\mathbf{T}[x_1, \dots, x_k]$ jsou většinou nekonečné. Abychom je mohli předložit počítači, potřebujeme nějaký konečný popis.

Definice. Řekneme, že prvky $a_1, \dots, a_n \in \mathbf{R}$ tvoří (konečnou) *bázi ideálu* $\mathbf{I}$ v okruhu $\mathbf{R}$, pokud je $\mathbf{I}$ nejmenším ideálem okruhu $\mathbf{R}$ obsahujícím tyto prvky. Budeme značit $\mathbf{I} = \langle a_1, \dots, a_n \rangle$. (V této kapitole tedy opouštíme předchozí značení, kde závorky $\langle\ \rangle$ značily *podokruhy* generované danými prvky.)

Připomeňme následující pozorování ze základního kurzu:

Lemma 20.1. *Nechť* $\mathbf{R}$ *je komutativní okruh a* $a_1, \dots, a_n$ *báze ideálu* $\mathbf{I}$. *Pak*

$$\mathbf{I} = \left\{ \sum_{i=1}^{n} r_i a_i : r_i \in \mathbf{R} \right\}.$$

Důkaz. Inkluze ($\supseteq$) plyne z uzavřenosti ideálu na operace $+, \cdot$. Stačí tedy dokázat, že pravá strana výrazu tvoří ideál: $\sum r_i a_i \pm \sum q_i a_i = \sum (r_i \pm q_i) a_i$ a $r \cdot \sum r_i a_i = \sum (r r_i) a_i$. $\square$

Konečná báze by tedy byla dobrým popisem ideálu. Má ale každý ideál nějakou konečnou bázi? Obecně ne, nicméně dokážeme, že v případě oborů polynomů nad tělesem ano.

Definice. Komutativní okruh $\mathbf{R}$ se nazývá *noetherovský*, pokud má každý ideál v $\mathbf{R}$ konečnou bázi.

Příklad.
- Obory integrity hlavních ideálů jsou noetherovské, protože každý ideál má bázi sestávající z jediného prvku (je hlavní). Připomeňme např. obory $\mathbb{Z}$, $\mathbb{Z}[x]$, $\mathbf{T}[x]$ ($\mathbf{T}$ těleso), atd.
- Obory $\mathbb{Z}[x_1, \dots, x_k]$ a $\mathbf{T}[x_1, \dots, x_k]$ ($\mathbf{T}$ těleso) jsou noetherovské (i když to nejsou obory hlavních ideálů), jak plyne z níže uvedené Hilbertovy věty o bázi.
- Obor $\mathbb{Z}[X]$, kde X je nekonečná množina, noetherovský není. Např. ideál $\mathbf{I} = \{p : p(0, 0, \dots) = 0\}$ nemá konečnou bázi.

Lemma 20.2. *Nechť* $\mathbf{R}$ *je komutativní okruh. Pak* $\mathbf{R}$ *je noetherovský právě tehdy, když v* $\mathbf{R}$ *neexistuje nekonečná posloupnost ideálů* $\mathbf{I}_1 \subsetneq \mathbf{I}_2 \subsetneq \mathbf{I}_3 \subsetneq \dots$

Důkaz. ($\Rightarrow$) Uvažujme nekonečnou posloupnost ideálů $\mathbf{I}_1 \subset \mathbf{I}_2 \subset \dots$ v $\mathbf{R}$ a položme $\mathbf{I} = \bigcup_{j=1}^{\infty} \mathbf{I}_j$. Pak $\mathbf{I}$ je také ideál a předpokládejme, že $a_1, \dots, a_n$ je jeho konečná báze (taková existuje, neboť $\mathbf{R}$ je noetherovský). Máme $a_1, \dots, a_n \in \mathbf{I} = \bigcup_{j=1}^{\infty} \mathbf{I}_j$, takže pro každé i existuje j_i splňující $a_i \in \mathbf{I}_{j_i}$. Označme $k = \max_{i=1,\dots,n} j_i$. Pak ale $a_1, \dots, a_n \in \mathbf{I}_k$, tedy $\mathbf{I}_k = \langle a_1, \dots, a_n \rangle$, tedy $\mathbf{I}_k = \mathbf{I}$, spor.

($\Leftarrow$) Předpokládejme, že nějaký ideál $\mathbf{I}$ nemá konečnou bázi. Definujme následující posloupnost ideálů: položme $\mathbf{I}_1 = \langle a_1 \rangle$, kde $a_1 \in \mathbf{I}$ je zvoleno libovolné. Dále, indukcí, zvolme a_{i+1} tak, aby $a_{i+1} \in \mathbf{I} \smallsetminus \mathbf{I}_i$ (takové a_{i+1} existuje, protože $\mathbf{I} \neq \langle a_1, \dots, a_i \rangle$) a položme $\mathbf{I}_{i+1} = \langle a_1, \dots, a_{i+1} \rangle$. Dostali jsme nekonečnou posloupnost ideálů $\mathbf{I}_1 \subset \mathbf{I}_2 \subset \mathbf{I}_3 \subset \dots$, spor. $\square$

Věta 20.3 (Hilbertova věta o bázi)**.** *Nechť* $\mathbf{R}$ *je komutativní noetherovský okruh. Pak je okruh polynomů* $\mathbf{R}[x]$ *noetherovský.*

Důkaz. Pro spor předpokládejme, že ideál $\mathbf{I}$ v oboru $\mathbf{R}[x]$ nemá konečnou bázi. Zvolme p_1 některý z polynomů nejmenšího stupně v $\mathbf{I}$. Dále, indukcí, zvolme p_{i+1} některý z polynomů v $\mathbf{I} \smallsetminus \langle p_1, \dots, p_i \rangle$ nejmenšího možného stupně. Zřejmě $\deg p_1 \leq \deg p_2 \leq \deg p_3 \leq \dots$ Pro každé i definujme $a_i = \mathrm{lc}(p_i)$ a položme $\mathbf{J}_i = \langle a_1, \dots, a_i \rangle$. Máme tedy rostoucí posloupnost $\mathbf{J}_1 \subseteq \mathbf{J}_2 \subseteq \mathbf{J}_3 \subseteq \dots$ ideálů v $\mathbf{R}$. Protože je okruh $\mathbf{R}$ noetherovský, nemůže tato posloupnost obsahovat ostře rostoucí podposloupnost, a tedy existuje k splňující $\mathbf{J}_k = \mathbf{J}_{k+1} = \dots$. Pak ale a_{k+1} můžeme vyjádřit jako $\sum_{i=1}^{k} r_i a_i$ pro jistá $r_i \in \mathbf{R}$. Definujme polynom $\tilde{p}_i$ jako $x^s p_i$ pro takové s, aby $\deg \tilde{p}_i = \deg p_{k+1}$ a uvažujme polynom $p = \sum_{i=1}^{k} \tilde{p}_i r_i$. Ten má stejný vedoucí člen jako p_{k+1} a přitom $p \in \langle p_1 \dots, p_k \rangle$, takže polynom $p_{k+1} - p$ má menší stupeň než p_{k+1}. Ale $p_{k+1} - p \in \mathbf{I} \smallsetminus \langle p_1, \dots, p_k \rangle$, což je spor s volbou p_{k+1}. $\square$

Na závěr této sekce nastíníme princip metody Gröbnerových bází. Uvažujme obor $\mathbf{T}[x_1,\dots,x_k]$ ($\mathbf{T}$ je nějaké těleso) a jeho ideál $\mathbf{I}$. Metoda sestává ze dvou algoritmů:

(1) je-li dána „hezká báze” ideálu $\mathbf{I}$, existuje algoritmus rozhodující problém náležení ideálu $\mathbf{I}$;
(2) je-li dána nějaká konečná báze ideálu $\mathbf{I}$, existuje algoritmus, který vyrobí „hezkou bázi” ideálu $\mathbf{I}$.

Pod pojmem „hezká báze” zde rozumíme právě tzv. *Gröbnerovu bázi.* Její definici, která je poněkud technická, uvedeme později.

Úloha. Uvažujme ideál $\mathbf{I}=\langle x^2-x+1\rangle$ v oboru $\mathbf{T}[x]$. Rozhodněte, zda polynom x^5+x^3+1 náleží $\mathbf{I}$.

Úloha má samozřejmě velmi snadné řešení: protože je $\mathbf{I}$ hlavní ideál, platí $h\in\mathbf{I}$ právě tehdy, když x^2-x+1 dělí h. Stačí tedy vydělit $(x^5+x^3+1):(x^2-x+1)$ a zjistit zbytek. Zkusíme ovšem na této úloze ilustrovat princip metody Gröbnerových bází.

Protože $x^2-x+1\in\mathbf{I}$, neboli $x^2-x+1\equiv 0 \pmod{\mathbf{I}}$, platí

$$x^2\equiv x-1 \pmod{\mathbf{I}}.$$

Modulo ideál $\mathbf{I}$ tedy můžeme přepsat libovolné x^2, které se vyskytuje v daném polynomu f, na $x-1$. Označme g výsledný přepsaný polynom. Všimněte si, že $f\in\mathbf{I}$ právě tehdy, když $g\in\mathbf{I}$: přepíšeme-li člen $q\cdot x^2$ v polynomu f na člen $q\cdot(x-1)$ (kde q je libovolný polynom), máme $g=f-q\cdot x^2+q\cdot(x-1)=f-q\cdot(x^2-x+1)$. Protože $x^2-x+1\in I$, platí $f\in\mathbf{I}$ právě tehdy, když $g\in\mathbf{I}$.

Přepisujeme-li daný polynom x^5+x^3+1, dostáváme

$$\begin{aligned}
x^5+x^3+1=x^2\cdot x^3+x^3+1 &\to x^3(x-1)+x^3+1=x^4+1=x^2\cdot x^2+1\\
&\to x^2(x-1)+1=x^3-x^2+1=x\cdot x^2-x^2+1\\
&\to x(x-1)-x^2+1=-x+1\notin\mathbf{I}
\end{aligned}$$

Později nahlédneme, že v tomto případě tvoří polynom x^2-x+1 Gröbneorvu bázi ideálu $\mathbf{I}$, a proto $h\in\mathbf{I}$ právě tehdy když jej lze přepsat na nulový polynom. Tedy polynom $x^5+x^3+1\notin\mathbf{I}$.

Úloha. Uvažujme ideál $\mathbf{I}=\langle y^2-yz-1, yz-x-1\rangle$ v oboru $\mathbf{T}[x,y,z]$. Rozhodněte, zda polynom y^2z+xz^2 náleží $\mathbf{I}$.

Zde již s výše uvedeným elementárním postupem neuspějeme. Problém ale bude i s metodou přepisování — nejprve musíme vyřešit následující problémy:

- Který člen z bázových polynomů je ten přepisovaný? Pro polynomy jedné proměnné je výběr přirozený: ten vedoucí (tj. s nejvyšším exponentem). Jak definovat vedoucí člen obecných polynomů?
- Je-li bázových polynomů více, který v daném případě použít? Uvažujme v našem případě přepisovací pravidla

$$(1)\ y^2\to yz+1 \quad \text{a} \quad (2)\ yz\to x+1.$$

Pak lze přepisovat

$$y^2z+xz^2 \overset{(1)}{\to} z(yz+1)+xz^2=yz^2+xz^2+z \overset{(2)}{\to} z(x+1)+xz^2+z=xz^2+xz+2z,$$

ale také

$$y^2z + xz^2 \xrightarrow{(2)} y(x+1) + xz^2 = xz^2 + xy + y,$$

tedy přepisování není jednoznačné.

Problém vedoucích členů bude řešen v jedné z následujících sekcí. Odpovědí na druhou otázku pak je fakt, že pro Gröbnerovy báze je výběr pravidel lhostejný, výsledek vyjde vždy stejně. Jak Gröbnerovu bázi daného ideálu poznat a najít, odpoví také jedna z následujících sekcí.

Vrátíme-li se k výše uvedenému příkladu, uvažujme polynom $g = z(y^2 - yz - 1) - y(yz - x - 1)$. Zřejmě $g \in \mathbf{I}$. Ovšem $g = y^2z - yz^2 - z - y^2z + yx + y = -yz^2 - z + yx + y \xrightarrow{\#2} xy - xz + y - 2z \neq 0$, tedy g se nepřepisuje na 0. Zadaná báze ideálu $\mathbf{I}$ tedy není Gröbnerova.

Úloha. Uvažujme ideál $\mathbf{I} = \langle xy + 2x^3 + 2x^2, x^4 + x^3 + x^2 \rangle$ v oboru $\mathbf{T}[x, y]$. Rozhodněte, zda polynom $x^4y^4 + x^3y^3$ náleží $\mathbf{I}$ pokud víte, že zadaná báze je Gröbnerova vzhledem k lexikografickému uspořádání $x < y$. Dokažte, že tato báze není Gröbnerova vzhledem k lexikografickému uspořádání $y < x$.

21. Konvergentní relace

V celé této sekci uvažujme orientovaný graf $(X, \to)$, tj. X je (ne nutně konečná) množina a $\to$ binární relace na X. Typickým příkladem pro nás bude množina $X = T[x_1, \dots, x_k]$ a $\to$ relace přepisování (redukce) polynomů vzhledem k dané bázi.

Definice. Pro daný orientováný graf $(X, \to)$ definujme symboly

$x \xrightarrow{n} y \quad \equiv \quad$ existují $x_0, \dots, x_n$ splňující $x = x_0 \to x_1 \to x_2 \to \dots \to x_n = y$;

$x \xrightarrow{*} y \quad \equiv \quad$ existuje $n \geq 0$ splňující $x \xrightarrow{n} y$, tj. „z x lze dojít do y po šipkách";

$x \leftrightarrow y \quad \equiv \quad$ $x \to y$ nebo $y \to x$;

$x \overset{n}{\leftrightarrow} y \quad \equiv \quad$ existují $x_0, \dots, x_n$ splňující $x = x_0 \leftrightarrow x_1 \leftrightarrow x_2 \leftrightarrow \dots \leftrightarrow x_n = y$

$x \overset{*}{\leftrightarrow} y \quad \equiv \quad$ existuje $n \geq 0$ splňující $x \overset{n}{\leftrightarrow} y$ tj. „existuje neorientovaná cesta mezi x a y".

Definice. Řekneme, že vrchol x je *terminál*, jestliže neexistuje y splňující $x \to y$.

Definice. Řekneme, že relace $\to$ je

terminující $\equiv$ neexistuje nekonečná posloupnost $x_1 \to x_2 \to x_3 \to \dots$

normální $\equiv$ kdykoliv $x \overset{*}{\leftrightarrow} y$, pak existuje z splňující $x \xrightarrow{*} z$ a $y \xrightarrow{*} z$;

konvergentní $\equiv$ terminující a normální.

Metoda Gröbnerových bází je založena na (poněkud naivním, avšak účinném) algoritmu, který pro konvergentní relace umožňuje rozhodnout, zda mezi danými vrcholy existuje cesta.

Algoritmus 21.1.

`VSTUP:` $x, y \in X$.

`VÝSTUP:` Rozhodne, zda $x \overset{*}{\leftrightarrow} y$.

1. Přepisuj libovolně $x = x_0 \to x_1 \to x_2 \to \dots \to t$, kde t je terminál.
2. Přepisuj libovolně $y = y_0 \to y_1 \to y_2 \to \dots \to s$, kde s je terminál.
3. Pokud $s = t$, odpověz ANO.

Věta 21.2. *Nechť $\to$ je konvergentní relace a předpokládejme existenci algoritmu, který*

- *rozhoduje, zda je daný prvek x terminál;*
- *není-li x terminál, pak nalezne nějaké y splňující $x \to y$.*

Pak Algoritmus 21.1 funguje.

Důkaz. Ukážeme si správnost algoritmu. Díky terminovanosti se výpočet v krocích `1.` a `2.` zastaví v nějakém terminálu.

Zbývá ukázat, že $x \overset{*}{\leftrightarrow} y$ právě tehdy, když $s = t$. Implikace ($\Leftarrow$) je zřejmá, neboť $x \to x_1 \to \ldots \to t = s \leftarrow \ldots \leftarrow y_1 \leftarrow y$ a tedy $x \overset{*}{\leftrightarrow} y$. Předpokládejme, že $x \overset{*}{\leftrightarrow} y$. Protože $x \overset{*}{\to} t$ a $y \overset{*}{\to} s$, platí $t \overset{*}{\leftrightarrow} s$ ($\overset{*}{\leftrightarrow}$ je tranzitivní relace). Relace $\to$ je normální, takže existuje z takové, že $t \overset{*}{\to} z \overset{*}{\leftarrow} s$. Ale s a t jsou terminály a tedy $t = z = s$. $\square$

Důsledek 21.3. *Je-li $(X, \to)$ konvergentní a $t \overset{*}{\leftarrow} x \overset{*}{\to} s$, kde t a s jsou terminály, pak $t = s$.*

Definice. Tento (jednoznačně určený) terminál se nazývá *normální forma x*.

Jak ale zaručit, aby byl přepisující systém konvergentní? Terminovanost budeme zaručovat tak, že každé přepisující pravidlo bude mít na levé straně polynom v jistém smyslu větší než na pravé straně (např. polynom většího stupně, apod.). Místo normality budeme ověřovat slabší vlastnost — lokální konfluenci.

Definice. Řekneme, že $(X, \to)$ je *konfluentní*, pokud pro všechny u, x, y takové, že $x \overset{*}{\leftarrow} u \overset{*}{\to} y$, existuje z takové, že $x \overset{*}{\to} z \overset{*}{\leftarrow} y$.

Řekneme, že $(X, \to)$ je *lokálně konfluentní*, pokud pro všechny u, x, y takové že $x \leftarrow u \to y$, existuje z takové, že $x \overset{*}{\to} z \overset{*}{\leftarrow} y$.

Řekneme, že $(X, \to)$ je *(lokálně) konfluentní v bodě u*, pokud pro všechny x, y platí podmínka z definice (lokální) konfluence.

Nyní jsme si zadefinovali slabší vlastnosti než je normalita. Ukážeme si, že jsou postačující.

Poznámka. V důkazu následující věty a některých dalších využijeme následující variantu důkazu indukcí: Máme-li graf $(X, \to)$, kde $\to$ je terminující relace, a chceme-li dokázat tvrzení $P(x)$ pro všechna x, pak stačí dokázat tvrzení $P(x)$ za předpokladu, že $P(y)$ platí pro každé y takové, že $x \overset{*}{\to} y$.

Tomuto principu budeme říkat *zobecněná indukce.* Všimněte si, že pro $X = \mathbb{N}$ a $\to = >$ jde o důkaz indukcí.

Věta 21.4. *Nechť $(X, \to)$ je graf. Pak*

(1) *$(X, \to)$ je normální právě tehdy, když je konfluentní.*
(2) *Nechť $(X, \to)$ je terminující. Pak je normální právě tehdy, když je lokálně konfluentní.*

Terminovanost je v bodě (2) nutná. Následující graf je lokálně konfluentní ale není normální.

$$\begin{tikzcd} \bullet & \bullet \arrow[l] \arrow[r, bend left] & \bullet \arrow[l, bend left] \arrow[r] & \bullet \end{tikzcd}$$

Důkaz. Implikace ($\Rightarrow$) v (1) i (2) jsou triviální.

(1 $\Leftarrow$) Indukcí podle n dokážeme vlastnost z definice normality pro x, y splňující $x \overset{n}{\leftrightarrow} y$.

Pro $n = 1$ máme buď $x \to y$, nebo $y \to x$, tedy tvrzení platí.

Nechť nyní tvrzení platí pro n.

Máme $x \overset{n+1}{\longleftrightarrow} y$, t.j. existuje z takové, že $x \overset{n}{\leftrightarrow} z \leftrightarrow y$. Z indukčního předpokladu existuje u t.ž. $x \overset{*}{\to} u \overset{*}{\leftarrow} z$. Nyní buď $z \to y$ nebo $y \to z$.

Předpokládejme, že $z \to y$. Z konfluence užité na $u \overset{*}{\leftarrow} z \to y$ dostáváme existenci u, pro něž $u \overset{*}{\to} v \overset{*}{\leftarrow} y$. Situaci ilustruje následující obrázek.

$$\begin{tikzcd}
x \arrow[dr, "*"] & & \\
& u \arrow[dr, "*"] & \\
z \arrow[ur, "*"] \arrow[dr] & & v \\
& y \arrow[ur, "*"] &
\end{tikzcd}$$

Předpokládejme, že $y \to z$. Nyní $y \overset{*}{\to} u$ a jsme hotovi. Situaci ilustruje následující obrázek

$$\begin{tikzcd}
& x \arrow[dr, "*"] & \\
& & u \\
& z \arrow[ur, "*"] & \\
y \arrow[ur, "*"] & &
\end{tikzcd}$$

(2 $\Leftarrow$) Podle bodu (1) stačí dokázat, že $(X, \to)$ je konfluentní. Zobecněnou indukcí dokážeme tvrzení

$P(u)$: Pokud $x \overset{*}{\leftarrow} u \overset{*}{\to} y$, pak existuje z takové, že $x \overset{*}{\to} z \overset{*}{\leftarrow} y$.

Předpokládáme tedy, že $P(v)$ platí, pokud $u \underset{v}{\overset{*}{\to}}$.

Označme x_1 a y_1 libovolné vrcholy, pro které $x \overset{*}{\leftarrow} x_1 \leftarrow u \to y_1 \overset{*}{\to} y$. Protože $x_1 \leftarrow u \to y_1$ a $\to$ je lokálně konfluentní, existuje w, pro které $x_1 \overset{*}{\to} w \overset{*}{\leftarrow} y_1$. Z indukčního předpokladu dostáváme v takové, že $x \overset{*}{\to} v \overset{*}{\leftarrow} w$ (indukční předpoklad můžeme použít, neboť $u \to x_1$ a $x \overset{*}{\leftarrow} x_1 \overset{*}{\to} w$). Dále najdeme z takové, že $v \overset{*}{\to} z \overset{*}{\leftarrow} y$ (opět využíváme indukčního předpokladu). Tedy z je společný následník x a y. Situaci ilustruje následující obrázek

$$\begin{tikzcd}
& & x \arrow[dr, "*"] & & \\
& x_1 \arrow[ur, "*"] \arrow[dr, "*"] & & v \arrow[ddr, "*"] & \\
u \arrow[ur] \arrow[dr] & & w \arrow[ur, "*"] & & \\
& y_1 \arrow[ur, "*"] \arrow[dr, "*"] & & & z \\
& & y \arrow[urr, "*"] & &
\end{tikzcd}$$

$\square$

Připomeňme pojem uspořádání na množině.

Definice. Relace $<$ na množině X se nazývá *uspořádání*, pokud $<$ je tranzitivní (tedy z $x < y$ a $y < z$ plyne $x < z$) a antisymetrická (nestane se současně $x < y$ a $y < x$.

Lemma 21.5. *Mějme na X uspořádání $<$ takové, že jestliže $x \to y$, pak $x > y$. Předpokládejme, že $>$ je terminující (neboli také $\to$ je terminující).*

Definujme $x \overset{*}{\underset{<u}{\longleftrightarrow}} y$, pokud existují $x_1, \ldots, x_{n-1} < u$ takové, že $x = x_0 \leftrightarrow x_1 \leftrightarrow x_2 \leftrightarrow \ldots \leftrightarrow x_n = y$. Předpokládejme, že pokud $x \leftarrow u \to y$, pak $x \overset{*}{\underset{<u}{\longleftrightarrow}} y$.

Pak $(X, \to)$ je konvergentní.

Důkaz. Podle předchozí věty stačí dokázat, že $(X, \to)$ je konfluentní. Tvrzení dokážeme zobecněnou indukcí pro $(X, >)$. Zvolíme u takové, že $x \overset{*}{\leftarrow} u \overset{*}{\to} y$ a předpokládáme, že pro libovolné $u' < u, x', y'$, takové, že $x' \overset{*}{\leftarrow} u' \overset{*}{\to} y'$, platí $x' \overset{*}{\to} z \overset{*}{\leftarrow} y'$ pro nějaké z.

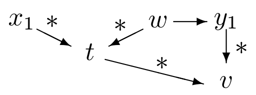
Nechť opět x_1, y_1 značí takové prvky X, pro které $x \overset{*}{\leftarrow} x_1 \leftarrow u \to y_1 \overset{*}{\to} y$. Podle předpokladu $x_1 \overset{*}{\underset{<u}{\longleftrightarrow}} y_1$, tedy $x_1 \overset{n}{\underset{<u}{\leftrightarrow}} y_1$ pro nějaké n. Indukcí podle n dokážeme, že $x_1 \overset{*}{\to} v \overset{*}{\leftarrow} x_2$ pro nějaké v.

Pro $n = 1$ platí $x_1 \to y_1$ nebo $y_1 \to x_1$.

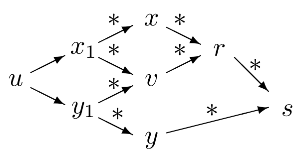
Z platnosti tvrzení pro n nyní odvodíme platnost pro $n+1$. Označme $w < u$ ten prvek, pro který $x_1 \overset{n-1}{\underset{<u}{\longleftrightarrow}} w \leftrightarrow y_1$. Z indukčního předpokladu máme existenci t, že $x_1 \overset{*}{\to} t \overset{*}{\leftarrow} w \leftrightarrow y_1$. Nyní buď a) $y_1 \to w$, nebo b) $w \to y_1$.

a) Máme $x_1 \overset{*}{\to} t \leftarrow w \overset{*}{\leftarrow} y_1$ a tvrzení platí.

b) Protože $w < u$ můžeme použít předpoklad zobecněné indukce na $t \overset{*}{\leftarrow} w \to y_1$. Dostáváme v takové, že $t \overset{*}{\to} v \overset{*}{\leftarrow} y_1$. Situaci ilustruje následující obrázek



Nyní dokončíme důkaz lemmatu. Protože $x_1 < u$ a $x \overset{*}{\leftarrow} x_1 \overset{*}{\to} v$ dostáváme r takové, že $x \overset{*}{\to} r \overset{*}{\leftarrow} v$. Dále $y_1 < u$ a $r \overset{*}{\leftarrow} y_1 \overset{*}{\to} y$, takže máme s takové, že $r \overset{*}{\to} s \overset{*}{\leftarrow} y$. Nyní $x \overset{*}{\to} s \overset{*}{\leftarrow} y$ a důkaz je hotov. Situaci ilustruje následující obrázek



□

22. Uspořádání termů

Termem rozumíme výraz $x_1^{a_1} \cdot \ldots \cdot x_k^{a_k}$ (tedy monočleny s koeficientem 1 v $\mathbf{T}[x_1, \ldots, x_k]$). Označme $[x_1, \ldots, x_k]$ množinu všech termů v proměnných $x_1, \ldots, x_k$. Všimněte si, že tato množina je monoid vzhledem k $\cdot$, kde jednotka je $1 = x_1^0 \cdot \ldots \cdot x_k^0$.

Definice. Uspořádání $<$ nazveme *přípustné uspořádání* na $[x_1, \ldots, x_k]$, pokud

(1) $>$ je lineární a terminující.
(2) $1 \leq t$ pro všechna t.

(3) Jestliže $s \leq t$, pak $rs \leq rt$ pro všechna t.

Příklad. Ukážeme si nejběžnější příklady přípustných uspořádání:

- *lexikografické* uspořádání. Položíme $x_1^{a_1} \cdot \ldots \cdot x_k^{a_k} <_{\text{LEX}} x_1^{b_1} \cdot \ldots \cdot x_k^{b_k}$ právě tehdy, když existuje i t.ž. $a_1 = b_1, \ldots, a_i = b_i, a_{i+1} < b_{i+1}$. O uspořádání tedy rozhoduje stupeň proměnné x_1, jsou-li rovné, pak stupeň u x_2, při rovnosti obou stupeň u x_3, ...

  Proto například v daném uspořádání platí

$$x_1 > x_2 > x_3 > \ldots > x_k > 1$$
$$x_1 > x_2^{100} > x_2 x_3 > x_3^{100}$$
$$x_1 x_2^3 x_3^3 > x_1 x_2^2 x_3^{10} > x_1 x_2^2 x_3^4$$

- *graduovaně lexikografické* uspořádání. Položíme $s <_{\text{GLEX}} t$ právě tehdy, když buď $\deg s < \deg t$ nebo $\deg s = \deg t$ a zároveň $s <_{\text{LEX}} t$. O uspořádání tedy rozhoduje celkový stupeň, až při rovnosti pak $<_{\text{LEX}}$.

  V daném uspořádání tedy platí

$$x_1 > x_2 > \ldots > x_k > 1$$
$$x_2^{100} > x_3^{100} > x_2 x_3 > x_1$$
$$x_1 x_2^2 x_3^{10} > x_1 x_2^3 x_3^3 > x_1 x_2^2 x_3^4$$

- *společné zobecnění* je dáno váhovým vektorem $w = (w_1, \ldots, w_k)$, kde $w_i \geq 0$. Položíme $s <_w t$ právě tehdy, když buď $\sum a_i w_i < \sum b_i w_i$, nebo $\sum a_i w_i = \sum b_i w_i$ a zároveň $s <_{\text{LEX}} t$.

  Tedy například pro váhový vektor $w = (0, 0, \ldots, 0)$ dostáváme lexikografické uspořádání, zatímco pro $w = (1, 1, \ldots, 1)$ dostáváme graduovaně lexikografické uspořádání.

Poznámka. Pro $[x, y, z]$ vždy specifikujeme pořadí, obvykle $x > y > z$ nebo naopak.

Úloha. Dokažte, že uspořádání v příkladu jsou přípustná.

Definice. Nechť $<$ je přípustné uspořádání na $[x_1, \ldots, x_k]$ a nechť $f \in \mathbf{T}[x_1, \ldots, x_k]$. Pak definujeme

- *vedoucí term* jako největší term v f vzhledem k danému uspořádání, značíme jej $\text{lt}(f)$
- *vedoucí koeficient* jako koeficient u vedoucího termu, značíme $\text{lc}(f)$
- *vedoucí monočlen* jako člen $\text{lc}(f) \cdot \text{lt}(f)$, značíme $\text{lm}(f)$

Příklad. Uvažujme polynom $f = 3xy^2 + 2x^2 - x + 1$. Určete $\text{lt}(f), \text{lc}(f), \text{lm}(f)$ vzhledem k (1) lexikografickému uspořádání (2) graduovaně lexikografickému uspořádání, jestliže $x > y$.

(1) Uspořádání termů polynomu f vzhledem k lexikografickému uspořádání je $x^2 > xy^2 > x > 1$. Proto $\text{lt}(f) = x^2$, $\text{lc}(f) = 2$ a $\text{lm}(f) = 2x^2$.
(2) Uspořádní termů polynomu f vzhledem ke graduovaně lexikografickému uspořádání je $xy^2 > x^2 > x > 1$. Proto $\text{lt}(f) = xy^2$, $\text{lc}(f) = 3$ a $\text{lm}(f) = 3xy^2$.

Definice. Nechť $<$ je přípustné uspořádání na $[x_1, \ldots, x_k]$. Pro $f, g \in \mathbf{R}[x_1, \ldots, x_k]$ položíme $f \ll g$, pokud platí jedna z následujících podmínek

- $f = 0$ a $g \neq 0$

- $\mathrm{lt}(f) < \mathrm{lt}(g)$
- $0 \neq \mathrm{lt}(f) = \mathrm{lt}(g)$ a $f - \mathrm{lm}(f) \ll g - \mathrm{lm}(g)$

Pozorování 22.1. *$\ll$ je uspořádání na $\mathbf{R}[x_1, \ldots, x_k]$.*

Úloha. Seřaďte následující polynomy $3xy^2 + 2x^2 - 1$, $xy^2 + x^2y$, $x^2y + 2x^2y^2 + 2$, $x^2y + x + 2y^2$ v následujících uspořádáních a) $\ll_{\mathrm{LEX}}$, $x > y$ b) $\ll_{\mathrm{LEX}}$, $x < y$ c) $\ll_{\mathrm{GLEX}}$, $x > y$ d) $\ll<_{\mathrm{GLEX}}$, $x < y$

Tvrzení 22.2. *Nechť $\mathbf{T}$ je těleso a $<$ je přípustné uspořádání na $[x_1, \ldots, x_k]$. Pak $\gg$ je terminující.*

Důkaz. Pro spor předpokládejme, že existuje nekonečná posloupnost $f_1 \gg f_2 \gg f_3 \gg \ldots$. Pro každé $i = 1, 2, \ldots$ označme $J_i = \langle f_1, \ldots, f_i \rangle$. Pro každé $j \leq i$ platí $f_{i+1} \ll f_j$, tedy $f_{i+1} \ll t_j f_j$ pro libovolná $t_j \mathbf{T}$, takže $f_{i+1} \ll \sum_{j \leq i} t_j f_j$. Dokázali jsme, že $f_{i+1} \ll g$ pro libovolné $g \in J_i$. Proto $f_{i+1} \notin J_i$ (neboť $f_{i+1} \not\ll f_{i+1}$) a máme tedy nekonečenou ostře rostoucí posloupnost $J_1 \subsetneq J_2 \subsetneq \ldots$ ideálů $\mathbf{T}[x_1, \ldots, x_k]$. To je spor, protože $\mathbf{T}[x_1, \ldots, x_n]$ je podle Hilberovy věty o bázi noetherovský. □

23. Redukce polynomů vzhledem k dané bázi

Definice. Pro $p, q, r \in \mathbf{T}[x_1, \ldots, x_k]$ definujeme relaci $p \xrightarrow{r} q$ následujícími ekvivalentními způsoby

- existuje monočlen m v polynomu p takový, že $\mathrm{lt}(r)|m$ a $q = p - \frac{m}{\mathrm{lm}(r)} \cdot r$
- platí $p = p' + m$, kde $m = a \cdot \mathrm{lm}(r)$, $q = p' + a \cdot (\mathrm{lm}(r) - r)$, přičemž p' neobsahuje člen s termem $\mathrm{lt}(m)$

Dále pro $R \subseteq \mathbf{T}[x_1, \ldots, x_k]$ definujeme $p \xrightarrow{\mathbf{R}} q$ právě tehdy, když $p \xrightarrow{r} q$ pro nějaké $r \in \mathbf{R}$.

Příklad. Uvažujme polynom $r = 2y^2 + 3x^2y - 1$ v uspořádání $<_{\mathrm{LEX}}$, $y > x$. Tedy $\mathrm{lm}(r) = 2y^2$. Definice $\xrightarrow{r}$ říká, že jakýkoliv člen tvaru $t \cdot 2y^2$ můžeme přepsat na $t \cdot (-3x^2y + 1)$, kde t je nějaký monočlen. Takže například

$$6x^2y^2 + 1 = 3x^2 \cdot 2y^2 + 1 \xrightarrow{r} 3x^2(-3x^2y + 1) + 1 = -9x^4y + 3x^2 + 1.$$

.

Příklad. Uvažujme polynomy $r = x - y$ a $p = 2x = x + x$. Neplatí $p \xrightarrow{r} x + y$. Musíme přepsat celý monočlen $2x$, tedy správně je $p \xrightarrow{r} 2y$.

Pozorování 23.1. *Nechť $p, q \in \mathbf{T}[x_1, \ldots, x_k]$, $R \subseteq \mathbf{T}[x_1, \ldots, x_k]$.*

- *Relace $\xrightarrow{\mathbf{R}}$ je terminující, dokonce pokud $p \xrightarrow{\mathbf{R}} q$, pak $p \gg q$*
- *Jestliže $p \xrightarrow{\mathbf{R}} q$, pak $mp \xrightarrow{\mathbf{R}} mq$ pro libovolný monočlen m m*
- *Jestliže $p \xrightarrow{\mathbf{R}} q$, pak pro každý polynom s existuje polynom u takový, že $p + s \xrightarrow{\mathbf{R}} u \xleftarrow{\mathbf{R}} q + s$*

Poznámka. Obecně neplatí $p + s \xrightarrow{\mathbf{R}} q + s$: Nechť $\mathbf{R} = \{x - y\}$, $x > y$. Pak neplatí $x + x \xrightarrow{\mathbf{R}} x + y$ (ale $x + x \xrightarrow{\mathbf{R}} 2y \xleftarrow{\mathbf{R}} x + y$).

Definice. Nechť $R \subseteq \mathbf{T}[x_1, \dots, x_k]$. Řekneme, že $\mathbf{R}$ je *Gröbnerova báze*, pokud relace $\overset{R}{\to}$ je konvergentní.

Věta 23.2. *Nechť $p, q \in \mathbf{T}[x_1, \dots, x_k]$ a $\mathbf{R} \subseteq \mathbf{T}[x_1, \dots, x_k]$. Pak $p \xleftrightarrow[*]{\mathbf{R}} q$ právě tehdy, když $p - q \in \langle \mathbf{R} \rangle$, neboli $p \equiv q \bmod \langle \mathbf{R} \rangle$.*

Speciálně $p \in \langle \mathbf{R} \rangle$ právě tehdy, když $p \xleftrightarrow[*]{\mathbf{R}} 0$.

Důkaz. ($\Rightarrow$) Budeme postupovat indukcí podle délky n cesty $p \xleftrightarrow[*]{\mathbf{R}} q$. Pro $n = 1$ můžeme předpokládat, že $p \overset{\mathbf{R}}{\to} q$, tedy $p - q = \frac{m}{\mathrm{lm}(r)} \cdot r$ pro nějaké $r \in \mathbf{R}$. Protože $r \in \langle \mathbf{R} \rangle$, je pravá strana prvkem $\langle R \rangle$.

Nyní provedeme indukční krok. Cestu $p \xleftrightarrow[n+1]{\mathbf{R}} q$ můžeme psát $p \xleftrightarrow[n]{\mathbf{R}} s \xleftrightarrow{\mathbf{R}} q$. Z indukčního předpokladu víme, že $p - s \in \langle \mathbf{R} \rangle$ a z 1. kroku indukce víme, že $s - q \in \langle \mathbf{R} \rangle$. Tedy $(p - s) + (s - q) = p - q \in \langle \mathbf{R} \rangle$.

($\Leftarrow$) Předpokládejme, že $p - q = \sum_{r \in \mathbf{R}} a_r \cdot r$ pro nějaká a_r. Roznásobením dostáváme $\sum_{r \in \mathbf{R}} a_r \cdot r = \sum_{i=1}^{N} b_i r_i$, kde b_i jsou monočleny a $r_i \in \mathbf{R}$. Potom $\sum_{i=1}^{N} b_i r_i \overset{r_1}{\to} \sum_{i=2}^{N} b_i r_i \overset{r_2}{\to} \sum_{i=3}^{N} b_i r_i \overset{r_3}{\to} \dots \overset{r_N}{\to} 0$. Tedy $p - q \xrightarrow[*]{\mathbf{R}} 0$.

Podle posledního bodu Pozorování 23.1 platí $(p - q) + q = p \xleftrightarrow[*]{\mathbf{R}} q = 0 + q$. □

Důsledek 23.3. *Je-li $p \in \mathbf{T}[x_1, \dots, x_k$ a $\mathbf{R} \subseteq \mathbf{T}[x_1, \dots, x_k]$ Gröbnerova báze, pak $p \in \langle \mathbf{R} \rangle$ právě tehdy, když $p \overset{\mathbf{R}}{\underset{*}{\to}} 0$.*

Důkaz. Gröbnerova báze je konvergentní a tedy $p \xleftrightarrow[*]{\mathbf{R}} 0$ právě tehdy, když $p \overset{\mathbf{R}}{\underset{*}{\to}} 0$. Užitím předchozí věty dostáváme $p \overset{\mathbf{R}}{\underset{*}{\to}} 0$ právě tehdy, když $p \in \langle \mathbf{R} \rangle$. □

24. Buchbergerova věta a Buchbergerův algoritmus

V dalších kapitolách se budeme zabývat problémem, jak poznat Gröbnerovu bázi a ukážeme si způsob, jak nějakou Gröbnerovu bázi najít.

Mějme $\mathbf{R} \subseteq \mathbf{T}[x_1, \dots, x_k]$ a $r_1, r_2 \in \mathbf{R}$ a nějaké přípustné uspořádání $<$.

Definice. Řekneme, že (h_1, h_2) je *kritický pár* pro polynomy r_1, r_2, pokud $h_2 \overset{r_2}{\leftarrow} t \overset{r_1}{\to} h_1$, kde $t = \mathrm{NSN}(\mathrm{lt}(r_1), \mathrm{lt}(r_2))$, kde NSN značí nejmenší společný násobek.

S-polynomem pro polynomy r_1, r_2 rozumíme polynom $\mathrm{spol}(r_1, r_2) = h_1 - h_2$.

Příklad. Mějme $r_1 = x^2 y - 1$, $r_2 = xy^2 - 1$. Pak $t = \mathrm{NSN}(\mathrm{lt}(r_1), \mathrm{lt}(r_2)) = \mathrm{NSN}(x^2 y, xy^2) = x^2 y^2$. Máme $t \overset{r_1}{\to} y$ a $t \overset{r_2}{\to} x$, tedy (y, x) je kritický pár, $\mathrm{spol}(r_1, r_2) = y - x$.

Tvrzení 24.1. *Jestliže $p \xrightarrow[*]{\mathbf{R}} q$, pak $p + s \xleftrightarrow[* \ll p+s]{\mathbf{R}} q + s$.*

Důkaz. Protože $p \xrightarrow[*]{\mathbf{R}} q$, platí $p \overset{\mathbf{R}}{\to} u_1 \overset{\mathbf{R}}{\to} u_2 \overset{\mathbf{R}}{\to} \dots \overset{\mathbf{R}}{\to} q$. Užitím Pozorování 23.1 dostáváme $p + s \to v_0 \leftarrow u_1 + s \to v_1 \leftarrow u_2 + s \to \dots \leftarrow q + s$. Ale $p + s \gg u_1 + s \gg u_2 + s \gg \dots$, přičemž $u_1 + s \gg v_0$, $u_2 + s \gg v_1, \dots$ □

Věta 24.2 (Buchbergerova). *Následující tvrzení jsou ekvivalentní*

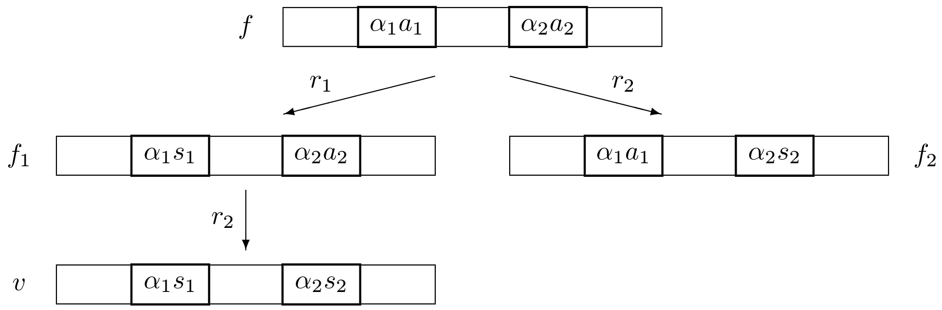
(1) $\mathbf{R}$ *je Gröbnerova báze*
(2) *Pro každý kritický pár* (h_1, h_2) *nějakých dvou prvků z* $\mathbf{R}$ *existuje polynom* u *takový, že* $h_1 \xrightarrow[*]{\mathbf{R}} u \xleftarrow[*]{\mathbf{R}} h_2$.
(3) $\mathrm{spol}(r_1, r_2) \xrightarrow[*]{\mathbf{R}} 0$ *pro libovolné polynomy* $r_1, r_2 \in \mathbf{R}$.

Důkaz. (1) $\Rightarrow$ (2). $\mathbf{R}$ je konvergentní, tedy konfluentní (Věta 21.4). Speciálně platí podmínka z bodu (2).

(1) $\Leftarrow$ (2). Podle Lemmatu 21.5 stačí ukázat následující. Pokud $f_1 \overset{r_1}{\leftarrow} f \overset{r_2}{\rightarrow} f_2$, pak $f_1 \underset{* \ll f}{\overset{\mathbf{R}}{\longleftrightarrow}} f_2$. Může nastat několik případů.

a)r_1, r_2 se přepisují v f různé termy. Nechť r_1 přepisuje a_1 na s_1 a r_2 přepisuje a_2 na s_2. Polynom f tedy můžeme přepsat pomocí r_1 na f_1. Protože však $\alpha_1 s_1 \ll \alpha_2 a_2$ (tedy v $\alpha_1 s_1$ se člen se stejným termem nevyskytuje), můžeme ještě f_1 přepsat pomocí r_2 na v.

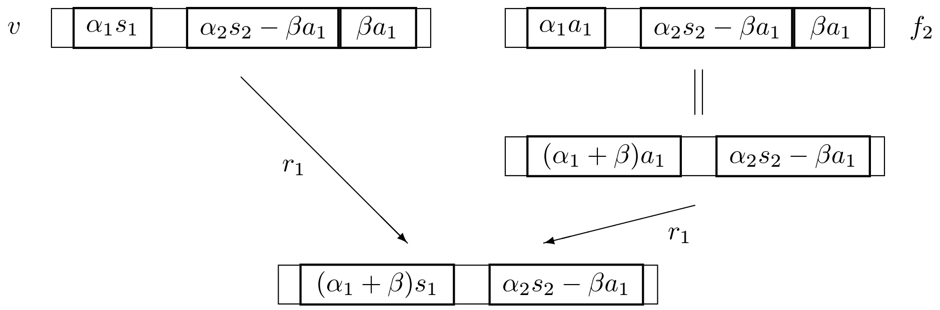
Pomocí r_2 můžeme f přepsat na polynom f_2. Situaci ilustruje následující obrázek



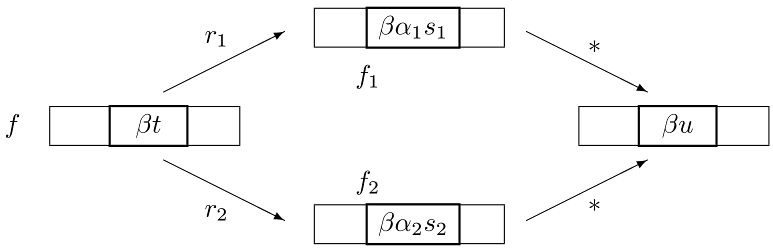
Při dalším přepisování rozlišíme následující situace:

i) V $\alpha_2 s_2$ se nevyskytuje člen se stejným termem jako $\alpha_1 a_1$. Pak v polynomu f_2 můžeme $\alpha_1 a_1$ přepsat pomocí r_1 na $\alpha_1 s_1$. Tím dostaneme stejný polynom, jako je v a tedy $f_1 \underset{* \ll f}{\overset{\mathbf{R}}{\longleftrightarrow}} f_2$.

ii) V $\alpha_2 s_2$ se vyskytuje člen se stejným termem jako $\alpha_1 a_1$. Pak člen $\alpha_2 s_2$ lze rozepsat na dva členy $\alpha_2 s_2 - \beta a_1$, βa_1 tak, že $\mathrm{lt}(\beta a_1) = \mathrm{lt}(\alpha_1 a_1)$. Oba polynomy v, f_1 přepíšeme podle r_1 a dostaneme tentýž polynom. Takže $f_1 \underset{* \ll f}{\overset{\mathbf{R}}{\longleftrightarrow}} f_2$. Situaci ilustruje následující obrázek.



b) r_1, r_2 se přepisují v f ten samý term, označme jej βt, kde $t = \mathrm{NSN}(\mathrm{lt}(r_1), \mathrm{lt}(r_2))$. Označme $a_1 := \mathrm{lt}(r_1)$ a $a_2 := \mathrm{lt}(r_2)$. Tedy $t = \alpha_1 a_1 = \alpha_2 a_2$. Nechť r_1 přepisuje a_1 na s_1 a nechť r_2 přepisuje a_2 na s_2. Z předpokladu věty máme zaručenou existenci polynomu u takové, že pokud $\alpha_1 s_1 \leftarrow t \rightarrow \alpha_2 s_2$, pak $\alpha_1 s_1 \overset{*}{\rightarrow} u \overset{*}{\leftarrow} \alpha_2 s_2$. Tedy pokud $\beta\alpha_1 s_1 \leftarrow \beta t \rightarrow \beta\alpha_2 s_2$, pak $\beta\alpha_1 s_1 \overset{*}{\rightarrow} \beta u \overset{*}{\leftarrow} \beta\alpha_2 s_2$. Situaci ilustruje následující obrázek.



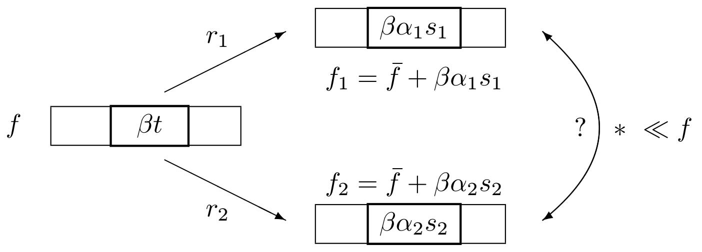
Vidíme, že $f_1 \overset{\mathbf{R}}{\underset{*}{\longleftrightarrow}} f_2$. Platí $f_1 \ll f$ a $f_2 \ll f$, proto z Tvrzení 24.1 vyplývá $f_1 \overset{\mathbf{R}}{\underset{* \ll f}{\longleftrightarrow}} f_2$.

(1) $\Rightarrow$ (3) Protože $h_1 \leftarrow t \rightarrow h_2$, t.j. $h_1 \overset{\mathbf{R}}{\underset{*}{\longleftrightarrow}} h_2$, máme $h_1 - h_2 \in \langle \mathbf{R} \rangle$. Podle Pozorování 23.1 existuje polynom u, takový, že $h_1 - h_2 \overset{*}{\rightarrow} u \overset{*}{\leftarrow} 0 = h_2 - h_2$. Ale 0 je terminál, a proto $u = 0$. Takže $\mathrm{spol}(r_1, r_2) = h_1 - h_2 \overset{\mathbf{R}}{\underset{*}{\longrightarrow}} 0$.

(1) $\Leftarrow$ (3) Tento bod se dokáže analogicky jako bod (1) $\Leftarrow$ (2) užitím Lemma 21.5. Opět rozlišíme dvě situace

a) r_1, r_2 se přepisují v f různé termy. Postupujeme jako v (1) $\Leftarrow$ (2) *a)*.

b) r_1, r_2 se přepisují v f na stejný term, označme jej βt. Nechť r_1 přepisuje βt na $\beta\alpha_1 s_1$ a nechť r_2 přepisuje βt na $\beta\alpha_2 s_2$. Označme $\bar{f} + \beta\alpha_1 s_1$ polynom vzniklý přepsáním f pomocí r_1 a $\bar{f} + \beta\alpha_2 s_2$ polynom vzniklý přepsáním f pomocí r_2. Z předpokladu víme, že $\mathrm{spol}(r_1, r_2) = \alpha_1 s_1 - \alpha_2 s_2 \overset{\mathbf{R}}{\underset{*}{\rightarrow}} 0$, tedy i $\beta\alpha_1 s_1 - \alpha_2 s_2 \overset{\mathbf{R}}{\underset{*}{\rightarrow}} 0$. Užitím Tvrzení 24.1 se analogicky ukáže, že $\beta\alpha_1 s_1 \overset{\mathbf{R}}{\underset{* \ll f}{\longleftrightarrow}} \beta\alpha_2 s_2$. Proto $\bar{f} + \beta\alpha_1 s_1 = f_1 \overset{\mathbf{R}}{\underset{* \ll f}{\longleftrightarrow}} f_2 = \bar{f} + \beta\alpha_1 s_1$. Situaci ilustruje následující obrázek.



□

Právě dokázaná věta nám dává návod jak na ověření, že daná báze je Gröbnerova, tak na hledání Gröbnerovy báze. Algoritmus na ověření je zřejmý: Pro každou dvojici $r_1, r_2 \in \mathbf{R}$ budeme přepisovat $\mathrm{spol}(r_1, r_2)$ tak dlouho, dokud to bude možné. Jestliže se nám podaří přepsat všechny S-polynomy na 0, pak testovaná báze je Gröbnerova.

Na podobném principu funguje také algoritmus, který z dané báze vyrobí Gröbnerovu bázi. Tento algoritmus se nazývá *Buchbergerův algoritmus.*

Algoritmus 24.3 (Buchberger)**.**

`VSTUP:` konečná množina polynomů $\mathbf{R}$
`VÝSTUP:` Gröbnerova báze $\mathbf{S}$, t.ž. $\langle \mathbf{R} \rangle = \langle \mathbf{S} \rangle$
`1.` Definujme množinu všech dvojic $P := \{(r, s) : r, s \in \mathbf{R}\}$
`2.` `WHILE` $P \neq 0$ `DO`
vezmi nějakou dvojici $(r, s) \in P$
spočti $\mathrm{spol}(r, s) \xrightarrow[*]{\mathbf{R}} p$ pro nějaký terminál p
`IF` $p = 0$ `THEN` $P := P \setminus \{(r, s)\}$
`ELSE` $P := (P \cup \{(r, p) : r \in \mathbf{R}\}) \setminus \{(r, s)\}$ a $\mathbf{R} := \mathbf{R} \cup \{p\}$

Tvrzení 24.4. *Buchbergerův algoritmus při libovolném vstupu skončí v konečně mnoha krocích a výsledkem bude Gröbnerova báze.*

Důkaz. Snadno nahlédneme, že každé p přidané do $\mathbf{R}$ náleží $\langle \mathbf{R} \rangle$. Pokud se algoritmus zastaví, výsledek bude podle Buchbergerovy věty Gröbnerova báze.

Zbývá dokázat, že se algoritmus zastaví. Předpokládejme, že ne. Označme $\mathbf{R}_0 := \mathbf{R}, \mathbf{R}_1, \mathbf{R}_2, \ldots$ postupně se zvětšující bázi. Definujme ideály $J_i := \langle \mathrm{lt}(r) : r \in \mathbf{R}_i \rangle$. Máme $J_0 \subseteq J_1 \subseteq J_2 \subseteq \ldots$. Z noetherovskosti existuje k takové, že $J_k = J_{k+1} = J_{k+2} = \ldots$. Nechť $q \in \mathbf{R}_{k+1} \setminus \mathbf{R}_k$, t.j. q je terminál vzniklý redukcí nějakého S-polynomu. Pak $\mathrm{lt}(q) \in J_{k+1} = J_k$, t.j. $\mathrm{lt}(q) = \sum_{r \in \mathbf{R}_k} a_r \cdot \mathrm{lt}(r)$ pro nějaké $a_r \in \mathbf{T}[x_1, \ldots, x_k]$. Roznásobením a sečtením stejných členů dostáváme $\sum_{r \in \mathbf{R}_k} a_r \cdot \mathrm{lt}(r) = \sum_{i=1}^{N} b_i \cdot \mathrm{lt}(r_i)$, kde $r_i \in \mathbf{R}_k$ a b_i jsou monočleny. Protože $\mathrm{lt}(q)$ je monočlen, je $N = 1$ a $\mathrm{lt}(q) = b \cdot \mathrm{lt}(r)$ pro nějaký monočlen b a $r \in \mathbf{R}_k$. Tedy q lze redukovat pomocí r. To je spor s tím, že q je terminál. □

Příklad. Nechť $\mathbf{R} = \{r_1, r_2\}$ a $r_1 = x^2y^2 + y - 1$, $r_2 = x^2y + x$ v uspořádání $<_{\mathrm{GLEX}}$, $y > x$.

V daném uspořádání platí $\mathrm{lt}(r_1) = x^2y^2$ a $\mathrm{lt}(r_2) = x^2y$. Budeme postupovat Buchbergerovým algoritmem.

(1) platí $-y + 1 \overset{r_1}{\leftarrow} x^2y^2 \overset{r_2}{\rightarrow} -xy$, tedy $\mathrm{spol}(r_1, r_2) = xy - y + 1 := r_3$

(2) platí $-y+1 \overset{r_1}{\leftarrow} x^2y^2 \overset{r_3}{\rightarrow} xy^2 - xy$, tedy $\mathrm{spol}(r_1,r_3) = -xy^2 + xy - y + 1 \overset{r_3}{\rightarrow} -xy^2 := r_4$
(3) platí $-x \overset{r_2}{\leftarrow} x^2y \overset{r_3}{\rightarrow} xy - x$, tedy $\mathrm{spol}(r_2,r_3) = -xy \overset{r_3}{\rightarrow} -y+1 := r_5$
(4) platí $y-1 \overset{r_3}{\leftarrow} xy \overset{r_5}{\rightarrow} x$, tedy $\mathrm{spol}(r_3,r_5) = y - x - 1 \overset{r_5}{\rightarrow} -x := r_6$
(5) platí $y-1 \overset{r_3}{\leftarrow} xy \overset{r_6}{\rightarrow} 0$, tedy $\mathrm{spol}(r_3,r_6) = y - 1 \overset{r_5}{\rightarrow} 0$
(6) platí $x \overset{r_5}{\leftarrow} xy \overset{r_6}{\rightarrow} 0$, tedy $\mathrm{spol}(r_5,r_6) = x \overset{r_6}{\rightarrow} 0$

Lze ověřit, že všechny další kombinace se redukují na 0. Tedy jsme našli Gröbnerovu bázi $S = \{r_1, r_2, r_3, r_4, r_5, r_6\}$.

Lze nahlédnout, že $\langle \mathbf{R} \rangle = \langle S \rangle = \langle r_5, r_6 \rangle$ (ostatní se dají pomocí r_5, r_6 přepsat na 0). Přitom r_5, r_6 je Gröbnerova báze. V další kapitole se budeme zabývat problémem, jak nalezenou Gröbnerovu redukovat.

Úloha. Zjistěte, zda $x^2y^3 - 2xy + 3y \in \langle \mathbf{R} \rangle$.

Úloha. Najděte Gröbnerovu bázi pro $\langle xy^2 + x^2 + x, x^2y + x \rangle$ v $\mathbb{Q}[x,y]$ v uspořádání $<_{\mathrm{LEX}}$, kde a) $x > y$ b) $x < y$

25. Redukce Gröbnerových bází

V minulé kapitole jsme pospali algoritmus na nalezení nějaké Gröbnerovy báze. Jak jsme viděli na příkladu, nalezená báze nemusí být (a obvykle nebývá) nejmenší možná. V této kapitole se zaměříme na redukci nalezené Gröbnerovy báze na v jistém smyslu minimální tvar.

Tvrzení 25.1. *Nechť $I = \langle \mathbf{R} \rangle$, kde $\mathbf{R}$ je konečná. Pak následující tvrzení jsou ekvivalentní*
(1) *$\mathbf{R}$ je Gröbnerova báze I*
(2) *pro každý polynom $f \in I$ platí $f \xrightarrow[*]{\mathbf{R}} 0$*
(3) *0 je jediný terminál v I*

Důkaz. (1) $\Rightarrow$ (2) Nechť $f \xleftrightarrow[*]{\mathbf{R}} 0$, pak existuje polynom u takový, že $f \overset{*}{\rightarrow} u \overset{*}{\leftarrow} 0$. Ale 0 je terminál v každé bázi $\mathbf{R}$ a proto $u = 0$.

(2) $\Rightarrow$ (3) Protože $f \xrightarrow[*]{\mathbf{R}} 0$, je buď $f = 0$, nebo f není terminál.

(3) $\Rightarrow$ (1) Nechť $\bar{p} \xleftarrow[*]{\mathbf{R}} p \xleftrightarrow[*]{\mathbf{R}} q \xrightarrow[*]{\mathbf{R}} \bar{q}$, kde $\bar{p}, \bar{q}$ jsou terminály. Chceme zjistit, zda $\bar{p} = \bar{q}$. Protože $\bar{p} \xleftrightarrow[*]{\mathbf{R}} \bar{q}$, platí podle Věty 23.2 $\bar{p} - \bar{q} \in I$. Ale $\bar{p} - \bar{q}$ je rovněž terminál, neboť $\bar{p}$ ani $\bar{q}$ nelze přepsat. Tedy podle (3) platí $\bar{p} - \bar{q} = 0$. □

Tvrzení 25.2. *Nechť $\mathbf{R}$ je Gröbnerova báze ideálu I a nechť $p, q \in \mathbf{R}$. Pak*
(1) *jestliže $\mathrm{lt}(p) | \mathrm{lt}(q)$, pak $\mathbf{R} \setminus \{q\}$ je Gröbnerova báze ideálu I,*
(2) *jestliže $p \overset{q}{\rightarrow} \bar{p}$, pak $(\mathbf{R} \setminus \{p\}) \cup \{\bar{p}\}$ je Gröbnerova báze ideálu I*

Důkaz. (1) Je zřejmé, že $\mathbf{R} \setminus \{q\}$ je Gröbnerova báze, neboť všechny S-polynomy se redukují na 0. Zbývá ověřit, že $I = \langle \mathbf{R} \setminus \{q\} \rangle$. K tomu stačí dokázat, že $q \in \langle \mathbf{R} \setminus \{q\} \rangle$, ekvivalentně $q \xrightarrow[*]{\mathbf{R} \setminus \{q\}} 0$. Protože $q \overset{p}{\rightarrow} \bar{q} \in I$ a $\mathbf{R}$ je Gröbnerova báze, máme $\bar{q} \xrightarrow[*]{\mathbf{R}} 0$. Ale $\bar{q} \ll q$, tedy $\bar{q} \xrightarrow[*]{\mathbf{R} \setminus \{q\}} 0$. Proto $q \in \langle \mathbf{R} \rangle \setminus \{q\}$.

(2) Označme si $\mathbf{S} := (\mathbf{R} \setminus \{p\}) \cup \{\bar{p}\}$. Jelikož $p \xrightarrow{q} \bar{p}$, platí $p - \bar{p} = a \cdot q$ pro nějaké a. Protože $q \in \mathbf{R}$ a rovněž $q \in \mathbf{S}$, máme $\bar{p} \in \langle \mathbf{R} \rangle$ a $p \in \langle \mathbf{S} \rangle$. Tedy $\langle \mathbf{R} \rangle = \langle \mathbf{S} \rangle$.

Zbývá ověřit, že $\mathbf{S}$ je Gröbnerova báze. Nechť $r \neq 0$ je terminál v $\mathbf{S}$. Polynom r nemůže být podle Tvrzení 25.1 terminálem v $\mathbf{R}$. Nyní $r \xrightarrow{p} \bar{r}$, ale pak $r \xrightarrow{\bar{p}}$, a tedy r se přepisuje rovněž v $\mathbf{S}$, protože $\bar{p} \in \mathbf{S}$. To je spor s předpokladem pro r.

□

Definice. Řekneme, že $\mathbf{R}$ je *redukovaná* Gröbnerova báze, jestliže pro všechny prvky $p, q \in \mathbf{R}$ platí $p \overset{q}{\not\to} \bar{p}$.

Řekneme, že $\mathbf{R}$ je *normovaná* Gröbnerova báze, jestliže pro všechny $p \in \mathbf{R}$ platí $\mathrm{lc}(p) = 1$.

Věta 25.3. *Pro každý ideál v $\mathbf{T}[x_1, \dots, x_k]$ existuje právě jedna normovaná redukovaná Gröbnerova báze vzhledem k danému přípustnému uspořádání.*

Důkaz. Existence redukované normované Gröbnerovy báze plyne z Buchbergerova algoritmu a předchozího tvrzení.

Zbývá dokázat jednoznačnost. Uvažujme dvě normované redukované báze $\mathbf{R}_1, \mathbf{R}_2$ v daném uspořádání takové, že $I = \langle \mathbf{R}_1 \rangle = \langle \mathbf{R}_2 \rangle$. Nechť $\mathbf{R}_1 = \{p_1, \dots, p_n\}$ a $\mathbf{R}_2 = \{q_1, \dots, q_m\}$. Protože $\langle \mathbf{R}_1 \rangle = \langle \mathbf{R}_2 \rangle$, platí $p_1 \xrightarrow[*]{\mathbf{R}_2} 0$, tedy existuje i, pro které r_1 lze přepsat pomocí q_i. Bez újmy na obecnosti můžeme předpokládat $i = 1$, tedy $\mathrm{lt}(q_1) | \mathrm{lt}(p_1)$. Analogicky $q_1 \xrightarrow[*]{\mathbf{R}_1} 0$, tedy existuje k, pro něž $\mathrm{lt}(p_k) | \mathrm{lt}(q_1)$. Protože $\mathrm{lt}(p_k) | \mathrm{lt}(q_1)$ a $\mathrm{lt}(q_1) | \mathrm{lt}(p_1)$, platí $\mathrm{lt}(p_k) | \mathrm{lt}(p_1)$. Protože $p_1, p_k \in \mathbf{R}_1$ a $\mathbf{R}_1$ je redukovaná báze, musí být $k = 1$. Potom ale $\mathrm{lt}(p_1) | \mathrm{lt}(q_1)$ a $\mathrm{lt}(q_1) | \mathrm{lt}(p_1)$ a tedy $p_1 = q_1$. Analogickým postupem dostaneme $m = n$ a $\mathrm{lt}(p_i) = \mathrm{lt}(q_i)$ pro všechna i.

Zbývá dokázat, že $p_i = q_i$ pro všechna i. Platí $p_i - q_i \in I$, neboť $p_i, q_i \in I$. Všimněme si, že $p_i - q_i$ je terminál v $\mathbf{R}_2$ (také v $\mathbf{R}_1$), neboť jej nelze přepsat pomocí q_i ($\mathrm{lt}(p_i) = \mathrm{lt}(q_i)$, tedy $p_i - q_i$ neobsahuje původní vedoucí člen). Dále $p_i - q_i$ nelze přepsat pomocí q_j pro žádné $j \neq i$, jinak báze $\mathbf{R}_2$ není redukovaná. Kdyby polynom p_i bylo možno přepsat pomocí q_j, tak by se také polynom p_i dal přepsat pomocí p_j, spor s redukovaností $\mathbf{R}_1$. Tedy $p_i - q_i$ je terminál a protože $\mathbf{R}_1$ i $\mathbf{R}_2$ jsou Gröbnerovy báze, máme $p_i - q_i = 0$ pro všechna i. Tedy $\mathbf{R}_1 = \mathbf{R}_2$. □

Úloha (výpočetně náročné)**.** Najděte redukovanou Gröbnerovu bázi pro ideál $\langle xz - 3x^2 + x + 6x^3 + 1, y^2 + x^2 - 2, x^5 - 6x^3 + x^2 - 1 \rangle$ v $\mathbb{Q}[x, y, z]$ v uspořádání $<_{\mathrm{LEX}}$, $x < y < z$.

Úloha (výpočetně náročné)**.** Najděte redukovanou Gröbnerovu bázi pro ideál $\langle x^3 y - 2y^2 - 1, x^2 y^2 + x + y \rangle$ v $\mathbb{Q}[x, y]$ v uspořádání a) $<_{\mathrm{GLEX}}$, $x > y$ b) $<_{\mathrm{LEX}}$, $x > y$

26. Aplikace Gröbnerových bází

26.1. **Náležení a rovnost ideálů.**

Algoritmus 26.1 (náležení do ideálu).
`VSTUP:` $f, f_1 \dots, f_m \in \mathbf{T}[x_1, \dots, x_k]$
`VÝSTUP:` $f \overset{?}{\in} \langle f_1, \dots, f_m \rangle$
`1.` spočti nějakou Gröbnerovu bázi $\mathbf{R}$ ideálu $\langle f_1, \dots, f_m \rangle$
`2.` zjistěte, zda $f \underset{*}{\overset{\mathbf{R}}{\to}} 0$

Algoritmus 26.2 (rovnost ideálů).
`VSTUP:` $f_1, \dots, f_m, g_1, \dots, g_n \in \mathbf{T}[x_1, \dots, x_k]$
`VÝSTUP:` $\langle f_1, \dots, f_m \rangle \overset{?}{=} \langle g_1, \dots, g_n \rangle$
`1.` spočti normovanou redukovanou bázi $\mathbf{R}$ ideálu $\langle f_1, \dots, f_m \rangle$
`2.` spočti normovanou redukovanou bázi $\mathbf{S}$ ideálu $\langle g_1, \dots, g_n \rangle$
`3.` zjisti, zda $\mathbf{R} = \mathbf{S}$

26.2. **Náležení a rovnost radikálů.** Abychom si vysvětlili pojem *radikál*, uděláme odbočku do algebraické geometrie.

Definice. Nechť $\mathbf{R} \subseteq \mathbf{T}[x_1, \dots, x_k]$. Pak definujeme $V(\mathbf{R}) = \{(a_1, \dots, a_k) \in \mathbf{T}^k : f(a_1, \dots, a_k) = 0 \text{ pro všechna } f \in \mathbf{R}\}$.

Nechť $X \subseteq \mathbf{T}^k$. Pak definujeme $I(X) := \{f \in \mathbf{T}[x_1, \dots, x_k] : f(a_1, \dots, a_k) = 0 \text{ pro všechny } \bar{a} \in X\}$.

I je zřejmě ideál. Navíc I je *radikál*, t.j. ideál takový, že jestliže $g^m \in I$, pak $g \in I$.

Množiny tvaru $V(\mathbf{R})$ se nazývají *algebraické*. Pro $\dim V(\mathbf{R}) = 1$ mluvíme o *algebraických křivkách*.

Příklad. Kružnice lze popsat jako $V(x^2 + y^2 - r^2)$. Přímku můžeme popsat množinou $V(ax + by + c)$.

Algebraická geometrie se zabývá studiem algebraických křivek přes odpovídající ideály polynomů. Má uplatnění v aplikacích teorie čísel, kryptografii,.... Gröbnerovy báze nám dávají nástroj na počítání s algebraickými množinami.

Definice. Definujeme $\mathrm{Rad}(I) := \{g : \text{ existuje } m \text{ t.ž. } g^m \in I\}$.

Pozorování 26.3. $\mathrm{Rad}(I)$ *je nejmenší radikál obsahující ideál* I.

Věta 26.4 (Hilbertova věta o nulách). *Nechť* $\mathbf{T}$ *je algebraicky uzavřené těleso a* $\mathbf{R} \subseteq \mathbf{T}[x_1, \dots, x_k]$. *Pak* $\mathrm{Rad}(\langle \mathbf{R} \rangle) = I(V(\mathbf{R}))$.

Důkaz. Nebude (viz. Komutativní okruhy). □

Na předchozí větu se lze dívat také následujícím způsobem. $\mathrm{Rad}(\langle \mathbf{R} \rangle)$ je nejmenší radikál obsahující $\mathbf{R}$. Dále $I(V(\mathbf{R})) = \{g \in \mathbf{T}[x_1, \dots, x_k] : \text{ jestliže } f(\bar{a}) = 0 \text{ pro všechna } f \in \mathbf{R} \text{ pak } g(\bar{a}) = 0\}$, tedy $I(V(\mathbf{R}))$ je množina všech polynomů, které mají za kořen všechny $\bar{a} \in V(\mathbf{R})$. Přitom víme, že $V(\mathbf{R})$ je množina všech společných kořenů pro $f \in \mathbf{R}$.

Toto pozorování lze uplatnit v úlohách analytické geometrie, kde na vstupu dostaneme $f, f_1, \dots, f_m \in \mathbf{T}[x_1, \dots, x_k]$ a na výstupu se dozvíme, jestli platí $f \in I(V(f_1, \dots, f_m))$.

Tvrzení 26.5. *Nechť* $\mathbf{T}$ *je algebraicky uzavřené těleso. Pak* $f \in I(V(f_1, \dots, f_m))$ *právě tehdy, když* $1 \in \langle f_1, \dots, f_m, zf - 1 \rangle$, *kde* z *je nová proměnná různá od* $x_1, \dots, x_k$ *(je to tedy ideál v* $\mathbf{T}[x_1, \dots, x_k, z]$*).*

Poznámka. Obecně platí jen implikace $\Leftarrow$, v algebraicky uzavřeném tělese platí ekvivalence.

Problém se tedy převádí na náležení do ideálu.

Důkaz. Tvrzení dokážeme ve dvou krocích, ve kterých dokážeme dvě pomocná tvrzení.

(1) Nejprve dokážeme, že $f \in I(V(f_1, \ldots, f_m))$ právě tehdy, když $f_1, \ldots, f_m, zf-1$ nemají společný kořen. K důkazu nepotřebujeme algebraickou uzavřenost tělesa $\mathbf{T}$.

($\Rightarrow$) Nechť $(a_1, \ldots, a_m, b)$ je společný kořen polynomů $f_1, \ldots, f_m, zf-1$ a nechť $\bar{a} := (a_1, \ldots, a_m)$. Pak $f_1(\bar{a}) = f_2(\bar{a}) = \ldots = f_m(\bar{a}) = 0$, tedy i $f(\bar{a}) = 0$. Proto $(zf-1)(\bar{a}, b) = b \cdot f(\bar{a}) - 1 = b \cdot 0 - 1 = -1$. To znamená, že $(\bar{a}, b)$ není kořenem $zf-1$, což je spor s předpokladem.

($\Leftarrow$) Nechť $\bar{a}$ je společný kořen $f_1, \ldots, f_m$, tedy $f_1(\bar{a}) = f_2(\bar{a}) = \ldots = f_m(\bar{a}) = 0$. Chceme dokázat, že i $f(\bar{a}) = 0$. Nechť $(\bar{a}, b)$ není kořen $zf-1$ pro žádné b, t.j. $b \cdot f(\bar{a}) - 1 \neq 0$ pro všechna b. Ale pokud $f(\bar{a}) \neq 0$, pak rovnost nastává pro $b = \frac{1}{f(\bar{a})}$, což je spor s předpokladem neexistence společného kořene. Proto $f(\bar{a}) = 0$.

(2) Nyní ukážeme následující tvrzení. $g_1, \ldots, g_m$ nemají společný kořen právě tehdy, když $1 \in \langle g_1, \ldots, g_m \rangle$. Pro důkaz implikace $\Rightarrow$ budeme potřebovat algebraickou uzavřenost tělesa $\mathbf{T}$.

($\Leftarrow$) Nechť $1 = \sum_{i=1}^{m} r_i g_i$ pro nějaké $r_i \in \mathbf{T}[x_1, \ldots, x_k]$. Kdyby $\bar{a}$ byl společný kořen polynomů $g_1, \ldots, g_m$, pak platí $0 = \sum_{i=1}^{m} r_i(\bar{a}) g_i(\bar{a}) \neq 1$, spor.

($\Rightarrow$) Nechť $g_1, \ldots, g_m$ nemají společný kořen, tedy $V(g_1, \ldots, g_m) = \emptyset$. Potom $I(V(g_1, \ldots, g_m)) = \mathbf{T}[x_1, \ldots, x_k]$ a $1 \in \mathbf{T}[x_1, \ldots, x_k]$. Protože $\mathbf{T}[x_1, \ldots, x_k]$ je algebraicky uzavřené, z Hilbertovy věty o nulách vyplývá $I(V(g_1, \ldots, g_m)) = \mathrm{Rad}(\langle g_1, \ldots, g_m \rangle)$. Víme, že $g \in \mathrm{Rad}(I)$ právě tehdy, když existuje m takové, že $g^m \in I$. Protože $1 \in \mathrm{Rad}(\langle g_1, \ldots, g_m \rangle)$, existuje m takové, že $1 = 1^m \in \langle g_1, \ldots, g_m \rangle$.

□

Pomocí předchozí věty můžeme nyní zformulovat algoritmus náležení radikálu pro algebraicky uzavřené těleso $\mathbf{T}$.

Algoritmus 26.6 (náležení radikálu pro algebraicky uzavřené těleso **T**).

`VSTUP:` $f, f_1, \ldots, f_m \in \mathbf{T}[x_1, \ldots, x_k]$

`VÝSTUP:` $f \stackrel{?}{\in} \mathrm{Rad}(\langle f_1, \ldots, f_m \rangle) = I(V(f_1, \ldots, f_m))$

`1.` spočti normovanou redukovanou Gröbnerovu bázi $\mathbf{R}$ pro $\langle f_1, \ldots, f_m, zf-1 \rangle$, kde z je nová proměnná

`2.` zjisti, zda $1 \in \mathbf{R}$

Pro obecné těleso máme následující částečný algoritmus pro náležení $I(V(f_1, \ldots, f_m))$. Kroky `1.`,`2.` proběhnou stejně. Jestliže $1 \in \mathbf{R}$, pak $f \in I(V(f_1, \ldots, f_m))$. Jestliže $1 \notin \mathbf{R}$, pak nemůžeme nic říct. Je teotiž možné, že každý společný kořen $f_1, \ldots, f_m$ v tělese $\mathbf{T}$ je rovněž kořenem f, ale může se stát, že existuje společný kořen $f_1, \ldots, f_m$ v algebraicky uzavřeném tělese, který f nemá.

Podobně můžeme zformulovat algoritmus pro rovnost radikálů (resp. $I(V(\mathbf{R}))$).

Algoritmus 26.7 (rovnost radikálů)**.**

`VSTUP:` $f_1, \dots, f_m, g_1, \dots, g_n$

`VÝSTUP:` $\mathrm{Rad}(\langle f_1, \dots, f_m\rangle \stackrel{?}{=} \mathrm{Rad}(\langle g_1, \dots, g_n\rangle))$

`1.` pro všechna $i = 1, \dots, m$ ověř, zda $f_i \in \mathrm{Rad}(\langle g_1, \dots, g_n\rangle)$

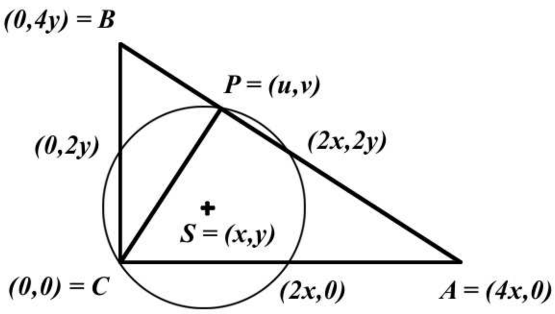
`2.` pro všechna $j = 1, \dots, n$ ověř, zda $g_j \in \mathrm{Rad}(\langle f_1, \dots, f_m\rangle)$

`3.` jestliže bylo odpovězeno vždy ano, pak odpověz `ANO`, jestli bylo odpovězeno aspoň jednou ne, pak v algebraicky uzavřeném tělese odpověz `NE`, v obecném tělese odpověz `NEVÍM`

Příklad. V $\mathbb{Q}[x]$ mějme $f_1 = (x-1)(x^2-2)$ a $f = x-1$. Zjistěte, zda $f \in I(V(f_1))$.

Hned vidíme, že $V(f_1) = \{1\}$. Použitím algoritmu chceme zjistit, zda $1 \in \langle x^3 - x^2 - 2x + 2, xz - z - 1\rangle$. Algoritmus odpoví NE (nemůže odpovědět ANO, celý výpočet by bylo možné provést stejně nad $\mathbb{C}$). Gröbnerova báze vyjde $z - x - 1, x^2 - 2$.

Příklad. Vraťme se nyní k naší motivační úloze z úvodu kapitoly o Gröbnerových bazích. Chtěli jsme dokázat, že v pravoúhlém trojúhelníku leží vrchol C, středy stran AB, AC, BC a pata výšky na stranu AB na kružnici. Označme si body trojúhelníku podle obrázku.



Víme, že

- strana PC je kolmá na stranu AB, což můžeme zapsat rovnicí $f_1 = xu - yv = 0$,
- bod P náleží straně AB, což můžeme zapsat rovnicí $f_2 = yu + xv - 4xy = 0$,

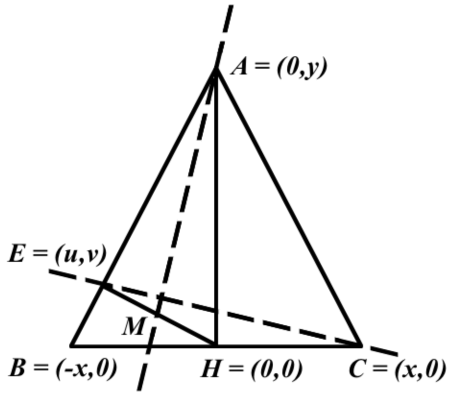
Nyní chceme ukázat, že $|PS| = |SC|$, neboli $(u-x)^2 + (v-y)^2 = x^2 + y^2$. Po rozepsání dostaneme podmínku $f = u^2 - 2xu - 2yv + v^2 = 0$.

Problém převedeme na problém náležení do radikálu. Chceme zjistit, zda libovolný kořen (a, b, c, d) polynomů f_1, f_2 je rovněž kořenem f. Chceme tedy zjistit, zda $f \in I(V(f_1, f_2))$. K tomu stačí ověřit, že $1 \in \langle f_1, f_2, zf - 1\rangle$. Výpočtem dostáváme Gröbnerovu bázi $x, y, z(u^2 + v^2) - 1$. Vidíme, že 1 do nalezené báze nepatří.

V čem je tedy problém? Pokud $x = y = 0$, tak rovnice $f_1 = 0$ $f_2 = 0$ splňují libovolné u, v, ale ne rovnici $f = 0$. Předpokládejme tedy, že $x, y \neq 0$. Jak to ale do naší úlohy zabudovat? Všimněme si, že $g(\bar{a}) \neq 0$ právě tehdy, když $wg(\bar{a}) - 1$ má kořen. Podmínku $g \neq 0$ tedy nahradíme za $wg - 1$ a do báze přidáme polynom $w(x^2 + y^2) - 1$, kde w je nová proměnná. Nyní již zjistíme, že $1 \in \langle f_1, f_2, w(x^2 + y^2) - 1, zf - 1\rangle$.

Příklad. Nechť ABC je rovnoramenný trojúhelník t.ž. $|AB| = |AC|$. Nechť H je pata výšky na stranu BC, E je kolmý průmět bodu H stranu AB, M je střed strany EH. Dokažte, že přímka EC je kolmá na přímku AM.

Situaci ilustruje následující obrázek



Bod M má souřadnice $(\frac{u}{2}, \frac{v}{2})$. Sestavíme si rovnice báze ideálu

- bod E patří straně AB, což můžeme zapsat rovnicí $f_1 = -yu + xv - xy = 0$
- strana EH je kolmá na stranu AB, což můžeme zapsat rovnicí $f_2 = xu + yv = 0$

Chceme dokázat, že přímka EC je kolmá na přímku AM, což lze zapsat rovnicí $f = \frac{u}{2}(u-x) + v(\frac{v}{2} - y) = 0$. Po úpravě dostáváme rovnici $f = u^2 - 2xu + v^2 - 2yv = 0$.

Chceme tedy dokázat, že $f \in I(V(f_1, f_2))$. K tomu stačí dokázat, že $1 \in \langle f_1, f_2, zf - 1\rangle$. To bohužel není pravda ze stejného důvodu v předchozím příkladu. Pokud $x = y = 0$, pak libovolné u, v jsou kořenem f_1, f_2, ale nikoliv f. Analogicky jako v předchozím příkladu přidáme podmínku $x^2 + y^2 \neq 0$. Zjistíme, že $1 \in \langle f_1, f_2, w(x^2 + y^2) - 1, zf - 1\rangle$.

26.3. **Soustavy polynomiálních rovnic.** Metoda Gröbnerových bazí je zobecněním jak Eukleidova algoritmu, tak Gaussovy eliminace.

Na základě tohoto pozorování můžeme sestavit algoritmus, na jehož vstupu budou polynomy $f_1, \ldots, f_m \in \mathbf{T}[x_1, \ldots, x_k]$ a na výstupu budou prvky $\bar{a} \in \mathbf{T}^k$ takové, že $f_i(\bar{a}) = 0$ pro všechna i, případně říct, kolik jich je a vypsat je. Definujme si $J := \langle f_1, \ldots, f_m\rangle$ a nechť $\mathbf{R}$ je nějaká normovaná Gröbnerova báze J.

Pozorování 26.8. *$V(\mathbf{R}) = V(J) = V(f_1, \ldots, f_m)$ je množina řešení soustavy $f_i(\bar{a}) = 0$.*

Lemma 26.9 (Eliminační lemma)**.** *Nechť $\mathbf{R}$ je Gröbnerova báze ideálu I vzhledem k uspořádání $<_{\mathrm{LEX}}$, $x_1 < x_2 < x_2 < \ldots < x_k$. Pak $I \cap \mathbf{T}[x_1, \ldots, x_i] = \langle \mathbf{R} \cap \mathbf{T}[x_1, \ldots, x_i]\rangle$ pro všechna i.*

Důkaz. ($\supseteq$) Triviální.

($\subseteq$) Nechť $f \in I \cap \mathbf{T}[x_1, \ldots, x_i]$, tedy $f \xrightarrow[*]{\mathbf{R}} 0$. $x_{i+1}, \ldots, x_k$ se v průběhu toho přepisování nevyskytují, neboť $x_1 < \ldots < x_i < x_{i+1} < \ldots < x_k$. Proto $f \xrightarrow[*]{\mathbf{R} \cap \mathbf{T}[x_1, \ldots, x_i]} 0$ a tedy $\mathbf{R} \cap \mathbf{T}[x_1, \ldots, x_i]$ je Gröbnerova báze pro $I \cap \mathbf{T}[x_1, \ldots, x_i]$. $\square$

Právě dokázané lemma nám říká, že řešení pro $x_1, \ldots, x_i$ lze vyjádřit z rovnic v $x_1, \ldots, x_i$.

Příklad. Mějme soustavu rovnic

$$x^2y^2 + y - 1 = 0$$
$$x^2y + x = 0$$

V jednom z předchozích příkladů jsme ukázali, že Gröbnerova báze ideálu generovaného těmito rovnicemi je $\{x, y-1\}$. Tedy jediné řešení této soustavy je $(0,1)$.

Příklad. Mějme soustavu rovnic

$$xy^2 + x^2 + x = 0$$
$$x^2y + x = 0$$

V jednom z předchozích příkladů jsme ukázali, že Gröbnerova báze této soustavy v uspořádání $<_{\mathrm{LEX}}$, $y > x$ je $\{xy - x^3 - x^2, x^4 + x^3 + x\}$. Dostali jsme sice nekonečně mnoho řešení v $\mathbb{C}$, ale dobře popsatelných: Z první rovnice máme pokud $x = 0$ tak y může být libovolné a pokud $x \neq 0$, pak $y = x^2 + x$. Z druhé rovnice dostáváme čtyři kořeny pro x. a to $0, \alpha, \bar{\alpha}, \beta$. Všechna řešení jsou tedy: $(0, y)$ pro všechna y, $(\alpha, \alpha^2 + \alpha)$, $(\bar{\alpha}, \bar{\alpha}^2 + \bar{\alpha})$, $(\beta, \beta^2 + \beta)$.

Kdybychom použili uspořádání $<_{\mathrm{LEX}}$, $y < x$, dostali bychom Gröbnerovu bázi $\{xy^3 + xy - x, xy^2 + x^2 + x\}$, ze které nejsou žádná řešení vidět.

Tvrzení 26.10. *Nechť* $\mathbf{T}$ *je algebraicky uzavřené těleso. Pak* $V(\mathbf{R}) = \emptyset$ *právě tehdy, když* $1 \in \mathbf{R}$.

Důkaz. ($\Leftarrow$) Jestliže $1 \in \mathbf{R}$, pak $\langle \mathbf{R} \rangle = T[x_1, \ldots, x_k]$ a tedy $V(\mathbf{R}) = V(\langle \mathbf{R} \rangle) = \emptyset$

($\Rightarrow$) Jestliže $V(\mathbf{R}) = \emptyset$, pak $I(V(\mathbf{R})) = \mathbf{T}[x_1, \ldots, x_k] \ni 1$. Podle Hilbertovy věty o nulách platí $\mathrm{Rad}(\langle \mathbf{R} \rangle) = I(V(\mathbf{R}))$. Tedy existuje m takové, že $1 = 1^m \in \langle \mathbf{R} \rangle$. □

Poznámka. Předchozí tvrzení nám dává algoritmus na zjištění, zda má daná soustava nad algebraicky uzavřeným tělesem řešení.

Tvrzení 26.11. *Nechť* $\mathbf{T}$ *je algebraicky uzavřené těleso. Pak* $V(\mathbf{R})$ *konečná právě tehdy, když pro všechna* $i = 1, \ldots, k$ *existuje* $g_i \in \mathbf{R}$ *t.ž.* $\mathrm{lt}(g_i) = x_i^{k_i}$.

Důkaz. Nebude. □

Poznámka. Jako důsledek Eliminačního lemmatu navíc platí: Je-li $\mathbf{R}$ redukovaná v uspořádání $<_{\mathrm{LEX}}$, kde $x_1 < x_2 < \ldots$ a je-li $V(\mathbf{R})$ konečná, pak výsledek je ve tvaru $g_1(x_1), g_2(x_1, x_2), \ldots, g_k(x_1, \ldots, x_k)$, přičemž z 1.rovnice můžeme vyjádřit x_1, z 2.rovice pak x_2, ...

Algebraická uzavřenost tělesa je nutná, protože Matijasevič někdy v 70. letech dokázal, že neexistuje algoritmus, který by určil, zda má rovnice $p(x_1, \ldots, x_n) = 0$ (polynom nad $\mathbb{Z}$) řešení v $\mathbb{Z}$ (a tedy ani v $\mathbb{Q}$). Proto nemůže existovat algoritmus, jak řešení nad algebraicky uzavřeným tělesem efektivně vyjádřit, jinak by stačilo otestovat, zda jsou některá z nich v $\mathbb{Z}$ (v $\mathbb{Q}$)!

Příklad. Mějme soustavu

$$4xz - 4xy^2 - 16x^2 - 1 = 0$$
$$2y^2z + 4x + 1 = 0$$
$$2x^2z + 2y^2 + x = 0$$

Tato soustava nemá řešení v $\mathbb{Q}$, ale má řešení v $\mathbb{C}$.

Příklad. Mějme soustavu

$$\begin{aligned} xz + yx - x + z^2 - 2 &= 0 \\ xy^2 + 2xz - 3x + y + z - 1 &= 0 \\ 2z^2 + 2y^2 - 3z + 2zy + y^3 - 3y &= 0 \end{aligned}$$

Tato soustava nemá řešení v $\mathbb{Q}$, ovšem má nekonečně mnoho řešení v $\mathbb{C}$. Dokonce při uspořádání $x < y < z$ je lze dobře vyjádřit.

Obsah