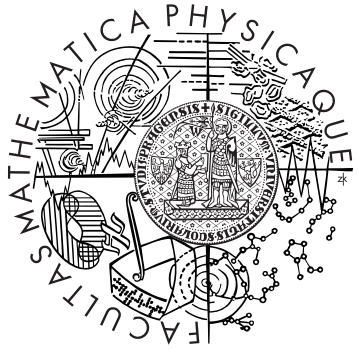
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jana Divišová

Mříže a faktorizace celočíselných polynomů

Katedra algebry

Vedoucí bakalářské práce:
RNDr. David Stanovský, Ph.D.

Studijní program:
Matematika
Matematické metody informační bezpečnosti

2007

Na tomto místě bych velmi ráda poděkovala svému vedoucímu bakalářské práce RNDr. Davidovi Stanovskému, Ph.D. za pomoc a cenné rady, kterými mi pomohl. Dále bych ráda poděkovala Ivanu Štubňovi za korektury textu a za podnětné připomínky k implementaci algoritmů.

Prohlašuji, že jsem svou bakalářskou práci napsala samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 2. srpna 2007 Jana Divišová

Obsah

1 Úvod **5**

2 Bezčtvercová faktorizace **7**

3 Berlekamp–Henselův algoritmus **9**

3.1 Henselův algoritmus 9

3.2 Berlekampův algoritmus 12

3.3 Berlekamp–Henselův algoritmus 15

4 Lenstra–Lenstra–Lovászův algoritmus **18**

4.1 Úvod do teorie mříží 18

4.2 Redukce báze 20

4.3 Faktorizace a mříže 25

4.4 Faktorizační algoritmus 30

5 Testy **35**

6 Závěr **37**

Přílohy **39**

Název práce: Mříže a faktorizace celočíselných polynomů
Autor: Jana Divišová
Katedra: Katedra algebry
Vedoucí bakalářské práce: RNDr. David Stanovský, Ph.D.
e-mail vedoucího: stanovsk@karlin.mff.cuni.cz

Abstrakt: V předložené práci se věnujeme faktorizaci celočíselných polynomů. Podrobně tu popisujeme dva algoritmy, které daný problém řeší. Jmenovitě se jedná o Berlekamp–Henselův algoritmus a Lenstra–Lenstra–Lovászův algoritmus. První z nich pracuje s teoreticky exponenciální časovou složitostí. Druhý algoritmus pracuje na podobném principu jako první, avšak využívá poznatky z teorie mříží k docílení polynomiální časové složitosti. Součástí práce je i shrnutí základních poznatků z teorie mříží, konkrétně algoritmus na redukci báze mřížky. Cílem práce je porovnat jak teoretickou tak i praktickou časovou náročnost těchto dvou algoritmů.

Klíčová slova: Faktorizace celočíselných polynomů, Berlekamp–Henselův algoritmus, Lenstra–Lenstra–Lovászův algoritmus, redukce báze mřížky

Title: Lattices and factorization of integer polynomials
Author: Jana Divišová
Department: The Department of Algebra
Supervisor: RNDr. David Stanovský, Ph.D.
Supervisor's e-mail address: stanovsk@karlin.mff.cuni.cz

Abstract: The aim of this work is to study factorization of integer polynomials. We describe in detail two algorithms, which solve this problem. Namely we discuss Berlekamp–Hensel algorithm and Lenstra–Lenstra–Lovász algorithm. The first one works theoretically in exponential time complexity. The second one works in a similar way, but it uses piece of knowledge of lattices to achieve polynomial time complexity. One part of the work is also recapitulation of theory of lattices, concretely lattice basis reduction algorithm. The goal of this work is to compare both theoretical and practical efficiency of these two algorithms.

Keywords: Factorization of integer polynomials, Berlekamp–Hensel algorithm, Lenstra–Lenstra–Lovász algorithm, lattice basis reduction

Kapitola 1

Úvod

Faktorizace polynomu nad celými čísly, neboli rozklad polynomu na součin ireducibilních činitelů, je následující problém: vstupem je polynom $f \in \mathbb{Z}[x]$, výstupem jsou po dvou neasociované ireducibilní polynomy $a_1, \ldots, a_m \in \mathbb{Z}[x]$ a nezáporná celá čísla $n_1, \ldots, n_m$ 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.

V této práci představíme dva algoritmy, které se tímto problémem zabývají a to Berlekamp–Henselův algoritmus a Lenstra–Lenstra–Lovászův algoritmus. Rozdíl mezi těmito algoritmy spočívá v jejich složitosti. Zatímco Berlekamp–Henselův algoritmus pracuje s teoreticky exponenciální složitostí, Lenstra–Lenstra–Lovászův pracuje v polynomiálním čase.

Oba algoritmy fungují na podobném principu. Základní kroky obou algoritmů můžeme shrnout v těchto bodech:

- Jako vstup se vždy předpokládá bezčtvercový polynom. Proto se v první kapitole věnujeme algoritmu na bezčtvercovou faktorizaci.
- V dalším kroku se používá Berlekampův algoritmus pro rozklad polynomu na ireducibilní činitele nad konečným tělesem.
- Tento rozklad se pak "naliftuje" pomocí Henselova algoritmu.
- V posledním kroku používá Berlekamp–Henselův algoritmus hrubou sílu na rekonstrukci ireducibilních faktorů v $\mathbb{Z}[x]$. Právě zde se Lenstra–Lenstra–Lovászův algoritmus odklání a používá poznatky z teorie mříží ke snížení časové náročnosti tohoto kroku.

Vzhledem k tomu, že bezčtvercová faktorizace i Berlekamp–Henselův algoritmus jsou součástí základního kurzu na Matematicko–fyzikální fakultě na Karlově univerzitě, jsou v této práci zmíněny pouze jako přehled, tj. uvedeny bez důkazů. Druhá část práce je věnována podrobnému popisu Lenstra–Lenstra–Lovászova algoritmu. Tato část obsahuje i základní poznatky z teorie mříží. Poslední část práce je věnována testování obou algoritmů na náhodných vstupech. K tomuto testování byla využita volně dostupná knihovna NTL pro práci s polynomy nad celými čísly v jazyku C++.

Kapitola 2

Bezčtvercová faktorizace

Většina algoritmů na faktorizaci vyžaduje, aby byl vstupní polynom f bezčtvercový. V této kapitole předvedeme algoritmus, který libovolný polynom rozloží na součin bezčtvercových polynomů (ne nutně ireducibilních).

Nejprve si zadefinujeme bezčtvercovost jako takovou.

Definice 1
Libovolný 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 $n_1 = \ldots = n_m = 1$. *Bezčtvercovým rozkladem* polynomu f rozumíme po dvou nesoudělné bezčtvercové polynomy $g_1, \ldots, g_k$ splňující

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

(tedy g_i je dáno součinem právě těch ireducibilních činitelů, které se v f vyskytují v i-té mocnině).

Věta 2.0.1
Nechť $\mathbb{T}$ je těleso a $0 \neq f \in \mathbb{T}[x]$. Pak

1. *f je bezčtvercový právě tehdy, když* $\mathrm{NSD}(f, f') = 1$
2. *jestliže* $\mathrm{char}(\mathbb{T}) = 0$ *a* $f = \prod_{i=1}^{k} g_i^i$ *je bezčtvercový rozklad, pak*

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

Právě vyslovenou Větu 2.0.1 použijeme v následujícím algoritmu. Nejprve si

předvedeme jednodušší verzi pro speciální případ rozkladu nad tělesem charakteristiky 0, poté o něco složitější pro případ tělesa nenulové charakteristiky p.

Algoritmus 2.0.2 (BezctvercovaFaktorizaceChar0)

Vstup: $f \in \mathbb{T}[x]$, předpokládáme $\mathrm{char}(\mathbb{T}) = 0$
Výstup: bezčtvercový rozklad $g_1, \ldots, g_k$ polynomu f

0. $f_0 := f$.
i. $f_i := \mathrm{NSD}(f_{i-1}, f'_{i-1})$ pro $i = 1, \ldots, k$,
$h_i := \frac{f_{i-1}}{f_i}$,
if $i \geq 2$ then $g_{i-1} := \frac{h_{i-1}}{h_i}$,
if $f_i = 1$ then return $g_1, \ldots, g_{i-1}, g_i = f_{i-1}$, stop.

Tvrzení 2.0.3
Časová složitost Algoritmu 2.0.2 (BezctvercovaFaktorizaceChar0) je $\mathcal{O}(n^3)$, kde $n = \deg(f)$, přičemž za jednotku výpočtu bereme operaci v okruhu koeficientů.

Algoritmus 2.0.4 (BezctvercovaFaktorizaceCharP)

Vstup: $f \in \mathbb{T}[x]$, předpokládáme $\mathrm{char}(\mathbb{T}) = p \neq 0$
Výstup: bezčtvercový rozklad $g_1, \ldots, g_k$ 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 $h_1, \ldots, h_k$ polynomu g,
return $\underbrace{1, \ldots, 1}_{p-1}, h_1, \underbrace{1, \ldots, 1}_{p-1}, h_2, \underbrace{1, \ldots, 1}_{p-1}, h_3, \ldots, h_k$.
4. if $1 \neq d \neq f$ then
polož $g := \frac{f}{d}$,
spočti bezčtvercový rozklad $h_1, \ldots, h_k$ polynomu d, polož $h_{i+1} := 1$,
pro $i = 1, \ldots, k$ 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 $h_1 \cdot g, h_2, \ldots, h_{k+1}$.

Tvrzení 2.0.5
Časová složitost Algoritmu 2.0.4 (BezctvercovaFaktorizaceCharP) je $\mathcal{O}(n^4)$, kde $n = \deg(f)$, přičemž za jednotku bereme operaci v okruhu koeficientů.

Kapitola 3

Berlekamp–Henselův algoritmus

První algoritmus na faktorizaci, který v této práci popíšeme je algoritmus modulární, který využívá Berlekampův algoritmus pro faktorizaci nad konečným tělesem charakteristiky p a Henselův algoritmus na "liftování" faktorů modulo p^k, kde k je vhodně zvolené přirozené číslo. Přesnější vysvětlení pojmu "liftování" bude objasněno v Henselově lemmatu 3.1.5. Poslední část Berlekamp–Henselova algoritmu sestává z rekonstrukce faktorů nad celými čísly.

3.1 Henselův algoritmus

V této podkapitole popíšeme Henselův algoritmus, který použijeme na "liftování" faktorů. Henselovu algoritmu na vstupu stačí rozklad modulo jediné prvočíslo p a to takové, aby nedělilo vedoucí člen vstupního polynomu f, a zároveň tak, aby $f \bmod p$ byl bezčtvercový polynom v $\mathbb{Z}_p[x]$. Pomocí Henselova algoritmu se tento rozklad "naliftuje" na rozklad modulo p^k. Připomeňme, 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

$$\sum_{i=0}^{m} |b_i| \leq 2^m \cdot \left| \frac{b_m}{a_n} \right| \cdot \sqrt{\sum_{i=0}^{n} a_i^2} \ .$$

Označíme-li tedy

$$\mathrm{LM}(f) = 2^n \cdot \sqrt{\sum_{i=0}^{n} a_i^2} \ ,$$

pak všechny koeficienty jakéhokoliv členu ireducibilního rozkladu polynomu f v $\mathbb{Z}[x]$ jsou v absolutní hodnotě menší než $\mathrm{LM}(f)$. Zvolíme-li k tak velké, abychom převýšili $\mathrm{LM}(f)$, dostaneme téměř rozklad nad celými čísly. Tento rozklad

nemusí být bohužel přesný, proto je potřeba ještě po Henselově algoritmu provést rekonstrukci faktorů. Tato rekonstrukce má bohužel exponenciální složitost.

Nyní zformulujeme pomocný algoritmus, který potom využijeme v algoritmu Henselově.

Lemma 3.1.1
Nechť $\mathbb{T}$ je těleso, f, a, b polynomy z $\mathbb{T}[x]$ a předpokládejme, že a, b jsou nesoudělné a $\deg(f) < \deg(a) + \deg(b)$. Pak existuje právě jedna dvojice polynomů $u, v \in \mathbb{T}[x]$ splňující

$$f = u \cdot a + v \cdot b$$

taková, že $\deg(u) < \deg(b)$ a $\deg(v) < \deg(a)$.

Tvrzení 3.1.2
Nechť $\mathbb{T}$ je těleso, $f, a_1, \ldots, a_n$ polynomy z $\mathbb{T}[x]$ a předpokládejme, že $a_1, \ldots, a_n$ 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 $b_1, \ldots, b_n \in \mathbb{T}[x]$ splňující

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

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

Tvrzení 3.1.2 je zobecněním Lemmatu 3.1.1. Obě tato tvrzení se používají při důkazu správnosti a na odhad časové složitosti pomocného Algoritmu 3.1.3 (PomocnyHensel).

Na základě Tvrzení 3.1.2 již můžeme zformulovat algoritmus.

Algoritmus 3.1.3 (PomocnyHensel)

Vstup: $f, a_1, \ldots, a_n \in \mathbb{T}[x]$ ($a_1, \ldots, a_n$ po dvou nesoudělné, $\deg(f) < \sum_i \deg(a_i)$)
Výstup: $b_1, \ldots, b_n$ 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 $i = 1, \ldots, n-1$ 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 $b_1, \ldots, b_n$.

Tvrzení 3.1.4
Časová složitost Algoritmu 3.1.3 (PomocnyHensel) je $\mathcal{O}(n \cdot m^2)$, kde m je definováno předpisem $m = \sum_{i=1}^{n} \deg(a_i)$, n je počet polynomů na vstupu, tj. počet cyklů, přičemž za jednotku bereme operaci v okruhu koeficientů.

Nyní vyslovíme větu, na které je založen Henselův algoritmus.

Věta 3.1.5 (Henselovo lemma)
Buď f primitivní bezčtvercový polynom v $\mathbb{Z}[x]$, p prvočíslo takové, které nedělí $lc(f)$, a nechť $a_1, \ldots, a_n \in \mathbb{Z}_p[x]$ jsou po dvou nesoudělné polynomy splňující

$$f \equiv a_1 \cdot \ldots \cdot a_n \pmod{p},$$

přičemž $lc(a_1) = lc(f) \bmod p$ a $lc(a_2) = \ldots = lc(a_n) = 1$.
Pak pro všechna přirozená čísla k existují $a_1^{(k)}, \ldots, a_n^{(k)} \in \mathbb{Z}_{p^k}[x]$ splňující

$$f \equiv a_1^{(k)} \cdot \ldots \cdot a_n^{(k)} \pmod{p^k},$$

přičemž $lc(a_1^{(k)}) = lc(f) \bmod p^k$ a $lc(a_2^{(k)}) = \ldots = lc(a_n^{(k)}) = 1$ a pro všechna i platí

$$a_i^{(k)} \equiv a_i \pmod{p}.$$

Algoritmus, který provádí Henselovo liftování je založený přímo na Henselově lemmatu 3.1.5. Tímto algoritmem "naliftujeme" rozklad modulo p na rozklad modulo p^k.

Algoritmus 3.1.6 (Henselův)

Vstup: $f, a_1, \ldots, a_n$ jako v Henselově lemmatu 3.1.5, $K \in \mathbb{N}$
Výstup: $a_1^{(K)}, \ldots, a_n^{(K)}$ jako v Henselově lemmatu 3.1.5

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 $lc(f) \bmod p^k$,
   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 $\bar{a}_1, \ldots, \bar{a}_n$.

Tvrzení 3.1.7
Časová složitost Henselova algoritmu 3.1.6 je $\mathcal{O}(K \cdot n \cdot m^2)$, kde $m = \deg(f)$.

3.2 Berlekampův algoritmus

Nyní stručně popíšeme Berlekampův algoritmus. Pomocí tohoto algoritmu můžeme počítat ireducibilní rozklad polynomu nad konečným tělesem. Jeho složitost je $\mathcal{O}(n^3)$. Celý algoritmus je založen na následující větě.

Věta 3.2.1
Nechť $f \in \mathbb{F}_q[x]$ je monický polynom, $h \in \mathbb{F}_q[x]$ takový, že $h^q \equiv h \bmod f$. Pak platí

$$f(x) = \prod_{c \in \mathbb{F}_q} \mathrm{NSD}(f(x), h(x) - c) \ .$$

Abychom mohli tuto větu použít, potřebujeme umět hledat polynomy $h(x)$. Zajímat nás budou takové polynomy $h(x)$, pro které platí $h^q \equiv h \bmod f$ a rozklad $\prod_{c \in \mathbb{F}_q} \mathrm{NSD}(f(x), h(x) - c)$ je netriviální. Těmto polynomům h se říká *f-redukující*.

Každý polynom $h(x) \in \mathbb{F}_q[x]$, pro který platí kongruence $h^q \equiv h \bmod f$ a zároveň $0 < \deg(h) < \deg(f)$, je f-redukující. Na hledání f-redukujících polynomů použijeme Čínskou větu o zbytcích.

Předpokládejme, že $f = f_1 \cdot \ldots \cdot f_k$ pro navzájem různé ireducibilní polynomy $f_1, \ldots, f_k$ a nechť $\deg(f) = n$. Zvolíme uspořádanou k-tici $(c_1, \ldots, c_k)$ prvků tělesa $\mathbb{F}_q$. Podle Čínské věty o zbytcích existuje právě jeden polynom $h(x) \in \mathbb{F}_q[x]$ stupně menšího než n takový, že

$$h(x) \ \equiv \ c_i \bmod f_i(x) \ \text{ pro všechna } i = 1, \ldots, k \ .$$

Z toho zřejmě plyne

$$h^q(x) \ \equiv \ c_i^q = c_i \ \equiv \ h(x) \bmod f_i(x) \ \text{ pro všechna } i = 1, \ldots, k \ ,$$

to znamená, že

$$f_i(x) \mid h^q(x) - h(x) \ \text{ pro všechna } i = 1, \ldots, k \ .$$

Protože jsou $f_1, \ldots, f_k$ po dvou nesoudělné, platí

$$f(x) = f_1(x) \cdot \ldots \cdot f_k(x) \mid h^q(x) - h(x) \ .$$

Je-li $h(x)$ libovolný polynom stupně menšího než n, pak podle Věty 3.2.1 platí

$$f(x) = \prod_{c \in \mathbb{F}_q} \mathrm{NSD}(f(x), h(x) - c) \ ,$$

a proto pro všechna $i = 1, \ldots, k$ platí

$$f_i(x) \mid f(x) = \prod_{c \in \mathbb{F}_q} \mathrm{NSD}(f(x), h(x) - c) \ .$$

Protože $f_i(x)$ je ireducibilní v $\mathbb{F}_q[x]$, existuje $c_i \in \mathbb{F}_q$ takový, že

$$f_i(x) \mid \mathrm{NSD}(f(x), h(x) - c_i) \ .$$

Tedy $f_i(x) \mid h(x) - c_i$ a proto $h(x) \equiv c_i \bmod f_i(x)$. Z jednoznačnosti existence takového polynomu $h(x)$, která je dána Čínskou větou o zbytcích, pak plyne, že počet polynomů $h(x)$, pro které platí $h^q(x) \equiv h(x) \bmod f(x)$ a $\deg(h) < n$, je přesně q^k.

Pro hledání těchto polynomů h použijeme následující lemma. Pro potřeby tohoto lemmatu si zadefinujeme matici B následovně:
Nejprve spočítáme

$$x^{jq} \bmod f(x) \quad \text{pro} \quad j = 0, \ldots, n-1 \ .$$

Platí

$$x^{jq} \bmod f(x) = \sum_{i=0}^{n-1} b_{ij} x^i, \quad \text{kde } b_{ij} \in \mathbb{F}_q \ .$$

Matici B pak definujeme $B = (b_{ij})$ (čtvercová matice řádu n).

Lemma 3.2.2
Polynom $h(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$ je řešením rovnice $h^q(x) \equiv h(x) \bmod f(x)$ právě tehdy, když platí

$$B \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix},$$

neboli

$$(B - I) \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = 0 \ .$$

Díky tomuto Lemmatu 3.2.2 nám pak stačí najít nějakou bázi nulového prostoru matice $(B-I)$. Jeden prvek báze je $(1, 0, \ldots, 0)$. Tomuto prvku odpovídá polynom $h_1(x) = 1$. Doplněním dalších vektorů do báze nulového prostoru, dostaneme další polynomy $h_2(x), \ldots, h_k(x)$.

Z následujícího lemmatu nám plyne, že pokud najdeme tyto polynomy $h(x)$, dokážeme najít i ireducibilní rozklad.

Lemma 3.2.3
Jsou-li f_1, f_2 dva různé ireducibilní faktory $f(x)$, potom existuje polynom $h_j(x)$ pro $j = 2, \ldots, k$ takový, že $f_1(x)$ a $f_2(x)$ dělí v součinu $\prod_{c\in\mathbb{F}_q} \mathrm{NSD}(f(x), h_j(x)-c)$ různé činitele.

Nyní prodiskutujeme případ, že pracujeme v tělese, které má mnoho prvků. V tomto případě je $q >> n$, a proto bude $\mathrm{NSD}(f(x), h(x) - c) = 1$ pro většinu $c \in \mathbb{F}_q$. Pokusíme se tedy omezit množinu používaných prvků $c \in \mathbb{F}_q$ pouze na takové, pro které je $\mathrm{NSD}(f(x), h(x) - c) > 1$.

Budeme postupovat podobně jako v předchozím případě. Začneme Berlekampovým algoritmem a řešíme soustavu $(B-I)x = 0$, najdeme nějakou bázi nulového prostoru $h_1(x) = 1,\ h_2(x), \ldots, h_k(x)$. Buď $h(x)$ jeden z těchto polynomů a

$$f(x) = \prod_{c\in\mathbb{F}_q} \mathrm{NSD}(f(x), h(x) - c) \ .$$

Označme

$$C = \{c \in \mathbb{F}_q;\ \mathrm{NSD}(f(x), h(x) - c) \neq 1\} \ .$$

Potom platí rovnost

$$f(x) = \prod_{c\in C} \mathrm{NSD}(f(x), h(x) - c) \ .$$

Odtud plyne, že $f(x) \mid \prod_{c\in C}(h(x) - c)$.
Označme $G(y) = \prod_{c\in C}(y - c)$. Pak $f(x) \mid G(h(x))$.

Následující věta nám ulehčí hledání polynomu $G(h(x))$ s nízkým stupněm.

Věta 3.2.4
Mezi všemi polynomy $G(y) \in \mathbb{F}_q[y]$, pro které platí $f(x) \mid G(h(x))$, má monický polynom $G(y)$ nejmenší stupeň.

Z vlastností $G(y)$ (konkrétně z vlastnosti, že $G(y)$ je polynom nejmenšího stupně takový, že $f(x) \mid G(h(x))$) plyne, že posloupnost polynomů

$$1 = h^0(x) \bmod f(x), h(x) = h^1(x) \bmod f(x), h^2(x) \bmod f(x), \ldots, h^{m-1}(x) \bmod f(x)$$

je lineárně nezávislá. Dále z rovnosti

$$f(x) \ = \ \prod_{c\in C} \mathrm{NSD}(f(x), h(x) - c)$$

plyne, že $m \leq k$, kde k je počet ireducibilních činitelů v rozkladu $f(x)$, protože počet činitelů na pravé straně poslední rovnosti (každý z nich má stupeň aspoň 1) nemůže být větší než počet různých ireducibilních dělitelů polynomu $f(x)$.

Nakonec shrneme postup hledání polynomu $G(y)$.
1) Počítáme postupně $h^j(x) \bmod f(x)$.
2) První index j, pro který platí, že $h^j(x) \bmod f(x)$ je lineární kombinací polynomů $h^i(x) \bmod f(x)$ pro $i = 0, \ldots, j-1$, si označíme m.
3) Když spočítáme tuto lineární kombinaci, dostaneme koeficienty $b_0, b_1, \ldots, b_{m-1}$. Tak najdeme polynom $G(y)$.
4) Spočítáme množinu jeho kořenů C a pak dokončíme Berlekampův algoritmus.

Berlekampův algoritmus s využitím polynomu $G(y)$ se nazývá *Zassenhausův algoritmus*.

3.3 Berlekamp–Henselův algoritmus

Po představení obou algoritmů zvlášť popíšeme nyní jejich kombinaci. Nejprve popíšeme Berlekamp–Henselův algoritmu slovně.

1) Zvolíme prvočíslo p tak, aby nedělilo vedoucí člen vstupního polynomu f, a tak, aby $f \bmod p$ byl bezčtvercový polynom v $\mathbb{Z}_p[x]$.
2) Pomocí Berlekampova algoritmu spočteme rozklad polynomu $f \bmod p$ v $\mathbb{Z}_p[x]$.
3) Tento rozklad dále "naliftujeme" pomocí Henselova algoritmu na rozklad modulo p^k. Hodnotu k zvolíme tak, aby $p^k > 2 \cdot \mathrm{LM}(f)$, $\mathrm{LM}(f) = 2^n \cdot \sqrt{\sum_{i=0}^{n} a_i^2}$.

Toto by mohl být rozklad polynomu f na ireducibilní faktory, ale obvykle se stává, že se polynom f rozkládá modulo p na více faktorů. Pak to znamená, že některé složky rozkladu, které nyní máme, ani nedělí polynom f. Z toho vyplývá, že je potřeba nalezené faktory nakombinovat do větších celků tak, aby odpovídaly dělitelům f, a tedy ireducibilnímu rozkladu.

4) Kombinace faktorů.
Tato fáze má bohužel v nejhorším případě exponenciální složitost. V Lenstra–Lenstra–Lovászově algoritmu je změněna tak, že Lenstra–Lenstra–Lovászův algoritmus má celkově polynomiální složitost.

Nyní následuje formální zápis algoritmu.

Algoritmus 3.3.1 (Berlekamp–Hensel)

Vstup:	$f \in \mathbb{Z}[x]$ primitivní bezčtvercový
Výstup:	ireducibilní rozklad $a_1, \ldots, a_k$ polynomu f

1. Zvol prvočíslo p takové, že $p \nmid lc(f)$ a $f \bmod p$ je bezčtvercový v $\mathbb{Z}_p[x]$.
2. Spočti ireducibilní rozklad $b_1, \ldots, b_l$ polynomu $f \bmod p$ v $\mathbb{Z}_p[x]$, a to tak, aby $lc(b_1) = lc(f) \bmod p$ a $lc(b_2) = \ldots = lc(b_l) = 1$.
3. Najdi nejmenší k takové, že $p^k > 2 \cdot \mathrm{LM}(f)$.
4. Spočti $c_1, \ldots, c_l \in \mathbb{Z}_{p^k}[x]$ splňující $f \equiv c_1 \cdot \ldots \cdot c_l \pmod{p^k}$, $lc(c_1) = lc(f) \bmod p^k$, $lc(c_2) = \ldots = lc(c_l) = 1$ a $c_i \equiv b_i \pmod{p}$ $\forall\, i$.
5. Kombinace faktorů:
   $C := \{2, \ldots, l\}$, $i := 0$, $m := 0$,
   while $m < |C|$ do
     $m := m + 1$
     pro všechna $i_1, \ldots, i_m \in C$ různá dělej
       $\tilde{g} := lc(f) \cdot c_{i_1} \cdot \ldots \cdot c_{i_m} \bmod p^k \in \mathbb{Z}[x]$,
       $g := pp(\tilde{g})$,
       if $g \mid f$ then $i := i + 1$, $a_i := g$, $f := \frac{f}{g}$, $C := C \setminus \{i_1, \ldots, i_m\}$,
   return $a_1, \ldots, a_i, f$.

Bezčtvercovost polynomu $f \bmod p$ v kroku 1. můžeme testovat pomocí kritéria $\mathrm{NSD}(f \bmod p, (f \bmod p)') = 1$ (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 $f \bmod p$ a výsledné polynomy znormujeme tak, aby platila podmínka na vedoucí členy.
V kroku 4. použijeme Henselův algoritmus na polynomy $b_1, \ldots, b_l$.
V kroku 5. kombinujeme faktory $c_1, \ldots, c_l$ 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 $lc(f)$, a z výsledku vezmeme primitivní část; zde využíváme předpokladu, že f je primitivní).
- Pokud jej tento součin 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 zbude v množině C po doběhu cyklu, je potřeba zkombinovat se členem c_1.

Kroky 1.– 4. mají tedy složitost polynomiální, krok 5. exponenciální, ale pouze v nejhorším případě.

Kapitola 4

Lenstra–Lenstra–Lovászův algoritmus

Lenstra–Lenstra–Lovászův algoritmus se od Berlekamp–Henselova algoritmu příliš neliší. Jediný, za to ale velice důležitý rozdíl, je v poslední fázi, kde Berlekamp–Henselův algoritmus používá hrubou sílu na nalezení a nakombinování skutečných faktorů v rozkladu nad celými čísly. Právě v tomto se Lenstra–Lenstra–Lovászův algoritmus odklání. Pro rekonstrukci faktorů využívá poznatky z teorie mříží. Díky tomu pracuje v polynomiálním čase i v nejhorším případě.

Než se dostaneme k samotnému popisu algoritmu, shrneme několik poznatků o mřížích a zároveň uvedeme pomocné algoritmy.

4.1 Úvod do teorie mříží

V této části zadefinujeme některé základní pojmy z teorie mříží a Hadamardovu nerovnost, kterou později budeme využívat v důkazech.

Nejdříve si připomeneme Gram–Schmidtův ortogonalizační proces, který vytvoří z libovolné báze ortogonální bázi.

Poznámka 4.1.1
Z báze $b_1, \ldots, b_n$ vektorového prostoru $\mathbb{R}^n$ spočteme ortogonální bázi $b_1^*, \ldots, b_n^*$ pomocí Gram–Schmidtova ortogonalizačního procesu. Označme $\langle \cdot\,, \cdot \rangle$ skalární součin dvou vektorů z $\mathbb{R}^n$ a $\| \cdot \|$ euklidovskou délku vektoru z $\mathbb{R}^n$. Pak tedy máme $\|a\|^2 = \langle a, a \rangle$. V Gram–Schmidtově ortogonalizačním procesu induktivně definu-

jeme vektory b_i^* a reálná čísla $\mu_{i,j},\ 1 \leq j < i \leq n$ následovně:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \ ,$$

$$\mu_{i,j} = \frac{\langle b_i, b_i^* \rangle}{\|b_j^*\|^2} \ .$$

Tyto vlastnosti budeme později používat v důkazech.

Před formulováním Hadamardovy nerovnosti formálně definujeme mřížky a některé další pojmy.

Definice 2
Nechť n je přirozené číslo a $b_1, \ldots, b_n \in \mathbb{R}^n$ lineárně nezávislé vektory nad $\mathbb{R}$. Pak množinu

$$L = \sum_{i=1}^{n} \mathbb{Z} b_i = \Big\{ \sum_{i=1}^{n} a_i b_i \,;\ a_1, \ldots, a_n \in \mathbb{Z} \Big\}$$

nazýváme *mřížka* nad $b_1, \ldots, b_n$.

Definice 3
Řekneme, že $b_1, \ldots, b_n$ tvoří *bázi mřížky* L, pokud L je mřížka nad těmito vektory. Libovolná podmnožina $M \subseteq \mathbb{R}^n$ je mřížka, pokud existují vektory $b_1, \ldots, b_n$ takové, že M je mřížka nad $b_1, \ldots, b_n$.
Číslo n se nazývá *hodnost mřížky.*

Definice 4
Nechť L je mřížka nad $b_1, \ldots, b_n$ v $\mathbb{R}^n$, pak definujeme *determinant* $det(L)$ mřížky L předpisem

$$det(L) = \big| det(b_1, \ldots, b_n) \big| \ ,$$

kde b_i jsou sloupcové vektory. Determinant je nezávislý na volbě báze.

Tvrzení 4.1.2 (Hadamardova nerovnost)
Nechť L je mřížka, $det(L)$ její determinant, $b_1, \ldots, b_n \in \mathbb{R}^n$ báze mřížky. Pak

$$det(L) \leq \prod_{i=1}^{n} \|b_i\| \ .$$

Důkaz

Definujme $B_i = \|b_i^*\|^2$, jako ortogonální protějšky $b_1, \ldots, b_n$, potom z ortogo-

nality b_i^* plyne

$$\|b_i\|^2 = B_i + \sum_{j=1}^{i-1} \mu_{i,j}^2 B_j$$

a proto $det(L)^2 = \prod_{i=1}^{n} B_i \leq \prod_{i=1}^{n} \|b_i\|^2$. □

4.2 Redukce báze

Zde popíšeme základní algoritmus, který se v teorii mříží používá k redukování báze mřížky. Nejdříve definujeme pojem redukovaná báze, pak dokážeme některé její vlastnosti, které pak využijeme při faktorizačním algoritmu, resp. jednom jeho subalgoritmu.

Definice 5
Bázi $b_1, \ldots, b_n$, která splňuje

$$\left|\mu_{i,j}\right| \ \leq \ \frac{1}{2} \ \text{ pro } \ 1 \leq j < i \leq n$$

a zároveň

$$\left\|b_i^* + \mu_{i,i-1} b_{i-1}^*\right\|^2 \ \geq \ \frac{3}{4} \left\|b_{i-1}^*\right\|^2 \ \text{ pro } \ 1 < i \leq n \ ,$$

kde b_i^* a $\mu_{i,j}$ jsou definovány jako v Poznámce 4.1.1, nazýváme *redukovaná báze*.

Po definici redukované báze nyní vyslovíme některé její vlastnosti.

Tvrzení 4.2.1
Nechť $b_1, \ldots, b_n$ je redukovaná báze mřížky L v $\mathbb{R}^n$ a nechť $b_1^*, \ldots, b_n^*$ je ortogonální báze z Gram–Schmidtova ortogonalizačního procesu. Pak platí
a) $\|b_j\|^2 \leq 2^{i-1} \cdot \|b_i^*\|^2$ pro $1 \leq j < i \leq n$,
b) $\|b_1\|^2 \leq 2^{n-1} \cdot \|x\|^2$ pro každé $x \in L \setminus \{0\}$,
c) pokud $x_1, \ldots, x_t \in L$ jsou lineárně nezávislé, potom
$\max\{\|b_1\|^2, \ldots, \|b_t\|^2\} \ \leq \ 2^{n-1} \cdot \max\{\|x_1\|^2, \ldots, \|x_t\|^2\}$.

Důkaz
a) Definujme $B_i = \|b_i^*\|^2$.
Z Definice 5 redukované báze platí:

$$B_i \ \geq \ \left(\frac{3}{4} \ - \ \mu_{i,i-1}^2\right) \cdot \|b_{i-1}^*\|^2 \ = \ \left(\frac{3}{4} \ - \ \mu_{i,i-1}^2\right) \cdot B_{i-1} \ \geq \ \frac{B_{i-1}}{2} \ .$$

Použitím indukce dostáváme

$$B_j \;\geq\; 2^{i-j} \cdot B_i \;\text{ pro }\; i \;\geq\; j \;.$$

Tedy

$$\|b_j\|^2 \;=\; \left\|b_j^* \;+\; \sum_{k=1}^{j-1} \mu_{j,k} b_k^*\right\|^2 \;=\; B_j \;+ \sum_{k=1}^{j-1} \mu_{j,k}^2 B_k \;\leq$$

(Výše uvedená rovnost platí, protože b_j^* jsou ortogonální.)

$$\leq\; B_j \;+\; \frac{1}{4}\sum_{k=1}^{j-1} B_k \;\leq\; B_j \;+\; \frac{1}{4}\sum_{k=1}^{j-1} 2^{i-k} B_i \;\text{ pro }\; 1 \;\leq\; k \;<\; j \;\leq\; i \;\leq\; n \;.$$

Sečtením sumy dostáváme

$$=\; B_j \;+\; 2^{i-2} B_i \;-\; \frac{1}{2} B_i \;.$$

Pokud B_j odhadneme také pomocí B_i, pak máme

$$\leq\; 2^{i-j} B_i \;+\; 2^{i-2} B_i \;-\; \frac{1}{2} B_i \;.$$

Následnými úpravami dostáváme

$$=\; (2^{i-j-1} \;+\; 2^{i-2}) B_i \;\leq\; 2^{i-1} \|b_i^*\|^2 \;.$$

Z toho plyne požadovaná nerovnost

$$\|b_j\|^2 \leq 2^{i-1} \cdot \|b_i^*\|^2 \;.$$

b) Každý vektor $x \in L$ můžu vyjádři pomocí vektorů báze, tj. existuje i takové, že platí:

$$x = \sum_{1 \leq j \leq i} r_j b_j = \sum_{1 \leq j \leq i} s_j b_j^* \;\text{ kde } r_i \neq 0,\; r_j \in \mathbb{Z} \text{ a } s_j \in \mathbb{R}$$

Z definice b_j^*, tím že vznikly Gram–Schmidtovou ortogonalizací, plyne, že $r_i = s_i$ a proto můžeme normu x odhadnout následovně

$$\|x\|^2 = \left\| \sum_{i \leq j \leq i} s_j b_j^* \right\|^2 = \sum_{i \leq j \leq i} |s_j|^2 \|b_j^*\|^2 \geq s_i^2 \|b_i^*\|^2 = r_i^2 \|b_i^*\|^2 \geq \|b_i^*\|^2 \;.$$

Z bodu a) pro $j = 1$ plyne:

$$\|b_i^*\|^2 \geq 2^{1-i} \|b_1\|^2 \geq 2^{1-n} \|b_1\|^2$$

Tedy

$$\|b_1\|^2 \leq 2^{n-1} \|x\|^2 \;.$$

c) Dostáváme zobecněním bodu b). □

Nejdříve uvedeme dva pomocné algoritmy, které využijeme ve stěžejním algoritmu této kapitoly.

Algoritmus 4.2.2 (Redukce)
Vstup: k, l

if $|\mu_{k,l}| > \frac{1}{2}$
then $r :=$ číslo nejbližší k $\mu_{k,l}$,
$b_k := b_k - r.b_l$,
for $j = 1$ to $l - 1$ do $\mu_{k,l} := \mu_{k,l} - \mu_{l,j}$,
$\mu_{k,l} := \mu_{k,l} - r$.

Algoritmus 4.2.3 (Update)
Vstup: k

$\mu = \mu_{k,k-1}$, $B := B_k + \mu^2 B_{k-1}$, $\mu_{k,k-1} := \frac{\mu B_{k-1}}{B}$,
$B_k := \frac{B_{k-1} B_k}{B}$, $B_{k-1} := B$,
$(b_{k-1}, b_k) := (b_k, b_{k-1})$,
for $j = 1$ to $k - 2$ do $(\mu_{k-1,j},\ \mu_{i,j}) := (\mu_{i,j},\ \mu_{k-1,j})$,
for $i = k + l$ to n do
$(\mu_{i,k-1}, \mu_{i,k}) := (\mu_{i,k-1}\mu_{k,k-1} + \mu_{i,k}(1 - \mu\mu_{k,k-1}),\ \mu_{i,k-1} - \mu\mu_{i,k})$.

Nyní už můžeme popsat hlavní algoritmus. Tento algoritmus transformuje náhodnou bázi mřížky L na redukovanou bázi.

Algoritmus 4.2.4 (RedukceBaze)

Vstup: $a_1, \ldots, a_n$ (náhodná báze z $\mathbb{R}^n$)
Výstup: $b_1, \ldots, b_n$ (redukovaná báze mřížky L)

1. (Gram–Schmidtova ortogonalizace)
for $i = 1$ to n do
$b_i := a_i$, $b_i^* := a_i$,
for $j = 1$ to $i - 1$ do
$\mu_{i,j} := \frac{\langle b_i, b_j^* \rangle}{B_j}$,
$b_i^* := b_i^* - \mu_{i,j} b_i^*$,
$B_i := \|b_i^*\|^2$.

2. (redukce)
$k := 2$,
while $k \leq n$ do
$l := k - 1$,
$Redukce(k, l)$,
if $B_k < (\frac{3}{4} - \mu_{k,k-1}^2)B_{k-1}$ (Podmínka 1)
then $Update(k)$,
if $k > 2$ then $k := k - 1$,
else for $l = k - 2$ downto 1 do $Redukce(k, l)$,
$k := k + 1$.

Tvrzení 4.2.5
Algoritmus 4.2.4 (RedukceBaze) je korektní.

Důkaz
1) Nejdříve dokážeme správnost algoritmu.

V prvním kroku jsou pomocí Gram–Schmidtovy ortogonalizace spočítány ortogonální vektory $b_1^*, \ldots, b_n^*$ a odpovídající koeficienty $\mu_{i,j}$. Podmínky pro výstup platí podle Poznámky 4.1.1.

Během algoritmu jsou b_i několikrát přepočítávány, ale pořád tvoří bázi mřížky L. Vektory b_i^* a reálná čísla $\mu_{i,j}$ jsou počítány souběžně, proto stále platí podmínky z Gram–Schmidtovy ortogonalizace. Ve skutečnosti není nutné uchovávat hodnoty b_i^*, ale ponecháme je pro výpočet $B_i = \|b_i^*\|^2$. Ukážeme, že kdykoliv proběhne while–cyklus, platí následující podmínky:

$$|\mu_{i,j}| \leq \frac{1}{2} \text{ pro } 1 \leq j < i < k$$

a zároveň

$$\|b_i^* + \mu_{i,i-1}b_{i-1}^*\|^2 \geq \frac{3}{4}\|b_{i-1}^*\|^2 \quad \text{pro} \quad 1 < i < k\,.$$

Z těchto podmínek plyne, že když algoritmus skončí, tak $k = n + 1$ a máme $b_1, \ldots, b_n$ redukovanou bázi mřížky L generovanou vstupní bází $a_1, \ldots, a_n$.

Výše uvedené podmínky zřejmě platí, když algoritmus vstoupí do while–cyklu poprvé. Pro $k = 2$ je druhá podmínka ($1 < i < k$) prázdná. Číslo k bude při běhu algoritmu vždy v rozmezí $2 \leq k \leq n + 1$.

Předpokládejme, že while–cyklus doběhl k maximální hodnotě k. Tedy $k \leq n$ a podmínka výše platí pro všechna i, $1 < i < k$. Nejdříve jsou proměnné počítány

subalgoritmem *Redukce* tak, že $|\mu_{i,i-1}| \leq \frac{1}{2}$. Nyní otestujeme, zda platí

$$\|b_k^* \ + \ \mu_{k,k-1} b_{k-1}^*\|^2 \ > \frac{3}{4}\|b_{k-1}^*\|^2 \qquad \text{(Podmínka 2)}$$

(Uvědomme si, že pro ortogonální vektory b_k^* a b_{k-1}^* platí

$$\|b_k^* \ + \ \mu_{k,k-1} b_{k-1}^*\|^2 \ = \|b_k^*\|^2 + \mu_{k,k-1}^2 \|b_{k-1}^*\|^2. \ )$$

Pokud podmínka na B_k z Algoritmu 4.2.4 (Podmínka 1) platí, jsou vektory b_k a b_{k-1} prohozeny Algoritmem 4.2.3 (Update) a všechny ostatní b_i jsou nezměněny. B_i a $\mu_{i,j}$ jsou přepočítávány průběžně. Číslo k je nahrazeno $k-1$. Testovaná podmínka (Podmínka 2) tedy platí.

Pokud zmíněná podmínka na B_k neplatí, pak nejdříve docílíme, že $|\mu_{i,j}| \ \leq \ \frac{1}{2}$ pro $1 \leq j \leq k-1$ pomocí Algoritmu 4.2.2 (Redukce). Potom nahradíme k za $k+1$. Tím podmínka (Podmínka 2) platí.

2) Nyní dokážeme, že algoritmus skončí.

Nejdříve si definujme pomocné hodnoty $d_i = \det(\langle b_j, b_l \rangle_{1 \leq j,l \leq i})$ pro $1 \leq i \leq n$ a $d_0 = 1$. Tedy $d_i = \prod_{j=1}^{i} \|b_j^*\|^2 \in \mathbb{R}^+$. Dále definujme $D = \prod_{i=1}^{n-1} d_i$. V algoritmu je

$$B_{k-1} = \|b_{k-1}^*\|^2 \ \text{ nahrazeno } \ \|B_k \ + \ \mu_{k,k-1}^2 B_{k-1}\|,$$

což je menší než $\frac{3}{4} B_{k-1}$. Takže B_{k-1} je redukováno faktorem $\frac{3}{4}$. Čísla d_i jsou zespodu omezena kladným reálným číslem, které závisí pouze na mřížce L. Proto existuje současně dolní kladná hranice pro D, a proto také existuje horní hranice pro počet opakování. Počet opakování při nesplnění podmínky (Podmínka 1) je nejvíce o $n-1$ vyšší než hranice pro D. Algoritmus tedy skončí. □

Tvrzení 4.2.6
Nechť $L \subset \mathbb{Z}^n$ je mřížka s bází $a_1, \ldots, a_n$ a nechť $B \in \mathbb{R}, B \geq 2$ takové, že $\|a_i\|^2 \leq B$ pro $1 \leq i \leq n$. Pak počet aritmetických operací Algoritmu 4.2.4 (RedukceBaze) na vstupu $a_1, \ldots, a_n$ je $\mathcal{O}(n^4 \cdot \log B)$ a čísla, na kterých se operace provádí, jsou délky $\mathcal{O}(n \cdot \log B)$.
Pokud použijeme klasické násobení, je složitost Algoritmu 4.2.4 (RedukceBaze) $\mathcal{O}(n^6 \cdot (\log B)^3)$. Tento odhad lze zlepšit použitím jiných technik násobení.

4.3 Faktorizace a mříže

V této části budeme používat následující značení:

- nechť p je prvočíslo
- nechť k je přirozené číslo
- nechť f je primitivní, bezčtvercový polynom stupně $n > 0$ v $\mathbb{Z}[x]$
- nechť h je monický polynom stupně l, $0 < l \leq n$, v $\mathbb{Z}_{p^k}[x]$

Budeme předpokládat, že:

- h dělí $(f \bmod p^k)$ v $\mathbb{Z}_{p^k}[x]$
- $(h \bmod p)$ je ireducibilní v $\mathbb{Z}_p[x]$
- $(f \bmod p)$ je bezčtvercový v $\mathbb{Z}_p[x]$

Než začneme s formulací Lenstra–Lenstra–Lovászova algoritmu, dokážeme si nejdříve několik pomocných tvrzení, která použijeme později pro dokazování správnosti algoritmu.

Tvrzení 4.3.1
1) Existuje až na znaménko jednoznačně určený ireducibilní faktor h_0 polynomu f v $\mathbb{Z}[x]$ takový, že $(h \bmod p)$ dělí $(h_0 \bmod p)$.

2) Pokud g dělí f v $\mathbb{Z}[x]$, pak jsou následující tvrzení ekvivalentní:
a) $(h \bmod p)$ dělí $(g \bmod p)$ v $\mathbb{Z}_p[x]$
b) h dělí $(g \bmod p^k)$ v $\mathbb{Z}_{p^k}[x]$
c) h_0 dělí g v $\mathbb{Z}[x]$

Důkaz
1) Nechť $f = \prod_{i=0}^{s} h_i$ je rozklad f v $\mathbb{Z}[x]$. Potom $(h \bmod p)$ dělí některý z faktorů $(h_i \bmod p)$, ať je to h_0. Jednoznačnost h_0 plyne z bezčtvercovosti $(f \bmod p)$ v $\mathbb{Z}_p[x]$.

2) Zřejmě b) ⇒ a) a c) ⇒ a)

Nyní dokážeme a) ⇒ c):
Předpokládejme, že platí a). Protože $(f \bmod p)$ je bezčtvercový v $\mathbb{Z}_p[x]$, $(h \bmod p)$

nedělí $(f/g \bmod p)$ v $\mathbb{Z}_p[x]$. Také $(h_0 \bmod p)$ nedělí $(f/g \bmod p)$ v $\mathbb{Z}_p[x]$ a navíc h_0 nedělí f/g v $\mathbb{Z}[x]$. Proto h_0 musí být faktor g v $\mathbb{Z}[x]$.

Nakonec dokážeme a) $\Rightarrow$ b):
$(h \bmod p)$ a $(f/g \bmod p)$ jsou nesoudělné v $\mathbb{Z}_p[x]$, proto existují $r, s \in \mathbb{Z}_p[x]$, pro které platí

$$r \cdot h \ + \ s \cdot (f/g) \ \equiv \ 1 \pmod{p}.$$

"Liftováním" dostaneme $r', s' \in \mathbb{Z}_{p^k}[x]$ takové, že

$$r' \cdot h \ + \ s' \cdot (f/g) \ \equiv \ 1 \pmod{p^k}$$

nebo

$$r' \cdot (g \bmod p^k) \cdot h \ + \ s' \cdot f \ \equiv \ g \pmod{p^k}.$$

Protože h dělí levou stranu této rovnice, pak h dělí i pravou stranu tj. h dělí $(g \bmod p^k)$. □

Důsledek 4.3.2
Polynom h_0 dělí $(h \bmod p^k)$ v $\mathbb{Z}_{p^k}[x]$.

Důkaz
Plyne z předchozího $g = h_0$. □

Doteď jsme pracovali s mřížkami nad čísly. Pro práci s mřížkami nad polynomy použijeme následující značení:

$$L_{m,h} = (g \in \mathbb{Z}[x] \text{ takové, že } \deg(g) \leq m \text{ a } h \text{ dělí } (g \bmod p^k) \text{ v } \mathbb{Z}_{p^k}[x])$$

Díky izomorfismu mezi $\mathbb{R}^{m+1}$ a polynomy z $\mathbb{R}[x]$ stupně menšího než m je $L_{m,h}$ mřížka nad bází

$$\{p^k x^i, \text{ kde } 0 \ \leq \ i \ < \ l\} \cup \{h x^j, \text{ kde } 0 \ \leq \ j \ \leq \ m - l\} \ ,$$

pro $m \in \mathbb{N}$ a l stupeň polynomu h.

Tvrzení 4.3.3
Nechť $b \in L_{m,h}$ vyhovuje podmínce $p^{kl} > \|f\|^m \cdot \|b\|^n$. Potom h_0 dělí b v $\mathbb{Z}[x]$ a speciálně $\mathrm{NSD}(f, b) \neq 1$.

Důkaz
Bez újmy na obecnosti můžeme předpokládat, že $b \neq 0$. Nechť $s = \deg(b)$, $g = \mathrm{NSD}(f, b)$ v $\mathbb{Z}[x]$ a $t = \deg(g)$. Všimněme si, že $0 \leq t \leq s \leq m$. Pokud chceme ukázat, že h_0 dělí b, pak stačí ukázat, že h_0 dělí g, což je podle Tvrzení 4.3.1 b)

ekvivalentní s tím, že (h mod p) dělí (g mod p) v $\mathbb{Z}_p[x]$.
Předpokládejme, že toto neplatí. Potom (h mod p) dělí (f/g mod p). Uvažujme množinu polynomů

$$M \;=\; \{\lambda f \;+\; \mu b,\ \text{kde } \lambda, \mu \in \mathbb{Z}[x],\ \deg(\lambda) \;<\; s-t,\ \deg(\mu) \;<\; n-t\}.$$

Nechť

$$M' \;=\; \Big\{ \sum_{i=t}^{n+s-t-1} a_i x^i,\ \text{kde} \sum_{i=0}^{n+s-t-1} a_i x^i \in M \Big\}.$$

tj. M' je projekce M na její poslední souřadnici. Tato projekce je lineárně nezávislá. Proto M je mřížka hodnosti $n+s-2t$. Z Hadamardovy nerovnosti 4.1.2 plyne

$$det(M') \;\leq\; \|f\|^{s-t} \cdot \|b\|^{n-t} \leq\; \|f\|^m \cdot \|b\|^n \;<\; p^{kl}\ .$$

Nechť $b_t, b_{t+1}, \ldots, b_{n+s-t-1}$ je báze M' se stupni $\deg(b_j) = j$. Všimněme si, že pokud g dělí b a (h mod p) dělí (f/g mod p), pak $t \;+\; l \;-\; 1 \;\leq\; n \;+\; s \;-\; t \;-\; 1$. Vedoucí koeficienty $b_t, b_{t+1}, \ldots, b_{n+l-1}$ jsou dělitelné p^k. Proto

$$det(M') \;=\; \Big| \prod_{i=t}^{n+s-t-1} lc(b_i) \Big| \;\geq\; p^{kl}\ .$$

A to je spor. Takže musí platit, že (h mod p) dělí (g mod p) v $\mathbb{Z}_p[x]$. □

Lemma 4.3.4
Nechť $q(x) \;=\; b_0 \;+\; b_1 x \;+ \ldots +\; b_l x^l \;\in \mathbb{Z}[x]$ *je dělitel polynomu* $p(x) \in \mathbb{Z}[x]$. *Pak*
a) $|b_i| \;\leq\; \binom{l}{i} \|p\|$ *pro* $0 \;\leq\; i \;\leq\; l$,
b) $\|q\| \;\leq\; \binom{2l}{l}^{1/2} \|p\|$.

Důkaz
a) Tento důkaz je velmi technický a nezáživný, proto jej uvedeme pouze v bodech. Snadno se ukáže, že platí

$$\|(x \;+\; c)f\| \;=\; |c| \cdot \|(x \;+\; c^{-1})f\| \;\text{ pro } c \;\neq\; 0\ .$$

Opakováním dostáváme:

$$\|(x \;-\; x_1) \cdot \ldots \cdot (x \;-\; x_t) \cdot (x \;-\; x_{t+1}) \cdot \ldots \cdot (x \;-\; x_m)\| \;=$$

$$= \;|x_1 \cdot \ldots \cdot x_t| \cdot \|(x \;-\; x_1^{-1}) \cdot \ldots \cdot (x \;-\; x_t^{-1}) \cdot (x \;-\; x_{t+1}) \cdot \ldots \cdot (x \;-\; x_m)\|$$

$$\text{pro } x_1 \neq 0, \ldots, x_t \neq 0$$

Nyní potřebujeme, aby platilo:

$$\Big\| \prod_{i=1}^{m} (x \;-\; x_i) \Big\|^2 \;\geq\; |x_1 \cdot \ldots \cdot x_t|^2 \;+\; |x_{t+1} \cdot \ldots \cdot x_m|^2$$

Tato nerovnost platí jednoduše dle předchozího pro $x_1 \neq 0, \ldots, x_t \neq 0$,
pro $x_1 = 0, \ldots, x_s = 0$ a $x_{s+1} \neq 0, \ldots, x_t \neq 0$ dostáváme

$$\Big\| \prod_{i=1}^{m} (x \ - \ x_i) \Big\|^2 \ = \ \Big\| \prod_{i=s+1}^{m} (x \ - \ x_i) \Big\|^2 \ \geq \ |x_{s+1} \cdot \ldots \cdot x_t|^2 \ + \ |x_{t+1} \cdot \ldots \cdot x_m|^2 \ .$$

Tímto máme:

$$\|p\| \ \geq \ |x_{t+1} \cdot \ldots \cdot x_m|$$

$$|p_i| \ \leq \ \binom{m}{i} \cdot |x_{t+1} \cdot \ldots \cdot x_m| \ \leq \ \binom{m}{i} \|p\|$$

Z předchozího již jednoduše dostáváme původní tvrzení

$$|b_i| \ \leq \ \binom{l}{i} \|p\| \ .$$

Vzhledem k tomu, že využíváme odhad pomocí kořenů a rozkladu na lineární koeficienty, které nemusí v $\mathbb{Z}[x]$ existovat, použijeme odpovídající kořenové rozšíření. Závěrečná nerovnost pak platí i pro polynomy v $\mathbb{Z}[x]$.

b) Z bodu a) vyplývá:

$$\|q\| \ \leq \ \sqrt{\sum_{i=0}^{l} \binom{l}{i}^2} \cdot \|p\| \ .$$

Z Vardemondovy rovnosti

$$\sum_{k=0}^{n} \binom{r}{k} \binom{s}{n-k} = \binom{r+s}{n}$$

aplikované na $s \ = \ r \ = \ n \ = \ l$ plyne

$$\|q\| \ \leq \ \binom{2l}{l}^{1/2} \|p\| \ .$$

□

Tvrzení 4.3.5
Nechť je $b_1, \ldots, b_{m+1}$ redukovaná báze mřížky $L_{m,h}$, $h_0 \ = \ \mathrm{NSD}(b_1, \ldots, b_t)$ a nechť platí

$$p^{kl} \ > \ 2^{mn/2} \cdot \binom{2m}{m}^{n/2} \cdot \|f\|^{m+n} \ .$$

Potom

a) $\deg(h_0) \leq m$ *právě tehdy, když* $\|b_1\| < \sqrt[n]{p^{kl}/\|f\|^m}$

b) Předpokládejme, že existuje index $j \in \{1, \ldots, m+1\}$*, pro který platí*

$$\|b_j\| \ < \ \sqrt[n]{p^{kl}/\|f\|^m}$$

Nechť t je největší z těchto indexů. Potom $\deg(h_0) = m+1-t$ *a předchozí nerovnost platí pro všechna* $j,\ 1 \leq j \leq t$.

Důkaz

a)"$\Leftarrow$" Pokud je b_1 takováto hranice, pak podle Tvrzení 4.3.3 polynom h_0 dělí b_1, $\deg(b_1) \leq m$, a proto i $\deg(h_0) \leq m$.

"$\Rightarrow$" Pokud $\deg(h_0) \leq m$, potom $h_0 \in L_{m,h}$. Takže podle Tvrzení 4.2.1 b) a Lemmatu 4.3.4 b)

$$\|b_1\| \ < \ 2^{m/2} \cdot \|h_0\| \ \leq \ 2^{m/2} \cdot \binom{2m}{m}^{1/2} \cdot \|f\| \ .$$

Použitím nerovnosti z předpokladů dostáváme mez pro $\|b_1\|$.

b) Definujme množinu J následovně:

$J \ = \ \{j;\ 1 \ \leq \ j \ \leq \ m+1$ *a* j splňuje podmínku z předpokladů tvrzení$\}$

Podle Tvrzení 4.3.3 pro každé $j \in J$, polynom h_0 dělí b_j. Takže h_0 dělí h_1, které definujeme

$$h_1 \ = \ \mathrm{NSD}(\{b_j;\ j \ \in \ J\}).$$

Každé b_j, takové že $j \in J$, je dělitelné h_1 a nemá stupeň větší než m. To znamená, že náleží do matice

$$\mathbb{Z} \cdot h_1 \ + \ \mathbb{Z} \cdot h_1 \cdot x \ + \ldots + \ \mathbb{Z} \cdot h_1 \cdot x^{m-\deg(h_1)}$$

hodnosti $m + 1 - \deg(h_1)$. Navíc b_j jsou lineárně nezávislé, proto

$$|J| \ \leq \ m \ + \ 1 \ - \ \deg(h_1) \ .$$

Stejně jako v a) ukážeme, že

$$\|h_0 \cdot x^i\| \ = \ \|h_0\| \ \leq \ \binom{2m}{m}^{1/2} \cdot \|f\| \ \text{ pro všechna } i \ \geq \ 0 \ .$$

Pro $i \in \{0, 1, \ldots, m - \deg(h_0)\}$ máme $h_0 \cdot x^i \in L_{m,h}$. Takže z Tvrzení 4.2.1 c) dostáváme

$$\|b_j\| \ < \ 2^{m/2} \cdot \binom{2m}{m}^{1/2} \cdot \|f\| \ \text{ pro } 1 \ \leq \ j \ \leq \ m \ + \ 1 \ - \ \deg(h_0) \ .$$

Takže z předpokladů tvrzení plyne

$$\{1, \ldots, m \ + \ 1 \ - \ \deg(h_0)\} \ \subseteq \ J \ .$$

Ale h_0 dělí h_1, proto z odvozené nerovnosti a inkluze o množině J máme

$\deg(h_0) \ = \ \deg(h_1) \ = \ m + 1 - t$,

$J = \{1, \ldots, t\}$,
$h_1 = a \cdot h_0$ pro $a \in \mathbb{Z}$.
Dále ještě máme $\deg(h_0) \leq m$ plynoucí z a), proto $h_0 \in L_{m,h}$.

Polynom h_0 je primitivní, proto dokazujeme, že h_0 je rovno h_1, navíc to postačí k ukázání, že h_1 je také primitivní. Nechť j je libovolný element z J. Polynom h_0 dělí $pp(b_j)$. Protože $h_0 \in L_{m,h}$, tak i $pp(b_j) \in L_{m,h}$. Ale b_j náleží do báze mřížky $L_{m,h}$. Proto b_j musí být primitivní, a proto také faktor h_1 polynomu b_j je primitivní. Takže $h_0 = \pm h_1$. □

4.4 Faktorizační algoritmus

Než popíšeme samotný Lenstra–Lenstra–Lovászův algoritmus, začneme se dvěma subalgoritmy.

Algoritmus 4.4.1 (SubLLL1)

Vstup: $f \in \mathbb{Z}[x]$ primitivní a bezčtvercový polynom,
p prvočíslo, které nedělí $lc(f)$ takové, že $(f \bmod p)$ je bezčvercový,
k kladné celé číslo,
h polymom v $\mathbb{Z}_{p^k}[x]$, který má $lc(h) = 1$, h dělí $(f \bmod p^k)$, $(h \bmod p)$ je ireducibilní v $\mathbb{Z}_p[x]$,
m je celé číslo větší nebo rovno $\deg(h)$ takové, že
$p^{k \cdot \deg(h)} > 2^{mn/2} \cdot \binom{2m}{m}^{n/2} \cdot \|f\|^{m+n}$

Výstup: h_0 je ireducibilní faktor f, pro který platí, že $(h \bmod p)$ dělí $(h_0 \bmod p)$, pokud tento faktor má stupeň $\leq m$,
$h_0 = error$ jinak

1. $n := \deg(f)$, $l := \deg(h)$.
2. $(b_1, \ldots, b_{m+1}) := RedukceBaze(p^k x^0, \ldots, p^k x^{l-1}, hx^0, \ldots hx^{m-1})$.
3. if $\|b_1\| \geq \sqrt[n]{p^{kl}/\|f\|^m}$
   then $h_0 := error$,
   else $t :=$největší celé číslo takové, že $\|b_t\| < \sqrt[n]{p^{kl}/\|f\|^m}$,
   $h_0 := \mathrm{NSD}(b_1, \ldots, b_t)$.

Tvrzení 4.4.2
a) Algoritmus 4.4.1 (SubLLL1) je korektní.
b) Počet aritmetických operací v Algoritmu 4.4.1 (SubLLL1) je $\mathcal{O}(m^4 \cdot k \cdot \log p)$*, celá čísla, která algoritmus používá jsou délky maximálně* $\mathcal{O}(m \cdot k \cdot \log p)$.

Důkaz

a) $b_1, \ldots, b_{m+1}$ tvoří redukovanou bázi mřížky $L_{m,h}$. Pokud $\|b_1\| \geq \sqrt[n]{p^{kl}/\|f\|^m}$, pak podle Tvrzení 4.3.5 a) "error" je správná odpověď. Jinak podle Tvrzení 4.3.5 b) $h_0 := \mathrm{NSD}(b_1, \ldots, b_t)$, kde t je největší index, pro který platí $\|b_t\| < \sqrt[n]{p^{kl}/\|f\|^m}$.

b) Pro každý faktor a ve výchozí bázi mřížky $L_{m,h}$ je jeho norma shora omezena nerovností $\|a\|^2 \leq 1 + l \cdot p^{2k} =: B$. Z nerovnosti $l \leq n$ a ze vstupní podmínky na m vidíme, že m je omezeno $k \cdot \log p$. Dále platí $\log l < l \leq m$. Proto je i $\log B$ ohraničen $k \cdot \log p$. Použitím Tvrzení 4.2.6 máme hranice pro krok 2.

V kroku 3. potřebujeme spočítat největší společný dělitel $b_1, \ldots, b_t$. Každý koeficient c v b_j, $1 \leq j \leq t$, je omezen $\sqrt[n]{p^{kl}/\|f\|^m}$, takže $\log c < k \cdot \log p$. Podle Landau–Mignottovy meze jsou koeficienty v NSD velikosti $\mathcal{O}(2^m \|b_1\|)$, takže jejich délka je omezena $m + \log B \sim \log B$. Stejná mez platí pro všechny postupné výpočty NSD. Jeden výpočet NSD trvá $\mathcal{O}(m^2)$. My potřebujeme nejvýše m výpočtů NSD, takže počet aritmetických operací pro všechny výpočty NSD je $\mathcal{O}(m^3)$. □

Algoritmus 4.4.3 (SubLLL2)

Vstup: $f \in \mathbb{Z}[x]$ primitivní a bezčtvercový polynom,
p prvočíslo, které nedělí $lc(f)$ takové, že $(f \bmod p)$ je bezčvercový,
h polymom v $\mathbb{Z}_{p^k}[x]$, který má $lc(h) = 1$, h dělí $(f \bmod p^k)$, $(h \bmod p)$ je ireducibilní v $\mathbb{Z}_p[x]$

Výstup: h_0 je ireducibilní faktor f, pro který platí, že $(h \bmod p)$ dělí $(h_0 \bmod p)$

1. $n := \deg(f),\ l := \deg(h)$,
   if $l = n$ then $h_0 = f$.
2. $k :=$ nejmenší kladné číslo, pro které platí
   $$p^{kl} > 2^{(n-1)n/2} \cdot \binom{2(n-1)}{n-1}^{n/2} \cdot \|f\|^{2n-1},$$
   $(h',\ h'') := Hensel(f,\ (f/h)\ mod\ p,\ h,\ k)$.
3. $u :=$ největší celé číslo takové, že $l \leq (n-1)/2^u$,
   while $u > 0$ do
   $m := \lfloor (n-1)/2^u \rfloor$,
   $h_0 := SubLLL1(f,\ p,\ k,\ h',\ m)$,
   $if\ h_0 \neq error$ then return,
   $u := u - 1$.
4. $h_0 := f$.

Tvrzení 4.4.4

a) Algoritmus 4.4.3 (SubLLL2) je korektní.

b) Nechť m_0 je stupeň výsledného polynomu h_0. Potom počet aritmetických operací

v Algoritmu 4.4.3 (SubLLL2) je $\mathcal{O}(m_0(n^5 + n^4 \cdot \log \|f\| + n^3 \cdot \log p))$. *Čísla mají délku* $\mathcal{O}(n^3 + n^2 \cdot \log \|f\| + n \cdot \log p)$.

Důkaz

a) Pokud $l = n$, pak f je ireducibilní v $\mathbb{Z}_{p^k}[x]$, proto je ireducibilní i nad celými čísly.

Když se dostaneme ke kroku 2. Víme, že

$$f \;\equiv\; h' \cdot h'',$$

kde $(h' \bmod p) = h$ a $lc(h') = 1$. Nechť je nyní m takové, že $1 \leq m \leq n-1$. Číslo m splňuje vstupní podmínku algoritmu SubLLL1. Takže pokud ireducibilní faktor h_0 polynomu f odpovídající h' nemá stupeň vyšší jak m, algoritmus SubLLL1 ho spočítá. Pokud algoritmus SubLLL1 vrátí hodnotu error pro všechny hodnoty m, pak neexistuje faktor odpovídající h', proto je výsledek $h_0 = f$.

b) Pokud k je nejmenší kladné číslo splňující podmínku v kroku 2., pak máme

$$p^{k-1} \;\leq\; p^{(k-1)l} \;\leq\; 2^{(n-1)n/2} \cdot \binom{2(n-1)}{n-1}^{n/2} \cdot \|f\|^{2n-1} \;.$$

Použitím nerovnosti

$$\log \binom{2(n-1)}{n-1}^{n/2} \;\leq\; \log(2^{2(n-1)})^{n/2} \;=\; (n-1) \cdot n \cdot \log 2$$

vidíme, že

$$k \cdot \log p \;=\; (k-1) \cdot \log p \;+\; \log p \;\preceq\; n^2 \;+\; n \cdot \log \|f\| \;+\; \log p \;.$$

Nechť m_1 je největší hodnota m uvažována v SubLLL2. Pokud začneme malými hodnotami m a skončíme, když m přesáhne stupeň h_0, pak z toho vyplývá, že $m_1 < 2m_0$. Všechny ostatní hodnoty m jsou tvaru $\lfloor \frac{m_1}{2} \rfloor$, $\lfloor \frac{m_1}{4} \rfloor, \ldots, \lfloor \frac{m_1}{2^u} \rfloor$. Takže pokud sečteme všechny tyto hodnoty m, máme

$$\sum_{m \text{ uvažované } SubLLL2} m \;\leq\; \frac{m_1}{2^u} + \ldots + \frac{m_1}{2} + m_1 \;=\; m_1 \,.\, \frac{(\frac{1}{2})^{u+1} - 1}{\frac{1}{2} - 1} \;\leq\; 2m_1 \;<\; 4m_0 \,.$$

Proto $\sum m^4 \leq (\sum m)^4 = \mathcal{O}(m_0^4)$. Použitím Tvrzení 4.4.2 b) dostáváme, že počet aritmetických operací potřebných v kroku 3. je omezen

$$m_0^4 \cdot k \cdot \log p \preceq m_0^4(n^2 + n \cdot \log \|f\| + \log p) \preceq m_0(n^5 + n^4 \cdot \log \|f\| + n^3 \cdot \log p)$$

a že čísla používaná v těchto operacích mají délku omezenou

$$m_0^4(n^2 + n \cdot \log \|f\| + \log p) \preceq n^3 + n^2 \cdot \log \|f\| + n \cdot \log p.$$

Pro Henselovo liftování platí stejné meze v kroku 2. □

Nyní můžeme zformulovat celkový algoritmus LLL, který používá již zmíněné subalgoritmy.

Algoritmus 4.4.5 (LLL)

Vstup: $f \in \mathbb{Z}[x]$ primitivní a bezčtvercový polynom
Výstup: $F = [f_1, \ldots, f_s]$, kde f_i jsou různé ireducibilní faktory f

1. $n := \deg(f)$.
2. $p :=$ nejmenší prvočíslo nedělící $lc(f)$, takže $(f \bmod p)$ je stupně n,
   $pfactors := Berlekamp(f,\ p)$.
3. $F := [\,]$,
   $\widehat{f} := f$,
   while $\deg(\widehat{f}) > 0$ do
   $h :=$ první faktor z pfactors,
   $h := h/lc(h)$,
   $h_0 := SubLLL2(\widehat{f},\ p,\ h)$,
   (h_0 je ireducibilní faktor $\widehat{f}$, pro který h dělí $(h_0 \bmod p)$)
   přidej h_0 do F,
   $\widehat{f} := \widehat{f}/h_0$,
   for $g \in pfactors$ do
   if g dělí $(h_0 \bmod p)$,
   then smaž g z $pfactors$.

Tvrzení 4.4.6
a) Algoritmus 4.4.5 (LLL) je korektní.
b) Počet aritmetických operací v Algoritmu 4.4.5 (LLL) je $\mathcal{O}(n^6 + n^5 \cdot \log \|f\|)$.
Čísla mají délku $\mathcal{O}(n^3 + n^2 \cdot \log \|f\|)$.

Důkaz
a) Korektnost Algoritmu 4.4.5 (LLL) vychází ze správnosti Berlekampova algoritmu a Algoritmu 4.4.3 (SubLLL2).

b) Vzhledem k tomu, že p je voleno jako nejmenší prvočíslo, které nedělí $lc(f)$, pak můžeme v odhadu složitosti členy s $\log p$ vynechat. Berlekampův algoritmus má složitost $\mathcal{O}(n^3)$.

Počet aritmetických operací v kroku 3. je omezen $\mathcal{O}(m_0(n^5 + n^4 \cdot \log \|f\| + n^3))$, kde m_0 je stupeň h_0. Lemma 4.3.4 dává, že $\log \|\widehat{f}\| \preceq n + \log \|f\|$. Všechny stupně ireducibilních faktorů jsou nejvýše n. Takže počet aritmetických operací v kroku 3. je $\preceq n^6 + n^5 \cdot \log \|f\|$.

Podle Tvrzení 4.4.4 jsou všechna čísla v Algoritmu 4.4.5 (LLL) délky maximálně $\mathcal{O}(n^3 + n^2 \cdot \log \|f\|)$. □

Tvrzení 4.4.7
Pokud jsou použité klasické algoritmy na aritmetické operace, pak složitost Algoritmu 4.4.5 (LLL) je $\mathcal{O}(n^{12} + n^9 \cdot (\log \|f\|)^3)$.

Kapitola 5

Testy

V této části práce se budeme věnovat testování obou algoritmů.

Algoritmy jsou naprogramovány v jazyku C++ s použitím knihovny NTL, která oba algoritmy obsahuje. Testy byly prováděny na počítači s procesorem AMD Turion 1.8 GHz.

Algoritmy budeme testovat na náhodných a ,,pseudonáhodných" polynomech stupně 100. Každý test proběhne dvacetkrát na 100 polynomech.

Vstupem prvnímu testu budou náhodně generované polynomy. V dalších testech budou vstupy pouze ,,pseudonáhodné". Vygenerujeme polynomy nižšího stupně než 100 a vynásobíme je, aby bylo zajištěné, že testované polynomy nejsou ireducibilní.

Výsledky zaneseme do tabulky a do grafu. Poté porovnáme průměrné časy obou algoritmů. Protože by porovnání absolutního rozdílu nedávalo objektivní pohled, provedeme srovnání pomocí procentuálního vyjádření tj. Berlekamp–Henselův algoritmus bereme jako základ, proto je jeho čas brán jako 100%. Lenstra–Lenstra–Lovászův algoritmus pak srovnáváme s Berlekamp–Henselovým právě vyjádřením jeho rychlosti v procentech. Tyto tabulky a grafy lze pak najít v přílohách této práce.

Výsledky testů nejsou mezi sebou navzájem příliš porovnatelné, protože s rostoucím počtem polynomů, které mezi sebou násobíme, abychom získali ,,pseudonáhodný" polynom, rostou koeficienty. Tímto narůstá i čas výpočtu obou algoritmů. Netestujeme tedy pouze polynomy s různým složením faktorů, ale necháváme růst koeficienty, aby se projevila i případná závislost rychlosti obou algoritmů na výši koeficientů testovaných polynomů.

Celkem bylo provedeno 6 testů. V každém testu byly generovány polynomy podle následujícího schématu:

Test 1: 1 polynom stupně 100

Test 2: 2 polynomy, každý stupně 50

Test 3: 2 polynomy, jeden stupně 99 a jeden stupně 1

Test 4: 10 polynomů, každý stupně 10

Test 5: 10 polynomů, jeden stupně 91 a 9 stupně 1

Test 6: 100 polynomů, každý stupně 1

Kapitola 6

Závěr

V této práci jsme se věnovali dvěma algoritmům na faktorizaci polynomů. Berlekamp–Henselův má v nejhorším případě složitost exponenciální. Exponenciální je pouze poslední krok v tomto algoritmu tzv. kombinace faktorů. Na druhou stranu Lenstra–Lenstra–Lovászův algoritmus má složitost polynomiální i v nejhorším případě, protože právě kombinaci faktorů nahrazuje polynomiálním krokem, který využívá vlastnosti redukované báze mřížky.

V první části práce jsme se věnovali teoretickému popisu obou algoritmů. V druhé části jsem pak testovali algoritmy na náhodných vstupech. Cílem práce bylo porovnat algoritmy a zjistit, jestli udávaná složitost v nejhorším případě odpovídá i složitosti v průměrném případě.

Výsledky testů ukazují, že oba algoritmy jsou naprosto srovnatelné. Nedá se říci, že by byl jeden výrazně rychlejší a druhý nějak výrazně pomalejší. Jejich rozdíl se neprojevuje ani v testech, kde jsou faktory podobného stupně, ale ani v testech, kde je jeden faktor výrazně vyššího stupně. Stejně tak na srovnání průměrné složitosti těchto algoritmů nemá vliv růst koeficientů. Ve všech testech vyšlo, že oba algoritmy pracují ve velmi podobném čase. Pokud bereme Berlekamp–Henselův algoritmus jako základ 100% pak Lenstra–Lenstra–Lovászův algoritmus pracoval v rozmezí menším než 99 - 101%.

Lenstra–Lenstra–Lovászův algoritmus je sice v nejhorším případě polynomiální, ale rozdíl v čase výpočtu obou algoritmů se projeví až při vygenerování polynomu, který bude mít mnoho faktorů v modulárním algoritmu ale málo skutečných faktorů. Takto zvolený polynom nelze ale považovat za náhodný.

Lze tedy říci, že na náhodném vstupu pracují algoritmy srovnatelnou rychlostí.

Literatura

[1] Cohen Henri: *A Course in Computational Algebraic Number Theory*, Springer, Berlin, 1995.

[2] Winkler Franz: *Polynomial algorithms in computer algebra*, Springer, Wien, 1996.

[3] Tůma Jiří, texty k prednášce Konečná tělesa, `http://www.karlin.mff.cuni.cz/~tuma/ffields/skripta.pdf`

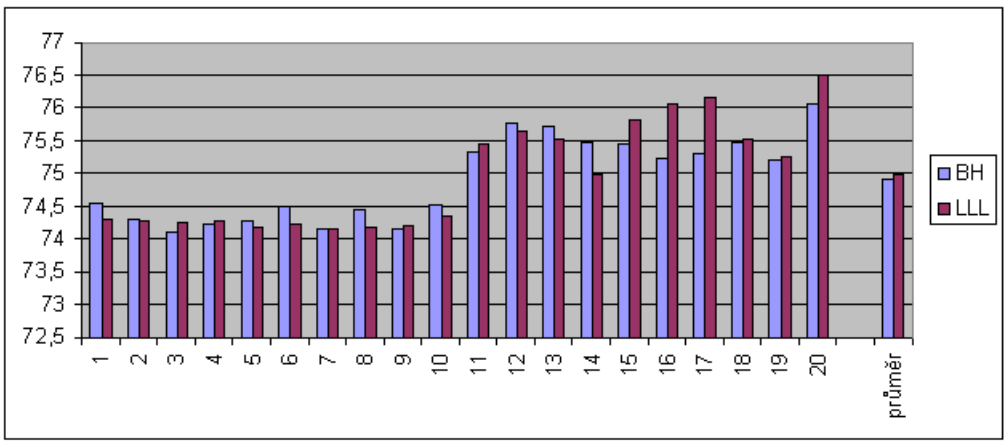
[4] Stanovský David, texty k prednášce Počítačová algebra, `http://www.karlin.mff.cuni.cz/~stanovsk/vyuka/skripta_palg.pdf`

[5] Silverman Joseph, texty k prednášce An Introduction to the Theory of Lattices and Applications to Cryptography.

Přílohy

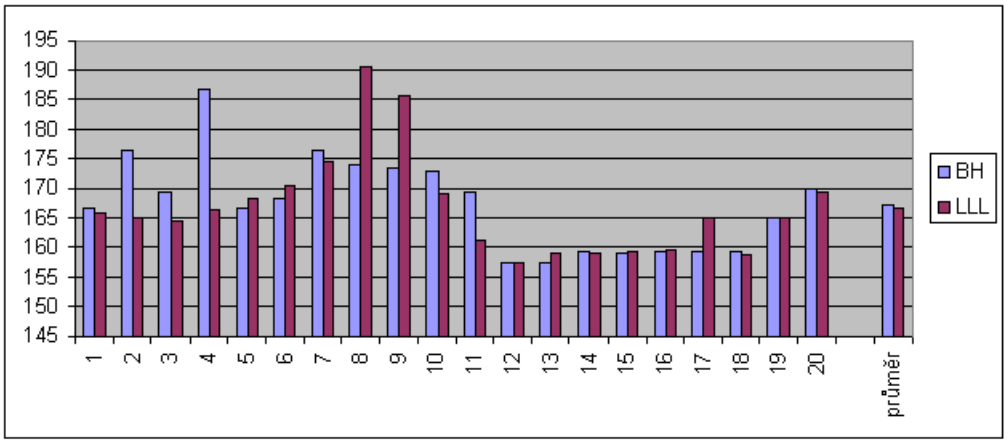
Test 1

Číslo měření	BH	LLL
1	74,562	74,297
2	74,297	74,282
3	74,109	74,25
4	74,219	74,281
5	74,281	74,172
6	74,5	74,234
7	74,156	74,141
8	74,438	74,172
9	74,156	74,188
10	74,516	74,359
11	75,313	75,438
12	75,766	75,657
13	75,734	75,532
14	75,469	74,985
15	75,438	75,828
16	75,219	76,047
17	75,297	76,141
18	75,469	75,532
19	75,203	75,25
20	76,047	76,5
Průměr	74,90945	74,9643
Procentuální poměr	100%	100,0732%



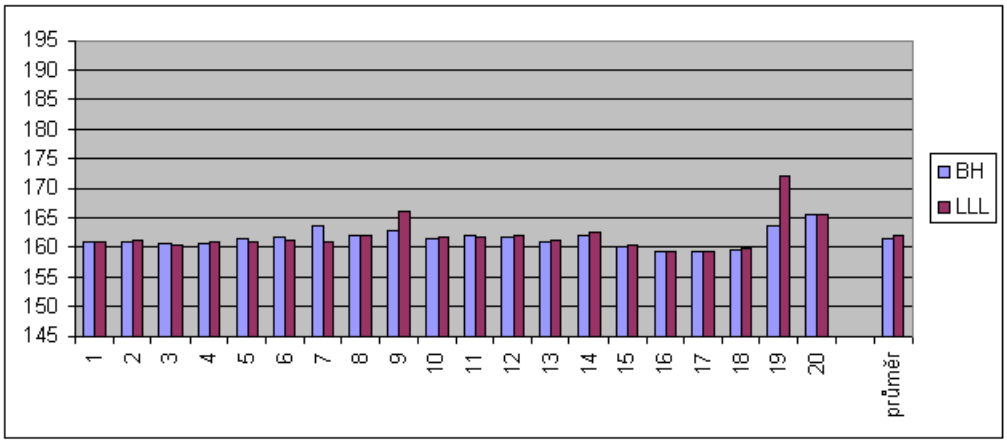
Test 2

Číslo měření	BH	LLL
1	166,641	165,703
2	176,36	164,875
3	169,421	164,541
4	186,625	166,344
5	166,735	168,26
6	168,391	170,422
7	176,484	174,453
8	173,984	190,422
9	173,453	185,516
10	172,734	169,235
11	169,375	161,182
12	157,594	157,421
13	157,453	158,843
14	159,25	158,968
15	158,938	159,094
16	159,187	159,437
17	159,031	164,906
18	159,266	158,64
19	164,867	164,997
20	170,009	169,561
Průměr	167,2899	166,641
Procentuální poměr	100%	99,6121%



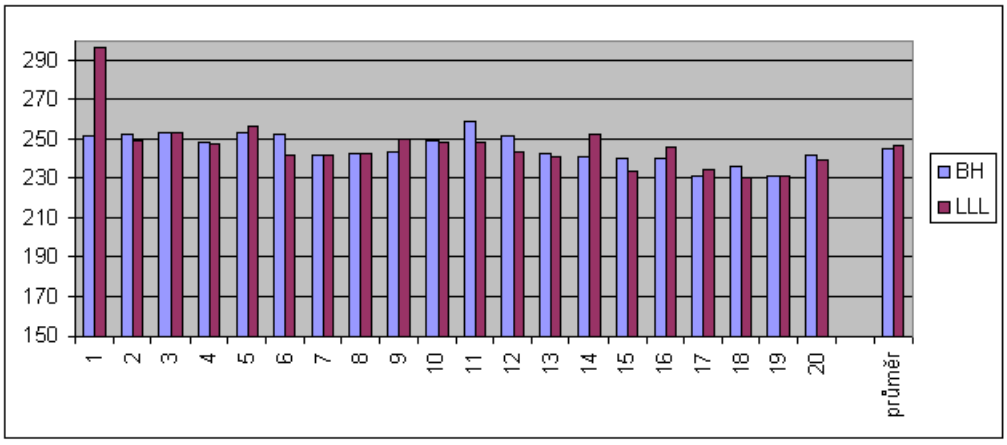
Test 3

Číslo měření	BH	LLL
1	160,967	160,875
2	160,875	160,985
3	160,672	160,36
4	160,641	160,875
5	161,484	160,703
6	161,61	161,125
7	163,656	160,797
8	162	161,984
9	162,813	166,141
10	161,281	161,688
11	161,813	161,734
12	161,563	161,969
13	160,875	161,125
14	162,063	162,438
15	160,016	160,344
16	159,297	159,188
17	159,125	159,11
18	159,328	159,61
19	163,609	172,078
20	165,484	165,578
Průměr	161,4586	161,93535
Procentuální poměr	100%	100,2952%



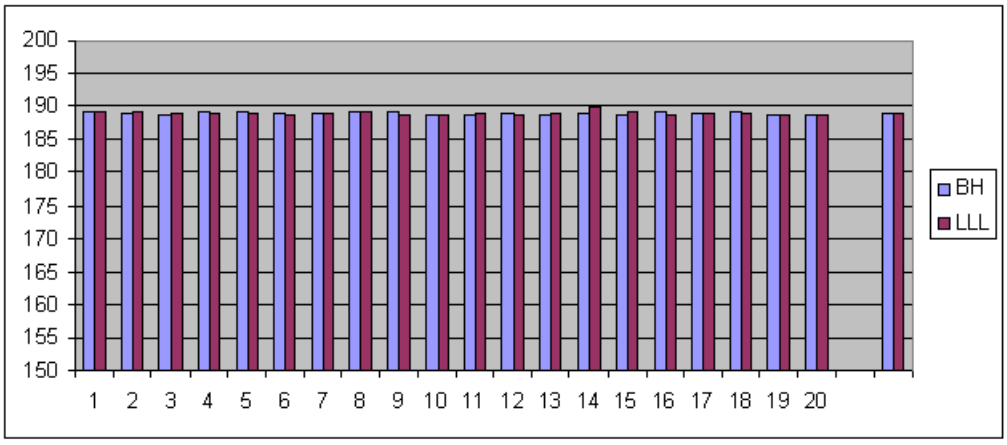
Test 4

Číslo měření	BH	LLL
1	251,546	296,679
2	252,878	248,816
3	253,547	253,703
4	248,422	247,578
5	252,953	257,078
6	252,235	241,984
7	241,875	241,828
8	242,219	242,875
9	243,156	249,641
10	249,5	248,75
11	258,766	248,078
12	251,531	243,593
13	242,188	240,969
14	240,563	252,829
15	240,047	233,734
16	240,047	245,422
17	231,187	234,219
18	235,813	230,141
19	230,594	230,469
20	242,031	239,563
Průměr	245,0549	246,39745
Procentuální poměr	100%	100,5478%



Test 5

Číslo měření	BH	LLL
1	189,25	189,406
2	189,015	189,218
3	188,859	188,969
4	189,234	189,125
5	189,312	189,032
6	189,109	188,922
7	189,172	189,109
8	189,297	189,203
9	189,297	188,812
10	188,75	188,922
11	188,922	188,937
12	189,062	188,922
13	188,797	189
14	189,094	189,922
15	188,922	189,203
16	189,25	188,844
17	189,016	188,953
18	189,25	188,969
19	188,906	188,828
20	188,828	188,766
Průměr	189,0671	189,0531
Procentuální poměr	100%	99,9925%



Test 6

Číslo měření	BH	LLL
1	377,328	376,312
2	376,5	373,781
3	374,625	376,078
4	376,312	378,968
5	376,422	375,375
6	379,047	379,36
7	383,109	380,688
8	376,953	375,031
9	374,5	375,641
10	391,953	372,797
11	374,422	374,921
12	374,906	373,531
13	405,328	399,156
14	394,546	394,796
15	395,781	403,344
16	409,328	409,682
17	380,536	381,012
18	392,301	391,989
19	385,722	387,351
20	378,638	373,632
Průměr	383,91285	382,67225
Procentuální poměr	100%	99,6768%