HLEDÁNÍ WIEFERICHOVÝCH PRVOČÍSEL

PETR LEŽÁK

Abstrakt. Článek pojednává o současném stavu hledání Wieferichových prvočísel. Jsou zde navrženy metody, jak toto hledání urychlit, a tyto metody jsou implementovány v software na jejich hledání. Tento software byl pak použit na vyhledání Wieferichových prvočísel o základu $2 - 9999$ menších než $3,5 \cdot 10^{10}$ a výsledky jsou v článku prezentovány. Dále je v článku navržen pravděpodobnostní model pro odhad hustoty Wieferichových prvočísel a tento model je porovnán s reálnými daty.

1. Úvod

Podle malé Fermatovy věty platí pro každé prvočíslo p a přirozené číslo a nedělitelné p rovnice (1.1), jak je uvedeno například v [1]. Wieferichovo prvočíslo o základu a navíc splňuje rovnici (1.2), viz [2].

$$a^{p-1} \equiv 1 \pmod{p} \tag{1.1}$$

$$a^{p-1} \equiv 1 \pmod{p^2} \tag{1.2}$$

Není-li uvedeno jinak, tak se většinou uvažují Wieferichova prvočísla o základu $a = 2$, pro něž jsou známa jen dvě řešení rovnice (1.2): $p = 1093$ a $p = 3511$. Tabulka 1 ukazuje, jak se v průběhu let podařilo rozšířit prozkoumaný interval čísel. Údaje jsou převzaty z [2] a [3]. Přesto se dosud nepodařilo najít třetí Wieferichovo prvočíslo. Důvody tohoto neúspěchu jsou prodiskutovány v sekci 8, kde je navržen pravděpodobnostní model hustoty Wieferichových prvočísel.

Cílem výzkumu bylo nalezení prvního a druhého Wieferichova prvočísla o základu $2 \leq a \leq 9999$. V současnosti není znám lepší algoritmus hledání Wieferichových prvočísel, než je prosté dosazování prvočísel do rovnice (1.2) a ověření její platnosti. Problémem tedy není, jak Wieferichova prvočísla hledat, ale jak je hledat co nejrychleji.

Protože obecné knihovny pro výpočty s velkými čísly jsou pomalé, tak bylo nutné vytvořit knihovnu optimalizovanou pro danou úlohu. Dále byl v prohledáván využit fakt, že hledáme Wieferichova prvočísla z velkého rozsahu základů.

2010 *MSC.* Primární 11A41.

Klíčová slova. Wieferich, prvočíslo, algoritmus, reprezentace.

Práce byla podporována projektem A-Math-Net – Síť pro transfer znalostí v aplikované matematice (CZ.1.07/2.4.00/17.0100).

Rok	Hranice	Vědec nebo organizace
1940	1.6×10^4	Beeger
1960	1×10^5	Kravitz
1964	5×10^5	Riesel
1971	3×10^9	Brillhart, Tonascia, Weinberger
1981	6×10^9	Lehmer
1996	6.1×10^{10}	Clark
1997	4×10^{12}	Crandall, Dilcher, Pomerance
2001	4.6×10^{13}	Brown, McIntosh
2005	1.25×10^{15}	Knauer, Richstein
2013	1.07×10^{17}	PrimeGrid

Tabulka 1. Prozkoumaný interval.

2. Výpočet Wieferichových prvočísel s různými základy

Wieferichova prvočísla se stejným exponentem jsou vždy hledána v jednom kroku. Lze tak využít různé optimalizace. Zavedeme-li označení (2.1) představující levou stranu rovnice (1.2), lze ukázat, že platí rovnice (2.2).

$$e(a,p) = a^{p-1} \bmod p^2 \tag{2.1}$$

$$e(ab,p) \equiv (ab)^{p-1} \equiv a^{p-1}b^{p-1} \equiv e(a,p) \cdot e(b,p) \pmod{p^2} \tag{2.2}$$

Díky ní je nutné vypočítat hodnoty $e(a,p)$ modulárním umocňováním jen pro prvočíselné základy a, hodnoty $e(a,p)$ pro složené základy a lze dopočítat mnohem rychleji pomocí modulárního násobení. Důsledkem rovnice (2.2) je i fakt, že je-li p Wieferichovo prvočíslo o základu a, pak je p také Wieferichovo prvočíslo o základu a^n.

K testování prvočíselnosti je využito Eratosthenovo síto, viz [4]. Protože je prosívání pomocí velkých prvočísel neefektivní (prvočíslo 2 odstraní každé druhé složené číslo, zatímco prvočíslo 997 zhruba každé tisícé), tak se prohledávaný interval prosívá jen prvočísly do určité velikosti. Výstupem tohoto kroku jsou proto tzv. pseudoprvočísla - tedy všechna prvočísla a některá složená čísla, která prošla sítem. Dále lze ukázat, že z platnosti $p \mid p^2$, plyne rovnice (2.3).

$$(x \bmod p^2) \bmod p = (x + n_1 \cdot p^2) + n_2 \cdot p = x + (n_1 p + n_2) \cdot p = x \bmod p \tag{2.3}$$

Toho lze využít k provedení Fermatova testu prvočíselnosti téměř bez dodatečné režie. Hodnotu $e(a,p) = a^{p-1} \bmod p^2$ spočítáme při testování Wieferichova prvočísla. Pokud se ukáže, že $e(a,p) \bmod p \neq 1$, pak není splněna rovnice (1.1) a p proto není prvočíslo. Nebudeme proto dále počítat $e(a,p)$ pro ostatní základy a a přejdeme k dalšímu pseudoprvočíslu p. Takto lze zavčas rozpoznat složená čísla, která prošla Eratosthenovým sítem, dříve, než jim věnujeme příliš mnoho strojového času, a přitom tento test nás prakticky nic nestojí.

3. Modulární umocňování

Pro ověření platnosti rovnice (1.2) je nutné provádět modulární umocňování. Před zahájením výpočtu jsou vypočítány mocniny

$$m(a,x) = a^x = a \cdot a^{x-1} = a \cdot m(a,x-1); m(a,0) = 1$$

pro všechny prvočíselné základy a a pro $m(a,x) < 2^{64}$.

Mocnina je pak spočítána podle rovnice (3.1), přičemž n se rozloží podle vztahu (3.2) na součet skupin bitů x_i začínajících na bitech s indexem s_i. Skupiny bitů x_i jsou voleny tak, aby $m(a,x_i) < p$. Tento algoritmus vychází z [5].

$$a^n \equiv \prod_{i=1}^{k} \left(m(a,x_i)\right)^{2^{s_i}} \pmod{c} \tag{3.1}$$

$$n = \sum_{i=1}^{k} x_i \cdot 2^{s_i}; s_1 = 0, s_i < s_{i+1} \tag{3.2}$$

Důkaz platnosti rovnice (3.1):

$$a^n \equiv a^{\sum_{i=1}^{k} x_i \cdot 2^{s_i}} \equiv \prod_{i=1}^{k} a^{x_i \cdot 2^{s_i}} \equiv \prod_{i=1}^{k} \left(a^{x_i}\right)^{2^{s_i}} \equiv \prod_{i=1}^{k} \left(m(a,x_i)\right)^{2^{s_i}} \pmod{c}$$

Rovnici (3.1) lze přepsat iterativně, což vede k efektivnějšímu výpočtu:

$$\begin{aligned} r_k &= m(a,x_k) \\ r_{i-1} &= \left(r_i\right)^{2^{s_i - s_{i-1}}} \cdot m(a,x_k) \bmod c \\ a^n \bmod c &= r_1 \end{aligned}$$

Výpočet $\left(r_i\right)^{2^{s_i - s_{i-1}}}$ je realizován pomocí $s_i - s_{i-1}$ modulárního umocňování nadruhou.

4. Reprezentace velkých čísel a modulární násobení

Budeme předpokládat, že vytvořený program poběží na 64-bitovém procesoru, protože prakticky všechny nové osobní počítače mají 64-bitový procesor. Pro uložení maximálního prozkoumaného prvočísla p je podle tabulky 1 nutné použít 54 bitů. Pokud pro uložení prvočísla p použijeme 64 bitový registr, budeme pracovat s dostatečnou rezervou. Zbytek po dělení p^2 je pak nutné reprezentovat pomocí 128 bitů, tedy dvou registrů.

Pro implementaci modulárního umocňování je třeba mít k dispozici algoritmus modulárního násobení čísel. Proto zvolená reprezentace musí být taková, aby modulární násobení bylo pokud možno co nejefektivnější. Přirozeně lze zbytek po dělení reprezentovat dvěma číslicemi x_0 a x_1 čísla v soustavě o základu 2^{64}. Lepší je ale pracovat v soustavě o základu p. Libovolné číslo $0 \leq x < p^2$ lze pak reprezentovat podle rovnice (4.1).

$$x = x_1 p + x_0; 0 \leq a, b < p \tag{4.1}$$

Výpočet $z = x \cdot y \bmod p^2$ popisuje rovnice (4.2).

$$\begin{aligned} z = x \cdot y &\equiv (x_1 p + x_0) \cdot (y_1 p + y_0) \equiv x_1 y_1 p^2 + (x_1 y_0 + x_0 y_1) p + x_0 y_0 \\ &\equiv (x_1 y_0 + x_0 y_1) p + x_0 y_0 \equiv (x_1 y_0 + x_0 y_1 + k - lp) \cdot p + (x_0 y_0 - kp) \pmod{p^2}, \end{aligned} \tag{4.2}$$

kde

$$k = \left\lfloor \frac{x_0 y_0}{p} \right\rfloor, l = \left\lfloor \frac{x_1 y_0 + x_0 y_1 + k}{p} \right\rfloor.$$

Rovnici (4.2) lze implementovat pomocí následujícího algoritmu výpočtu:

- Vstupy: $x = x_1 p + x_0; y = y_1 p + y_0; p$
- Výstup: $z = z_1 p + z_0 = x \cdot y$
- $t_1 = x_0 y_0$
- $k = \left\lfloor \frac{t_1}{p} \right\rfloor$
- $z_0 = t_1 - kp$
- $t_2 = x_1 y_0 + x_0 y_1 + k$
- $l = \left\lfloor \frac{t_2}{p} \right\rfloor$
- $z_1 = t_2 - lp$

Srovnáme-li algoritmus s modulárním násobením čísel reprezentovaných v soustavě o základu 2^{64}, tak odpadne jeden člen při násobení a stačí jen dvě dělení p oproti redukci modulo p^2.

5. Redukce

Jak bylo popsáno v sekci 4, tak pro modulární násobení je třeba provádět dělení čísla o velikosti nejvýše 128 bitů číslem p o velikosti nejvýše 64 bitů. Na současných procesorech je však dělení cca 14x pomalejší než násobení, viz [6]. Proto je vhodné využít algoritmus, který dělení nahradí i několika násobeními. Jedna z možností je využít Barrettův algoritmus, který je detailně popsán v [7]. Algoritmus je vhodný tehdy, pokud je třeba provést mnoho dělení se stejným dělitelem p. Vychází z myšlenky, že dělení lze nahradit násobením převrácenou hodnotou čísla, toto násobení je ale třeba uzpůsobit aritmetice celých čísel.

Nejdříve se pro dané p předpočítá $\mu = \left\lfloor \frac{2^{128}}{p} \right\rfloor$. Dělení se pak provádí podle rovnice (5.1).

$$\left\lfloor \frac{a}{p} \right\rfloor = \left\lfloor \frac{\mu \cdot a}{2^{128}} \right\rfloor + k \tag{5.1}$$

První člen rovnice (5.1) představuje odhad podílu a korekce k je odchylka od skutečné hodnoty podílu. Čísla μ a a mají velikost 128 bitů, každé je tedy reprezentováno dvěma registry o velikosti 64 bitů. Rovnice (5.2) určuje způsob výpočtu prvního členu rovnice (5.1).

$$\left\lfloor \frac{\mu \cdot a}{2^{128}} \right\rfloor = \left\lfloor \frac{(2^{64}\mu_1 + \mu_0) \cdot (2^{64} a_1 + a_0)}{2^{128}} \right\rfloor \approx \mu_1 a_1 + \left\lfloor \frac{\mu_1 a_0 + \mu_0 a_1}{2^{64}} \right\rfloor \tag{5.2}$$

Jedno dělení se tak nahradí třemi násobeními a následně ještě jedním násobením p pro výpočet zbytku po dělení. Korekce k se dopočítá ve smyčce po dělení tak, aby

zbytek dělení $r = a - p \cdot \left\lfloor \frac{a}{p} \right\rfloor$ splňoval podmínku $0 \leq r < p$. Barettův algoritmus zaručuje, že $-1 \leq k \leq 2$.

6. Celkový algoritmus

Nyní všechny uvedené dílčí kroky sestavíme do algoritmu pro hledání Wieferichových prvočísel.

- Předpočítej hodnoty $m(a, x)$ pro prvočíselná a a pro $m(a, x) < 2^{64}$.
- Rozděl prohledávaný interval na intervaly konstantní délky. V nich vyhledej pseudoprvočísla Erastothenovým sítem.
- Pro každé pseudoprvočíslo p dělej:
  - Pro každý prvočíselný základ $a < p$ vypočítej $e(a, p) = a^{p-1} \bmod p^2$, pokud pro některý prvočíselný základ a platí $e(a, p) \bmod p \neq 1$, tak další hodnoty $e(a, p)$ již nepočítej a přejdi k dalšímu pseudoprvočíslu p.
  - Každý složený základ $a < p$ rozlož na součin 2 čísel $a = b \cdot c$ a vypočítej $e(a, p) = e(b, p) \cdot e(c, p) \bmod p^2$, postupuj od nejnižšího základu k nejvyššímu.
  - Pokud $e(a, p) = 1$, tak zapiš $[a, p]$ do výsledků.

Hodnoty základů $a \geq p$ nemohou být tímto algoritmem spočítány, protože by docházelo k chybnému modulárnímu umocňování, proto byly spočítány s využitím standardních knihoven pro práci s velkými čísly. Je jich relativně malé množství, proto pomalejší výpočet nebyl na škodu.

7. Dosavadní výsledky

Nalezená Wieferichova prvočísla jsou umístěna na stránce [8] na záložce files v souboru results/table.txt. V tabulce 7 je výběr některých z nich. V intervalu 2 až $3,5 \cdot 10^{10}$ nebylo nalezeno žádné Wieferichovo prvočíslo pro celkem 433 základů. Nejvyšší nalezené Wieferichovo prvočíslo je $a = 4795, p = 34974531887$. Na uvedené stránce jsou i kompletní zdrojové kódy vytvořeného programu volně ke stažení.

8. Hustota Wieferichových prvočísel

Mohlo by nás zajímat, kolik Wieferichových prvočísel bychom měli ve zkoumaném intervalu najít. Uvažujme hodnotu $e(a, p) = a^{p-1} \bmod p^2$ jako náhodný pokus. Výsledek modulo p^2 může nabývat celkem p^2 různých hodnot. Pouze každá p-tá hodnota ale zároveň vyhovuje rovnici (1.1). Pravděpodobnost, že prvočíslo p je Wieferichovo, je proto určena rovnicí (8.1).

$$P\left(a^{p-1} \equiv 1 \pmod{p^2}\right) = \frac{1}{p} \tag{8.1}$$

Protože je hustota prvočísel rovna $\frac{1}{ln(p)}$, tak je celkový počet prvočísel v intervalu $p_1 \leq p < p_2$ popsán rovnicí (8.2), N zde představuje počet základů a.

a	p
2	1093, 3511
3	11, 1006003
4	1093, 3511
5	2, 20771, 40487, 53471161, 1645333507, 6692367337
6	66161, 534851, 3152573
7	5, 491531
8	3, 1093, 3511
9	2, 11, 1006003
10	3, 487, 56598313
11	71
9996	72461
9997	2, 7, 15973
9998	3, 31
9999	5, 25763597

Tabulka 2. Vybraná Wieferichova prvočísla.

$$n = N \sum_{p=p_1}^{p_2-1} \frac{1}{p \cdot \ln(p)} \approx N \int_{p_1}^{p_2} \frac{dp}{p \cdot \ln p} = N \left(\ln |\ln p_2| - \ln |\ln p_1| \right) \tag{8.2}$$

V této rovnici byla suma aproximována určitým integrálem. Dosadíme do této rovnice $p_1 = 10^k, p_2 = 10^{k+1}$ a získáme tak počet čísel v k-té dekádě:

$$n(k) = N \ln \left(\frac{\ln p_2}{\ln p_1} \right) = N \ln \left(\frac{\log 10^{k+1}}{\log 10^k} \right) = N \ln \left(1 + \frac{1}{k} \right)$$

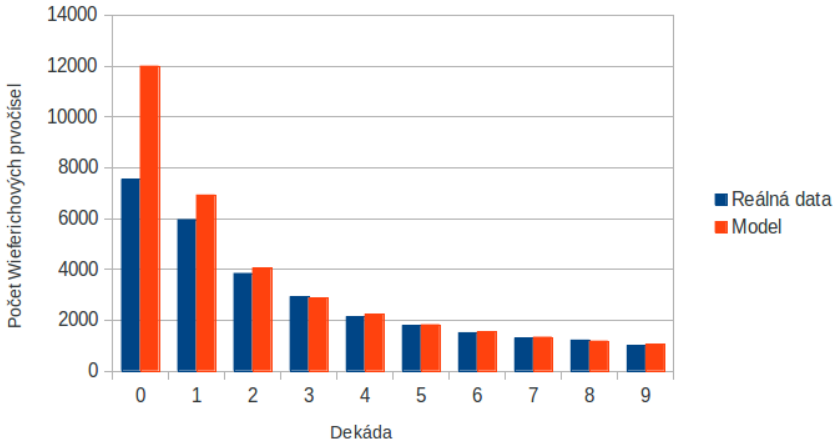
Povšimněme si, že $\lim_{n\to\infty} n(k) = N \ln 1 = 0$, s rostoucím číslem dekády k tak klesá počet Wieferichových prvočísel v ní obsažených. Přestože tedy budeme vynakládat stále větší úsilí (počet prvočísel v dekádě, která je nutné prozkoumat, roste), budeme nacházet stále méně Wieferichových prvočísel.

Obrázek 1 zobrazuje předpokládaný a nalezený počet Wieferichových prvočísel v jednotlivých dekádách. Povšimněme si, že kromě prvních 2 dekád, kde je počet prvočísel p relativně malý, a nejsou tak splněny předpoklady použité pro odvození modelu, model prakticky koresponduje s reálnými daty.

9. Závěr

Na základě uvedeného výzkumu byla publikována tabulka obsahující nalezená Wieferichova prvočísla. Ukázalo se, že hledání Wieferichových prvočísel s N základy současně je mnohem méně náročné, než prohledávání jednotlivých základů odděleně. Vytvořený software je publikován v [8].

Dalšího rozšíření intervalu p je možné dosáhnout distribucí výpočtu na mnoho počítačů. Na tomto problému aktuálně pracuji a nalezené výsledky budu v budoucnu publikovat.



Obrázek 1. Četnost Wieferichových prvočísel.

Reference

[1] A. Manindra: *Primality tests based on Fermats little theorem*, `http://www.cse.iitk.ac.in/users/manindra/survey/FLTBasedTests.pdf`, 7 pp.
[2] F. G. Dorais, D. Klyve: *A Wieferich Prime Search up to 6.7 x* 10^{15}, Journal of Integer Sequences **14** (2011), 14 pp, `https://cs.uwaterloo.ca/journals/JIS/VOL14/Klyve/klyve3.pdf`.
[3] *PrimeGrid's PRPNet: Wieferich Prime Search*, `http://prpnet.mine.nu:13000/`.
[4] M. E. O'Neill: *The Genuine Sieve of Eratosthenes*, `http://www.cs.hmc.edu/ oneill/papers/Sieve-JFP.pdf`, 12 pp.
[5] N. Smart: *Cryptography: An Introduction*, 3rd ed., `http://www.cs.bris.ac.uk/ nigel/Crypto_Book/`.
[6] Advanced Micro Devices, Inc.: *Software Optimization Guide for AMD64 Processors*, `http://support.amd.com/us/Processor_TechDocs/25112.PDF`, 384 pp.
[7] D. Hankerson, A. Menezes, S. Vanstone: *Guide to Elliptic Curve Cryptography*, New York, Springer, 2004.
[8] P. Ležák: *SourceForge: Wieferich*, `https://sourceforge.net/projects/wieferich/`.

Petr Ležák, Ústav telekomunikací, Fakulta elektrotechniky a komunikačních technologií, Vysoké učení technické v Brně, Technická 3082/12, 616 00, Brno, Česká republika,
e-mail: `petr.lezak@email.cz`