Gymnázium Přírodní škola, o.p.s.
Profilová práce — třída Mí
Vyšší stupeň studia
2020/2021

Matěj Zeman

Rozložení prvočísel na intervalech mocnin dvou a prvočíselné testy

Vedoucí práce: Ing. Mgr. Filip Hložek
Datum odevzdání: 12. listopadu 2020

Poděkování

Na úvod bych rád poděkoval všem, kteří mi poskytli cenné připomínky v průběhu tvorby této práce. V prvé řadě mému vedoucímu práce Ing. Mgr. Filipu Hložkovi a mému otci a zároveň odbornému konzultantovi Mgr. Martinu Zemanovi za vedení a držení ochranné ruky nad prací. V další řadě se pak jedná o Mgr. Davida Breberu a Mgr. Marka Cútha, Ph.D., jejichž zpětná vazba k prvotní verzi popisující bitové řezy pomohla práci výrazně vylepšit.

Úvod

V mé práci se věnuji především matematice, konkrétně zkoumání prvočísel. Původní motivací bylo spíše naučit se programovat a matematiku omezit na nezbytné minimum. Ale postupně jsem v rozporu s tímto předsevzetím pronikal do matematických otázek více a proto má současná práce vypovídá více o matematice než o programování.

Mé první zaměření v prosinci 2019 bylo na tak zvaný BPSW prvočíselný test. Záměrem bylo ověřit jeho spolehlivost nad dosud známou hranici a možná snad i najít BPSW pseudoprvočíslo. Rychle se však ukázalo, že tento cíl je velmi ambiciózní.

Zprvu bylo velmi pracné vybudovat všechny konfigurace vývojového prostředí a naprogramovat do něj jednoduché algoritmy pro test prvočíselnosti. Nicméně v dalším (a výrazně náročnějším kroku) jsem převzal již existující kódy pro BPSW a ECP testy a zahájil měření.

A v této fázi jsem se dostal do slepé uličky. Měření v březnu 2020 vypadala zcela beznadějně. Doba běhu programu pro dosažení hlavního cíle vycházela řádově na 100 milionů let. I přes poměrně značné úsilí o optimalizaci algoritmů se mi však doposud nepodařilo snížit tuto dobu běhu na přijatelný čas, který by umožňoval dosáhnout podstatného posunu dosud známé hranice před termínem odevzdání této práce (tedy v horizontu několika měsíců). Hlavní cíl práce byl formulován tak, že není záměrem hranici přímo posunout, ale změřit možnosti jejího zdvojnásobení v domácích podmínkách. Takový výsledek však nepředstavuje příliš inspirativní čtení. Zároveň jsem se nevyhnul určitému zklamání a také pochybám, zda přece jen neexistuje cesta, jak dosáhnout výrazného zrychlení výpočtů.

Již před samotným zahájením prací bylo zřejmé, že pochopit podstatu BPSW testu by vyžadovalo studium pokročilých částí matematiky, které vysoce převyšují moji úroveň. Předpokládal jsem, že test bude možné pouze uživatelsky pouštět a že tento nedostatek nebude na překážku. Dnes vím, že to byl mylný předpoklad. Snahy o výpočetní zrychlení BPSW testu vyžadují pochopení jeho podstaty, konkrétního algoritmu výpočtu a porozumění toho, co stojí počítač nejvíce času.

Zpočátku vypadala velmi nadějně myšlenka nahrazení specializované knihovny pro práci s velkými čísly (GNU MP) datovým typem integer s rozsahem 128 bitů, do něhož by se výpočet převedl. Tento datový typ zatím netvoří[1] součást standardu jazyků C ani C++, ale v praxi je již možné jej využívat. Po pochopení podstaty BPSW testu se však ukázalo, že ani 128 bitový datový typ nestačí pro některé pomocné výpočty BPSW testu. Konkrétně algoritmus požaduje opakovaný výpočet zbytku po dělení čísla 2 umocněného na testované prvočíslo (funkce *mpz_powm* z knihovny GNU MP), což není jednoduše uskutečnitelné uvnitř 128 bitového datového typu. A podobných slepých uliček bylo více.

Během přemýšlení a různých úvah, zda se podaří tuto zeď prorazit, jsem se postupně začal věnovat zcela jinému tématu, kdy jsem využil již zhotovených algoritmů pro testování prvočísel a začal zkoumat jejich rozložení na intervalech mocnin 2. Tato myšlenka vznikla náhodou při pronikání do hlubin programování v jazyku C a později i C++. Při objevování toho, jakým způsobem počítač ukládá a pracuje s celočíselnými datovými typy a jak jejich vnitřní reprezentace souvisí s dvojkovou soustavou, mě napadlo prozkoumat, zda reprezentace

[1] https://gcc.gnu.org/onlinedocs/gcc/_005f_005fint128.html

prvočísel ve dvojkové soustavě není něčím zajímavá. Postupně jsem tomuto tématu věnoval více a více času. Dnes leží těžiště mé práce právě v této původně neplánované odbočce.

Práce se tedy skládá ze dvou oddělených celků. První se věnuje různým metodám testování prvočíselnosti. Od nejjednodušších metod za pomoci dělení nebo Eratosthenova síta postupně přechází k BPSW pravděpodobnostnímu testu a ECP testu. Hlavním cílem první části je měření výkonnosti těchto metod pro čísla v rozsahu $2^{64} - 2^{65}$, tedy v rozsahu, na který nestačí 64-bitový datový typ dnešních počítačů. Výsledkem první části je závěr, že v běžných domácích podmínkách není realistické prověřit spolehlivost BPSW testu v uvedeném rozsahu. Věnoval jsem se i zajímavé otázce složitosti výpočtu koeficientu D vyžadovaným Lucas-Selfridgeovým testem.

Druhá část práce se pak zabývá rozložením prvočísel na intervalech mocniny dvou. Ústřední otázkou je studium prvočísel v jejich dvojkovém rozvoji a rovnoměrnost jejich rozložení při povinném nastavení dvou bitů. V závěrečné části je pak zkoumáno rozložení prvočísel s jinou podmínkou: povinném nastavení zvoleného počtu bitů. Toto zkoumání vedlo na objevení pilovité anomálie v rozložení prvočísel u lichých řezů, které se nepodařilo vysvětlit a představuje tak nejzajímavější část práce.

BPSW test

V mnoha odvětvích matematiky se řeší úloha, zda je nějaké velké číslo prvočíslo, či ne. Test, zda x je prvočíslo, je výpočetně náročný. V praxi se proto často používají pravděpodobnostní testy, které jsou výpočetně mnohem rychlejší než tzv. deterministické testy. Například známé aplikace Maple a Mathematica údajně využívají tzv. BPSW pravděpodobnostní test (a asi jej kombinují i s dalšími pravděpodobnostními testy).

BPSW test[2] byl vytvořený čtyřmi matematiky: Robert Baillie, Carl Pomerance, John Selfridge a Samuel Wagstaff, podle nichž se test jmenuje. O tomto testu víme, že pro všechna čísla až do 2^{64} (18 446 744 073 709 551 616) vrací správný výsledek, tzn. že všechna čísla označená jako pravděpodobná prvočísla opravdu jsou prvočísla. Zároveň ale jeden z otců testu (Carl Pomerance) v roce 1984 předložil hypotézu[3], že existuje nekonečně mnoho tzv. BPSW pseudoprvočísel. BPSW pseudoprvočíslo je takové číslo, o kterém BPSW test tvrdí, že je pravděpodobně prvočíslem, ale ve skutečnosti je to číslo složené. Na objevení (slabého) BPSW pseudoprvočísla je již od roku 1980 vypsána autory testu odměna 30$. Podobné drobné odměny bývají často vyplaceny krátce po jejich vypsání. Toto však není případ odměny na BPSW pseudoprvočíslo, které domnívám se představuje oproti původnímu očekávání výrazně složitější výzvu, než zřejmě autoři testu předpokládali.

Cílem profilové práce je otestovat čísla i za hranicí 2^{64} a zjistit, zda se za touto hranicí nevyskytuje BPSW pseudoprvočíslo. Pokud nedojde k objevení BPSW pseudoprvočísla, dojde naopak k rozšíření hranice, kdy je možné tento test bez obav používat místo pomalejších deterministických testů. V ideální variantě by došlo k otestování čísel v rozmezí od 2^{64} až do 2^{65}, čímž by došlo ke zdvojnásobení dosud známé hranice spolehlivosti testu.

Existují indicie[4], že BPSW pseudoprvočíslo nebude v testované hranici existovat. Konkrétně program PRIMO, obsahuje ve standardním běhu i test na BPSW pseudoprvočíslo. Za několik let jeho běhu se dosud žádný kandidát neověřil jako BPSW pseudoprvočíslo a autor programu M. Martin, který je zároveň matematik zkoumající eliptické křivky, odhaduje, že nebude existovat žádné BPSW pseudoprvočíslo menší než $10^{10\,000}$

Výpočetní složitost úlohy (a tím i realistické dosažení cíle) nebylo možné stanovit dopředu, zejména s ohledem na skutečnost, že současné počítačové procesory založené na 64bitové architektuře zvládají operace dělení pouze pro čísla do 2^{64}. Pro provádění matematických operací pro větší čísla je nutné používat speciální knihovny (např. GNU MP[5]), které dokáží pracovat v podstatě s libovolně velkými čísly. Nejsou tedy omezeny standardní velikostí datového typu 64bitové architektury počítačů a limit na jejich velikost je v praxi dán jen velikostí dostupné operační paměti počítače a výkonu procesoru. Umožňují tak operace s čísly, které mají například 100 milionů cifer. Nevýhodou těchto knihoven je výrazně pomalejší výpočet. Například i prosté dělení čísla je o několik řádů pomalejší. Není tedy jasné, zda by prověření všech čísel do zmíněné hranice trvalo měsíce, roky, či stovky milionů let.

[2] https://en.wikipedia.org/wiki/Baillie%E2%80%93PSW_primality_test
[3] https://faculty.lynchburg.edu/~nicely/misc/bpsw.html
[4] http://mathworld.wolfram.com/Baillie-PSWPrimalityTest.html
[5] https://gmplib.org/

Postup práce

Volba a příprava prostředí

Nejprve bylo nutné zvolit vhodné nástroje. Po prvotním průzkumu možností bylo zřejmé, že dostupné algoritmy jsou nejčastěji založené na jazyce C, který se používá pro implementaci algoritmů, kde jde v prvé řadě o rychlost a efektivitu výpočtů. Pro jazyk C existuje několik různých překladačů. Nejvíce používané jsou Microsoft C, Clang a GCC.

Pohledem do zdrojových kódů některých algoritmů pro výpočet prvočísel bylo jasné, že autoři se nejvíce opírali o překladač g++ ze sady kompilátorů GNU Compiler Collection GCC[6] a o knihovnu pro práci s velkými čísly GNU MP evidentně v prostředí Linux.

Linux, Ubuntu, WSL

Prvním krokem tedy bylo zajistit prostředí Linux:

- Nakonec jsem zvolil prostředí Ubuntu[7] ve verzi 18 a později i ve verzi 20. Využil jsem možnost rozběhnout Ubuntu přímo ve Windows 10 pod WLS[8] (Windows Linux Subsystem) verze 1, později Microsoft vydal aktualizovanou verzi WSL verze 2. Jedná se o relativně nedávnou aktivitu společnosti Microsoft umožňující běh některých distribucí Linuxu přímo pod Windows, která využívá schopnosti virtualizace současného hardwaru.
- Zároveň jsem nainstaloval plnohodnotný operační systém Xubuntu 18 v režimu dual boot, který jsem používal jen pro porovnání rychlosti běhu algoritmu pod WLS a přímo pod Linuxem. Ukázalo se, že běh algoritmů pod Linuxem byl rychlejší jen o necelé jedno procento, takže jsem pak nadále využíval systém WLS pod Windows, na kterém proběhla i veškerá měření.

Výpočty byly provedeny na domácím notebooku: Lenovo Yoga 720 s procesorem Intel i7 7700HQ @ 2.8GHz, 16GB RAM.

Překladač C++

V průběhu práce jsem používal různé verze překladače g++, při dokončení pak verzi z GCC 10.2. Při překladu jsem pak vynucoval verzi standardu C++ z roku 2017 zejména z důvodu dostupnosti standardizované knihovny pro práci se souborovým systémem (tzv. knihovna std::filesystem[9]), která v nižších verzích standardu C++ není dostupná.

V případě potřeby tvorby programu by bylo možné zdrojový kód přeložit přímo pod implementací GCC pro prostředí Windows, konkrétně například v prostředí MinGW[10], které jsem ale dosud neuskutečnil. Je možné, že zdrojový kód by vyžadoval drobné úpravy (například při ukládání souboru s prvočísly do hranice 2^{32}). Zároveň by bylo obtížné provést optimalizaci knihovny GNU MP, pravděpodobně by se musela využívat verze dodávaná spolu s MinGW.

[6] https://en.wikipedia.org/wiki/GNU_Compiler_Collection
[7] https://en.wikipedia.org/wiki/Ubuntu
[8] https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux
[9] https://en.cppreference.com/w/cpp/filesystem
[10] https://en.wikipedia.org/wiki/MinGW

Jako vývojové prostředí jsem používal Visual Studio Code[11] od společnosti Microsoft, které velmi dobře spolupracuje se systémem WSL.

Knihovna pro práci s velkými čísly GNU MP

Většinu měření jsem pak provedl na GNU MP knihovny verze 6.2.0, která oproti starším verzím obsahuje optimalizaci klíčové funkce mpz_powm[12]. Knihovnu jsem pro dosažení lepšího výkonu linkoval staticky, což znamená, že celá GNU MP knihovna je přímo součástí vytvořeného programu. Tato volba má výhodu v rychlejším volání jednotlivých funkcí a drobnou nevýhodu ve větší velikosti programu v paměti počítače. Na druhou stranu dynamicky linkovaná knihovna může být sdílená vícero programy, což ale způsobuje malé zpoždění při volání funkcí knihovny. Je nutné vzít do úvahy, že pro otestování jednoho prvočísla je nutné mnoho volání různých funkcí knihovnu GNU MP. A vytvořený program má za úkol otestovat v podstatě nepředstavitelně mnoho čísel (jistě více než 2^{62}. Celý rozsah čísel je 2^{64}, ale netestujeme čísla sudá a dále násobky nízkých prvočísel). I drobná úspora při každém volání tak představuje přínos. Tato optimalizace vedla na zrychlení do 3% v závislosti na konkrétním algoritmu.

Knihovna GNU MP umožňuje i optimalizaci pro konkrétní počítač, kde dojde k otestování možností a konfigurace konkrétního počítače (procesor, velikost paměti...) a zároveň je možné nastavit i rozsáhlou sadu různých nastavení knihovny (například rozsah různých dopředu spočítaných hodnot, které šetří dobu běhu některých algoritmů na úkor požadované paměti). Takto nakonfigurovanou knihovnu je pak nutné přeložit. Tento krok nebyl zdaleka tak jednoduchý, jak se zdálo z dokumentace a nakonec jsem neuspěl. Představuje jednu z možností optimalizace algoritmů. Nicméně od této možnosti lze očekávat jen velmi drobné zlepšení výkonu, ne zásadní řešení výkonnostních problémů.

[11] https://code.visualstudio.com/
[12] https://gmplib.org/manual/Integer-Exponentiation

Deterministické testy

Ověření spolehlivosti BPSW pravděpodobnostního testu vyžaduje, aby každé číslo označené BPSW testem za pravděpodobné prvočíslo bylo následně prověřené na prvočíselnost deterministickým testem. Deterministické testy prvočíselnosti lze rozdělit na testy nazývané jednoduché, které byly známy již ve starověku a není je obtížné naprogramovat, a ostatní, které vychází z moderní matematiky a jejichž pochopení i algoritmy jsou velmi složité. Nejprve jsem se rozhodl vyzkoušet použitelnost jednoduchých testů.

Test prvočíselnosti dělením

Toto je nejjednodušší prvočíselný test. Testované číslo postupně dělíme a zkoumáme zbytek po dělení. Je-li zbytek po dělení nějakým dělitelem 0, testované číslo není prvočíslo, ale číslo složené. Pro prokázání prvočíselnosti je třeba postupně vyzkoušet všechna čísla až do odmocniny z testovaného čísla. Při tomto testu se nejčastěji používá jedna z následujících dvou optimalizací:

1. Po testu na dělitelnost 2 se již přeskakuje dělení všemi sudými čísly
2. Po testu na dělitelnost 2 a 3 se přeskakují všechny násobky čísel 2 a 3. Stačí pak otestovat jen čísla ve tvaru
   - $6k + 1$ a
   - $6k + 5$.

   Čísla v následujících tvarech jsou evidentně složená:

   - $6k$ jsou dělitelná čísly 2 a 3
   - $6k + 2$ jsou dělitelná číslem 2
   - $6k + 3$ jsou dělitelná číslem 3

Eratosthenovo síto, primoriál

Popsané optimalizace jsou speciálním případem tzv. Eratosthenova síta[13]. Popsaný přístup lze zobecnit i na násobky vyšších prvočísel. V mém případě se osvědčilo netestovat násobky těchto prvočísel: $2, 3, 5, 7, 11, 13, 17, 19, 23$. Přeskakovat vyšší prvočísla se postupně stává nepraktické. Toto tvrzení je potřeba důkladněji rozebrat. Nejprve však bude potřeba popsat nový matematický termín, tzv. primoriál, který je definován jako součet po sobě jdoucích prvočísel. Přesněji, je-li $p_n n$-té prvočíslo, označme jeho promoriál $p_n\#$ jako:

$$p_n\# = \prod_{k=1}^{n} p_k$$

Symbol Π je standardní označení pro součin prvků posloupnosti.

Příklady:

- $2\# = 2$
- $3\# = 2 \cdot 3 = 6$
- $5\# = 2 \cdot 3 \cdot 5 = 30$
- $7\# = 2 \cdot 3 \cdot 5 \cdot 7 = 210$
- $11\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2\,310$
- $13\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 = 30\,030$
- $17\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 = 510\,510$

[13] https://cs.wikipedia.org/wiki/Eratosthenovo_s%C3%ADto

- $19\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 9\,699\,690$
- $23\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 = 223\,092\,870$
- $29\# = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 = 6\,469\,693\,230$

Primoriál je funkce, která velmi rychle roste. Lze ji srovnávat s faktoriálem, který v součinu nevynechává složená čísla a roste tak ještě rychleji. Hodnoty primoriálů jsou vždy tzv. bezčtvercová čísla[14], neboť v jejich prvočíselném rozkladu není žádný činitel v druhé nebo vyšší mocnině.

Podívejme se, jak by vypadala úprava testu prvočíselnosti, ve kterém by se netestovaly všechny násobky prvočísel $2, 3, 5, 7$.

- Nejprve je třeba otestovat dělitelnost testovaného čísla na prvočísla $2, 3, 5, 7$.
- V dalším kroku postupně procházíme potenciální dělitele až do odmocniny z testovaného čísla.
- Potenciální dělitele musíme rozdělit do dvou skupin na ty, které jsou dělitelné prvočísly $2, 3, 5, 7$, a ty které dělitelné nejsou.
- Jak zjistit, zda potenciální dělitel x je, či není dělitelný prvočísly $2, 3, 5, 7$? Toto lze snadno rozhodnout následující úvahou:
    - Testovaného dělitele x čísla lze rozložit na součet $x = n \cdot 7\# + k$.
    - Jde o rozklad na násobek primoriálu a zbytek k, který nabývá hodnoty mezi 0 a $7\# - 1$.
    - Libovolný násobek primoriálu $n \cdot 7\#$ je nutně dělitelný všemi prvočísly $2, 3, 5, 7$.
    - Otázka, zda potenciální dělitel x je či není dělitelný prvočísly $2, 3, 5, 7$ se tedy redukuje na otázku, zda je, či není dělitelný zbytek k prvočísly $2, 3, 5, 7$.
    - Je-li zbytek k číslo soudělné s primoriálem $7\#$, je i celý potenciální dělitel $x$násobkem některého prvočísla $2, 3, 5, 7$.
    - Naopak, je-li zbytek k nesoudělný s primoriálem $7\#$, není potenciální dělitel x násobkem prvočísel $2, 3, 5, 7$ a je třeba jej použít v testu prvočíselnosti ke zkoumání zbytku po dělení.
- Uvedený příklad lze samozřejmě zobecnit pro libovolnou větší posloupnost prvočísel.

Po tomto příkladu je vhodný okamžik vrátit se k tvrzení, že sada prvočísel $2, 3, 5, 7, 11, 13, 17, 19, 23$ je pro algoritmus optimální.

- Nechť je nyní y testované číslo a $r = \sqrt{y}$.
- Nejprve je třeba zjistit zbytek po dělení testovaného čísla y postupně všemi prvočísly $2, 3, 5, 7, 11, 13, 17, 19, 23$.
- Dále je třeba oblast potenciálních dělitelů $0..r$ rozdělit na úseky dlouhé $23\# = 223\,092\,870$. Takových úseků je $\frac{r}{23\#}$ s tím, že poslední úsek může být kratší a první úsek má hodnotu $n = 0$.
- Uvnitř každého úseku n je potřeba pro všechna $k = 0..223\,092\,869$ rozhodnout, zda je nutné potenciálního dělitele použít, nebo jej rovnou přeskočit.
- Kritériem je nesoudělnost čísel k a $23\#$.
- Čísla k v intervalu $0..223\,092\,869$ lze otestovat na nesoudělnost s primoriálem jednou na začátku algoritmu a tuto informaci uložit do pomocného pole, odkud bude využívána opakovaně pro všechny úseky n .
- Je zřejmé, že jednorázový výpočet tohoto pole stojí určitý strojový čas a uložené pole zabírá paměť.

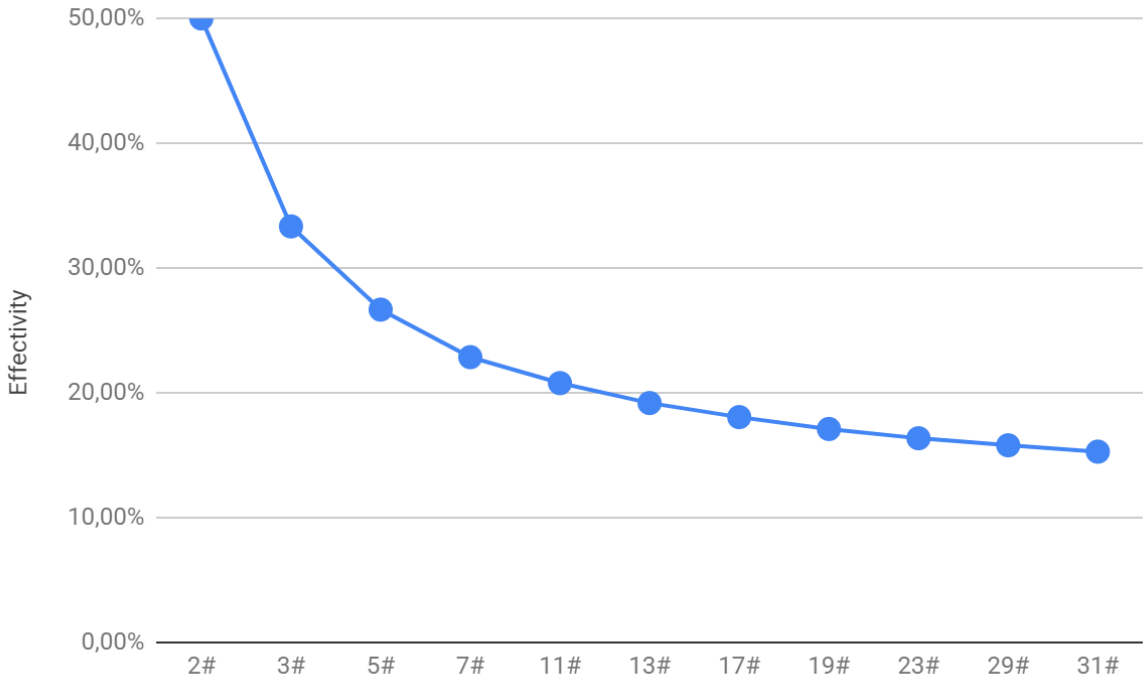
[14] https://cs.wikipedia.org/wiki/Bez%C4%8Dtvercov%C3%A9_cel%C3%A9_%C4%8D%C3%ADslo

- Při použití delší sady prvočísel tyto nároky prudce rostou. Zejména potřebná paměť pro uložení nesoudělných čísel začne poměrně rychle omezovat chod celého počítače a běh algoritmu nebo rovnou překročí dostupnou operační paměť počítače.
- V praxi je potřeba tedy sadu prvočísel optimalizovat. Hodnota $23\#$ představuje právě takovou rozumnou hranici.

Ukazuje se, že s rostoucí sadou prvočísel prudce roste délka úseků (daná hodnotou primoriálu) i počet nesoudělných čísel s příslušným primoriálem. Efektivita algoritmu je daná podílem těchto dvou hodnot. Nabízí se tedy otázka tuto efektivitu vyhodnotit spočtením procentuálního zastoupení hodnot k, které je nutné testovat, vůči délce úseku.

Tuto otázku pro několik prvních primoriálů řeší nově vytvořená metoda programu *EratosthenesEffectivity*. Výpočet se postupně zpomaluje. Je zajímavé, že tento výsledek se nepodařilo nikde na internetu nalézt.

Primoriál	Hodnota primoriálu	Nesoudělných dělitelů	Efektivita
2#	2	1	50,00%
3#	6	2	33,33%
5#	30	8	26,67%
7#	210	48	22,86%
11#	2 310	480	20,78%
13#	30 030	5 760	19,18%
17#	510 510	92 160	18,05%
19#	9 699 690	1 658 880	17,10%
23#	**223 092 870**	**36 495 360**	**16,36%**
29#	6 469 693 230	1 021 870 080	15,79%
31#	200 560 490 130	30 656 102 400	15,29%



Graf 1

Z výpočtu plyne, že při přeskakování násobků prvočísel $2, 3, 5, 7, 11, 13, 17, 19, 23$ je oblast do odmocniny z testovaného čísla rozdělena na úseky dlouhé přibližně 223 milionů a v paměti je nutné držet seznam asi 36 miliónů nesoudělných dělitelů, což vede na efektivitu 16,36%. Při přidání dalšího prvočísla by úseky byly dlouhé asi 6 470 milionů a seznam nesoudělných prvočísel by narostl asi na jednu miliardu, přičemž efektivita by se zlepšila jen nepatrně na 15,79%. Tím je úsudek o efektivitě řádně podložený.

Test prvočíselnosti pomocí dělení prvočísly

Pokud jsou k dispozici spočtená prvočísla až do odmocniny testovaného čísla, je možné předchozí algoritmus zefektivnit dělením pouze těmito prvočísly. Při testování čísel s horní hranicí 2^{65} je potřeba uložit do paměti počítače prvočísla do hranice $2^{32,5}$. Podívejme se na paměťové nároky. Prvočísla do 2^{32} je možné uložit do 16-bitového datového typu, zatímco vyšší prvočísla je nutné ukládat již do 64-bitového datového typu (a tedy do jiného pole), každé číslo zabírá dvojnásobek paměti. Prvočísel do 2^{32} je $203\,280\,221$ a zabírají necelých 776 MB paměti. Prvočísel nad tuto hranici menších než $2^{32,5}$ je $79\,550\,350$ a zabírají necelých 607 MB paměti. Běžný současný domácí počítač zvládne uložit do operační paměti obě tyto pole současně. Tato otázka nebyla do okamžiku naprogramování a praktického vyzkoušení vůbec jasná. Hrozilo riziko, že vypočtená prvočísla nebude možné v paměti počítače najednou ukládat, což by znamenalo, že bude algoritmus nepoužitelný pro praktické použití.

Výpočet těchto polí pomocí Eratosthenova síta zabírá řádově desítky minut, což na první pohled není mnoho. Přesto pro rychlé opakované spouštění programu (například při ladění) je naprosto nutné obě spočtené pole uložit do dvou souborů. Jejich načtení z disku pak při spuštění programu je otázka několika málo vteřin.

Test prvočíselnosti založený na kombinaci předchozích přístupů

Oba přístupy lze kombinovat. Takový algoritmus při výpočtu zbytku po dělení nejprve čerpá z předem připravených prvočísel menších než 2^{32}a po jejich vyčerpání pokračuje prvočísly v 64-bitovém datovém typu až po jejich vyčerpání. Pak algoritmus pokračuje Eratosthenovým sítem s přeskakováním násobků prvočísel $2,3,5,7,11,13,17,19,23$.

Použitelnost jednoduchých deterministických testů

Přestože uvedené testy postupně zlepšovaly výkonnost, ověření jednoho čísla ve zkoumaném rozmezí $2^{64} - 2^{65}$ trvalo řádově několik minut. I bez přesného výpočtu je okamžitě zřejmé, že uvedené jednoduché testy nejsou pro praktické ověření BPSW testu použitelné.

Bylo nutné najít efektivnější deterministický test. Po studiu výkonnosti existujících testů byl vybrán tzv. APR-CL test.

Deterministický APR-CL test

APR-CL[15] deterministický test vynalezený v roce 1984 je nazván podle iniciálů svých autorů: Leonard Adleman, Carl Pomerance, Robert Rumely. O rok později test dále vylepšili matematici Henri Cohen a Hendrik Willem Lenstra. Pro progam jsem převzal implementaci algoritmu nazvanou mpz_aprcl[16] poskytovaného zdarma pod licencí svobodného software GNU Library or Lesser General Public License[17] version 3.0 (LGPLv3).

Pravděpodobnostní testy

Všechny následující testy jsem převzal ze stránek[18] Thomase R. Nicelyho, který dal testy v roce 2009 volně k dispozici bez jakéhokoliv licenčního omezení (freeware). V kódu jsem provedl některé úpravy:

- Převod z jazyka C do jazyku C++.
- Odstranění statických proměnných pro budoucí podporu paralelního běhu.
- Převod na 64-bitové celočíselné proměnné

Při úpravách ve složitém kódu, kde není zřejmý smysl algoritmu, hrozí riziko zavlečené chyby. Test jsem tedy otestoval na všech prvočíslech vygenerovaných pomocí Eratosthenova síta, dále pak na několika větších prvočíslech získaných kombinovaným algoritmem a několika prvočísly získaných z internetu.

Miller-Rabinův test

Miller-Rabinův test[19] pochází z roku 1980, kdy matematik Michael O. Rabin navázal na objev Garyho L. Millera z roku 1976. Test není příliš složitý na implementaci a je tedy velmi rozšířený.

---

[15] https://en.wikipedia.org/wiki/Adleman%E2%80%93Pomerance%E2%80%93Rumely_primality_test
[16] https://sourceforge.net/projects/mpzaprcl/
[17] https://cs.wikipedia.org/wiki/GNU_Lesser_General_Public_License
[18] https://faculty.lynchburg.edu/~nicely/misc/bpsw.html
[19] https://cs.wikipedia.org/wiki/Miller%C5%AFv%E2%80%93Rabin%C5%AFv_test_prvo%C4%8D%C3%ADselnosti

V praktických algoritmech se test několikrát opakuje pro různé tzv. báze, čímž se snižuje pravděpodobnost chybného označení testovaného čísla za pravděpodobné prvočíslo. Každá báze generuje jinou sadu pseudoprvočísel. Pokud se test pouští pouze jednou a není řečeno jinak, předpokládá se báze dána číslem 2.

Zásadní část testu je tato:

- Nechť X je testované číslo a zvolená báze je 2.
- Nejprve číslo X zmenšíme o 1, čímž získáme číslo sudé. Z něj odstraníme v jeho rozkladu na součin případné mocniny 2. Samozřejmě velmi často v takovém rozkladu žádná dvojka být nemusí a výsledkem tohoto kroku tedy může být hodnota $X-1$. Označme takto získané číslo Y .
- V dalším kroku spočteme $2^Y \ mod \ X$.
- Je jasné že hodnota 2^Y může být velmi velké číslo, nestačí na něj 64 bitů, ani 128 bitů, na jeho reprezentaci v počítači bychom potřeboval Y bitů. Toto číslo nedává smysl přímo počítat, stačí zjistit zbytek po dělení testovaným číslem X. Existují různé algoritmy s různými strategiemi optimalizace. Založené na opakovaném mocnění a průběžné analýze výsledného zbytku.
- Konkrétně knihovna GNU MP obsahuje tento výpočet v metodě *mpz_powm* a ve verzi 6.2 obsahuje i specifickou optimalizaci pro bázi 2. Snaha o odstranění knihovny GNU MP a její nahrazení vlastním rychlým kódem založeným na 128-bitovém datovém typu se tedy jeví jako velmi složitá a nepříliš nadějná. Proto jsem od tohoto optimalizačního nápadu upustil.

Lucas-Selfridgeův test

Tento test je opět pojmenován po dvou matematicích. Prvním z nich je François Édouard Anatole Lucas (1842–91), který se zabýval Fibonacciho posloupností, zlatým řezem a tzv. Lucasovými posloupnostmi. Druhým matematikem byl John Lewis Selfridge (1927-2010). Tento test existuje ve slabé a silné variantě. Nezkoumal jsem jejich vzájemné rozdíly ani bližší podstatu těchto testů.

BPSW test

Vhodným propojením uvedených dvou testů získáme BPSW test. Konkrétní popis testu je tento:

- Na vstupu se předpokládá liché číslo. Prvočíslo 2 a sudá čísla musí být otestovány před vlastním algoritmem.
- Volitelně je vhodné provést test na dělitelnost několika malých prvočísel (uvádí se např. do 101). Tento krok zavádí určitou vágnost do definice testu. Krok je míněn pouze jako optimalizace. Zřejmě se nepředpokládá, že by mohlo existovat BPSW pseudoprvočíslo dělitelné nějakým malým prvočíslem, ale tuto otázku nejsem schopen zodpovědět.
- Provede se Miller-Rabinův test s bází 2. Pokud označí testované číslo jako složené, test končí.
- Dále je třeba nalézt první číslo D z posloupnosti $5, -7, 9, -11, 13, -15, 17, -19, \ldots$, pro které bude platit, že Jakobiho symbol[20] $\left(\frac{D}{X}\right) = -1$.

[20] https://cs.wikipedia.org/wiki/Jacobiho_symbol

    - Jakobiho symbol je důležitý pro teorii čísel. představuje zobecnění tzv. kvadratického zbytku[21]. Jeho názorné vysvětlení lze nalézt v mnoha videích na YouTube. Do jeho hlubšího pochopení jsem se nepouštěl.
    - Každopádně jeho výpočet není složité pochopit. V knihovně GNU MP existuje implementace výpočtu Jakobiho symbolu. V případě přechodu na výpočet pomocí 128-bitového datového typu by mohlo dojít ke značné optimalizaci. Opět se ale jedná o optimalizaci, kterou jsem se po zvážení a studiu náročnosti rozhodl nerealizovat.
    - Výpočet koeficientu D může být teoreticky velmi náročný. V úplně nejhorším případě se zkoušení členů popsané posloupnosti zastaví až u odmocniny z testovaného čísla. V praxi se však ukazuje, že se koeficient Ddaří nacházet velmi rychle.
- V dalším kroku se nastaví dvojice čísel $P = 1$ a $Q = (1 - D)/4$. Tyto dvě hodnoty se použijí jako vstup pro Lucasův test. Právě kombinace splnění těchto vstupních podmínek a Lucasova testu se nazývá Lucas-Selfridgeův test.

V GNU MP knihovně je naprogramován BPSW test, jehož algoritmus se odlišuje od výše popsaného postupu. Je prováděno více Miller-Rabinových testů a vůbec nedochází k hledání koeficientu D. Tento test ale není součástí oficiální dokumentace. Přesto však ukazuje, že definice BPSW testu nemusí být vnímána zcela jednotně.

Existuje další varianta, kdy je místo standardního Lucas-Selfridgeova testu použitá jeho silnější varianta (tzv. silný Lucas-Selfridgeův test). Touto silnou variantou jsem se ale nezabýval.

Algoritmus

Vzhledem ke katastrofickým výsledkům měření doby běhu jednotlivých testů jsem ani nevytvářel algoritmus pro ověřování BPSW pravděpodobných prvočísel. Jádro algoritmu by však vypadalo přibližně takto:

- Algoritmus zahájí výpočet na hranici 2^{64} a postupně zvyšuje testované číslo. Pokud BPSW test vrátí, že se jedná o pravděpodobné prvočíslo, dojde k ověření deterministickým testem APR-CL.
- Testovaný interval by algoritmus procházel pomocí Eratosthenova síta, kdy by se automaticky přeskakovaly násobky následujících prvočísel: $2, 3, 5, 7, 11, 13, 17, 19, 23$. Toto vychází z předpokladu, že v praxi by každá implementace BPSW testu měla na začátku obsahovat jako optimalizaci test dělitelnosti těmito prvočísly. Tímto by se snížil počet testovaných čísel z původních 2^{64} na 16,36% z této hodnoty.
- Algoritmus by nutně musel být připraven na opakované spouštění a zastavování svého běhu. Velikost bloku, který se v případě nečekaného ukončení musel počítat znovu by byla dána primoriálem $23\# = 223\,092\,870$.
- Bude nutné počítat s postupným zpomalováním programu, kdy lze předpokládat, že výpočet pro větší čísla bude většinou pomalejší. Program si bude sám měřit, ukládat dobu běhu a optimálně i odhadovat dobu pro dokončení.

[21] https://cs.wikipedia.org/wiki/Kvadratick%C3%BD_zbytek

- Všechny metody jsou připraveny na paralelní spuštění na více jádrech procesoru. Paralelní programování představuje poměrně komplikovanou oblast programování, takže jsem tuto problematiku ani nezačal studovat.

Většina stavebních kamenů je připravena pro složení do výsledného algoritmu. Nedává však smysl ladit algoritmus, který by nikdy nedoběhl.

Měření koeficientu D

Vedlejším cílem práce je i analýza koeficientu D, který je třeba vypočítat pro Lucas-Selfridgeův test. Výše tohoto koeficientu je teoreticky omezena pouze odmocninou z testovaného čísla. Pro praktické užití jsou ale následující otázky velmi praktické:

- Kolik koeficientů D je obvykle potřeba vyzkoušet, než je nalezen vyhovující koeficient D?
- Nenastávají výjimečně případy, kdy se koeficient D hledá extrémně dlouhou dobu?

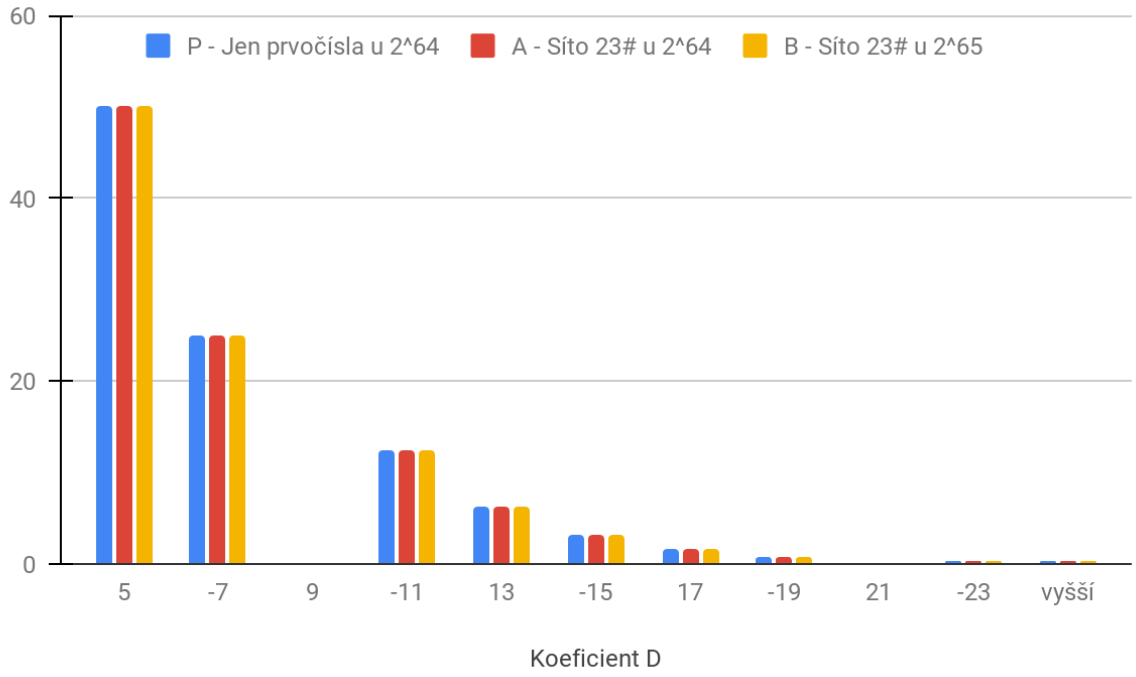
Provedl jsem tři měření procentuální četnosti konkrétních koeficientů D na třech různých vzorcích čísel:

Vzorek	**Začátek**	**Konec**	**Délka**	**Popis**
A	$2^{64}+1$	$2^{64}+1+2^{31}$	$2^{31}=.$ $2\,147\,483\,648$	Eratosthenovo síto spuštěné od hranice 2^{64} za využití primoriálu $23\#$. Šlo tedy o směs prvočísel a běžných čísel, ze kterých jsou odstraněny násobky malých prvočísel.
B	$2^{65}-1-2^{31}$	$2^{65}-1$	$2^{31}=.$ $2\,147\,483\,648$	Shodně jako u A.

U vzorků A a B jde o směs prvočísel a běžných čísel, ze kterých jsou odstraněny násobky malých prvočísel do 23.

Vzorek	**Začátek**	**Délka**	**Popis**
P	První prvočíslo po 2^{64}	$2^{26}=67\,108\,864$	Vzorek složený jen z prvočísel.

Pomocí vytvořeného programu jsem napočítal histogramy koeficientů D. Všechny tři histogramy pro vzorky A, B a P jsou zobrazeny na následujícím grafu. Osa x zobrazuje koeficient D, na ose y jsou procenta, jejichž základem je počet čísel v daném vzorku. Údaj pak vyjadřuje relativní četnost konkrétního koeficientu ve vzorku.



Graf 2

Měření ukázalo, že histogramy pro všechny tři vzorky jsou velmi podobné, na grafech histogramů nejsou odchylky okem pozorovatelné. Ukazuje se u všech vzorků až na velmi malou odchylku platí:

- První koeficient 5 vyhovuje polovině všech testovaných čísel,
- Druhý koeficient -7 vyhovuje asi čtvrtině všech testovaných čísel,
- Čtvrtý koeficient -11 vyhovuje asi osmině testovaných čísel
- Pátý koeficient 13 vyhovuje zhruba šestnáctině testovaných čísel

Třetí koeficient 9 a stejně tak i koeficient 21 nevyhovují žádnému číslu. Tento poznatek nebyl nikde v literatuře zmíněn. Není zřejmé, zda by toto pozorování platilo obecně pro všechna čísla. V takovém případě by jejich vynechání při výpočtu koeficientu D představovalo vhodnou optimalizaci.

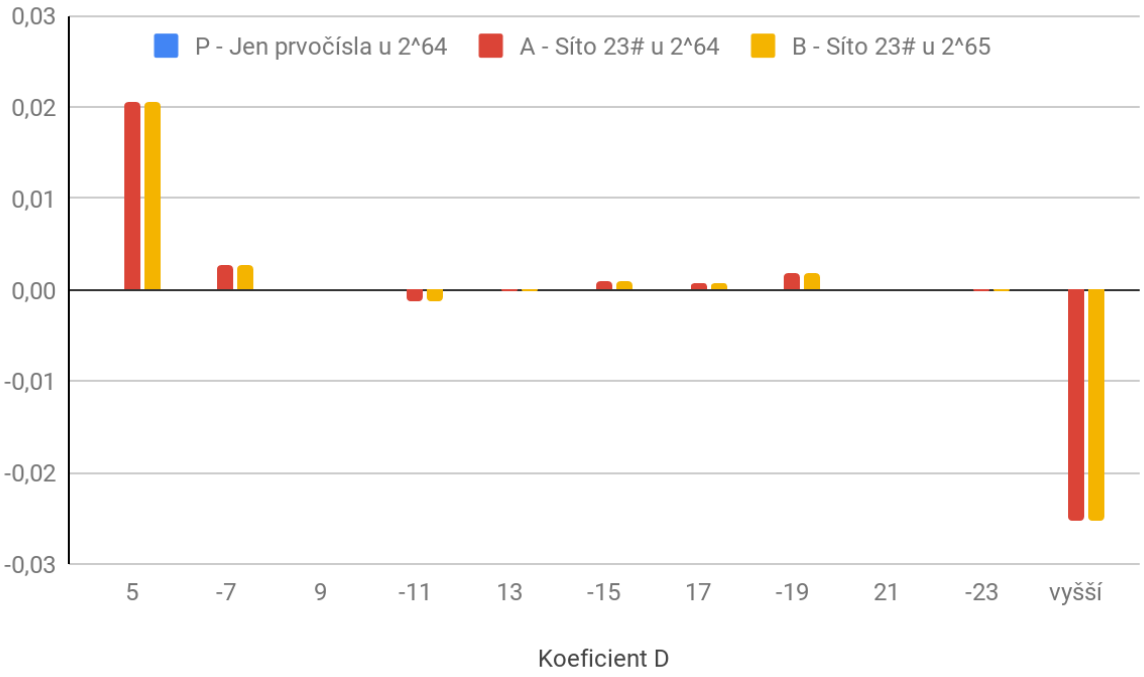
Zdrojová data pro tvorbu výše uvedeného grafu jsou vidět v následující tabulce:

Koeficient D	P	A	B
5	50,003403	50,023956	50,024017
-7	24,997267	25,000003	24,999999
9	0,000000	0,000000	0,000000
-11	12,501185	12,500003	12,500002
13	6,250198	6,250000	6,250002
-15	3,124112	3,124997	3,125000
17	1,561783	1,562497	1,562502
-19	0,779399	0,781249	0,781252
21	0,000000	0,000000	0,000000
-23	0,390801	0,390625	0,390617
vyšší	0,391851	0,366670	0,366609

Hodnoty koeficientu D , kde bylo potřeba nejvyšší počet kroků a vlastní testované číslo zobrazuje následující tabulka:

Vzorek	D	Testované číslo
A	-107	18446744075355579421
B	113	36893488147322959681
P	137	18446744076220818901

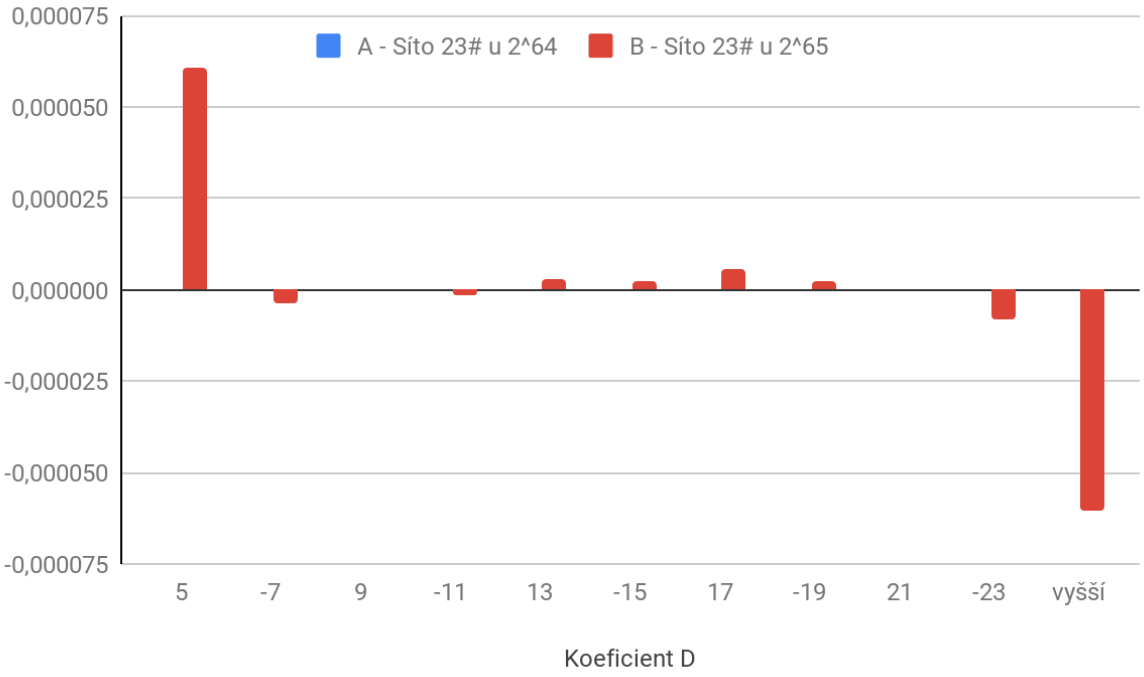
I přes velmi podobné chování koeficientu D v podkladových datech existují drobné odchylky. Ty je vhodné zkoumat v podobě rozdílu. Následující graf zobrazuje odchylky histogramů pro vzorky A a B od histogramu pro prvočísla, který sám leží na ose x:



Graf 3

Z grafu je vidět, že prvočísla jsou specifická tím, že koeficient 5 jim ve velmi malé míře vyhovuje o něco méně a tento neúspěch se daří napravit nalezením vhodného koeficientu až po více jak deseti krocích. Pro pochopení tohoto grafu je zásadní vzít v úvahu, že měřítko osy *y* pokrývá pouze rozsah odchylky do $0{,}3\%$.

Posledním grafem je porovnání histogramů A a B, který tak popisuje případné rozdíly na začátku zkoumaného intervalu a na jeho konci. Pro jeho řádnou interpretaci je opět nutné vzít v úvahu, že měřítko osy y je nyní velmi jemné a pokrývá odchylky pouze do $0{,}00005\%$.



Graf 4

Opět lze usuzoval, že větší čísla jsou specifická tím, že koeficient 5 jim v opravdu velmi malé míře vyhovuje o něco méně a tento neúspěch se daří napravit nalezením vhodného koeficientu až po více jak deseti krocích.

Každopádně pro praktické výpočty jsou rozdíly zanedbatelné, neznámý počet kroků nutných pro výpočet koeficientu D nepředstavuje výraznou překážku u naprosté většiny čísel. Konkrétně vyšší počet kroků než deset je potřeba jen u méně než $0{,}4\%$ čísel (na všech vzorcích). Detailní rozpad těchto čísel je v následující tabulce, která seskupuje náročnost výpočtu koeficientu D po desítkách:

Počet kroků pro výpočet D	P	A	B
do 10	99,60814863	99,63332989	99,63339052
do 20	0,37956536	0,35650453	0,35649143
do 30	0,01148582	0,00955358	0,00953365
do 40	0,00077635	0,00059806	0,00057016
do 50	0,00002086	0,00001338	0,00001309
do 60	0,00000149	0,00000057	0,00000114
do 70	0,00000149		
do 80			
do 90			
do 100			
vyšší			

Prázdné buňky mají význam hodnoty 0.

Měření výkonnosti jednotlivých metod

Měření výkonnosti jednotlivých metod je kritické pro odhad doby běhu algoritmu, který by ověřoval spolehlivost BPSW testu. Doba běhu testu může být odlišná na začátku intervalu než na jeho konci. Stejně tak doba běhu jednotlivých algoritmů může být zcela jiná, pokud jsou na vstupu pouze prvočísla, což byl případ všech popsaných triviálních algoritmů. S tímto cílem byly zkonstruovány tři vzorky A, B a P popsané v předešlé kapitole. Postupně jsem prováděl měření následujících metod:

Označení	**Popis**
MR	Miller-Rabinův test s bází 2
LS	Lucas-Selfridgeův test
LS-D	Lucas-Selfridgeův test, který končí předčasně výpočtem koeficientu D
BPSW	BPSW test, ve kterém do první fáze MR testu vstoupí všechna čísla a do druhé LS části vstoupí jen čísla, která MR test neoznačil za složená.
BPSW2	Jiná implementace BPSW testu z knihovny APRCL.
APRCL	Deterministický APRCL test

Výsledná měření pro vzorky A a B shrnuje následující tabulka:

Označení	Minuty u 2^{64}	Minuty u 2^{65}	Zpoždění	Roky
MR	7,20	8,07	112,1%	124 681
LS	88,41	162,98	184,3%	2 052 952
LS-D	2,15	2,21	102,6%	35 624
BPSW	19,47	30,32	155,7%	**406 645**
BPSW2	22,76	28,17	123,8%	415 929
APRCL	499,00	632,04	126,7%	9 236 262

Z tabulky je zřejmé, že BPSW test správně začíná Miller-Rabinovým testem. Případné prohození pořadí s Lucas-Selfridgeovým testem by bylo nevýhodné. Podobně nedává smysl začínat deterministickým APRCL testem a do BPSW testu pouštět jen složená čísla.

Z časových hodnot je očividné, že výpočet Lucas-Selfridgeova testu, končící výpočtem koeficientu D se s vyššími čísly nijak výrazně nezpomaluje. Nárůst ke konci intervalu byl jen $2{,}6\%$.

Zato u všech ostatních testů jsou nárůsty časů pro větší čísla již významná. Naměřené časy je nutné pro odhad případného použití na celý interval vynásobit. Vzorky A, B mají délku 2^{31}takže je nutné časy v minutách násobit hodnotou 2^{33}.

Pro odhad doby běhu algoritmů jsem zvolil průměr mezi dobou běhu na vzorku A a na vzorku B. Postupné zpožďování metod nebude lineární, takže měření uprostřed intervalu pravděpodobně nebude odpovídat tomuto průměru. I přes popsaný nedostatek ve způsobu odhadu je ale tabulka dostatečně vypovídající.

Výsledek převedený z minut na roky je uveden v posledním sloupci tabulky. **Je zřejmé, že pouze první fáze programu (generování BPSW prvočísel) by trvala více než 400 tisíc let.**

Výsledná měření pro vzorky A a B shrnuje následující tabulka:

	Minuty	Roky
MR	1,46	17 065
LS	16,65	194 502
LS_D	0,41	4 807
BPSW	18,13	211 804
BPSW2	22,48	262 555
APRLCL	1037,95	12 122 995

Prvočíselný vzorek P má délku 2^{26}. Dále je nutné zjistit počet všech prvočíslem mezi 2^{64} a 2^{65}. Ten jsem odhadl pomocí funkce Li, která bude více popsána ve druhé části práce. Zde je důležitý výsledek, že přibližný počet prvočísel odhadnutý touto funkcí je $412\,246\,861\,507\,631\,552$. Naměřené časy je nutné vynásobit podílem velikosti vzorku P a odhadnutého počtu prvočísel. Výsledek je opět v tabulce v pravém sloupci.

Tento zjednodušený přístup vůbec nezohledňuje zpomalení algoritmů pro rostoucí čísla.

Druhá fáze programu (ověřování prvočísel deterministickým testem) by trvala více než 12 milionů let.

Zhodnocení výsledků měření

Vzhledem k obrovským požadovaným časům všech variant výpočtů se ukazuje, že zdvojnásobení dosud platné hranice není v domácích podmínkách realistické. Případné další optimalizace uvedené ve volitelných cílech záměru práce by určitě přinesly další zrychlení. Velikost tohoto zrychlení by byla různá:

- paralelní výpočet: podle počtu jader jednoho procesoru například 8x
- Azure: podle počtu jader a počtu procesorů například 100x
- distribuovaný výpočet: podle počtu uživatelů například 10 000x

V každém případě by ale cíl zdvojnásobení hranice až na hodnotu 2^{65} byl stále nedosažitelný. Realizace původního cíle vyžaduje zcela jiný přístup, než opakované spouštění volně dostupných algoritmů. Takový přístup by musel kombinovat hlubší znalosti:

- podstaty prvočíselných testů,
- konkrétního algoritmu a jejich implementace,
- vnitřního chodu knihovny GNU MP.

Pravděpodobně by rovněž kombinoval rychlé operace dostupných celočíselných typů o 64 a 128 bitech s objekty reprezentující větší čísla. Především by pak neprocházel testovaný rozsah čísel lineárně, ale v jiném pořadí, které by umožňovalo sdílet některé části výpočtu pro mnoho čísel najednou. Všechny tyto popsané náměty jsou však velmi náročné.

Na základě tohoto poznání jsem se rozhodl již neinvestovat čas a energii na realizaci těchto cílů a přesunul svoji pozornost na zkoumání prvočísel na intervalech délky 2, které jsou předmětem další kapitoly.

Rozložení prvočísel na intervalech mocnin 2

Dirichletův teorém

Starořecký matematik Euklides dokázal ve své deváté matematické knize, že prvočísel existuje nekonečně mnoho. Tímto tématem se v devatenáctém století zabýval i německý matematik P. G. Lejeune Dirichlet (1805-1859), který mimo jiné zkoumal rozmístění prvočísel na aritmetických řadách. Jeho objev je názorně vysvětlen v tomto videu na YouTube: https://www.youtube.com/watch?v=EK32jo7i5LQ

Aritmetická řada je nekonečná posloupnost čísel, u nichž je rozdíl dvou následujících prvků (tzv. diference) vždy konstantní. Následující obrázek ukazuje rozdělení přirozených čísel na aritmetické řady definované dle zbytku po dělení 10 (zelená barva) a 4 (okrová barva).

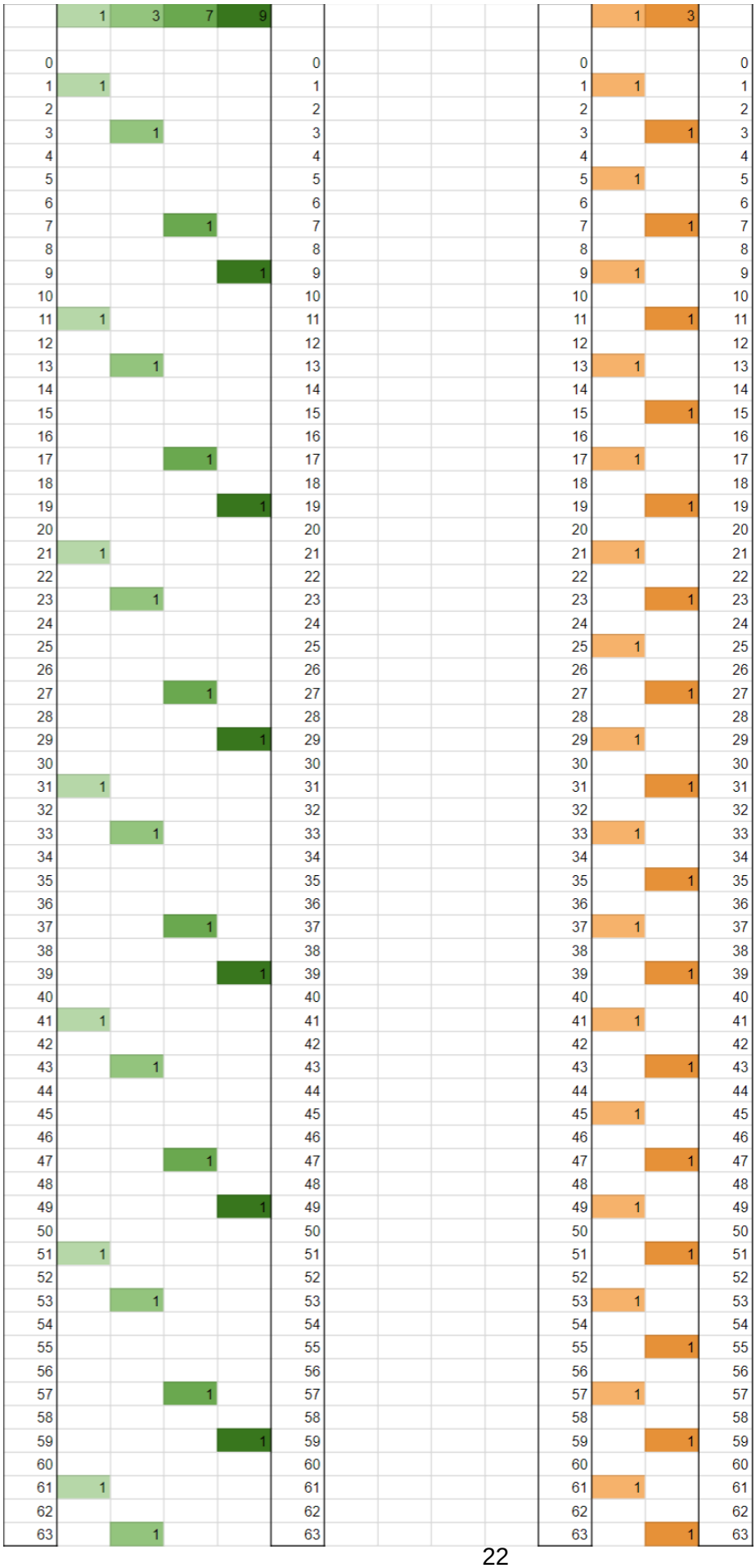
Pravý příklad ukazuje rozdělení všech přirozených čísel dle zbytku po dělení 4 na čtyři množiny:

- $4k$
- $4k + 1$
- $4k + 2$
- $4k + 3$

Například číslo 11 má po dělení 4 zbytek 3 a patří tak do aritmetické posloupnosti $4k + 3$.

Je zřejmé, že v aritmetických řadách $4k$ a $4k + 2$ se žádná prvočísla (s výjimkou prvočísla 2) vyskytnout nemohou, neboť všechny prvky jsou dělitelné 2. Všechna prvočísla se nacházejí v aritmetických posloupnostech $4k + 1$ (často jsou označované za elfy) a $4k + 3$ (často označované za skřety - anglicky ‘orcs’). Číslo 2 jediné nezapadá ani mezi elfy, ani mezi skřety a poeticky se nazývá ‘Tom Bombadil’[22].

[22] https://www.quora.com/Is-there-a-relation-between-prime-numbers-and-the-sum-of-squares-If-a-number-can-be-represented-as-the-sum-of-squares-does-it-comes-closer-to-being-a-prime



Obrázek 1

Dirichlet dokázal, že na každé aritmetické posloupnosti $nk + q$, kde n (skok, dělitel) a q(počátek, zbytek po dělení) jsou nesoudělná a k představuje všechna přirozená čísla, leží nekonečně mnoho prvočísel. Zobecnil tak zmiňovaný Euklidův důkaz. Konkrétně lze Dirichletův výsledek pro $n = 4$ zapsat pomocí následujících rovnic:

$$\lim_{i\to\infty} \frac{\pi(i;\ 4;\ 1)}{\pi(i)} = \frac{1}{2}$$

a zároveň platí:

$$\lim_{i\to\infty} \frac{\pi(i;\ 4;\ 3)}{\pi(i)} = \frac{1}{2}$$

kde:

- $\pi(i)$ je obvyklé označení pro funkci, která vrací počet prvočísel menších nebo rovných než číslo i.
- $\pi(i; 4; 3)$ je obvyklé označení pro funkci, která vrací počet prvočísel menších nebo rovných i, jejichž zbytek po dělení 4 je 3.

Pojem limity si zaslouží přesnou definici i její stručný výklad. Říkáme, že limita posloupnosti x_j je číslo C (tj. zápis $\lim_{j\to\infty} x_j = C$) právě tehdy a jen tehdy, pokud pro každé $\varepsilon > 0$ existuje index N takový, že pro všechny vyšší indexy $j \geq N$ platí že $|x_j - C| < \varepsilon$.

Vztáhneme-li definici limity na uvedený případ, konkrétně vyjadřuje skutečnost, že čím více číslo i roste,

- tím více se podíl blíží jedné polovině
- tj. pro dostatečně velké hodnoty i se uvedený podíl přiblíží jedné polovině s libovolnou požadovanou přesností.

Uvedený příklad platí i pro libovolné n. Na místo aritmetických posloupností definovaných dle zbytku po dělitelnosti 4, můžeme definovat řady dle zbytku po dělení zcela libovolným číslem n. Výsledek limity potom nebude $\frac{1}{2}$, ale obrácená hodnota z počtu nesoudělných čísel s číslem n. Například bude-li $n = 10$, počet nesoudělných čísel s 10 je 4 (konkrétně $1,3,7,9$). Limita pro každou aritmetickou posloupnost bude $\frac{1}{4}$.

Dirichletův teorém[23] přitom platí pro každou aritmetickou posloupnost, kde její skok n a počátek q jsou nesoudělná čísla (a takovýchto kombinací je nekonečně mnoho!). Dirichletův teorém představuje tedy podstatné zobecnění původního Euklidova objevu.

Nekonečný počet prvočísel se tedy vyskytuje například v každé uvedené aritmetické řadě:

- $4k + 3, 4k + 1$
- $10k + 1, 10k + 3, 10k + 7, 10k + 9$

[23] https://en.wikipedia.org/wiki/Dirichlet%27s_theorem_on_arithmetic_progressions

- $1000000k + q$ pro všechna q nesoudělná s 1000000

Každá aritmetická řada přitom představuje zcela specifický řez množiny přirozených čísel a takovýchto řezů všech přirozených čísel existuje nekonečně mnoho.

O nespoutanosti prvočísel a anomáliích

Teorém vždy popisuje rozřazení všech prvočísel do několika aritmetických řad, přičemž vždy vypovídá o rozřazení do řady jako celku, tj. do nekonečné řady neomezené žádnou konečnou hranicí. Teorém nevypovídá přímo o rozložení konečného počtu prvočísel menších než nějaké konkrétní přirozené číslo n.

Lze si tedy představit možnost, že zhruba rovnovážného zastoupení počtu prvočísel může být v některých řadách dosaženo až po překonání obrovských (a prakticky výpočetně nedosažitelných) hodnot, s tím, že rozložení prvočísel pro menší hodnoty nemusí být vůbec rovnoměrné.

Praktické měření [24]ale ukazuje, že počty prvočísel pro libovolně zvolenou sadu aritmetických řad jsou vyrovnané velmi brzy. Samozřejmě, že každé praktické měření nemá hodnotu důkazu a nevypovídá o chování pro nekonečně mnoho čísel. Naštěstí se ale jedná o intenzivně zkoumanou část matematiky. V této souvislosti je vhodné uvést zejména následující dosažené úspěchy.

Prvním z nich je snaha aproximovat funkci $\pi(x)$ nějakou jinou funkcí se snadnou definicí, a tím pádem i jasného a predikovaného chování, kterou lze na rozdíl od funkce $\pi(x)$ snadno zkoumat. Zde matematici objevili funkci inverzního logaritmického integrálu $li(x)$, kterou budeme definovat v kapitole *Funkce* $li(x), Li(x)$ *a* $x/ln(x)$. Nyní je důležitá následující aproximace[25]:

$$|\pi(x) - li(x)| < \frac{\sqrt{x} \cdot log(x)}{8\pi}$$

Současně je nutné sdělit, že platnost uvedené nerovnosti je podmíněna platností slavné Riemannovy hypotézy[26]. Nicméně platí, že dosažené odhady, které nejsou závislé na platnosti Riemannovy hypotézy, poskytují o něco horší odhady.

Druhým dosaženým úspěchem je tzv. Siegel–Walfisz teorém[27], který představuje zobecnění Prvočíselné věty a Dirichletova teorému. Ve svém důsledku stanovuje určitou mez na odchylku počtu prvočísel na aritmetické posloupnosti, která v tomto případě není nekonečná, ale omezená nějakou konečnou hranicí.

Již uvedené věty zde mají ilustrovat následující představu. Krása prvočísel spočívá ve skutečnosti, že mají velmi triviální definici, ale přesto je nelze nějak snadno spoutat. Dodnes v sobě skrývají řadu tajemství, které současní matematici intenzivně zkoumají.

---

[24] viz např. již zmíněné Youtube video popisující Dirichletův teorém
[25] https://en.wikipedia.org/wiki/Prime_number_theorem#Prime-counting_function_in_terms_of_the_logarithmic_integral
[26] https://cs.wikipedia.org/wiki/Riemannova_hypot%C3%A9za
[27] https://en.wikipedia.org/wiki/Siegel%E2%80%93Walfisz_theorem

Ano, prvočísla jsou nespoutaná, ale zároveň se chovají poměrně ukázněně, což bylo například ilustrováno výše uvedenými úspěchy, které v podstatě stanovují chování prvočísel určitou mez.

Na druhou stranu je nutné zmínit, že menší anomálie samozřejmě existují. Nejvýraznější z nich přímo se vztahující ke zkoumanému tématu je tzv. Chebyshev's bias[28] popisující pozoruhodně vytrvalou převahu výše zmíněných skřetů nad elfy, která trvá pro prvočísla až do zhruba 23 miliard.

Zobecňovat platnost nějakého pozorování učiněného pro určitou konečnou množinu prvočísel může být ošidné. Hezkým příkladem je tzv. Pólyova hypotéza[29] z roku 1919, která na základě tehdy dostupných pozorování předpokládala, že většinu (tj. více než 50%) přirozených čísel lze rozložit na lichý počet prvočinitelů. Hypotéza byla vyvrácena až v roce 1958 sestrojením protipříkladu v okolí čísla 1.845×10^{361}.

Jiný příklad anomálie týkající se mezer mezi prvočísly je uvedený v podkapitole *Interpretace pozorování*.

Každopádně rovnoměrnost rozložení prvočísel a jeho případných anomálií představuje fascinující předmět zkoumání matematiky. V dalších kapitolách budou zkoumány prvočísla na konečných intervalech definovaných bitovými řezy. Intuitivně lze předpokládat, že i zde se prvočísla budou chovat ukázněně a velikosti odchylek oproti teoretickému očekávání bude rovněž mít nějakou svoji hranici.

[28] https://mathworld.wolfram.com/ChebyshevBias.html
[29] https://en.wikipedia.org/wiki/P%C3%B3lya_conjecture

Bitové řezy

Pokud připustíme, že prvočísla jsou poměrně rovnoměrně rozmístěna napříč různými aritmetickými posloupnostmi, nabízí se předpoklad, že rovnoměrnost rozmístění prvočísel bude možné pozorovat i při jiném způsobu rozdělení číselné osy než na aritmetické posloupnosti.

Pro počítače je přirozené pracovat s dvojkovou soustavou. Každé přirozené číslo lze rozdělit jako součet mocnin 2. Jednotlivé bity pak označují, která mocnina dvou je v rozkladu zastoupena:

- $37 = 32 + 4 + 1 = 2^5 + 2^2 + 2^0 = 10\,0101_2$
- Nejnižší bit s indexem 0 je umístěn nejvíce vpravo a do součtu přispívá nultou mocninou dvojky, tedy jedničkou. Představuje řád jednotek a vyjadřuje, zda je číslo liché (bit nastaven) nebo sudé (bit nenastaven).
- Nejvyšší bit z příkladu, bit číslo 5, přispívá do součtu hodnotou 32.

Nejprve je potřeba rozdělit číselnou osu na intervaly, které se budou dále dělit na řezy. Dirichlet dělil číselnou osu na aritmetické řady, my rozdělíme číselnou osu tak, jak je přirozené pro počítač, to znamená za pomocí dvojkové soustavy.

Interval

Zkoumaný interval I_N je vždy konečný a určený nějakou mocninou 2 jako

$$I_N = < 2^N;\ 2^{N+1} - 1 >\ pro\ N \geq 1$$

Vlastnosti:

- Například interval I_7pro mocninu 7 začíná na hodnotě $2^7 = 128$ a končí na hodnotě $2^8 - 1 = 255$.
- V binárním zápisu tedy začátek intervalu lze zapsat pomocí 8 bitů jako $1000\,0000_2$ a konec jako$1111\,1111_2$.
- Všechna čísla v I_7mají nastaven nejvyšší bit číslo 7, ostatní bity číslo 0 až 6 probíhají postupně všemi kombinacemi, kterých je celkem 128 (neboli 2^7).
- Každý následující interval je dvakrát delší než předchozí interval.
- $I_1 = < 2;\ 3 >, I_2 = < 4;\ 7 >, I_3 = < 8;\ 15 >, I_4 = < 16;\ 31 >$
- Interval I_0není intervalem, obsahuje pouze jedno číslo $I_0 = < 1;\ 1 >$ a není ho možné dále řezat.

Řezy

Zkoumaný interval I_N lze “rozřezat” vícero způsoby povinným nastavením dalšího bitu j nižšího než N na 1. V dalším textu budeme krátkým označením ‘nastavený bit’ vždy rozumět nastavení bitu na hodnotu 1.

$$I_{N,j} = I_N \cap \{všechna\ čísla\ mající\ nastavený\ bit\ j\}, 0 \leq j < N$$

Vlastnosti:

- Každý řez má nastavené povinně vždy dva bity: bit nejvyšší a jeden z nižších bitů. Ostatní bity postupně prochází všemi možnými kombinacemi.
- Nastavením bitu 0 dojde k vybrání všech lichých čísel ve zkoumaném intervalu. Jen tento řez se skládá z pouze lichých čísel. Ostatní řezy obsahují vždy polovinu čísel sudých a polovinu čísel lichých.
- Nastavením bitu 1 dojde k rozřezání intervalu I^7 na 32 čtveřic a do výběru je vždy zařazena vyšší polovina každé čtveřice.
- Nastavením bitu 2 dojde k rozřezání intervalu na osmice a do výběru je vždy zařazena horní čtveřice z každé osmice.
- Nastavením bitu 6 u intervalu I_7dojde k rozdělení celého intervalu na dvě poloviny, přičemž do výběru je zařazena jen horní polovina.
- Pro I_7 dostaneme celkem 7 různých řezů, které získáme postupným nastavením bitů číslo 0 až 6.
- $I_{1,0} = \{3\}$
- $I_{2,0} = \{5,7\}$, $I_{2,1} = \{6,7\}$

  Jak můžeme vidět ve dvojkové soustavě na obrázku:

  - interval I_2 má nastavený bit 2, čemuž odpovídají 4 čísla 4, 5, 6, 7.
  - interval $I_{2,0}$má nastavené bity 2 a 0, čemuž odpovídají čísla 5 a 7.
  - interval $I_{2,1}$má obdobně nastavené bity 2 a 1. Z obrázku je zřejmé, že tomuto nastavení odpovídají čísla 6 a 7.

Číslo	Bit			
	2	1	0	
4	1			4
5	1		1	5
6	1	1		6
7	1	1	1	7

Obrázek 2

- $I_{3,0} = \{9,11,13,15\}$, $I_{3,1} = \{10,11,14,15\}$, $I_{3,2} = \{12,13,14,15\}$

	I3	I3,2	I3,1	I3,0	
8	1				8
9	1			1	9
10	1		1		10
11	1		1	1	11
12	1	1			12
13	1	1		1	13
14	1	1	1		14
15	1	1	1	1	15

Obrázek 3

- Každý řez má v součtu shodnou délku, a to přesně polovinu zkoumaného intervalu.
- Průnik dvou různých řezů by představoval jiný způsob řezů, kdy by byly vždy nastaveny právě 3 bity a v součtu by byla vybrána čtvrtina původního intervalu. Dva různé řezy tedy nejsou navzájem disjunktní.

Všechny řezy pro interval definovaný bitem číslo 5 jsou znázorněny na následujícím obrázku.

	I5	I5,4	I5,3	I5,2	I5,1	I5,0	
32	1						32
33	1					1	33
34	1				1		34
35	1				1	1	35
36	1			1			36
37	1			1		1	37
38	1			1	1		38
39	1			1	1	1	39
40	1		1				40
41	1		1			1	41
42	1		1		1		42
43	1		1		1	1	43
44	1		1	1			44
45	1		1	1		1	45
46	1		1	1	1		46
47	1		1	1	1	1	47
48	1	1					48
49	1	1				1	49
50	1	1			1		50
51	1	1			1	1	51
52	1	1		1			52
53	1	1		1		1	53
54	1	1		1	1		54
55	1	1		1	1	1	55
56	1	1	1				56
57	1	1	1			1	57
58	1	1	1		1		58
59	1	1	1		1	1	59
60	1	1	1	1			60
61	1	1	1	1		1	61
62	1	1	1	1	1		62
63	1	1	1	1	1	1	63

Obrázek 4

Pokud bychom připustili v uvedené definici $j = N$, došlo by k nastavení pouze jednoho bitu a nejednalo by se tedy o řez, ale o celý interval. Přirozeně lze tedy dále definovat:

$$A_{N,N} = A_N$$

Měření

Cílem měření je spočítat počet prvočísel pro jednotlivé řezy intervalů $I_0, \ldots, I_{36}$. Výpočet pro vyšší mocniny by byl neúměrně zdlouhavý. Označme

$$\pi_N = počet\ prvočísel\ intervalu\ I_N$$

$$\pi_{N,j} = počet\ prvočísel\ řezu\ I_{N,j}$$

Program postupně vypočítal hodnoty $\pi_{N,j}$ pro $N \in < 1, 36 >, j \in < 0, N >, N \in \mathbb{N}$. Výpočet pro I_{36} trval několik dní, pro vyšší mocniny by trval neúměrně dlouho. Pro ilustraci: $\pi_{36} = 2\,712\,103\,833$, což považuji za číslo dostatečně velké pro měření rovnoměrnosti rozmístění prvočísel na řezech intervalu I_{36} a zároveň jde o prakticky zvládnutelný výpočet na domácím počítači. Na druhou stranu počet prvočísel na krátkých intervalech je velmi nízký. V takové situaci lze jen těžko mluvit o rovnoměrném rozmístění prvočísel, neboť i velmi malý rozdíl v počtu prvočísel způsobí poměrně významnou odchylku. V praxi tedy vyhodnocuji jen mocniny dvou od 25 do 36, čímž získáme pozorování pro všechny řezy, tj. celkem pro celkem 12 intervalů. Pro ilustraci $\pi_{25} = 1\,894\,120$.

Všechna naměřená data jsou dostupné v této tabulce[30] (na prvním řádku klíčového listu *BitStat* obsahují názvy sloupců vysvětlující komentář, který se zobrazí, pokud na danou buňku namíříme myší).

Porovnání pomocí průměru

Pro zkoumání rovnoměrnosti rozmístění prvočísel pro jednotlivé intervaly a jejich řezy je potřeba počty prvovočísel navzájem porovnávat. Jednak mezi řezy jednoho intervalu, ale i mezi řezy různých intervalů. Nejjednodušším způsobem jak tohoto docílit je použít následující úvahu: *každý řez obsahuje sadu dílčích intervalů, jejichž součet odpovídá přesně délce poloviny celého intervalu.* Na toto tvrzení lze nahlížet například takto:

1. délka intervalu je mocnina dvou, lze jej tedy rozdělit na dvě stejně dlouhé části
2. nastavením libovolného bitu dojde k výběru určité množiny čísel
3. vybraná čísla mají zvolený bit nastavený na 1
4. nevybraná čísla mají zvolený bit naopak nastavený na 0
5. průnik obou výběrů je prázdný
6. situace je symetrická, oba výběry tedy musí obsahovat stejný počet prvků - tedy přesně polovinu čísel celého intervalu

Jako základ pro výpočet procent tedy stanovíme polovinu počtu prvočísel celého intervalu, tedy: $\frac{1}{2}\pi_N$. Definujme výraz:

30 https://docs.google.com/spreadsheets/d/1YZKVsGMMEgvizjicBX59jQYdXLOTNTPxQfWsvlOgZO4/edit?usp=sharing

$$A_{N,j} = \frac{\pi_{N,j} - \frac{1}{2}\pi_N}{\frac{1}{2}\pi_N} \; pro\ N \geq 2, pro\ j \in < 0, N >$$

Čitatel vyjadřuje odchylku skutečného počtu prvočísel v daném řezu od očekávaného stavu, celý výraz pak vyjadřuje poměr vůči základně. Případným vynásobením 100 se na výraz můžeme dívat jako na hodnotu v procentech. $A_{N,j}$ je relativní veličina, a můžeme ji tedy vzájemně porovnávat napříč různými řezy stejného intervalu ale i napříč různými intervaly.

Podívejme se nejdříve na speciální případy.
Co říká hodnota $A_{N,0}$?

- $A_{N,0}$ se vždy vztahuje k intervalu, jehož nultý bit je vždy nastaven.
- Řez obsahuje všechna lichá čísla v intervalu I_N.
- S výjimkou dvojky jsou všechna prvočísla lichá.
- Na námi měřeném vzorku čísel se prvočíslo 2 nevyskytuje ($< I_{25}; I_{36} >$).
- Hodnota $A_{N,0} = 100\%$ a obsahuje všechna prvočísla v intervalu.

Co říká hodnota $A_{N,N}$?

- Hodnota $A_{N,N}$ se vztahuje k intervalu, který má povinně nastavený pouze jeden bit s indexem N
- Nejedná se tedy o řez, ale o celý interval I_Na tedy $A_{N,N} = 100\%$.

Oba tyto speciální případy ze společného porovnávání vylučujeme.

Jaké by byly hodnoty $A_{1,0}$, $A_{1,1}$?

- I_1 obsahuje jen čísla $2,3$, takže i $\pi_1 = 2$.
- $I_{1,0}$ obsahuje jen číslo 3, $A_{1,0} = 0$, což je výjimka vytvořená jediným sudým prvočíslem 2, pro všechny vyšší intervaly již platí, že $A_{N,0} = 1$.
- $I_{1,1}$ obsahuje čísla $2,3$, $A_{1,0} = 1$.
- Tento speciální případ nás nezajímá, stačí tedy definovat hodnoty $A_{N,j}$ pro $N \geq 2$.

Jaké jsou hodnoty $A_{2,0}$, $A_{2,1}$?

- I_2 obsahuje jen čísla $4,5,6,7$, takže i $\pi_2 = 2$.
- $I_{2,0}$ obsahuje čísla $5,7$, $A_{2,0} = 1$ v souladu s výše uvedeným tvrzením pro $A_{N,N}$.
- $I_{2,1}$ obsahuje čísla $6,7$, $A_{2,0} = 0$, tady žádná odchylka.

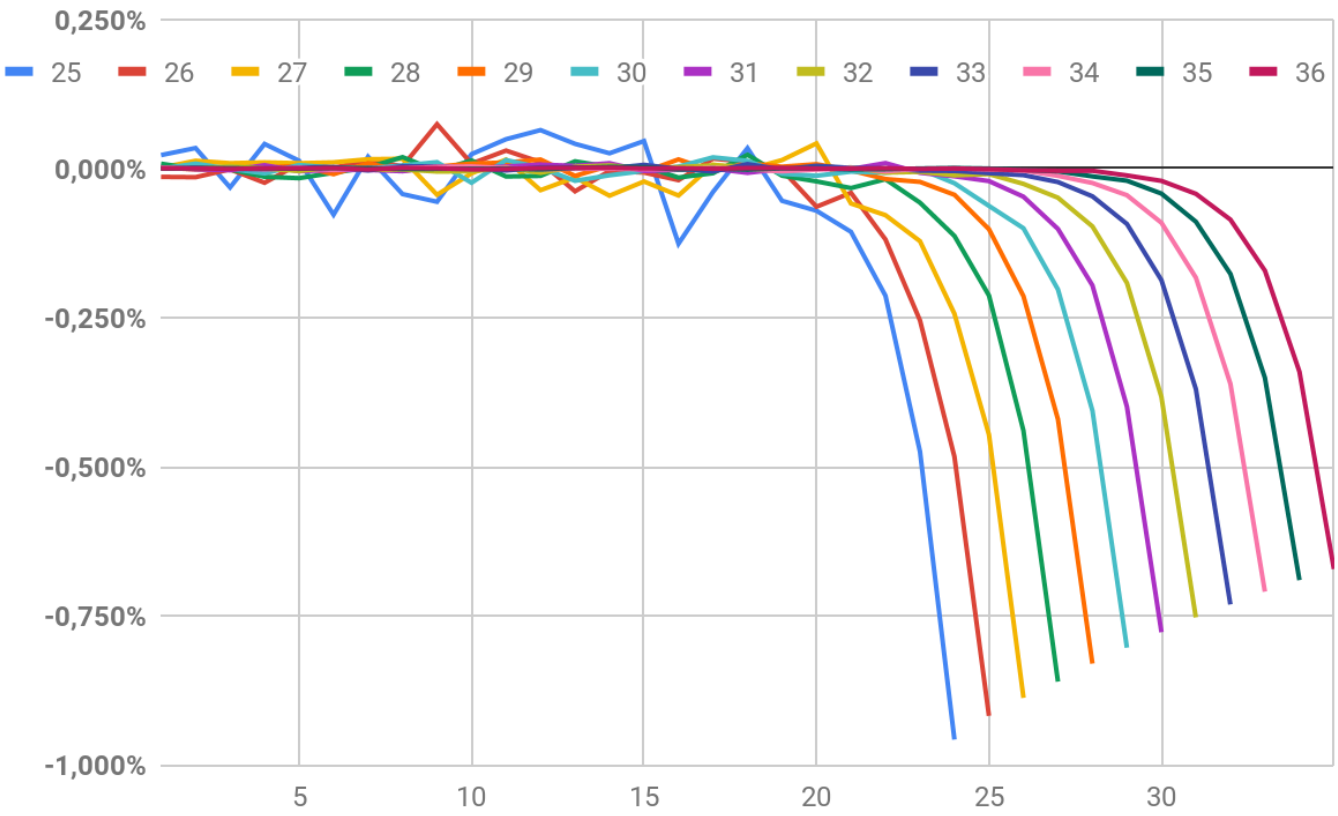
Jaké jsou hodnoty $A_{3,0}$, $A_{3,1}$, $A_{3,2}$?

- I_3 obsahuje jen čísla $8,9,10,11,12,13,14,15$, takže i $\pi_3 = 2$.
- $I_{3,0}$ obsahuje čísla $9, \mathbf{11}, \mathbf{13}, 15$, $A_{3,0} = 1$.

- $I_{3,1}$ obsahuje čísla $10,11,14,15$, $A_{3,1} = 0$.
- $I_{3,2}$ obsahuje čísla $12,13,14,15$, $A_{3,2} = 0$.

Hodnoty $A_{N,j}$ pro $N \leq 3$vždy nabývají jen dvou krajních hodnot 0 (žádná odchylka oproti očekávání) nebo 1 (extrémní odchylka oproti očekávání).

Teprve u vyšších intervalů začínají hodnoty $A_{N,j}$ nabývat i jiných (a tedy zajímavějších) hodnot. Porovnání vybraných hodnot $A_{N,j}$ je na následujícím grafu. Index j definující řez je na ose x, na ose y je procentuální hodnota $A_{N,j}$vztažená k řezu j. Jedna barva pak určuje všechny řezy určitého intervalu. Na grafu jsou zobrazeny výsledky $A_{N,j}$ pro intervaly I_{25} až I_{36}.



Graf 5

Z grafu je vidět, že pro řezy od 1 do přibližně 20 (v různých intervalech), je velikost odchylky menší než 0.1% a výše zmíněná úvaha o očekávaném počtu prvočísel funguje velmi dobře. Na druhou stranu pro každý interval je několik jeho posledních řezů zatíženo výrazně vyšší chybou. Velikost odchylky je až desetkrát větší. Přestože se procentuální velikost odchylky pro vyšší mocniny postupně zmenšuje, je z uvedeného grafu velmi problematické usuzovat na rovnoměrnost rozmístění prvočísel a znázorněný jev je potřeba nejdříve vysvětlit.

Interpretace pozorování

Předpokládejme nejprve, že graf nepopisuje nějakou anomálii v rozmístění prvočísel, ale že vysvětlení pozorovaného jevu spočívá v chybném nastavení základny $\frac{1}{2}\pi_N$, které se nejvýznaměji projevuje jen pro řezy s vyšším indexem.

Pro další úvahu je potřeba zkombinovat dvě úvahy.

1. Řezy $I_{N,j}$s nízkým indexem $j$se skládají z mnoha malých částí, které rovnoměrně vyplňují celý interval. Naproti tomu řez $I_{N,N-1}$ vybírá ze zkoumaného intervalu jen jeho horní polovinu. Řez $I_{N,N-2}$ se skládá ze dvou úseků: horní polovina z první poloviny intervalu a horní polovina z druhé poloviny intervalu. Podobně i ostatní vyšší indexy dělí zkoumaný interval na několik málo částí, ze kterých je vždy vybírána jejich horní polovina. V definovaných řezech vždy převažují horní poloviny a tento efekt je nejvýraznější právě u řezů s vyšším indexem.

2. Je prokázáno, že hustota výskytu prvočísel pro větší čísla se postupně snižuje. Konkrétně Prvočíselná věta[31] říká, že při náhodném výběru čísla blízko nějakého velkého čísla N pravděpodobnost, že toto číslo bude prvočíslem, je přibližně $1/ln(N)$. Použitím věty například na I_{36} získáme:
   - Začátek intervalu: $ln(2^{36}) = ln(68\,719\,476\,736) = 24.953$
   - Konec intervalu: $ln(2^{37}) = ln(137\,438\,953\,472) = 25.646$

   Tedy na začátku intervalu se vyskytuje prvočíslo v průměru každých 24.953, zatímco na jeho konci v průměru již jen každých 25.646 čísel.

   Zde je klíčové porozumět skutečnosti, že řada individuálních případů se od tohoto průměrného chování značně liší. Vzdálenost mezi prvočísly[32] je předmětem intenzivního studia. Například je dokázáno, že v posloupnosti o $n-1$ prvcích $n!+2, n!+3, \ldots, n!+n$ se nevyskytuje žádné prvočíslo. Důkaz je jednoduchý: první člen je dělitelný 2, další 3 a tak dále až po poslední člen, který je dělitelný n. Můžeme tedy snadno sestrojit libovolně dlouhou posloupnost, kde se žádné prvočíslo nevyskytuje. Faktoriál roste velmi rychle, takže bychom mohli předpokládat, že se dlouhé mezery mezi prvočísly budou vyskytovat až u velmi vysokých čísel, ale není tomu tak úplně. Například vzdálenost prvočísla:

   2937032340680225901587237661044194634257090755748117620985887982178957288586767281432 27,

   které obsahuje 87 cifer. Podle aproximace z Prvočíselné věty je hodnota ln uvedeného prvočísla zhruba 199. To znamená, že v okolí tohoto prvočísla by zhruba každé 199 číslo mělo být prvočíslem. Ale další prvočíslo je ve skutečnosti vzdálené dalších 8350 hodnot, což je téměř 42 krát více, než předpovídaná hodnota z Prvočíselné věty. Je to dosud největší známá anomálie.

[31] https://cs.wikipedia.org/wiki/Prvo%C4%8D%C3%ADseln%C3%A1_v%C4%9Bta
[32] https://en.wikipedia.org/wiki/Prime_gap

Interval I_{N+1}je přesně dvakrát delší než předchozí interval I_N. Délka intervalů tedy rychle roste a postupně se začne projevovat přirozený úbytek prvočísel uvnitř intervalu.

Vezměme si například Interval $I_{36} = < 2^{36}; 2^{37} - 1 >$. Řez $I_{36,35}$ je horní polovina intervalu I_{36}, tedy interval $< 2^{36} + 2^{35}; 2^{37} - 1 >$, což je souvislý interval délky 2^{35}. Obsahuje nepochybně méně prvočísel než nevybraná spodní spodní polovina intervalu $< 2^{36}; 2^{36} + 2^{35} - 1 >$.

Na druhou stranu řez $I_{36,1}$ dělí interval I_{36} na velmi mnoho čtveřic, ze kterých vždy vybírá horní dva prvky. Čísla tohoto řezu jsou velmi rovnoměrně rozmístěna uvnitř celého intervalu. Přirozený úbytek prvočísel se neprojeví, stanovená základna $\frac{1}{2}\pi_N$ pro takovýto řez funguje dobře.

V řezech s vyšším indexem j se tedy musí vyskytovat menší počet prvočísel, neboť v porovnání s rovnoměrnými řezy s nižšími indexy j zabírají větší část oblasti vyšších čísel.

Nyní je evidentní, že stanovení konstantní základny pro všechny řezy intervalu je tedy nepochybně zatížené chybou danou postupným úbytkem prvočísel.

Pozorovaný pokles na grafu nedosahuje ani jednoho procenta. Zároveň je efekt úbytku prvočísel také poměrně malý. Nabízí se tedy otázka, zda chyba při stanovení základny daná úbytkem prvočísel dokáže pozorování plně vysvětlit.

Další postup se tedy zaměří na výpočet přirozeného úbytku prvočísel a stanovení vhodnější základny pro vzájemné porovnávání hodnot $A_{N,j}$.

Funkce $li(x), Li(x)$ a $x/ln(x)$

Ze zmíněné Prvočíselné věty lze odvodit aproximaci funkce $\pi(x)$.

$$\pi(x) \sim \frac{x}{ln(x)}$$

Ukazuje se však, že pro naše účely je vhodnější použít integrál funkce obráceného logaritmu [33], běžně označované jako $li(x)$:

$$li(x) = \int_0^x \frac{dt}{ln\, t}$$

Konkrétně použijeme drobnou úpravu této funkce a zavedeme funkci $Li(x)$:

$$Li(x) = li(x) - li(2)$$

Z prvočíselné věty opět vyplývá, že funkce $Li(x)$dobře aproximuje funkci $\pi(x)$:

$$\pi(x) \sim Li(x)$$

Tato funkce je v námi zkoumané oblasti přesnější[34] aproximací funkce $\pi(x)$.

[33] https://en.wikipedia.org/wiki/Logarithmic_integral_function
[34] https://en.wikipedia.org/wiki/Prime_number_theorem#Table_of_%CF%80(x),_x_/_log_x,_and_li(x)

Pro vlastní výpočet $li(x)$ je nejvhodnější zvolit tzv. Ramanujanův[35] rozvoj funkce do nekonečné řady, která ale rychle konverguje. Není cílem popisovat integrály a pochopit podstatu této funkce. Pro účely této práce postačí převzít Ramanujanův vzorec bez hlubšího pochopení problematiky.

Korekce pomocí funkce $Li(x)$

Nyní je třeba pomocí funkce $Li(x)$ spočítat očekávaný počet prvočísel pro každý řez $I_{N,j}$ a použít jej pro odvození individuální základny pro měření odchylky. Každý řez se skládá ze sady dílčích intervalů. Pro každý z nich je možné odečíst hodnotu funkce na jeho konci a začátku a takto vypočtené hodnoty pro všechny dílčí intervaly sečíst. Označme takto definovanou hodnotu pro řez $I_{N,j}$ symbolem $Li_{N,j}$. Rozdíl funkce $Li(x)$ pro začátek a konec intervalu I_N obdobně označíme Li_N. Zde je na místě velká preciznost při definování začátku a konce intervalu. Podívejme se na příklad pro I_5:

- Interval I_5 obsahuje dle výše uvedené kapitoly o bitových řezech všechna celá čísla 32 až 63. Začátek intervalu je 32.
- Konec intervalu I_5 je ve dvojkové soustavě číslo 63 a číslo 64 již patří do následujícího intervalu. Pro zapsání čísla 64 ve dvojkové soustavě by bylo zapotřebí o jeden bit více. Při aproximaci počtu prvočísel na intervalu I_5potřebujeme odhadnout počet prvočísel na interval délky 32. Prosté odečtení $Li(63) - Li(32)$ by představovalo odhad pro interval délky pouze 31. Při výpočtu je nutné považovat konec každého intervalu až bodem, kde začíná interval následující. Zde konkrétně tedy je korektní odhad pro I_5 dán rozdílem $Li(64) - Li(32)$.

Uvedené pravidlo pro konec intervalu je nutné aplikovat obecně na délky všech dílčích intervalů každého řezu. Do algoritmu pro výpočet vždy vstupuje konec intervalu vyjádřený ve dvojkové soustavě. Konec je tedy nutné zvýšit o 1. Zavedeme tedy pomocnou funkci$DeltaLi$ pro výpočet na celočíselném intervalu $< x, y >$ takto:

$$DeltaLi(x, y) = Li(y + 1) - Li(x)$$

Pomocí ní pak zavedeme precizně $Li_{N,j}$ a Li_N takto:

$x = min(s), y = max(s)$ označují nejmenší a největší celé číslo obsažené v intervalu s

$Li_{N,j} = \sum_{s \in \text{dílčí intervaly } I_{N,j}} DeltaLi(x, y)$, kde $x = min(s), y = max(s)$

$$Li_N = Li(2^{N+1}) - Li(2^N)$$

Zároveň platí, že $Li_N = DeltaLi(x, y)$, kde x je nejmenší celé číslo v I_N a y je poslední celé číslo obsažené v I_N. Definice Li_N je tedy konzistentní s definicí $Li_{N,j}$. Například konkrétně pro I_5 dostáváme $Li_5 = Li(2^6) - Li(2^5)$ = $Li(64) - Li(32) = DeltaLi(32,63)$.

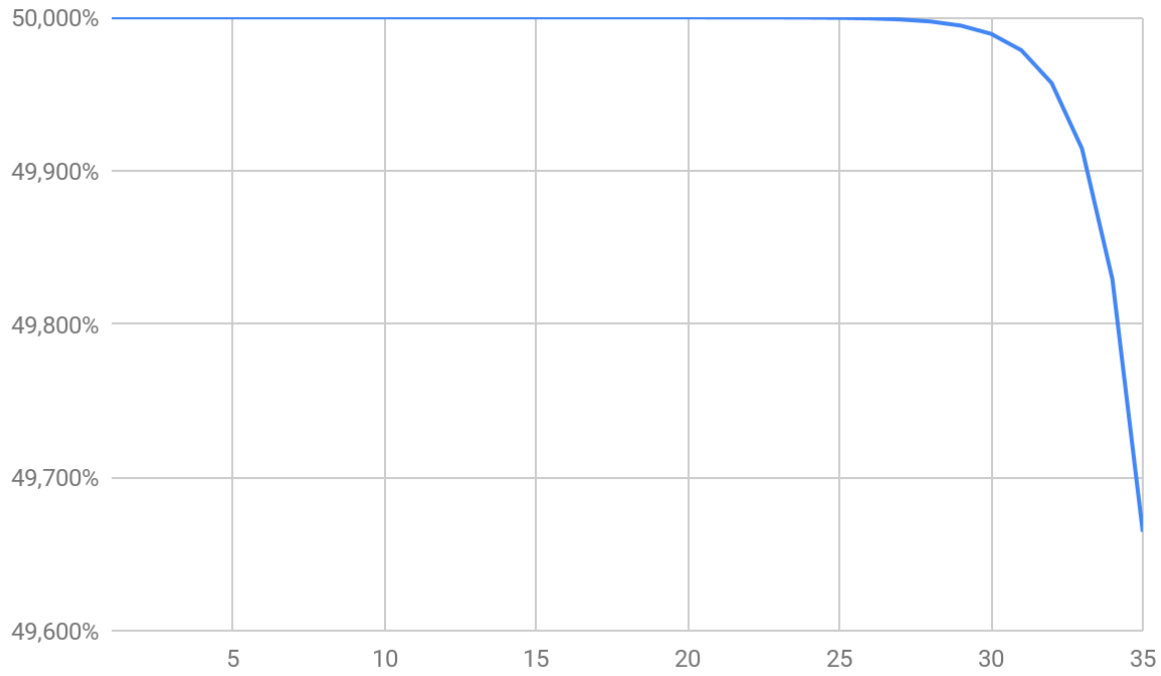
Není vůbec zřejmé, že takto vypočtená hodnota je dobrou aproximací počtu prvočísel pro zkoumaný řez, protože Prvočíselná věta se nevyjadřuje o chování prvočísel na konečném intervalu, ale mluví o celkové konvergenci, když x roste nade všechny meze. Naštěstí pro

[35] https://en.wikipedia.org/wiki/Logarithmic_integral_function#Series_representation

výpočet základny není nutné používat přímo hodnotu $Li_{N,j}$. Pro opravu základny stačí použít relativní pokles hodnot $Li_{N,j}$ pro jednotlivé řezy. Konkrétně pro všechny přípustné řezy i intervalu I_N spočteme poměr $Li_{N,j}$ k hodnotě Li_N, která je vztažená k celému intervalu:

$$LiP_{N,j} = \frac{Li_{N,j}}{Li_N}$$

Následující graf zobrazuje vypočtené hodnoty $LiP_{36,j}$ pro $j = 1..35$. Hodnota j označující řez je na ose x. Osa y zobrazuje hodnoty LiP v procentech. Pokud bychom chtěli zanést do grafu ekvivalent předchozí základny, byla by to konstantní hodnota 50% pro všechny řezy. Hodnota $LiP_{36,35} \approxeq 49{,}664\%$, takže rozdíl oproti předchozí základně převyšuje 0.3%.



Graf 6

Takto spočítané poměry $LiP_{N,j}$ stačí aplikovat na skutečný počet prvočísel intervalu π_N, čímž získáme nové hodnoty základny pro všechny řezy.

$$LiB_{N,j} = LiP_{N,j} \cdot \pi_N$$

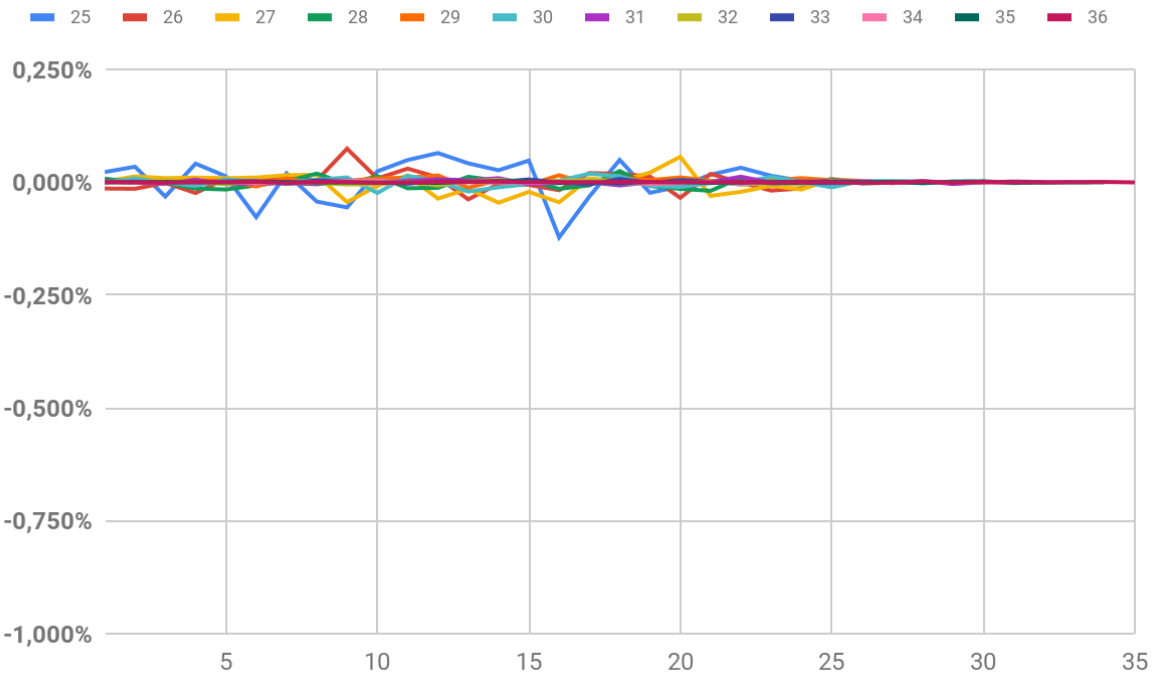
Hodnoty $LiB_{N,j}$ tedy vyžadují očekávaný počet prvočísel pro daný řez $I_{N,j}$.

Měření se zohledněním úbytku prvočísel

Použitím nově definovaného základu $LiB_{N,j}$ můžeme spočítat očištěné hodnoty očekávaného počtu prvočísel pro daný řez analogicky k výpočtu původní hodnoty $A_{N,j}$

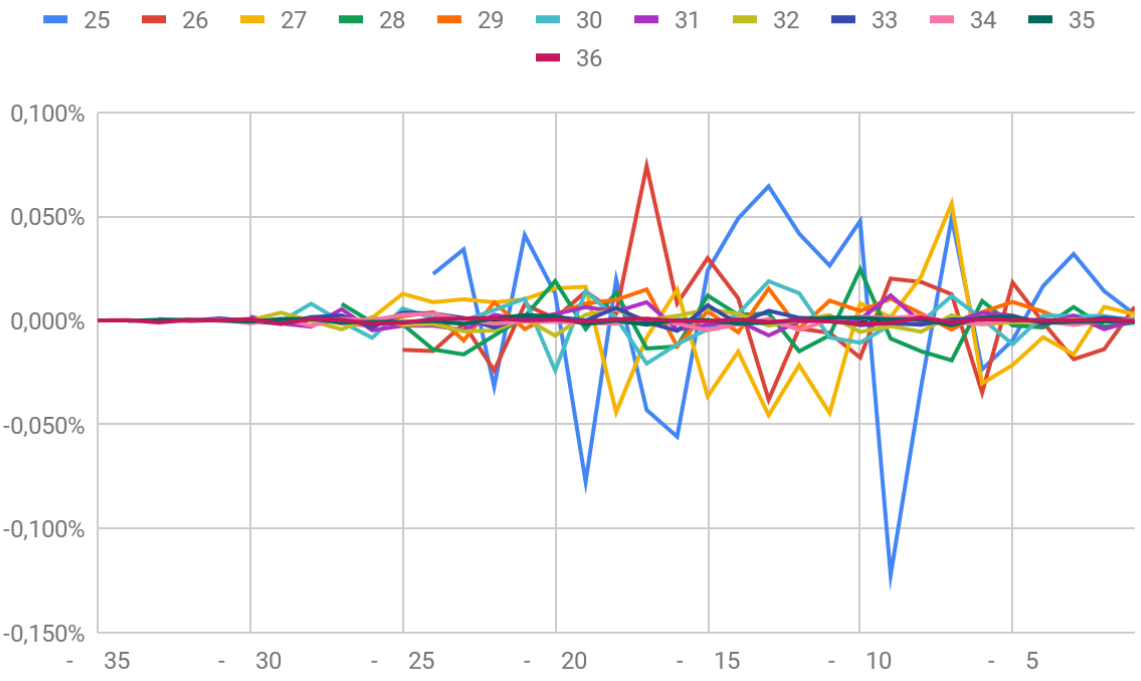
$$B_{N,j} = \frac{\pi_{N,j} - LiB_{N,j}}{LiB_{N,j}} \; pro\; N > 2, pro\; j \in < 0, N >$$

Praktické měření skutečně ukazuje, že pozorovaná anomálie z předchozího měření byla odstraněna a chyba je celkově malá pro všechny možné řezy všech zkoumaných intervalů, což znázorňuje následující graf. Jeho měřítko odpovídá měřítku grafu, který zobrazoval data analogické měření hodnot $A_{N,j}$ před korekcí.



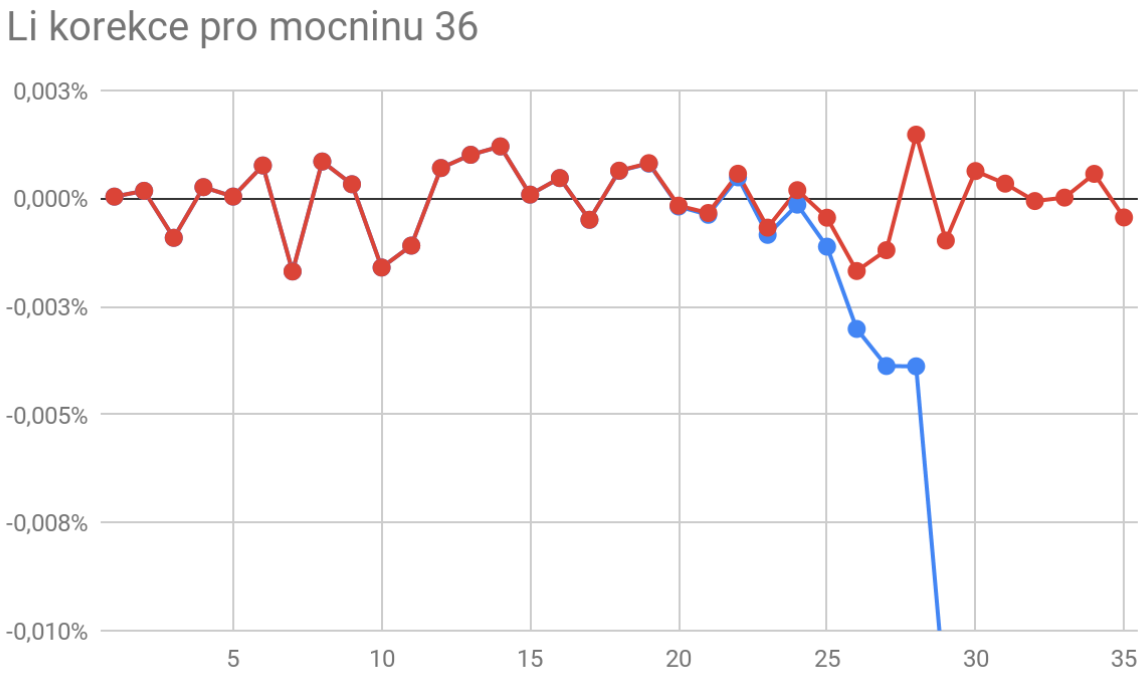
Graf 7

Při analyzování tohoto grafu je nutné nezapomenout, že zobrazené křivky jsou různě dlouhé a hodnoty 35 dosáhne pouze jedna křivka – vztažená k intervalu I_{35}. Všechny křivky jsou zarovnané zleva a osa x má přímo význam indexu povinně nastaveného bitu každého řezu. Křivky ovšem můžeme zarovnat i zprava. Potom nejvíce pravý bod bude zobrazovat vždy nejhrubší řez $I_{N,N-1}$, který vybírá z celého intervalu jen jeho horní polovinu. Hodnoty na ose x se pak nebudou vztahovat ke stejnému bitu, ale budou mít význam relativní pozice počítané od nejvyššího bitu a dává tedy smysl ji číslovat záporně. Například index nejvyššího bitu zmenšený o 1 atd. Tento pohled znázorňuje následující graf. Na tomto grafu je i díky menšímu rozsahu osy y lépe vidět skutečnost, že pro vyšší mocniny má měřená odchylka tendenci se ve všech měřených řezech zmenšovat.



Graf 8

Další graf ukazuje detail pouze pro nejvyšší spočítanou mocninu 36. Na grafu je úmyslně omezené měřítko osy y stokrát zmenšeno (z původní hodnoty -1% na -0,01%), aby byl dobře vidět efekt nastupující korekce pomocí funkce $Li(x)$. Zároveň je vidět, že původní základna je pro řezy s nízkým indexem odpovídá nové základně odvozené od li a vliv korekce se projevuje teprve pro řezy s vyšším indexem. Červeně je zde vyobrazen interval pro $B_{N,j}$ a modře pro $A_{N,j}$.



Graf 9

Korigovaná měření je možné interpretovat takto:

- Velikost odchylky je nyní pro všechny řezy a všechny pozorované intervaly srovnatelná a bez nějakého trendu.
- Korekce tedy byla dobře definována a odstranila pozorovanou anomálii plně na všech měřených intervalech a na všech jejich řezech.
- Korigovaný graf zobrazuje pouze drobné odchylky, které vyplývají přímo z podstaty prvočísel a které jsou nevyhnutelné pro měření na libovolném ale konečném intervalu či jeho řezu.
- Dosud univerzálně pozorovaná rovnoměrnost rozmístění prvočísel se vždy vyjadřuje k nějakému velkému celku (celá aritmetická řada nebo v tomto případě dostatečně dlouhý interval). Nikdy ale nepředstavovala snahu o nalezení exaktního vzorce, který by postihl každé individuální prvočíslo. Na rozdíl od grafu hodnot $A_{N,j}$ korigovaný graf vypovídá o podstatě prvočísel, nezobrazuje žádnou chybu, kterou by bylo možné dále očistit. Pokud připustíme trochu poezie: prvočísla se sice podřizují celkovému řádu, individuálně je však nelze spoutat žádným jednoduchým předpisem (přirozeně s výjimkou jejich vlastní definice).

Směrodatná odchylka

Míru konvergence pro konkrétní interval je možné vyjádřit pomocí směrodatné odchylky (anglicky standard deviation[36] - SD). Definujme nejprve součet kvadrátů rozdílů skutečné hodnoty prvočísel a odpovídajícího základu:

[36] https://en.wikipedia.org/wiki/Standard_deviation

$$dB_N = \sum_{j=1}^{N-1} (\pi_{N,j} - LiB_{N,j})^2$$

A tento základ je nutné normalizovat vydělením počtem rozdílů a aplikováním odmocniny:

$$SDB_N = \sqrt{\frac{dB_N}{N}}$$

Výsledný vztah je inspirován vzorcem pro výpočet směrodatné odchylky statistické veličiny a používá se běžně pro vyjádření velikosti chyby. Jeho velkou výhodou je, že je vyjádřen ve stejných jednotkách, jako je měřená veličina, a je tedy možné porovnávat měřenou veličinu a její směrodatnou odchylku. Roli měřené veličiny zde spolehlivě zastane polovina skutečného počtu prvočísel daného intervalu.

$$SDBP_N = \frac{SDB_N}{0.5 \cdot \pi_N}$$

Tento výraz vyjadřuje relativní velikost chyb (rozdíl skutečnosti a základu) všech řezů intervalu I_N vztaženou k přibližnému počtu prvočísel každého řezu (zde zjednodušeně polovina prvočísel celého intervalu).

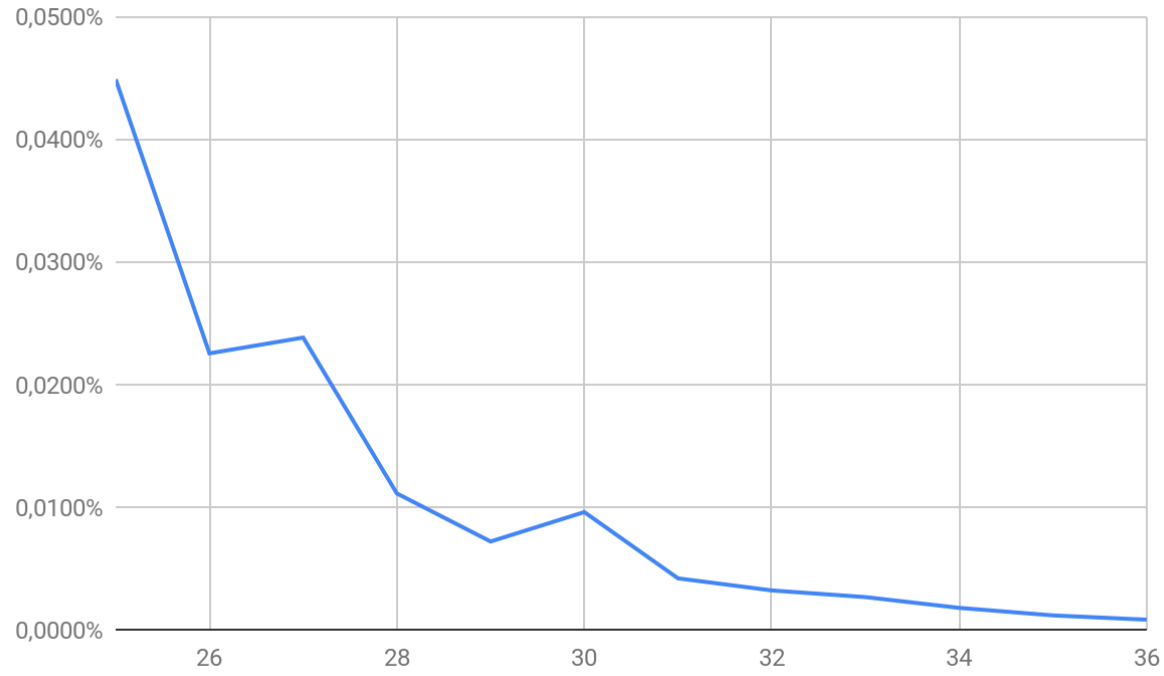
Zcela analogicky lze uvedené definice vztáhnout ke skutečným hodnotám prvočísel a původně navrženému základu založeném na prostém průměru.

$$dA_N = \sum_{j=1}^{N-1} (\pi_{N,j} - 0.5 \cdot \pi_N)^2$$

$$SDA_N = \sqrt{\frac{dA_N}{N}}$$

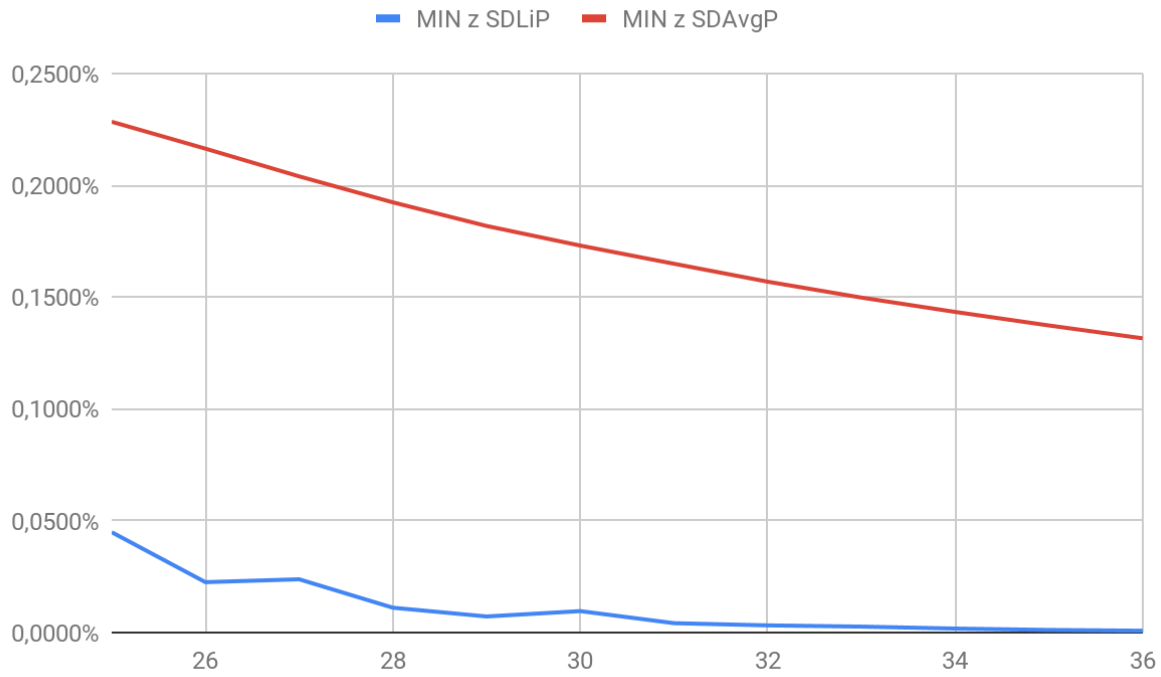
$$SDAP_N = \frac{SDA_N}{0.5 \cdot \pi_N}$$

Hodnoty $SDBP_N$ lze spočítat a zobrazit v následujícím grafu, který zobrazuje tyto hodnoty pro $N = 25..36$. Z grafu je evidentní postupné snižování chyby SDB_N.



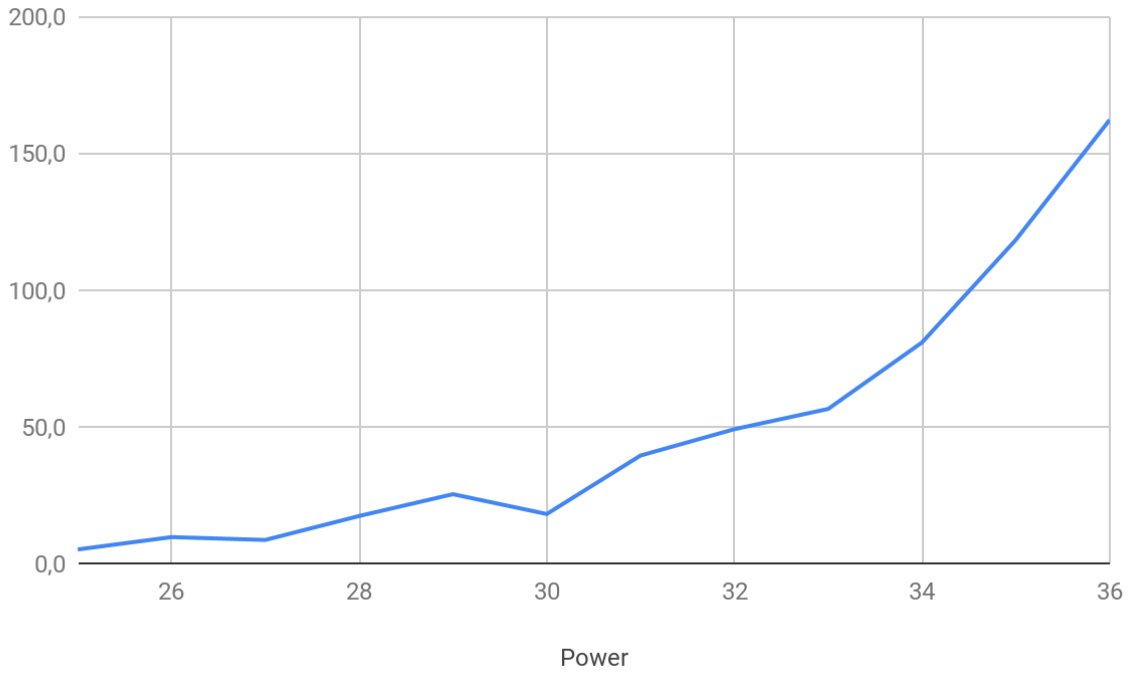
Graf 10

Analogicky je možné porovnat hodnoty SDA_N a SDB_N. Následující graf ukazuje, že korekce pomocí funkce $Li(x)$ vede k výrazně menší SDB_N chybě.



Graf 11

Z grafu však nevyplývá, jak významně lepší jsou hodnoty SDB_N oproti hodnotám $SDBA_N$. Jejich podíl zobrazuje následující graf. Z grafu je vidět, že chyba SDB_N je řádově lepší a tento trend s rostoucím N prudce roste.



Graf 12

Hypotéza založená na korigovaném měření

Uvedené měření a klesající trend chyby SDB_N nás opravňuje k formulaci nové hypotézy:

$$\lim_{N\to\infty} max_{0<j<N}(B_{N,j}) = 0$$

Hypotéza tak očekává obdobný výsledek měření i pro všechny delší intervaly a předpokládá, že s délkou intervalu se postupně velikosti odchylek pro všechny řezy budou snižovat. Hypotéza se vyjadřuje najednou o všech možných řezech podobně jako se Dirichletův teorém vyjadřuje o všech aritmetických posloupnostech. Hypotéza hovoří o rovnoměrnosti rozmístění prvočísel novým způsobem.

Hypotéza založená na nekorigovaném měření

Zkusme se nyní zamyslet, zda je opodstatněné zformulovat analogickou hypotézu pro nekorigovanou hodnotu $A_{N,j}$.

$$\lim_{N\to\infty} max_{0<j<N}(A_{N,j}) = 0$$

Zde je na místě značná opatrnost, neboť u hrubších dělení každého intervalu odchylka $A_{N,j}$ výrazně klesá (tj. její velikost v absolutní hodnotě roste). Vraťme se znovu k interpretaci této odchylky:

- Hlavní příspěvek odchylky je dán přirozeným úbytkem prvočísel.
- Úbytek prvočísel se nejvýrazněji projevuje u hrubých dělení intervalu a u každého intervalu I_N je největší u řezu $I_{N,N-1}$.
- Absolutní hodnota odchylky $A_{N,N-1}$se s rostoucím N snižuje.

Pro řádné podložení hypotézy tedy stačí odhadnout velikost vlivu úbytku prvočísel pro nejhrubší řezy $A_{N,N-1}$ a zkoumat, zda s rostoucím N se tento vliv bude blížit 0. Kontrolně pak porovnáme tyto odhady s vypočtenými hodnotami odchylek $A_{N,N-1}$.

Odhad vlivu úbytku prvočísel pro nejhrubší řezy

Interval $I_{N,N-1}$ se skládá jen z jednoho intervalu, který

- začíná hodnotou $2^N + 2^{N-1} = 2^{N+1} - 2^{N-1} = 3 \cdot 2^{N-1}$
- a končí hodnotou $2^{N+1} - 1$.

Dosazením do definice získáme:

- $Li_{N,N-1} = Li(2^{N+1}) - Li(3 \cdot 2^{N-1})$
- $Li_N = Li(2^{N+1}) - Li(2^N)$.

Přirozený úbytek prvočísel pak byl vyjádřen podílem těchto hodnot označovaným $LiP_{N,N-1}$. Pro zkoumání chování této funkce pro velmi velká čísla by bylo nutné pracovat s definicí funkce Li, což by vyžadovalo pracovat s integrály. Pokusme se nyní provést odhad hodnot $LiP_{N,N-1}$ za využití jednoduššího matematického aparátu a integrálům se vyhnout. Tuto možnost nám dává již zmiňovaná Prvočíselná věta, ze které vyplývá, že chyba mezi funkcemi $Li(x)$a $x/ln(x)$ postupně konverguje k 0. Při výpočtu LiP tedy nahradíme všude funkci $Li(x)$ funkcí $x/ln(x)$ a postupně krátkým výpočtem získáme:

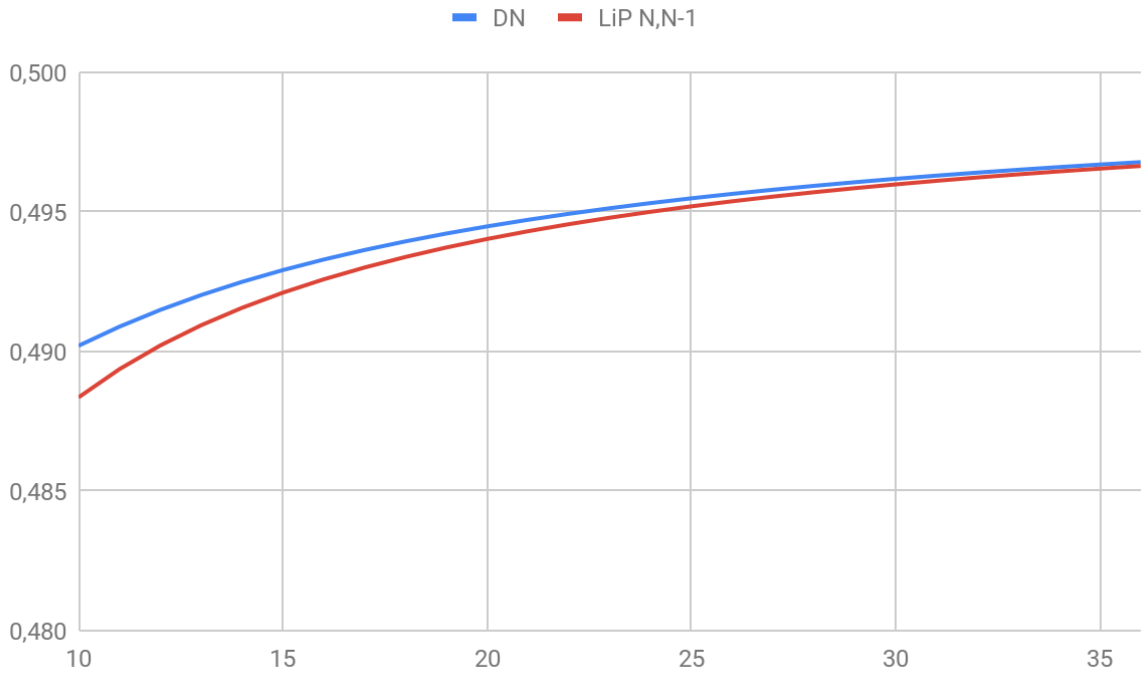
$$Li_N \sim \frac{2^{N+1} \cdot ln_2 e}{N+1} - \frac{2^N \cdot ln_2 e}{N} = \frac{2^N \cdot (N-1) \cdot ln_2 e}{N \cdot (N+1)}$$

$$Li_{N,N-1} \sim \frac{2^{N+1} \cdot ln_2 e}{N+1} - \frac{3 \cdot 2^{N-1}}{ln(3 \cdot 2^{N-1})} = 2^N \cdot ln_2 e \cdot \frac{(0.5 \cdot N - 3.5 + 2 \cdot ln_2 3) \cdot N}{(N+1) \cdot (N-1+ln_2 3)}$$

Podíl obou výrazů pak představuje dobrou aproximaci $LiP_{N,N-1}$:

$$LiP_{N,N-1} \sim D_N = \frac{N \cdot (0.5 \cdot N - 3.5 + 2 \cdot ln_2 3)}{(N-1) \cdot (N-1+ln_2 3)}$$

Kontrolní porovnání spočtených hodnot $LiP_{N,N-1}$ a jejich aproximací D_N zobrazuje následující graf:



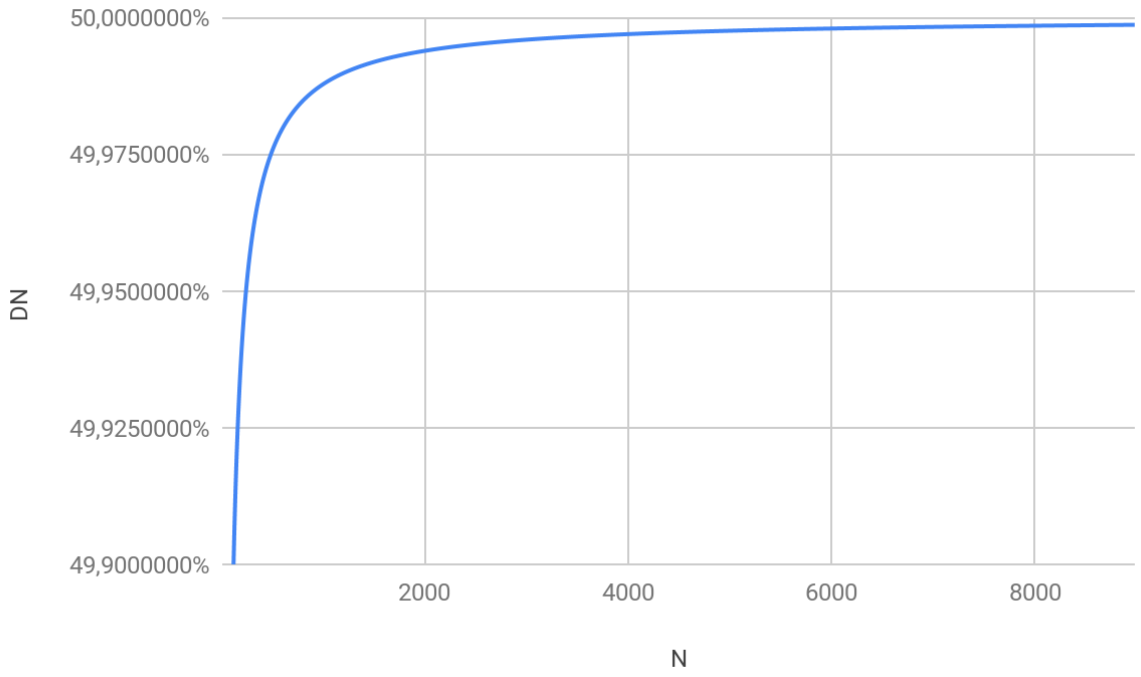
Graf 13

Výhodou funkce D_N však je, že se jedná o jednoduchý vzorec, který lze snadno spočítat i pro velké hodnoty N a zkoumat její konvergenci. Zde konkrétně stačí zkoumat koeficienty u kvadratických členů u proměnné N v čitateli i jmenovateli, neboť pro dostatečně veliká N lze vliv ostatních členů (lineární členy N a konstanty) v porovnání s kvadratickým členem N^2 zanedbat. Získáme:

$$\lim_{N\to\infty} D_N = \frac{0.5 \cdot N^2 + \dots}{N^2 + \dots} = 0.5$$

Velikost přirozeného úbytku prvočísel tedy s rostoucím N postupně klesá a počet prvočísel i u řezu $I_{N,N-1}$ s největší odchylkou se postupně blíží k polovině prvočísel celého intervalu. Touto úvahou je tedy i hypotéza pro hodnoty $A_{n,j}$ řádně podložena.

Následující graf zobrazuje vývoj D_N pro větší hodnoty N a potvrzuje výpočet a konvergenci D_N k hodnotě 0.5, která v grafu odpovídá hodnotě 50%.



Graf 14

Úvaha nad možným důkazem hypotéz

Hypotézy jsou inspirované Dirichletovou větou. Nicméně jejich případný důkaz zřejmě nebude jednoduchá modifikace důkazu Dirichletovy věty[37]. Velmi zběžným pohledem do tohoto důkazu je zřejmé, že obsahuje velmi pokročilý matematický aparát (Fourierova funkce, Dirichletova L funkce, Riemannova zeta funkce v reálném oboru a její převod na Eulerův součin prvočísel a mnohé další), který ale nevypovídá o chování na konečných intervalech. Případný důkaz tedy může být obtížný. Je samozřejmě možné, že matematik zběhlý v teorii čísel by byl schopen předložit jednoduchý důkaz uvedené hypotézy vyplývající z poznatků oboru, které nemám ani já, ani můj odborný konzultant.

Řezy definované počtem bitů

Definice řezů

Zkoumané intervaly lze rozřezat zcela jiným způsobem. Čísla v intervalu lze rozdělit do několika tříd podle toho, kolik obsahují nastavených bitů. Označme

$$S(x) = \sum_{i=0}^{N} x_i, \text{ kde } x = \sum_{i=0}^{N} x_i \cdot 2^i, x_i \in \{0,1\}$$

Vstupem do této funkce je celé kladné číslo. To je jednoznačně možné rozložit na součet mocnin 2, což přímo odpovídá reprezentaci čísla ve dvojkové soustavě. Funkce vrátí počet nastavených bitů. Příklady:

[37] http://math.uchicago.edu/~may/REU2012/REUPapers/LiAng.pdf

- $S(280) = S(1000\,1100_2) = S(2^8 + 2^4 + 2^3) = 3$
- $S(2^N) = 1$ pro libovolné N. Všechny mocniny dvou mají nastavený pouze jeden (nejvyšší) bit.
- $S(2^N - 1) = N - 1$ pro libovolné N. Mocnina 2 zmenšená o 1 má nastaveny všechny bity.
- $S(0) = 0$
- $S(1) = 1$

V daném intervalu $I_N$mají všechny hodnoty nastaven nejvyšší bit s indexem N. Zaveďme nový typ řezu:

$$R_{N,j} = \{x: x \in I_N, S(x) = j + 1\} \text{ pro } 0 \leq j \leq N, N > 1$$

Všechna čísla intervalu $I_N$mají nutně nastavený nejvyšší bit. Příspěvek tohoto bitu v definici ošetříme pomocí přičtení čísla 1 k indexu j. Příklad:

Desítkově	Dvojkově	Desítkově	Dvojkově
16	$1\,0000_2$	24	$1\,1000_2$
17	$1\,0001_2$	25	$1\,1001_2$
18	$1\,0010_2$	26	$1\,1010_2$
19	$1\,0011_2$	27	$1\,1011_2$
20	$1\,0100_2$	28	$1\,1100_2$
21	$1\,0101_2$	29	$1\,1101_2$
22	$1\,0110_2$	30	$1\,1110_2$
23	$1\,0111_2$	31	$1\,1111_2$

- $R_{4,0} = \{16\}$
- $R_{4,1} = \{17,18,20,24\}$
- $R_{4,2} = \{19,21,22,25,26,28\}$
- $R_{4,3} = \{23,27,29,30\}$
- $R_{4,4} = \{31\}$
- Řez $R_{4,1}$ tedy obsahuje pouze čísla z intervalu I_4, která mají vedle povinného nejvyššího bitu s indexem 4 nastaven právě jeden nižší bit.

Rozmístění obsažených čísel tedy není v porovnání s bitovými řezy intuitivní. Důvodem je skutečnost, že pokud nahradíme při rozkladu čísla na součet mocnin 2 vyšší mocninu mocninou nižší, vzniklé číslo zůstane ve stejném řezu, přestože se jeho umístění uvnitř intervalu (někdy i značně) změní.

Dále vidíme, že tyto řezy nejsou stejně dlouhé. Počet prvků v řezu lze spočítat pomocí tzv. kombinačního čísla[38], které udává, kolika způsoby (bez ohledu na pořadí a bez opakování) lze vybrat z dané množiny (zde množina bitů) několik prvků (bity, které nastavíme na 1). Příklad:

- Počet prvků řezu $R_{4,3}$lze spočítat pomocí kombinačního čísla $\binom{4}{3} = \frac{4!}{1!\cdot 3!} = 4$. Každé číslo v řezu má 5 bitů, ale nejvyšší bit je vždy nastaven. Vybíráme tedy z množiny zbývajících 4 bitů s indexy $0,1,2,3$ celkem 3 bity.

Řezy s nízkým indexem ja symetricky i řezy s vysokým indexem j jsou výrazně kratší. Naopak řezy s indexem j blízké $\frac{N}{2}$ jsou velmi dlouhé.

Všechna prvočísla (s výjimkou čísla 2) jsou lichá a mají nastaven i nejnižší (nultý) bit. Definujme tedy nový řez, který se omezí pouze na lichá čísla:

$$T_{N,j} = \{x: x \in R_{N,j}, x\ je\ liché\}$$

Každé číslo v řezu $T_{N,j}$ se skládá z $N+1$bitů, přičemž bity 0 a N jsou vždy povinně nastaveny (příslušnost k intervalu I_N a lichost). K výběru tedy zbývá $N-1$ bitů. Počet prvků řezu $T_{N,j}$ je pak dán následujícím kombinačním číslem:

$$|T_{N,j}| = \binom{N-1}{j-1}$$

Příklad:

Desítkově	Dvojkově	Desítkově	Dvojkově
17	$1\,0001_2$	25	$1\,1001_2$
19	$1\,0011_2$	27	$1\,1011_2$
21	$1\,0101_2$	29	$1\,1101_2$
23	$1\,0111_2$	31	$1\,1111_2$

- $T_{4,0} = \{\}$
- $T_{4,1} = \{17\}$
- $T_{4,2} = \{19,21,25\}$
- $T_{4,3} = \{23,27,29\}$
- $T_{4,4} = \{31\}$

[38] https://cs.wikipedia.org/wiki/Kombina%C4%8Dn%C3%AD_%C4%8D%C3%ADslo

Počet prvočísel řezu

Nyní můžeme přistoupit ke zkoumání rovnoměrnosti rozmístění prvočísel na řezech $T_{N,j}$. Počet prvočísel obsažených v řezu $T_{N,j}$ označme:

$$\Pi_{N,j} = \left|\{x: x \in T_{N,j}\ a\ x\ je\ prvočíslo\}\right|$$

K vzájemnému srovnání hodnot $\Pi_{N,j}$ opět potřebujeme jejich počet vztáhnout relativně k počtu prvočísel celého intervalu:

$$q_{N,j} = \frac{\Pi_{N,j}}{\pi_N}$$

Očekávaný počet prvočísel

Každý řez má ale jiný počet prvků. Řezy $T_{N,j}$ jednoho intervalu I_N obsahují dohromady všechna lichá čísla intervalu. Jejich počet je tedy 2^{N-1}.

$$Z_{N,j} = \frac{|T_{N,j}|}{2^{N-1}}$$

$Z_{N,j}$ tak vlastně říká, kolik procent lichých čísel řez $T_{N,j}$ zabírá.

Vyváženost řezů

Hodnota $Z_{N,j}$ nezohledňuje přirozený úbytek prvočísel, naopak předpokládá, že každé liché číslo je stejně dobrý kandidát pro výskyt prvočísla. Nabízí se otázka, změřit, nakolik jsou jednotlivé řezy nevyvážené a zda případná nevyváženost nezpůsobí, že některý řez by představoval nevhodně nastavené očekávání ohledně předpokládaného počtu prvočísel. Skutečně již na výše uvedeném příkladu řezů $T_{4,j}$ je možné nevyváženost pozorovat:

$T_{4,1} = \{17\}$	$Z_{4,1} = \frac{1}{8}$	pouze jedna hodnota zcela nevyvážená směrem k nízkým číslům
$T_{4,2} = \{19,21,25\}$	$Z_{4,2} = \frac{3}{8}$	nejdelší řez nevyvážený ve prospěch menších čísel
$T_{4,3} = \{23,27,29\}$	$Z_{4,3} = \frac{3}{8}$	nejdelší řez nevyvážený ve prospěch vyšších čísel
$T_{4,4} = \{31\}$	$Z_{4,4} = \frac{1}{8}$	pouze jedna hodnota zcela nevyvážená směrem k velkým číslům

Každopádně bychom vzhledem k přirozenému úbytku prvočísel měli předpokládat, že řezy nevyvážené směrem k nižším číslům by měly vykazovat nadbytečný počet prvočísel oproti takto nastavenému očekávání (a naopak pro opačně nevyvážené řezy).

Nevyváženost řezů lze měřit různými způsoby.

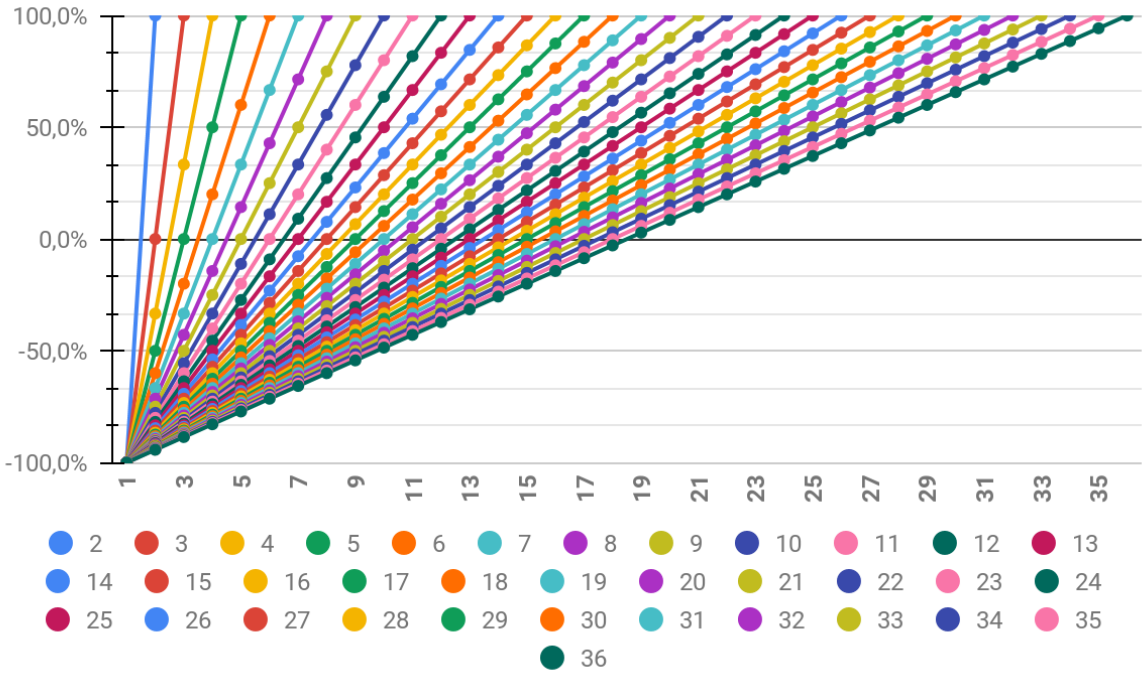
Nevyváženost řezů – průměr

První způsob měření míry nevyváženosti řezu pouze jednoduchým způsobem počítá průměr ze všech čísel obsažených v řezu.

Aby bylo možné porovnávat navzájem řezy různých intervalů, je vhodné spočtené průměry jednotlivých řezů převést na relativní umístění této veličiny v rámci intervalu:

- Hodnotu průměru nějakého řezu ležící přesně ve středu intervalu umístíme na hodnotu 0.
- Hodnotu průměru řezu ležící přesně na nejnižším lichém čísle intervalu umístíme na hodnotu -100%. Toto je vždy případ řezů $T_{N,1}$, které obsahují jen jedno nejnižší liché číslo řezu, které má nastavené právě dva bity: bit nejvyšší, který definuje interval a bit nejnižší, který mají nastavená všechna čísla.
- Hodnotu průměru řezu ležící přesně na nejvyšším lichém čísle intervalu umístíme na hodnotu -100%. Toto je vždy případ řezů T_{NN}, které obsahují jen jedno největší liché číslo řezu, které má nastavené všechny bity.

Tako umístěný průměr odpovídá fyzikální představě těžiště, kdy jednotlivé hodnoty jsou umístěné v určité vzdálenosti od středu intervalu, přičemž dvě hodnoty umístěné symetricky okolo středu intervalu nerovnováhu jednotlivých čísel navzájem vyruší. Na druhou stranu jediné číslo umístěné na okraji intervalu má na pomyslné váze díky největší páce největší dopad. Výsledek pro řezy $T_{N,j}$, kde $N = 2..36$ a $j = 1..N$ je zobrazen na následujícím grafu:



Graf 15

Na ose x je index j. Jednotlivé intervaly I_N jsou znázorněny pomocí různých barev. Osa y popisuje míru nevyvážení těžiště řezu. Například červený graf vlevo odpovídá intervalu I_3, který má celkem 3 řezy:

- $T_{3,1}$ má těžiště maximálně nevyvážené vlevo (ostatně jako i všechny ostatní řezy $T_{N,1}$, které obsahují vždy pouze nejmenší liché číslo intervalu) na hodnotu -100%.
- $T_{3,1}$ je rovnoměrně vyvážený, neboť jeho hodnoty $\{11,13\}$ jsou symetricky umístěné kolem středu intervalu a těžiště má tedy hodnotu 0%.
- $T_{3,1}$ má těžiště maximálně nevyvážené pravo (ostatně jako i všechny ostatní řezy $T_{N,N}$, které obsahují vždy pouze největší liché číslo intervalu) na hodnotu 100%.

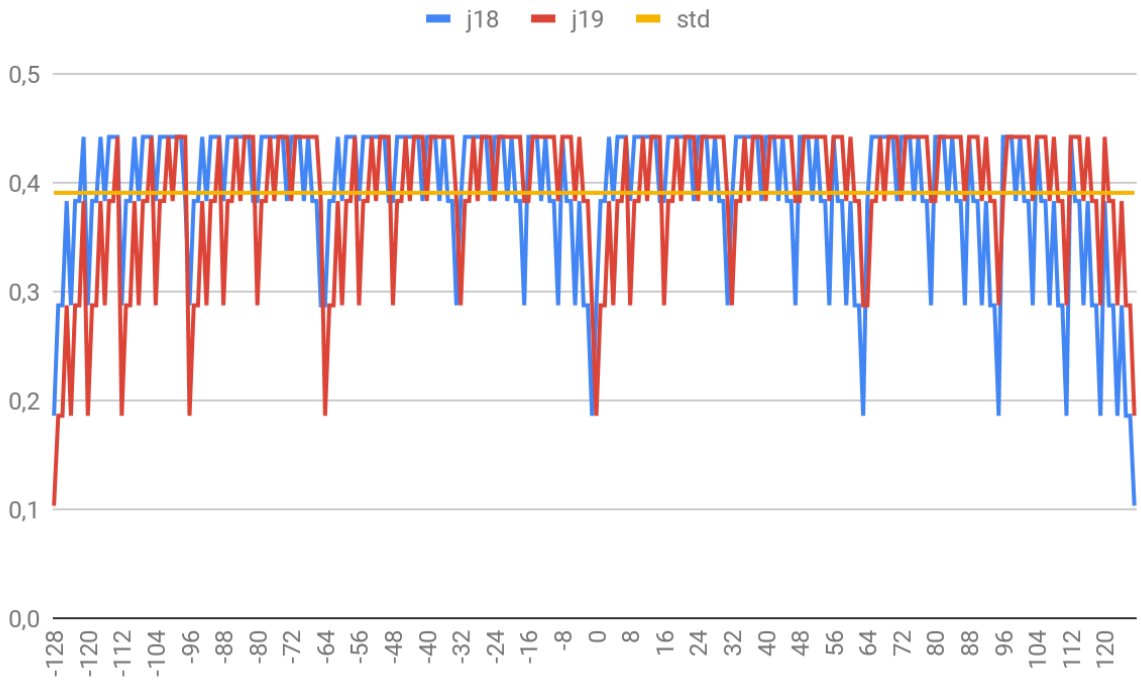
Z grafu je vidět, že těžiště u všech intervalů se zvyšujícím se j rovnoměrně posouvá doprava. Tento závěr je podložený pouze výpočtem a grafem. Pravděpodobně by šel dokázat i obecně, ale zvolil jsem nejprve metodu praktického měření a o obecný důkaz se ani nepokoušel.

Dále je z grafu možné pozorovat, že všechny liché intervaly $I_3, I_5, \dots I_{33}, I_{35}$ mají prostřední řez dokonale vyvážený s těžištěm v 0, ale všechny ostatní řezy nejsou vyvážené. Tento pohled na vyváženost intervalu však nijak nepopisuje, nakolik je řez symetrický kolem středu intervalu.

Nevyváženost řezů – histogram

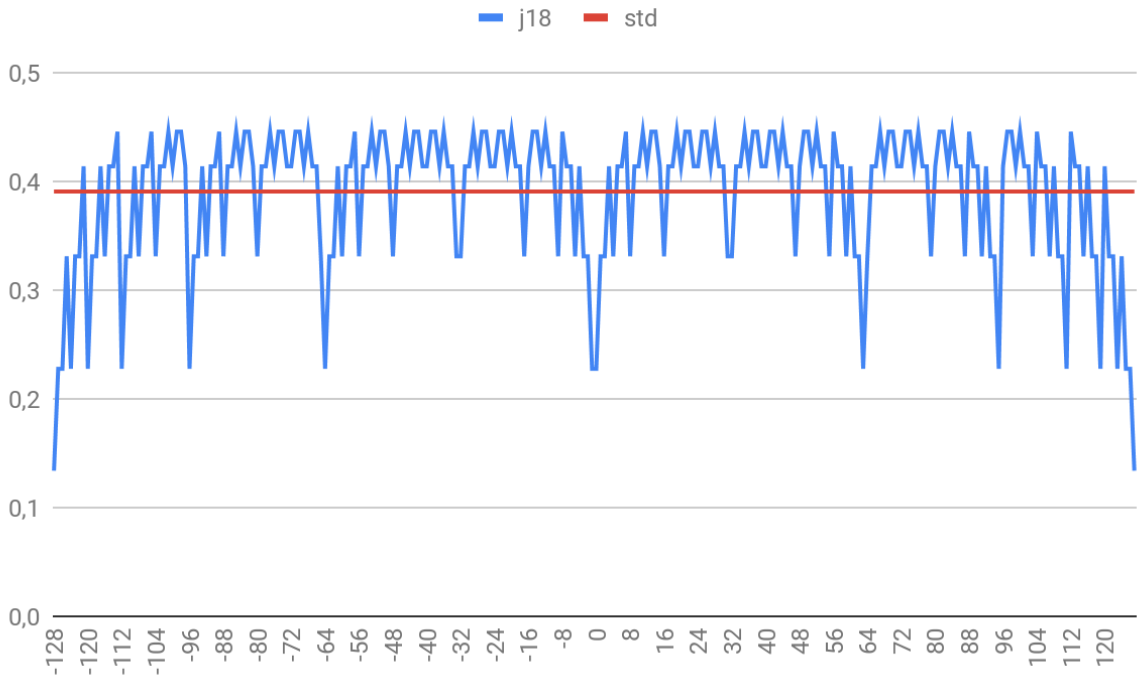
Druhý způsob, jak zkoumat míru nevyváženosti řezů je pomocí histogramu. Každý měřený interval je možné rozdělit na několik stejně dlouhých úseků a zkoumat, kolik čísel z daného řezu je součástí konkrétních úseků. Četnost zastoupení čísel je možné měřit v procentech vůči základu, který tvoří počet čísel daného řezu. Při tvorbě histogramu o 256 sloupcích by měl každý sloupec při naprosto rovnoměrném rozložení obsahovat $\frac{1}{256}$ všech čísel. Tato hranice je na následujících grafech označena jako *std.* První graf zobrazuje dva prostřední řezy intervalu I_{36}. Konkrétně histogramy řezů $T_{36,18}$ a $T_{36,19}$. Z grafu je vidět, že tyto dva řezy jsou

rozprostřeny po celé délce intervalu, na jeho krajích je čísel méně a uvnitř hodnoty fluktuují s oblastí mírného poklesu ve středu intervalu. Oba grafy mají stejnou strukturu: křivku jednoho získáme z druhého otočením kolem středu.



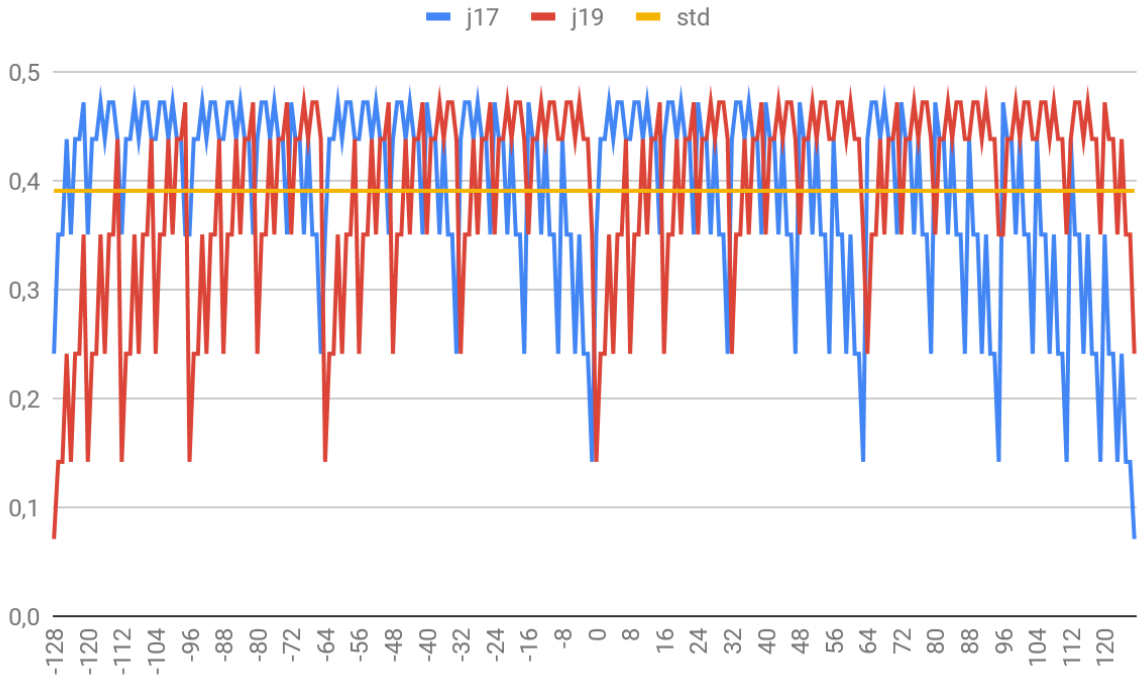
Graf 16

Na následujícím obrázku je histogram prostředního řezu intervaluI_{35}. Tento řez má těžiště ve středu intervalu a navíc je symetrický.



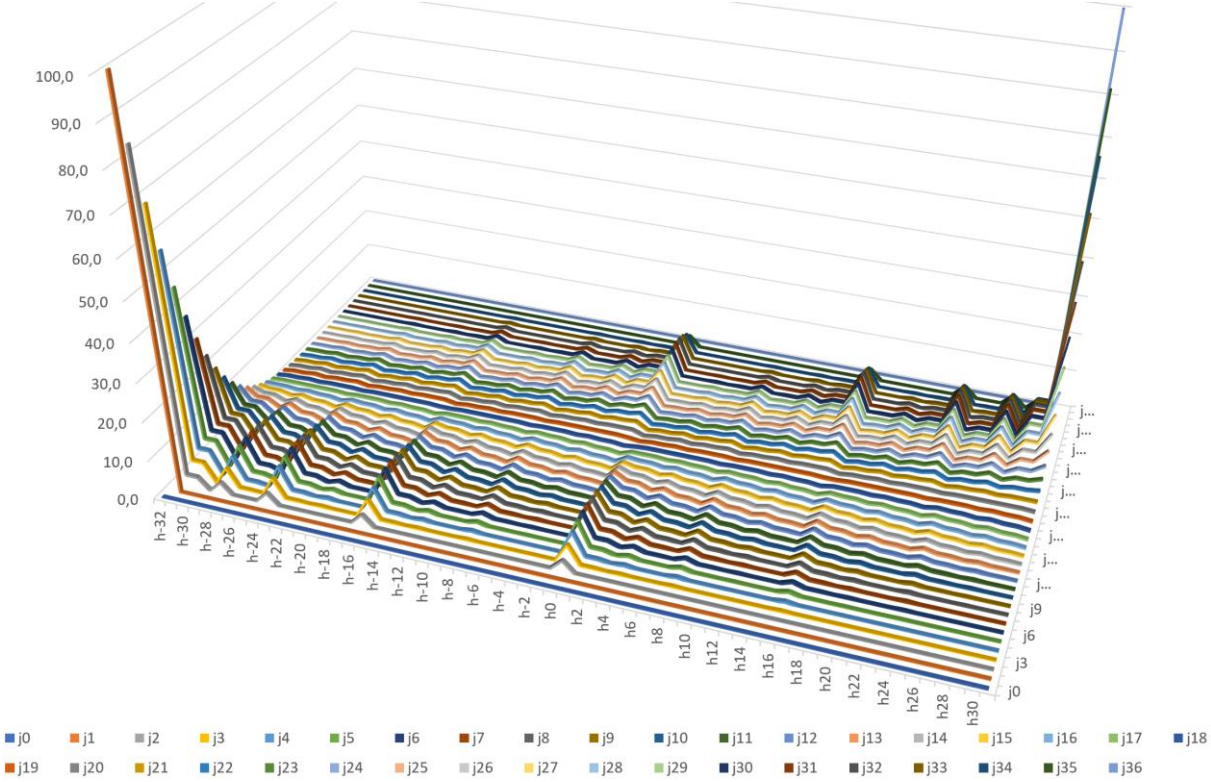
Graf 17

Na následujícím obrázku je histogram dvou řezů intervaluI_{35} sousedících s prostředním řezem. Řezy jsou opět rozprostřeny do celého intervalu, řezy 17 a 19 lze získat otočením kolem středu. Na tomto histogramu je ale vidět, že řez 17 je nevyvážen spíše k menším číslům a řez 19 naopak obsahuje více větších čísel.



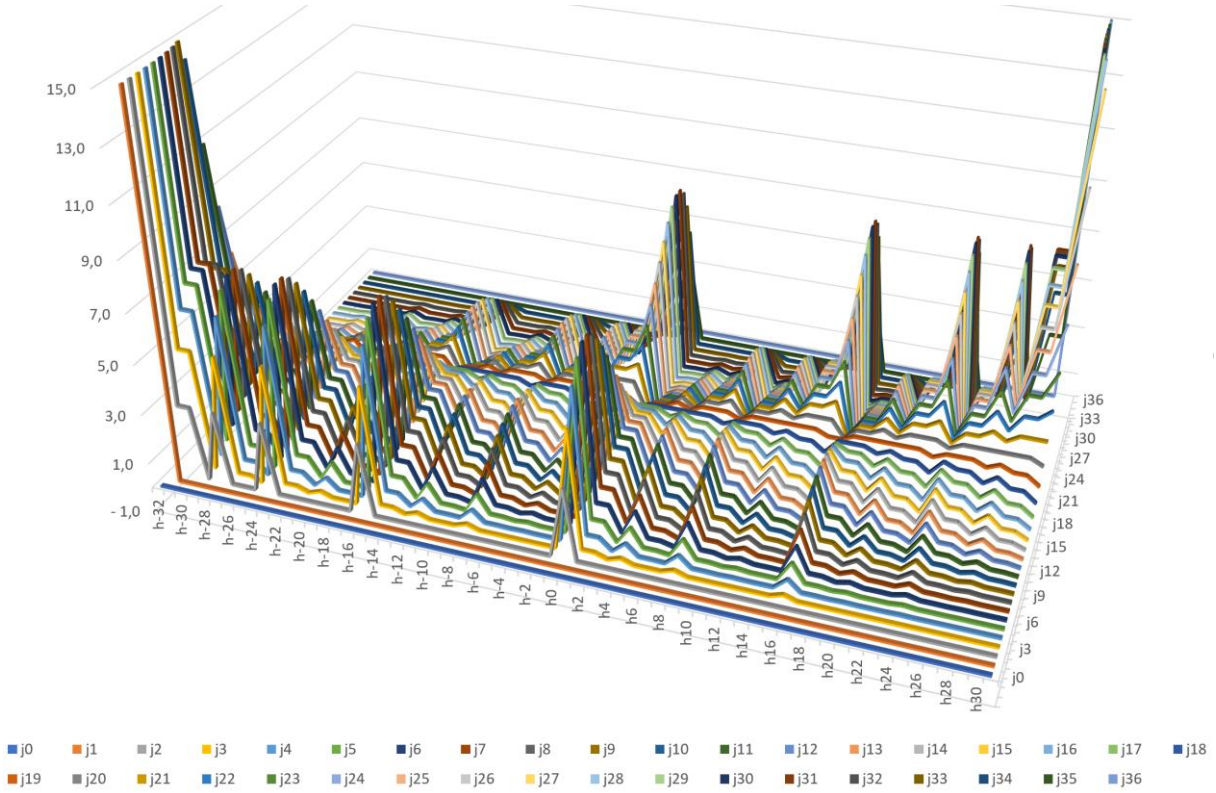
Graf 18

Následující trojrozměrný graf zobrazuje histogramy (rozdělené do 64 sloupců) všech řezů intervalu I_{36}.



Graf 19

Následující graf je pouze detailem předchozího grafu, kdy byla osa x omezena na hodnotu 15%, aby vynikly lépe detaily po oříznutí extrémních hodnot krajních řezů.



Graf 20

Všechny tyto grafy představují snahu pochopit neobvyklou charakteristiků bitových řezů $T_{N,j}$ ve snaze vysvětlit pilovité chování, které bude popsáno dále v textu.

Oba uvedené přístupy byly naměřeny za pomocí metody *SliceTBalanceTest* hlavního programu. Výsledky výpočtů jsou uloženy ve zvláštní tabulce:

https://docs.google.com/spreadsheets/d/1GyoSyiu4HjIPPDOo9XDm0DqVPG7EgTvbP-n8RLWHHd4/edit?usp=sharing

Porovnání očekávání a skutečnosti

Poměry $Z_{N,j}$ a $q_{N,j}$ lze přímo porovnávat, což je jejich hlavní výhoda. Definujme jejich rozdíl jako:

$$F_{N,j} = q_{N,j} - Z_{N,j}$$

Podobně jako v předchozí kapitole bychom mohli zavést i jejich podíl:

$$V_{N,j} = \frac{q_{N,j}}{Z_{N,j}}$$

Nicméně se ukazuje, že jejich hodnota se velmi často pohybuje velmi blízko hodnoty 1, takže rozdíl je v tomto případě praktičtější metrika.

Případně bychom mohli zavést i očekávaný počet prvočísel jako součin počtu prvočísel intervalu I_N a poměru $Z_{n,j}$:

$$L_{N,j} = Z_{N,j} \cdot \pi_N$$

A v dalším kroku bychom mohli zkoumat rozdíl skutečnosti od tohoto očekávání:

$$P_{N,j} = \Pi_{N,j} - L_{N,j}$$

Tyto hodnoty ovšem nelze mezi sebou vzhledem k různé délce řezů porovnávat.

Veličina $F_{N,j}$ je pro další postup nejvhodnější.

Příklad

Následující tabulka ukazuje všechny definované veličiny pro intervaly I_4 a I_5. Rozčleňovat několik kusů prvočísel do řezů nevypovídá o chování prvočísel, ale poslouží dobře pro ilustraci všech uvedených definic na konkrétním a snadno představitelném příkladě. Studium takto krátkých intervalů není cílem.

$\pi_4 = 5$

j	$T_{4,j}$	$\Pi_{4,j}$	$Z_{4,j}$	$q_{4,j}$	$F_{4,j}$	$V_{4,j}$	$L_{4,j}$	$P_{4,j}$
1	$\{\mathbf{17}\}$	1	$\frac{1}{8} = 12{,}5\%$	$\frac{1}{5} = 20\%$	$7{,}5\%$	$\frac{8}{5} = 1{,}6$	0,6	0,4
2	$\{\mathbf{19}, 21{,}25\}$	1	$\frac{3}{8} = 37{,}5\%$	$\frac{1}{5} = 20\%$	$-17{,}5\%$	$\frac{8}{15} \doteq 0{,}53$	1,9	$-0{,}9$
3	$\{\mathbf{23}, 27, \mathbf{29}\}$	2	$\frac{3}{8} = 37{,}5\%$	$\frac{2}{5} = 40\%$	$2{,}5\%$	$\frac{16}{15} \doteq 1{,}06$	1,9	0,1
4	$\{\mathbf{31}\}$	1	$\frac{1}{8} = 12{,}5\%$	$\frac{1}{5} = 20\%$	$7{,}5\%$	$\frac{8}{5} = 1{,}6$	0,6	0,4

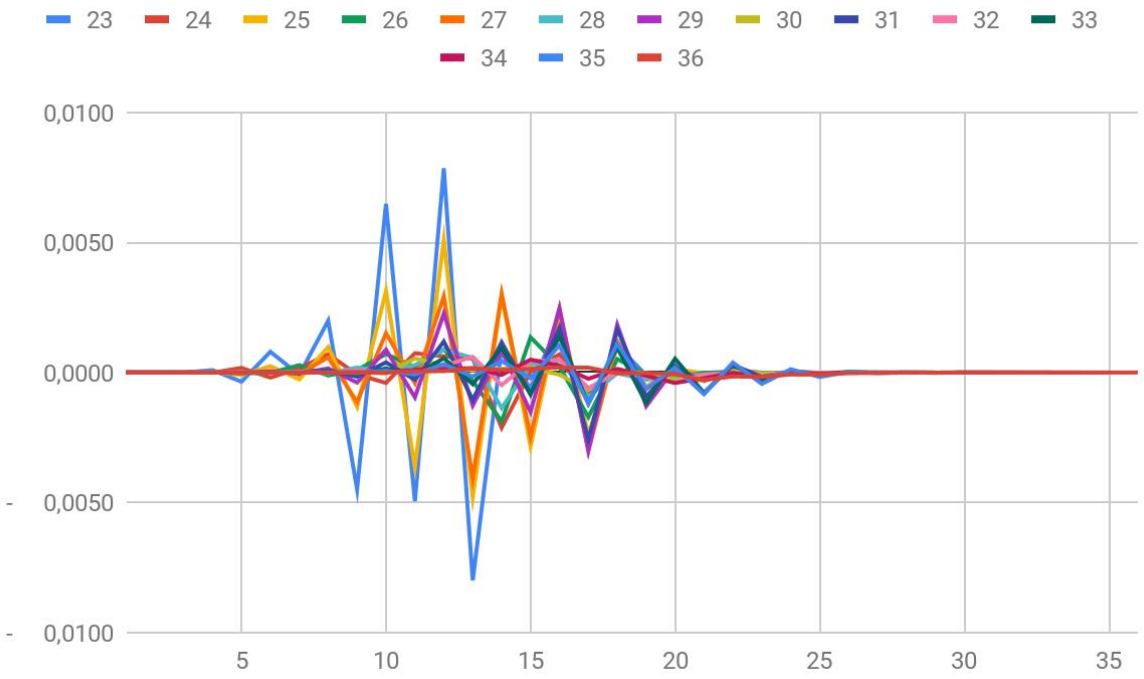
$\pi_5 = 7$

j	$T_{5,j}$	$\Pi_{5,j}$	$Z_{5,j}$	$q_{5,j}$	$F_{5,j}$	$V_{5,j}$	$L_{5,j}$	$P_{5,j}$
1	$\{33\}$	0	$\frac{1}{16} = 6{,}25\%$	0%	$-6{,}25\%$	0	$0{,}4$	$-0{,}4$
2	$\{35, \mathbf{37}, \mathbf{41}, 49\}$	2	$\frac{4}{16} = 25\%$	$\frac{2}{7} \doteq 28{,}57\%$	$3{,}57\%$	$1{,}14$	$1{,}8$	$0{,}3$
3	$\{39, \mathbf{43}, 45{,}51, \mathbf{53}, 57\}$	2	$\frac{6}{16} = 37{,}5\%$	$\frac{2}{7} \doteq 28{,}57\%$	$-8{,}93\%$	$0{,}76$	$2{,}6$	$-0{,}6$
4	$\{\mathbf{47}, 55, \mathbf{59}, \mathbf{61}\}$	3	$\frac{4}{16} = 25\%$	$\frac{3}{7} \doteq 42{,}86\%$	$17{,}86\%$	$1{,}71$	$1{,}8$	$1{,}3$
5	$\{63\}$	0	$\frac{1}{16} = 6{,}25\%$	0%	$-6{,}25\%$	0	$0{,}4$	$-0{,}4$

Měření

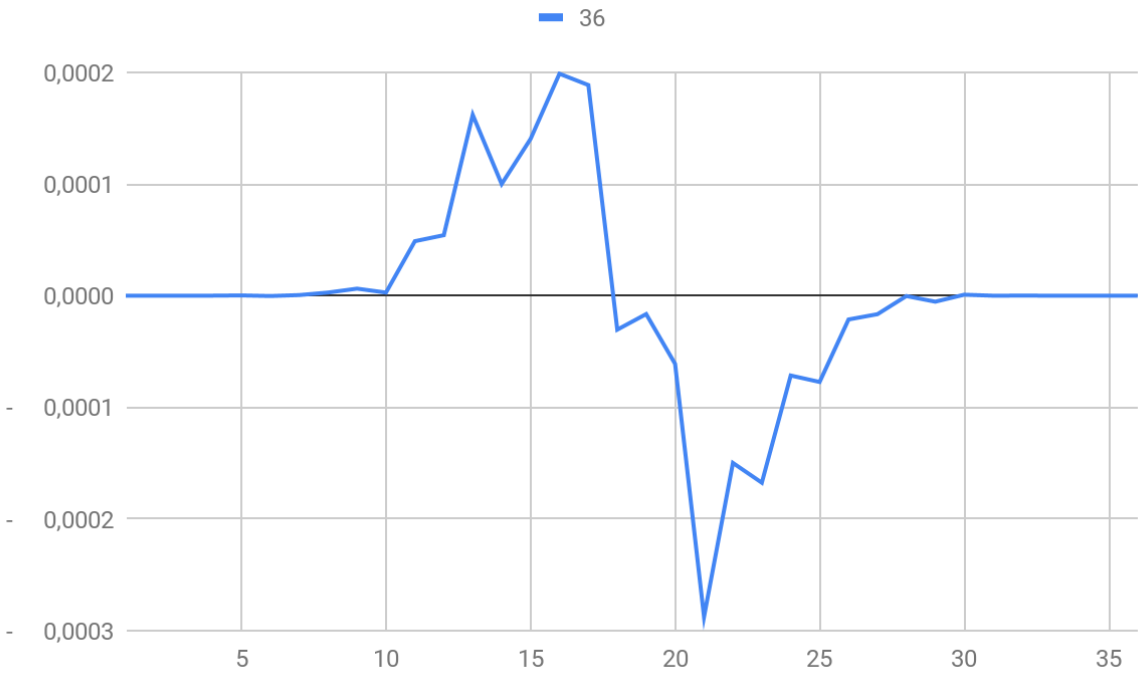
Následující graf zobrazuje rozdíly $F_{N,j}$ pro intervaly I_{23} až I_{36}. Osa x zobrazuje hodnoty j, osa y je v procentech. Z grafu je ihned vidět, že odchylka mezi očekávaným a skutečným počtem prvočísel nedosahuje ani 0.01%, přičemž pro vyšší intervaly se postupně zmenšuje.

Na tomto typu grafu je potřeba brát do úvahy, že řady jsou různě dlouhé a jsou zarovnané zleva. Nejdelší řezy pro každý interval jsou vždy pro hodnoty j blízko středu rozsahu $0..N$, zatímco krajní řezy jsou velmi krátké.



Graf 21

Pilovitý charakter většiny řad je skutečně podivný. Kladné hodnoty vyskytující se typicky pro lichá j indikují převahu prvočísel nad očekáváním - tj. jakoby prvočísla rozložitelná na sudý počet (lichý počet j nastavených bitů a nejvyšší bit určující celý interval) mocnin 2 převažovaly. Uvedený jev se však nevyskytuje na všech intervalech. Na následujícím obrázku je vidět detail pro $F_{36,j}$, kde tento jev neuvidíme. Takovýto průběh naopak odpovídá předpokladu, že přirozený úbytek prvočísel v kombinaci s nevyvážeností řezů vede na nadbytek prvočísel pro řezy j nalevo od středových řezů a naopak nedostatek prvočísel oproti definovanému očekávání u vyšších řezů napravo od středových řezů.



Graf 22

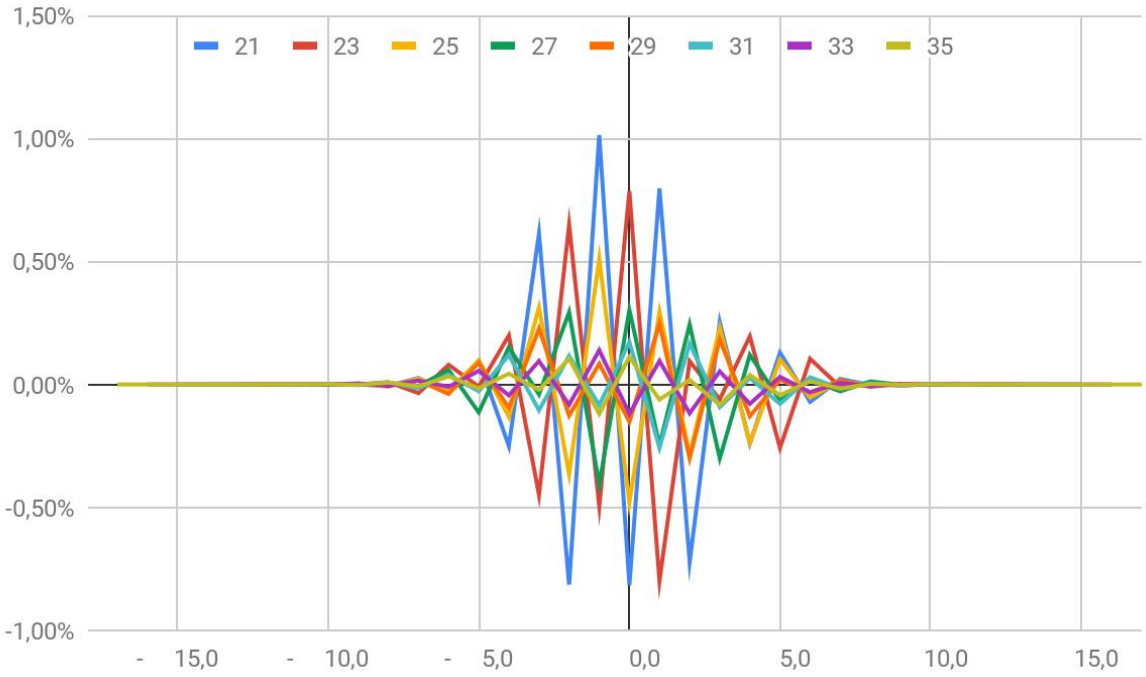
Liché a sudé řezy

Jde o překvapivé a rozhodně nejzajímavější pozorování této práce, které si zasluhuje další pozornost. Prvním krokem je zarovnání řad. Řada pro každý interval je různě dlouhá a přirozeným a logickým způsobem zarovnání je všechny řady seřadit dle jejich středu. V následujících grafech hodnota 0 na ose x představuje střed řezu. Situace je však odlišná pro intervaly s lichou a sudou hodnotou N, od které se dále odvíjí počet počet řezů:

- V případě lichých řezů (např. $T_{31,j}, T_{33,j}, T_{35,j}$) jsou všechny indexy j zarovnány tak, aby prostřední hodnoty indexu j daného řezu a zároveň nejdelší řezy (tj. např. řezy $T_{31,16}, T_{33,17}, T_{35,18}$) byly zobrazeny na ose x v hodnotě 0. Zde je dobré připomenout, že všechny řezy $T_{N,0}$ jsou prázdné (řez $R_{N,0}$ neobsahuje žádný další nastavený bit a tvoří jej jedno sudé číslo, kterým začíná interval I_N).
- V případě sudých řezů (např. $T_{30,j}, T_{32,j}, T_{34,j}$) jsou všechny indexy j zarovnány tak, aby prostřední dva nejdelší řezy (tj. např. řezy $T_{31,16}, T_{33,17}, T_{35,18}$) byly zobrazeny na ose x v hodnotě $-0{,}5$ a $0{,}5$.

Liché řezy

Na následujícím obrázku jsou znázorněny všechny liché řezy.

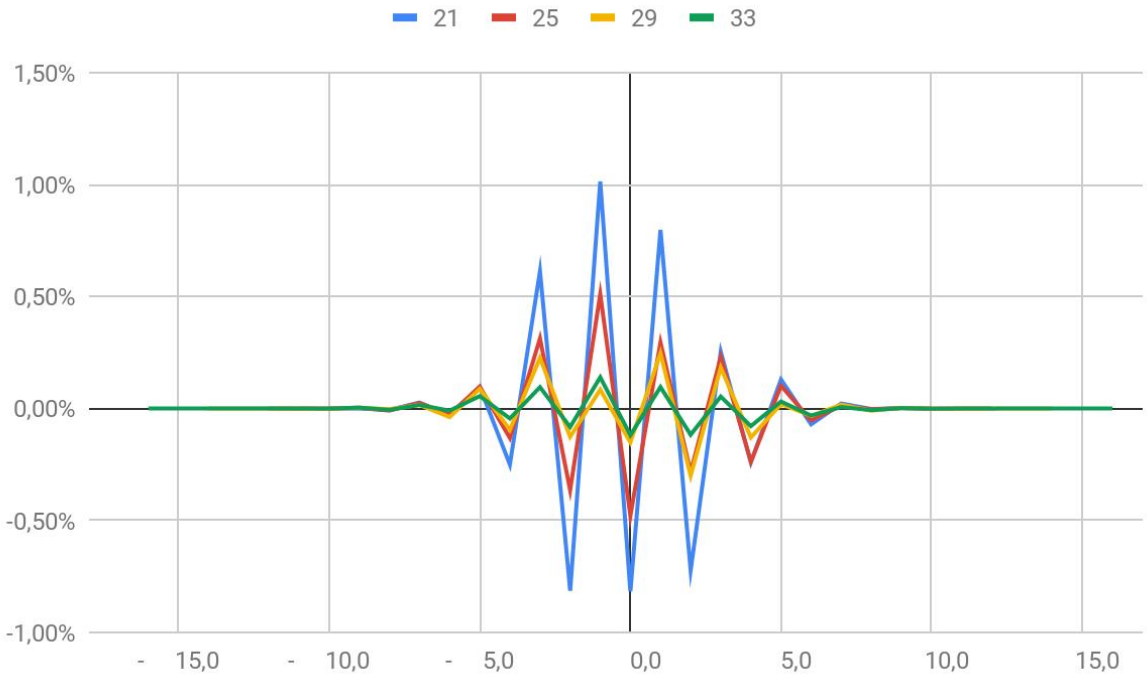


Graf 23

Z grafu je vidět, že kladné a záporné hodnoty se pravidelně střídají. Je to velmi zajímavé, ale nemám pro tento jev uspokojivé vysvětlení.

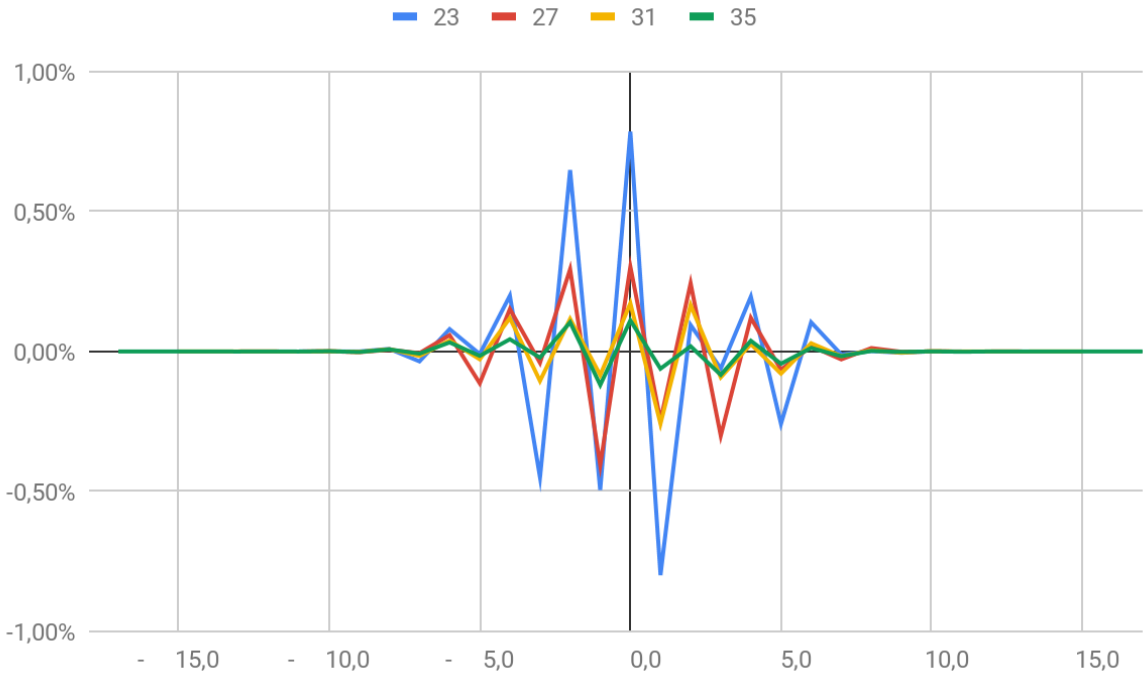
Dále je možné pozorovat vzorec, dle kterého je nejdelší řez kladný (přebytek prvočísel oproti očekávání), nebo naopak záporný (nedostatek prvočísel oproti očekávání), kde dochází k opakování s cyklem 4.

Na následujícím obrázku je vidět, že každý čtvrtý řez s počátečním intervalem I_{21} má v nejdelším řezu nedostatek prvočísel.



Graf 24

Na následujícím obrázku je naopak vidět, že každý čtvrtý řez s počátečním intervalem I_{23} má v nejdelším řezu přebytek prvočísel.

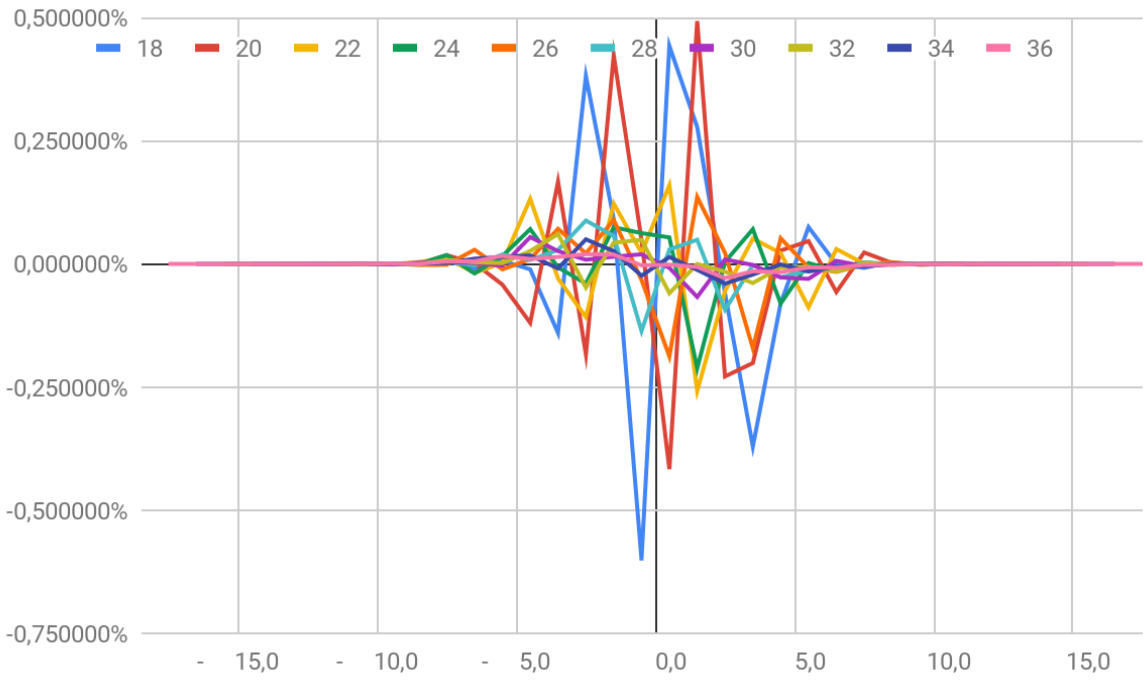


Graf 25

Samozřejmě, že počet naměřených intervalů je nízký, přesto je možné uvažovat o obecněji platném vzorci. Jeho možné vysvětlení bude v kapitole o spojených řezech.

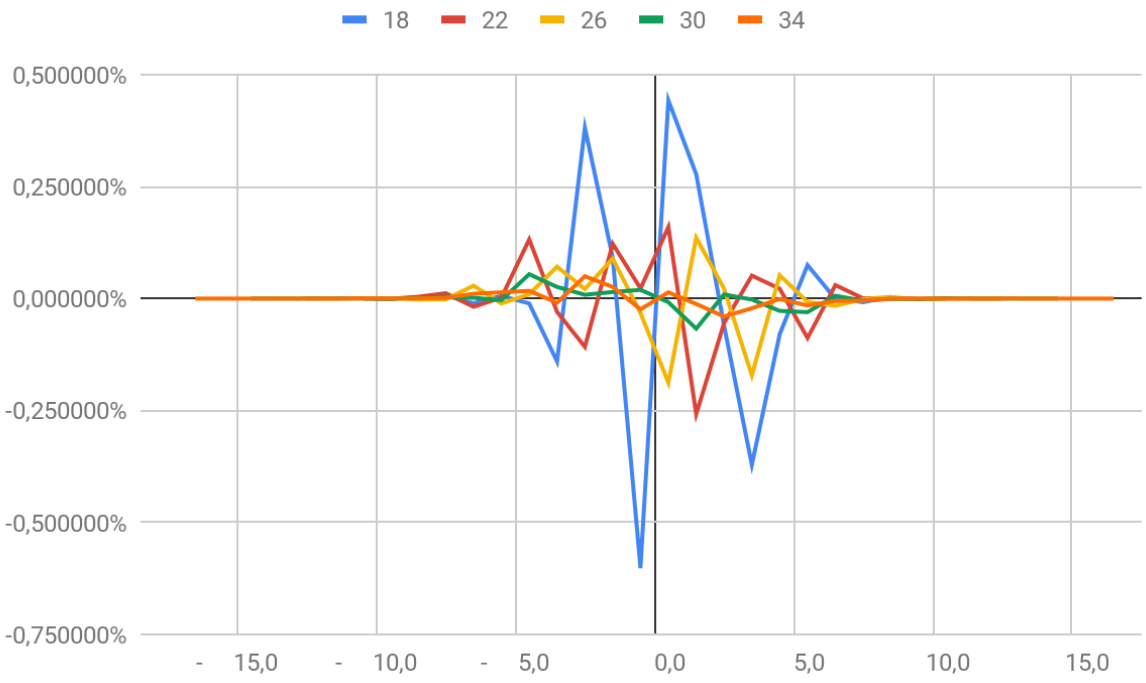
Sudé řezy

Sudé řezy nevykazují žádný podobný vzorec, jak je vidět na následujících grafech.



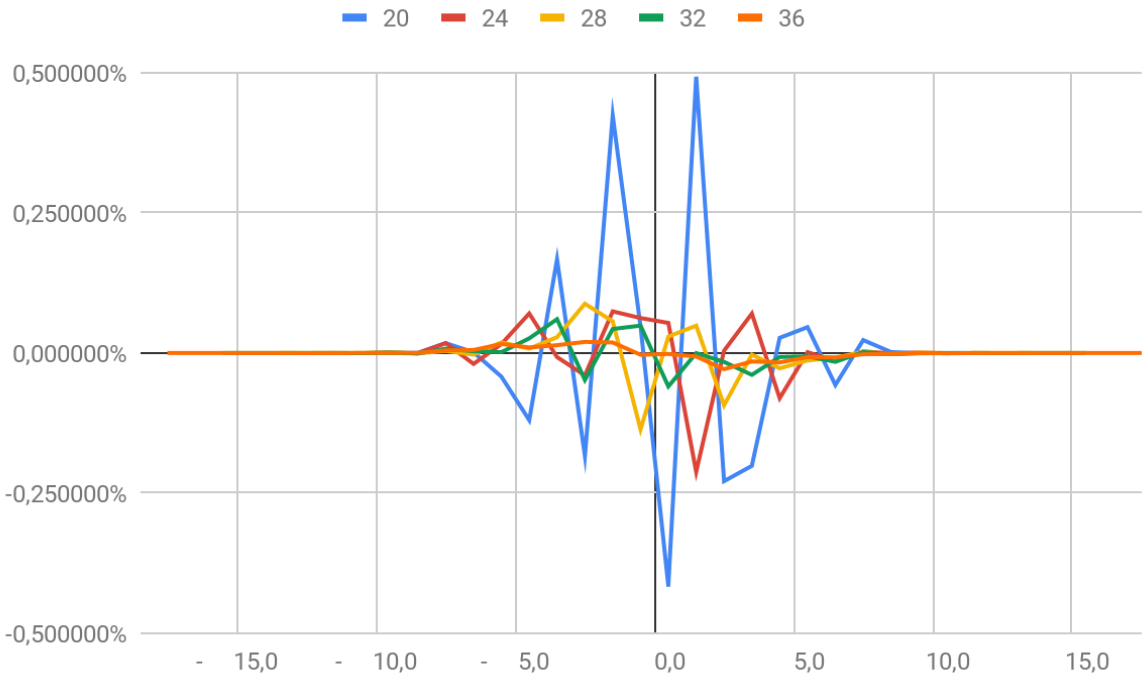
Graf 26

Výběr každého čtvrtého intervalu s počátkem v I_{18}.



Graf 27

Výběr každého čtvrtého intervalu s počátkem v I_{20}.



Graf 28

Hodnocení

Absolutní velikost odchylek je velmi nízká, takže bychom mohli na první pohled usuzovat, že zvolený přístup definice řezů a studia očekávanému počtu prvočísel se osvědčil. Při bližším zamyšlení je ale znepokojivé, že pilovité charakteristiky lichých intervalů nedokážu vysvětlit. Předpokládám, že pilovitý charakter lichých řezů nepředstavuje nějaký revoluční objev, ale že je spíše nějakým způsobem svázán se specifickou definicí řezů a podstaty jejich nevyváženosti. Otázka si zasluhuje hlubší zkoumání, které je předmětem následujících kapitol.

Symetrie kolem středu intervalu

Některé ze spočtených grafů ukazují na symetrické chování kolem středu intervalu. Nabízí se myšlenka toto chování blíže prozkoumat. Označme střed intervalu I_N jako

$$M_N = 2^N + 2^{N-1}$$

Z uvedené definice ihned plyne, že:

- je to číslo sudé, které není součástí žádného řezu $T_{N,j}$
- jeho dvojkový zápis je $110\ldots0_2$, kde počet nul je $N-1$

Každé liché číslo intervalu I_N má uvnitř intervalu zrcadlový protějšek. Definujme vzdálenost čísla od středu intervalu takto:

$$d(x) = |x - M_N|$$

Dále zavedeme dvojici čísel P_x a L_x tak, aby byly umístěny symetricky v intervalu I_N symetricky kolem jeho středu - L_x nalevo od středu a P_x napravo od středu:

$$L_x = M_N - d(x)$$

$$P_x = M_N + d(x)$$

Je zřejmé, že vždy jedno číslo z dvojice P_x a L_x splývá s x a druhé je jeho zrcadlovým obrazem. Pokud je x liché je dvojice P_x a L_x rovněž lichá, neboť střed M_N je sudé číslo.

Pokusme se nyní určit pravidlo pro určení řezů pro čísla L_x a P_x za pomocí jejich dvojkových rozvojů:

$$L_x = 1\ 0\ l_{N-2}\ l_{N-3} \ldots l_1\ 1$$

$$P_x = 1\ 1\ p_{N-2}\ p_{N-3} \ldots p_1\ 1$$

Odůvodnění výše uvedeného zápisu:

- Obě čísla jsou lichá, takže nejnižší bit 1 musí být nastavený na 1.
- Obě čísla patří do I_N, takže nejvyšší bit N musí být nastavený na 1.
- Všechna čísla menší než M_N mají bit $N-1$ nastavený na 0, protože střed intervalu M_N je nejmenší číslo, které má nastavený bit $N-1$.
- Opačně všechna čísla větší než M_N mají nastavený bit $N-1$ na 1.
- Hodnoty bitů l_i pro levé číslo a p_i pro pravé číslo nabývají hodnot 0 anebo 1 a jejich počet je $N-2$, protože všechna čísla intervalu I_N mají $N+1$ bitů a tři výše popsané bity jsou pevně stanovené.

Bitový rozvoj $d(x)$ získáme odečtením bitových rozvojů:

$$d(x) = P_x - M_N = 0\,0\,p_{N-2}\,p_{N-3}\ldots p_1\,1$$

Hodnotu L_x získáme odečtením bitového rozvoje $d(x)$ od bitového rozvoje hodnoty M_N. Zde je výhodné použít algoritmus odčítání čísel ve dvojkové soustavě, které využívají počítače, kdy úlohu odečítání převedou na úlohu sčítání:

- Zarovnání menšitele a menšence zleva nulami na stejný počet bitů.
- U menšitele nejprve algoritmus otočí jednotlivé bity (bitová negace, kdy se z každé hodnoty bitu 1stane 0 a naopak).
- K takto získanému číslu přičte jedničku.
- Výsledné číslo přičte k menšenci.
- U sčítání vždy dojde k přetečení nejvyššího bitu, který počítačový algoritmus vždy ignoruje a nahradí ho nulou.

Uvedený algoritmus je popsán například na stránce Wikipedie *Method of complements* v sekci *Binary method*[39].

Aplikací algoritmu odečítání získáme bitový rozvoj:

$$L_x = M_N - d(x) = 1\,0\,p^*_{N-2}\;p^*_{N-3}\ldots p^*_1\,1$$

Hodnoty bitů p_i^* představují opačné hodnoty k bitům (otočené bity) k hodnotám bitů p_i.

Odůvodnění:

- Otočením bitů menšitele získá nejnižší bit hodnotu 0.
- Přičtením jedničky získá nazpátek hodnotu 1. Zde je důležité si uvědomit, že při sčítání nemůže dojít k přetečení hodnoty do vyšších bitů.
- Bity 1 až $N-2$ u M_N jsou nulové, takže při sčítání prostě přenesou otočené hodnoty bitů $N-2$ až 1 z bitového rozvoje d_x. Opět je zde důležité, že při sčítání nemůže dojít k přetečení hodnoty do vyšších bitů.
- Nejvyšší dva bity získáme sečtením dvou nejvyšších bitů M_N tedy 11 a dvou nejvyšších bitů d_x, které byly předtím z hodnot dvou nul otočeny na dvě jedničky. Tedy dvojkově $11_2 + 11_2 = 110_2$.
- Závěr postupu sčítání je ignorování nejvyššího bitu ze sčítání $11_2 + 11_2 = 110_2$, čímž dostaneme úvodní sekvenci 10_2.

Z uvedených bitových rozvojů je možné snadno určit vzájemný vztah mezi hodnotami bitů l_i levého čísla a hodnotami bitů pravého čísla p_i:

$$P_x = 1\,1\,p_{N-2}\,p_{N-3}\ldots p_1\,1$$

$$L_x = 1\,0\,p^*_{N-2}\;p^*_{N-3}\ldots p^*_1\,1$$

Tento rozvoj nám umožňuje stanovit příslušnost čísel P_x a L_x prostým spočtením jejich bitů. Při stanovování příslušností čísel P_x a L_x do řezů $T_{N,j}$se vždy ignoruje nejvyšší bit. Stačí tedy určit vzájemný vztah mezi vnitřními bity p_i a jejich negacemi p_i^*. Jedná se o $N-2$ bitů, které jsou vždy navzájem otočené. Jejich sečtením ihned získáme následující užitečný vztah:

$$\sum_{i=1}^{N-2} p_i + \sum_{i=1}^{N-2} p_i^* = N-2$$

Je-li $P_x \in T_{N,k}$, pak z definice řezu nutně platí:

[39] https://en.wikipedia.org/wiki/Method_of_complements#Binary_method

$$k = 0 + 1 + \sum_{i=1}^{N-2} p_i + 1$$

A analogicky, je-li $L_x \in T_{N,r}$ nutně platí:

$$r = 0 + 0 + \sum_{i=1}^{N-2} p_i^* + 1$$

Oba výrazy pro indexy k a r jsou napsány tak, že jednotlivé sčítance kopírují bitový rozvoj čísel a pro názornost jsou v něm uvedeny i hodnoty 0 pro bity, které neovlivňují zařazení do řezu. První 0zastupuje v součtu bit N, který je povinně nastaven na 1 a v definici řezů $T_{N,j}$se vždy ignoruje.

Nyní je třeba vyjádřit výraz $\sum_{i=1}^{N-2} p_i^*$. Z výše uvedené užitečné rovnosti dostáváme:

$$\sum_{i=1}^{N-2} p_i^* = N - 2 - \sum_{i=1}^{N-2} p_i$$

$$\sum_{i=1}^{N-2} p_i^* = N - 2 - \sum_{i=1}^{N-2} p_i$$

Dosazením do rovnice pro r získáme:

$$r = 0 + 0 + (N - 2 - \sum_{i=1}^{N-2} p_i) + 1$$

Jednoduchou závěrečnou úpravou získáme vztah mezi k a r takto:

$$r = N - k + 1$$

$$k = N - r + 1$$

Příklad:

- Zvolme $x = 29$ z intervalu I_4
- Střed intervalu $M_4 = 16 + 8 = 24 = 11000_2$
- Pravá hodnota $P_{29} = 13 = 1\,\mathbf{1101}_2 \in T_{4,3}$ a tedy $k = 3$
- Levá hodnota $L_{29} = 19 = 1\,001\mathbf{1}_2 \in T_{4,2}$ a tedy $r = 2$
- Pokud bychom neznali hodnotu r a tedy i zařazení levé hodnoty x do řezu, mohli bychom použít nově odvozený vzorec $r = N - k + 1 = 4 - 3 + 1 = 2$

Definice symetrických řezů

Uvedeným postupem jsem našel obecný vztah pro zařazení symetrického protějšku libovolného čísla do správného řezu. Když tento postup platí pro jedno libovolné číslo, platí i pro všechna čísla libovolného řezu.

Jinými slovy, pokud například vezmeme pravou polovinu řezu $T_{15,11}$ (tedy všechna čísla větší než M_{15}) dostaneme levou polovinu řezu $T_{15,5}$ (neboť pro $k = 11$ dostaneme $r = 15 - 11 + 1 = 5$) a nutně i naopak překlopením levé poloviny řezu $T_{15,11}$ dostaneme pravou polovinu řezu $T_{15,5}$ (neboť pro $r = 11$ dostaneme $k = 15 - 11 + 1 = 5$).

Tento příklad lze zobecnit v tvrzení, že řezy $T_{N,i}$ a $T_{N,N-i+1}$ jsou navzájem symetrické kolem středu intervalu a jeden lze získat z druhého překlopením kolem osy. Nabízí se tedy myšlenka každou takovou dvojici řezů spojit do nového typu řezu:

$$TT_{N,i} = T_{N,i} \cup T_{N,N-i+1} \text{ pro } i = 1..N$$

Příklad:

- $TT_{6,1} = T_{6,1} \cup T_{6,6}$
- $TT_{6,2} = T_{6,2} \cup T_{6,5}$
- $TT_{6,3} = T_{6,3} \cup T_{6,4}$
- $TT_{6,4} = T_{6,4} \cup T_{6,3} = TT_{6,3}$
- $TT_{6,5} = T_{6,5} \cup T_{6,2} = TT_{6,2}$
- $TT_{6,6} = T_{6,6} \cup T_{6,1} = TT_{6,1}$

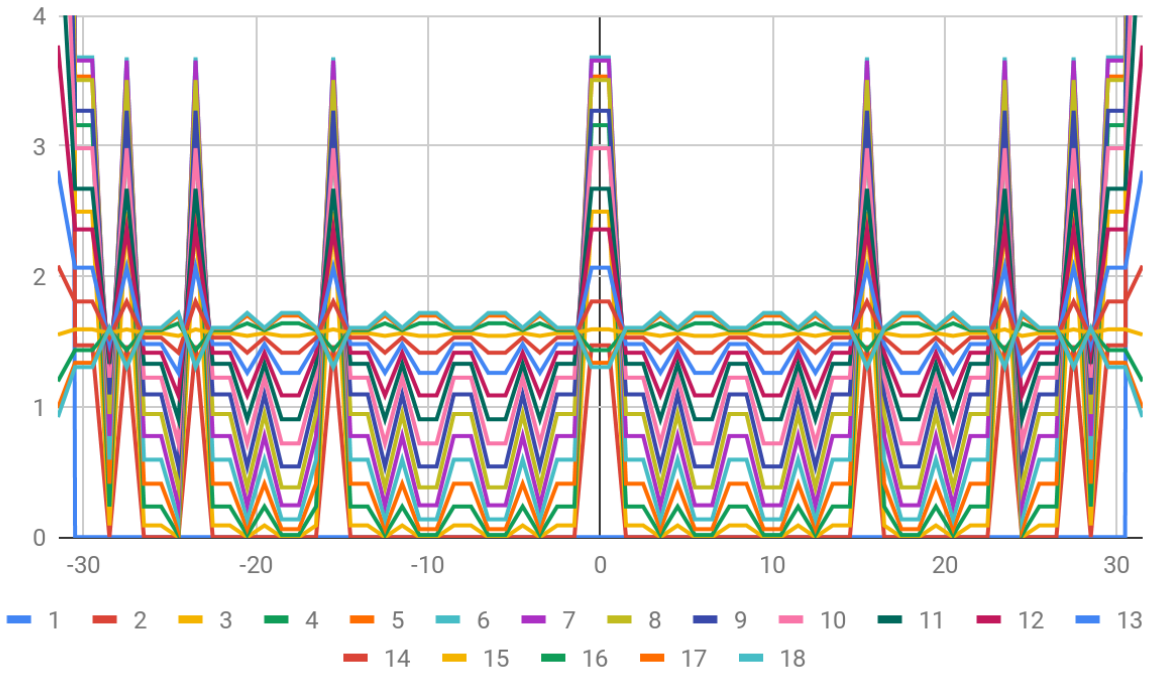
Je jasné, že pro intervaly I_N, kde N je sudé získáme $N/2$ nových intervalů. Pokud ale naopak bude N liché, získáme jeden prostřední interval, který nevznikne spojením dvou různých intervalů, což je případ intervalu $TT_{5,3}$ z následujícího příkladu:

- $TT_{5,1} = T_{5,1} \cup T_{5,5}$
- $TT_{5,2} = T_{5,2} \cup T_{5,4}$
- $TT_{5,3} = T_{5,3} \cup T_{5,3}$
- $TT_{5,4} = T_{5,4} \cup T_{5,2} = TT_{5,2}$
- $TT_{5,5} = T_{5,5} \cup T_{5,1} = TT_{5,1}$

Vzhledem k popsané symetrii budou nutně spojené intervaly $TT_{N,i}$ vyvážené s těžištěm ve středu intervalu M_N, neboť prostřední řez lichých intervalů je symetrický sám o sobě a u ostatních řezů vždy nevyváženost jednoho dílčího řezu přesně vykompenzuje nevyváženost druhého řezu.

Histogram vyvážených řezů

Výpočtem jsem ověřil, že všechny nově řezy mají vyvážené těžiště v přesném prostředku intervalu. Dále jsem pro kontrolu spočítal histogramy všech řezů $TT_{N,j}$. Histogramy původních řezů $T_{N,j}$ byly asymetrické, zatímco histogramy řezů $TT_{N,j}$ jsou již v souladu s očekáváním dokonale symetrické. Na následujícím obrázku jsou vidět histogramy všech řezů intervalu I_{35}.



Graf 29

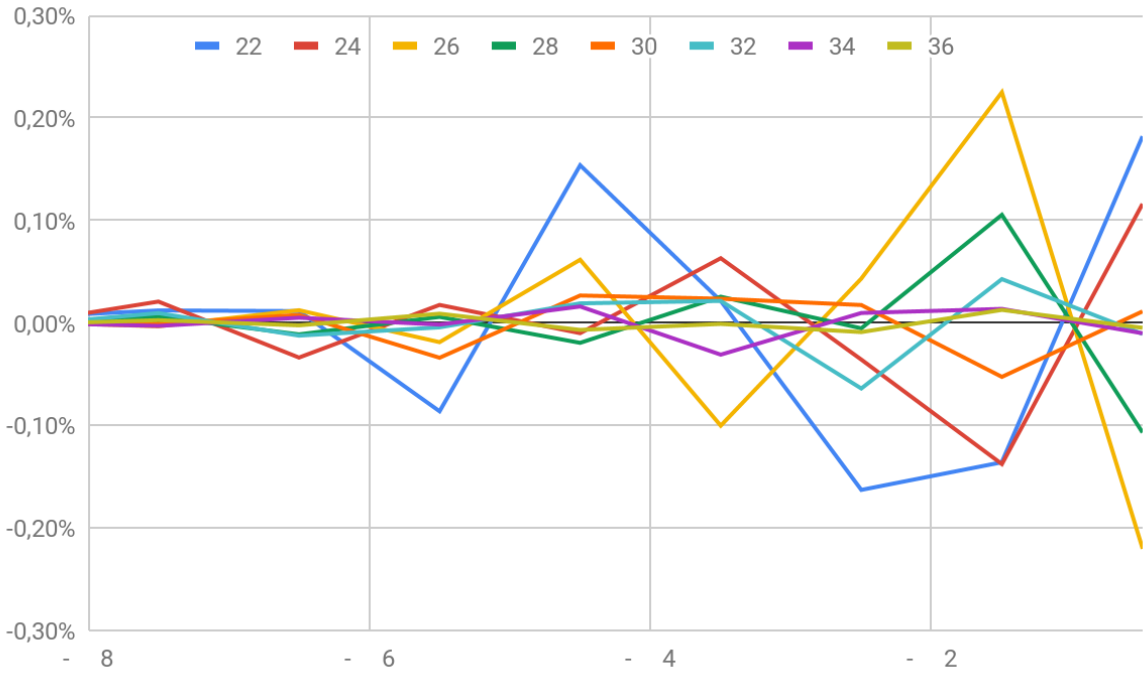
Dále se měřením ukázalo, že všechny liché intervaly jsou vyvážené i jiným způsobem. Pokud sečteme počet prvků všech lichých řezů, dostaneme stejný počet čísel jako je součet všech prvků ze všech sudých řezů. Toto chování pro sudé intervaly neplatí, což znázorňuje následující tabulka:

N	Součet délky sudých řezů	Součet délky lichých řezů
4	75,00%	25,00%
6	31,25%	68,75%
8	65,63%	34,38%
10	36,33%	63,67%
12	62,30%	37,70%
14	38,72%	61,28%
16	60,47%	39,53%
18	40,18%	59,82%
20	59,27%	40,73%
22	41,19%	58,81%
24	58,41%	41,59%
26	41,94%	58,06%
28	57,75%	42,25%
30	42,53%	57,47%
32	57,22%	42,78%
34	43,00%	57,00%
36	56,79%	43,21%

Měření na symetrických intervalech

Opět na takto definovaných intervalech můžeme zkoumat rozmístění prvočísel. Při porovnání výsledků z různých intervalů je potřeba opět vyřešit zarovnání různě dlouhých křivek. Logickým způsobem je zarovnání podle prostředního řezu u lichých intervalů anebo podle dvou prostředních (a spojených) řezů pro sudé intervaly I_N se sudým N. Tyto řezy jsou označené v grafech zarovnány na hodnotu nula nejvíce doprava. Záporné hodnoty na ose x pak vyjadřují vzdálenost k těmto středovým řezům. Typicky tedy platí, že směrem vpravo se vyskytují řezy s větším počtem prvků, přičemž u lichých intervalů je poslední středový řez jediný nespojený a nemá tedy nejvíce prvků.

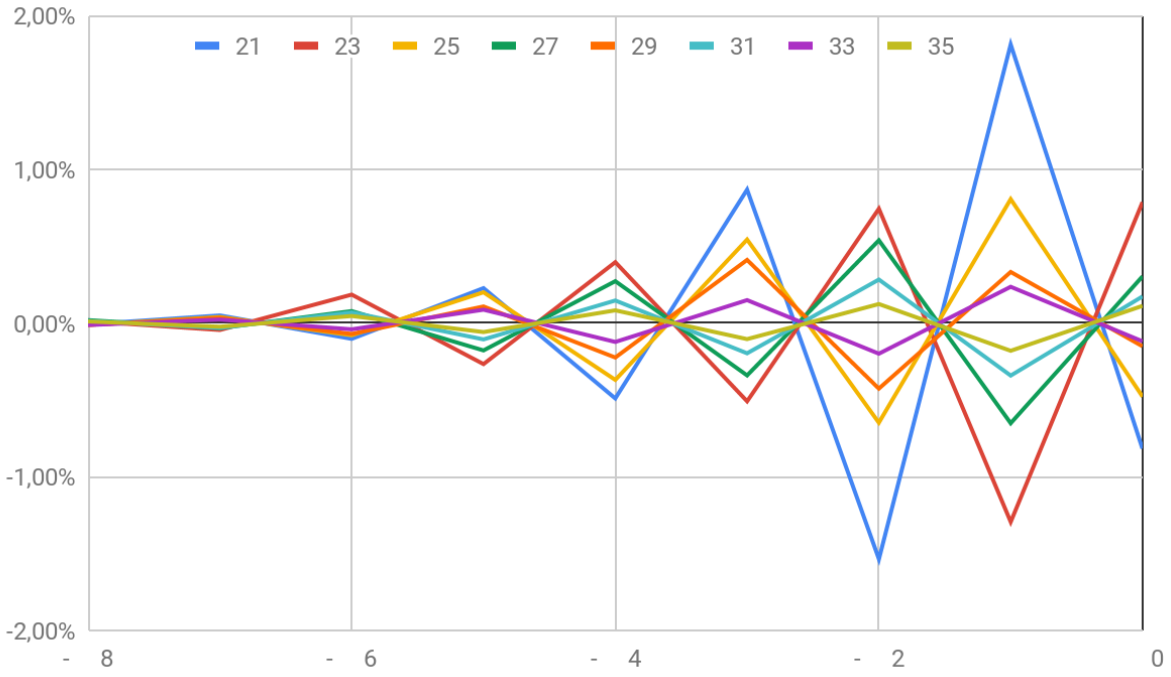
Opět budeme zkoumat zvlášť sudé a liché intervaly. Na následujícím grafu je výsledek analýzy pro sudé intervaly I_{22} až I_{36}.



Graf 30

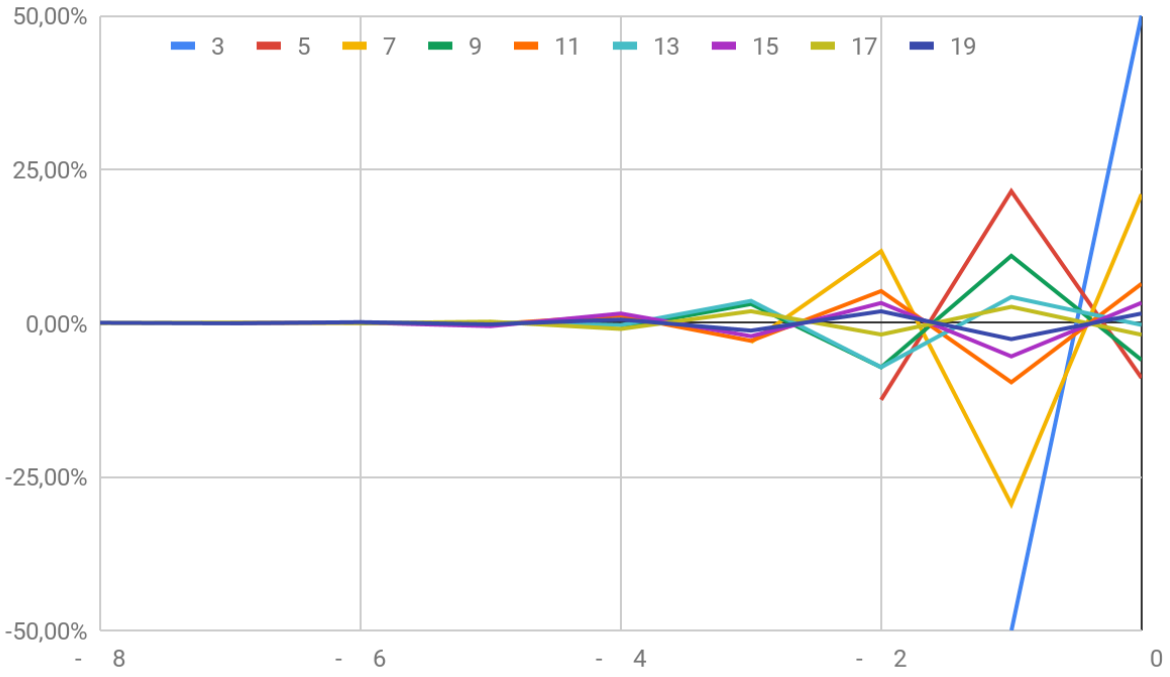
Spojené řezy pro sudé intervaly nevykazují žádné překvapivé chování ani žádný rozpoznatelný vzorec. Podobně jako u řezů u řezů $T_{N,j}$ jen odráží nepředvídatelné odchylky počtu prvočísel v jednotlivých řezech.

Na druhou stranu graf pro liché intervaly obsahuje podobně pravidelné pilovité chování, které se vyskytovalo i u řezů $T_{N,j}$. Řezy $TT_{N,j}$odstraňují nedokonalost řezů $T_{N,j}$ v podobě jejich asymetrie a posunutého těžiště, takže je lze pro zkoumání rovnoměrnosti prvočísel považovat za vhodnější. Přesto se překvapivé pilovité chování řezů $T_{N,j}$ lichých intervalů přeneslo i do řezů lichých intervalů řezů $TT_{N,j}$. Nedokonalost původní definice tedy nebyla jejich příčinou.



Graf 31

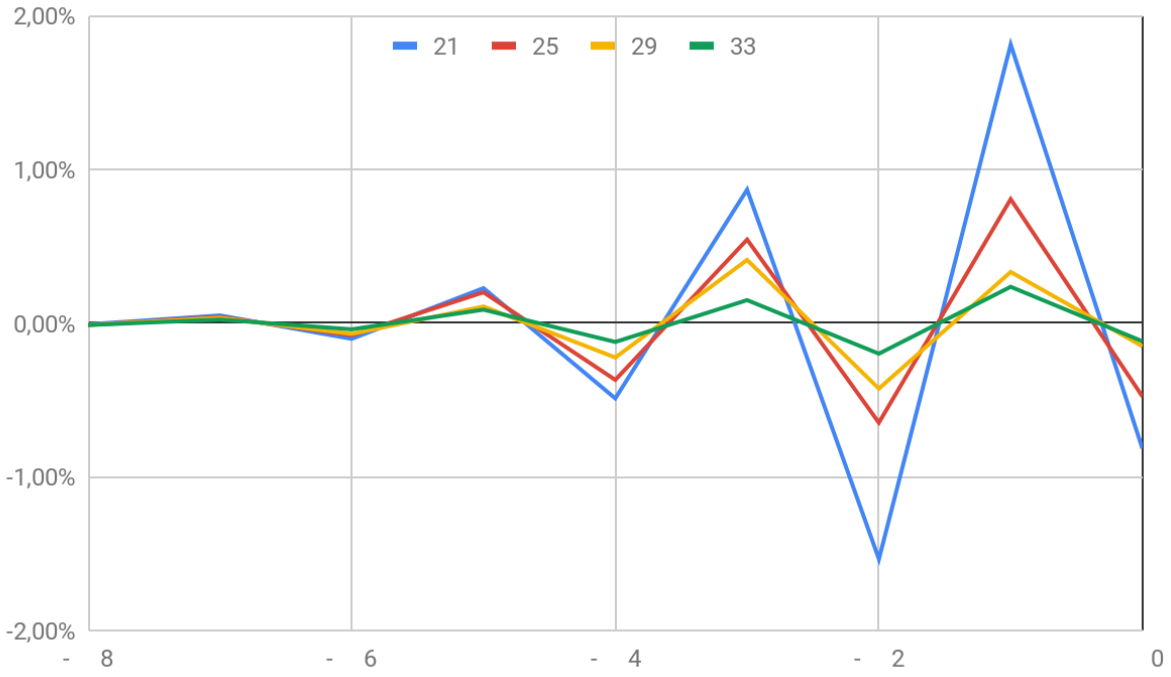
Překvapivé je rovněž chování na nízkých intervalech. Dosud jsem nikdy nevěnoval pozornost nízkým intervalům do přibližně $N = 20$, protože celkový počet prvočísel na těchto intervalech je tak nízký, že mi nedávalo smysl studovat rovnoměrnost jejich rozmístění. Každopádně je překvapivé, že záhadné pilovité chování se naprosto systematicky vyskytuje již od nejnižších intervalů.



Graf 32

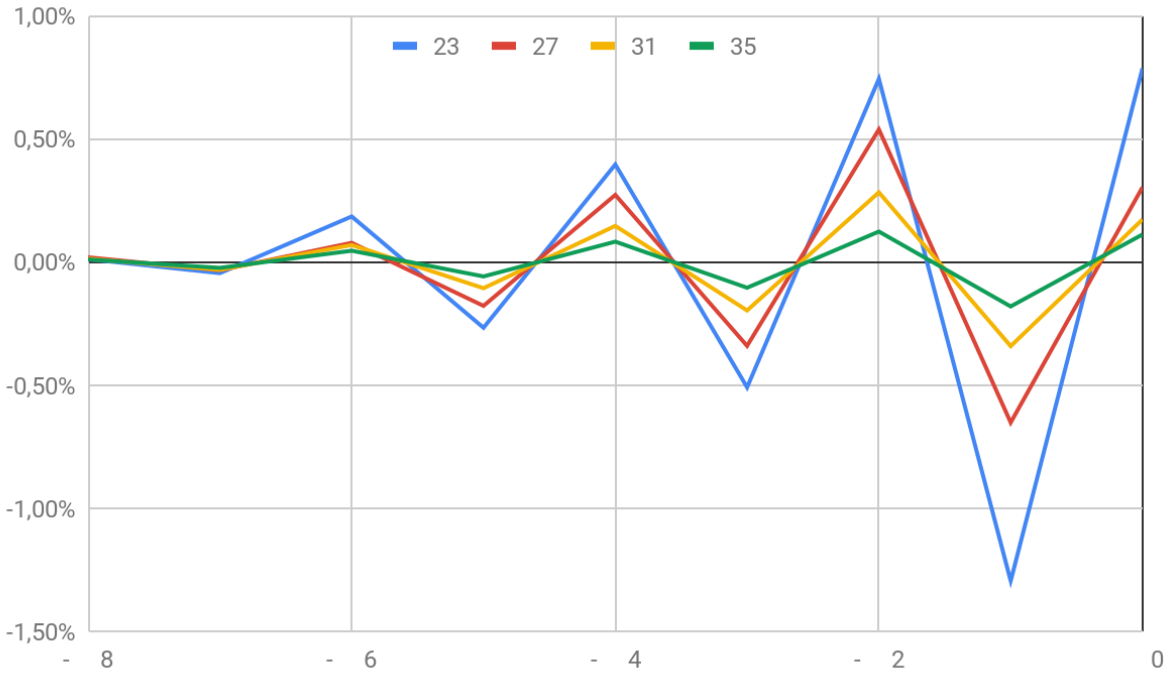
Cyklus 4 pro liché intervaly

Podobně lze pozorovat, že se zachoval i výše zmíněný cyklus 4, který je možné pro řezy $TT_{N,j}$ pozorovat na následujících dvou grafech. Prvním je graf řezů intervalů s počátkem v intervalu 21 a skokem 4.



Graf 33

A druhým je graf řezů intervalů s počátkem v intervalu 23 a skokem 4.



Graf 34

Všechny prostřední řezy intervalů $21, 27, 31, 35$, které jsou v grafu zarovnány na hodnotu 0 ukazují na nedostatek prvočísel oproti očekávání. Naopak prostřední řezy intervalů $23, 27, 31, 35$ rovněž zarovnané na hodnotu 0 vykazují přebytek prvočísel oproti očekávání.

Tento jev lze snadno vysvětlit jako důsledek pozorovaného pravidelného pilovitého chování, které je vynucené situací na nejnižším řezu $TT_{N,1}$.

Tento řez vznikne sloučením řezů $T_{N,1} \cup T_{N,N}$. Řez $T_{N,1}$ vždy obsahuje jen jedno nejnižší liché číslo intervalu, které má vedle bitu N (určující příslušnost do intervalu I_N) nastevený právě jeden další bit. Řez $T_{N,N}$ naopak obsahuje vždy jen jedno číslo, a to nejvyšší liché číslo intervalu I_N, neboť má vedle bitu N nastavny i všechny ostatní bity $0..N-1$. Žádný lichý interval v rozsahu $N = 1..35$ neobsahuje prvočíslo. Tedy pro všechna spočtená lichá N jsou hodnoty $q_{N,0} = 0$ a $q_{N,N} = 0$. Na druhou stranu nastavené očekávání $Z_{N,1}$ a $Z_{N,N}$ odráží jen potenciál pro výskyt prvočísla podílem počtu lichých číslel v řezu oproti počtu všech lichých čísel intervalu a je to tedy kladné číslo. Při jejich složení do symetrického řezu se toto očekávání zdvojnásobí. Je zřejmé, že na takto složeném řezu je vždy nedostatek prvočísel oproti jejich očekávání. A pozorované pravidelné střídání přebytku a nedostatku prvočísel v každém sousedním řezu se postupně přenáší až na konec do prostředního řezu. Případný nedostatek nebo přebytek prvočísel je dán jen hodnotou N.

Tato úvaha uspokojivě vysvětluje, že pokud začneme s intervalem I_3, tak každý čtvrtý vyšší interval má ve středovém řezu přebytek prvočísel, a naopak při startu v I_5 má každý čtvrtý následující interval prostřední řez s nedostatkem prvočísel. Ale zde je nutné si uvědomit, že úvaha se opírá o pravidelné střídání nedostatku a přebytku, který u lichých řezů nebyl uspokojivě vysvětlený.

Dále je možné, že na vyšších intervalech, které nebyly analyzovány, se může vyskytnout prvočíslo ve tvaru $2^N + 1$. Prvočísla ve tvaru tvaru $2^N - 1$ lze snadno vyloučit. Taková prvočísla se nazývají Mersennovými prvočísly[40] a dosud je jich známo 51 (konkrétně jde o exponenty *2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091, 756839, 859433, 1257787, 1398269, 2976221, 3021377, 6972593, 13466917, 20996011, 24036583, 25964951, 30402457, 32582657, 37156667, 42643801, 43112609*).

Konkrétně exponenty $2, 3, 5, 7, 13, 17, 19, 31$ vedou na Mersennova prvočísla M_2 až M_{31}, které byly analyzovány. Z nich ale jen to první je součástí lichého intervalu:

$$M_2 = 3 = 11_2 \in I_1$$

A interval $I_1 = \{2,3\}$ obsahuje jen dvě čísla, nemá smysl jej dále řezat a tuto krajní situaci jakkoliv dále zkoumat. Všechna ostatní Mersennova prvočísla patří do sudých intervalů. Například:

$$M_3 = 7 = 111_2 \in I_2$$

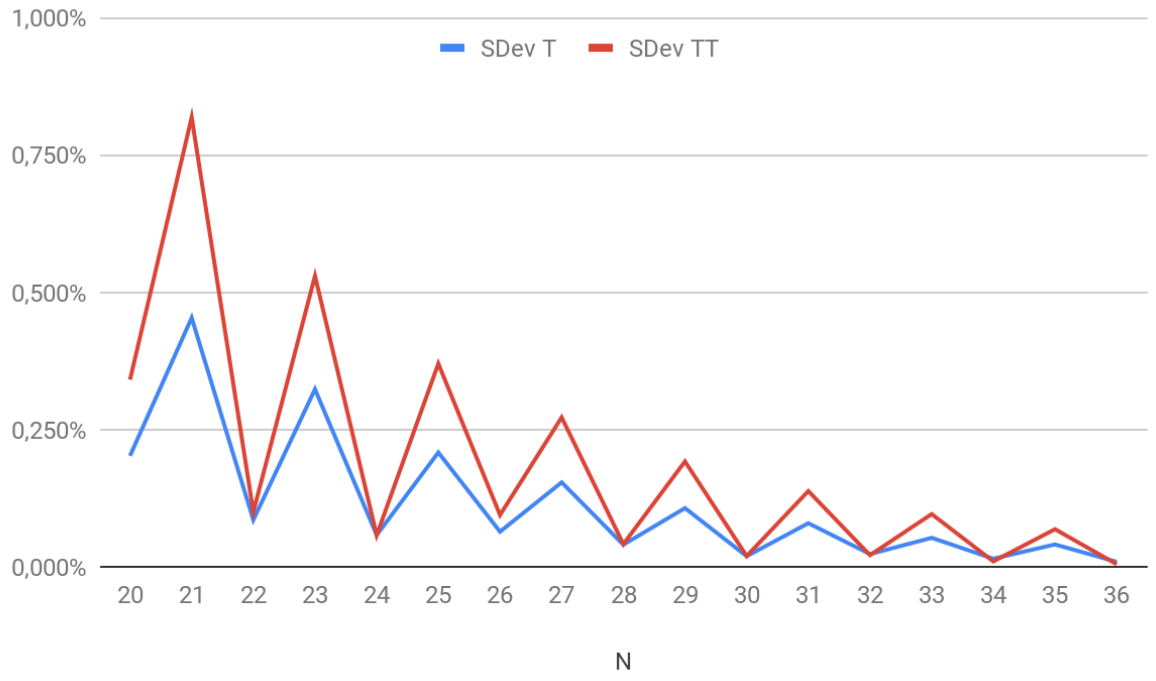
A toto platí i obecně. Existuje důkaz, že pokud je k složené, pak je složené i $2^k - 1$. Zbývá tedy možnost, že bude existovat prvočíslo ve tvaru $2^N + 1$, o kterých se mi ale nepodařilo nic zjistit.

Každopádně případný výskyt prvočísla v tomto tvaru by otočil typický nedostatek prvočísel v řezu $T_{N,1}$ na přebytek a porušil by jinak typické pilovité chování. Bylo by pak zajímavé provést analýzu celého intervalu a podívat se na sousední řezy, ale to je úloha mimo možnosti jednoho domácího počítače.

[40] https://en.wikipedia.org/wiki/Mersenne_prime

Směrodatná odchylka

Podobně jako u bitových řezů $I_{N,j}$ lze zkoumat směrodatnou odchylku pro řezy $T_{N,j}$ a symetrické řezy $TT_{N,j}$. Výsledek je zobrazen na následujícím obrázku:



Graf 35

Modrá křivka se vztahuje k bitovým řezům $T_{N,j}$.

Ihned je možné pozorovat, že s postupně zvětšujícími se intervaly dochází k postupnému poklesu směrodatné odchylky a je tedy možné opět formulovat hypotézu, že zvolená veličina $q_{N,j}$ je dobrou aproximací očekávaného počtu prvočísel, což lze matematicky formulovat jako očekávání, že

$$\lim_{N\to\infty} max_{0<j<N}(F_{N,j}) = 0$$

Z grafu je dále patrné, že pro liché intervaly popsané pilovité chování způsobují vyšší směrodatnou odchylku a naopak na sudých intervalech námi definované očekávání $q_{N,j}$ funguje lépe a má systematicky nižší směrodatnou odchylku. Spojené řezy $TT_{N,j}$ vykazují obdobnou výkonnost na sudých intervalech, ale na lichých intervalech je směrodatná odchylka výrazně vyšší.

Závěrečná úvaha nad pilovitým chováním

Pravidelné pilovité chování způsobuje, že při skládání dvou řezů vždy dochází k sečtení nedostatků v obou dílčích řezech, nebo naopak k sečtení přebytku v obou dílčích řezech. Na pozorovaných datech nikdy nenastala situace, kdy by při skládání došlo ke kompenzaci přebytku v jednom řezu s nedostatkem v druhém řezu. Vhodnější definice symetrických řezů tedy naopak pozorovanou anomálii systematicky prohlubuje. Toto pozorování považuji za nejzajímavější část této práce.

Domnívám se tedy, že je vhodné ještě jednou bez použití vzorců vyložit, co vlastně pilovité chování znamená. Libovolný interval I_N, kde N je liché, vykazuje pravidelné střídání nedostatku a přebytku v sousedních řezech $T_{N,j}$, ale i $TT_{N,j}$.

Konkrétní příklad pro interval I_{21}:

j	osa *x* v grafech (zarovnání na střed)	skutečný počet prvočísel v řezu	očekávaný počet prvočísel v řezu	skutečný počet prvočísel v řezu [%]	očekávaný počet prvočísel v řezu [%]	rozdíl očekávání - skutečnost	rozdíl očekávání - skutečnost [%]
1	- 10	0	0,27	0,000%	0,000%	-0,27	0,000%
2	- 9	7	5,35	0,005%	0,004%	1,65	0,001%
3	- 8	39	50,86	0,028%	0,036%	-11,86	-0,008%
4	- 7	374	305,14	0,267%	0,217%	68,86	0,049%
5	- 6	1 152	1 296,86	0,821%	0,924%	-144,86	-0,103%
6	- 5	4 468	4 149,95	3,184%	2,957%	318,05	0,227%
7	- 4	9 686	10 374,88	6,902%	7,393%	-688,88	-0,491%
8	- 3	21 967	20 749,75	15,653%	14,786%	1 217,25	0,867%
9	- 2	31 562	33 718,35	22,490%	24,027%	-2 156,35	-1,537%
10	- 1	47 500	44 957,80	33,847%	32,036%	2 542,20	1,812%
11	0	23 581	24 726,79	16,803%	17,620%	-1 145,79	-0,816%

Poslední řádek reprezentuje středový řez, který nevznikl spojením dvou řezů a v grafech je zarovnán na hodnotu 0. Délky řezů s rostoucím indexem j rostou, což se odráží v rostoucím očekávaném počtu prvočísel.

Grafy jsou zobrazují poslední sloupec, ve kterém je vidět pravidelné střídání znaménka (tedy pily). Záporné znaménko označuje nedostatek prvočísel oproti očekávání. Rozdíl procent je vhodné zobrazovat v grafech, protože umožňuje srovnání různých intervalů a různých řezů, ale pro bližší pochopení je nutné upustit od procent a analyzovat přímo počty prvočísel (očekávané i skutečné). Z tabulky je vidět, že například řez $TT_{21,7}$ obsahuje nedostatek prvočísel $(688{,}880)$, a oba sousední řezy obsahují nadbytek prvočísel. Konkrétně řez $TT_{21,6}$ o $318{,}5$ a řez $TT_{21,8}$ o $1\,217{,}25$. Řezy jsou různě dlouhé, takže je přirozené, že na delším řezu je absolutní rozdíl větší. To, co je skutečně zajímavé, je pravidelné střídání nedostatku a nadbytku.

Řez $TT_{21,7}$ obsahuje čísla, které mají vedle bitu 21 nastaveno dalších 7 anebo 15 bitů, neboť vznikl sloučením řezů $T_{21,7}$ a řezu $T_{21,21-7+1} = T_{21,15}$. Sousední řez $TT_{21,6}$ obsahuje čísla, které mají vedle bitu 21 nastaveno dalších 6 anebo 16 bitů, neboť vznikl sloučením řezů $T_{21,6}$ a řezu $T_{21,21-6+1} = T_{21,16}$. A nakonec sousední řez $TT_{21,8}$ obsahuje čísla, které mají vedle bitu 21 nastaveno dalších 8 anebo 14 bitů, neboť vznikl sloučením řezů $T_{21,8}$ a řezu $T_{21,21-8+1} = T_{21,14}$.

Pokud sečteme nedostatky prvočísel ze všech lichých řezů a přebytky ze všech sudých řezů intervalu I_{21} dostaneme tuto tabulku:

Interval 21	skutečný počet prvočísel v řezech	očekávaný počet prvočísel v řezech	rozdíl
Liché	66 020	70 168,00	-4 148
Sudé	74 316	70 168,00	4 148

Zobecněním tohoto příkladu dospíváme k pozorování, že na lichých intervalech prvočísla více preferují sudý počet vnitřních bitů, který vždy vede na nadbytek oproti očekávání a naopak řezy s lichým počtem vnitřních bitů vedou na nedostatek.

Každý interval můžeme rozdělit na dva super řezy tak, že do sudého super řezu zahrneme všechny řezy se sudým indexem j a naopak do lichého super řezu zahrneme všechny řezy s lichým indexem j. Níže uvedená tabulka zobrazuje nedostatek nebo přebytek prvočísel v těchto dvou super řezech oproti jejich očekávání (daném poměrem součtu jejich délky v celém intervalu). Z tabulky je ihned vidět, že ve všech lichých super řezech je nedostatek prvočísel.

	sudý super řez	lichý super řezy
N		
1		- 1,0
3	1,0	- 1,0
5	1,5	- 1,5
7	7,5	- 7,5
9	10,5	- 10,5
11	32,5	- 32,5
13	69,0	- 69,0
15	246,0	- 246,0
17	515,5	- 515,5
19	1 603,5	- 1 603,5
21	4 148,0	- 4 148,0
23	10 895,0	- 10 895,0
25	30 147,0	- 30 147,0
27	85 012,0	- 85 012,0
29	232 904,0	- 232 904,0
31	672 134,0	- 672 134,0
33	1 842 947,5	- 1 842 947,5
35	5 204 767,0	- 5 204 767,0

V případě sudých intervalů takové pozorování neplatí. Navíc velikost odchylky oproti očekávání dosahuje řádově menších hodnot.

N	sudý super řez	lichý super řez
2		0,0
4	- 0,8	0,8
6	- 1,1	1,1
8	- 5,2	5,2
10	- 1,8	1,8
12	- 6,1	6,1
14	12,8	- 12,8
16	- 51,4	51,4
18	- 12,9	12,9
20	- 682,0	682,0
22	- 509,6	509,6
24	348,6	- 348,6
26	3 705,0	- 3 705,0
28	- 19 641,6	19 641,6
30	- 33 320,9	33 320,9
32	- 134 222,1	134 222,1
34	- 157 132,6	157 132,6
36	- 660 370,5	660 370,5

Při rozkladu čísla na součet mocnin dvou se uvedené pozorování projeví tak, že prvočísla na lichých intervalech “preferují” ve svém dvojkovém rozvoji lichý počet sčítanců (sudý počet vnitřních bitů plus jeden nejvyšší bit definující interval). Toto si nedokážu nijak vysvětlit. Je možné, že při zkoumání vyšších intervalů by tato anomálie vymizela nebo se postupně nedostatek obrátil v nadbytek. Každopádně by si toto téma zasloužilo další analýzu, případně vysvětlení od někoho více zkušeného v oblasti prvočísel.

Možnosti zobecnění

Nabízí se myšlenka pro další zobecnění uvedených hypotéz pro jiné typy řezů. Použité řezy předpokládají pevné nastavení dvou bitů. První definuje interval a druhý definuje jeho řez. Pokud bychom otočili nastavení bitu definující řez a předpokládali jeho nastavení vždy na 0, došli bychom k velmi obdobnému výsledku: z jednotlivých částí řezů by došlo k výběru spodních polovin místo horních polovin, řezy by byly nevyvážené směrem k nižším hodnotám, měřené odchylky by tedy měly opačné znaménko, avšak celkový výsledek by zůstal platný.

Další možností by bylo pevné nastavené 3 bitů. První by definoval interval a druhé dva bity by definovaly jejich řez, který by v takovém případě měl celkovou délku odpovídající čtvrtině délky celého intervalu. I bez provedeného měření lze předpokládat, že by bylo možné postupovat

analogicky: provést měření a formulovat podobnou hypotézu, tentokrát o konvergenci ke čtvrtině celkového počtu prvočísel celého intervalu.

Dalším zobecněním by bylo postupné nastavení většího počtu bitů a tvorba řezů, které z intervalu vybírají stále menší součet délky. Konkrétně při pevném nastavení j bitů by došlo k výběru $1/2^{j-1}$násobku celkové délky intervalu. Opět lze předpokládat, že by bylo možné formulovat a měřením podložit analogické hypotézy. Přirozeně s výběrem menší části intervalu bychom logicky museli očekávat větší fluktuace, takže výpočet by bylo vhodné provést na delších intervalech, ale na celkové konvergenci by se velmi pravděpodobně nic nezměnilo.

Dosud diskutované intervaly a jejich řezy vychází z dvojkové soustavy, což umožňuje snadnou implementaci výpočtů v počítačovém programu. Samozřejmě je možné definovat zcela jiné intervaly a jejich řezy a zkoumat na nich rozložení prvočísel. Lze předpokládat, že pokud by taková definice splňovala požadavky na jejich ‘nestrannost’, bylo by možné dospět k podobným výsledkům ohledně rovnoměrnosti rozmístění prvočísel. Všechny tyto úvahy však nebyly podloženy bližším zkoumáním, měřením nebo teoretickou úvahou a představují tak spíše oblasti možné další práce.

Závěry a diskuse

Oba úseky práce představují podle mého názoru zajímavá témata. Jak je již uvedeno v závěru k první části práce, nelze vyloučit, že hledaná optimalizace existuje a jen se ji s mou úrovní znalostí nepodařilo nalézt. Každopádně úloha byla mnohem složitější než jsem na začátku předpokládal. Přesto úsilí a čas strávený nad tímto úsekem nepovažuji za ztracený.

Druhá část práce představuje krásné propojení matematiky a programování. Je dokonce možné, že myšlenka bitových řezů nebyla dosud zkoumána. Pokusy o vyhledání článků na podobné téma na internetu skončily neúspěchem. Zkoumání bitových řezů $I_{N,j}$ v mých očích představuje úspěšný postup: formulace nápadu, první výpočty, anomálie, formulace hypotézy jejího původu, úspěšná korekce anomálie završená ověřením pomocí výpočtu. Souhrn těchto kroků považuji za tyto kroky považuji za dotažený celek.

Na druhou stranu bitové řezy $T_{N,j}$ vedly na popsané pilovité grafy, které se mi nepodařilo vysvětlit a tento úsek práce zůstává nedotaženým. Tak jako první část čeká na svého mistra programátora, tak druhá část čeká na rozlousknutí nějakým matematikem působícím v oboru teorie čísel nazývaným anglicky *partition theory*[41], do jejíž oblasti studium pil asi nejvíce spadá.

Prvočísla představují velmi intenzivně zkoumané odvětví matematiky. Pokusit se o nový pohled je tedy odvážné a cenné samo o sobě.

[41] https://en.wikipedia.org/wiki/Partition_(number_theory)

Přílohy

Poznámky k provedeným výpočtům

Výpočetně nejnáročnější část bylo generování prvočísel intervalu I_{36}. Generování prvočísel pro intervaly do I_{32} až I_{33} proběhlo pomocí triviálního dělení předpočítanými prvočísly. Pro vyšší intervaly byly nutné optimalizace:

1. Testování pouze čísel, které nejsou násobkem 2,3,5,7,11,13,17,19.
2. Pro test prvočíselnosti byl použit BPSW test, který byl ale implementovaný za pomoci ne zcela optimální implementace založené na GNU MP[42] knihovně pro výpočty s velkými čísly.

Přesto trval výpočet pro I_{36} více než 24 hodin.

Další poměrně náročnou úlohou byl výpočet hodnot $Li_{N,j}$. Zejména pro jemná dělení velkých intervalů bylo potřeba spočítat velké množství hodnot funkce Li. Při jejich agregaci se projevila numerická chyba daná sčítáním obrovského množství hodnot s přesností datového typu *long double*. Tuto chybu je možné pozorovat např. u hodnot $Li_{36,1} = 1\,356\,053\,142{,}46$ a $Li_{36,2} = 1\,356\,053\,143{,}62$. Spočítaná hodnota $Li_{36,1}$ teoreticky musí být nutně větší než hodnota $Li_{36,2}$(viz kapitola Korekce pomocí funkce $li(x)$), ale vlivem numerické chyby výpočtu tomu tak není. Naštěstí je numerická chyba vzhledem pro naše použití velmi malá bez dopadu na celkový výpočet.

[42] https://gmplib.org/

Zdrojové kódy, git, licence

Zdrojové kódy jsou k dispozici na cloudovém úložišti GitHub [43]na následující adrese:

https://github.com/MatejZeman02/BPSW-primes-and-bit-slices

Veškeré převzaté kódy nemají žádné přísné licenční omezení, jedná se o licence pro svobodný software:

- BSD[44] - jedna z nejsvobodnějších licencí
- některou verzi licence GNU GPL[45]

Nestudoval jsem detailně důsledky využívání zdrojových kódů různých licencí, předpokládám však, že se jedná o běžnou praxi a že jsem žádné licenční ujednání neporušil, především zveřejněním mnou vytvořeného kódu na výše uvedené adrese.

Následující tabulka uvádí seznam souborů projektu se stručným popisem jejich funkce. Každá třída je typicky uložena ve dvou souborech. jeden je hlavičkový soubor s příponou *hpp* a druhý soubor obsahuje vlastní implementaci třídy a má příponu *cpp*. Tyto soubory jsou v tabulce reprezentovány jen jedním řádkem. Program adresuje více někdy nesouvisejících oblastí, proto obsahuje několik metod s názvem končícím slovem Test. Každá se zaměřuje na jednu konkrétní oblast a program main typicky spouští právě jednu takovou zvolenou funkci.

BitRange	Třída umožňující generovat čísla, které mají nastavené konkrétní bity
BitRangeTest	
BitStatistics	Třída umožňující výpočty bitových řezů
BitStatisticsTest	Generování prvočísel nad 2^{64}.
BPSWTest	Třída, která implementuje BPSW test, LS a MR test. Z velké části jde o převzatý kód.
BPSWTestTest	
GeneratorFunction	Virtuální třída jejíž potomek představuje vstup do Eratosthenova síta zajišťující výpočet některé z testovaných metod.
GeneratorFunctionBitStatistics	
GeneratorFunctionPrime	Potomek třídy GeneratorFunction, společný předek všech tříd pro testy prvočíselnosti.
GeneratorFunctionPrimeAPRCL	Třída pro Eratosthenovo síto zajišťující deterministický APRLCL test.
GeneratorFunctionPrimeBPSW	Třída pro Eratosthenovo síto zajišťující BPSW test.
GeneratorFunctionPrimeBPSW128	Nedokončený pokus pro implementaci BPSW testu na 128 bitovém datovém typu.

[43] https://cs.wikipedia.org/wiki/GitHub
[44] https://cs.wikipedia.org/wiki/BSD_licence
[45] https://cs.wikipedia.org/wiki/GNU_General_Public_License

GeneratorFunctionPrimeBPSW2	Třída pro Eratosthenovo síto zajišťující BPSW test z knihovny APRCL.
GeneratorFunctionPrimeCreator	Třída umožňující vygenerovat prvočísla v daném rozsahu. Využívá BPSW test a Eratosthenovo síto.
GeneratorFunctionPrimeLS	Třída pro Eratosthenovo síto zajišťující Lucas-Selfridgeův test.
GeneratorFunctionPrimeLS_D	Třída pro Eratosthenovo síto umožňující měření koeficientu D u Lucas-Selfridgeova testu.
GeneratorFunctionPrimeLS_DHistogram	Třída pro Eratosthenovo síto umožňující výpočet histogramů koeficientu D u Lucas-Selfridgeova testu.
GeneratorFunctionPrimeMR	Třída pro Eratosthenovo síto zajišťující Miller-Rabinův test.
GeneratorFunctionPrimeMR128	Nedokončený pokus pro implementaci Miller-Rabinova testu na 128 bitovém datovém typu.
int128Test	tester 128 bitového datového typu
IsPrime	Primitovní testy prvočíslenosti pomocí dělení a Eratosthenova síta. Velmi starý kód
jacobi_sum	Převzatý kód
Li	Funkce pro výpočet hodnoty funkce Li
main	Funkce main, která je spuštěna první po startu programu. Slouží jako rozcestník pro ostatní metody.
Makefile	Makefile zajišťuje kompilaci a sestavení programu z příkazové řádky napsáním příkazu make
mpz_aprcl	Převzatý kód pro deterministický APRCL test
PrimesRange	Třída pro výpočet prvočísel pomocí Eratosthenova síta, která uloží do pole o 32 nebo 64 bitech. Pro opakované použití se spočtené pole umí uložit a načíst z a do souboru.
PrimesRangeService	Třída, která spravuje více polí s vypočítanými prvočísly tak, aby zbytečně žádné pole nebylo v paměti počítače vícekrát.
PrimesRangeServiceTest	
SieveGenerator	Generátor čísel dle Eratosthenova síta, který pro každé vygenerované číslo dokáže volat různé testy.
SieveGeneratorTest	

SliceTBalance	Třída pro generování bitových řezů T a TT a měření jejich těžiště a histogramů.
SliceTBalanceTest	
Split	Pomocná funkce split
UInt128tompz	Pomocná funkce na převod 128-bitového čísla do čísla mpz_t z knihovny GNU MP.
VectorToFile	Pomocná funkce pro uložení pole typu vector do souboru

Což je celkově přes 13 800 řádků kódu (včetně převzatých algoritmů).

Odkazy

Tento dokument byl vytvořen pomocí Google Doc a následně převeden do jiných formátů. Původní zdrojový dokument je možné nalézt online na této adrese:

https://docs.google.com/document/d/191X1eMjp72DpxHHzX8mmrTakhvO9K6nQU0wAJulR4cw/edit?usp=sharing

Převážná většina grafů a tabulek byla vytvořena v Google Table, kterou je možné nalézt online pod tímto odkazem:

https://docs.google.com/spreadsheets/d/1YZKVsGMMEgvizjicBX59jQYdXLOTNTPxQfWsvlOgZO4/edit?usp=sharing

Doplňující výpočty zaměřené na nevyváženost řezů $T_{N,j}$ je možné nalézt pod tímto odkazem:

https://docs.google.com/spreadsheets/d/1GyoSyiu4HjIPPDOo9XDm0DqVPG7EgTvbP-n8RLWHHd4/edit?usp=sharing

Zdrojové kódy je možné nalézt v cloudovém úložišti GitHub pod tímto odkazem:

https://github.com/MatejZeman02/BPSW-primes-and-bit-slices

Seznam zdrojů a literatury

Nejdůležitějším zdrojem byla Wikipedie v české a anglické verzi. Uvědomuji si, že obsah Wikipedie může kdokoliv měnit a informace na ní obsažené nemusí být vždy korektní. Na druhou stranu jsem přesvědčený, že pokud bych se pokoušel vždy vycházet z původních zdrojů, tuto práci bych nikdy nenapsal. Osobně hodnotím přínos Wikipedie pro mou práci jako zásadní a jednoznačně převažující nad riziky.

Samozřejmě mě i po celou dobu mé práce intenzivně vedl můj odborný konzultant, který mi trpělivě vysvětloval všechny matematické problémy, které převyšovaly mou úroveň.

Zásadní byly rovněž stránky gmplib.org věnované knihovně GNU MP.

Pro otázky spojené s programováním jsem vedle běžných internetových zdrojů (StackOverflow.com, GeeksForGeeks.com, git-scm.com, Microsoft.com, gcc.gnu.org ...) využíval klasickou tištěnou knihu:

VIRIUS, Miroslav. Programování v C++: od základů k profesionálnímu použití. Praha: Grada Publishing, 2018. ISBN 978-80-271-0502-1

Poznámky pod čarou

1. https://gcc.gnu.org/onlinedocs/gcc/__005f_005fint128.html
2. https://en.wikipedia.org/wiki/Baillie%E2%80%93PSW_primality_test
3. https://faculty.lynchburg.edu/~nicely/misc/bpsw.html
4. http://mathworld.wolfram.com/Baillie-PSWPrimalityTest.html
5. https://gmplib.org/
6. https://en.wikipedia.org/wiki/GNU_Compiler_Collection
7. https://en.wikipedia.org/wiki/Ubuntu
8. https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux
9. https://en.cppreference.com/w/cpp/filesystem
10. https://en.wikipedia.org/wiki/MinGW
11. https://code.visualstudio.com/
12. https://gmplib.org/manual/Integer-Exponentiation
13. https://cs.wikipedia.org/wiki/Eratosthenovo_s%C3%ADto
14. https://cs.wikipedia.org/wiki/Bez%C4%8Dtvercov%C3%A9_cel%C3%A9_%C4%8D%C3%ADslo
15. https://en.wikipedia.org/wiki/Adleman%E2%80%93Pomerance%E2%80%93Rumely_primality_test
16. https://sourceforge.net/projects/mpzaprcl/
17. https://cs.wikipedia.org/wiki/GNU_Lesser_General_Public_License
18. https://faculty.lynchburg.edu/~nicely/misc/bpsw.html
19. https://cs.wikipedia.org/wiki/Miller%C5%AFv%E2%80%93Rabin%C5%AFv_test_prvo%C4%8D%C3%ADselnosti
20. https://cs.wikipedia.org/wiki/Jacobiho_symbol
21. https://cs.wikipedia.org/wiki/Kvadratick%C3%BD_zbytek

22. https://www.quora.com/Is-there-a-relation-between-prime-numbers-and-the-sum-of-squares-If-a-number-can-be-represented-as-the-sum-of-squares-does-it-comes-closer-to-being-a-prime
23. https://en.wikipedia.org/wiki/Dirichlet%27s_theorem_on_arithmetic_progressions
24. https://www.youtube.com/watch?v=EK32jo7i5LQ
25. https://en.wikipedia.org/wiki/Prime_number_theorem#Prime-counting_function_in_terms_of_the_logarithmic_integral
26. https://cs.wikipedia.org/wiki/Riemannova_hypot%C3%A9za
27. https://en.wikipedia.org/wiki/Siegel%E2%80%93Walfisz_theorem
28. https://mathworld.wolfram.com/ChebyshevBias.html
29. https://en.wikipedia.org/wiki/P%C3%B3lya_conjecture
30. https://docs.google.com/spreadsheets/d/1YZKVsGMMEgvizjicBX59jQYdXLOTNTPxQfWsvlOgZO4/edit?usp=sharing
31. https://cs.wikipedia.org/wiki/Prvo%C4%8D%C3%ADseln%C3%A1_v%C4%9Bta
32. https://en.wikipedia.org/wiki/Prime_gap
33. https://en.wikipedia.org/wiki/Logarithmic_integral_function
34. https://en.wikipedia.org/wiki/Prime_number_theorem#Table_of_%CF%80(x),_x_/_log_x,_and_li(x)
35. https://en.wikipedia.org/wiki/Logarithmic_integral_function#Series_representation
36. https://en.wikipedia.org/wiki/Standard_deviation
37. http://math.uchicago.edu/~may/REU2012/REUPapers/LiAng.pdf
38. https://cs.wikipedia.org/wiki/Kombina%C4%8Dn%C3%AD_%C4%8D%C3%ADslo
39. https://en.wikipedia.org/wiki/Method_of_complements#Binary_method
40. https://en.wikipedia.org/wiki/Mersenne_prime
41. https://en.wikipedia.org/wiki/Partition_(number_theory)
42. https://gmplib.org/
43. https://cs.wikipedia.org/wiki/GitHub
44. https://cs.wikipedia.org/wiki/BSD_licence
45. https://cs.wikipedia.org/wiki/GNU_General_Public_License