CHALMERS

GÖTEBORGS UNIVERSITET

Matematiska såll och deras tillämpningar

En introduktion och algoritmisk implementation

Mathematical Sieves and their Applications

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet

Nils Alexandersson
Erik Dagobert
Coën Lorcan Olofsson

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Matematiska såll och deras tillämpningar

En introduktion och algoritmisk implementation

Examensarbete för kandidatexamen i matematik inom Matematikprogrammet vid Göteborgs universitet

Nils Alexandersson
Erik Dagobert
Coën Lorcan Olofsson

Handledare: Lucile Devin
Anders Södergren

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Förord

Denna kandidatrapport har skrivits i syfte att introducera några grundläggande idéer inom sållteori och koppla dessa till nutidens framsteg inom ämnet. Under arbetets utförande har en gruppdagbok, samt individuella loggböcker förts. Dessa loggböcker innehåller detaljer angående utvecklingen av rapportens övergripande struktur, mötesanteckningar, och individuella rapporteringar av hur tiden har tillbringats.

Nedan listas huvudförfattare till rapportens respektive avsnitt.

- **Nils Alexandersson**: Populärvetenskaplig presentation, 5 Bruns såll, 8 Datorimplementation av Eratosthenes såll (inledningen), 8.2 Implementation av algoritmerna i Python, samt appendix B.3 och D med tillhörande kod.
- **Erik Dagobert**: 2 Förberedelser, 3 Det allmänna sållproblemet, 4 Eratosthenes generaliserade såll, 8.1 Grundläggande teori och algoritmer, samt appendix A, B.2 och C.
- **Coën Lorcan Olofsson**: 1 Inledning, 6 Selbergs såll, 7 Diskussion av sållmetoder, 8.3 Tillämpningar och resultat, samt appendix B.1, B.4 och B.5.

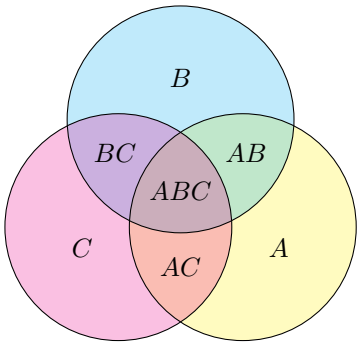
Vi vill också tacka våra handledare Anders Södergren och Lucile Devin för all den hjälp de har givit oss. Deras engagemang och vägledning har varit ovärderlig för detta arbete.

Populärvetenskaplig presentation

Primtal är förrädiskt oförutsägbara. De dyker upp när du minst anar det och kan till synes bete sig totalt oregelbundet. Trots detta är de helt deterministiska i sin natur och gränsen för vad som är och inte är ett primtal är mycket tydlig. Det är kanske just av denna anledning som primtalen, även i modern tid, är så intressanta att studera.

Hur många primtal finns det? Svaret är att det finns oändligt många. Om vi istället frågar oss hur många primtal det finns som är mindre än en miljon, så är svaret inte lika lätt. Till att börja med vet vi att primtal, med undantag för talet 2, aldrig är jämna. Därför kan vi åtminstone utesluta vartannat tal och säga att svaret är mindre än en halv miljon. Hur går vi vidare härifrån? Ett naturligt andra steg vore att föra samma resonemang för talet 3; förutom just 3 så är primtal aldrig delbara med 3 och därmed borde vi kunna dra bort ytterligare en tredjedels miljon från svaret. Detta är dessvärre inte riktigt sant. Tal som både är jämna och delbara med 3 har ju uteslutits två gånger. Det har alltså skett en dubbelräkning av alla tal som kan delas med 6, men vi kan kompensera för detta genom att addera en sjättedels miljon till svaret.

Denna uppskattning av svaret är bättre än den vi hade tidigare och givetvis kan vi fortsätta vidare med talet 5. På så sätt skulle vi komma ännu närmre det faktiska svaret, men vi skulle också behöva hantera fler dubbelräkningar. Den generella idén här kallas för *inklusions-exklusionsprincipen* och illustreras nedan för tre mängder med hjälp av figur 1.



Figur 1: Illustration av inklusions-exklusionsprincipen för tre stycken överlappande mängder A, B och C.

Säg att vi vill beräkna den totala storleken för de överlappande mängderna A, B och C. Vi kan börja med att summera storleken på mängderna var för sig. Om vi betecknar storleken på A som $|A|$ och gör på samma vis för de andra mängderna så kan vi skriva summan som $|A| + |B| + |C|$. Som bekant har det skett en dubbelräkning där mängderna överlappar varandra. Låt oss skriva AB när vi menar överlappet mellan A och B, och liknande för de andra överlappen. Precis som i tidigare exempel kompenserar vi för dubbelräkningen genom att dra bort dessa överlapp ifrån summan, vilket ger oss $|A| + |B| + |C| - |AB| - |BC| - |AC|$. Vidare har vi även ABC där alla tre mängderna överlappar varandra. Denna ingår i alla tidigare nämnda mängder och har därmed lagts till och dragits bort tre gånger om. Vi lägger till ABC en gång till och kommer fram till det slutgiltiga svaret

$$|A| + |B| + |C| - |AB| - |BC| - |AC| + |ABC|.$$

Detta är inklusions-exklusionsprincipen och den kan användas för ett godtyckligt antal mängder. Metoden att använda denna princip för att räkna primtal kallas för *Legendres såll* och är fundamental i teorin om matematiska såll.

Ett (matematiskt) såll är i stora drag en metod för att rensa bort vissa tal ur en mängd heltal. Som ovan exempelvis, då vi använde Legendres såll för att sålla bort icke-primtal ur mängden av heltal upp till en miljon. I denna rapport presenteras tre stycken såll; *Eratosthenes generaliserade såll*, *Bruns såll* samt *Selbergs såll* som alla bygger på idéer liknande det vi har sett här. Gemensamt för de tre sållen är att de inte ger något exakt svar, eftersom ett sådant ofta är opraktiskt att jobba med. Istället nöjer vi oss med approximationer, vilket ändå kan vara användbart förutsatt att vi känner till storleken på uppskattningens fel. Dessa feltermer och deras beteende utgör en väsentlig

skillnad mellan de olika sållen och en diskussion samt jämförelse av feltermerna är således relevant. Därför har detta tillägnats en egen del i rapporten vilken följer efter att de tre sållen presenterats.

I rapportens sista avsnitt redogör vi för hur Harald Helfgott i [1] har utvecklat idéer från ett grundläggande såll för att skapa effektiva algoritmer som kan utföras med en dator. Därefter framlägger vi vår implementation av algoritmerna i ett program skrivet i Python. Avslutningsvis presenteras även några resultat som vi fått från att köra programmet, där ett av resultaten knyter an till frågan om att räkna primtal. Men istället för att som ovan, titta på tal mellan noll och en miljon, har vi valt att svara på frågan för ett intervall med mittpunkt i 10^{19} och en bredd på 2.5×10^9.

Sammanfattning

Syftet med denna rapport är att ge läsaren en inblick i det matematiska delområdet sållteori genom att redogöra för dess grundläggande idéer och tillämpningar, samt att presentera en datorimplementation av Eratosthenes såll. I rapporten presenteras Eratosthenes generaliserade såll, samt Bruns och Selbergs såll. Först ges en kortfattad historisk kontext till sållen, följt av en översiktlig härledning, och därtill ett exempel på hur sållen kan tillämpas för att ge resultat om bland annat primtalstvillingar och primtal i aritmetiska serier. Efter att de tre sållen introducerats diskuteras och jämförs orsaken till deras feltermer. Avsikten med detta är att belysa de möjligheter och begränsningar som finns i sållen som verktyg.

Efter att ha etablerat viss grundläggande teori övergår rapportens fokus till en algoritmisk implementation av Eratosthenes såll baserad på Harald Helfgotts arbete [1]. Här beskrivs de underliggande matematiska principerna till algoritmen och dess övergripande struktur. Därefter redovisas den metod som har använts, och väsentliga beslut som fattats för att översätta algoritmen till ett effektivt program skrivet i programmeringsspråket Python. I den sista delen av rapporten presenteras resultat utifrån kvantitativ data som genererats av programmet vid sållning av primtal i intervallet $10^{19} \pm 1.25 \times 10^9$. För att styrka denna datas giltighet jämförs den mot primtalssatsen och med stöd i detta undersöks den förmodade fördelningen av primtalstvillingar, i förhållande till det som uppmätts i intervallet. Avslutningsvis betraktas datan med avseende på frekvensen av primtalsgap, följt av en kort diskussion om hur detta knyter an till framsteg som gjorts om primtalsgap i modern tid.

Abstract

This report aims to introduce the reader to the mathematical area of sieve theory through the elucidation of its fundamental principles and applications, as well as presenting an implementation in code of Eratosthenes sieving algorithm. We present the generalised sieve of Eratosthenes, as well as the sieves of Brun and Selberg; briefly describing their history and then focusing on outlining their general derivation as well as giving a thorough example of their application on sets such as the twin primes and primes in arithmetic progressions. Following the presentation of the sieves is a discussion which aims to focus the reader's attention on the origins of the error terms attached to the various methods and the effects they have on the quality of estimations which can be made.

Having presented the theory in some detail, we move our attention to an implementation of Eratosthenes original algorithm as based upon the work of Helfgott [1]. We present the underpinning mathematical principles and general algorithmic structure of Helfgott's implementation as well as our interpretation of his code and eventual improvements to the implementation. Following that is a presentation of results regarding the distributions of the prime numbers, the twin primes, and the behaviour of prime gaps in the interval $10^{19} \pm 1.25 \times 10^9$. Through the comparison of our code with the prime number theorem as applied to this interval we bolster its legitimacy, which leads to our investigation of the conjectured distribution of twin primes in our interval. We conclude this report with an analysis of the frequency of the prime gaps in our interval and discuss briefly its connection to modern day advances in and about the theory.

Innehåll

1 Inledning

Den som någon gång har funderat kring primtal, kanske har provat att ta en lista med naturliga tal upp till någon gräns och börjat försöka hitta de primtalen som finns. Efter en liten stund kanske man börjar lägga märke till mönster som uppträder; såsom att det finns vissa par av primtal som bara har ett tal mellan sig. Man kanske också börjar stryka igenom alla multipler av primtal för att ha något system för att uteslutta sammansatta tal. Så småningom kanske man inser att för att hitta alla primtal upp till ett visst tal behöver man bara alla primtal mindre än eller lika med kvadratroten av det talet.

Idén bakom denna process, att hitta primtal i en lista av naturliga tal, har funnits sedan antikens grekland med en algoritm som har tillskrivits den grekiska polyhistorn Eratosthenes (ca. 276 - 194 f.v.t.). Algoritmen har följande struktur;

1. Börja en lista med talet 2 och lista alla naturliga tal upp till någon gräns.
2. Ringa in 2 och stryk över alla andra tal som är delbara med 2.
3. Ringa in nästa tal som inte är struket och stryk alla andra tal delbara med det nya inringade talet.
4. Upprepa steg 3 tills varje tal på listan är struket eller inringat.

När algoritmen är avslutad så har varje primtal i listan blivit inringat och alla andra tal har strukits. Eratosthenes algoritm lade grunden för ämnet sållteori; ett område inom matematiken som försöker uppskatta storkleken på så kallade *siktade mängder*.

En siktad mängd är en mängd där alla element är heltal och har någon gemensam egenskap som till exempel en mängd som består av endast primtal eller mängden av alla tal i en aritmetisk talföjld. Den största fördelen med den grundläggande sållteorin är att metoderna är relativt elementära och flexibla, speciellt jämfört med andra metoder inom analytisk talteori. Det krävs inga idéer från komplex analys som till exempel Dirichlets L-funktioner eller serier för att ha användning av de enklare sållen och om man kan formulera mängderna på ett korrekt sätt, går det att effektivt tillämpa metoderna på ett stort antal mängder. Effektiviteten innebär att trots sin enkla konstruktion, kan dessa matematiska såll fortfarande ge starka resultat, även om bättre resultat kan hittas genom att använda mer raffinerade analytiska metoder. Exempelvis gäller det att antalet primtal mindre än x har det asymptotiska beteendet av $x/\log(x)$ vilket ges i primtalssatsen. Detta icke trivialaresultat kan nästan bevisas utan användning av zeta-funktioner eller då man utnyttjar Selbergs såll, dock så får man bara en övre gräns av $x/\log(x)$ istället för det asymptotiska beteendet. Mer avancerade sållmetoder har givit svar på frågor närliggande till primtalstvillingsförmodan i [2], och att det finns oändligt många par av primtal med maximalt 600 tal emellan sig [3].

Vår rapport fokuserar på små såll av både kombinatorisk och viktad form. Först, att sållen+ är små innebär det att deras sållning använder ett få antal restklasser modulo primtal. Näst, att ett såll är kombinatoriskt innebär att den använder sig av inklusion-exklusionsprincipen för att uppskatta mängdens storlek på ett lämpligt sätt. Slutligen, att ett såll är viktat innebär att någon viktfunktion används för att uppskatta storleken. Sållen vi håller oss till är Eratosthenes generaliserade såll, Bruns såll, och Selbergs såll. För varje såll ger vi en kortfattad härledning till dess formulering, där vi lyfter fram de centrala idéerna för dess konstruktion, och en förklaring till hur sållet används med exempel. En diskussion angående de sållens feletermerna följer sista teoretiska tillämpningen. Efteråt redovisar och analyserar vi en implementering i kod av Eratosthenes ursprungliga algoritm, vilket följer metoden som beskrivs i [1]. Med hjälp av de primtal som genereras av algoritmen undersökar vi slutligen vissa egenskaper hos primtalen såsom deras fördelning, fördelningen av primtalstvillingar, och frekvensen av olika stora avstånd mellan två på varandra följande primtal (primtalsgap). Dock, innan vi börjar redovisa något av sållen går vi igenom några förberedelser och förklarar det allmänna sållproblemet.

2 Förberedelser

Den här texten lånar större delen av sin notation från boken *An Introduction to Sieve Methods and their Applications* av Alina Carmen Cojocaru och M. Ram Murty [4]. Exempelvis om inget annat

nämns skriver vi p, q när vi menar primtal, n, d, k för positiva heltal och x, y, z för positiva reella tal. Den största gemensamma delaren betecknas här (d, k) och $\pi(z)$ är antalet primtal mindre än eller lika med z. Nedanstående avsnitt ämnar till att etablera mer notation och tekniker som är vanligt förekommande i sållteori med utgångspunkt i [4].

2.1 Multiplikativa funktioner

En särskilt trevlig delmängd av alla funktioner i talteoretiska sammanhang är de multiplikativa funktionerna. Vi säger att en funktion f är *multiplikativ* om $f(1) = 1$ och $f(mn) = f(m)f(n)$ då $(m, n) = 1$. Om detta håller för alla m, n så kallar vi f för *fullständigt multiplikativ*.

Multiplikativa funktioner har fördelen att en viss typ av summor kan omskrivas till produkter, s.k. Eulerprodukter. Om f är multiplikativ så följer det av aritmetikens fundamentalsats att

$$\sum_{n=1}^{\infty} f(n) = \prod_{p} \sum_{i=0}^{\infty} f(p^i)$$

så länge den första summan absolutkonvergerar.

2.2 Möbiusfunktionen

En ytterst väsentlig multiplikativ funktion i sållteori är Möbiusfunktionen,

$$\mu(n) = \begin{cases} 1, & \text{om } n \text{ är ett kvadratfritt, naturligt tal med jämnt antal primtalsdelare} \\ -1, & \text{om } n \text{ är ett kvadratfritt, naturligt tal med udda antal primtalsdelare} \\ 0, & \text{om } n \text{ inte är kvadratfri} \end{cases}$$

där *kvadratfri* innebär att n saknar delare på formen p^α för $\alpha > 1$. Vi kommer se att Möbiusfunktionen, bland annat, förenklar inklusion-exklusionsprincipen i nästa kapitel. Två andra viktiga egenskaper hos Möbiusfunktionen är

$$\sum_{d|n} \mu(d) = \begin{cases} 1, & \text{om } n = 1 \\ 0, & \text{om } n > 1 \end{cases}$$

och Möbius inverteringsformel som säger

$$f(n) = \sum_{d|n} g(d) \implies g(n) = \sum_{d|n} \mu(d) f(n/d) \tag{1}$$

om $f, g : \mathbb{N} \to \mathbb{C}$. Det finns flera varianter av Möbius inverteringsformel, en annan som vi kommer ha användning av hittas i appendix B.6.

3 Det allmänna sållproblemet

Det allmänna fallet i sållteori är att vi har en mängd heltal $\mathcal{A}$, en mängd primtal $\mathcal{P}$, samt delmängder $\mathcal{A}_p, \forall p \in \mathcal{P}$ och är intresserade av att sålla fram kardinaliteten $S(\mathcal{A}, \mathcal{P}) := \#(\mathcal{A} \setminus \cup_{p \in \mathcal{P}} \mathcal{A}_p)$. Vi låter $P(z)$ beteckna produkten av alla $p < z$ i $\mathcal{P}$ med specialfallet $P(z) = P_z$ då $\mathcal{P}$ är mängden av alla primtal. Med denna notation så skriver vi

$$S(\mathcal{A}, \mathcal{P}, z) := \#(\mathcal{A} \setminus \cup_{p|P(z)} \mathcal{A}_p).$$

För d, en kvadratfri produkt av element $p \in \mathcal{P}$, definierar vi $\mathcal{A}_d = \cap_{p|d} \mathcal{A}_p$ och $\mathcal{A}_1 = \mathcal{A}$. Använder vi oss av inklusion-exklusionsprincipen, formulerad med hjälp av möbiusfunktionen, får vi då

$$S(\mathcal{A}, \mathcal{P}, z) = \sum_{d|P(z)} \mu(d) \# \mathcal{A}_d. \tag{2}$$

Ett gemensamt drag hos de matematiska sållen diskuterade nedan är att låta X beteckna kardinaliteten av $\mathcal{A}$ och hitta δ_p så att

$$\#\mathcal{A}_p = \delta_p X + R_p \tag{3}$$

där $\delta_p \in [0, 1)$ kan betraktas som en uppskattning av andelen element i delmängden $\mathcal{A}_p$ och R_p som en felterm. Utvidgat, för ett kvadratfritt tal $d = p_1 \cdot ... \cdot p_n$, låter vi $\delta_d = \delta_{p_1} \cdot ... \cdot \delta_{p_n}$ så att $\#\mathcal{A}_d = \delta_{p_1} \cdot ... \cdot \delta_{p_n} X + R_d$.

I avsnitt 4 om Eratosthenes generaliserade såll och avsnitt 5 om Bruns såll så kommer vi välja $\delta_p = \omega(p)/p$ där $\omega(p)$ betecknar antalet utvalda restklasser modulo p vi vill sålla bort. Som följd låter vi i dessa fall $\mathcal{A}_p$ vara alla element som tillhör någon av dessa restklasser modulo p och för kvadratfria d låter vi $\omega(d) := \prod_{p|d} \omega(p)$. Sätter vi in det här δ_d:t i (2) ser vi att vi får en huvudterm som är X multiplicerat med summan $\sum_{d|P(z)} \mu(d)\omega(d)/d$. Vi kan se att denna summa är lika med

$$W(z) := \prod_{\substack{p \in \mathcal{P} \\ p < z}} \left(1 - \frac{\omega(p)}{p}\right). \tag{4}$$

då produkten ovan genererar en summa av alla kvadratfria produkter av $p \in \mathcal{P}$ som är multiplicerat med -1 om vi inkluderat ett udda antal primtal.

4 Eratosthenes generaliserade såll

Även om Eratosthenes såll fortfarande är mest känt som en enkel algoritm vilken utan större möda kan utföras med papper och penna så har dess idé raffinerats genom historien till att inkorporera mer avancerad teori för att tillämpas på nya sätt. År 1808 utvecklade Adrien-Marie Legendre (1752-1833) sållet med hjälp av inklusion-exklusionsprincipen formulerad på formen från avsnitt 3. Nedan kommer vi gå igenom denna form av Eratosthenes såll samt, i efterföljande avsnitt, följa hur [4] presenterar en utvidgning av sållet till en mer generell form.

4.1 Legendres såll

Första steget för att utveckla Eratosthenes såll är att omformulera sållexemplet från inledningen i de termer vi definierade i föregående avsnittet. I exemplet börjar vi med en lista av alla positiva heltal upp till en övre gräns, säg x, vilken vi kan skriva som $\mathcal{A} = \{n \in \mathbb{N} : n \leq x\}$. Låt alla primtal upp till z redan vara inringade och mängderna vi kryssar över vara $\mathcal{A}_p = \{a \in \mathcal{A} : a \equiv 0 \pmod{p}\}$ — mängden av multiplar av p som är mindre än eller lika med x. Vi kan se att kardinaliteten av $\mathcal{A}_d$ är $\lfloor x/d \rfloor$ och väljer vi $z = \sqrt{x}$ så med hjälp av (2),

$$\pi(x) - \pi(\sqrt{x}) + 1 = \sum_{d|P(\sqrt{x})} \mu(d) \left\lfloor \frac{x}{d} \right\rfloor$$

där 1:an på vänstersidan tar i hänsyn att $1 \in \mathcal{A}$ inte sållas bort på högersidan och $\pi(\sqrt{x})$ är de inringade primtalen. Detta var Legendres idé när han omformulerade Eratosthenes såll till att räkna primtal [5, kapitel 1.1].

Legendres formel är bra för att räkna primtal exakt men är långsam och golvfunktionen kan vara svårhanterlig. Vi kan underlätta för oss genom att ersätta golvfunktionen med x/d och med bråkdelen som restterm. Skriver vi bråkdelen som $\{x/d\} := x/d - \lfloor x/d \rfloor = O(1)$ leder det oss till $\#\mathcal{A}_d = x/d + O(1)$. Dessvärre resulterar det här i en väldigt stor felterm då restermerna ackumulerar, så att $\pi(x) - \pi(\sqrt{x}) + 1 = x \sum_{d|P(\sqrt{x})} \frac{\mu(d)}{d} + O(2^{\pi(\sqrt{x})})$.

Eftersom $\mathcal{A}_p$, för varje p, är definierad som en restklass modulo p så säger vi att antalet utvalda restklasser modulo p är $\omega(p) = 1$ för alla p – vi kallar Eratosthenes för ett endimensionellt eller linjärt såll av den här anledningen. Mer allmänt betecknas dimensionen av ett såll med parametern κ om det är ett minimum för en genomsnittlig övre gräns för $\omega(p)$ för alla p [6], så att för alla z

$$\sum_{p|P(z)} \frac{\omega(p)\log(p)}{p} \leq \kappa \log(z) + O(1). \tag{5}$$

Vi ser således att ett naturligt nästa steg är att generalisera Eratosthenes-Legendres såll för fler dimensioner.

4.2 Eratosthenes generaliserade såll

Vi vill alltså härleda ett generellt verktyg utifrån idéerna från föregående avsnitt vilket vi kommer göra genom att följa [4, kap.5.4]. Om vi låter $\mathcal{A}_d$, $\mathcal{P}$, $P(z)$ och $\omega(d)$ vara definierade som i avsnitt 3 så kan vi, med utgångspunkt i Eratosthenes och Legendres grundtankar, skapa ett mer flexibelt och effektivt såll. Den stora skillnaden mot vad vi gjorde i föregående avsnitt är att vi här tillåter att fler restklasser sållas bort per primtal p.

Utöver vad som står i avsnitt 3 kommer vi göra antagandena att $\#\mathcal{A}_d = 0$ för alla d större än något positivt y, att $|R_d| = O(\omega(d))$ samt att (5) håller. Vi kan lätt övertyga oss om att dessa är rimliga antaganden i det klassiska Eratosthenes såll-fallet. Eftersom vi utgår från en ändlig mängd $\mathcal{A}$ så håller det första antagandet då $\#\mathcal{A}_p = 0$ för $p > X$ vilket innebär att, väldigt grovt, $y \leq P(X)$. Att det andra antagandet gäller såg vi i föregående avsnitt ty $\{x/d\} \leq \omega(d) = 1$. Sist så ser vi att (5) håller genom att sätta in $\omega(p) = 1$ och utnyttja att

$$\sum_{p \leq n} \frac{\log p}{p} = \log n + O(1). \tag{6}$$

Vi börjar återigen med Legendres grundidé, inklusion-exklusionsprincipen, men infogar det mer generella uttrycket för $\#\mathcal{A}_d$ i formeln

$$S(\mathcal{A}, \mathcal{P}, z) = \sum_{d|P(z)} \mu(d)\#\mathcal{A}_d = X \sum_{\substack{d|P(z) \\ d \leq y}} \frac{\omega(d)}{d}\mu(d) + \sum_{\substack{d|P(z) \\ d \leq y}} \mu(d)R_d,$$

där trunkeringen av summan kommer av antagandet $\#\mathcal{A}_d = 0, \forall d > y$.

Studerar vi den första summan ovan ser vi att vi kan skriva om den som summan över alla $d \mid P(z)$ minus summan över alla $d \mid P(z)$ så att $d > y$. Med $W(z)$ på summaformen från avsnitt 3 kan vi således dela upp den siktade mängden i en huvudterm och en rest på följande vis

$$S(\mathcal{A}, \mathcal{P}, z) = XW(z) + \left(-X \sum_{\substack{d|P(z) \\ d > y}} \frac{\omega(d)}{d}\mu(d) + \sum_{\substack{d|P(z) \\ d \leq y}} \mu(d)R_d \right).$$

De två summorna innanför parentesen i uttrycket ovan är vår felterm. Fokuserar vi på den andra summan så kan vi uppskatta dess storlek, först med hjälp av triangelolikheten och sedan med $\left|\mu(d)\right| \leq 1$ och vårt antagande $|R_d| = O(\omega(d))$, så att summanden $\left|\mu(d)R_d\right| \leq O(\omega(d))$. Flyttar vi in summan innanför ordo-tecknet så kan vi förenkla den med Rankins trick som, enligt [4, s.68] säger att indikatorfunktionen, $1_{n \leq x} \leq \frac{x}{n}$. Därav skriver vi

$$\sum_{\substack{d|P(z) \\ d \leq y}} \omega(d) = \sum_{d|P(z)} \omega(d)(1_{d \leq y})^\delta \leq \sum_{d|P(z)} \omega(d) \left(\frac{y}{d}\right)^\delta$$

för alla $\delta > 0$. Eftersom $\omega(d)/d^\delta$ är multiplikativ kan vi skriva summan som en produkt

$$y^\delta \prod_{p|P(z)} \left(1 + \frac{\omega(p)}{p^\delta}\right) = \exp\left(\delta \log y + \sum_{p|P(z)} \log\left(1 + \frac{\omega(p)}{p^\delta}\right)\right) \leq \exp\left(\delta \log y + \sum_{p|P(z)} \frac{\omega(p)}{p^\delta}\right)$$

vari sista steget vi använder att $\log(1+x) \leq x$. Väljer vi nu $\delta = 1 - 1/\log z$ och utnyttjar olikheten $e^x \leq 1 + xe^x$, som gäller för $x \geq 0$, och sist partiell summering av (5) får vi att

$$\sum_{\substack{d|P(z) \\ d \leq y}} \omega(d) = O\left(\frac{y}{\log(z)}(\log(z))^{\kappa+1} \exp\left(-\frac{\log(y)}{\log(z)}\right)\right).$$

På ett liknande sätt omvandlar vi den första summan i feltermen med partiell summering av antagandet (5) och sedan använder resultatet vi fick för den andra summan.

Sammanställt, erhåller vi nästa sats, [4, sats 5.4.1]:

Sats 4.1 (Eratosthenes generaliserade såll)**.** *Med notationen från avsnitt 3 och följande antaganden*

1. $\#\mathcal{A}_d = 0, \forall d > y$ *för något* $y \in \mathbb{R}_+$,
2. $|R_d| = O(\omega(d))$,
3. $\sum_{p|P(z)} \frac{\omega(p)\log(p)}{p} \leq \kappa \log(z) + O(1)$,

så gäller att

$$S(\mathcal{A}, \mathcal{P}, z) = XW(z) + O\left(\left(X + \frac{y}{\log z}\right)(\log z)^{\kappa+1} \exp\left(-\frac{\log y}{\log z}\right)\right).$$

4.3 En högredimensionell tillämpning av Eratosthenes såll

Ett av målen med avsnitt 4.2 var att kunna använda Eratosthenes såll till att sålla bort flera kongruensklasser per primtal p. I det här avsnittet kommer vi utnyttja denna egenskap till att sålla fram primtalstvillinigar och sedan bevisa en variant av Bruns sats [4, korollarium 5.4.5],

Sats 4.2 (Bruns sats)**.** *Summan av reciproker,*

$$\sum_{\substack{p \\ p+2 \text{ prima}}} \frac{1}{p} < \infty.$$

Satsen bevisades först av Brun 1919 i artikeln *La série* $1/5+1/7+1/11+1/13+1/17+1/19+1/29+1/31+1/41+1/43+1/59+1/61\ldots$ *ou les dénominateurs sont «nombres premiers jumeaux »est convergente ou finie* men vi kommer här följa ett annat bevis, ur [4, s.72f], som använder sig av sats 4.1.

Primtalstvillingar i det här fallet är tal i mängden $\mathcal{P}_2(x) := \{p < x : p + 2 \text{ är ett primtal}\}$. För att hitta dessa tal så vill vi utesluta alla multiplar av primtal samt alla tal två mindre än dessa multiplar. Med andra ord väljer vi kongruensklasserna 0 och -2 modulo p i sålldefinitionen så att $\omega(2) = 1$ och $\omega(p) = 2$ för alla primtal $p > 2$ samt att dimensionen $\kappa = 2$. Det andra antagandet i sats 4.1 är en svagare variant av ett antagande i Bruns såll och verifieras i avsnitt 5.2. Vårt val av kongruensklasser ger att $y = x + 2$ i det första antagandet av sats 4.1 som då säger att

$$\#\mathcal{P}_2(x) \leq \pi(z) + S(\mathcal{A}, \mathcal{P}, z) \leq z + xW(z) + O\Big(x(\log z)^3 \exp\Big(-\frac{\log x}{\log z}\Big)\Big).$$

Vi kan enkelt se att $W(z)$ i huvudtermen är lika med

$$\prod_{p<z}\left(1 - \frac{\omega(p)}{p}\right) = \frac{1}{2}\exp\left(\sum_{2<p<z} \log\left(1 - \frac{2}{p}\right)\right) \asymp \exp\left(-\sum_{2<p<z}\frac{2}{p}\right) \tag{7}$$

där sista steget följer av olikheten $\log(1 + x) \leq x$, om $x > -1$ i $\ll$–riktningen och den andra riktningen visas i sats B.4 från appendix. Med en partiell summation av (6) (se sats B.3) erhåller vi att $\sum_{p\leq z}\frac{1}{p} = \log\log z + O(1)$ som, när vi sätter in i (7), ger oss att

$$W(z) \asymp \exp\left(-2\sum_{2<p<z}\frac{1}{p}\right) \asymp \exp\big(-2\log\log z + O(1)\big) \asymp (\log z)^{-2}. \tag{8}$$

Slutligen får vi alltså att huvudtermen $xW(z) \ll x(\log z)^{-2}$ och väljer vi $\log z = \log x/(6\log\log x)$ så leder det oss till [4, sats 5.4.4] som säger

$$\#\mathcal{P}_2(x) \ll \frac{x(\log\log x)^2}{\log^2 x}. \tag{9}$$

Genom att partialsummera summan av de reciproka värdena, med c_n som indikatorfunktionen på $\mathcal{P}_2$ och $f(n) = 1/n$ i notationen från sats B.1, så får vi

$$\sum_{\substack{p \leq x \\ p+2 \text{ prima}}} \frac{1}{p} = \#\mathcal{P}_2(x) \cdot \frac{1}{x} + \int_2^x \#\mathcal{P}_2(t) \cdot \frac{1}{t^2} \mathrm{d}t.$$

Med hjälp av (9) får vi att summan i Bruns sats är $\ll \int_2^\infty \frac{(\log\log t)^2}{t \log^2 t} \mathrm{d}t$ som är ändlig, vilket medför att summan är begränsad.

Vad vi nu visat för summan över $\mathcal{P}_2$ är i kontrast mot en annan identitet vi använde i ovanstående bevis, $\sum_{p \leq n} \frac{1}{p} = \log\log n + O(1)$ vilken implicerar att summan av reciproker över primtalen divergerar. Detta innebär inte att primtalstvillinigar är en ändlig delmängd av primtalen men det säger oss att andelen primtalstvillingar är relativt liten. I nästa del kommer vi se ett annat resultat tillskrivet Viggo Brun, Bruns såll, och med hjälp av detta visa ett resultat i motsatt riktning för en mängd snarlik mängden primtalstvillinigar.

5 Bruns såll

När den norska matematikern Viggo Brun (1881-1978) år 1915 publicerade artikeln *Über das Goldbachsche Gesetz und die Anzahl der Primzahlpaare*, lades grunden till modern sållteori [7]. I artikeln presenterade Brun ett såll baserat på Eratosthenes idéer, som numera kallas Bruns såll och fyra år senare använde han det för att bevisa Bruns sats, vilken finns beskriven i avsnitt 4.3 ovan. I detta avsnitt ges en kortfattad beskrivning av Bruns såll. Vi nöjer oss med att illustrera huvudidéen bakom sållet och tar stöd i [4, kap.6] för att göra detta. Därefter presenterar vi ett exempel hämtat ur [4] på hur sållet kan tillämpas för att frambringa ett resultat som angränsar till primtalstvillingsförmodan.

5.1 Beskrivning av sållet

Med målet att förminska feltermerna i Eratosthenes generaliserade såll, lät Viggo Brun modifiera uttrycket (2). Han gjorde detta genom införandet av en funktion g i högerledets summa, vilket efter omskrivning, resulterade i uttrycket:

$$\sum_{d \mid P(z)} \mu(d) g(d) \#\mathcal{A}_d = S(\mathcal{A}, \mathcal{P}, z) + \sum_{d \mid P(z)} \sum_{\substack{p \mid P(z) \\ p < q(d)}} \mu(d)(g(d) - g(pd)) \mathcal{S}(\mathcal{A}_{pd}, \mathcal{P}, p), \tag{10}$$

där $q(d)$ är det minsta primtal som delar d för $d \geq 2$ och $q(1) = z + 1$. Brun visade att (10) håller för varje funktion g som uppfyller $g(1) = 1$.

Därefter konstruerade han två funktioner g_U och g_L, så att $\mu(d)(g_U(d) - g_U(pd)) \geq 0$ och $\mu(d)(g_L(d) - g_L(pd)) \leq 0$ håller för alla $d \mid P(z)$ och $p < q(d)$. Således kunde han erhålla en undre och övre begränsning till $S(\mathcal{A}, \mathcal{P}, z)$, nämligen

$$\sum_{d \mid P(z)} \mu(d) g_L(d) \#\mathcal{A}_d \leq S(\mathcal{A}, \mathcal{P}, z) \leq \sum_{d \mid P(z)} \mu(d) g_U(d) \#\mathcal{A}_d. \tag{11}$$

Enligt [4] var detta Bruns innovativa idé. Genom att sedan skriva $\#\mathcal{A}_d = \frac{\omega(d)}{d} X + R_d$ kunde han, efter mycken möda, hitta en approximation och en felterm för begränsningarna i (11) och på så vis formulera Bruns såll. Följande sats är hämtad ur [4, kap.6.2].

Sats 5.1 (Bruns såll)**.** *Med notation från avsnitt 3, låt b vara ett positivt heltal och låt λ vara ett reellt tal som uppfyller $0 < \lambda e^{1+\lambda} < 1$. Antag att $|R_d| \leq \omega(d) \leq dA_1$ för varje kvadratfritt tal d med faktorer ur $\mathcal{P}$ och någon konstant $A_1 < 1$, samt att det finns konstanter $A_2 \geq 1$ och $\kappa > 0$ som uppfyller*

$$\sum_{w \leq p < z} \frac{\omega(p) \log p}{p} \leq \kappa \log(z/w) + A_2, \quad \text{för } 2 \leq w \leq z.$$

Då begränsas $S(\mathcal{A}, \mathcal{P}, z)$ ovanifrån av

$$XW(z)\left(1+\lambda c_1 \exp\left(c_2\frac{2b+3}{\lambda\log z}\right)\right)+O\left(z^{c_3}\right), \tag{12}$$

och underifrån av

$$XW(z)\left(1-c_1 \exp\left(c_2\frac{2b+2}{\lambda\log z}\right)\right)+O\left(z^{c_3-1}\right). \tag{13}$$

Konstanterna c_1, c_2 och c_3 definieras som

$$c_1 := \frac{2\lambda^{2b}e^{2\lambda}}{1-\lambda^2 e^{2+2\lambda}}, \quad c_2 := \frac{A_2}{2}\left(1+\frac{\kappa+\frac{A_2}{\log 2}}{1-A_1}\right) \quad \textit{och} \quad c_3 := 2b+\frac{2.01}{e^{2\lambda/\kappa}-1}.$$

Precis som i avsnitt 4 betraktar vi här κ som dimensionen av sållet. Notera att konstanten b kan väljas relativt fritt och kan bidra till att hålla nere feltermen. Nedan följer ett exempel på hur detta kan göras, där en tillämpning av satsen presenteras.

5.2 En tillämpning av Bruns såll

Låt oss uttrycka en modifierad version av primtalstvillingsförmodan på följande vis:

Det finns oändligt många par av heltal $(n, n+2)$ där båda talen har högst r primtalsfaktorer.

För valet $r = 1$ erhålls exakt primtalstvillingsfömodan, vilken står obevisad. Däremot går det att med hjälp av Bruns såll bevisa påståendet för $r = 7$. Vi följer samma tillvägagångssätt som i [4, kap.6.2]. Definiera mängden

$$\mathcal{T} := \{n \le x\text{: } n \textit{ och } n+2 \textit{ har som mest } 7 \textit{ primtalsfaktorer}\}.$$

Målet är att med hjälp av (13), hitta en undre begränsning till $\#\mathcal{T}$ som divergerar då $x \to \infty$.

Låt $\mathcal{P}$ vara mängden av alla primtal och låt

$$\mathcal{A} := \{n(n+2) : n \le x\}.$$

Precis som i avsnitt 4.3 vill vi sålla mängden $\mathcal{A}$ med avseende på kongruensklasserna $0, -2$ modulo p. Således har vi även här att $\omega(2) = 1$ och $\omega(p) = 2$ för $p > 2$, samt $\kappa = 2$.

För att kunna använda (13) måste givetvis satsens alla antaganden vara uppfyllda. Sätt därför $A_1 := 2/3$ så att antagandet $\omega(d) \le dA_1$ gäller. Härnäst kontrollerar vi att $|R_d| \le \omega(d)$ håller genom att visa det något starkare påståendet $|R_p| \le \omega(p)$. Med användning av (3) har vi att

$$\#\mathcal{A}_p = \left\lfloor \frac{x}{p} \right\rfloor + \left\lfloor \frac{x+2}{p} \right\rfloor \le \frac{2}{p}x + 2, \quad \text{för alla } p > 2.$$

En liknande beräkning ger oss att $\#\mathcal{A}_p \ge 2x/p$ vilket visar att $|R_p| \le 2$ för alla $p > 2$. I fallet då $p = 2$ har vi istället att $\#\mathcal{A}_2 = \lfloor x/p \rfloor$ som implicerar $|R_2| \le 1$. Därmed kan vi dra slutsatsen att $|R_p| \le \omega(p)$, vilket implicerar att $|R_d| \le \omega(d)$ håller för alla kvadratfria tal d. Att det sista antagandet i sats 5.1 är uppfyllt kan vi se genom användning av (6) som tillsammans med $\omega(p) \le 2$ ger oss att

$$\sum_{w \le p < z} \frac{\omega(p)\log p}{p} \le 2\log\left(z/w\right) + O(1),$$

varpå vi ser att $\kappa = 2$ uppfyller kriterierna och att A_2 existerar. Något explicit värde på A_2 är inte nödvändigt, vilket kan ses genom att betrakta exponenten i (13). Taylorutveckling av denna ger

$$\exp\left(c_2\frac{2b+2}{\lambda\log z}\right) = 1 + c_2\frac{2b+2}{\lambda}(\log z)^{-1} + \left(c_2\frac{2b+2}{\lambda}\right)^2(\log z)^{-2} + \cdots \tag{14}$$

Fixera b, λ och A_2 så att även c_2 är fixerat, då följer det att (14) begränsas av $1 + O((\log z)^{-1})$, då $z \to \infty$. Nu är vi redo att tillämpa (13). Vi sätter in begränsningen ovan tillsammans med $W(z) \gg (\log z)^{-2}$, vilket ges av (8), och får att

$$S(\mathcal{A}, \mathcal{P}, z) \gg \frac{x}{(\log z)^2}\big(1 - c_1\big) + O\big(z^{c_3 - 1}\big). \tag{15}$$

Nu vill vi välja b och λ, så att $c_1 < 1$ och c_3 är litet. Detta uppfylls av $b = 1$ samt $\lambda = 0.253$ som medför att $c_3 - 1 < 8$.

I fallet att n eller $n + 2$ har mer än 7 primtalsfaktorer måste $p \mid n(n + 2)$ för något primtal $p < n^{1/8}$. Således erhålls $\#\mathcal{T}$ ur $S(\mathcal{A}, \mathcal{P}, z)$ genom att sålla för primtal upp till $z = x^{1/u}$, där $u < 8$. Insättning av detta i (15) ger

$$\frac{u^2 x}{(\log x)^2}\big(1 - c_1\big) + O\big(x^{\frac{c_3 - 1}{u}}\big) \gg \frac{x}{(\log x)^2} + O\big(x^{\frac{c_3 - 1}{u}}\big).$$

Genom att ställa det ytterligare kravet $u > c_3 - 1$, har vi att feltermen går mot noll då $x \to \infty$, och vi får slutligen:

Sats 5.2.

$$\#\mathcal{T} \gg \frac{x}{(\log x)^2}.$$

Lägg märke att vi inte bara har bevisat påståendet att det finns oändligt många par av heltal $(n, n+2)$ där båda talen har högst 7 primtalsfaktorer, uttrycket ovan ger oss även en undre gräns för det asymptotiska beteendet hos mängden ifråga. Härnäst riktas fokus mot denna rapports tredje och sista såll.

6 Selbergs såll

Ett ytterligare perspektiv på sållmetoder gavs av en annan norsk matematiker, Atle Selberg (1917-2007), under 1940 talet. Hans metod var av en kombinatorisk stil som liknade de tidigare. Dock använde han sig av en ny typ av vikt och en uppskattning av en summa av Möbiusfunktioner för att uppskatta kardinaliteten av $\mathcal{A}$. Denna metod kallas för Selbergs såll och är ett exempel på ett viktat såll. I detta avsnitt så ges en beskrivning av sållets uppbyggnad följt av en tillämpning av sållet för att uppskatta antalet primtal i en aritmetisk talföljd.

6.1 Beskrivning av sållet

Selbergs nya vikter, vilka motsvarar δ_d i (3), består av

$$\frac{1}{f(n)}, \quad f(n) = \sum_{d \mid n} f_1(d)$$

där $f(n)$ och $f_1(d)$ är multiplikativa funktioner och $f_1(d)$ är entydigt bestämd med hjälp av Möbius inverteringsformel. Det krävs också att $f(p) > 1$ för alla $p \in \mathcal{P}$. Till exempel, om vi vill sålla bort alla tal i mängden $\mathcal{A}_d = \{n \leq x : n \equiv 0 \pmod{d}\}$, där $x > 0$, då blir $\#\mathcal{A}_d = x/d + O(1)$ och dessutom $f(d) = d$. Föregående exempel är relativt enkelt men i mer komplexa fall kan en sådan mer allmän vikt ge sållteorin möjligheten att uppskatta nya typer av mängder som tidigare hade inte varit möjligt att uttrycka med endast restklasser modulo primtal.

Enligt [4, s. 114], bestod grundläggande uppskattningen av Selbergs såll huvudsakligen av att för en given reell talföljd, (λ_d), med $\lambda_1 = 1$, har vi att

$$\sum_{d \mid n} \mu(d) \leq \left(\sum_{d \mid n} \lambda_d \right)^2.$$

För att förstå varför denna uppskattning av Möbiusfunktionens summa fungerar, erinrar vi den första egenskapen som redovisas i 2.2. Om $n = 1$ är både höger- och vänsterledet ovan lika med 1,

medan $n = 0$ ger att vänsterledet är 0 och högerledet är ickenegativt ty detta är en kvadrat. Slutmålet med sållets härledning blir att konstruera talföljden (λ_d) så att uppskattningen minimeras, vilket kommer göras senare. Med hjälp av talföljden (λ_d) kan vi nu uppskatta $S(\mathcal{A}, \mathcal{P}, z)$ eftersom

$$S(\mathcal{A}, \mathcal{P}, z) = \sum_{\substack{a \in \mathcal{A} \\ a \notin \mathcal{A}_p, \forall p | P(z)}} 1 = \sum_{d | P(z)} \mu(d) \sum_{a \in \mathcal{A}_d} 1 = \sum_{a \in \mathcal{A}} \left(\sum_{\substack{d | P(z) \\ a \in \mathcal{A}_d}} \mu(d) \right) \leq \sum_{a \in \mathcal{A}} \left(\sum_{\substack{d | P(z) \\ a \in \mathcal{A}_d}} \lambda_d \right)^2. \quad (16)$$

Låt oss nu anta att $\lambda_d = 0$ för alla $d > z$. Med antagandet kan vi nu skriva om kvadraten av summan som en summa över två olika d och omvandla (16) till

$$S(\mathcal{A}, \mathcal{P}, z) \leq \sum_{a \in \mathcal{A}} \left(\sum_{\substack{d_1 | P(z) \\ a \in \mathcal{A}_{d_1}}} \lambda_{d_1} \sum_{\substack{d_2 | P(z) \\ a \in \mathcal{A}_{d_2}}} \lambda_{d_2} \right) = \sum_{a \in \mathcal{A}} \sum_{\substack{d_1, d_2 | P(z) \\ a \in \mathcal{A}_{[d_1, d_2]}}} \lambda_{d_1} \lambda_{d_2} = \sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z)}} \lambda_{d_1} \lambda_{d_2} \# \mathcal{A}_{[d_1, d_2]}$$

där $[a, b]$ betecknar den minsta gemensamma multipeln av a och b. Vi kombinerar summorna av λ_{d_1} och λ_{d_2} ovan genom att inse att om $a \in \mathcal{A}_{d_1}$ och $a \in \mathcal{A}_{d_2}$ måste den vara ett element i $\mathcal{A}_{[d_1, d_2]}$ på grund av definitionen av $\mathcal{A}$. Med infogande av $\delta_d = 1/f(d)$ i (3) får vi uppskattningen som är grundläggande för Selbergs såll;

$$S(\mathcal{A}, \mathcal{P}, z) \leq X \sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z)}} \frac{\lambda_{d_1} \lambda_{d_2}}{f([d_1, d_2])} + O\left(\sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z)}} |\lambda_{d_1}| |\lambda_{d_2}| |R_{[d_1, d_2]}| \right). \quad (17)$$

Vi kommer att fokusera vår härledning på huvudtermen av sållet eftersom den kräver att vi visar hur vi bestämmer talföjlden (λ_d) vilket är avgörande för sållets konstruktion. Med [4, s. 120] som utgångspunkt, börjar vi med att använda oss av ett lemma som låter oss skriva om multiplikativa funktioner av minsta gemensamma multipler, se Appendix B.5. Lemmat tillsammans med definitionen av f tillåter följande omskrivning:

$$\sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z)}} \frac{\lambda_{d_1} \lambda_{d_2}}{f([d_1, d_2])} = \sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z)}} \frac{\lambda_{d_1} \lambda_{d_2}}{f(d_1) f(d_2)} f((d_1, d_2)) = \sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z)}} \frac{\lambda_{d_1} \lambda_{d_2}}{f(d_1) f(d_2)} \sum_{\delta | (d_1, d_2)} f_1(\delta). \quad (18)$$

Nu byter vi summationsordningen i (18), genom att inse att om $\delta | (d_1, d_2)$ då är $\delta \leq z$ och $\delta | P(z)$. Med detta, har vi också att d_1 och d_2 måste vara delbara med δ. Därför kan vi skriva om (18) till

$$\sum_{\substack{\delta \leq z \\ \delta | P(z)}} f_1(\delta) \sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2 | P(z) \\ \delta | (d_1, d_2)}} \frac{\lambda_{d_1} \lambda_{d_2}}{f(d_1) f(d_2)} = \sum_{\substack{\delta \leq z \\ \delta | P(z)}} f_1(\delta) \left(\sum_{\substack{d \leq z \\ d | P(z) \\ \delta | d}} \frac{\lambda_d}{f(d)} \right)^2.$$

Sätter vi

$$u_\delta = \sum_{\substack{d \leq z \\ d | P(z) \\ \delta | d}} \frac{\lambda_d}{f(d)} \implies \sum_{\substack{\delta \leq z \\ \delta | P(z)}} f_1(\delta) \left(\sum_{\substack{d \leq z \\ d | P(z) \\ \delta | d}} \frac{\lambda_d}{f(d)} \right)^2 = \sum_{\substack{\delta \leq z \\ \delta | P(z)}} f_1(\delta) u_\delta^2. \quad (19)$$

Vi kan nu tillämpa en variant på Möbius inverteringsformel, se Appendix B.6, på u_δ, vilket ger att

$$\frac{\lambda_\delta}{f(\delta)} = \sum_{\substack{d \leq z \\ d | P(z) \\ \delta | d}} \mu(d/\delta) u_d \quad (20)$$

Eftersom vi har diagonaliserat huvudtermen i (19) kan vi enkelt bestämma ett värde på u_δ sådant att summan antar sitt minimivärde genom att använda kvadratkomplettering. Vi gör detta med hjälp av (20) och införandet av en ny funktion $V(z) = \sum_{d \leq z,\, d | P(z)} \frac{\mu^2(d)}{f_1(d)}$, vilket direkt underlätter kvadratkompletteringen av summans termer. Det är också värt att påpeka att $V(z)$

är strikt positivt eftersom $f_1(d)$ alltid är större än noll. Detta gäller eftersom för varje primtal är $f_1(p) = \sum_{d|p} \mu(d)f(p/d) = \mu(1)f(p) + \mu(p)f(1) = f(p) - 1 > 0$. Dessutom har vi att f_1 är multiplikativ och att d är kvadratfri vilket tillsammans med föregående resonemang medför att $f_1 > 0$ för alla d i summan. Därför blir $V(z)$ en summa av strikt positiva tal och därmed strikt positiv själv. Om vi sätter $\delta = 1$ i (20) då får vi att

$$1 = \frac{\lambda_1}{f(1)} = \sum_{\substack{d \leq z \\ d|P(z)}} \mu(d)u_d.$$

vilket vi kan använda för att hitta en minimum på följande sätt:

$$\begin{aligned}\sum_{\substack{\delta \leq z \\ \delta|P(z)}} f_1(\delta)u_\delta^2 &= \sum_{\substack{\delta \leq z \\ \delta|P(z)}} f_1(\delta)u_\delta^2 + \frac{V(z)}{V(z)^2} - \frac{2}{V(z)} + \frac{1}{V(z)} \\ &= \sum_{\substack{\delta \leq z \\ \delta|P(z)}} f_1(\delta)u_\delta^2 + \sum_{\substack{\delta \leq z \\ \delta|P(z)}} \frac{\mu^2(\delta)}{f_1(\delta)V^2(z)} - \sum_{\substack{\delta \leq z \\ \delta|P(z)}} \frac{2\mu(\delta)u_\delta}{V(z)} + \frac{1}{V(z)}. \qquad (21)\end{aligned}$$

Vi kan samla alla summor i (21) till en och då vi kvadratkompletterar summans termer har vi att

$$\sum_{\substack{\delta \leq z \\ \delta|P(z)}} f_1(\delta)u_\delta^2 = \sum_{\substack{\delta \leq z \\ \delta|P(z)}} f_1(\delta)\left(u_\delta - \frac{\mu(\delta)}{f_1(\delta)V(z)}\right)^2 + \frac{1}{V(z)}.$$

Givetvis blir ovanstående summa noll då vi väljer $u_\delta = \frac{\mu(\delta)}{f_1(\delta)V(z)}$, vilket måste vara en minimivärde eftersom $f_1(\delta$ är strikt positiv. Det går också att bevisa att med detta val av u_δ blir $|\lambda_d||V(z)| \leq |V(z)|$ [4, s. 122-123] och därför härleder vi följande sats.

Sats 6.1 (Selbergs såll)**.** *Behåll notationen från tidigare i detta avsnitt och avsnitt 2. Då har vi att*

$$S(\mathcal{A}, \mathcal{P}, z) \leq \frac{X}{V(z)} + O\left(\sum_{\substack{d_1, d_2 \leq z \\ d_1, d_2|P(z)}} |R_{[d_1,d_2]}|\right).$$

Ovanstående formulering är hur satsen redovisas i [4, sats 7.2.1]. Notera att i Selbergs såll har vi färre paratmetrar att bestämma än i de tidigare sållen, vilket hjälper en del med dess tillämpning. Ett exempel på hur sållet kan tillämpas redovisas i nästa avsnitt.

6.2 Selbergs såll och primtal i aritmetiska talföljder

Med hjälp av Selbergs såll kan vi uppskatta antalet primtal som finns i en aritmetisk talföjld, det vill säga antalet primtal på formen $p = a + tk$ där a, k är valda konstanter och t är godtyckligt. Det finns också ett krav på a och k, nämligen att de är relativt prima, utan det kravet så skulle maximalt ett primtal finnas i talföjlden. Slutligen antar vi att k är kvadratfritt, vilket kommer underlätta beräkningar senare. Om vi vill uppskatta antalet primtal på ovanstående form mindre än något x, då är det samma som att uppskatta;

$$\pi(x; k, a) = \#\{p \leq x : p \equiv a \pmod{k}\}. \qquad (22)$$

Låt oss nu ta något $z \leq x,\ z \in \mathbb{R}^+$ som vi kommer att bestämma senare. Då kan vi uppskatta (22) genom att dela upp mängden på följande sätt.

$$\begin{aligned}\pi(x; k, a) &= \pi(z; k, a) + \#\{z < p \leq x : p \equiv a \pmod{k}\} \\ &\leq z + \#\{n \leq x : n \equiv a \pmod{k},\ n \not\equiv 0 \pmod{q},\ \forall q \leq z\}. \qquad (23)\end{aligned}$$

Uppskattningen av den andra kardinaliteten som utförs i (23) motsvarar en sållning av alla tal som finns i talföjlden med primtal mindre än z. Det är klart att det är en uppskattning eftersom om

det finns ett sammansatt tal i följden som endast är delbart med primtal större än z så kommer de inte sållas bort. Dock så behöver vi faktiskt inte sålla med avseende på alla primtal mindre än z, bara de som är relativt prima med k. Om till exempel $k = 6$ så får a inte vara delbar med varken 2 eller 3. Som konsekvens av det kommer inga element i talföjlden vara delbar med ks delare heller. Följaktligen ger detta oss att vi bara behöver primtal ur $\mathcal{P} = \{p : (p, k) = 1\}$ och kardinaliteten vi vill uppskatta är då $S(\mathcal{A}, \mathcal{P}, z) = \#\{n \leq x : n \equiv a \pmod{k},\ n \not\equiv 0 \pmod{p},\ p \leq z, (p, k) = 1\}$.

För att kunna använda sållet så behöver vi bestämma $\mathcal{A}$, $\mathcal{A}_d$, X, $f(d)$ samt $f_1(d)$, och R_d. Det är uppenbart att $\mathcal{A} = \{n \leq x : n \equiv a \pmod{k}\}$ och dessutom att $\mathcal{A}_d = \{n \leq x : n \equiv a \pmod{k},\ n \equiv 0 \pmod{d}\}$. Eftersom k och d är relativt prima ger kinesiska restsatsen att det finns ett unikt tal modulo kd som är kongruent med n. Detta delar då upp intervalet $[0, x]$ i $\lfloor x/kd \rfloor$ stycken delintervall där det i varje delintervall finns ett element som ska sållas bort, vilket medför att $\#\mathcal{A}_d = \frac{x}{kd} + O(1)$. Vi erhåller $X = \frac{x}{k}$, $f(d) = d$, och $R_d = O(1)$. För att bestämma $f_1(d)$ använder vi oss av dess definition, att f är multiplikativ, och att varje $d = p_1 p_2 ... p_j$ är kvadratfri. Detta görs på följande sätt:

$$\begin{aligned} f_1(d) &= \sum_{m|d} \mu(m) f(d/m) = \sum_{m|p_1 ... p_j} \mu(m) f((p_1 ... p_j)/m) \\ &= \sum_{b_1=0}^{1} ... \sum_{b_j=0}^{1} \mu(p_1^{b_1} ... p_j^{b_j}) f((p_1 ... p_j)/(p_1^{b_1} ... p_j^{b_j})) = \prod_{h=1}^{j} \sum_{b_h=0}^{1} \mu(p_h^{b_h}) f(p_h / p_h^{b_h}) \\ &= \prod_{h=1}^{j} \left(f(p_h) - 1 \right) = \prod_{h=1}^{j} p_h \left(1 - \frac{1}{p_h} \right) = d \prod_{h=1}^{j} \left(1 - \frac{1}{p_h} \right) = \phi(d). \end{aligned} \tag{24}$$

I (24) ser vi att $f_1(d) = \phi(d)$ där $\phi(d)$ är Eulers ϕ-funktion. Tillsammans med 6.1 ger ovanstående att $S(\mathcal{A}, \mathcal{P}, z) \leq x/(kV(z)) + O(z^2)$.

För att uppskatta $1/V(z)$ tillämpar vi ett lemma från [4, lemma 7.2.3] som säger att

$$f(\overline{P}(z)) V(z) \geq f_1(\overline{P}(z)) \sum_{\delta \leq z} \frac{1}{\tilde{f}(\delta)} \tag{25}$$

där $\overline{P}(z) = \prod_{p \leq z,\ p \notin \mathcal{P}} p$ och $\tilde{f}(d)$ är en fullständigt multiplikativ funktion med $\tilde{f}(p) = f(p)$ för alla primtal p. Eftersom de enda primtal som inte är i $\mathcal{P}$ är de som delar k har vi enligt (25) att

$$kV(z) \geq \phi(k) \sum_{\delta \leq z} \frac{1}{\delta} = \phi(k)(\log z + O(1)) \iff \frac{1}{V(z)} \leq \frac{k}{\phi(k)(\log z + O(1))} \tag{26}$$

där vi har använt oss av partiell summation vid likhetstecknet. Med hjälp av (26) har vi nu att

$$\pi(x; k, a) \leq z + \frac{x}{\phi(k)(\log z + O(1))} + O(z^2) = \frac{x}{\phi(k)(\log z + O(1))} + O(z^2)$$

där likheten gäller eftersom z^1 absorberas av ordonotationen. Genom att likställa de två termerna kan vi bestämma z så att båda termerna har samma vikt. Vi följer förslaget på z i [4, s. 127] och tar $z = (2x/k)^{\frac{1}{2} - \varepsilon_0}$ för något $\varepsilon_0 \in (0, 1)$. Detta val av z ger oss den slutliga uppskattningen

$$\pi(x; k, a) \leq \frac{(2 + \varepsilon)x}{\phi(k) \log(2x/k)}$$

för alla $\varepsilon > 0$ och x större än något $x_0(\varepsilon) > 0$. Resultatet är en variant på Brun-Titchmarshsatsen som bevisades först år 1930 av Titchmarsh [8] med hjälp av Bruns såll. Varianten som vi har härledt ovan är en starkare form än vad om bevisades av Titchmarsh vilket i viss mån beror på skillnaden mellan storleken på feltermerna i Bruns och Selbergs såll. Denna idé kommer utvecklas mer i nästa avsnitt.

7 Diskussion av sållmetoder

Som nämndes i inledningen kan sållmetoder, även om de är användbara, kräva en stor del arbete eller förfining för att spegla de resultat vilka ges av mer analytiska metoder. En uppenbar orsak till

detta är feltermerna kopplade till de sållmetoder som används. Det är ett återkommande tema i sållmetoder som helhet att försöka minska dessa feltermer för att förbättra de slutliga uppskattningarna. Feltermens påverkan på dessa uppskattningar kan enkelt upptäckas i alla tillämpningsdelar i de föregående avsnitten. Den har störst påverkan i det kritiska steget då z bestäms till någon funktion av x vilket i sin tur förenar huvudtermen och feltermen. Av de sållmetoderna vi har redovisat är skillnaden mellan feltermerna störst mellan Eratosthenes generaliserade såll och de andra två sållmetoderna. För att visa påverkan av denna skillnad redovisar vi ett sista exempel.

Låt oss nu återgå till en av sållteorins musor som är att uppskatta storleken på $\pi(x)$. Som nämndes i 4.1 motsvarar problemet en linjär sållning av $\mathcal{A} = \{n \leq x : n \in \mathbb{N}\}$ med $\mathcal{A}_d = \{n \leq x : n \equiv 0 \pmod{d}\}$. Vi tillämpar Eratosthenes såll på problemet genom att välja $y = x$,och $\kappa = 1$. Med detta val av parametrar erhåller vi en felterm av storleksordning $O(x(\log z)^2 \exp(-\log x/\log z))$. Vi jämför den nu emot feltemerna för både Bruns och Selbergs såll, och vad för uppskattning av $\pi(x)$ man kan få från de. Vi tillämpar Bruns såll genom att sätta $b = 1$ och, återigen, $\kappa = 1$. Vi sätter $\lambda = 0.26$, för att minska potensen av feltermen och då får vi att $c_3 \leq 9$ och storleksordningen på feltermen är $O(z^9)$. För att tillämpa Selbergs såll inser vi att $\#\mathcal{A}_d = x/d + O(1)$ vilket medför att feltermen i Selbergs såll av storleksordning $O(z^2)$, precis som i tillämpningen i avsnitt 6.2. Skillnaden mellan felen i Bruns och Selbergs såll kan ses lätt men hur de jämförs med Eratosthenes är inte lika uppenbart. För att förstå skillnaden blandar vi in huvudtermen, vilket är av storleksordning $O(x/\log z)$ för alla tre sållmetoder. Vi kan förena huvud och feltermen för båda Bruns och Selbergs såll genom att sätta $z = x^{1/u}$, med $u > 9$ respektive $u > 2$ och både ger uppskattningen $\pi(x) \ll x/\log x$. Dock så kan vi inte välja z på samma stil när vi tillämpar Eratosthenes allmänna såll. I det fallet måste vi välja $\log z = \log x/C \log\log x$ för något liten konstant C sådan att feltermen och huvudtermen har samma vikt. Med en sådan z erhåller vi från Eratosthenes allmänna såll att $\pi(x) \ll x \log\log x/\log x$. Att både Brun och Selbergs såll når resultat som börjar likna primtalssatsen men inte Eratosthenes generaliserade såll exemplifierar hur feltermerna påverkar kvalitén av slutsatsen som kan dras från de olika metoderna.

Var och en av sållmetoderna har olika faktorer som påverkar storleken på deras feltermer. Det finns dock åtminstone en gemensam faktor mellan de tre metoderna som är värt att nämna, vilket är hur de olika metoderna hanterar Möbiusfunktionen. Som utgångpunkt har vi Eratosthenes allmänna såll som uppskattar Möbiusfunktionen med $|\mu(d)| \leq 1$ då vi kommer till feltermen. Bruns såll gör lite mer med Möbiusfunktionens och (10) kan formuleras på ett sådant sätt på grund av Möbiusfunktionen och en tillämpning av Möbius inverteringsformeln på den och funktionen g. Selbergs såll går ett steg vidare och baserar en stor del av hela sin härledning på en uppskattning av en summa av Möbiusfunktioner. Att sållet som försöker hårdast att undvika Möbiusfunktionen är oftast metoden som ger bäst resultat av de tre påpekar hur svårhanterat Möbiusfunktionen är. I själva verket, enligt Tao [9], är Möbiusfunktionen roten till en ännu mer central utmaning för sållteorin än att minimera feltermer, nämligen paritetsproblemet. Detta problem säger i huvudsak att för specifika typer av mängder kommer sållmetoder antingen att ge övre gränser som är för stora med faktor av minst två eller undre gränser som är triviala. Detta kan ses bero på den binära karaktären hos Möbiusfunktionen vilket leder till att för många kardinaliteter tas bort eller för många läggs tillbaka då man trunkerar Möbiusfunktionen, det vill säga uppskattar den. Moderna sållmetoder såsom de utvecklade av Friedlander och Iwaniec [10] försöker undvika paritetsproblemet för vissa mängder av primtal. Deras metoder har haft succé med att visa att det finns oändlig många primtal på formen $a^2 + b^4$ och även ger en asymptotisk formel för det.

8 Datorimplementation av Eratosthenes såll

I numeriska sammanhang är Eratosthenes såll ett kraftfullt verktyg. Sållet ger oss en metod för att sålla bland eller faktorisera alla tal upp till något N, där förhållandet mellan N och antalat beräkningar som krävs är nära linjärt [1, s.333]. I detta avsnitt utforskas de idéer och algoritmer baserade på Eratosthenes såll som presenteras i *An Improved Sieve of Eratosthenes* av Harald Helfgott [1], samt en datorimplementation av detta skriven i programmeringsspråket Python. Avsnittet består av tre delar. I den första delen beskrivs algoritmernas funktion och deras underliggande matematiska principer. Del två är en metoddel där vi beskriver vilka beslut som gått in i att skriva ett snabbt och välfungerande program baserat på dessa algoritmer. Slutligen visar vi hur pro-

grammet kan användas för att ge resultat om fördelningen av primtalstvillingar och frekvensen av primtalsgap.

8.1 Grundläggande teori och algoritmer

När Eratosthenes först formulerade sitt såll var det, som vi såg i inledningen, på formen av en algoritm. Det är med utgångspunkt i Eratosthenes ursprungliga idé som [1] utvecklar algoritmen för att optimera processen till en effektiv kod som kan leta efter primtal i ett intervall, $[n-\Delta, n+\Delta] \subset \mathbb{R}_+$. Delvis gör [1] detta med algoritmen SimpleSiev, vilken är nära en direkt översättning av Eratosthenes klassiska såll. Vi ser i Algorithm 1 exakt hur sållet implementerats i pseudokod. Algoritmen startar med att sålla bort alla jämna tal från en lista och går sedan igenom resterande tal, kontrollerar om de sållats bort och om inte så tas alla multiplar av talet bort från listan. Resultatet blir en lista av alla primtal upp till ett givet tal N. Denna metoden fungerar väl för små N men vill vi hitta primtal i ett intervall $[n-\Delta, n+\Delta]$ för något stort n och relativt litet Δ så ger algoritmen oss betydligt mer information än vad vi behöver och kräver mer tid och minnesutrymme.

Algorithm 1 En implementation av Eratosthenes såll från [1]

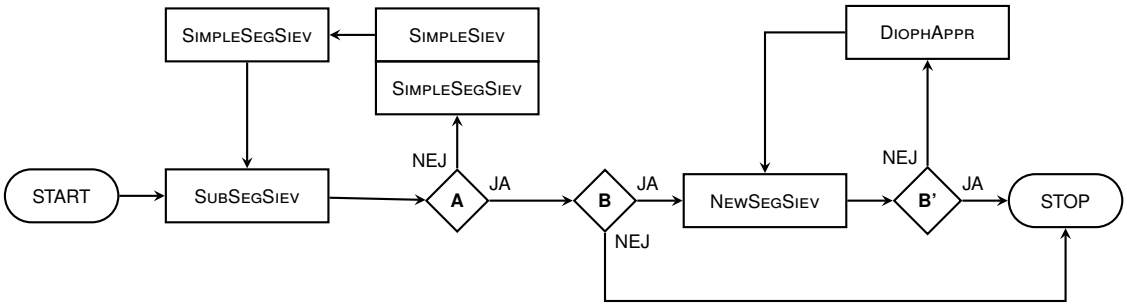
```
1: function SimpleSiev(N)
Ensure: for 1 ≤ n ≤ N, P_n = 1, if n is prime, P_n = 0 otherwise
2:     P_1 ← 0, P_2 ← 1, P_n ← 0 for n ≥ 2 even, P_n ← 1 for n ≥ 3 odd
3:     m ← 3, n ← m · m
4:     while n ≤ N do
5:         if P_m = 1 then
6:             while n ≤ N do
7:                 P_n ← 0, n ← n + 2m                ▷ sieves out multiples ≥ m^2 of m
8:         m ← m + 2, n ← m · m
9:     return P
```

Flaggskeppet i [1], algoritmen NewSegSiev, löser problemet genom att dela upp talen vi vill sålla bort multiplar av i två fall — tal som är så pass små att vi är garanterade att det finns en multipel av dessa i vårt intervall och större tal som har högst en multipel i intervallet. Med den här uppdelningen kan vi behandla de två fallen olika för att spara tid och precis som i Eratosthenes såll behöver vi inte sålla med tal större än $\sqrt{n+\Delta}$. I figur 2 ser vi ett flödesschema av NewSegSiev med den ovannämnda uppdelningen — först sållar algoritmen bort multiplar av små primtal innan kriterium A och sedan går den in i den andra loopen för att hantera multiplar av större tal.

Eftersom vi kan vara säkra på att alla tal mindre än längden av intervallet (det vill säga $m \leq 2\Delta+1$) har en multipel i intervallet så börjar vi att sålla med dessa. Algoritmen gör detta för några fler m än de vi kan vara säkra på har en multipel i intervallet — för alla $m \leq K\Delta$ — bara så att nästa steg med $m > K\Delta$ ska gå smidigt. Vi behöver som bekant inte sålla för alla tal utan det räcker med att vi utesluter multiplar av primtal från intervallet. Processen i NewSegSiev för $p \leq K\Delta$ sköts av funktionen SubSegSiev som delar upp dessa primtal i segment $[M_i', M_i' + \Delta']$, där $M_0' = 1, \Delta' = \lfloor\sqrt{K\Delta}\rfloor$ och $M_i' = M_{i-1}' + \Delta' + 1$, och sållar sedan bort alla multiplar av dessa i vårt intervall. Primtalen i intervallen $[M_i', M_i' + \Delta']$ ges i sin tur av en annan funktion, SimpleSegSiev, som på samma vis sållar segmentet med en lista av primtal $p \leq \lfloor\sqrt{M_i' + \Delta'}\rfloor$. Sista pusselbiten, listan av primtal, erhålls med den tidigarenämnda algoritmen SimpleSiev på klassiskt vis.

Givetvis hade vi kunnat sålla intervallet med $p \leq K\Delta$ utan att segmentera primtalen i intervall och bara genererat alla primtal upp till och med $K\Delta$ direkt med SimpleSiev. Tidskomplexiteten är enligt [1] samma för båda metoderna men anledningen till att vi segmenterar sållet är för att bespara minneskomplexitet — $O(\sqrt{K\Delta} + 2\Delta)$ för SubSegSiev gentemot $O(K\Delta + 2\Delta)$ för SimpleSegSiev.

Nästa steg, huvudalgoritmen i NewSegSiev, kräver mer matematisk eftertanke. Uppgiften som kvarstår för funktionen efter SubSegSiev är att sålla intervallet $[n-\Delta, n+\Delta]$ med resterande primtal, $p \geq K\Delta$ där $K \geq 5/2$. Problemet är löst genom att leta efter alla multiplar av $m \geq K\Delta$



Figur 2: Ett övergripande flödesschema för algoritmen NewSegSiev. Observera att det finns två huvudloopar i algoritmen, en där kriterium A är uppfyllt när intervallet $[n-\Delta, n+\Delta]$ sållats med alla primtal $p \le K\Delta$ och den andra där kriterium B' är uppfyllt då algoritmen sållat för resterande tal upp till $\sqrt{n+\Delta}$. Slutligen är kriterium B i koden identiskt med B' men illustrerar i flödesschemat möjligheten för programmet att aldrig gå in i den andra loopen vilket inträffar då $\Delta > \sqrt{n+\Delta}/K$.

i vårt intervall. Låt ℓm vara en sådan multipel, då ser vi att

$$n-\Delta \le \ell m \le n+\Delta \iff -\frac{\Delta}{m} \le \frac{n}{m} - \ell \le \frac{\Delta}{m} \iff \left\{\frac{n}{m}\right\} \in \left[-\frac{\Delta}{m}, \frac{\Delta}{m}\right] \bmod 1, \tag{27}$$

där $\left[-\frac{\Delta}{m}, \frac{\Delta}{m}\right] \bmod 1 = \bigcup_{k\in\mathbb{Z}} \left[k-\frac{\Delta}{m}, k+\frac{\Delta}{m}\right]$. Med den här presentationen av problemet gör [1] två approximationer – först en Taylorutveckling och därefter en diofantisk approximation.

Den första approximationen syftar till att ersätta hyperbeln $f(m) := \frac{n}{m}$ med en (diskontinuerlig) mängd tangenter till kurvan givna av en Taylorapproximation vid olika punkter m_0,

$$f(m) = f(m_0+r) = \frac{n}{m_0} - \frac{n}{m_0^2}r + O\left(\frac{n}{m_-^3}r^2\right)$$

där $m_- = \min(m, m_0)$. Eftersom hyperbeln planar ut för större m så kan vi approximera större och större intervall av kurvan med samma linjesegment utan att förstora feltermen. Mer specifikt så låter [1] approximera kurvan med tangenter till kurvan på mitten, m_0, av intervallet $[M_i, M_i+2R_i]$ där $M_{i+1} = M_i + 2R_i + 1$ med $M_0 = \lfloor K\Delta \rfloor + 1$ och

$$R_i = \left\lfloor \sqrt{\Delta/(4n)} M_i \right\rfloor.$$

Vi delar in kurvan i segment $[M, M+2R]$ tills vi har täckt alla tal $m \in [\lfloor K\Delta \rfloor + 1, \sqrt{n+\Delta}]$. Orsaken till att [1] väljer att definiera M, m_0 och R som ovan är så att restermen inte övertar storleken på intervallet, med andra ord är resttermen $\lesssim nr^2/m_-^3 \le nR^2/M^3 = \Delta/(4M)$. Tar vi hänsyn till feltermen i problemformuleringen, (27), så har vi omformulerat problemet till att hitta $r \in [-R, R]$ så att $P(r) = (\frac{n}{m_0} - \frac{n}{m_0^2}r) \in [-5\Delta/(4M), 5\Delta/(4M)] \bmod 1$. Helfgotts val av $K \ge 5/2$ fyller nu två syften: $R \ge 1$ så att intervallen vi söker inte är tomma och $5\Delta/(4M) < 1/2$ så att intervallet vi försöker pricka inte är hela $\mathbb{R}$.

Vi kan ställa ytterligare krav på $r \in [-R, R]$ för att slippa gå över hela intervallet. Vad [1] gör är att hitta en approximation för $\alpha_1 := -n/m_0^2$ på formen a/q där a, q är två relativt prima heltal med ett krav på nämnaren, $q \le 2R$. Detta är precis funktionen av en så kallad diofantisk approximation och för att förstå den här processen krävs förkunskaper om kedjebråk som kan hittas i appendix C. Med kedjebråksnotationen från appendix så vill vi omformulera α_1 till ett enkelt kedjebråk, $\langle a_0, a_1, \dots \rangle$. Följer vi algoritmen i [11, sats 21.5] så ges a_0 av $\lfloor \alpha_1 \rfloor$ och om inte α_1 är ett heltal, i vilket fall vi är färdiga, så blir resten $0 < \alpha_1 - a_0 < 1$. Detta ger oss $\xi_1 := 1/(\alpha_1 - a_0) > 1$ och vi kan återupprepa samma steg: låt $a_1 = \lfloor \xi_1 \rfloor$ och $\xi_2 := 1/(\xi_1 - a_1) > 1$. Genom att fortsätta på samma vis får vi en algoritm som genererar ett enkelt kedjebråk.

Vi kan vara säkra på att algoritmen slutar av sig självt eftersom vi utvecklar kedjebråket av ett rationellt tal, $\alpha_1 = -n/m_0^2$, som därav är ett ändligt kedjebråk (se [11, sats 21.5]) men vi vill

sannolikt avbryta processen redan tidigare. Vårt mål är att hitta en rationell approximation till α_1 med nämnare $q \leq 2R$ och eftersom $(q_n)_{n>0}$ är en växande följd så kan vi välja att stoppa algoritmen när $q_n \leq 2R$ men $q_{n+1} > 2R$. Del 2 av sats C.2 garanterar att vår approximation blir bättre för varje iteration och, för sådant n, så är $\left|\alpha_1 - \frac{a}{q}\right| \leq \frac{1}{q \cdot 2R}$. Till sist observerar vi att konvergenterna, p_n, q_n, är relativt prima då del 2 av sats C.1 ger oss att (p_{n-1}, q_{n-1}) är en heltalslösning till den diofantiska ekvationen

$$ax + by = 1 \quad \text{där} \quad a = (-1)^n q_n, \quad b = (-1)^{n-1} p_n,$$

vilket endast är möjligt om högerledet, som här är lika med 1, är en multipel av största gemensamma nämnaren, (a, b), (ett resultat från elementär talteori, se förslagsvis [11, sats 3.1]).

Ovanstående algoritm heter DiophAppr i [1] och beräknar en diofantisk approximation a/q av den ledande koefficienten α_1 i $P(r)$ med kravet på q som vi nämnde tidigare. Algoritmen returnerar täljare och nämnare separat samt passar på att beräkna $a^{-1} \pmod q$ då del 2 av sats C.1 ger oss en möjlighet att räkna ut inversen i termer av konvergenterna.

Målet med att introducera DiophAppr är så att vi kan gå från att leta lösningar till (27) i $r \in [-R, R]$ till att endast leta bland ett urval av heltal. Vi har redan hittat en rationell approximation av den ledande koefficienten, α_1, i $P(r)$. Vi kan också approximera konstantkoefficienten $\alpha_0 := n/m_0$ med bråket $\lfloor \alpha_0 q + 1/2 \rfloor / q := c/q$. Således ser vi att

$$\begin{aligned} \big|q \cdot P(r) - (c + ar)\big| &= \left|\left(\alpha_1 - \frac{a}{q}\right) qr + (\alpha_0 q - c)\right| \leq \left|\alpha_1 - \frac{a}{q}\right| q|r| + |\alpha_0 q - c| \\ &\leq \frac{1}{q \cdot 2R} \cdot qR + \frac{1}{2} = 1 \end{aligned}$$

tack vare noggrannheten av den diofantiska approximationen, hur vi definierade c och att $r \in [-R, R]$. Därav, om $P(r) \in [-5\Delta/(4M), 5\Delta/(4M)]$ mod 1 så medför det att

$$c + ar \in \{-k - 1, -k, ..., k, k + 1\} \bmod q$$

där $k = \lfloor q \cdot 5\Delta/(4M) \rfloor$. Det räcker alltså att sålla med $r \equiv -a^{-1}(c + j) \pmod q$ för heltal $j \in [-k - 1, k + 1]$, där den multiplikativa inversen av $a \pmod q$ tillhandahålls av DiophAppr.

Vi har alltså sett hur [1] först sparar minnesplats genom att segmentera första delen av algoritmen och sedan tid i den andra delen genom att vara selektiv med vilka tal som vi sållar bort multiplar av. I det senare fallet finns som mest en multipel i intervallet, per konstruktion av $K\Delta$, och vi har visat att alla $m = m_0 + r$ med en multipel i intervallet har $r \equiv -a^{-1}(c + j) \pmod q$, för a, c, j definierade ovan. Däremot har vi inte visat det motsatta och det kan vara så att vissa r som genereras är falska lösningar orsakade av Taylor- och den diofantiska approximationen. Därför behöver vi kontrollera att multipeln av m ingår i intervallet, det vill säga att

$$\ell m \in [n - \Delta, n + \Delta] \quad \text{och} \quad \ell m > m. \tag{28}$$

I nästa avsnitt kommer vi se vilka val som gick in i Python-implementationen av pseudokoden samt studera tidsbesparingar och möjliga förbättringar från originalalgoritmen.

8.2 Implementation av algoritmerna i Python

Algoritmerna i [1] presenteras i form av pseudokod som för att kunna användas, måste översättas till något programmeringsspråk. Vår implementation av algoritmerna är skrivna i språket Python. Det var möjligt att översätta pseudokoden mer eller mindre ordagrant, vilket gjordes och resulterade i en första version av programmet. Därefter kunde flera förbättringar av koden göras för att korta ned dess körningstid. Vissa av förbättringarna var möjliga då pseudokoden i [1] är skriven i syfte att tydligt illustrera algoritmerna, och är således inte ämnad till att vara färdig, optimerad kod. Andra förbättringar var språkspecifika och åstadkoms genom att jämföra beräkningstiden hos olika funktioner och metoder i Python, för att sedan implementera de som visade sig vara snabba. Den förbättrade versionen av koden finns bifogad i appendix. Nedan följer, utan inbördes ordning, några av de gjorda förbättringar som har haft större inverkan.

- I den ursprungliga pseudokoden representeras den sållade mängden av en vektor bestående av booleaner, vilken vi har ersatt med en bitsträng. Denna idé föreslås redan i [1] och sparar i första hand minne, som i sin tur kan leda till snabbare beräkningar på grund av bättre användning av cache. Här användes Python-biblioteket *Bitarray*.
- På vissa ställen har det varit möjligt att flytta ut beräkningar utanför loopar så att samma beräkning inte behöver göras flera gånger. Dessutom har flera beräkningar kunnat kortas ned eller skrivas ihop för att undvika temporära variabler.
- I Python kan operatorn `x//y` användas för division utan rest. Denna har visat sig vara snabbare än sammansättningen `floor(x/y)` och har därför fått ersätta den senare där det varit möjligt. På ett liknande sätt har `ceil(x/y)` ersatts med `-(x//-y)`.

En ytterligare förbättring kunde göras genom att titta närmare på (28). För att en lösning m ska vara en äkta lösning måste alla villkor i (28) hålla och detta bör således kontrolleras i algoritmen. Men i själva verket är det överflödigt att kontrollera alla tre villkoren.

Algoritmen konstruerar nämligen multipeln ℓm genom att sätta $\ell := \lfloor (n+\Delta)/m \rfloor$, vilket direkt implicerar att $\ell m \leq n + \Delta$ alltid är uppfyllt. Vidare inspekterar vi villkoret $\ell m > m$, där vi har att

$$\ell m > m \iff \left\lfloor \frac{n+\Delta}{m} \right\rfloor > 1 \iff \left\lfloor \frac{n+\Delta}{m} \right\rfloor \geq 2 \iff (n+\Delta)/2 \geq m.$$

Men m väljs ur intervallet $[M, M+2R]$ så det räcker med att undersöka vilka förhållanden som $M + 2R \leq (n+\Delta)/2$ är uppfyllt under: Vi har att $M \leq \sqrt{n+\Delta}$, $R = \lfloor M\sqrt{\Delta/4n} \rfloor$ samt $\Delta \leq n$ vilket ger oss att $M + 2R \leq 2\sqrt{n+\Delta}$. Detta är i sin tur mindre eller lika med $(n+\Delta)/2$ om $n + \Delta \geq 16$. Att sålla efter primtal i en lista av positiva heltal mindre än 16 är ointressant. Därför kan vi rimligtvis introducera kravet $n + \Delta \geq 16$ i början av NewSegSiev, så att $\ell m > m$ alltid håller och därmed inte behöver testas senare i algoritmen.

Det har följaktligen visat sig att av de tre villkor i (28), är två av dem alltid uppfyllda och behöver därmed inte kontrolleras. Det totala antalet gånger som detta steg utförs är

$$O\left(\Delta \log n + \sqrt{n/\Delta}(\log n)^2\right),$$

enligt [1, s.346] och vi sparar således en betydande mängd tid på att reducera antalet beräkningar här till en tredjedel av det ursprungliga.

För att visa att de förändringar som gjorts i koden, faktiskt har haft inverkan så lät vi utföra tester. Testerna gjordes på en hemdator och jämförde körningstid mellan den första versionen av programmet, mot en senare version där förbättringarna har införts. Som tidigare nämnts så sållar algoritmen fram primtal i ett angivet intervall $[n - \Delta, n + \Delta]$ och dess tillvägagångssätt varierar något beroende på förhållandet mellan Δ och n. Av denna anledning utfördes två stycken tester där detta förhållande sattes till att vara så stort som möjligt respektive så litet som möjligt. I det första testet valdes således $\Delta := n$ och tre mätningar utfördes för $n = 5 \cdot 10^3, 5 \cdot 10^5, 5 \cdot 10^7$. I det andra testet valdes $\Delta := \sqrt[3]{n}$ och n fick anta värdena 10^9, 10^{12} respektive 10^{15}. För alla mätningar visade sig den förbättrade versionen vara minst 10 gånger så snabb som den ursprungliga versionen, dessutom visar mätningarna på en antydan till att denna faktor ökar då n växer. De uppmätta tiderna finnes i tabell 1.

Givetvis är programmet inte perfekt och det finns flera knep som kan utforskas ifall ytterligare förbättring av koden eftertraktas. I den ursprungliga artikeln [1] nämns ett fåtal eventuella förbättringar, här ges ytterligare två stycken.

- Istället för att flera gånger om låta SimpleSiev(M) generera en lista med alla primtal upp till M kan det eventuellt spara tid att generera en godtycklig lista vid start av programmet. Värdet på M överskrider inte $\sqrt{K\Delta + \sqrt{K\Delta}}$, vilket gör denna taktik rimlig. Exempelvis tar en lista med alla primtal upp till en miljard omkring 1GB att spara och kan användas för $n \leq 10^{35}$, så länge som vi håller oss till något mindre interval där $\Delta \leq \sqrt{n}$. Detta handlar självklart om en avvägning mellan hur snabbt det går att läsa in sparad data mot hur snabbt det går att beräkna den från grunden och bör undersökas mer innan idén tillämpas.
- Flera beräkningar kan utföras parallellt, vilket redan föreslås i [1]. I synnerhet kan detta nyttjas i andra delen av NewSegSiev där oberoende beräkningar utförs för varje $j \in [-k -$

$1, k+1]$. Dessa beräkningar är alla enkla och på samma form, och det kan således vara gynnsamt att överlåta dessa till datorns grafikkort. Detta eftersom grafikkortet ofta visar sig snabbare än processorn på att utföra denna typ av uppgifter.

Tabell 1: Uppmätta körningstider för den första, respektive den förbättrade versionen av programmet. Programmet sållade fram alla primtal i ett intervall på formen $[n-\Delta, n+\Delta]$ där $\Delta = n$ för de tre första mätningarna och $\Delta = \sqrt[3]{n}$ för de tre sista. Den förbättrade versionen var snabbare än den första versionen med en faktor på minst 10, för alla mätningar.

Intervall	Körningstid i sekunder	
	Utan förbättringar	Med förbättringar
$[0, 10^4]$	0.0236	0.0018
$[0, 10^6]$	1.9253	0.1002
$[0, 10^8]$	179.7834	9.2633
$[10^9 \pm 10^3]$	0.1489	0.0142
$[10^{12} \pm 10^4]$	2.4071	0.2319
$[10^{15} \pm 10^5]$	78.1001	3.2894

I nästkommande avsnitt presenteras diverse resultat som erhållits ifrån körning av programmet.

8.3 Tillämpningar och resultat

Genom rapportens gång har vi hänvisat till, och uppskattat fördelningen av, ett antal klasser av primtal och deras egenskaper. Nu kommer vi att redovisa några resultat som ovan nämnda Python-program givit. Vi börjar med att räkna primtal i ett intervall och jämföra det med primtalssatsen, detta i syftet att övertyga oss om kodens validitet. Sedan undersöker vi trovärdigheten hos en förmodad fördelning för mängden av primtalstvillingar. Slutligen redovisas frekvensen av primtalsgap, och mönster som uppstår från detta.

8.3.1 Fördelningen av primtal

Vi börjar med att jämföra antalet primtal som hittats med NewSegSiev, mot fördelningen som ges av primtalssatsen, nämligen

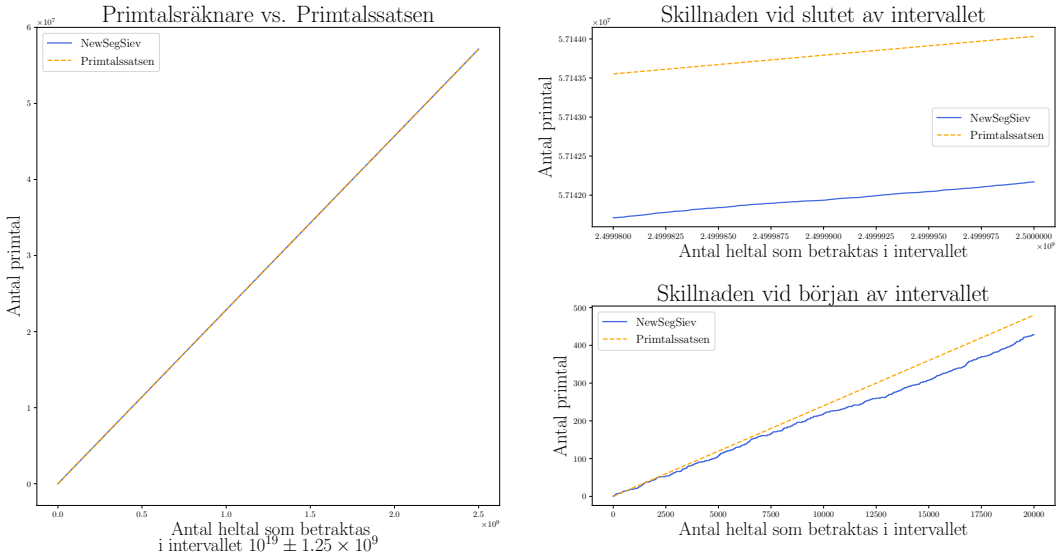
$$\pi(x) \sim \mathrm{Li}(x) = \int_2^x \frac{1}{\log t} dt. \tag{29}$$

För att skapa figur 3 använder vi vår implementering av Helfgotts kod och en enkel räknefunktion.

Som vi ser i vänstra grafen av figur 3 är skillnaden mellan kurvorna nästan osynlig på makronivå, och eftersom primtalssatsen gäller är det troligt att vårt program har hittat det korrekta antalet primtal i intervallet. Att de inte ligger exakt på varandra är huvudsakligen beroende på felet vilket inte visas i (29). Notera att vi normaliserar felet vid början av intervallet eftersom vi betraktar skillnaden $\mathrm{Li}(x) - \mathrm{Li}(n-\Delta)$, $x \in [n-\Delta,\ n+\Delta]$. Att vår kod hittar ungefär 100 färre primtal än förväntat efter 20 000 heltal och ungefär 2000 färre primtal efter 2.5×10^9 heltal är mycket rimligt då felet i (29) kan som bäst vara $O(2\Delta)^{1/2+\varepsilon})$, $\varepsilon > 0$, under antagandet att Riemannhypotesen stämmer [12, kap. 5].

Värdet på n valdes till 10^{19} för att detta är stort nog för att vara intressant men inte så stort att programmets körningstid blir orimlig[1]. Vi valde Δ till 1.25×10^9 får att nyttja de tidbesparingar som fås av att gå in i den andra loopen i NewSegSiev. Slutligen valde vi K till 2.5 för att det är minsta möjliga värde per konstruktion av algoritmen. Vi kommer nu att studera den data som programmet gav efter körning med ovanstående parametervärden.

[1] Att n inte valdes större i detta fallet är på grund av körningstiden för koden.



Figur 3: I figuren till vänster redovisas antalet primtal som förväntas enligt primtalssatsen, *orange*, och antalet primtal som hittades enligt NewSegSiev, *blå*, då $n = 10^{19}$, $\Delta = 1.25 \cdot 10^9$, och $K = 2.5$. Notera att kurvorna ligger nästan på varandra. I grafen till höger redovisas inzoomade versioner av grafen till vänster för att visa skillnaden mellan vår mätning och det som förväntas vid början av intervallet, där algoritmen returnerar kring 100 färre primtal än förväntat, och vid slutet av intervallet, där algoritmen returnerar kring 2000 färre. Dock att kurvorna inte ligger på varandra är väntat eftersom det finns ett fel i 29 vilket inte visas.

8.3.2 Fördelningen av primtalstvillingar

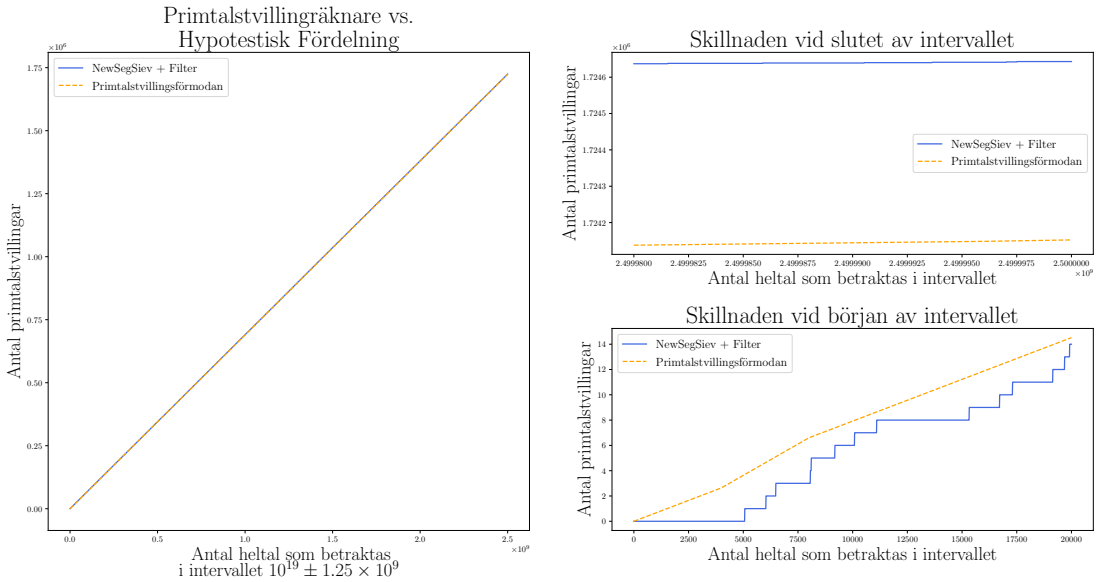
Med hjälp av de primtal vi hittade i 8.3.1, vänder vi oss nu till primtalstvillingar. Det finns ingen sats för primtalstvillingar motsvarande primtalssatsen, dock så finns det en förmodan framlagd av Hardy och Littlewood [13, Förmodan B]. Den lyder

$$\pi_2(x) \sim 2\mathrm{C}_2 \cdot \mathrm{Li}_2(x) = 2\mathrm{C}_2 \int_2^x \frac{1}{(\log t)^2} dt \tag{30}$$

där $\mathrm{C}_2 = 0.66016...$ är primtalstvillingskonstanten [14]. Jämför vi den hypotetiska fördelningen emot antalet primtalstvillingar vi har genererat, så får vi figur 4.

Datan i figur 4 ger oss några intressanta insikter angående primtalstvillingar. Den första är att datan ger stöd till den hypotetiska fördelningen av primtalstvillingar. Kurvorna ligger mycket nära varandra och stödjer därmed hypotes 30, eftersom vi nu litar på vårt program. Skillnaden mellan det förväntade antalet primtalstvillingar och antalet som hittades är som mest ungefär 500, och hittas vid slutet av intervallet. I sammanhanget av ett intervall av längd 2.5×10^9 så är en skillnad av 500 ekvivalent med en avvikelse av 2×10^{-7} mer primtalstvillingar per heltal än vad som förväntades. Att antalet primtalstvillingar blir större än det hypotetiskt förväntade antalet vid slutet på intervallet beror möjligen på att vi avrundar C_2 till 0.66 i kombination med att vi normaliserar felet vid början av intervallet precis som i föregående tillämpning. Dock så kunde avrundningen av C_2 tillsammans med normaliseringen av felet också ha ökat antalet förväntade primtalstvillingar till mer än vad det egentligen skulle vara. I sin tur leder detta till att förmodan verkar stödjas av vår implementering mer än vad den borde.

Den andra insikten som grafen ger är kopplad till avsnitt 4.3 där vi visade att primtalstvillingarna är en relativt liten andel av primtalen. Låt oss undersöka detta genom att jämföra figur 3 och 4. Vi ser direkt att bland de första 20 000 heltalen som betraktas, så finns det drygt 400 primtal och 14 primtalstvillingar. Detta är en faktor på ungefär 30, och tittar vi istället vid slutet av intervallet ser vi att denna faktor ökat till ungefär 33.



Figur 4: I grafen till vänster redovisas antalet primtalstvillingar som förväntas enligt den hypotetiska fördelningen, *orange*, och de primtalstvillingar som hittades enligt NewSegSiev, *blå*, då $n = 10^{19}$, $\Delta = 1.25 \times 10^9$, och C_2 har avrundats till 0.66. Notera återigen att kurvorna ligger nästan på varandra. I graferna till höger så redovisas inzoomade versioner av grafen till vänster. Översta grafen till höger visar skillnaden mellan antalet primtalstvilling vi har hittat och antalet som förväntas vid slutet av intervallet, där algoritmen returnerar 500 fler primtalstvillingar. Efter de första 20 000 heltalen i intervallet så har algoritmen genererat nästan det exakta antalet primtalstvillingar som förväntas.

Vi kan inte använda vår kod som bevis för den förmodade fördelningen av primtalstvillingar. Dock så ger den oss en känsla för om fördelningen verkar rimlig på detta intervall, vilket den gör. Vi kan också få en känsla för hur få primtalstvillingar det finns jämfört med antalet primtal, vilket kan göra det lättare att acceptera Bruns sats (sats 5.2).

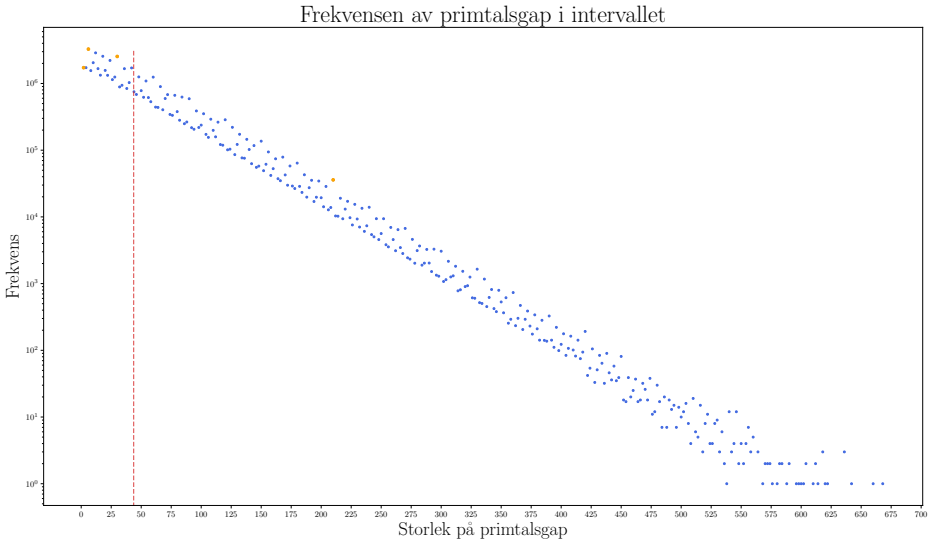
8.3.3 Frekvens av primtalsgap

Vi fortsätter undersöka och illustrera egenskaper hos vår genererade data, genom att studera frekvensen av primtalsgap. Ett primtalsgap är avståndet mellan ett primtal och det efterföljande primtalet. Det går att visa, med hjälp av primtalssatsen, att det förväntade avståndet mellan ett primtal p_n och nästa, p_{n+1}, är $\log p_n$. Vi jämför nu detta med den data som vi genererat, se figur 5.

I figur 5 får vi några insikter kring beteendet av primtalsgap i vårt intervall. En kanske överaskande aspekt av grafen är hur rak den är efter logaritmering av y-axeln. Detta antyder att frekvensen på primtalsgap minskar exponentiellt då dess längd ökar. Vi ser att det vanligaste storleken på ett primtalsgap i intervallet är 6 och att det störta gapet är 668. En detalj som är svår att utläsa från figuren är att det för varje jämnt tal upp till och inklusive 560, finns ett primtalsgap i intervallet med denna storlek. Enligt primtalssatsen är den förväntade storleken på ett primtalsgap i intervallet ungefär 43.7491, och om vi beräknar den genomsnittliga storleken på primtalsgap i vårt intervall så får vi det till 43.7505. Återigen stödjer detta resultat legitimiteten av vår kod eftersom datan reflekterar vad som har bevisats i primtalssatsen.

Om vi återgår till primtalstvillingar så säger figuren även något om dem. Dessa representeras av primtalsgap med storlek 2, vilket är det sjunde vanligaste primtalsgapet och utgör 3 procent av alla gap i intervallet.

Till sist ger figur 5 intuition för en sats som omnämndes kortfattat i inledningden och säger att det finns oändligt många par av primtal med maximalt 600 steg emellan sig. Detta betyder attdet



Figur 5: I figuren redovisas frekvensen av primtalsgap i intervallet $[n - \Delta, n + \Delta]$, då $n = 10^{19}$ och $\Delta = 1.25 \cdot 10^9$. Notera att y-axeln är logaritmerad på grund av den höga frekvensen av små primtalsgap. De orange prickarna anger frekvensen på primtalsgap vars storlek är en primorial (en produkt av de första n primtalen). Röda linjen markerar den genomsnittliga storleken på primtalsgap i vårt intervall vilket är 43.705.

även för stora primtal finns så finns det relativt små primtalsgap. I vårt intervall är nästan alla primtalsgap mindre än 600, trots att talen i intervallet är stora. Satsen bevisades med hjälp av nya sållmetoder, specifikt Goldston-Pintz-Yıldırım-metoder utvecklades som år 2005 [15]. I Maynards artikel används vikter liknande de som finns i Selbergs såll, fast på en ännu mer allmän form. Sedan 2005 så har 600 steg reducerades ned till 246. Ett mål med att fortsätta minska längden är att en dag reducera det till 2 och därmed bevisa primtalstvillingshypotesen.

Såsom mycket annat angående primtal, finns det även en förmodan kopplad till dessa frekvenser ovan och vilket primtalsgap som är mest frekvent upp till en given gräns. Som vi ser i figur 5 är 6 (den andra orange pricken) den mest frekventa storleken på primtalsgap i vårt intervall, och speciellt med talet 6 är att det är produkten av de första två primtalen. Vi ser också att 30 (den tredje orange pricken) vilket är produkten av de första 3 primtalen, hoppar upp lite från översta delen av bandet, och på samma vis är 210 (den fjärde orange pricken) nästa tal med detta beteende. Förmodan är då att varje *primorial*, produkten av de första m primtalen, någon gång blir "hoppmästare". Förmodan har undersökts direkt med hjälp av numeriska uppskattningsmetoder [16] som har lyckats visa att 6 är hoppmästaren fram till $\approx 1.74 \cdot 10^{35}$ och att 30 sedan tar över titeln fram till $\approx 10^{425}$.

Referenser

1. Helfgott HA. An Improved Sieve Of Eratosthenes. Mathematics Of Computation 2020; 89(321):333–50. DOI: 10.1090/mcom/3438
2. Chen JR. On the representation of a large even integer as the sum of a prime and the product of at most two primes. Kexue Tongbao 1966; 11(9):385–6
3. Maynard J. Small gaps between primes. Annals of Mathematics 2015; 181(1):383–413. DOI: 10.4007/annals.2015.181.1.7
4. Cojocaru AC, Murty MR. An Introduction to Sieve Methods and Their Applications. Cambridge: Cambridge University Press, 2005
5. Friedlander J, Iwaniec H. Opera de cribro. Providence, Rhode Island: American Mathematical Society, 2010
6. Tenenbaum G. Introduction à la théorie analytique et probabiliste des nombres. (Français) [Introduction to Analytic and Probabilistic Number Theory]. 3. utg. Providence, Rhode Island: American Mathematical Society, 2015
7. O'Connor JJ, Robertson EF. Maths history, Viggo Brun. 2014 Mar. [citerad 2021-05-14]. Hämtad från: https://mathshistory.st-andrews.ac.uk/Biographies/Brun/
8. Titchmarsh EC. A divisor problem. Rend. Circ. Mat. Palermo 1930; 54:414–29
9. Tao T. 254A, Notes 1: Elementary multiplicative number theory. 2014 Nov. [citerad 2021-05-14]. Hämtad från: https://terrytao.wordpress.com/2014/11/23/254a-notes-1-elementary-multiplicative-number-theory/#more-7855
10. Friedlander J, Iwaniec H. The Polynomial X2 + Y4 Captures Its Primes. Annals of Mathematics 1998 Nov; 148(3):945–1040. DOI: 10.2307/121034
11. Lindahl LÅ. Elementär talteori. Uppsala, 2012
12. Hildebrand AJ. Introduction to Analytic Number Theory. 2005. [citerad 2021-05-14]. Hämtad från: http://www.math.uiuc.edu/~hildebr/ant
13. Hardy G, Littlewood J. Some problems of 'Partitio numerorum'; III: On the expression of a number as a sum of primes. Acta Mathematica 1923; 44:1–70. DOI: 10.1007/BF02403921
14. Inc. OF. The On-Line Encyclopedia of Integer Sequences. 2019. [citerad 2021-05-14]. Hämtad från: https://oeis.org/A005597
15. Goldston DA, Pintz J, Yildirim CY. Primes in Tuples I. Annals Of Mathematics 2009; 170:819–57
16. Odlyzko A, Rubinstein M, Wolf M. Jumping Champions. Experimental Mathematics 1999; 8(2):107–9

A Ordonotation

I den här uppsatsen gör vi flitig användning av ordonotation för att beteckna olika asymptotiska gränser. För $D \subset \mathbb{C}$ och två funktioner, $f : D \to \mathbb{C}$ och $g : D \to \mathbb{R}_+$ skriver vi

$$f(x) = O(g(x)) \quad \text{om det finns } A > 0 \text{ så att } \quad \big|f(x)\big| \leq Ag(x), \forall x \in D.$$

Omväxlande skriver vi även $f(x) \ll g(x)$ med samma betydelse som ovan. Om vi har att $f(x) \ll g(x)$ och $g(x) \ll f(x)$ så skriver vi $f(x) \asymp g(x)$. Sist så kallar vi $f(x)$ och $g(x)$ för *asymptotiskt ekvivalenta* och skriver $f(x) \sim g(x)$ om $g(x)$ är nollskild för alla $x > x_0$, för någon konstant x_0, och $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1$.

B Några användbara resultat

B.1 Partiell summation

Följande sats är hämtad från [4, Sats 1.3.1] dock så ger vi lite mer förklaring i beviset.

Sats B.1. *Låt $c_1, c_2, ...$ vara en foljd av komplexa tal och sätt*

$$S(x) := \sum_{n\leq x} c_n.$$

Låt n_0 vara ett fixt positivt heltal. Om $c_j = 0$ för $j < n_0$ och $f : [n, \infty) \longrightarrow \mathbb{C}$ har en kontinuerlig derivata i $[n_0, \infty)$, då för något heltal $x > n_0$ har vi att

$$\sum_{n\leq x} c_n f(n) = S(x)f(x) - \int_{n_0}^{x} S(t)f'(t)dt.$$

Bevis. Vi bevisar satsen genom att först inse att

$$c_n = S(n) - S(n-1).$$

Vi kan därför skriva om vänsterleden i satsen till

$$\begin{aligned}
\sum_{n\leq x} c_n f(n) &= \sum_{n\leq x} (S(n) - S(n-1))f(n) \\
&= \sum_{n\leq x} S(n)f(n) - \sum_{n\leq x-1} S(n)f(n+1) \\
&= S(x)f(x) + \sum_{n\leq x-1} S(n)f(n) - \sum_{n\leq x-1} S(n)f(n+1) \\
&= S(x)f(x) - \sum_{n\leq x-1} S(n)(f(n+1) - f(n)) \\
&= S(x)f(x) - \sum_{n\leq x-1} S(n) \int_{n}^{n+1} f'(t)dt \\
&= S(x)f(x) - \int_{n_0}^{x} S(t)f'(t)dt
\end{aligned}$$

Att vi kan flytta in $S(x)$ i integralen beror på att $S(x)$ är en stegfunktion och är konstant på intervaller på formen $[n, n+1)$. □

B.2 Summan av primtalsreciproker

Avsnitt 4.1 introducerar dimensionen av ett såll, κ, som ska uppfylla (5). Vi visar här ett resultat tillskrivet Tjebysjov, hämtat ur [4, Sats 1.4.3], som hjälper till vid kontroll av detta.

Sats B.2.

$$\sum_{p\le n}\frac{\log p}{p}=\log n+O(1).$$

Bevis. Nyckeln till beviset av ovanstående sats i [4] är två observationer om fakultetfunktionen, $n!=1\cdot 2\cdot ...\cdot n$. Först vill vi omformulera $n!$ som en produkt av primtalspotenser. Eftersom det finns $\lfloor n/p^\alpha\rfloor$ multiplar av p^α som är mindre än eller lika med n och inga sådana multiplar av p^α för $\alpha>\log n/\log p$, ser vi att den bidragande faktorn för varje $p\le n$ kan skrivas som

$$\prod_{\substack{p^\alpha\parallel k\\ k\le n}}p^\alpha=\prod_{p^\alpha\le n}p^{\lfloor n/p^\alpha\rfloor}=p^{\lfloor n/p\rfloor+\lfloor n/p^2\rfloor+...+\lfloor n/p^{\lfloor \log n/\log p\rfloor}\rfloor}$$

där $p^\alpha\parallel k$ betyder att α är den största heltalsexponenten så att p^α delar k. Detta ger oss att

$$\log n!=\sum_{p\le n}\left(\left\lfloor\frac{n}{p}\right\rfloor+\left\lfloor\frac{n}{p^2}\right\rfloor+...+\left\lfloor n/p^{\lfloor\log n/\log p\rfloor}\right\rfloor\right)\log p$$

vari en restterm kan urskiljas och uppskattas till

$$\sum_{p\le n}\left(\left\lfloor\frac{n}{p^2}\right\rfloor+...+\left\lfloor n/p^{\lfloor\log n/\log p\rfloor}\right\rfloor\right)\log p\le\sum_{p\le n}\left(\frac{n}{p^2}\sum_{k=0}^{\infty}\frac{1}{p^k}\right)\log p=n\sum_{p\le n}\frac{1}{p^2}\cdot\frac{\log p}{1-1/p}\ll n.$$

Den första observationen är alltså att

$$\log n!=n\sum_{p\le n}\frac{\log p}{p}+O(n). \tag{31}$$

Den andra observationen ser vi genom att utföra en partiell summation,

$$\log n!=\log\Big(\prod_{k\le n}k\Big)=\sum_{k\le n}1\cdot k=\lfloor n\rfloor\log n-\int_1^n\frac{\lfloor t\rfloor}{t}\mathrm{d}t$$

och att $\lfloor t\rfloor=t+O(1)$ ger att detta är lika med

$$(n+O(1))\log n-\int_1^n\frac{t}{t}\mathrm{d}t+O(1)\int_1^n\frac{1}{t}\mathrm{d}t=n\log n-n+O(\log n). \tag{32}$$

Sätter vi samman observationerna, (31) och (32) får vi

$$n\sum_{p\le n}\frac{\log p}{p}+O(n)=n\log n-n+O(\log n)$$

$$n\sum_{p\le n}\frac{\log p}{p}=n\log n+O(n)$$

vilket efter division med n ger det önskade resultatet. □

När vi hanterar Eratosthenes generaliserade såll, sats 4.2, och Bruns såll, sats 5.1, resulterar det ofta i att vi vill uppskatta $W(z)$-funktionen. En användbar sats för detta är [4, Sats 1.4.4] som säger

Sats B.3.

$$\sum_{p\le n}\frac{1}{p}=\log n+O(1).$$

Bevis. Låt

$$c_k := \begin{cases} \log p & \text{för } k = p \\ 0 & \text{annars} \end{cases}$$

då får vi att $S(x) = \sum_{k \leq x} c_k$ är lika med summan i sats B.2. Om vi använder föregående sats efter en partiell summation så ser vi att

$$\sum_{p \leq n} \frac{1}{p} = \frac{S(n)}{\log n} + \int_2^n \frac{S(t)}{t(\log t)^2} \mathrm{d}t = \frac{\log n + O(1)}{\log n} + \int_2^n \frac{\log t}{t(\log t)^2} \mathrm{d}t + \int_2^n \frac{O(1)}{t(\log t)^2} \mathrm{d}t$$
$$= O(1) + [\log \log t]_2^n + O(1) \left[-\frac{1}{\log t} \right]_2^n = \log \log n + O(1).$$

□

B.3 Asymptotiskt beteende för W(z), ett specialfall

Följande sats beskriver det asymptotiska beteendet hos $W(z)$ i specialfallet då $\omega(2) = 1$ och $\omega(p) = 2$ för primtal $p > 2$. Resultatet används i (5.2) samt (4.3) och idéerna till beviset är hämtade från beviset av *Merten's Theorem* i [4, kap.5.2].

Sats B.4. *Antag att $\omega(2) = 1$ och $\omega(p) = 2$ för alla primtal $p > 2$, och definiera*

$$W(z) := \prod_{p<z} \left(1 - \frac{\omega(p)}{p}\right).$$

Då gäller det att

$$W(z) \gg \exp\left(-\sum_{2<p<z} \frac{2}{p}\right), \tag{33}$$

då $z \to \infty$.

Bevis. Låt oss betrakta

$$-\log(W(z)) = \sum_{p<z} -\log\left(1 - \frac{\omega(p)}{p}\right) = \sum_{p<z} \log\left(1 + \frac{\omega(p)}{p - \omega(p)}\right).$$

Användning av antagandet på ω implicerar att ovanstående högerled är lika med

$$\log 2 + \sum_{2<p<z} \log\left(1 + \frac{2}{p-2}\right) \leq \log 2 + \sum_{2<p<z} \frac{2}{p-2}. \tag{34}$$

där olikheten håller eftersom $\log(1+x) \leq x$, för alla $x \geq 0$. Betrakta nu summan i högerledet och gör omskrivningen

$$\sum_{2<p<z} \frac{2}{p-2} = \sum_{2<p<z} \frac{2}{p} + \sum_{2<p<z} \frac{2}{p(p-2)}.$$

Här gäller det att den andra summan konvergerar då $z \to \infty$ och kan ses genom att betrakta

$$\sum_{2<p<z} \frac{2}{p(p-2)} < 2 \sum_{2<p<z} \frac{1}{p^2} < 2 \sum_{n=1}^{\infty} \frac{1}{n^2},$$

där summan till höger konvergerar och är lika med $\pi^2/3$. Om vi nu återgår till högerledet i (34) får vi slutligen att

$$-\log(W(z)) \gg \sum_{2<p<z} \frac{2}{p},$$

vilket implicerar satsen. □

B.4 Ett lemma angående multiplikativa funktioner

Följande lemmat är hämtad från [4, Lemma 7.2.2], där beviset utelämnas.

Lemma B.5. *Låt f vara en multiplikativ funktion med d_1, d_2 positiva, kvadratfria heltal. Då*

$$f([d_1, d_2]) \cdot f((d_1, d_2)) = f(d_1)f(d_2)$$

Bevis. Vi delar upp beviset i två fall där d_1 och d_2 är antingen relativt prima eller inte. Då d_1 och d_2 är relativt prima är $(d_1, d_2) = 1$ och $[d_1, d_2] = d_1 d_2$. Detta medför att

$$f([d_1, d_2]) \cdot f((d_1, d_2)) = f(d_1 d_2)f(1) = f(d_1)f(d_2)$$

där sista likheten gäller för att d_1 och d_2 är relativt prima.

Då d_1 och d_2 inte är relativt prima är både deras lägsta gemensamma multipel och största gemensamma delare också kvadratfria. Genom att faktorisera minsta gemensamma multipeln $[d_1, d_2]$ och största gemensamma delaren (d_1, d_2) som

$$[d_1, d_2] = p_1 p_2 ... p_m$$
$$(d_1, d_2) = q_1 q_2 ... q_l$$

så får vi med användning av formeln $[d_1, d_2] \cdot (d_1, d_2) = d_1 d_2$ att

$$f([d_1, d_2]) \cdot f((d_1, d_2)) = f(p_1)f_(p_2)...f(p_m)f(q_1)f(q_2)...f(q_l) = f(d_1)f(d_2)$$

där i sista likheten arrangerar om de primtal som behövs för att få tillbaka d_1 och d_2. Detta kan vi göra på grund av formeln som nämndes tidigare. □

B.5 En variant på Möbius inverteringsformel

Formuleringen av satsen är hämtad från [4, Sats 1.2.3], där beviset utelämnas.

Sats B.6. *Låt $\mathcal{D}$ vara en sluten delare mängd av naturliga tal (det vill säga, om $d \in \mathcal{D}$ och $d' \mid d$, då är $d' \in \mathcal{D}$. Låt f och g vara två komplexvärda funktioner på de naturliga talen. Om*

$$f(n) = \sum_{\substack{n|d \\ d\in\mathcal{D}}} g)(d)$$

då

$$g(n) = \sum_{\substack{n|d \\ d\in\mathcal{D}}} \mu\left(\frac{d}{n}\right) f(d)$$

och motsatsen också gällller (om man antar att alla serier är absolutkonvergenta).

Bevis. Vi bevisar bara framåt riktningen, eftersom omvändningen bevisas på likadant sätt. Låt

$$f(n) = \sum_{\substack{n|d \\ d\in\mathcal{D}}} g(d)$$

då är

$$\sum_{\substack{n|d \\ d\in\mathcal{D}}} \mu\left(\frac{d}{n}\right) f(d) = \sum_{\substack{n|d \\ d\in\mathcal{D}}} \mu\left(\frac{d}{n}\right) \sum_{\substack{d|d' \\ d'\in\mathcal{D}}} g(d').$$

Vi nu använder att

$$n \mid d \implies \exists l \in \mathbb{N} : nl = d$$

och likadant för d och d'. Vi får att summan kan då skrivas på formen

$$\sum_{\substack{nl=d \\ d\in\mathcal{D}}} \mu(l) \sum_{\substack{dk=d' \\ d'\in\mathcal{D}}} g(d') = \sum_{\substack{n|d' \\ d'\in\mathcal{D}}} g(d') \sum_{k|\frac{d'}{n}} \mu(k). \tag{35}$$

där vi nu summerar över l på yttersta summan i vänsterled. Enligt den första egenskapen som redovisas i 2.2, så är inre summan i (35) antingen 1 eller 0 beroende på om $d'/n = 1$ eller inte. Detta medför att

$$\sum_{\substack{n|d \\ d\in\mathcal{D}}} \mu\left(\frac{d}{n}\right) f(d) = g(n)$$

eftersom $n \in \mathcal{D}$ på grund av att $\mathcal{D}$ är en sluten delare mängd. □

C Kedjebråk

I avsnitt 8.1 går vi igenom en implementering av Eratosthenes såll och gör då bruk av en *diofantisk approximation*, [1, algoritm 4]. För att förstå det här steget krävs först lite förkunskaper om kedjebråk.

Ett *ändligt kedjebråk* definieras i [11, definition 20.1] som

$$\langle a_0, a_1, ..., a_n\rangle := a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots \; a_{n-2} + \cfrac{1}{a_{n-1} + \cfrac{1}{a_n}}}}}$$

där $a_0, a_1, ..., a_n$ är reella tal och $a_i > 0$ för $i > 0$. I den här uppsatsen berör vi endast särfallet då $a_0, a_1, ..., a_n$ är heltal vilket då kallas för ett *enkelt kedjebråk*.

En omedelbar utmaning är att, givet ett kedjebråk $\langle a_0, a_1, ..., a_n\rangle$, beräkna $\langle a_0, a_1, ..., a_{n+1}\rangle$ utan att räkna om hela kedjebråket från term a_{n+1} till a_0. Lösningen på det här problemet är så kallade *konvergenter*, [11, Definition 20.4], och definieras som ett par (p_n, q_n), där

$$p_{-2} = 0, p_{-1} = 1, p_n = a_n p_{n-1} + p_{n-2} \text{ då } n \geq 0$$
$$q_{-2} = 1, q_{-1} = 0, q_n = a_n q_{n-1} + q_{n-2} \text{ då } n \geq 0,$$

och deras kvot, $c_n := p_n/q_n$, $n \geq 0$. Observera att definitionen ger oss att $q_0 = 1$ och $(q_n)_{n=1}^N$ är en strikt växande följd av heltal om motsvarande kedjebråk är enkelt. Anledningen till att vi introducerar konvergenter ges av följande sats, [11, Sats 20.5i) och ii)],

Sats C.1. *Låt $(a_n)_{n=0}^N$ vara en följd av reella tal där $a_i > 0$ för $i > 0$ och (p_n, q_n) är deras respektive konvergenter. Då gäller att*

1. $\langle a_0, a_1, ..., a_n\rangle = c_n$ *för alla* $n \geq 0$,
2. $p_n q_{n-1} - p_{n-1} q_n = (-1)^{n-1}$ *för alla* $n \geq -1$.

Bevis. Vi följer här beviset från [11].

(i): För att visa det första påståendet använder vi ett induktionsargument. Vi ser att påståendet håller för $n = 0$ ty $\langle a_0\rangle = a_0$ och $c_0 = p_0/q_0 = (a_0 \cdot 1 + 0)/1 = a_0$.

Antag nu att påståendet gäller för kedjebråk, $\langle a_0, a_1, ..., a_n\rangle$, med $n+1$ stycken element. Vi ska visa att detta medför att $\langle a_0, a_1, ..., a_n, a_{n+1}\rangle = c_{n+1}$. Vi gör detta genom att skriva om det sista elementet i kedjebråket så att vi får samma kedjebråk fast med $n + 1$ stycken element,

$$\langle a_0, a_1, ..., a_n, a_{n+1}\rangle = \langle a_0, a_1, ..., a_n + 1/a_{n+1}\rangle. \tag{36}$$

Låt a'_n beteckna det nya kedjebråkets $(n + 1)$:te element. Det nya kedjebråket har då samma konvergenter upp till index n. Vi skriver den sista konvergenten som $c'_n = p'_n/q'_n$. Alltså, per

induktionsantagandet, har vi

$$\begin{aligned} c'_n = \frac{p'_n}{q'_n} &= \frac{\left(a_n + 1/a_{n+1}\right) p_{n-1} + p_{n-2}}{\left(a_n + 1/a_{n+1}\right) q_{n-1} + q_{n-2}} \\ &= \frac{a_{n+1}\left(a_n p_{n-1} + p_{n-2}\right) + p_{n-1}}{a_{n+1}\left(a_n q_{n-1} + q_{n-2}\right) + q_{n-1}} \\ &= \frac{a_{n+1}p_n + p_{n-1}}{a_{n+1}q_n + q_{n-1}} \end{aligned}$$

vilket är definitionen av $c_{n+1} = p_{n+1}/q_{n+1}$. Därav ger (36) att

$$\langle a_0, a_1, ..., a_n, a_{n+1}\rangle = c_{n+1}$$

Det följer nu av induktion att $\langle a_0, a_1, ..., a_n\rangle = c_n$ för alla $n \geq 0$.

(ii): För det andra påståendet gör vi ett till induktionsbevis. Vi börjar den här gången med att kolla basfallet $n = -1$,

$$p_{-1}q_{-2} - p_{-2}q_{-1} = 1 \cdot 1 - 0 \cdot 0 = (-1)^{-2}$$

vilket stämmer med påståendet vi vill visa.

Härnäst antar vi att $z_n := p_n q_{n-1} - p_{n-1}q_n = (-1)^{n-1}$ för något n och vill visa att $z_{n+1} = (-1)^n$. Detta är lätt att se med hjälp av den rekursiva definitionen av konvergenterna:

$$z_{n+1} = p_{n+1}q_n - p_n q_{n+1} = (a_n p_n + p_{n-1})q_n - p_n(a_n q_n + q_{n-1}) = p_{n-1}q_n - p_n q_{n-1} = -z_n.$$

Således ger induktionsantagandet att $z_{n+1} = (-1) \cdot (-1)^{n-1}$ vilket skulle visas och påståendet följer av induktion. □

Sats C.1 säger oss något om hur konvergenter och kedjebråk är relaterade. Vi kan nu använda den kunskapen för att säga hur väl vi kan approximera kedjebråk genom trunkering [11, sats 20.9].

Sats C.2. *Låt $(a_n)_{n=0}^N$ vara en följd av heltal med positiva termer då $n > 0$ och med konvergenter $p_n/q_n = c_n$. Kedjebråket $\xi = \langle a_0, a_1, \ldots, a_N\rangle$ uppfyller då för alla $n \geq 0$*

1. $c_{2n} < \xi < c_{2n+1}$ *för* $2n + 1 < N$,
2. $|\xi - c_n| < \frac{1}{q_n q_{n+1}}$.

Bevis. (i): Vi börjar med att se att (ii) i sats C.1 ger oss, för $n \geq 1$,

$$\begin{aligned} p_n q_{n-1} - p_{n-1}q_n &= (-1)^{n-1} \\ \frac{p_n}{q_n} - \frac{p_{n-1}}{q_{n-1}} &= \frac{(-1)^{n-1}}{q_{n-1}q_n} \end{aligned} \tag{37}$$

där sista vänsterledet är lika med $c_n - c_{n-1}$. Då $q_{n-1}q_n$ är en produkt av två positiva tal medför detta att $c_{2n} < c_{2n-1}$ för $n \geq 1$. Vidare, för $n \geq 2$, får vi

$$\begin{aligned} (c_n - c_{n-1}) + (c_{n-1} - c_{n-2}) &= \frac{(-1)^{n-1}}{q_{n-1}q_n} + \frac{(-1)^{n-2}}{q_{n-2}q_{n-1}} \\ c_n - c_{n-2} &= (-1)^n \frac{q_n - q_{n-1}}{q_{n-2}q_{n-1}q} = (-1)^n A_n \end{aligned}$$

där A_n är en positiv konstant eftersom $(q_n)_{n=0}^N$ är en strikt växande följd av positiva tal. Alltså har vi att delföljden $(c_{2n})_{n=0}^{\lfloor N/2\rfloor}$ är strikt växande och delföljden av udda index, $(c_{2n+1})_{n=0}^{\lfloor (N-1)/2\rfloor}$, är strikt avtagande.

Sammantaget ger detta oss att

$$c_0 < c_2 < c_4 < ... < c_N = \xi < ... < c_5 < c_3 < c_1$$

vilket bevisar (i).

(ii): Från första delen har vi att $\xi \in [c_{n+1}, c_n]$ så att

$$|\xi - c_n| \leq |c_{n+1} - c_n|$$

och högerledet är enligt (37) lika med $1/(q_n q_{n+1})$. □

D Python-kod tillhörande avsnitt 8

D.1 Vår förbättrade version av programmet

Nedan följer väsentliga delar av den förbättrade versionen av programmet som omnämns i 8.2. Koden är skrivet i Python och är en tolkning av den pseudokod som Helfgott presenterar i [1]. Om läsaren vill testa programmet kan all nedanstående kod förslagsvis läggas i samma fil. För att programmet ska fungera behöver Python-bibliotek *Bitarray* vara installerat.

Vi börjar med import av några externa funktioner, detta ska göras först i koden för att de senare funktionerna ska fungera. Biblioteket `bitarray` ger oss möjlighet att arbeta direkt med bitsträngar och funktionen `zeros(N)` konstruerar en sådan bitsträng bestående av `N` stycken nollor. Funktionerna `sqrt` och `floor` ger kvadratroten respektive golvfunktionen av ett tal och `isqrt` returnerar heltalsdelen av kvadratroten.

```
1 from bitarray import bitarray
2 from bitarray.util import zeros
3 from math import floor, sqrt, isqrt
```

Härnäst följer algoritmerna vars funktion beskrivs i 8.1, på flera rader finns det extra kommentarer (på engelska) som förklarar radens syfte. Alla funktioner, med undantag för DiophAppr, returnerar en bitsträng. Strängen representerar något sållat heltalsintervall $[a, b]$ där biten med index i är en nolla om talet $i + a$ har sållats bort och en etta annars.

SimpleSiev

Detta är det vanliga Eratosthenes såll. Funktionen tar in ett heltal N och returnerar en lista med alla primtal i intervallet $[0, N]$. Exempelvis ger `SimpleSiev(7)` svaret `00110101` vilket representerar alla primtal från 0 till 7.

```
1 def SimpleSiev(N):  # Traditional Sieve of Eratosthenes
2     P = zeros(N+1)  # Create bitarray of N+1 '0's, representing numbers 0 to N
3     P[2] = True     # Set 2 to '1'
4     P[3::2] = True  # Set odd numbers >1 to '1'
5     for num in range(3, isqrt(N)+1, 2):
6         if P[num]: P[num*num::2*num] = False  # Set odd mult's of num to '0'
7     return P
```

SimpleSegSiev

Denna funktion tar in tre argument; n, Δ och M. Därefter returnerar den en lista vilken representerar alla tal i intervallet $[n, n + \Delta]$ som antingen är prima eller saknar delare mindre än, eller lika med M. Använder vi notation från avsnitt 3, representerar denna lista mängden $S(\mathcal{A}, \mathcal{P}, z)$ där $A = [n, n + \Delta]$, $z = M$ och $\mathcal{P}$ är mängden av alla primtal. För att utföra sållningen behövs alltså alla primtal i intervallet $[0, M]$. Dessa genereras av SimpleSiev som åkallas på rad 5.

```
1 def SimpleSegSiev(n, Delta, M):
2     S = zeros(Delta+1)
3     S.setall(True)  # Set all bits in S to '1'
4     if n <= 1: S[0:2-n] = False  # Set 0 and 1 to '0' if present
5     Primes = SimpleSiev(M)  # Get list of primes up to M
6     for p in Primes.itersearch(bitarray('1')):  # Iterate through primes
7         S[max(-(n//-p), 2)*p - n::p] = False  # Set multiples of p to '0'
8     return S
```

SubSegSiev

Denna funktion är exakt som SimpleSegSiev gällande indata och utdata. Skillnaden mot förstnämnda är att intervallet $[0, M]$ delas upp i mindre delintervall. Funktionen tar sedan ett delintervall i taget, skapar en lista med alla primtal i delintervallet för att sedan sålla med hjälp av dessa

primtal. På så sätt behöver inte alla primtal i intervallet $[0, M]$ ligga i arbetsminnet på samma gång. För att generera listorna används SimpleSegSiev.

```
1  def SubSegSiev(n, Delta, M):
2      S = zeros(Delta+1)
3      S.setall(True)
4      if n <= 1: S[0:2-n] = False
5      D_s = isqrt(M)    # Size (-1) of each segment
6      for M_s in range(1, M+1, D_s+1):  # Iterate through segments
7          Primes = SimpleSegSiev(M_s, min(M-M_s,D_s), isqrt(M_s+D_s))  # Get primes in segment
8          for p in Primes.itersearch(bitarray('1')): # Iterate through primes
9              p = p+M_s  # shift p to get the actual prime number
10             S[max(-(n//-p),2)*p-n::p] = False
11     return S
```

NewSegSiev

Detta är algoritmens huvudfunktion. Den tar in tre argument; n, Δ och K, och returnerar alla primtal i intervallet $[n - \Delta, n + \Delta]$. Det sista argumentet K ska vara ett flyttal större eller lika med 2.5 och påverkar hur funktionen går tillväga för att sålla. Funktionen sållningen sker i två delar. Först sållas intervallet för multiplar av primtal mindre än eller lika med $K\Delta$. Detta utförs av SubSegSiev och sker på rad 16. Därefter sållas intervallet för resterande tal, vilket är multiplar av tal mellan $K\Delta$ och $\sqrt{n + \Delta}$. Detta görs med hjälp av DiophAppr i while-loopen som börjar på rad 17 och slutar på rad 31.

```
1  def NewSegSiev(n, Delta, K): # Main algorithm
2      # Argument parsing:
3      if n**(1/3)>=Delta or Delta>n:
4          raise Exception("Delta should be between n^(1/3) and n")
5      if K < 2.5:
6          raise Exception("K must be at least 2.5")
7      if n+Delta < 16:
8          raise Exception("n + Delta must be at least 16")
9
10     # Calculation of some repeatedly used values:
11     upper, lower = n+Delta, n-Delta
12     f = sqrt(Delta/(4*n))
13     bound1 = isqrt(upper)
14
15     M = floor(K*Delta) + 1
16     S = SubSegSiev(lower, 2*Delta, M-1) # SubsegSiev sieves by primes < M
17     while M <= bound1: # Sieve by the larger primes not covered by SubSegSiev
18         R = floor(M*f)
19         m0 = M + R
20         alpha0 = n/m0 % 1
21         alpha1 = -n/(m0*m0) % 1
22         ainv, q = DiophAppr(alpha1, 2*R)
23         c = floor(alpha0*q + 0.5)
24         k = floor(5*Delta*q/(4*M)) # Need int-type. floor(x/y) is faster than int(x//y)
25         bound2 = M + 2*R + 1
26         for j in range(-k-1, k+2):
27             r0 = -ainv*(c+j) % q
28             for m in range(m0+r0-((m0+r0-M)//q)*q, bound2, q): # for m in the intersection
29                 n_s = (upper//m)*m - lower
30                 if n_s >= 0: S[n_s] = False # Checking n_s <= n+Delta and n_s > m is redundant
31         M = bound2
32     return S
```

DiophAppr

Denna funktion tar in två tal; α och Q, och beräknar en rationell approximation a/q av α med hjälp av kedjebråk. Heltalen a och q uppfyller $(a, q) = 1$ samt att $q \leq Q$ och $\left|a/q - \alpha\right| \leq 1/qQ$. Funktionen returnerar talparet (q, a^{-1}) där a^{-1} är den multiplikativa inversen till a modulo q.

```
def DiophAppr(alpha, Q):
    b = p = floor(alpha)
    q = p_m = s = 1
    q_m = 0
    while q <= Q:
        if alpha == b:
            return -s*q_m, q
        alpha = 1/(alpha-b)
        b = floor(alpha)
        p, q, p_m, q_m, s = b*p+p_m, b*q+q_m, p, q, -s
    return s*q, q_m
```

D.2 Några extra funktioner

RemoveNonTwins

Här presenteras den funktion som använts i 8.3. Denna tar in en redan sållad lista med primtal och sållar bort alla primtal p där $p+2$ inte är prima. De tal i listan som är kvar efter detta är primtalstvillingarna i intervallet. Om det sista eller näst sista talet i listan är prima så är det omöjligt för funktionen att avgöra om talet är ett tvillingprimtal eller ej, funktionen sållar inte bort talet men utfärdar en varning om att detta har inträffat.

```
def RemoveNonTwins(S): # Sieve for twin primes in prime list.
    # If the last or second to last number in the given list is prime,
    #  then that prime won't get removed even though it might not be a twin prime.
    from bitarray import bitarray
    try:  # Removes 2 if present
        # Searches bits 0-3 for occurrence of '11' corresponing to primes 2,3
        #   then replaces '11' with '01'
        two = S.search(bitarray('11'),3)
        S[two[0]] = False
    except:
        pass
    for i in S.itersearch(bitarray('100')): # Replaces strings of 100 to 000 in bitarray S
        S[i:i+3] = False
    if S[-1]|S[-2]:
        print("Warning! The last prime in the list may not be a twin prime.")
    return
```

Till sist ger vi även en modifierad version av NewSegSiev och SubSegSiev som vi har valt att ge namnen NewSegSievTwins respektive SubSegSievTwins. Som namnet antyder kan NewSegSiev användas för att från grunden sålla fram alla primtalstvillingar i det angivna intervallet och funktionen brukas på samma sätt som NewSegSiev. Funktionen SubSegSievTwins är nödvändig för att NewSegSievTwins ska fungera.

NewSegSievTwins

```
def NewSegSievTwins(n, Delta, K):  # Sieve for twin primes, small mod of NewSegSiev
    if n**(1/3)>=Delta or Delta>n:
        raise Exception("Delta should be between n^(1/3) and n.")
    if K < 2.5:
        raise Exception("K must be at least 2.5")
    if n+Delta < 16:
        raise Exception("n + Delta must be at least 16")
    upper = n+Delta
    lower = n-Delta
    f = sqrt(Delta/(4*n))
    bound1 = sqrt(upper)
    M = floor(K*Delta) + 1
    S = SubSegSievTwins(lower, 2*Delta, M-1)
    while M <= bound1:
        R = floor(M*f)
        m0 = M + R
        alpha0 = n/m0 % 1
```

```
        alpha1 = -n/(m0*m0) % 1
        ainv, q = DiophAppr(alpha1, 2*R)
        c = floor(alpha0*q + 0.5)
        k = floor(5*Delta*q/(4*M))
        bound2 = M + 2*R + 1
        for j in range(-k-1, k+2):
            r0 = -ainv*(c+j) % q
            for m in range(m0+r0-((m0+r0-M)//q)*q, bound2, q):
                n_s = (upper//m)*m
                if n_s >= lower+2:  # both n_s and its twin n_s-2 is in the interval
                    S[n_s - lower] = False
                    S[n_s-2 - lower] = False
                elif n_s >= lower:  # n_s is in the interval but not n_s-2
                    S[n_s - lower] = False
                    if n_s+m-2 <= upper:  # the twin to the next multiple of m is in the interval
                        S[n_s+m-2 - lower] = False
        M = bound2
    return S
```

NewSegSievTwins

```
def SubSegSievTwins(n, Delta, M):  # Sieve for twin primes, small mod of SubSegSiev, needed in NewSegSievTwins
    S = zeros(Delta+1)
    S.setall(True)
    S[0:2-n] = False
    D_s = isqrt(M)
    for M_s in range(1, M+1, D_s+1):
        Primes = SimpleSegSiev(M_s, D_s, isqrt(M_s+D_s))
        for p in Primes.itersearch(bitarray('1')):
            p = p+M_s
            start = max(-(n//-p),2)*p-n
            S[start::p] = False
            if start-2<0:
                start = start+p
            S[start-2::p] = False
    return S
```