rijksuniversiteit groningen

faculteit Wiskunde en Natuurwetenschappen

Het tellen van priemgetallen in $\mathbb{Z}[i]$ en $\mathbb{Z}[\omega]$

Bacheloronderzoek Wiskunde

Juli 2013
Student: C.A. Verschoor
Eerste begeleider: Prof.Dr. J. Top
Tweede begeleider: Dr. A. Meijster

Samenvatting

In de gehelen van Gauss en de gehelen van Eisenstein proberen we priem telfuncties te ontwikkelen. Hiervoor zullen we eerst wat inleidende theorie behandelen en priem telfuncties bekijken in $\mathbb{Z}$. Daarna zullen we een brug slaan naar de gehelen van Gauss en gehelen van Eisenstein, om zo te ontdekken of in deze ringen priem telfuncties te vinden zijn. Ook zullen we ontdekken dat we in deze ringen methodes tegen komen voor het tellen van priemgetallen van de vorm $3 \pmod 4$ en $5 \pmod 6$.

Inhoudsopgave

1 Priemgetallen tellen in $\mathbb{Z}$ **3**
1.1 Zeef van Eratosthenes . . . 3
1.2 Legendre's Methode . . . 3
1.3 Meissel's Methode . . . 5
1.4 Mapes' Methode . . . 7
1.5 De modulaire priem telfunctie . . . 8

2 Gehelen van Gauss **8**
2.1 Euclidische Afstand . . . 10
2.2 Chebyshev Afstand . . . 11
2.2.1 Roosterpunten in een vierkant . . . 12
2.3 Zeven . . . 21
2.3.1 Pythagoreaanse Driehoeken . . . 21
2.3.2 Zeef van Atkin . . . 23
2.3.3 Tellen van priemgetallen 3 (mod 4) . . . 25

3 Gehelen van Eisenstein **29**
3.1 Zeven in $\mathbb{Z}[\omega]$ . . . 31
3.2 Tellen in $\mathbb{Z}[\omega]$ . . . 32

4 Conclusie **33**

1 Priemgetallen tellen in $\mathbb{Z}$

Voordat we priemgetallen gaan tellen in $\mathbb{Z}[i]$ en $\mathbb{Z}[\omega]$ willen we ons eerst gaan richten op de priemgetallen in $\mathbb{Z}$, we zouden methoden kunnen gebruiken voor het tellen van deze priemgetallen om deze vervolgens toe te passen op $\mathbb{Z}[i]$ en $\mathbb{Z}[\omega]$. We zullen hierbij eerst de zeef van Eratosthenes kort behandelen. Zo'n zeef is meestal een goede basis om een methode op te stellen. Daarna zullen we de Methode van Legendre behandelen en hoe deze methode tot stand komt en in zijn werk gaat. Daarna zullen we stil staan bij de methode van Meissel en tot slot nog de methode van Mapes'. Door deze methoden te onderzoeken, kunnen we vaststellen of we ze kunnen gebruiken om gevonden methoden om priemgetallen te tellen in $\mathbb{Z}[i]$ en $\mathbb{Z}[\omega]$ te kunnen versnellen of uitbreiden.

$\mathbb{Z}$ is een euclidisch domein. En $\mathbb{Z}$ heeft twee eenheden 1 en -1. Laten we daarom voor de methoden alleen kijken naar de verzameling getallen $\mathbb{Z}_{\geq 0}$ en daar de priemgetallen in zeven, de andere priemgetallen kunnen we namelijk vinden door vermenigvuldiging met een eenheid. Op deze verzameling bestaat ook een priem telfunctie, deze zal de priemgetallen tot en met een bepaalde bovengrens tellen. En is daarom ook als volgt gedefinieerd:

Definitie 1.1. De priem telfunctie $\pi(x)$ is gedefinieerd als:

$$\pi(x) = \#\{p : 1 \leq p \leq x, p \text{ priem}\}$$

1.1 Zeef van Eratosthenes

Eén van de simpelste methodes om priemgetallen te tellen in $\mathbb{Z}$ is via de zeef van Eratosthenes, niet alleen zal deze zeef alle priemgetallen tellen, hij zal ze ook nog vinden. De methode werkt als volgt, stel we willen alle priemgetallen onder N vinden. Dan schrijven we eerst alle getallen van 2 tot en met N op. Dan pakken we het eerste getal dat we vinden, dat is 2, en we zeven daarna alle veelvouden van 2 weg en we zetten 2 nu apart as priemgetal. Dan bij de overgebleven reeks getallen, doen we hetzelfde. Totdat we een lijst priemgetallen hebben. Dit proces herhaalt zich totdat het eerste element in de overgebleven reeks groter is dan $\sqrt{N}$, immers de eerste veelvoud van een priemgetal $p > \sqrt{N}$ die nog niet weg gezeefd is, is $p^2 > N$.

Hoewel we nu alle priemgetallen met gemak kunnen vinden, willen we ze eigenlijk wel heel graag tellen, zonder al deze priemgetallen nog apart te vinden. Hiervoor moeten we de zeef van Eratosthenes ombouwen.

1.2 Legendre's Methode

De Legendre methode is een methode om priemgetallen te tellen. En het maakt op een handige manier gebruik van de zeef van Eratosthenes. Deze methode is een basis voor wat we gaan gebruiken in $\mathbb{Z}[i]$ en $\mathbb{Z}[\omega]$, dat komt omdat in die ringen ook een zeef mogelijk is. Als we alle stappen van de zeef van Eratosthenes opschrijven, en daarnaast hoeveel je eigenlijk weghaalt kunnen we erachter komen hoe de Legendre Methode werkt. Laat x nu de bovengrens zijn en laat p_i alle priemgetallen zijn zodat $p_i \leq \sqrt{x}$, dan is de Legendre functie als volgt gedefinieerd:

$$\phi(x) = \lfloor x \rfloor - \sum_i \left\lfloor \frac{x}{p_i} \right\rfloor + \sum_{i<j} \left\lfloor \frac{x}{p_i p_j} \right\rfloor - \sum_{i<j<k} \left\lfloor \frac{x}{p_i p_j p_k} \right\rfloor + \dots$$

Laten we erachter komen, wat hier eigenlijk gebeurt. Laten we eerst alle gehele getallen nemen van 1 t/m x, dit zijn er uiteraard $\lfloor x \rfloor$. De volgende stap bij een zeef van Eratosthenes zou zijn om alle veelvouden van 2 onder x weg te halen. Dit zijn er in totaal $\left\lfloor \frac{x}{2} \right\rfloor$ veelvouden van 2. Zelfde doen we met alle andere priemgetallen. tot dusver hebben we dus $\lfloor x \rfloor - \sum_i \left\lfloor \frac{x}{p_i} \right\rfloor$. Maar er is hier een probleem, als we bijvoorbeeld eerst alle veelvouden van 2 weghalen en daarna veelvouden van 3 op deze manier. Dan halen we de veelvouden van 6 dubbel weg. En dit geldt niet alleen voor de priemgetallen 2 en 3, maar voor al die priemgetallen p_i. Om dit te compenseren moeten we dus elke twee priemgetallen $p_i \neq p_j$, met elkaar vermenigvuldigen en dan alle veelvouden van $p_i p_j$

onder x weer erbij op tellen. Ook dit zal even goed gaan, totdat we bij een punt komen dat we drie priemgetallen met elkaar moeten corrigeren. Neem bijvoorbeeld de priemgetallen 2, 3 en 5, dan worden nu in het eerste gedeelte alle veelvouden van $2, 3, 5$ afgetrokken. Daarna alle veelvouden van $2 \cdot 3$, $2 \cdot 5$ en $3 \cdot 5$ weer opgeteld, dus om dit te corrigeren moeten we alle veelvouden van $2 \cdot 3 \cdot 5$ onder x weer enkel aftrekken. Dit principe herhaalt zich voor hogere vermenigvuldigingen. Ook zijn er maar een eindig aantal van dit soort optellingen en aftrekkingen, dit komt omdat elke vermenigvuldiging van priemgetallen uniek is. En er in totaal maar x getallen zijn die we zeven. Dus zijn er maximaal x van dit soort optellingen en aftrekkingen.

We kunnen hier nog meer over zeggen, we hebben nu elk priemgetal p_i en de veelvouden van p_i precies één keer weg gezeefd, dus wat we overhouden zijn die priemgetallen die groter zijn dan $\sqrt{x}$ en het getal 1. Dus:

$$\phi(x) = \pi(x) - \pi(\sqrt{x}) + 1$$

Hier zouden we wat mee kunnen, als we nu namelijk $\phi(x)$ weten en $\pi(\sqrt{x})$ weten, kunnen we $\pi(x)$ berekenen. En zo dus priemgetallen gaan tellen. Ons doel is nu dus, om de functie $\phi(x)$ te versnellen, zodat we $\pi(x)$ sneller kunnen berekenen. De methoden van Meissel en Mapes spelen hier op in en zorgen ervoor dat $\phi(x)$ sneller kan worden berekend. Een andere manier om de boel te versnellen is om de Legendre functie recursief te maken. Om dit te doen gaan we de eerst kijken naar $q = \pi(\sqrt{x})$. Deze q heeft de eigenschap dat p_q het grootste priemgetal is onder $\sqrt{x}$. Omdat we alle priemgetallen onder de wortel nodig hebben voor de Legendre methode, kunnen we ook wel het volgende schrijven:

$$\phi(x, q) = \lfloor x \rfloor - \sum_{i \leq q} \left\lfloor \frac{x}{p_i} \right\rfloor + \sum_{i<j \leq q} \left\lfloor \frac{x}{p_i p_j} \right\rfloor - \sum_{i<j<k \leq q} \left\lfloor \frac{x}{p_i p_j p_k} \right\rfloor + \dots \tag{1}$$

Nu laten als $q = \pi(\sqrt{x})$, dan inderdaad $\phi(x, q) = \phi(x)$, maar dit gaat niet op als $q < \pi(\sqrt{x})$, omdat we dan met minder priemgetallen zeven en zo een aantal getallen kunnen overslaan. Neem $q = \pi(\sqrt{x})$, laten we nu onder x gaan zeven met de eerste $q - 1$ priemgetallen. Om p_q aan de zeef toe te voegen, moeten we dus alle veelvouden van p_q aftrekken, deze veelvouden zouden ook weer dubbelen kunnen bevatten en moeten we dus op dezelfde manier kunnen compenseren. Het maximaal aantal veelvouden zijn er $\left\lfloor \frac{x}{p_q} \right\rfloor$. Dus we doen dan niks anders dan:

$$\phi(x, q) = \phi(x, q-1) - \phi(\left\lfloor \frac{x}{p_q} \right\rfloor, q-1)$$

Verder als er geen priemgetallen meer over zijn om te zeven, dan is $\phi(x, 0) = \lfloor x \rfloor$. Verder kunnen we gebruik maken van $\left\lfloor \frac{\lfloor \frac{x}{a} \rfloor}{b} \right\rfloor = \lfloor \frac{x}{ab} \rfloor$. Zodat we dan een recursie hebben zonder alle vloerfuncties:

$$\phi(x, q) = \phi(x, q-1) - \phi(\frac{x}{p_q}, q-1) \tag{2}$$

Om deze Legendre functie iets sneller te maken, kunnen we de Euler totiënt functie gebruiken.

Definitie 1.2. De Euler totiënt functie voor $x \in \mathbb{Z}_{>0}$ is:

$$\varphi(x) = \#\left\{a : 1 \leq a < x, \gcd(a, x) = 1\right\}$$

Stelling 1.3. *Laat $m_k = \prod_{i \leq k} p_i$, dan:*

$$\phi(m_k, k) = \varphi(m_k)$$

Bewijs. $\phi(m_k, k)$ zeeft weg alle veelvouden van p_i met $i \leq k$. Dus alle getallen die we niet weg zeven hebben allemaal een gcd van 1 vergeleken met elk priemgetal p_i. Aangezien de grootste gemene deler multiplicatief is, betekent dat het product van alle p_i ook een grootste gemene deler heeft van 1 ten opzichte van wat we overhouden na het zeven. Ten tweede telt $\varphi(m_k)$ al deze getallen onder m_k en dat doet $\phi(m_k, k)$ dus ook. □

Stelling 1.4. *Neem* $m_k = \prod_{i \leq k} p_i$ *en neem verder* $a = bm_k + r$ *waarbij* $r = a \bmod m_k$ *en* $b = \frac{a-r}{m_k}$, *dan*

$$\phi(a, k) = b\varphi(m_k) + \phi(r, k) \tag{3}$$

Bewijs. Neem een element $q+ \in \mathbb{Z}$ en stel $\gcd(q, m_k) = 1$ dan natuurlijk $\gcd(q + um_k, m_k) = 1$ voor $u \in \mathbb{Z}$ ook, dit betekent dat we een patroon hebben van lengte m_k dat zich herhaalt. Dus als $r = a \bmod m_k$ en er bestaan b veelvouden van m_k onder a, dan $\phi(a, k) = b\phi(m_k, k) + \phi(r, k) = b\varphi(m_k) + \phi(r, k)$. Nu hoeven we alleen nog na te gaan dat $b = \frac{a-r}{m_k}$. Dit is zo want $a - r$ is de eerste veelvoud van m_k onder a. Dus het aantal veelvouden onder a is een simpele deling door m_k. □

Een nog iets snellere uitbreiding vind plaats door middel van een soortgelijk principe

Stelling 1.5. *Neem* $m_k = \prod_i p_i$ *voor alle* $i \leq k$ *en zijn alle priemgetallen tussen* 0 *en* p_k *en verder geldt* $a < m_k$, *dan*

$$\phi(a, k) = \varphi(m_k) - \phi(m_k - a - 1, k) \tag{4}$$

Bewijs. Ook hier is de kleinste gemene veelvoud van alle p_i gelijk aan m_k. En ontstaat er dus een patroon om de m_k veelvouden in. In ons geval is $a < m_k$, dus is er geen volledig patroon. Toch zouden we dan omgekeerd kunnen denken. Stel we maken dit patroon af, dan vinden we $\varphi(m_k)$ copriemen. Maar dan hebben we dus $m_k - a - 1$ meer ruimte gebruikt dan was afgesproken. Dus die moeten we er dan weer van af trekken. □

(3) en (4), zijn erg belangrijk voor een snel verloop van de functie $\phi(a, k)$, namelijk door middel van de recursie in (2) kunnen we k steeds verkleinen, totdat we m_k klein genoeg hebben om vervolgens ook a weer te verkleinen. Toch kan m_k zelf al een vrij groot getal zijn, waardoor we dit principe pas kunnen gebruiken als onze priemgetallen bijna op raken.

De tijdscomplexiteit van de Legendre functie $\phi(x)$ is hier $O(x)$, we moeten immers all mogelijk combinaties van producten $p_{a_1} p_{a_2} \dots p_{a_j}$ bij langs, het liefst via de recursie in (2). Deze recursie zal alle mogelijkheden producten van priemgetallen kleiner dan x bij langs gaan en alleen alle priemgetallen tussen $\sqrt{x}$ en x overslaan. Volgens de Priemgetal stelling is $\pi(x) \sim \frac{x}{\ln(x)}$ (zie [3], [6]), dus voor dat recursie proces zijn er ongeveer $O(x - \frac{x}{\ln(x)}) = O(x)$ stappen nodig. Met gebruik van de totiënt functie kan het aantal stappen verder verkleind worden, maar dit zal voornamelijk gebruikt worden als k klein is en a heel groot en zal de $O(x)$ barrière daarom niet verlagen. Voor het geheugen hoeven we alleen bij te houden, alle priemgetallen onder de wortel. En wellicht nog wat voorgerekende waardes van $\phi(x)$. Deze voorgerekende waardes zullen vooral bedoeld zijn voor m_k. Dit komt dus neer op een opslagcomplexiteit van $O(\sqrt{x})$.

1.3 Meissel's Methode

Meissel's Methode en ook de Meissel-Lehmer Methode, maken niet noodzakelijk gebruik van (3) en (4), maar ook hier wordt wel gekeken naar het versimpelen van de functie $\phi(a)$.

De methode die Meissel beschreef, zorgt ervoor dat we deze manier van denken opsplitsen in stukjes. De methode die Lehmer hieraan toevoegde, is een uitbreiding hiervan, maar volgt hetzelfde principe. Laten we dit principe kort samenvatten. Het begint met de functie $P_k(x, a)$

$$P_k(x, a) = \#\left\{p_{i_1} p_{i_2} \dots p_{i_k} \leq x : i_k \geq \dots \geq i_2 \geq i_1 \geq a + 1\right\} \tag{5}$$

Een paar dingen zijn hieruit al vrij snel duidelijk, bijvoorbeeld $P_1(x, a) = \pi(x) - a$, dit zijn namelijk alle getallen van de vorm p_{i_1} met $i_1 \geq a + 1$. Een speciaal geval is $P_0(x, a) = 1$, omdat 1 het enige getal is dat kleiner is dan x zonder priemdelers. Ook voldoet $P_k(x, a)$ aan een andere eigenschap, die ons kan helpen bij het vinden van $\phi(x)$:

Stelling 1.6.

$$\phi(x, a) = \sum_{k=0}^{\infty} P_k(x, a) \tag{6}$$

Bewijs.

$$\begin{aligned}\sum_{k=0}^{\infty} P_k(x,a) &= \sum_{k=0}^{\infty} \#\left\{p_{i_1}p_{i_2}\dots p_{i_k} \le x : i_k \ge \dots \ge i_2 \ge i_1 \ge a+1\right\} \\ &= \#\bigcup_{k=0}^{\infty}\left\{p_{i_1}p_{i_2}\dots p_{i_k} \le x : i_k \ge \dots \ge i_2 \ge i_1 \ge a+1\right\} \\ &= \#\left\{p_{i_1}p_{i_2}\dots p_{i_k} \le x : i_k \ge \dots \ge i_2 \ge i_1 \ge a+1, k \ge 0\right\} \\ &= \#\left\{n \le x : \forall_{j\le a} p_j \nmid n\right\} = \phi(x,a)\end{aligned}$$

□

Legendre's Methode maakt gebruik van (6) en dat $P_1(x,a) = \pi(x) - a$. Voor $a = \pi(\sqrt{x})$ hebben $P_2(x,a) = 0$, want alle priemgetallen boven p_a, zijn ook groter dan $\sqrt{x}$ en het eerste product $p_{a+1}p_{a+1} > \sqrt{x}\sqrt{x} = x$. Dus Hieruit volgt: $\phi(x, \pi(\sqrt{x})) = \pi(x) - \pi(\sqrt{x}) + 1$, wat neer komt op de Legendre Methode.

Meissel's Methode gaat uit van $\pi(\sqrt[3]{x}) \le a < \pi(\sqrt{x})$, hier is $P_2(x,a) \ge 0$, maar $P_3(x,a) = 0$.

Lehmer's Uitbreiding op Meissel gaat uit van $\pi(\sqrt[4]{x}) \le a < \pi(\sqrt[3]{x})$, hier is $P_3(x,a) \ge 0$, maar $P_4(x,a) = 0$.

Voor berekening van $P_2(x,a)$, kunnen we het beste gewoon de definitie gebruiken. We nemen hiervoor $\pi(\sqrt[3]{x}) \le a < \pi(\sqrt{x})$. Neem verder $b = \pi(\sqrt{x})$

$$\begin{aligned}P_2(x,a) &= \#\left\{p_i p_j \le x : j \ge i \ge a+1\right\} \\ &= \sum_{i=a+1}^{\infty} \#\left\{p_j \le \frac{x}{p_i} : j \ge i\right\} \\ &= \sum_{i=a+1}^{\infty} P_1\left(\frac{x}{p_i}, i-1\right) \\ &= \sum_{i=a+1}^{b} P_1\left(\frac{x}{p_i}, i-1\right) \\ &= \sum_{i=a+1}^{b} \pi\left(\frac{x}{p_i}\right) - (i-1) \\ &= -\frac{(b-a)(b+a-1)}{2} + \sum_{i=a+1}^{b} \pi\left(\frac{x}{p_i}\right) \qquad (7)\end{aligned}$$

(6) samen met (7) vormt nu een formule voor $\pi(x)$, in het bijzonder als $a = \pi(\sqrt[3]{x})$, namelijk

$$\begin{aligned}\pi(x) &= \phi(x,a) - 1 + a + \frac{(b-a)(b+a-1)}{2} - \sum_{i=a+1}^{b} \pi\left(\frac{x}{p_i}\right) \\ &= \phi(x,a) + \frac{(b-a+1)(b+a-2)}{2} - \sum_{i=a+1}^{b} \pi\left(\frac{x}{p_i}\right) \qquad (8)\end{aligned}$$

Hoe snel kunnen we $\pi(x,a)$ berekenen, we weten dat we hiervoor elk priemgetal $p_k \le p_a$ bij langs gaan, en alle veelvouden onder x hiervan willen weten. Dus deze functie is qua complexiteit in de buurt van $O(x)$. Ook krijgen we nu een extra som met nieuwe priem telfuncties. In deze som zal $\sqrt[3]{x} < p_i \le \sqrt{x}$, hoewel dit er veel zijn, zijn ze wel sneller te berekenen. En als we de waardes in de recursie van $\phi(x,a)$ opslaan, kunnen we veel van die $\pi(\frac{x}{p_i})$ berekenen zonder te veel moeite te

doen. Deze Meissel methode zorgt ervoor dat $\pi(x)$ nog iets sneller te berekenen is dan de Legendre functie.

De Meissel-Lehmer methode verschilt niet veel van de Meissel methode en gaat alleen om het verlagen van a en berekenen van $P_3(x,a)$. Dit levert nog een rij sommen op. Eigenlijk kunnen we dit proces steeds herhalen, maar dan worden de formules zeer complex en dat komt de snelheid niet ten goede.

1.4 Mapes' Methode

Ook Mapes' Methode maakt net als de Methoden van Meissel en Lehmer, gebruik van het anders schrijven van $\phi(a,k)$. En ook deze kan in combinatie gebruikt worden met de Meissel-Lehmer Methode. Toch is de wijze waarop hier naar het probleem wordt gekeken anders. We introduceren eerst de volgende functie:

$$T_k(x,a) = (-1)^{\beta_0+\beta_1+\ldots+\beta_{a-1}} \left\lfloor \frac{x}{p_1^{\beta_0} p_2^{\beta_1} \ldots p_a^{\beta_{a-1}}} \right\rfloor \tag{9}$$

Hier geldt voor elke n dat $\beta_n \in \{0,1\}$. En k is het getal, in de volgende binaire vorm: $k = \beta_{a-1}\beta_{a-2}\ldots\beta_1\beta_0$ Ook definiëren we dat $T_k(-x,a) = -T_k(x,a)$. Uit (1) volgt nu direct dat:

$$\phi(x,a) = \sum_{k=0}^{2^a-1} T_k(x,a) \tag{10}$$

Kies nu een $M < 2^a$ dan bestaat er een $i \geq 0$ zodat $2^i | M$, nou ten eerste merken we op dat we $T_M(x,a)$ iets handiger kunnen definiëren:, namelijk als:

$$T_M(x,a) = (-1)^{\beta_i+\beta_{i+1}+\ldots+\beta_{a-1}} \left\lfloor \frac{x}{p_{i+1}^{\beta_i} p_{i+2}^{\beta_{i+1}} \ldots p_a^{\beta_{a-1}}} \right\rfloor \tag{11}$$

immers $\beta_j = 0$ voor $j < i$. Neem nu een het binaire getal $k = \alpha_{i-1}\alpha_{i-2}\ldots\alpha_0 < 2^i$, dan volgens (11) geldt:

$$\begin{aligned}
T_k(T_M(x,a),i) &= (-1)^{\beta_i+\beta_{i+1}+\ldots+\beta_{a-1}} \left\lfloor \frac{T_M(x,a)}{p_{i+1}^{\beta_i} p_{i+2}^{\beta_{i+1}} \ldots p_a^{\beta_{a-1}}} \right\rfloor \\
&= \operatorname{sign}(T_M(x,a))(-1)^{\beta_i+\beta_{i+1}+\ldots+\beta_{a-1}} \left\lfloor \frac{|T_M(x,a)|}{p_{i+1}^{\beta_i} p_{i+2}^{\beta_{i+1}} \ldots p_a^{\beta_{a-1}}} \right\rfloor \\
&= (-1)^{\alpha_0+\alpha_1+\ldots+\alpha_{i-1}}(-1)^{\beta_i+\beta_{i+1}+\ldots+\beta_{a-1}} \left\lfloor \frac{\left\lfloor \frac{x}{p_1^{\alpha_0} p_2^{\alpha_1} \ldots p_i^{\alpha_{i-1}}} \right\rfloor}{p_{i+1}^{\beta_i} p_{i+2}^{\beta_{i+1}} \ldots p_a^{\beta_{a-1}}} \right\rfloor \\
&= (-1)^{\alpha_0+\alpha_1+\ldots+\alpha_{i-1}+\beta_i+\beta_{i+1}+\ldots+\beta_{a-1}} \left\lfloor \frac{x}{p_1^{\alpha_0} p_2^{\alpha_1} \ldots p_i^{\alpha_{i-1}} p_{i+1}^{\beta_i} p_{i+2}^{\beta_{i+1}} \ldots p_a^{\beta_{a-1}}} \right\rfloor \\
&= T_{M+k}(x,a)
\end{aligned} \tag{12}$$

Deze laatste stap is mogelijk omdat $k < 2^i$ is gekozen, we kunnen daarom deze α's simpelweg samenvoegen met de β 's zonder dat de binaire representatie van k en M overlapt. En dus $M + k = \beta_{a-1}\beta_{a-2}\ldots\beta_i\alpha_{i-1}\alpha_{i-2}\ldots\alpha_0$.

Nu gaan we iets verder, neem nu niet alleen die i zodat $2^i | M$ maar ook $2^{i+1} \nmid M$, dan definiëren we hierop de volgende functie:

$$\gamma(M,x,a) = \sum_{k=M}^{M+2^i-1} T_k(x,a) \tag{13}$$

Ook nu door (10) een beetje om te schrijven, vinden we

$$\phi(x,a) = T_0(x,a) + \sum_{k=0}^{a-1} \gamma(2^k, x, a) \tag{14}$$

Maar ook geldt met behulp van (10) en (12) het volgende:

$$\phi(T_M(x,a),i) = \sum_{k=0}^{2^i-1} T_k(T_M(x,a),i) = \sum_{k=0}^{2^i-1} T_{M+k}(x,a) = \sum_{j=M}^{M+2^i-1} T_j(x,a) = \gamma(M,x,a) \tag{15}$$

Combinerend geeft ons dit dat:

$$\phi(x,a) = T_0(x,a) + \sum_{k=0}^{a-1} \phi(T_{2^k}(x,a), 2^k - 1) \tag{16}$$

Dit vormt het algemene recept van de Mapes' methode. We kunnen immers (16) herhaaldelijk toepassen door x te vervangen met $T_M(x,a)$ en de a met een bijbehorende i. Dit proces kunnen we herhalen en vereenvoudigen zodat we snel ϕ kunnen berekenen. Hiervoor zullen kleine waarden opgeslagen moeten worden. Het verdere verloop van Mapes' methode gaat volgende een bepaald schema dat met behulp van alle stellingen in $\mathcal{O}(x^{0.7})$ tijd en $O(x^{0.7})$ geheugen priemgetallen telt [6]. Hoewel dus de snelheid van dit algoritme erg snel is vergeleken met Legendre's methode en Meissel's methode, moeten we hier wel veel voor opslaan.

1.5 De modulaire priem telfunctie

Stel we zijn niet geïnteresseerd in de gewone priemgetallen in $\mathbb{Z}$, maar meer geïnteresseerd in priemgetallen van de vorm $ak+b$ met $k \in \mathbb{Z}_{\geq 0}$ en $\gcd(a,b) = 1$. Deze priem telfuncties noteren we met $\pi(x;a,b) = \#\{k \in \mathbb{Z}_{\geq 0} : ak+b \text{ priem}, ak+b \leq x\}$. We ouden iets kunnen zeggen over $\pi(x;4,3)$ en $\pi(x;6,5)$ met behulp van de gehelen van Gauss en gehelen van Eisenstein. Dit zullen we verder op ook onderzoeken. Het toepassen van methoden die gebruikt worden in $\mathbb{Z}$ om vervolgens $\pi(x;a,b)$ te berekenen zijn niet altijd een goede manier om dit soort priem telfuncties te berekenen. Een voorbeeld van een simpele methode is om gewoon de zeef van Erathostenes toe te passen op alle getallen onder x en daarna de getallen van de vorm $ak+b$ eruit te halen, en deze te tellen. Alleen dit kunnen we niet omzetten naar een Legendre functie. Ook zouden we eerst alle getallen van de vorm $ak+b$ op kunnen schrijven en dan daarna zeven. Dit kan in sommige gevallen goed gaan, alleen moeten we wel eerst onderzoeken met welke priemgetallen je mag zeven. Een voorbeeld als je alle getallen van de vorm $7k+3$ neemt, dan is het niet voldoende om te zeven met priemgetallen van diezelfde vorm. Als $k=1$, dan zou 10 als priemgetal worden gezien namelijk. Een simpele oplossing zou zijn, om gewoon alle priemgetallen te gebruiken om te zeven. En hoewel hier een Legendre functie van te maken is, zal deze te ingewikkeld zijn om uit te werken tot een Meissel of Mapes methode. Hoewel dus deze methoden tekort schieten, zijn er snelle algoritmen om modulaire priem telfuncties te berekenen ontwikkeld. [2]

2 Gehelen van Gauss

De gehelen van Gauss, zijn niets anders dan de getallen $\mathbb{Z}[i]$. En zijn dus allen van de vorm $a+bi$, verder bezit $\mathbb{Z}[i]$ over een aantal bijzondere eigenschappen. Het is namelijk een Euclidisch domein, wat delen met rest mogelijk maakt. Verder is de norm van een element $a+bi \in \mathbb{Z}[i]$ gelijk aan $N(a+bi) = a^2+b^2$. En de eenheden zijn ± 1 en $\pm i$. Om priemgetallen in deze ring te tellen, gaan we ons eerst richten op het identificeren van de priemgetallen. Daarna zullen we proberen of we een Legendre methode kunnen maken, die deze priemgetallen telt. Dit zullen we ook op meerdere manieren bekijken. Hierna gaan we onderzoeken of een andere manier van zeven, misschien een

betere methode oplevert. Maar voordat we beginnen met het vinden van priem telfuncties, laten we eerst de basis behandelen.

Een priemgetal p in een euclidisch domein is ook irreducibel, dus de enige delers van een priemgetal zijn up en u, waarbij u een eenheid is in het domein. Omdat de norm multiplicatief is, is het handig om eerst hier de priemgetallen te zoeken. De norm is een functie van $\mathbb{Z}[i] \mapsto \mathbb{Z}$, dus als voor $p \in \mathbb{Z}[i]$ geldt dat het een priemgetal is, dan moet $N(p)$ een priemgetal zijn in $\mathbb{Z}$ of voor alle $q \in \mathbb{Z}[i]$ met $q \neq up$ en $q \neq u$, met u een eenheid, geldt dat $N(q) \nmid N(p)$.

Nu zouden we ons af kunnen vragen welke priemgetallen in $\mathbb{Z}$ nou ook direct priemgetallen zijn in $\mathbb{Z}[i]$. Laten we hiervoor kijken naar de norm van $a+bi$, deze was $N(a+bi) = a^2+b^2$, als $a+bi \in \mathbb{Z}$, dan $b=0$. Dus is de norm dan niks anders dan $N(a) = a^2$, als a nu een priemgetal is in $\mathbb{Z}$, dan is deze ook een priemgetal in $\mathbb{Z}[i]$ als we geen $p \in \mathbb{Z}[i]$ kunnen vinden, zodat $N(p) = a$. Aangezien $a^2 \equiv 0 \pmod 4$ als a even is, en $a^2 \equiv 1 \pmod 4$ als a oneven. Moet het zo zijn dat $N(p) \not\equiv 3 \pmod 4$, en dus als $a \equiv 3 \pmod 4$ moet a een priemgetal zijn in $\mathbb{Z}[i]$. Voor $a \equiv 1 \pmod 4$ is dit niet het geval, het zou namelijk hier wel kunnen dat er een $p \in \mathbb{Z}[i]$ bestaat zodat $N(p) = a$. Om dit te laten zien, gebruiken we twee stellingen. De tweede laat zien dat (a) geen priemideaal is in $\mathbb{Z}[i]$ en dus is a ook geen priemgetal en ook geen irreducibele. Dit betekent dat we een getal $c \in \mathbb{Z}[i]$ kunnen vinden die a deelt. En bovendien zal $N(c) = a$.

Stelling 2.1. *Stel $p \in \mathbb{Z}$ is een priemgetal van de vorm $4k+1$, dan bestaat er een $x \in \mathbb{Z}$, zodat $p|x^2+1$.*

Bewijs. Neem $p = 4k+1$ een priemgetal in $\mathbb{Z}$, laten we nu eerst $(p-1)! = (4k)! \bmod p$ berekenen. Aangezien elk getal in $\mathbb{F}_p$ een inverse heeft, en de enige getallen waarvan de inverse gelijk is aan hun zelf 1 en -1 zijn. Is $(p-1)!$ niks anders dan een vermenigvuldiging met allemaal enen en en een enkele -1. Dus $(p-1)! \equiv (4k)! \equiv -1 \pmod p$. Verder geldt ook $(4k)! \equiv 1 \cdot 2 \cdot \ldots \cdot (4k-1) \cdot 4k \equiv 1 \cdot 2 \cdot \ldots \cdot -2 \cdot -1 \equiv (-1)^{2k} 1^2 \cdot 2^2 \cdot (2k)^2 \equiv ((2k)!)^2 \equiv -1 \pmod p$. Dus neem $x = ((2k)!)^2$, dan $p|x^2+1$. □

Stelling 2.2. *Stel $p \in \mathbb{Z}$ is een priemgetal van de vorm $4k+1$, dan is het ideaal (p) geen priemideaal in $\mathbb{Z}[i]$*

Bewijs. We weten dat we een x kunnen vinden zodat $p|x^2+1$. In $\mathbb{Z}[i]$ is dit gelijk aan zeggen dat $p|(x+i)(x-i)$. Als $p \in \mathbb{Z}[i]$ een priemgetal is in $\mathbb{Z}[i]$ dan moet deze dus volgens de definitie ofwel $x+i$ delen of $x-i$ delen. Maar als $p = 4k+1 \in \mathbb{Z}[i]$, dan is dit duidelijk niet het geval omdat $\frac{1}{p} \notin \mathbb{Z}$. Dus bestaan er $a, b \in \mathbb{Z}[i]$ zodat $ab = p \in (p)$ maar $a \notin (p)$ en $b \notin (p)$ □

Dus $p = 4k+1$ is geen priemgetal in $\mathbb{Z}[i]$, aangezien $N(p) = p^2$ kunnen we dus gaan zoeken naar een priemgetal $q \in \mathbb{Z}[i]$ zodat $N(q)|N(p)$, of te wel een q zodat $N(q) = p$. En omdat p geen priemgetal is en dus niet irreducibel, moet deze q ook bestaan. Dan houden we nog één geval over, namelijk het priemgetal 2 in $\mathbb{Z}$. Deze wordt behandelt als speciaal geval en is geen priemgetal in $\mathbb{Z}[i]$, want $2 = -i(1+i)^2$. Het element $1+i$ is wel een priemgetal omdat $N(1+i) = 2$. Dit priemgetal wordt als een speciaal geval behandelt.

Voor het maken van een Legendre functie, gebruiken we meestal een soort zeef van Eratosthenes. Maar $\mathbb{Z}[i]$ heeft de vorm van een rooster en $\mathbb{Z}$ is gewoon een lijn. Om op dit rooster een bovengrens te bepalen waaronder we priemgetallen willen vinden, kunnen we metrieken gebruiken. Als we alle punten op een afstand x van de oorsprong met een metriek d markeren, zullen we een grens krijgen waaronder we tellen. Ook moeten we ervoor zorgen dat binnen deze grens een eindig aantal elementen van $\mathbb{Z}[i]$ zitten.

De meest voor de hand liggende metriek is misschien wel $\mathcal{E} = d(a+bi, c+di) = \sqrt{(a-c)^2+(b-d)^2}$, dit is de *Euclidische afstand*. Verder in deze metriek is $|u| = d(u,0) = 1$ voor alle eenheden $u \in \mathbb{Z}[i]$. Deze zullen niet meetellen in de priem telfunctie, maar het geeft wel aan dat de priemgetallen dan een hogere absolute waarde zullen hebben dus kun je ze minder snel verwarren. Ook is hier $|a+bi| = \sqrt{N(a+bi)}$, wat betekent dat de absolute waarde ook nog multiplicatief is. Nog een andere manier om het te bekijken is met $\mathcal{M} = d(a+bi, c+di) = |a-c|+|b-d|$. Dit is niets anders dan de *Manhattan Afstand*, en over gehele getallen zal deze ook altijd als afstand een positief

geheel getal geven. We weten met deze metriek precies hoeveel punten dezelfde afstand hebben tot de oorsprong, in tegenstelling tot de euclidische metriek, waarbij we het niet zo snel kunnen zien. Dan als laatst hebben we nog de metriek $\mathcal{C} = d(a+bi, c+di) = \max(|a-c|, |b-d|)$, de *Chebyshev Afstand* deze zal $1+i$ op dezelfde afstand zetten als een eenheid zoals 1, maar ook hier weten we precies hoeveel punten er op een bepaalde afstand van de oorsprong zitten en kan het daarom handig zijn. Alleen moeten we hier niet de fout maken om $1+i$ op te vatten als een eenheid.

2.1 Euclidische Afstand

Laten we beginnen met het construeren van een klassieke Legendre functie. Hierbij nemen we de euclidische metriek met als gevolg dat we dus alle priemgetallen gaan tellen in een cirkel. Deze euclidische metriek komt erg overeen met de norm over de gehelen van Gauss. Het is namelijk de wortel van deze norm, waar we in geïnteresseerd zijn. Dit betekent dat ook de euclidische afstand tussen de oorsprong en $a+bi$ een multiplicatieve functie is. Als we nu het aantal elementen van een ideaal willen tellen, waarbij elk element op maximale afstand n van de oorsprong zit, kunnen we door deling van de radius van de cirkel het probleem verkleinen. Neem bijvoorbeeld alle veelvouden van $a+bi \in \mathbb{Z}[i]$ in een cirkel van radius n. Deze verzameling getallen bevat evenveel getallen als de verzameling getallen $c+di \in \mathbb{Z}[i]$ die op afstand $\frac{n}{\sqrt{a^2+b^2}}$ van de oorsprong staan. Dit komt door de multiplicativiteit van de norm over de getallen van Gauss. Dit betekent dat het aantal veelvouden van $a \in \mathbb{Z}[i]$ binnen een straal n van de oorsprong, gelijk is aan $C(\frac{n}{a})$ waarbij $C(x)$ de functie is die alle getallen $a+bi \in \mathbb{Z}[i]$ telt zodat de euclidische afstand tot de oorsprong maximaal x is. De functie is daarom als volgt gedefinieerd: $C(x) = \{z \in \mathbb{Z}[i] : |z| < x\}$. Als we nu alle priemgetallen $p_i \in \mathbb{Z}[i]$ nemen, met $|p_i| \leq \sqrt{x}$, dan kunnen we op dezelfde manier zeven als bij de zeef van Erathosthenes. Dat mag omdat vanwege multipliciteit van $d(z,0)$ elk niet-priemgetal q met $\sqrt{x} < |q| \leq x$ een deler moet hebben die een absolute waarde heeft kleiner of gelijk aan $\sqrt{x}$. En verder zal deze zeef nu een bijbehorende Legendre functie hebben:

$$\phi_{\mathbb{Z}[i];\mathcal{E}}(x,a) = C(x) - \sum_i C\left(\frac{x}{|p_i|}\right) + \sum_{i<j} C\left(\frac{x}{|p_i||p_j|}\right) - \sum_{i<j<k} C\left(\frac{x}{|p_i||p_j||p_k|}\right) + \dots \tag{17}$$

Hoewel deze $C(x)$ een prima formule is, hebben we alleen een probleem, deze functie is niet makkelijk te berekenen. Het berekenen van $C(x)$ heet ook wel Gauss' cirkel probleem. Een manier om $C(x)$ te berekenen is om te kijken naar een kwartcirkel. Stel $a^2+b^2 = x^2$, dan kunnen we dit omschrijven tot $a = \sqrt{x^2-b^2}$. Nu om na te gaan hoeveel punten $c+di \in \mathbb{Z}$ een afstand hebben kleiner dan x, kunnen we $0 < b \in \mathbb{Z}$ variëren en dan a naar beneden afronden om het aantal $c+di$ te krijgen met $d = b > 0$ en $c > 0$. Wat we dan nog over houden zijn alle $c+di$ met $d < 0$ of $c < 0$ die we kunnen vinden door te vermenigvuldigen met 4. En alle $c+di$ zodat $c = 0$ of $d = 0$ kunnen we apart behandelen. Dit brengt ons tot:

$$C(x) = 1 + 4\lfloor x \rfloor + 4\sum_{b=0}^{x} \left\lfloor \sqrt{x^2-b^2} \right\rfloor \tag{18}$$

Dit is te berekenen in $O(x)$ tijd, wat in samenwerking met de Legendre functie alleen maar nadelig zal werken. Ook als we de Legendre functie opschrijven door middel van een recursie, zal dit probleem niet oplossen, we moeten dan aan het eind van de recursie als nog $C(x)$ berekenen.

Als we $C(x)$ wel snel konden berekenen en we zouden daarna de Legendre functie uitwerken tot een snellere methode zoals Mapes' Methode, dan kunnen we gebruik maken van de volgende relatie:

$$\frac{1}{4}\pi_{\mathbb{Z}[i];\mathcal{E}}(x) = 2\pi(x;4,1) + \pi(\sqrt{x};4,3) + 1 \quad , x > \sqrt{2} \tag{19}$$

Deze relatie geldt, omdat alle priemgetallen van de vorm $4k+1$, twee delers heeft in $\mathbb{Z}[i]$ en $4k+3$ niet. Voor een priemgetal $4k+1 \in \mathbb{Z}$ kunnen we een priemgetal in $a+bi \in \mathbb{Z}[i]$ vinden zodat $(a+bi)(a-bi) = 4k+1$. Dus elke $4k+1$ komt overeen met twee priemgetallen. Elk priemgetal

$4k+3 \in \mathbb{Z}$ is ook priem in $\mathbb{Z}[i]$, en elke norm is hier $(4k+3)^2$ wat maakt dat we hiervoor alleen priemgetallen van de vorm $4k+3$ hoeven te tellen onder $\sqrt{x}$ in plaats van x. De bijbehorende $+1$ is een bijdrage van het priemgetal $2 \in \mathbb{Z}$ en dus $1+i \in \mathbb{Z}[i]$ die als speciaal geval wordt gezien. En de deling door 4 is vanwege de vier eenheden in $\mathbb{Z}[i]$. Als we nu voor twee van de drie priem telfuncties een snelle methode kunnen vinden, kunnen we daarom ook een snelle methode vinden door gebruik te maken van de vergelijking.

Sterker nog, als we een redelijke formule voor $\pi(x;4,3)$ kunnen vinden zouden we samen met $\pi_{\mathbb{Z}[i];\mathcal{E}}(x)$ een vrij snel algoritme kunnen vinden voor $\pi(x;4,1)$. We weten al uit [2] dat $\pi(x;4,1)$ en $\pi(x;4,3)$ beide berekend kunnen worden in $O\left(\frac{x^{2/3}}{log^2(x)}\right)$ tijd. Hier is $x = 10^{20}$ de hoogst bekende waarde voor beide functies op het moment van schrijven. Dit betekent, over euclidische afstand gekeken, dat $\pi_{\mathbb{Z}[i];\mathcal{E}}(x)$ ook tot deze hoogte berekent kan worden. De functie $C(x)$ die in de Legendre methode en dus ook Meissel methode zit, zal ons niet helpen om meer priemgetallen te tellen in $\mathbb{Z}[i]$, deze is gewoonweg te langzaam. We hebben iets efficiënters nodig.

2.2 Chebyshev Afstand

Ook onder de Chebyshev afstand kunnen we de afstand tot de oorsprong gebruiken om Legendre-achtige functies te creëren. Laten we daarom eerst een aantal eigenschappen van deze metriek naar voren brengen. Ten eerste willen we kijken naar deling met rest, dit is een belangrijke eigenschap en die gebruiken we om het aantal veelvouden te tellen binnen een bepaalde omtrek.

Stelling 2.3.
Voor $\alpha \in \mathbb{Z}[i]$ en $\beta = k+li \in \mathbb{Z}[i], \beta \neq 0$ en afstandsfunctie $d(a+bi,0) = \max\{|a|,|b|\}$

(a) Als $\beta = i^m(n+ni)$, voor $m,n \in \mathbb{Z}$, dan kunnen we een α vinden, zodat er geen $\gamma,\rho \in \mathbb{Z}[i]$ bestaan met $d(\rho,0) < d(\beta,0)$ zodat $\alpha = \gamma\beta + \rho$.

(b) Voor alle andere β is het voor elke α mogelijk, om een $\gamma,\rho \in \mathbb{Z}[i]$ te vinden met $d(\rho,0) < d(\beta,0)$ zodat $\alpha = \gamma\beta + \rho$

Bewijs. (a). Zonder verlies van algemeenheid kunnen we aannemen dat $m = 0$ en $n > 0$, Verder nemen we $\alpha = n$ als tegenvoorbeeld. Stel we kunnen n nu schrijven als $n = \gamma\beta + \rho$, dan willen we dat $d(\rho,0) < d(\beta,0) = n$. Maar aangezien $\beta = n+ni$ moet ρ een veelvoud zijn van n. Dus nemen we $\rho = 0$ om te voldoen aan $d(\rho,0) < d(\beta,0)$, maar dan moet er een $\gamma \in \mathbb{Z}[i]$ bestaan zodat $n = \gamma(n+ni)$, maar dit is niet mogelijk.

(b). Stel we hebben nu een $\alpha,\beta \in \mathbb{Z}[i]$, met $\beta \neq 0$. Dan moeten we nu een $\gamma,\rho \in \mathbb{Z}[i]$ vinden, zodat $\alpha = \gamma\beta + \rho$ en verder $d(\rho,0) < d(\beta,0)$.

Laat $\alpha = a+bi$ en $\beta = c+di$. Dan bestaat er $u,v \in \mathbb{Q}$ zodat $\frac{\alpha}{\beta} = u+vi$. Nu kunnen we gehele getallen $k,m \in \mathbb{Z}$ vinden zodat $|u-k| \leq \frac{1}{2}$ en $|v-m| \leq \frac{1}{2}$. Neem nu $p = u-k$ en $q = v-m$, dan is het duidelijk dat $\frac{\alpha}{\beta} = (k+mi)+(p+qi)$. Met andere woorden:

$$\alpha = (k+mi)\beta + (p+qi)\beta$$

$k+mi \in \mathbb{Z}[i]$ en dus ook $(k+mi)\beta \in \mathbb{Z}[i]$. Verder $(p+qi)\beta = \alpha - (k+mi)\beta \in \mathbb{Z}[i]$. Neem nu $\gamma = k+mi$ en $\rho = (p+qi)\beta$. Dan $\alpha = \gamma\beta + \rho$. Nu hoeven we alleen nog aan te tonen dat

$d(\rho,0) < d(\beta,0)$. Dit gaat via een rechtstreekse berekening, gebruikmakend van $|p|,|q| \leq \frac{1}{2}$:

$$\begin{aligned}
d(\rho,0) &= d((p+qi)\beta,0) = d((p+qi)(c+di),0) && (20)\\
&= d(pc-dq+(pd+cq)i,0) && (21)\\
&= \max\{|pc-dq|,|pd+cq|\} && (22)\\
&\leq \max\{|pc|+|dq|,|pd|+|cq|\} && (23)\\
&\leq \max\left\{\frac{|c|+|d|}{2},\frac{|d|+|c|}{2}\right\} && (24)\\
&= \frac{|c|+|d|}{2} && (25)\\
&< \max\{|c|,|d|\} = d(\beta,0) && (26)
\end{aligned}$$

De $<$ geldt omdat $|c| \neq |d|$ wat klopt omdat $\beta \neq i^u(v+vi), u,v \in \mathbb{Z}$. □

Dus alleen als $\beta = i^m(n+ni)$, kunnen we niet altijd gebruik maken van delen met rest. Dit gebeurt dus bij maar een enkel priemgetal, namelijk $1+i$. Dit priemgetal was al een speciaal geval, laten we daarom ook hier het als een speciaal geval behandelen. Een manier om nu een Legendre functie op te stellen gaat via een functie $S(x,p)$ die de veelvouden in een vierkant telt

$$S(x,p) = \#\{z : d(z,0) \leq x, p|z, z \neq 0\} \tag{27}$$

De Legendre Functie opstellen kan nu zoals we gewend zijn:

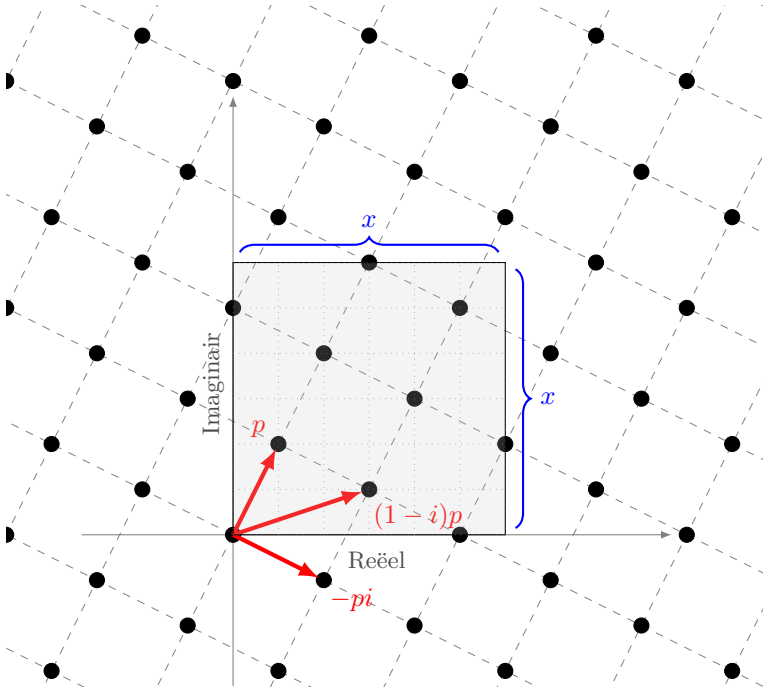
$$\phi_{\mathbb{Z}[i];\mathcal{C}}(x,a) = S(x,1) - \sum_i S(x,p_i) + \sum_{i<j} S(x,p_ip_j) - \sum_{i<j<k} S(x,p_ip_jp_k) + \ldots \tag{28}$$

Hierbij zouden we p_i in het eerste kwadrant van $\mathbb{Z}[i]$ kunnen nemen, om de eenheden niet dubbel te tellen. $S(x,p)$ kan berekent worden in $O(log(|p|))$ tijd, wat een verbetering is ten opzichte van $C(x)$. $S(x,p)$ zijn alle veelvouden van p en $\overline{p}$ in het vierkant met hoekpunten $-x-xi$, $-x+xi$, $x-xi$, $x+xi$. Dus dit vierkant heeft zijdes van lengte $2x+1$. Als $p = \overline{p}$ of $p = 1+i$, dan moeten we uitkijken dat we deze niet dubbel gaan tellen. Een ander problem is dat p en $\overline{p}$ beide delers zijn van $N(p)$. Deze norm ligt op de reële as dus ook hier geldt $N(p) = \overline{N(p)}$. Nu telt $S(x,p)$ alle veelvouden in een vierkant met een middelpunt in de oorsprong en zijden van lengte $2x+1$. Dit kunnen we natuurlijk opdelen in stukken van 4 rechthoeken van omvang $(x+1)\times x$. Het tellen van priemgetallen in dit rechthoek doen we met de functie $B(x,p)$.

2.2.1 Roosterpunten in een vierkant

Figuur 1 Laat een schematische weergave zien van deze $B(x,p)$. De lijn $x+0i$ wordt niet meegeteld, deze bevat namelijk dezelfde getallen als op de lijn $0+xi$, maar dan vermenigvuldigt met een eenheid. We moeten dus één van beide lijnen niet mee laten doen. Alle veelvouden van $p = a+bi$ zijn van de vorm $(c+di)p = (c+di)(a+bi) = ca-bd+(cb+ad)i$ we tellen dus alle $c+di$ zodat $ca-bd$ binnen de zijdes blijft en zodat $cb+ad$ binnen de zijdes blijft, dus $B(x,p) = \#\{c+di : 0 < ca-bd \leq x, 0 \leq ad+bc \leq x\}$. Let hierbij op dat we de veelvouden op de reële lijn (of imaginaire lijn) ook meetellen. Laten we nu $S(x,p)$ berekenen met behulp van $B(x,p)$ en laten we verder gevallen waar we niet mogen delen met rest apart behandelen:

$$\begin{aligned}
S(x,1) &= \lfloor 2x+1 \rfloor^2 - 1 && (29)\\
S(x,b+bi) &= \left\lfloor \frac{S(\lfloor \frac{x}{b} \rfloor,1)}{2} \right\rfloor && (30)\\
S(x,a) &= S\left(\left\lfloor \frac{x}{|a|} \right\rfloor,1\right) \qquad ,a \in \mathbb{Z} && (31)\\
S(x,p) &= 4B(x,p) && (32)\\
B(x,a+bi) &= \#\{(k,l) : |ak-bl| \leq x \wedge |bk+al| \leq x, k+li \neq 0\} && (33)
\end{aligned}$$



Figuur 1: Rooster van punten gegenereerd door het priemgetal $p = 1 + 2i \in \mathbb{Z}[i]$ en $-pi$, het doel is het tellen van alle veelvouden van p in en op het vierkant met zijde $x = 6$, dit is de essentie van $B(x, p)$, de lijn $u + 0i$ wordt niet geteld omdat we die kunnen krijgen door de lijn $0 + vi$ te vermenigvuldigen met een eenheid, daarom tellen we in essentie niet in een vierkant maar een rechthoek, in dit voorbeeld $B(6, 1 + 2i) = 8$

$S(x, 1)$ telt alle getallen in een vierkant van omvang $(2x + 1) \times (2x + 1)$ behalve het getal 0. Dan komt het speciale geval $S(x, b+bi)$ Wat eigenlijk gewoon neer komt op een soort dambord patroon, dit wordt opgelost met een simpele deling en delen door 2. Dan hebben we natuurlijk $S(x, a)$ met $a \in \mathbb{Z}$. De veelvouden zien er uit als vierkanten in het complexe vlak en kunnen we dus ook berekenen door een simpele deling. De rest gaan we eigenlijk oplossen met behulp van $B(x, p)$ en het versnellen van deze functie. Wat dus niets anders is dan het tellen van alle getallen p in een vierkant van zijde x, waarbij we een horizontale of verticale zijde erbij nemen.

Lemma 2.4. *Het kleinste getal $p \in \mathbb{Z}_{>0}$ dat deelbaar is door een $\pi = a+bi \in \mathbb{Z}[i]$ met $\gcd(a, b) = 1$ is $p = N(\pi)$*

Bewijs. Stel er is een getal $0 < q < N(p)$ zodat $\pi | q$, dan $(a + bi)(c + di) = q$. Dus $(ac - bd) + (bc + ad)i = q + 0i$. Dat geeft ons de vergelijking $bc + ad = 0$ en dus $d = \frac{-bc}{a}$. En ook $ac - bd = q$, als we d invullen, krijgen we dan $ac + \frac{b^2c}{a} = q$. Aangezien $\gcd(a, b) = 1$, zal q pas geheel zijn als $a|c$. Neem nu $k \in \mathbb{Z}$ zodat $c = ak$. Dan bij invulling krijgen we $q = a^2k + b^2k = N(\pi)k$ en dat is een tegenspraak. □

Ook willen we hierbij aangeven dat $(p) \cap \mathbb{Z} = p\mathbb{Z}[i] \cap \mathbb{Z}$ en dit is een ideaal. Aangezien $N(p)$ de kleinste veelvoud is van p in $\mathbb{Z}$, wordt dit ideaal dus voortgebracht door $N(p)$.

Stelling 2.5. *Voor $p = a + bi \in \mathbb{Z}[i]$ en $x \in \mathbb{Z}$ met de eigenschap dat $g = \gcd(a, b) \neq 0$, neem $y = x \bmod N(p)$ dan*

$$B(x, p) = \begin{cases} B(y, p) + \left\lfloor \frac{x}{N(p)} \right\rfloor (x + y + 1) & g = 1 \\ B\left(\left\lfloor \frac{x}{g} \right\rfloor, \frac{p}{g} \right) & g > 1 \end{cases} \tag{34}$$

waarbij $c \bmod d = c - d \left\lfloor \frac{c}{d} \right\rfloor$, $c, d \in \mathbb{Z}$

Bewijs. Ten eerste als $g > 1$ is het zo dat alle veelvouden van $a + bi$ allemaal g als deler hebben. Dus als we delen door g, zullen we ook de grens moeten schalen door deling met g.

Als $g = 1$, dan merken we op dat $N(p)$ de kleinste veelvoud is van p in $\mathbb{Z}$. Dus alle veelvouden van p in het vierkant met zijde $N(p)$, kunnen we kopiëren op de posities van veelvouden van $N(p)$. Vanzelfsprekend zijn alle nieuwe punten ook weer veelvouden van p. Dit zijn tevens alle veelvouden, omdat als we een veelvoud nemen van p we er een veelvoud van $N(p)$ kunnen aftrekken zodat de Chebyshev afstand tot de oorsprong weer kleiner is dan $N(p)$. $B(y, p)$ berekent nu het aantal veelvouden van p in een omgeving van $y \times y$. We voegen nu rechthoeken toe die een zijde hebben die een veelvoud is van $N(p)$. Het aantal veelvouden van p op een horizontale of verticale lijn van lengte $N(p) - 1$ is precies 1, immers deze lijn beslaat $N(p)$ punten en $N(p)$ is de kleinste gehele veelvoud van p. Dus in een rechthoek van met dimensie $x \times x - y$ vinden we precies $x \frac{x-y}{N(p)}$ veelvouden. Laten we deze rechthoek boven het vierkant van zijde y plaatsen. Dan is er nog een rechthoek over met zijden $x - y \times y$, deze bevat precies $y \frac{x-y}{N(p)}$. Ook moeten we nog ofwel een horizontale of verticale zijde meerekenen om alle gehele veelvouden van $N(p)$ te tellen.

Dit geeft ons nog eens $1 \cdot \frac{x-y}{N(p)}$ veelvouden. Bij elkaar opgeteld:

$$B(x, p) = B(y, p) + \frac{x - y}{N(p)}(x + y + 1) = B(y, p) + \left\lfloor \frac{x}{N(p)} \right\rfloor (x + y + 1)$$

□

Stelling 2.6. *Voor* $p = a + bi \in \mathbb{Z}[i]$ *en* $x \in \mathbb{Z}$*, waarvoor* $\gcd(a, b) = 1$ *en* $\frac{N(p)}{2} \leq x < N(p)$*, geldt:*

$$B(x, p) = B(N(p) - x - 1, p) + 2x - N(p) + 1 \tag{35}$$

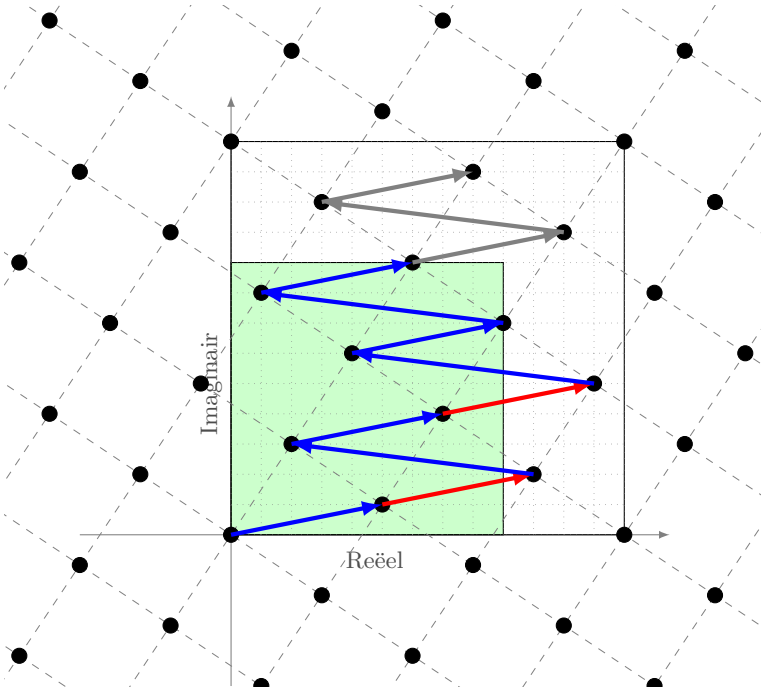
Bewijs. Laten we een vierkant van zijde $N(p) - 1$ pakken en hier rechthoeken van aftrekken zodat we een rechthoek van zijde x overhouden. Dus eigenlijk halen we eerst twee rechthoeken van zijde $(N(p) - 1) \times (N(p) - 1 - x)$ en $(N(p) - 1 - x) \times (N(p) - 1)$ eraf. En dan moet er vanwege overlap een vierkant van zijde $N(p) - 1 - x$ weer bij opgeteld worden. Waarom we geen $N(p)$ doen, is omdat we weten dat de enige veelvouden van p die ook deelbaar zijn door $N(p)$ zich bevinden op $N(p)$, $N(p)i$ en $N(p) + N(p)i$. En hierin zijn we niet geïnteresseerd. De rechthoeken met een zijden $N(p) - 1 - x$ en $N(p) - 1$ bevatten $N(p) - 1 - x$ veelvouden. En $B(N(p) - 1, p) = N(p) - 1$, bij elkaar opgeteld levert dat ons:

$$B(x, p) = N(p) - 1 - 2(N(p) - 1 - x) + B(N(p) - x - 1, p) = B(N(p) - x - 1, p) + 2x - N(p) + 1$$

□

Met deze stelling kunnen we $B(x, p)$ voor $\frac{N(p)}{2} < x < N(p)$ berekenen uit $B(x, p)$ voor $x \leq \frac{N(p)}{2}$. Wat het nog sneller maakt voor het berekenen van het aantal roosterpunten in een vierkant. Nou zijn er natuurlijk nog een aantal triviale en ook niet-triviale gevallen die het nog sneller maken en voor kleine gevallen van p kun je exacte vergelijkingen vinden die werken voor alle x. Zo kunnen we aantonen dat $B(x, 1+2i) = \left\lfloor \frac{x(x+1)}{5} \right\rfloor$, maar in de meeste gevallen zijn er geen makkelijke exacte gevallen te vinden. Toch zijn er snellere methoden die in figuur 2 en figuur 3 worden weergegeven en toegelicht.

In Figuur 4 zien we een schematische weergave van een ander sneller algoritme, hierbij beginnen we bij $p = a + bi$ en in één keer berekenen we dan links en rechts het aantal veelvouden dat zich in het vierkant bevind en op de lijn $z = p + piu$ met $u \in \mathbb{Z}$. Dan gaan we naar $2p$ en doen hetzelfde, etc. Op een geven moment gaan de veelvouden van p het vierkant uit en komen er fantoomlijnen. De snelheid van dit algoritme hangt af van het aantal fantoomlijnen, het is makkelijk te zien dat het aantal veelvouden kp in het vierkant met $k \in \mathbb{N}^+$, er $\left\lfloor \frac{x}{\max a,b} \right\rfloor$ zijn. Het aantal totaal aantal



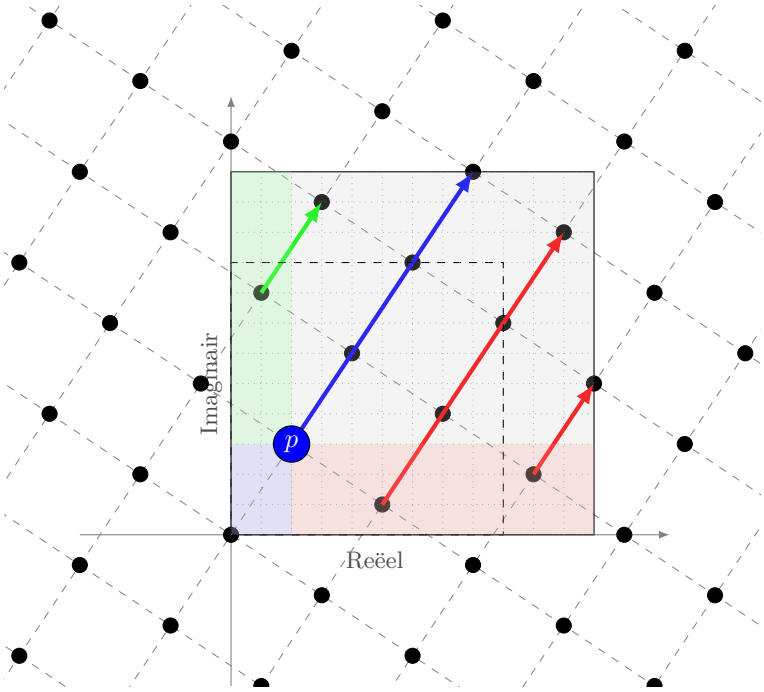
Figuur 2: een rooster gevormd door $p = 2 + 3i$, Hier staat schematisch hoe elk veelvoud gevonden zou kunnen worden. Alle gevonden veelvouden buiten het groene vierkant tellen niet mee, aangegeven met rood. En de rest in of op het groene vierkant telt wel mee en is aangegeven met blauw. De grijze pijlen worden in de stelling overgeslagen omdat ze toch al buiten het vierkant liggen. Het buitenste vierkant heeft zijde $N(p)$ en kun je zien als een cylinder. Elke pijl die naar links gaat passeert namelijk de rechterzijde en komt er links uit (hoewel dat niet zo is getekend). De eerste pijl wijst naar $\alpha + 1i$, de tweede naar $2\alpha + 2i$, etc.

iteraties is de laatste k zodat er een u bestaat zodat de vergelijking $\mathrm{Re}(kp+piu) = ak - bu \leq x$ en $\mathrm{Im}(kp + piu) = bk + au \leq x$. Met een simpele berekening kunnen we aantonen dat $k = \left\lfloor \frac{a+b}{N(p)} x \right\rfloor$, en dus is de complexiteit van het algoritme $O\left(\frac{a+b}{N(p)} x\right)$. De formule die hierbij hoort is als volgt:

Stelling 2.7. *Neem aan dat $a \leq b$ en neem $L = \left\lfloor \frac{ax}{N(p)} \right\rfloor$, $R = \left\lfloor \frac{bx}{N(p)} \right\rfloor$ en $E = \left\lfloor \frac{(a+b)x}{N(p)} \right\rfloor$*

$$\begin{aligned} B(x,p) =& E + \sum_{t=1}^{L} \left(\left\lfloor \frac{tb}{a} \right\rfloor + \left\lfloor \frac{ta}{b} \right\rfloor \right) + \sum_{t=L+1}^{R} \left(\left\lfloor \frac{ta}{b} \right\rfloor + \left\lfloor \frac{x-ta}{b} \right\rfloor \right) \\ &+ \sum_{t=R+1}^{E} \left(\left\lfloor \frac{x-tb}{a} \right\rfloor + \left\lfloor \frac{x-ta}{b} \right\rfloor \right) \end{aligned} \tag{36}$$

Bewijs. We hebben hier figuur 4 in vier stukken gehakt. Ten eerste hebben we alle punten op de blauwe lijn die een helling heeft van $\frac{b}{a} \geq 1$. Deze punten tellen we met E, deze punten liggen ook buiten het vierkant, deze zullen we later op moeten heffen. Nu merken we op dat om de rode diagonalen te tellen, we een aantal hoekpunten tegenkomen. Aangezien $a \leq b$, zullen we eerst links onderin een hoekpunt tegenkomen bij het tellen van rode diagonalen. Voordat we daar aan belanden kunnen we alvast de andere diagonalen tellen, de punten die op deze diagonalen zijn bevinden mogen niet buiten het vierkant treden. Dat betekent dat $\mathrm{Re}(tp + piu) = at - bu \geq 0$ moet gelden aan de linkerkant van de blauwe diagonaal en $\mathrm{Im}(tp+piu) = bt+au \geq 0$ rechts ervan. Respectievelijk zijn er dan $\left\lfloor \frac{at}{b} \right\rfloor$ en $\left\lfloor \frac{bt}{a} \right\rfloor$ veelvouden op de rode lijnen links en rechts. Het hoekpunt



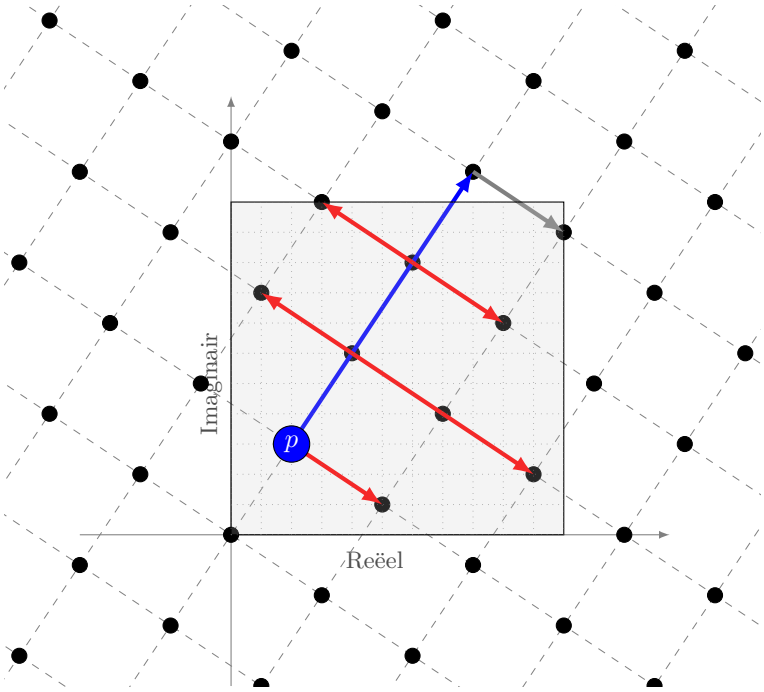
Figuur 3: een rooster gevormd door $p = 2+3i$, het bevat vier delen, de blauwe regio, hier bevinden zich geen veelvouden van p. Groen en Rood geven de verticale en horizontale regionen aan die wel veelvouden van p kunnen bevatten (randen niet meegerekend). De blauwe cirkel is p zelf en doet ook mee en hoort als uitzondering bij de blauwe regio. Het vierkant gevormd door de stippellijn heeft zijde x, en hoewel we steeds dezelfde veelvouden in rood en groen gebruikt, zien we dat er een veelvoud in het rode gebied overbodig wordt

linksonder verandert één van de condities, de rode lijnen mochten eerst niet door de onderkant en nu mogen ze niet door de linkerzijde van het vierkant heen. Als we een lijn loodrecht aan de blauwe lijn tekenen die de linkeronderhoek snijdt, hoeveel blauwe cirkels hebben we dan al geteld? Elke keer als we p opschuiven op de blauwe lijn, gaan we b stappen omhoog en snijdt een loodrechte rode diagonaal de onderzijde $\frac{N(p)}{b}$ stappen verder. Na dit hoekpunt gepasseerd te hebben vervangen we de conditie $\mathrm{Im}(tp + piu) = bt + au \geq 0$ door $\mathrm{Re}(tp + piu) = at - bu \leq x$ en dus vervangen we $\left\lfloor \frac{bt}{a} \right\rfloor$ voor $\left\lfloor \frac{x-at}{b} \right\rfloor$. Op soortgelijke wijze komen we vervolgens bij de rechterbovenhoek, waar we nu de andere conditie veranderen. Als we nu sommeren over al deze rode diagonalen, plus de blauwe diagonaal, krijgen we de gewenste formule. De blauwe cirkels die uit het vierkant steken, worden opgeheven door de laatste rode fantoom diagonalen, die zullen soms een negatieve waarde opleveren, die de overtollige cirkels boven de bovenzijde weghaalt. □

Hoewel deze formule al vrij snel is, kunnen we deze methode nu in logaritmische tijd berekenen, door wat slimme trucjes toe te passen op de sommen.

Stelling 2.8. *Laat* $a, b \in \mathbb{Z}$ *en* $\gcd(a, b) = 1$*, dan*

$$\sum_{t=1}^{a-1} \left\lfloor \frac{tb}{a} \right\rfloor = \frac{(a-1)(b-1)}{2} \tag{37}$$



Figuur 4: een rooster gevormd door $p = 2+3i$, deze geeft schematisch een erg snel algoritme weer, we volgen de blauwe pijl omhoog en door simpele delingen weten we hoeveel veelvouden er op elke rode lijn ligt, de grijze lijnen zijn fantoomlijnen, maar mogen niet overgeslagen worden, omdat ze nog wel veelvouden vinden.

Bewijs.

$$\begin{aligned}
\sum_{t=1}^{a-1} \left\lfloor \frac{tb}{a} \right\rfloor &= \sum_{t=1}^{a-1} \frac{tb}{a} - \sum_{t=1}^{a-1} \left\{ \frac{tb}{a} \right\} \\
&= \frac{b}{a} \sum_{t=1}^{a-1} t - \sum_{t=1}^{a-1} \left\{ \frac{tb}{a} \right\} \\
&= \frac{b}{a} \frac{a(a-1)}{2} - \sum_{t=1}^{a-1} \left\{ \frac{tb}{a} \right\} \\
&= \frac{b(a-1)}{2} - \sum_{t=1}^{a-1} \left\{ \frac{tb}{a} \right\}
\end{aligned}$$

Hierbij is $\{x\}$ het deel achter de komma van x. Nu is het zo dat $\gcd(a,b) = 1$ dus als $1 \leq t \leq a-1$, dan komt elke $\left\{\frac{tb}{a}\right\}$ overeen met een unieke breuk en dit zijn ook direct alle mogelijke breuken met deler a. Dus $\sum_{t=1}^{a-1} \left\{\frac{tb}{a}\right\} = \sum_{t=1}^{a-1} \frac{t}{a} = \frac{1}{a} \sum_{t=1}^{a-1} t = \frac{a-1}{2}$. In totaal krijgen we dan:

$$\begin{aligned}
\frac{b(a-1)}{2} - \sum_{t=1}^{a-1} \left\{ \frac{tb}{a} \right\} &= \frac{b(a-1)}{2} - \frac{a-1}{2} \\
&= \frac{(b-1)(a-1)}{2}
\end{aligned}$$

□

Laten we nu dezelfde stelling uitbreiden, door te sommeren over k.

Stelling 2.9. *Laat $a, b, k \in \mathbb{Z}$ en $\gcd(a, b) = 1$, en neem $u = \left\lfloor \frac{k}{a} \right\rfloor$, dan*

$$\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor = \frac{(a-1)(b-1)}{2} u + \frac{u(u-1)}{2} ab + \sum_{t=ua}^{k} \left\lfloor \frac{tb}{a} \right\rfloor \tag{38}$$

Bewijs.

$$\begin{aligned}
\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor &= \sum_{m=0}^{u-1} \sum_{t=0}^{a-1} \left\lfloor \frac{tb}{a} + mb \right\rfloor + \sum_{t=ua}^{k} \left\lfloor \frac{tb}{a} \right\rfloor \\
&= \sum_{m=0}^{u-1} amb + u \sum_{t=0}^{a-1} \left\lfloor \frac{tb}{a} \right\rfloor + \sum_{t=ua}^{k} \left\lfloor \frac{tb}{a} \right\rfloor \\
&= \frac{u(u-1)}{2} ab + \frac{(a-1)(b-1)}{2} u + \sum_{t=ua}^{k} \left\lfloor \frac{tb}{a} \right\rfloor
\end{aligned}$$

□

Nu herschrijven we de bovenstaande stelling totdat de volgende stelling ontstaat:

Stelling 2.10. *Laat $a, b, k \in \mathbb{Z}$ en $\gcd(a, b) = 1$, en neem $u = \left\lfloor \frac{k}{a} \right\rfloor$, dan*

$$\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor = bu(k+1) - \frac{u}{2}(bau + b + a - 1) + \sum_{t=1}^{k-ua} \left\lfloor \frac{tb}{a} \right\rfloor \tag{39}$$

Bewijs. We gebruiken dat:

$$\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor = \frac{(a-1)(b-1)}{2} u + \frac{u(u-1)}{2} ab + \sum_{t=ua}^{k} \left\lfloor \frac{tb}{a} \right\rfloor$$

Het enige wat ons nu interesseert is die laatste som.

$$\begin{aligned}
\sum_{t=ua}^{k} \left\lfloor \frac{tb}{a} \right\rfloor &= \sum_{t=ua}^{k} \frac{tb}{a} - \sum_{t=ua}^{k} \left\{ \frac{tb}{a} \right\} \\
&= \sum_{t=ua}^{k} \frac{tb}{a} - \sum_{t=1}^{k-ua} \left\{ \frac{tb}{a} \right\} \\
&= \sum_{t=0}^{k-ua} \frac{(t+ua)b}{a} - \sum_{t=0}^{k-ua} \left\{ \frac{tb}{a} \right\} \\
&= \sum_{t=0}^{k-ua} \frac{tb}{a} + \sum_{t=0}^{k-ua} ub - \sum_{t=0}^{k-ua} \left\{ \frac{tb}{a} \right\} \\
&= \sum_{t=1}^{k-ua} \left\lfloor \frac{tb}{a} \right\rfloor + ub(k - ua + 1)
\end{aligned}$$

Laten we nu deze som invoegen:

$$\begin{aligned}
\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor &= \frac{(a-1)(b-1)}{2} u + \frac{u(u-1)}{2} ab + \sum_{t=1}^{k-ua} \left\lfloor \frac{tb}{a} \right\rfloor + ub(k - ua + 1) \\
&= bu(k+1) - \frac{u}{2}(bau + b + a - 1) + \sum_{t=1}^{k-ua} \left\lfloor \frac{tb}{a} \right\rfloor
\end{aligned}$$

□

Nu we een formule hebben om een k te verkleinen, kunnen we kijken of we misschien b kunnen verkleinen. Als we vervolgens a en b nog kunnen omwisselen, zouden we een soort van euclidisch algoritme moeten krijgen.

Stelling 2.11. *Laat $a, b, k \in \mathbb{Z}$, en neem $v = \lfloor \frac{b}{a} \rfloor$, dan*

$$\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor = \frac{vk(k+1)}{2} + \sum_{t=1}^{k} \left\lfloor \frac{t(b \bmod a)}{a} \right\rfloor \tag{40}$$

Bewijs. Merk op dat $b \bmod a = b - va$, dus:

$$\begin{aligned}
\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor &= \sum_{t=1}^{k} \left\lfloor \frac{t(b - va + va)}{a} \right\rfloor \\
&= \sum_{t=1}^{k} \left\lfloor \frac{t(b - va)}{a} \right\rfloor + \sum_{t=1}^{k} tv \\
&= \sum_{t=1}^{k} \left\lfloor \frac{t(b \bmod a)}{a} \right\rfloor + \frac{vk(k+1)}{2}
\end{aligned}$$

□

Stelling 2.12. *Laat $a, b, k \in \mathbb{Z}$ en $m = \lfloor \frac{kb}{a} \rfloor$ met $b \nmid ta$ voor $1 \leq t \leq m$, dan*

$$\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor = mk - \sum_{t=1}^{m} \left\lfloor \frac{ta}{b} \right\rfloor \tag{41}$$

Bewijs. Neem $\lfloor \frac{tn}{a} \rfloor = m'$, dan is het duidelijk dat de volgende dingen gelden:

$$\begin{aligned}
m' \leq \frac{tn}{a} < m' + 1 \\
\frac{m'a}{b} \leq t < \frac{(m'+1)a}{b}
\end{aligned}$$

Als we er nu even van uit gaan dat $\frac{(m'+1)a}{b} < k$, dan vinden we dus dat het laatste geval $\left\lceil \frac{(m'+1)a}{b} \right\rceil - \left\lceil \frac{m'a}{b} \right\rceil$ keer voor komt. Dus de som van al die $m' \leq \frac{tb}{a} < m' + 1$ is dan $m' \left(\left\lceil \frac{(m'+1)a}{b} \right\rceil - \left\lceil \frac{m'a}{b} \right\rceil \right)$. De laatste m' die gemeten wordt is $m' = m$, maar dan is $\frac{(m'+1)a}{b} \geq k$. Dus voor die laatste m' gebruiken we $k + 1 - \lceil \frac{ma}{b} \rceil$ en dus wordt de som daar $m(k + 1 - \lceil \frac{ma}{b} \rceil)$. Dit betekent dat:

$$\begin{aligned}
\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor &= m(k + 1 - \left\lceil \frac{ma}{b} \right\rceil) + \sum_{m'=1}^{m-1} m' \left(\left\lceil \frac{(m'+1)a}{b} \right\rceil - \left\lceil \frac{m'a}{b} \right\rceil \right) \\
&= m(k+1) - \sum_{t=1}^{m} \left\lceil \frac{ta}{b} \right\rceil
\end{aligned} \tag{42}$$

De laatste stap komt omdat de som via telescoping teruggebracht kan worden tot een kleiner geheel. Nu willen we alleen van alle naar boven afrondingen, naar beneden afronden om weer aan te sluiten bij de andere stellingen. Hiervoor moeten we aannemen dat $b \nmid ta$ voor alle $1 \leq t \leq m$ zodat er altijd wordt afgerond naar boven, dan geldt er namelijk dat $\lceil x \rceil = \lfloor x \rfloor + 1$ en dus:

$$m(k+1) - \sum_{t=1}^{m} \left\lceil \frac{ta}{b} \right\rceil = mk - \sum_{t=1}^{m} \left\lfloor \frac{ta}{b} \right\rfloor \tag{43}$$

□

Met de bovenstaande stelling kunnen we een algoritme maken dat in logaritmische tijd, voordat we hieraan beginnen moeten we nog één ding ontleden:

Stelling 2.13. *Laat $a, b, c, m, n, x \in \mathbb{Z}$ en $\gcd(a, b) = 1$ en neem $g, h \in \mathbb{Z}$ zodat $ag + bh = x$, dan geldt dat:*

$$\sum_{t=m}^{n} \left\lfloor \frac{x - tb}{a} \right\rfloor = \left(g + \left\lfloor \frac{\delta}{a} \right\rfloor b \right) (n - m + 1) + \sum_{t=1}^{n-m+1+\bar{\delta}} \left\lfloor \frac{tb}{a} \right\rfloor - \sum_{t=1}^{\bar{\delta}} \left\lfloor \frac{tb}{a} \right\rfloor \tag{44}$$

waarbij $\delta = h - n - 1$, $\bar{\delta} = h - n - 1 \bmod a$

Bewijs. Om deze formule te bewijzen, substitueren we eerst $ag + bh = x$ en vervolgens gaan we alles proberen uit te schrijven. □

Met gebruik van al deze stellingen kunnen we een $O(\log(a))$ algoritme opstellen, waar $p = a+bi$ en $b < a$, voor het berekenen van $B(x, p)$. Als p een priemgetal is zal gelden $\gcd(a, b) = 1$, zo niet dan kunnen we toch hieraan voldoen door te delen door zowel x als p te delen door de grootste gemene deler van a en b. Om het algoritme een succes te maken proberen we ten tweede $B(x, p)$ te schrijven als een som van allemaal stukken vloersommen van de vorm:

$$\sum_{t=1}^{k} \left\lfloor \frac{tb}{a} \right\rfloor \tag{45}$$

Dit is mogelijk vanwege stellingen 2.10 en 2.13. Ook zal stelling 2.10 de waarde van k verkleinen. Dan wordt nu in het algoritme elke vloersom apart behandelt. Eerst wisselen we b en a om met behulp van stelling 2.12. Nu moet gelden $b > a$ en nog steeds $\gcd(a, b) = 1$. Nu wordt de b waarde verkleint naar $b \bmod a$ door middel van stelling 2.11 en begint het proces opnieuw. De reden waarom we stelling 2.12 mogen gebruiken is omdat als $b < a$, dan is $m < k$. We hadden met stelling 2.10 deze k al zo verkleint dat deze kleiner was dan a. Dus $m = \left\lfloor \frac{kb}{a} \right\rfloor < b$. Dus als $b|ta$ en we hebben al dat $\gcd(a, b) = 1$. Dat maakt $b|ta$ equivalent aan $b|t$ en dat kan niet omdat $t < m < b$. Als we daarna a en b omwisselen, zullen we dus nog wel k verder moeten verkleinen met stelling 2.10, om zo nog steeds aan deze condities te voldoen. Aangezien we een cyclus krijgen van $b \to b \bmod a$ en $a \leftrightarrow b$, is de snelheid vergelijkbaar met het euclidische algoritme.

Met dit algoritme zouden we een aardig snelle Legendre functie op kunnen stellen, die sneller is dan de Legendre functie met een euclidische metriek. $S(x, p)$ is het aantal veelvouden van p in het totale vierkant, dus door wat kleine aanpassingen aan $B(x, p)$ zouden we deze kunnen berekenen. $S(x, p)$ zal daarom ook in logaritmische tijd berekend kunnen worden.

$$\phi_{\mathbb{Z}[i];\mathcal{C}}(x, a, b) = S(x, b) - \sum_{i} S(x, bp_i) + \sum_{i<j} S(x, bp_i p_j) - \sum_{i<j<k} S(x, bp_i p_j p_k) + \dots \tag{46}$$

En natuurlijk in recursieve vorm:

$$\phi_{\mathbb{Z}[i];\mathcal{C}}(x, a, b) = \phi_{\mathbb{Z}[i];\mathcal{C}}(x, a - 1, b) - \phi_{\mathbb{Z}[i];\mathcal{C}}(x, a - 1, bp_a) \tag{47}$$

Ook deze Legendre functie kunnen we omzetten tot een priem telfunctie, en ook hier zullen we de priem telfunctie weer op kunnen splitsen in het vinden van priemgetallen $\equiv 1 \bmod 4$ en $\equiv 3 \bmod 4$. Dan ziet dit er zo uit:

$$\frac{1}{4}\pi_{\mathbb{Z}[i];\mathcal{C}}(x) = 2\tilde{\pi}(x; 4, 1) + \pi(x; 4, 3) + 1 \quad , x > 2 \tag{48}$$

De $\frac{1}{4}$ is hier om het aantal eenheden in $\mathbb{Z}[i]$ te compenseren, en de priemgetallen van de vorm $4k+1$ hebben allemaal een conjugatie, dus worden extra geteld. Hieruit kunnen we ook een snelle manier vinden om alle $4k + 1$ te tellen, in het vierkant vandaar dat we hier $\tilde{\pi}$ gebruiken.

$$\tilde{\pi}(x; k, p) = \left\{ y : y \text{ priem in } \mathbb{Z}, y \equiv k \pmod{p}, y = a^2 + b^2, 0 < a < b \leq x, y \leq x^2 \right\} \tag{49}$$

Ook als blijkt dat voor deze $\tilde{\pi}$ er een snelle formule bestaat kunnen we alle $4k+3$ heel snel tellen. Het probleem zit hem echter nu in de functie $S(x,p)$ hoewel die duidelijk sneller is dan $C(x)$, is deze wel erg complex. Versnellen met behulp van Meissel's methode zou misschien nog te doen zijn, hiervoor hoeven we alleen maar de Legendre functie te herschrijven zonder al te veel handigheden te gebruiken. Maar Mapes' methode gebruikt ook handigheden met behulp van de Euler totiënt functie en het binair ontleden in ϕ, wat erg lastig is om na te bootsen met een functie als $S(x,p)$, die niet dezelfde eigenschappen heeft als een simpele functie als $\left\lfloor \frac{x}{p} \right\rfloor$.

2.3 Zeven

Het zeven van priemgetallen in $\mathbb{Z}[i]$ kunnen we soortgelijk houden aan de zeef van Eratosthenes. Toch is dit niet effectief voor de priemgetallen die zich in het complexe vlak bevinden. Namelijk als a^2+b^2 een priemgetal is in $\mathbb{Z}$ dan is $a+bi$ een priemgetal in $\mathbb{Z}[i]$. Dit betekent dat in verhouding het aantal priemgetallen van de vorm $4k+1$, dus in het complexe vlak, kwadratisch is ten opzichte van $4k+3$. Als we dit zeven omzetten in een telmethode voor priemgetallen van de vorm $4k+3$, zoals we eigenlijk in de vorige paragraaf deden, is totaal niet efficiënt in dat op zicht. Laten we ons daarom vooral focussen op die priemgetallen van de vorm $4k+1$ en proberen de bijbehorende paren $a+bi$ zo min mogelijk te laten zijn. Daarom gaan we ons richten op andere zeven. Eén zo'n zeef is de zeef van Atkin, deze werkt door te kijken naar 3 verschillende rijtjes getallen. Eén van die rijtjes zijn alle getallen van de vorm x^2+y^2. In ons geval $N(x+yi) = x^2+y^2$, en aangezien de norm multiplicatief is, dan geldt als $x+yi$ een priemgetal is in $\mathbb{Z}[i]$, dan x^2+y^2 een priemgetal in $\mathbb{Z}$ (als $x,y \neq 0$). Verder is het zo dat $N(x+yi)$ een priemgetal van de vorm $4k+1$ is en $x-yi$ en $u(x+yi)$ en $u(x-yi)$ voor $u \in \mathbb{Z}[i]^*$ de enige andere priemgetallen zijn. Dus dan zijn er maar 8 mogelijke oplossingen. Andere getallen van de vorm x^2+y^2 hebben allemaal delers, dus deze getallen hebben minimaal 8 mogelijke oplossingen.

Stel $x^2+y^2 = pq$ waarbij $p = a^2+b^2$ en $q = c^2+d^2$ beide priemgetal met $p \neq q$ en $p \equiv q \equiv 1 \pmod 4$, dan is het makkelijk te zien dat er meer dan 8 oplossingen zijn. Een basis oplossing zou zijn $x = ac-bd$ en $y = ad+bc$ en de andere zou zijn $x = ac+bd$ en $y = ad-bc$. Vermenigvuldigd met 8 symmetriëen, geeft dat ons 16 oplossingen. Als $p = q$ gaat dit echter niet meer op en zijn er gewoon 8 oplossingen. Als we kijken naar het aantal basisoplossingen, zullen we zien dat als $x^2+y^2 = p$, en p is een priemgetal van de vorm $4k+1$ of een priemmacht van een priemgetal van de vorm $4k+1$, dan is er maar 1 basis oplossing. Als p geen priemmacht of priemgetal is, maar wel minstens twee verschillende priemgetallen deelt met $p \equiv q \equiv 1 \pmod 4$, dan is het aantal basis oplossingen altijd even. Dit heeft te maken met het voorbeeld van $x^2+y^2 = pq$, alleen nu zijn p en q geen priemgetallen van de vorm $4k+1$ zijn, maar zullen ze alleen priemdelers hebben van de vorm $4k+1$. Toch weten we dat ook hier nu $p = a^2+b^2$ en $q = c^2+d^2$. Omdat we elk getal dat geen priemgetal is, in twee van dit soort stukken kunnen opdelen en omdat elk paar p en q weer twee basisoplossingen toevoegt, moet het aantal basisoplossingen dan even zijn. Dit principe wordt gebruikt bij zowel de zeef van Atkin als de pythagoreaanse driehoeken.

2.3.1 Pythagoreaanse Driehoeken

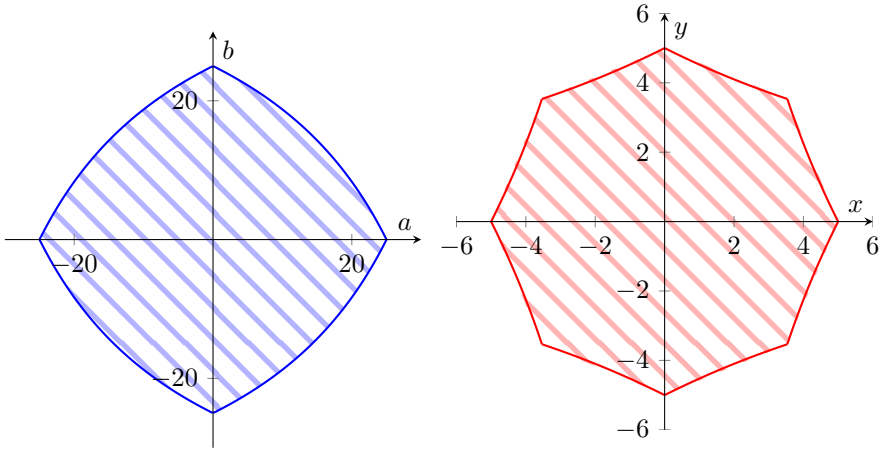
Een andere mogelijk zeef gebruikt primitieve Pythagoreaanse driehoeken, primitief want $\gcd(x,y) = 1$. Pythagoreaanse driehoeken van de vorm $a^2+b^2 = c^2$. Aangezien we elk priemgetal van de vorm $N(x+yi) = x^2+y^2 = c$ hebben, kunnen we dus ook die a en b vinden zodat $a^2+b^2 = c^2$. Een a en b die hieraan voldoen zijn $a = x^2-y^2$ en $b = 2xy$, immers wat we kunnen doen is $c^2 = N(x+yi)N(x+yi) = N((x+yi)(x+yi)) = N(x^2-y^2+2xyi) = (x^2-y^2)^2+(2xy)^2 = a^2+b^2$. Verder is het makkelijk aan te tonen dat als $\gcd(x,y) = 1$ dan ook $\gcd(x^2-y^2, 2xy) = \gcd((x+y)(x-y), 2xy) = \gcd(x+y, 2xy)\gcd(x-y, 2xy) = \gcd(x+y, 2y)\gcd(x+y, x)\gcd(x-y, 2x)\gcd(x-y, y) = \gcd(x,y)\gcd(x,y)\gcd(x,y)\gcd(x,y) = 1$. Verder weten we dat als $x+yi$ hieraan voldoet dat we de andere 7 symmetrieën kunnen vinden door middel van de afbeeldingen $x+yi \mapsto y+xi$ en $x+yi \mapsto x-yi$. Het is makkelijk na te gaan dat na herhaaldelijk toe te passen van de verschillende afbeeldingen de 8 symmetriën tevoorschijn komen. We zeggen dat $x+yi$ een primitieve Pythagoreaanse driehoek genereert als geldt: $\gcd(x,y) = 1$ en $x+y \equiv 1 \pmod 2$.

Stelling 2.14. *Als $x+yi$ een primitieve Pythagoreaanse driehoek genereert dan genereert $2x+y+xi$ ook een primitieve Pythagoreaanse driehoek.*

Bewijs. Ten eerste $\gcd(x, y) = 1$, dus dan ook $\gcd(2x + y, x) = \gcd(y, x) = 1$. Verder omdat $x + y \equiv 1 \pmod 2$, dan $2x + y + x \equiv 3x + y \equiv x + y \equiv 1 \pmod 2$ □

Sterker nog met behulp van de afbeeldingen $x + yi \mapsto y + xi$ en $x + yi \mapsto x - yi$, weten we ook dat $2y + x + yi$ en $2x - y + xi$ een primitief Pythagoreaanse driehoek voort moet brengen. Maar brengen deze getallen nou alle primitieve Pythagoreaanse driehoeken voort? De kleinste $x + yi$ die een primitief Pythagoreaanse driehoek genereert is $2 + i$. Wat we dan moeten aantonen is dat elke primitief Pythagoreaanse driehoek hier van afstamt als wij alleen de afbeeldingen $x + yi \mapsto 2x + y + xi$, $x + yi \mapsto 2x - y + xi$ en $x + yi \mapsto 2y + x + yi$ gebruiken. Verder nemen we aan dat $x > y > 0$, het is makkelijk na te gaan dat deze eigenschap bewaard blijft onder elk van deze afbeeldingen. Neem nu een getal $a + bi$, $a > b > 0$ die een primitief Pythagoreaans driehoek genereert en neem $a + bi \neq 2 + i$. Dan is het zo dat één van de volgende getallen ook een primitief Pythagoreaanse driehoek voortbrengt: $x + yi = b + (a - 2b)i$, $x + yi = b + (2b - a)i$, $x + yi = a + (b - 2a)i$, dit zijn inversen van de lineaire afbeeldingen. Het is aan te tonen dat er minimaal 1 $x + yi$ is die dan ook $x > y > 0$, $\gcd(x, y) = 1$ en waarbij $x + y \equiv 1$. Dat betekent dat $a + bi$ een afstammeling heeft. De enige $a + bi$ die geen afstammeling heeft is $2 + i$ aangezien we dan niet meer aan de eis van $x > y > 0$ kunnen voldoen. Dus alles valt terug op die $2 + i$. Als we dus beginnen bij $2 + i$ kunnen we elke $x^2 + y^2$ vinden door middel van lineaire afbeeldingen. Verder vanwege de keuze van $x > y > 0$ hebben we geen symmetrieën meer en als we dus een combinatie van $x + yi$ twee keer tegenkomen dan is dit niet een priemgetal. Als we $x + yi$ 1 keer zien voorkomen in deze stamboom dan is het of een priemgetal of een priemmacht.

Om de snelheid van dit algoritme te weten, moeten we weten hoeveel stappen het moet doorlopen. Lehmer bewees dat alle primitieve driehoeken waarvan hun omtrek kleiner is dan L kan worden afgeschat door $\frac{\ln(2)}{\pi^2} L$. Met de methode gaan we precies deze primitieve Pythagoreaanse driehoeken af. Dus is de tijdscomplexiteit dan $O(L)$. Maar merk hier op dat we L zo kunnen kiezen zodat er $a, b, c, x, y \in \mathbb{Z}$ bestaan zodat $L = |a| + |b| + |c| = |x^2 - y^2| + |2xy| + |x^2 + y^2|$. In figuur 2.3.1 zien we voor $L = 50$ beide vormen terug. [5] Wat we echt willen is gebruik maken



Figuur 5: Links zijn alle mogelijke $a + bi$ gemarkeerd zodat we een primitieve Pythagoreaanse driehoek met zijden a, b, c kunnen vormen zodat $a^2 + b^2 = c^2$ en $|a| + |b| + |c| \leq 50$. Hetzelfde rechts maar dan wordt gekeken naar alle $x + yi$, zodat $a = x^2 - y^2$, $b = 2xy$, $c = x^2 + y^2$ en $|a| + |b| + |c| \leq 50$

van de norm van elke $x + yi$, deze norm is $x^2 + y^2 = c$. Ook hier heeft Lehmer iets voor bewezen, namelijk dat het aantal primitieve Pythagoreaanse driehoeken met $c \leq L$ asymptotisch is aan $\frac{L}{2\pi}$. We gaan dan dus ook precies zoveel keer langs een primitieve Pythagoreaanse driehoek, daarmee

is de tijdscomplexiteit ook hier $O(L)$. Verder hebben we met deze methode alle priemgetallen en priemmachten gevonden. Dus moeten we alle gevonden priemgetallen bij langs en alle machten wegstrepen. Aangezien we maximaal $O(L)$ priemgetallen vinden, kan dat dus in $O(L\log(L))$ tijd. Een betere methode is om een subklasse te gebruiken van de zeef van Atkin, zoals beschreven in [1].

2.3.2 Zeef van Atkin

Deze zeef van Atkin is in eerste instantie niet veel sneller dan een zeef van Eratosthenes of een hierboven beschreven methode, maar qua complexiteit toch lichtelijk sneller en het kan direct gebruikt worden op de gehelen van Gauss. Ook kan deze methode door de zeef in stukken te hakken erg efficiënt uitgevoerd worden. Zo nemen we $60L$ hier als onderlimiet en $60B$ als spanwijdte, dus $60L+60B$ is bovenlimiet van een aantal getallen die we gaan zeven. De waardes van L zullen per iteratie met B worden verhoogd en de waarde van B zal later duidelijk worden. Verder gebruikt het algoritme een techniek genaamd wielfactorisatie. De grootte van het wiel is hier 60 en we weten dat als $d < 60$ en een copriem is van 60, dan kan $60k + d$ priemgetal opleveren voor een zekere $k \geq 0$. Als d niet copriem was, dan uiteraard niet. In dit geval zoeken we naar priemgetallen van de vorm $4k + 1$, dus zullen ook alle copriemen van deze vorm moeten zijn. Dus gaan we elke $d \in \{1, 13, 17, 29, 37, 41, 49, 53, 57\}$ bij langs. Het laatste punt is dat we hier gebruik maken van de formule $4x^2 + y^2$ in tegenstelling tot $x'^2 + y'^2$. Het verschil is dat we nu aannemen dat x' even is, dus eigenlijk verandert er nog niks.

Het algoritme zal eerst een geheugen aanmaken en op 0 zetten. In het geheugen betekent 0, dat op die positie geen priemgetal is en 1 dat er wel een priemgetal is. Daarna vinden we alle $L \leq k < L + B$ met $4x^2 + y^2 = 60k + d$, elke keer als we zo'n k vinden gaan we het geheugen aanpassen van 0 naar 1 of juist terug. Daarna gaan we alle priemgetallen onder $\sqrt{60(L + B)}$ bij langs en ook alle $60k + d$ die het kwadraat van dit priemgetal delen. Natuurlijk hoort op deze plaatsen geen 1, want dit zijn geen priemgetallen, dus hier wordt het geheugen op 0 gezet. In

Algoritme 1 Subklasse van de Zeef van Atkin

```
1: m_{0...B-1} ← {0...0}
2: for all k met 4x^2 + y^2 = 60k + d,  L ≤ k < L + B,  x, y > 0 do
3:     m_{k-L} ← 1 − m_{k-L}
4: end for
5: for all q ≥ 7, q priem, q^2 < 60(L + B) do
6:     for all k met q^2 | 60k + d do
7:         m_{k-L} ← 0
8:     end for
9: end for
```

algoritme 1 wordt een deel van de zeef weergegeven in pseudocode. Dit is niet de totale zeef, zo moet dit onderdeel voor elke L en d opnieuw uitgevoerd worden. Buiten deze subklasse om, zitten dus nog twee lussen.

Regel 1

Hier wordt het geheugen op 0 gezet, we gebruiken hier posities 0 tot en met $B - 1$. Elke positie k in het geheugen komt nu overeen met een getal van de vorm $60L + 60k + d$ in werkelijkheid.

Regel 2-4

Deze lus is gebaseerd op het feit dat als $60k+d$ een priemgetal of priemmacht is, dan zullen er een oneven aantal oplossingen zijn voor $4x^2+y^2 = 60k+d$. We hebben al gezien bij de pythagoreaanse driehoeken dat priemmachten en priemgetallen 8 oplossingen hebben voor $x'^2 + y'^2 = p$, indien $p \equiv 1 \pmod 4$. Nu nemen we dus x even en $x > 0$ en $y > 0$, waardoor we maar 1 oplossing

overhouden. Ook hadden we al gezien dat als $x'^2 + y'^2 = pq$, we meer dan acht oplossingen kunnen vinden als $p, q \equiv 1 \pmod 4$ en $p \neq q$. Nu als we een even aantal oplossingen vinden, staat ons geheugen op 0, op die positie. En als we een oneven aantal oplossingen vinden dan staat ons geheugen op 1. Hierna hoeven we dus alleen nog alle priemmachten weg te halen.

Regel 5-9

Hier gaan we de priemmachten uit het geheugen zeven. We nemen hier alle priemkwadraten onder $60(L + B)$, en dan gaan we kijken welke posities in het geheugen zo'n kwadraat delen. Als dit zo is en het geheugen staat daar op 1, dan moet dit een priemmacht zijn. Dus zullen we hier het geheugen op 0 zetten.

Nu is het zo dat de lus bij regel 2, niet een vanzelfsprekendheid is. Om al deze k, x en y te vinden die voldoen aan de vergelijking $4x^2 + y^2 = 60k + d$. Het algoritme om deze getallen te vinden, gaat ook via een wiel, maar dan van grootte 450. De wiel bestaat uit een lus met f en een lus met g. Hier nemen we de lus zodat $0 \leq f \leq 15$ en $0 \leq g \leq 30$. Verder moeten ze voldoen aan $d \equiv 4f^2 + g^2 \pmod{60}$. Deze f en g kunnen we ook voor het hele algoritme begint, geïnitialiseerd worden. Het is namelijk voor elke d al duidelijk welke f en g hier aan voldoen, of we nou in een lus zitten, of niet. Het uit de lus halen van de f en g bespaart ons al een enorme tijd. Verder f en g zijn de kleinste x en y zodat $4x^2 + y^2 = 60k + d$. En we gaan nu spelen met verlagingen van x en verhogingen van y, om zo al die k tegen te komen met $L \leq k < L + B$. Algoritme 2 laat zien hoe deze lus doorlopen kan worden.

Algoritme 2 Het berekenen van de eerste lus in Algoritme 1

```
 1: x ← f; y0 ← g; k0 ← (4f² + g² − d)/60
 2: while k0 < L + B do                          ▷ neem k0 omhoog over de bovenlimiet
 3:     k0 ← k0 + 2x + 15
 4:     x ← x + 15
 5: end while
 6: x ← x − 15                                   ▷ ga een stap terug, om net onder de bovenlimiet te komen
 7: k0 ← k0 − 2x − 15
 8: while x > 0 do
 9:     while k0 < L do                          ▷ zorg ervoor dat k0 ook boven de onderlimiet blijft
10:         k0 ← k0 + y0 + 15
11:         y0 ← y0 + 30
12:     end while
13:     k ← k0
14:     y ← y0
15:     while k < L + B do
16:         m_{k−L} ← 1 − m_{k−L}                ▷ uit Algoritme 1
17:         k ← k + y + 15
18:         y ← y + 30
19:     end while
20:     x ← x − 15
21:     k0 ← k0 − 2x − 15
22: end while
```

Regel 1

Hier beginnen we met voorbereiden van een aantal variabelen. We zetten x eerst gelijk aan f, deze x willen we maximaal houden. En we zetten y_0 gelijk aan g. Deze willen we minimaal houden. Met andere woorden, we nemen x de maximale waarde zodat $L \leq k_0 < L + B$. En we nemen y_0 minimaal zodat dit nog geldt. Vanwege de vergelijking $4x^2 + y^2 = 60k + d$, zal onze $k_0 = (4x^2 + y^2 - d)/60$.

Regel 2-5

In deze lus maximaliseren we k_0 en daarmee ook x. We nemen hier 15 als stapgrootte van x. En k_0 zouden we nu weer kunnen berekenen met behulp van $(4x^2 + y^2 - d)/60$, maar door een simpel verschil te nemen, zien we ook dat deze een stapgrootte zal hebben. Als de lus eindigt, zal k_0 een stap over de limiet van $L + B$ zitten.

Regel 6-7

Aangezien we iets te ver zijn doorgeschoten, zullen we beide x en k_0 weer een stap terug moeten doen.

Regel 8-14 en 20-21

We gaan hier een lus beginnen waar we x maximaal willen houden en y_0 minimaal. Dit gaan we doen door y_0 steeds te verhogen totdat $k_0 \geq L$, dit gaat met stappen van 30. Ook hier kunnen we berekenen wat de stapgrootte voor k_0 zal zijn in dit geval. Hierna slaan we y_0 en k_0 op in respectievelijk y en k. Deze y_0 en k_0 willen we bewaren zodat we de minimaliteit van y_0 kunnen gebruiken in een volgende iteratie. Aan het eind van de iteratie gaan we de x een enkele keer verlagen. Zo blijft x niet alleen maximaal, maar zullen we ook alle mogelijke k kunnen bereiken.

Regel 15-19

Beginnend aan deze lus, hebben we y minimaal en k ook, maar met $k \geq L$. Nu kunnen we nog oplossingen vinden tot en met $k < L + B$. Dus die gaan we gewoon allemaal bij langs, door elke keer y en k een stap te verhogen. Verder bij elke stap die we doen wordt het geheugen aangepast zoals te zien in Algoritme 1 regel 3.

Het totale algoritme blijkt bij aanpassing van wielgrootte te kunnen lopen in $O(N/\log(\log(N)))$ tijd en met $O(N^{\frac{1}{2}})$ geheugen. Zonder aanpassing van wielgrootte en dus gelijk aan 60, zal de tijdscomplexiteit slechts $O(N)$ zijn. Dat het geheugen zo goed kan, ligt vooral aan onze keuze van B. Als we dit een veelvoud van een wortel nemen, hebben we namelijk ook maar een wortel aan geheugen nodig. We mogen B niet te laag kiezen, omdat dit weer nadelige gevolgen zou kunnen hebben voor de snelheid [1]. We verhogen L nu steeds met B totdat we ons maximum N hebben bereikt, waaronder we priemgetallen wilden tellen.

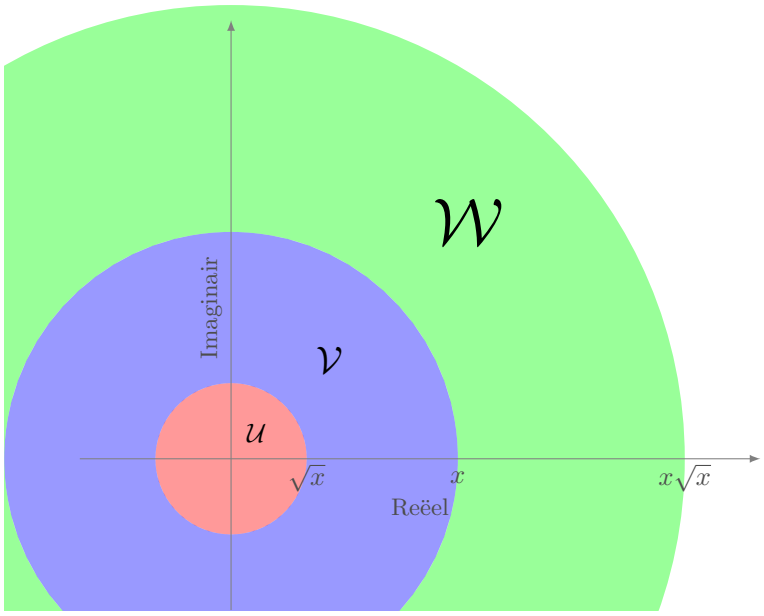
2.3.3 Tellen van priemgetallen 3 (mod 4)

Ook zouden we $4k + 3$ kunnen proberen te zeven, door gebruik te maken van de eerder gevonden priemgetallen van de vorm $4k + 1$. in Figuur 6 zien wij drie verzamelingen. $\mathcal{U}$ is de regio waarmee we het liefst zouden willen zeven. We kunnen met een zeef van Eratosthenes of met een Legendre methode daarmee alle priemgetallen in $\mathcal{V}$ vinden. De vraag is nu, als we ons specifiek gaan richten op het vinden van priemgetallen van de vorm $4k + 3$, zouden we dan misschien nog meer priemgetallen kunnen vinden zonder te veel extra priemgetallen toe te voegen in de zeef.

Stelling 2.15. *Door te zeven met alle priemgetallen in $\mathcal{U}$ kunnen we alle priemgetallen van de vorm $4k + 3$ in $\mathcal{V} \cup \mathcal{W}$ vinden.*

Met andere woorden, neem alle priemgetallen $p \in \mathbb{Z}[i]$ met $|p| \leq \sqrt{x}$. Dan kunnen we door middel van zeven alle priemgetallen $q \in \mathbb{Z}$ met $q \equiv 3 \pmod 4$ en $q \leq x\sqrt{x}$ vinden of tellen.

Bewijs. Neem de verzameling getallen $\mathbb{Z}_{3\text{mod}4}$ dit zijn alle getallen $3 \bmod 4$ in $\mathbb{Z}$. Neem nu een willekeurig getal $q \in \mathbb{Z}_{3\text{mod}4}$ met $q \leq x\sqrt{x}$, en dit is geen priemgetal. Deze willen we nu dus weg zeven. Als deze q een priemgetal deelt die in $\mathcal{U}$ zit, dan zijn we klaar. Nu zijn er twee mogelijkheden die we moeten uitsluiten:



Figuur 6: Een diagram met drie regio's $\mathcal{U}, \mathcal{V}, \mathcal{W}$, deze verzameling hebben geen elementen gemeenschappelijk.

- q deelt een priemgetal p van de vorm $4k+3$ die niet in $\mathcal{U}$ zit. Zo'n priemgetal p heeft als eigenschap dat $|p| > \sqrt{x}$. Dus automatisch $r = \frac{q}{p} < x$, verder zal $r \equiv 1 \pmod 4$. Als r niet een priemgetal is, dan zal één van de delers kleiner of gelijk zijn aan $\sqrt{x}$, en zal q dus weg gezeefd worden. Als r een priemgetal is, bestaat er een priemgetal $\pi \in \mathbb{Z}[i]$ zodat $N(\pi) = r$. Aangezien $\pi | N(\pi)$ en $|\pi| = \sqrt{N(\pi)} < \sqrt{x}$, zal ook in dit geval q worden weg gezeefd.
- q deelt een priemgetal π van de vorm $a+bi$, met beide a, b ongelijk aan 0 en $\pi \notin \mathcal{U}$. Hier maken we gebruik dat $N(\pi)$ de kleinste veelvoud is in $\mathbb{Z}$ van π en dat deze q deelt. Verder omdat $\pi \notin \mathcal{U}$ zal $|\pi| > \sqrt{x}$ en dus $N(\pi) > x$ en $\frac{q}{N(\pi)} < \sqrt{x}$ wat betekent dat q wordt weg gezeefd.

□

We zouden dus kunnen zeven met $\mathcal{U}$, en via een aanpassing aan de Legendre methode, zouden we dit ook kunnen tellen. Toch zijn er in verhouding nog steeds veel priemgetallen die we hiervoor moeten vinden. Namelijk alle priemgetallen in $\mathcal{U}$, om die te vinden is een zeef van Eratosthenes niet al te efficiënt. En kunnen we dus de gedeeltelijke zeef van Atkin gebruiken voor het vinden van de priemgetallen van de vorm $a+bi$ en de rest met een zeef van Eratosthenes.

Een algoritme die de methode in 6 omzet in een Legendre functie wordt weergegeven in algoritme 3.

regel 1-3

Tijdens het initialisatie proces, zullen we eerst alle priemgetallen van de vorm $4k+1$ onder $n^{\frac{2}{3}}$ zeven. Dit gebeurt via algoritme 1. Dan gaan we ook alle priemgetallen onder $\sqrt[3]{n}$ zeven, dit kan met de zeef van Eratosthenes of met de volledige zeef van Atkin. Nu voegen we beide verzamelingen bij elkaar in A en sorteren we de priemgetallen, zodat ze beter tot hun recht komen.

Algoritme 3 Een Legendre implementatie voor het tellen van priemgetallen 3 (mod 4) onder n

```
1: P ← Atkin(n^(2/3))                                   ▷ zie Algoritme 1
2: Q ← Priemgetallen(n^(1/3)) \ P \ {2}                 ▷ Of Alle priemgetallen 3 (mod 4) onder ∛n
3: A = sorteer(P ∪ Q)
4: function LEGENDRE(x, k, r)
5:     if k = −1 then
6:         return ⌊x/4⌋ + ⌊(4−r+(x mod 4))/4⌋
7:     else if A_k ≡ 1 (mod 4) then
8:         return Legendre(x, k − 1, r) − Legendre(⌊x/A_k⌋, k − 1, r)
9:     else
10:        return Legendre(x, k − 1, r) − Legendre(⌊x/A_k⌋, k − 1, 4 − r)
11:    end if
12: end function
13: Legendre(n, |A| − 1, 3)
```

regel 4-12

Nu definiëren we een functie genaamd Legendre, dit is de basis van de Legendre functie in de vorm van een recurrente betrekking. De functie heeft drie parameters. x is hierbij het getal waaronder we willen zeven met de eerste k priemgetallen in A. r geeft aan dat we alleen getallen van de vorm $4k + r$ toelaten om gezeefd te worden. We beginnen met alle getallen die voldoen aan 3 mod 4, dus $r = 3$. Als we delen door een priemgetal die voldoet aan 3 mod 4 dan zal $r = 1$ worden omdat als $p \equiv 3 \pmod 4$ en $xp \equiv 3 \pmod 4$ dan $x \equiv 1 \pmod 4$. Dus moeten we verder met zeven op $r = 1$. Als we delen door een priemgetal die voldoet aan 1 mod 4 verandert er niets aan r. Ook als $r = 1$ ons beginpunt is geldt dat als we delen door een priemgetal van de vorm 3 mod 4, dan zal de volgende recursie $r = 3$ hebben en bij een priemgetal die voldoet aan 1 mod 4 verandert er weer niets aan r. Verder als de priemgetallen op zijn ($k = -1$), dan kunnen we de bijbehorende waarde afdrukken van het aantal $4k + r$ onder x. Dit is gelijk aan $\lfloor \frac{x}{4} \rfloor + \left\lfloor \frac{4-r+(x \bmod 4)}{4} \right\rfloor$.

regel 13

Hier zeggen we dat we dus het aantal priemgetallen willen tellen onder n, met $|A|$ priemgetallen en al de gevonden priemgetallen zijn van de vorm $4k + 3$.

Nu is de functie Legendre nog niet optimaal. Zo zouden we aangezien A gesorteerd is, k slimmer kunnen kiezen voor kleine x, zodat we niet alle priemgetallen bij langs hoeven te gaan. k slim kiezen kan in logaritmische tijd, denk bijvoorbeeld aan bisectie totdat A_k net onder x ligt. Ook kunnen we de Legendre$(x, k - 1, r)$ vervangen door een while-lus om zo te voorkomen dat we teveel functies gestapeld krijgen in het geheugen. Een dergelijk algoritme zal er dan zo uit komen te zien:

Algoritme 4 geeft een verbetering op algoritme 3. Hierbij maken we gebruik van het feit dat x en r niet verandert voor één deel van de recursie in algoritme 3, hiervoor gebruiken we de while lus. Het eerste deel van de recursie Legendre$(x, k, r) =$ Legendre$(x, k - 1, r) + \ldots$ is het deel wat we omzetten in een while-lus, dit kan omdat als we dit uitwerken tot en met Legendre$(x, -1, r)$ het teken niet verandert en omdat overal in de recursie dezelfde waarde zal worden gebruikt van s, deze s wordt normaal gesproken aan het eind van de recursie berekent en daarna ingevoegd in de recursie. Maar omdat we vanaf het begin van de recursie deze waarde van s al nodig hebben, kunnen we die net zo goed voor de while-lus in het begin berekenen.

Ook wordt er in dit algoritme een geschikte k gevonden voor elke x. De functie vindBetereK wordt gedaan in $O(\log(k))$ tijd, omdat we hier het principe van bisectie gebruiken. De functie Legendre heeft een optimale tijdscomplexiteit van $O(n)$, aangezien we door vermenigvuldiging van elk priemgetal met elkaar, alle getallen bij langs zullen gaan op de priemgetallen van de vorm $4k+3$

Algoritme 4 Een verbeterde implementatie van algoritme 4

$P \leftarrow \text{Atkin}(n^{\frac{2}{3}})$ ▷ zie Algoritme 1
$Q \leftarrow \text{Priemgetallen}(n^{\frac{1}{3}}) \setminus P \setminus \{2\}$ ▷ Of Alle priemgetallen 3 (mod 4) onder $\sqrt[3]{n}$
$A = \text{sorteer}(P \cup Q)$

function VINDBETEREK(x, k_0, k_1)
    $m \leftarrow \left\lfloor \frac{k_0+k_1}{2} \right\rfloor$
    if $x > A_{k_1}$ **or** $k_0 = k_1$ **then**
        return k_1
    else if $x > A_m$ **then**
        return vindBetereK($x, m+1, k_1$)
    else if $x < A_m$ **then**
        return vindBetereK($x, k_0, m-1$)
    else
        return m
    end if
end function

function LEGENDRE(x, k, r)
    $s \leftarrow \left\lfloor \frac{x}{4} \right\rfloor + \left\lfloor \frac{4-r+(x \mod 4)}{4} \right\rfloor$
    if $k = -1$ **then**
        return s
    end if
    $k \leftarrow$ vindBetereK($x, 0, k$)
    while $k > -1$ **do**
        if $A_k \equiv 1 \pmod 4$ **then**
            $s \leftarrow s - \text{Legendre}(\left\lfloor \frac{x}{A_k} \right\rfloor, k-1, r)$
        else
            $s \leftarrow s - \text{Legendre}(\left\lfloor \frac{x}{A_k} \right\rfloor, k-1, 4-r)$
        end if
    end while
    return s
end function

Legendre($n, |A|-1, 3$)

boven $\sqrt[3]{n}$ na, maar die zijn in de minderheid. De complexiteit van het geheugen is $O(\frac{n^{\frac{2}{3}}}{\log(n)})$ dit heeft te maken met het aantal priemgetallen die opgeslagen worden.

De vraag nu is, kunnen we deze methode nog verbeteren door gebruik te maken van Meissel's of Mapes' technieken. Het grote probleem wat we hier echter hebben is dat de parameter r de Legendre functie in tweeën splitst, wat het lastig maakt om ook die methodes verder uit te werken onder deze omstandigheden. Ook het vormen van snellere methodes door gebruik te maken van de Totiënt functie is niet eenvoudig vanwege diezelfde r. We kunnen ook in plaats van alle priemgetallen van de vorm $4k+1$ opslaan, elke Legendre recursie gaan opslaan. Dus dan werken we andersom, telkens als we een nieuw priemgetal vinden dan gaan we elke Legendre recursie bijlangs en verder berekenen. Het probleem met zo'n methode is echter dat je dan elke recursie moet opslaan in plaats van alle priemgetallen. Normaal gesproken zonder de r, zouden we maximaal $2\sqrt{n}$ unieke Legendre recursies kunnen krijgen. Met de r verdubbelt dit aantal. Dit heeft alles te maken met de delingen door getallen in het algemeen. Laten we daarom kijken naar de functie $f(x) = \lfloor \frac{n}{x} \rfloor$. Als $1 \leq x \leq \sqrt{n}$ dan zullen er maximaal $\sqrt{n}$ beeldpunten mogelijk zijn, omdat x tot zever beperkt is. En als $x > \sqrt{n}$ dan is $f(x) < \sqrt{n}$, dus kunnen we ook in dit geval maximaal $\sqrt{n}$ beeldpunten vinden. Aangezien r wisselt tussen 3 en 1, verdubbelen we nu het aantal unieke Legendre recursies. Maar de opslagcapaciteit wordt hierdoor wel minder, namelijk $\mathcal{O}(\sqrt{n})$. Alleen moeten al deze recursies nog wel in goede banen worden geleid, waardoor de snelheid er niet aan ten goede komt. Stel namelijk dat je $\sqrt{n}$ aan recursies hebt opgeslagen, dan moet je per nieuwe priemgetal dat je vindt in de zeef van Atkin al deze recursies bij langs. Hoewel we voor grote priemgetallen natuurlijk heel veel gevallen zullen krijgen met $f(x) = 0$, zou je daar een paar omwegen voor kunnen verzinnen. Een reden waarom de de totale berekening hier niet sneller van wordt, is omdat als we al deze priemgetallen onderling vermenigvuldigen, we alle getallen onder n moeten krijgen die niet priemgetal zijn. Dus volgens dat principe zitten we nog steeds op $O(n)$ berekeningen.

Een andere manier om opslag capaciteit te verminderen, is om niet de priemgetallen van de vorm $4k + 1$ te vinden, maar in plaats daarvan alle getallen van deze vorm te nemen. Hoewel dit onze opslag van priemgetallen vermindert tot $\mathcal{O}(n^{\frac{1}{3}} / \log(n))$ voor het vinden van de priemgetallen van vorm $4k + 3$. Worden veel berekeningen herhaalt en zullen we dus uitkomsten van recursies op moeten slaan waardoor we als nog een opslagcapaciteit krijgen van $\mathcal{O}(n^{\frac{1}{2}})$. Bijvoorbeeld als we de getallen 5 en 9 gebruiken om te zeven, dan nemen we automatisch ook $5 \cdot 9 = 45$ mee. Maar ook 45 is van de vorm $4k + 1$, dus in het algoritme zullen we die ook weer tegenkomen. De tijdscomplexiteit zal ook hier niet veranderen.

Ook zouden we ervoor kunnen kiezen om alle getallen $a + bi$ met $\gcd(a, b) = 1$ en $a \equiv b + 1 \pmod 2$, deze kunnen we vinden met behulp van Pythagoreaanse driehoeken. Ook is het zo dat als $N(a+bi)$ geen priemgetal is in $\mathbb{Z}$ dan moet het een deler hebben van de vorm $4k+1$. Wat het aantal herhalingen vermindert. Nu zouden we al deze getallen kunnen vinden met de eigenschap $|a+bi| \leq n^{\frac{1}{3}}$. Maar dat geeft ons een geheugencomplexiteit van $\mathcal{O}(n^{\frac{2}{3}})$. Wat we ook kunnen doen is alle recursies van de Legendre functie opslaan, en nieuwe generen zodra we een nieuwe $a + bi$ gevonden hebben. Hoewel we hier minder getallen bij langs gaan dan dat we alle $4k + 1$ af gaan werken. Worden er als nog heel veel getallen onnodig dubbel berekent.

Dus de beste methode is om als een priemgetal gevonden is, deze direct toe te passen op een Legendre Recursie, zo slaan we uiteindelijk minder priemgetallen op en toch is het even snel. De structuur van het algoritme moet hierdoor wel drastisch veranderen. En alle recursies moeten goed opgeslagen worden. Onze opslag capaciteit verbetert dan naar $O(n^{\frac{1}{2}})$.

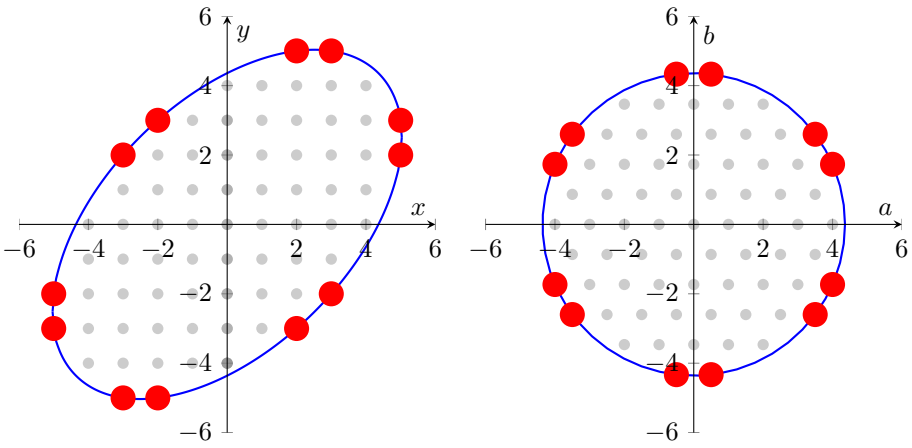
3 Gehelen van Eisenstein

De gehelen van Eisenstein zijn de getallen van de vorm $\mathbb{Z}[\omega]$, met $\omega = \frac{-1+\sqrt{-3}}{2}$. Verder is dit een euclidische ring en heeft eenheden $\pm 1, \pm\omega, \pm\omega^2$. De norm van deze ring is gelijk aan $N(a+b\omega) = a^2 - ab + b^2$ en het is tevens goed om te vermelden dat $(a + b\omega)(a + b\omega^2) = a^2 + ab(-1 - \omega) + ab\omega + b^2\omega^3 = a^2 - ab + b^2 = N(a + b\omega)$ en dus is $N(a + b\omega)$ nooit een priemgetal in $\mathbb{Z}[\omega]$. Ook hier zullen we eerst de priemgetallen identificeren en vervolgens zullen we proberen hier in een priem

telfunctie te maken. Wat we verder gaan doen is het gebruiken van methodes gevonden voor de gehelen van Gauss, om toe te passen op de gehelen van Eisenstein.

We zouden nu naar het volgende product kunnen kijken $(a+b\omega)(a+b\omega^2) = a^2 + ab(-1-\omega) + ab\omega + b^2\omega^3 = a^2 - ab + b^2 = N(a+b\omega)$, dit betekent dat de norm geen priemgetal is in $\mathbb{Z}[\omega]$. Ook kan de norm alleen maar van de vorm $N(x) \equiv 0 \pmod 3$ of $N(x) \equiv 1 \pmod 3$ zijn. Met als gevolg dat alle priemgetallen van de vorm $3k+1$ in $\mathbb{Z}$, geen priemgetallen zijn in $\mathbb{Z}[\omega]$. Om te zien dat de norm inderdaad voldoet aan $N(a+b\omega) \not\equiv 2 \pmod 3$. Merken we eerst op dat als a en b beide veelvouden zijn van 3, dan $N(a+b\omega) \equiv 0 \pmod 3$. Verder als a of b een veelvoud is van 3 en de ander niet, dan $N(a+b\omega) \equiv 1 \pmod 3$. Dan als a en b geen veelvouden zijn van 3 en $a \equiv b \pmod 3$, dan $N(a+b\omega) \equiv 1 \pmod 3$. En als laatst als $a \not\equiv b \pmod 3$ en beide geen veelvoud van 3, dan $N(a+b\omega) \equiv 0 \pmod 3$.

Om een priem telfunctie te maken in deze ring, moeten we kijken naar alle mogelijkheden. Maar ook naar de vorm van het rooster en misschien wel een andere metriek. In figuur 7 is te zien



Figuur 7: In het rood, alle twaalf oplossingen voor $N(x+y\omega) = 19$, $x+y\omega \in \mathbb{Z}[\omega]$ en in het grijs alle oplossingen voor $N(x+y\omega) < 19$. Deze rode oplossingen zijn priemgetallen, want 19 is een priemgetal in $\mathbb{Z}$. Links de oplossingen in het vlak met ω en 1 als basis vectoren. Rechts het complexe vlak met $x+y\omega = a+bi$, dus $a = x - \frac{y}{2}$ en $b = \frac{\sqrt{3}}{2}y$.

dat beide representaties van $a+b\omega$ voordelen en nadelen hebben met tellen van priemgetallen. Zo is in de afbeelding links het voordeel dat elke $x+y\omega$ verdeelt is over een vierkant rooster. Nadeel is dat de norm geen cirkel meer is maar een ellips. In de afbeelding rechts, hebben we als voordeel dat de norm een vorm heeft van een cirkel en daarom meer spiegelingen heeft. Aan de andere kant is het rooster nu een driehoeksrooster, wat een andere manier van rekenen kan betekenen. Ook is het zo voor een getal in $\mathbb{Z}[\omega]$ elke veelvoud makkelijk weer te geven is in het complexe vlak door middel van een gelijkzijdige driehoeksrooster. Deze veelvouden zullen in het complexe vlak ook gelijkzijdige driehoeken vormen. Het lijkt misschien makkelijker om met een vierkant rooster te werken. Vanwege de ellipsen zijn deze gelijkzijdige driehoeken vervormd en het is onduidelijk hoe hierin efficiënt geteld kan worden.

Ook is er nog een voordeel aan het normale complexe vlak. Zo is $\omega = e^{\frac{2}{3}\pi i}$. Alle eenheden in $\mathbb{Z}[\omega]$ zijn $\pm 1, \pm\omega, \pm\omega^2$. Maar als we dit uitwerken zijn dit dus eigenlijk alle getallen $e^{\frac{k}{3}\pi i}$, $k \in \{0,1,2,3,4,5\}$. Dit betekent dat in het complexe vlak bij vermenigvuldiging van $a+b\omega$ met een eenheid, we de bijbehorende vector gewoon om de oorsprong heen draaien. Ook in het Eisenstein vlak, met als basis ω en 1, is er een draaiing, maar deze is minder snel duidelijk omdat we hier te maken hebben met een ellips. Het complexe vlak kunnen we hierdoor in zes gelijke stukken verdelen. Om het zoekgebied te verkleinen, kunnen we dus één van die zes regio's kiezen. Het gebied heeft de vorm van een driehoek en maakt in de oorsprong een hoek van 60°. Als we nu de driehoek 60° doordraaien, krijgen we alle bijbehorende driehoeken. Elke eenheid in $\mathbb{Z}[\omega]$ is

immers een rotatie van 60°. Het enige wat we weten over deze driehoek is dat het begint in de oorsprong, dus hij mag al gedraaid zijn. Toch voor het berekenen is het handiger om uit te gaan van een driehoek die met een zijde rust op de lijn $x + 0\omega$. Of om uit te gaan van een driehoek waarvan de middellijn op de lijn $x + 0\omega$ ligt.

3.1 Zeven in $\mathbb{Z}[\omega]$

Laten we beginnen hier met het zeven van priemgetallen in het vlak en dus niet op de randen. Ook dit kunnen we het beste doen met behulp van een subklasse van de zeef van Atkin. We kunnen eigenlijk een soortgelijke methode gebruiken als dat we in algoritme 1 en 2 hebben gedaan, behalve dat we nu $4x^2 + y^2$ vervangen door $3x^2 + y^2$. En moeten we een aantal waardes daarop aanpassen. Maar waarom nou $3x^2 + y^2$. Dit komt omdat we nu graag in plaats van met $a + b\omega$ willen werken met $u + vi$, met $u = a - \frac{b}{2}$ en $v = \frac{\sqrt{3}}{2}b$. Als we dit inverteren en invullen in $N(a + b\omega)$ krijgen we

$$\begin{aligned} N(a + b\omega) &= N(u + \frac{1}{\sqrt{3}}v + \frac{2}{\sqrt{3}}v\omega) \\ &= (u + \frac{1}{\sqrt{3}}v)^2 - (u + \frac{1}{\sqrt{3}}v)(\frac{2}{\sqrt{3}}v) + (\frac{2}{\sqrt{3}}v)^2 \\ &= u^2 + \frac{2}{\sqrt{3}}uv + \frac{1}{3}v^2 - \frac{2}{\sqrt{3}}uv - \frac{2}{3}v^2 + \frac{4}{3}v^2 \\ &= u^2 + v^2 \end{aligned} \tag{50}$$

Alleen hebben we hier dat u halven kan bevatten en v allemaal veelvouden zijn van $\frac{\sqrt{3}}{2}$. Hoewel we dit niet kunnen gebruiken kunnen we het idee wel gebruiken, kijk nu eens naar $a = x + y$ en $b = 2x$. Dan op soortgelijke wijze vinden we dat $N(a + b\omega) = 3x^2 + y^2$. Verder merken we op dat b hier altijd even zal zijn, maar $a \in \mathbb{Z}$. Als a en b beide even waren dan zal er ook geen priemgetal zijn, in verband met het getal 2. Verder als a en b beide oneven waren, konden we door vermenigvuldiging van ω als nog 1 van beide even maken, immers $(a + b\omega)\omega = -b + (a - b)\omega$. Aangezien $3x^2 + y^2$ afkomstig is van de norm, geld hier ook een soort van multiplicativiteit, alleen moeten we hiervoor wel eerst terug rekenen naar $a + b\omega$. Om alle priemgetallen te vinden van de vorm $3x^2 + y^2$ herschrijven we nu algoritme 1 tot:

Algoritme 5 Algoritme 1 met $3x^2 + y^2$

```
m_{0...B-1} ← {0...0}
for all k met 3x^2 + y^2 = 60k + d,  L ≤ k < L + B,  x, y > 0 do
    m_{k-L} ← 1 − m_{k-L}
end for
for all q ≥ 7, q priem, q^2 < 60(L + B) do
    for all k met q^2 | 60k + d do
        m_{k-L} ← 0
    end for
end for
```

Om de eerste lus te berekenen kunnen we algoritme 2 makkelijk herschrijven door voor x steeds stappen van 10 te nemen en y stappen van 30. De stapgrootte van k kan makkelijk hier uit berekent worden, om zo het algoritme te voltooien. Verder moet opgemerkt worden dat we d nu moeten kiezen uit $d \in \{1, 7, 13, 19, 31, 37, 43, 49\}$.

Voor het zeven van priemgetallen van de vorm $\omega^n a$ met $a, n \in \mathbb{Z}$ kunnen we ook een zeef van Atkin construeren. Alleen kunnen we dan niet de eigenschappen gebruiken van de norm van $\mathbb{Z}[\omega]$ aangezien deze priemgetallen van de vorm $3k + 2$ zijn. Het zal hiervoor dus lastiger zijn om een geschikte functie te vinden. Maar aangezien we toch weinig van deze priemgetallen nodig hebben, zouden we ook een zeef van Eratosthenes kunnen toepassen hier.

3.2 Tellen in $\mathbb{Z}[\omega]$

Om de priemgetallen te tellen zouden we nu de methode van figuur 6 opnieuw kunnen uitvoeren. Deze werkte het best voor de gehelen van Gauss. Alleen in plaats van priemgetallen van de vorm $4k+3$, tellen we in dit geval priemgetallen $6k+5$. Ook worden de cirkels in figuur 6, nu ellipsen zoals in 7.

Laat $\mathcal{U}$ de verzameling van alle priemgetallen p zijn met $N(p) \leq x$. Nu gaan we proberen te zeven met deze priemgetallen. Nu beweren we dus:

Stelling 3.1. *Met alle priemgetallen in $\mathcal{U}$ kunnen we alle priemgetallen van de vorm $p = 6k+5 \in \mathbb{Z}$ zeven waarvoor $N(p) \leq x^3$.*

Bewijs. We nemen hier alleen getallen van de vorm $6k+5$, de enige mogelijke delers van zulke getallen zijn getallen van de vorm $6k+1$ of $6k+5$.

Neem een willekeurig getal $q \leq x^3$ zodat $q \equiv 5 \pmod 6$ en q is geen priemgetal. Stel q deelt een priemgetal in $\mathcal{U}$, dan wordt q inderdaad weg gezeefd. Stel q deelt een priemgetal r niet in $\mathcal{U}$. Dan zijn er twee mogelijkheden

- $r \equiv 5 \pmod 6$, dan $p/r \equiv 1 \pmod 6$ nu bestaat er een getal $u \in \mathbb{Z}[\omega]$ zodat $N(u) = u\bar{u} = p/r$. Dus er zijn minimaal 3 delers van p. Omdat de norm multiplicatief is, betekend dit dat er minstens één deler d moet zijn met $N(d) \leq \sqrt[3]{p} \leq x$. Dus is er een $d \in \mathcal{U}$ die p zeeft.
- $r \equiv 1 \pmod 6$, dan $p/r \equiv 5 \pmod 6$. Aangezien $r \equiv 1 \pmod 6$, moet er een $u \in \mathbb{Z}[\omega]$ zodat $r = u\bar{u}$ en dus zijn ook hier minimaal 3 delers. En dus volgens hetzelfde argument moet er een deler d zijn met $N(d) \leq \sqrt[3]{p} \leq x$, dus p wordt weg gezeefd.

□

Nu kunnen we dit opvatten als een zeef van Eratosthenes. Maar we kunnen hier nu ook een Legendre functie voor maken. We nemen hiervoor alle getallen van de vorm $6k+5$. Dit algoritme lijkt erg op Algoritme 3.

Algoritme 6 Een Legendre implementatie voor het tellen van priemgetallen $5 \pmod 6$ onder n

```
1: P ← Atkin(n^(2/3))                                   ▷ zie Algoritme 5
2: Q ← Priemgetallen(n^(1/3)) \ P \ {2}                 ▷ Of Alle priemgetallen 5 (mod 6) onder ∛n
3: A = sorteer(P ∪ Q)
4: function LEGENDRE(x, k, r)
5:     if k = −1 then
6:         return ⌊x/6⌋ + ⌊(6 − r + (x mod 6))/6⌋
7:     else if A_k ≡ 1 (mod 6) then
8:         return Legendre(x, k − 1, r) − Legendre(⌊x/A_k⌋, k − 1, r)
9:     else
10:        return Legendre(x, k − 1, r) − Legendre(⌊x/A_k⌋, k − 1, 6 − r)
11:    end if
12: end function
13: Legendre(n, |A| − 1, 3)
```

Algoritme 6 geeft een simpel algoritme om de priemgetallen van vorm $6k+5$ te tellen. Dit algoritme kan verbeterd worden door net als in Algoritme 4 een functie *vindBetereK* toe te voegen en om het aantal recursies te beperken, kunnen we van een enkele recursie een while-lus maken.

Ook kunnen we in plaats van alle priemgetallen opslaan, de recursies opslaan en zodra we een priemgetal gevonden hebben, deze updaten. Zo is ons algoritme te berekenen in $O(n)$ tijd en $O(n^{\frac{1}{2}})$ geheugen.

4 Conclusie

In deze scriptie hebben we eerst een korte uitleg gegeven van de methoden van Legendre, Meissel (en Lehmer), en Mapes om het aantal priemgetallen onder een gegeven grens te tellen. Daarna hebben we geprobeerd dezelfde technieken toe te passen op het tellen van priemgetallen in de gehelen van Gauss en in de gehelen van Eisenstein. In de gehelen van Gauss, kunnen we met behulp van de Chebyshev metriek een redelijk goede telmethode vinden gebaseerd op de Legendre methode. Ook hebben we met behulp van de gehelen van Gauss methodes beschreven voor het maken van andere priem telfuncties, zoals het tellen van de gewone priemgetallen van de vorm $4k+1$. In de gehelen van Eisenstein blijkt het minder eenvoudig hoe we gebaseerd op de Legendre methode een priem telfunctie kunnen maken. Toch hebben we wel met behulp van de gehelen van Eisenstein methodes gevonden voor het vinden en tellen van de gewone priemgetallen van de vorm $6k+5$.

Referenties

[1] A. O. L. Atkin, D. J. Bernstein, *Prime Sieves using binary quadratic forms*, `http://cr.yp.to/papers/primesieves.pdf`

[2] M. Deléglise, P. Dusart, X. Roblot, *Counting Primes in Residue Classes*, `http://www.ams.org/journals/mcom/2004-73-247/S0025-5718-04-01649-7/S0025-5718-04-01649-7.pdf`, 2004

[3] Wolfram, *Prime Number Theorem*, `http://mathworld.wolfram.com/PrimeNumberTheorem.html`

[4] Wolfram, *Primitive Pythogorean Triple*, `http://mathworld.wolfram.com/PrimitivePythagoreanTriple.html`

[5] Wolfram, *Pythagorean Triple*, `http://mathworld.wolfram.com/PythagoreanTriple.html`

[6] H. Riesel, *Prime numbers and Computer Methods for Factorization*, Birkhäuser, 1994

[7] B. van Geemen, H. W. Lenstra, F. Oort, J. Top, *Algebraïsche structuren dictaat*