Factorisatie van gehele getallen

Raymond Papenburg (0469998)

2014

Inhoudsopgave

1. Inleiding

> *“”Het probleem van het onderscheiden van priemgetallen en samengestelde getallen en het ontbinden van samengestelde getallen in hun priemfactoren is bekend als een van de belangrijkste en meest nuttige in de rekenkunde. ”*
>
> – C.F. Gauss, *Disquisitiones arithmeticae, art. 329 (1801)*

Priemgetallen vormen al eeuwenlang een populair onderzoeksgebied voor wiskundigen. Ze vormen de bouwstenen van de rekenkunde, en zolang men al geïnteresseerd is in priemgetallen is men ook benieuwd naar de priemfactorisatie van een samengesteld getal. Dankzij de Hoofdstelling van de rekenkunde weten we dat een geheel getal op een unieke manier in factoren te ontbinden is.

In deze scriptie zal ik een aantal methoden om samengestelde getallen te ontbinden in zijn priemfactoren onder de loep nemen. Na een introductie over complexiteit en een beknopte inleiding in de benodigde getaltheorie zullen een aantal ontbindingsalgoritmen beschreven worden.

Dit werk is bedoeld voor eerstejaars studenten als inleiding in de getaltheorie en de toepassingen ervan.

2. Geschiedenis

Zoals genoemd worden wiskundigen al eeuwen bezig gehouden door priemgetallen. Omstreeks 240 v.C. bedacht Eratosthenes een efficiënt algoritme om alle priemgetallen tot een vastgelegde waarde N te genereren: de *zeef van Eratosthenes*. Dit algoritme gaat ervan uit door eerst een lijst te maken van alle getallen van 2 tot en met N. Vervolgens kiezen we het kleinste getal in de lijst en strepen vervolgens alle veelvouden van dit getal in de lijst door (maar niet het getal zelf). Kies nu het volgende getal (dat nog niet is doorgestreept) in de lijst en streep de veelvouden van dit getal door. Dit proces herhalen we tot we alle getallen kleiner of gelijk aan $\sqrt{N}$ hebben gehad. Uit Stelling 8 volgt waarom niet getallen groter dan $\sqrt{N}$ worden gekozen. De getallen die overblijven zijn de priemgetallen.

In de zeventiende eeuw had men al tabellen gemaakt met factorisaties van getallen tot 100.000.[1] In de negentiende eeuw bestonden er al tabellen met getallen tot 10 miljoen. Een aardige anekdote bestaat er van F.N. Cole. In 1903 presenteerde hij een paper genaamd *on the factorization of large numbers*, waarbij zonder een woord te zeggen $2^{67} - 1$ helemaal uitschreef op het bord, vervolgens $193707721 \times 761838257287$ uitrekende, en daarmee aantoonde dat deze twee gelijk aan elkaar waren. Volgens eigen zeggen kostte het hem "three years of sundays".

Hoewel men sinds de opkomst van de computer in staat was om steeds grotere getallen van tientallen decimalen te ontbinden, bleek het factoriseren van zeer grote getallen bestaande uit twee ongeveer even grote factoren problematisch. Zelfs de modernste algoritmes uitgevoerd door de snelste computers weten dit soort getallen, wanneer ze meer dan 150 decimalen bevatten, maar nauwelijks te kraken. Van dit gegeven wordt gebruik gemaakt in allerlei cryptografische toepassingen, zoals de beveiliging van bankpassen.

[1] In 1668 publiceerde J. Pell een werk met alle getallen tot 100.000.

3. Complexiteit

Voordat we ons gaan bezighouden met priemfactorisatie, stellen we ons eerst de vraag hoeveel tijd onze berekeningen ongeveer in beslag zullen nemen als we deze door een computer laten uitvoeren.

Een computer doet zijn bewerkingen met behulp van bits. Een bit is de kleinste eenheid van informatie en kan twee waarden aannemen, welke we kunnen weergegeven met 1 en 0. Om rekenkundige bewerkingen te kunnen doen moeten decimale getallen worden vertaald naar binaire (tweetallige) getallen.

Net zoals een decimaal getal $a_k a_{k-1}...a_0$ wordt gebruikt om het getal

$$a_0 + 10a_1 + ... + 10^k a_k$$

weer te geven, wordt een binair getal $b_k b_{k-1}...b_0$ gebruikt om het getal

$$b_0 + 2b_1 + ... + 2^k b_k$$

weer te geven.

We noemen 2 en 10 de bases van respectievelijk de binaire en de decimale getallen. Om verwarring te voorkomen wordt ook wel de notatie $(a_k a_{k-1}...a_0)_m$ gebruikt voor getallen met basis m. Zo geldt $118_{10} = 1110110_2$. In het vervolg zullen we de subscript 10 bij decimale getallen niet gebruiken.
Een geheel getal n, waarbij $2^{k-1} \leq n < 2^k$, is in het binaire stelsel een getal van k cijfers. Hieruit volgt:

$$k - 1 \leq \log_2 n < k$$

en omdat $\log_p k = \frac{\log k}{\log p}$

$$k - 1 \leq \frac{\log n}{\log 2} < k$$

En dus wordt het aantal bits van n gegeven door $\lfloor \frac{\log n}{\log 2} \rfloor + 1$

3.1 Bitoperaties

Eenvoudige rekenkundige bewerkingen (optellen, aftrekken, vermenigvuldigen, delen) kunnen ook met binaire getallen worden uitgevoerd.

Om de tijd die nodig is om rekenkundige bewerkingen uit te voeren te kunnen afschatten, voeren we het begrip *complexiteit* in. Met de complexiteit T van een berekening bedoelen we het aantal bitoperaties dat nodig is om een berekening uit te voeren. De complexiteit geeft ons een bovengrens voor de looptijd:

Zij $k, \ell \in \mathbb{N}$ met $k > \ell$, dan gelden de volgende lemma's:

Lemma 1.

$$
\begin{aligned}
&T(\textit{Optelling van een } k\textit{-bit getal bij een } \ell\textit{-bit getal}) \leq k \\
&T(\textit{Aftrekking van een } \ell\textit{-bit getal van een } k\textit{-bit getal}) \leq k \\
&T(\textit{Vermenigvuldiging van een } k\textit{-bit getal met een } \ell\textit{-bit getal}) < kl \\
&T(\textit{Deling van een } k\textit{-bit getal door een } \ell\textit{-bit getal}) < kl
\end{aligned}
$$

3.2 Grote-O-notatie

Een in de complexiteitstheorie meer conventionele manier om deze bovengrens te beschrijven is met behulp van de *Grote-O-notatie*

Zij $f, g : \mathbb{N} \to \mathbb{R}$ functies. We noemen

$$f(n) = O(g(n)) \text{ of kortweg } f = O(g)$$

wanneer er een een constante $C \geq 0$ bestaat zodanig dat

$$f(n) \leq Cg(n) \text{ voor alle } n \in \mathbb{N}$$

Voor $k \geq l$ geldt dus

$$
\begin{aligned}
&T(\text{optelling van een } k\text{-bit getal bij een } l\text{-bit getal}) = O(k) \\
&T(\text{Aftrekking van een } \ell\text{-bit getal van een } k\text{-bit getal}) = O(k) \\
&T(\text{vermenigvuldiging van een } k\text{-bit getal met een } l\text{-bit getal}) = O(k^2) \\
&T(\text{Deling van een } k\text{-bit getal door een } \ell\text{-bit getal}) = O(k^2)
\end{aligned}
\tag{3.2.1}
$$

4. Getaltheoretische bagage

4.1 Deelbaarheid

Laat a, b gehele getallen ongelijk aan 0 zijn. We noemen a *deelbaar* door b als er een $d \in \mathbb{Z}$ bestaat zodanig dat $a = bd$. We noteren a *is deelbaar door* b (of b *is een deler van* a) als $b|a$. Wanneer b geen deler is van a schrijven we $b \nmid a$.

Uit deze definitie volgen direct de volgende eigenschappen:

Voor elke $a, b, d \in \mathbb{Z}$

- $d|a,\ d|b \Rightarrow d|(a \pm b)$
- $d|a \Rightarrow db|ab$
- $d|a\,, a|b \Rightarrow d|b$

Een getal $p \in \mathbb{N}$ heet *priem* wanneer $p > 1$ en wanneer p geen andere positieve delers heeft dan 1 en p. Een geheel getal groter dan 1 dat niet priem is heet *samengesteld*.
Dankzij de volgende stelling kunnen we garanderen dat samengestelde getallen kunnen worden ontbonden in priemfactoren.

Stelling 2 (Hoofdstelling van de rekenkunde)**.** *Elk natuurlijk getal kan (op volgorde van factoren na) op unieke wijze worden geschreven als product van priemgetallen.*

Een instructief bewijs hiervan is te vinden in [1] Zo is bijvoorbeeld 14520 te schrijven als $2^3 \times 3 \times 5 \times 11^2$, maar ook als $11 \times 2 \times 3 \times 2 \times 5 \times 2 \times 11$.

Met behulp van de Stelling 2 kunnen we vaststellen dat (a) 14520 is op te schrijven als product van priemgetallen en (b) dat dit product altijd bestaat uit drie 2'en, één 3, één 5 en twee 11'en.

Gevolg 3. *Als p priem is en $p|ab$, dan geldt $p|a$ of $p|b$*
Als $m|a, n|a$ en m en n hebben geen delers groter dan 1 gemeen, dan $mn|a$

De grootste deler die a en b gemeen hebben wordt de *grootste gemene deler* genoemd, en noteren we als $\text{ggd}(a, b)$ of (a, b).

Het tweede gevolg van hierboven kunnen we dus herformuleren:

- $m|a$, $n|a$ en $\text{ggd}(m, n) = 1$, dan $mn|a$

Er bestaat een eenvoudige manier om de ggd van twee getallen te bepalen: het *Euclidisch algoritme*.

Stel $a, b \in \mathbb{N}$ en $a = qb + r$, waarbij $0 \leq r < b$. We zien dat iedere gemeenschappelijke deler van a en b tevens een gemeenschappelijke deler is van r (immers als $d|a$ en $d|b$, dan is ook

$d|(a - qb) = r)$. Hieruit volgt dat $\text{ggd}(a, b) = \text{ggd}(b, r)$. Dit principe wordt herhaald toegepast in het volgende voorbeeld:

$$\begin{aligned}\text{ggd}(3521, 2051) &= \text{ggd}(2051, 3521 - 2051) = \text{ggd}(2051, 1470)\\ &= \text{ggd}(1470, 2051 - 1470) = \text{ggd}(1470, 581)\\ &= \text{ggd}(581, 1470 - 2 \cdot 581) = \text{ggd}(581, 308)\\ &= \text{ggd}(308, 581 - 308) = \text{ggd}(308, 273)\\ &= \text{ggd}(273, 308 - 273) = \text{ggd}(273, 35)\\ &= \text{ggd}(35, 273 - 7 \cdot 35) = \text{ggd}(35, 28)\\ &= \text{ggd}(28, 35 - 28) = \text{ggd}(28, 7) = \text{ggd}(7, 0)\\ &= 7\end{aligned}$$

Het Euclidisch algoritme geeft altijd de ggd in een eindig aantal stappen. De resten worden immers per stap strikt kleiner, dat wil zeggen $r_{i+1} < r_i$ en moeten (omdat $r \geq 0$) uiteindelijk 0 opleveren.

Stelling 4. *T(vind ggd(a, b) m.b.v. het Euclidisch algoritme)=*$O(\log^3 a)$

Bewijs. Elke stap van het algoritme is niets anders dan een deling met rest van twee gehele getallen kleiner dan a. Zoals we in het vorige hoofdstuk hebben gezien zijn er per stap $O(\log^2 a)$ bitoperaties nodig. Dan is het alleen nog de vraag hoeveel stappen er hoogstens nodig zijn.

Claim. $r_{i+2} \leq \frac{r_i}{2}$, voor $i = 1, 2, ...$

Om dit in te zien beschouwen we twee gevallen:

Stel allereerst dat $r_{i+1} \leq \frac{r_i}{2}$. Dan geldt $r_{i+2} < r_{i+1} \leq \frac{r_i}{2}$ en is aan de ongelijkheid voldaan.

Neem nu aan dat $r_{i+1} > \frac{r_i}{2}$.

Het Euclidisch algoritme schrijft voor dat $r_i = q_{i+2}r_{i+1} + r_{i+2}$. We beschouwen nu q_{i+2}.
Als $q_{i+2} \geq 2$ dan

$$r_i \geq 2r_{i+1} + r_{i+2} > 2r_{i+1}$$

waaruit volgt dat $r_{i+1} < \frac{r_i}{2}$, wat in tegenspraak is met de aanname.
Dus $q_{i+2} = 1$ en

$$\begin{aligned}r_{i+2} &= r_i - r_{i+1}\\ &< r_i - \frac{r_i}{2} = \frac{r_i}{2}\end{aligned}$$

Onze claim is juist, en hierdoor zien we een vlot verval van de resten. In hoogstens twee stappen neemt het aantal bits met 1 af. Daardoor zijn er maximaal $\left\lfloor \frac{\log a}{\log 2} \right\rfloor$ delingen nodig, wat neerkomt op $O(\log a)$ delingen. Dit maakt de totale tijd die nodig is om $\text{ggd}(a, b)$ te vinden $O(\log a) \cdot O(\log^2 a) = O(\log^3 a)$ □

4.2 Congruentie

Zij m een natuurlijk getal groter dan 1. We noemen $a, b \in \mathbb{Z}$ *congruent modulo* m wanneer $m|(a-b)$. Dit komt erop neer dat twee getallen congruent zijn precies dan als ze bij deling door

m dezelfde rest hebben.

we noteren dit als $a \equiv b \pmod{m}$

Eigenschappen:

Stel dat $a \equiv b \pmod{m}$ en $c \equiv d \pmod{m}$, dan

$$a \pm c \equiv b \pm d \pmod{m}$$
$$ac \equiv bd \pmod{m}$$

Congruentie modulo m is een equivalentierelatie. Dat wil zeggen dat

$$a \equiv a \pmod{m} \text{ reflexiviteit}$$
$$a \equiv b \pmod{m} \Leftrightarrow b \equiv a \pmod{m} \text{ symmetrie}$$
$$a \equiv b \pmod{m},\ b \equiv c \pmod{m} \Rightarrow a \equiv c \pmod{m} \text{ transitiviteit}$$

Elke equivalentieklasse (restklasse) bevat precies één getal tussen 0 en $m-1$. De verzameling restklassen wordt genoteerd als $\mathbb{Z}/m\mathbb{Z}$.
We besluiten dit hoofdstuk met een belangrijke stelling van Pierre de Fermat. Hij liet zelf na het bewijs te geven. [1] Euler publiceerde in 1706 een bewijs.

Stelling 5 (Fermat)**.** *Stel $a \in \mathbb{Z}$ en $p \in \mathbb{N}$ priem. Als $p \nmid a$, dan*

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

Bewijs. Laat $a \in \mathbb{Z}$ en $p \in \mathbb{N}$ priem en $p \nmid a$. We maken eerst een lijst van de eerste $p-1$ veelvouden van a:

$$a, 2a, 3a, ..., (p-1)a$$

We tonen aan dat deze $p-1$ getallen modulo p allemaal verschillend zijn. Neem het tegendeel aan: stel dat $ra \equiv sa \pmod{p}$. Dan geldt $p|(r-s)a$. Omdat $p \nmid a$ moet gelden $p|(r-s)$. Omdat $r, s < p$ moet r gelijk zijn aan s. Hieruit volgt dat $a, 2a, 3a, ..., (p-1)a \pmod{p}$ niets anders is dan een herschikking van $1, 2, 3, ..., p-1$. Hieruit volgt dat

$$a \cdot 2a \cdot 3a \cdot ... \cdot (p-1)a = 1 \cdot 2 \cdot 3 \cdot ... \cdot (p-1) \pmod{p}$$

Anders geschreven:

$$a^{p-1}(p-1)! = (p-1)! \pmod{p}$$

Wanneer we beide leden door $(p-1)!$ delen vinden we het gevraagde. □

[1] Fermat zei daarbij "I would send you the demonstration, if I did not fear its being too long" [2, p. 79]

5. Priemtesten

Voordat we ons gaan richten op de factorisatie van gehele getallen is het aardig om ons eerst af te vragen of het te factoriseren getal niet priem is. Hiervoor zijn in de loop der tijd allerlei priemtesten ontwikkeld.

Een voor de hand liggende (en naïeve) manier om de primaliteit van een geheel getal n te testen is gewoon door te kijken of er een $d \in \left\{2, 3, ... \lfloor\sqrt{n}\rfloor\right\}$ bestaat waarvoor geldt dat $d|n$. Wanneer dit het geval is, weten we dat n samengesteld is. Wanneer n echter uit veel (>20 cijfers) bestaat is deze methode echter onbruikbaar, omdat daar te veel tijd voor nodig is, namelijk $O(\sqrt{n})$ (hierover in het volgende hoofdstuk meer).

Praktischere methoden maken gebruik van de omkering van de Kleine stelling van Fermat.

Stelling 6 (Fermat)**.** *Zij $a \in \mathbb{Z}$ en $n \in \mathbb{N}$ zodanig dat $n \nmid a$. Dan*

$$a^{n-1} \not\equiv 1 \pmod{n} \Rightarrow n \text{ is samengesteld.}$$

Zouden we ons bijvoorbeeld afvragen of 119 priem is, dan kiezen we $n = 119$ en bijvoorbeeld $a = 2$. Allereerst controleren we $\gcd(a, n)$. Wanneer dit groter is dan 1 dan hebben we natuurlijk meteen een deler van n te pakken en weten we dat n niet priem is. Echter $\gcd(2, 119) = 1$ en we vervolgen met het bepalen van $2^{118} \pmod{119}$. Dit lijkt een heel karwei, maar we kunnen heel wat rekenwerk besparen door 118 te noteren in bits en vervolgens herhaaldelijk te kwadrateren modulo n. We maken daarbij gebruik van de volgende identiteit [12]:

$$a^{2^k} \pmod{N} = \begin{cases} (a^{2^{k-1}})^2 & \pmod{N}, \text{ als } k \text{ is even} \\ a(a^{2^{k-1}})^2 & \pmod{N}, \text{ als } k \text{ is oneven} \end{cases}$$

$$\begin{aligned} 118 &= 1110110_2 \\ 2^1 &\equiv 2 \pmod{119} \\ 2^{11} &\equiv 2 \cdot 2^2 \equiv 8 \pmod{119} \\ 2^{111} &\equiv 2 \cdot 8^2 \equiv 9 \pmod{119} \\ 2^{1110} &\equiv 9^2 \equiv 81 \pmod{119} \\ 2^{11101} &\equiv 2 \cdot 81^2 \equiv 32 \pmod{119} \\ 2^{111011} &\equiv 2 \cdot 32^2 \equiv 25 \pmod{119} \\ 2^{1110110} &\equiv 25^2 \equiv 30 \pmod{119} \end{aligned}$$

We zien dat $2^{118} \not\equiv 1 \pmod{119}$ wat inhoudt dat 119 niet priem is ($119 = 7 \cdot 17$). De tijd die deze methode gebruikt om de samengesteldheid van n aan te tonen is een stuk vlotter in vergelijking met de naïeve manier.

Lemma 7. $T(\textit{bepaal } a^{n-1} \pmod{n}) = O(\log(n-1)\log^2(n))$

Immers, er zijn zoveel stappen als het aantal bits dat $n - 1$ bevat, en elke stap bestaat uit het vermenigvuldigen van twee getallen modulo n.

6. Getallen ontbinden

6.1 Klassieke ontbindingsmethoden

Voor de opkomst van computers was men nog niet zo ver met het ontbinden van grote getallen. Nu was men er tegen het begin van de twintigste eeuw wel in geslaagd om van getallen tot 10.000.000 de priemfactorisatie ontcijferen, maar

Laten we in het vervolg aannemen dat $N \in \mathbb{N}$ samengesteld is. Laten we een poging doen het te ontbinden.

6.1.1 Een naïeve methode

De eerste manier om dit te doen ligt, net als bij het primaliseren, voor de hand: we proberen een deler van N te vinden door achtereenvolgens alle gehele getallen tot $\lfloor\sqrt{N}\rfloor$ te proberen.

Stelling 8. *Laat $N \in \mathbb{N}$ bestaan uit niet-triviale factoren $a, b \in \mathbb{N}$, zodanig dat $N = ab$ en neem aan dat $a \leq b$. Dan geldt $a \leq \sqrt{n}$.*

Bewijs. Stel dat $a > \sqrt{N}$, dan geldt ook $b > \sqrt{N}$ (immers $b \geq a$). Hieruit volgt dat $ab > N$ en dit is in tegenspraak met onze aanname. □

Het vervelende aan deze methode is dat zij nogal tijdrovend is. Het aantal stappen om een factor van N te vinden kan immers oplopen tot $\sqrt{N}$. Stel dat N uit k decimalen bestaat, dan geldt dus

$$k \approx \log_{10} N = \frac{\log N}{\log 10}$$

ofwel

$$\log N = k \log 10$$

$$O(\sqrt{N}) = O(e^{\frac{\log N}{2}}) = O(e^{ck}) \text{ waarbij } c = \tfrac{\log 10}{2}$$

Het aantal stappen neemt dus exponentieel toe met de invoerlengte. Zoals gezegd is dit veel te langzaam voor het ontbinden van grotere getallen. We kunnen het algoritme verbeteren om het wat sneller te maken. We gaan ervan uit dat ons te factoriseren getal oneven is. Wanneer het even is hebben we met 2 al een factor gevonden. We hoeven de even getallen dus niet meer proberen als deler en halveren daarmee de bewerkingstijd (merk op dat deze nog wel exponentieel is). Om dit idee door te voeren kunnen we ook alleen priemgetallen proberen als deler. We moeten dan echter wel weten wat de priemgetallen beneden $\sqrt{N}$ zijn. Hiervoor moeten we of vooraf een lijst maken van priemgetallen en vervolgens het algoritmes die getallen laten proberen, of een priemgenerator (zoals bijvoorbeeld de *zeef van Eratosthenes*) inbouwen bij het algoritme. In beide gevallen kost dit extra rekentijd die het algoritme niet veel sneller maakt. Een goed alternatief zou zijn om eerst 2 en 3 als delers te proberen en daarna de getallen van de vorm $6k - 1$ en $6k + 1$ ($k \in \mathbb{N}$). Op die manier worden toch alle priemgetallen (alsmede sommige samengestelde getallen) gebruikt. De rekentijd van dit algoritme is ongeveer eenderde van die van de oorspronkelijke.

6.1.2 De methode van Fermat

Het algoritme leent zich nog prima voor de kleine factoren, maar wanneer de factoren dichter bij $\sqrt{N}$ blijken te liggen, kunnen we beter gebruikmaken van de *methode van Fermat*. Pierre de Fermat maakte hierbij gebruik van het gegeven dat ieder oneven samengesteld getal kan worden geschreven als verschil van twee kwadraten.

Laat $N \in \mathbb{N}$ oneven en samengesteld. Dan kunnen we $N = ab$ schrijven, waarbij $a, b \in \mathbb{N}$ oneven en $b \geq a > 0$. Tevens kunnen we N als verschil van twee kwadraten schrijven: $N = x^2 - y^2$, waarbij $x, y \in \mathbb{N}$ en $x > y \geq 0$. We kunnen de volgende stelling poneren:

Stelling 9. *Er bestaat een bijectie tussen de factorisaties van N in de vorm ab en representaties van N in de vorm $x^2 - y^2$*

Bewijs. Definieer $x = \frac{a+b}{2}$ en $y = \frac{a-b}{2}$.
Dan geldt

$$N = ab = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2 = x^2 - y^2$$

Laat nu $a = x + y$ en $b = x - y$, dan

$$N = x^2 - y^2 = (x + y)(x - y) = ab$$

□

Om een factor van N te vinden moeten we een x vinden zodanig dat $x^2 - N = \square$ ($waarbij we \square$ definiëren als kwadraat van een geheel getal
$Laten we het algoritme beginnen met het kleinste gehele getal groter dan \sqrt{N}$, ofwel $x_0 = \lfloor \sqrt{N} \rfloor + 1$. De eerste stap van het algoritme berekent $x_0^2 - N$. Indien dit een kwadraat is hebben we een factor van N gevonden. Als het geen kwadraat is definiëren we $x_1 = x_0 + 1$, en berekenen vervolgens $x_1^2 - N$ en bepalen hiervan of het een kwadraat is. Zo vervolgens we het algoritme totdat we bij de $p - 1$-de stap een kwadraat vinden.

$$\begin{aligned} x_0^2 - N &\neq \square \\ x_1^2 - N &\neq \square \\ &\vdots \\ x_{p-1}^2 - N &= \square =: q^2 \end{aligned}$$

We hebben met $x_{p-1}^2 - N$ een kwadraat van geheel getal q gevonden en dus geldt

$$\begin{aligned} N &= x_{p-1}^2 - q^2 \\ &= (\lfloor \sqrt{N} \rfloor + p)^2 - q^2 \\ &= (\lfloor \sqrt{N} \rfloor + p + q)(\lfloor \sqrt{N} \rfloor + p - q) \end{aligned}$$

Neem bijvoorbeeld $N = 33127$, dan $\lfloor \sqrt{N} + 1 \rfloor = 183$. We voeren het algoritme uit en vinden achtereenvolgens

$$\begin{aligned} 183^2 - 33127 &= 362 \neq \square \\ 184^2 - 33127 &= 729 = 27^2 \end{aligned} \tag{6.1.1}$$

We vinden dus al na twee stappen dat $N = 184^2 - 27^2 = (184 + 27)(184 - 27) = 211 \cdot 157$. Met de eerder beschreven verbeterde naïeve probeermethode hadden we dit resultaat pas na ruim 50 stappen bereikt (en bij de oorspronkelijke pas na 157 stappen). We kunnen dus stellen dat deze factorisatiemethode erg snel is wanneer a en b relatief dicht bij elkaar liggen. Echter, wanneer a en b verder uit elkaar liggen, is de verwerkingstijd in het slechtste geval nog langer dan bij de vorige methode.
Voor zowel de naïeve methode als de Fermat-factorisatie is een algoritme geschreven in `Python`[1], waarbij tijd van de ontbindingen is gemeten. In tabel 6.1 nemen p en q (beide steeds priem) steeds met een factor 10 toe. Dat betekent dat bij de naïeve steeds 10 keer zoveel getallen als deler geprobeerd moeten worden. De rekentijd bevestigt hier ook de expontentiële rekentijd. Omdat p en q wel steeds relatief bij elkaar blijven liggen, blijft de rekentijd voor Fermat laag. In tabel 6.2 zien we het omgekeerde: daar is p vast en laag, en q neemt steeds met een factor 10 toe. De naïeve methode vindt steeds de kleine factor in ongeveer dezelfde tijd, terwijl de Fermat-methodemoeite heeft met het feit het verschil tussen p en q steeds een factor 10 groter wordt.

Tabel 6.1: Berekentijd (in sec.) voor de naïeve methode en de Fermat-methode waarbij p en q allebei steeds met een factor groter worden

p	q	Trial	Fermat
61	43	0.00019-0.00025	0.00007-0.00009
503	547	0.00045-0.00050	0.00010-0.00015
5303	5417	0.0035-0.0040	0.00012-0.00018
50363	53381	0.040-0.050	0.00025-0.00028
525167	496399	0.49-0.52	0.0011-0.0014
5021683	5002747	5.3-5.7	0.00027-0.00030
48205429	63563359	59-61	3.2-3.4
460966463	535314859	470-485	7.1-7.6

Tabel 6.2: Berekentijd (in sec.) voor de naïeve methode en de Fermat-methode waarbij p vastligt en q steeds met een factor groter wordt

p	q	Trial	Fermat
61	43	0.00019-0.00025	0.00007-0.00009
61	547	0.00022-0.00026	0.00060-0.00070
61	5417	0.00022-0.00027	0.010-0.011
61	53381	0.00022-0.00027	0.11-0.13
61	496399	0.00023-0.00027	1.2-1.3
61	5002747	0.00023-0.00028	13-14
61	63563359	0.00023-0.00029	168-170

6.1.3 De methode van Euler

Tot slot beschouwen we nog kort de methode van Euler. Leonhard Euler (1707-1783) beschreef deze methode in 1745 in een brief aan Christian Goldbach [9]. Deze methode is alleen toepasbaar op gehele getallen N die *op twee manieren* in de vorm $N = a^2 + kb^2$ geschreven kunnen worden

[1] De codes hiervan staan in het Appendix

met dezelfde k. Hierbij wordt gebruik gemaakt van de volgende identiteit die zegt dat het product van twee gehele getallen van de vorm $a^2 + kb^2$ opnieuw een geheel getal van deze vorm is.

$$(a^2 + kb^2)(c^2 + kd^2) = \begin{cases} (ac + kbd)^2 + k(bc - ad)^2 \\ (ac - kbd)^2 + k(bc + ad)^2 \end{cases} \tag{6.1.2}$$

Dit kunnen we als volgt inzien:

$$\begin{aligned}(a^2 + kb^2)(c^2 + kd^2) &= (ac)^2 + k(ad)^2 + k(bc)^2 + k^2(bd)^2 \\ &= (ac)^2 \pm 2k(abcd) + k(ad)^2 + k(bc)^2 \mp 2k(abcd) + k^2(bd)^2 \\ &= (ac)^2 \pm 2k(abcd)k^2(bd)^2 + k(bc)^2 \mp 2k(abcd) + k(ad)^2 \\ &= (ac \pm kbd)^2 + k(bc \mp ad)^2\end{aligned}$$

Euler bewees het omgekeerde: als een getal op twee manieren geschreven kan worden in de vorm $a^2 + kb^2$, dus $N = p^2 + kq^2$ en $N = r^2 + ks^2$ waarbij $\text{ggd}(qs, N) = 1$, dan kan N worden geschreven als product van twee getallen van deze vorm.
Om de factoren te vinden schrijven we $N = p^2 + kq^2$ en $N = r^2 + ks^2$. Respectievelijk met s^2 en q^2 vermenigvuldigen levert

$$\begin{aligned}p^2s^2 + kq^2s^2 &= Ns^2 \\ r^2q^2 + kq^2s^2 &= Nq^2\end{aligned}$$

Dus

$$p^2s^2 - r^2q^2 = N(s^2 - q^2)$$

waaruit volgt dat

$$p^2s^2 \equiv r^2q^2 \pmod{N}$$

Lemma 10 (Legendre)**.** *Zij $x, y \in \mathbb{Z}$, zodanig dat $x^2 \equiv y^2 \pmod{m}$, $0, x < y < m$ en $x + y \neq N$, dan zijn ggd$(x + y, m)$ en ggd$(x - y, m)$ niet-triviale factoren van m.*

Er geldt immers dat $m|(x+y)(x-y)$, maar tevens $m \nmid (x+y)$ en $m \nmid (x-y)$. Passen we Lemma 10 toe op $p^2s^2 \equiv r^2q^2 \pmod{N}$ dan zien we dat $\text{ggd}(ps + rq, N)$ en $\text{ggd}(ps - rq, N)$ factoren van N zijn.
We zullen niet ingaan op de vraag hoe we getallen van de vorm $a^2 + kb^2$ kunnen vinden. Wel volgt hier tot slot nog een voorbeeld. Neem $N = 377$. $377 = 4^2 + 19^2 = 11^2 + 16^2$. Dus $p = 19$,$q = 4$,$r = 11$,$s = 16$. Nu volgt

$$19^2 \cdot 16^2 \equiv 4^2 \cdot 11^2 \pmod{377}$$

Als factoren van N vinden we dan $\text{ggd}(19 \cdot 16 + 11 \cdot 4, 377) = \text{ggd}(348, 377) = 29$ en $\text{ggd}(19 \cdot 1611 \cdot 4, 377) = \text{ggd}(260, 377) = 13$. En inderdaad $377 = 13 \cdot 29$.

6.2 Moderne factorisatiemethoden

Het mooie aan de behandelde methoden is dat, mits N samengesteld is, ze gegarandeerd een deler opleveren. Bij de moderne methoden, die sinds de opkomst van computers wordt gebruikt is dit niet het geval. Deze methoden worden dan wel *probabilistisch* genoemd.

6.2.1 De Pollard-ρ-methode

Een aanmerkelijk snellere methode is de $\rho - methode$, bedacht door J.M. Pollard in 1975. Pollard was ook de uitvinder van de $p-1$-methode en de eerste getaltheoriezeef.
Bekijk de afbeelding van $f : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$ gegeven door een $f(x) = x^2 + a$ met hele coëfficiënten en $a \notin \{-2, 0\}$. We definiëren

$$\begin{aligned} x_0 &= \text{ een willekeurig getal } r \in \mathbb{Z}/N\mathbb{Z} \\ x_i &= f(x_{i-1}) \pmod N, \text{ i=0,1,2,...} \end{aligned}$$

De rij $x_0, x_1, x_2, ..., x_n$ die zo ontstaat gedraagt zich als een rij willekeurig gegenereerde getallen. Omdat het aantal elementen van $\mathbb{Z}/N\mathbb{Z}$ eindig is (het zijn er immers niet meer dan N), zullen we uiteindelijk in een cykel terechtkomen. Dat betekent dat er $m, n \in \mathbb{N}$ (met $n > m$) zijn zodanig dat $x_m = x_n$, en dus ook $x_{m+1} = x_{n+1}$, $x_{m+2} = x_{n+2}$, ... Het kleinste getal $k = n - m$ noemen we de periode. Met een beetje fantasie zien we de aanloop naar de cykel (dus $x_0, x_1, ...x_{k-1}$) als de staart en de cykel als de kop van ρ.

Vervolgens bekijken we het verschil tussen de x_i's. Het is zaak om een paar (i, j) te vinden waarvoor geldt dat $x_i \equiv x_j \pmod s$ en $x_i \not\equiv x_j \pmod N$, (waarbij $s \in \mathbb{N}$ een niet-triviale deler is van N). Dan geldt immers dat $s|(x_i - x_j)$ en $s|N$. Hieruit volgt dat $s|\text{ggd}(x_i - x_j, N)$. omdat $s \geq 2$ geldt $\text{ggd}(x_i - x_j, N) \geq 2$. Daarnaast geldt ook $N \nmid (x_i - x_j)$ en daarom ook $N \nmid \text{ggd}(x_i - x_j, N)$. Hieruit volgt dat $\text{ggd}(x_i - x_j, N)$ een niet-triviale factor van N moet zijn. Hoe vinden we zulke i, j? We kunnen niet alle mogelijke paren proberen, aangezien dit er $O(p)$ zijn. Pollard stelt voor om de paren (x_i, x_{2i}) te proberen, volgens *Floyd's cycle algorithm.*

We gaan proberen $N = 8051$ te ontbinden. Neem $f(x) = x^2 + 1$, $x_0 = 5$.

$$\begin{aligned} x_1 &= 5^2 + 1 \equiv 26 \pmod{8051} \\ x_2 &= 26^2 + 1 \equiv 677 \pmod{8051} \\ x_3 &= 677^2 + 1 \equiv 7474 \pmod{8051} \\ x_4 &= 7474^2 + 1 \equiv 2839 \pmod{8051} \\ x_5 &= 2839^2 + 1 \equiv 871 \pmod{8051} \\ x_6 &= 871^2 + 1 \equiv 1848 \pmod{8051} \end{aligned}$$

tegelijkertijd bepalen we $\text{ggd}(x_{2i} - x_i, 8051)$ voor $i = 1, 2, 3$. we vinden:

$$\begin{aligned} \text{ggd}(x_2 - x_1, 8051) &= \text{ggd}(651, 8051) = 1 \\ \text{ggd}(x_4 - x_2, 8051) &= \text{ggd}(2162, 8051) = 1 \\ \text{ggd}(x_6 - x_3, 8051) &= \text{ggd}(5626, 8051) = 97 \end{aligned}$$

En inderdaad, $8051 = 97 \cdot 83$.

6.2.2 Het algoritme van Kraitchik

We kijkens nogmaals naar de methode van Fermat. We hebben in paragraaf 6.1.2 gezien dat deze alleen goed werkt wanneer het verschil tussen de factoren relatief klein is.

Stelling 11. *Zij $n \in \mathbb{N}$ oneven, samengesteld en niet een priemmacht. Dan bestaan er $x, y \in \mathbb{Z}$ zodanig dat $x^2 = y^2 \pmod n$ en $x \not\equiv \pm y \pmod n$*

Bewijs. Neem aan dat n samengesteld is, zodat we kunnen schrijven $n = ab$, waarbij $a, b \in \mathbb{N}$, $a, b > 1$ en $\text{ggd}(a, b) = 1$.
Laat nu $y \in \mathbb{Z}$ zodanig dat $\text{ggd}(y, n) = 1$ (en dus ook $\text{ggd}(y, a) = \text{ggd}(y, b) = 1$) en $x \in \mathbb{Z}$ zodanig dat $x \equiv y \pmod{a}$ en $x \equiv -y \pmod{b}$. Volgens de Chinese reststelling bestaat er zo'n x. Dan geldt $a|(x - y)$ en $b|(x + y)$. Omdat $\text{ggd}(a, b) = 1$, geldt $ab|(x - y)(x + y)$ ofwel $n|(x^2 - y^2)$, wat hetzelfde betekent als $x^2 \equiv y^2 \pmod{n}$.

Stel dat $x \equiv y \pmod{n}$, dan geldt ook $x \equiv y \pmod{b}$, en dus $-y \equiv y \pmod{b}$, waaruit volgt dat $b|2y$. Omdat $\text{ggd}(y, b) = 1$ moet $b = 2$, maar ab is oneven en dus b ook: tegenspraak.
Stel nu dat $x \equiv -y \pmod{n}$, dan geldt ook $x \equiv -y \pmod{a}$, en dus $y \equiv -y \pmod{b}$, waaruit volgt dat $a|2y$. Omdat $\text{ggd}(y, a) = 1$ moet $a = 2$, maar ab is oneven en dus a ook: tegenspraak. □

Het is nu zaak om een een congruentie van de vorm $x^2 \equiv y^2 \pmod{n}$ te vinden. Hierbij kunnen we de methode van Fermat als uitgangspunt gebruiken. De Belg Maurice Kraitchik deed dit in de jaren '20 van de negentiende eeuw, en zijn aanpassingen op Fermats methode hebben de basis gevormd voor de meeste ontbindingsalgoritmes [6].
In plaats van het vinden van $u, v \in \mathbb{Z}$ waarvoor $u^2 - v^2 = n$ is het voldoende om u en v te vinden waarvoor $u^2 - v^2 = kn$ (waarbij $k \in \mathbb{Z}$), oftewel $u^2 \equiv v^2 \pmod{n}$.

We bekijken $z_k := x_k^2 - n$, waarbij $x_k = \lfloor \sqrt{n} \rfloor + 1 + k$ en $k = 1, 2, 3....$ We bekijken vervolgens de priemontbinding van z_k. Wanneer we $i_1, i_2, ..., i_r \in \{1, ..., k\}$ vinden waarvoor $z_{i_1} z_{i_2} ... z_{i_r}$ een kwadraat is hebben we onze congruentie $u^2 \equiv v^2 \pmod{n}$ gevonden, waarbij

$$u = \prod_{m=1}^{r} x_{i_m} \text{ en } v = \sqrt{\prod_{m=1}^{r} z_{i_m}}$$

Bekijk bijvoorbeeld $n = 1717$. dan

$$x_k = \lfloor \sqrt{1717} \rfloor + 1 + k = 42 + k \text{ en } z_k = (42 + k)^2 - 1717 = k^2 + 84k + 47$$

k	x_k	z_k	Factorisatie
0	42	47	47
1	43	132	$2^2 \cdot 3 \cdot 11$
2	44	219	$3 \cdot 73$
3	45	308	$2^2 \cdot 7 \cdot 11$
4	46	399	$3 \cdot 7 \cdot 19$
5	47	492	$2^2 \cdot 3 \cdot 41$
6	48	587	587
7	49	684	$2^2 \cdot 3^2 \cdot 19$

We zien dat

$$z_1 z_3 z_4 z_7 = 2^6 \cdot 3^4 \cdot 7^2 \cdot 11^2 \cdot 19^2 = (2^3 \cdot 3^2 \cdot 7 \cdot 11 \cdot 19)^2 = 105336^2$$
$$x_1 x_3 x_4 x_7 = 43 \cdot 45 \cdot 46 \cdot 49 = 4361490$$

waaruit volgt dat

$$4361490^2 \equiv 105336^2 \pmod{1717}$$

Uit stelling 10 volgt dat $\text{ggd}(4361490 - 105336, 1717) = 17$ en $\text{ggd}(4361490 + 105336, 1717) = 101$ factoren zijn van 1717, en dit klopt. We hebben de factoren van n nu na 7 stappen gevonden,

terwijl we met de Fermat-methode pas bij $k = 17$ een kwadraat zouden hebben gevonden.

We hebben nu geluk gehad dat we de niet-triviale factoren van n hebben kunnen vinden. De kans bestaat echter dat we $x, y \in \mathbb{Z}$ vinden waarvoor $x^2 \equiv y^2 \pmod{n}$ maar tevens $x \equiv \pm y \pmod{n}$. In dat geval kunnen we volgens Stelling 11 geen niet-triviale factoren van n vinden.

Maar hoe vinden we zo effectief mogelijk een deelverzameling van $\{z_1, z_2, ..., z_r\}$ waarvan het product een kwadraat is?

Een positief geheel getal n heet b-glad wanneer alle priemfactoren kleiner of gelijk zijn aan p_b. $B := \{p_1, p_2, , .., p_b\}$ noemen we de *factorbasis*.

Zij $n \in \mathbb{N}$ een b-glad getal. We kunnen (dankzij de Hoofdstelling van de rekenkunde) n schrijven als $n = p_1^{k_1} p_2^{k_2} \dots p_b^{k_b}$, waarbij p_i het i-de priemgetal is. Definieer nu $\upsilon(n) = (k_1, k_2, \dots, k_r)$. Het vorige voorbeeld beschouwd zien we bijvoorbeeld dat $\upsilon(132) = (2, 1, 0, 0, 1, 0, 0, 0)$ (waarbij $b = 8$ is genomen). Omdat we enkel geïnteresseerd zijn in kwadraten en omdat n een kwadraat is dan en slechts dan als $k_i \equiv 0 \pmod{2}$ voor alle i, bekijken we $\upsilon(n)$ modulo 2. Dat betekent dat $\upsilon(132) \equiv (0, 1, 0, 0, 1, 0, 0, 0) \pmod{2}$.

Om een kwadraat te vinden gaan we de z_k bekijken die b-glad zijn. Stel dat we er k hebben gevonden. We noteren ze als $z_{b_1}, z_{b_2}, \dots, z_{b_k}$. We kunnen de rijvectoren $\upsilon(z_{b_1}), \upsilon(z_{b_2}), \dots, \upsilon(z_{b_k})$ (mod 2) onder elkaar zetten om zo een $k \times b$ exponentmatrix te vormen. Het is nu zaak om minstens twee $\upsilon(z_{b_i})$ te vinden die lineair afhankelijk zijn. In $\mathbb{Z}/2\mathbb{Z}$ betekent dit niets anders dan dat de som van de rijvectoren de nulvector oplevert.
Immers, voor afhankelijkheid van verschillende vectoren $v_1, v_2, \dots, v_m$ in een vectorruimte V moeten er scalairen $a_1, a_2, \dots, a_m \in V$ bestaan die niet allemaal 0 zijn, zodanig dat

$$a_1 v_1 + a_2 v_2 + \dots + a_m v_m = \mathbf{0}$$

In $\mathbb{Z}/2\mathbb{Z}$ zijn 0 en 1 de enige scalairen, en zodoende vereist afhankelijkheid minstens twee vectoren in $\{v_1, v_2, ..., v_m\}$ waarvan de som de nulvector is.

We bekijken nogmaals $n = 1717$ en kiezen $b = 8$, ofwel $B = \{2, 3, 5, 7, 11, 13, 17, 19\}$.
We zien dat we vier k's hebben gevonden waarvoor z_k b-glad is. De exponentvectoren hiervan zijn

$$\begin{aligned}
\upsilon(z_1) &= (2, 1, 0, 0, 1, 0, 0, 0) \equiv (0, 1, 0, 0, 1, 0, 0, 0) \pmod{2} \\
\upsilon(z_3) &= (2, 0, 0, 1, 1, 0, 0, 0) \equiv (0, 0, 0, 1, 1, 0, 0, 0) \pmod{2} \\
\upsilon(z_4) &= (0, 1, 0, 1, 0, 0, 0, 1) \equiv (0, 1, 0, 1, 0, 0, 0, 1) \pmod{2} \\
\upsilon(z_7) &= (2, 2, 0, 0, 0, 0, 0, 1) \equiv (0, 0, 0, 0, 0, 0, 0, 1) \pmod{2}
\end{aligned}$$

De exponentmatrix hiervan is dus

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \pmod{2}$$

We zien direct dat $\upsilon(z_1) + \upsilon(z_3) + \upsilon(z_4) + \upsilon(z_7) \equiv \mathbf{0} \pmod{2}$, en zo hebben we een kwadraat gevonden. Zoals we hebben gezien leidt een kwadraat niet automatisch tot een ontbinding. Wanneer $x \equiv \pm y \pmod{1717}$ het geval was geweest hadden we een andere lineaire combinatie

van vectoren moeten proberen. Met de matrix als hierboven had dit geen afhankelijkheidsrelatie opgeleverd.

Om meer exponentvectoren te verkrijgen moeten we meer k proberen om zo meer b-gladde z_k te krijgen. Echter neemt z_k toe naarmate k groter wordt, en daarmee wordt de kans dat we z_k vinden met priemfactoren van hoogstens de b-de priem kleiner. Om toch meer z_k te vinden laten we negatieve k toe. Op die manier kunnen we een grens M kiezen waarbij we voor k de getallen in het interval $[-M, M]$ kiezen.

We bekijken nu de gehele getallen n waarvan p_b de grootste priemmacht in de factorisatie is. We voegen -1 toe aan de factorbasis: $B := \{-1, p_1, p_2, , .., p_b\}$. We kunnen -1 immers zien als priemgetal in $\mathbb{Z}$.

We bekijken opnieuw $n = 1717$, maar nu laten we $M = 4$ en $b = 13$.

k	x_k	z_k	Factorisatie
-4	38	-273	$(-1) \cdot 3 \cdot 7 \cdot 13$
-3	39	-196	$(-1) \cdot 2^2 \cdot 7^2$
-2	40	-117	$(-1) \cdot 3^2 \cdot 13$
-1	41	-36	$(-1) \cdot 2^2 \cdot 3^2$
1	43	132	$2^2 \cdot 3 \cdot 11$
3	45	308	$2^2 \cdot 7 \cdot 11$

De exponentmatrix is dan $\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \pmod 2$

We zien dat $v(z_{-4}) + v(z_{-2}) + v(z_1) + v(z_3) \equiv \mathbf{0} \pmod 2$, en zodoende is $z_{-4}z_{-2}z_1z_3$ een kwadraat.

$$(38 \cdot 40 \cdot 43 \cdot 45)^2 \equiv (-273) \cdot (-117) \cdot 132 \cdot 308 \pmod{1717}$$

Echter geldt dat zowel het linker- als het rechterlid 1696 (mod 1717) zijn, en dat we daarom geen niet-triviale oplossing zullen vinden. We zullen daarom een andere lineaire combinatie moeten vinden, maar dat lukt niet met deze matrix. We zullen dus andere keuzes voor b en M moeten maken. We willen b eigenlijk klein kiezen omdat getallen met dezelfde grotere priemfactoren schaarser zijn. Echter moeten we dan wel M groot kiezen. Kiezen we b groot, dan laten we meer factorisaties toe en kunnen we M kleiner kiezen.

6.2.3 De kwadratische zeef

Het grote probleem van deze methode is dat voor elke z_k bepaald moet worden of het b-glad is, wat een nogal tijdrovende klus is. We zullen middels een zeefproces de z_k's die niet b-glad zijn eruit filteren. We bekijken nu $k \in [-M, M]$. We vragen ons dus af welke voor welke k een priem $p \in \{2, 3, 5, 7..., p_b\}$ een deler is van $z_k := x_k^2 - n$. Hiertoe moeten we de congruentievergelijking $x_k^2 \equiv n \pmod p$ oplossen. Indien er voor een bepaalde p geen oplossingen zijn dan

komt p niet voor in de factorisatie van elke z_k. Indien de vergelijking wel oplossingen heeft, dan zijn dit er twee: $t_1 \pmod p$ en $t_2 := p - t_1 \pmod p$ (als $p = 2$ dan geldt $t_1 = t_2$=1). De k waarvoor de vergelijking een oplossing heeft noemen we k_t. Nu geldt dat ook $k_t + ap$, waarbij $a \in \mathbb{Z}$, een oplossing is. Op deze manier kunnen we voor elke p een deelverzameling oplossingen $D_p \subset \{x_{-M}, x_{1-M}, ..., x_M\}$ genereren.
Omdat z_k ook hogere machten van p kan bevatten doen we hiervoor. Indien $x_k^2 \equiv n \pmod p$ geen oplossingen heeft dan heeft $x_k^2 \equiv n \pmod{p^r}$ ook geen oplossingen. Voor elke p waarvoor $x_k^2 \equiv n \pmod p$ oplossingen heeft bepalen we $x_k^2 \equiv n \pmod{p^r}$, en genereren we de verzameling oplossingen. Indien deze verzameling niet-leeg is voeren we de r met 1 op totdat de verzameling oplossingen wel leeg is. Vervolgens bekijken we de machten van de volgende $p \leq p_b$.
We hebben nu voor elke macht van p een deelverzameling van oplossingen. Vervolgens delen we elke z_{k_t} door de p_i's die horen bij de deelverzamelingen waarin x_{k_t} zit. Alle z_k die aan het eind van dit proces ± 1 zijn geworden bevatten dus alleen priemfactoren $\leq p_b$ en zijn dus b-glad.

We bekijken nogmaals $n = 1717$. We zoeken b-gladde getallen voor $B = \{-1, 2, 3, 5, 7, 11\}$ en $x_k \in [25, 59]$.

$x^2 \equiv 1717 \pmod 2$
$x^2 \equiv 1 \pmod 2$
$x \equiv 1 \pmod 2$

$x^2 \equiv 1717 \pmod 4$
$x^2 \equiv 1 \pmod 4$
$x \equiv 1, 3 \pmod 4$

$x^2 \equiv 1717 \pmod 8$
$x^2 \equiv 1 \pmod 8$
geen oplossingen

$x^2 \equiv 1717 \pmod 3$
$x^2 \equiv 1 \pmod 3$
$x \equiv 1, 2 \pmod 3$

$x^2 \equiv 1717 \pmod 9$
$x^2 \equiv 7 \pmod 9$
$x \equiv 4, 5 \pmod 9$

$x^2 \equiv 1717 \pmod{27}$
$x^2 \equiv 16 \pmod{27}$
$x \equiv 4, 23 \pmod{27}$

$x^2 \equiv 1717 \pmod 5$
$x^2 \equiv 2 \pmod 5$
geen oplossingen

$x^2 \equiv 1717 \pmod 7$
$x^2 \equiv 2 \pmod 7$
$x \equiv 3, 4 \pmod 7$

$x^2 \equiv 1717 \pmod{49}$
$x^2 \equiv 2 \pmod{49}$
$x \equiv 10, 39 \pmod{49}$

$D_2 = D_4 = \{25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59\}$
$D_3 = \{25, 26, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44, 46, 47, 49, 50, 52, 53, 55, 56, 58, 59\}$
$D_9 = \{31, 32, 40, 41, 49, 50, 58, 59\}$
$D_{27} = \{31, 50, 58\}$
$D_7 = \{25, 31, 32, 38, 39, 45, 46, 52, 53, 59\}$
$D_{49} = \{39, 59\}$

Voor het volledige zeefproces, zie Appendix B. We zien dat voor $x_k \in \{31, 32, 39, 41, 43, 45, 59\}$ geldt dat z_k b-glad is. Een zeefmethode als deze doet het ontbindingsalgoritme veel tijdwinst boeken. Het aantal stappen om b-gladde getallen te vinden binnen het interval $[-M, M]$ is ongeveer $\log \log b$ per kandidaat b-glad getal. De tijd om n te factoriseren is $O(\exp(c\sqrt{\log n \log \log n}))$ Hierbij is c een goed gekozen constante.

6.2.4 Andere methoden

Over het algemeen gebruikt men bij het ontbinden van getallen een mengvorm aan methoden. Algoritmen zoals het naïeve delingsalgoritme, Fermat en Euler worden ingezet om kleinere factoren tevoorschijn te halen. Middelgrote factoren worden gezocht met behulp van Pollard-ρ en Lenstra's ECM-methode. Voor de grotere factoren worden met name de kwadratische zeef en de getallenlichamenzeef gebruikt.

De getaltheoriezeef

Aan het einde van de jaren tachtig van de vorige eeuw ontwierp John Pollard de getallenlichamenzeef.Het idee hierbij is min of meer gelijk aan die van de kwadratische zeef. Echter wordt er niet langer gewerkt met louter kwadratische veeltermen, en ook worden er naast $\mathbb{Z}$ en $\mathbb{Z}/n\mathbb{Z}$ over andere getallenlichamen gewerkt, wat dit algoritme veel gecompliceerder maakt. Dit algoritme bleek vooral voor zeer grote getallen (> 100 cijfers) nog veel sneller te werken dan de kwadratische zeef. De complexiteit van dit algoritme is $O(\exp(c \log n^{\frac{1}{3}} (\log \log n)^{\frac{2}{3}}))$, waar c een goed gekozen constante is.

Lenstra's *Elliptic Curve Method (ECM)*

Hendrik Lenstra ontwikkelde een algoritme dat gebruik maakt van elliptische krommen. In tegenstelling tot de kwadratische zeef en de getallenlichamenzeef hangt de snelheid van dit algoritme af van de grootte van de kleinste gevonden factor p. De complexiteit van dit algoritme voor het ontbinden van n is $O(\exp(c\sqrt{\log p \log \log p}) \cdot \log n)$, waar $c \approx 2$ [11]. Voor een goede introductie op het gebied van elliptische krommen is het raadzaam om [8] te lezen. In [5] staat een aardige beschrijving van het ECM-algoritme.

A. Programmeercodes in Python

A.1 Trial

```
import math
import time

def isqrt(N):
  x = N
  y = (x + N // x) // 2
  while y < x:
    x = y
    y = (x + N // x) // 2
  return x

def trial(N):

    starting_time = time.clock()
    for a in range(2,isqrt(N)+1):
        if N % a == 0:
            print(N, '=', a, '*', N//a)
            break
    else:
        print(N,'is priem')
    elapsed_time = time.clock() - starting_time


    print('Benodigde tijd is', elapsed_time,'seconden')
```

A.2 Fermat

```
import math
import time

def isqrt(N):
  x = N
  y = (x + N // x) // 2
  while y < x:
    x = y
    y = (x + N // x) // 2
  return x

def fermat(N):
    starting_time = time.clock()

    if N == isqrt(N)*isqrt(N):
        print(N,'=', isqrt(N),'*',isqrt(N))
    else:
        a = 1
        b = isqrt(N)
        m = ((a + b)**2 - N)**0.5
        while m != int(m):
            a = a + 1
            m = ((a + b)**2 - N)**0.5
        elapsed_time = time.clock() - starting_time
        p = a + b + int(m)
        q = a + b - int(m)
        print(N, '=', p, '*', q)
        print('Benodigde tijd is', elapsed_time)
```

B. Kwadratische zeef

Tabel B.1: Kwadratische zeef voor $n = 1717$ en $B = \{-1, 2, 3, 5, 7, 11\}$

x_k	x^2	z_k	÷2	÷2	÷3	÷3	÷3	÷7	÷7	÷11
25	625	-1092	-546	-273	-91			-13		
26	676	-1041			-347					
27	729	-988	-494	-247						
28	784	-933			-311					
29	841	-876	-438	-219	-73					
30	900	-817								
31	961	-756	-378	-189	-63	-21	-7	**-1**		
32	1024	-693			-231	-77		-11		**-1**
33	1089	-628	-314	-157						
34	1156	-561			-187					17
35	1225	-492	-246	-123	-41					
37	1369	-348	-174	-87	-29					
38	1444	-273			-91			-13		
39	1521	-196	-98	-49				-7	**-1**	
40	1600	-117			-39	-13				
41	1681	-36	-18	-9	-3	**-1**				
43	1849	132	66	33	11					**1**
44	1936	219			73					
45	2025	308	154	77				11		**1**
46	2116	399			133			19		
47	2209	492	246	123	164					
49	2401	684	342	171	57	19				
50	2500	783			261	87	29			
51	2601	884	442	221						
52	2704	987			329			47		
53	2809	1092	546	273	91			13		
54	2916	1199								109
55	3025	1308	654	327	109					
56	3136	1419			473					43
57	3249	1532	766	383						
58	3364	1647			549	183	61			
59	3481	1764	882	441	147	49		7	**1**	

Bibliografie

[1] Beukers, F.: *Getaltheorie voor beginners*. Epsilon Uitgaven, 1993.

[2] Burton, D.M.: *Elementary number theory*. Allyn and Bacon, 1980.

[3] Everest, G., Ward, T.: *An introduction to number theory*. Springer-Verlag, 2005.

[4] Garrett, S.L.: *On the quadratic sieve* The University of North Carolina, Greensboro, 2008.

[5] Koblitz, N.: *A course in number theory and cryptography*, Springer-Verlag, 1987.

[6] Pomerance,C.: *A tale of two sieves*. Notices of the AMS, December 1996, [pp.1473-1485]

[7] Riesel, H.: *Prime numbers and computer methods for factorization*, Birkhaüser, 1994.

[8] Silverman, J.H., Tate, J.: *Rational points on ellipcitc curves*, Springer, 1994

[9] Smith, F.J.: *A brief history of factorization techniques*. University of Washington, 2006.

[10] Yan, S.Y.: *Number theory for computing*. Springer-Verlag, 2002.

[11] Yan, S.Y.: *Cryptanalytic attacks on RSA*. Springer-Verlag, 2008.

[12] Wikipedia: *Exponentiation by squaring*. `http://en.wikipedia.org/wiki/Exponentiation_by_squaring`