Joris van der Hoeven
CNRS, Department of Mathematics
Simon Fraser University, Burnaby, B.C.
vdhoeven@lix.polytechnique.fr

De oplossing

Getallen vermenigvuldigen in $O(n \log n)$ stappen

Onlangs werd door David Harvey en Joris van der Hoeven bewezen dat twee getallen van n cijfers kunnen worden vermenigvuldigd in $O(n \log n)$ stappen [12]. In 1971 werd het bestaan van zo'n methode door Schönhage en Strassen geuit als vermoeden [21]. Ook verwachtten zij dat het sneller niet gaat, iets dat we nog steeds niet weten. Joris van der Hoeven vat in dit artikel eerst een lange geschiedenis kort samen en geeft dan inzicht in het nieuwe bewijs.

Wat is de beste manier om twee gehele getallen met elkaar te vermenigvuldigen? Deze ogenschijnlijk eenvoudige vraag is misschien wel het oudste onopgeloste wiskundeprobleem. Zo is het zeker tien maal ouder dan het vermoeden van Fermat, dat inmiddels de stelling van Wiles heet.

Wél moet hier natuurlijk duidelijk worden gemaakt wat we met 'beste' bedoelen. Dit werd eigenlijk pas mogelijk na de uitvinding van de Turingmachine: uitgerust met een precies rekenmodel kunnen we spreken over het aantal stappen $M(n)$ dat nodig is om twee getallen van n cijfers met elkaar te vermenigvuldigen. De beste methode is dan diegene waarvoor $M(n)$ zo langzaam mogelijk stijgt als n groter wordt.

Neem bijvoorbeeld de methode van de lagere school. Dan vermenigvuldigen we elk cijfer van het eerste getal met elk cijfer van het tweede getal terwijl we de resultaten op de juiste posities bij elkaar optellen. Dat geeft $M(n) = O(n^2)$: er bestaat een constante $C > 0$ met $M(n) \leq Cn^2$ voor iedere n.

Om onze hoofdvraag onafhankelijk te maken van de daadwerkelijke Turingmachine waarop we onze berekening doen, zijn we enkel geïnteresseerd in rekentijd op een constante factor na. Een honderd maal snellere methode is dus geen wezenlijke verbetering, maar een $\log \log \log n$ maal sneller algoritme is een enorme stap voorwaarts.

Foto: Nina Ramaswamy

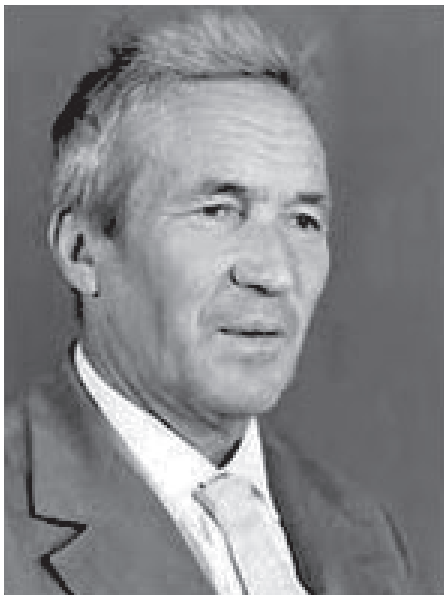
Figuur 1 Gedenksteen in de Pieterskerk in Leiden, met een reconstructie van de oorspronkelijke tekst van de grafsteen van Ludolph van Ceulen (1540–1610), die vijfentwintig jaar van zijn leven besteedde aan het uitrekenen van 35 decimalen van π.

Waarschijnlijk formuleerde Kolmogorov de vraag voor het eerst op deze manier, tijdens een beroemd seminarium in Moskou. In zijn enthousiasme lanceerde hij ook meteen het vermoeden dat $n^2 = O(M(n))$. Want, zo redeneerde hij, als er een betere methode bestond dan die van school, dan zou die sinds zesduizend jaar toch wel zijn uitgevonden.

De nauwkeurige formulering van een probleem maakt het zoeken naar oplossingen makkelijker. Maar het ontbreken daarvan sluit het zoeken naar oplossingen nou ook weer niet uit. De Babyloniërs berekenden $\sqrt{2}$ al tot op zestien decimalen achter de komma. Wellicht vinden we ooit een kleitablet waarop hogesnelheidsvermenigvuldigingsmethodes staan uitgelegd. Liefhebbers van het getal π, zoals de bij ons bekende Ludolph van Ceulen, zouden daar vast wel wat aan kunnen hebben gehad (zie Figuur 1).

Karatsuba

Kolmogorovs vermoeden hield niet lang stand. Na drie weken werd hij verrast door een jonge student met een vermenigvuldigingsmethode in $O(n^{\log 3/\log 2})$ stappen. Meteen schreef hij het nieuwe algoritme op en publiceerde het onder de naam van de bedenker, Karatsuba, samen met een ander artikel dat er niets mee te maken had [15, 16].



Andrej Kolmogorov

Anatoly Karatsuba

Laten we Karatsuba's idee uitleggen met een voorbeeld:

$$u = 220629012020,$$
$$v = 314159265358.$$

Eerst splitsen we beide getallen in tweeën:

$$u = \overbrace{220629}^{a} \times \overbrace{1000000}^{x} + \overbrace{012020}^{b},$$
$$v = \underbrace{314159}_{c} \times 1000000 + \underbrace{265358}_{d}.$$

Daarna doen we twee optellingen en drie vermenigvuldigingen:

$$a + b = 232649,$$
$$c + d = 579517,$$
$$ac = 069312586011,$$
$$bd = 003189603160,$$
$$(a+b)(c+d) = 134824050533.$$

Nu merken we op dat het slechts twee keer aftrekken vergt om

$$ad + bc = (a+b)(c+d) - ac - bd$$
$$= 62321861362$$

te bepalen. Tot slot geldt

$$uv = (ax+b)(cx+d)$$
$$= acx^2 + (ad+bc)x + bd,$$

zodat we klaar zijn na een laatste optelsom

$$\begin{array}{rr} 069312586011000000000000 & acx^2 \\ 62321861362000000 & (ad+bc)x \\ 003189603160 & bd \\ \hline 069312648332864551603160 & uv \end{array}$$

Al met al hebben we een vermenigvuldiging van twaalf cijfers teruggevoerd tot drie vermenigvuldigingen van zes cijfers en een stel optel- en aftreksommen.

In het algemeen kost het vermenigvuldigen van twee maal zo lange getallen ongeveer drie maal zoveel tijd. Dit kan goed worden samengevat in een recurrente ongelijkheid:

$$M(2n) \leq 3M(n) + O(n).$$

Voor een zekere constante C en $n = 2^r$ hebben we dus

$$\begin{aligned} M(n) &\leq 3M(n/2) + Cn \\ &\leq 9M(n/4) + \left(1 + \tfrac{3}{2}\right)Cn \\ &\leq 27M(n/8) + \left(1 + \tfrac{3}{2} + \tfrac{9}{4}\right)Cn \\ &\vdots \\ &\leq 3^r M(1) + O\left(\left(\tfrac{3}{2}\right)^r n\right) \\ &= O(3^r). \end{aligned}$$

Nu kunnen we links altijd extra nullen aan een getal toevoegen zodat het aantal cijfers een tweemacht wordt. Uiteindelijk bewijzen we zo dat

$$M(n) = O(n^{\log 3/\log 2}).$$

Van getallen naar veeltermen

Een van de ingrediënten van Karatsuba's methode is het opsplitsen van getallen in twee delen. Dit maakte het mogelijk om u en v te zien als veeltermen $ax+b$ en $cx+d$ van graad < 2. In feite kwam dat simpelweg neer op het werken in een talstelsel met grondtal $x = 1000000$ in plaats van 10.

Dit idee om het rekenen met gehele getallen te vervangen door het rekenen met veeltermen werd in the negentiende eeuw reeds geopperd door Kronecker. Zo kunnen we een getal van n cijfers opsplitsen in l brokken van $p := \lceil n/l \rceil$ cijfers en daar weer een veelterm van maken.

Door het aantal brokken l groter dan twee te kiezen, werden er in de zestiger jaren een reeks snellere varianten van Karatsuba's methode ontwikkeld [17, 20, 22]; zie Tabel 1 voor een historisch overzicht.

Zonder al te ver op details in te gaan, is elk van deze varianten gebaseerd op een veralgemening van Karatsuba's rekentruc. In dit geval wordt de vermenigvuldiging van twee veeltermen van graad $< l$ gereduceerd tot $2l-1$ vermenigvuldigingen van coëfficiënten, waarna

$$M(n) = O(n^{\log(2l-1)/\log(l-1)}).$$

-4000	Onbekend	$O(n^2)$
1962	Karatsuba	$O(n^{\log 3/\log 2})$
1963	Toom	$O(n 2^{5\sqrt{\frac{\log n}{\log 2}}})$
1966	Schönhage	$O(n 2^{\sqrt{\frac{2\log n}{\log 2}}} (\log n)^{\frac{3}{2}})$
1969	Knuth	$O(n 2^{\sqrt{\frac{2\log n}{\log 2}}} \log n)$
1971	Pollard	$O(n \log n \log\log n \cdots)$
1971	Schönhage-Strassen	$O(n \log n \log\log n)$
2007	Fürer	$O(n \log n\, 2^{O(\log^* n)})$
2014	Harvey-vdH-Lecerf	$O(n \log n\, 8^{\log^* n})$
2017	Harvey	$O(n \log n\, 6^{\log^* n})$
2017	Harvey-vdH	$O(n \log n (4\sqrt{2})^{\log^* n})$
2018	Harvey-vdH	$O(n \log n\, 4^{\log^* n})$
2019	Harvey-vdH	$O(n \log n)$

Tabel 1 Betere en betere bovengrenzen voor $M(n)$ in de loop van de jaren... In de tabel is de functie $\log^\times$ gedefinieerd door $\log^* x = \min\{n \in \mathbb{N} : (\log \circ \overset{n\times}{\cdots} \circ \log)(x) \leqslant 1\}$.

Van veeltermen naar kringtermen

Vanaf nu gaan we het vooral hebben over het vermenigvuldigen van veeltermen en is het handig om te werken met coëfficiënten in een algemene ring R. De precieze keus van R laten we voorlopig in het midden.

Voor wat komen gaat, is het ook handig om met zogenaamde 'kringtermen' in plaats van veeltermen te werken. Een *kringterm* van lengte l is een element van $R[x]/(x^l-1)$.

De relatie $x^l=1$ treedt pas in actie zodra de graad van een veelterm over de l gaat. Het uitrekenen van een product van twee veeltermen in $R[x]$ van graad $< l/2$ kunnen we dus net zo goed in $R[x]/(x^l-1)$ doen.

Het voordeel van kringtermen is dat we een aantal nieuwe rekentrucs rijker worden. Veronderstel bijvoorbeeld dat $l=2$ en dat 2 omkeerbaar is in R. Dan is het mogelijk om Karatsuba's methode verder te verbeteren: modulo x^2-1 geldt

$$\begin{aligned}&(a_1x+a_0)(b_1x+b_0)\\&\quad=\frac{\hat a_0\hat b_0-\hat a_1\hat b_1}{2}x+\frac{\hat a_0\hat b_0+\hat a_1\hat b_1}{2},\end{aligned}$$

waar

$$\begin{aligned}\hat a_0&=a_0+a_1,\quad \hat b_0=b_0+b_1,\\ \hat a_1&=a_0-a_1,\quad \hat b_1=b_0-b_1.\end{aligned}$$

In $R[x]/(x^2-1)$ vergt het uitrekenen van een product dus slechts twee vermenigvuldigingen in R. Ook zijn er een aantal optellingen, aftrekkingen en delingen door twee nodig.

Bovenstaande relaties zijn het gevolg van

$$(x^2-1)=(x-1)(x+1) \tag{1}$$

en toepassing van de Chinese reststelling op deze ontbinding:

$$\begin{aligned}&R[x]/(x^2-1)\cong R[x]/(x-1)\times R[x]/(x+1),\\&\qquad\overline{a_1x+a_0}\mapsto(\overline{a_0+a_1},\overline{a_0-a_1}).\end{aligned}$$

Dit maakt het mogelijk willekeurige berekeningen in $R[x]/(x^2-1)$ te vervangen door berekeningen in $R[x]/(x-1)\times R[x]/(x+1)$ $\cong R^2$.

FFT-vermenigvuldigen

De ontbinding (1) geldt voor elke ring R. In sommige ringen kan ook x^l-1 op soortgelijke wijze ontbonden worden voor $l>2$. Dat is het geval zodra R een element ω bezit met $\sum_{k=0}^{l-1}(\omega^j)^k=0$. Zo'n element ω heet een *principiële eenheidswortel* van orde l en leidt tot de ontbinding

$$x^l-1=\prod_{0\leqslant k<l}(x-\omega^k).$$

Is l ook nog eens omkeerbaar in R, dan kan bovendien de Chinese reststelling worden toegepast op deze ontbinding:

$$R[x]/(x^l-1)\cong\prod_{0\leqslant k<l}R[x]/(x-\omega^k).$$

Van links naar rechts heet dit isomorfisme de *discrete fouriertransformatie* of DFT. Voor alle $A\in R[x]$ en $0\leq k<l$ hebben we $x=\omega^k$ en $P(x)=P(\omega^k)$ modulo $x-\omega^k$, zodat

$$\mathrm{DFT}(\bar A)=(\overline{A(1)},\overline{A(\omega)},\ldots,\overline{A(\omega^{l-1})}).$$

De algebra $R[x]/(x-\omega^k)$ is op zijn beurt weer isomorf met R voor elke k, zodat

$$R[x]/(x^l-1)\cong R^l.$$

Nu is rekenen aan de rechterkant goedkoop: één vermenigvuldiging in R^l komt bijvoorbeeld neer op l vermenigvuldigingen in R. Als we zowel de DFT en de inverse DFT^{-1} snel uit kunnen rekenen, dan hebben we dus een goede methode om kringtermen $A,B\in R[x]/(x^l-1)$ te vermenigvuldigen:

$$AB=\mathrm{DFT}^{-1}(\mathrm{DFT}(A)\,\mathrm{DFT}(B)).$$

Dit heet *FFT-vermenigvuldigen*.

De snelle fouriertransformatie

Maar hoe rekenen we zo'n DFT snel uit? Het geval $l=2$ bestudeerden we al eerder. Meer in het algemeen leidt de ontbinding

$$x^{2l}-1=(x^l-1)(x^l+1)$$

voor even lengtes $2l$ tot een isomorfisme

$$R[x]/(x^{2l}-1)\cong R[x]/(x^l-1)\times R[x]/(x^l+1).$$

Als $A,B\in R[x]$ veeltermen van graad $<l$ zijn, dan zendt dit isomorfisme $A+Bx^l$ naar $(A+B,A-B)$, waar we voor het gemak de modulostrepen hebben weggelaten. Het uitrekenen van $A\pm B$ komt neer op l maal optellen en aftrekken in R. Voor het isomorfisme in de andere richting komen daar $2l$ delingen door twee bij.

Als ω een principiële eenheidswortel van orde $2l$ is, dan hebben we bovendien het volgende isomorfisme:

$$\begin{aligned}&R[x]/(x^l+1)\cong R[x]/(y^l-1),\\&\qquad\sum_{0\leqslant k<l}a_kx^k\mapsto\sum_{0\leqslant k<l}(a_k\omega^k)\,y^k.\end{aligned}$$

Het uitrekenen van dit isomorfisme komt neer op l vermenigvuldigingen met machten van ω. Hier is het belangrijk om op te merken dat een vermenigvuldiging met een macht van ω soms goedkoper is dan een willekeurige vermenigvuldiging in R.

Al met al laat dit zien hoe een DFT van lengte $2l$ kan worden gereduceerd tot twee DFTs van lengte l. Schrijven we $F(l)$ voor de tijd die het kost om een DFT van lengte l uit te rekenen, $T_\pm$ voor de tijd van een optelling of aftrekking in R, en T_ω voor de tijd van een vermenigvuldiging met een macht van ω, dan hebben we

$$F(2l)\leq 2F(l)+2lT_\pm+lT_\omega.$$

Passen we deze formule recursief toe in het geval dat $l=2^{\lg l}$ een tweemacht is, dan vinden we

$$\begin{aligned}F(l)&\leq 2F(l/2)+l\big(T_\pm+\tfrac12T_\omega\big)\\&\leq 4F(l/4)+2l\big(T_\pm+\tfrac12T_\omega\big)\\&\;\;\vdots\\&\leq (l/2)F(2)+(\lg l-1)\,l\big(T_\pm+\tfrac12T_\omega\big)\\&\leq l\lg l\big(T_\pm+\tfrac12T_\omega\big).\end{aligned} \tag{2}$$

Voor de inverse transformatie komen daar ook nog l delingen door l bij.

Intermezzo

Voordat we verder gaan, is het een mooi moment voor wat opmerkingen. De FFT brak door in 1965, na de publicatie van Cooley en Tukey's artikel [3]. Maar een soortgelijke methode werd eerder beschreven in ongepubliceerd werk van Gauss [6, 14].

Daarnaast hebben we gezien dat een vermenigvuldiging van kringtermen kan worden gereduceerd tot twee directe plus één inverse DFT. In 1970 merkte Bluestein op dat een DFT van lengte l ook kan worden gereduceerd tot een kringproduct van dezelfde lengte l [2].

Laten we, om dit uit te leggen, voor het gemak veronderstellen dat l even is en η een principiële eenheidswortel van orde $2l$ met $\eta^2=\omega$. Voor gehele j, k geldt dan

$$\eta^{(j+l)^2}=\eta^{j^2}\eta^{2l(j+l/2)}=\eta^{j^2}$$

en

$$\omega^{jk}=\eta^{j^2}\eta^{k^2}\eta^{-(j-k)^2}.$$

De j-de coëfficiënt van de DFT van een kringterm $u_0+\cdots+u_{l-1}x^{l-1}$ is dus gelijk aan

$$\sum_{0\leqslant k<l}u_k\omega^{jk}=\eta^{j^2}\sum_{0\leqslant k<l}\big(u_k\eta^{k^2}\big)\eta^{-(j-k)^2}.$$

De rechtersom herkennen we nu als de

Foto: MFO, Konrad Jakobs

Volker Strassen (links) en Arnold Schönhage (rechts)

j-de coëfficiënt van het product van de volgende twee kringtermen:

$$V = \sum_{0 \leqslant k < l} \left(u_k \eta^{k^2}\right) x^k,$$
$$W = \sum_{0 \leqslant k < l} \eta^{-k^2} x^k.$$

Terug naar de gehele getallen

In mijn geboortejaar 1971 verschenen er maar liefst drie methodes om de FFT toe te passen op het vermenigvuldigen van gehele getallen [19, 21]. Elk van deze methodes was gebaseerd op een aparte keuze van R.

Het meest voor de hand ligt het om $R = \mathbb{C}$ en $\omega = e^{2\pi i/l}$ te nemen. In ons geval kunnen we natuurlijk alleen maar met benaderingen van complexe getallen werken. Voor het vermenigvuldigen van getallen van n cijfers kan worden bewezen dat benaderingen tot op $C \log n$ cijfers achter de komma voldoende zijn voor een zekere constante $C > 0$. Dat betekent dat we kunnen werken met $l = O(n/\log n)$ brokken van $O(\log n)$ cijfers. In combinatie met (2) vinden we dan

$$M(n) = O(l \log l M(\log n)) = O(nM(\log n)).$$

Laten we deze formule recursief op zichzelf los, dan vinden we

$$M(n) = O(n \log n \log\log n \log\log\log n \cdots).$$

Dit was het 'eerste algoritme' in het artikel van Schönhage–Strassen [21].

Het nulde algoritme werd echter al iets eerder door Pollard ontdekt [19]. Hij stelde voor om $R = \mathbb{F}_p$ te nemen, waar p een priemgetal is van de vorm $p = s2^r + 1$; hoe kleiner s, hoe beter. Voor zulke priemgetallen p weten we dat er primitieve eenheidswortels van orde 2^r bestaan. Opnieuw is het dan mogelijk om $\log p = O(\log n)$ en $l = O(n/\log n)$ te kiezen, en we vinden een soortgelijke bovengrens voor $M(n)$ als net. Eerlijkheidshalve moet er wel worden bijgezegd dat dit laatste níet in Pollards artikel stond. Hij was meer uit op een praktisch algoritme en beschreef verschillende optimalisaties. Voor zeer grote getallen is zijn algoritme op hedendaagse computers het beste [8].

Voor het 'tweede algoritme' van Schönhage–Strassen schakelen we over naar het binaire stelsel en nemen we $R = \mathbb{Z}/(2^m + 1)\mathbb{Z}$, waar m een tweemacht is. In deze ring hebben we per definitie $2^m = -1$, zodat $\omega := 2$ een principiële eenheidswortel van orde $2m$ is. Een ander voordeel is dat ω een *snelle eenheidswortel* is. Daarmee bedoelen we dat we snel kunnen vermenigvuldigen met machten van ω: we schuiven gewoon de bits van het getal op terwijl we de relatie $2^m = -1$ gebruiken zodra we de 2^m passeren. Voor $l \in \{m, 2m\}$ laten Schönhage en Strassen vervolgens zien hoe een vermenigvuldiging in $\mathbb{Z}/(2^{lm/2} + 1)\mathbb{Z}$ kan worden gereduceerd tot l vermenigvuldigingen in R. Schrijven we $M'(m)$ voor de kosten van één vermenigvuldiging in R, dan geeft dit

$$M'(lm/2) \leq lM'(m) + O(ml \log l). \quad (3)$$

De term $O(ml \log l)$ komt van de DFTs, waar we gebruik maken van het feit dat $T_\omega = O(l)$ in R. De term $lM'(m)$ komt van de 'interne' vermenigvuldiging in $\prod_{k=0}^{l-1} R[x]/(x - \omega^k) \cong R^l$.

Omgerekend ten opzichte van n leidt (3) grofweg tot de ongelijkheid

$$M'(n) \leq 2n^{1/2} M'(n^{1/2}) + Cn \log n, \quad (4)$$

voor een zekere constante C. Deze formule kunnen we vervolgens 'uitrollen':

$$\begin{aligned} M'(n) &\leq 2n^{1/2} M'(n^{1/2}) + Cn \log n \\ &\leq 4n^{3/4} M'(n^{1/4}) + 2Cn \log n \\ &\leq 8n^{7/8} M'(n^{1/8}) + 3Cn \log n \\ &\vdots \\ &\leq n \log n\, M'(O(1)) + Cn \log n \log\log n. \end{aligned}$$

Dit bewijst dat $M(n) = O(n \log n \log\log n)$, een bovengrens die vijfenveertig jaar stand hield.

Hoe verder?

Van de drie op de FFT gebaseerde methodes uit 1971 heeft de laatste methode één groot nadeel: omdat $l \leq 2m$ wordt een vermenigvuldiging van lengte n 'slechts' gereduceerd tot vermenigvuldigingen van lengte $O(\sqrt{n})$ in plaats van $O(\log n)$. In tegenstelling tot de twee andere methodes kosten de DFTs daarentegen bijna niets.

John Pollard

Dit levert twee sporen op voor verdere verbeteringen.

Een eerste mogelijkheid is om de kosten van DFTs over $\mathbb{C}$ en/of $\mathbb{F}_p$ te drukken. In 2007 lukte Fürer dit voor het eerst [5]. Zijn methode leidt tot een ongelijkheid van de vorm

$$\frac{M(n)}{n\log n} \leq K\frac{M(n')}{n'\log n'} + O(1),$$
$$n' = O((\log n)^2),$$

en de verbeterde bovengrens

$$M(n) = O(n\log nK^{\log^* n}),$$
$$\log^* x = \min\{k\in\mathbb{N}: (\log\circ\overset{k}{\ldots}\circ\log)(x)\leq 1\}.$$

Fürer besteedde geen aandacht aan de precieze 'uitdijingsfactor' $K>1$. Vanaf 2014 lukte het David Harvey, Grégoire Lecerf en mijzelf om deze factor steeds verder naar beneden te jagen [9, 10, 11, 13]; zie Tabel 1.

Onze tweede optie is een verscherping van de ongelijkheid (4). Hier merken we op dat de factor 2 in $\underline{2}n^{1/2}M'(n^{1/2})$ precies wordt gecompenseerd door het feit dat $\log\sqrt{n} = \frac{1}{2}\log n$: bij het uitrollen kost elke iteratie precies $Cn\log n$ stappen extra. Kunnen we deze factor 2 maar een fractie verbeteren, dan verloopt het uitrollen zich als volgt:

$$\begin{aligned} M(n) &\leq 1{,}98n^{1/2}M(n^{1/2}) + Cn\log n\\ &\leq 1{,}98^2n^{3/4}M(n^{1/4})\\ &\quad + (1+0{,}99)Cn\log n\\ &\leq 1{,}98^3n^{7/8}M(n^{1/8})\\ &\quad + (1+0{,}99+0{,}99^2)Cn\log n\\ &\vdots\\ &\leq o(n\log n) + 100Cn\log n. \end{aligned}$$

Ho! Lezen we dat goed?

$$M(n) = O(n\log n).$$

Op deze wens kunnen we voortborduren. Zo is het ook genoeg om te bewijzen dat

$$M(n) \leq \alpha n^{(d-1)/d}M(n^{1/d}) + O(n\log n), \quad (5)$$

waar $d\geq 2$ en $\alpha < 1/d$.

Op naar hogere dimensies

Nu heeft de methode van Schönhage en Strassen inderdaad een frustrerend aspect: de kunstmatig aangemaakte eenheidswortel $\omega = 2$ in R wordt eigenlijk reuze 'ondergebruikt'. Om die reden is onze lengtereductie $n\rightsquigarrow\sqrt{n}$ zeer bescheiden.

Nu zijn er ook DFTs die in meer dan één richting werken. Stel opnieuw dat R een willekeurige ring is met een principiële eenheidswortel ω van orde l. Een d-dimensionale DFT bewerkstelligt dan het isomorfisme

$$\begin{aligned} &R[x_1,\ldots,x_d]/(x_1^l-1,\ldots,x_d^l-1)\\ &\cong \prod_{0\leq k_1,\ldots,k_d<l} R[x_1,\ldots,x_d]/(x_1-\omega_1^{k_1},\ldots,x_d-\omega_d^{k_d}). \end{aligned}$$

Als we de DFTs in $x_2,\ldots,x_d$ nu vervangen door DFTs over de ring $R[x_1]/(x_1^l-1)$, dan kosten ze veel minder, want x_1 is een snelle eenheidswortel in deze ring. Het systematische gebruik van dit soort DFTs werd voor het eerst voorgesteld door Nussbaumer en Quandalle [18].

Met dat idee maken we meteen al een grote stap in de richting van (5). Als $n = l^d$, dan vergt een vermenigvuldiging in $R[x_1,\ldots,x_d]/(x_1^l-1,\ldots,x_d^l-1)$ namelijk $O(n\log n)$ optellingen en aftrekkingen in R plus $n^{(d-1)/d}$ vermenigvuldigingen in $R[x_1]/(x_1^l-1)$.

Er is echter één probleem: snel vermenigvuldigen in $R[x_1,\ldots,x_d]/(x_1^l-1,\ldots,x_d^l-1)$ is niet hetzelfde als snel vermenigvuldigen in $R[x]/(x^{l^d}-1)$. We hebben dus een manier nodig om kringtermen in één variabele x om te smeden tot kringtermen in d variabelen $x_1,\ldots,x_d$.

Met hulp van de oude Chinezen

Nu is het in bepaalde gevallen inderdaad mogelijk om van dimensie te veranderen. Stel dat $l_1,\ldots,l_d$ onderling ondeelbaar zijn. Volgens de Chinese reststelling hebben we dan

$$\mathbb{Z}/(l_1\cdots l_d)\mathbb{Z} \cong \mathbb{Z}/l_1\mathbb{Z} + \cdots + \mathbb{Z}/l_d\mathbb{Z}.$$

Formeel gesproken hebben we dus ook

$$x^{\mathbb{Z}/(l_1\cdots l_d)\mathbb{Z}} \cong x_1^{\mathbb{Z}/l_1\mathbb{Z}}\times\cdots\times x_d^{\mathbb{Z}/l_d\mathbb{Z}}.$$

Nemen we vervolgens lineaire combinaties, dan zien we uiteindelijk dat

$$\begin{aligned} &R[x]/(x^{l_1\cdots l_d}-1)\\ &\quad\cong R[x_1,\ldots,x_d]/(x_1^{l_1}-1,\ldots,x_d^{l_d}-1). \end{aligned}$$

In verband met DFTs werd dit voor het eerst opgemerkt door Good [7]. Agarwal en Cooley gebruikten dit isomorfisme daarna voor het uitrekenen van convoluties [1].

Wel zitten we nu weer met een nieuw probleem: in de vorige paragraaf vroegen we om een isomorfisme waarvoor alle l_i gelijk zijn. Maar ons isomorfisme werkt juist alleen in het geval wanneer $l_1,\ldots,l_d$ onderling ondeelbaar zijn.

Wat er tot slot nog ontbreekt is een manier om de lengte van een DFT iets te veranderen. Dit zou het mogelijk maken om een d-dimensionale DFT van lengte $(l_1,\ldots,l_d)$ om te smeden tot eentje van lengte $(l,\ldots,l)$. Met behulp van een d-dimensionale versie van Bluesteins algoritme kan zo'n DFT vervolgens worden gereduceerd tot een vermenigvuldiging in $R[x_1,\ldots,x_d]/(x_1^l-1,\ldots,x_d^l-1)$. En dat kan uiteindelijk weer efficiënt worden gedaan met behulp van snelle eenheidswortels.

Gaussiaans overmonsteren

Hoe kunnen we een DFT van lengte s vervangen door een DFT van een iets grotere lengte t? Om dit te bereiken veronderstellen we vanaf hier dat $R = \mathbb{C}$.

Neem een DFT van lengte s. In de signaaltheorie wordt de invoer gezien als een reeks monsters van een signaal. De frequentie van de monstername is hierbij evenredig aan s. Is het mogelijk het oorspronkelijke signaal uit onze reeks monsters te reconstrueren? Dan zouden we daarna een nieuwe reeks monsters kunnen nemen, met een andere frequentie.

De meest voor de hand liggende manier om een digitaal signaal analoog te maken is via convolutie met een Gaussiaan $G_\alpha(x) = e^{-\alpha x^2}$. Hoe kleiner α, hoe gladder (en minder scherp) het analoge signaal. Een andere prettige eigenschap is dat de fouriertransformatie van G_α opnieuw een Gaussiaan is.

Laten we deze ideeën in formules vertalen. In plaats van kringtermen van lengte s beschouwen we nu de bijbehorende vectoren van coëfficiënten $u\in\mathbb{C}^s$. Wel spreken we af dat $u_{k+js}\equiv u_k$ voor alle $j\in\mathbb{Z}$. Gegeven $u\in\mathbb{C}^s$ definiëren we $\mathcal{F}_s(u)\in\mathbb{C}^s$ door

$$(\mathcal{F}_s u)_k := \sum_{0\leq k<s} u_k e^{-2\pi i\frac{jk}{s}}.$$

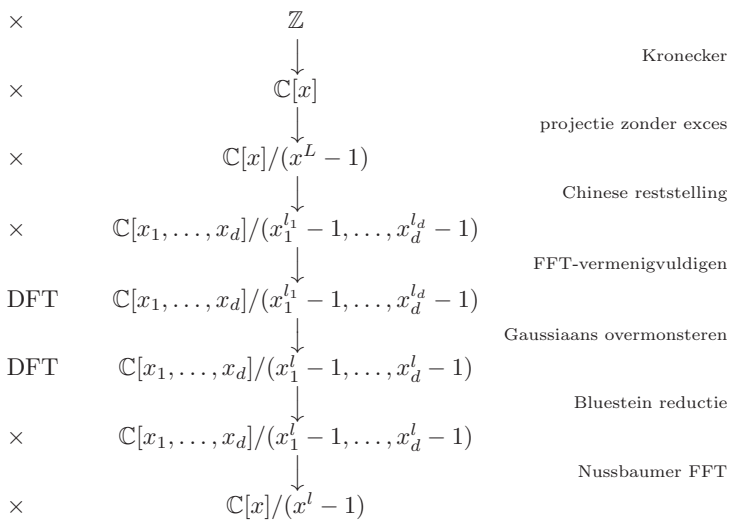
Dit is een variant op onze eerdere definitie van een DFT (nu nemen we $\omega = e^{-2\pi i/s}$). Vervolgens definiëren we twee lineaire afbeeldingen $\mathcal{S},\mathcal{T}:\mathbb{C}^s\to\mathbb{C}^t$ door

$$(\mathcal{S}u)_k := \alpha^{-1}\sum_{j\in\mathbb{Z}} e^{-\pi\alpha^{-2}s^2(\frac{k}{t}-\frac{j}{s})^2}u_j,$$
$$(\mathcal{T}u)_k := \sum_{j\in\mathbb{Z}} e^{-\pi\alpha^{-2}t^2(\frac{k}{t}-\frac{j}{s})^2}u_j.$$

Tot slot definiëren we twee permutaties $\mathcal{P}_s:\mathbb{C}^s\to\mathbb{C}^s$ en $\mathcal{P}_t:\mathbb{C}^t\to\mathbb{C}^t$ door

$$(\mathcal{P}_s u)_j := u_{tj},$$
$$(\mathcal{P}_t u)_k := u_{-sk}.$$

In [12, Theorem 4.2] bewijzen we dan dat het volgende diagram commuteert:



Figuur 2 Schematische weergave van de verschillende reducties in het nieuwe algoritme. Hier en daar hebben we ietwat gesmokkeld om alles simpel te houden.

$$\begin{array}{ccccc} \mathbb{C}^s & \xrightarrow{\mathcal{F}_s} & \mathbb{C}^s & \xrightarrow{\mathcal{P}_s} & \mathbb{C}^s \\ \mathcal{S}\downarrow & & & & \downarrow\mathcal{T} \\ \mathbb{C}^t & \xrightarrow{\mathcal{F}_t} & \mathbb{C}^t & \xrightarrow{\mathcal{P}_t} & \mathbb{C}^t \end{array}$$

Dit is wat we gebruiken om het berekenen van $\mathcal{F}_s$ terug te voeren op het berekenen van $\mathcal{F}_t$. Omdat $t > s$, zijn de matrices voor $\mathcal{S}$ en $\mathcal{T}$ niet vierkant. Wél is het zo dat de elementen die niet op de diagonaal liggen razendsnel afnemen. Dat komt doordat Gaussianen uit het centrum ook razendsnel afnemen. Door $t-s$ goed gekozen rijen uit $\mathcal{T}$ weg te nemen blijft er om die reden een bijna diagonale matrix over. Dit kan worden benut om $\mathcal{T}^{-1}\mathcal{P}_t\mathcal{F}_t\mathcal{S}$ snel uit te rekenen.

Kiezen we a en de precisie met beleid, dan kunnen we bewijzen dat $\mathcal{F}_s$ op deze manier met bijna evenveel precisie als $\mathcal{F}_t$ kan worden berekend en dat de rekentijd van $\mathcal{S}$ and $\mathcal{T}^{-1}$ verwaarloosbaar is ten opzichte van die van $\mathcal{F}_t$.

Daarmee is de kous af en hebben we bewezen dat $M(n) = O(n \log n)$. In Figuur 2 worden alle benodigde reducties nog eens samengevat.

Een variant van onze overmonsteringsmethode werd voor het eerst gepubliceerd door Dutt en Rokhlin [4]. Deze variant is algemener en maakt het mogelijk DFTs te berekenen van onregelmatig gemonsterde signalen. Daarentegen werkt hun methode alleen voor $\log s = O(a)$ en is daarom net iets te traag voor ons uiteindelijke doel.

Wat hebben we eraan?

Praktisch gezien moeten we dat afwachten. Voor theoretische doeleinden is de functie $M(n)$ van belang om de kosten van allerlei rekenkundige operaties nauwkeurig te beschrijven. Zo kost het delen van getallen van n cijfers $O(M(n)) = O(n \log n)$ en het uitrekenen van een grootste gemene deler $O(M(n) \log n) = O(n \log^2 n)$. Ook n decimalen van π kunnen nu in $O(M(n) \log n) = O(n \log^2 n)$ stappen worden berekend. Een DFT van lengte l over de complexe getallen kost $O(M(lp))$ als we rekenen met een precisie van $p \geq \log l$ cijfers [13]. Dat bepaalt ook de minimale CO_2-uitstoot die men kan verwachten bij grote berekeningen aan het weer. Al met al speelt $M(n)$ als elementaire 'rekensnelheid' dus een zelfde soort rol als de lichtsnelheid c in de natuurkunde.

Kan het nog sneller?

Dat weten we niet!

Referenties

1 R. Agarwal en J. Cooley, New algorithms for digital convolution, *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25(5) (1977), 392–410.

2 Leo I. Bluestein, A linear filtering approach to the computation of discrete Fourier transform, *IEEE Transactions on Audio and Electroacoustics* 18(4) (1970), 451–455.

3 J.W. Cooley en J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. Computat.* 19 (1965), 297–301.

4 A. Dutt en V. Rokhlin, Fast Fourier transforms for nonequispaced data, *SIAM J. Sci. Comput.* 14(6) (1993), 1368–1393.

5 M. Fürer, Faster integer multiplication, in *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*, San Diego, CA, 2007, pp. 57–66.

6 C.F. Gauss, Nachlass: theoria interpolationis methodo nova tractata, in *Werke*, Vol. 3, Königliche Gesellschaft der Wissenschaften, Göttingen, 1866, pp. 265–330.

7 I.J. Good, The interaction algorithm and practical Fourier analysis, *Journal of the Royal Statistical Society, Series B.* 20(2) (1958), 361–372.

8 D. Harvey, Faster arithmetic for numbertheoretic transforms, *J. Symbolic Comput.* 60 (2014), 113–119.

9 D. Harvey, Faster truncated integer multiplication, 2017, arXiv:1703.00640.

10 D. Harvey en J. van der Hoeven, Faster integer and polynomial multiplication using cyclotomic coefficient rings, Technical Report, 2017, arXiv:1712.03693.

11 D. Harvey en J. van der Hoeven, Faster integer multiplication using short lattice vectors, in R. Scheidler en J. Sorenson, eds., *Proc. of the 13-th Algorithmic Number Theory Symposium*, Open Book Series 2, Mathematical Sciences Publishes, 2019, pp. 293–310.

12 D. Harvey en J. van der Hoeven, Integer multiplication in time $O(n \log n)$, Technical Report, HAL, 2019, http://hal.archivesouvertes.fr/hal-02070778.

13 D. Harvey, J. van der Hoeven en G. Lecerf, Even faster integer multiplication, *J. Complexity* 36 (2016), 1–30.

14 M.T. Heideman, D.H. Johnson en C.S. Burrus, Gauss and the history of the FFT, *IEEE Acoustics, Speech and Signal Processing Magazine* 1 (1984), 14–21.

15 A.A. Karatsuba, The complexity of computations, *Proc. of the Steklov Inst. of Math.* 211 (1995), 169–183. English translation; Russian original at pp. 186–202.

16 A. Karatsuba en J. Ofman, Multiplication of multidigit numbers on automata, *Soviet Physics Doklady* 7 (1963), 595–596.

17 D.E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, 1969.

18 H.J. Nussbaumer en P. Quandalle, Computation of convolutions and discrete Fourier transforms by polynomial transforms, *IBM J. Res. Develop.* 22(2) (1978), 134–144.

19 J.M. Pollard, The fast Fourier transform in a finite field, *Mathematics of Computation*, 25(114) (1971), 365–374.

20 A. Schönhage, Multiplikation großer Zahlen, *Computing* 1(3) (1966), 182–196.

21 A. Schönhage en V. Strassen, Schnelle Multiplikation großer Zahlen, *Computing* 7 (1971), 281–292.

22 A.L. Toom, The complexity of a scheme of functional elements realizing the multiplication of integers, *Soviet Mathematics* 4(2) (1963), 714–716.