PRIMALITEIT EN FACTORIZATIE

H.W. LENSTRA, Jr.

1. AANVANG

Zij $n > 1$ een geheel getal. We beschouwen de volgende twee problemen:

(a) (*primaliteit*) is n priem?

(b) (*factorizatie*) zo nee, vind $a,b > 1$ met $n = ab$.

We zullen niet een volledig overzicht geven van alle voorgestelde methoden, maar ons beperken tot de *theoretisch* best bekende. Voor al het andere, zoals *practische* methoden, geschiedkundige opmerkingen, toepassingen en verdere literatuurverwijzingen zie men WILLIAMS [15], GUY [5], MONIER [9], SCHNORR [12], KNUTH [6] en voorts de uitgewerkte teksten van voordrachten, gehouden tijdens de studieweek "Getaltheorie en Computers" [4].

Zoals gebruikelijk zullen we een algoritme *goed* noemen als de rekentijd begrensd wordt door een polynoom in de lengte van de input. Voor problemen (a) en (b) is de input het getal n, dat binaire lengte $[\log n/\log 2] + 1$ heeft. De lengte van de input heeft dus dezelfde orde van grootte als $\log n$.

De bekendste methode om de problemen (a) en (b) op te lossen bestaat uit het achtereenvolgens proberen of n deelbaar is door $2,3,4,\ldots,[\sqrt{n}]$. Dit kan $\sqrt{n}$ stappen kosten, hetgeen exponentieel is in de lengte van de input. Deze algoritme is dus niet "goed".

Voordat men op zoek gaat naar een kort bewijs dat n priem is, of een kort bewijs dat n samengesteld is, kan men zich afvragen of een dergelijk bewijs wel bestaat. In deze richting hebben we ten eerste het volgende resultaat; onder een *rekenkundige bewerking* verstaan we een optelling, aftrekking en vermenigvuldiging van twee gehele getallen.

STELLING 1. *Als* n *samengesteld is bestaat hiervoor een bewijs bestaande uit* O(1) *rekenkundige bewerkingen. Als* n *priem is, dito.*

BEWIJS. Voor samengestelde n is de stelling triviaal: om te bewijzen dat n samengesteld is is het voldoende twee gehele getallen a,b > 1 op te schrijven en de enkele vermenigvuldiging uit te voeren waaruit blijkt dat ab = n.

Voor n priem maken we gebruik van een nevenresultaat van de negatieve oplossing van het tiende probleem van Hilbert. Dit resultaat zegt dat er een polynoom

$$f \in \mathbb{Z}[\underline{A},\underline{B},\ldots,\underline{Z}]$$

in 26 variabelen bestaat met de eigenschap dat de verzameling priemgetallen samenvalt met de verzameling *positieve* waarden die f aanneemt als niet-negatieve gehele getallen voor $\underline{A},\underline{B},\ldots,\underline{Z}$ gesubstitueerd worden. Om te bewijzen dat n priem is is het dus voldoende 26 gehele getallen A,B,...,Z op te schrijven en de begrensde hoeveelheid rekenwerk uit te voeren waaruit blijkt dat n = f(A,B,...,Z). Men vindt dat niet meer dan 87 rekenkundige bewerkingen hiervoor vereist zijn. Dit bewijst Stelling 1. □

Het belang van Stelling 1 is om twee redenen louter theoretisch. In de eerste plaats vanwege het non-deterministische karakter van de methode: het bestaan van zekere bewijzen wordt verzekerd, maar er wordt niet bij verteld hoe men ze snel vindt. In de tweede plaats is het meten van de lengte van een bewijs door middel van het aantal rekenkundige bewerkingen volslagen onrealistisch. Wil men met het in bovenstaand bewijs bedoelde polynoom f bewijzen dat n priem is, dan zal tenminste één van A,B,...,Z groter zijn dan

$$n^{n^{n^{n}}}.$$

Het is kennelijk onrealistisch om een rekenkundige bewerking met getallen van deze orde van grootte als één stap te tellen. Daarom zullen we in het vervolg - behalve in Stelling 6 - *bit-operaties* tellen; deze kunnen we definiëren als rekenkundige bewerkingen op getallen van één (binair) cijfer.

Een vermenigvuldiging van twee getallen < n vereist niet meer dan $O((\log n)^2)$ bit-operaties, dus er geldt de volgende stelling:

STELLING 2. *Als* n *samengesteld is bestaat hiervoor een bewijs bestaande uit* $O((\log n)^2)$ *bit-operaties.*

Met behulp van snellere vermenigvuldig-routines kan men dit resultaat verbeteren tot $O((\log n)^{1+\varepsilon})$, voor elke $\varepsilon > 0$. Evenzo kan in de volgende stelling de exponent 4 door $3+\varepsilon$ vervangen worden, voor elke $\varepsilon > 0$.

STELLING 3 (PRATT [11]). *Als* n *priem is bestaat hiervoor een bewijs bestaande uit* $O((\log n)^4)$ *bit-operaties.*

BEWIJS. We nemen n oneven. Uit elementaire getaltheorie volgt dat n priem is dan en slechts dan als er een geheel getal a is, $0 < a < n$, waarvoor geldt

$$a^{(n-1)/2} \equiv -1 \bmod n,$$
$$a^{(n-1)/q} \not\equiv 1 \bmod n \text{ voor elke priemfactor } q \text{ van } n-1.$$

Dus, om te bewijzen dat n priem is schrijven we een geheel getal a op, $0 < a < n$, we schrijven de ontbinding van n-1 op:

(1) $\quad n-1 = q_0 q_1 \cdots q_t \qquad$ met $q_0 = 2$,

we verifiëren dat

(2) $\quad a^{(n-1)/2} \equiv -1 \bmod n,$

(3) $\quad a^{(n-1)/q_i} \not\equiv 1 \bmod n \qquad$ voor $1 \le i \le t$

en we controleren recursief:

(4) $\quad q_i$ is priem $\qquad (1 \le i \le t)$.

Dit alles vereist t vermenigvuldigingen in (1), en t+1 machtsverheffingen in (2) & (3), plus wat voor (4) vereist is. Geven we met f(n) het totale aantal vermenigvuldigingen en machtsverheffingen aan, dan geldt dus

$$f(n) \le t + t + 1 + \sum_{i=1}^{t} f(q_i);$$

hier definiëren we $f(2) = 1$. Inductief bewijzen we dat $f(n) \le 3(\log n/\log 2) - 2$. Dit geldt voor $n = 2$, en als het voor de q_i geldt, dan

$$f(n) \le 2t + 1 + \sum_{i=1}^{t} (3(\log q_i/\log 2) - 2) = \left(\sum_{i=0}^{t} 3(\log q_i/\log 2)\right) - 2 =$$

$$= 3(\log(n-1)/\log 2) - 2 < 3(\log n/\log 2) - 2,$$

zoals verlangd.

Er zijn dus niet meer dan O(log n) vermenigvuldigingen en machtsverheffingen nodig. Elke machtsverheffing in (2), (3) kan men uitvoeren door middel van O(log n) kwadrateringen en vermenigvuldigingen modulo n. Elke vermenigvuldiging, kwadratering of vermenigvuldiging modulo n (of een getal kleiner dan n) kost $O((\log n)^2)$ bit-operaties. Dit leidt tot de grens $O((\log n)^4)$, waarmee Stelling 3 bewezen is. □

Stellingen 2 en 3 hebben nog het eerste gebrek van Stelling 1: het non-deterministische karakter van de methode. Desondanks is het gegeven bewijs van Stelling 3 niet uitsluitend van theoretische waarde: er zijn verscheidene in de practijk toegepaste primaliteitstests waarvan de grondgedachte dezelfde is. De voornaamste moeilijkheid is dan het vinden van de priemfactorontbinding van n-1. Er zijn varianten waarvoor een gedeeltelijke ontbinding van n-1 ook al voldoende is. Lukt het niet om genoeg factoren van n-1 te vinden, dan zijn er verwante tests waarvoor men factoren van n+1 dient te kennen, en beide soorten tests kunnen ook gecombineerd worden. Het vervolg van n-1, n+1 luidt niet n-2, n+2, n-3, ... zoals men zou kunnen denken, maar

$$n^2+n+1,\ n^2+1,\ n^4+n^3+n^2+n+1,\ n^2-n+1,\ \ldots$$

waarbij de k-de term gegeven is door

$$\Phi_k(n) = \prod_{1\le a\le k,\ ggd(a,k)=1} (n - e^{2\pi i a/k}),$$

(dus $\Phi_1(n) = n-1$, $\Phi_2(n) = n+1$).

Voor getallen n van de vorm $n = 2^k\pm1$ is $n\mp1$ eenvoudig in factoren te ontbinden, en in deze gevallen blijken de bovengenoemde tests een bijzonder eenvoudige vorm aan te nemen. Dat is een plezierige samenloop van omstandigheden, want priemgetallen van de vorm $2^k\pm1$ spelen een bijzondere rol in de wiskunde: met behulp van de priemgetallen van de vorm 2^k+1 (*Fermat*-priemgetallen) wist *Gauss* alle getallen n te karakteriseren waarvoor een regelmatige n-hoek met passer en liniaal te construeren is; en priemgetallen van de vorm 2^k-1 (*Mersenne*-priemgetallen) komen voor in een stelling van *Euclides* en *Euler* over volmaakte getallen. De enige bekende Fermat-priemgetallen (met $k > 0$) zijn 3,5,17,257 en 65537, en er zijn redenen om aan te nemen dat dit ze alle zijn. Daarentegen vermoedt men dat er oneindig veel

Mersenne-priemgetallen zijn. Met deze stand van zaken is het weinig verbazend dat het "grootst bekende priemgetal", zoals dat tot ons komt via de nieuwsmedia en het *Guinness book of records*, steeds een Mersenne-priemgetal is; op het ogenblik is $2^{44497}-1$ de gelukkige.

Voordat we deze het theoretische kader van deze voordracht ietwat te buiten gaande uitweiding beeindigen noemen we nog een tweede aardige toepassing van de primaliteitstests die op ontbindingen van $n\pm1$ berusten, namelijk de jacht op *priemgetaltweelingen*. Een priemgetaltweeling is een tweetal priemgetallen met verschil 2, zoals 3, 5 of 101, 103. Men vermoedt dat er oneindig veel priemgetaltweelingen bestaan; de grootst bekende is

$$256200945 \cdot 2^{3426} \pm 1 = 2^{3426} \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 113 \cdot 151 \pm 1.$$

Zoals men ziet is het getal dat tussen de tweeling-priemgetallen in ligt uit kleine factoren opgebouwd, zodat de ontbinding ervan volledig bekend is. Dit maakt het mogelijk de primaliteit van beide getallen te bewijzen door op de grootste een n-1 - test toe te passen, en op de kleinste een n+1 - test.

Voor meer details over deze zaken verwijzen we naar de aan het begin aangehaalde literatuur.

We keren terug tot onze theoretische beschouwingen. Stellingen 2 en 3 impliceren dat, in modern jargon, het probleem of een getal priem is behoort to NP ∩ coNP. Behoort het, zo zal de expert zich dan direct afvragen, wellicht ook tot de klasse P? Dat wil zeggen, is er "goede" en *deterministische* algoritme die beslist of een getal n priem is? Het antwoord hierop luidt waarschijnlijk bevestigend, maar zoals de zaken nu staan moeten we voor een definitief antwoord wachten op het bewijs van de zgn. gegeneraliseerde Riemann-hypothese. Deze hypothese zegt iets over de ligging van de nulpunten van zekere in de analytische getaltheorie optredende complexe functies, en een bewijs ervan zou verreikende consequenties hebben. Voor een precieze formulering verwijzen we naar de literatuur.

STELLING 4 (G.L. MILLER [8]). *Laat de gegeneraliseerde Riemann-hypothese waar zijn. Dan is er een algoritme die in* $O((\log n)^5)$ *stappen beslist of* n *priem is.*

Beneden geven we een beschrijving van de algoritme. Zolang de Riemann-hypothese onbewezen blijft is het interessant na te gaan welke rol de hypothese in de algoritme speelt. Het blijkt dat de terminatie van de

algoritme in $O((\log n)^5)$ stappen niet van de hypothese afhankelijk is, maar de correctheid van het antwoord wel. Nauwkeuriger geformuleerd: als de algoritme het getal n samengesteld verklaart is n inderdaad samengesteld; maar als de algoritme n priem verklaart, kan het zijn dat n samengesteld is, en de Riemann-hypothese fout.

Neem aan dat n oneven is, en schrijf $n-1 = u \cdot 2^t$, met u oneven en $t \geq 1$. In navolging van Rabin noemen we een geheel getal a een *getuige* voor de samengesteldheid van n, of kortweg een getuige voor n, als de volgende drie voorwaarden vervuld zijn:

(5) $\quad a \not\equiv 0 \bmod n,$

(6) $\quad a^u \not\equiv 1 \bmod n,$

(7) $\quad a^{u \cdot 2^i} \not\equiv -1 \bmod n \quad$ voor $i = 0,1,\ldots,t-1.$

Of a al dan niet een getuige van n is hangt alleen van a modulo n af; dus we mogen ons beperken tot $0 \leq a < n$. Voor zo'n a kan men in $O((\log n)^3)$ stappen nagaan of (5), (6) en (7) vervuld zijn.

Getuigen zijn betrouwbaar: als a een getuige is voor de samengesteldheid van n dan is n inderdaad samengesteld. Neem namelijk aan dat n priem is. Uit (5) en de stelling van Fermat (zie beneden) volgt dan dat

$$a^{n-1} \equiv 1 \bmod n,$$

dus van de rij

$$a^u, a^{u \cdot 2}, \ldots, a^{u \cdot 2^t}$$

is de laatste term 1 mod n. De eerste niet, wegens (6). Zij $b = a^{u \cdot 2^i}$ de laatste term in de rij die niet 1 mod n is. Dan geldt $0 \leq i \leq t-1$, en $b^2 = a^{u \cdot 2^{i+1}} \equiv 1 \bmod n$. Dus n deelt $b^2-1 = (b+1)(b-1)$, maar uit $b \not\equiv 1 \bmod n$ volgt dat n geen deler is van b-1. Omdat n priem is impliceert dit dat n een deler van b+1 is, dus $b \equiv -1 \bmod n$, in tegenspraak met (7).

In de practijk zijn getuigen niet lastig te vinden, als n tenminste samengesteld is: Rabin heeft bewezen dat in dat geval tenminste 75% van de getallen $1,2,\ldots,n-1$ uit getuigen voor n bestaat. Dit leidt tot een probabilistische primaliteitstest: trek honderd willekeurige getallen uit $\{1,2,\ldots,n-1\}$; is er een getuige voor n bij, dan is n samengesteld; en is er geen getuige bij dan verklaart men n priem. In ten hoogste één op de 4^{100} gevallen leidt dit tot een verkeerd antwoord. Dit is voor de commerciële productie van priemgetallen een aanvaardbaar risico, maar het levert

geen bewijs van Stelling 4. Om dit te doen vervangen we de honderd willekeurige getallen door de getallen a met $1 < a < 70 \cdot (\log n)^2$; uit de Riemann-hypothese kan men namelijk afleiden dat elke samengestelde n een getuige in dit interval heeft. Vindt men dus geen getuige, dan moet n priem zijn, of de Riemann-hypothese fout. Dit besluit onze schets van het bewijs van Stelling 4. □

Zonder Riemann-hypothese blijven we zitten met een slechte algoritme:

STELLING 5 (ADLEMAN, POMERANCE, RUMELY [1,2]). *Er is een algoritme die in* $O((\log n)^{c \log\log\log n})$ *stappen beslist of* n *priem is, voor* $n > e^e$. *Hier geeft* c *een effectief berekenbare constante aan.*

Het bewijs van Stelling 5 vereist gecompliceerde beschouwingen uit zowel de algebraische als de analytische getaltheorie, die buiten het kader van deze voordracht vallen. Met een verbeterde versie van de algoritme uit Stelling 5 (zie LENSTRA [7]) verwacht men in de practijk de primaliteit van getallen van ruim honderd cijfers gemakkelijk te kunnen beslissen.

De meeste van de bovenbeschreven primaliteitstests vertonen een verbazend gedrag wanneer men ze op niet-priemgetallen n toepast: men krijgt te horen dat n samengesteld is, maar een factor van n wordt er niet bijgeleverd, en is uit de gedane berekeningen ook niet rechtstreeks af te leiden. Stel bijvoorbeeld dat we weten dat n samengesteld is omdat we een geheel getal a hebben gevonden met

$$a^{n-1} \not\equiv 1 \bmod n, \quad \gcd(a,n) = 1,$$

hetgeen volgens de stelling van Fermat voor n priem onmogelijk is. Om in te zien waarom dit ons geen factor van n levert, moeten we nagaan hoe de stelling van Fermat bewezen wordt. Dit kan men doen door op te merken dat de afbeelding $i \mapsto ai \pmod n$ een permutatie van $\{1,2,\ldots,n-1\}$ is, en dat bijgevolg

$$a^{n-1} \cdot (n-1)! = \prod_{i=1}^{n-1} (ai) \equiv \prod_{i=1}^{n-1} i = (n-1)! \bmod n.$$

Als nu $a^{n-1} \not\equiv 1 \bmod n$ dan moet $(n-1)!$ een factor gemeenschappelijk hebben met n, hetgeen ons niets meer of minder vertelt dan dat n samengesteld is.

Ondertussen is het de moeite waard op te merken dat snelle manieren om faculteiten of binomiaalcoëfficiënten modulo n te berekenen behulpzaam kunnen zijn bij factorizatie. Deze opmerking ligt ten grondslag aan de volgende

twee stellingen, in de eerste waarvan weer, als in Stelling 1, op weinig realistische wijze het aantal rekenkundige bewerkingen geteld wordt; bovendien beschouwen we een deling met rest nu ook als een rekenkundige bewerking.

STELLING 6 (SHAMIR [13]). *Er bestaat een algoritme die voor elke samengestelde* n *een niet-triviale deler van* n *levert, en die niet meer dan* O(log n) *rekenkundige bewerkingen vereist.*

BEWIJS. Het getal n is samengesteld dan en slechts dan als $1 < \text{ggd}(a_0!,n) < n$ voor een positief geheel getal a_0. Aangezien ggd(a!,n) als functie van a monotoon niet-dalend is, en in a = 1, n de waarden 1,n aanneemt, kan een dergelijke a_0 door middel van O(log n) bisecties bepaald worden, mits we weten hoe ggd(a!,n) te berekenen.

Als we a! kennen, kunnen we de ggd met de Euclidische algoritme in O(log n) rekenkundige stappen bepalen. Ter berekening van a! passen we de formules

$$(2b+1)! = (2b+1)\cdot(2b)!,$$

$$(2b)! = (b!)^2\cdot\binom{2b}{b}$$

O(log a) keer toe. De benodigde binomiaalcoëfficiënt $\binom{2b}{b}$ bepaalt men door naar het middelste blok van n binaire cijfers in $(2^n+1)^{2b} = \sum_{i=0}^{2b}\binom{2b}{i}\cdot 2^{in}$ te kijken, voor $2b < n$, waarbij de machtsverheffing met O(log(2b)) vermenigvuldigingen kan worden gedaan.

De beschreven algoritme loopt in $O((\log n)^3)$ rekenkundige stappen. SHAMIR [13] brengt dit met enkele kunstgrepen omlaag tot O(log n). Dit besluit onze schets van het bewijs van Stelling 6. □

Tellen we bit-operaties, dan is het best bekende resultaat veel poverder:

STELLING 7 (POLLARD [10]). *Voor elke* $\varepsilon > 0$ *bestaat er een algoritme die elk geheel getal* $n > 1$ *volledig factorizeert in* $O(n^{(1/4)+\varepsilon})$ *stappen.*

BEWIJS. Het vereenvoudigde bewijs dat we hier schetsen is afkomstig van V. STRASSEN [14]. Om n te factorizeren is het voldoende zijn factoren $\leq \sqrt{n}$ te kennen, en evenals in het bewijs van Stelling 6 kan men

inzien dat men deze kan bepalen als men weet hoe a! mod n te berekenen, voor $a \leq \sqrt{n}$. De stelling is dus bewezen als we aantonen dat a! mod n in $O(a^{1/2} \cdot n^{\varepsilon})$ stappen bepaald kan worden.

We mogen aannemen dat a een kwadraat is: $a = b^2$. Definieer

$$f(x) = \prod_{i=1}^{b} (x+i),$$

dan geldt

$$a! = \prod_{j=0}^{b-1} f(jb).$$

Uit algemene resultaten over polynoom-arithmetiek volgt dat de coëfficiënten van het polynoom f in $O(b \cdot n^{\varepsilon})$ stappen modulo n berekend kunnen worden, en dat, gegeven f, de b waarden $f(0), f(b), \ldots, f((b-1)b)$ ook in $O(b \cdot n^{\varepsilon})$ stappen modulo n berekend kunnen worden. We kunnen, tenslotte, deze b waarden in $O(b \cdot n^{\varepsilon})$ stappen modulo n vermenigvuldigen, hetgeen a! mod n oplevert. Hiermee is Stelling 7 bewezen. □

Met de Riemann-hypothese gaat het een beetje sneller:

STELLING 8. *Voor elke* $\varepsilon > 0$ *bestaat er een algoritme die elk geheel getal* $n > 1$ *volledig factoriseert en, als de gegeneraliseerde Riemann-hypothese waar is, termineert in* $O(n^{(1/5)+\varepsilon})$ *stappen.*

Men heeft, voor het bewijs van Stelling 8, de keuze uit verschillende algoritmen. Deze maken alle gebruik van de door *Gauss* (1801) ontwikkelde theorie van binaire quadratische vormen $aX^2 + bXY + cY^2$ met gehele coëfficiënten a, b, c, en meer in het bijzonder van het verband dat er bestaat tussen de zogenaamde *ambigue* vormen, waarvoor b deelbaar is door a, en de ontbindingen van de discriminant $\Delta = b^2 - 4ac$ in twee factoren. Hierbij kan men zowel met positieve als met negatieve Δ werken. Voor meer informatie over deze methoden, waarvan de grondgedachte afkomstig is van *Shanks*, zie men de bijdrage van R.J. Schoof in [4].

De Riemann-hypothese die in Stelling 8 bedoeld wordt is algemener dan die uit Stelling 4, en speelt bovendien een andere rol: het is niet de correctheid van het antwoord die erdoor gegarandeerd wordt, maar het feit dat de algoritme binnen de gestelde rekentijd termineert.

Een wezenlijk verschil tussen de problemen (a) (primaliteit) en (b) (factorizatie) bestaat uit het volgende. Bij (a) "weet" men vrij snel het antwoord op de vraag of n priem is, de grote moeilijkheid bestaat eruit

de *juistheid* van dit antwoord te bewijzen. Bij (b) is het echter juist de grote kunst om factoren a,b > 1 van n te *vinden*; als ze eenmaal gevonden zijn is het triviaal om te controleren dat a·b = n. Deze omstandigheid leidt, bij probleem (b), tot een grotere nadruk op algoritmen van probabilistische aard, waarvan het termineren binnen een bepaalde tijd niet gegarandeerd wordt, maar wel op grond van een probabilistische redenering *verwacht* wordt. Verscheidene van de beste in de practijk gebruikte methoden zijn van deze soort; we noemen in het bijzonder de rho-methode van *Pollard* en de "square-form factorization"-algoritme (SQUFOF) van *Shanks*, die beide verwachte rekentijd $O(n^{(1/4)+\varepsilon})$ hebben. Hier willen we vooral aandacht besteden aan methoden waarvan de verwachte rekentijd sneller dan elke macht van n is. Zie SCHNORR [12] voor een uitgebreidere behandeling.

De te bespreken methoden beginnen ermee een reeks paren gehele getallen (c_i,d_i) te construeren waarvoor geldt

$$c_i^2 \equiv d_i \bmod n,$$

alle priemfactoren van $|d_i|$ zijn "klein", zeg $\leq B$.

Iedere d_i ontbindt men in priemfactoren:

$$d_i = (-1)^{n_{i1}} \cdot \prod_{p \text{ priem},\ p\leq B} p^{n_{ip}}.$$

Van de vectoren $(n_{i1},n_{i2},n_{i3},n_{i5},\ldots)$ beschouwt men nu de coördinaten modulo 2. Heeft men genoeg paren (c_i,d_i) dan kan men met behulp van lineaire algebra over $\mathbb{Z}/2\mathbb{Z}$ een relatie tussen deze vectoren vinden:

$$\sum_{i\in I} n_{ip} \equiv 0 \bmod 2 \quad \text{voor } p = 1,2,3,5,\ldots .$$

Dan is $\prod_{i\in I} d_i$ een *kwadraat*, zeg e^2, en met $g \equiv \prod_{i\in I} c_i \bmod n$ geldt dan

$$g^2 \equiv e^2 \bmod n,$$

dus n is een deler van $g^2 - e^2 = (g-e)\cdot(g+e)$. Nu hoopt men dat $g \not\equiv \pm e \bmod n$, dan zijn ggd(n,g-e) en ggd(n,g+e) niet-triviale factoren van n.

De diverse algoritmen die op het bovenstaande principe berusten verschillen voornamelijk in de manier waarop de paren (c_i,d_i) gegenereerd worden.

Het eenvoudigst gaat DIXON [3] te werk. Hij kiest de c_i willekeurig uit $\{1,2,\ldots,n\}$, zet $d_i = (c_i^2 \bmod n)$, en verwerpt de paren (c_i,d_i) waarvoor d_i

een priemfactor > B heeft, met $B = \exp(\sqrt{2 \log n \operatorname{loglog} n})$. Dixon is dan in staat een stelling te formuleren en te bewijzen, die ruwweg zegt dat men mag verwachten dat de algoritme in $O(B^3) = O(\exp(3\sqrt{2 \log n \operatorname{loglog} n}))$ stappen termineert. Uit

$$\exp(\sqrt{\log n \operatorname{loglog} n}) = n^{\sqrt{\operatorname{loglog} n / \log n}} = (\log n)^{\sqrt{\log n / \operatorname{loglog} n}}$$

blijkt dat dit sneller dan elke vaste macht van n is, maar langzamer dan elke macht van log n. Het geheugengebruik van Dixon's methode is $O(B^2)$.

Een tweede, en in de practijk ook werkelijk gebruikte methode om de paren (c_i, d_i) te genereren maakt gebruik van de kettingbreukontwikkeling van $\sqrt{n}$ (of $\sqrt{kn}$, met k klein). De kettingbreukontwikkeling levert rationale benaderingen A_i/B_i van $\sqrt{n}$, en men neemt $c_i = A_i \bmod n$, $d_i \equiv A_i^2 \bmod n$, $|d_i|$ minimaal. Omdat A_i "dichtbij" $B_i\sqrt{n}$ ligt, ligt A_i^2 "dichtbij" het veelvoud $B_i^2 n$ van n, dus men verwacht dat d_i "klein" is, en inderdaad kan men bewijzen

$$|d_i| < 2\sqrt{n}.$$

Dit verkleint de kans dat een paar (c_i, d_i) verworpen moet worden vanwege een te grote priemfactor in d_i. Met een heuristische redenering heeft *Wunderlich* laten zien dat men met $B = \exp(\sqrt{(1/3) \log n \operatorname{loglog} n})$ terminatie in tijd $O(B^3) = \exp(\sqrt{3 \log n \operatorname{loglog} n})$ mag verwachten. Dit is sneller dan *Dixon* maar er is hier geen sprake van een precieze stelling.

De net beschreven methode werd door *Lehmer* en *Powers* voorgesteld in 1931, de tijd dat men nog met de hand rekende. Er bleek een niet te verwaarlozen kans te bestaan dat men, hopende bijna klaar te zijn, ontdekt dat $g \equiv \pm e \bmod n$, zodat men practisch weer helemaal opnieuw moet beginnen. Door dit frustrerende effect werd de methode niet populair. *Brillhart* en *Morrison* realiseerden zich dat een computer niet gevoelig is voor dit nadeel van de methode, en haalden hem in 1970 weer van stal. Voor erg grote getallen (30 à 40 cijfers) is het de beste in de practijk gebruikte algoritme.

Tenslotte is er een algoritme van *Schroeppel*, die dezelfde grondgedachte iets anders uitwerkt. In plaats van met congruenties $c_i^2 \equiv d_i \bmod n$ werkt hij met congruenties

$$\prod_{c \in C} c^{m_{ic}} \equiv d_i \bmod n$$
$$m_{ic} \in \mathbb{Z}_{\geq 0}, \quad \sum_{c \in C} m_{ic} = 2,$$

waarbij C een vaste collectie getallen in de buurt van $\sqrt{n}$ is, zodat d_i weer klein is. Men kiest het aantal elementen van C gelijk aan het aantal priemgetallen $\leq B$. De vectoren $(n_{i1}, n_{i2}, \ldots)$ maakt men 2× zo lang door er de m_{ic} aan toe te voegen. Dit leidt dan weer op dezelfde manier tot een congruentie $g^2 \equiv e^2 \bmod n$.

Schroeppel geeft de verzameling C een speciale structuur, waardoor het tijdrovende factorizeren van $d_i = (\prod c^{m_{ic}}) - n$ kan geschieden door te zeven. Verder beredeneert hij dat $B = \exp(\frac{1}{2}\sqrt{\log n \, \text{loglog } n})$ de optimale keuze is. Dit leidt tot een algoritme met verwachte rekentijd $O(B^3) = \exp(\frac{3}{2}\sqrt{\log n \, \text{loglog } n})$ (de in een ongepubliceerd geschrift van *Boonstra* genoemde grens $O(B^2)$, die Schroeppel claimt, wordt niet door de daar genoemde argumenten ondersteund). Opnieuw is hier geen sprake van een precies geformuleerde en bewezen stelling.

Samenvattend kunnen we concluderen dat probleem (a) "doenlijk" is, vooral als men aan een morele zekerheid over de juistheid van het antwoord voldoende heeft, maar dat probleem (b) met de tegenwoordige kennis "ondoenlijk" is voor algemene n.

2. LITERATUUR

[1] ADLEMAN, L.M., *On distinguishing prime numbers from composite numbers* (abstract), Proc. 21st Annual IEEE Symp. Found. Comp. Sci. (1980), 387-406.

[2] ADLEMAN, L.M., C. POMERANCE & R.S. RUMELY, *On distinguishing prime numbers from composite numbers*, Preprint.

[3] DIXON, J.D., *Asymptotically fast factorization of integers*, Math. Comp. 36 (1981), 255-260.

[4] GETALTHEORIE EN COMPUTERS, Studieweek, Mathematisch Centrum 1980.

[5] GUY, R.K., *How to factor a number*, Proc. Fifth Manitoba Conf. Numer. Math, Utilitas, Winnipeg (1975), 49-89.

[6] KNUTH, D.E., *The art of computer programming, vol. 2*, Seminumerical Algorithms, second edition, Addison-Wesley, Reading 1980.

[7] LENSTRA, H.W., JR., *Primality testing algorithms*, Séminaire Bourbaki 33 (1980/81), exp. 576, Lecture Notes in Mathematics, Springer, to

appear.

[8] MILLER, G.L., *Riemann's hypothesis and tests for primality*, J. Comput. System Sci. 13 (1976), 300-317.

[9] MONIER, L., *Algorithmes de factorization d'entiers*, Thèse, Orsay 1980.

[10] POLLARD, J.M., *Theorems on factorization and primality testing*, Proc. Cambridge Philos. Soc. 76 (1974), 521-528.

[11] PRATT, V.R., *Every prime has a succinct certificate*, SIAM J. Comput. 4 (1975), 214-220.

[12] SCHNORR, C.P., *Refined analysis and improvements on some factoring algorithms*, to appear in: Automata, Languages and Programming, Eighth Colloquium, Haifa 1981, Lecture Notes in Computer Science, Springer.

[13] SHAMIR, A., *Factoring numbers in* O(log n) *arithmetic steps*, Inform. Process. Lett. 8 (1979), 28-31.

[14] STRASSEN, V., *Einige Resultate uber Berechnungskomplexität*, Jber. Deutsche Math. Verein. 78 (1976), 1-8.

[15] WILLIAMS, M.C., *Primality testing on a computer*, Ars Combin. 5 (1978), 127-185.